ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Βελτιστοποίηση επιλογής γειτόνων σε δομημένα δίκτυα ομοτίμων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Σ. Κλουδάς

Επιβλέπων : Νεκτάριος Κοζύρης
Αν. Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Βελτιστοποίηση επιλογής γειτόνων σε δομημένα δίκτυα ομοτίμων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Σ. Κλουδάς

Επιβλέπων : Νεκτάριος Κοζύρης
Αν. Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7η Ιουλίου 2009.

.....	.....	.....
Ν.Κοζύρης	Τ.Σελλής	Ν.Παπασπύρου
Αν.Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Επ.Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2009

.....
Κωνσταντίνος Σ. Κλουδάς
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα δίκτυα ομοτίμων (peer-to-peer) αποτελούν σήμερα τους πιο δημοφιλείς εκπροσώπους των υπερκείμενων δικτύων προσφέροντας πλήθος υπηρεσιών όπως μετάδοση αρχείων, video streaming και υπηρεσίες VOIP (voice-over-IP) και όντας υπεύθυνα για σημαντικό ποσοστό της παγκόσμιας κίνησης στο διαδίκτυο. Το τελευταίο, σε συνδυασμό με το πλήθος εφαρμογών που έχουν ως δομικό συστατικό ένα δίκτυο ομοτίμων, έχει οδηγήσει μεγάλο μέρος της έρευνας γύρω από τα p2p δίκτυα να στραφεί προς τη μελέτη τρόπων μείωσης του φόρτου που συνεπάγεται η διατήρηση της δομής των δικτύων ομοτίμων και πιο αποτελεσματικής χρήσης του διαθέσιμου εύρους ζώνης.

Στην εργασία αυτή παρουσιάζουμε τη μελέτη της χρήσης του διαθέσιμου εύρους ζώνης ως μέτρου για την επιλογή γειτόνων σε δομημένα δίκτυα ομοτίμων. Στόχος, είναι η μείωση των μηνυμάτων αυτορύθμισης του δικτύου που ανταλλάσσονται μεταξύ των κόμβων. Για το σκοπό αυτό χρησιμοποιήσαμε το Bamboo DHT, μια open source υλοποίηση ενός δομημένου δικτύου ομοτίμων βασισμένου στην αρχιτεκτονική του Pastry, ενώ για τη μέτρηση του διαθέσιμου εύρους ζώνης χρησιμοποιήσαμε το Spruce, ένα open source εργαλείο βασισμένο στο μοντέλο PGM (Probe Gap Model). Η αξιολόγηση του νέου αλγόριθμου επιλογής γειτόνων έγινε μέσω της εγκατάστασής του σε 75 κόμβους του Planetlab και της σύγκρισής του με τον αρχικό αλγόριθμο του Bamboo. Από τις μετρήσεις που συλλέξαμε είδαμε ότι με την υλοποίησή μας, τα μηνύματα μεταξύ των κόμβων για την αυτορύθμιση του δικτύου μειώνονται κατά 98.5% ενώ ο χρόνος απόκρισης κατά τη δρομολόγηση ερωτημάτων στο δίκτυό μας είναι κατά μόλις 17% χειρότερος από αυτόν του αρχικού αλγορίθμου επιλογής γειτόνων του Bamboo στο 90% των περιπτώσεων.

Λέξεις Κλειδιά

Δομημένα δίκτυα ομοτίμων, μέτρηση εύρους ζώνης, επιλογή γειτονικών κόμβων, Planetlab, κατανεμημένα συστήματα, επιδημικοί αλγόριθμοι, Bamboo, Spruce, κατανεμημένοι πίνακες κατακερματισμού.

Abstract

Peer-to-peer networks constitute the most popular representatives of overlay networks, offering numerous services such as file sharing, video streaming, VOIP services, on-line gaming services and being responsible for a large fraction of the global internet traffic. This last observation, in addition to the large amount of applications built on top of peer-to-peer overlays, has led a large portion of the research taking place on the field of p2p networks to focus on the aspect of more effective bandwidth utilization and decrease of the traffic created for the self-organization of these networks.

In this thesis, we present the study of the use of the available bandwidth as a metric for the neighbour selection in structured peer-to-peer networks. Our goal is the reduction of the messages exchanged between nodes for the self-organization of the overlay network. For that, we use the Bamboo DHT, an open source implementation of a structured peer-to-peer network based on the architecture of the Pastry structured overlay while for the measurement of the available bandwidth we used Spruce, an open source tool based on the PGM(Probe Gap Model) model. The evaluation of the new proximity neighbour selection algorithm was made on Planetlab, where it was deployed on 75 nodes and compared to the initial algorithm. From the measurements we collected we saw that with our implementation, the messages for the self-organization of the network were decreased by 98.5% while the mean lookup time was only 17% worse than that of the initial Bamboo on the 90% of the cases.

Keywords

Structured peer-to-peer networks, bandwidth measurement, proximity neighbor selection, Planetlab, distributed systems, epidemic algorithms, Bamboo, Spruce, distributed hash tables.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΠΙΝΑΚΑΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1: ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

1.1 Peer-to-peer Υπερκείμενα Δίκτυα

1.1.1 Υπερκείμενα δίκτυα

Η αλλαγή στους βασικούς μηχανισμούς του Διαδικτύου (στα βασικά πρωτόκολλα του), ακόμα και αν αυτό είναι η πρόσθεση επιπλέον λειτουργικότητας σε αυτούς, δεν είναι απλή υπόθεση. Σε αυτήν την περίπτωση θα χρειαστεί να αλλάξει η υλοποίηση των πρωτοκόλλων σε κάθε υπάρχοντα κόμβο. Σαν απάντηση στο παραπάνω πρόβλημα, προτάσσεται η λύση των *υπερκείμενων δικτύων* κι εκεί είναι που αυτά οφείλουν και την τεράστια δημοτικότητά τους τόσο σε επίπεδο τελικών χρηστών (μέσω διαφόρων εφαρμογών) όσο και μεταξύ ερευνητών. Τα υπερκείμενα δίκτυα δημιουργούν ένα εικονικό δίκτυο πάνω από την ήδη υπάρχουσα τοπολογία χρησιμοποιώντας τα δικά τους σχήματα δρομολόγησης και καμιά φορά και το δικό τους χώρο διευθύνσεων. Έτσι, οποιαδήποτε επιπλέον λειτουργικότητα μπορεί να προστεθεί «πάνω» από το επίπεδο δικτύου και με τρόπο ανεξάρτητο από τους περιορισμούς που αυτό επιβάλει.

Τα “peer-to-peer” υπερκείμενα δίκτυα ή δίκτυα ομοτίμων αποτελούν τους πιο γνωστούς εκπροσώπους των υπερκείμενων δικτύων. Ο όρος “peer-to-peer” χρησιμοποιήθηκε αρχικά στα μέσα της δεκαετίας του 1980 από τους σχεδιαστές τοπικών δικτύων (LAN) για την περιγραφή της αρχιτεκτονικής διασύνδεσης των υπολογιστών σε αυτά. Ουσιαστικά, “peer-to-peer” απλά σημαίνει επικοινωνία μεταξύ ίσων. Ωστόσο, σήμερα, δεδομένης της ραγδαίας εξάπλωσης του διαδικτύου, ο όρος “peer-to-peer” έχει αποκτήσει ένα πιο εξειδικευμένο νόημα. Ορίζεται ως μια ομάδα εφαρμογών η οποία εκμεταλλεύεται πόρους – αποθηκευτικό χώρο, υπολογιστικούς κύκλους, ανθρώπινη παρουσία - διαθέσιμους σε απομακρυσμένα μηχανήματα διάσπαρτα στο διαδίκτυο.

Σε ένα δίκτυο ομοτίμων (“peer-to-peer”) κάθε μέλος μπορεί να συμμετέχει ανά πάσα στιγμή είτε ως εξυπηρετητής, είτε ως πελάτης, είτε ως δρομολογητής ανάλογα με τις συνθήκες που επικρατούν και τις ανάγκες που διαμορφώνονται στο δίκτυο κάθε στιγμή. Τα δίκτυα P2P επιτρέπουν το διαμοιρασμό απομακρυσμένων πόρων μεταξύ των μελών τους σε ευρεία κλίμακα, χωρίς τους περιορισμούς που θα επέβαλε κάποια κεντρική δομή ελέγχου.

1.1.2 Δίκτυα peer-to-peer

1.1.2.1 Γενικά

Ένα peer-to-peer δίκτυο υπολογιστών δεν έχει σταθερό αριθμών πελατών και εξυπηρετητών, αλλά

αποτελείται από ένα μεταβλητό αριθμό αλληλένδετων κόμβων. Κύρια χαρακτηριστικά των δικτύων αυτών είναι η ικανότητά τους να αυτο-οργανώνονται, η συμμετρική φύση της επικοινωνίας μεταξύ των συμμετεχόντων κόμβων και η απουσία κάποιας κεντρικής δομής ελέγχου. Έτσι, κάθε κόμβος μπορεί να στείλει ή να λάβει αιτήσεις εξυπηρέτησης ή απλά να προωθήσει μια εισερχόμενη αίτηση σε ένα γειτονικό κόμβο.

Τα δίκτυα peer-to-peer αποτελούν μια κατηγορία εφαρμογών που εκμεταλλεύεται τους αποθηκευτικούς πόρους, τους υπολογιστικούς κύκλους, το περιεχόμενο και την ανθρώπινη παρουσία που είναι διαθέσιμη στα διάφορα σημεία που συνδέονται μέσω Διαδικτύου. Ωστόσο, επειδή η πρόσβαση σε αυτούς τους κατανεμημένους πόρους συνεπάγεται λειτουργία σε ένα περιβάλλον όπου η συνδεσιμότητα μεταξύ των συμμετεχόντων δε μπορεί να θεωρηθεί δεδομένη και οι IP διευθύνσεις τους είναι μη προβλέψιμες, πρέπει οι κόμβοι ενός δικτύου peer-to-peer να λειτουργούν έξω από το σύστημα DNS και να μην εξαρτώνται από κεντρικούς εξυπηρετητές. Τα δομημένα peer-to-peer υπερκείμενα δίκτυα δεν κάνουν διακρίσεις στους ρόλους των κόμβων που συμμετέχουν σε αυτά, δε χρησιμοποιούν συγκεντρωμένες υπηρεσίες και δε χρειάζονται υποστήριξη από ISPs. Τέλος, χρησιμοποιούν μεγάλο όγκο πόρων χωρίς να απαιτείται κεντρικός σχεδιασμός ή μεγάλες επενδύσεις σε υλικό, εύρος ζώνης.

1.1.2.2 Αδόμητα peer-to-peer δίκτυα

Τα συστήματα peer-to-peer αποτελούν δημοφιλές πεδίο έρευνας στο χώρο της πληροφορικής και των επικοινωνιών, και χρησιμοποιούνται κυρίως για την υλοποίηση εντελώς αποκεντρωμένων δικτύων από αλληλένδετους κόμβους. Τέτοια δίκτυα ενδείκνυνται για εφαρμογές ομαδικής επικοινωνίας, αποθήκευσης αρχείων, κατανομής περιεχομένου και ανταλλαγής πόρων.

Τα πρώτα peer-to-peer συστήματα ήταν συστήματα ανταλλαγής αρχείων και αναφέρονται ως αδόμητα δίκτυα p2p. Ο χαρακτηρισμός «αδόμητα» αναφέρεται στη μη ύπαρξη μιας καθολικής δομής για τη ρύθμιση των μηχανισμών αναζήτησης και εύρεσης γειτόνων. Χαρακτηριστικοί εκπρόσωποι αυτής της γενειάς είναι το Napster, το Gnutella, το FastTrack (που έγινε γνωστό μέσω του KaZaA) και το Freenet.

Το Napster εμφανίστηκε στα μέσα του 1999 και μέχρι το τέλος του 2000 το λογισμικό του αποκτήθηκε από περισσότερους από 50 εκατομμύρια χρήστες. Σε αυτά τα συστήματα τα αρχεία αποθηκεύονται σε μηχανήματα τελικών χρηστών (peers) αντί σε έναν κεντρικό εξυπηρετητή και σε αντίθεση με το μοντέλο client-server, τα αρχεία ανταλλάσσονται μεταξύ των peers. Τα συστήματα ανταλλαγής αρχείων peer-to-peer μπορούν να οδηγήσουν σε νέα μοντέλα κατανομής περιεχομένου για εφαρμογές όπως η διανομή λογισμικού, η ανταλλαγή αρχείων και η στατική διανομή περιεχομένου web. Ωστόσο, η αρχιτεκτονική του Napster δε φάνηκε να παρουσιάζει ιδιαίτερες ικανότητες κλιμάκωσης ως προς το μέγεθος του δικτύου. Κι αυτό γιατί στο Napster ένας κεντρικός εξυπηρετητής αποθηκεύει τον κατάλογο όλων των αρχείων της κοινότητας χρηστών του Napster. Για την ανάκτηση ενός αρχείου ο χρήστης ρωτά τον κεντρικό εξυπηρετητή χρησιμοποιώντας το ευρύτερα γνωστό όνομα του αρχείου και παίρνει ως απάντηση την IP διεύθυνση του μηχανήματος που

αποθηκεύει το παραπάνω αρχείο. Έτσι, αν και το Napster χρησιμοποιεί μοντέλο επικοινωνίας peer-to-peer για τη μεταφορά του αρχείου, η διαδικασία εντοπισμού του αρχείου εξακολουθεί να είναι κεντρική με αποτέλεσμα να κοστίζει στο σύστημα η αυξομείωση του κεντρικού κατάλογου και να είναι τρωτό, αφού η αποτυχία εξαρτάται από ένα και μόνο σημείο (single point of failure).

Από την άλλη, το Gnutella προχωράει ένα βήμα και αποκεντρώνει τη διαδικασία εντοπισμού του αρχείου. Οι χρήστες ενός δικτύου Gnutella οργανώνονται σε ένα πλέγμα επιπέδου εφαρμογής, στο οποίο διακινούνται μαζικά οι αιτήσεις για ένα αρχείο με συγκεκριμένο περιεχόμενο. Η πλημμύρα για κάθε αίτηση προφανώς δεν είναι κλιμακούμενη ανάλογα με το μέγεθος του δικτύου και επειδή πρέπει να σταματήσει σε κάποιο σημείο, είναι πιθανό η αναζήτηση να αποτύχει και να μη βρεθεί το ζητούμενο περιεχόμενο στο σύστημα, αν και αυτό μπορεί να υπάρχει. Ωστόσο, παρά τη βελτίωση ως προς την κλιμάκωση, το Gnutella δε φαίνεται κι αυτό να προσαρμόζεται καλά στις απότομες αυξομειώσεις στο φορτίο του. Μάλιστα, τη μέρα μετά το κλείσιμο του Napster, το Gnutella φαίνεται να κατέρρευσαι υπό το φόρτο που προστέθηκε από νέους χρήστες προερχόμενους από το Napster.

Το FastTrack φαίνεται να ακολουθεί μια άλλη προσέγγιση στο θέμα της κλιμάκωσης, εισάγοντας την έννοια των «υπερκόμβων» (“superpeers”) και δομώντας το δίκτυο με σαφή ιεραρχία, θυμίζοντας την αρχιτεκτονική του συστήματος DNS. Ωστόσο, αυτή η αρχιτεκτονική αν και παρουσιάζει βελτιώσεις στον τομέα της κλιμάκωσης, πάσχει από το μειονέκτημα του κινδύνου της αποτυχίας κάποιου υπερκόμβου κοντά στην κορυφή της ιεραρχίας του δικτύου. Επιπλέον, και αυτή η αρχιτεκτονική, δε μπορεί να εγγυηθεί την επιτυχία μιας αναζήτησης ενός αντικειμένου.

Το Freenet είναι ένα δίκτυο ανταλλαγής αρχείων σχεδιασμένο για να αντιστέκεται στη λογοκρισία. Ούτε το Gnutella ούτε το Freenet εγγυώνται τον εντοπισμό των αρχείων – ακόμα και σε ένα κανονικό δίκτυο που λειτουργεί χωρίς μεγάλα προβλήματα.[1]

1.1.2.3 Δομημένα peer-to-peer δίκτυα

Λύση σε κάποια από τα προβλήματα που αντιμετωπίζουν τα αδόμητα συστήματα peer-to-peer δίνουν τα δομημένα peer-to-peer υπερκείμενα δίκτυα όπως είναι το Tapestry, το Chord, το Pastry, το CAN, το Kademlia, το Bamboo. Αυτά τα υπερκείμενα δίκτυα υλοποιούν ένα βασικό interface δρομολόγησης με βάση το κλειδί (key-based routing – KBR), το οποίο υποστηρίζει μία ντετερμινιστική δρομολόγηση μηνυμάτων στον κόμβο - υπεύθυνο για το κλειδί προορισμού. Επίσης, μπορούν να υποστηρίξουν interfaces υψηλότερου επιπέδου όπως έναν κατανεμημένο πίνακα κατακερματισμού (Distributed Hash Table – DHT) ή ένα στρώμα κατανεμημένου εντοπισμού και δρομολόγησης αντικειμένων (Distributed Object Location and Routing – DOLR). Αυτά τα συστήματα παρουσιάζουν καλές ιδιότητες κλιμάκωσης και εγγυώνται ότι οι ερωτήσεις βρίσκουν τα ζητούμενα αντικείμενα υπό φυσιολογικές συνθήκες.

Ένα χαρακτηριστικό που διαφοροποιεί αυτά τα συστήματα μεταξύ τους είναι το εάν λαμβάνουν υπόψη

δικτυακές αποστάσεις, όταν φτιάχνουν το υπερκείμενο στρώμα δρομολόγησης. Έτσι, στο CAN και στο Chord ένα υπερκείμενο βήμα μπορεί να είναι όσο η διάμετρος του δικτύου. Και τα δύο πρωτόκολλα δρομολογούν κατά μήκος της μικρότερης διαθέσιμης διαδρομής και χρησιμοποιούν σε χρόνο εκτέλεσης διάφορες τεχνικές για τη βελτιστοποίηση της συμπεριφοράς τους. Από την άλλη, το Tapestry και το Pastry φτιάχνουν τοπικά βελτιστοποιημένους πίνακες δρομολόγησης κατά την αρχικοποίηση και τους διατηρούν προκειμένου να μειώσουν την έκταση που θα πάρει η δρομολόγηση.

Μια καινούρια γενιά εφαρμογών έχει προταθεί πάνω σε αυτά τα συστήματα peer-to-peer επικυρώνοντάς τα ως υποδομές νέων και πιο εξελιγμένων εφαρμογών. Πολλά συστήματα παρέχουν multicast επιπέδου εφαρμογής: CAN-MC (CAN), Scribe (Pastry)[3] και Bayeux (Tapestry). Επιπλέον, έχουν προταθεί διάφορα αποκεντρωμένα συστήματα αρχείων: CFS (Chord), Mnemosyne (Chord, Tapestry), OceanStore (Tapestry) και PAST [5] (Pastry). Τα δομημένα peer- to-peer υπερκείμενα δίκτυα υποστηρίζουν επίσης νέες εφαρμογές, όπως τα δίκτυα αντοχής σε επιθέσεις, στρώματα δικτυακής ανακατεύθυνσης και η αναζήτηση ομοιότητας.

1.2 Τα βασικότερα δομημένα peer-to-peer συστήματα

1.2.1 CAN (Content-Addressable Networks)

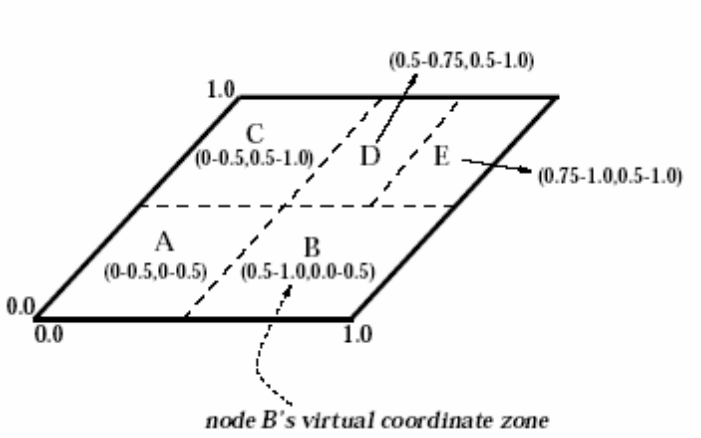
1.2.1.1 Γενικά

Η διαδικασία μεταφοράς ενός αρχείου σε ένα δίκτυο peer-to-peer είναι από τη φύση της κλιμακούμενη, αλλά η πραγματική δυσκολία στη σχεδίαση ενός τέτοιου δικτύου έγκειται στην εύρεση του κόμβου που διαθέτει το αρχείο (lookup process). Τα δίκτυα διευθυνσιοδότησης σύμφωνα με το περιεχόμενο (Content-Addressable Networks - CANs) είναι δίκτυα που χρησιμοποιούν ένα σχήμα δεικτοδότησης για την αντιστοίχηση ονομάτων αρχείων σε θέσεις στο σύστημα.

Τα CAN μοιάζουν με πίνακες κατακερματισμού (hash tables) : οι βασικές λειτουργίες που επιτελούν είναι η εισαγωγή, η αναζήτηση και η διαγραφή ζευγαριών (κλειδιών, τιμών). Κάθε τέτοιο δίκτυο αποτελείται από διάσπαρτους, μεμονωμένους κόμβους, καθένας από τους οποίους αποθηκεύει κάποιο συγκεκριμένο κομμάτι του συνολικού πίνακα κατακερματισμού, το οποίο καλείται ζώνη. Επιπλέον, κάθε κόμβος διατηρεί πληροφορίες για ένα μικρό αριθμό γειτονικών κόμβων. Οι αιτήσεις για ένα συγκεκριμένο κλειδί δρομολογούνται από ενδιάμεσους κόμβους που συμμετέχουν στο CAN, προς τον κόμβο του οποίου η ζώνη περιέχει το κλειδί που αναζητείται. Το δίκτυο CAN, όπως φαίνεται και από τα παραπάνω, είναι πλήρως κατανεμημένο (δεν απαιτείται καμιάς μορφής κεντρική δομή ελέγχου), κλιμακούμενο (κάθε κόμβος διατηρεί πληροφορίες μόνο για ένα μικρό νούμερο γειτόνων του, ανεξάρτητα από το μέγεθος τού δικτύου) και ανθεκτικό σε αποτυχίες (οι κόμβοι μπορούν να παρακάμψουν κόμβους που φαίνεται να έχουν φύγει από το δίκτυο). Επίσης, δε χρησιμοποιεί καμιά μορφή ιεραρχικής δομής ονοματοδοσίας (όπως το DNS ή η IP δρομολόγηση), για να πετύχει καλή κλιμάκωση και μπορεί να υλοποιηθεί εξολοκλήρου σε επίπεδο εφαρμογής.[7]

1.2.1.2 Σχεδιασμός ενός δικτύου CAN

Το κυριότερο στοιχείο αυτής της σχεδίασης είναι ένας εικονικός d-διάστατος καρτεσιανός χώρος συντεταγμένων. Ο χώρος αυτός αποτελεί μια λογική κατασκευή και δεν σχετίζεται με κανένα πραγματικό σύστημα συντεταγμένων. Σε κάθε χρονική στιγμή ολόκληρος ο χώρος κατανέμεται δυναμικά ανάμεσα σε όλους τους κόμβους του συστήματος, έτσι ώστε κάθε κόμβος να έχει τη δική του ζώνη μέσα στο χώρο, για την οποία να είναι υπεύθυνος. Μια εικόνα του παραπάνω εικονικού χώρου και των ζωνών στις οποίες αυτός χωρίζεται δίνεται στην εικόνα που ακολουθεί.



Εικόνα 1 : Διδιάστατος [0,1]*[0,1] χώρος συντεταγμένων

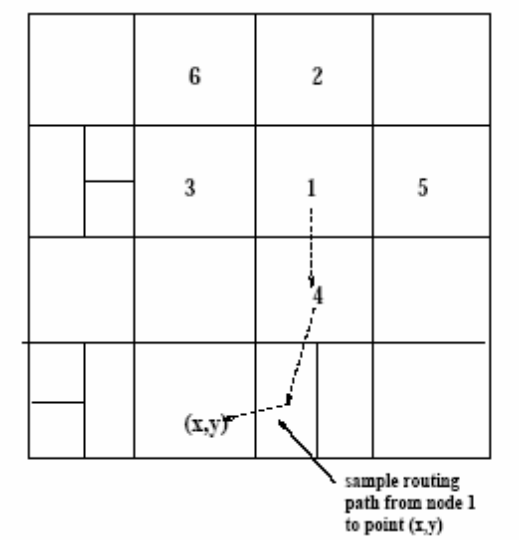
Αυτός ο εικονικός χώρος συντεταγμένων χρησιμοποιείται, για να αποθηκεύσει ζεύγη της μορφής (κλειδί, τιμή). Για την αποθήκευση του ζεύγους (K1,V1), το κλειδί K1 αντιστοιχίζεται συγκεκριμένα σε ένα σημείο P του χώρου με τη χρήση μιας ομοιόμορφης συνάρτησης κατακερματισμού (hash function). Το ζεύγος (K1,V1) θα αποθηκευτεί στον κόμβο, ο οποίος είναι υπεύθυνος για τη ζώνη, στην οποία ανήκει το σημείο P. Για την ανάκτηση του αντικειμένου με κλειδί K1, ο εκάστοτε κόμβος μπορεί να εφαρμόσει την ίδια συνάρτηση κατακερματισμού, για να αντιστοιχίσει το κλειδί K1 στο σημείο P και μετά να ανακτήσει την αντίστοιχη τιμή από το P. Εάν το σημείο P δεν ανήκει στον κόμβο που κάνει την αναζήτηση ή στους γείτονές του, η αίτηση πρέπει να δρομολογηθεί μέσω της CAN υποδομής μέχρι να φτάσει στον κόμβο με τη ζώνη που περιέχει το P. Συνεπώς, η αποδοτική δρομολόγηση είναι ένα κρίσιμο κομμάτι του CAN.

Οι κόμβοι στο CAN αυτο-οργανώνονται σε ένα υπερκείμενο δίκτυο, το οποίο αναπαριστά τον εικονικό χώρο συντεταγμένων. Κάθε κόμβος μαθαίνει και διατηρεί τις IP διευθύνσεις των κόμβων που κατέχουν τις ζώνες, οι οποίες γειτνιάζουν στο χώρο συντεταγμένων με τη δική του ζώνη. Αυτό το σύνολο των άμεσων γειτόνων που διατηρείται από κάθε κόμβο είναι, ουσιαστικά, ένας πίνακας δρομολόγησης που επιτρέπει τη δρομολόγηση ενός μηνύματος μεταξύ δύο οποιονδήποτε κόμβων.[7]

1.2.1.3 Δρομολόγηση στο CAN

Όπως αναφέρθηκε και παραπάνω, ένας κόμβος CAN διατηρεί έναν πίνακα δρομολόγησης με τις IP διευθύνσεις και εικονικές ζώνες όλων των άμεσων γειτόνων του στο χώρο των συντεταγμένων. Σε έναν d-διάστατο χώρο δύο κόμβοι θεωρούνται γειτονικοί, εάν τα διαστήματα των συντεταγμένων τους δεν επικαλύπτονται για (d-1) διαστάσεις και συνορεύουν κατά μήκος μίας μόνο διάστασης. Έτσι, στην εικόνα που ακολουθεί ο κόμβος 5 είναι γείτονας του κόμβου 1, ενώ ο κόμβος 6 δεν είναι γείτονας του κόμβου 1.

Ο πίνακας δρομολόγησης περιέχει αρκετή πληροφορία για την επιτυχή δρομολόγηση ανάμεσα σε δύο αυθαίρετα σημεία στο χώρο των συντεταγμένων. Η μόνη πληροφορία που απαιτείται να φέρει ένα μήνυμα CAN είναι οι συντεταγμένες του προορισμού του. Χρησιμοποιώντας τις συντεταγμένες των γειτόνων του, ένας κόμβος δρομολογεί ένα μήνυμα προς τον προορισμό του προωθώντας το στο γείτονα με τις πλησιέστερες συντεταγμένες στις συντεταγμένες του προορισμού. Στην παρακάτω εικόνα (Εικόνα 2) φαίνεται ένα απλό μονοπάτι δρομολόγησης.



Εικόνα 2 : Παράδειγμα διδιάστατου χώρου. Το σύνολο γειτόνων

Από την παραπάνω περιγραφή της οργάνωσης των κόμβων σε ένα δίκτυο CAN, καθίσταται σαφές ότι υπάρχουν πολλά διαφορετικά μονοπάτια ανάμεσα σε δύο σημεία στο χώρο. Αυτό συνεπάγεται ότι, εάν ο γείτονας ενός κόμβου είναι εκτός δικτύου, ο κόμβος μπορεί αυτόματα να δρομολογήσει την αίτηση του κατά μήκος της επόμενης καλύτερης διαδρομής. Εάν, ωστόσο, ένας κόμβος χάσει όλους τους γείτονές του προς μία συγκεκριμένη κατεύθυνση και οι διορθωτικοί μηχανισμοί δεν καταφέρουν να γεμίσουν το κενό που δημιουργείται στο χώρο των συντεταγμένων, τότε η «άπληστη» προώθηση μπορεί να αποτύχει προσωρινά. Σε αυτήν την περίπτωση, ο κόμβος μπορεί να χρησιμοποιήσει μια αναζήτηση διευρυνόμενου δακτυλίου (με ελεγχόμενη πλημμύρα στο unicast υπερκείμενο πλέγμα CAN), για να εντοπίσει έναν κόμβο που είναι πλησιέστερα στον προορισμό από τον εαυτό του. Έπειτα, το μήνυμα προωθείται σε αυτόν τον κόμβο και επαναφέρεται η «άπληστη» προώθηση. [7]

1.2.1.4 Δόμηση του CAN

Όλοι οι κόμβοι σε ένα CAN είναι υπεύθυνοι για τη δική τους ζώνη, δηλαδή για το δικό τους σύνολο αποθηκευμένων δεδομένων τού δικτύου. Για να επιτραπεί σε ένα CAN να αυξήσει τον αριθμό των κόμβων του, κάθε καινούριος κόμβος που προσχωρεί στο σύστημα θα πρέπει να αποκτήσει τη δική του ζώνη στο χώρο των συντεταγμένων.

Αυτό μπορεί να γίνει, εφόσον ένας ήδη υπάρχων κόμβος χωρίσει στη μέση τη δική του ζώνη και παραχωρήσει τη μισή στον καινούριο κόμβο.

Συγκεκριμένα, η διαδικασία ένταξης ενός νέου κόμβου στο δίκτυο ολοκληρώνεται σε τρία βήματα:

• *Ο νέος κόμβος βρίσκει έναν ήδη υπάρχοντα κόμβο στο CAN :*

Ένας νέος κόμβος CAN πρώτα ανακαλύπτει την IP διεύθυνση κάποιου κόμβου που είναι ήδη στο σύστημα. Η λειτουργία του CAN δεν εξαρτάται από τον τρόπο που γίνεται αυτό. Υποθέτουμε ότι ένα CAN έχει ένα DNS όνομα δικτύου, το οποίο αναλύεται στην IP διεύθυνση ενός ή περισσοτέρων κόμβων αρχικοποίησης (bootstrap nodes) του CAN. Ένας κόμβος αρχικοποίησης διατηρεί μια μερική λίστα των κόμβων CAN που πιστεύει ότι ανήκουν στο σύστημα. Για να προσχωρήσει στο CAN, ο νέος κόμβος αναζητά το όνομα δικτύου του CAN στο DNS, για να αποκτήσει την IP διεύθυνση ενός κόμβου αρχικοποίησης. Στη συνέχεια, ο κόμβος αρχικοποίησης δίνει τις IP διευθύνσεις μερικών τυχαία επιλεγμένων κόμβων του συστήματος.

• *Με χρήση των μηχανισμών δρομολόγησης CAN βρίσκει έναν κόμβο, του οποίου η ζώνη θα μοιραστεί :*

Ο νέος κόμβος επιλέγει τυχαία ένα σημείο P στο χώρο και στέλνει μία αίτηση JOIN με προορισμό το σημείο P. Αυτό το μήνυμα στέλνεται στο CAN μέσω οποιουδήποτε υπάρχοντος κόμβου CAN και προωθείται σύμφωνα με τους μηχανισμούς δρομολόγησης στον κόμβο που είναι υπεύθυνος για το P. Στη συνέχεια, ο κόμβος που είναι υπεύθυνος για το P, χωρίζει τη ζώνη του στα δύο και αναθέτει το ένα κομμάτι στον καινούριο κόμβο. Αυτός ο χωρισμός γίνεται υποθέτοντας μια συγκεκριμένη ταξινόμηση των διαστάσεων για την επιλογή της διάστασης που θα χωριστεί, ώστε οι ζώνες να μπορούν να ενωθούν, όταν κόμβοι αποχωρούν από το σύστημα. Έτσι, σε ένα δισδιάστατο χώρο μια ζώνη πρώτα θα χωριζόταν κατά την X διάσταση, μετά κατά την Y κ.ο.κ. Τα ζεύγη (κλειδί, τιμή) της μισής ζώνης που παραχωρείται μεταφέρονται στον καινούριο κόμβο.

• *Οι γείτονες της ζώνης που μοιράστηκε ενημερώνονται, ώστε να συμπεριληφθεί στους πίνακες δρομολόγηση ο νέος κόμβος* :

Αφού έχει αποκτήσει τη ζώνη του, ο νέος κόμβος μαθαίνει τις IP διευθύνσεις των γειτόνων του από τον κόμβο που του παραχώρησε τη μισή ζώνη του. Προφανώς, και αυτός ο κόμβος ανήκει στο σύνολο των γειτόνων του νέου κόμβου και πρέπει και αυτός να ανανεώσει τη λίστα των γειτόνων του και να διαγράψει αυτούς που δεν ανήκουν πλέον σε αυτή. Τέλος, οι γείτονες του νέου και του παλιού κόμβου πρέπει να ενημερωθούν για την αλλαγή στο χώρο των συντεταγμένων. Κάθε κόμβος στο σύστημα στέλνει ένα άμεσο μήνυμα ενημέρωσης στους γείτονές του, ακολουθούμενο από περιοδικές ενημερώσεις, με περιεχόμενο τη ζώνη που του έχει ανατεθεί τη δεδομένη χρονική στιγμή. Αυτές οι ενημερώσεις "χαλαρού" τύπου (soft-state style)

εξασφαλίζουν ότι όλοι οι γείτονες θα μάθουν γρήγορα τις τυχόν αλλαγές και θα ενημερώσουν ανάλογα το σύνολο των γειτόνων τους.

Η πρόσθεση καινούριων κόμβων επηρεάζει μόνο ένα μικρό αριθμό κόμβων σε μια μικρή περιοχή του χώρου των συντεταγμένων. Ο αριθμός των γειτόνων ενός κόμβου εξαρτάται μόνο από τις διαστάσεις του χώρου και είναι ανεξάρτητος του συνολικού αριθμού των κόμβων στο σύστημα. Έτσι, η εισαγωγή ενός κόμβου επηρεάζει μόνο O(αριθμός διαστάσεων) υπάρχοντες κόμβους, πράγμα πολύ σημαντικό για ένα CAN με πάρα πολλούς κόμβους.[7]

1.2.1.5 Αποχώρηση κόμβου, ανάκαμψη και διατήρηση του CAN

Όταν κόμβοι αποχωρούν από ένα CAN, είναι απαραίτητο να διασφαλίζεται ότι οι ζώνες που είχαν καταλάβει θα περάσουν υπό την επίβλεψη των κόμβων που παραμένουν. Αυτό γίνεται με την παραχώρηση της ζώνης και της σχετικής βάσης δεδομένων (κλειδί, τιμή) του αποχωρούντος κόμβου σε κάποιον από τους γείτονές του. Εάν η ζώνη ενός γείτονα μπορεί να ενωθεί με τη ζώνη του αποχωρούντος κόμβου παράγοντας μία έγκυρη ζώνη, τότε γίνεται η ένωση. Εάν αυτό δεν είναι εφικτό, τότε η ζώνη παραχωρείται στο γείτονα με τη μικρότερη ζώνη, ο οποίος προσωρινά θα μεταχειρίζεται δύο ζώνες.

Το CAN πρέπει επίσης να παραμένει σταθερό παρά τις αποτυχίες κόμβων ή δικτύου (όταν ένας ή περισσότεροι κόμβοι χάνουν συνδεσιμότητα). Αυτό ρυθμίζεται με έναν άμεσο αλγόριθμο απόκτησης ελέγχου, ο οποίος διασφαλίζει ότι ένας γείτονας του νεκρού κόμβου θα αναλάβει τη ζώνη του. Ωστόσο, σε αυτήν την περίπτωση τα ζεύγη (κλειδί, τιμή) του νεκρού κόμβου χάνονται μέχρι να ανανεωθεί η κατάσταση από τους κόμβους που κρατούν τα δεδομένα.

Υπό κανονικές συνθήκες ένας κόμβος στέλνει περιοδικά μηνύματα ανανέωσης σε κάθε έναν από τους γείτονές του περικλείοντας τις συντεταγμένες της ζώνης για την οποία είναι υπεύθυνος και τη λίστα των γειτόνων του με τις συντεταγμένες των ζωνών τους. Μία παρατεταμένη απουσία ενός τέτοιου μηνύματος από ένα γείτονα ερμηνεύεται ως αποτυχία κόμβου. Μόλις ένας κόμβος αποφασίσει ότι ένας γείτονας του είναι νεκρός, ξεκινά το μηχανισμό κατάληψης μαζί με ένα χρονιστή απόκτησης ελέγχου. Κάθε γείτονας του νεκρού κόμβου θα δράσει ανάλογα, με τον χρονιστή ενεργοποιημένο σε αναλογία με το μέγεθος της ζώνης του. Όταν ο χρονιστής αποπνέει, ο κόμβος στέλνει ένα μήνυμα TAKEOVER μαζί με το μέγεθος της ζώνης του σε όλους τους γείτονες του νεκρού κόμβου.

Λαμβάνοντας ένα μήνυμα TAKEOVER, κάθε κόμβος ακυρώνει τον χρονιστή του, εάν το μέγεθος της ζώνης στο μήνυμα είναι μικρότερο από το μέγεθος της δικής του ζώνης. Διαφορετικά, απαντά με το δικό του μήνυμα TAKEOVER. Με αυτόν τον τρόπο, επιλέγεται ο γείτονας-κόμβος που είναι ζωντανός και έχει τη μικρότερη ζώνη σε μέγεθος.

Στην περίπτωση που «πέσουν» ταυτόχρονα πολλοί διπλανοί κόμβοι, είναι δυνατό να ανιχνευτεί η αποτυχία, εάν λιγότεροι από τους μισούς γείτονες του νεκρού κόμβου είναι ακόμα προσιτοί. Εάν ένας κόμβος

αναλάβει μια άλλη ζώνη υπό αυτές τις συνθήκες, μπορεί το CAN να βρεθεί σε ασυνεπή κατάσταση. Σε αυτήν την περίπτωση, πριν πυροδοτηθεί ο μηχανισμός διόρθωσης, ο κόμβος ξεκινά μια επεκτεινόμενη αναζήτηση δακτυλίου για κόμβους που βρίσκονται πέρα από την περιοχή αποτυχίας και έτσι αποκτά ξανά την απαραίτητη πληροφορία για τους γειτονικούς κόμβους και μπορεί να εκκινήσει πάλι την κατάληψη με ασφάλεια.

Τέλος, και η κανονική διαδικασία αποχώρησης και ο άμεσος αλγόριθμος κατάληψης μπορούν να καταλήξουν σε έναν κόμβο που κρατά περισσότερες από μία ζώνες. Για να αποφευχθεί περαιτέρω τεμαχισμός του χώρου, ένας αλγόριθμος συναρμολόγησης ζωνών τρέχει στο παρασκήνιο, για να διασφαλίσει ότι το CAN θα τείνει πάλι σε μία ζώνη ανά κόμβο. [7]

1.2.1.6 Σχεδιαστικές βελτιώσεις

Ο βασικός αλγόριθμος CAN, όπως περιγράφτηκε παραπάνω, χαρακτηρίζεται από μια ισορροπία ανάμεσα σε πληροφορία ανά κόμβο (O(d) για d-διάστατο χώρο) και σε σύντομα μήκη διαδρομής (O(d n1/d) βήματα για d διαστάσεις και n κόμβους). Αυτά τα βήματα αναφέρονται σε επίπεδο εφαρμογής και όχι σε επίπεδο IP, πράγμα που σημαίνει ότι η καθυστέρηση σε κάθε βήμα μπορεί να είναι μεγάλη. Δύο κόμβοι, οι οποίοι είναι διπλανοί στο CAN, μπορεί να είναι πολλά χιλιόμετρα και πολλά βήματα IP μακριά. Η μέση συνολική καθυστέρηση μιας αναζήτησης ισούται με το μέσο αριθμό των βημάτων CAN επί τη μέση καθυστέρηση ενός βήματος CAN. Επιθυμητή είναι μια καθυστέρηση αναζήτησης, η οποία να είναι συγκρίσιμη με τις υποκείμενες IP καθυστερήσεις ανάμεσα στον αιτούντα και στον κόμβο CAN που κρατά το κλειδί.

Στη συνέχεια παρουσιάζονται τεχνικές που στόχο έχουν την βελτίωση του χρόνου αναζήτησης σε ένα δίκτυο CAN. Πολλές από τις παρακάτω τεχνικές, πέρα από τη βελτίωση του χρόνου, βελτιώνουν και την αξιοπιστία ενός δικτύου CAN προσφέροντας τόσο μεγαλύτερη διαθεσιμότητα δεδομένων, όσο και πολλές εναλλακτικές διαδρομές για την εύρεση ενός δεδομένου.

- *Πολυδιάστατος χώρος συντεταγμένων*

Αυξάνοντας τη διάσταση του χώρου συντεταγμένων μειώνεται το μήκος της διαδρομής κι άρα η καθυστέρηση που εισάγεται λόγω της δικτυακής απόστασης των κόμβων, με αντάλλαγμα μία μικρή αύξηση στον πίνακα δρομολόγησης κάθε κόμβου.

Επιπλέον, πέρα από τη μείωση στην καθυστέρηση, η αύξηση της διάστασης τού χώρου συντεταγμένων συνεπάγεται την αύξηση των γειτόνων που κάθε κόμβος διατηρεί στον πίνακα δρομολόγησης του, κι άρα τη μεγαλύτερη ανοχή του δικτύου σε ταυτόχρονες αποχωρήσεις κόμβων.

- *Multiple Realities: πολλαπλοί χώροι συντεταγμένων*

Μία ακόμα τεχνική η οποία λέγεται πολλαπλές πραγματικότητες, κάνει χρήση πολλαπλών χώρων συντεταγμένων με σκοπό να βελτιώσει και το χρόνο εύρεσης ενός αντικειμένου στο DHT αλλά και την ανθεκτικότητα του CAN.

Σύμφωνα με την τεχνική αυτή, σε κάθε κόμβο ανατίθεται μία συγκεκριμένη ζώνη δεδομένων σε κάθε χώρο συντεταγμένων. Αντίγραφα των κλειδιών υπάρχουν σε κάθε χώρο συντεταγμένων, βελτιώνοντας έτσι την ανθεκτικότητα του δικτύου σε περίπτωση αποτυχίας ενός κόμβου. Για την προώθηση ενός μηνύματος, ένας κόμβος ελέγχει τους γείτονές του σε κάθε χώρο συντεταγμένων και προωθεί το μήνυμα στον κοντινότερο από αυτούς.

- *Επιλογή γειτόνων βάσει κάποιας δικτυακής μετρικής*

Κατ' αυτή την τεχνική, κάθε κόμβος μετράει το RTT καθενός από τους γείτονές του. Έτσι, όταν λάβει ένα μήνυμα αναζήτησης, το προωθεί στον γείτονα με τον καλύτερο λόγο εγγύτητας στον προορισμό στο συγκεκριμένο χώρο συντεταγμένων προς το RTT του γείτονα.

- *Ανάθεση κάθε ζώνης σε παραπάνω του ενός κόμβου*

Σε αυτή την τεχνική, κάθε ζώνη, αντίθετα με ότι γινόταν ως τώρα, ανατίθεται, σε παραπάνω του ενός κόμβου. Κόμβοι υπεύθυνοι για την ίδια ζώνη δεδομένων αποκαλούνται ομότιμοι.

Η εν λόγω τεχνική, βελτιώνει την καθυστέρηση της αναζήτησης μειώνοντας το μήκος της διαδρομής προς το δεδομένο που ψάχνουμε, μειώνει την per hop καθυστέρηση αφού τώρα κάθε κόμβος έχει τη δυνατότητα να επιλέξει μεταξύ πολλών κόμβων για την κάθε θέση στον πίνακα δρομολόγησης του (και φυσικά θα επιλέξει αυτόν με το μικρότερο RTT) και τέλος αυξάνει την ανθεκτικότητα του δικτύου σε αποτυχίες κόμβων, λόγω του ότι πλέον, χρειάζεται η αποτυχία περισσότερων του ενός κόμβου για να καταστεί μια ζώνη δεδομένων μη διαθέσιμη.

- *Χρήση πολλαπλών συναρτήσεων hash*

Με τη χρήση k συναρτήσεων κατακερματισμού, κάθε ζεύγος (κλειδί, τιμή) ανατίθεται σε k διαφορετικούς κόμβους. Με τον τρόπο αυτό για να χαθεί μια τιμή, απαιτείται η ταυτόχρονη αποτυχία k κόμβων. Επιπλέον, ερωτήματα για ένα συγκεκριμένο δεδομένο μπορούν να σταλούν ταυτόχρονα και προς τους k κόμβους που το έχουν με αποτέλεσμα τη μείωση της μέσης καθυστέρησης αναζήτησης.

- *Κατασκευή του CAN δικτύου λαμβάνοντας υπόψιν την τοπολογία των κόμβων*

Η βασική ιδέα πίσω από αυτή την τεχνική είναι ότι γεωγραφικά γειτονικοί κόμβοι, θα έχουν και μικρότερη "δικτυακή απόσταση" με όρους RTT. Για το σκοπό αυτό γίνεται χρήση κάποιων σταθερών σημείων στο διαδίκτυο που ονομάζονται φάροι, όπως DNS servers, όπου με βάση το RTT κάθε κόμβου από αυτούς τους φάρους γίνεται και η ανάθεση των ζωνών.

- *Καλύτερη ισοκατανομή των ζωνών*

Όταν ένας κόμβος μπαίνει στο δίκτυο, ένα μήνυμα JOIN αποστέλλεται στον υπεύθυνο κόμβο ενός τυχαίου σημείου στο χώρο συντεταγμένων. Αυτός ο κόμβος, πέρα από τις συντεταγμένες της δικής του ζώνης, ξέρει και αυτές των ζωνών των γειτόνων του. Έτσι, αντί να διασπάσει κατευθείαν τη δική του ζώνη και να την αναθέσει στον νέο κόμβο, συγκρίνει αρχικά τον όγκο της ζώνης του με αυτόν των γειτονικών του κόμβων. Τελικά, η ζώνη που διασπάται είναι αυτή με το μεγαλύτερο όγκο.

- *Caching και Διατήρηση Αντιγράφων Δημοφιλών Δεδομένων*

Όπως και στο διαδίκτυο, πολλά δεδομένα είναι πιο δημοφιλή από άλλα. Για να είναι τα δεδομένα αυτά άμεσα διαθέσιμα και να μην υπερφορτώνονται και οι κόμβοι που τα φιλοξενούν, γίνεται χρήση τεχνικών που τις συναντάμε στο διαδίκτυο, όπως :

Caching : Κάθε κόμβος, εκτός από τον πίνακα δρομολόγησης του κρατά και μια cache με τα κλειδιά των δεδομένων που έχει ήδη επισκεφθεί. Έτσι, κάθε φορά που πρέπει να προωθήσει μία αίτηση αναζήτησης κάποιου κλειδιού, ελέγχει πρώτα την cache κι αν το βρει εκεί, τότε ικανοποιεί μόνος του την αίτηση χωρίς να προωθήσει το μήνυμα παρακάτω. Έτσι, η διαθεσιμότητα των δεδομένων καθίσταται ανάλογη του πόσο δημοφιλή είναι.

Διατήρηση Αντιγράφων : Ένας κόμβος που λαμβάνει πολλές αιτήσεις για ένα συγκεκριμένο δεδομένο μπορεί να το δημιουργήσει αντίγραφά του σε γειτονικούς του κόμβους. Έτσι, όταν ένας κόμβος, που διατηρεί ένα αντίγραφο του συγκεκριμένου δεδομένου, λάβει μια αίτηση για αυτό, μπορεί είτε να την εξυπηρετήσει ο ίδιος είτε να την εκτρέψει προς έναν άλλο γειτονικό κόμβο που έχει ένα αντίγραφο του συγκεκριμένου δεδομένου. Με τον παραπάνω τρόπο επιτυγχάνεται ο διαμοιρασμός του φόρτου που συνεπάγεται ένα δημοφιλές δεδομένο σε μία ομάδα κόμβων παρά η συνολική εξυπηρέτηση των αιτήσεων από έναν κόμβο.[7]

1.2.2 Chord

1.2.2.1 Γενικά

Το Chord είναι μια κατανεμημένη υπηρεσία αναζήτησης, η οποία είναι επιδεκτική σε ευρεία κλιμάκωση, πλήρως αποκεντρωμένη και μπορεί να χρησιμοποιηθεί ως βάση για συστήματα peer-to-peer γενικού σκοπού. Τα πιο πολλά συστήματα peer-to-peer εμφανίζουν κάποια χαρακτηριστικά που ενισχύουν την απόδοση τους, όπως η εφεδρική αποθήκευση, η μονιμότητα και αποδοτική τοποθέτηση των δεδομένων, η επιλογή των κοντινών εξυπηρετητών, η πιστοποίηση και η ιεραρχική ονοματοδοσία.

Το Chord δεν υλοποιεί αυτές τις υπηρεσίες στον πυρήνα του, αλλά μάλλον επιλέγει μια εύκαμπτη και υψηλής απόδοσης αρχή αναζήτησης, πάνω στην οποία μπορούν να βασιστούν αποδοτικά οι παραπάνω λειτουργίες. Με άλλα λόγια, η σχεδιαστική φιλοσοφία έγκειται στο διαχωρισμό του προβλήματος της αναζήτησης από την επιπλέον λειτουργικότητα. Τοποθετώντας επιπλέον χαρακτηριστικά πάνω σε μία υπηρεσία αμιγούς αναζήτησης, το σύστημα κερδίζει σε ευρωστία και δυνατότητα κλιμάκωσης.

Είναι σχεδιασμένο έτσι ώστε να προσφέρει την απαραίτητη λειτουργικότητα για την υλοποίηση συστημάτων γενικού σκοπού διατηρώντας παράλληλα τη μέγιστη ελαστικότητα. Παρέχει μία μονοσήμαντη αντιστοίχηση ανάμεσα στο χώρο των αναγνωριστικών και στο σύνολο των κόμβων. Ένας κόμβος μπορεί να είναι ένας υπολογιστής ή μια διεργασία με IP διεύθυνση και αριθμό πόρτας. Κάθε κόμβος συνδέεται με ένα αναγνωριστικό Chord. [14, 22]

1.2.2.2 Το πρωτόκολλο του Chord

Στην καρδία του πρωτοκόλλου του Chord βρίσκεται ο γρήγορος κατανεμημένος υπολογισμός μιας συνάρτησης κατακερματισμού, που αντιστοιχίζει κλειδιά σε κόμβους, οι οποίοι και είναι υπεύθυνοι για αυτά. Το Chord χρησιμοποιεί συνεχή κατακερματισμό, μια τεχνική με πολλά καλά χαρακτηριστικά μεταξύ των οποίων είναι η ομοιόμορφη κατανομή των κλειδιών στους κόμβους καθώς και η εξασφάλιση ότι, κατά πάσα πιθανότητα, όταν ένας κόμβος μπαίνει ή βγαίνει από το δίκτυο, μόνο ένα O(1/N) κλάσμα των κλειδιών θα κληθεί να αλλάξει θέση μέσα στο δίκτυο.

Η συνάρτηση συνεχούς κατακερματισμού αντιστοιχεί σε κάθε κόμβο ένα αναγνωριστικό των m-bit, κάνοντας χρήση μιας συνάρτησης βάσης όπως η SHA-1. Το αναγνωριστικό αυτό παράγεται από την εφαρμογή της συνάρτησης κατακερματισμού στην IP του κόμβου, ενώ το αναγνωριστικό ενός δεδομένου προς αποθήκευση, παράγεται από την εφαρμογή της παραπάνω συνάρτησης στο κλειδί τού δεδομένου. Το μέγεθος του κλειδιού πρέπει να είναι αρκούντως μεγάλο ώστε να αποφευχθεί το ενδεχόμενο δύο κλειδιά ή κόμβοι να έχουν την ίδια τιμή κατακερματισμού.

Μέσω του συνεχούς κατακερματισμού, τα δεδομένα ανατίθενται σε κάποιον κόμβο του δικτύου με τον

παρακάτω τρόπο : τα αναγνωριστικά διατάσσονται σε ένα κύκλο modulo 2^m μονοδιάστατων αναγνωριστικών με το 0 να ακολουθεί το μεγαλύτερο αναγνωριστικό. Το κλειδί k ανατίθεται στον κόμβο του οποίου το αναγνωριστικό είναι ίσο με το k ή είναι το μικρότερο που έπεται του k στον κύκλο των αναγνωριστικών. Αυτός ο κόμβος καλείται ο *διάδοχος(successor)* του κλειδιού k.

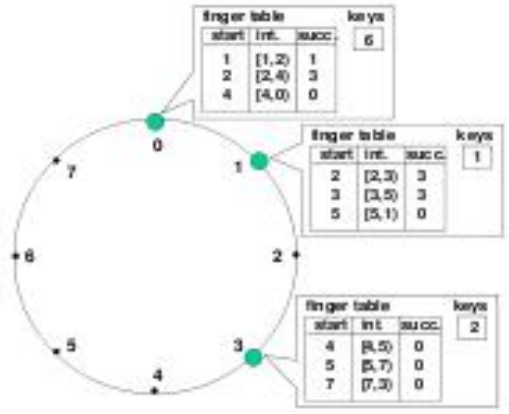
Με δεδομένα τα όσα περιγράψαμε παραπάνω, κάθε κόμβος αρκεί να ξέρει τον άμεσο διάδοχο του, δηλαδή τον κόμβο με το μικρότερο αναγνωριστικό που έπεται του δικού του για να είναι σε θέση να δρομολογεί αιτήσεις επιτυχώς. Έτσι, παρατηρούμε ότι μια μικρή ποσότητα πληροφορίας αρκεί για την ορθή δρομολόγηση μηνυμάτων σε ένα κατανεμημένο περιβάλλον.

Η αναζήτηση ενός αναγνωριστικού σε ένα δίκτυο Chord, μπορεί να επιτευχθεί ως εξής : το ερώτημα για ένα κλειδί, προωθείται από ένα κόμβο(που δεν το έχει) στον κόμβο που έπεται στον κύκλο των αναγνωριστικών, έως ότου φτάσει στο διάδοχο τού κλειδιού. Το παραπάνω, αν και εξασφαλίζει την ορθότητα στο Chord, δεν είναι αποδοτικό αφού στη χειρότερη περίπτωση μπορεί να χρειαστούν N hops για την εύρεση του κλειδιού. Για λόγους αποδοτικότητας, κάθε κόμβος στο Chord διατηρεί κι επιπλέον πληροφορία δρομολόγησης.

Notation	Definition
$finger[k].start$	$(n + 2^{k-1}) \bmod 2^m, 1 \leq k \leq m$
$.interval$	$[finger[k].start, finger[k+1].start)$
$.node$	first node $\geq n.finger[k].start$
$successor$	the next node on the identifier circle; $finger[1].node$
$predecessor$	the previous node on the identifier circle

Εικόνα 3 : Ορισμός των μεταβλητών ενός κόμβου n με αναγνωριστικό των m bits

Κάθε κόμβος του Chord διατηρεί έναν πίνακα δρομολόγησης με το πολύ m εγγραφές, όπου m το πλήθος των bits των αναγνωριστικών. Ο πίνακας αυτός καλείται finger table του κόμβου. Στη θέση i του finger table του κόμβου n, περιέχονται πληροφορίες σχετικά με τον πρώτο κόμβο του οποίου το αναγνωριστικό έπεται του n κατά 2^{i-1}. Από το παραπάνω φαίνεται ότι οι θέσεις στο finger table αντιστοιχούν σε διαστήματα στο χώρο των αναγνωριστικών. Έτσι , στη θέση i αντιστοιχεί ο κόμβος όπου προωθούνται ερωτήματα για το διάστημα $(n+2^{i-1}, n+2^i)$.



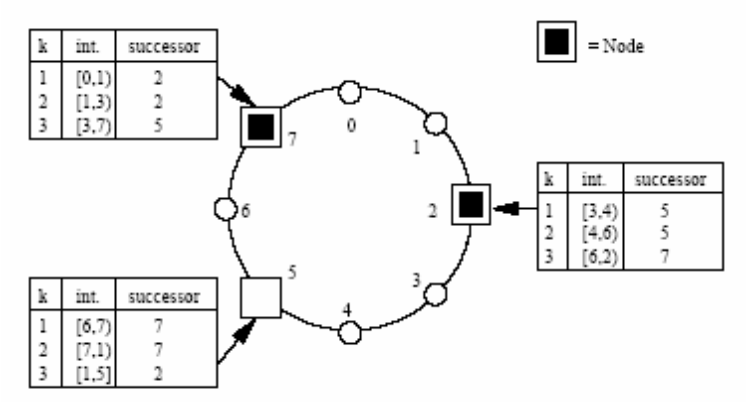
Εικόνα 4 : Στο παραπάνω σχήμα φαίνεται τα finger tables και τα κλειδιά

Από τα παραπάνω μπορούμε να εξάγουμε τα ακόλουθα σημαντικά χαρακτηριστικά για το Chord :

1. *Κάθε κόμβος αποθηκεύει πληροφορίες για ένα μικρό αριθμό άλλων κόμβων που συμμετέχουν στο δίκτυο και ξέρει περισσότερα για κόμβους που έπονται του δικού του αλλά είναι κοντά του, παρά για κόμβους μακριά του.*
2. *Η πληροφορία που φέρει ένας κόμβος, δεν είναι αρκετή ώστε να του επιτρέπει να εντοπίζει το διάδοχο (με άλλα λόγια τον υπεύθυνο κόμβο) ενός οποιοδήποτε κλειδιού βάσει μόνο του δικού του finger table.* [14, 22]

1.2.2.3 Δρομολόγηση στο Chord

Η αναζήτηση του διαδόχου του f ξεκινώντας από τον κόμβο r αρχίζει ελέγχοντας εάν το f είναι ανάμεσα στον r και στον άμεσο διάδοχο του r. Εάν ναι, η αναζήτηση τερματίζεται και επιστρέφεται ο διάδοχος του r. Σε αντίθετη περίπτωση, ο r προωθεί την αίτηση αναζήτησης στο μεγαλύτερο κόμβο του πίνακα finger που προηγείται του f (έστω κόμβος s). Η ίδια διαδικασία ακολουθείται από τον s μέχρι να τερματιστεί η αναζήτηση. Επί παραδείγματι, υποθέτουμε ότι το σύστημα είναι σε σταθερή κατάσταση (όλοι οι πίνακες δρομολόγησης περιέχουν σωστές πληροφορίες) και έχει ξεκινήσει από τον κόμβο 2 (βλ. παρακάτω εικόνα) αναζήτηση για το διάδοχο του αναγνωριστικού 6. Ο μεγαλύτερος κόμβος με αναγνωριστικό μικρότερο του 6 είναι ο 5. Ο στόχος της αναζήτησης (6) είναι στο διάστημα ανάμεσα στον 5 και στο διάδοχο του 5 (7). Άρα, 7 είναι η επιστρεφόμενη τιμή. Από τα παραπάνω, λοιπόν, βλέπουμε πως κατά τη διάρκεια μιας αναζήτησης, κάθε κόμβος αρχικά ελέγχει αν ο διάδοχος του κλειδιού προς αναζήτηση περιέχεται στον πίνακα διαδόχων του κι αν όχι, τότε το ερώτημα προωθείται σε κόμβο του πίνακα finger ο οποίος είναι πλησιέστερα στο αναζητούμενο κλειδί στο χώρο των αναγνωριστικών.



Εικόνα 5 : Παράδειγμα εύρεσης διαδόχου

Όπως σκιαγραφείται και παραπάνω, ο αλγόριθμος έχει αναδρομική (recursive) μορφή : εάν μία αίτηση αναζήτησης χρειάζεται πολλαπλά βήματα για την ολοκλήρωση της, το n-ιοστό βήμα ξεκινά από το (n-1)-ιοστό κόμβο εκ μέρους του αρχικού κόμβου. Η συνάρτηση αναζήτησης μπορεί να υλοποιηθεί και επαναληπτικά

(iterative). Στην επαναληπτική υλοποίηση, ο αρχικός κόμβος είναι αυτός που κάνει αιτήσεις για πληροφορίες του finger table σε κάθε στάδιο του πρωτοκόλλου. Και οι δύο υλοποιήσεις έχουν πλεονεκτήματα : η επαναληπτική προσέγγιση είναι ευκολότερη στην υλοποίηση και βασίζεται λιγότερο σε ενδιάμεσους κόμβους, ενώ η αναδρομική προσέγγιση ενδείκνυται περισσότερο για επιπλέον λειτουργίες, όπως είναι η προσωρινή αποθήκευση και η επιλογή εξυπηρετητή. [14, 22]

1.2.2.4 Εισαγωγή ενός κόμβου

Σε ένα δυναμικό δίκτυο, κόμβοι εισέρχονται και φεύγουν συνεχώς. Η πραγματική πρόκληση είναι η υλοποίηση των παραπάνω διαδικασιών με τρόπο που να επιτρέπει τον ανά πάσα στιγμή εντοπισμό οποιοδήποτε κλειδιού στο δίκτυο. Αυτό συνεπάγεται ότι η ορθότητα των αναζητήσεων δε θα πρέπει να επηρεάζεται από την ταυτόχρονη αποτυχία κόμβων ή τη συνεχή διαδικασία εισαγωγής κι αποχώρησης κόμβων. Για το σκοπό αυτό, το Chord φροντίζει να διατηρεί πάντα δύο αναλλοίωτες συνθήκες :

1. *Κάθε κόμβος ξέρει πάντα τον διάδοχό του.*
2. *Για κάθε κλειδί, ο κόμβος που είναι υπεύθυνος για αυτό είναι ο κόμβος του οποίου το αναγνωριστικό είναι το μικρότερο που έπεται του δικού του στον κύκλο των αναγνωριστικών.*

Πέρα από τα παραπάνω που εξασφαλίζουν την ορθότητα, για λόγους απόδοσης είναι επιθυμητό και οι εγγραφές στο finger table να είναι πάντα ενημερωμένες.

Για την απλοποίηση των διαδικασιών ένταξης και αποχώρησης κόμβων από το δίκτυο αλλά και την υλοποίηση των μηχανισμών σταθεροποίησης του δικτύου μετά από ταυτόχρονες εισαγωγές κι αποχωρήσεις κόμβων, κάθε κόμβος διατηρεί επίσης έναν δείκτη προς τον κόμβο που προηγείται του δικού του στο χώρο των αναγνωριστικών. Ο κόμβος αυτός καλείται *προηγούμενος (predecessor).* Ο δείκτης αυτός επιτρέπει την διάσχιση του δακτυλιδιού των κόμβων του Chord αντιωρολογιακά κατά τη διαδικασία αναζήτησης.

Για τη διατήρηση των παραπάνω αναλλοίωτων συνθηκών, το Chord πρέπει να προβεί στις παρακάτω τρεις ενέργειες όταν ένας νέος κόμβος εντάσσεται στο δίκτυο :

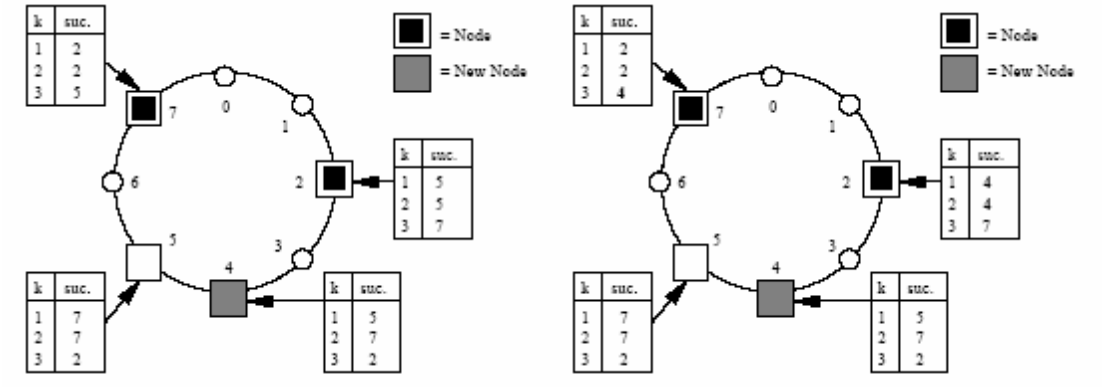
1. *Την αρχικοποίηση του finger table και του προηγούμενου κόμβου του νέου κόμβου.*
2. *Ανανέωση των finger tables των ήδη υπαρχόντων κόμβων στο δίκτυο ώστε να συμπεριλάβουν τον νέο κόμβο.*
3. *Ενημέρωση του λογισμικού πάνω από το Chord ώστε να γίνουν οι απαραίτητες αλλαγές π.χ. σε μια εφαρμογή αποθήκευσης αρχείων η μεταφορά των κλειδιών που πλέον θα σχετίζονται με τον νέο κόμβο.*

Για την εύρεση του διαδόχου του, ένας νέος κόμβος ζητάει από έναν υπάρχοντα κόμβο, τον οποίο μαθαίνει από έναν εξωτερικό μηχανισμό, να του τον βρει μέσα από μια διαδικασία αναζήτησης του αναγνωριστικού του νέου κόμβου.

Στη συνέχεια, και για λόγους σταθεροποίησης του δικτύου σε περιπτώσεις ταυτόχρονων εισαγωγών κόμβων, αναλαμβάνει το πρωτόκολλο σταθεροποίησης, το οποίο ο κάθε κόμβος εκτελεί περιοδικά. Το πρωτόκολλο αυτό παρουσιάζουμε στη συνέχεια με ένα παράδειγμα για λόγους πιο εύκολης κατανόησης.

Υποθέτουμε πως ένας κόμβος n εισέρχεται στο δίκτυο και το αναγνωριστικό του είναι μεταξύ του κόμβου p και του s. Ο κόμβος n θα πάρει τον s σαν διάδοχο. Στη συνέχεια, όταν ο s ειδοποιηθεί από τον n, θα βάλει τον n στη θέση του p ως προηγούμενο κόμβο. Όταν ο p τρέξει το πρωτόκολλο σταθεροποίησης, θα ρωτήσει τον s για τον προηγούμενό του κόμβο (που τώρα είναι ο n) κι έτσι θα βάλει τον n ως διάδοχό του. Τέλος, ο p θα ειδοποιήσει τον n και ο τελευταίος θα βάλει τον p σαν προηγούμενό του κόμβο.

Στη συνέχεια, για την ανανέωση του finger table του, κάθε κόμβος τρέχει περιοδικά μια διαδικασία βάσει της οποίας επιλέγει τυχαία μια θέση του finger table κι εκτελεί μια αναζήτηση για την αρχή της ζώνης που η θέση αυτή καλύπτει. Ο κόμβος που επιστρέφεται είναι και αυτός που καταλαμβάνει την εν λόγο θέση στο finger table. Το Chord δεν παρέχει κάποιον πιο "δυνατό" και άμεσο μηχανισμό ανανέωσης του finger table ενός κόμβου, κι αυτό γιατί αποδεικνύεται πως αρκεί απλά η ύπαρξη κόμβων στο finger table για την αποδοτική δρομολόγηση. Ένα μη ενημερωμένο finger table λόγω εισαγωγής νέων κόμβων, δεν επηρεάζει ιδιαίτερα τη διαδικασία αναζήτησης, εκτός βέβαια και αν εισαχθεί ταυτόχρονα και στην ίδια περιοχή του δικτύου ένας τεράστιος αριθμός νέων κόμβων. Οι νέοι κόμβοι που εντάσσονται στο δίκτυο, μπαίνουν ανάμεσα σε ήδη υπάρχοντες κόμβους. Έτσι, σε μια διαδικασία αναζήτησης, ακόμα κι αν χρειαστεί να τους διατρέξουμε γραμμικά, τελικά πάλι O(logN) βήματα θα χρειαστούν για την εύρεση του κλειδιού. [14, 22]

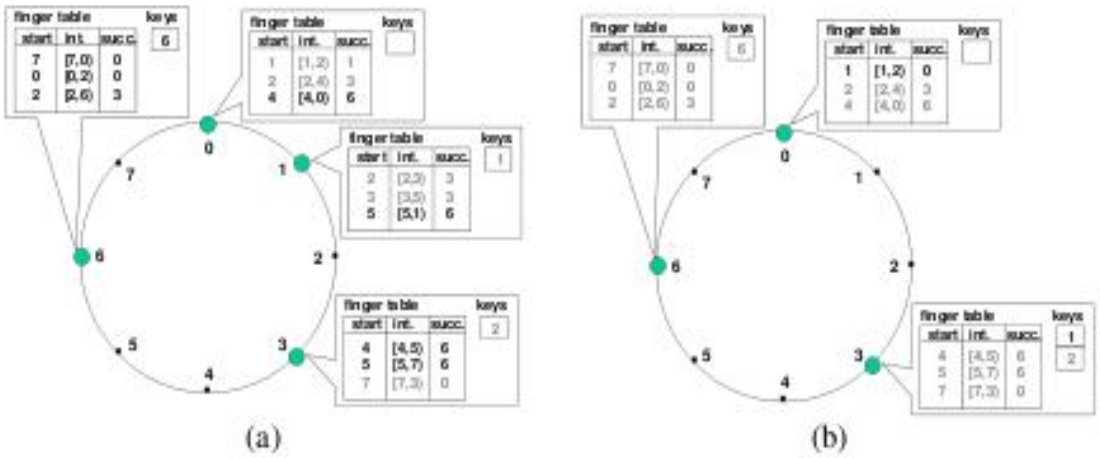


Εικόνα 6 : Αρχικοποίηση κι ανανέωση πινάκων δρομολόγησης.

Τέλος, ο νέος κόμβος πρέπει επίσης να αποκτήσει και όλες τις τιμές για τις οποίες θα είναι υπεύθυνος. Η δομή του Chord εξασφαλίζει ότι όλες αυτές οι τιμές θα προέρχονται από τον διάδοχο του νέου κόμβου.

1.2.2.5 Αποτυχία κόμβων και διαχείριση αντιγράφων

Όταν ένας κόμβος αποσυνδέεται, οι κόμβοι που τον είχαν στο finger table τους πρέπει να βρουν τον διάδοχό του. Επιπλέον, η αποτυχία αυτή δεν πρέπει να διακόψει ερωτήματα που βρίσκονται σε εξέλιξη όσο το δίκτυο του Chord ανακάμπτει από την αποτυχία.



Εικόνα 7 : σχήμα α) φαίνονται τα finger tables και τα κλειδιά μετά την εισαγωγή του κόμβου 6 ενώ στο β) φαίνονται τα finger tables και τα κλειδιά μετά την αποτυχία του κόμβου 3. Οι εγγραφές που άλλαξαν είναι με μαύρο ενώ αυτές που δεν επηρεάστηκαν είναι με γκρι.

Το κλειδί για τη μη αποτυχία των ερωτημάτων σε περίπτωση αποτυχίας ενός κόμβου είναι η ορθή διατήρηση του διαδόχου κάθε κόμβου. Κι αυτό γιατί στη χειρότερη περίπτωση η αναζήτηση μπορεί να συνεχιστεί έστω και μόνο μέσω των διαδόχων κάθε κόμβου. Για το λόγο αυτό, κάθε κόμβος, πέρα από το finger table, διατηρεί και μια λίστα με τους άμεσους διαδόχους του η οποία ονομάζεται "λίστα διαδόχων" (successor list). Η λίστα αυτή διατηρείται από μία τροποποιημένη έκδοση του μηχανισμού σταθεροποίησης. Μέσω αυτής, αν ο διάδοχος ενός κόμβου αποτύχει, τότε αντικαθίσταται από το διάδοχό του στην "λίστα διαδόχων". Από αυτό το σημείο κι έπειτα, κάθε αναζήτηση για κλειδί για το οποίο ο κόμβος που αποσυνδέθηκε ήταν υπεύθυνος, δρομολογείται στον διάδοχό του, που είναι και ο νέος υπεύθυνος για το κλειδί αυτό. Στη συνέχεια, ο μηχανισμός σταθεροποίησης αναλαμβάνει την ανανέωση των finger table κόμβων που περιείχαν τον κόμβο που αποσυνδέθηκε.

Μετά την αποχώρηση ενός κόμβου και πριν την ολοκλήρωση της σταθεροποίησης του δικτύου, κόμβοι θα προσπαθούν να δρομολογήσουν αναζητήσεις μέσω του κόμβου που έπεσε. Στις περισσότερες των περιπτώσεων, η δρομολόγηση θα μπορέσει να συνεχιστεί μέσω εναλλακτικών κόμβων μετά την εκπνοή κάποιου χρονικού διαστήματος. Οι κόμβοι αυτοί μπορεί να προέρχονται είτε από το finger table είτε από τη "λίστα διαδόχων" του κόμβου.

Η συντήρηση της λίστας διαδόχων από κάθε κόμβο βοηθά τις εφαρμογές που δομούνται πάνω στο

Chord στο να κρατούν αντίγραφα των δεδομένων που κάθε κόμβος αποθηκεύει. Μία τέτοια εφαρμογή μπορεί να αντιγράφει τα δεδομένα κάθε κόμβου στους κόμβους της λίστας διαδόχων του. Επιπλέον, η ύπαρξη της λίστας θα μπορεί να ενημερώνει την εφαρμογή για την αποτυχία ενός κόμβου και την ανάγκη για περαιτέρω αντιγραφή των δεδομένων του. [14, 22]

1.2.2.6 Το API του Chord

Το API του Chord παρέχει τη βασική λειτουργικότητα για τον εντοπισμό κλειδιών σε έναν διαστρωματωμένο σχεδιασμό. Δύο αρχές σχεδιασμού διευκολύνουν τη χρήση του Chord σε μία διαστρωματωμένη αρχιτεκτονική : ελάχιστη λειτουργικότητα και ελάχιστη εκτεθειμένη πληροφορία. Ελαχιστοποιώντας την έκταση της λειτουργικότητας που ενσωματώνεται στο Chord (κατώτερο επίπεδο), ελαχιστοποιούνται οι περιορισμοί που τίθενται σε υψηλότερα επίπεδα, τα οποία εξαρτώνται από το Chord. Η χρησιμότητα του Chord API είναι περισσότερο εμφανής κατά το σχεδιασμό των στρωμάτων που τοποθετούνται πάνω στο βασικό στρώμα Chord και τα οποία είναι απαραίτητα για τη δόμηση μεγάλων peer-to-peer εφαρμογών ανταλλαγής αρχείων.

Ωστόσο τα μεγάλα συστήματα που βασίζονται στο Chord υπόκεινται σε αρκετούς περιορισμούς. Το παραπάνω δε μπορεί να αποδοθεί στο ότι το Chord δεν παρέχει ένα αρκετά ελαστικό σύνολο χαρακτηριστικών, αλλά στο ότι τα υψηλότερα στρώματα επιθυμούν πρόσβαση στην εσωτερική κατάσταση του Chord κατά τη διάρκεια του υπολογισμού της. Για να παρέχεται αυτή η πρόσβαση με ταυτόχρονη διατήρηση των αφαιρετικών ορίων, επιτρέπεται σε όλα τα στρώματα να εγγράφονται σε συναρτήσεις "callback" για γεγονότα, για τα οποία ενδιαφέρονται και να αποτιμούν τη συνάρτηση διαδόχου σε κάθε βήμα ξεχωριστά.

Χαρακτηριστικά αναφέρουμε ότι η next_hop(j, k) στέλνει ένα μήνυμα στον κόμβο j ρωτώντας τον για τη μικρότερη είσοδο στον finger πίνακά του που είναι μεγαλύτερη του k. Αυτό επιτρέπει στους κόμβους που καλούν τη συνάρτηση να ελέγχουν την εκτέλεση του αλγόριθμου αναζήτησης του Chord βήμα προς βήμα. [10, 14, 22]

1.2.3 Kademlia

1.2.3.1 Γενικά

Το Kademlia είναι ένα peer-to-peer σύστημα αποθήκευσης και αναζήτησης ζευγών (κλειδί, τιμή). Βασικό χαρακτηριστικό του είναι ότι ελαχιστοποιεί τον αριθμό των μηνυμάτων ρύθμισης που πρέπει να στείλουν οι κόμβοι, για να μάθουν όσα χρειάζεται να ξέρουν για τους άλλους κόμβους. Οι πληροφορίες ρύθμισης εξαπλώνονται αυτόματα, καθώς διεξάγονται οι αναζητήσεις κλειδιών. Επιπλέον, οι κόμβοι έχουν την απαραίτητη γνώση και ευελιξία, ώστε να δρομολογούν ερωτήσεις μέσω διαδρομών χαμηλής καθυστέρησης. Το Kademlia χρησιμοποιεί παράλληλα ασύγχρονα ερωτήματα, για να αποφύγει καθυστερήσεις λήξης χρόνου από νεκρούς κόμβους και ο αλγόριθμος με τον οποίο ενημερώνονται οι κόμβοι για την κατάσταση των άλλων κόμβων στο σύστημα είναι ανθεκτικός απέναντι στη βασική επίθεση DoS.

Τα κλειδιά στο Kademlia είναι λέξεις των 160 bits (π.χ. ο SHA-1 κατακερματισμός μεγαλύτερων δεδομένων). Κάθε ένας από τους συμμετέχοντες στο σύστημα υπολογιστές έχει ένα node ID σε έναν χώρο κλειδιών των 160 bits. Τα ζεύγη (κλειδί, τιμή) αποθηκεύονται σε κόμβους με IDs «κοντά» στο κλειδί. Τέλος, ο αλγόριθμος δρομολόγησης βασίζεται στα node IDs και επιτρέπει τον εντοπισμό των εξυπηρετητών κοντά σε ένα κλειδί-προορισμό από κάθε κόμβο.

Ένα από τα καινοτόμα στοιχεία του Kademlia είναι η XOR τεχνική μέτρησης της απόστασης ανάμεσα σε δύο σημεία στο χώρο των κλειδιών. Η XOR τεχνική είναι συμμετρική και επιτρέπει στους κόμβους του Kademlia να λαμβάνουν ερωτήσεις αναζήτησης από ακριβώς την ίδια κατανομή κόμβων που περιέχεται στους πίνακες δρομολόγησης τους. Ένας κόμβος Kademlia μπορεί να στείλει ερώτηση σε οποιονδήποτε κόμβο εντός ενός διαστήματος και έτσι μπορεί να επιλέξει διαδρομές ανάλογα με την καθυστέρηση ή ακόμα και να στείλει παράλληλα ασύγχρονες ερωτήσεις. Για τον εντοπισμό κόμβων κοντά σε ένα συγκεκριμένο ID το Kademlia χρησιμοποιεί έναν απλό αλγόριθμο δρομολόγησης από την αρχή μέχρι το τέλος (άλλα συστήματα χρησιμοποιούν έναν αλγόριθμο για να πλησιάσουν το ID-στόχο και έναν άλλο για τα τελευταία λίγα βήματα).

1.2.3.2 Περιγραφή συστήματος Kademlia

Κάθε κόμβος Kademlia έχει ένα node ID των 160 bits. Κάθε μήνυμα που μεταδίδεται από έναν κόμβο περιλαμβάνει και το node ID του, επιτρέποντας κατ' αυτόν τον τρόπο στον παραλήπτη να καταγράψει την ύπαρξη του αποστολέα.

Τα κλειδιά είναι και αυτά αναγνωριστικά των 160 bits. Για τη δημοσίευση και την εύρεση ζευγών (κλειδί, τιμή) το Kademlia χρησιμοποιεί μια δική του έννοια για την απόσταση ανάμεσα σε δύο αναγνωριστικά. Δεδομένων δύο αναγνωριστικών των 160 bits, x και y, το Kademlia ορίζει ως απόσταση ανάμεσα στα δύο το ανά bit XOR τους εκφρασμένο σαν ακέραιο, d (x, y) = x XOR y. Σημειώνουμε ότι το XOR είναι ένα έγκυρο και

μη-Ευκλείδειο μέτρο, για το οποίο ισχύει :

- d (x, x) = 0 και d (x, y) > 0, εάν x ≠ y
- □ x, y : d (x, y) = d (y, x)
- d (x, z) = d (x, y) XOR d (y, z)
  - □ a ≥0, b≥0 : a + b ≥ a XOR b =>
    - d (x, y) + d (y, z) ≥d (x, z)

Το XOR είναι μονοκατευθυντικό. Για κάθε δεδομένο σημείο x και απόσταση Δ>0, υπάρχει ένα και μόνο σημείο y, για το οποίο ισχύει d (x, y) = Δ. Η μονοκατευθυντικότητα εξασφαλίζει ότι όλες οι αναζητήσεις για το ίδιο κλειδί θα συγκλίνουν στην ίδια διαδρομή ανεξάρτητα του κόμβου που την ξεκίνησε. Έτσι, η προσωρινή αποθήκευση ζευγών (κλειδί, τιμή) κατά μήκος της διαδρομής αναζήτησης μπορεί να ανακουφίσει τα δημοφιλή ζεύγη. Τέλος, η XOR τοπολογία είναι συμμετρική (□ x, y: d (x, y) = d (y, x)).

1.2.3.3 Πληροφορία ανά κόμβο

Οι κόμβοι Kademlia αποθηκεύουν πληροφορίες επικοινωνίας με του άλλους κόμβους, για να δρομολογούν μηνύματα-ερωτήσεις. Για κάθε $0 \leq i < 160$, κάθε κόμβος διατηρεί μια λίστα από τριάδες της μορφής (IP διεύθυνση, UDP πόρτα, node ID) για κόμβους με απόσταση μεταξύ 2^i και 2^{i+1} από τον εαυτό του. Αυτές οι λίστες καλούνται k-κάδοι (k-buckets). Κάθε k-κάδος κρατείται ταξινομημένος σύμφωνα με το χρόνο τελευταίας συνάντησης. Στην κεφαλή της λίστας βρίσκεται ο κόμβος που «ακούστηκε» νωρίτερα και στην ουρά ο κόμβος που «ακούστηκε» πιο πρόσφατα. Για μικρές τιμές του i, οι k-κάδοι θα είναι γενικά άδειοι, αφού δε θα υπάρχουν οι αντίστοιχοι κόμβοι. Για μεγάλες τιμές του i, οι λίστες μπορούν να φτάσουν μέχρι και μέγεθος k, όπου k είναι η παράμετρος αντιγραφής του συστήματος. Το k επιλέγεται έτσι ώστε οποιοιδήποτε k κόμβοι να είναι όσο το δυνατό πιο απίθανο να αποτύχουν με διαφορά λιγότερο της μίας ώρας ο ένας από τον άλλο.

Όταν ένας κόμβος Kademlia λαμβάνει ένα μήνυμα (αίτηση ή απάντηση) από έναν άλλο κόμβο, ενημερώνει τον αντίστοιχο k-κάδο με το node ID του αποστολέα. Εάν τα στοιχεία του κόμβου αποστολέα υπάρχουν ήδη στον k-κάδο του παραλήπτη, τότε ο παραλήπτης μεταφέρει την τριάδα του αποστολέα στην ουρά της λίστας. Εάν δεν υπάρχουν και ο k-κάδος έχει λιγότερες από k στοιχεία, τότε ο παραλήπτης απλά εισάγει τον αποστολέα στην ουρά της λίστας. Αν πάλι ο k-κάδος είναι γεμάτος, τότε ο παραλήπτης κάνει ping στον τελευταίο κόμβο από τον οποίο «άκουσε», για να αποφασίσει τι θα κάνει. Εάν αυτός ο κόμβος δεν απαντήσει, διαγράφεται από τον κάδο και εισάγεται στην ουρά της λίστας ο νέος αποστολέας. Διαφορετικά, μεταφέρεται στην ουρά της λίστας και ο νέος αποστολέας απορρίπτεται από τον κάδο.

Οι k-κάδοι υλοποιούν αποδοτικά μια τεχνική απόρριψης του κόμβου που συναντήθηκε λιγότερο πρόσφατα χωρίς να απομακρύνουν ζωντανούς κόμβους. Αυτή η προτίμηση για παλιές επαφές πηγάζει από μία ανάλυση, κατά την οποία, όσο περισσότερο είναι ζωντανός ένας κόμβος, τόσο πιο πιθανό είναι να παραμείνει

ζωντανός για άλλη μια ώρα. Έτσι, κρατώντας τις παλιότερες επαφές στους κάδους οι k-κάδοι μεγιστοποιούν την πιθανότητα να περιέχουν κόμβους που θα παραμένουν συνδεδεμένοι.

Ένα δεύτερο πλεονέκτημα των k-κάδων είναι ότι παρέχουν ασφάλεια απέναντι σε συγκεκριμένες επιθέσεις DoS. Κανείς δεν μπορεί να εξαφανίσει την πληροφορία δρομολόγησης που φυλάει ένας κόμβος πλημμυρίζοντας το σύστημα με νέους κόμβους, γιατί οι νέοι κακόβουλοι κόμβοι θα εισαχθούν στους k-κάδους μόνο όταν αποχωρήσουν από το σύστημα οι παλιοί κόμβοι.

1.2.3.4 Πρωτόκολλο Kademlia

Το πρωτόκολλο Kademlia περιλαμβάνει τέσσερις απομακρυσμένες κλήσεις διαδικασίας (Remote Procedure Calls – RPCs): PING, STORE, FIND_NODE και FIND_VALUE. Η PING RPC επιχειρεί να μάθει εάν ένας κόμβος είναι συνδεδεμένος. Η STORE δίνει την εντολή σε έναν κόμβο να αποθηκεύσει ένα ζεύγος (κλειδί, τιμή), για να ανακτηθεί αργότερα.

Η FIND_NODE παίρνει ένα ID των 160 bits σαν όρισμα. Ο παραλήπτης της RPC επιστρέφει μια τριάδα (διεύθυνση IP, UDP πόρτα, node ID) για καθένα από τους k κόμβους που γνωρίζει ότι είναι κοντινότερα στον στόχο-ID. Αυτές οι τριάδες μπορεί να προέρχονται από έναν και μόνο κάδο, ή και από πολλούς κάδους, ανάλογα με το αν ο κοντινότερος k-κάδος είναι ή όχι γεμάτος. Σε κάθε περίπτωση, ο παραλήπτης RPC πρέπει να στείλει k τεμάχια, εκτός εάν υπάρχουν λιγότεροι από k κόμβοι σε όλους συνολικά τους κάδους του, οπότε και στέλνει πληροφορία μόνο για τους κόμβους που γνωρίζει.

Η FIND_VALUE συμπεριφέρεται σαν την FIND_NODE –επιστρέφει και αυτή τριάδες- με μία εξαίρεση. Εάν ο παραλήπτης RPC έχει λάβει κλήση STORE για το κλειδί, απλά επιστρέφει την αποθηκευμένη τιμή.

Σε όλες τις RPCs, ο παραλήπτης πρέπει να συμπεριλάβει ένα τυχαίο RPC ID των 160 bits, το οποίο παρέχει κάποια αντίσταση σε απόπειρες πλαστογραφίας. Τα PINGs μπορούν επίσης να γίνουν piggy-backed σε απαντήσεις RPC, έτσι ώστε ο παραλήπτης RPC να λάβει μια επιπλέον επιβεβαίωση της διεύθυνσης δικτύου του αποστολέα.

Η πιο σημαντική εργασία για έναν κόμβο Kademlia είναι ο εντοπισμός των k κοντινότερων κόμβων σε ένα δεδομένο ID. Αυτή η εργασία καλείται αναζήτηση κόμβου. Το Kademlia εφαρμόζει έναν αναδρομικό αλγόριθμο για τις αναζητήσεις κόμβων. Ο κόμβος – εκκινητής της αναζήτησης – (τον ονομάζουμε αρχικό) διαλέγει α κόμβους από τον κοντινότερο μη-άδειο k-κάδο του. Εάν ο κάδος έχει λιγότερα από α στοιχεία, παίρνει απλά του κοντινότερους α κόμβους που γνωρίζει. Ο αρχικός στέλνει στη συνέχεια παράλληλα και ασύγχρονα FIND_NODE RPCs στους α κόμβους που έχει επιλέξει. Το α είναι μια παράμετρος ταυτοχρονισμού του συστήματος.

1.2.3.5 Αλγόριθμος αναζήτησης,προσωρινής αποθήκευσης και εισαγωγή κόμβου

Στο αναδρομικό βήμα ο αρχικός ξαναστέλνει FIND_NODE στους κόμβους που γνωρίζει από προηγούμενα RPCs. Αυτή η αναδρομή μπορεί μάλιστα να ξεκινήσει πριν επιστρέψουν και τα α προηγούμενα RPCs. Από τους k κόμβους που έχει ακούσει ο αρχικός να είναι πλησιέστερα στο στόχο, διαλέγει α που δεν έχει ακόμα ρωτήσει και ξαναστέλνει FIND_NODE RPC σε αυτούς. Οι κόμβοι που δεν απαντούν γρήγορα αγνοούνται μέχρι και εάν απαντήσουν. Εάν ένας κύκλος από FIND_NODEs αποτύχει να επιστρέψει έναν κόμβο κοντύτερα από αυτόν που θεωρείται μέχρι τώρα κοντινότερος, ο αρχικός ξαναστέλνει FIND_NODE σε όλους τους k κοντινότερους κόμβους που δεν έχει ακόμα ρωτήσει. Η αναζήτηση τελειώνει, όταν ο αρχικός έχει ρωτήσει και λάβει απαντήσεις από όλους τους k κοντινότερους κόμβους που έχει «δει». Το Kademlia μπορεί να δρομολογήσει με χαμηλότερη καθυστέρηση, γιατί έχει την ευχέρεια της επιλογής ανάμεσα σε αυτούς τους k κόμβους, για να προωθήσει ένα μήνυμα.

Οι περισσότερες λειτουργίες γίνονται σε σχέση με τον παραπάνω αλγόριθμο αναζήτησης. Για να αποθηκεύσει ένας κόμβος ένα ζεύγος (κλειδί, τιμή), εντοπίζει τους k κοντινότερους κόμβους στο κλειδί και στέλνει STORE RPCs σε αυτούς. Επιπλέον, κάθε κόμβος δημοσιεύει τα ζεύγη που φυλάει κάθε ώρα. Αυτό εξασφαλίζει στο σύστημα συνέπεια με μεγάλη πιθανότητα. Γενικά, απαιτείται και οι κόμβοι που δημοσίευσαν πρώτοι ένα ζεύγος να το δημοσιεύουν ξανά κάθε 24 ώρες. Διαφορετικά, όλα τα ζεύγη (κλειδί, τιμή) «λήγουν» ύστερα από 24 ώρες από την αρχική τους δημοσίευση προκειμένου να περιοριστεί η παλιά πληροφορία στο σύστημα.

Τέλος, για να υπάρχει συνέπεια στον κύκλο ζωής (δημοσίευση-αναζήτηση) ενός ζεύγους (κλειδί, τιμή), απαιτείται οποτεδήποτε ένα κόμβος x παρατηρεί έναν νέο κόμβο y, ο οποίος είναι κοντύτερα σε κάποια από τα ζεύγη που έχει ο x, να αντιγράφονται αυτά τα ζεύγη του x στον y χωρίς να αφαιρούνται από τη βάση δεδομένων του.

Για να βρει ένας κόμβος ένα ζεύγος (κλειδί, τιμή), ξεκινά με αναζήτηση των k κόμβων με τα πλησιέστερα IDs στο κλειδί. Ωστόσο, οι αναζητήσεις τιμής χρησιμοποιούν FIND_VALUE αντί για FIND_NODE RPCs. Επιπλέον, η διαδικασία διακόπτεται, μόλις κάποιος κόμβος επιστρέψει την τιμή. Για λόγους προσωρινής αποθήκευσης (caching), όταν πετύχει μια αναζήτηση, ο ζητών κόμβος αποθηκεύει το ζεύγος (κλειδί, τιμή) στον κοντινότερο στο κλειδί κόμβο που «είδε» και που δεν επέστρεψε την τιμή.

Λόγω της μονοκατευθυντικότητας της τοπολογίας, μελλοντικές αναζητήσεις για το ίδιο κλειδί αναμένεται να πετύχουν προσωρινά αποθηκευμένες εγγραφές πριν ρωτήσουν τον κοντινότερο κόμβο. Εάν ένα κλειδί γίνει πολύ δημοφιλές, το σύστημα θα καταλήξει να το αποθηκεύει σε πολλούς κόμβους. Για να αποφευχθεί το "over- caching", ο χρόνος «λήξης» ενός ζεύγους στη βάση δεδομένων ενός κόμβου είναι αντίστροφα εκθετικά ανάλογος του αριθμού των κόμβων ανάμεσα στον κόμβο και στον κόμβο με ID πλησιέστερα στο κλειδί. Ενώ μια απλή διαγραφή LRU (Least Recently Used) θα έδινε παρόμοια κατανομή στους χρόνους ζωής, δεν υπάρχει τρόπος να επιλεχτεί το μέγεθος της μνήμης προσωρινής αποθήκευσης (cache),

εφόσον οι κόμβοι δε γνωρίζουν εκ των προτέρων πόσες τιμές θα αποθηκεύσει το σύστημα.

Οι κάδοι θα διατηρούνται γενικά «φρέσκοι», καθώς θα ανανεώνουν τα δεδομένα τους από την κίνηση των αιτήσεων στο σύστημα. Για την αποφυγή παθολογικών καταστάσεων, όταν δεν υπάρχει κίνηση, κάθε κόμβος ανανεώνει έναν κάδο του, όταν δεν έχει κάνει κανένας κόμβος αναζήτηση στην περιοχή του για μια ώρα. Η ανανέωση αυτή έγκειται στην επιλογή ενός τυχαίου ID στην περιοχή του κάδου και στην εκκίνηση μιας αναζήτησης για αυτό το ID.

Για να προσχωρήσει στο δίκτυο ένας κόμβος x, πρέπει να έχει επαφή με έναν τουλάχιστον κόμβο του δικτύου, έστω y. Ο x εισάγει τον y στον κατάλληλο k-κάδο του και μετά ξεκινά μια αναζήτηση κόμβου για το δικό του ID. Έτσι, γεμίζει όλους τους k-κάδους του και γνωστοποιεί την παρουσία του στο υπόλοιπο δίκτυο.

1.2.3.6 Συμπερασματικά

Λόγω της πρωτότυπης τοπολογίας που βασίζεται στο XOR μέτρο απόστασης, το Kademlia είναι ένα σύστημα peer-to-peer που συνδυάζει συνέπεια, απόδοση, δρομολόγηση με ελαχιστοποιημένη καθυστέρηση και συμμετρική, μονοκατευθυντική τοπολογία. Επιπλέον, εισάγει μία παράμετρο ταυτοχρονισμού α, που επιτρέπει την επιλογή ανάμεσα σε εύρος ζώνης για την ασύγχρονη επιλογή του βήματος με την ελάχιστη καθυστέρηση και σε ανάκαμψη από αποτυχίες ανεξαρτήτως χρόνου.Τέλος, το Kademlia είναι το πρώτο σύστημα peer-to-peer το οποίο εκμεταλλεύεται το γεγονός ότι οι αποτυχίες κόμβων είναι αντίστροφα σχετιζόμενες με το χρονικό διάστημα που οι ίδιοι κόμβοι είναι ζωντανοί.

1.2.4 Pastry

1.2.4.1 Γενικά

Ένα από τα μεγαλύτερα προβλήματα στις peer-to-peer εφαρμογές μεγάλης κλίμακας είναι η εύρεση αποδοτικών αλγορίθμων για την αναζήτηση αντικειμένων στο δίκτυο, μια διαδικασία που ονομάζεται και δρομολόγηση. Το Pastry είναι ένα γενικό σχήμα peer-to-peer εντοπισμού αντικειμένων και δρομολόγησης βασισμένο σε ένα αυτο-διοργανούμενο υπερκείμενο δίκτυο κόμβων που συνδέονται μέσω του Διαδικτύου. Είναι πλήρως αποκεντρωμένο, ανθεκτικό σε λάθη, κλιμακούμενο σε μέγεθος και αξιόπιστο.

Το Pastry προορίζεται για γενική υποδομή για τη δημιουργία πολλών και διαφορετικών peer-to-peer εφαρμογών, όπως η εκτεταμένη ανταλλαγή και αποθήκευση αρχείων, η επικοινωνία ομάδων και τα συστήματα ονοματοδοσίας. Πολλές εφαρμογές έχουν δομηθεί πάνω στο Pastry μέχρι στιγμής, όπως το PAST [21,41], μια υπηρεσία συνεχούς αποθήκευσης και το SCRIBE [16], ένα κλιμακούμενο σύστημα δημοσίευσης/ εγγραφής.

Η λειτουργικότητα που παρέχει στις εφαρμογές που δομούνται πάνω σε αυτό είναι η ακόλουθη : κάθε κόμβος στο δίκτυο Pastry έχει ένα μοναδικό αναγνωριστικό (nodeId). Με ένα μήνυμα και ένα κλειδί ένας κόμβος που συμμετέχει σε ένα δίκτυο Pastry δρομολογεί αποδοτικά το μήνυμα στον κόμβο με nodeId που είναι αριθμητικά πλησιέστερο στο κλειδί σε σχέση με όλους τους άλλους "ζωντανούς" κόμβους Pastry. Ο αναμενόμενος αριθμός βημάτων δρομολόγησης είναι O(logN), όπου N το πλήθος των κόμβων Pastry στο δίκτυο. Από κάθε κόμβο Pastry που περνάει ένα μήνυμα κατά την προώθησή του προς τον προορισμό, ενημερώνεται η αντίστοιχη εφαρμογή και της δίνεται η δυνατότητα να τροποποιήσει το μήνυμα ανάλογα με τις ανάγκες της.

Το Pastry λαμβάνει υπόψη του την έννοια της τοπικότητας των κόμβων που συμμετέχουν στο δίκτυο : προσπαθεί να ελαχιστοποιήσει τις αποστάσεις που ταξιδεύουν τα μηνύματα σύμφωνα με ένα μέτρο εγγύτητας των κόμβων, όπως ο αριθμός των IP βημάτων δρομολόγησης. Κάθε κόμβος Pastry είναι ενήμερος για τους άμεσους γείτονές του στο χώρο των nodeIds και πληροφορεί τις εφαρμογές για αφίξεις νέων κόμβων, αποτυχίες και ανακάμψεις κόμβων. Επειδή τα nodeIds ανατίθενται τυχαία, είναι πολύ πιθανό μια ομάδα κόμβων με γειτονικά αναγνωριστικά να βρίσκονται αρκετά μακριά γεωγραφικά. Μια ευρηματική τεχνική διασφαλίζει ότι ανάμεσα στο σύνολο των k κόμβων με τα κοντινότερα nodeIds στο κλειδί το μήνυμα μάλλον θα φτάσει πρώτο στον κόμβο που είναι «κοντινότερα» (όσον αφορά το μέτρο εγγύτητας) στον κόμβο από τον οποίο προέρχεται το μήνυμα[2].

1.2.4.2 Σχεδιασμός του Pastry

Το σύστημα Pastry είναι ένα αυτο-διοργανούμενο υπερκείμενο δίκτυο κόμβων όπου κάθε κόμβος δρομολογεί αιτήσεις πελατών και αλληλεπιδρά με τοπικά στιγμιότυπα μίας ή περισσοτέρων εφαρμογών.

Οποιοσδήποτε υπολογιστής είναι συνδεδεμένος στο Διαδίκτυο και τρέχει λογισμικό κόμβου Pastry μπορεί να δράσει σαν κόμβος Pastry και υπόκειται μόνο στις πολιτικές ασφαλείας της εφαρμογής.

Σε κάθε κόμβο στο peer-to-peer υπερκείμενο δίκτυο Pastry ανατίθεται τυχαία (όταν ο κόμβος εντάσσεται στο σύστημα) ένα αναγνωριστικό των 128 bits (nodeId). Το nodeId χρησιμοποιείται για να προσδιορίσει τη θέση του κόμβου σε έναν κυκλικό χώρο των nodeIds που κυμαίνεται από το 0 έως το 2^{128}-1. Τα nodeIds παράγονται έτσι ώστε το προκύπτον σύνολο των nodeIds να είναι ομοιόμορφα κατανεμημένο στο χώρο των nodeIds. Επί παραδείγματι, τα nodeIds θα μπορούσαν να παράγονται υπολογίζοντας τον κρυπτογραφημένο κατακερματισμό του δημοσίου κλειδιού ενός κόμβου ή της IP διεύθυνσής του. Σαν αποτέλεσμα της τυχαίας ανάθεσης τους, με πολύ μεγάλη πιθανότητα, οι κόμβοι με διπλανά nodeIds διαφοροποιούνται σε γεωγραφική θέση, σε ιδιοκτήτη και σε δικαιοδοσία. Αυτή η ιδιότητα εξασφαλίζει, όπως θα δούμε και παρακάτω, παίζει πολύ σημαντικό ρόλο στη αντοχή του δικτύου σε ταυτόχρονες αποτυχίες κόμβων.

Εάν έχουμε ένα δίκτυο με N κόμβους, το Pastry μπορεί να δρομολογήσει στον αριθμητικά πλησιέστερο κόμβο σε δεδομένο κλειδί σε λιγότερα από $\log_2{}^b N$ βήματα υπό κανονικές συνθήκες (b είναι μια παράμετρος ρύθμισης με τυπική τιμή 4). Παρ'όλες τις ταυτόχρονες αποτυχίες κόμβων η τελική παράδοση είναι εγγυημένη, εάν δεν «πέσουν» ταυτόχρονα |L|/2 κόμβοι με διπλανά nodeIds (|L| είναι μια παράμετρος ρύθμισης με τυπική τιμή 16 ή 32).

Για να εξυπηρετηθεί η δρομολόγηση, τα nodeIds και τα κλειδιά θεωρούνται ακολουθίες ψηφίων με βάση 2^b. Το Pastry δρομολογεί μηνύματα στον κόμβο του οποίου το nodeId είναι αριθμητικά κοντινότερα στο δεδομένο κλειδί. Αυτό επιτυγχάνεται ως εξής: σε κάθε βήμα δρομολόγησης, ένας κόμβος προωθεί το εισερχόμενο μήνυμα σε έναν κόμβο του οποίου το nodeId μοιράζεται με το κλειδί ένα πρόθεμα τουλάχιστον κατά ένα ψηφίο (ή b bits) μεγαλύτερο από το πρόθεμα που μοιράζεται το κλειδί με τον τοπικό κόμβο. Εάν δεν υπάρχει τέτοιος κόμβος, το μήνυμα προωθείται στον κόμβο του οποίου το nodeId μοιράζεται ένα πρόθεμα με το κλειδί τόσο μακρύ όσο και ο τοπικός κόμβος, αλλά είναι αριθμητικά κοντύτερα στο κλειδί από ότι το nodeId του τοπικού κόμβου. Για να υποστηριχθεί αυτή η διαδικασία δρομολόγησης, κάθε κόμβος διατηρεί κάποια πληροφορία δρομολόγησης.[2]

1.2.4.3 Πληροφορία ανά κόμβο στο Pastry

Κάθε κόμβος Pastry διατηρεί έναν πίνακα δρομολόγησης (*routing table*), ένα σύνολο γειτονιάς (*neighborhood set*) και ένα σύνολο φύλλων (*leaf set*).

Ο πίνακας δρομολόγησης R ενός κόμβου οργανώνεται σε $\log_2{}^b N$ γραμμές με 2^b–1 εγγραφές η κάθε μία. Κάθε μία από τις 2^b–1 εγγραφές στη γραμμή n του πίνακα δρομολόγησης αναφέρεται στον κόμβο με nodeId που μοιράζεται τα πρώτα n ψηφία με το nodeId του τοπικού κόμβου, αλλά το n+1–ιοστό ψηφίο του έχει μία από τις 2^b–1 πιθανές τιμές (εκτός από το n+1 – ιοστό ψηφίο του τοπικού κόμβου).

Μια τέτοια εγγραφή στον πίνακα δρομολόγησης περιέχει την IP διεύθυνση του κόμβου του οποίου το

nodeId έχει το κατάλληλο πρόθεμα. Στην πράξη, για κάθε θέση του πίνακα δρομολόγησης επιλέγεται μεταξύ των κόμβων με το αντίστοιχο πρόθεμα, αυτός που είναι κοντά στον τοπικό κόμβο σύμφωνα με ένα μέτρο εγγύτητας. Εάν κανένας κόμβος δεν είναι γνωστός με το κατάλληλο nodeId, τότε η θέση του πίνακα δρομολόγησης παραμένει άδεια. Έτσι, κατά μέσο όρο μόνο $\log_2{}^b N$ γραμμές είναι κατειλημμένες στον πίνακα δρομολόγησης.

Η επιλογή του b περιλαμβάνει επιλογή ανάμεσα στο μέγεθος του κατειλημμένου ποσοστού του πίνακα δρομολόγησης (περίπου $(\log_2{}^b N)x(2^b-1)$ εγγραφές) και στο μέγιστο αριθμό των βημάτων που απαιτούνται για τη δρομολόγηση ανάμεσα σε δύο τυχαίους κόμβους ($\log_2{}^b N$). Με b = 4 και 106 κόμβους, ο πίνακας δρομολόγησης περιέχει περίπου 75 εγγραφές και ο αναμενόμενος αριθμός των βημάτων για δρομολόγηση είναι 4, ενώ με 109 κόμβους ο πίνακας δρομολόγησης περιέχει περίπου 105 εγγραφές και ο αναμενόμενος αριθμός των βημάτων για δρομολόγηση είναι 7.

Το σύνολο γειτονιάς M περιέχει τα nodeIds και τις IP διευθύνσεις των |M| κόμβων που είναι κοντινότερα (σύμφωνα με το μέτρο εγγύτητας) στον τοπικό κόμβο. Το σύνολο αυτό δε χρησιμοποιείται κανονικά στα μηνύματα δρομολόγησης. Η χρησιμότητά του έγκειται στη διατήρηση των ιδιοτήτων τοπικότητας του δικτύου.

Το σύνολο φύλλων L είναι το σύνολο των κόμβων με τα |L| / 2 αριθμητικά αμέσως μεγαλύτερα nodeIds, και τα |L| / 2 αριθμητικά αμέσως μικρότερα nodeIds σε σχέση πάντα με το nodeId του τοπικού κόμβου. Το σύνολο φύλλων χρησιμοποιείται κατά τη δρομολόγηση μηνυμάτων. Τυπικές τιμές του |L| και του |M| είναι 2b ή 2x2b.

Στην εικόνα 8 φαίνεται η πληροφορία ενός υποθετικού κόμβου Pastry με nodeId 10233102 (βάση 4) σε ένα σύστημα που χρησιμοποιεί nodeIds των 16 bits και b=2.

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Εικόνα 8 : Η πληροφορία ενός υποθετικού κόμβου Pastry με nodeId 10233102 (βάση 4) σε ένα σύστημα που χρησιμοποιεί nodeIds των 16 bits και b = 2. Η πρώτη γραμμή του πίνακα δρομολόγησης είναι η γραμμή 0. Το γκρι κελί σε κάθε γραμμή του πίνακα δρομολόγησης δείχνει το αντίστοιχο ψηφίο του nodeId του τοπικού κόμβου. Τα nodeIds σε κάθε εγγραφή είναι χωρισμένα έτσι ώστε να φαίνεται το κοινό πρόθεμα με το 10233102 – επόμενο ψηφίο – υπόλοιπο nodeId. Έχουν παραλειφθεί οι IP διευθύνσεις των κόμβων.

1.2.4.4 Δρομολόγηση στο Pastry

Όταν ένας κόμβος έχει ένα μήνυμα για αποστολή, πρώτα ελέγχει να δει εάν το κλειδί ανήκει στην περιοχή των nodeIds που καλύπτεται από το σύνολο φύλλων του. Εάν αυτό ισχύει, το μήνυμα προωθείται αμέσως στον κόμβο-προορισμό που είναι ο κόμβος στο σύνολο φύλλων με nodeId πλησιέστερα στο κλειδί.

Εάν το κλειδί δεν ανήκει στην παραπάνω περιοχή, τότε χρησιμοποιείται ο πίνακας δρομολόγησης και το μήνυμα προωθείται σε έναν κόμβο που μοιράζεται κοινό πρόθεμα με το αναζητούμενο κλειδί τουλάχιστον κατά ένα ψηφίο περισσότερο από τον τοπικό κόμβο. Σε ορισμένες περιπτώσεις είναι δυνατό η κατάλληλη εγγραφή στον πίνακα δρομολόγησης να είναι άδεια ή ο κόμβος που υποδεικνύει να μην είναι προσπελάσιμος. Τότε, το μήνυμα προωθείται σε έναν κόμβο που το κοινό του πρόθεμα με το αναζητούμενο κλειδί είναι ίδιου μήκους με το κοινό πρόθεμα του τοπικού κόμβου αλλά αριθμητικά πλησιέστερο στο κλειδί. Ένας τέτοιος κόμβος πρέπει να υπάρχει στο σύνολο φύλλων, εάν το μήνυμα δεν έχει ήδη φτάσει στον τελικό κόμβο. Κι, εάν εξαιρέσουμε την περίπτωση |L|/2 διπλανοί κόμβοι στο σύνολο φύλλων να έχουν «πέσει» ταυτόχρονα, τουλάχιστον ένας από αυτούς τους κόμβους πρέπει να είναι ζωντανός. Μάλιστα, λόγω της τυχαίας απόδοσης αναγνωριστικών σε κόμβους, κόμβοι με παραπλήσια αναγνωριστικά μπορεί να διαφέρουν πολύ σε γεωγραφική θέση. Αυτό εξασφαλίζει τη μηδαμινή πιθανότητα να συμβεί το παραπάνω σενάριο.

Αυτή η απλή διαδικασία δρομολόγησης πάντα συγκλίνει, γιατί κάθε βήμα οδηγεί το μήνυμα σε έναν κόμβο που είτε μοιράζεται ένα μακρύτερο πρόθεμα με το κλειδί από ότι ο τοπικός κόμβος είτε μοιράζεται ένα πρόθεμα με το ίδιο μήκος αλλά αριθμητικά κοντύτερα στο κλειδί από ότι ο τοπικός κόμβος. Ο αναμενόμενος αριθμός βημάτων για τη δρομολόγηση ενός μηνύματος είναι $\log_2{}^b N$, με την προϋπόθεση ότι έχουμε ακριβείς πίνακες δρομολόγησης και καμία πρόσφατη αποτυχία κόμβου. [2]

1.2.4.5 Το API του Pastry

Στην παράγραφο αυτή αναφέρουμε επιγραμματικά τις κύριες λειτουργίες του API του Pastry σε απλοποιημένη μορφή.

Οι λειτουργίες που παρέχει το Pastry στις εφαρμογές που δομούνται πάνω από αυτό είναι :

• *nodeId = pastryInit (Credentials, Application)* : Επιτρέπει στον τοπικό κόμβο να προσχωρήσει σε ένα υπάρχον δίκτυο Pastry (ή να ξεκινήσει ένα καινούριο) και να αρχικοποιήσει όλη την απαραίτητη πληροφορία. Επιστρέφει το nodeId του τοπικού κόμβου. Τα Credentials, τα οποία εξαρτώνται από την εφαρμογή, περιέχουν την απαραίτητη πληροφορία για την πιστοποίηση της αυθεντικότητας του τοπικού κόμβου. Το όρισμα Application είναι ένας χειριστής του αντικειμένου της εφαρμογής, ο οποίος παρέχει στο Pastry τις διαδικασίες που πρέπει να εκκινεί, όταν συμβαίνουν συγκεκριμένα γεγονότα (π.χ. άφιξη μηνύματος).

• *route (msg, key)* : Δρομολογεί το δεδομένο msg στον κόμβο με nodeId αριθμητικά κοντύτερα -σε σχέση με

όλους τους ζωντανούς κόμβους του δικτύου Pastry - στο κλειδί key. Οι εφαρμογές που βρίσκονται ένα στρώμα πάνω από το Pastry πρέπει να εξάγουν τις ακόλουθες λειτουργίες:

Επιπλέον, κάθε εφαρμογή που δομείται πάνω από το Pastry, πρέπει να υλοποιεί τις παρακάτω λειτουργίες :

• *deliver (msg, key)* : Καλείται από το Pastry, όταν λαμβάνεται ένα μήνυμα και το nodeId του τοπικού κόμβου είναι το πλησιέστερο στο κλειδί.

• *forward (msg, key, nextId)* : Καλείται από το Pastry, μόλις πριν ένα μήνυμα προωθηθεί στον κόμβο με nodeId = nextId. Θέτοντας το nextId NULL το μήνυμα τερματίζει την πορεία του στον τοπικό κόμβο.

• *newLeafs(leafSet)* : Καλείται από το Pastry όποτε υπάρχει αλλαγή στο σύνολο φύλλων του τοπικού κόμβου. Αυτό παρέχει στην εφαρμογή τη δυνατότητα να προσαρμόζει σταθερές συγκεκριμένες για κάθε εφαρμογή στο σύνολο των φύλλων. [2]

1.2.4.6 Αυτο-οργάνωση του δικτύου και προσαρμογή

1.2.4.6.1 Άφιξη κόμβου

Όταν εισάγεται ένας νέος κόμβος στο δίκτυο, πρέπει να αρχικοποιήσει τους πίνακές του και μετά να ενημερώσει τους άλλους κόμβους για την παρουσία του.

Υποθέτουμε ότι ο καινούριος κόμβος X γνωρίζει έναν κοντινό -σύμφωνα με το μέτρο εγγύτητας- κόμβο A, που ανήκει ήδη στο δίκτυο Pastry (ένας τέτοιος κόμβος μπορεί να εντοπιστεί αυτόματα με χρήση IP multicast «επεκτεινόμενου δακτυλίου» ή να δοθεί απευθείας από το διαχειριστή του συστήματος μέσω εξωτερικών καναλιών).

Ο X ζητά από τον A να δρομολογήσει ένα ειδικό μήνυμα “join” με κλειδί ίσο με X (όπου X το αναγνωριστικό του νέου κόμβου). Το μήνυμα θα παραδοθεί στον κόμβο Z με nodeId πλησιέστερα στο X. Αυτή η αίτηση για “join” έχει σαν αποτέλεσμα να στείλουν οι A, Z, και όλοι οι ενδιάμεσοι κόμβοι από τους οποίους πέρασε το “join”, τους πίνακές τους στον X.

Για την ορθή αρχικοποίηση των πινάκων του, ο νέος κόμβος κάνει χρήση της παραπάνω πληροφορίας που λαμβάνει. Για τον πίνακα φύλλων, ο νέος κόμβος κάνει χρήση του πίνακα φύλλων του κόμβου Z ο οποίος, βάσει της διαδικασίας δρομολόγησης, είναι και αυτός με το αριθμητικά πλησιέστερο αναγνωριστικό στον νέο κόμβο. Για την αρχικοποίηση του πίνακα γειτονιάς, ο νέος κόμβος κάνει χρήση του πίνακα γειτονιάς του κόμβου A αφού, όπως περιγράψαμε και νωρίτερα, ο A είναι επιλεγμένος βάσει της εγγύτητας του στον προς εισαγωγή κόμβο.

Τέλος, για την αρχικοποίηση του πίνακα δρομολόγησης, γίνεται χρήση πληροφορίας από όλους τους κόμβους που συναντήθηκαν κατά τη δρομολόγηση της αίτησης “join”. Στη γενική περίπτωση όπου ο A δεν έχει

κανένα πρόθεμα που να ταυτίζεται με αυτό του νέου κόμβου, η συμπλήρωση του πίνακα δρομολόγησης γίνεται με τον παρακάτω τρόπο. Αρχικά, η γραμμή 0 του νέου κόμβου πρέπει να περιέχει κόμβους με αναγνωριστικά που δεν έχουν κανένα κοινό πρόθεμα με το δικό του nodeId. Έτσι, οι κόμβοι στη γραμμή 0 του κόμβου A είναι κατάλληλοι για τη γραμμή 0 του νέου κόμβου. Ωστόσο, οι υπόλοιπες γραμμές του πίνακα δρομολόγησης τού A δεν πληρούν τις προϋποθέσεις για τις υπόλοιπες σειρές του πίνακα δρομολόγησης του νέου κόμβου. Αντίθετα, αναλογιζόμενοι ότι ο πρώτος κόμβος, έστω B, που συναντήθηκε κατά τη δρομολόγηση του "join" μοιράζεται ένα κοινό ψηφίο με το αναγνωριστικό του νέου κόμβου, μπορούμε να πάρουμε την πρώτη γραμμή από τον πίνακα δρομολόγησης αυτού του κόμβου για την πρώτη γραμμή του πίνακα δρομολόγησης του νέου κόμβου. Ομοίως συμπληρώνονται και οι υπόλοιπες γραμμές του πίνακα δρομολόγησης του νέου κόμβου.

Τελειώνοντας τη συμπλήρωση των πινάκων με πληροφορία για τους γείτονές του, ο νέος κόμβος στέλνει την κατάστασή του (όλη την παραπάνω πληροφορία) σε όλους τους κόμβους που περιέχονται στον πίνακα δρομολόγησης, τον πίνακα φύλλων και τον πίνακα γειτονιάς του κι αυτοί με τη σειρά τους ενημερώνουν τις αντίστοιχες δομές τους λαμβάνοντας υπόψιν τους την ύπαρξη του νέου κόμβου.

1.2.4.6.2 Αποχώρηση κόμβου

Οι κόμβοι στο δίκτυο Pastry μπορεί να πέσουν ή να αναχωρήσουν χωρίς προειδοποίηση. Ένας κόμβος θεωρείται εκτός δικτύου, όταν οι άμεσοι γείτονές του στο χώρο των nodeIds δεν μπορούν πλέον να επικοινωνήσουν μαζί του. Για να αντικατασταθεί ένας μη προσπελάσιμος κόμβος στο σύνολο φύλλων, ο γείτονας του "πεσμένου" κόμβου στο χώρο των nodeIds έρχεται σε επαφή με το ζωντανό κόμβο με το μεγαλύτερο δείκτη από την πλευρά του "πεσμένου" κόμβου και του ζητά τον πίνακα φύλλων του. Το σύνολο που λαμβάνει επικαλύπτεται μερικώς από το σύνολο φύλλων του τοπικού κόμβου και περιέχει κόμβους με κοντινά Ids που δε βρίσκονται στο σύνολο φύλλων του τοπικού κόμβου. Ανάμεσα σε αυτούς τους κόμβους επιλέγεται ο κατάλληλος για να εισαχθεί στο σύνολο φύλλων έχοντας πρώτα εξακριβωθεί ότι είναι ζωντανός. Αυτή η διαδικασία εγγυάται ότι κάθε κόμβος διορθώνει το σύνολο φύλλων του αρκεί να μην έχουν πέσει ταυτόχρονα |L|/2 κόμβοι με γειτονικά nodeIds. Λόγω της διαφοροποίησης (σε θέση, ιδιοκτησία κ.λπ.) των κόμβων με γειτονικά nodeIds μια τέτοια αποτυχία τείνει να είναι απίθανη.

Η αποτυχία ενός κόμβου που εμφανίζεται στον πίνακα δρομολόγησης ενός δεύτερου κόμβου ανιχνεύεται, όταν ο δεύτερος κόμβος επιχειρεί να επικοινωνήσει με τον πρώτο και δε λαμβάνει απάντηση. Αυτό το γεγονός, συνήθως, δεν οδηγεί σε αποτυχία τη δρομολόγηση ενός μηνύματος, αφού η πληροφορία που αποθηκεύει κάθε κόμβος για γειτονικούς του κόμβους, του επιτρέπει να δρομολογήσει το μήνυμα μέσω κάποιου εναλλακτικού μονοπατιού.

Ωστόσο, πρέπει να βρεθεί μια εγγραφή αντικατάστασης, για να διατηρηθεί η ακεραιότητα του πίνακα δρομολόγησης. Ο δεύτερος κόμβος ζητά αντίστοιχες εγγραφές από άλλους κόμβους της ίδιας σειράς μέχρι να βρει έναν κόμβο με το κατάλληλο πρόθεμα.

Το σύνολο γειτονιάς δε χρησιμοποιείται κανονικά στη δρομολόγηση μηνυμάτων, ωστόσο είναι σημαντικό να παραμένει έγκυρο και ενημερωμένο, καθώς παίζει σημαντικό ρόλο στην ανταλλαγή πληροφοριών με κοντινούς κόμβους. Για αυτό το λόγο κάθε κόμβος επιχειρεί να επικοινωνεί περιοδικά με κάθε κόμβο του συνόλου γειτονιάς, για να διαπιστώσει εάν είναι ζωντανός. Εάν κάποιο μέλος δεν ανταποκρίνεται, ο κόμβος ζητά από άλλα μέλη του συνόλου τους πίνακες γειτόνων τους, ελέγχει την απόσταση από κάθε νέο κόμβο που ανακαλύπτει και ενημερώνει τον πίνακά του ανάλογα.

1.2.4.7 Τοπικότητα

Το Pastry φροντίζει η διαδρομή που θα επιλεχθεί για ένα μήνυμα να είναι η μικρότερη δυνατή, κάνοντας χρήση ενός μέτρου εγγύτητας για την επιλογή των γειτόνων κάθε κόμβου. Η αντίληψη του Pastry για δικτυακή εγγύτητα βασίζεται σε ένα κλιμακούμενο σε σχέση με το μέγεθος του δικτύου μέτρο εγγύτητας, όπως ο αριθμός των IP βημάτων δρομολόγησης ή η γεωγραφική απόσταση. Θεωρούμε ότι κάθε εφαρμογή παρέχει μια συνάρτηση σε κάθε κόμβο Pastry, για να μπορεί να καθορίζει την «απόστασή» του από έναν κόμβο με δεδομένη IP διεύθυνση. Μια εφαρμογή αναμένεται να υλοποιεί αυτή τη συνάρτηση ανάλογα με την επιλογή του μέτρου εγγύτητας, χρησιμοποιώντας υπηρεσίες δικτύου όπως το traceroute, και να χρησιμοποιεί την κατάλληλη προσωρινή αποθήκευση και τεχνικές για την ελαχιστοποίηση των επικεφαλίδων.

Σε σχέση με τον πίνακα δρομολόγησης επιδιώκεται όλες οι εγγραφές να αναφέρονται σε κόμβους κοντινούς, σύμφωνα με το μέτρο εγγύτητας, στον τοπικό κόμβο σε σύγκριση με όλους τους άλλους ζωντανούς κόμβους με κατάλληλο πρόθεμα για την εγγραφή. Έτσι, σε κάθε βήμα δρομολόγησης, ένα μήνυμα προωθείται σε έναν σχετικά κοντινό κόμβο με nodeId που μοιράζεται ένα μεγαλύτερο κοινό πρόθεμα ή είναι αριθμητικά πλησιέστερο στο κλειδί, απ' ότι ο τοπικός κόμβος. Έτσι, κάθε βήμα μετακινεί το μήνυμα πλησιέστερα στον προορισμό του στο χώρο των nodeIds, ενώ ταξιδεύει την ελάχιστη δυνατή απόσταση στο χώρο. Η παραπάνω διαδικασία, την οποία δε θα περιγράψουμε περαιτέρω, προφανώς δεν εγγυάται τη συντομότερη διαδρομή από την πηγή στον επιλεγμένο προορισμό, αλλά δίνει αρκετά καλές διαδρομές.

Τέλος, κάποιες εφαρμογές peer-to-peer που χρησιμοποιούν το Pastry, όπως το PAST, αντιγράφουν πληροφορία στους k κόμβους Pastry με τα αριθμητικά πλησιέστερα nodeIds στο κλειδί (τους κόμβους του leafset). Έτσι, εξασφαλίζεται υψηλή διαθεσιμότητα παρά τις αποτυχίες.

Το Pastry δρομολογεί τα μηνύματά του περιμένοντας ένας τουλάχιστον κόμβος από τους k πλησιέστερους στο κλειδί κόμβους να λάβει το αντίστοιχο για αυτόν μήνυμα.

Οι ιδιότητες τοπικότητας του Pastry προσθέτουν το εξής πλεονέκτημα στην παραπάνω δρομολόγηση: κατά μήκος της διαδρομής από έναν πελάτη στον αριθμητικά πλησιέστερο κόμβο, το μήνυμα πρώτα θα φτάσει στον κόμβο που είναι πιο «κοντά» (σύμφωνα με το μέτρο εγγύτητας) στον πελάτη ανάμεσα στους k κόμβους. Έτσι, μειώνεται και η καθυστέρηση για τον πελάτη και ο φόρτος του δικτύου. [2]

1.2.4.8 Συμπερασματικά

Το Pastry είναι ένα peer-to-peer σύστημα εντοπισμού περιεχομένου και δρομολόγησης. Χρησιμοποιείται σαν δομικό στοιχείο για το χτίσιμο μιας ποικιλίας peer-to-peer δικτυακών εφαρμογών. Δρομολογεί σε οποιονδήποτε κόμβο στο υπερκείμενο δίκτυο σε λιγότερα από O (log N) βήματα, και διατηρεί πίνακες δρομολόγησης με μόνο O (log N) εισόδους. Λαμβάνει υπόψη την τοπικότητα κατά τη δρομολόγηση των μηνυμάτων.

1.2.5 Tapestry

1.2.5.1 Γενικά

Το Tapestry είναι μια επεκτάσιμη υποδομή, η οποία παρέχει αποκεντρωμένο εντοπισμό και δρομολόγηση ενός αντικειμένου (Decentralized Object Location and Routing – DOLR). Το DOLR interface εστιάζει στη δρομολόγηση μηνυμάτων σε τελικά σημεία, όπως είναι οι κόμβοι και τα αντίγραφα αντικειμένων. Το DOLR παρέχει εικονικούς πόρους, αφού τα τελικά σημεία παίρνουν ονόματα αναγνωριστικών κωδικοποιώντας μηδενική πληροφορία για τη φυσική τους θέση. Με αυτήν την υλοποίηση επιτρέπεται η παράδοση μηνυμάτων σε κινητά τελικά σημεία ή αντίγραφα τελικών σημείων κατά την παρουσία αστάθειας στην υποκείμενη υποδομή.

Σαν αποτέλεσμα, ένα δίκτυο DOLR παρέχει μια απλή πλατφόρμα πάνω στην οποία μπορούν να υλοποιηθούν κατανεμημένες εφαρμογές, ενώ απαλλάσσει τον σχεδιαστή των εφαρμογών από το να πρέπει να λάβει υπόψιν τη δυναμικότητα του δικτύου. Ήδη το Tapestry έχει κάνει δυνατή την ανάπτυξη εφαρμογών αποθήκευσης μεγάλης κλίμακας όπως το OceanStore και συστημάτων multicast διανομής όπως το Bayeux.

Το Tapestry χρησιμοποιεί προσαρμοστικούς αλγόριθμους, για να επιδείξει αντοχή σε λάθη. Η αρχιτεκτονική του είναι τμηματική και περιλαμβάνει μια εκτεταμένη λειτουργικότητα "upcall" γύρω από έναν απλό, υψηλής απόδοσης μηχανισμό δρομολόγησης. Αυτό το API επιτρέπει στους προγραμματιστές να επεκτείνουν την υπάρχουσα λειτουργικότητα, όταν αυτή κρίνεται ανεπαρκής για την εφαρμογή τους.

Το Tapestry επιτρέπει στις εφαρμογές να τοποθετούν αντικείμενα ανάλογα με τις ανάγκες τους και δε ρυθμίζει, όπως άλλα δομημένα συστήματα peer-to-peer, τον αριθμό και τη θέση των αντιγράφων των αντικειμένων μέσω ενός DHT interface. Το Tapestry «δημοσιεύει» δείκτες θέσης μέσα στο δίκτυο, για να διευκολυνθεί η δρομολόγηση στα αντικείμενα που έχουν μικρή διασπορά στο δίκτυο. Αυτή η τεχνική δίνει στο Tapestry ιδιότητες τοπικότητας: οι ερωτήσεις για κοντινά αντικείμενα ικανοποιούνται γενικά σε χρόνο ανάλογο της απόστασης ανάμεσα στην πηγή της ερώτησης και στο πλησιέστερο αντίγραφο του αντικειμένου.

1.2.5.2 API Δικτύωσης του DOLR

Οι κόμβοι Tapestry συμμετέχουν στο υπερκείμενο δίκτυο και τους ανατίθενται nodeIDs ομοιόμορφα και τυχαία από ένα μεγάλο χώρο αναγνωριστικών. Περισσότεροι από δύο κόμβοι μπορούν να φιλοξενούνται από το ίδιο φυσικό μηχάνημα. Στα τελικά σημεία μιας συγκεκριμένης εφαρμογής ανατίθενται μοναδικά αναγνωριστικά (globally unique identifiers – GUID) επιλεγμένα από τον ίδιο χώρο αναγνωριστικών. Το Tapestry χρησιμοποιεί έναν χώρο αναγνωριστικών των 160 bits με μία γενικά ορισμένη βάση (π.χ. η δεκαεξαδική βάση μας δίνει αναγνωριστικά των 40 ψηφίων). Το Tapestry υποθέτει ότι τα nodeIDs και τα GUIDs είναι γενικά ομοιόμορφα κατανεμημένα στο χώρο αναγνωριστικών, πράγμα που επιτυγχάνεται

χρησιμοποιώντας έναν αλγόριθμο ασφαλούς κατακερματισμού όπως ο SHA-1. Γενικά, ένας κόμβος N έχει nodeID Nid, και ένα αντικείμενο O έχει GUID OG.

Για την ομοιόμορφη κατανομή των αναγνωριστικών, που με τη σειρά της εξασφαλίζει την ισοκατανομή του φορτίου μεταξύ των κόμβων, ισχύει ότι όσο πιο πολλοί είναι οι συμμετέχοντες κόμβοι στο δίκτυο, τόσο πιο ομοιόμορφη είναι και η κατανομή τους. Έτσι, η αποδοτικότητα του Tapestry γενικά βελτιώνεται όσο μεγαλώνει το μέγεθος του δικτύου. Για το λόγο αυτό, ωφελεί πολλές εφαρμογές να μοιράζονται το ίδιο μεγάλο υπερκείμενο δίκτυο Tapestry. Για να είναι δυνατή η συνύπαρξη πολλών εφαρμογών, κάθε μήνυμα περιέχει ένα αναγνωριστικό της εφαρμογής Aid (applicationID), το οποίο χρησιμοποιείται για την επιλογή μιας διεργασίας ή εφαρμογής για την παράδοση του μηνύματος στον προορισμό (ομοίως με το ρόλο της πόρτας στο TCP/IP), ή για την επιλογή ενός χειριστή upcall όπου χρειάζεται.

Το API Δικτύωσης του DOLR αποτελείται από τέσσερις βασικές κλήσεις:

- *PUBLISHOBJECT (OG, Aid)* : δημοσιεύει, ή κάνει διαθέσιμο, το αντικείμενο O στον τοπικό κόμβο. Αυτή η κλήση είναι καλύτερης δυνατής προσπάθειας, και δε λαμβάνει επιβεβαίωση.
- *UNPUBLISHOBJECT (OG, Aid)*: κλήση καλύτερης δυνατής προσπάθειας για την κατάργηση αντιστοιχίσεων θέσης για το O.
- *ROUTETOOBJECT (OG, Aid)* : δρομολογεί μήνυμα στη θέση ενός αντικειμένου με GUID OG.
- *ROUTETONODE (N, Aid, Exact)* : δρομολογεί μήνυμα στην εφαρμογή Aid στον κόμβο N. Το "Exact" προσδιορίζει εάν το ID του προορισμού πρέπει να ταιριάζει απόλυτα, για να παραδοθεί το περιεχόμενο του μηνύματος.

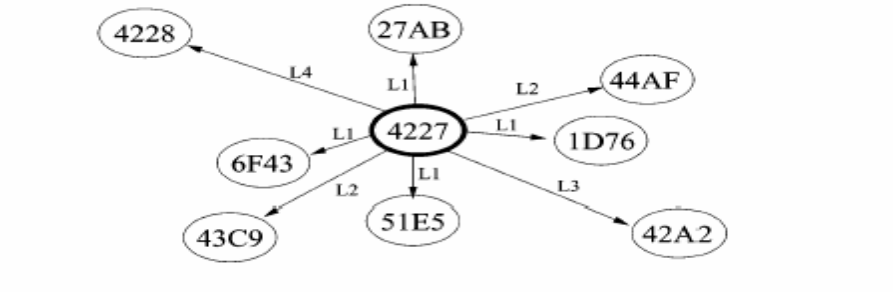
1.2.5.3 Δρομολόγηση και εντοπισμός αντικειμένου

Το Tapestry αντιστοιχεί δυναμικά κάθε αναγνωριστικό G σε έναν μοναδικό ζωντανό κόμβο, ο οποίος καλείται ρίζα(root) του αναγνωριστικού, GR. Εάν ένας κόμβος N υπάρχει με Nid = G , τότε αυτός ο κόμβος είναι η ρίζα του G. Για να είναι σε θέση να δρομολογεί μηνύματα, κάθε κόμβος διατηρεί έναν πίνακα δρομολόγησης που περιέχει nodeIDs και IP διευθύνσεις των κόμβων με τους οποίους επικοινωνεί. Αυτοί οι κόμβοι καλούνται γείτονες του τοπικού κόμβου. Κατά τη δρομολόγηση προς το GR, τα μηνύματα προωθούνται κατά τις συνδέσεις των γειτόνων προς κόμβους με nodeIDs σταδιακά πλησιέστερα (σύμφωνα με τη σύγκριση προθεμάτων) στο G στο χώρο των αναγνωριστικών.

1.2.5.3.1 Δίκτυο δρομολόγησης

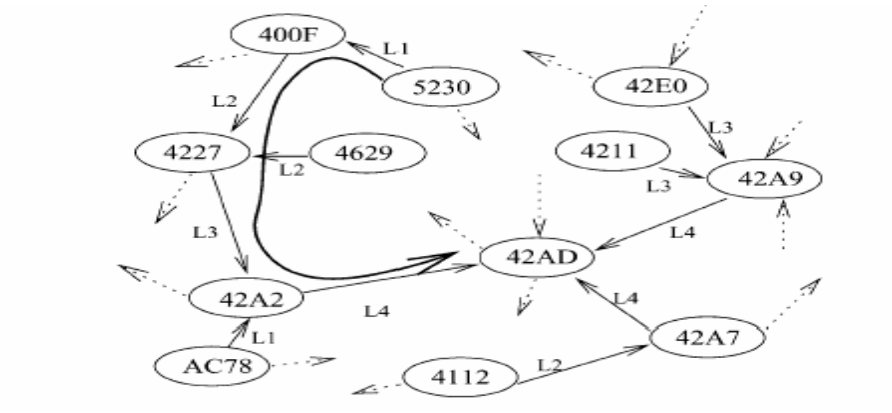
Το Tapestry χρησιμοποιεί τοπικούς πίνακες δρομολόγησης σε κάθε κόμβο, οι οποίοι καλούνται χάρτες γειτόνων (neighbor maps), για τη δρομολόγηση μηνυμάτων στο ID προορισμού. Η δρομολόγηση γίνεται ψηφίο

προς ψηφίο (π.χ. 4*** □ 42** □ 42A* □ 42AD, όπου το * αντιπροσωπεύει έναν άγνωστο δεκαεξαδικό αριθμό). Αυτή η προσέγγιση είναι παρόμοια με τη δρομολόγηση μακρύτερου προθέματος που χρησιμοποιείται από την CIDR (Classless InterDomain Routing) ανάθεση IP διεύθυνσης. Ένας κόμβος N έχει έναν χάρτη γειτόνων πολλαπλών επιπέδων, όπου σε κάθε επίπεδο περιέχονται συνδέσεις σε κόμβους που έχουν το ίδιο πρόθεμα μέχρι μια θέση ψηφίου στο ID και ο αριθμός των εγγραφών σε κάθε επίπεδο ισούται με τη βάση του ID. Η i-ιοστή εγγραφή στο j-ιοστό επίπεδο είναι το ID και η θέση του κοντινότερου κόμβου που ξεκινά με πρόθεμα (N, j – 1) + "i" (π.χ. η 9η εγγραφή του 4ου επιπέδου για τον κόμβο 325AE είναι ο κοντινότερος κόμβος με ID που ξεκινά με 3259). Αυτή η έννοια του «κοντινότερου» κόμβου προσδίδει στο Tapestry ιδιότητες τοπικότητας. Στην εικόνα που ακολουθεί φαίνεται το δίκτυο δρομολόγησης του Tapestry από την πλευρά ενός κόμβου. Οι εξερχόμενες γειτονικές συνδέσεις δείχνουν σε κόμβους με μερικώς κοινό πρόθεμα. Οι εγγραφές υψηλότερου επιπέδου έχουν περισσότερα κοινά ψηφία με τον τοπικό κόμβο.



Εικόνα 9 : Ο τοπικός πίνακας δρομολόγησης ενός κόμβου υπό μορφή δικτύου δρομολόγησης.

Στο επόμενο σχήμα φαίνεται η διαδρομή που μπορεί να πάρει ένα μήνυμα μέσα στην υποδομή Tapestry. Ο δρομολογητής για το n-ιοστό βήμα μοιράζεται ένα πρόθεμα μήκους ≥ n με το ID του προορισμού. Έτσι, το Tapestry, για να δρομολογήσει, ψάχνει στο (n+1) – ιοστό επίπεδο του χάρτη για την εγγραφή που ταιριάζει με το επόμενο ψηφίο του ID του προορισμού. Αυτή η μέθοδος εγγυάται ότι οποιοσδήποτε κόμβος στο σύστημα θα βρεθεί σε το πολύ $\log_{\beta}N$ βήματα επιπέδου εφαρμογής, όταν το σύστημα έχει μέγεθος χώρου αναγνωριστικών NIDs βάσης β, και υποθέτοντας ότι οι χάρτες γειτόνων είναι συνεπείς. Όταν ένα ψηφίο δεν μπορεί να ταιριάξει, το Tapestry ψάχνει για ένα «κοντινό» ψηφίο στον πίνακα δρομολόγησης. Αυτό καλείται surrogate routing. Όταν, δηλαδή, ένα ID που δεν υπάρχει αντιστοιχίζεται σε κάποιο ζωντανό κόμβο με παρόμοιο ID.



Εικόνα 10 : Η διαδρομή ενός μηνύματος που ξεκινά από τον κόμβο 5230 και κατευθύνεται στον κόμβο 42AD.

Κάθε αναγνωριστικό G αντιστοιχίζεται σε έναν μοναδικό κόμβο-ρίζα GR με συνεχή χρήση της συνάρτησης NEXTHOP, η οποία επιλέγει τον εξερχόμενο σύνδεσμο. Αυτή η συνάρτηση εντοπίζει το επόμενο βήμα προς τη ρίζα του αναζητούμενου αναγνωριστικού, δεδομένου του αριθμού του προηγούμενου βήματος, n, και του GUID προορισμού G. Η τιμή επιστροφής είναι είτε η IP του κόμβου που θα αποτελέσει το επόμενο βήμα είτε "self", εάν ρίζα είναι ο τοπικός κόμβος.

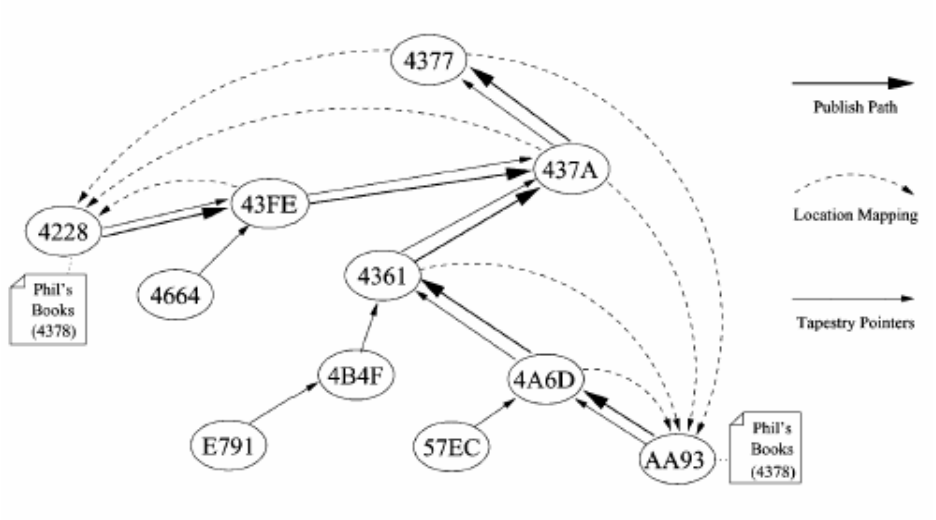
Οι κύριοι σύνδεσμοι γειτόνων, όπως φαίνονται στην παραπάνω εικόνα, έχουν εφεδρικούς συνδέσμους με ίδια προθέματα. Στο n-ιοστό επίπεδο δρομολόγησης, οι c σύνδεσμοι γειτόνων διαφέρουν μόνο στο n-ιοστό ψηφίο. Υπάρχουν c x β δείκτες σε κάθε επίπεδο, και το συνολικό μέγεθος ενός χάρτη γειτόνων είναι c x β x $\log_{\beta}$N. Κάθε κόμβος αποθηκεύει επίσης αντίστροφες αναφορές (backpointers) στους κόμβους που δείχνουν σε αυτόν. Ο συνολικός αριθμός τέτοιων εγγραφών είναι c x β x $\log_{\beta}$N.

1.2.5.3.2 Εντοπισμός και δημοσίευση αντικειμένου

Όπως προαναφέρθηκε, κάθε αναγνωριστικό G έχει έναν μοναδικό κόμβο-ρίζα GR που του ανατίθεται από τη διαδικασία δρομολόγησης. Κάθε τέτοιος κόμβος-ρίζα αποκτά ένα μοναδικό δέντρο με μηνύματα από κόμβους-φύλλα να διασχίζουν ενδιάμεσους κόμβους στην πορεία τους για τη ρίζα. Αυτή η ιδιότητα χρησιμεύει για τον εντοπισμό αντικειμένων διανέμοντας "soft-state" πληροφορία καταλόγου στους κόμβους συμπεριλαμβανομένου και της ρίζας του αντικειμένου.

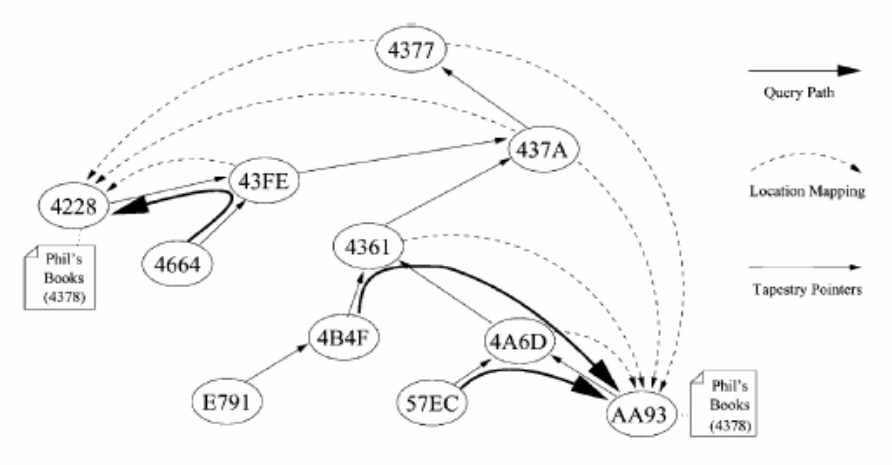
Ένας εξυπηρετητής S που αποθηκεύει ένα αντικείμενο O με GUID OG και ρίζα OR περιοδικά διαφημίζει ή δημοσιεύει αυτό το αντικείμενο δρομολογώντας ένα μήνυμα δημοσίευσης προς τον OR. Γενικά, το nodeID του OR είναι διαφορετικό από OG. Ο OR είναι ο μοναδικός κόμβος που φτάνει η δρομολόγηση με διαδοχικές κλήσεις της NEXTHOP (*, OG). Κάθε κόμβος κατά μήκος της διαδρομής δημοσίευσης αποθηκεύει μία αντιστοίχηση δείκτη {OG, S} αντί για ένα αντίγραφο του αντικειμένου. Όταν υπάρχουν αντίγραφα ενός αντικειμένου σε διαφορετικούς εξυπηρετητές, κάθε εξυπηρετητής δημοσιεύει το αντίγραφό του. Οι κόμβοι

Tapestry αποθηκεύουν αντιστοιχίσεις θέσης για αντίγραφα αντικειμένων ταξινομημένες σύμφωνα με τη δικτυακή καθυστέρηση από τον εαυτό τους.



Εικόνα 11 : Παράδειγμα δημοσίευσης αντικειμένου στο Tapestry. Δύο αντίγραφα του αντικειμένου 4378 διαφημίζονται / δημοσιεύονται στον κόμβο-ρίζα τους, 4377. Τα μηνύματα δημοσίευσης κατευθύνονται προς τη ρίζα αφήνοντας ένα δείκτη θέσης προς το αντικείμενο σε κάθε βήμα που συναντούν στη διαδρομή τους.

Ένας πελάτης εντοπίζει το Ο δρομολογώντας ένα μήνυμα στο OR. Κάθε κόμβος στη διαδρομή ελέγχει εάν αποθηκεύει αντιστοίχηση θέσης για το Ο. Εάν ναι, ανακατευθύνει το μήνυμα στον κάτοχο S. Διαφορετικά, προωθεί το μήνυμα παρακάτω προς το OR. Στην παρακάτω εικόνα φαίνονται τρεις αναζητήσεις αντικειμένων.



Εικόνα 12 : Παράδειγμα δρομολόγησης σε αντικείμενο στο Tapestry. Κόμβοι από διαφορετικά σημεία του δικτύου δρομολογούν μηνύματα προς (για) το αντικείμενο 4378. Όλα τα μηνύματα κινούνται προς τη ρίζα του 4378. Όταν συναντούν το μονοπάτι δημοσίευσης, ακολουθούν το το δείκτη θέσης προς το πλησιέστερο αντίγραφο.

Σε κάθε βήμα μειώνονται οι κόμβοι που ικανοποιούν τον περιορισμό του προθέματος κατά έναν παράγοντα ίσο με τη βάση του αναγνωριστικού. Μηνύματα που στέλνονται στον ίδιο προορισμό από δύο κοντινούς κόμβους γενικά θα συναντηθούν σύντομα κατά την πορεία τους προς τον προορισμό, για τον εξής λόγο: κάθε βήμα αυξάνει το μήκος του προθέματος που απαιτείται για το επόμενο βήμα, η διαδρομή προς τη ρίζα είναι συνάρτηση μόνο του ID του προορισμού, και όχι του nodeID της πηγής (όπως στο Chord), και τα γειτονικά βήματα διαλέγονται με γνώμονα την τοπικότητα του δικτύου, η οποία αποτελεί δυναμικό στοιχείο. Συνεπώς, όσο πιο κοντά σε απόσταση δικτύου είναι ένας πελάτης σε ένα αντικείμενο, τόσο το συντομότερο οι ερωτήσεις του για το αντικείμενο θα συναντήσουν το μονοπάτι δημοσίευσης του αντικειμένου και τόσο πιο γρήγορα θα το φτάσουν. Εφόσον οι κόμβοι ταξινομούν τους δείκτες στα αντικείμενα σύμφωνα με την απόσταση από τους εαυτούς τους, οι ερωτήσεις δρομολογούνται σε κοντινά αντίγραφα των αντικειμένων.

1.2.5.4 Δυναμικοί αλγόριθμοι του Tapestry

1.2.5.4.1 Εισαγωγή κόμβου

Υπάρχουν τέσσερα βήματα κατά την εισαγωγή ενός κόμβου N σε ένα δίκτυο Tapestry.

1. *Οι κόμβοι που χρειάζονται να γνωρίζουν την είσοδο του N ενημερώνονται για τον N, γιατί ο N συμπληρώνει μια άδεια είσοδο στον πίνακα δρομολόγησής τους.*
2. *Ο N μπορεί να γίνει η νέα ρίζα για υπάρχοντα αντικείμενα. Οι αναφορές σε αυτά τα αντικείμενα πρέπει πλέον να δείχνουν προς τον N, για να διατηρηθεί η διαθεσιμότητα των αντικειμένων.*
3. *Οι αλγόριθμοι πρέπει να φτιάξουν ένα βελτιστοποιημένο πίνακα δρομολόγησης για τον N με ιδιότητες τοπικότητας.*
4. *Οι κόμβοι κοντά στον N ενημερώνονται και μπορεί να χρησιμοποιήσουν τον N στους πίνακες δρομολόγησής τους σαν βελτιστοποίηση.*

Η εισαγωγή του κόμβου N ξεκινά στον S που είναι ο κόμβος-ρίζα στον οποίο αντιστοιχεί το Nid στο υπάρχον δίκτυο. Ο S βρίσκει το p, το μήκος του μεγαλύτερου προθέματος που μοιράζεται το ID του με το Nid. Ο S στέλνει ένα μήνυμα Acknowledged Multicast που φτάνει σε όλους τους υπάρχοντες κόμβους που μοιράζονται το ίδιο πρόθεμα. Όταν οι κόμβοι λαμβάνουν το μήνυμα, προσθέτουν τον N στους πίνακες δρομολόγησής τους και τροποποιούν αναφορές σε τοπικούς δείκτες όπως χρειάζεται. Έτσι, συμπληρώνονται τα βήματα 1 και 2.

Οι κόμβοι που λαμβάνουν το multicast μήνυμα επικοινωνούν με τον N και γίνονται το πρώτο σύνολο γειτόνων που χρησιμοποιείται για τη δόμηση του πίνακα δρομολόγησης του N. Ο N διεξάγει μια επαναληπτική αναζήτηση κοντινότερων γειτόνων ξεκινώντας με το επίπεδο δρομολόγησης p. Χρησιμοποιεί το σύνολο των γειτόνων του, για να γεμίσει το επίπεδο δρομολόγησης p, αφήνει στη λίστα τους πλησιέστερους k κόμβους, και

ζητά από τους k κόμβους να στείλουν σε αυτόν τους δείκτες προς τους προηγούμενούς τους σε αυτό το επίπεδο. Το σύνολο που προκύπτει περιέχει όλους τους κόμβους που δείχνουν σε οποιονδήποτε από τους k κόμβους στο προηγούμενο επίπεδο δρομολόγησης, και γίνεται το επόμενο σύνολο γειτόνων. Στη συνέχεια μειώνεται το p και επαναλαμβάνεται η παραπάνω διαδικασία μέχρι να γεμίσουν όλα τα επίπεδα. Έτσι ολοκληρώνεται το βήμα 3. Οι κόμβοι με τους οποίους έρχεται σε επαφή ο N κατά τον επαναληπτικό αλγόριθμο χρησιμοποιούν τον N, για να βελτιστοποιήσουν, όπου είναι δυνατό, τους πίνακες δρομολόγησής τους. Συνεπώς, συμπληρώνεται και το βήμα 4.

Για να διασφαλιστεί ότι οι κόμβοι που εισάγονται στο δίκτυο δεν αποτυγχάνουν να ενημερώσουν (και να ενημερωθούν) για την ύπαρξή τους, κάθε κόμβος A που ανήκει στο multicast δέντρο κρατά πληροφορία για κάθε κόμβο B που κάνει ακόμα multicast προς τα κάτω σε κάποιον από τους γείτονές του. Αυτή η πληροφορία συνήθως πληροφορεί κάθε κόμβο C που έχει τον A στο multicast δέντρο του για τον B. Τέλος, το multicast δέντρο αφήνει μια λίστα από κενά στον πίνακα δρομολόγησης του νέου κόμβου. Ο κόμβοι ελέγχουν τους πίνακές τους σε σχέση με τον πίνακα δρομολόγησης του νέου κόμβου και τον ενημερώνουν με εγγραφές που μπορούν να γεμίσουν τα κενά.

1.2.5.4.2 Εκούσια διαγραφή κόμβου

Εάν ένα κόμβος N εγκαταλείπει το Tapestry ηθελημένα, ενημερώνει το σύνολο D των κόμβων που δείχνουν οι δείκτες προς τους προηγούμενούς του για την πρόθεσή του, στέλνοντας μαζί και έναν κόμβο-αντικαταστάτη από κάθε επίπεδο δρομολόγησης του δικού του πίνακα δρομολόγησης. Οι κόμβοι που ενημερώνονται στέλνουν κίνηση για την αναδημοσίευση του αντικειμένου και στον N και στον αντικαταστάτη του.

Εν τω μεταξύ, ο N δρομολογεί αναφορές σε αντικείμενα, για τα οποία αποτελεί ρίζα, στις νέες τους ρίζες και στέλνει σήμα στους κόμβους που ανήκουν στο D, όταν τελειώνει.

1.2.5.4.3 Ακούσια διαγραφή κόμβου

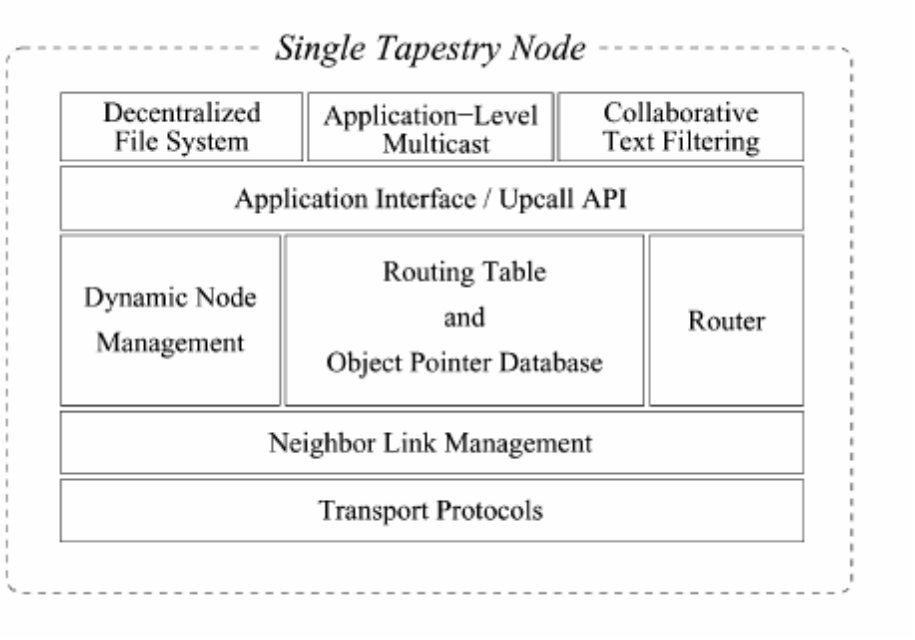
Σε ένα δυναμικό, επιρρεπές σε λάθη δίκτυο όπως είναι το Διαδίκτυο οι κόμβοι γενικά εγκαταλείπουν το δίκτυο απροετοίμαστοι και αυτό οφείλεται σε αποτυχίες συνδέσεων ή σε αποτυχίες άλλων τμημάτων του δικτύου. Επιπλέον, κόμβοι μπορεί να μπουν και να βγουν από το δίκτυο πολλές φορές σε ένα μικρό χρονικό διάστημα. Το Tapestry βελτιώνει τη διαθεσιμότητα και δρομολόγηση αντικειμένου σε ένα τέτοιο περιβάλλον προσθέτοντας εφεδρεία στους πίνακες δρομολόγησης και στις αναφορές θέσης αντικειμένων (π.χ. οι c-1 εφεδρικοί δείκτες προώθησης για κάθε είσοδο του πίνακα δρομολόγησης).

Για τη διατήρηση της διαθεσιμότητας και της εφεδρείας, οι κόμβοι χρησιμοποιούν περιοδικούς φάρους (beacons), για να ανιχνεύσουν αποτυχίες κόμβων και εξερχόμενων συνδέσεων. Τέτοια γεγονότα πυροδοτούν

διορθώσεις στο δίκτυο δρομολόγησης και εκκινούν αναδιανομή και αντιγραφή αναφορών σε θέσεις αντικειμένων. Επιπλέον, η διαδικασία διόρθωσης εντείνεται από την soft-state αναδημοσίευση των αναφορών σε αντικείμενα. Το Tapestry είναι πολύ αποδοτικό και διατηρεί σχεδόν 100% ποσοστό επιτυχίας στη δρομολόγηση μηνυμάτων σε κόμβους και αντικείμενα.

1.2.5.5 Αρχιτεκτονική και υλοποίηση ενός κόμβου Tapestry

Στην εικόνα 13, που ακολουθεί, απεικονίζεται η λειτουργική διαστρωμάτωση ενός κόμβου Tapestry. Στην κορυφή φαίνονται οι εφαρμογές που έρχονται σε επαφή με το υπόλοιπο σύστημα μέσω του Tapestry API. Κάτω από αυτό βρίσκονται τα στρώματα δρομολόγησης (router) και δυναμικής διαχείρισης κόμβων (dynamic node management). Το πρώτο επεξεργάζεται μηνύματα δρομολόγησης και θέσης, ενώ το δεύτερο χειρίζεται την άφιξη και την αναχώρηση κόμβων στο δίκτυο. Αυτά τα δύο στρώματα επικοινωνούν μέσω του πίνακα δρομολόγησης. Στη βάση βρίσκονται τα στρώματα μεταφοράς (transport) και γειτονικών συνδέσεων (neighbor links), τα οποία μαζί παρέχουν ένα στρώμα ανταλλαγής μηνυμάτων μεταξύ των κόμβων.



Εικόνα 13 : Αρχιτεκτονική επιμέρους τμημάτων του Tapestry. Τα μηνύματα κατευθύνονται προς τα πάνω από τα στρώματα φυσικού δικτύου και προς τα κάτω από τα στρώματα εφαρμογών. Κρίσιμο στρώμα για την επικοινωνία είναι αυτό του δρομολογητή.

• *Στρώμα μεταφοράς (Transport layer)*

Το στρώμα μεταφοράς παρέχει την αφαίρεση των καναλιών επικοινωνίας ανάμεσα σε έναν υπερκείμενο κόμβο και σε έναν άλλο. Αντιστοιχεί στο στρώμα 4 της OSI αρχιτεκτονικής. Χρησιμοποιώντας τις δυνατότητες του εκάστοτε λειτουργικού συστήματος είναι δυνατές πολλές υλοποιήσεις καναλιών. Αυτή τη

στιγμή υποστηρίζονται δύο υλοποιήσεις (TCP/IP και UDP/IP).

• *Στρώμα γειτονικών συνδέσεων (Neighbor link layer)*

Πάνω από το στρώμα μεταφοράς είναι το στρώμα γειτονικών συνδέσεων. Παρέχει ασφαλείς αλλά αναξιόπιστες υπηρεσίες δεδομενογραφήματος(UDP) στα παραπάνω στρώματα συμπεριλαμβανομένου του τεμαχισμού και της συναρμολόγησης μεγάλων μηνυμάτων. Την πρώτη φορά που ένα υψηλότερο στρώμα επιθυμεί να επικοινωνήσει με έναν άλλο κόμβο πρέπει να δώσει στο στρώμα γειτονικών συνδέσεων τη φυσική διεύθυνση (IP διεύθυνση και πόρτα) του προορισμού. Σε περίπτωση που ένα υψηλότερο στρώμα θέλει να κάνει χρήση ενός ασφαλούς καναλιού, τότε πρέπει να δώσει επίσης ένα δημόσιο κλειδί για τον απομακρυσμένο κόμβο. Το στρώμα γειτονικών συνδέσεων χρησιμοποιεί αυτή την πληροφορία, για να εγκαταστήσει μια σύνδεση με τον απομακρυσμένο κόμβο.

Οι συνδέσεις εγκαθίστανται μετά από αίτηση ανώτερων επιπέδων του Tapestry. Για να αποφευχθεί η κατάχρηση πόρων του λειτουργικού συστήματος που δεν αφθονούν, όπως οι δείκτες σε αρχεία, το στρώμα γειτονικών συνδέσεων μπορεί να κλείνει περιοδικά μερικές συνδέσεις. Οι κλεισμένες συνδέσεις μπορούν να ανοιχθούν μετά από αίτηση. Μια σημαντική λειτουργία αυτού του στρώματος είναι η συνεχής παρακολούθηση των συνδέσεων και η προσαρμογή. Παρέχει ανίχνευση λαθών μέσω soft-state μηνυμάτων keep-alive και εκτιμήσεις καθυστέρησης και ποσοστού απωλειών. Το στρώμα γειτονικών συνδέσεων ενημερώνει τα υψηλότερα στρώματα όποτε τα χαρακτηριστικά ενός συνδέσμου αλλάζουν σημαντικά.

Αυτό το στρώμα βελτιστοποιεί επίσης την επεξεργασία μηνυμάτων, καθώς αναλαμβάνει το parsing των επικεφαλίδων των μηνυμάτων και το deserialising μόνο του περιεχομένου των μηνυμάτων, όταν χρειάζεται. Τέλος, η πιστοποίηση κόμβου και οι κώδικες πιστοποίησης μηνυμάτων (message authentication codes – MACs) μπορούν να ενσωματωθούν σε αυτό το στρώμα για επιπλέον ασφάλεια.

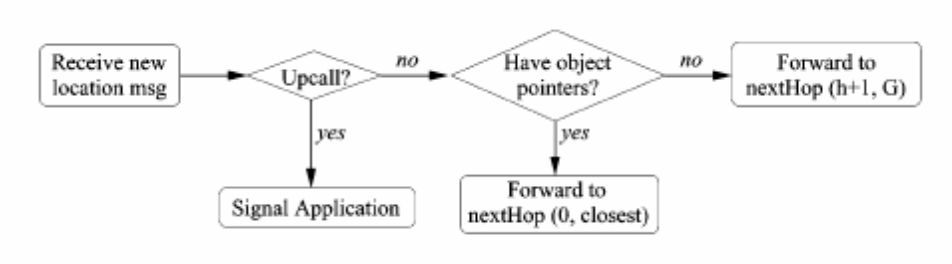
• *Στρώμα δρομολογητή (Router layer)*

Ενώ το στρώμα γειτονικών συνδέσεων παρέχει βασικές δικτυακές υπηρεσίες, το στρώμα δρομολογητή παρέχει μία και μοναδική λειτουργικότητα στο Tapestry. Σε αυτό το στρώμα περιλαμβάνονται ο πίνακας δρομολόγησης και οι δείκτες τοπικών αντικειμένων.

Όπως έχει ήδη αναφερθεί, το δίκτυο δρομολόγησης είναι μια λίστα από γείτονες ταξινομημένους σύμφωνα με το πρόθεμα, που αποθηκεύονται στον πίνακα δρομολόγησης. Ο δρομολογητής εξετάζει το GUID προορισμού των μηνυμάτων που λαμβάνει και καθορίζει το επόμενο βήμα τους χρησιμοποιώντας τον πίνακα και τους δείκτες τοπικών αντικειμένων. Τα μηνύματα περνάνε στη συνέχεια στο στρώμα γειτονικών συνδέσεων για παράδοση.

Στην παρακάτω εικόνα βλέπουμε το διάγραμμα ροής της διαδικασίας εντοπισμού ενός αντικειμένου.

Τα μηνύματα φτάνουν από το στρώμα γειτονικών συνδέσεων στα αριστερά. Κάποια από τα μηνύματα πυροδοτούν περαιτέρω upcalls και βάζουν αμέσως σε λειτουργία τους χειριστές των upcalls. Διαφορετικά, οι δείκτες στα τοπικά αντικείμενα ελέγχονται μήπως υπάρξει ταίριασμα με το GUID που αναζητείται. Εάν πράγματι βρεθούν δύο GUIDs που να ταυτίζονται, το μήνυμα προωθείται στον κοντινότερο κόμβο από το σύνολο των δεικτών που ταιριάζουν με το αναζητούμενο GUID. Αλλιώς, το μήνυμα προωθείται στο επόμενο βήμα προς τον κόμβο-ρίζα.



Εικόνα 14 : Διάγραμμα ροής επεξεργασίας μηνυμάτων.

Πρέπει να σημειώσουμε ότι ο πίνακας δρομολόγησης και η βάση δεδομένων των δεικτών προς αντικείμενα μεταβάλλονται συνεχώς από το στρώμα δυναμικής διαχείρισης κόμβων και το στρώμα γειτονικών συνδέσεων. Επί παραδείγματι, λόγω συνεχών αλλαγών στις καθυστερήσεις συνδέσεων, το στρώμα γειτονικών συνδέσεων μπορεί να αναδιατάξει τις προτιμήσεις που έχουν ανατεθεί στους γείτονες που καταλαμβάνουν την ίδια εγγραφή στον πίνακα δρομολόγησης. Ομοίως, το στρώμα δυναμικής διαχείρισης κόμβων μπορεί να προσθέσει ή να αφαιρέσει δείκτες προς αντικείμενα μετά την άφιξη ή αναχώρηση γειτόνων.

1.2.5.6 Upcall Interface του Tapestry

Ενώ το DOLR API αποτελεί ένα ισχυρό interface εφαρμογών, άλλες λειτουργίες όπως το multicast απαιτούν περισσότερο έλεγχο πάνω στις λεπτομέρειες της δρομολόγησης και της αναζήτησης αντικειμένου. Για να είναι δυνατός αυτός ο επιπλέον έλεγχος, το Tapestry υποστηρίζει έναν εκτεταμένο upcall μηχανισμό. Μάλιστα, αναμένεται με την ωρίμανση της χρήσης των υπερκείμενων υποδομών να δημιουργηθεί η ανάγκη για προσαρμογή σε ένα σύνολο από καλά δοκιμασμένες και συχνά χρησιμοποιούμενες συμπεριφορές δρομολόγησης.

Η αλληλεπίδραση ανάμεσα στο Tapestry και στους χειριστές των εφαρμογών γίνεται μέσω τριών βασικών κλήσεων. Το G είναι ένα γενικευμένο κλειδί – έτσι, θα μπορούσε να είναι ένα nodeID Nid ή ένα GUID OG.

- *DELIVER (G, Aid, Msg)* : εφαρμόζεται στα εισερχόμενα μηνύματα που προορίζονται για τον τοπικό κόμβο. Είναι ασύγχρονη κλήση. Η εφαρμογή μπορεί να παράγει περαιτέρω γεγονότα καλώντας τη ROUTE( ).

- *FORWARD (G, Aid, Msg)* : εφαρμόζεται σε εισερχόμενα μηνύματα που ενεργοποιούν τα upcalls. Η εφαρμογή πρέπει να καλέσει τη ROUTE( ), για να προωθηθεί και άλλο το μήνυμα.
- *ROUTE (G, Aid, Msg, NextHopNode)* : καλείται από τον χειριστή εφαρμογής, για να προωθήσει ένα μήνυμα στον NextHopNode.

Επιπλέον interfaces δίνουν πρόσβαση στον πίνακα δρομολόγησης και στη βάση δεδομένων των δεικτών σε αντικείμενα. Όταν φτάνει ένα μήνυμα που ενεργοποιεί τα upcalls, το Tapestry στέλνει το μήνυμα στην εφαρμογή μέσω του FORWARD(). Ο χειριστής είναι υπεύθυνος για την κλήση του ROUTE() με τον τελικό προορισμό. Στο τέλος, το Tapestry καλεί την DELIVER() σε μηνύματα που προορίζονται για τον τοπικό κόμβο και έτσι τελειώνει η δρομολόγηση.

Αυτό το upcall interface παρέχει επαρκή λειτουργικότητα, για να υλοποιηθεί – επί παραδείγματι- το Bayeux σύστημα multicast. Τα μηνύματα μαρκάρονται, έτσι ώστε να πυροδοτούν upcalls σε κάθε βήμα και το Tapestry να «προκαλεί» την FORWARD() κλήση σε κάθε μήνυμα. Ο χειριστής Bayeux ελέγχει στη συνέχεια τη λίστα μελών, την ταξινομεί σε ομάδες και προωθεί ένα αντίγραφο του μηνύματος σε κάθε εξερχόμενη εγγραφή.

1.2.5.7 Tapestry και εφαρμογές

Έχοντας εξετάσει την υλοποίηση και τη συμπεριφορά του Tapestry, καταλήγουμε στο ότι το Tapestry παρέχει ένα σταθερό interface υπό μια ποικιλία συνθηκών δικτύου. Μένει να δούμε πως το Tapestry μπορεί να σταθεί απέναντι στις προκλήσεις που αντιμετωπίζουν οι μεγάλης κλίμακας εφαρμογές.

Με την αυξανόμενη χρησιμοποίηση του Διαδικτύου, οι μηχανικοί εφαρμογών έχουν αρχίσει να εστιάζουν σε εφαρμογές μεγάλης κλίμακας που εκμεταλλεύονται κοινούς πόρους του δικτύου. Παραδείγματα αποτελούν το multicast επιπέδου εφαρμογής, τα συστήματα αποθήκευσης μεγάλης κλίμακας, και τα συστήματα ανακατεύθυνσης κίνησης για ανθεκτικότητα και ασφάλεια. Αυτές οι εφαρμογές μοιράζονται νέες προκλήσεις, όταν πλέον μιλάμε για μεγάλες περιοχές εφαρμογής: θα είναι πιο δύσκολο για τους χρήστες να εντοπίζουν κοντινούς πόρους, καθώς μεγαλώνει το δίκτυο, και η εξάρτηση από περισσότερα κατανεμημένα επιμέρους τμήματα σημαίνει μικρότερος μέσος χρόνος μεταξύ αποτυχιών (mean time between failures – MTBF) για το σύστημα. Επί παραδείγματι, ένας χρήστης ενός συστήματος ανταλλαγής αρχείων θέλει να εντοπίσει και να ανακτήσει ένα κοντινό αντίγραφο ενός αρχείου αποφεύγοντας αποτυχίες εξυπηρετητή ή δικτύου.

Η ασφάλεια είναι επίσης ένα σημαντικό θέμα. Η επίθεση Sybil είναι μια επίθεση όπου ένας χρήστης παίρνει έναν μεγάλο αριθμό από IDs, για να αυξήσει τις επιθέσεις συνωμοσίας (collusion attacks). Το Tapestry αντιμετωπίζει την παραπάνω επίθεση χρησιμοποιώντας μία έμπιστη υποδομή δημόσιου κλειδιού (public-key infrastructure–PKI) για την ανάθεση των nodeIDs. Για να μειωθεί η ζημιά από ελεγχόμενους κόμβους, οι κόμβοι του Tapestry μπορούν να δουλεύουν σε ζευγάρια δρομολογώντας μηνύματα μεταξύ τους μέσω γειτόνων

και επαληθεύοντας στη συνέχεια τη διαδρομή που ακολουθήθηκε. Τέλος, το Tapestry υποστηρίζει τη χρήση MACs, για να διατηρήσει την ακεραιότητα της υπερκείμενης κίνησης.

Το Tapestry, εκτός του ότι υποστηρίζει αποδοτική δρομολόγηση των μηνυμάτων σε ονομαζόμενα αντικείμενα ή τελικά σημεία στο δίκτυο, αυξομειώνεται λογαριθμικά με το μέγεθος του δικτύου σε πληροφορία δρομολόγησης ανά κόμβο και σε αναμενόμενο αριθμό υπερκείμενων βημάτων σε μία διαδρομή. Επιπλέον, εμφανίζει ανθεκτικότητα σε αποτυχίες εξυπηρετητών και αποτυχίες δικτύου επιτρέποντας στα μηνύματα να δρομολογούνται γύρω τους σε εφεδρικές διαδρομές. Οι εφαρμογές μπορούν να πετύχουν πρόσθετη αντοχή αντιγράφοντας δεδομένα σε πολλαπλούς εξυπηρετητές και περιμένοντας από το Tapestry να κατευθύνει αιτήσεις πελατών σε κοντινά αντίγραφα.

Μια ποικιλία από διαφορετικές εφαρμογές έχουν σχεδιαστεί, υλοποιηθεί και λειτουργήσει πάνω στο Tapestry. Το OceanStore είναι μία μεγάλης κλίμακας υπηρεσία αποθήκευσης υψηλής διαθεσιμότητας, η οποία έχει δοκιμαστεί στο PlanetLab. Οι εξυπηρετητές OceanStore χρησιμοποιούν το Tapestry, για να διασκορπίσουν αποδοτικά κωδικοποιημένα τμήματα αρχείων (blocks). Οι πελάτες μπορούν να εντοπίσουν γρήγορα και να ανακτήσουν κοντινά τμήματα αρχείων από το ID τους ανεξάρτητα από αποτυχίες εξυπηρετητών ή αποτυχίες δικτύου. Άλλες εφαρμογές είναι το Mnemosyne, ένα σύστημα αρχείων, το Bayeux, ένα αποδοτικό αυτο-διοργανούμενο σύστημα multicast επιπέδου εφαρμογής, και το SpamWatch, ένα αποκεντρωμένο σύστημα που φιλτράρει το "spamming" και χρησιμοποιεί μία μηχανή αναζήτησης ομοιότητας υλοποιημένη στο Tapestry.

1.2.5.8 Συμπερασματικά

Η αρχιτεκτονική εντοπισμού και δρομολόγησης του Tapestry είναι μία αυτο- διοργανούμενη, κλιμακούμενη σε μέγεθος και εύρωστη υποδομή ευρείας κλίμακας που δρομολογεί αποτελεσματικά αιτήσεις περιεχομένου υπό την παρουσία υψηλού φορτίου και λαθών δικτύου ή κόμβων. Ένα υπερκείμενο δίκτυο Tapestry μπορεί να δομηθεί αποτελεσματικά, για να υποστηρίξει δυναμικά δίκτυα χρησιμοποιώντας κατανεμημένους αλγορίθμους. Ενώ το Tapestry είναι παρόμοιο με την κατανεμημένη τεχνική αναζήτησης Plaxton, έχει πρόσθετους μηχανισμούς που εκμεταλλεύονται την soft-state πληροφορία και παρέχουν αυτόματη οργάνωση, ευρωστία, κλιμάκωση σε μέγεθος, δυναμική προσαρμογή, και ικανοποιητική υποβάθμιση/ μείωση της απόδοσης παρουσία αποτυχιών και υψηλού φορτίου.

Το Tapestry αποτελεί την πλέον κατάλληλη λύση για δυναμικά, ευρείας κλίμακας συστήματα ονοματοδοσίας αντικειμένου και δρομολόγησης μηνυμάτων, όταν αυτά τα συστήματα πρέπει να παραδώσουν μηνύματα στο πλησιέστερο αντίγραφο των αντικειμένων ή των υπηρεσιών με έναν τρόπο ανεξάρτητο θέσης, χρησιμοποιώντας μόνο τις από σημείο σε σημείο συνδέσεις και χωρίς συγκεντρωμένες υπηρεσίες. Το Tapestry το πετυχαίνει αυτό χρησιμοποιώντας την τυχαιότητα, για να επιτύχει και την κατανομή φορτίου και την τοπικότητα της δρομολόγησης.

ΚΕΦΑΛΑΙΟ 2 : Περιγραφή αρχιτεκτονικής

2.1 Στόχος της διπλωματικής

Σκοπός αυτής της διπλωματικής εργασίας είναι η βελτίωση του αλγορίθμου επιλογής γειτονικών κόμβων για τον πίνακα δρομολόγησης του Pastry έτσι ώστε να λαμβάνονται υπόψη περισσότεροι παράμετροι της απόδοσης ενός κόμβου.

Στο προηγούμενο κεφάλαιο παρουσιάστηκε ο αρχικός αλγόριθμος επιλογής γειτόνων του Pastry. Η ιδέα που εξετάζεται σε αυτή την εργασία είναι αυτή της επιλογής του καλύτερου γείτονα για μια θέση στον πίνακα δρομολόγησης βάσει του διαθέσιμου εύρους ζώνης από-άκρη-σε-άκρη.

Έτσι, ένας κόμβος θα συμπεριλαμβάνεται στον πίνακα δρομολόγησης ενός άλλου αν και μόνο αν το διαθέσιμο εύρος ζώνης του μονοπατιού μεταξύ του τοπικού κόμβου και του υποψήφιου είναι μεγαλύτερο από αυτό των άλλων κόμβων που συμμετέχουν στο δίκτυο και που "ανταγωνίζονται" για την ίδια θέση στον πίνακα δρομολόγησης. Χρήσιμο θα ήταν εδώ να αναφέρουμε ότι δεν εξετάζονται κάθε φορά όλοι οι κόμβοι του δικτύου αλλά αυτό γίνεται σταδιακά με το δίκτυο να τείνει προς μία σταθερή κατάσταση κάνοντας χρήση επιδημικών αλγορίθμων. Οι αλγόριθμοι αυτοί περιγράφονται με μεγαλύτερη λεπτομέρεια στη συνέχεια.

Το σύστημα Pastry που χρησιμοποιήθηκε είναι το Bamboo, μία open source υλοποίηση του Pastry σε κώδικα Java. Το Bamboo, όπως θα περιγράψουμε λεπτομερέστερα και παρακάτω, είναι ένα DHT που *βασίζεται* στο Pastry αλλά περιλαμβάνει τροποποιήσεις των αρχικών αλγορίθμων με σκοπό την καλύτερη απόδοση σε συχνές και ταυτόχρονες εισαγωγές κι αποχωρήσεις κόμβων *(churn).* Για την επιλογή γειτόνων για τον πίνακα δρομολόγησης, το Bamboo κάνει χρήση του rtt (round trip time).

Για τη μέτρηση του εύρους ζώνης μεταξύ των κόμβων τού δικτύου έγινε χρήση του Spruce, ενός open source εργαλείου που αναπτύχθηκε στο MIT. Στο εργαλείο αυτό καταλήξαμε μετά από δοκιμή άλλου ενός το οποίο, για λόγους που θα αναφερθούν στη συνέχεια, δεν προτιμήθηκε. Τα δύο εργαλεία που δοκιμάστηκαν είναι οι βασικότεροι εκπρόσωποι των δύο επικρατέστερων μεθοδολογιών στην από-άκρο-σε-άκρο μέτρηση του διαθέσιμου εύρους ζώνης μεταξύ δύο κόμβων.

Τέλος, η αξιολόγηση της παραπάνω υλοποίησης πραγματοποιήθηκε στο Planetlab, ένα ερευνητικό δίκτυο με κόμβους σε όλο τον κόσμο το οποίο επιτρέπει μια όσο το δυνατό ρεαλιστικότερη αξιολόγηση κατανεμημένων και γενικότερα δικτυακών εφαρμογών.

2.2 Bamboo

2.2.1 Γενικά

Το Bamboo είναι ένας κατανεμημένος πίνακας κατακερματισμού (*DHT – Distributed Hash Table*) που στοχεύει κυρίως σε ένα σύστημα ανθεκτικό σε μεγάλα επίπεδα "churn". Churn είναι η συνεχής διαδικασία ταυτόχρονης εισαγωγής και αποχώρησης κόμβων από το δίκτυο. Το κύριο πλεονέκτημα του Bamboo σε σύγκριση με άλλα δομημένα συστήματα peer-to-peer, όπως το Chord και το Pastry, είναι ότι δεν καταρρέει σε υψηλά επίπεδα churn.

Το Bamboo βασίζεται στο Pastry, αλλά τροποποιεί ελαφρά τα πρωτόκολλα και τους αλγορίθμους του ώστε να είναι σε θέση να κάνει αποδοτικότερη χρήση του διαθέσιμου εύρους ζώνης του δικτύου. Έτσι, καταφέρνει κι αντιμετωπίζει καλύτερα μεγάλες ή συνεχείς αλλαγές μελών ειδικά σε περιβάλλοντα περιορισμένου εύρους δικτύου. Κατά τα άλλα χρησιμοποιεί ίδιους μηχανισμούς με το Pastry, το οποίο και περιγράψαμε σε προηγούμενη παράγραφο. [15, 23]

2.2.2 Γεωμετρία δικτύου και πληροφορία ανά κόμβο

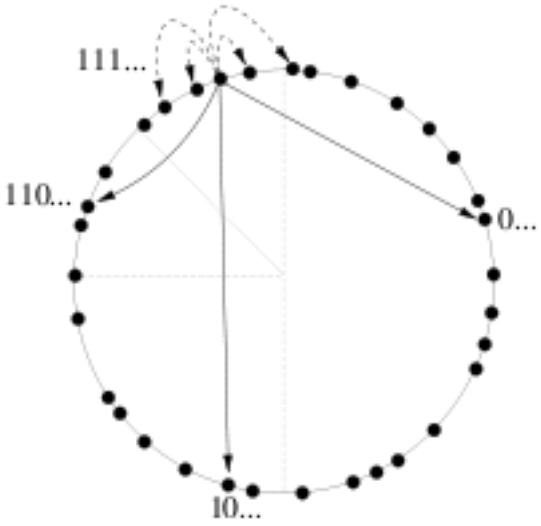
Τα DHT είναι δομημένοι γράφοι και με τον όρο *γεωμετρία* εννοούμε το πρότυπο βάσει του οποίου οι κόμβοι ενός τέτοιου δικτύου ενώνονται στο υπερκείμενο δίκτυο, ανεξάρτητα από τους αλγόριθμους δρομολόγησης και διαχείρισης που το δίκτυο χρησιμοποιεί.

Η γεωμετρία του Bamboo είναι παρόμοια με αυτήν του Pastry. Σε κάθε κόμβο του Bamboo ανατίθεται ένα αριθμητικό αναγνωριστικό (guid) μεγέθους 160 bits από την περιοχή $[0,2^{160})$. Το αναγνωριστικό αυτό μπορεί να προέρχεται είτε από τον κατακερματισμό, με βάσει τη συνάρτηση SHA-1, της IP διεύθυνσης και της πόρτας στην οποία ο κόμβος λαμβάνει τα διάφορα μηνύματα ή από τον SHA-1 κατακερματισμό ενός δημόσιου κλειδιού. Κατά αυτόν τον τρόπο τα αναγνωριστικά κατανέμονται ομοιόμορφα σε όλο το χώρο των αναγνωριστικών.

Κάθε κόμβος στο δίκτυο Bamboo διατηρεί ένα σύνολο φύλλων L (leafset) μεγέθους 2κ. Το σύνολο αυτό αποτελείται από τους k κόμβους που προηγούνται και τους k που έπονται στο κυκλικό διάστημα αναγνωριστικών. Το συνηθέστερο μέγεθος leafset για έναν κόμβο είναι 16 κι άρα k=8. Με L_i , όπου $-k \leq i \leq k$, συμβολίζονται τα μέλη του L και L_0 είναι ο ίδιος ο κόμβος. Το σύνολο αυτό παρέχει στον κόμβο αυξημένη στατική προσαρμοστικότητα στις μεταβολές των μελών του δικτύου *(static resilience)*. Με τον όρο αυτό εννοούμε την ικανότητα δρομολόγησης μηνυμάτων ακόμα και μετά την ταυτόχρονη αποτυχία κόμβων του δικτύου και πριν την λήψη μέτρων από τον μηχανισμό επανόρθωσης του δικτύου. Έχει αποδειχτεί ότι με ένα σύνολο φύλλων 16 κόμβων ακόμα και εάν αποτύχει ένα τυχαίο 30% των συνδέσεων, υπάρχουν ακόμα διαδρομές μεταξύ όλων των ζευγαριών κόμβων. Τέτοια στατική ανθεκτικότητα είναι σαφώς χρήσιμη στο

χειρισμό των αποτυχιών γενικότερα και του churn ειδικότερα, και για αυτό επιλέχθηκε η γεωμετρία του Pastry για χρήση στο Bamboo.

Εκτός από το σύνολο φύλλων, κάθε κόμβος Bamboo διατηρεί έναν πίνακα δρομολόγησης (routing table). Θεωρώντας κάθε αναγνωριστικό ως ακολουθία ψηφίων βάσης 2^b και συμβολίζοντας την εγγραφή του πίνακα δρομολόγησης στη σειρά l και τη στήλη i με $R_l[i]$, ένας κόμβος επιλέγει τους γείτονές του έτσι ώστε η εγγραφή $R_l[i]$ να είναι ένας κόμβος, του οποίου το αναγνωριστικό ταιριάζει με το δικό του ακριβώς στα πρώτα l ψηφία και του οποίου το (l+1)-ιοστό ψηφίο είναι ίσο με i. Από την παραπάνω περιγραφή, καθίσταται σαφές ότι για κάθε θέση στον πίνακα δρομολόγησης ενός κόμβου ανταγωνίζονται περισσότεροι του ενός κόμβοι. Έτσι, το Bamboo επιλέγει ως γείτονα για τον πίνακα δρομολόγησης του τον "πλησιέστερο" γείτονα μεταξύ αυτών που ταιριάζουν σε κάθε θέση. Ως κριτήριο της εγγύτητας ενός κόμβου, το Bamboo χρησιμοποιεί την καθυστέρηση δικτύου (rtt).



Εικόνα 15 : Με διακεκομμένη γραμμή φαίνεται το leafset του

Όπως ειπώθηκε και νωρίτερα, για την εισαγωγή κόμβων στον πίνακα δρομολόγησης ενός κόμβου, δεν εξετάζονται κάθε φορά όλοι οι κόμβοι του δικτύου αλλά η σύγκλιση προς μια σταθερή κατάσταση επέρχεται σταδιακά, μέσω της δράσης επιδημικών αλγορίθμων. Για την εύρεση νέων κοντινότερων γειτόνων για την ανανέωση του πίνακα δρομολόγησης, κάθε κόμβος του Bamboo εκτελεί περιοδικά την παρακάτω διαδικασία : επιλέγει τυχαία έναν από τους γείτονές του και του ζητάει να του στείλει τον πίνακα δρομολόγησης του. Στη συνέχεια, μετράει το rtt προς τους κόμβους του γειτονικού πίνακα δρομολόγησης που δεν περιέχονται στον πίνακα δρομολόγησής του κι σε περίπτωση που διαπιστώσει ότι κάποιος είναι πιο κοντά (βάσει του rtt) από τη δική του εγγραφή για τη συγκεκριμένη θέση, τότε τον εισάγει στον πίνακα δρομολόγησής αντικαθιστώντας τον ήδη υπάρχοντα κόμβο. Επιπλέον, για να καλύψει κενές θέσεις ("τρύπες") που μπορεί να υπάρχουν στον πίνακα δρομολόγησης, κάθε κόμβος εκτελεί περιοδικά αναζητήσεις για τυχαία αναγνωριστικά που όμως πληρούν τις

προϋποθέσεις ώστε ο κόμβος που θα επιστραφεί από την αναζήτηση να ταιριάζει στην κενή θέση βάσει του αλγορίθμου του μακρύτερου κοινού προθέματος. Έτσι, γεμίζει τον πίνακα δρομολόγησης εξασφαλίζοντας ότι κάθε αναζήτηση θα θέλει O(logN) βήματα για να ολοκληρωθεί. [15, 23]

2.2.3 Δρομολόγηση

Η βασική λειτουργία ενός DHT είναι η συνεπής αντιστοίχηση αναγνωριστικών σε κόμβους του δικτύου, μια διαδικασία που ονομάζεται *αναζήτηση (lookup)* ή *δρομολόγηση (routing)*.

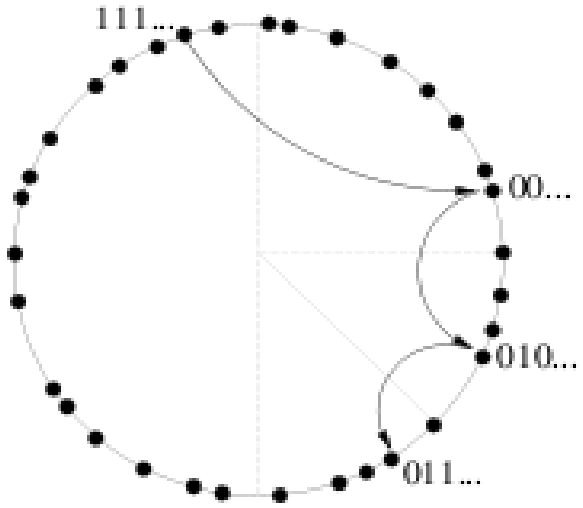
Αλγοριθμικά, η δρομολόγηση στο Bamboo εξελίσσεται βάσει μιας διαδικασίας σταδιακής ταύτισης προθέματος και εξελίσσεται με τον ακόλουθο τρόπο. Για τη δρομολόγηση ενός μηνύματος με κλειδί D, ένας κόμβος πρώτα ελέγχει εάν το D ανήκει στο σύνολο φύλλων του (όπου περιέχεται και ο εαυτός του), και σε αυτή την περίπτωση, το προωθεί στο αριθμητικά πλησιέστερο (προς το αναζητούμενο κλειδί) μέλος αυτού του συνόλου που είναι και ο κόμβος που το φιλοξενεί. Εάν αυτό το μέλος είναι ο τοπικός κόμβος, η δρομολόγηση ολοκληρώνεται. Εάν το D δεν εμπίπτει στο σύνολο φύλλων, ο τοπικός κόμβος υπολογίζει το μήκος l του μέγιστου προθέματος του αναζητούμενου αναγνωριστικού που ταυτίζεται με το δικό του. Έστω ότι D[i] είναι το i-ιοστό ψηφίο του D. Εάν η εγγραφή R_l[D[l]] δεν είναι κενή, το μήνυμα προωθείται σε αυτόν τον κόμβο. Εάν καμία από τις παραπάνω συνθήκες δεν ισχύει, το μήνυμα προωθείται στο μέλος του συνόλου φύλλων του τοπικού κόμβου που είναι αριθμητικά πλησιέστερα στο D. Όταν, τελικά βρεθεί ο κόμβος-κάτοχος του αναζητούμενου κλειδιού, αυτός αποστέλλει στον κόμβο που ξεκίνησε την αναζήτηση ένα μήνυμα με το αναγνωριστικό του και την διεύθυνση IP του. Έτσι τερματίζεται η διαδικασία αναζήτησης.

Η παραπάνω δρομολόγηση είναι αναδρομική *(recursive)* (στο βήμα n της διαδικασίας δρομολόγησης συμμετέχουν ο κόμβος n και ο n-1), αλλά είναι, φυσικά, δυνατό η δρομολόγηση να γίνει επαναληπτικά *(iteratively)*. Όπως φαίνεται και στην παρακάτω εικόνα, η επαναληπτική δρομολόγηση εμπλέκει τους ίδιους κόμβους με τη αναδρομική, με τη διαφορά ότι τώρα, συντονιστής της όλης διαδικασίας είναι ο κόμβος που ξεκινάει την αναζήτηση. Ο αρχικός κόμβος, αντί να ζητήσει από τον αμέσως επόμενό του (σύμφωνα με τη διαδικασία αναδρομικής αναζήτησης) να συνεχίσει την αναζήτηση εκ μέρους του, του ζητά απλά τη διεύθυνση του επόμενου βήματος της αναζήτησης και προωθεί ο ίδιος το ερώτημα σε αυτόν τον κόμβο. Η διαδικασία αυτή επαναλαμβάνεται μέχρι την τελική εύρεση του κλειδιού που αναζητείται.

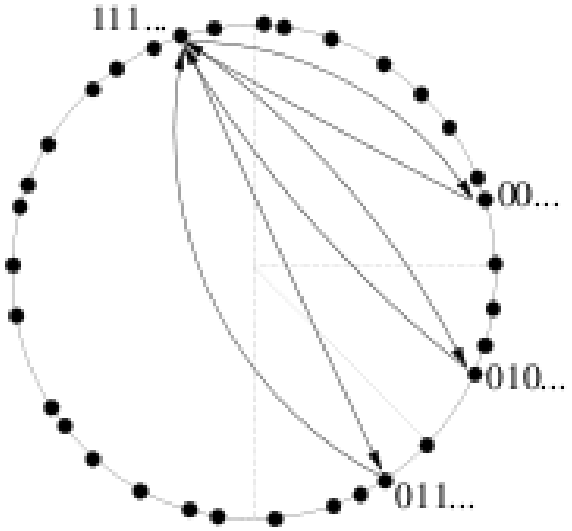
Εντούτοις, η επιλογή της αναδρομικής δρομολόγησης στο Bamboo αποδεικνύεται σημαντική για το χειρισμό του churn. Σε ένα δίκτυο με N κόμβους ένα μήνυμα μπορεί να δρομολογηθεί σε οποιονδήποτε κόμβο σε λιγότερα από $\log_{2^b}N$ βήματα κατά μέσο όρο, όπου b μία ρυθμιζόμενη παράμετρος που επηρεάζει το μέγεθος του πίνακα δρομολόγησης, και είναι συνήθως 4. Το σύνολο φύλλων επιτρέπει την πρόοδο της προώθησης (με αντάλλαγμα ενδεχομένως μεγαλύτερες διαδρομές) στην περίπτωση που ο πίνακας δρομολόγησης είναι ελλιπής.

Από το παραπάνω φαίνεται ότι, όπως είπαμε και κατά την περιγραφή άλλων αρχιτεκτονικών δομημένων δικτύων p2p, το σύνολο φύλλων εξασφαλίζει την επιτυχία της αναζήτησης κι άρα είναι απαραίτητο

για την ορθότητά της. Αντίθετα, ο κύριος λόγος ύπαρξης του πίνακα δρομολόγησης είναι η βελτίωση της απόδοσης του δικτύου. [15, 23]



Εικόνα 16 : Στο παραπάνω σχήμα παρατηρούμε την πορεία ενός μηνύματος που δρομολογείται **αναδρομικά** (recursive routing)



Εικόνα 17 : Στο παραπάνω σχήμα παρατηρούμε την πορεία ενός *μηνύματος που δρομολογείται* ***επαναληπτικά*** *(iterative routing).*

2.2.4 Bamboo και churn

Το Bamboo, όπως είπαμε και παραπάνω, παρουσιάζει ευρωστία κάτω από συνθήκες churn. Η ευρωστία αυτή μπορεί να διακριθεί σε δύο επίπεδα. Την *ορθότητα* των αναζητήσεων και την *αποδοτική* τους εκτέλεση.

Με τον όρο "στατική ανθεκτικότητα" εννοούμε τη δυνατότητα ενός δικτύου να συνεχίσει να εκτελεί αναζητήσεις σωστά, ακόμα και σε περιπτώσεις όπου παρατηρούνται ταυτόχρονες αποτυχίες κόμβων. Στο Bamboo, αυτό καθίσταται δυνατό μέσω της αποθήκευσης από κάθε κόμβο πληροφορίας σχετικής όχι μόνο με τον άμεσο γείτονά του, αλλά κι ενός πλήθους άλλων κόμβων (στον πίνακα φύλλων και τον πίνακα δρομολόγησης). Η πληροφορία αυτή του επιτρέπει να δρομολογεί μηνύματα ακόμα και μετά από ταυτόχρονη αποτυχία αρκετών γειτονικών κόμβων μέσω εναλλακτικών διαδρομών. Ο κάθε κόμβος αποκτά και διατηρεί πληροφορία σχετικά με γείτονές του μέσω διαρκούς επικοινωνίας και συντονισμού του με άλλους κόμβους του δικτύου. Για τη σύγκλιση του δικτύου σε μια σταθερή κατάσταση χωρίς τη σπάταλη χρήση του διαθέσιμου εύρους ζώνης, η επικοινωνία μεταξύ των κόμβων γίνεται με επιδημικό τρόπο, δηλαδή μέσω διαδικασιών όπου ένας κόμβος περιοδικά ζητάει(pull) πληροφορία από έναν άλλο κόμβο ή δημοσιοποιεί(push) πληροφορία σχετικά με τον ίδιο. Η παραπάνω διαδικασία αποδεικνύεται ότι οδηγεί στη σύγκλιση του δικτύου χωρίς την ανάγκη κάθε κόμβος να επικοινωνεί με όλους τους κόμβους που συμμετέχουν στο δίκτυο.

Επιπλέον, πέρα από τη δυνατότητα ορθών αναζητήσεων υπό υψηλό churn που εξασφαλίζεται μέσω της στατικής ανθεκτικότητας, το Bamboo δίνει ιδιαίτερη βαρύτητα και στην απόδοση των αναζητήσεων αυτών. Για τη βελτίωση της απόδοσης, στο Bamboo έχουν υιοθετηθεί οι παρακάτω σχεδιαστικές αποφάσεις :

- *περιοδική αντιμετώπιση των αποτυχιών κόμβων*
- *ορθό υπολογισμό των timeout για τα μηνύματα που στέλνει κάθε κόμβος*
- *προτίμηση κοντινών έναντι μακρινών κόμβων για γείτονες*

Το Bamboo ενσωματώνει τους νέους κόμβους και συνέρχεται από την αποτυχία παλιών με ένα τρόπο καθοδηγούμενο από τη συμφόρηση. Η δυναμική αποκατάσταση, όπου ένας κόμβος προσπαθεί να αντιδράσει αμέσως όταν αντιλαμβάνεται την αποτυχία ενός γείτονά του, μονάχα προσθέτει επιπλέον πίεση σε ένα ήδη πιεσμένο δίκτυο. Για να αποφύγει την κατάρρευση λόγω συμφόρησης, το Bamboo χρησιμοποιεί περιοδικούς αλγορίθμους για την αυτόματη (ανάλογα με τη συμφόρηση) κλιμάκωση των περιόδων αποκατάστασης.

Επιπλέον, σημαντικό ρόλο στην απόδοση του Bamboo παρατηρήθηκε ότι παίζει και ο ορθός καθορισμός του timeout των μηνυμάτων. Το παραπάνω είναι λογικό αν αναλογιστούμε το εξής : μια μεγάλη τιμή για timeout μηνύματος σε ένα ενδιάμεσο βήμα της διαδικασίας αναζήτησης, θα συνεπαγόταν τη μεγάλη αναμονή του κόμβου-αποστολέα πριν την αναγνώριση της αποτυχίας του κόμβου-προορισμού του μηνύματος, κι άρα τη συνολική καθυστέρηση της διαδικασίας. Αντίθετα, μια μικρή τιμή, θα οδηγούσε στη λανθασμένη υπόθεση από πλευράς αποστολέα ότι ο κόμβος παραλήπτης είναι εκτός δικτύου. Αυτό, με τη σειρά του, θα οδηγούσε σε απομάκρυνση του κόμβου από το δίκτυο, σε νέα επιβάρυνση του δικτύου λόγω του ότι ο αποστολέας θα επιδιώξει να ενημερώσει τους γείτονές του για την αλλαγή στον πίνακα δρομολόγησής του μέσω νέων μηνυμάτων και, τελικά, την άσκοπη καθυστέρηση της ίδιας της αναζήτησης γιατί ο αποστολέας θα επιδιώξει τη συνέχιση της αναζήτησης μέσω άλλου γείτονα, ο οποίος δε θα είναι και μέρος της βέλτιστης

διαδρομής. Ένας συνδυασμός ενεργού εξέτασης των κόμβων και αναδρομικής δρομολόγησης επιτρέπει στο Bamboo να κάνει αποτελεσματικά αυτή τη διάκριση. Έτσι, για τον υπολογισμό του timeout, κάθε κόμβος λαμβάνει υπόψιν το rtt του κόμβου προορισμού (το οποίο και μετράει περιοδικά) και βάσει αυτού υπολογίζει το timeout του μηνύματος που θα στείλει.

Τέλος, όπως και στο Pastry, κάθε κόμβος του Bamboo επιλέγει τους γείτονες του βάσει μετρήσεων της εγγύτητας των υποψήφιων γειτόνων. Το Bamboo, χρησιμοποιεί ως μετρική για την εγγύτητα το rtt. Βάσει λοιπόν των παραπάνω, κάθε θέση στον πίνακα δρομολόγησης ενός κόμβου καταλαμβάνεται από τον κόμβο που ταιριάζει σε αυτή τη θέση βάσει της λογικής του μακρύτερου ταυτιζόμενου προθέματος που αναλύσαμε παραπάνω και που έχει το μικρότερο rtt από όλους τους άλλους που πληρούν την παραπάνω προϋπόθεση. [15, 23]

2.2.5 Συμπερασματικά

Η ανθεκτικότητα του Bamboo σε συνθήκες churn είναι το συνδυασμένο αποτέλεσμα διάφορων παραγόντων. Κατ' αρχάς, η στατική ανθεκτικότητα της υβριδικής γεωμετρίας του επιτρέπει στο δίκτυο να συνεχίσει μετά από μια αποτυχία έως ότου ξεκινήσει η αποκατάσταση. Αυτή η στατική ανθεκτικότητα είναι ένα απαραίτητο πρώτο βήμα για το χειρισμό των αποτυχιών γενικά και του churn ειδικότερα. Δεύτερον, η αποκατάσταση εκτελείται κατά τρόπο περιοδικό: η ίδια η διαδικασία ανάκαμψης δε φορτώνει περαιτέρω το δίκτυο, αποφεύγοντας ένα πιθανό θετικό σύστημα ανατροφοδότησης πληροφοριών στο οποίο η συμφόρηση ερμηνεύεται ως περαιτέρω αποτυχία κόμβων. Τρίτον, η χρήση της αναδρομικής δρομολόγησης από το Bamboo επιτρέπει την ενεργό εξέταση των συνδέσεων γειτόνων, η οποία επιτρέπει στη συνέχεια τον υπολογισμό των αποτελεσματικών χρόνων λήξης απάντησης για τις αιτήσεις που στέλνονται πάνω από αυτές τις συνδέσεις. Χωρίς τόσο ακριβείς πληροφορίες για τους χρόνους λήξης, μια πιθανή αποτυχία κόμβων είναι δύσκολο να διακριθεί από συμφόρηση δικτύου ή επεξεργαστή, και η δυνατότητα του DHT να προσαρμοστεί εμποδίζεται. Επιπλέον, στα δίκτυα με μεγάλη κίνηση, αναμένεται αυτή η ανθεκτικότητα στη συμφόρηση να φανεί σημαντική στο χειρισμό χαμηλότερων επιπέδων churn. [15,23]

2.3 Λεπτομέρειες Υλοποίησης του Bamboo

2.3.1 SEDA (Staged Event-Driven Architecture) και Bamboo

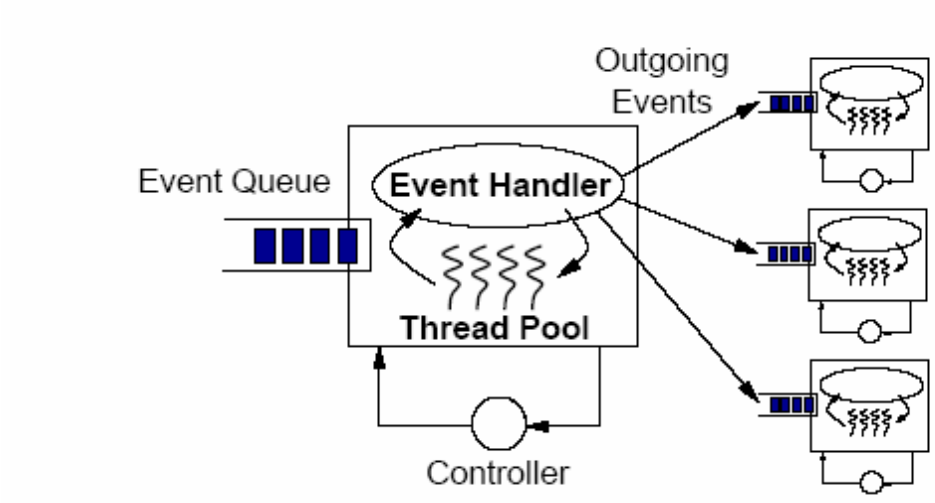
Στην παρούσα παράγραφο παρουσιάζουμε το προγραμματιστικό μοντέλο SEDA βάσει του οποίου είναι γραμμένο το Bamboo.

Το Bamboo είναι γραμμένο σύμφωνα με μία οδηγούμενη-από-γεγονότα (event-driven) μονονηματική (single-threaded) προγραμματιστική τεχνική. Για την ακρίβεια κληρονομεί τη δομή της αρχιτεκτονικής SEDA

(Staged Event-Driven Architecture). Όπως φαίνεται και από το όνομα, κάθε νέα εφαρμογή είναι ένα νέο στάδιο (stage). Η επικοινωνία γίνεται με το πέρασμα γεγονότων (events) σε κάθε στάδιο, ενώ όλη η εφαρμογή παραμένει μονονηματική (single-threaded) και σειριακή.

Η αρχιτεκτονική SEDA έχει σχεδιαστεί, για να υποστηρίζει την εξυπηρέτηση πολλαπλών ταυτόχρονων αιτήσεων και να απλουστεύσει τη δόμηση υπηρεσιών Διαδικτύου που απαιτούν υψηλό συγχρονισμό. Συνδυάζει στοιχεία του πολυνηματικού (multi-threaded) προγραμματισμού και του οδηγούμενου–από-γεγονότα προγραμματι-σμού. Στο SEDA οι εφαρμογές αποτελούνται από ένα δίκτυο σταδίων (stages) που οδηγούνται από γεγονότα και συνδέονται μεταξύ τους με ουρές (queues). Συγκεκριμένα, το στάδιο αποτελεί τη βασική μονάδα επεξεργασίας στο SEDA και είναι ένα ανεξάρτητο επιμέρους κομμάτι της εφαρμογής (βλ. παρακάτω εικόνα). Αυτή η αρχιτεκτονική επιτρέπει στις εφαρμογές να μπορούν να αντεπεξέλθουν στο φόρτο εμποδίζοντας την υπεραπασχόληση των πόρων, όταν η ζήτηση ξεπερνά την ικανότητα εξυπηρέτησης.

Το SEDA χρησιμοποιεί ένα σύνολο από δυναμικούς ελεγκτές πόρων (dynamic resource controllers), έτσι ώστε τα στάδια να παραμένουν στην κανονική περιοχή λειτουργίας τους ανεξάρτητα από τις διακυμάνσεις στο φόρτο. [12]



Εικόνα 18 : Ένα στάδιο SEDA αποτελείται από μία εισερχόμενη ουρά γεγονότων (event queue), μία δεξαμενή νημάτων (thread pool) και έναν χειριστή γεγονότων (event handler) που παρέχεται από την εφαρμογή. Τη λειτουργία του σταδίου διαχειρίζεται ο ελεγκτής (controller), ο οποίος προσαρμόζει δυναμικά την ανάθεση των πόρων και τη δρομολόγηση.

2.4 Μέτρηση Εύρους ζώνης

2.4.1 Εργαλεία μέτρησης τού εύρους ζώνης

Το διαθέσιμο εύρος ζώνης ορίζεται ως η αχρησιμοποίητη χωρητικότητα ενός μονοπατιού στο διαδίκτυο. Δεδομένου ότι ανά πάσα στιγμή, ένας σύνδεσμος μεταξύ δύο δικτυακών τοποθεσιών είτε μεταδίδει

στο μέγιστο ρυθμό που μπορεί, είτε είναι ανενεργός, μπορούμε να ορίσουμε το διαθέσιμο εύρος ζώνης ως το χρονικό μέσο όρο του αχρησιμοποίητου εύρους ζώνης για μια χρονική περίοδο T. Έτσι, προκύπτει:

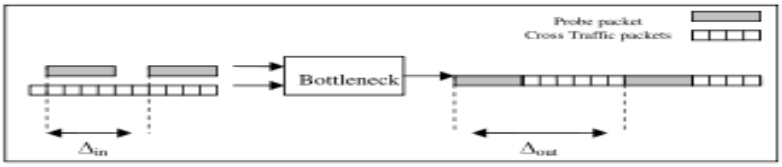
$$A_i(t,T) = \frac{1}{T}\int_t^{T+t} (C_i - \lambda_i(t))\, dt,$$

όπου Ai(t,T) είναι το διαθέσιμο εύρος ζώνης στο σύνδεσμο i τη στιγμή t, Ci είναι η χωρητικότητα του δεσμού και λi είναι η κίνησή του. Το διαθέσιμο εύρος ζώνης όλου του μονοπατιού είναι το ελάχιστο διαθέσιμο εύρος ζώνης όλων των συνδέσμων που το αποτελούν.

Όλα τα σύγχρονα εργαλεία μέτρησης του εύρους ζώνης μπορούν να χωριστούν σε δύο κατηγορίες σύμφωνα με την τεχνική που ακολουθούν.

- Το PGM (Probe Gap Model) εκμεταλλεύεται την πληροφορία που του παρέχεται από τη χρονική διαφορά μεταξύ των αφίξεων δύο διαδοχικών δοκιμαστικών πακέτων στον παραλήπτη. Ένα δοκιμαστικό ζεύγος πακέτων στέλνεται με διαφορά Δ_{in} μεταξύ των πακέτων που το απαρτίζουν και φτάνει στον προορισμό του με διαφορά Δ_{out}. Υποθέτοντας ότι ο σύνδεσμος με το μικρότερο διαθέσιμο εύρος ζώνης κατά μήκος του μονοπατιού είναι μοναδικός και η ουρά του συνδέσμου αυτού δεν αδειάζει μεταξύ της αποχώρησης από αυτή του πρώτου πακέτου και της άφιξης του δεύτερου, τότε το Δ_{out} είναι ο χρόνος που απαιτείται για να στείλει ο σύνδεσμος με το μικρότερο διαθέσιμο εύρος ζώνης ποσότητα δεδομένων ίση με το δεύτερο πακέτο και τα δεδομένα που έφτασαν στην ουρά του κατά το Δ_{in} όπως φαίνεται και στο παρακάτω σχήμα. Έτσι, το διάστημα που απαιτείται για την αποστολή των δεδομένων που έφτασαν στην ουρά του συνδέσμου είναι Δ_{out}-Δ_{in}, και ο ρυθμός αποστολής των δεδομένων αυτών είναι Δ_{out}-Δ_{in}/Δ_{in} x C, όπου C είναι η χωρητικότητα του παραπάνω συνδέσμου. Το διαθέσιμο εύρος ζώνης στην παραπάνω περίπτωση δίνεται από τον τύπο :

$$A = C \times \left(1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}}\right).$$



Εικόνα 19 :Στο παραπάνω σχήμα φαίνεται το μοντέλο PGM για τη μέτρηση *του διαθέσιμου εύρους ζώνης.*

- Το PRM (Probe Rate Model) βασίζεται στην αυτο-συμφόρηση του μονοπατιού που συνδέει δύο τοποθεσίες. Μια διαισθητική περιγραφή του παραπάνω αλγορίθμου είναι η ακόλουθη : αν ένα κόμβος στέλνει δεδομένα προς έναν άλλο με ρυθμό μικρότερο από το διαθέσιμο εύρος ζώνης, τότε ο ρυθμός άφιξης των δεδομένων αυτών στον παραλήπτη θα είναι ίδιος με το ρυθμό αποστολής τους. Αντίθετα, αν ο ρυθμός

αποστολής είναι μεγαλύτερος από το διαθέσιμο εύρος ζώνης, τότε θα δημιουργηθούν ουρές αναμονής κατά μήκος του μονοπατιού, με συνέπεια την καθυστέρηση στην άφιξη των απεσταλμένων δεδομένων. Σαν αποτέλεσμα, ο ρυθμός άφιξης στον παραλήπτη θα είναι μικρότερος από το ρυθμό αποστολής τους. Έτσι, για τη μέτρηση του διαθέσιμου εύρους ζώνης αρκεί ο όσο το δυνατό ακριβέστερος προσδιορισμός της τιμής του ρυθμού που προκαλεί τη δημιουργία ουρών κατά μήκος του μονοπατιού. Αυτός ο συγχρονισμός επιτυγχάνεται με την ανταλλαγή μηνυμάτων μεταξύ του τμήματος που τρέχει στο μετρούμενο κόμβο και αυτού που τρέχει στον κόμβο που μετράει.

Όλα τα εργαλεία, ανεξάρτητα από το ποια από τις παραπάνω τεχνικές χρησιμοποιούν, για την εξαγωγή ασφαλών μετρήσεων αποφεύγοντας αλλοιώσεις που μπορούν να προκληθούν από αστάθμητους παράγοντες, επαναλαμβάνουν πολλές φορές την παραπάνω διαδικασία πριν την παρουσίαση της τελικής μέτρησης.
Και οι δύο προσεγγίσεις, PGM και PRM, υποθέτουν :

1. *FIFO ουρές αναμονής κατά μήκος του μονοπατιού.*
2. *η δικτυακή κίνηση πέρα από αυτή που δημιουργείται από το εκάστοτε εργαλείο για τις ανάγκες της μέτρησης, ακολουθεί ένα μοντέλο ρευστών.*
3. *οι μέσοι ρυθμοί της παραπάνω δικτυακής κυκλοφορίας αλλάζουν αργά και μπορεί να θεωρηθεί ότι παραμένουν σταθεροί κατά τη διάρκεια μιας μέτρησης.*

Επιπλέον, το PGM υποθέτει μοναδικό σύνδεσμο με το μικρότερο εύρος ζώνης και μάλιστα, θεωρεί ότι ο σύνδεσμος αυτός ταυτίζεται και με το σύνδεσμο με τη μικρότερη χωρητικότητα. Οι παραπάνω υποθέσεις είναι αναγκαίες για την ανάλυση των μοντέλων αλλά αποδεικνύεται πειραματικά πως τα εργαλεία παραμένουν αξιόπιστα ακόμα κι αν κάποιες από τις παραπάνω υποθέσεις δεν ισχύουν.

Τα δύο εργαλεία που δοκιμάστηκαν είναι το Pathload και το Spruce και είναι εκπρόσωποι των παραπάνω δύο μεθοδολογιών. Το Pathload υλοποιεί το PRM ενώ το Spruce το PGM μοντέλο. Για τις ανάγκες των μετρήσεων ενσωματώσαμε το κάθε εργαλείο στο Bamboo DHT και το ανεβάσαμε σε 12 κόμβους. Από την έξοδο που συλλέξαμε βγάλαμε κάποια συμπεράσματα τα οποία και παρουσιάζουμε στη συνέχεια.

Αρχικά δοκιμάσαμε το Pathload και παρατηρήσαμε τα εξής αρνητικά χαρακτηριστικά, τα οποία και μας έκαναν να μην το προτιμήσουμε :

- Αρχικά παρατηρήσαμε ότι κάθε μέτρηση χρειαζόταν κατά μέσο όρο 40 με 50 sec (με κάποιες μετρήσεις να ξεπερνούν τα 2 λεπτά), κάτι μη αποδεκτό για τις δικές μας ανάγκες. Επιπλέον, μία τέτοια καθυστέρηση πιθανά θα επηρέαζε και την ακρίβεια της μέτρησης αφού το χρονικό διάστημα αυτό είναι αρκετό για μεγάλες αλλαγές στην κίνηση ενός συνδέσμου (κίνηση προερχόμενη από άλλες εφαρμογές).

- Επιπλέον, ως προς την ακρίβεια της μέτρησης, έχουμε να παρατηρήσουμε δύο πράγματα. Αρχικά ότι σε πολλές μετρήσεις, το άνω και το κάτω όριο της μέτρησης απείχαν αρκετά και μάλιστα, η απαίτηση για μικρότερη απόκλιση συνεπαγόταν σημαντική αύξηση στο μέσο χρόνο μέτρησης. Και δεύτερον, ότι για ένα τυχαίο μονοπάτι μεταξύ των κόμβων Α και Β, παρατηρούνταν μεγάλες αποκλίσεις στη μέτρηση του διαθέσιμου εύρους ζώνης όταν η μέτρηση γινόταν από τον Α στο Β κι όταν αυτή γινόταν από τον Β στον Α. Ακόμα κι αν οι δύο μετρήσεις γίνονταν ταυτόχρονα.

Για τους παραπάνω λόγους καταλήξαμε στη χρήση του Spruce, το οποίο και περιγράφουμε με μεγαλύτερη λεπτομέρεια παρακάτω. [17,18]

2.4.2 SPRUCE

Το Spruce (Spread PaiR Unused Capacity Estimate) είναι ένα εργαλείο για τη μέτρηση του διαθέσιμου εύρους ζώνης μεταξύ δύο κόμβων από-άκρη-σε-άκρη, υλοποιημένο σε C που βασίζεται στο μοντέλο PGM. Έτσι, αρχικά, στέλνει ζευγάρια από πακέτα στο μονοπάτι με χρονική διαφορά μεταξύ τους τέτοια ώστε το πρώτο πακέτο να μην έχει φύγει από την ουρά του συνδέσμου με το μικρότερο εύρος ζώνης του μονοπατιού πριν το δεύτερο φτάσει σε αυτή. Στη συνέχεια, υπολογίζει τα bytes που έχουν φτάσει στην ουρά μεταξύ των δύο πακέτων, από την απόσταση που παρατηρεί μεταξύ τους στον παραλήπτη. Τέλος, το Spruce υπολογίζει το διαθέσιμο εύρος ζώνης ως τη διαφορά μεταξύ της χωρητικότητας του μονοπατιού και του ρυθμού άφιξης στο σύνδεσμο με το μικρότερο εύρος ζώνης.

Το Spruce υπολογίζει το εύρος ζώνης βάσει της εξίσωσης $(\Delta_{out}-\Delta_{in})/\Delta_{in}$? C. Έτσι, απαιτεί ως όρισμα και την χωρητικότητα C του μονοπατιού, που ορίζεται ως η ελάχιστη χωρητικότητα των συνδέσμων που απαρτίζουν το μονοπάτι.

Μερικά από τα χαρακτηριστικά που ξεχωρίζουν το spruce από άλλα εργαλεία για τη μέτρηση του εύρους ζώνης είναι τα ακόλουθα:

1. Το Spruce, αντί για "τρένα" από πακέτα, στέλνει ζευγάρια πακέτων σύμφωνα με μια κατανομή Poisson. Αυτό του επιτρέπει να είναι non-intrusive and robust.
2. Επιλέγοντας προσεκτικά την τιμή τού Δin, το Spruce εξασφαλίζει ότι η ουρά αναμονής του συνδέσμου με το μικρότερο εύρος ζώνης δεν αδειάζει μεταξύ των δύο μηνυμάτων του ζεύγους πακέτων, γεγονός που αποτελεί μια από τις προϋποθέσεις για την ορθή εκτίμηση του εύρους ζώνης.
3. Το Spruce δεν κατακλύζει με μηνύματα το σύνδεσμο με το μικρότερο εύρος ζώνης κατά μήκος του μονοπατιού γιατί ο ρυθμός αποστολής του δεν είναι παρά ο ελάχιστος των 240Kb/s και 5% της χωρητικότητας του συνδέσμου αυτού.

4. Πέρα από το πλήθος των ζευγών μηνυμάτων K επί του οποίου υπολογίζεται και ο μέσος όρος για την τελική μέτρηση, το Spruce δεν έχει άλλες παραμέτρους που ρυθμίζει ο χρήστης. [17]

2.5 Περιγραφή υλοποίησης

2.5.1 Γενικά

Για τις ανάγκες της διπλωματικής τροποποιήσαμε τον αλγόριθμο δρομολόγησης του Bamboo ώστε η επιλογή των γειτόνων στον πίνακα δρομολόγησης να γίνεται βάσει του διαθέσιμου εύρους ζώνης του μονοπατιού μεταξύ των κόμβων.

Στη συνέχεια, δίνουμε μία γενική περιγραφή της αρχιτεκτονικής της δικής μας υλοποίησης και στο τέλος, περιγράφουμε λεπτομερέστερα την υλοποίησή μας και διάφορα προβλήματα που συναντήσαμε κατά την διάρκειά της.

2.5.2 Περιγραφή αρχιτεκτονικής

Για την υλοποίηση της τροποποίησης του αλγόριθμου δρομολόγησης του Bamboo, χρειάστηκε, αρχικά, η τροποποίηση του Spruce ώστε να είναι δυνατή η ταυτόχρονη εκτέλεση sender και receiver στο ίδιο μηχάνημα. Αυτό, είναι απαραίτητο ώστε να μπορεί ένα μηχάνημα να συμμετέχει σε πολλές ταυτόχρονες μετρήσεις του διαθέσιμου εύρους ζώνης του, είτε αν σε αυτό τρέχει ο sender είτε ο receiver. Για το σκοπό αυτό κάναμε και τις δύο διεργασίες του Spruce, τον sender και τον receiver, να μπορούν να δέχονται ως όρισμα τις TCP και UDP πόρτες που θα δεσμευτούν για τη διεξαγωγή της μέτρησης (αρχικά ήταν σταθερές). Επιπλέον, προσθέσαμε την επιλογή να μπορεί να τυπώνει το αποτέλεσμα σε ένα αρχείο αντί για τη οθόνη ώστε να είναι εφικτή η απόκτηση του αποτελέσματος από το Bamboo.

Στο SEDA, όπως είπαμε και παραπάνω, κάθε εφαρμογή είναι ουσιαστικά ένα στάδιο το οποίο επικοινωνεί με τα άλλα μέσω γεγονότων (events). Έτσι, για το συγχρονισμό των ενεργειών που απαιτούνται για τη μέτρηση του διαθέσιμου εύρους ζώνης με το Spruce (του Sender και του Receiver), δημιουργήσαμε δύο επιπλέον μηνύματα ή γεγονότα, βάσει της ορολογίας του SEDA, τα οποία περιγράφονται στη συνέχεια:

- *BWMeasurementStartReq* : περιέχει τα στοιχεία *(IP, port, guid)* των δύο κόμβων μεταξύ των οποίων θα γίνει η μέτρηση, τις TCP και UDP θύρες που χρησιμοποιεί ο κόμβος του Receiver για το Spruce κι έναν ακέραιο, ο οποίος θα περιγραφεί και αργότερα, που δηλώνει τον αριθμό των αποτυχημένων προσπαθειών που έχουν γίνει για τη μέτρηση του εύρους ζώνης μεταξύ των δύο κόμβων.

- *BWMeasurementResult* : περιέχει τα στοιχεία *(IP, port, guid)* των δύο κόμβων μεταξύ των οποίων θα

γίνει η μέτρηση, το αποτέλεσμα της μέτρησης, μία τιμή boolean που δηλώνει την επιτυχία ή την αποτυχία της μέτρησης και τον αριθμό των προσπαθειών που έχουν γίνει (όμοια με το παραπάνω).

Ο τρόπος με τον οποίο επιτυγχάνεται ο συγχρονισμός των διεργασιών παρουσιάζεται στο ακόλουθο παράδειγμα. Έστω ότι ένας κόμβος Α θέλει να μετρήσει το διαθέσιμο εύρος ζώνης του μονοπατιού προς έναν κόμβο Β. Ο κόμβος Α, θα ξεκινήσει τον Reciever του Spruce και θα στείλει ένα μήνυμα BWMeasurementStartReq στον Β. Αυτός με τη σειρά του, μόλις λάβει το νέο γεγονός, θα ξεκινήσει τον Sender του Spruce, με ορίσματα που προέρχονται από το μήνυμα BWMeasurementStartReq, θα περιμένει να τελειώσει η μέτρηση και θα στείλει πίσω στον Α το αποτέλεσμα σε kbps. Τέλος, ο Α θα χρησιμοποιήσει το αποτέλεσμα για να εξετάσει το αν ο Β θα μπει στον πίνακα δρομολόγησής του.

2.5.3 Λεπτομερέστερη περιγραφή υλοποίησης

Η παραπάνω περιγραφή, αν και πολύ γενική, δίνει μια εποπτική εικόνα του τρόπου που επιτυγχάνεται ο συγχρονισμός των διεργασιών για τη μέτρηση του διαθέσιμου εύρους ζώνης. Στη συνέχεια της παραγράφου παρουσιάζεται λεπτομερέστερα η υλοποίηση του τροποποιημένου Bamboo.

Κατά την περιγραφή του Bamboo, περιγράψαμε το πώς αυτό συμπληρώνει και ανανεώνει τον πίνακα δρομολόγησής του κάνοντας χρήση επιδημικών αλγορίθμων. Οι παραπάνω αλγόριθμοι στη βασική τους δομή παραμένουν αναλλοίωτοι και στο τροποποιημένο Bamboo. Οι αλλαγές αφορούν στη μετρική που χρησιμοποιείται για την εκτλιμηση της εγγύτητας των κόμβων, η οποία από το rtt γίνεται το διαθέσιμο εύρος ζώνης και στις περιόδους των διαδικασιών αναζήτησης καλύτερων γειτόνων για τον πίνακα δρομολόγησης του κάθε κόμβου.

Όσον αφορά την αλλαγή της μετρικής της εγγύτητας των γειτόνων, η αντικατάσταση του rtt από το διαθέσιμο εύρος ζώνης δεν συνεπάγεται την απομάκρυνση της περιοδικής μέτρησης του rtt, απλά την απεμπλοκή της από τη διαδικασία επιλογής γειτόνων και την αύξηση της περιόδου μέτρησης. Δεν αφαιρέσαμε εντελώς τη μέτρηση του rtt γιατί ο λόγος που το αρχικό Bamboo μέτραγε το rtt των γειτονικών του κόμβων δεν ήταν μόνο η εισαγωγή τους στον πίνακα δρομολόγησης αλλά και ο ορθός υπολογισμός του timeout των μελλοντικών μηνυμάτων που θα δρομολογούσε σε αυτούς. Μάλιστα, ο ορθός υπολογισμός του timeout αποδεικνύεται πειραματικά ότι παίζει πολύ σημαντικό ρόλο στην απόδοση του Bamboo υπό churn.

Έτσι, στο τροποποιημένο Bamboo, πέρα από την περιοδική μέτρηση του rtt, προστίθεται και η περιοδική μέτρηση του διαθέσιμου εύρους ζώνης των κόμβων του πίνακα δρομολόγησης κάθε κόμβου, η οποία πραγματοποιείται κάθε 5 λεπτά ώστε η κάθε εγγραφή να είναι έγκυρη κι επικαιροποιημένη ανά πάσα στιγμή. Στην τιμή των 5 λεπτών για την περίοδο μέτρησης του διαθέσιμου εύρους ζώνης καταλήξαμε μετά από μετρήσεις που κάναμε με το iptraf (ένα open source εργαλείο για την παρακολούθηση της κίνησης στο δίκτυο) για την αποφυγή υπερβολικής φόρτωσης του δικτύου του τοπικού κόμβου.

Για την εύρεση καλύτερων γειτόνων, όπως περιγράψαμε και νωρίτερα, το Bamboo εκτελεί περιοδικά αναζητήσεις είτε ρωτώντας γείτονές του για τους πίνακες δρομολόγησής τους είτε εκτελώντας αναζητήσεις για τυχαία κλειδιά που όμως πληρούν τις προϋποθέσεις ώστε ο κόμβος που θα επιστραφεί να είναι κατάλληλος για την θέση που θέλει να γεμίσει. Βάσει μετρήσεων που κάναμε με το iptraf, και δεδομένου ότι πλέον για την αξιολόγηση των γειτόνων χρησιμοποιείται το διαθέσιμο εύρος ζώνης, καταλήξαμε να αυξήσουμε την περίοδο που εκτελούνται οι παραπάνω αναζητήσεις. Έτσι, από 10sec που ήταν η περίοδος όπου ο κάθε κόμβος ρώταγε τους γείτονές του για τους πίνακες δρομολόγησής τους και 20 όπου εκτελούσε αναζητήσεις για τυχαία κλειδιά, τώρα οι περίοδοι αυτοί έγιναν 90 και 120sec αντίστοιχα.

Κατά την περιγραφή των δύο νέων γεγονότων που εισάγαμε (*BWMeasurementStartReq* και *BWMeasurementResult*), αναφέραμε αλλά δεν εξηγήσαμε επακριβώς το ρόλο του αριθμού των αποτυχημένων προσπαθειών που περιέχεται στα δύο νέα μηνύματα. Αυτό θα προσπαθήσουμε να κάνουμε στη συνέχεια.

Το Spruce, για την διεκπεραίωση της μέτρησης, απαιτεί την παροχή από το χρήστη (ως όρισμα) της χωρητικότητας του προς μέτρηση μονοπατιού. Η μέτρηση της χωρητικότητας, αν αυτή γινόταν κάθε φορά που θέλουμε να μετρήσουμε το διαθέσιμο εύρος ζώνης ενός κόμβου ή έστω κάθε φορά που αντικαθίσταται ένας κόμβος στον πίνακα δρομολόγησης κι ένας νέος εισέρχεται (κάτι που προϋποθέτει την αρκετά καλή ακρίβεια των εργαλείων μέτρησης της χωρητικότητας και το μη επηρεασμό της μέτρησης από την κίνηση κάθε φορά στο δίκτυο), θα προκαλούσε περαιτέρω φόρτο στο δίκτυο και θα κατέληγε, όσο το δίκτυο θα μεγάλωνε, να θέτει σοβαρούς περιορισμούς στην ικανότητά κλιμάκωσής του *(scalability)*. Μάλιστα, μετά από μια πρόχειρη δοκιμή του Pathrate , ενός εργαλείου φτιαγμένου για μέτρηση της χωρητικότητας, είδαμε ότι η μέτρηση αυτή είναι κι αρκετά χρονοβόρα.

Για τους παραπάνω λόγους, καταλήξαμε σε μία λύση που θυμίζει τη διαδικασία της δυαδικής αναζήτησης. Έτσι, στη θέση του να μετράμε τη χωρητικότητα του κάθε μονοπατιού, θέτουμε ως αρχική τιμή της χωρητικότητας όλων των μονοπατιών τα 200 Mbps και βάσει αυτής εκτελούμε το Spruce. Στη φάση αυτή ο μετρητής των αποτυχημένων προσπαθειών που περιέχεται στο μήνυμα *BWMeasurementStartReq* που αποστέλλεται στον προς μέτρηση κόμβο είναι 0 καταδεικνύοντας ότι πρόκειται για την πρώτη προσπάθεια εκτίμησης του διαθέσιμου εύρους ζώνης μεταξύ των δύο κόμβων. Αν η προσπάθεια πετύχει, τότε το αποτέλεσμα επιστρέφεται μέσω του μηνύματος *BWMeasurementResult* που αποστέλλεται πίσω στον κόμβο που ξεκίνησε τη μέτρηση. Σε περίπτωση αποτυχίας, το *BWMeasurementResult* μήνυμα πληροφορεί τον αρχικό κόμβο για την αποτυχία της μέτρησης και του επιστρέφει την τιμή του μετρητή που περιεχόταν στο αρχικό *BWMeasurementStartReq* μήνυμα. Σε απόκριση του παραπάνω και δεδομένου ότι η αποτυχία της μέτρησης δεν οφείλεται σε αποτυχία του κόμβου (το οποίο ελέγχεται από τους ήδη υπάρχοντες μηχανισμούς του Bamboo μέσω περιοδικής αποστολής μηνυμάτων τύπου ping από κάθε κόμβο στους γείτονές του), ο αρχικός κόμβος, μειώνει την τιμή της χωρητικότητας που είχε αρχικά δώσει ως όρισμα στα 150 Mbps κι εκτελεί εκ νέου το Spruce. Η παραπάνω διαδικασία συνεχίζεται με τις τιμές της χωρητικότητας να παίρνουν, σε περίπτωση

αποτυχίας, τις παρακάτω τιμές : 200, 150, 100, 50, 40, 30, 20, 10 Mbps.

Όσον αφορά στην εισαγωγή των κόμβων στο πίνακα δρομολόγησης, αυτό γίνεται, όπως έχουμε πει, βάσει του ποιος από τους γείτονες που ταιριάζουν στην κάθε θέση βάσει προθέματος έχει και το υψηλότερο διαθέσιμο εύρος ζώνης. Μάλιστα, για να λαμβάνεται υπόψιν και το σφάλμα που πιθανά κάνει το Spruce στις μετρήσεις του, ορίσαμε ότι το μετρούμενο εύρος ζώνης ενός κόμβου πρέπει να είναι μεγαλύτερο από το 110% του κόμβου που είναι ήδη στον πίνακα δρομολόγησης (υποθέτοντας έτσι σφάλμα της τάξης του 10%). Επιπλέον, όπως είπαμε και κατά την περιγραφή του Bamboo, είναι προτιμότερο για λόγους απόδοσης οι θέσεις του πίνακα δρομολόγησης να είναι γεμάτες ακόμα κι αν δεν περιέχουν τους καλύτερους δυνατούς γείτονες. Για το λόγο αυτό, σε περίπτωση που αποτύχουν και οι 8 προσπάθειες μέτρησης ενός κόμβου, αν η θέση του πίνακα δρομολόγησης για την οποία είναι υποψήφιος είναι άδεια, τότε ο κόμβος εισάγεται με μηδενικό εύρος ζώνης. Έτσι, ο τοπικός κόμβος μπορεί να δρομολογεί μηνύματα προς την κατεύθυνση του κόμβου εξασφαλίζοντας το O(logN) ενώ ταυτόχρονα, ο κόμβος αυτός θα αντικατασταθεί από τον πρώτο κόμβο που ταιριάζει στη θέση του βάσει προθέματος και έχει καλύτερης ποιότητας μονοπάτι προς τον τοπικό κόμβο.

Από τα παραπάνω βλέπουμε ότι για τη μέτρηση του εύρους ζώνης ενός κόμβου μπορούν να γίνουν έως και 8 συναπτές προσπάθειες. Κάτι τέτοιο, εύλογα εγείρει το ερώτημα του αν πραγματικά η παραπάνω τεχνική εξοικονομεί εύρος ζώνης ή αν το σπαταλάει. Για το λόγο αυτό αφήσαμε το τροποποιημένο Bamboo να τρέχει στο Planetlab και παρατηρήσαμε ότι πάνω από το 65% των μετρήσεων πετυχαίνει στην πρώτη προσπάθεια. Έτσι, με τον παραπάνω αλγόριθμο εξοικονομούμε το εύρος ζώνης που θα απαιτούνταν για τον υπολογισμό της χωρητικότητας στις παραπάνω περιπτώσεις. Τέλος, αν μία προσπάθεια αποτύχει και η επόμενη προσπάθεια που είναι να ξεκινήσει είναι για χωρητικότητα μονοπατιού μικρότερη του εύρους ζώνης του κόμβου που βρίσκεται ήδη στη θέση του πίνακα δρομολόγησης στη θέση που θα ταίριαζε ο μετρούμενος κόμβος, τότε η προσπάθεια εγκαταλείπεται. Έτσι, αποφεύγονται επιπλέον άσκοπες μετρήσεις.

Εδώ θεωρούμε σκόπιμο να τονίσουμε πως το γεγονός ότι μια μέτρηση πετυχαίνει με την πρώτη, κι άρα με όρισμα για την χωρητικότητα 200Mbps, δε συνεπάγεται ότι και η εκτίμηση για το διαθέσιμο εύρος ζώνης είναι της τάξης των 200Mbps. Παρατηρήσαμε κατά την εφαρμογή του Spruce ότι σε κάποιες περιπτώσεις, αν το όρισμα που αφορά στη χωρητικότητα του μονοπατιού είναι αρκετά μεγαλύτερο από την πραγματική τιμή του, τότε η μέτρηση με το Spruce τερματίζεται χωρίς να δίνει κάποια εκτίμηση για το διαθέσιμο εύρος ζώνης. Κάνοντας χρήση του αλγορίθμου που περιγράψαμε παραπάνω είδαμε ότι το φαινόμενο σχεδόν εξαλείφθηκε. Αιτίες για το παραπάνω φαινόμενο θεωρούμε ότι μπορούν να αναζητηθούν στην "ποιότητα" του μονοπατιού που ενώνει τους δύο κόμβους και πιο συγκεκριμένα σε χαρακτηριστικά όπως ο βαθμός απώλειας πακέτων. Ωστόσο, οι λόγοι του παραπάνω φαινομένου δε διερευνήθηκαν περαιτέρω μιας και ξεφεύγουν από τους σκοπούς της εργασίας μας.

Τέλος, από την περιγραφή της λειτουργίας του Spruce μπορούμε να δούμε ότι για την απόκτηση της μέτρησης, θα πρέπει (τουλάχιστον) ο κόμβος που εκτελεί τον sender να περιμένει για την ολοκλήρωση της μέτρησης. Κάτι τέτοιο όμως, δεδομένης και της μονονηματικής δομής του Bamboo, θα ήταν καταστρεπτικό για

την απόδοσή του, διότι κατά τη διάρκεια της μέτρησης θα έπρεπε να παγώνει ο κόμβος και να μην επεξεργάζεται κανένα μήνυμα από αυτά που πιθανά περιμένουν στις ουρές του. Για να αποφύγουμε αυτό το πρόβλημα, κάναμε τον sender να τρέχει σε ξεχωριστό thread και να ειδοποιεί το κύριο όταν ολοκληρωθεί η μέτρηση. Ο receiver τρέχει εξαρχής σε άλλο thread και ειδοποιείται από τον κόμβο που τρέχει τον sender μέσω του μηνύματος *BWMeasurementResult* για το αποτέλεσμα της μέτρησης.

ΚΕΦΑΛΑΙΟ 3 : ΠΕΙΡΑΜΑΤΙΚΗ ΑΞΙΟΛΟΓΗΣΗ ΤΟΥ ΔΙΚΤΥΟΥ

3.1 Γενικά

Στο κεφάλαιο αυτό θα παρουσιάσουμε την αξιολόγηση του τροποποιημένου Bamboo. Ένα δίκτυο ομότιμων κόμβων, όπως αυτό του Bamboo, παρουσιάζει ιδιαίτερη δυσκολία στην αξιολόγησή του λόγω της κατανεμημένης του φύσης. Η προσπάθεια εξομοίωσης της πληθώρας των προβληματικών καταστάσεων που μπορεί να βρεθεί ένα τέτοιο δίκτυο, συμπεριλαμβανομένης της τυχαιότητας που τις χαρακτηρίζει, θα αποτελούσε μία επίπονη και αμφιβόλου αποτελέσματος προσπάθεια, αφού τα αποτελέσματα στα οποία θα καταλήγαμε θα ήταν τόσο έγκυρα όσο και ο εξομοιωτής που θα χρησιμοποιούσαμε. Φαινόμενα όπως αποτυχίες κόμβων, καθυστερήσεις λόγω ουρών στους δρομολογητές, κίνηση στους κόμβους του δικτύου από άλλες εφαρμογές κ.α., είναι παράγοντες οι οποίοι δε μπορούν να εξομοιωθούν παρά μόνο από μια υποδομή η οποία είναι κι αυτή κατανεμημένη κι άρα υπόκειται στις ίδιες συνθήκες δικτύωσης με έναν πραγματικό κόμβο του δικτύου.

Για τον παραπάνω λόγο θεωρήσαμε σκόπιμο η αξιολόγηση του συστήματός μας να γίνει στο PlanetLab, μία κατανεμημένη υποδομή αξιολόγησης κατανεμημένων συστημάτων η οποία περιγράφεται παρακάτω.

3.2 Planetlab

Το PlanetLab είναι ένα ερευνητικό δίκτυο αποτελούμενο από υπολογιστές σε όλο τον κόσμο, το οποίο σχεδιάστηκε και χρησιμοποιείται ως ένα περιβάλλον δοκιμής εφαρμογών που άπτονται της δικτύωσης υπολογιστών και των κατανεμημένων συστημάτων. Φτιάχτηκε το 2002 και μέχρι τον Αύγουστο του 2008 μετρούσε 913 κόμβους σε 460 τοποθεσίες παγκοσμίως. Κάθε πείραμα που εκτελείται στο Planetlab έχει τη δική του "φέτα" (slice). Μια "φέτα" είναι, ουσιαστικά, εικονικά μηχανήματα που τρέχουν σε ένα υποσύνολο των κόμβων του Planetlab και στα οποία έχει πρόσβαση ο χρήστης. Από το παραπάνω καθίσταται σαφές ότι πολλές "φέτες", κι άρα πολλά πειράματα, μπορούν να τρέχουν παράλληλα σε ένα μηχάνημα.

Πρόσβαση επιτρέπεται σε άτομα που σχετίζονται με πανεπιστήμια ή εταιρείες που φιλοξενούν κόμβους του Planetlab. Ωστόσο, υπάρχουν μερικές δημόσιες και δωρεάν υπηρεσίες που παρέχονται από το Planetlab.

Για τον εύκολο χειρισμό των slices, έχουν αναπτυχθεί εργαλεία από μέλη της κοινότητας του Planetlab τα οποία παρέχονται στα υπόλοιπα μέλη.

3.3 Μεθοδολογία ελέγχου

Για τις ανάγκες τής αξιολόγησης, εγκαταστήσαμε το τροποποιημένο Bamboo και το αρχικό Bamboo σε 75 κόμβους και τα αφήσαμε να τρέχουν παράλληλα για διάστημα 8 ημερών. Όπως αναφέραμε και πριν, σε κάθε κόμβο του PlanetLab μπορούν να φιλοξενούνται πολλές slices, κι άρα να τρέχουν διάφορα πειράματα. Το γεγονός αυτό εξασφαλίζει τον τυχαίο και μεταβαλλόμενο φόρτο κάθε μηχανήματος, ο οποίος θα είναι ανάλογος των πειραμάτων που εκτελούνται ανά πάσα στιγμή σε αυτό και, φυσικά, των απαιτήσεων καθενός πειράματος. Επιπλέον, λόγω της κατανεμημένης φύσης του PlanetLab, οι κόμβοι μας υπόκεινται στους ίδιους μηχανισμούς δρομολόγησης (κι άρα και στους ίδιους παράγοντες αποτυχίας και καθυστέρησης) με έναν οποιοδήποτε τελικό χρήστη του Bamboo. Τα παραπάνω μας εξασφαλίζουν την εγκυρότητα των αποτελεσμάτων μας. Έτσι, οι κόμβοι του PlanetLab αποτελούν ένα ρεαλιστικό "δοκιμαστικό σωλήνα" για το δίκτυό μας.

Για την εγκατάσταση και το χειρισμό των απομακρυσμένων κόμβων του PlanetLab απαιτήθηκε η δημιουργία επιπλέον προγραμμάτων (κυρίως σε perl και bash shell scripting language) ενώ έγινε επίσης χρήση του εργαλείου PlMan που έχει αναπτυχθεί στο πανεπιστήμιο της Washington.

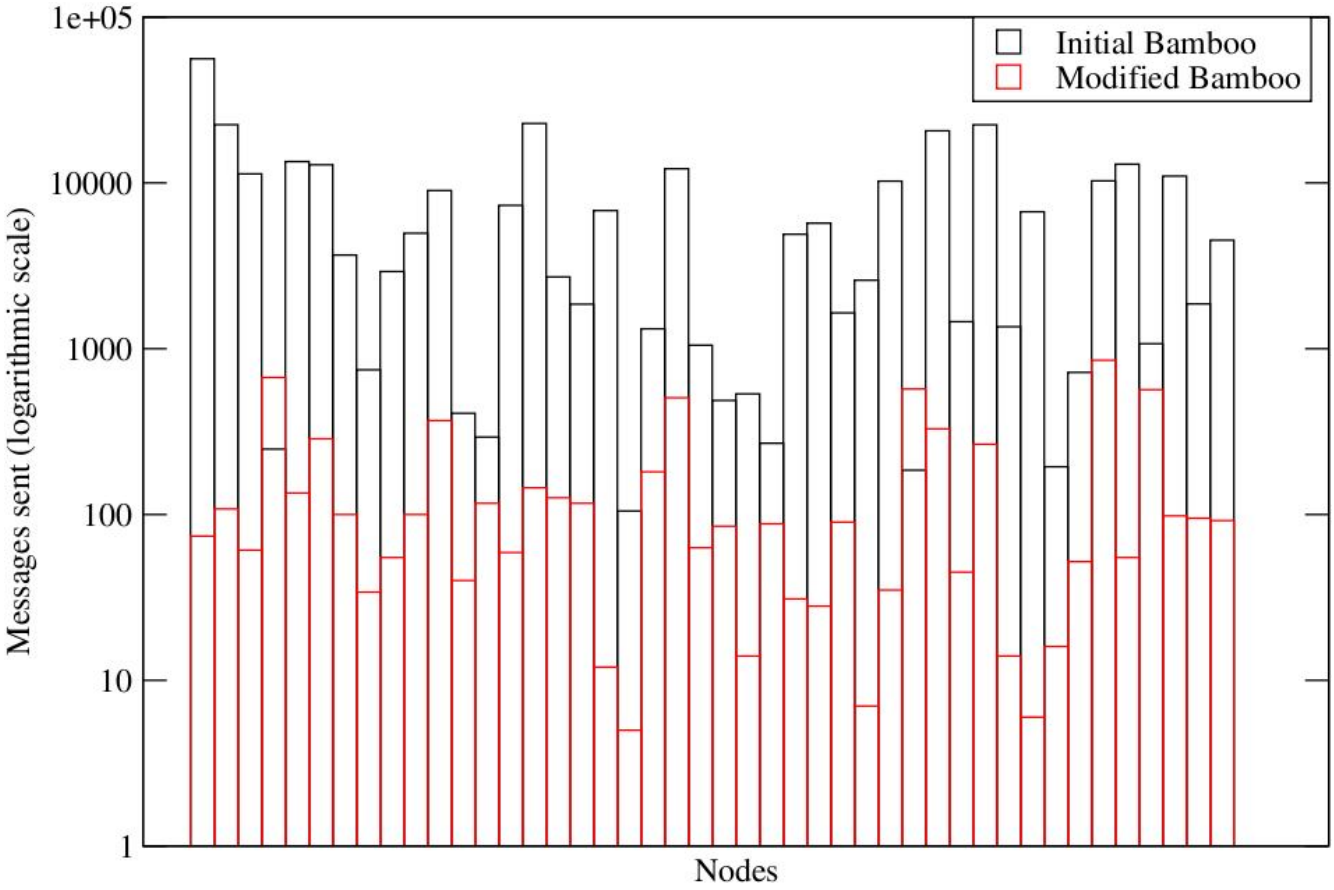
3.4 Παρουσίαση μετρήσεων

Όπως είπαμε και παραπάνω, για την αξιολόγηση του τροποποιημένου Bamboo σε σχέση με το αρχικό, αφήσαμε και τα δύο να εκτελούνται στα ίδια μηχανήματα του Planetlab για 8 ημέρες. Ωστόσο, όταν ήρθε η ώρα να κατεβάσουμε τα αποτελέσματα, μόνο 44 κόμβοι ήταν προσπελάσιμοι. Αυτό μπορεί να οφείλεται είτε σε αποτυχίες των υπολοίπων κόμβων είτε σε πιθανές διαδικασίες συντήρησης του Planetlab.

Σε γενικές γραμμές, ο έλεγχος και η αξιολόγηση του συστήματος έγιναν με σκοπό την διερεύνηση του αν και κατά πόσο, η αλλαγή στον τρόπο επιλογής των γειτόνων που περιγράψαμε νωρίτερα, βελτιώνει την απόδοση του Bamboo όσον αφορά την αποτελεσματικότερη χρήση του διαθέσιμου εύρους ζώνης κάθε κόμβου κι άρα και του συνολικού εύρους ζώνης στο δίκτυο.

Η μέτρηση της αποδοτικότητας του συστήματος βασίστηκε κατά κύριο λόγο στον αριθμό των μηνυμάτων που ανταλλάσσονται μεταξύ των κόμβων και αφορούν στην ενημέρωση των γειτόνων τους σχετικά με αλλαγές που συνέβησαν στον πίνακα δρομολόγησής τους. Το Bamboo, στις παραπάνω περιπτώσεις στέλνει έναν ειδικό τύπο μηνυμάτων στους γείτονές του που ονομάζεται RoutingTableChanged.

Στο σχήμα που ακολουθεί παρουσιάζονται οι μετρήσεις που κάναμε σχετικά με τα μηνύματα που στέλνει κάθε κόμβος στους γείτονές του κι αφορούν σε αλλαγές στον πίνακα δρομολόγησής του. Τα μηνύματα που ανταλλάσσονται είναι τύπου RoutingTableChanged. Στο σχήμα αυτό, ο άξονας X είναι οι κόμβοι του δικτύου μας, ενώ στον άξονα των Y παρουσιάζεται το πλήθος των μηνυμάτων που έστειλε ο κάθε κόμβος κατά τη διάρκεια του πειράματος. Με κόκκινο παρουσιάζονται τα μηνύματα του τροποποιημένου Bamboo ενώ με μαύρο, αυτά του αρχικού.



ΔΙΑΓΡΑΜΜΑ 1:το σχήμα αυτό φαίνονται τα μηνύματα που εστάλησαν ανά κόμβο.

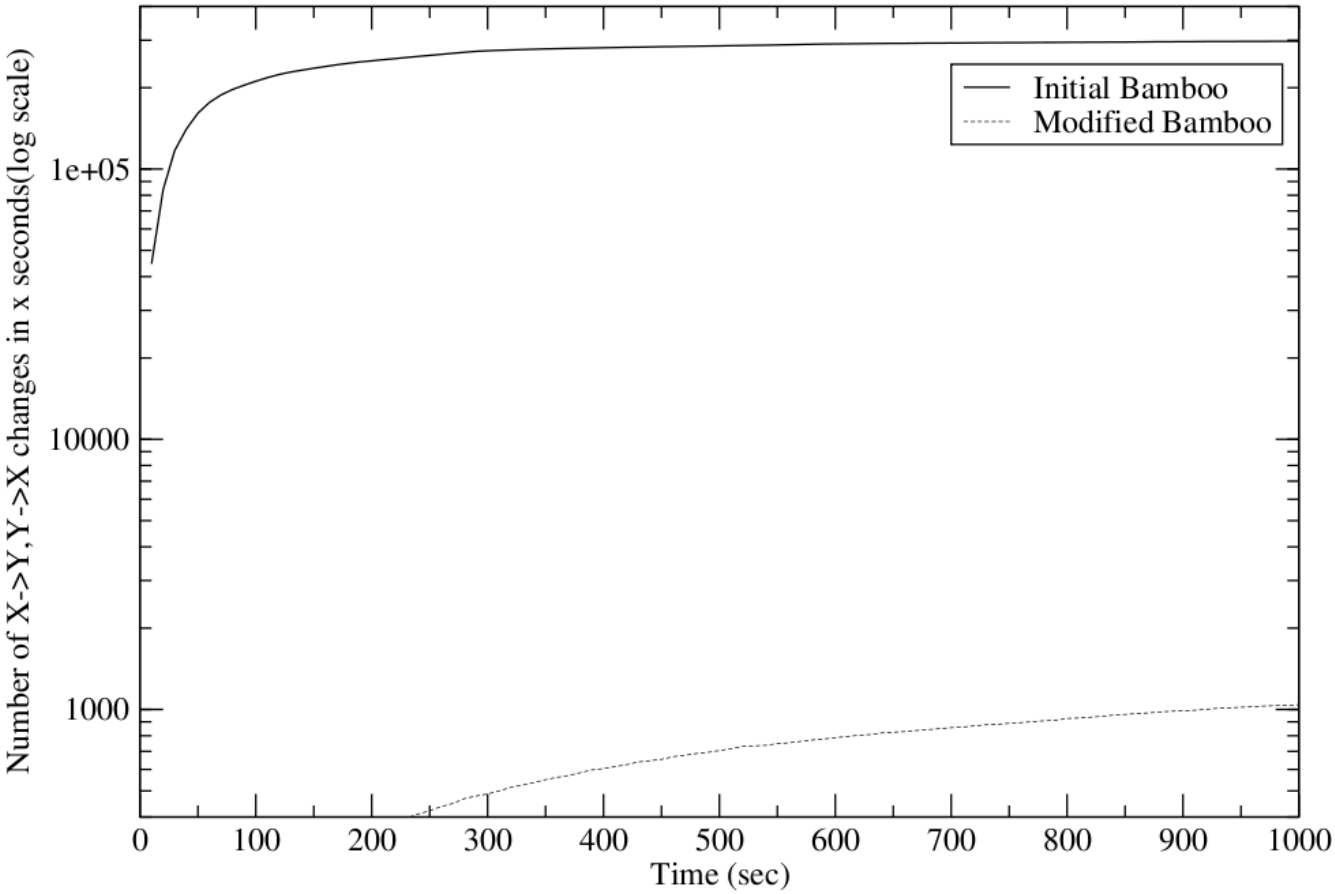
Από το παραπάνω σχήμα, φαίνεται ξεκάθαρα ότι με το νέο αλγόριθμο δρομολόγησης, η αυτορύθμιση του δικτύου μπορεί να επιτευχθεί με πολύ λιγότερα μηνύματα να ανταλλάσσονται μεταξύ των κόμβων του δικτύου. Μάλιστα, στο τροποποιημένο Bamboo παρατηρείται μείωση των μηνυμάτων σχετικά με την αυτορύθμιση του δικτύου που φτάνει τη μία ή και δύο τάξεις μεγέθους ανάλογα τον κόμβο.

Όσον αφορά στο αρχικό Bamboo, και για τους κόμβους που παρατηρούμε, τα μηνύματα που ανταλλάχθηκαν μεταξύ των κόμβων είναι συνολικά 310149 ενώ στο τροποποιημένο είναι μόλις 4504 (μείωση κατά 98.5%). Έτσι, καταλήγουμε στο συμπέρασμα ότι στο τροποποιημένο Bamboo, το διαθέσιμο εύρος ζώνης κάθε κόμβου αξιοποιείται αποδοτικότερα.

Κατά τη μελέτη του αρχικού Bamboo, παρατηρήθηκε επίσης και το φαινόμενο όπου ένας κόμβος, κατά την προσαρμογή των γειτόνων που έχει στον πίνακα δρομολόγησης, αλλάζει τον κόμβο X με τον κόμβο Y, και μετά από σύντομο χρονικό διάστημα (της τάξης των millisecond) αναιρεί την προηγούμενη αλλαγή, βάζοντας πάλι τον κόμβο X στη θέση του κόμβου Y. Αυτές, προφανώς, αποτελούν άσκοπες αλλαγές που έχουν ως αποτέλεσμα, πέρα από την αύξηση του φορτίου στο συγκεκριμένο κόμβο, την περαιτέρω επιβάρυνση συνολικά του δικτύου αφού, ο κόμβος θα ενημερώσει τους γείτονές του για την αλλαγή μέσω της αποστολής επιπλέον μηνυμάτων σε αυτούς κι αυτοί με τη σειρά τους τους δικούς τους γείτονες, κ.ο.κ. Έτσι, επιβαρύνεται μια

ολόκληρη “περιοχή” του δικτύου.

Στο σχήμα που ακολουθεί, φαίνονται οι συνολικές αλλαγές τέτοιου είδους που παρουσιάστηκαν στο δίκτυό μας. Με διακεκομμένη γραμμή παρουσιάζονται οι μετρήσεις στο τροποποιημένο Bamboo, ενώ με συνεχή μαύρη αυτές του αρχικού. Ο άξονας των X είναι το χρονικό διάστημα που μεσολαβεί μεταξύ της πρώτης αλλαγής (X->Y) και της αναίρεσής της (Y->X) ενώ αυτός των Y είναι το πλήθος των ζευγών των αλληλοαναιρούμενων αλλαγών που παρατηρήθηκαν με το διάστημα μεταξύ τους να είναι μικρότερο του αντίστοιχου X. Στο σχήμα παρουσιάζονται οι αλλαγές τέτοιου είδους τον οποίων το διάστημα που μεσολαβεί δεν υπερβαίνει τα 1000 sec.



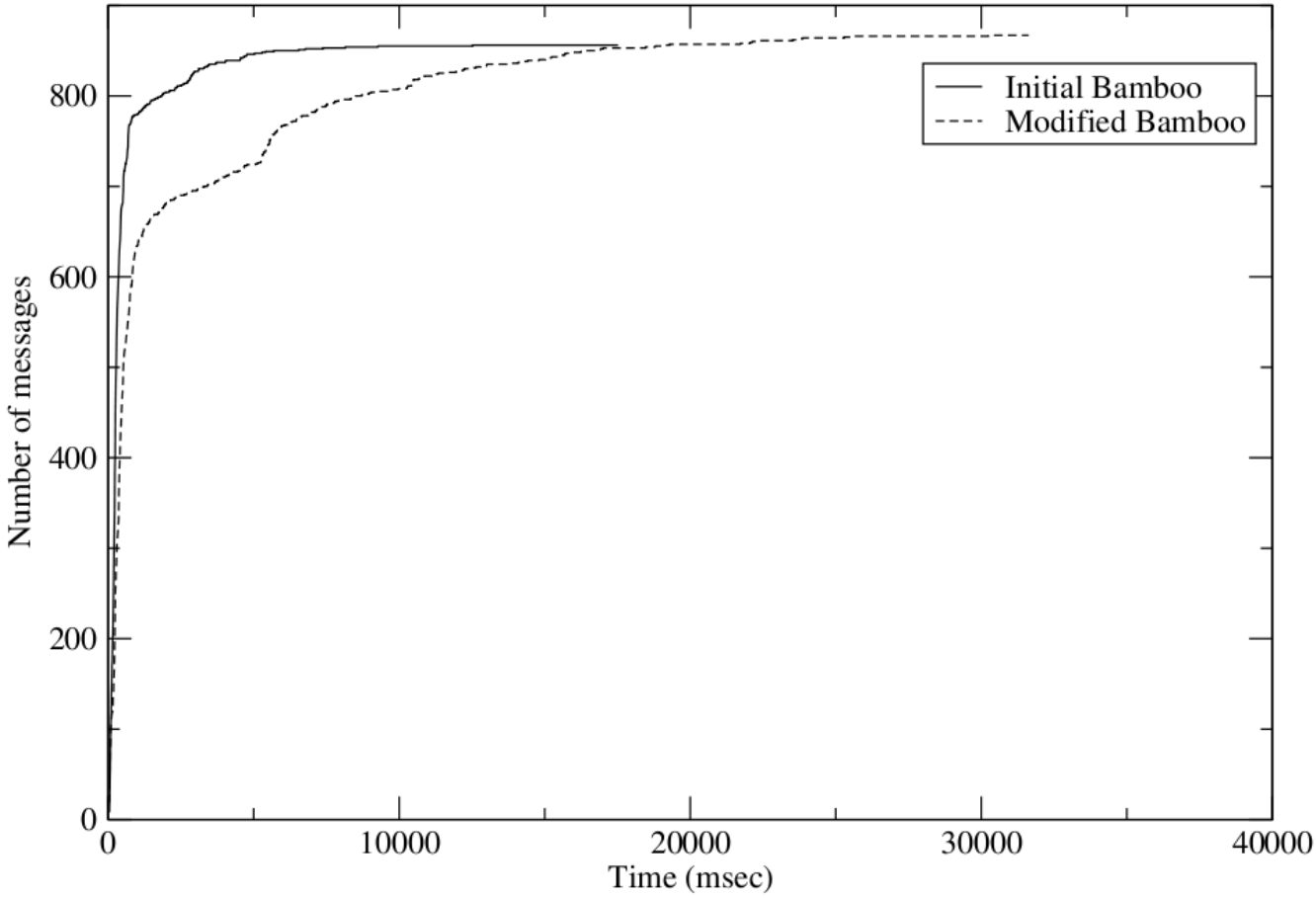
ΔΙΑΓΡΑΜΜΑ 2:Στο σχήμα αυτό φαίνεται το πλήθος των ζευγών των αλλαγών(X->Y,Y->X) στους πίνακες δρομολόγησης του δικτύου, σε σχέση με το χρόνο που μεσολάβησε μεταξύ των δύο αυτών αλλαγών. Παρουσιάζονται μόνο οι αλλαγές των οποίων το ενδιάμεσο διάστημα δεν υπερβαίνει τα 1000 sec.

Από το παραπάνω σχήμα φαίνεται πως το φαινόμενο αυτό περιορίστηκε δραστικά τροποποιημένο Bamboo. Μάλιστα, μπορούμε να πούμε ότι πρακτικά εξαλείφθηκε ιδίως για αλληλοαναιρούμενες αλλαγές που συμβαίνουν σε πολύ μικρό χρονικό διάστημα.

Τέλος, για την εξαγωγή συμπερασμάτων σχετικά με το πώς η τροποποίησή που περιγράψαμε παραπάνω επηρεάζει την επίδοση του Bamboo όσον αφορά στην αναζήτηση, κάναμε χρήση του PutGetTest, το

οποίο και αφήσαμε να τρέχει για 8 ώρες. Το PutGetTest είναι ένα στάδιο *(σύμφωνα με την ορολογία του SEDA),* ήδη υλοποιημένο στο αρχικό Bamboo. Η διεξαγωγή του πειράματος έχει ως εξής : το PutGetTest δημιουργεί τυχαία κλειδιά και δεδομένα κι εκτελεί *put* και *get* διαδικασίες γι' αυτά. Τα put και get γίνονται ανά τυχαία χρονικά διαστήματα. Κάθε επιτυχημένο put αποθηκεύει το κλειδί των δεδομένων που αποθήκευσε στο δίκτυο σε μια λίστα από όπου το τεστ μας ανασύρει τα κλειδιά για τα οποία εκτελεί ένα get. Μάλιστα, για την ορθότητα των μετρήσεων, βάλαμε να εκτελούνται παράλληλα τα δύο τεστ ούτως ώστε, δεδομένης της ταυτόχρονης εκτέλεσης του αρχικού και του τροποποιημένου Bamboo στα ίδια μηχανήματα του Planetlab, οι αναζητήσεις να υπόκεινται στις ίδιες συνθήκες φόρτου του δικτύου και των μηχανημάτων.

Στο σχήμα που ακολουθεί παρουσιάζονται τα αποτελέσματα του παραπάνω πειράματος. Ο Χ άξονας είναι ο άξονας του χρόνου ενώ ο Υ δείχνει πόσα put και get ολοκληρώθηκαν σε χρόνο μικρότερο ή ίσο του αντίστοιχου Χ. Με διακεκομμένη γραμμή παρουσιάζονται τα αποτελέσματα που αφορούν στο τροποποιημένο Bamboo ενώ με συνεχή γραμμή αυτά του αρχικού. Από τα παραπάνω καθίσταται σαφές ότι οι παρακάτω καμπύλες είναι της μορφής P(X<x).



ΔΙΑΓΡΑΜΜΑ 3:Αποτελέσματα put get test.

Από το παραπάνω σχήμα βλέπουμε πως το τροποποιημένο Bamboo έχει μεγαλύτερη διασπορά των χρόνων ολοκλήρωσης των παραπάνω διαδικασιών. Έτσι, παρατηρούμε πως στο αρχικό Bamboo τα περισσότερα put και get διαρκούν έως περίπου 6sec (6000ms) ενώ στο τροποποιημένο έχουμε αρκετά τα οποία διαρκούν έως και 16sec. Σε επίπεδο μέσων όρων παρατηρούμε ότι η διαφορά αγγίζει το 300%, με το μέσο χρόνο του αρχικού Bamboo να είναι 570ms και του τροποποιημένου να είναι 2235ms.

Ωστόσο, ύστερα από με μια πιο ενδελεχή ανάλυση, προκύπτει ότι αυτή η διαφορά οφείλεται σε μόλις 10% των μετρήσεων. Όσον αφορά το υπόλοιπο 90%, το αρχικό Bamboo είναι ταχύτερο κατά μόλις 17%, με το μέσο χρόνο αναζήτησης του τροποποιημένου Bamboo να ανέρχεται σε 611 ms. Το ποσοστό είναι ανεκτό δεδομένου του πολύ μειωμένου αριθμού μηνυμάτων που απαιτούνται για την αυτορύθμιση τού δικτύου μας, σε σχέση με την αρχική υλοποίηση.

Από τα παραπάνω διαγράμματα, καθίσταται σαφές ότι με την τροποποίηση του αλγορίθμου επιλογής γειτόνων του Bamboo που περιγράψαμε παραπάνω, οι αλλαγές γειτόνων στους πίνακες δρομολόγησης των κόμβων μειώθηκαν δραστικά, χωρίς αυτό να συνεπάγεται χειροτέρευση των χρόνων απόκρισης του δικτύου. Αυτό, οφείλεται κυρίως στο ότι αυξάνοντας τα διαστήματα που ο κάθε κόμβος εκτελεί τις περιοδικές του αναζητήσεις για καλύτερους κόμβους-γείτονες, μειώνουμε τις αυξομειώσεις της κίνησης στο δίκτυο που αυτές προκαλούν και που είναι υπεύθυνες για αλλοιώσεις στις μετρήσεις δικτυακών μεγεθών. Επιπλέον, η στιγμιαία φύση της μέτρησης του rtt το καθιστά πιο ευάλωτο σε τέτοιου είδους αυξομειώσεις της κίνησης του δικτύου σε αντίθεση με τη μέτρηση του διαθέσιμου εύρους ζώνης που λόγω του ότι επαναλαμβάνει τη μέτρηση πολλές φορές πριν την εξαγωγή αποτελέσματος, μπορεί και αποφεύγει τέτοιου είδους αλλοιώσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes in Computer Science*, 2001, pp. 46-66.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Citeseer, 2001, pp. 329-350.

[3] M. Castro, P. Druschel, A.M. Kermarrec, and A.I.T. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in communications*, vol. 20, 2002, pp. 1489-1499.

[4] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," *Lecture Notes in Computer Science*, 2001, pp. 14-29.

[5] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," *ACM SIGOPS Operating Systems Review*, vol. 35, 2001, pp. 188-201.

[6] M. Waldman, A.D. Rubin, and L.F. Cranor, "Publius: A Robust, Tamper-Evident Censorship-Resistant Web Publishing System," *9th USENIX Security Symposium*, 2000, pp. 59-72.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," *Proceedings of the 2001 SIGCOMM conference*, ACM New York, NY, USA, 2001, pp. 161-172.

[8] S. Saroiu, P.K. Gummadi, and S.D. Gribble, "A measurement study of peer-to-peer file sharing systems," *Proceedings of Multimedia Computing and Networking*, 2002, p. 152.

[9] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Proceedings of IPTPS02, Cambridge, USA*, vol. 1, 2002, p. 2.2.

[10] F. Dabek, E. Brunskill, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building peer-to-peer systems with Chord, a distributed lookup service," *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Schloss Elmau, Germany, 2001, p. 81?86.

[11] B.Y. Zhao, J. Kubiatowicz, and A.D. Joseph, "Tapestry: An infrastructure for fault-

tolerant wide-area location and routing," *Computer*, vol. 74, 2001.

[12] M. Welsh, D. Culler, and E. Brewer, "SEDA: An architecture for well-conditioned, scalable internet services," *ACM SIGOPS Operating Systems Review*, vol. 35, 2001, pp. 230-243.

[13] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," *Lecture Notes in Computer Science*, 2001, pp. 14-29.

[14] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM New York, NY, USA, 2001, pp. 149-160.

[15] S.C. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, *Handling churn in a DHT*, Computer Science Division, University of California, 2003.

[16] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a common API for structured peer-to-peer overlays," *Lecture Notes in Computer Science*, 2003, pp. 33-44.

[17] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, ACM New York, NY, USA, 2003, pp. 39-44.

[18] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," *In Proceedings of Passive and Active Measurements (PAM) Workshop*, 2002.

[19] H. Balakrishnan, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in P2P systems," *Communications of the ACM*, vol. 46, 2003, pp. 43-48.

[20] Gnutella. http://gnutella.wego.com

[21] Napster. http://www.napster.com

[22] Chord. http://pdos.lcs.mit.edu/chord

[23] Bamboo-dht. http://www.bamboo-dht.org

[24] Pastry. http://research.microsoft.com/~antr/Pastry/

ΠΑΡΑΡΤΗΜΑ Α : Planetlab

Δημιουργία λογαριασμού στο Planetlab

Για τη χρήση μηχανημάτων στο Planetlab, αρχικά απαιτείται η αίτηση για ένα λογαριασμό. Αυτό μπορεί να γίνει μέσω της σελίδας Account Registration και προϋποθέτει ότι το site του αιτούντα είναι εγκεκριμένο από το Planetlab. Με την αποδοχή της τήρησης των όρων που περιέχονται στην Acceptable Use Policy (AUP) θα επιτραπεί η αίτηση για λογαριασμό στο Planetlab μέσω του site του εργαστηρίου (στο Planetlab).

Για την απόκτηση πρόσβασης σε κόμβους, συμπεριλαμβανομένων κι αυτών του site του εργαστηρίου, απαιτείται η δημιουργία ενός ζεύγους SSH κλειδιών. Μάλιστα, μόνο τα SSH κλειδιά που χρησιμοποιούν RSA πρωτόκολλο πιστοποίησης επιτρέπονται στο Planetlab. Το RSA βασίζεται σε κρυπτογράφηση δημοσίου κλειδιού. Επιπλέον, το κλειδί θα πρέπει να είναι συμβατό με το OpenSSH πρότυπο. Αυτό σημαίνει ότι σε ένα σύστημα που "τρέχει" Linux, η πρώτη γραμμή θα πρέπει να μοιάζει με την παρακάτω :

- ssh-rsa AAAAB3Nza...

Μετά τη δημιουργία του παραπάνω κλειδιού, το δημόσιο κλειδί id_rsa.pub πρέπει να "ανέβει" στο site του Planetlab μέσω της σελίδας Manage My Keys.

Μετά την ολοκλήρωση των παραπάνω βημάτων, πρέπει ο διαχειριστής του site του εργαστηρίου, να δημιουργήσει μία καινούρια slice για τον νέο χρήστη ή να του δώσει δικαιώματα χρήσης μιας ήδη υπάρχουσας.

Από εδώ και πέρα, ο χρήστης μπορεί να χρησιμοποιήσει τους κόμβους στη slice του. Για το σκοπό αυτό μπορεί να συνδέεται μέσω SSH με τους κόμβους που σχετίζονται με το slice του και να εκτελεί έτσι τα πειράματα. Για το upload και των download των απαραίτητων αρχείων μπορεί να γίνει χρήση του SCP.

PlMan

Τα παραπάνω είναι αρκετά για τη διεξαγωγή πειραμάτων στο Planetlab. Ωστόσο, για την ευκολότερη διαχείριση και συντήρηση των slices και λόγω της μεγάλης αποδοχής που τυγχάνει το Planetlab μεταξύ της ακαδημαϊκής κοινότητας, υπάρχουν αρκετά open source εργαλεία δημιουργημένα από χρήστες του Planetlab για το σκοπό αυτό.

Για τις ανάγκες της διπλωματικής κάναμε χρήση του PlMan, ενός εργαλείου από το πανεπιστήμιο της

Washington. Το εργαλείο αυτό δημιουργήθηκε για την απλοποίηση των διαδικασιών χειρισμού, εκτέλεσης και παρακολούθησης ενός πειράματος. Η εφαρμογή περιέχει ένα απλό GUI για τη διενέργεια απλών και συχνών εργασιών όπως :

- *επιλογή κόμβων για τη slice μας*
- *επιλογή κόμβων για το πείραμά μας βάσει CoMon πληροφοριών για τους κόμβους*
- *αξιόπιστο χειρισμό των αρχείων που αφορούν στο πείραμα*
- *παράλληλη εκτέλεση εντολών σε κάθε κόμβο*
- *παρακολούθηση συνολικά της προόδου του πειράματος, καθώς και την έξοδο κάθε κόμβου χωριστά*

Το PlMan, όπως είπαμε και παραπάνω, είναι ένα εργαλείο που βοηθά στην διεκπεραίωση απλών και συχνών εργασιών οι οποίες είναι κοινές για όλους τους κόμβους όπως "ανέβασμα" και "κατέβασμα" αρχείων, παράλληλη εκτέλεση εντολών, κλπ.

Ωστόσο, σε κάποιες περιπτώσεις, κρίθηκε απαραίτητη η δημιουργία επιπλέον προγραμμάτων για τη διεκπεραίωση συγκεκριμένων εργασιών.

Έτσι, για τη δημιουργία των .cfg αρχείων που είναι απαραίτητα για την εκκίνηση του Bamboo σε ένα μηχάνημα, δημιουργήσαμε ένα πρόγραμμα σε perl, το *GatewayCFGconstructor.pl*, που μας επιτρέπει τη δημιουργία του αρχείου αυτού σε κάθε κόμβο μέσω της παράλληλης εκτέλεσης της εντολής :

perl -w GatewayCFGconstructor.pl `uname -n` 3640

Το παραπάνω κατέστησε εφικτή τη δημιουργία ενός προγράμματος σε shell scripting language για την αυτόματη εκκίνηση του Bamboo σε κάθε κόμβο χωρίς να απαιτείται ξεχωριστός χειρισμός κάθε κόμβου για το "ανέβασμα" του αντίστοιχου .cfg.

Τέλος, δημιουργήσαμε προγράμματα σε Java για τον αυτόματο χειρισμό των log αρχείων κάθε κόμβου. Τα προγράμματα αυτά αναλαμβάνουν αυτόματα το "κατέβασμα" των παραπάνω αρχείων, την επεξεργασία των μετρήσεων και την παρουσίασή τους μέσω διαγραμμάτων.

Το "κατέβασμα" των log αρχείων γίνεται με scp. Για το σκοπό αυτό κάναμε χρήση της βιβλιοθήκης Trilead SSH, μιας open source βιβλιοθήκης της Java που υλοποιεί το πρωτόκολλο SSH. Η επεξεργασία των μετρήσεων γίνεται μέσω ενός parser που αναπτύξαμε γι' αυτό το σκοπό και ο οποίος είναι ειδικός για κάθε μορφή διαγράμματος που θέλουμε. Τέλος, οι επεξεργασμένες πλέον μετρήσεις αποτυπώνονται σε διαγράμματα μέσω του εργαλείου *xmgrace.*