ΔΗΜΙΟΥΡΓΙΑ ΑΝΤΙΓΡΑΦΩΝ XML ΕΓΓΡΑΦΩΝ ΜΕ ΤΗ ΜΕΘΟΔΟ ΚΩΔΙΚΟΠΟΙΗΣΗΣ ΑΠΑΛΟΙΦΗΣ ΣΕ ΔΙΚΤΥΑ ΟΜΟΤΙΜΩΝ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

Παναγιώτη Γιωτάκη

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΗΝ ΘΕΩΡΙΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ιούλιος 2009

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Τα Δίκτυα Ομότιμων (Peer-to-Peer Networks) είναι κατανεμημένα συστήματα ομότιμων κόμβων τα οποία χρησιμοποιούν την υποδομή του Διαδικτύου και τους πόρους του για την επικοινωνία μεταξύ των κόμβων. Κύριο χαρακτηριστικό των κόμβων είναι το γεγονός ότι είναι *ομότιμοι*, δηλαδή κάθε κόμβος συμμετέχει ισάξια στο σύστημα, έχει τα ίδια δικαιώματα και τις ίδιες υποχρεώσεις με τους υπόλοιπους. Βασικός παράγοντας στην αξιοπιστία και βιωσιμότητα ενός Δικτύου Ομότιμων είναι η διαθεσιμότητα των αρχείων που διαμοιράζονται σε αυτό.

Στην παρούσα εργασία αναλύεται η μέθοδος της Κωδικοποίησης Απαλοιφής (Erasure Coding) ως μέσον για διατήρηση αντιγράφων σε ένα δίκτυο ομότιμων. Βάση της Κωδικοποίησης Απαλοιφής είναι το γεγονός ότι ένα έγγραφο μπορεί να χωριστεί σε m ίσα τμήματα και να κωδικοποιηθεί σε νέα n (n>m) τμήματα. Η ιδιότητα των n τμημάτων αυτών είναι ότι οποιαδήποτε m από αυτά μπορούν να μας δώσουν το αρχικό έγγραφο. Συγκεκριμένα, χρησιμοποιούμε τη μέθοδο των Luby Transform Κωδίκων (LT Codes), η οποία στηρίζεται στην ιδέα των αραιών γράφων (Sparse Graphs) για την κωδικοποίηση και αποκωδικοποίηση των εγγράφων.

Τους LT Κώδικες τους εφαρμόζουμε σε XML έγγραφα, τα οποία διαμοιράζονται οι κόμβοι σε ένα αδόμητο και μη κεντρικοποιημένο δίκτυο ομότιμων. Εξετάζουμε τα πλεονεκτήματα και μειονεκτήματα της μεθόδου κατά τη διαχείριση τέτοιων εγγράφων. Επιπλέον, περιγράφουμε τη διαδικασία αποθήκευσης και ανάκτησης XML εγγράφων που είναι κωδικοποιημένα με τη μέθοδο αυτή, έπειτα από κάποια αναζήτηση. Τέλος, τη συγκρίνουμε με άλλη μια δημοφιλή μέθοδο διατήρησης αντιτύπων, αυτή της Αντιγραφής (Replication).

EXTENDED ABSTRACT IN ENGLISH

Peer-to-Peer (p2p) networks are distributed systems, which use the structure and resources of the Internet for the communication among the nodes. Main characteristic of the nodes is that they are peers, meaning they participate equally in the network, have the same rights and same obligations as the rest. One of the most important factors for the reliability and viability of p2p networks is the availability of the files shared in them.

In this thesis we analyze the method of Erasure Coding, as the means of maintaining copies of a document in a p2p network. Base of Erasure Coding s the fact that a document can be split into m equal pieces and then coded into n (n>m) new pieces. The characteristic of these pieces is that any m of these n pieces can reconstruct the original document. Specifically, we use the Luby Transform (LT) Coding, which relies on the use of sparse graphs for the coding and decoding of the documents.

We apply LT Codes on XML based documents being shared in an unstructured and decentralized p2p network. We describe the advantages and disadvantages of this method, when handling such documents. Moreover, we present the way of storing and retrieving of XML fragments coded with the Erasure Coding method, after a query from a node. Finally, we compare LT Codes against another popular method of maintaining copies in a p2p network, that of Replication.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Στόχοι

Τα Δίκτυα Ομότιμων (Peer-to-Peer Networks) είναι κατανεμημένα συστήματα ομότιμων κόμβων τα οποία χρησιμοποιούν την υποδομή του Διαδικτύου και χρησιμοποιούν τους πόρους του για την επικοινωνία μεταξύ των κόμβων. Τα συστήματα αυτά χρησιμοποιούνται για το διαμοιρασμό διαφόρων αρχείων μεταξύ των χρηστών. Κύριο χαρακτηριστικό των κόμβων είναι το γεγονός ότι είναι *ομότιμοι*, δηλαδή κάθε συμμετέχει ισάξια στο σύστημα, έχει τα ίδια δικαιώματα και τις ίδιες υποχρεώσεις με τους υπόλοιπους. Βασικός παράγοντας στην αξιοπιστία και βιωσιμότητα ενός Δικτύου Ομότιμων είναι η διαθεσιμότητα των αρχείων που διαμοιράζονται σε αυτό. Μια ευρέως διαδεδομένη μέθοδος διατήρησης αντιγράφων των αρχείων σε ένα τέτοιο Δίκτυο, είναι η μέθοδος της Αντιγραφής (Replication). Με τη μέθοδο αυτή δημιουργούνται διάφορα αντίγραφα ενός εγγράφου, τα οποία αποθηκεύονται σε διαφορετικούς κόμβους που συμμετέχουν στο Δίκτυο. Έτσι, αν αποχωρήσει ο κόμβος ο οποίος περιέχει το συγκεκριμένο έγγραφο, αυτό θα είναι διαθέσιμο στο Δίκτυο μέσω των αντιγράφων του.

Στην παρούσα εργασία εισάγεται μια νέα μέθοδος δημιουργίας και διατήρησης αντιγράφων σε ένα Δίκτυο Ομότιμων. Η μέθοδος αυτή αποκαλείται Κωδικοποίηση Απαλοιφής (Erasure Coding). Βάση της Κωδικοποίησης Απαλοιφής είναι το γεγονός

ότι ένα έγγραφο μπορεί να χωριστεί σε m ίσα τμήματα και να κωδικοποιηθεί σε νέα n (n>m) τμήματα. Η ιδιότητα των n τμημάτων αυτών είναι ότι οποιαδήποτε m από αυτά μπορούν να μας δώσουν το αρχικό έγγραφο. Συνεπώς, αποθηκεύοντας τα τμήματα αυτά σε n διαφορετικούς κόμβους σε ένα Δίκτυο Ομότιμων, πετυχαίνουμε μεγάλη διαθεσιμότητα του εγγράφου, κερδίζοντας παράλληλα σε αποθηκευτικό κόστος.

Η μέθοδος της Κωδικοποίησης Απαλοιφής που θα χρησιμοποιήσουμε είναι οι Luby-Transform (LT) Κώδικες Απαλοιφής. Η μέθοδος αυτή ανήκει στην κατηγορία των Κωδικών Πηγής, οι οποίοι είναι πιθανοτικοί αλγόριθμοι και στηρίζονται στην ιδέα των Αραιών Γράφων (Sparse Graphs) για την Κωδικοποίηση και Αποκωδικοποίηση των εγγράφων.

Στόχος της παρούσας εργασίας είναι να αναλύσουμε τη μέθοδο της Κωδικοποίησης Απαλοιφής και συγκεκριμένα των LT Κωδικών, παρουσιάζοντας τα χαρακτηριστικά τους. Επιπλέον, αναλύουμε τα πλεονεκτήματα και μειονεκτήματά τους ως μέθοδο διατήρησης αντιγράφων. Επιπλέον, επιθυμούμε να ερευνήσουμε τη συμπεριφορά των Κωδικών Απαλοιφής σε ένα Δίκτυο Ομότιμων και να τη συγκρίνουμε με τη μέθοδο της Αντιγραφής, ώστε να επισημάνουμε σε ποιους τομείς υπερτερεί και σε ποιους όχι.

Τέλος χρησιμοποιούμε τη μέθοδο των LT Κωδικών σε XML έγγραφα. Στόχος είναι να εξετάσουμε τα χαρακτηριστικά που πρέπει να εισαχθούν στη μέθοδο της Κωδικοποίησης Απαλοιφής που χρησιμοποιούμε, ώστε να διαχειρίζεται τέτοια έγγραφα μέσα σε ένα Δίκτυο Ομότιμων.

1.2 Σχετική Εργασία

Στο [5] οι Rodrigues και Liscov εστιάζουν στη σύγκριση ανάμεσα στην Κωδικοποίηση Απαλοιφής (Erasure Coding) και την Αντιγραφή (Replication), σε δίκτυα ομότιμων που χρησιμοποιούν κατανεμημένους πίνακες κατακερματισμού (Distributed Hash Tables-DHTs). Ιδιαίτερα λαμβάνουν υπ’ όψιν τα χαρακτηριστικά των κόμβων που συμμετέχουν σε αυτό. Η σύγκριση των δύο μεθόδων γίνεται με βάση τη διαθεσιμότητα των κόμβων του δικτύου, καθώς και με βάση την απαιτούμενη επιπρόσθετη πληροφορία (redundancy) που πρέπει να αποθηκευτεί σε

σχέση με τη διαθεσιμότητα και το χρησιμοποιούμενο εύρος δικτύου. Η μελέτη καταλήγει στο συμπέρασμα ότι σε συνθήκες όπου η διαθεσιμότητα των κόμβων είναι μεγάλη, ευνοείται η χρήση της Αντιγραφής, ενώ στις περιπτώσεις που είναι χαμηλότερη, προτιμάται η Κωδικοποίηση Απαλοιφής. Τα πειράματα διεξήχθησαν με τη χρήση των συστημάτων PlanetLab, Overnet και Farsite.

Στο [4] οι Weatherspoon και Kubiatowicz συγκρίνουν ποσοτικά τις μεθόδους Κωδικοποίησης Απαλοιφής και Αντιγραφής σε κατανεμημένα συστήματα αποθήκευσης. Η σύγκριση αυτή γίνεται με βάση τη χρήση εύρους δικτύου και το αποθηκευτικό κόστος που επιβάλλονται σε τέτοια συστήματα με παρόμοιο μέσο χρόνο αποτυχιών (Mean Time To Failure- MTTF). Για τη διεξαγωγή της συγκεκριμένης έρευνας έγινε η υπόθεση ότι οι αποτυχίες των μέσων αποθήκευσης είναι ανεξάρτητες και κατανεμημένες με τον ίδιο τρόπο. Μελλοντική εργασία από τους συγγραφείς σκοπεύει στην έρευνα των δύο μεθόδων με εξαρτώμενες αποτυχίες, τυχαία κατανεμημένες. Η μελέτη τους καταλήγει στο ότι η Κωδικοποίηση Απαλοιφής χρησιμοποιεί μια τάξη μεγέθους λιγότερο εύρος δικτύου και αποθηκευτικό κόστος σε σχέση με την Αντιγραφή.

Στο [22] πραγματοποιείται σύγκριση των μεθόδων Κωδικοποίησης Απαλοιφής και Αντιγραφής με στόχο να καθοριστούν οι συνθήκες κάτω από τις οποίες υπερτερεί η μία μέθοδος έναντι της άλλης και το αντίστροφο. Για την επίτευξη του στόχου αυτού έγιναν πειράματα ώστε να διαπιστωθεί η απόδοση των μεθόδων αυτών έπειτα από σημαντικές αλλαγές στη διαθεσιμότητα των κόμβων, καθώς και το επιπρόσθετο αποθηκευτικό κόστος που επιβάλλουν για τη διατήρηση της κανονικής λειτουργίας του δικτύου. Εκτός από τα πειράματα, οι συγγραφείς πράττουν μια θεωρητική ανάλυση πάνω στα πειράματα και καταλήγουν στα ίδια συμπεράσματα μέσω μαθηματικών αναλύσεων. Τα αποτελέσματα που βγήκαν και σε αυτή την έρευνα, όπως και στο [5], δείχνουν ότι η Κωδικοποίηση Απαλοιφής πρέπει να προτιμάται όταν η διαθεσιμότητα των κόμβων είναι μικρή. Αντιθέτως, ενδείκνυται η Αντιγραφή όταν έχουμε μεγάλη διαθεσιμότητα κόμβων στο δίκτυο.

Στο [23] ερευνάται η διαδικασία εφαρμογής της Κωδικοποίησης Απαλοιφής, ως μεθόδου διατήρησης αντιγράφων, σε ένα δίκτυο ομότιμων. Επιπλέον, γίνεται

ανάλυση ώστε να επιδειχθούν οι διαφορές σε σχέση με τη μέθοδο της Αντιγραφής. Η σύγκριση πραγματοποιείται με παραμέτρους τη CPU, το κόστος αποθήκευσης, ανοχή αποτυχιών και χρόνο αποθήκευσης και ανάκτησης δεδομένων. Το σύστημα αποθήκευσης που προτείνεται είναι το p2peur, το οποίο χτίστηκε πάνω στο σύστημα αποθήκευσης Hispread [24]. Το Hispread είναι ένα δίκτυο ομότιμων το οποίο χρησιμοποιεί την Αντιγραφή για διατήρηση των δεδομένων στους κόμβους. Το πλήθος των αντιγράφων καθορίζεται από το χρήστη και κάθε ένα από αυτά αποθηκεύεται σε διαφορετικό κόμβο. Οι ιδιότητες που λαμβάνονται υπ' όψιν είναι η διαθεσιμότητα των κόμβων και του αποθηκευτικού χώρου. Το Hispread χρησιμοποιεί ένα κατακερματισμένο σχήμα ηλεκτρονικών υπογραφών (hash based signature scheme) για την κρυπτογράφηση των αρχείων του, ώστε να διασφαλιστεί ότι τα δεδομένα δεν αλλάζουν κατά τη διάρκεια αναζήτησης και αποθήκευσής τους στον κόμβο που τα ζήτησε. Το p2peur εφαρμόζει τις βιβλιοθήκες Κωδικοποίησης Απαλοιφής του προγράμματος PASIS [25]. Το PASIS χρησιμοποιεί αλγορίθμους για τν καλύτερη δυνατή επιλογή μεθόδου κατανομής δεδομένων. Αυτό περιλαμβάνει τρία βήματα: απαρίθμηση των πιθανών σχημάτων για κατανομή δεδομένων σε ένα σύστημα αποθήκευσης, μοντελοποίηση των συνεπειών έπειτα από τη χρήση των σχημάτων αυτών και προσδιορισμό της καλύτερης κατανομής για ένα σύστημα με συγκεκριμένες παραμέτρους διαθεσιμότητας και ασφάλειας των δεδομένων. Τα συμπεράσματα που προκύπτουν δείχνουν ότι η Κωδικοποίηση Απαλοιφής λειτουργεί καλύτερα όταν έχουμε πτώση στη διαθεσιμότητα των κόμβων του δικτύου. Επιπλέον, ο χρόνος αποθήκευσης και ανάκτησης είναι καλύτερος από ότι στην Αντιγραφή. Τέλος, η χρήση των πόρων του συστήματος (CPU) είναι μεγαλύτερη στην Κωδικοποίηση Απαλοιφής (μεγαλύτερη αναζήτηση κόμβων για την εύρεση των τμημάτων που απαιτούνται).

Στο [21] προτείνεται η χρήση κατανεμημένων πινάκων κατακερματισμού (DHT) οι οποίοι δρομολογούν την αποθήκευση και ανάκτηση δεδομένων σε ένα δίκτυο ομότιμων. Ο DHT που χρησιμοποιείται στη συγκεκριμένη εργασία αποκαλείται Reperasure και περιλαμβάνει πρωτόκολλα για την αποθήκευση και ανάκτηση δεδομένων που έχουν υποστεί Κωδικοποίηση Απαλοιφής. Τα πρωτόκολλα αυτά είναι πρωτόκολλα ανάγνωσης και αποθήκευσης κωδικοποιημένων τμημάτων, καθώς και αναβάθμισης των τμημάτων, ώστε να μην υπάρχει σύγκρουση (collision) όταν δύο

έγγραφα υφίστανται επεξεργασία από διαφορετικούς χρήστες ταυτόχρονα. Οι συγγραφείς καταλήγουν στο ότι η χρήση του DHT δίνει μεγάλη αξιοπιστία στη λειτουργία του δικτύου και επίσης ότι αυξάνει τη συνοχή (consistency) των δεδομένων που αποθηκεύονται σε αυτό.

1.3 Δομή της Διατριβής

Η διατριβή περιέχει 6 κεφάλαια: Το Κεφάλαιο 2 εισάγει την έννοια της Κωδικοποίησης Απαλοιφής και τις ιδιότητές της. Επίσης παρουσιάζονται οι LT Κώδικες Απαλοιφής και αναλύονται οι Αλγόριθμοι Κωδικοποίησης και Αποκωδικοποίησης που χρησιμοποιούν.

Στο Κεφάλαιο 3 παρουσιάζουν το Δίκτυο Ομότιμων που θα υλοποιήσουμε, αναλύοντας την αρχιτεκτονική του. Εξετάζουμε τις ιδιότητες των κόμβων του Δικτύου αυτού, ώστε να χρησιμοποιείται η LT μέθοδος για διατήρηση αντιγράφων των εγγράφων μέσα σε αυτό. Επιπλέον, αναλύουμε τους τις διαδικασίες εισαγωγής και αποθήκευσης ενός νέου εγγράφου στο δικτύου, αλλά και τη διαδικασία αναζήτησης και ανάκτησης ενός κωδικοποιημένου εγγράφου από κάποιο κόμβο.

Στο Κεφάλαιο 4 κάνουμε μια εισαγωγή στη γλώσσα Σήμανσης XML, παρουσιάζοντας τα χαρακτηριστικά της. Στη συνέχεια επεκτείνουμε τη λειτουργία των LT Κωδικών, ώστε να διαχειρίζονται XML δεδομένα σε ένα Δίκτυο Ομότιμων. Ακόμη, παρουσιάζονται οι διάφοροι τύποι επερωτήσεων που μπορεί να κάνει ένας κόμβος στο Δίκτυο, για την ανάκτηση ενός εγγράφου κωδικοποιημένου με τους LT Κώδικες Απαλοιφής.

Στο Κεφάλαιο 5 πραγματοποιείται μια σειρά από πειράματα, τα οποία έχουν δύο κατευθύνσεις. Η πρώτη σειρά πειραμάτων αφορά στη γενικότερη συμπεριφορά της μεθόδου των LT Κωδικών. Εκτιμάται η απόδοση της Αποκωδικοποίησης ανάλογα με το πλήθος των Κωδικοποιημένων Τμημάτων που λαμβάνονται, καθώς και οι χρόνοι Κωδικοποίησης και Αποκωδικοποίησης. Η δεύτερη σειρά πειραμάτων εξάγει

ορισμένα συμπεράσματα για τη λειτουργία της μεθόδου σε ένα δίκτυο Ομότιμων και πραγματοποιούνται συγκρίσεις με τη μέθοδο της Αντιγραφής. Οι τομείς που θα πραγματοποιηθεί η σύγκριση αφορούν τη διαθεσιμότητα ενός εγγράφου στο Δίκτυο, το κόστος αποθήκευσης που επιβάλλουν οι δύο μέθοδοι σε αυτό, καθώς και τα ποσοστά επιτυχών αναζητήσεων των μεθόδων σε συνθήκες μικρού ή μεγάλου ποσοστού αποχωρήσεων κόμβων από το Δίκτυο.

Τέλος, στο κεφάλαιο 6 συνοψίζουμε τα συμπεράσματα που εξάγουμε από τα πειράματα που θα πραγματοποιηθούν και παραθέτουμε μελλοντικές προεκτάσεις που μπορούν να πραγματοποιηθούν ως εξέλιξη της υπάρχουσας ερευνητικής προσπάθειας.

ΚΕΦΑΛΑΙΟ 2. ΚΩΔΙΚΟΠΟΙΗΣΗ ΑΠΑΛΟΙΦΗΣ

2.1 Κανάλια Απαλοιφής (Erasure Channels)

Παραδοσιακά, στο Διαδίκτυο, η επικοινωνία μεταξύ των υπολογιστών βασιζόταν στο πρωτόκολλο TCP (Transmission Control Protocol). Με το πρωτόκολλο TCP παρέχεται μια αξιόπιστη υπηρεσία μεταφοράς δεδομένων από τον αποστολέα προς τον παραλήπτη. Το πρωτόκολλο αυτό αντιμετωπίζει τα δεδομένα ως μια ταξινομημένη αλληλουχία πακέτων τα οποία στέλνονται μέσω ενός καναλιού επικοινωνίας προς τον προορισμό τους. Ο παραλήπτης πρέπει να λάβει το πλήθος των πακέτων που συνιστούν ένα αρχείο και μάλιστα με τη σωστή τους σειρά. Για επίτευξη μεγαλύτερης αξιοπιστίας, το πρωτόκολλο TCP περιλαμβάνει και την αποστολή ξανά των πακέτων σε περίπτωση που υπάρξει κάποιο χαμένο πακέτο κατά την αρχική αποστολή τους. Ωστόσο, σε πολλές εφαρμογές η ανάγκη για αποστολή και λήψη πακέτων με τη σωστή τους σειρά αποδεικνύεται ιδιαίτερα ανασφαλής και περιοριστική.

Ας θεωρήσουμε, για παράδειγμα, το πρόβλημα διανομής μιας μεγάλης αναβάθμισης ενός λογισμικού σε χιλιάδες υπολογιστές που βρίσκονται σε διαφορετικά (γεωγραφικά) κατανεμημένα συστήματα. Κάποιοι από τους υπολογιστές αυτούς μπορεί να συνδέονται μεταξύ τους με δορυφόρο ή με κάποιο άλλο ασύρματο δίκτυο. Η

αποστολή της αναβάθμισης αυτής μέσω του TCP καναλιού προκαλεί την περιττή κατανάλωση εύρους γραμμής (bandwidth), εξαιτίας της αναμετάδοσης των πακέτων προς τον προορισμό τους. Επίσης, υπάρχει ο κίνδυνος, εξαιτίας της δυναμικής φύσης των δικτύων, τα πακέτα να φτάσουν στους υπολογιστές με λάθος σειρά ή κάποια να μη φτάσουν καθόλου, κάτι που έχει ως συνέπεια την αποτυχία λήψης του αρχείου. Αυτό εγείρει και ένα άλλο ζήτημα, την ανάγκη ύπαρξης ενός καναλιού με διπλή κατεύθυνση, από τον αποστολέα προς τον παραλήπτη και το αντίστροφο. Αυτό είναι ουσιαστικό, διότι θα πρέπει ο παραλήπτης ενός αρχείου να μπορεί να ενημερώνει τον αποστολέα για τυχόν αποτυχίες σε κάποια πακέτα, έτσι ώστε να σταλούν ξανά μόνο αυτά και όχι ολόκληρο το αρχείο.

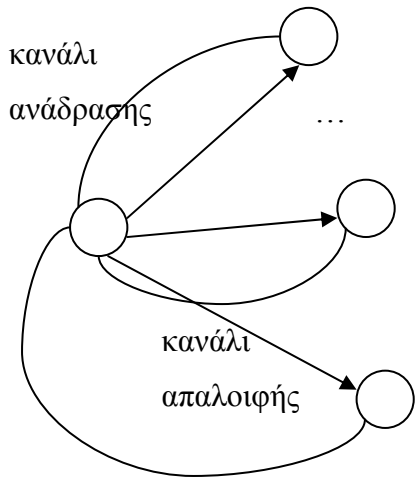
Το ζήτημα της αξιόπιστης μετάδοσης δεδομένων στο Διαδίκτυο έχει λάβει ιδιαίτερη μελέτη και έρευνα. Λόγω των προβλημάτων που ανακύπτουν από τα κανάλια με τη χρήση του TCP πρωτοκόλλου, έχουν προταθεί κανάλια επικοινωνίας με ορισμένες επιθυμητές ιδιότητες. Ένα διαφορετικό είδος καναλιού επικοινωνίας σε ένα δίκτυο είναι τα αποκαλούμενα Κανάλια Απαλοιφών (Erasure Channels) [1].

Ορισμός 1: Ως Κανάλι Απαλοιφής ορίζουμε ένα δίαυλο επικοινωνίας μεταξύ δύο συστημάτων, ενός αποστολέα και ενός παραλήπτη, μέσω του οποίου ένα πακέτο δεδομένων που στέλνεται από τον αποστολέα είτε φτάνει αυτούσιο χωρίς σφάλμα στον παραλήπτη, είτε δεν λαμβάνεται καθόλου.

Μέσω των Καναλιών Απαλοιφής, ένα αρχείο χωρίζεται σε πακέτα, τα οποία στη συνέχεια στέλνονται στον παραλήπτη. Στην περίπτωση αυτή δεν είναι απαραίτητο όλα τα πακέτα να αποσταλούν και να ληφθούν με συγκεκριμένη σειρά, όπως γίνεται μέσω του πρωτοκόλλου TCP. Εάν κάποιο τμήμα δεν ληφθεί, αρκεί η επανάληψη της αποστολής του και όχι ολόκληρου του αρχείου. Είναι προφανές ότι με τον τρόπο αυτό υπάρχει κέρδος σε εύρος γραμμής, δηλαδή δεν χρειάζεται μεγάλη κατανάλωση των πόρων του δικτύου για την επιτυχή αποστολή των αρχείων.
Πολλές φορές στο κανάλι απαλοιφής μεταξύ αποστολέα και παραλήπτη, τα τμήματα που συνθέτουν ένα αρχείο δεν στέλνονται πλήρη στον προορισμό τους. Αυτό έχει ως αποτέλεσμα, το αρχείο να μην μπορεί να ανακτηθεί από τον παραλήπτη. Για την αντιμετώπιση αυτής της ανεπιθύμητης κατάστασης, πολλές φορές χρησιμοποιείται ένα

κανάλι ανάδρασης (feedback channel) στο οποίο στέλνονται μηνύματα από τον παραλήπτη προς τον αποστολέα. Συγκεκριμένα, αν κάποιο τμήμα ενός αρχείου δεν έχει φτάσει με επιτυχία προς τον παραλήπτη, τότε στέλνει ένα μήνυμα στον αποστολέα, ζητώντας την μετάδοση του συγκεκριμένου τμήματος που απέτυχε. Λαμβάνοντας αυτό το μήνυμα από το κανάλι ανάδρασης, ο αποστολέας επαναλαμβάνει την αποστολή του συγκεκριμένου τμήματος.

Σε ένα κανάλι απαλοιφής μεταξύ ενός αποστολέα και ενός παραλήπτη, το πρόβλημα με την αποστολή τμημάτων ενός αρχείου, από τα οποία κάποιο μπορεί να μη φτάσει στον προορισμό του, μπορεί να μην είναι απόλυτα εμφανές. Αυτό διότι στην περίπτωση αποτυχίας κάποιου τμήματος, αυτό διορθώνεται με την αποστολή μηνύματος από τον παραλήπτη, μέσω του καναλιού ανάδρασης, στον αποστολέα για την αποτυχία και την αποστολή από τον τελευταίο ξανά του επίμαχου τμήματος. Το πρόβλημα, ωστόσο, διογκώνεται στην περίπτωση που έχουμε έναν (ή λίγους) αποστολέα και πολλούς παραλήπτες. Την κατάσταση αυτή τη συναντάμε όταν έχουμε για παράδειγμα ένα δίκτυο στο οποίο στέλνεται ένα αρχείο από ένα κόμβο σε πολλούς άλλους κόμβους.



Σχήμα 2.1: Δίκτυο σύνδεσης όπου οι κόμβοι συνδέονται με κανάλι απαλοιφής και ανάδρασης

Κατά την αποστολή ενός αρχείου από τον αποστολέα στους κόμβους με τους οποίους συνδέεται, ενδέχεται κάποια τμήματα να αποτύχουν να φτάσουν στους παραλήπτες

τους, είτε λόγω αποτυχίας κάποιου κόμβου (σε ένα δίκτυο ομότιμων), είτε εξαιτίας της ύπαρξης θορύβου αν το κανάλι είναι ευαίσθητο στο θόρυβο, είτε σε άλλους λόγους. Στην περίπτωση αυτή όλοι οι κόμβοι με τους οποίους συνδέεται ο αποστολέας θα στείλουν μέσω του καναλιού ανάδρασης μηνύματα που να ζητούν τα χαμένα τμήματα. Αυτό έχει ως αποτέλεσμα ο αποστολέας να λάβει πολύ μεγάλο αριθμό μηνυμάτων και να πρέπει να στείλει πολλά διαφορετικά τμήματα του αρχείου στους κόμβους που τα ζήτησαν. Ο φόρτος με τον οποίο επιβαρύνεται ο συγκεκριμένος κόμβος είναι μεγάλος, κάτι που μπορεί να καταλήξει ως και στην αδυναμία του να αποστείλει το αρχείο ολοκληρωμένο στους γείτονές του.

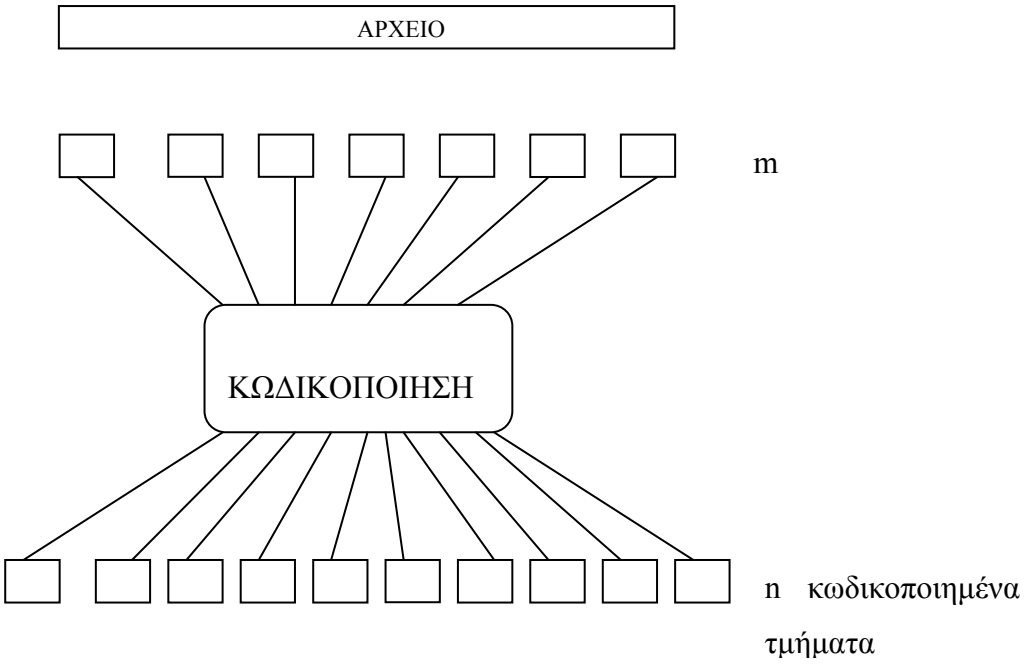
Μία άλλη περίπτωση όπου μπορεί να υπάρξει μεγάλο πρόβλημα κατά τη μετάδοση δεδομένων μέσω ενός καναλιού απαλοιφής, είναι η περίπτωση όπου δεν υπάρχει κανάλι ανάδρασης από τον παραλήπτη στον αποστολέα. Για παράδειγμα, όταν έχουμε πολλά αυτοκίνητα τα οποία είναι εφοδιασμένα με γεωγραφικά πληροφοριακά συστήματα (GPS) στα οποία ένας αναμεταδότης (δορυφόρος) στέλνει διάφορους χάρτες ανάλογα με την περιοχή όπου βρίσκονται τα αυτοκίνητα. Η κατάσταση αυτή είναι ιδιαίτερα δυναμική, καθώς τα αυτοκίνητα μπορεί να περάσουν μέσω ενός τούνελ, ή να φτάσουν σε κάποια περιοχή όπου η ισχύς του σήματος από το δορυφόρο να είναι μικρή. Αυτό έχει ως συνέπεια να μην σταλούν ακέραιοι οι χάρτες στον προορισμό τους. Επιπλέον, δεν υπάρχει κάποιο κανάλι μέσω του οποίου τα αυτοκίνητα να στείλουν μήνυμα στο δορυφόρο για το τμήμα που δεν έχουν λάβει επιτυχώς. Ο δορυφόρος, θα πρέπει, λοιπόν, να μεταδώσει ξανά ολόκληρο το χάρτη, έτσι ώστε το αυτοκίνητο να λάβει το τμήμα που επιθυμεί. Αυτό, βέβαια, έχει επίπτωση στον όγκο των περιττών δεδομένων που αποστέλλονται στους παραλήπτες.

Αυτό που ζητούμε στα κανάλια απαλοιφής, για την αποφυγή ανεπιθύμητων καταστάσεων όπως οι παραπάνω, είναι η ύπαρξη μιας αξιόπιστης και σταθερής κωδικοποίησης των δεδομένων που στέλνονται μέσω των καναλιών αυτών. Η κωδικοποίηση αυτή επιτυγχάνεται μέσω των αποκαλούμενων σχημάτων Προώθησης Διόρθωσης Σφαλμάτων (Forward Error Correction schemes – FEC). Με τα FEC σχήματα, τα δεδομένα υφίστανται επεξεργασία πριν τη μετάδοσή τους, με στόχο να δημιουργήσουν επιπρόσθετη πληροφορία με τη μορφή κωδικοποιημένων συμβόλων. Αυτή η επιπλέον πληροφορία στη συνέχεια μεταδίδεται μαζί με τα αρχικά δεδομένα

και δίνει τη δυνατότητα στους παραλήπτες να ανακτήσουν πιθανά χαμένα δεδομένα χωρίς την ανάγκη επανάληψης της μετάδοσης. Όταν τα κωδικοποιημένα σύμβολα φτάσουν στον παραλήπτη, τότε υφίστανται μια διαδικασία αποκωδικοποίησης ώστε να μας δώσουν τα αρχικά δεδομένα που κωδικοποίησαν. Γενικά, όσα περισσότερα κωδικοποιημένα σύμβολα δημιουργηθούν για ένα αρχείο, τόσο μεγαλύτερη είναι και η ασφάλεια για την πλήρη μεταφορά του. Ωστόσο, θα πρέπει να υπάρχει κάποιο όριο στο πλήθος των συμβόλων αυτών, ώστε να μην επιφορτίζεται το κανάλι επικοινωνίας με την αποστολή μεγάλου όγκου περιττών δεδομένων [1] [2].

2.2 Κωδικοποίηση Απαλοιφής

Με τη μέθοδο της Κωδικοποίησης Απαλοιφής (Erasure Coding) [3][4][5] υλοποιείται ικανοποιητικά ο στόχος των FEC σχημάτων στα κανάλια απαλοιφής. Με τη μέθοδο αυτή κάθε αρχείο χωρίζεται αρχικά σε m (ίσα) τμήματα και στη συνέχεια κωδικοποιείται σε n (n>m) νέα τμήματα. Ο παράγοντας επιβάρυνσης στο κανάλι επικοινωνίας από την κωδικοποίηση αυτή είναι n-m. Το βασικό χαρακτηριστικό της μεθόδου αυτής είναι ότι μπορούμε να κατασκευάσουμε το αρχικό αρχείο λαμβάνοντας οποιαδήποτε m από τα n τμήματα που υπάρχουν. Η Κωδικοποίηση Απαλοιφής δίνει μια νέα ισχύ στη μετάδοση των δεδομένων, καθώς διαφορετικοί παραλήπτες μπορούν να ανακτήσουν το αρχείο που τους αποστέλλεται λαμβάνοντας m διαφορετικά τμήματα ο καθένας από τα n, συνολικά, που μεταδίδονται.



Σχήμα 2.2: Μέθοδος Κωδικοποίησης Απαλοιφής

Χρησιμοποιώντας την Κωδικοποίηση Απαλοιφής σε ένα αρχείο (Σχήμα 2.2), ο αποστολέας στέλνει τα n κωδικοποιημένα τμήματα που συνθέτουν το αρχείο στους παραλήπτες. Η επιπλέον πληροφορία που μεταδίδεται στο κανάλι είναι n-m τμήματα. Οι παραλήπτες αρκεί να λάβουν m από τα n τμήματα και αφού γίνει η αποκωδικοποίηση των τμημάτων, αποκτούν το αρχείο. Με τον τρόπο αυτό εξαλείφεται το πρόβλημα κατά το οποίο πολλοί παραλήπτες μπορεί να έχουν "χάσει" κάποια μεταδιδόμενα τμήματα. Λαμβάνοντας κάποιο άλλο τμήμα, αντισταθμίζεται η απώλεια του χαμένου πακέτου. Επομένως, δε χρειάζεται ούτε η αποστολή μηνύματος από κάθε παραλήπτη για κάποιο τμήμα που χάθηκε, ούτε η αποστολή διαφορετικών τμημάτων από τον αποστολέα. Επίσης, περιορίζεται (ή και δεν απαιτείται) η χρήση των καναλιών ανάδρασης, διότι δε χρειάζεται ο παραλήπτης να στείλει μήνυμα απώλειας ενός συγκεκριμένου τμήματος που δε λήφθηκε, απλώς λαμβάνει ένα ακόμη τμήμα έως ότου συμπληρωθεί το απαιτούμενο πλήθος για την ανακατασκευή του αρχείου.

Υπάρχουν διάφορες τεχνικές που υλοποιούν τη μέθοδο της Κωδικοποίησης Απαλοιφής στα δεδομένα. Οι τεχνικές αυτές μπορούν να ομαδοποιηθούν σε δύο κατηγορίες, τους Μπλοκ Κώδικες Απαλοιφής (Block Erasure Codes) και τους Κώδικες Πηγής (Fountain Codes).

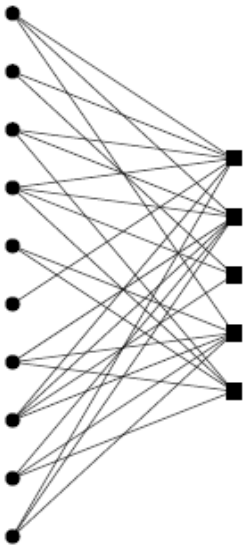
2.2.1 Μπλοκ Κώδικες Απαλοιφής

Οι Μπλοκ Κώδικες Απαλοιφής (Block Erasure Codes) [3] αποτελούν την πρώτη προσέγγιση στην Κωδικοποίηση Απαλοιφής των δεδομένων. Στους κώδικες αυτούς, το περιεχόμενο ενός αρχείου χωρίζεται αρχικά σε μία αλληλουχία συμβόλων ($s_1,s_2,\ldots,s_m$) τα οποία ονομάζονται σύμβολα εισόδου. Από τα σύμβολα εισόδου, ένας κωδικοποιητής παράγει μια σειρά ($t_1,t_2,\ldots,t_n$) κωδικοποιημένων συμβόλων. Ο Μπλοκ Κώδικας Απαλοιφής που κωδικοποιεί ένα αρχείο που χωρίζεται σε m τμήματα και στη συνέχεια κωδικοποιείται σε n κωδικοποιημένα τμήματα, αναφέρεται και ως (k, n) Κώδικας Απαλοιφής. Στους κώδικες αυτούς υπάρχει η έννοια του ρυθμού κωδικοποίησης, ο οποίος ορίζεται ως η ποσότητα $r = \frac{m}{n}$. Για παράδειγμα, ένας Μπλοκ Κώδικας Απαλοιφής ρυθμού $\frac{1}{4}$ χωρίζει ένα αρχείο σε 16 τμήματα και το κωδικοποιεί σε 64 κωδικοποιημένα τμήματα. Το κλειδί επιτυχίας της κωδικοποίησης έγκειται στην ικανότητα να παράγουμε το αρχείο λαμβάνοντας οποιαδήποτε k από τα n κωδικοποιημένα τμήματα.

Ένας από τους βασικούς αντιπροσώπους των Μπλοκ Κωδικών Απαλοιφής είναι οι Reed Solomon (RS) Κώδικες [3]. Οι RS Κώδικες χρησιμοποιούν στοιχεία γραμμικής άλγεβρας για την κωδικοποίηση των τμημάτων ενός αρχείου και την αποκωδικοποίησή τους. Τα τμήματα του αρχείου σχηματίζουν ένα πολυώνυμο και τα κωδικοποιημένα τμήματα σχηματίζονται πολλαπλασιάζοντας κάθε παράγοντα του πολυωνύμου με ένα πίνακα κατανομής B. Αντίστροφα, κατά την αποκωδικοποίηση, τα κωδικοποιημένα τμήματα που λαμβάνονται, πολλαπλασιάζονται με τον αντίστροφο του B ώστε να παραχθούν τα αρχικά τμήματα. Είναι προφανές ότι για την κωδικοποίηση και αποκωδικοποίηση απαιτούνται αρκετές πράξεις εσωτερικού γινομένου (πολλαπλασιασμοί και προσθέσεις στοιχείων). Χαρακτηριστικό τους είναι ότι χρειάζονται ακριβώς k τμήματα για να συνθέσουν το αρχικό αρχείο, η επιτυχία τους δηλαδή αγγίζει το ιδανικό. Ωστόσο, τόσο ο χρόνος κωδικοποίησης των τμημάτων του αρχείου, όσο και ο χρόνος αποκωδικοποίησης είναι αρκετά μεγάλοι και αυξάνονται κατά πολύ όσο αυξάνονται τα τμήματα που κωδικοποιούνται. Επίσης, το

κόστος σε πράξεις που απαιτείται μπορεί να είναι ασύμφορα μεγάλο. Συγκεκριμένα, με γραμμική αύξηση των τμημάτων του αρχείου προς κωδικοποίηση, το κόστος των υπολογισμών αυξάνεται εκθετικά. Το γεγονός αυτό τους καθιστά ανεπιθύμητους στην περίπτωση που σε κάποιο δίκτυο διακινούνται μεγάλου μεγέθους αρχεία, υπονοώντας σχετικά μεγάλο αριθμό τμημάτων, ή σε δίκτυα όπου ο χρόνος αποστολής και λήψης παίζει ουσιαστικό ρόλο, όπως για παράδειγμα στα ασύρματα δίκτυα.

Ένα άλλο είδος Μπλοκ Κωδικών Απαλοιφής είναι αυτό που προτάθηκε από τον R. Gallager το 1960 και βασίζεται στους διμερείς γράφους (bipartite graphs)[2] [7] [8]. Ένας διμερής γράφος φαίνεται στο Σχήμα 2.3. Στους γράφους αυτούς τα τμήματα του αρχείου ενώνονται με τα κωδικοποιημένα τμήματα μέσω κάποιων ακμών. Η τιμή κάθε κωδικοποιημένου τμήματος ισούται με το λογικό XOR των τμημάτων του αρχείου με τα οποία συνδέεται. Ο γράφος αυτός μπορεί να αναπαρασταθεί με έναν πίνακα, κατά αντιστοιχία με τον πίνακα κατανομής στην Reed Solomon μέθοδο, θέτοντας το στοιχείο (i,j) = 1 αν το i-οστό κωδικοποιημένο τμήμα ενώνεται με το j-οστό τμήμα δεδομένων ή (i, j)= 0 στην αντίθετη περίπτωση.



Σχήμα 2.3: Διμερής γράφος. Αριστερά βρίσκονται τα τμήματα δεδομένων και δεξιά τα κωδικοποιημένα τμήματα

Οι Κώδικες αυτοί ονομάζονται (Low Density Parity Check – LDPC) [6] [7] [8]. Οι Κώδικες αυτοί διαφέρουν σε σχέση με τους Reed Solomon Κώδικες στο γεγονός ότι πιθανώς να χρειάζονται λίγα περισσότερα από k τμήματα για να παράγουμε τα k τμήματα του αρχικού δεδομένου. Επιπλέον, οι χρόνοι κωδικοποίησης και

αποκωδικοποίησης είναι πολύ καλύτεροι από τους αντίστοιχους της RS μεθόδου. Οι LDPC Κώδικες μοιάζουν πολύ με κάποιους Κώδικες Πηγής που θα περιγράψουμε παρακάτω. Ωστόσο, το γεγονός που τους κατατάσσει στους Μπλοκ Κώδικες Απαλοιφής είναι το ότι πρέπει να καθοριστεί εκ των προτέρων το πλήθος των κωδικοποιημένων τμημάτων που θα παραχθούν. Ο ρυθμός κωδικοποίησης παραμένει σταθερός και δεν μπορεί να επεκταθεί ανάλογα με τις συνθήκες.

Δυστυχώς, τα μειονεκτήματα των Μπλοκ Κωδικών Απαλοιφής είναι αρκετά σημαντικά στην εφαρμογή τους στα Κανάλια Απαλοιφής. Το μοντέλο ενός μοναδικού καναλιού επικοινωνίας δεν είναι επαρκές όταν δεδομένα πρέπει να αποσταλούν ταυτόχρονα από έναν αποστολέα σε πολλούς παραλήπτες. Στην περίπτωση αυτή τα Κανάλια Απαλοιφής από τον αποστολέα σε κάθε παραλήπτη πιθανόν να έχουν διαφορετική πιθανότητα αντιμετώπισης σφαλμάτων. Σε κάποια κανάλια, δηλαδή, τα πακέτα ενδέχεται να φτάνουν με μεγαλύτερη δυσκολία στους παραλήπτες, πιθανώς λόγω θορύβου. Είναι δύσκολο ο αποστολέας να γνωρίζει τους ρυθμούς αποτυχιών σε δυναμικά κανάλια, π.χ. ασύρματα δίκτυα, όπου μεμονωμένοι παραλήπτες υφίστανται ξαφνικές αποτυχίες στη λήψη πακέτων. Για το λόγο αυτό ο αποστολέας οφείλει να επιλέξει τέτοιο ρυθμό κωδικοποίησης, δηλαδή την παραγωγή κωδικοποιημένων τμημάτων που να ικανοποιούν τη χειρότερη κατάσταση. Αυτό όμως επιβαρύνει ιδιαίτερα το δίκτυο αν ο ρυθμός αποτυχιών είναι μικρότερος από τον εκτιμώμενο, ή δυσκολεύει την αξιόπιστη μετάδοση των δεδομένων αν ο πραγματικός ρυθμός αποτυχιών είναι ακόμη μεγαλύτερος από αυτόν που έχει θεωρηθεί ότι υπάρχει στα κανάλια.

2.2.2 Κώδικες Πηγής

Οι Κώδικες Πηγής (Fountain Codes) [9] [10] είναι Κώδικες Απαλοιφής που σχεδιάστηκαν ώστε να επιλύουν κάποια από τα προβλήματα που ανακύπτουν από τους Μπλοκ Κώδικες Απαλοιφής. Προτάθηκαν από τον Bayers et al. το 1998 και περιγράφουν έναν αποδοτικό τρόπο μεταφοράς δεδομένων σε Κανάλια Απαλοιφών. Χαρακτηριστικό των Κωδικών αυτών είναι το ότι μπορούν, θεωρητικά, να παράγουν απεριόριστο πλήθος κωδικοποιημένων τμημάτων για κάποιο αρχείο. Μάλιστα, αυτή η διαδικασία μπορεί να γίνει κατά τη διάρκεια της κωδικοποίησης των τμημάτων του

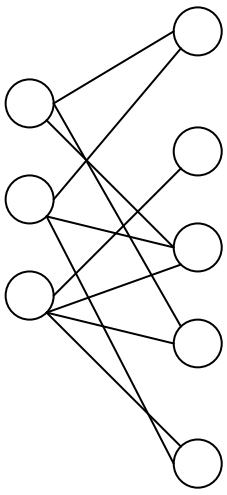
αρχείου ή ακόμη και κατά τη διάρκεια αποστολής των τμημάτων αυτών στον παραλήπτη τους. Δεν είναι απαραίτητο να προκαθοριστεί κατά την κωδικοποίηση ένα σταθερό πλήθος των κωδικοποιημένων τμημάτων, όπως συμβαίνει στους LDPC κώδικες, αλλά μπορεί να προσαρμοστεί στους ρυθμούς αποτυχιών του εκάστοτε συστήματος αποθήκευσης όπου χρησιμοποιούνται οι συγκεκριμένοι Κώδικες. Αν, δηλαδή, το σύστημα δεν είναι ιδιαίτερα σταθερό (έχουμε πολλές αποτυχίες στους κόμβους που αποθηκεύουν τα δεδομένα), χρειάζεται να παραχθούν περισσότερα τμήματα, από ότι σε ένα σταθερό και αξιόπιστο σύστημα. Οι Κώδικες Πηγής πήραν το συγκεκριμένο όνομα, καθώς παρομοιάζονται με μια πηγή που παράγει απεριόριστο πλήθος από σταγόνες. Αν έχουμε ένα ποτήρι το οποίο γεμίσει με m σταγόνες, τότε αρκεί να το τοποθετήσουμε κάτω από την πηγή μέχρι να μπουν σε αυτό m σταγόνες. Κάποιες σταγόνες μπορεί να πέσουν έξω από το ποτήρι αλλά δεν έχει σημασία, καθώς το ποτήρι χρειάζεται οποιεσδήποτε σταγόνες πλήθους m για να γεμίσει (Σχήμα 2.4).

Σχήμα 2.4: Η Κωδικοποίηση Απαλοιφής με Κώδικες Πηγής, μοιάζει με το πρόβλημα γεμίσματος ενός ποτηριού από μια πηγή.

Η Κωδικοποίηση Απαλοιφής με τη χρήση των Κωδικών Πηγής είναι παρόμοια με τη διαδικασία γεμίσματος του ποτηριού. Κατά την κωδικοποίηση, παράγονται αυθαίρετου πλήθους κωδικοποιημένα τμήματα από τα m τμήματα ενός αρχείου. Τα τμήματα αυτά μοιάζουν με τις σταγόνες της πηγής, είναι απεριόριστου πλήθους και δεν διαφέρουν στη σημασία ή τη σειρά με την οποία θα φτάσουν στον προορισμό τους. Ο παραλήπτης επιθυμεί να ανακτήσει ολόκληρο το αρχείο που επιθυμεί, όπως το ποτήρι επιθυμεί να γεμίσει. Δεν έχει σημασία ποια κωδικοποιημένα τμήματα θα φτάσουν τελικά, αρκεί να είναι πλήθους m, ίσως και λίγο περισσότερα όπως θα αναλυθεί παρακάτω. Αυτό αρκεί για να μπορέσει ο παραλήπτης να λάβει το αρχείο.

Οι Κώδικες Πηγής στηρίζονται στην ιδέα των *Αραιών Γράφων* (Sparse Graphs) [10] και ικανοποιούν πολλές από τις ιδιότητες που ζητούμε για μια αξιόλογη Κωδικοποίηση Απαλοιφής. Ένα παράδειγμα αραιού γράφου που χρησιμοποιείται από τους Κώδικες Πηγής φαίνεται στην παρακάτω εικόνα.

Η ιδέα των αραιών γράφων είναι η εξής: χωρίζουμε το αρχείο σε m τμήματα, τα οποία αποτελούν τους κόμβους του γράφου που βρίσκονται στα αριστερά. Οι κόμβοι αυτοί περιέχουν τα δεδομένα του αρχείου και αποκαλούνται τμήματα δεδομένων ή κόμβοι δεδομένων. Στη δεξιά πλευρά του γράφου υπάρχουν κάποιοι επιπρόσθετοι (έστω πλήθους n) κόμβοι οι οποίοι περιέχουν δεδομένα ελέγχου για τα τμήματα του αρχείου. Οι κόμβοι αυτοί λέγονται τμήματα ελέγχου (ή κόμβοι ελέγχου). Οι κόμβοι δεδομένων και ελέγχου συνδέονται μεταξύ τους με ένα σύνολο ακμών. Το πλήθος των ακμών μπορεί να διαφέρει μεταξύ των κόμβων ελέγχου. Η ένωση ενός κόμβου ελέγχου με έναν, ή περισσότερους, κόμβο δεδομένων σημαίνει ότι ο συγκεκριμένος κόμβος ελέγχου κωδικοποιεί το συγκεκριμένο κόμβο δεδομένων.



Σχήμα 2.5: Ένας αραιός γράφος όπου οι $m_1 \ldots m_3$ είναι οι κόμβοι δεδομένων και οι $n_1 \ldots n_5$ είναι οι κόμβοι ελέγχου

Ορισμός 2. Ως βαθμό ενός κόμβου ελέγχου ονομάζουμε το πλήθος των ακμών που πρόσκεινται σε αυτόν, δηλαδή το πλήθος των κόμβων ελέγχου που κωδικοποιεί.

Στο Σχήμα 2.5, για παράδειγμα, ο βαθμός του κόμβου ελέγχου m_2 ισούται με τρία καθώς υπάρχουν τρεις ακμές που πρόσκεινται σε αυτόν.

Η τιμή κάθε κόμβου ελέγχου σχετίζεται με την τιμή των κόμβων δεδομένων με τους οποίους συνδέεται. Συγκεκριμένα, η τιμή κάθε κόμβου ελέγχου ισούται με το XOR των τιμών των κόμβων δεδομένων με τους οποίους συνδέεται. Για παράδειγμα, στην παραπάνω εικόνα η τιμή του κόμβου ελέγχου m_1 ισούται με $n_1 \oplus n_3 \oplus n_4$. Όσον αφορά το βαθμό του κάθε κόμβου, δηλαδή το πλήθος των κόμβων δεδομένων που κωδικοποιεί, αυτός εξαρτάται από μια συνάρτηση κατανομής (degree distribution). Η συνάρτηση κατανομής αναθέτει κάποια βάρη στο πλήθος των ακμών, δηλαδή το βαθμό, κάθε κόμβου. Μπορεί για παράδειγμα, ο βαθμός 1 να έχει μεγαλύτερο βάρος από το βαθμό 3. Αυτό σημαίνει ότι υπάρχει μεγαλύτερη πιθανότητα κάποιος κόμβος ελέγχου να έχει βαθμό 1 και μικρότερη να έχει βαθμό 3. Υπάρχουν διάφοροι Κώδικες Πηγής οι οποίοι είναι αρκετά ικανοποιητικοί για την επίτευξη της Κωδικοποίησης Απαλοιφής των δεδομένων. Από τους δημοφιλέστερους είναι οι Luby Transform Κώδικες (LT codes) [10] [11] [12] [13] [14]και πιο πρόσφατα οι Raptor Κώδικες [14] [15], οι οποίοι συνδυάζουν τους LDPC και LT Κώδικες για την υλοποίησή τους.

Πριν προχωρήσουμε στην διεξοδικότερη ανάλυση της μεθόδου Κωδικοποίησης Απαλοιφής που θα χρησιμοποιήσουμε, είναι σκόπιμο να αναφερθούμε στα πλεονεκτήματα και μειονεκτήματα των Κωδικών Πηγής. Οι Κώδικες Πηγής δεν έχουν κάποιο σταθερό ρυθμό παραγωγής κωδικοποιημένων τμημάτων (rateless), υπό την έννοια ότι μπορούν να παράγουν θεωρητικά απεριόριστο πλήθος από τμήματα ελέγχου για τα τμήματα ενός αρχείου. Αυτό ενδέχεται να είναι ιδιαίτερα εύχρηστο, καθώς μπορούν εύκολα να προσαρμόζονται στις ανάγκες για αξιοπιστία που επιβάλλουν οι ρυθμοί αποτυχιών στα εκάστοτε Κανάλια Απαλοιφών που χρησιμοποιούνται. Αυτό σημαίνει ότι σε ένα αξιόπιστο και σταθερό Κανάλι Απαλοιφής, μπορούν να παραχθούν σχετικά λίγα κωδικοποιημένα τμήματα, καθώς είναι πολύ πιθανό να φτάσουν επαρκή τμήματα στους παραλήπτες. Αν, πάλι, το κανάλι μετάδοσης παρουσιάζει μεγάλο ρυθμό σφαλμάτων, τότε ο Κώδικας Πηγής που χρησιμοποιείται μπορεί να αυξήσει τα

παραγόμενα τμήματα ελέγχου, ώστε να μεγαλώσει η πιθανότητα να φτάσουν τα απαιτούμενα σε πλήθος τμήματα για την επιτυχή λήψη του αρχείου που εστάλη. Επιπλέον, μπορούν να προσαρμοστούν σε ένα δυναμικό Κανάλι Απαλοιφής, στο οποίο για κάποιο χρονικό διάστημα ο ρυθμός σφαλμάτων είναι μικρός, ενώ σε κάποιο άλλο χρονικό διάστημα αυξάνεται. Την προσαρμογή αυτή την παρέχει η ικανότητα των Κωδικών Πηγής να παράγουν κωδικοποιημένα τμήματα κατά την αποστολή των τμημάτων (on the fly). Αυτή είναι και η βασική διαφορά για την οποία οι LDPC Κώδικες δεν ανήκουν στην κατηγορία των Κωδικών Πηγής, παρά το γεγονός ότι στηρίζονται στην ιδέα της κωδικοποίησης μέσω γράφου.

Ωστόσο, ένα μειονέκτημα των μεθόδων αυτών είναι ότι περισσότερο βασίζονται σε πιθανοτικούς αλγορίθμους για την κωδικοποίηση και αποκωδικοποίηση. Δηλαδή, δεν εγγυώνται ότι αν χωρίσουμε ένα αρχείο σε m τμήματα και το κωδικοποιήσουμε σε νέα n τμήματα, τότε οποιαδήποτε m από τα n τμήματα θα μας δώσουν σίγουρα το αρχείο, όπως για παράδειγμα η Reed Solomon μέθοδος. Ωστόσο, αποδεικνύεται ότι με την κατάλληλη επιλογή της συνάρτησης κατανομής και με λίγο παραπάνω από n τμήματα, μπορούμε με πιθανότητα που ξεπερνά το 90% να ανακτήσουμε ακέραιο το αρχείο που επιθυμούμε [10].

2.3 Luby Transform (LT) Κώδικες

Στην παρούσα εργασία μελετούμε τη μέθοδο της Κωδικοποίησης Απαλοιφής σε δίκτυα ομότιμων. Όπως θα δούμε στο επόμενο κεφάλαιο, η Κωδικοποίησης Απαλοιφής βρίσκει χρήση και στην αποθήκευση δεδομένων σε κατανεμημένα συστήματα και μάλιστα με πολλά πλεονεκτήματα. Η μέθοδος Κωδικοποίησης Απαλοιφής που υλοποιήθηκε είναι οι αποκαλούμενοι Luby Transform Κώδικες (LT Codes). Οι LT Κώδικες προτάθηκαν από τον Michael Luby και την εταιρεία Digital Fountain το 2002 και αποτελούν την πρώτη ολοκληρωμένη και υλοποιήσιμη μέθοδο Κωδικών Πηγής που εφαρμόστηκε σε διάφορες περιπτώσεις. Οι λόγοι που οδήγησαν στην επιλογή της συγκεκριμένης μεθόδου περικλείονται κυρίως στις ιδιότητες των Κωδικών Πηγής και είναι οι ακόλουθοι.

Οι LT Κώδικες δεν απαιτούν τον καθορισμό από πριν του αριθμού των κωδικοποιημένων τμημάτων που θα παραχθούν για ένα αρχείο, δεν έχουν δηλαδή καθορισμένο ρυθμό κωδικοποίησης. Το γεγονός αυτό είναι αρκετά σημαντικό καθώς δεν επιβάλλει περιορισμούς και μπορεί να προσαρμόζεται στις εκάστοτε ανάγκες. Αν, για παράδειγμα, εφαρμόζονται σε ένα δυναμικό δίκτυο με μεγάλο ρυθμό αποτυχιών, μπορούν να επιλέξουν τη δημιουργία μεγάλου αριθμού κωδικοποιημένων τμημάτων. Το ίδιο μπορεί να συμβεί όταν θέλουμε να αποθηκεύσουμε ένα αρχείο σε ένα δίκτυο, π.χ. ομότιμων. Αν το αρχείο είναι ιδιαίτερα σημαντικό και δημοφιλές μπορούμε να αυξήσουμε την παραγωγή κωδικοποιημένων τμημάτων για μεγαλύτερη αξιοπιστία.

Μπορούν να παράγουν νέα κωδικοποιημένα τμήματα κατά τη διάρκεια της αποστολής τους (on the fly). Αυτό είναι επίσης σημαντικό, καθώς μπορούν να αλλάξουν το ρυθμό κωδικοποίησής τους για να αντιμετωπίσουν απρόβλεπτες καταστάσεις, που είναι συχνό φαινόμενο σε ένα δυναμικό δίκτυο όπως τα δίκτυα ομότιμων.

Οι αλγόριθμοι κωδικοποίησης και αποκωδικοποίησης είναι αρκετά κατανοητοί και υλοποιήσιμοι. Οι πράξεις που περιλαμβάνονται στην κωδικοποίηση και αποκωδικοποίηση δεν είναι πολύπλοκες ούτε ιδιαίτερα χρονοβόρες.

Οι LT Κώδικες είναι εύχρηστοι σε αρχεία τόσο μικρού όσο και μεγάλου μεγέθους. Το κάθε τμήμα δεδομένων μπορεί να κυμαίνεται από ένα bit έως k bits (όπου το k μπορεί να είναι αρκετά μεγάλο). Αυτό είναι ιδιαίτερα αποδοτικό, καθώς άλλοι Κώδικες (π.χ. Reed Solomon) επιφέρουν μεγάλο υπολογιστικό κόστος όσο αυξάνεται το μέγεθος του αρχείου που διαμοιράζεται. Στους LT Κώδικες η επιπρόσθετη πολυπλοκότητα είναι σαφώς μικρότερη.

Στη συνέχεια αναλύονται οι αλγόριθμοι Κωδικοποίησης και Αποκωδικοποίησης των LT Κωδικών, καθώς και οι παράμετροι που πρέπει να ληφθούν υπό όψη για την αύξηση της απόδοσής τους. Θεωρούμε ότι το αρχείο που υφίσταται την Κωδικοποίηση Απαλοιφής έχει χωριστεί σε k τμήματα $s_1,s_2,\ldots,s_k$ και κάθε τμήμα ελέγχου t_n που δημιουργείται κωδικοποιεί ένα υποσύνολο των τμημάτων δεδομένων.

Οι LT Κώδικες αποτελούν μια μέθοδο Κωδικών Πηγής και συνεπώς η κωδικοποίησή τους βασίζεται στους αραιούς γράφους, όπως είδαμε προηγουμένως. Τα κωδικοποιημένα τμήματα αποτελούν τους κόμβους ελέγχου και τα τμήματα στα οποία χωρίζεται το αρχείο αποτελούν τους κόμβους δεδομένων του γράφου. Κάθε κόμβος ελέγχου ενώνεται μέσω ακμών με έναν ή περισσότερους κόμβους δεδομένων. Οι κόμβοι δεδομένων που ενώνονται με κάποιο κόμβο ελέγχου, κωδικοποιούνται από αυτόν. Στον αλγόριθμο κωδικοποίησης (Σχήμα 2.6) αρχικά πρέπει να καθοριστεί ο βαθμός του κάθε κόμβου ελέγχου, δηλαδή το πλήθος των ακμών που πρόσκεινται σε αυτόν. Η επιλογή αυτή πραγματοποιείται μέσω μιας συνάρτησης κατανομής και ο κατάλληλος σχεδιασμός της αποτελεί, ίσως, το σημαντικότερο παράγοντα που επηρεάζει την απόδοση ολόκληρης της μεθόδου. Αφού επιλεγεί ο βαθμός για τον κόμβο ελέγχου, στη συνέχεια επιλέγουμε τυχαία και ομοιόμορφα τόσους κόμβους δεδομένων, όσους υποδεικνύει ο βαθμός. Οι κόμβοι δεδομένων που θα επιλεγούν ενώνονται με τον κόμβο ελέγχου και έτσι σχηματίζεται ο αραιός γράφος. Όσον αφορά την τιμή του κόμβου ελέγχου, ισούται με το λογικό XOR των τιμών των κόμβων δεδομένων με τους οποίους ενώνεται. Η τιμή αυτή αποτελεί και την κωδικοποίηση των συγκεκριμένων κόμβων δεδομένων.

Κάθε κωδικοποιημένο πακέτο t_n κωδικοποιείται από τα $s_1,s_2,\ldots,s_k$ αρχικά τμήματα του αρχείου ως εξής:

1. Τυχαία επέλεξε το βαθμό d_n του πακέτου t_n από μια συνάρτηση κατανομής ρ(d). Η επιλογή της κατάλληλης κατανομής εξαρτάται από το μέγεθος k του αρχείου.

2. Επέλεξε ομοιόμορφα και τυχαία d_n διαφορετικά πακέτα εισόδου (δεδομένων) και θέσε το t_n ίσο με το λογικό XOR των πακέτων εισόδου που επιλέχθηκαν.

Σχήμα 2.6 Αλγόριθμος 1: Αλγόριθμος κωδικοποίησης με βάση τους LT κώδικες

Είπαμε παραπάνω ότι πολύ σημαντικό ρόλο στην κωδικοποίηση και εν γένει σε ολόκληρη τη μέθοδο των LT Κωδικών παίζει η συνάρτηση κατανομής που

χρησιμοποιούμε. Σκοπός της συνάρτησης αυτής είναι να καθορίσει το πλήθος των ακμών κάθε κόμβου ελέγχου. Συγκεκριμένα, ανατίθεται σε κάθε πιθανό βαθμό ένα βάρος το οποίο καθορίζει ποιοι βαθμοί θα εμφανιστούν περισσότερες φορές στους κόμβους ελέγχου και ποιοι λιγότερες. Τα βάρη μπορούμε να τα θεωρήσουμε ως πιθανότητες να εμφανιστούν οι διάφοροι βαθμοί στον αραιό γράφο της κωδικοποίησης. Συγκεκριμένα, στους μικρότερους βαθμούς ανατίθενται μεγαλύτερα βάρη, δηλαδή η πιθανότητα να εμφανιστούν είναι μεγαλύτερη από τους μεγαλύτερους βαθμούς. Αυτή η διαδικασία είναι ιδιαίτερα σημαντική για την εν γένει ορθή λειτουργία της μεθόδου, διότι όπως θα δούμε στον αλγόριθμο 2, η έναρξη της αποκωδικοποίησης των τμημάτων γίνεται από ένα κωδικοποιημένο τμήμα το οποίο ενώνεται με ένα μόνο τμήμα δεδομένων. Τα κριτήρια που χρησιμοποιούμε για να επιλέξουμε την πιο κατάλληλη συνάρτηση κατανομής θα τα παρουσιάσουμε στη συνέχεια του κεφαλαίου.

Όσον αφορά το πλήθος των κωδικοποιημένων τμημάτων που θα δημιουργηθούν, μπορούμε να το επιλέξουμε εκ των προτέρων. Ωστόσο, αυτό δεν είναι δεσμευτικό καθώς μπορούμε να δημιουργήσουμε επιπλέον κόμβους ελέγχου αν το επιβάλλουν οι περιστάσεις. Συνήθως για κάθε τμήμα που είναι κάποιες δεκάδες ή εκατοντάδες bits σε μέγεθος, δημιουργούμε σχετικά μικρό πλήθος κωδικοποιημένων τμημάτων (π.χ. τέσσερα ή οχτώ). Βέβαια, όπως είπαμε, οι LT Κώδικες παρέχουν την ελευθερία να δημιουργήσουμε το πλήθος των κωδικοποιημένων τμημάτων που αρμόζουν στο συγκεκριμένο σύστημα στο οποίο χρησιμοποιούνται.

Όταν ο παραλήπτης λάβει επαρκή αριθμό από κωδικοποιημένα τμήματα, τότε εφαρμόζει μια διαδικασία αποκωδικοποίησης των τμημάτων αυτών, ώστε να λάβει τα τμήματα δεδομένων που συνθέτουν το αρχείο που επιθυμεί. Για να επιτύχει η αποκωδικοποίηση πρέπει να έχουν ληφθεί τουλάχιστον m κωδικοποιημένα τμήματα, αν θεωρήσουμε ότι το αρχείο έχει χωριστεί σε m τμήματα δεδομένων. Στους LT Κώδικες, ο αλγόριθμος αποκωδικοποίησης αναλαμβάνει τη διαδικασία αυτή και σχετίζεται άμεσα με τον τρόπο που έχουν κωδικοποιηθεί τα δεδομένα. Στην ουσία, ακολουθείται μια αντίστροφη πορεία από την κωδικοποίηση ώστε να ληφθούν τα τμήματα δεδομένων και συνεπώς το αρχείο.

Στους LT Κώδικες, για να λειτουργήσει η αποκωδικοποίηση των δεδομένων, πρέπει να είναι γνωστές κάποιες πληροφορίες για τα τους κόμβους ελέγχου εκ των προτέρων. Προφανώς, αυτές οι πληροφορίες προκύπτουν από τον αλγόριθμο κωδικοποίησης και συνεπώς είναι αποθηκευμένες στους κόμβους ελέγχου πριν ξεκινήσει η αποκωδικοποίηση. Πιο συγκεκριμένα, αυτές οι πληροφορίες που οφείλουμε να γνωρίζουμε πριν την εκτέλεση του αλγόριθμου αποκωδικοποίησης είναι οι εξής:

Ο βαθμός κάθε κόμβου ελέγχου, δηλαδή το πλήθος των ακμών που πρόσκεινται σε κάθε κόμβο στον αραιό γράφο.

Οι κόμβοι δεδομένων με τους οποίους είναι συνδεδεμένος κάθε κόμβος ελέγχου. Οφείλουμε, δηλαδή, να γνωρίζουμε ποιους συγκεκριμένους κόμβους δεδομένων κωδικοποιεί κάθε κόμβος ελέγχου.

Ο αλγόριθμος αποκωδικοποίησης των LT Κωδικών φαίνεται στον αλγόριθμο 2.

1. Βρες ένα κόμβο ελέγχου t_n ο οποίος είναι συνδεδεμένος με ένα μόνο κόμβο δεδομένων s_k (είναι δηλαδή βαθμού 1).
2. Θέσε $s_k=t_n$
3. Πρόσθεσε την τιμή του s_k σε όλους τους κόμβους t_n' που συνδέονται με τον s_k. Δηλαδή

$$t_n' = t_n' + s_k$$

4. Απομάκρυνε όλες τις ακμές που συνδέονται με τον κόμβο δεδομένων s_k

5. Επανέλαβε το (1) μέχρι όλα τα s_k να υπολογιστούν.

Σχήμα 2.7 Αλγόριθμος 2. Αλγόριθμος αποκωδικοποίησης με βάση τους LT κώδικες

Για να ξεκινήσει τη λειτουργία του ο αλγόριθμος αποκωδικοποίησης, είναι απαραίτητο να εντοπίζουμε αρχικά ένα κόμβο ελέγχου t_n ο οποίος να έχει βαθμό ίσο με ένα, δηλαδή να συνδέεται με ένα μόνο κόμβο δεδομένων, έστω s_k. Εφόσον ο t_n κωδικοποιεί

μόνο τον s_k, είναι προφανές από την κωδικοποίηση ότι οι τιμές τους είναι οι ίδιες, επομένως θέτουμε $s_k=t_n$. Με τον τρόπο αυτό αποκωδικοποιούμε τον κόμβο δεδομένων s_k. Υπολογίζοντας τον s_k, κοινοποιούμε την τιμή του στους υπόλοιπους κόμβους ελέγχου με τους οποίους ενδεχομένως ενώνεται. Αυτό το κάνουμε προσθέτοντας την τιμή του s_k στις τιμές όλων των κόμβων ελέγχου t_n' με τους οποίους συνδέεται στον αραιό γράφο. Επιπλέον, αφαιρούμε κάθε ακμή που συνδέει τον s_k με κάποιο κόμβο t_n'. Η διαδικασία αυτή γίνεται έτσι ώστε να γνωστοποιηθεί ότι οι κόμβοι ελέγχου δε χρειάζεται να προσπαθήσουν να αποκωδικοποιήσουν τον κόμβο δεδομένων s_k, αφού αυτός είναι ήδη γνωστός. Απομακρύνοντας τις ακμές που συνδέουν τον κόμβο δεδομένων που αποκωδικοποιήθηκε με τους t_n', αυτομάτως μειώνεται ο βαθμός των κόμβων ελέγχου t_n' κατά ένα. Για να αποκωδικοποιήσουμε τους υπόλοιπους κόμβους δεδομένων, εντοπίζουμε τον επόμενο κόμβο ελέγχου που προκύπτει ο οποίος έχει βαθμό ίσο με ένα. Βρίσκουμε, λοιπόν, τον κόμβο δεδομένων που αυτός κωδικοποιεί και ακολουθούμε ακριβώς την ίδια διαδικασία που περιγράφηκε προηγουμένως. Ο αλγόριθμος αποκωδικοποίησης ολοκληρώνεται με επιτυχία αν αποκωδικοποιηθούν όλοι οι κόμβοι δεδομένων που υπάρχουν στο γράφο. Αυτό σημαίνει ότι μπορούμε να παραλάβουμε το αρχείο χωρίς πρόβλημα. Εάν σε κάποιο βήμα του αλγορίθμου προκύψει η κατάσταση όπου δεν υπάρχει κάποιος κόμβος ελέγχου με βαθμό ίσο με ένα, τότε ο αλγόριθμος αποκωδικοποίησης δεν μπορεί να συνεχίσει, με αποτέλεσμα να αποτύχει. Αυτή είναι μια ανεπιθύμητη κατάσταση και συνεπώς θέλουμε σε κάθε βήμα του αλγορίθμου 2 να εμφανίζεται τμήμα με βαθμό 1 (το οποίο θα προκύψει από αφαίρεση ακμών κατά τα προηγούμενα βήματα).

Από την ανάλυση του αλγόριθμου αποκωδικοποίησης που προηγήθηκε, παρατηρούμε ότι καθοριστικό ρόλο στην επιτυχία του διαδραματίζει η ύπαρξη σε κάθε βήμα του ενός τουλάχιστον κόμβου ελέγχου με βαθμό ίσο με ένα. Αυτό καθορίζεται στον αλγόριθμο κωδικοποίησης κατά την ανάθεση βαθμού σε κάθε κόμβο ελέγχου από τη συνάρτηση κατανομής που χρησιμοποιείται. Συμπεραίνουμε, λοιπόν, ότι ο βασικός παράγοντας που διασφαλίζει την επιτυχία των LT Κωδικών είναι η επιλογή της κατάλληλης κατανομής που να αναθέτει με το σωστό τρόπο το βαθμό σε κάθε κόμβο ελέγχου. Εκτός όμως από την επιτυχή κατάληξη της αποκωδικοποίησης, σημαντικό στοιχείο στην επιθυμητή απόδοση της μεθόδου είναι και το πλήθος των κωδικοποιημένων τμημάτων που χρειαζόμαστε για να ανακτήσουμε το αρχείο. Στόχος

της κατανομής είναι επίσης η διασφάλιση ότι με επιλογή πλήθους κωδικοποιημένων τμημάτων όσο γίνεται πιο κοντά στο k (όπου k το πλήθος των τμημάτων δεδομένων στα οποία χωρίζεται το αρχείο) μπορούμε να αποκωδικοποιήσουμε το αρχείο με μεγάλη πιθανότητα.

2.3.1 Κατανομή Βαθμού LT Κωδικών

Μια σημαντική παράμετρος που παρατηρήθηκε από τον αλγόριθμο αποκωδικοποίησης που περιγράφηκε προηγουμένως, είναι η επιλογή της κατάλληλης κατανομής που να αναθέτει βαθμό σε κάθε κόμβο ελέγχου με τέτοιο τρόπο ώστε να μπορεί να ολοκληρωθεί επιτυχώς η αποκωδικοποίηση με μεγάλη πιθανότητα. Στην περίπτωση των LT Κωδικών, η κατανομή που επιλέγεται ανήκει σε μια συγκεκριμένη κατηγορία συναρτήσεων κατανομής που λέγεται Κατανομή Βαθμού (Degree Distribution). Η ονομασία των Κατανομών Βαθμού προέρχεται από τη λειτουργία που καλούνται να επιτελέσουν οι κατανομές αυτές, δηλαδή την ανάθεση βαθμών (βαρών) σε κάποια δεδομένα.

Ορισμός 3 (Κατανομή Βαθμού- Degree Distribution): Για κάθε ακέραιο αριθμό d, ορίζουμε ως Κατανομή Βαθμού ρ(d) την πιθανότητα ένα κωδικοποιημένο πακέτο να έχει βαθμό ίσο με d.

Όπως είπαμε παραπάνω, βασική παράμετρος για την απόδοση της μεθόδου των LT Κωδικών είναι η συνάρτηση κατανομής που χρησιμοποιείται για την επιλογή του βαθμού κάθε κωδικοποιημένου πακέτου. Η κατανομή αυτή είναι ουσιαστική καθώς τα κωδικοποιημένα τμήματα που θα παραχθούν πρέπει να ικανοποιούν την ιδιότητα ότι αν λάβουμε k τέτοια τμήματα (όπου k το πλήθος των αρχικών τμημάτων δεδομένων) θα πρέπει (με μεγάλη πιθανότητα) να μπορούμε να ανακτήσουμε το αρχείο.

Για την επιλογή της κατάλληλης Κατανομής Βαθμού πρέπει να ληφθούν υπ’ όψιν ορισμένοι παράγοντες:

Στόχος είναι όσο το δυνατόν λιγότερα πακέτα να κωδικοποιούν ολόκληρο το αρχείο. Αυτό σημαίνει ότι αν λάβουμε k (ή λίγα περισσότερα όπως θα δούμε) κωδικοποιημένα τμήματα, διασφαλίζεται με μεγάλη πιθανότητα η επιτυχία της μεθόδου.

Ο βαθμός κάθε κόμβου ελέγχου, δηλαδή το πλήθος των κόμβων δεδομένων με τους οποίους ενώνεται στο γράφο, απεικονίζει το πλήθος των πράξεων που απαιτούνται για την κωδικοποίησή του. Αυτό είναι προφανές διότι όσο μεγαλύτερος είναι ο βαθμός, τόσο περισσότερες πράξεις XOR χρειάζονται για τον υπολογισμό της τιμής κάθε κόμβου ελέγχου. Επιπλέον, το σύνολο των πράξεων που απαιτούνται για την αποκωδικοποίηση ισούται με το μέσο βαθμό των κόμβων ελέγχου επί το πλήθος των λαμβανόμενων κόμβων. Με βάση τα παραπάνω, αντιλαμβανόμαστε ότι ο βαθμός κάθε πακέτου πρέπει να είναι κατά το δυνατόν μικρότερος.

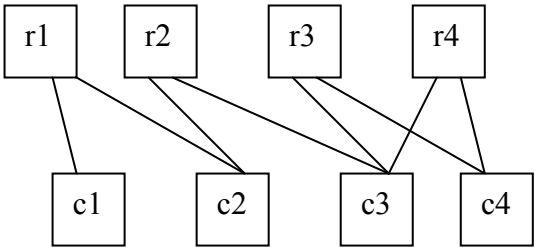
Σε κάθε βήμα του αλγορίθμου αποκωδικοποίησης πρέπει να υπάρχει (τουλάχιστον) ένα κωδικοποιημένο τμήμα το οποίο να ενώνεται με ένα μόνο άγνωστο κόμβο δεδομένων το οποίο και αποκωδικοποιείται. Αυτό υποδεικνύει ότι η κατανομή θα πρέπει να δίνει μεγαλύτερο βάρος στην επιλογή μικρού βαθμού και μικρότερη κατά την επιλογή μεγαλύτερου βαθμού. Ωστόσο, δεν θα πρέπει να ανατίθεται μηδενική πιθανότητα στην επιλογή του μεγαλύτερου βαθμού.

2.3.1.1 Ideal Soliton Κατανομή

Για την επιλογή της σωστής Κατανομής Βαθμού είναι σημαντικό να ισχύει το γεγονός ότι ο ρυθμός αποκωδικοποίησης των τμημάτων εισόδου είναι ανάλογος με το ρυθμό δημιουργίας κωδικοποιημένων τμημάτων βαθμού ένα που να πρόσκεινται σε τμήμα εισόδου που δεν έχει αποκωδικοποιηθεί ακόμη.

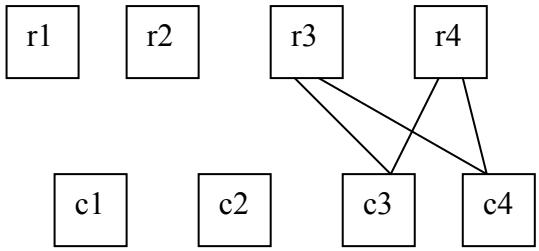
Για να γίνει πιο κατανοητή η παρατήρηση αυτή, ας θεωρήσουμε ότι έχουμε τον ακόλουθο γράφο (Σχήμα 2.8):

Παράδειγμα 1



Σχήμα 2.8 αραιός γράφος με τέσσερις κόμβους δεδομένων και τέσσερις κόμβους ελέγχου

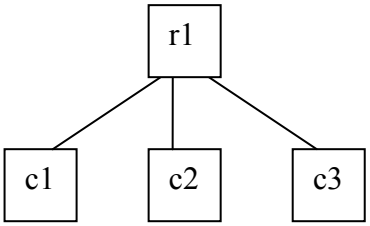
Τα r1…r4 είναι τα τμήματα δεδομένων του αρχείου και τα c1…c4 είναι τα κωδικοποιημένα τμήματα. Το c1 (βαθμού 1) αποκωδικοποιεί το r1 στο οποίο πρόσκειται. Αφού ανακτήθηκε με επιτυχία το r1, η ακμή (r1,c2) απομακρύνεται και συνεπώς προκύπτει ότι το c2 είναι τώρα βαθμού 1 και ενώνεται μόνο με το άγνωστο r2. Ομοίως, υπολογίζεται το r2 από το c2 και η ακμή (r2, c3) απομακρύνεται. Στο σημείο αυτό, όμως, προκύπτει ένα πρόβλημα. Τα c3 και c4 δεν είναι πλέον βαθμού 1 (ο βαθμός τους είναι 2), με αποτέλεσμα να μην μπορεί να συνεχίσει η αποκωδικοποίηση (Σχήμα 2.9). Ο ρυθμός, δηλαδή, αποκωδικοποίησης τμημάτων δεδομένων δεν είναι ίδιος με το ρυθμό δημιουργίας κωδικοποιημένων τμημάτων βαθμού 1. Αυτή είναι μια κατάσταση που δεν επιθυμούμε και έγκειται στην κατανομή βαθμού να διασφαλίσει την επιτυχή αποκωδικοποίηση.



Σχήμα 2.9 Ανεπιθύμητη κατάσταση κατά την αποκωδικοποίηση των κόμβων δεδομένων

Ακόμη, επιδίωξη μιας καλής κατανομής βαθμού είναι να μη δίνει τμήματα που να κωδικοποιούν το ίδιο τμήμα δεδομένου. Έχουμε, για παράδειγμα, το γράφο του Σχήματος 2.10.

Παράδειγμα 2



Σχήμα 2.10 Ένας κόμβος δεδομένων κωδικοποιείται από πολλαπλούς κόμβους ελέγχου

Οι κόμβοι ελέγχου c1…c3 κωδικοποιούν τον ίδιο κόμβο δεδομένων r1 (Σχήμα 2.10). Αυτό όμως μας οδηγεί σε περιττούς κόμβους ελέγχου, καθώς μόνο ένας από αυτούς χρειάζεται για να αποκωδικοποιήσει τον r1. Προφανώς, είναι βέλτιστο να αποφευχθεί μια τέτοια κατάσταση.

Μια Κατανομή Βαθμού που προτάθηκε, η οποία (στη θεωρία τουλάχιστον) έχει κάποιες από τις επιθυμητές ιδιότητες που ζητάμε, είναι η Ideal Soliton Κατανομή [10]. Η κατανομή αυτή έχει την μορφή που φαίνεται στην Εξίσωση (2.1)

:

$p(1)=1/k$

$$p(i)= \frac{1}{i*(i-1)} \quad , i = 2,\ldots,k \qquad \text{Εξ. 2.1}$$

όπου k είναι το πλήθος των τμημάτων εισόδου στα οποία χωρίζεται το αρχείο.

Δυστυχώς, αποδεικνύεται (πειραματικά) ότι χρησιμοποιώντας την Ideal Soliton Κατανομή στη μέθοδο των LT Κωδίκων, τα αποτελέσματα δεν είναι ικανοποιητικά. Η

απόδοση είναι αρκετά χαμηλή και χρειαζόμαστε πολύ περισσότερα από k κωδικοποιημένα τμήματα για να επιτύχουμε ικανοποιητικά αποτελέσματα.

Παρατηρώντας τη συμπεριφορά της κατανομής αυτής, βλέπουμε ότι όταν ένας κόμβος δεδομένου αποκωδικοποιείται, δημιουργείται ακριβώς ένας κόμβος ελέγχου βαθμού ένα, ο οποίος συνεχίζει και όχι περισσότεροι. Αυτό όμως εγκυμονεί την ύπαρξη της ανεπιθύμητης κατάστασης που είδαμε στο παράδειγμα 1. Δηλαδή, ενδέχεται να υπάρξει περίπτωση, μετά την αποκωδικοποίηση κάποιου κόμβου δεδομένων να σταματήσει ο αλγόριθμος διότι δεν υπάρχουν επαρκείς κόμβοι ελέγχου για την αποκωδικοποίηση. Επιθυμία μας είναι, μετά την αποκωδικοποίηση κάπου κόμβου δεδομένων, να υπάρχουν περισσότεροι κόμβοι ελέγχου υποψήφιοι για να συνεχίσουν. Σε ένα δυναμικό σύστημα (όπως τα Δίκτυα Ομότιμων) αυτή η επάρκεια είναι ουσιαστική για την αποφυγή του πρόωρου τερματισμού της διαδικασίας.

Τα συμπεράσματα αυτά προκύπτουν από το γεγονός ότι η Ideal Soliton κατανομή δεν ευνοεί την ύπαρξη τμήματος βαθμού ένα για μεγάλο πλήθος τμημάτων δεδομένων. Για παράδειγμα, αν το πλήθος των τμημάτων δεδομένων είναι 1000, η πιθανότητα δημιουργίας (αλγόριθμος 1) τμήματος ελέγχου βαθμού 1 είναι $\rho(1)=1/1000$. Αντιθέτως, για μεγαλύτερους βαθμούς, η κατανομή αυτή λειτουργεί ικανοποιητικά. Για βαθμό 2 η πιθανότητα είναι 0.5, για βαθμό 3 είναι 1/6 και μειώνεται σταδιακά για μεγαλύτερους βαθμούς. Επιθυμία μας κατά την δημιουργία των τμημάτων ελέγχου είναι τα τμήματα με μεγαλύτερο βαθμό, να μπορούν να προκύψουν από τμήματα με μικρότερους βαθμούς. Αν όμως η ύπαρξη τμήματος βαθμού ένα είναι τόσο μικρή (που αποτελεί και τη βάση έναρξης της αποκωδικοποίησης με βάση τον αλγόριθμο 2), τότε ενδέχεται να μην προχωρήσει η αποκωδικοποίηση των τμημάτων, μετά από κάποιο βήμα.

2.3.1.2 Robust Soliton Κατανομή

Η Ideal Soliton Κατανομή λειτουργεί ικανοποιητικά μόνο στην ιδανική περίπτωση όπου ένας κόμβος δεδομένων που αποκωδικοποιείται οδηγεί στην ύπαρξη ενός μόνο κόμβου ελέγχου που να συνεχίζει τη διαδικασία. Στην περίπτωση αυτή, επομένως,

αρκούν ακριβώς k κωδικοποιημένα τμήματα για την επιτυχή ανάκτηση του κατακερματισμένου αρχείου. Ωστόσο, αυτό δεν ισχύει συνήθως.

Η Robust Soliton Κατανομή [10] [16] [17] αποδεικνύεται πειραματικά ότι έχει πολύ καλύτερη συμπεριφορά από την Ideal Soliton Κατανομή. Η κατανομή αυτή εξασφαλίζει με μεγάλη πιθανότητα ότι αφού αποκωδικοποιηθεί ένας κόμβος δεδομένων σε κάποιο βήμα του αλγορίθμου, προκύπτουν πάνω από ένας κόμβοι ελέγχου βαθμού ένα που μπορούν να συνεχίσουν την αποκωδικοποίηση. Με λίγα παραπάνω από k τμήματα, αποδεικνύεται ότι με μεγάλη πιθανότητα (πάνω από 99%) μπορούμε να αποκωδικοποιήσουμε το αρχείο. Ακολουθεί η περιγραφή της Robust Soliton Κατανομής.

Ορισμός 4: Έστω δ η πιθανότητα αποτυχίας του αλγορίθμου αποκωδικοποίησης για ένα πλήθος K κόμβων ελέγχου. Η Robust Soliton Κατανομή μ() ορίζεται ως εξής [10]:

Έστω $R = c * \ln(\frac{k}{\delta}) * \sqrt{k}$, για κάποια σταθερά c>0

Ορίζουμε,

$$\tau(\mathrm{i}) = \begin{cases} \frac{R}{i*k}, i = 1,\ldots,\frac{k}{R}-1 \\ \frac{R*\ln(\frac{R}{\delta})}{k}, i = \frac{k}{R} \\ 0, i = \frac{k}{R}+1,\ldots,k \end{cases} \qquad \text{Εξ. 2.2}$$

όπου k το πλήθος τμημάτων δεδομένων

Προσθέτοντας στην τ(i) την Ideal Soliton Κατανομή p(i) και κανονικοποιώντας, προκύπτει η μ(i):

$$\mu(\mathrm{i}) = \frac{p(i) + \tau(\mathrm{i})}{\sum_{i=1}^{\mathrm{k}} p(i) + \tau(\mathrm{i})} \qquad \text{Εξ. 2.3}$$

Υπολογίζοντας την Robust Soliton Κατανομή μ(i) για συγκεκριμένο c και δ (Εξίσωση (2.2) και (2.3)), προκύπτει ένας πίνακας μεγέθους k, όπου το i-στό στοιχείο του πίνακα δείχνει την πιθανότητα το κωδικοποιημένο τμήμα το οποίο θα προκύψει κατά την αποκωδικοποίηση να έχει βαθμό i. Το άθροισμα των στοιχείων του πίνακα αυτού ισούται με τη μονάδα.

Η Robust Soliton κατανομή λειτουργεί ικανοποιητικότερα από την Ideal Soliton κατανομή, διότι ευνοεί την ύπαρξη κωδικοποιημένων τμημάτων ελέγχου με βαθμό ένα. Για μεγαλύτερους βαθμούς, προσεγγίζει την απόδοση της Ideal Soliton κατανομής, η οποία είναι ικανοποιητική. Για παράδειγμα, όταν το k=1000, τότε η πιθανότητα εμφάνισης βαθμού 1 είναι μ(1)=1/5. Για μεγαλύτερους βαθμούς, έχουμε μ(2)=1/10, μ(3)=1/15 και μειώνεται σταδιακά για μεγαλύτερους βαθμούς.

Με τη χρήση της Robust Soliton Κατανομής πετυχαίνουμε ικανοποιητική απόδοση του αλγορίθμου αν λάβουμε λίγο πάνω από k κωδικοποιημένα τμήματα. Για την επιβεβαίωση της καταλληλότητας της Robust Soliton Κατανομής στους LT Κώδικες, ελέγχουμε κάποιες μετρικές που είναι σημαντικές στην απόδοση της μεθόδου. Συγκεκριμένα ελέγχουμε το αναμενόμενο πλήθος των κόμβων ελέγχου που πρέπει να λάβουμε ώστε να αποκωδικοποιήσουμε το αρχείο που μας ενδιαφέρει. Επίσης, υπολογίζουμε το μέσο βαθμό των κόμβων ελέγχου που προκύπτουν κατά την κωδικοποίηση των κόμβων δεδομένων. Τέλος, υπολογίζουμε την πολυπλοκότητα του αλγόριθμου αποκωδικοποίησης των LT Κωδικών.

Α) Αναμενόμενο πλήθος κόμβων ελέγχου

Όπως είπαμε παραπάνω, το άθροισμα των στοιχείων του πίνακα της Robust Soliton Κατανομής ισούται με τη μονάδα, δηλαδή από την Εξίσωση (2.4):

$$\mu(1) + \mu(2) + \ldots + \mu(\mathrm{k}) = 1 \qquad \text{Εξ. 2.4}$$

Χρειαζόμαστε, επομένως, k* $(\mu(1)+\mu(2)+...+\mu(k))$ κωδικοποιημένα τμήματα για την επιτυχία του αλγορίθμου. Ο Luby στο [10] έπειτα από περαιτέρω ανάλυση της Εξ(2.4), κατέληξε στο συμπέρασμα για το μέγιστο πλήθος τμημάτων ελέγχου που μπορούν να παραχθούν, χωρίς να έχουμε ύπαρξη περιττών τμημάτων, δηλαδή δύο ή περισσοτέρων τμημάτων που να κωδικοποιούν τα ίδια τμήματα δεδομένων (Εξ(2.5)).

$$k + R*(\sum_{i=1}^{\frac{k}{R}-1}\frac{1}{i}+\ln(\frac{R}{\delta})) \le k+O(\sqrt{k}*\ln^2(\frac{k}{\delta})) \qquad \text{Εξ. 2.5}$$

Παρατηρούμε ότι η αύξηση αυτή εξαρτάται και από το ποσοστό αποτυχίας δ το οποίο θεωρούμε αποδεκτό. Γεγονός είναι, ωστόσο, ότι ο επιπλέον αριθμός κόμβων ελέγχου που χρειαζόμαστε για την αποκωδικοποίηση k κόμβων δεδομένων είναι ικανοποιητικά χαμηλός.

Β) Μέσος Βαθμός κόμβων ελέγχου

Ο μέσος βαθμός των κόμβων ελέγχου μας δίνει μια εκτίμηση για την πολυπλοκότητα των αλγορίθμων κωδικοποίησης και αποκωδικοποίησης, καθώς σχετίζεται με το πλήθος των πράξεων που απαιτούνται τόσο για τη δημιουργία των κόμβων ελέγχου, όσο και για την αποκωδικοποίηση των κόμβων δεδομένων. Το γεγονός αυτό ισχύει διότι ο βαθμός κάθε κόμβου ελέγχου καθορίζει και το πλήθος των πράξεων XOR που απαιτούνται για τον υπολογισμό της τιμής του από τους κόμβους δεδομένων που κωδικοποιεί. Στο [10] η ανάλυση που γίνεται όσον αφορά το μέσο βαθμό D των κόμβων ελέγχου που παράγονται, καταλήγει στην εξίσωση Εξ(2.6) :

$$D \le \ln(k)+1+\ln(\frac{R}{\delta}) \qquad \text{Εξ. 2.6}$$

Προκύπτει, συνεπώς, ότι ο μέσος βαθμός των κόμβων ελέγχου που παράγει η Robust Soliton Κατανομή είναι $O(\ln(\frac{k}{\delta}))$. Αυτή είναι και η ιδανική τιμή που διασφαλίζει ότι τα k τμήματα δεδομένων κωδικοποιούνται με πιθανότητα 1-δ, χωρίς κάποιο να λείπει από την κωδικοποίηση.

Γ) Πολυπλοκότητα Αλγορίθμου Κωδικοποίησης/ Αποκωδικοποίησης

Είδαμε παραπάνω ότι ο μέσος βαθμός των κόμβων ελέγχου του γράφου είναι της τάξης $O(\ln(\frac{k}{\delta}))$. Εάν θεωρήσουμε ότι δημιουργούμε K κόμβους ελέγχου για την κωδικοποίηση, τότε είναι προφανές ότι χρειαζόμαστε $K * \ln(\frac{k}{\delta})$ πράξεις XOR για να υπολογίσουμε τις τιμές τους. Αυτό είναι λογικό διότι ο βαθμός κάθε κόμβου ελέγχου υποδηλώνει το πλήθος των κόμβων δεδομένων που αυτός κωδικοποιεί και, συνεπώς, το πόσες πράξεις XOR πρέπει να γίνουν για να υπολογιστεί η τιμή του. Το ίδιο πλήθος πράξεων χρειάζεται και για την αποκωδικοποίηση, καθώς στην ουσία πρόκειται για μια αντίστροφή διαδικασία από αυτή της κωδικοποίησης. Προκύπτει, επομένως, ότι η πολυπλοκότητα των αλγορίθμων είναι $O(k * \ln(\frac{k}{\delta}))$ πράξεις. Η πράξη της κωδικοποίησης μοιάζει με το πρόβλημα ρίψης μπαλών σε k κάδους. Για να διασφαλίσουμε ότι, εφόσον ρίχνουμε τυχαία και ομοιόμορφα μπάλες στους κάδους, δεν θα υπάρχει άδειος κάδος χρειαζόμαστε μπάλες της τάξης $O(k * \ln(k))$ τουλάχιστον. Αν, λοιπόν, θεωρήσουμε ότι κάθε κάδος αντιστοιχεί με ένα κόμβο δεδομένων και κάθε μπάλα με μία ακμή στο γράφο, τότε είναι κατανοητό ότι πρέπει να υπάρχει τουλάχιστον μία ακμή που να αντιστοιχίζεται σε κάθε κόμβο δεδομένων. Επομένως, χρειαζόμαστε $O(k * \ln(k))$ ακμές (και άρα πράξεις) για να κωδικοποιήσουμε k κόμβους δεδομένων. Βλέπουμε, δηλαδή, ότι η Robust Soliton Κατανομή καταλήγει στις αναμενόμενες βέλτιστες τιμές όσον αφορά στην πολυπλοκότητα των αλγορίθμων των LT Κωδικών. Η ύπαρξη του δ στην πολυπλοκότητα δηλώνει ότι η απόκλιση της

τάξης των πράξεων μπορεί να είναι το πολύ δ, δηλαδή ότι θα έχουμε πολυπλοκότητα $O(k * \ln(k))$ με πιθανότητα 1-δ.

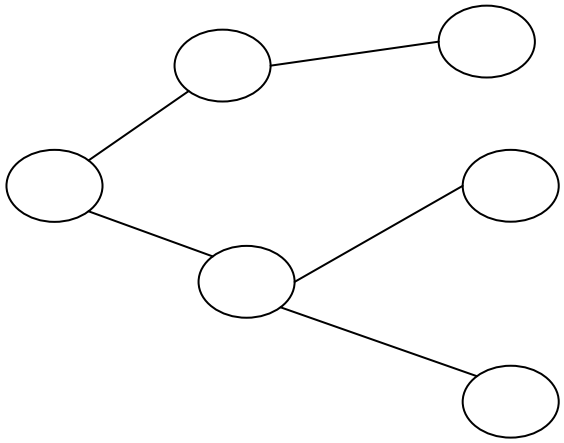
ΚΕΦΑΛΑΙΟ 3. ΚΩΔΙΚΟΠΟΙΗΣΗ ΑΠΑΛΟΙΦΗΣ ΣΕ ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

3.1 Κωδικοποίηση Απαλοιφής Σε Δίκτυα Ομότιμων

Παραδοσιακά, η απόκτηση της πληροφορίας μέσω του Διαδικτύου γινόταν μέσω του μοντέλου "πελάτη-εξυπηρετητή"(client-server). Με βάση το μοντέλο αυτό, τα δεδομένα βρίσκονται αποθηκευμένα σε ένα κεντρικό υπολογιστή που ονομάζεται εξυπηρετητής (server). Όταν ένας χρήστης του Διαδικτύου, ο οποίος είναι ο πελάτης (client), επιθυμεί να αποκτήσει ένα αρχείο το οποίο βρίσκεται σε κάποιον εξυπηρετητή, τότε στέλνει την IP διεύθυνσή του σε αυτόν και ο εξυπηρετητής του στέλνει το αρχείο που ζήτησε. Κάθε εξυπηρετητής είναι ένας υπολογιστής ο οποίος έχει αρκετά μεγάλος εύρος δικτύου (bandwidth), καθώς είναι συχνό φαινόμενο να υπάρχουν πολλοί πελάτες που να ζητούν διαφορετικά δεδομένα την ίδια χρονική στιγμή από έναν εξυπηρετητή. Πρέπει, λοιπόν, ο εξυπηρετητής αυτός να έχει τη δυνατότητα να ικανοποιεί πολλαπλές ταυτόχρονες αιτήσεις. Ένα βασικό μειονέκτημα της μεθόδου πελάτη-εξυπηρετητή είναι το γεγονός ότι τα δεδομένα τα διαχειρίζεται ένας κεντρικός υπολογιστής. Αν ο υπολογιστής αυτός δεν είναι διαθέσιμος για κάποιο λόγο, τότε δεν είναι διαθέσιμα και τα δεδομένα που περιέχει. Το γεγονός αυτό είναι ιδιαίτερα δυσάρεστο διότι είναι πολύ πιθανές οι επιθέσεις από κακόβουλα

προγράμματα στον εξυπηρετητή με στόχο την αποτυχία του. Συνεπώς, η διαθεσιμότητα των αρχείων είναι αρκετά επισφαλής στο μοντέλο πελάτη-εξυπηρετητή.

Ένα άλλο είδος κατανεμημένων συστημάτων το οποίο χρησιμοποιείται ολοένα και περισσότερο από εκατομμύρια χρήστες είναι τα δίκτυα ομότιμων κόμβων (Peer-to-Peer, p2p) [18] [19] [20]. Τα δίκτυα ομότιμων είναι κατανεμημένα συστήματα ομότιμων κόμβων τα οποία χρησιμοποιούν την υποδομή του Διαδικτύου και τους πόρους του για την επικοινωνία μεταξύ των κόμβων. Τα συστήματα αυτά ασχολούνται με το διαμοιρασμό διαφόρων αρχείων μεταξύ των χρηστών. Κύριο χαρακτηριστικό των κόμβων είναι το γεγονός ότι είναι *ομότιμοι*, δηλαδή κάθε συμμετέχει ισάξια στο σύστημα, έχει τα ίδια δικαιώματα και τις ίδιες υποχρεώσεις με τους υπόλοιπους. Προκύπτει, επομένως, ότι τα p2p δίκτυα δεν είναι ως επί το πλείστον κεντρικοποιημένα, δηλαδή δεν υπάρχει ένας κεντρικός υπολογιστής (εξυπηρετητής) ο οποίος να διευθύνει και να δρομολογεί την επικοινωνία μεταξύ των κόμβων και τη διαμοίραση των αρχείων. Αυτό είναι επιθυμητό διότι με τον τρόπο αυτό το δίκτυο γίνεται πιο ανθεκτικό και βιώσιμο καθώς δεν υπάρχει ο κίνδυνος συγκεντρωμένης κακόβουλης επίθεσης σε συγκεκριμένο στόχο. Ένα δίκτυο ομότιμων φαίνεται στο Σχήμα 3.1.



Σχήμα 3.1. Διασύνδεση κόμβων σε ένα δίκτυο ομότιμων

Τα δίκτυα ομότιμων κόμβων είναι δυναμικά δίκτυα, υπό την έννοια ότι οι διάφοροι ομότιμοι (peers) που συμμετέχουν στο δίκτυο εισέρχονται και εξέρχονται από αυτό με αυθαίρετο τρόπο. Αυτό έχει ως αποτέλεσμα τα δεδομένα που κρατούν οι ομότιμοι να

μην είναι διαθέσιμα όταν βρίσκονται εκτός του δικτύου, κάτι που μπορεί να έχει επίπτωση στην απόδοσή του. Ένας ουσιαστικός παράγοντας που συμβάλλει στην αξιοπιστία και τη βιωσιμότητα ενός δικτύου ομότιμων είναι η δυνατότητα να έχουμε πρόσβαση σε δεδομένα οποιαδήποτε στιγμή, ακόμη και όταν κάποιοι κόμβοι που τα διαθέτουν βρίσκονται εκτός του δικτύου. Η διαθεσιμότητα (availability), λοιπόν, των δεδομένων που διακινούνται στο δίκτυο, πρέπει να είναι όσο το δυνατόν μεγαλύτερη, ώστε να εγγυάται η επιτυχής ανάκτησή τους από οποιονδήποτε ομότιμο τα ζητήσει.

Η Κωδικοποίηση Απαλοιφής βρίσκει εφαρμογή στα δίκτυα ομότιμων κατά την αποθήκευση των δεδομένων στους διάφορους κόμβους. Είναι λογικό ότι τα αρχεία θα πρέπει να αποθηκεύονται σε διαφορετικούς κόμβους και όχι μόνο σε αυτόν που το διαχειρίζεται αρχικά. Με τον τρόπο αυτό το αρχείο θα είναι διαθέσιμο ακόμη και αν κάποιος κόμβος που το διαθέτει βρεθεί εκτός δικτύου. Με την Κωδικοποίηση Απαλοιφής ένα αρχείο δεν αντιγράφεται αυτούσιο σε περισσότερους κόμβους. Συγκεκριμένα, όταν κάποιο αρχείο εισάγεται στο δίκτυο, τα m τμήματα δεδομένων που το συνθέτουν κωδικοποιούνται σε νέα n τμήματα ελέγχου, όπως είδαμε στο προηγούμενο κεφάλαιο. Τα τμήματα ελέγχου διασπείρονται κατά μήκος του δικτύου στους διάφορους κόμβους. Λαμβάνοντας οποιαδήποτε m από τα n αυτά τμήματα, μπορεί κάποιος κόμβος να ανακτήσει το αρχικό αρχείο με μεγάλη πιθανότητα. Τα τμήματα ελέγχου αποθηκεύονται σε n διαφορετικούς κόμβους, δηλαδή κάθε κόμβος κρατά ένα τμήμα, με στόχο τη μεγαλύτερη διαθεσιμότητα του αρχείου. Είναι προφανές από τον ορισμό της Κωδικοποίησης Απαλοιφής ότι το αρχείο δεν θα είναι διαθέσιμο όταν θα υπάρχουν στο δίκτυο λιγότεροι από m κόμβοι που να διαθέτουν κάποιο τμήμα ελέγχου. Προκύπτει, συνεπώς, ότι η διαθεσιμότητα του αρχείου σε ένα Δίκτυο Ομότιμων με Κωδικοποίηση Απαλοιφής ισούται με το πλήθος των $\binom{n}{m}$ συνδυασμών τμημάτων ελέγχου που μας δίνουν το αρχείο μέσω της αποκωδικοποίησης. Αν επιθυμούμε μεγαλύτερη διαθεσιμότητα, μπορούμε να δημιουργήσουμε αντίγραφα των τμημάτων ελέγχου τα οποία και αποθηκεύονται σε επιπλέον κόμβους. Παρατηρούμε, λοιπόν, ότι με την Κωδικοποίηση Απαλοιφής υπάρχει σημαντικό κέρδος όσον αφορά τη διαθεσιμότητα των αρχείων και γενικά της σταθερότητας και απόδοσης του δικτύου.

Το κέρδος που προκύπτει από την Κωδικοποίηση Απαλοιφής ως μέθοδο για την αποθήκευση των δεδομένων σε ένα δίκτυο ομότιμων είναι ακόμη μεγαλύτερο αν αναλογιστούμε και τον αποθηκευτικό χώρο που απαιτεί η αντιγραφή ενός αρχείου στο δίκτυο. Αν το μέγεθος ενός αρχείου είναι k, τότε η δημιουργία δύο αντιγράφων του αρχείου που θα διατίθενται στο δίκτυο επιβάλλει αποθηκευτικό κόστος 2k. Μάλιστα, η διαθεσιμότητα του αρχείου είναι 2, καθώς αν φύγουν οι δύο κόμβοι που κρατούν αντίγραφο του αρχείου τότε αυτό γίνεται αυτομάτως μη διαθέσιμο. Αν όμως χωρίσουμε το αρχείο σε m τμήματα δεδομένων και τα κωδικοποιήσουμε σε 2m τμήματα ελέγχου, προκύπτει ότι με ίδιο αποθηκευτικό κόστος (2k) έχουμε $\binom{2m}{m} = \frac{(2m)!}{m!(2m-m)!}$ συνδυασμούς που μπορούν να μας δώσουν το αρχείο. Παρατηρούμε, δηλαδή, ότι η ανθεκτικότητα του δικτύου των ομότιμων γίνεται ισχυρότερη με μικρό συγκριτικά κόστος σε χώρο. Για παράδειγμα, αν m=3, τότε έχουμε 40 συνδυασμούς τμημάτων ελέγχου που μπορούν να δώσουν το αρχείο, προσθέτοντας κόστος ίσο με το διπλάσιο του μεγέθους του αρχείου. Αυτό ισχύει χωρίς να χρειάζεται η περαιτέρω δημιουργία αντιγράφων των τμημάτων ελέγχου που να αποθηκευτούν σε κόμβους του δικτύου.

Γεγονός είναι, βέβαια, ότι απαιτούνται περισσότερα μηνύματα για τη συλλογή των απαιτούμενων τμημάτων για την αποκωδικοποίηση του αρχείου από ότι να υπήρχε ολόκληρο αντίγραφο σε κάποιο κόμβο. Αυτό ισχύει διότι τα τμήματα ελέγχου τα οποία πρέπει να ανακτηθούν για την ανάκτηση του αρχικού εγγράφου βρίσκονται διασκορπισμένα σε διαφορετικούς κόμβους. Δεν ψάχνουμε, δηλαδή, για ένα τμήμα μόνο, αλλά για περισσότερα, επομένως η αναζήτηση διατρέχει μεγαλύτερο τμήμα των κόμβων του δικτύου. Ωστόσο αυτό είναι ένα μικρό τίμημα που οφείλουμε να πληρώσουμε σε σχέση με τη σταθερότητα και αξιοπιστία που αποκτάται από την αυξημένη διαθεσιμότητα του αρχείου.

Με τη χρήση των κωδικών πηγής, όπως οι LT κώδικες που χρησιμοποιούμε, αν θέλουμε να αυξήσουμε κι άλλο τη διαθεσιμότητα ενός αρχείου, μπορούμε να μην ασχοληθούμε καθόλου με τη δημιουργία αντιγράφων των ίδιων των τμημάτων ελέγχου, ώστε να υπάρχει μεγαλύτερο πλήθος από αυτά στο δίκτυο. Η δυνατότητα που μας παρέχουν οι κώδικες αυτοί είναι το γεγονός ότι μπορούμε να παράγουμε

οσοδήποτε πλήθος τμημάτων ελέγχου επιθυμούμε. Μπορούμε, λοιπόν, να δημιουργήσουμε περισσότερα διαφορετικά τμήματα ελέγχου για τα τμήματα του αρχείου και να τα αποθηκεύσουμε σε διαφορετικούς κόμβους. Με τον τρόπο αυτό αποφεύγεται η ύπαρξη όμοιων τμημάτων ελέγχου και συνεπώς δεν υπάρχει ο κίνδυνος αποστολής δύο ίδιων τμημάτων ελέγχου σε κάποιο κόμβο που θα ζητήσει κάποιο αρχείο. Αποφεύγουμε, δηλαδή, την περιττή αποστολή των ίδιων δεδομένων, κερδίζοντας τόσο σε χρόνο όσο και σε εύρος δικτύου, καθώς οποιαδήποτε τμήματα φτάσουν στον κόμβο που τα ζήτησε, είναι εγγυημένο ότι είναι διαφορετικά.

Στη συγκεκριμένη εργασία ασχοληθήκαμε με την είσοδο της μεθόδου των LT κωδικών για δημιουργία αντιγράφων σε ένα δίκτυο ομότιμων και την ανάλυση των πλεονεκτημάτων και μειονεκτημάτων στην απόδοση του δικτύου. Στην παρούσα εργασία το δίκτυο ομότιμων που υλοποιήθηκε είναι ένα αδόμητο και μη κεντρικοποιημένο δίκτυο [19], στο οποίο οι κόμβοι συνδέονται μεταξύ τους χωρίς την ύπαρξη κάποιας κεντρικής οντότητας (π.χ. υπερ-κόμβος) η οποία να δρομολογεί την επικοινωνία και μεταφορά αρχείων μεταξύ τους.

Διαθεσιμότητα

Όπως είπαμε, ένας από τους σημαντικότερους παράγοντες που συμβάλλουν στην αξιοπιστία και βιωσιμότητα ενός p2p δικτύου είναι η διαθεσιμότητα (availability) των δεδομένων που υπάρχουν σε αυτό κάθε χρονική στιγμή. Η διαθεσιμότητα αναφέρεται στο μέγιστο πλήθος των μη διαθέσιμων κόμβων που μπορεί να ανεχτεί το δίκτυο χωρίς να υπάρχει πτώση στην συμπεριφορά του. Με άλλα λόγια ορίζουμε τη διαθεσιμότητα ενός εγγράφου ως το μέγιστο πλήθος αντιγράφων που πρέπει να υπάρχουν στο δίκτυο ώστε η πιθανότητα ανάκτησής του να είναι η ίδια, ανεξάρτητα από το ποσοστό των κόμβων που αποτυγχάνουν σε μια δεδομένη στιγμή.

Ακολούθως δίνουμε τους τύπους της διαθεσιμότητας ενός εγγράφου σε ένα p2p δίκτυο, με βάση τις μεθόδους της Αντιγραφής και της Κωδικοποίησης Απαλοιφής ως μεθόδους διατήρησης αντιγράφων.

Αντιγραφή

Κατά την Αντιγραφή δημιουργούμε k αντίγραφα ενός εγγράφου Α, έτσι ώστε αυτό να είναι διαθέσιμο ακόμη και αν φύγουν ταυτόχρονα k-1 κόμβοι από το δίκτυο. Ο υπολογισμός του k εξαρτάται από το ποσοστό e το Α να μην είναι διαθέσιμο και τη μέση διαθεσιμότητα των κόμβων f. Η πιθανότητα p το Α να μην είναι διαθέσιμο είναι η εξής:

$$\begin{aligned} p &= P(\text{ το Α δεν είναι διαθέσιμο}) \\ &= P(\text{ κανένα αντίγραφο δεν είναι διαθέσιμο}) \\ &= P(\text{ένα αντίγραφο δεν είναι διαθέσιμο})^k \end{aligned} \qquad \text{Εξ. 3.1}$$

Από την εξίσωση Εξ(3.1) προκύπτει η εξίσωση Εξ(3.2):

$$p = (1-f)^k \qquad \text{Εξ. 3.2}$$

Άρα, η πιθανότητα P_1 το Α να είναι διαθέσιμο φαίνεται στην εξίσωση Εξ(2.3):

$$P_1 = 1-p = 1-(1-f)^k \qquad \text{Εξ. 3.3}$$

Λύνοντας ως προς k, προκύπτει το πλήθος των αντιγράφων που επιθυμούμε (Εξ(3.4)):

$$k = \frac{\log e}{\log(1-f)} \qquad \text{Εξ. 3.4}$$

Κωδικοποίηση Απαλοιφής

Στην Κωδικοποίηση Απαλοιφής η πιθανότητα P ένα έγγραφο Α να είναι διαθέσιμο υπολογίζεται ως ακολούθως:

P_0 : πιθανότητα ένα τμήμα ελέγχου να είναι διαθέσιμο
n : το σύνολο των κωδικοποιημένων τμημάτων
m : το απαιτούμενο πλήθος τμημάτων για ανάκτηση εγγράφου
N : συνολικό πλήθος κόμβων

M : πλήθος κόμβων εκτός δικτύου

$$P_0 = \sum_{i=0}^{n-m} \frac{\binom{M}{i} \binom{N-M}{n-i}}{\binom{N}{n}} \qquad \text{Εξ. 3.5}$$

Η εξίσωση της διαθεσιμότητας Εξ(3.5) ισούται με το πλήθος των τρόπων που μπορούμε να αποθηκεύσουμε μη διαθέσιμα τμήματα σε μη διαθέσιμους κόμβους πολλαπλασιαζόμενο με το πλήθος των τρόπων που μπορούμε να αποθηκεύσουμε διαθέσιμα τμήματα σε διαθέσιμους κόμβους, διαιρούμενο με το πλήθος των τρόπων που μπορούμε να αποθηκεύσουμε όλα τα τμήματα σε όλους τους κόμβους.

3.2 Αρχιτεκτονική Δικτύου

Στο δίκτυο το οποίο σχεδιάστηκε οι κόμβοι συνδέονται μεταξύ τους χωρίς να υπάρχει κάποιος κεντρικός κόμβος που να αναλαμβάνει αυτή τη διαδικασία. Αυτό δίνει μια μεγαλύτερη ευελιξία και ανθεκτικότητα στο δίκτυο, υπό την έννοια ότι δεν υπάρχει κάποια κεντρικοποιημένη δομή που να είναι απαραίτητη για την ορθή λειτουργία του. Συνεπώς αποφεύγεται ο κίνδυνος αν αποτύχει η συγκεκριμένη δομή να συνδεθεί, να προκληθεί πτώση του δικτύου. Οι κόμβοι εισέρχονται και εξέρχονται από το δίκτυο ελεύθερα και χωρίς περιορισμούς.

Όταν ένας κόμβος εισέλθει στο δίκτυο, τότε ενώνεται με κάποιους άλλους, ήδη υπάρχοντες, κόμβους (γείτονες) με τους οποίους μπορεί να επικοινωνήσει μέσω ανταλλαγής μηνυμάτων και δεδομένων. Επιπλέον, οι κόμβοι έχουν ένα μοναδικό αναγνωριστικό (id) με βάση το οποίο μπορούν να τους αναγνωρίσουν και να επικοινωνήσουν οι γείτονές τους. Κάθε κόμβος στέλνει και δέχεται μηνύματα από τους γείτονές του. Για το λόγο αυτό κάθε κόμβος κρατά μια λίστα η οποία περιλαμβάνει τα αναγνωριστικά των γειτονικών κόμβων με τους οποίους συνδέεται. Όταν ένας νέος κόμβος εισέρχεται στο δίκτυο, τότε ενημερώνει τη λίστα το με τα αναγνωριστικά των κόμβων οι οποίοι αποτελούν τους γείτονές του. Σε κάθε περίπτωση, θα πρέπει να συνδέεται με τουλάχιστον ένα κόμβο ώστε να μπορεί να αποτελεί μέρος του δικτύου.

Με την είσοδο του κόμβου στο δίκτυο, οι γείτονές του ενημερώνουν αντίστοιχα τις δικές τους λίστες, προσθέτοντας το αναγνωριστικό του. Σε περίπτωση αποτυχίας κάποιου κόμβου, δηλαδή στην περίπτωση που αποχωρήσει από το δίκτυο, οι γείτονές του ενημερώνουν ξανά τη λίστα τους, αφαιρώντας το αναγνωριστικό του.

Κάθε κόμβος του δικτύου διαθέτει τη δυνατότητα επεξεργασίας των αρχείων με βάση την LT μέθοδο απαλοιφής. Συγκεκριμένα χρησιμοποιεί τις συναρτήσεις κωδικοποίησης και αποκωδικοποίησης της συγκεκριμένης μεθόδου. Κατά την είσοδό του εφαρμόζει τον αλγόριθμο κωδικοποίησης στα δεδομένα που πιθανώς διαθέτει και δημιουργεί τα κωδικοποιημένα τμήματα τα οποία θα διοχετεύσει στη συνέχεια σε διάφορους κόμβους του δικτύου. Τα τμήματα ελέγχου είναι αυτά τα οποία διαδίδονται στο δίκτυο (όπως θα δούμε στο επόμενο κεφάλαιο). Προφανώς η διαδικασία αυτή αυξάνει το μέγεθος των δεδομένων που αποθηκεύονται στο δίκτυο, ωστόσο αυξάνεται σημαντικά η βιωσιμότητα και ανθεκτικότητά του. Αυτό ισχύει διότι η διαθεσιμότητα κάθε δεδομένου αυξάνεται και είναι εφικτή η ανάκτησή του ακόμη και αν ένα σημαντικό ποσοστό κόμβων εξέλθει από το δίκτυο.

Στην περίπτωση που ζητήσει ο ίδιος κάποιο αρχείο, τότε θα πρέπει να ανακτήσει τον απαραίτητο αριθμό από κωδικοποιημένα τμήματα, στα οποία στη συνέχεια θα εφαρμόσει τον αλγόριθμο αποκωδικοποίησης, μέχρι να ολοκληρώσει την αναζήτησή του. Τη διαδικασία αποθήκευσης των τμημάτων ελέγχου στο δίκτυο, καθώς και την αναζήτηση και ανάκτηση ενός δεδομένου περιγράφουμε αναλυτικότερα στις επόμενες ενότητες.

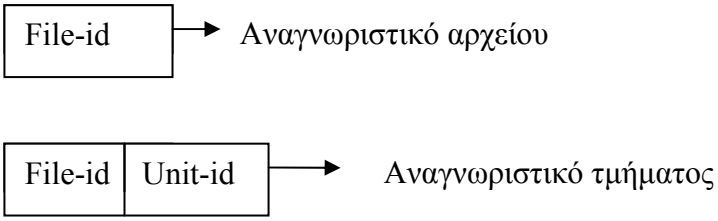
3.2.1. Αποθήκευση των τμημάτων στο Δίκτυο

Όταν ένας κόμβος εισέρχεται στο δίκτυο και αποκτήσει τους γείτονές του, το πρώτο πράγμα που κάνει είναι να εφαρμόσει τον αλγόριθμο κωδικοποίησης στα έγγραφα του ώστε να παράγει τα κωδικοποιημένα τμήματα που θα αποθηκευτούν στο δίκτυο. Επιπλέον, οι LT κώδικες, ως Κώδικες Πηγής, δίνουν τη δυνατότητα παραγωγής απεριόριστου πλήθους τμημάτων ελέγχου που κωδικοποιούν το αρχείο. Τα τμήματα ελέγχου είναι αυτά τα οποία θα αποθηκευτούν σε διάφορους κόμβους του δικτύου.

Κάθε τμήμα ελέγχου που δημιουργείται αποθηκεύεται ως μια πλειάδα (tuple) στο δίκτυο, η οποία χαρακτηρίζεται από δύο παράγοντες: ένα αναγνωριστικό, το οποίο χαρακτηρίζει μοναδικά το συγκεκριμένο τμήμα (πλειάδα) και επίσης τα δεδομένα του τμήματος. Τα χαρακτηριστικά αυτά περιγράφονται παρακάτω:

Α) **Αναγνωριστικό Πλειάδας.** Κάθε πλειάδα (τμήμα ελέγχου) συνδέεται με ένα αναγνωριστικό, το *file-id,* το οποίο είναι το όνομα του αρχείου στο οποίο ανήκει το τμήμα. Το file-id είναι σημαντικό, καθώς βοηθά στην ανάκτηση των σωστών τμημάτων ελέγχου, σε περίπτωση που γίνει αναζήτηση του συγκεκριμένου αρχείου από κάποιο κόμβο. Χωρίς την ύπαρξη του συγκεκριμένου αναγνωριστικού, θα ήταν αδύνατο να προσδιορίσουμε ποιο αρχείο αποκωδικοποιεί το συγκεκριμένο τμήμα ελέγχου, κάτι που οδηγεί σε αποτυχία της αναζήτησης.

Εκτός, όμως, από το file-id, το οποίο διαχωρίζει πλειάδες που κωδικοποιούν διαφορετικά αρχεία, χρειαζόμαστε και ένα επιπλέον αναγνωριστικό, το οποίο να διαχωρίζει πλειάδες που κωδικοποιούν το ίδιο αρχείο μεταξύ τους. Αυτό χρειάζεται για το γεγονός ότι ένας κόμβος ο οποίος θέλει να αποκωδικοποιήσει ένα αρχείο, δεν πρέπει να ανακτήσει περισσότερα αντίγραφα της ίδιας πλειάδας στην προσπάθειά του να αποκτήσει επαρκή αριθμό από αυτές για την επιτυχή αποκωδικοποίησης του αρχείου. Ένα τέτοιο ενδεχόμενο θα κατέληγε σε αποτυχία της αναζήτησης, διότι επί της ουσίας, ο απαραίτητος αριθμός από πλειάδες για την επιτυχία της ανάκτησης δε θα μπορούσε να συμπληρωθεί. Για να αποφευχθεί αυτή η ανεπιθύμητη κατάσταση, προσθέτουμε ένα επιπρόσθετο αναγνωριστικό (πεδίο στην πλειάδα), το *unit-id.* Το unit-id παίρνει τιμές από ένα μέχρι τον μέγιστο αριθμό των τμημάτων ελέγχου που έχουμε δημιουργήσει. Το πρώτο τμήμα λαμβάνει την τιμή 1 στο συγκεκριμένο αναγνωριστικό, το δεύτερο τμήμα παίρνει την τιμή 2 και συνεχίζεται αντίστοιχη ανάθεση τιμών στα unit-ids και των υπόλοιπων τμημάτων.



Σχήμα 3.2 Ο τύπος αναγνωριστικών που χρησιμοποιούνται στο σύστημα

Β) **Δεδομένα Πλειάδας.** Κάθε πλειάδα έχει ένα πεδίο data, το οποίο περιέχει τα δεδομένα του τμήματος ελέγχου στο οποίο αντιστοιχεί.

Αφού ολοκληρωθεί η δημιουργία των πλειάδων για το συγκεκριμένο αρχείο του κόμβου που εισήλθε στο δίκτυο, ακολουθεί στη συνέχεια η διαδικασία αποθήκευσης των πλειάδων σε διαφορετικούς κόμβους. Η αποθήκευση κάθε πλειάδας σε διαφορετικό κόμβο βοηθά στην επίτευξη μεγαλύτερης ανθεκτικότητας στη διαθεσιμότητα του αρχείου, καθώς θα πρέπει να φύγουν ταυτόχρονα πάνω από τρεις διαφορετικοί κόμβοι που να διαθέτουν από μια πλειάδα, για να μην είναι διαθέσιμο το αρχείο. Προφανώς, η πιθανότητα αυτή είναι σαφώς μικρότερη από την πιθανότητα να έφευγε ένας κόμβος που να είχε αποθηκευμένες περισσότερε πλειάδες για το ίδιο αρχείο. Επιπλέον, είναι επιθυμητό να μην είναι αποθηκευμένες όλες (ή οι περισσότερες) οι πλειάδες στους άμεσους γείτονες του εισερχόμενου κόμβου. Αυτό συμβαίνει διότι δεν επιθυμούμε να βρίσκονται όλες οι πλειάδες ενός αρχείου συγκεντρωμένες σε μια μαζεμένη ομάδα κόμβων, διότι αυτό θα έχει αρνητικό αντίκτυπο σε ενδεχόμενη αναζήτηση του αρχείου από κάποιο κόμβο που βρίσκεται αρκετά μακριά από τη συγκεκριμένη ομάδα.

Για τους παραπάνω λόγους ο νέος κόμβος επιλέγει να κρατήσει στη λίστα δεδομένων του την πλειάδα με unit-id =1 και στέλνει μηνύματα για να στείλει τις υπόλοιπες πλειάδες σε άλλους κόμβους. Κάθε μήνυμα που στέλνει ο κόμβος έχει πέντε πεδία:

Α) Το αναγνωριστικό (id) του αρχικού κόμβου που στέλνει το μήνυμα

Β) Το file-id της πλειάδας που στέλνει προς αποθήκευση

Γ) Το unit-id της συγκεκριμένης πλειάδας που θέλει να στείλει προς αποθήκευση

Δ) Ένα TTL (Time-To-Live) του μηνύματος. Το TTL είναι ένας τυχαίος αριθμός (επιλέγεται πάνω από τρία) ο οποίος υποδηλώνει τα βήματα που ακολουθούνται μέχρι να καταλήξουμε στον κόμβο που θα αποθηκεύσει τη συγκεκριμένη πλειάδα.

Ο αρχικός κόμβος στέλνει ένα μήνυμα που αφορά τη δεύτερη πλειάδα (αφού την πρώτη την περιέχει ο ίδιος) και θέτοντας το TTL του μηνύματος. Το μήνυμα αυτό το στέλνει τυχαία σε έναν από τους γείτονές του. Ο γείτονας που δέχεται το μήνυμα, μειώνει το TTL κατά ένα, υποδηλώνοντας ότι έχει γίνει το πρώτο βήμα (hop) του μηνύματος και το προωθεί σε κάποιον από τους γείτονές του. Κάθε κόμβος που λαμβάνει το συγκεκριμένο μήνυμα, ελέγχει αν το TTL είναι μηδέν. Σε περίπτωση που δεν είναι μηδέν, ακολουθείται η ίδια διαδικασία προώθησής του σε κάποιο γείτονα που δεν έχει πάει. Αν, ωστόσο, ισούται με μηδέν, τότε ο συγκεκριμένος κόμβος θεωρείται υποψήφιος για αποθήκευση της συγκεκριμένης πλειάδας. Στην περίπτωση αυτή, ο κόμβος αρχικά ελέγχει τη λίστα δεδομένων του, ώστε να δει αν υπάρχει αποθηκευμένη πλειάδα με file-id ίδιο με το file-id του μηνύματος. Αν δεν υπάρχει, τότε σημαίνει ότι ο κόμβος αυτός δεν έχει κάποια από τις πλειάδες του συγκεκριμένου αρχείου στο οποίο αναφέρεται το μήνυμα, επομένως μπορεί να αποδεχθεί τη συγκεκριμένη πλειάδα. Στέλνει, επομένως, ένα μήνυμα αποδοχής στον κόμβο με id ίσο με το id του μηνύματος. Το μήνυμα αποδοχής περιέχει ένα πεδίο με τιμή True (αποδοχή) και ένα πεδίο με το id του κόμβου που το στέλνει. Ο αρχικός κόμβος που δέχεται αυτό το μήνυμα μπορεί να επικοινωνήσει με τον αποδέκτη (αφού γνωρίζει το id του) και συνεπώς στέλνει την πλειάδα που αναφέρεται στο μήνυμα (με βάση τα file-id και unit-id του μηνύματος) στον κόμβο αυτό, ο οποίος και την αποθηκεύει.

Σε περίπτωση που ο κόμβος στον οποίο φτάνει το μήνυμα αποστολής της πλειάδας με TTL=0, έχει ήδη αποθηκευμένη πλειάδα με ίδιο file-id, τότε κάνει το TTL=1 και προωθεί το ίδιο μήνυμα σε κάποιον από τους υπόλοιπους γείτονές του στον οποίο δεν έχει πάει το μήνυμα. Στη συνέχεια ακολουθείται η ίδια διαδικασία αποθήκευσης της πλειάδας.

Στην περίπτωση που υπάρξει το ενδεχόμενο κάποιος κόμβος να έχει γείτονες οι οποίοι να έχουν ήδη δεχτεί το μήνυμα, τότε και πάλι αυξάνει το TTL κατά ένα και στέλνει το μήνυμα στον κόμβο από τον οποίο το έλαβε. Ο γειτονικός κόμβος στη συνέχεια το προωθεί σε κάποιον άλλο γείτονα που να μην το έχει ήδη λάβει. Η διαδικασία που περιγράφηκε, φαίνεται στον αλγόριθμο 3 που ακολουθεί.

1. Ο αρχικός κόμβος δημιουργεί n τμήματα ελέγχου με τον αλγόριθμο 1
Για κάθε τμήμα ελέγχου:
2. Αποθηκεύει το πρώτο τμήμα ελέγχου
3. Θέτει το TTL και στέλνει μήνυμα αποθήκευσης τυχαία σε ένα γείτονα
Γείτονας:
4. Ελέγχει αν το TTL=0
Αν ναι
5. Ελέγχει τη λίστα δεδομένων του αν υπάρχει τμήμα δεδομένων για το έγγραφο με id=file-id
Αν ναι
6. Αυξάνει το TTL κατά ένα και προωθεί το μήνυμα τυχαία σε γείτονα
7. Πήγαινε στο βήμα 4
Αλλιώς
8. Αποθηκεύει το τμήμα και στέλνει μήνυμα επιτυχίας στον αρχικό Κόμβο
Αν όχι
9. Μειώνει το TTL κατά ένα και προωθεί το μήνυμα τυχαία σε κάποιο Γείτονα
10. Πήγαινε στο βήμα 4

Σχήμα 3.3 Αλγόριθμος 3: Αλγόριθμος αποθήκευσης τμημάτων ελέγχου στο δίκτυο

Με την παραπάνω διαδικασία αποθήκευσης των πλειάδων διασφαλίζουμε αρχικά ότι κάποιος κόμβος μπορεί να αποθηκεύσει το πολύ μία πλειάδα που κωδικοποιεί ένα συγκεκριμένο αρχείο και όχι περισσότερες για το ίδιο αρχείο. Επιπλέον, με την ύπαρξη του TTL στο μήνυμα αποθήκευσης, πετυχαίνουμε την αποθήκευση των πλειάδων σε κόμβους που δεν βρίσκονται σε πολύ κοντινή απόσταση μεταξύ τους, εξασφαλίζοντας μεγαλύτερη διασπορά τους.

3.2.2 Αναζήτηση και ανάκτηση τμημάτων

Όταν κάποιος κόμβος ζητήσει κάποιο αρχείο, τότε ακολουθείται μια διαδικασία αναζήτησης του απαραίτητου πλήθους πλειάδων του συγκεκριμένου αρχείου που απαιτούνται για την επιτυχή αποκωδικοποίησή του. Η διαδικασία αναζήτησης περιγράφεται παρακάτω.

Αρχικά, ο κόμβος που κάνει την αναζήτηση (έστω N) ελέγχει τη λίστα δεδομένων του για να δει αν έχει ο ίδιος αποθηκευμένη κάποια από τις πλειάδες του αρχείου που ζητά. Αν δεν διαθέτει κάποια από τις πλειάδες αυτές τότε στέλνει ένα μήνυμα αναζήτησης (query) το οποίο αποτελείται από τα ακόλουθα πεδία.

Α) Το id του κόμβου N που κάνει την αναζήτηση

Β) Το αναγνωριστικό file-id του ζητούμενου αρχείου

Γ) Μια λίστα (έστω Α) η οποία αρχικά είναι κενή και η οποία θα περιέχει τα unit-ids των πλειάδων του αρχείου που θα ανακτηθούν επιτυχώς κατά τη διάρκεια της αναζήτησης.

Δ) Ένα TTL το οποίο δηλώνει το πλήθος των βημάτων που θα γίνουν μέχρι την ολοκλήρωση του μηνύματος

Ε) Ένα πεδίο count. Το count παίρνει την τιμή του αριθμού των ζητούμενων τμημάτων, ώστε να ξεκινήσει η αποκωδικοποίηση. Αυτό το πεδίο ελέγχεται από τους κόμβους στους οποίους φτάνει το μήνυμα και χρησιμοποιείται ώστε να τερματιστεί το μήνυμα σε περίπτωση που ανακτηθούν επαρκή τμήματα ελέγχου.

Ο N αρχικά επιλέγει έναν από τους γείτονές του στον οποίο θα στείλει την αίτηση. Επιπλέον επιλέγει ένα (σχετικά μεγάλο) TTL το οποίο δηλώνει τη διάρκεια ζωής του μηνύματος αναζήτησης. Σε περίπτωση που ο N έχει αποθηκευμένη κάποια από τις ζητούμενες πλειάδες, τότε προσθέτει το unit-id της στη λίστα Α του μηνύματος. Σε διαφορετική περίπτωση, η λίστα Α παραμένει κενή. Επίσης θέτει το count ίσο με το

πλήθος των τμημάτων που χρειάζεται για να ξεκινήσει την αποκωδικοποίηση. Ακολούθως, στέλνει την αίτηση στο γείτονά του.

Ο γείτονας που θα λάβει την αίτηση, αρχικά ελέγχει τη λίστα δεδομένων του για ενδεχόμενη ύπαρξη κάποιας πλειάδας με file-id ίσο με το file-id του μηνύματος. Αν έχει κάποια πλειάδα, τότε ελέγχει περαιτέρω και το unit-id της πλειάδας του. Αν αυτό είναι ίδιο με κάποιο από τα unit-ids που υπάρχουν στη λίστα Α του μηνύματος, τότε μειώνει το TTL κατά ένα και προωθεί το μήνυμα σε κάποιον γείτονά του. Η ίδια διαδικασία ακολουθείται και από τους υπόλοιπους κόμβους στους οποίους φτάνει η συγκεκριμένη αίτηση.

Αν, στην πορεία του μηνύματος, βρεθεί κάποιος κόμβος ο οποίος περιέχει πλειάδα που δεν έχει ανακτηθεί ακόμη από τον Ν (έχει δηλαδή ίδιο file-id με αυτό του μηνύματος αλλά διαφορετικό unit-id από τα αντίστοιχα unit-ids που υπάρχουν στη λίστα Α) τότε έχουμε επιτυχία. Ο συγκεκριμένος κόμβος μειώνει το count κατά ένα, υποδηλώνοντας ότι έχει βρεθεί ζητούμενο τμήμα, άρα τα εναπομείναντα είναι κατά ένα λιγότερα. Επιπλέον αποστέλλει και την εν λόγω πλειάδα. Αν το count γίνει μηδέν, τότε ο κόμβος τερματίζει το μήνυμα, στέλνοντας μήνυμα επιτυχίας στον αρχικό κόμβο. Αν δεν είναι μηδέν , τότε το προωθεί τυχαία σε κάποιον από τους γείτονές του. Αν το TTL είναι μηδέν όταν φτάσει το μήνυμα σε κάποιο κόμβο και δεν έχει βρεθεί ζητούμενη πλειάδα, τότε ο κόμβος στον οποίο έφτασε το μήνυμα στέλνει μήνυμα αποτυχίας στον Ν.

Όταν στον Ν φτάσει μήνυμα επιτυχίας από κάποιο κόμβο μαζί με τη ζητούμενη πλειάδα (τμήμα ελέγχου) τότε την αποθηκεύει στη λίστα δεδομένων του. Στη συνέχεια ελέγχει τη λίστα ξανά ώστε να δει αν έχει συμπληρώσει τον απαιτούμενο αριθμό από πλειάδες ώστε να ξεκινήσει η αποκωδικοποίηση. Εάν δεν έχει συμπληρωθεί το απαιτούμενο πλήθος από πλειάδες, τότε ο Ν στέλνει ένα νέο μήνυμα σε κάποιον από τους γείτονες που δεν είχε επιλέξει κατά την τελευταία αναζήτησή του, έτσι ώστε να διασφαλιστεί ότι δεν θα ακολουθηθεί το ίδιο μονοπάτι με προηγούμενη αναζήτηση (είτε επιτυχή είτε όχι). Αυτό που κάνει ο Ν είναι να ορίσει ξανά το TTL του νέου μηνύματος και επίσης να προσθέσει τα unit-ids των πλειάδων του αρχείου που έχει ήδη αποθηκευμένες στη λίστα Α του νέου μηνύματος. Η διαδικασία αναζήτησης και ανάκτησης των τμημάτων ελέγχου φαίνεται στον αλγόριθμο 4 που ακολουθεί.

Αρχικός κόμβος (N)

1. Θέτει TTL και count ίσο με το πλήθος των τμημάτων ελέγχου που απατούνται και ανανεώνει τη λίστα των unit-ids (A) αν έχει κάποιο από τα τμήματα ελέγχου (αλλιώς είναι κενή). Ακολούθως στέλνει μήνυμα αναζήτησης τυχαία σε γείτονά του
2. Αν λάβει ένα τμήμα ελέγχου από κάποιο κόμβο, το αποθηκεύει στη λίστα δεδομένων του
3. Όταν λάβει επαρκή τμήματα, ξεκινά την αποκωδικοποίηση χρησιμοποιώντας τον αλγόριθμο 2
4. Αν η αποκωδικοποίηση αποτύχει, θέτει ξανά TTL και count=1 και επαναλαμβάνει το βήμα 1.

Γείτονας

Λαμβάνοντας μήνυμα αναζήτησης από κάποιο κόμβο

5. Ελέγχει αν το TTL είναι μηδέν

Αν όχι

6. Μειώνει το TTL κατά ένα
7. Ελέγχει τα δεδομένα του για το αν έχει πλειάδες με το file-id του μηνύματος

Αν ναι

8. Ελέγχει αν το unit-id της πλειάδας του υπάρχει στη λίστα A

Αν ναι

9. προωθεί το μήνυμα τυχαία σε γείτονα

Αλλιώς

10. μειώνει το count κατά ένα και στέλνει την πλειάδα στον αρχικό κόμβο.
11. ελέγχει αν το count είναι μηδέν

Αν ναι

12. Τερματίζει το μήνυμα

Αλλιώς

13. Πηγαίνει στο βήμα 9

Αν όχι

Πηγαίνει στο βήμα 9

Αν ναι

Πηγαίνει στο βήμα 7

Αν ισχύει το βήμα 8, δεν προωθεί το μήνυμα αλλά το τερματίζει στέλνοντας αποτυχία

Σχήμα 3.4. Αλγόριθμος 4: Αλγόριθμος αναζήτησης τμημάτων ελέγχου

Σε περίπτωση που συμπληρωθεί ο απαιτούμενος αριθμός πλειάδων από τον N για το αρχείο που αναζήτησε, τότε χρησιμοποιεί αυτές τις πλειάδες στον αλγόριθμο αποκωδικοποίησης των LT Κωδικών που διαθέτει. Επειδή οι LT Κώδικες είναι πιθανοτικοί αλγόριθμοι, ένας παράγοντας είναι και το ποσοστό αποτυχίας το οποίο

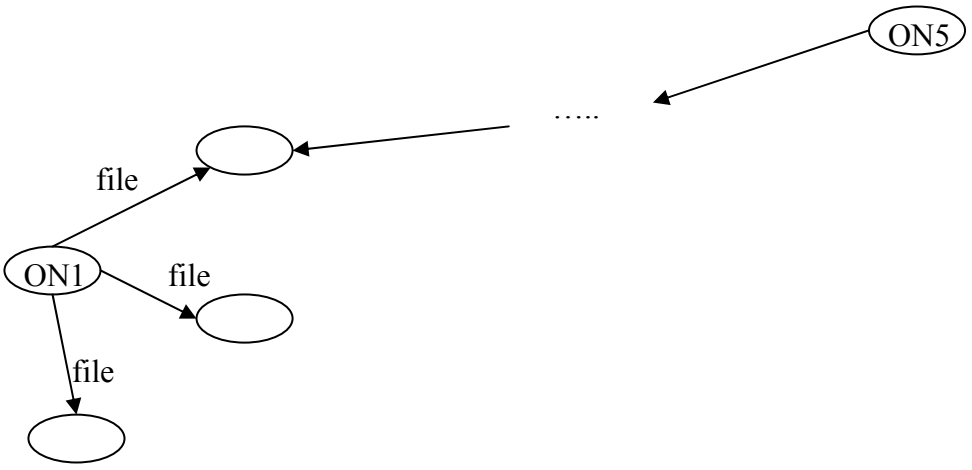
θεωρείται αποδεκτό (προφανώς το θέλουμε όσο το δυνατόν μικρότερο). Στη Robust Soliton Κατανομή που χρησιμοποιείται, το ποσοστό αυτό εκφράζεται από την ποσότητα δ. Για παράδειγμα, αν ένα αρχείο είναι χωρισμένο σε 4 τμήματα, λαμβάνοντας λίγο παραπάνω από 4 τμήματα, περίπου 6, μπορούν οι LT Κώδικες να παράγουν το αρχείο με δ=0.05, δηλαδή με περίπου 95% επιτυχία, ενώ αν ένα αρχείο είναι χωρισμένο σε 20 τμήματα, χρειαζόμαστε περίπου 25 τμήματα για την ανάκτησή του. Γενικότερα, το πλήθος των επιπλέον τμημάτων που χρειάζονται κατά μέσο όρο για την ανάκτηση ενός αρχείου χωρισμένου σε m τμήματα δεδομένων, είναι της τάξης $O(\sqrt{m})$. Ενδέχεται, λοιπόν, ακόμη και με την ανάκτηση των απαραίτητων πλειάδων από τον N, η αποκωδικοποίηση να μην είναι επιτυχής. Στην περίπτωση αυτή, η ανάκτηση μιας ακόμη πλειάδας εγγυάται ότι με πιθανότητα πάνω από 90 % η αποκωδικοποίηση θα είναι πετυχημένη. Για το λόγο αυτό ο N στέλνει νέο μήνυμα αναζήτησης, ανανεώνοντας κατάλληλα τη λίστα A και το TTL, σε κάποιον γείτονα με τυχαίο τρόπο, με σκοπό την ανάκτηση μιας επιπλέον πλειάδας , ώστε να διασφαλιστεί κατά πολύ μεγάλο ποσοστό η επιτυχής ανάκτηση του ζητούμενου αρχείου.

3.3 Μέθοδος Αντιγραφής Σε Δίκτυα Ομότιμων

Η πιο διαδεδομένη μέθοδος για την αύξηση της διαθεσιμότητας των δεδομένων σε ένα δίκτυο ομότιμων είναι η τεχνική της *αντιγραφής (replication)*. Με τη μέθοδο αυτή δημιουργούμε k πλήρη αντίγραφα ενός αρχείου, τα οποία αποθηκεύονται σε k διαφορετικούς κόμβους του δικτύου. Συνεπώς, το αρχείο αυτό είναι διαθέσιμο προς προσπέλαση από κάποιον κόμβο όταν υπάρχει τουλάχιστον ένας κόμβος στο δίκτυο, ο οποίος να έχει αποθηκευμένο ένα από τα αντίγραφά του. Αυτή η διαδικασία, βέβαια, επιβάλλει στο δίκτυο ένα επιπρόσθετο κόστος αποθήκευσης κατά παράγοντα k, όσα δηλαδή αντίγραφα του αρχείου είναι αποθηκευμένα σε αυτό.

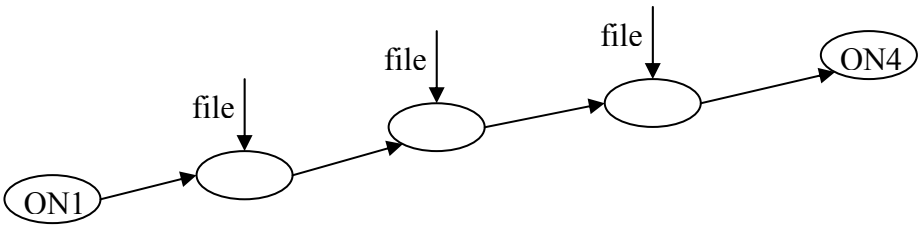
Υπάρχουν διάφοροι αλγόριθμοι οι οποίοι υλοποιούν την αντιγραφή σε ένα δίκτυο ομότιμων. Ένας από αυτούς είναι η δημιουργία αντιγράφων ενός αρχείου που διαθέτει ένας ομότιμος και η αποθήκευσή τους στους γείτονές του (Σχήμα 3.5). Αυτό, ωστόσο, έχει το μειονέκτημα ότι τα αντίγραφα του αρχείου βρίσκονται σε μια μικρή ομάδα κόμβων στο δίκτυο, δηλαδή η διασπορά τους δεν είναι μεγάλη. Αυτό έχει ως αποτέλεσμα, κάποιοι κόμβοι που ζητούν το συγκεκριμένο αρχείο και βρίσκονται

μακριά από αυτή την ομάδα κόμβων να χρειάζονται σημαντικό αριθμό βημάτων (hops) ώστε να μπορεί να βρεθεί κάποιος κόμβος που να περιέχει κάποιο αντίγραφο. Ενδέχεται, μάλιστα, αν υπάρχει ένας συγκεκριμένος αριθμός επιτρεπόμενων βημάτων (TTL-Time-To-Live) για κάθε αναζήτηση, να υπάρξει αποτυχία στην ανάκτηση του αρχείου διότι απαιτούνται περισσότερα βήματα για να βρεθεί ο κόμβος που να ικανοποιεί την αναζήτηση.



Σχήμα 3.5 Αντιγραφή ενός αρχείου (file) από τον κόμβο ON1 στους γείτονές του. Ο ON5 χρειάζεται αρκετά βήματα (hops) για να το ανακτήσει

Μια άλλη μέθοδος που χρησιμοποιείται για την υλοποίηση της αντιγραφής είναι η αποκαλούμενη *αντιγραφή μονοπατιού (path replication)* (Σχήμα 3.6). Η μέθοδος αυτή είναι διαφορετική σε σύγκριση με τη μέθοδο που περιγράφηκε προηγουμένως. Σύμφωνα με αυτή όταν ένας ομότιμος κάνει μια αναζήτηση για ένα συγκεκριμένο αρχείο, η αναζήτηση αυτή προωθείται στους γειτονικούς κόμβους οι οποίοι εξετάζουν αν διαθέτουν το αρχείο που ζητήθηκε. Η αναζήτηση ολοκληρώνεται όταν βρεθεί ένας κόμβος που να διαθέτει το αρχείο, το οποίο και στέλνει σε αυτόν που το ζήτησε. Αφού βρεθεί ο κόμβος που ικανοποιεί την αναζήτηση, το συγκεκριμένο αρχείο αντιγράφεται και αποθηκεύεται σε κάθε κόμβο που βρίσκεται στο μονοπάτι από εκείνον που ζήτησε το αρχείο, έως αυτόν που ικανοποιεί την αναζήτηση. Έχουμε δηλαδή τη δημιουργία αντιγράφων που βρίσκονται στους ομότιμους που συνθέτουν το μονοπάτι που ακολούθησε η αναζήτηση για το αρχείο, έως την ικανοποίησή της.

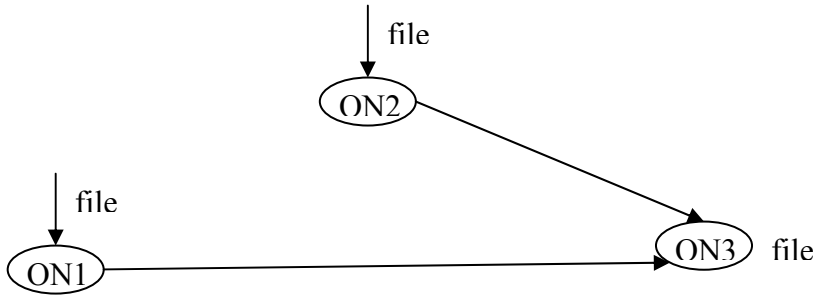


Σχήμα 3.6 Ο κόμβος ON1 ζητά το αρχείο file, το οποίο έχει ο ON4. Το file αποθηκεύεται στους ενδιάμεσους κόμβους που σχηματίζουν το μονοπάτι

Το πλεονέκτημα της μεθόδου αυτής είναι ότι αν το συγκεκριμένο αρχείο ζητηθεί ξανά από τον κόμβο που έκανε την αρχική αναζήτηση ή από κάποιον άλλο κόμβο που βρίσκεται κοντά στο μονοπάτι, η ανάκτηση του αρχείου θα είναι επιτυχής πολύ πιο γρήγορα, με μικρότερο αριθμό μηνυμάτων. Αυτό γίνεται διότι πλέον, αντίγραφα του αρχείου βρίσκονται πολύ κοντά στον κόμβο που ζητά το αρχείο, συνεπώς μειώνονται κατά πολύ τα απαιτούμενα βήματα για την ανάκτησή του. Ωστόσο, η μέθοδος αυτή επιβάλλει ένα σημαντικό, ενδεχομένως, κόστος για την αντιγραφή του αρχείου. Αυτό συμβαίνει διότι, εάν το μονοπάτι που δημιουργείται έως ότου ικανοποιηθεί η αναζήτηση ενός αρχείου είναι μεγάλο, αυτό έχει ως συνέπεια τη δημιουργία μεγάλου αριθμού αντιγράφων που θα αφορούν το συγκεκριμένο αρχείο. Αν το αρχείο αυτό δεν είναι δημοφιλές, δηλαδή δεν έχει ζητηθεί από ικανοποιητικό αριθμό ομότιμων, τότε ίσως η απόδοση του δικτύου να μην είναι η επιθυμητή, καθώς θα έχει χρησιμοποιηθεί περιττός αποθηκευτικός χώρος για δεδομένα που ζητούνται σπάνια. Γεγονός είναι, όμως, ότι η διαθεσιμότητα του αρχείου αυξάνεται σημαντικά.

Μία ακόμη μέθοδος, ιδιαίτερα απλοϊκή, είναι η *αντιγραφή κατόχου (owner replication)* (Σχήμα 3.7). Με βάση αυτή τη μέθοδο αντιγραφής, όταν ένας κόμβος κάνει μια αναζήτηση για ένα αρχείο, τότε μόνο αυτός κρατά ένα αντίγραφο του αρχείου αυτού με το πέρας της αναζήτησης. Η μέθοδος αυτή έχει το πλεονέκτημα ότι τα πιο δημοφιλή αρχεία, δηλαδή αυτά που έχουν ζητηθεί περισσότερο, θα έχουν μεγαλύτερη διαθεσιμότητα από εκείνα τα αρχεία που ζητούνται σπανιότερα. Αυτό έχει ως αποτέλεσμα τη μη σπατάλη μεγάλου αποθηκευτικού χώρου για σπάνια αρχεία, όπως γίνεται στην αντιγραφή μονοπατιού. Βέβαια, υπάρχει και το μειονέκτημα ότι ένα αρχείο που δεν έχει ζητηθεί θα έχει πολύ μικρό πλήθος αντιγράφων στο δίκτυο. Αυτό

εγκυμονεί τον κίνδυνο ότι αν φύγουν οι κόμβοι που το έχουν, το αρχείο αυτομάτως γίνεται μη διαθέσιμο.



Σχήμα 3.7 Αντιγραφή κατόχου. Οι ON1 και ON2 εξυπηρετούνται από τον ON3 που έχει το ζητούμενο αρχείο (file) και κρατούν από ένα αντίγραφό του

Στον πίνακα 3.1 εμφανίζονται συγκεντρωτικά τα πλεονεκτήματα και μειονεκτήματα των τριών μεθόδων αντιγραφής που παρουσιάστηκαν.

Πίνακας 3.1 Πλεονεκτήματα- Μειονεκτήματα των μεθόδων Αντιγραφής

Αντιγραφή στους γειτονικούς κόμβους	
Πλεονεκτήματα	Μειονεκτήματα
Ικανοποιητική διαθεσιμότητα αρχείων Γρήγορη ανάκτηση αρχείων από κόμβους που βρίσκονται κοντά σε κόμβους που προηγουμένως είχαν ανακτήσει το αρχείο	Αύξηση περιττού αποθηκευτικού κόστους για σπάνια αρχεία Απαιτείται μεγάλο πλήθος βημάτων για κόμβους που βρίσκονται μακριά τοπολογικά από κόμβους που προηγουμένως είχαν ανακτήσει το αρχείο Μικρό ποσοστό επιτυχίας ικανοποίησης αναζητήσεων

Αντιγραφή Μονοπατιού	
Πλεονεκτήματα	Μειονεκτήματα
Ικανοποιητική διαθεσιμότητα αρχείων Δεν υπάρχει περιορισμός ύπαρξης αντιγράφων σε μια μικρή ομάδα κόμβων Εύκολη και γρήγορη ανάκτηση αρχείων από κόμβους που βρίσκονται κοντά σε μονοπάτι προηγούμενης εξυπηρέτησης του αρχείου Μεγάλο ποσοστό επιτυχίας αναζήτησης με μικρό TTL	Αύξηση περιττού αποθηκευτικού κόστους για σπάνια αρχεία Εφαρμόζεται κυρίως σε δίκτυα αδόμητα και μη κεντρικοποιημένα, καθώς σε δίκτυα υπερ-κόμβων η δρομολόγηση γίνεται με αναζήτηση εγγραφών (index search) οπότε δεν ευνοείται η ύπαρξη μονοπατιού.

Αντιγραφή Κατόχου	
Πλεονεκτήματα	Μειονεκτήματα
Αυξημένη διαθεσιμότητα δημοφιλών αρχείων Ομοιόμορφη κατανομή αντιγράφων κατά μήκος του δικτύου Πολύ γρήγορη εξυπηρέτηση αναζητήσεων για δημοφιλή αρχεία Αύξηση απόδοσης και καλή εξισορρόπηση φόρτου (load balancing) καθώς το πλήθος των κόμβων που εξυπηρετούν την αναζήτηση είναι ανάλογο με το ρυθμό αναζήτησης του αρχείου	Πολύ μικρή διαθεσιμότητα για αρχεία που δεν ζητούνται σπάνια

ΚΕΦΑΛΑΙΟ 4. ΚΩΔΙΚΟΠΟΙΗΣΗ ΑΠΑΛΟΙΦΗΣ ΣΕ XML ΕΓΓΡΑΦΑ

4.1 Extensible Markup Language (XML)

Η XML (eXtensible Markup Language – Επεκτάσιμη Γλώσσα Σήμανσης), είναι μια γλώσσα που αναπτύχθηκε από το W3C τον Φεβρουάριο του 1998 και αποτελεί το πρότυπο για σήμανση (markup) εγγράφων, δηλαδή παρέχει επιπλέον πληροφορίες οι οποίες περιγράφουν και χαρακτηρίζουν το περιεχόμενο του εγγράφου. Η XML καθορίζει ένα γενικό συντακτικό το οποίο χρησιμοποιείται για να περιγράψουμε δεδομένα με τη χρήση απλών, κατανοητών και καθοριζόμενων από τους ανθρώπους, ετικετών (tags). Οι ετικέτες είναι αλφαριθμητικά (string), όπως για παράδειγμα το <*BODY*> της HTML, τα οποία περιγράφουν τα δεδομένα που είναι αποθηκευμένα σε ένα XML έγγραφο.

Η XML παρέχει ένα πρότυπο για έγγραφα υπολογιστών το οποίο είναι αρκετά ευέλικτο ώστε να μπορεί να ενσωματωθεί σε χώρους όπως οι Ιστοσελίδες (web pages), ηλεκτρονική ανταλλαγή δεδομένων, γραφικά, σειριοποίηση αντικειμένων (object serialization),συστήματα voice mail και άλλα. Προσφέρει έναν απλό τρόπο για να κωδικοποιήσουμε τόσο κείμενο όσο και δεδομένα και ευνοεί την ανταλλαγή και δημοσίευση ετερογενών δεδομένων, δηλαδή δεδομένων διαφορετικού τύπου και

κωδικοποίησης. Επιπλέον, δίνει τη δυνατότητα στους συγγραφείς ενός εγγράφου XML να περιγράψουν το περιεχόμενο του εγγράφου τους εύκολα και χωρίς μεγάλο κόπο, αφού μπορούν να φτιάξουν τις δικές τους ετικέτες, ώστε να περιγράψουν συγκεκριμένη πληροφορία, καθώς και να προσθέσουν δομή στα έγγραφά τους. Οι πληροφορίες που είναι αποθηκευμένες ως XML δεδομένα, μπορούν να διακινούνται σε διάφορες πλατφόρμες και λειτουργικά συστήματα και μπορούν να αξιοποιηθούν από πολλές εφαρμογές και υπολογιστικά εργαλεία, όπως για παράδειγμα στις μηχανές αναζήτησης (π.χ. Google). Για τους λόγους αυτούς η XML χρησιμοποιείται όλο και περισσότερο για διακίνηση πληροφορίας στο Διαδίκτυο.

Τα βασικά δομικά μέρη ενός XML κειμένου είναι τα *Στοιχεία (Elements)*. Ένα Στοιχείο περικλείεται ανάμεσα στις *ετικέτες αρχής και τέλους (opening and closing tags)*. Η ετικέτα αρχής περιέχει το όνομα του Στοιχείου ανάμεσα στα σύμβολα "<" και ">" (π.χ. `<first_name>, <profession>`) ενώ στην ετικέτα τέλους προστίθεται και το σύμβολο "/" μετά το "<". (π.χ `</first_name>,</profession>`). Τα Στοιχεία χρησιμοποιούνται για να περιγράψουν το κείμενο το οποίο περιέχουν. Η XML μας δίνει την ελευθερία να δημιουργήσουμε στοιχεία με όποιο όνομα θέλουμε, κάτι πολύ χρήσιμο διότι μπορεί ο καθένας που διαβάσει το XML έγγραφο, να κατανοήσει τα δεδομένα (κείμενο) που περιέχει από τα ονόματα των στοιχείων που χαρακτηρίζουν κάθε τμήμα του κειμένου. Για παράδειγμα, η παρακάτω πρόταση που είναι ένα απλό κείμενο :

" Alan Turing is a computer scientist"

μπορεί να αποδοθεί σε XML μορφή όπως παρακάτω :

```
<person>
<name>
<first_name> Alan </first_name>
<last_name> Turing </last_name>
</name>
<profession> computer scientist </profession>
</person>
```

Παρατηρώντας το συγκεκριμένο έγγραφο, όταν βλέπουμε ότι το κείμενο `computer scientist` περιέχεται στο στοιχείο `<profession>`, αντιλαμβανόμαστε ότι δηλώνει το επάγγελμα κάποιου και συγκεκριμένα κάποιου με όνομα `Alan` και επώνυμο `Turing`. Ακόμη, κοιτώντας το αρχικό στοιχείο (root element) `<person>`, το οποίο περιέχει όλα τα στοιχεία του εγγράφου, καταλαβαίνουμε ότι η πληροφορία του XML εγγράφου περιγράφει κάποιο πρόσωπο. Συμπεραίνουμε, λοιπόν, ότι η ελευθερία που παρέχει η XML για την ονομασία των στοιχείων είναι πολύ σημαντική για την εύκολη ανάγνωση και κατανόηση των XML εγγράφων από οποιονδήποτε.

Μια ακόμη δυνατότητα που προσφέρει η XML είναι το γεγονός ότι τα στοιχεία μπορούν να είναι εμφωλευμένα μέσα σε άλλα στοιχεία . Το γεγονός αυτό επιτρέπει στο συγγραφέα του XML εγγράφου να δομήσει την πληροφορία όπως αυτός επιθυμεί και να κάνει το έγγραφο ακόμη πιο κατανοητό. Για παράδειγμα, το στοιχείο `<name>` περιέχει τα στοιχεία `<first_name>, <last_name>,` κάτι που δηλώνει ότι τα εμφωλευμένα στοιχεία περιέχουν δεδομένα τα οποία χαρακτηρίζουν ένα όνομα (name), όπως υποδηλώνει το στοιχείο `<name>` που τα περιλαμβάνει. Μπορούμε να χαρακτηρίσουμε την εμφώλευση των στοιχείων μέσα σε άλλα στοιχεία ως μια σχέση πατέρα- παιδιού, υπό την έννοια ότι το παιδί (εμφωλευμένο στοιχείο) ανήκει και χαρακτηρίζει το πατρικό στοιχείο. Για να είναι καλά ορισμένο και συντακτικά ορθό ένα XML έγγραφο, πρέπει να υπάρχει ένα στοιχείο το οποίο να περικλείει στις ετικέτες αρχής και τέλους του όλα τα υπόλοιπα στοιχεία. Το αρχικό αυτό στοιχείο (root element) δεν μπορεί να είναι εμφωλευμένο σε κάποιο άλλο στοιχείο και συνεπώς αποτελεί τον πατέρα για όλα τα υπόλοιπα στοιχεία που εμφανίζονται μέσα στο έγγραφο.

Στα Δίκτυα Ομότιμων, η διακίνηση XML εγγράφων μεταξύ των χρηστών αυξάνεται σημαντικά μέσα στο χρόνο. Οι δυνατότητες και οι πληροφορίες που μπορούν να εκφραστούν μέσα από τέτοια έγγραφα καθιστούν την αξιόπιστη διαχείριση των XML δεδομένων σε Δίκτυα Ομότιμων μια ουσιαστική και σημαντική διαδικασία, η οποία αποτελεί κύριο τομέα έρευνας τα τελευταία χρόνια. Ιδιαίτερα η δημιουργία και αποθήκευση αντιγράφων XML δεδομένων στα δίκτυα για την αύξηση της διαθεσιμότητάς τους και εν γένει την βιωσιμότητα και αξιοπιστία των ίδιων των

δικτύων, έχει γνώμονα την εύκολη και γρήγορη πρόσβαση σε αυτά από τους διάφορους χρήστες. Ωστόσο, υπάρχουν κάποιοι παράγοντες που πρέπει να ληφθούν υπ' όψιν για την καλύτερη διανομή των XML εγγράφων. Ένας από αυτούς είναι το γεγονός ότι πρέπει να διατηρείται η σειρά (κατάταξη) εμφάνισης των Στοιχείων μέσα στο έγγραφο. Η δομή των διαφόρων στοιχείων που περικλείουν άλλα εμφωλευμένα στοιχεία δεν πρέπει να καταστρατηγείται, διότι σε διαφορετική περίπτωση κινδυνεύει να χαθεί η πληροφορία που περικλείεται. Ένας άλλος παράγοντας είναι το γεγονός ότι ορισμένοι χρήστες επιθυμούν την ανάκτηση ενός μόνο τμήματος πληροφορίας που περικλείεται σε ένα XML έγγραφο και όχι ολόκληρη την απόκτηση του εγγράφου. Για παράδειγμα, σε ένα έγγραφο που περιλαμβάνει πληροφορίες για πολλά τραγούδια και τους καλλιτέχνες τους, κάποιος χρήστης μπορεί να ενδιαφέρεται μόνο για την πληροφορία ενός τραγουδιού, το οποίο περιγράφεται ως ξεχωριστό στοιχείο μέσα στο έγγραφο. Είναι χρήσιμο, επομένως, να δίνεται η δυνατότητα στον χρήστη να ανακτά μόνο τις πληροφορίες των στοιχείων ενός εγγράφου που τον ενδιαφέρουν και όχι απαραίτητα να αναγκάζεται να αποκτήσει ολόκληρο το έγγραφο. Αυτό ωφελεί στο γεγονός ότι δεν υπάρχει περιττή μεταφορά πληροφορίας μεταξύ των χρηστών, κάτι που μπορεί να έχει αρνητικό αντίκτυπο στην γενική απόδοση του δικτύου. Ας μην ξεχνάμε ότι οι πόροι ενός δικτύου (π.χ. το εύρος δικτύου) είναι σημαντικό να μην χρησιμοποιούνται άσκοπα, καθώς μπορεί να βρίσκονται συνδεδεμένοι με αυτό χιλιάδες ή και εκατομμύρια κόμβοι (χρήστες).

4.2 Δενδρική Δομή XML Εγγράφων

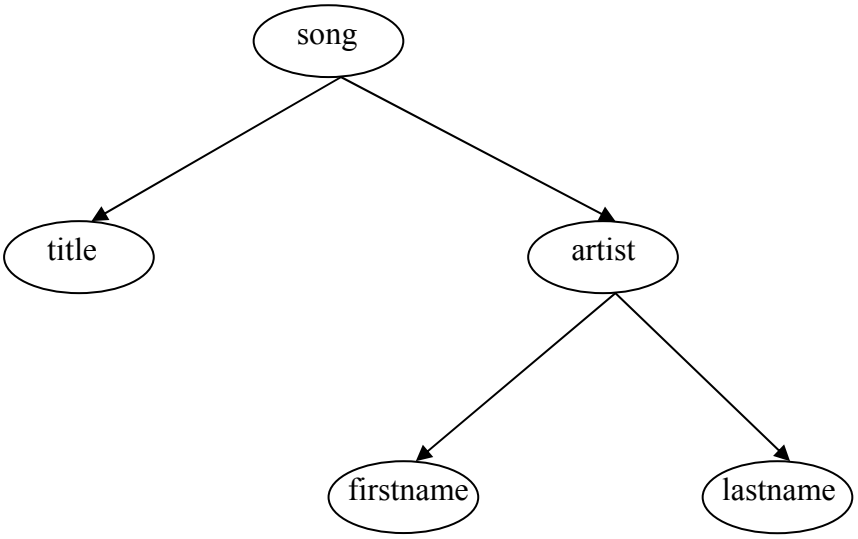
Ειπώθηκε στην προηγούμενη ενότητα ότι ορισμένοι χρήστες αναζητούν ένα μέρος της πληροφορίας που περιέχει ένα XML έγγραφο το οποίο βρίσκεται αποθηκευμένο σε ένα Δίκτυο Ομότιμων. Η ανάγκη αυτή καθιστά σημαντική την κατάτμηση των XML εγγράφων και την αποθήκευση των διαφόρων τμημάτων που προκύπτουν στους κόμβους του Δικτύου. Μπορούμε να αντιμετωπίσουμε κάθε τμήμα ως αυτόνομο έγγραφο που περιλαμβάνει πληροφορία χρήσιμη και ανεξάρτητη από αυτή των υπόλοιπων τμημάτων. Οφείλουμε, ωστόσο, να μην παραλείψουμε το γεγονός ότι δεν πρέπει η κατάτμηση αυτή να διαφοροποιήσει την καθορισμένη σειρά με την οποία εμφανίζονται τα στοιχεία μέσα στο έγγραφο.

Ένα XML έγγραφο μπορεί να αποδοθεί σχηματικά ως ένα σύνολο κόμβων οι οποίοι συνδέονται μεταξύ τους μέσω μιας συγκεκριμένης δενδρικής δομής. Κάθε κόμβος στο δέντρο είναι ένα στοιχείο του κειμένου και η ακμή που ενώνει δύο κόμβους υποδηλώνει τη σχέση πατέρα- παιδιού που υπάρχει μεταξύ τους. Στο αρχικό επίπεδο του δέντρου βρίσκεται το αρχικό στοιχείο του εγγράφου, το οποίο αποτελεί τον πατέρα για όλα τα υπόλοιπα στοιχεία. Στη συνέχεια ο αρχικός κόμβος ενώνεται μέσω ακμών με κάθε κόμβο (στοιχείο) που αποτελεί άμεσο παιδί του και όχι με οποιονδήποτε απόγονο. Για να γίνει πιο κατανοητή αυτή η παρατήρηση ας θεωρήσουμε ότι ο αρχικός κόμβος είναι ο D1, ο οποίος περικλείει ακριβώς από κάτω του τον κόμβο D2. Ο D2, ωστόσο, περιέχει επίσης ένα εμφωλευμένο κόμβο (στοιχείο), τον D3. Στη δενδρική δομή που αναπαριστά την κατάσταση αυτή, ο D1 θα ενωθεί άμεσα μέσω ακμής με τον D2, όχι όμως και με τον D3. Δηλαδή, ο D3 αποτελεί έναν απόγονο του D1, όχι όμως και άμεσο παιδί του. Ο D3 θα ενωθεί μέσω ακμής με τον D2, υποδηλώνοντας έτσι τη σχέση μεταξύ τους ως πατέρας- παιδί.

Ας θεωρήσουμε ότι έχουμε το ακόλουθο XML έγγραφο (έστω με τίτλο “song.xml”):

```
<song>
<title> Beautiful </title>
<artist>
<firstname> James </firstname>
<lastname> Blunt </lastname>
</artist>
</song>
```

Στο έγγραφο αυτό το στοιχείο <song> είναι ο αρχικός κόμβος, ή κόμβος- ρίζα (root node), ο οποίος έχει δύο παιδιά, τα <title> και <artist>. Το στοιχείο <artist> έχει επίσης δύο παιδιά, τα <firstname> και <lastname>. Η δενδρική δομή του συγκεκριμένου XML εγγράφου είναι αυτή που φαίνεται στο Σχήμα 4.1.



Σχήμα 4.1. Δενδρική δομή XML εγγράφου

Παρατηρούμε από τη δενδρική δομή του Σχήματος 4.1 ότι στο πρώτο επίπεδο του δέντρου βρίσκεται ο αρχικός κόμβος (<song>), ενώ τα παιδιά του (<title> και <artist>) βρίσκονται στο δεύτερο επίπεδο. Βρίσκονται, ωστόσο, στο ίδιο επίπεδο καθώς έχουν κοινό πατέρα. Στο τρίτο επίπεδο βρίσκονται τα παιδιά του κόμβου <artist>, δηλαδή τα <firstname> και <lastname>, τα οποία επίσης έχουν κοινό πατέρα. Τα δύο αυτά στοιχεία μαζί με το στοιχείο <title> αποτελούν τα φύλλα του δέντρου, διότι δεν υπάρχουν άλλοι κόμβοι που να συνδέονται με αυτούς σε κατώτερο επίπεδο.

Ένας κόμβος στο XML δέντρο μπορεί να χαρακτηριστεί από το όνομα του στοιχείου το οποίο αναπαριστά. Για παράδειγμα, ο κόμβος <title> που υπάρχει στο δέντρο, μπορεί να χαρακτηριστεί και να προσπελασθεί με βάση το όνομα του στοιχείου το οποίο περιγράφει, δηλαδή το title. Ωστόσο, πολύ χρήσιμο θα ήταν να μπορούσαμε να περιγράψουμε κάθε κόμβο στο δέντρο με ένα μοναδικό τρόπο ο οποίος θα υποδεικνύει και τη σχετική θέση που κατέχει ο κόμβος για το συγκεκριμένο αρχείο στο οποίο αναφέρεται. Ένας από τους δημοφιλέστερους τρόπους για το χαρακτηρισμό των κόμβων μέσα σε ένα XML δέντρο είναι η περιγραφή του *μονοπατιού (path)* που πρέπει να ακολουθήσουμε από τον αρχικό κόμβο έως τον κόμβο που θέλουμε να προσπελάσουμε. Το μονοπάτι που περιγράφει ένα κόμβο el_nστο δέντρο είναι ένα αλφαριθμητικό το οποίο έχει την εξής μορφή

$Path(el_n)= /el_1/el_2/\ldots/el_n$

Σχήμα 4.2 Μορφή μονοπατιού ενός XML υποδέντρου

Το μονοπάτι κάθε κόμβου εκφράζει τη διαδρομή που πρέπει να ακολουθηθεί στο δέντρο από τον αρχικό πατρικό κόμβο έως το συγκεκριμένο κόμβο που επιθυμούμε. Η διαδρομή αυτή εκφράζεται ως το σύνολο των πατρικών κόμβων (στοιχείων) που πρέπει να προσπελάσουμε ώστε να καταλήξουμε σε αυτόν που θέλουμε. Η ακμή που ενώνει δύο κόμβους el_1 και el_2 σε ένα δέντρο σημειώνεται με το χαρακτήρα "/", δηλαδή εκφράζεται από τη μορφή el_1/el_2. Ο χαρακτήρας '/" υποδηλώνει τη σύνδεση που έχουν δύο κόμβοι στο δέντρο (δηλαδή τη σχετική θέση των στοιχείων στο έγγραφο), ενώ το σύνολο των χαρακτήρων αυτών που υπάρχουν σε ένα μονοπάτι δηλώνει και το βάθος στο οποίο πρέπει να κατέβουμε ώστε να βρούμε τον κόμβο που μας ενδιαφέρει.

Για παράδειγμα, τα μονοπάτια που χαρακτηρίζουν κάθε κόμβο του XML δέντρου του Σχήματος 4.1 είναι τα ακόλουθα:

Path(song)= /song
Path(title)=/ song/ title
Path(artist)=/ song/ artist
Path(firstname)=/ song/ artist/ firstname
Path(lastname)=/ song/ artist/ lastname

Παρατηρούμε ότι η ανάθεση του μονοπατιού ως χαρακτηριστικό για τους κόμβους του δέντρου παρέχει χρήσιμες πληροφορίες. Μέσω του μονοπατιού, εκτός από το όνομα του στοιχείου που περιγράφει ο κόμβος, περιγράφεται και η σχετική θέση που κατέχει ο κόμβος στο δέντρο, δηλαδή η σχετική θέση που κατέχει το συγκεκριμένο στοιχείο μέσα στο έγγραφο. Επιπλέον, μέσω των μονοπατιών είναι πιο εύκολο να προσπελάσουμε ένα μέρος μόνο της πληροφορίας που υπάρχει σε ένα XML έγγραφο ακολουθώντας την αλληλουχία των κόμβων προς τον προορισμό μας, παραβλέποντας πληροφορία που δεν χρειαζόμαστε. Αυτό είναι ιδιαίτερα σημαντικό κατά την

αποθήκευση και διακίνηση XML πληροφορίας σε ένα δίκτυο ομότιμων, καθώς με τον έλεγχο των μονοπατιών μπορούμε να ανακτήσουμε ακριβώς την πληροφορία που ζητά η επερώτηση (query) που κάνουμε, χωρίς την άσκοπη προσπέλαση πληροφορίας που δεν μας ενδιαφέρει. Την ύπαρξη μονοπατιών ως χαρακτηριστικά των κόμβων ενός XML δέντρου την εκμεταλλεύονται διάφορες μέθοδοι επερωτήσεων σε δίκτυα ομότιμων, όπως η XQuery, προσαρμόζοντας τη μορφή των αναζητήσεων στη μορφή που έχουν τα μονοπάτια. Δηλαδή οι αναζητήσεις που γίνονται για κάποιο στοιχείο έχουν τη μορφή ενός μονοπατιού που υποδηλώνει το στοιχείο (ή τα στοιχεία) που επιθυμούμε. Όταν υπάρχει ταύτιση των μονοπατιών της επερώτησης και του χαρακτηριστικού που συνοδεύει τους κόμβους, τότε έχουμε επιτυχία της αναζήτησης και γρήγορη εξυπηρέτησή της.

4.3 Κατάτμηση XML εγγράφων

Στα δίκτυα ομότιμων είναι χρήσιμη, όπως αναφέρθηκε, η ανάκτηση συγκεκριμένης πληροφορίας που υπάρχει μέσα σε ένα XML έγγραφο είναι ουσιώδους σημασίας για την καλύτερη απόδοση και βιωσιμότητά του. Ένα XML έγγραφο μπορεί να περιέχει μεγάλου πλήθους πληροφορία, η οποία δεν είναι πάντα επιθυμητή. Κάποιος χρήστης πιθανώς ενδιαφέρεται για ένα μικρό κομμάτι της περικλειόμενης πληροφορίας και όχι για ολόκληρη. Είναι επομένως ιδιαίτερα αποδοτικό να μπορεί ο χρήστης να ανακτήσει την πληροφορία, δηλαδή το συγκεκριμένο στοιχείο ή σύνολο στοιχείων, που επιθυμεί δίχως να απαιτείται η ανάκτηση ολόκληρου του εγγράφου. Είναι προφανές ότι με αυτή τη δυνατότητα δεν επιβαρύνεται το δίκτυο με αναζήτηση και αποστολή περιττής πληροφορίας και επιπλέον ο χρήστης ανακτά ακριβώς αυτό που ζητά και όχι κάτι επιπλέον. Η βελτίωση της απόδοσης στο δίκτυο είναι σημαντική, όπως και η εξυπηρέτηση των διαφόρων επερωτήσεων έχει σαφώς μεγαλύτερο ποσοστό επιτυχίας. Για τους λόγους αυτούς, μία μέθοδος που χρησιμοποιούμε για την αποθήκευση των XML εγγράφων σε ένα δίκτυο ομότιμων είναι η *Κατάτμηση (fragmentation)* των εγγράφων αυτών σε μικρότερα τμήματα και η αποθήκευση των τμημάτων στους κόμβους του δικτύου. Τα τμήματα αυτά μπορούν να περιέχουν αυτόνομη πληροφορία, δηλαδή πληροφορία που δεν απαιτεί την υποχρεωτική ανάκτηση και άλλων τμημάτων, ή ενδέχεται να αποτελούν μέρος ενός μεγαλύτερου όγκου πληροφορίας που βρίσκεται αποθηκευμένος και σε άλλα τμήματα. Το ποσό της πληροφορίας που εξυπηρετείται

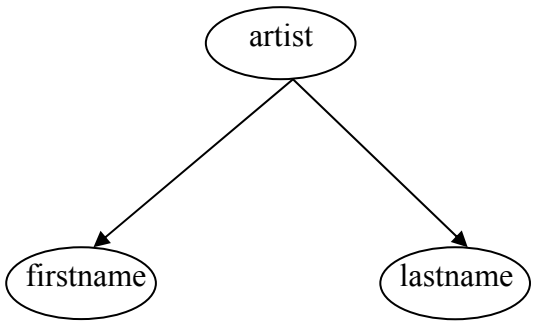
από ένα τμήμα XML εγγράφου, σχετίζεται με την επερώτηση που κάνει κάποιος κόμβος. Εάν αναζητείται ένα μικρό ποσοστό του εγγράφου, τότε ενδεχομένως να μπορεί να εξυπηρετηθεί από ένα μόνο τμήμα, ενώ αν αναζητείται μεγαλύτερος όγκος πληροφορίας, ίσως να απαιτείται συνδυασμός τμημάτων για την εξυπηρέτησή της. Γεγονός είναι, ωστόσο, ότι με τη διαδικασία της κατάτμησης οι χρήστες έχουν πρόσβαση σε οποιοδήποτε τμήμα πληροφορίας ενδιαφέρονται χωρίς την υποχρεωτική απόκτηση ολόκληρου του εγγράφου.

Η κατάτμηση του XML εγγράφου που υλοποιήθηκε στην παρούσα εργασία σχετίζεται άμεσα με τα στοιχεία που υπάρχουν μέσα σε αυτό. Αυτό οφείλεται στο γεγονός ότι κάθε στοιχείο σε ένα XML έγγραφο (μαζί με τα εμφωλευμένα σε αυτό στοιχεία, αν έχει) περιγράφει ένα κομμάτι πληροφορίας το οποίο μπορεί αυτόνομα να αναζητηθεί από κάποιον χρήστη. Είναι κατανοητό, ωστόσο, ότι ένα στοιχείο σε ένα XML έγγραφο το οποίο περιέχει και άλλα εμφωλευμένα στοιχεία, δεν μπορεί μόνο του να περιγράψει κάποια πληροφορία. Χρειάζεται, δηλαδή, η ύπαρξη και των εμφωλευμένων σε αυτό στοιχείων και μάλιστα με τη σωστή διάταξη (ως παιδιά του) μέσα στο έγγραφο, ώστε η πληροφορία να είναι ολοκληρωμένη και έγκυρη. Στο δέντρο του Σχήματος 4.1, για παράδειγμα, ο κόμβος <artist> μόνος του δεν μας δίνει κάποια χρήσιμη και επεξεργάσιμη πληροφορία. Περιγράφει απλώς ένα στοιχεία το οποίο δεν περιέχει κάποιο κείμενο, αλλά δύο επιπλέον στοιχεία (τα <firstname> και <lastname>). Για να λάβουμε την πληροφορία που περιγράφει ο κόμβος <artist>, οφείλουμε να λάβουμε ταυτόχρονα και τους κόμβους- παιδιά του, δηλαδή τους <firstname> και <lastname>.

Το σύνολο των κόμβων <artist>, <firstname> και <lastname> μαζί με τις ακμές που υποδηλώνουν τις σχέσεις πατέρα- παιδιών, αποτελούν ένα *υποδέντρο (subtree)* του δέντρου που περιγράφει το έγγραφο “song.xml”. Ένα υποδέντρο ενός XML δέντρου είναι ένα σύνολο από κόμβους και ακμές που τους ενώνουν, στο οποίο οι κόμβοι συνδέονται μεταξύ τους με σχέσεις πατέρα- παιδιού ή προγόνου- απογόνου. Ως πρόγονος ενός XML στοιχείου el1 θεωρείται ένα στοιχείο el2 το οποίο βρίσκεται σε υψηλότερο επίπεδο στο δέντρο και το οποίο ενώνεται μέσω μονοπατιού με τον πατέρα του el1. Δηλαδή το el2 περιλαμβάνει στο εσωτερικό του τον πατέρα του el1 και συνεπώς και το ίδιο το el1. Στο Σχήμα 4.1, για παράδειγμα, ο κόμβος <song> είναι πρόγονος του <lastname>, καθώς ενώνεται με τον πατέρα του (<artist>) μέσω ενός

μονοπατιού (/song/artist), δηλαδή περιέχει εμφωλευμένο στο εσωτερικό του το στοιχείο <artist>. Ωστόσο, το στοιχείο <title> δεν είναι πρόγονος του <lastname>, καθώς δεν ενώνεται με μονοπάτι με τον πατέρα του (<artist>), δηλαδή το <artist> δεν βρίσκεται εμφωλευμένο στο <title> μέσα στο έγγραφο. Αντίστροφα, ο κόμβος <lastname> είναι απόγονος του <song>, δηλαδή είναι παιδί ενός παιδιού (<artist>) του <song>. Απόγονος ενός κόμβου el1 σε ένα XML δέντρο, είναι ένας κόμβος el2, ο οποίος βρίσκεται σε χαμηλότερο επίπεδο στο δέντρο και ενώνεται έμμεσα μέσω ενός μονοπατιού (και όχι άμεσα μέσω μιας ακμής) με τον el1. Ο el1, δηλαδή, δεν είναι πατέρας του el2, αλλά πρόγονός του.

Επιπλέον, ένα υποδέντρο περιγράφει κάποια συγκεκριμένη πληροφορία η οποία είναι αυτόνομη, δηλαδή δεν απαιτείται η προσθήκη και άλλων κόμβων, ώστε να είναι ολοκληρωμένη. Για να γίνει κατανοητός ο ορισμός του υποδέντρου, στο σχήμα 4.2 βλέπουμε ένα υποδέντρο του δέντρου του Σχήματος 4.1.



Σχήμα 4.3. Υποδέντρο του δέντρου του Σχήματος 4.1. Ο κόμβος <artist> είναι ο αρχικός κόμβος (root node) του υποδέντρου και οι <firstname>, <lastname> τα φύλλα του.

Το υποδέντρο του Σχήματος 4.3 έχει ως αρχικό κόμβο τον <artist> και ως φύλλα τα παιδιά του. Ένα υποδέντρο, βέβαια, ενός δέντρου μπορεί να είναι και ένα φύλλο του. Δεν είναι απαραίτητο να χαρακτηρίζουμε ως υποδέντρο μόνο κόμβους που έχουν κάποια παιδιά. Για παράδειγμα, στο Σχήμα 4.1, ο κόμβος <firstname> αποτελεί υποδέντρο (δηλαδή υποσύνολο) του XML δέντρου και ας μην έχει παιδιά. Μπορούμε να φανταστούμε τα υποδέντρα ως ξεχωριστά XML έγγραφα και επομένως μπορούν να

αποθηκευτούν αλλά και να αναζητηθούν αυτόνομα στο δίκτυο. Κάποια υποδέντρα ενός XML δέντρου μπορεί να έχουν μεγαλύτερο ρυθμό αναζητήσεων από άλλα στο ίδιο δέντρο. Αυτό, πρακτικά, σημαίνει ότι κάποια τμήματα ενός εγγράφου περιέχουν πληροφορία που είναι πιο χρήσιμη από αυτή που υπάρχει αλλού στο ίδιο έγγραφο και συνεπώς είναι λογικό να τη ζητούν περισσότεροι χρήστες. Είναι ουσιαστικό για την απόδοση του δικτύου οι χρήστες αυτοί να μπορούν να προσπελάσουν τη συγκεκριμένη πληροφορία χωρίς να απαιτείται να ανακτήσουν ολόκληρο το έγγραφο, δηλαδή όλους τους κόμβους του δέντρου. Αυτή τη δυνατότητα την παρέχουν τα υποδέντρα, αποθηκεύοντάς τα ξεχωριστά κάπου στο δίκτυο. Κάποιος που επιθυμεί την πληροφορία που περικλείεται σε ένα υποδέντρο, μπορεί να λάβει μόνο την πληροφορία του και όχι ολόκληρο το έγγραφο. Προκύπτει, λοιπόν, ότι τα υποδέντρα αποτελούν ένα βολικό τρόπο αποθήκευσης XML εγγράφων σε ένα δίκτυο ομότιμων.

Κάθε υποδέντρο μέσα σε ένα XML δέντρο χαρακτηρίζεται από το μονοπάτι που ξεκινά από τον αρχικό κόμβο του δέντρου και καταλήγει στον αρχικό κόμβο του υποδέντρου. Είναι σημαντικό να υπάρχει τέτοιου είδους αναγνωριστικό για κάθε υποδέντρο, έτσι ώστε να είναι εφικτή η αναζήτησή του από κάποιο χρήστη του δικτύου. Στο δέντρο του Σχήματος 4.1, το υποδέντρο του Σχήματος 4.2 έχει ως αναγνωριστικό το μονοπάτι από τον αρχικό κόμβο <song> έως τον κόμβο <artist>, δηλαδή είναι το /song/artist. Αν κάποιος επιθυμήσει το συγκεκριμένο τμήμα του εγγράφου, θα πρέπει να το αναζητήσει με βάση αυτό το αναγνωριστικό.

Η διαδικασία δημιουργίας των τμημάτων που αποθηκεύονται στους κόμβους του δικτύου είναι η ακόλουθη. Αρχικά, οι πληροφορίες που αποθηκεύει κάθε τμήμα είναι α) το όνομα του υποδέντρου που περιγράφεται και β) το κείμενο που περικλείει. Αν το υποδέντρο που περιγράφεται από το τμήμα είναι ένας κόμβος- φύλλο του δέντρου, τότε το όνομα ισούται με το όνομα του συγκεκριμένου κόμβου. Αν, ωστόσο, το υποδέντρο αποτελείται από περισσότερους κόμβους, τότε το όνομα ισούται με το όνομα του αρχικού κόμβου του υποδέντρου. Για παράδειγμα, στο Σχήμα 4.1, αν το τμήμα περιγράφει τον κόμβο- φύλλο <title>, τότε το τμήμα που θα αποθηκευτεί περιέχει τις παρακάτω πληροφορίες:
Name= title
Value= <title> Beautiful </title>

Αν το τμήμα περιγράφει το υποδέντρο του Σχήματος 4.2, τότε περιέχει τις παρακάτω πληροφορίες:
Name= artist
Value= <artist>
<firstname> James </firstname>
<lastname> Blunt </lastname>
</artist>

Παρατηρούμε, δηλαδή, ότι το πεδίο value περιέχει ολόκληρο το κείμενο του XML εγγράφου, αυτούσιο, το οποίο αντιστοιχεί στο τμήμα που δημιουργήθηκε. Δηλαδή, αν κάποιος αναζητήσει το τμήμα <artist> του εγγράφου "song.xml", θα λάβει αυτούσιο το κείμενο που υπάρχει στο στοιχείο <artist>, μαζί με τα στοιχεία- παιδιά του.

Σε κάθε τμήμα που δημιουργείται ανατίθεται ένα μοναδικό κλειδί (χαρακτηριστικό) το οποίο αναφέρεται στο XML έγγραφο, μέρος του οποίου είναι και το στοιχείο που περιγράφεται από το τμήμα. Το κλειδί αυτό ονομάζεται *fileID* και έχει ως τιμή το όνομα του αρχείου. Είναι προφανές ότι τα τμήματα που περιγράφουν στοιχεία (υποδέντρα) που βρίσκονται στο ίδιο έγγραφο (δέντρο) θα έχουν το ίδιο fileID, καθώς αναφέρονται στο ίδιο έγγραφο. Για παράδειγμα, αν το έγγραφο που περιγράφεται από το δέντρο του Σχήματος 4.1 έχει όνομα "song.xml", τότε αυτή θα είναι και η τιμή του fileID για όλα τα τμήματα. Αναζητήσεις που στοχεύουν στην ανάκτηση ολόκληρου του εγγράφου έχουν ως παράμετρο το όνομά του (δηλαδή το fileID). Συνεπώς αναζητούνται όλα τα τμήματα που συσχετίζονται με το συγκεκριμένο fileID και η ανάκτησή τους ολοκληρώνει με επιτυχία την αναζήτηση.

Ωστόσο, κάθε τμήμα συσχετίζεται επίσης και με ένα δεύτερο κλειδί, το οποίο ονομάζεται *pathID*, το οποίο είναι διαφορετικό για κάθε τμήμα. Το pathID είναι ένα αλφαριθμητικό, η τιμή του οποίου ισούται με το μονοπάτι (path) στο XML δέντρο του συγκεκριμένου αρχείου το οποίο ξεκινά από τον αρχικό κόμβο και καταλήγει στον κόμβο που περιγράφεται από το τμήμα. Για παράδειγμα, το pathID του τμήματος που περιγράφει τον κόμβο <title> του Σχήματος 4.1 λαμβάνει την τιμή pathID=/song/title. Η αντιστοίχηση κάθε τμήματος με ένα pathID χρησιμεύει αρχικά στο να ξεχωρίσουμε

τα τμήματα του ίδιου εγγράφου μεταξύ τους. Αφού το fileID με το οποίο αντιστοιχίζονται τα τμήματα είναι κοινό για όλα, το pathID αποτελεί το χαρακτηριστικό που τα ξεχωρίζει μεταξύ τους. Επιπλέον, τα pathIDs των τμημάτων, δίνουν τη δυνατότητα στις επερωτήσεις που μπορούν να γίνουν στο δίκτυο να αναζητήσουν συγκεκριμένους κόμβους ενός XML δέντρου χωρίς να χρειάζεται η ανάκτηση ολόκληρου του δέντρου. Αυτό δεν θα μπορούσε να γίνει μόνο με το fileID, το οποίο είναι κοινό για όλους τους κόμβους του ίδιου δέντρου. Η ύπαρξη, επομένως, του pathID σε κάθε τμήμα δίνει αυτή τη δυνατότητα η οποία, όπως αναφέρθηκε προηγουμένως, είναι ιδιαίτερα σημαντική για την απόδοση του δικτύου. Η αναζήτηση για συγκεκριμένο κόμβο (υποδέντρο) του δέντρου γίνεται με παράμετρο το επιθυμητό μονοπάτι για τον κόμβο που ζητείται και όταν υπάρξει ταυτοποίηση του μονοπατιού με το pathID ενός τμήματος, τότε επιτυγχάνεται η επερώτηση. Μάλιστα, με το pathID μπορούμε να ανακτήσουμε την επιθυμητή πληροφορία από πολλαπλά XML δέντρα, καθώς διαφορετικοί κόμβοι σε διαφορετικά δέντρα ενδέχεται να έχουν το ίδιο pathID. Για παράδειγμα, αν υπάρχουν στο δίκτυο δύο διαφορετικά XML έγγραφα που περιγράφουν τραγούδια, είναι πιθανό να υπάρχουν κόμβοι που έχουν το ίδιο μονοπάτι (π.χ. /song/title). Μπορούμε, επομένως, να ανακτήσουμε τα τραγούδια που περιγράφονται από τα δύο XML έγγραφα κάνοντας μια αναζήτηση με παράμετρο το συγκεκριμένο μονοπάτι. Αν ωστόσο επιθυμούμε να ανακτήσουμε τίτλους τραγουδιών που υπάρχουν σε ένα έγγραφο (π.χ. “song.xml”) μπορούμε να κάνουμε μια επερώτηση που να έχει παραμέτρους τόσο το fileID όσο και το μονοπάτι (pathID) των κόμβων που επιθυμούμε.

Με βάση την παραπάνω ανάλυση, τα τμήματα στα οποία χωρίζεται ένα XML δέντρο έχουν την ακόλουθη μορφή (Σχήμα 4.4):

FileID	PathID	Name	Value

Σχήμα 4.4. Μορφή τμημάτων που αποθηκεύονται στο δίκτυο

Στο δέντρο του Σχήματος 4.1 τα τμήματα που περιγράφουν τους κόμβους <title> και <artist> είναι τα εξής (Σχήμα 4.5):

Song.xml	/song/title	Title	<title> Beautiful </title>

Song.xml	/song/artist	Artist	<artist><firstname>James</firstname><lastname>Blunt</lastname></artist>

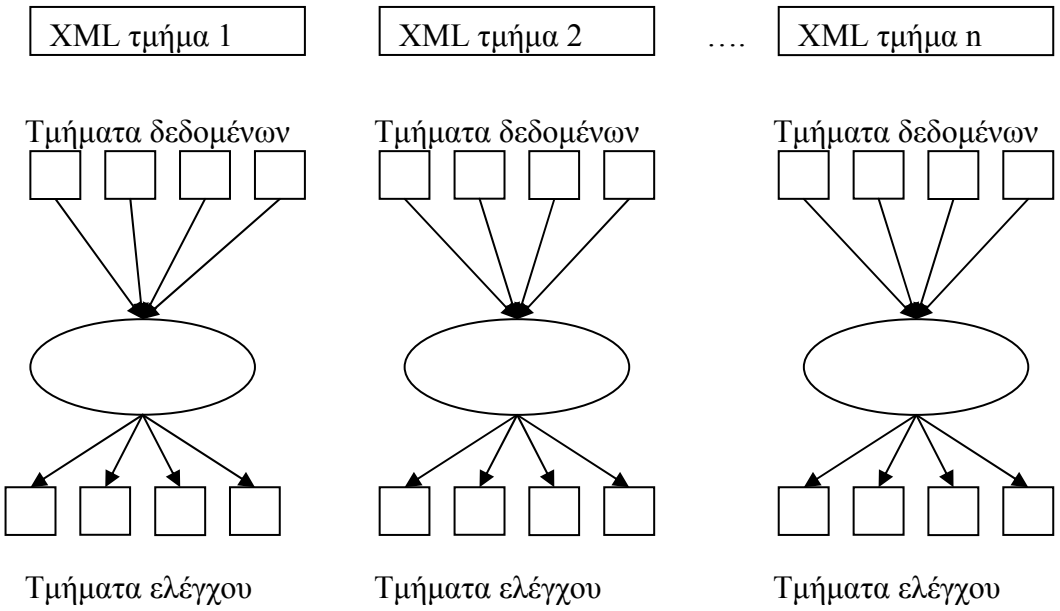
Σχήμα 4.5. Τα τμήματα που περιγράφουν τα υποδέντρα <title> και <artist> του δέντρου του Σχήματος 4.1

4.4 Κωδικοποίηση Απαλοιφής σε XML τμήματα

Στο προηγούμενο κεφάλαιο αναλύσαμε την αρχιτεκτονική του Δικτύου των Ομότιμων στο οποίο τα διάφορα αρχεία αποθηκεύονται με τη μέθοδο της Κωδικοποίησης Απαλοιφής και συγκεκριμένα με τη μέθοδο των LT κωδικών. Στην παρούσα ενότητα θα ασχοληθούμε με τη διαδικασία Απαλοιφής XML δεδομένων στο δίκτυό μας και τη χρήση των υπο-δέντρων.

Τα XML δέντρα τα οποία αναπαριστούν XML έγγραφα, είπαμε ότι χωρίζονται σε τμήματα τα οποία αποθηκεύονται στο δίκτυο και μπορούν αυτόνομα να εξυπηρετήσουν επερωτήσεις που αφορούν στην πληροφορία που περιγράφουν. Τα τμήματα αυτά αποτελούν τα υπο-δέντρα του XML δέντρου, δηλαδή κομμάτια του αρχείου που περιγράφουν μια συγκεκριμένη πληροφορία. Κάθε τέτοιο XML τμήμα αντιμετωπίζεται ως ξεχωριστό αρχείο από την Κωδικοποίηση Απαλοιφής. Συνεπώς, σε κάθε τέτοιο τμήμα εφαρμόζεται η μέθοδος των LT κωδικών, κωδικοποιώντας το σε νέα τμήματα δεδομένων και ελέγχου. Παρατηρούμε, δηλαδή, ότι αντί να εφαρμόσουμε την Κωδικοποίηση Απαλοιφής μία φορά στο σύνολο των στοιχείων του XML εγγράφου, την εφαρμόζουμε μεμονωμένα σε κάθε τμήμα του εγγράφου (Σχήμα 4.6). Αυτό γίνεται διότι αν θεωρούσαμε για τη μέθοδο των LT κωδικών ότι τα τμήματα του XML εγγράφου αποτελούσαν και τα τμήματα δεδομένων που θα υφίσταντο την κωδικοποίηση σε κάποια τμήματα ελέγχου, αυτό θα ήταν μη αποδοτικό για διάφορους λόγους. Ένας από τους λόγους αυτούς είναι το γεγονός ότι στην περίπτωση αυτή τα τμήματα ελέγχου που παράγονται κωδικοποιούν πάνω από ένα XML τμήμα. Αυτό

σημαίνει ότι για να ανακτήσουμε την πληροφορία που περιέχεται σε ένα XML τμήμα αναγκαζόμαστε να αποκτήσουμε τμήματα ελέγχου τα οποία κωδικοποιούν και τμήμα για το οποίο δεν ενδιαφερόμαστε. Αναγκαζόμαστε δηλαδή να ανακτήσουμε περιττή πληροφορία.



Σχήμα 4.6 . Κάθε XML τμήμα (υποδέντρο) υφίσταται την κωδικοποίηση των LT Κωδικών

Η ιδιομορφία των XML εγγράφων έγκειται στο γεγονός ότι διάφορα Στοιχεία σε ένα έγγραφο περιγράφουν διαφορετική πληροφορία, η οποία μπορεί να χρησιμοποιηθεί αυτούσια από κάποιον που θα τη ζητήσει. Αυτό σημαίνει ότι κάποιος κόμβος στο δίκτυο ενδέχεται να αναζητήσει ένα υποσύνολο των στοιχείων ενός εγγράφου, που να περιλαμβάνουν την πληροφορία που ζητά. Για το λόγο αυτό είναι ιδιαίτερα χρήσιμο να αποθηκεύονται, εκτός από το αυτούσιο XML έγγραφο, και υποδέντρα του εγγράφου τα οποία περικλείουν κάποια αυτοδύναμη πληροφορία. Ωστόσο, τα υποδέντρα δεν δημιουργούνται στο σύνολό τους κατά την αρχική είσοδο του εγγράφου στο δίκτυο. Αυτό θα προσέθετε μεγάλο φόρτο στο νέο κόμβο που θα έμπαινε στο δίκτυο. Αυτό γιατί θα έπρεπε να κωδικοποιήσει το έγγραφο αυτούσιο, δημιουργώντας τα τμήματα ελέγχου για αυτό και στη συνέχεια να επιφορτιστεί με τη δημιουργία αλλά και κωδικοποίηση όλων των υποδέντρων που θα μπορούσαν να εξαχθούν από αυτό. Τέλος θα έπρεπε να στείλει ένα μεγάλο αριθμό από μηνύματα στους γείτονές του, ώστε να

αποθηκευτούν όλα τα παραγόμενα τμήματα ελέγχου σε διάφορους κόμβους του δικτύου. Αντιθέτως, η δημιουργία και κωδικοποίηση ενός υποδέντρου πραγματοποιείται έπειτα από αναζήτηση της πληροφορίας που κατέχει το συγκεκριμένο τμήμα από κάποιο κόμβο. Δηλαδή, αν κάποιος κόμβος ζητήσει ένα δεδομένο που περιγράφεται από ένα υποσύνολο του εγγράφου, τότε ενεργοποιείται η διαδικασία δημιουργίας και κωδικοποίησης του εν λόγω υποδέντρου. Η διαδικασία αυτή περιγράφεται αναλυτικότερα στην επόμενη ενότητα. Αυτό είναι θεμιτό διότι ένα τμήμα μιας πληροφορίας κάποιου εγγράφου, η οποία έχει μεγάλη ζήτηση, αξίζει να αποθηκεύεται αυτόνομο στο δίκτυο υπό τη μορφή ενός υποδέντρου. Με τον τρόπο αυτό διευκολύνεται η γρήγορη και επιτυχής ανάκτησή του σε μια νέα αναζήτηση. Πληροφορίες οι οποίες δε ζητούνται, δεν χρειάζεται να κωδικοποιούνται εκ νέου ως υποδέντρα, με δεδομένο ότι είναι διαθέσιμες αφού το αρχικό έγγραφο βρίσκεται στο δίκτυο.

Επιλέγοντας να εφαρμόσουμε τη μέθοδο των LT Κωδικών σε κάποιο XML τμήμα του δέντρου ξεχωριστά, πετυχαίνουμε μεγάλη διαθεσιμότητα του XML εγγράφου στο σύνολό του αλλά και μεγαλύτερη ευελιξία στην ανάκτηση πληροφορίας. Πράγματι, χωρίζοντας ένα XML τμήμα σε m τμήματα δεδομένων και κωδικοποιώντας τα σε νέα τμήματα ελέγχου, πετυχαίνουμε μεγάλη διαθεσιμότητα του τμήματος. Το δίκτυο, συνεπώς, γίνεται πιο εύρωστο και πιο ανθεκτικό στις διάφορες αποχωρήσεις κόμβων, ενώ το XML τμήμα μπορεί να αναζητηθεί και να ανακτηθεί με πολλούς τρόπους, επιλέγοντας διαφορετικά τμήματα κάθε φορά. Επίσης, κάθε XML τμήμα περιέχει ένα περιορισμένο όγκο πληροφορίας που περιγράφει. Τα τμήματα δεδομένων στα οποία χωρίζεται, συνεπώς, θα έχουν και σχετικά μικρό μέγεθος. Η κωδικοποίησή τους, λοιπόν, γίνεται πολύ πιο γρήγορα καθώς απαιτούνται λιγότερες πράξεις XOR και επίσης η πιθανότητα ύπαρξης σφάλματος κατά την κωδικοποίηση περιορίζεται σημαντικά. Τα τμήματα που αποθηκεύονται στους κόμβους του δικτύου είναι μικρότερα, κάνοντας πιο εύκολη και γρήγορη την ταυτόχρονη απόκτησή τους από κάποιο κόμβο που τα ζητήσει.

Ένα μειονέκτημα της συγκεκριμένης εφαρμογής της Κωδικοποίησης Απαλοιφής σε XML τμήματα είναι το πλήθος των μηνυμάτων που απαιτούνται για την απόκτηση επαρκών τμημάτων, ώστε να είναι εφικτή η αποκωδικοποίηση. Είναι λογικό, τα

βήματα (hops) που απαιτούνται για τη λήψη των τμημάτων που συνθέτουν το αρχείο να είναι περισσότερα από αυτά που θα απαιτούνταν αν, για παράδειγμα, αποθηκευόταν αυτούσιο το XML τμήμα σε ένα κόμβο (μέθοδος Αντιγραφής – Replication). Προφανώς, αυτό είναι γενικό φαινόμενο της Κωδικοποίησης Απαλοιφής και δεν επικεντρώνεται ειδικά στα XML έγγραφα. Περισσότεροι κόμβοι πρέπει να προσπελαστούν, ώστε να συμπληρωθεί το απαιτούμενο πλήθος τμημάτων που μπορούν να μας δώσουν το αρχείο και όχι ένας μόνο. Αυτό το κόστος στο εύρος του δικτύου που καταναλώνεται για την αποστολή πολλαπλών μηνυμάτων μετριάζεται από το γεγονός ότι μπορούμε να δημιουργούμε αντίγραφα των τμημάτων ελέγχου που δίνουν δημοφιλή XML υπο-δέντρα. Τα αντίγραφα αυτά παρέχουν μεγαλύτερη διαθεσιμότητα και διευκόλυνση στη γρήγορη ανάκτηση των τμημάτων αυτών σε μια νέα αναζήτηση, χωρίς μεγάλο κόστος. Τη διαδικασία δημιουργίας αντιγράφων περιγράφουμε στην ενότητα 4.5.2.

Στο προηγούμενο κεφάλαιο, στην ενότητα 3.3, αναλύσαμε την αρχιτεκτονική του δικτύου στην οποία εφαρμόστηκε η μέθοδος των LT Κωδικών. Στη συγκεκριμένη αρχιτεκτονική, αφού γίνει η κωδικοποίηση ενός αρχείου προκύπτουν κάποια τμήματα ελέγχου. Τα τμήματα αυτά διατηρούν τα ακόλουθα πεδία (υπό τη μορφή πλειάδων):

File-Id	Unit-Id	Τιμή

Σχήμα 4.7. Πεδία των πλειάδων στο δίκτυο Ομότιμων

Για την υποστήριξη των XML εγγράφων, επιπλέον πληροφορία απαιτείται να βρίσκεται αποθηκευμένη σε κάθε πλειάδα. Η πληροφορία αυτή περιλαμβάνει το pathID κάθε XML τμήματος του δέντρου, καθώς και το όνομα του στοιχείου που περιγράφεται από αυτό. Όταν ένα XML δέντρο εισάγεται στο δίκτυο, το όνομα του στοιχείου που το περιγράφει είναι το αρχικό στοιχείο (root element) το εγγράφου. Όταν δημιουργείται ένα υποδέντρο, το όνομα αφορά το στοιχείο που αποτελεί το πατρικό στοιχείο όλων των άλλων στοιχείων του υποδέντρου. Σε κάθε τμήμα ελέγχου που συνθέτει ένα συγκεκριμένο XML τμήμα, αναθέτουμε τρία κλειδιά. Είδαμε στην ενότητα 3.3 του προηγούμενου κεφαλαίου ότι σε κάθε τμήμα που προκύπτει κατά την κωδικοποίηση αναθέτουμε ένα μοναδικό κλειδί, το file-Id, το οποίο ισούται με το

όνομα του αρχείου στο οποίο ανήκει, καθώς και το κλειδί unit-Id, το οποίο ξεχωρίζει τα τμήματα μεταξύ τους. Στα τμήματα, ωστόσο, που προκύπτουν από την κωδικοποίηση ενός XML τμήματος (αρχικού εγγράφου ή υποδέντρου), εκτός από τα δύο αυτά κλειδιά που ανατίθενται, αναθέτουμε και ένα τρίτο κλειδί, το pathID του (υπο)δέντρου που περιγράφει το συγκεκριμένο XML τμήμα (ενότητα 4.3). Τα τρία αυτά κλειδιά συσχετίζονται με κάθε τμήμα που προκύπτει από την Κωδικοποίηση Απαλοιφής και χρησιμοποιούνται για να εξυπηρετήσουν διαφόρων ειδών επερωτήσεις όπως θα δούμε στη συνέχεια του κεφαλαίου. Η μορφή, λοιπόν, των τμημάτων που αποθηκεύονται στο δίκτυο και αναφέρονται σε κάποιο XML τμήμα είναι η εξής:

<{fileID, pathID, unitID}, όνομα, τιμή τμήματος >

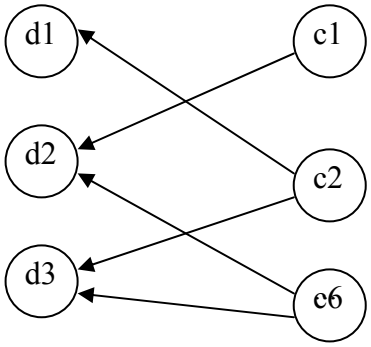
Σχήμα 4.8. Μορφή τμημάτων ελέγχου XML υποδέντρου

Όπως έχουμε πει, το fileID είναι μοναδικό και χρησιμοποιείται για να προσδιορίσουμε σε ποιο έγγραφο ανήκει το τμήμα ελέγχου. Το pathID χρησιμοποιείται για να περιγράψουμε το υπο-δέντρο μέσα στο έγγραφο. Στο σημείο αυτό είναι σημαντικό να αναφέρουμε ότι κάποιο XML έγγραφο μπορεί να περιέχει πολλά στοιχεία με το ίδιο όνομα. Για παράδειγμα, ένα έγγραφο bookstore.xml μπορεί να περιέχει πολλά στοιχεία με όνομα book, τα οποία περιγράφουν βιβλία. Προφανώς τα στοιχεία αυτά έχουν το ίδιο pathID και όνομα. Το σύνολο των στοιχείων αυτών αντιμετωπίζεται ως μια ενιαία οντότητα, ως ένα ενιαίο υποδέντρο, καθώς περιγράφουν ίδιο είδος πληροφορίας, δηλαδή βιβλία. Συνεπώς, κωδικοποιούνται μαζί και παράγουν τμήματα ελέγχου που αποκωδικοποιούν το σύνολο των Στοιχείων αυτών του εγγράφου.

Το unitID χρησιμοποιείται για να ξεχωρίσουν μεταξύ τους τα τμήματα ελέγχου που κωδικοποιούν το ίδιο δεδομένο. Παίρνει τιμές από ένα έως έξι (όσα και τα τμήματα ελέγχου που δημιουργούνται).

Με βάση την παραπάνω ανάλυση παραθέτουμε ένα παράδειγμα αποθήκευσης ενός XML τμήματος το οποίο βασίζεται στο δέντρο του Σχήματος 4.1 της ενότητας 4.3. Έστω ότι θέλουμε να αποθηκεύσουμε το XML τμήμα το οποίο περιγράφει τον κόμβο <firstname> του δέντρου που περιγράφει το XML έγγραφο “song.xml”. Θεωρούμε ότι

το τμήμα αυτό χωρίζεται σε 3 ίσα τμήματα δεδομένων, τα d1, d2 και d3, από τον κόμβο που εισάγει το XML τμήμα στο δίκτυο. Τα τμήματα αυτά κωδικοποιούνται σε έξι τμήματα ελέγχου c1, c2, …, c6. Έστω ότι ο αραιός γράφος της κωδικοποίησης είναι ο εξής:



Σχήμα 4.9. Αραιός γράφος Κωδικοποίησης του XML τμήματος <firstname>

Οι πλειάδες που περιγράφουν κάθε τμήμα ελέγχου είπαμε ότι περιέχουν κάποια κλειδιά και κάποιες επιπλέον πληροφορίες (όνομα, τιμή). Η μορφή τους είναι η ακόλουθη:

fileID	pathID	Unit-Id	Name	Value
song.xml	/song/artist/firstname	01	Firstname	
song.xml	/song/artist/firstname	02	Firstname	
song.xml	/song/artist/firstname	03	Firstname	
song.xml	/song/artist/firstname	04	Firstname	
song.xml	/song/artist/firstname	05	Firstname	
song.xml	/song/artist/firstname	06	Firstname	

Σχήμα 4.10. Πλειάδες για το υποδέντρο /song/artist/firstname του δέντρου του Σχήματος 4.1.

4.5 Αναζήτηση XML Τμημάτων Στο Δίκτυο

4.5.1 Αρχική Εισαγωγή XML Εγγράφου Στο Δίκτυο

Όταν ένας κόμβος εισέρχεται στο δίκτυο θεωρούμε ότι το XML έγγραφο που διαθέτει δεν υφίσταται διαδικασία δημιουργίας υποδέντρων, βρίσκεται δηλαδή αποθηκευμένο αυτούσιο σε αυτόν. Το έγγραφο αυτό στη συνέχεια κωδικοποιείται μέσω της κωδικοποίησης των LT Κωδικών και παράγονται τα τμήματα ελέγχου. Στη συνέχεια, τα τμήματα αυτά αποθηκεύονται σε διάφορους κόμβους του δικτύου για λόγους αύξησης της διαθεσιμότητας και βιωσιμότητας του εγγράφου στο δίκτυο. Η διαδικασία αυτή γίνεται με τον τρόπο που περιγράφηκε στο προηγούμενο κεφάλαιο (Ενότητα 3.3.1). Η επιπρόσθετη πληροφορία που εισάγεται στις πλειάδες είναι το pathID το οποίο για το αρχικό έγγραφο έχει την τιμή "/Αρχικό Στοιχείο" και το όνομα, το οποίο παίρνει την τιμή του ονόματος του αρχικού στοιχείου (root element) του εγγράφου. Εάν κάποιος κόμβος αναζητήσει ολόκληρο το έγγραφο (με βάση το fileID, pathID) τότε ακολουθείται η διαδικασία που περιγράφηκε στο προηγούμενο κεφάλαιο (Ενότητα 3.3.2.).

Για τη διευκόλυνση στην αναζήτηση ενός υποσυνόλου της πληροφορίας που υπάρχει σε ένα XML έγγραφο, θεωρούμε ότι το υποδέντρο το οποίο περιέχει την συγκεκριμένη πληροφορία υπάρχει αυτούσιο στο δίκτυο. Το υποδέντρο αυτό υφίσταται την κωδικοποίηση σε νέα τμήματα ελέγχου με βάση τον αλγόριθμο 1. Στη συνέχεια αποθηκεύεται στους κόμβους με βάση τον αλγόριθμο 3. Το path-id που συσχετίζεται με κάθε τμήμα ελέγχου, ισούται με το μονοπάτι που χαρακτηρίζει τον πατέρα του συγκεκριμένου υποδέντρου. Επίσης, το name παίρνει την τιμή του γονικού αυτού στοιχείου του υποδέντρου.

Αν κάποιος κόμβος ζητήσει πληροφορία που υπάρχει σε ένα υποδέντρο ενός εγγράφου, η διαδικασία που ακολουθείται είναι αυτή που περιγράφει ο αλγόριθμος 4. Η διαφορά εδώ είναι ότι η αναζήτηση μπορεί να γίνει με βάση το path-id που χαρακτηρίζει το υποδέντρο. Επιπλέον, αν δεν γνωρίζει το μονοπάτι ο κόμβος που κάνει την αναζήτηση, μπορεί να ψάξει με βάση το όνομα (name) του στοιχείου που τον ενδιαφέρει. Σε κάθε περίπτωση, ο αλγόριθμος 4 χρησιμοποιείται, απλώς

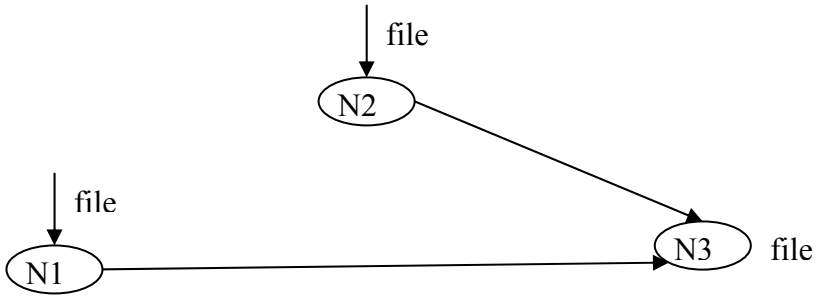
χρησιμοποιώντας το path-id ή το name ως το κλειδί του μηνύματος αναζήτησης, αντίστοιχα.

4.5.2 Δημιουργία Αντιγράφων XML Τμημάτων

Είναι κατανοητό ότι για την καλύτερη απόδοση του δικτύου, ένας σημαντικός παράγοντας είναι και ο ρυθμός με τον οποίο ζητούνται κάποια τμήματα από τους διάφορους κόμβους. Είναι φυσιολογικό κάποια τμήματα να ζητούνται περισσότερες φορές από κάποια άλλα. Κάποια τμήματα, δηλαδή, είναι πιο δημοφιλή προς αναζήτηση από κάποια άλλα για τα οποία οι αναζητήσεις είναι σπανιότερες. Η παράμετρος αυτή είναι αρκετά σημαντική και οφείλει το δίκτυο να τη λάβει υπ' όψιν κατά την αποθήκευση των XML τμημάτων από τους διάφορους κόμβους. Η Κωδικοποίηση Απαλοιφής που εφαρμόζεται στα διάφορα τμήματα είναι μία επικερδής μέθοδος για την αύξηση της διαθεσιμότητάς τους, ωστόσο δεν παρέχει κάποια ουσιαστικά αποτελέσματα στον τομέα της δημοφιλίας ορισμένων τμημάτων έναντι άλλων. Είναι προφανές ότι τα πιο δημοφιλή τμήματα θα πρέπει να έχουν μεγαλύτερη διαθεσιμότητα από τα λιγότερο δημοφιλή, καθώς αυτά τα αναζητούν περισσότεροι κόμβοι στο δίκτυο, άρα πρέπει να υπάρχουν επαρκή αντίγραφα ώστε η ανάκτησή τους να γίνεται και πιο γρήγορα αλλά και με μεγάλη πιθανότητα επιτυχίας. Η ύπαρξη ίδιου πλήθους αντιγράφων δημοφιλών και μη δημοφιλών τμημάτων στο δίκτυο είναι, γενικότερα, ανεπιθύμητη. Αν το πλήθος των αντιγράφων είναι μικρό, τότε τα δημοφιλή τμήματα υστερούν στην αναζήτηση, καθώς η πιθανότητα απόκτησής τους είναι σχετικά μικρή και επιπλέον, οι λίγοι κόμβοι που κρατούν κάποιο δημοφιλές τμήμα θα έχουν αυξημένο φόρτο εργασίας, με δυσάρεστες, πολλές φορές, συνέπειες (π.χ. αποτυχία εξυπηρέτησης επερωτήσεων). Εάν, πάλι, το πλήθος των αντιγράφων είναι αρκετά μεγάλο, ώστε να μπορούν να αντιμετωπιστούν οι παραπάνω ανεπιθύμητες καταστάσεις, τότε προκύπτει ότι τα μη δημοφιλή τμήματα καταλαμβάνουν περιττό χώρο στο δίκτυο, καθώς θα ζητούνται ελάχιστα άλλα θα υπάρχουν σε πολλά αντίγραφα.

Για την αντιμετώπιση των ακραίων αυτών καταστάσεων, επιλέχθηκε το πλήθος των αντιγράφων για κάθε XML τμήμα που βρίσκονται αποθηκευμένα στο δίκτυο να είναι

ανάλογο με το ποσοστό των επερωτήσεων για το συγκεκριμένο τμήμα. Είναι εύλογο ότι όσο περισσότερες αναζητήσεις γίνονται για κάποιο τμήμα από τους κόμβους του δικτύου, τόσο πιο δημοφιλές γίνεται αυτό το τμήμα. Η πιθανότητα, δηλαδή, να ζητηθεί ξανά είναι μεγαλύτερη σε σχέση με κάποιο τμήμα που έχει μικρό ρυθμό αναζητήσεων. Συνεπώς, επιθυμούμε τα δημοφιλή τμήματα να υπάρχουν σε περισσότερα αντίγραφα στο δίκτυο, ώστε να είναι πιο εύκολη και γρήγορη η ανάκτησή τους από τους κόμβους. Στο δίκτυο που υλοποιήθηκε, η μέθοδος που επιλέχθηκε όσον αφορά το πλήθος των αντιγράφων που κρατούνται για κάθε XML τμήμα είναι η αντιγραφή κατόχου (owner replication). Με βάση τη μέθοδο της αντιγραφής κατόχου, το πλήθος των αντιγράφων που δημιουργούνται για κάθε τμήμα, εξαρτάται από το ποσοστό των επερωτήσεων για το συγκεκριμένο τμήμα. Πιο συγκεκριμένα, όταν ένας απλός κόμβος λάβει ένα XML τμήμα το οποίο έχει ζητήσει, τότε δημιουργεί και ένα αντίγραφο αυτού του τμήματος, το οποίο και αποθηκεύει στη βάση του. Με τον τρόπο αυτό, τα αντίγραφα που δημιουργούνται και αποθηκεύονται στο δίκτυο είναι ανάλογα με το πλήθος των επερωτήσεων που γίνονται για αυτά (Σχήμα 4.11).



Σχήμα 4.11. Αντιγραφή κατόχου. Οι N1 και N2 εξυπηρετούνται από τον N3 που έχει το ζητούμενο αρχείο (file) και κρατούν από ένα αντίγραφό του

Η απλή αποθήκευση ενός αντιγράφου κάποιου αρχείου από ένα κόμβο λειτουργεί ικανοποιητικά στη μέθοδο της Αντιγραφής (Replication). Ωστόσο, όταν εφαρμόζουμε τη μέθοδο της Κωδικοποίησης Απαλοιφής, η διαδικασία είναι λίγο διαφορετική. Συγκεκριμένα, όταν ένας κόμβος λάβει το απαραίτητο πλήθος τμημάτων ελέγχου για κάποιο τμήμα XML εγγράφου, τότε επιλέγει να δημιουργήσει αντίγραφά τους. Ωστόσο, δεν κρατάει όλα τα τμήματα ελέγχου στη λίστα δεδομένων του, αλλά επιλέγει να τα προωθήσει σε άλλους κόμβους. Ο ίδιος αποθηκεύει το τμήμα με unitID =1 και

στη συνέχεια στέλνει τα υπόλοιπα για αποθήκευση στους γείτονές του. Το μήνυμα που στέλνει για την αποθήκευση των τμημάτων αυτών διαφέρει λίγο από αυτό που περιγράφηκε στην Ενότητα 3.3.1, υπό την έννοια ότι δεν υπάρχει TTL στο συγκεκριμένο μήνυμα. Ο κόμβος επιλέγει τους γείτονές του για να αποθηκεύσουν τα τμήματα ελέγχου. Α κάποιος από τους γείτονες περιέχει ήδη το τμήμα ελέγχου που του αποστέλλεται για αποθήκευση, απλώς το προωθεί σε κάποιον από τους δικούς του γείτονες. Η λειτουργία αυτή επαναλαμβάνεται έως ότου βρεθεί κόμβος που να μην έχει ήδη το τμήμα ελέγχου στη λίστα του.

Με τον τρόπο δημιουργίας αντιγράφων με βάση την αντιγραφή κατόχου, προκύπτει ότι τα δημοφιλή τμήματα καταλαμβάνουν περισσότερο χώρο στο δίκτυο. Τα δημοφιλή τμήματα είναι πιο εύκολο να ανακτηθούν με λιγότερο αριθμό μηνυμάτων (hops). Βέβαια, τα μη δημοφιλή τμήματα είναι πιο δύσκολο να ανακτηθούν, καθώς το πλήθος αντιγράφων τους είναι σαφώς μικρότερο από κάποιο δημοφιλές τμήμα. Ωστόσο, αυτό είναι ένα τίμημα που είναι ανεκτό στο σύστημά μας, αφού η απόδοση του δικτύου είναι λογικό να κρίνεται κυρίως από την μεγάλη διαθεσιμότητα των δημοφιλών τμημάτων. Τα σπάνια τμήματα, βέβαια, έχουν αρκετές φορές μεγαλύτερη διαθεσιμότητα σε σύγκριση με την εφαρμογή της μεθόδου της αντιγραφής σε αυτά. Συνεπώς, η διαθεσιμότητά τους είναι σε ένα βαθμό ικανοποιητική και αυξάνεται σημαντικά αν γίνει τουλάχιστον μία επιπλέον αναζήτηση για κάποιο από αυτά.

4.5.3 Τύποι Επερωτήσεων

Στην παρούσα ενότητα αναλύουμε τους τύπους των επερωτήσεων που κάνουν οι κόμβοι του δικτύου για την ανάκτηση XML τμημάτων που βρίσκονται αποθηκευμένα στο δίκτυο μέσω της Κωδικοποίησης Απαλοιφής. Εκτός από τη μορφή των επερωτήσεων που μπορούν να γίνουν στο δίκτυο, παρατίθενται και διάφορα παραδείγματα για την καλύτερη κατανόηση της διαδικασίας που ακολουθείται για την επιτυχή ολοκλήρωση κάθε μορφής αναζήτησης.

Βασικές παράμετροι με τις οποίες γίνεται μια αναζήτηση στο δίκτυο που υλοποιήσαμε είναι α) το όνομα του XML εγγράφου (fileID), τμήμα του οποίου αναζητείται και β) το

μονοπάτι που χαρακτηρίζει το υποδέντρο του XML δέντρου με το συγκεκριμένο όνομα, το οποίο αντιστοιχεί στο τμήμα Στοιχείων του εγγράφου που θέλει κάποιος χρήστης να λάβει (pathID). Επίσης, ένας κόμβος μπορεί να αναζητήσει ένα τμήμα, με βάση το όνομα του Στοιχείου που επιθυμεί, χωρίς να χρειάζεται να γνωρίζει το μονοπάτι του. Δηλαδή κάποιος κόμβος του δικτύου μπορεί να κάνει μια αναζήτηση για το υποδέντρο που χαρακτηρίζει τον κόμβο <lastname> στο δέντρο του Σχήματος 4.1 με βάση το ίδιο το όνομα του στοιχείου (lastname) και όχι με βάση το μονοπάτι που το χαρακτηρίζει (/song/artist/lastname).

Η μορφή των επερωτήσεων που μπορούν να γίνουν για την ανάκτηση ενός XML τμήματος φαίνεται στο Σχήμα 4.12.

FILE: όνομα XML εγγράφου (fileID) RETURN: op_1 el_1 op_2 el_2…op_n el_n (pathID)

FILE: όνομα XML εγγράφου RETURN: NAME=όνομα αρχικού κόμβου υποδέντρου

Σχήμα 4.12. *Πάνω*: Μορφή επερώτησης με βάση το όνομα του αρχείου και το μονοπάτι που χαρακτηρίζει το τμήμα. *Κάτω*: Μορφή επερώτησης με βάση το όνομα του αρχικού κόμβου που χαρακτηρίζει το τμήμα

Στο Σχήμα 4.12 βλέπουμε τους τύπους επερωτήσεων με βάση το μονοπάτι ή με βάση το όνομα ενός υποδέντρου για κάποιο XML έγγραφο. Στο πάνω μέρος του Σχήματος 4.12 βλέπουμε τη μορφή επερώτησης που γίνεται στο δίκτυο με βάση το όνομα του ζητούμενου εγγράφου και το μονοπάτι του υποδέντρου που ζητείται. Στην έκφραση "op_1 el_1 op_2 el_2…op_n el_n", που υποδηλώνει το μονοπάτι του υποδέντρου μας ενδιαφέρει, το "op_k" μπορεί να πάρει τιμές "/" ή το "//", αναφερόμενο σε σχέση πατέρα- παιδιού ή προγόνου- απογόνου ανάμεσα στα στοιχεία τα οποία συνδέει. Το "el_k" μπορεί να πάρει τιμή το όνομα ενός κόμβου (στοιχείου) στο XML δέντρο (έγγραφο). Στο κάτω μέρος του Σχήματος 4.12 βλέπουμε τη μορφή αναζήτησης σε ένα

XML έγγραφο για κάποιο τμήμα με βάση το όνομα του στοιχείου που μας ενδιαφέρει. Το όνομα αυτό αναφέρεται σε κάποιο υποδέντρο του δέντρου που χαρακτηρίζει το XML έγγραφο, το όνομα του οποίου δίνεται επίσης ως παράμετρος στην αναζήτηση. Εάν το υποδέντρο είναι, στην ουσία, ένας κόμβος – φύλλο, τότε η ανάκτηση απλώς του στοιχείου που περιγράφεται από τον κόμβο αυτό αρκεί για να ικανοποιήσει την αναζήτηση. Αν όμως το όνομα του υποδέντρου που δίνεται ως παράμετρος, ορίζει ένα κόμβο ο οποίος έχει και παιδιά, τότε μόνο η ανάκτηση ολόκληρου του υποδέντρου (κόμβου που αναζητήθηκε και τα παιδιά του) ικανοποιεί την αναζήτηση.

Στη συνέχεια της ενότητας παρατίθενται κάποιες ενδεικτικές επερωτήσεις που μπορούν να γίνουν στο δίκτυο που υλοποιήθηκε με τη χρήση της Κωδικοποίησης Απαλοιφής (LT κώδικες). Οι επερωτήσεις αφορούν σε XML τμήματα του εγγράφου "song.xml" της ενότητας 4.2. Το XML δέντρο που περιγράφει το έγγραφο αυτό φαίνεται στο Σχήμα 4.1. Στα παραδείγματα αυτά παρουσιάζονται οι ενέργειες που πρέπει να κάνει κάθε υπερ-κόμβος που δέχεται μια επερώτηση, καθώς και τα διάφορα δεδομένα που πρέπει να ανακτηθούν μέχρι να ολοκληρωθεί η επερώτηση με επιτυχία.

Παράδειγμα 4.1

Έστω ότι ένας κόμβος N1 κάνει την ακόλουθη επερώτηση:

FILE: song.xml RETURN: /song/artist/

Σχήμα 4.13 Αναζήτηση με βάση το path-id ενός υποδέντρου

Τα βήματα που ακολουθούνται για την ικανοποίηση της αναζήτησης είναι τα εξής:

1. Ο N1 δημιουργεί το μήνυμα αναζήτησης δημιουργώντας τις λίστες των unitIDs και των κόμβων στους οποίους φτάνει το μήνυμα (Ενότητα 3.3.2). Επίσης θέτει και το TTL του μηνύματος και το στέλνει τυχαία σε κάποιον από τους γείτονές του.

2. Ο γείτονας που θα λάβει το μήνυμα ελέγχει τη λίστα δεδομένων του αν υπάρχει πλειάδα που να πληρεί τις κατάλληλες συνθήκες (fileID, pathID, unitID). Αν έχει μια

τέτοια πλειάδα τότε στέλνει μήνυμα επιτυχίας στον N1 και την στέλνει σε αυτόν. Αν δεν περιέχει τέτοια πλειάδα, τότε μειώνει το TTL κατά ένα και το προωθεί σε κάποιο γείτονά του.

3. Αν κάποιος κόμβος διαθέτει ένα από τα ζητούμενα τμήματα ελέγχου, τότε στέλνει το εν λόγω τμήμα στον N1 και μειώνει το πεδίο count (αλγόριθμος 4). Αν γίνει μηδέν, τότε τερματίζεται επιτυχώς η αναζήτηση. Σε διαφορετική περίπτωση, απλώς προωθεί το τροποποιημένο μήνυμα σε ένα γείτονά του.

4. Αν ο N1 συμπληρώσει το απαιτούμενο πλήθος πλειάδων, τότε ξεκινά την αποκωδικοποίηση των τμημάτων ελέγχου με βάση τον αλγόριθμο Αποκωδικοποίησης των LT κωδικών. Αν δεν επιτύχει η ανάκτηση, τότε στέλνει ξανά την αναζήτηση για ένα ακόμη τμήμα ελέγχου, διαφορετικό από αυτά που έχει ήδη λάβει.

Παράδειγμα 4.2

Έστω ότι ένας κόμβος N1 κάνει την ακόλουθη επερώτηση (Σχήμα 4.14):

```
FILE: song.xml
RETURN: /song//lastname
```

Σχήμα 4.14 Αναζήτηση με βάση μονοπάτι που υποδηλώνει σχέση προγόνου-απόγονου

Η διαδικασία που ακολουθείται για την εξυπηρέτηση της συγκεκριμένης επερώτησης (Σχήμα 4.14) είναι η ίδια που περιγράφηκε στο Παράδειγμα 4.1. Η διαφορά εδώ έγκειται στο γεγονός ότι οι κόμβοι στους οποίους φτάνει το μήνυμα ελέγχουν τη λίστα δεδομένων τους για πλειάδες των οποίων το fileID ισούται με το “song.xml”, ενώ το pathID θα πρέπει να περιέχει σε κάποιο σημείο το όνομα “song” και να καταλήγει με το όνομα “lastname”. Στο δέντρο του Σχήματος 4.1, είναι προφανές ότι το υποδέντρο που θα ανακτηθεί είναι αυτό με pathID= /song/artist/lastname. Αυτό είναι και το pathID με βάση το οποίο θα γίνει η αναζήτηση των τμημάτων ελέγχου που θα συνθέσουν το ζητούμενο υποδέντρο.

Παράδειγμα 4.3

Έστω ότι ένας κόμβος N1 κάνει την ακόλουθη επερώτηση (Σχήμα 4.5):

```
FILE: song.xml
RETURN: //firstname
```

Σχήμα 4.15 Αναζήτηση με βάση ένα στοιχείο-απόγονο ενός υποδέντρου

Η επερώτηση του Σχήματος 4.15 εξυπηρετείται με τα ίδια βήματα που ακολουθήθηκαν και στις προηγούμενες επερωτήσεις. Η βάση για τη σωστή εξυπηρέτηση της αναζήτησης είναι η κατάλληλη επιλογή και ανάκτηση του σωστού πλήθους πλειάδων για το ζητούμενο υποδέντρο. Στο παράδειγμα αυτό, κάθε κόμβος ελέγχει αν στη λίστα δεδομένων του υπάρχουν πλειάδες με fileID=”song.xml” και το pathID τους να τελειώνει με “firstname”. Αυτό ισχύει διότι η ύπαρξη του “//” πριν το firstname υποδηλώνει ότι το στοιχείο αυτό είναι απόγονος κάποιου άλλου στοιχείου στο έγγραφο. Στο Σχήμα 4.1 το υποδέντρο που ικανοποιεί την επερώτηση αυτή είναι αυτό με pathID=/song/artist/firstname.

Παράδειγμα 4.4

Έστω ότι ένας κόμβος N1 κάνει την ακόλουθη επερώτηση (Σχήμα 4.16)

```
FILE: song.xml
RETURN: NAME=title
```

Σχήμα 4.16 Αναζήτηση με βάση το όνομα του πατέρα του υποδέντρου

Στην επερώτηση αυτή, η παράμετρος με την οποία γίνεται η αναζήτηση είναι , εκτός από το όνομα του XML εγγράφου, το όνομα του στοιχείου που μας ενδιαφέρει. Το υποδέντρο, λοιπόν, που αναζητεί ο N1ειναι αυτό που έχει ως αρχικό κόμβο τον <title>. Στην περίπτωση του Σχήματος 4.1 το υποδέντρο είναι ο κόμβος-φύλλο με pathID=/song/title. Το μήνυμα που θα στείλει ο N1 στον γείτονά του, πλέον, δεν έχει

ως παράμετρο το pathID, αλλά το πεδίο name στις πλειάδες των τμημάτων ελέγχου. Συνεπώς κάθε κόμβος στον οποίο φτάνει το συγκεκριμένο μήνυμα, ελέγχει τη λίστα δεδομένων του για πλειάδες οι οποίες έχουν fileID=”song.xml” και name=”title”. Κατά τα λοιπά, η διαδικασία εύρεσης των τμημάτων ελέγχου είναι η ίδια με αυτή που περιγράφηκε στο παράδειγμα 4.1. Στην περίπτως που το έγγραφο song.xml περιέχει πολλά στοιχεία με όνομα title, οι πλειάδες που πληρούν τις προϋποθέσεις κωδικοποιούν όλα τα Στοιχεία με όνομα title. Επομένως, ο N1 θα λάβει το σύνολο της πληροφορίας που περιέχουν.

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

5.1 Εισαγωγή

Στα προηγούμενα κεφάλαια περιγράψαμε τη μέθοδο Κωδικοποίησης Απαλοιφής, ως μία νέα μέθοδο για την αποθήκευση αρχείων σε ένα δίκτυο. Συγκεκριμένα, ασχοληθήκαμε με τη μέθοδο των LT κωδικών απαλοιφής, παρουσιάζοντας τα χαρακτηριστικά της (αλγόριθμος κωδικοποίησης και αποκωδικοποίησης), καθώς και το κόστος των πράξεων για την πραγματοποίησή τους. Επιπλέον, κάναμε μια εισαγωγή στη γλώσσα σήμανσης XML και στα XML έγγραφα, περιγράφοντας τα συστατικά ενός τέτοιου εγγράφου και πώς μπορεί να υλοποιηθεί ως ένα δέντρο. Δείξαμε τη δυνατότητα κατακερματισμού ενός XML δέντρου σε μικρότερα υπο-δέντρα και ποιες ιδιότητες αυτά διαθέτουν. Κατόπιν εισάγαμε την Κωδικοποίηση Απαλοιφής σε ένα τέτοιοτύπο εγγράφων. Τέλος, δημιουργήσαμε ένα δίκτυο ομότιμων, το οποίο θεωρήθηκε ότι περιλαμβάνει την αποθήκευση και διακίνηση XML δεδομένων. Χρησιμοποιήσαμε τους LT κώδικες απαλοιφής ως το μέσον για τη δημιουργία αντιγράφων των εγγράφων που υπάρχουν σε ένα τέτοιο δίκτυο, με στόχο την αύξηση της διαθεσιμότητάς τους, για τη διασφάλιση μεγαλύτερης ευρωστίας και βιωσιμότητας του δικτύου.

Στο παρόν κεφάλαιο παραθέτουμε ορισμένα πειράματα τα οποία πραγματοποιήθηκαν στη συγκεκριμένη εργασία. Τα πειράματα αυτά έγιναν με στόχο να διαπιστώσουμε τα

οφέλη της Κωδικοποίησης Απαλοιφής και για να δούμε τη συμπεριφορά της σε XML έγγραφα μέσα σε ένα Δίκτυο Ομότιμων. Μια συνήθης μέθοδος που χρησιμοποιείται πολύ συχνά για τη δημιουργία αντιγράφων σε ένα δίκτυο ομότιμων, είναι η μέθοδος της Αντιγραφής (Replication). Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, η μέθοδος αυτή δημιουργεί αυτούσια αντίγραφα των αρχείων, τα οποία αποθηκεύονται στους κόμβους του δικτύου με διάφορους τρόπους. Τα αντίγραφα αυτά αυξάνουν τη διαθεσιμότητα των αρχείων διότι είναι προσπελάσιμα σε ενδεχόμενη αποτυχία κάποιων κόμβων του δικτύου. Ωστόσο, το αποθηκευτικό κόστος ατής της δημιουργίας αντιγράφων είναι αρκετά μεγάλο και αποτελεί τροχοπέδη στην ομαλή λειτουργία του δικτύου. Για να λάβουμε χρήσιμα συμπεράσματα από τη μέθοδο της Κωδικοποίησης Απαλοιφής, πολλά από τα πειράματα πραγματοποιήθηκαν τόσο για τη μέθοδο των LT κωδικών, όσο και για τη μέθοδο της Αντιγραφής. Στόχος είναι να αντιπαραβάλλουμε τα αποτελέσματα που προκύπτουν από τις μεθόδους αυτές, ώστε να συγκρίνουμε και να αξιολογήσουμε τις επιδόσεις τους αλλά και τον αντίκτυπο που δημιουργούν στο Δίκτυο, με ίδιες συνθήκες.

Τα πειράματα που πραγματοποιήθηκαν μπορούν να ομαδοποιηθούν σε δύο κατηγορίες. Κάποια πειράματα έγιναν πάνω στη μέθοδο των LT Κωδικών, ώστε να βγάλουμε κάποια συμπεράσματα για την απόδοση και την πολυπλοκότητά τους. Η επόμενη ομάδα πειραμάτων ανήκουν στην κατηγορία αυτών που πραγματοποιήθηκαν στο Δίκτυο Ομότιμων που σχεδιάστηκε. Συγκεκριμένα, τα πειράματα έγιναν πάνω σε XML έγραφα που αποθηκεύονται στο δίκτυο μέσω Κωδικοποίησης Απαλοιφής (αλλά και κάποια με Αντιγραφή-Replication). Στόχος των πειραμάτων αυτών είναι να δούμε τη συμπεριφορά της μεθόδου σε ένα δυναμικό δίκτυο, όπως ένα δίκτυο ομότιμων, αλλά και τις επιμέρους διαφορές (πλεονεκτήματα – μειονεκτήματα) έναντι της μεθόδου της Αντιγραφής. Στην τελευταία ενότητα του κεφαλαίου ανακεφαλαιώνουμε τα συμπεράσματα που προκύπτουν από τα πειραματικά αποτελέσματα.

Στα πειράματα που πραγματοποιήθηκαν προσομοιώσαμε ένα δίκτυο ομότιμων, το οποίο είναι μη κεντρικοποιημένο και αδόμητο. Δεν έχει, δηλαδή, κεντρική δομή που να καθορίζει τη συμπεριφορά των κόμβων και επιπλέον, οι κόμβοι συνδέονται μεταξύ τους με αυθαίρετο τρόπο, χωρίς συγκεκριμένη δομή. Το δίκτυο που υλοποιήσαμε έχει πλήθος 100 κόμβων και ο μέγιστος αριθμός κάθε κόμβου ορίστηκε να είναι πέντε (5).

Τα έγγραφα που χρησιμοποιήθηκαν είναι XML μορφής και έχουν τάξη μεγέθους O(KBytes). Επιπλέον, θεωρήσαμε ότι δεν υπάρχει συγκεκριμένος αποθηκευτικός χώρος για κάθε κόμβο, αλλά ότι είναι απεριόριστος. Τα χαρακτηριστικά του δικτύου φαίνονται στον πίνακα 5.1.

Πίνακας 5.1. Χαρακτηριστικά δικτύου υλοποίησης

Τοπολογία δικτύου	Αδόμητο, μη κεντρικοποιημένο
Πλήθος κόμβων	100
Μέγιστο πλήθος γειτόνων κάθε κόμβου	5
Λειτουργίες κόμβου	Κωδικοποίηση, αποκωδικοποίηση, αποστολή και λήψη μηνυμάτων στους γείτονες
Τύπος εγγράφων	XML έγγραφα
Μέγεθος εγγράφων	O(Kbytes)

Επιπλέον, σε διάφορα πειράματα χρησιμοποιήσαμε κάποιες παραμέτρους οι οποίες μεταβάλλονται, ώστε να δείξουν τη συμπεριφορά των αλγορίθμων από αυτές τις μεταβολές. Οι βασικές παράμετροι που μεταβάλλονταν για κάποια πειράματα είναι το πλήθος των τμημάτων ελέγχου, στα οποία χωρίζεται ένα έγγραφο, καθώς και το πλήθος των τμημάτων ελέγχου που παράγονται από αυτά. Ακόμη, παράγοντας εισόδου είναι και το TTL των μηνυμάτων αποθήκευσης και αναζήτησης. Τέλος, χρησιμοποιήθηκαν, το πλήθος των κόμβων του δικτύου αλλά και το συνολικό μέγεθος των εγγράφων, ώστε να διαπιστωθούν τα ποσοστά επιτυχίας στην περίπτωση αύξησης του μεγέθους του δικτύου, αλλά και τους χρόνους και την εν γένει συμπεριφορά των LT κωδικών για διαφορετικά μεγέθη των εγγράφων. Οι παράμετροι εισόδου που χρησιμοποιήθηκαν, φαίνονται στον πίνακα 5.2.

Πίνακας 5.2 Παράμετροι εισόδου που μεταβάλλονται κατά τα πειράματα

Πλήθος τμημάτων δεδομένων
Πλήθος τμημάτων ελέγχου
Πλήθος βημάτων αποθήκευσης (TTL αποθήκευσης)
Πλήθος βημάτων αναζήτησης (TTL αναζήτησης)
Μέγεθος δικτύου (Συνολικός αριθμός κόμβων)
Μέγεθος εγγράφων

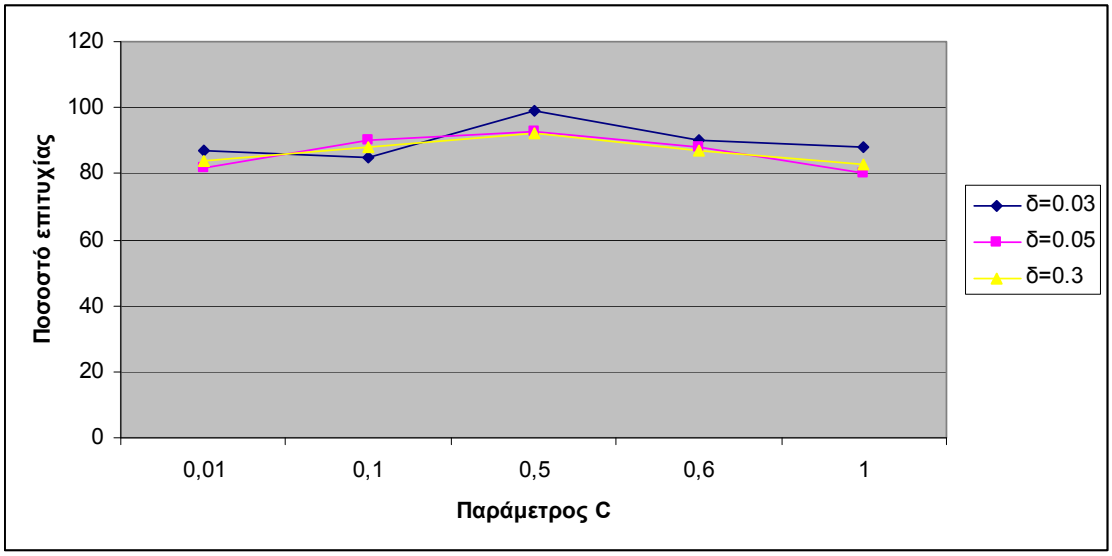
5.2 Μέθοδος LT Κωδικών Απαλοιφής

Στην Ενότητα αυτή υλοποιήσαμε κάποια πειράματα πάνω στους αλγορίθμους κωδικοποίησης και αποκωδικοποίησης των LT κωδικών (Κεφάλαιο 2). Στόχος των πειραμάτων αυτών είναι να δείξουν την απόδοση της μεθόδου όταν εφαρμόζεται σε κάποιο XML έγγραφο. Η έννοια της απόδοσης έγκειται στο γεγονός ότι αν λάβουμε διαφορετικό πλήθος από τμήματα ελέγχου που κωδικοποιούν το έγγραφο, ποιο είναι το ποσοστό της επιτυχημένης αποκωδικοποίησής τους ώστε να λάβουμε το αρχικό έγγραφο. Το έγγραφο που χρησιμοποιήθηκε είναι της τάξης μεγέθους 3KB, κάτι ρεαλιστικό καθώς ως επί το πλείστον τα XML έγγραφα έχουν μέγεθος της τάξης των KB. Το πλήθος των κόμβων που χρησιμοποιήσαμε για την προσομοίωση του δικτύου είναι εκατό (100). Στα πειράματα που ακολουθούν υπολογίσαμε την απόδοση του αλγορίθμου λαμβάνοντας διαφορετικού πλήθους τμημάτων ελέγχου. Επιπλέον, υπολογίστηκε ο χρόνος κωδικοποίησης και αποκωδικοποίησης των LT Κωδικών ανάλογα με το πλήθος αλλά και το μέγεθος των τμημάτων ελέγχου που δημιουργούνται.

Πείραμα 1

Στο πείραμα 1 εξετάζεται η συμπεριφορά της Robust Soliton κατανομής, με βάση τις παραμέτρους c και δ. Συγκεκριμένα, παρατηρούμε το ποσοστό των επιτυχημένων

αποκωδικοποιήσεων (αλγόριθμος 2) αν λάβουμε κωδικοποιήσεις για διάφορες τιμές των παραμέτρων αυτών. Παίρνουμε, λοιπόν, το Σχήμα 5.1.



Σχήμα 5.1. Απόδοση της LT μεθόδου για διαφορετικές τιμές των c και δ

Από το Σχήμα 5.1 παρατηρούμε ότι τα ποσοστά επιτυχίας είναι ιδιαίτερα υψηλά για τις παραμέτρους δ και c της Robust Soliton κατανομής. Είπαμε ότι η συγκεκριμένη κατανομή ορίζεται ως :

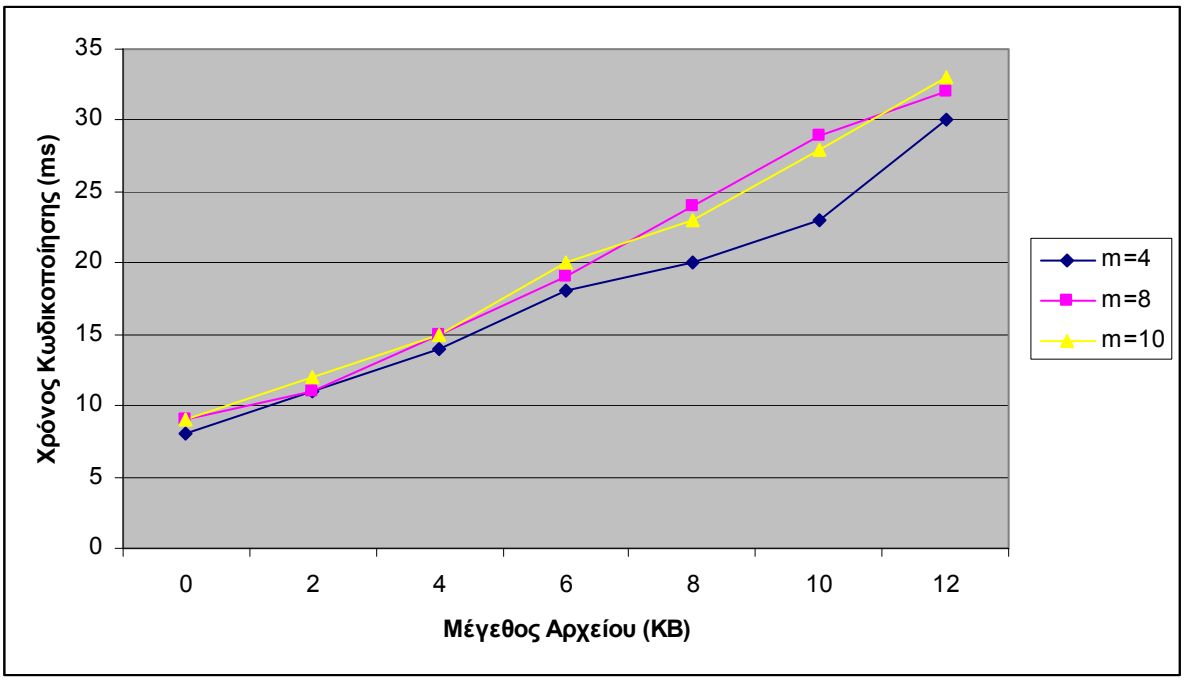
$$R = c * \ln(\frac{k}{\delta}) * \sqrt{k}$$

και υποδηλώνει την πιθανότητα κάποιος κόμβος να έχει βαθμό 1. Υπολογίζοντας το ποσοστό επιτυχίας της αποκωδικοποίησης για διάφορα c και δ προκύπτει ότι για την τιμή c=0.5 έχουμε πολύ καλά αποτελέσματα. Αυτό ισχύει για όλα τα δ που επιλέχθηκαν. Ωστόσο, βέλτιστή απόδοση (99%) παρουσιάζεται για δ=0.03. Συμπεραίνουμε, λοιπόν, ότι οι συγκεκριμένες τιμές λειτουργούν βέλτιστα, καθώς δίνουν τα μέγιστα ποσοστά επιτυχίας στην αποκωδικοποίηση, λαμβάνοντας k+O(k+k/2) τμήματα ελέγχου.

Πείραμα 2

Στο συγκεκριμένο πείραμα υπολογίζουμε το χρόνο που απαιτείται για την κωδικοποίηση ενός εγγράφου εισόδου σε ορισμένο πλήθος τμημάτων ελέγχου.

Συγκεκριμένα, υπολογίζουμε το χρόνο κωδικοποίησης για διάφορα μεγέθη αρχείων εισόδου, αν δημιουργήσουμε ίδιο πλήθος τμημάτων ελέγχου. Πήραμε τα ακόλουθα αποτελέσματα:

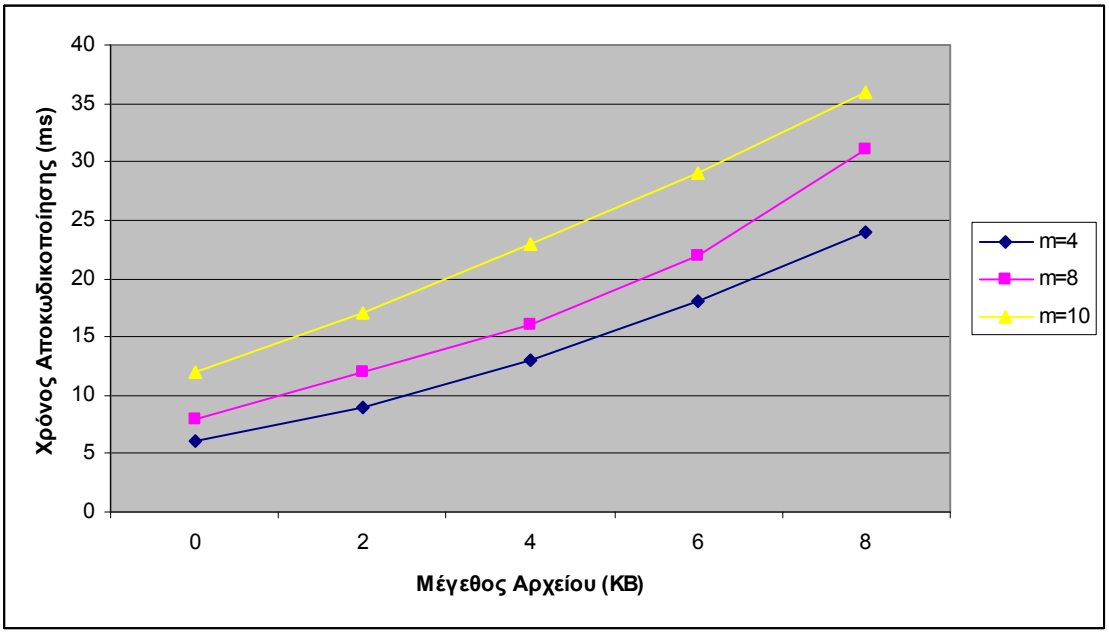


Σχήμα 5.2. Χρόνος Κωδικοποίησης για διαφορετικά μεγέθη αρχείων

Από τα αποτελέσματα που πήραμε παρατηρούμε ότι ο χρόνος κωδικοποίησης είναι ικανοποιητικός ακόμη και για μεγάλα μεγέθη αρχείων εισόδου (πχ. Για 10 KB αρχείου χρειάστηκαν περίπου 17 ms). Αυτό οφείλεται στο γεγονός ότι η κωδικοποίηση επί της ουσίας αποτελείται απο απλές XOR πράξεις, οι οποίες δεν απαιτούν ιδιαίτερους και περίπλοκους υπολογισμούς για τα δεδομένα των τμημάτων ελέγχου που δημιουργούνται. Επίσης, η αύξηση του χρόνου όσο μεγαλώνει το μέγεθος είναι περίπου γραμμική. Αυτό ισχύει διότι η δημιουργία κάθε κομματιού είναι ανεξάρτητη από τα υπόλοιπα τμήματα που δημιουργήθηκαν, επομένως η παράμετρος που επηρεάζει ουσιαστικά το χρόνο, είναι η αύξηση του πλήθους των XOR πράξεων που απαιτούνται. Παρατηρούμε επίσης ότι για διάφορα μεγέθη κατάτμισης του εγγράφου ο χρόνος δε μεταβάλλεται ιδιαίτερα. Αυτό ισχύει διότι παρόλο που το πλήθος των τμημάτων προς κωδικοποίηση αυξάνεται, ο συνολικός αριθμός πράξεων για την κωδικοποίησή τους παραμένει σταθερός.

Πείραμα 3

Στο συγκεκριμένο πείραμα υπολογίζουμε το χρόνο αποκωδικοποίησης για διάφορα μεγέθη αρχείων εισόδου. Τα τμήματα εισόδου έχουν το ίδιο μέγεθος, κάτι που σημαίνει ότι όσο αυξάνεται το μέγεθος του αρχείου, τόσο αυξάνεται το πλήθος των τμημάτων εισόδου (άρα και των τμημάτων ελέγχου) για την αποκωδικοποίηση. Τα αποτελέσματα φαίνονται στο ακόλουθο διάγραμμα.



Σχήμα 5.3. Χρόνος Αποκωδικοποίησης για διαφορετικά μεγέθη αρχείων

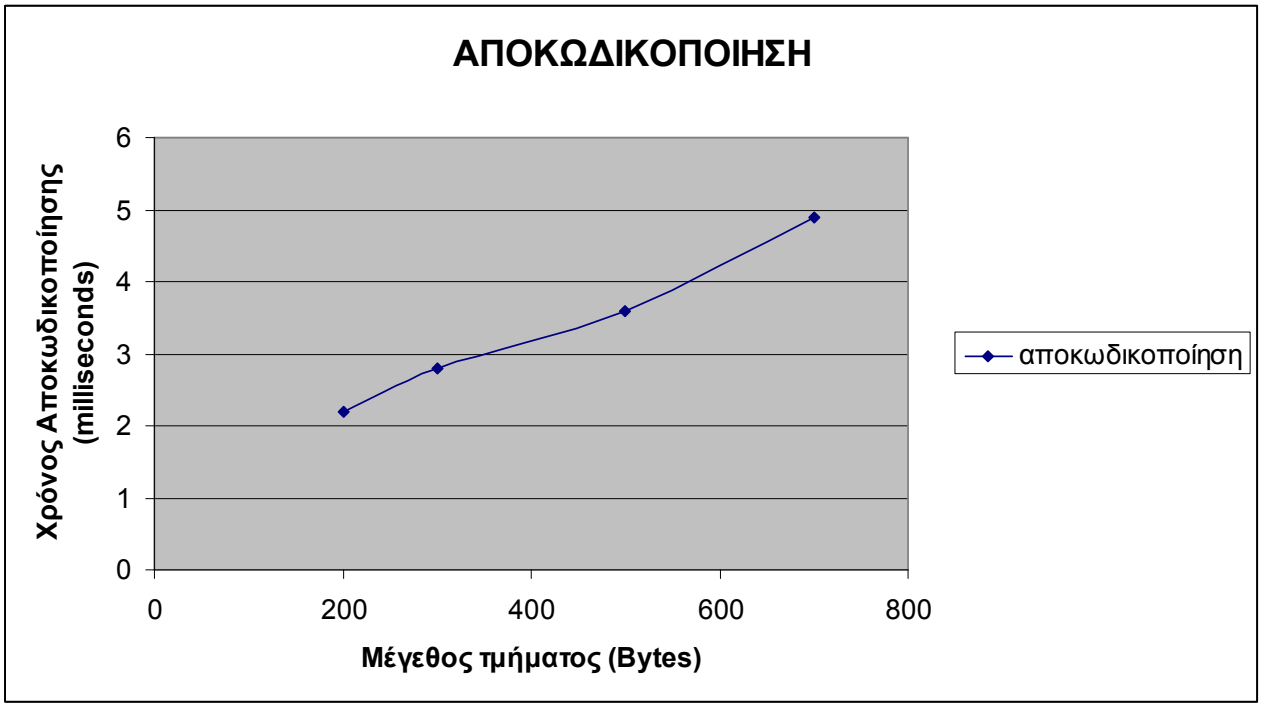
Παρατηρούμε από το Σχήμα 5.3 ότι καθώς αυξάνεται το μέγεθος του αρχείου εισόδου, αυξάνεται γραμμικά και ο χρόνος αποκωδικοποίησής του. Σημειώνουμε ότι λαμβάνουμε υπ' όψιν χρόνους αποκωδικοποίησης για επιτυχή αποκωδικοποίηση, δηλαδή δεν λαμβάνουμε τους χρόνους όταν δεν λαμβάνουμε το επιθυμητό αποτέλεσμα. Τα αποτελέσματα οφείλονται στο γεγονός ότι τα τμήματα ελέγχου που δημιουργούνται για την επιτυχή ολοκλήρωση του αλγορίθμου αυξάνονται καθώς αυξάνεται το μέγεθος του αρχείου εισόδου. Συνεπώς χρειαζόμαστε περισσότερα για την αποκωδικοποίηση, δηλαδή απαιτούνται περισσότερες πράξεις XOR και περισσότεροι έλεγχοι σε κάθε βήμα του αλγορίθμου για την εύρεση τμήματος ελέγχου βαθμού ένα. Επιπλέον, με κάθε αποκωδικοποίηση κάποιου τμήματος εισόδου,

χρειάζονται έλεγχοι για να απομακρυνθούν οι ακμές των τμημάτων ελέγχου που πρόσκεινται στο συγκεκριμένο τμήμα, παράγοντας που συντελεί στην ολική αύξηση του χρόνου αποκωδικοποίησης.

Παρατηρούμε, τέλος, ότι οι χρόνοι αποκωδικοποίησης αυξάνονται, με ανάλογη αύξηση των τμημάτων δεδομένων (m) στα οποία χωρίζεται το αρχείο (άρα και το πλήθος των τμημάτων ελέγχου). Αυτό οφείλεται στο γεγονός ότι, παρόλο που το πλήθος των XOR πράξεων μπορεί να μην αλλάζει ιδιαίτερα, απαιτείται περισσότερος χρόνος για την εύρεση , σε κάθε βήμα, τμήματος βαθμού ένα για να συνεχίσει η αποκωδικοποίηση.

Πείραμα 4

Στο συγκεκριμένο πείραμα υπολογίζουμε το χρόνο αποκωδικοποίησης σε ένα συγκεκριμένο αρχείο εισόδου (σταθερό μέγεθος) αν αυξήσουμε το πλήθος των τμημάτων ελέγχου που χρησιμοποιούμε για την επιτυχή αποκωδικοποίησή του. Παίρνουμε τα αποτελέσματα του Σχήματος 5.4.



Σχήμα 5.4. Χρόνος Αποκωδικοποίησης εγγράφου σταθερού μεγέθους αλλά με μεταβαλλόμενο μέγεθος (άρα και πλήθος) των παραγόμενων Τμημάτων Ελέγχου

Παρατηρούμε από το Σχήμα 5.4 ότι υπάρχει μια αύξηση στο χρόνο αποκωδικοποίησης του αρχείου, η οποία ωστόσο δεν είναι πολύ σημαντική. Αυτό οφείλεται στο γεγονός ότι με μικρή αύξηση των τμημάτων ελέγχου η επιτυχία του αλγορίθμου αυξάνεται σημαντικά, επομένως και με βάση τη Robust Soliton κατανομή δεν χρειάζονται ενδεχομένως όλα τα τμήματα ελέγχου να χρησιμοποιηθούν. Αυτό πρακτικά σημαίνει ότι όσα τμήματα ελέγχου και να λάβουμε, χρειαζόμαστε μικρό επιπλέον αριθμό (το πολύ $0(\sqrt{k})$ για να πετύχουμε την αποκωδικοποίηση, συνεπώς οι επιπλέον XOR πράξεις για την αποκωδικοποίηση όλων των τμημάτων εισόδου αυξάνονται σε μικρό αριθμό. Άρα και ο ολικός χρόνος δεν μεταβάλλεται σημαντικά.

Πείραμα 5

Στο πείραμα αυτό θέλουμε να υπολογίσουμε το κόστος αποθήκευσης των τμημάτων ελέγχου στο δίκτυο. Το κόστος αυτό ισούται με το πλήθος των βημάτων που ακολουθούν τα μηνύματα αποθήκευσης έως ότου αποθηκευτούν όλα τα τμήματα σε διάφορους κόμβους. Τα στοιχεία του δικτύου που μας αφορούν στο συγκεκριμένο πείραμα φαίνονται στον πίνακα 5.3.

Πίνακας 5.3 Στοιχεία δικτύου για μέτρηση κόστους αποθήκευσης

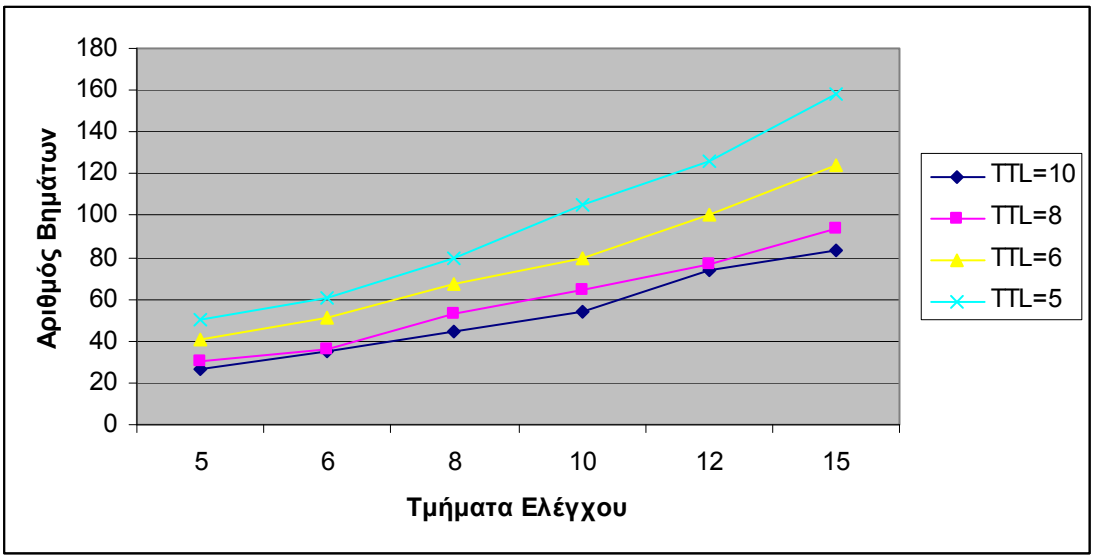
Πλήθος κόμβων	100
Πλήθος γειτόνων κάθε κόμβου	5

Για μεγαλύτερη διασπορά στην αποθήκευση των τμημάτων, επιλέγουμε ένα TTL το οποίο καθορίζει το βάθος στο οποίο θα φτάσει το μήνυμα αποθήκευσης μέχρι να βρεθεί ο κόμβος ο οποίος θα αποκτήσει το τμήμα. Συνοπτικά, η διαδικασία έχει ως εξής.

Ο αρχικός κόμβος επιλέγει τυχαία και ομοιόμορφα ένα γείτονά του στον οποίο στέλνει το μήνυμα αποθήκευσης με ορισμένο TTL. Ο γείτονας που θα το λάβει το προωθεί σε έναν από τους δικούς του γείτονες, μειώνοντας κατά ένα το TTL. Η διαδικασία επαναλαμβάνεται (χωρίς να προωθηθεί σε κόμβο που έχει ήδη φτάσει) μέχρι να μηδενιστεί το TTL. Στον κόμβο που θα καταλήξει, θα αποθηκευτεί το τμήμα.

Αν ο τελικός κόμβος έχει ήδη στη μνήμη του το συγκεκριμένο τμήμα, αυξάνει το TTL κατά ένα και το προωθεί σε έναν γείτονά του, στον οποίο θα καταλήξει το τμήμα.

Το πείραμα πραγματοποιήθηκε για διάφορα TTL και διαφορετικό πλήθος τμημάτων ελέγχου προς αποθήκευση. Τα αποτελέσματα φαίνονται στο σχήμα 5.5.

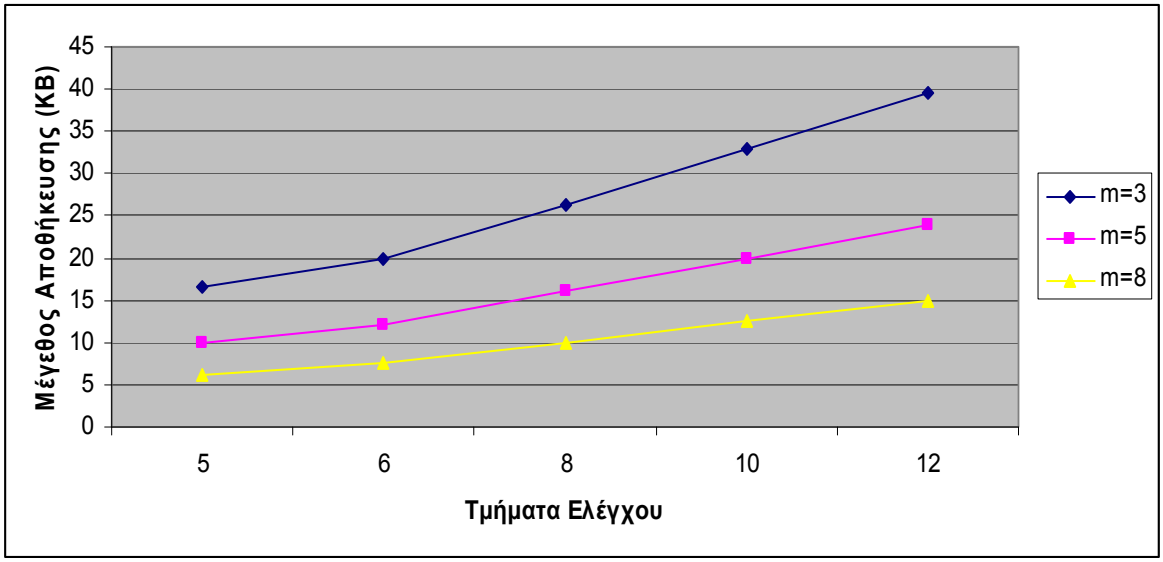


Σχήμα 5.5 Κόστος αποθήκευσης τμημάτων ελέγχου στο δίκτυο, για διάφορα TTL

Στο σχήμα 5.5 παρατηρούμε αρχικά ότι για περισσότερα τμήματα ελέγχου, το κόστος αποθήκευσης είναι αναλόγως μεγαλύτερο. Αυτό είναι λογικό καθώς όταν έχουμε περισσότερα τμήματα, πρέπει να σταλούν περισσότερα μηνύματα στους κόμβους, μέχρι να αποθηκευτούν όλα. Επίσης, παρατηρούμε μια γραμμική αύξηση των βημάτων για κάθε TTL, όσο αυξάνονται τα τμήματα. Αυτό ισχύει, διότι για κάθε επιπλέον τμήμα, απαιτούνται επιπρόσθετα μηνύματα, όσο είναι και το TTL. Αν ο τελικός κόμβος (TTL=0), έχει ήδη το τμήμα, τότε προστίθεται νέο βήμα, για να βρεθεί ο κόμβος που δεν το έχει, ώστε να το αποθηκεύσει.

Πείραμα 6

Στο πείραμα 6 θέλουμε να υπολογίσουμε το αποθηκευτικό κόστος με το οποίο επιβαρύνεται το δίκτυο αν προσθέσουμε επιπλέον τμήματα ελέγχου για κάποιο έγγραφο. Το έγγραφο που χρησιμοποιήθηκε έχει ορισμένο μέγεθος, ίσο με 10 KB και παίρνουμε περιπτώσεις όπου το έχουμε αρχικά χωρίσει σε διαφορετικό πλήθος τμημάτων δεδομένων. Τα τμήματα ελέγχου έχουν το ίδιο μέγεθος με τα τμήματα δεδομένων, συνεπώς το μέγεθος κάθε τέτοιου τμήματος αυξάνεται ανάλογα. Παίρνουμε τα κάτωθι αποτελέσματα (Σχήμα 5.6).



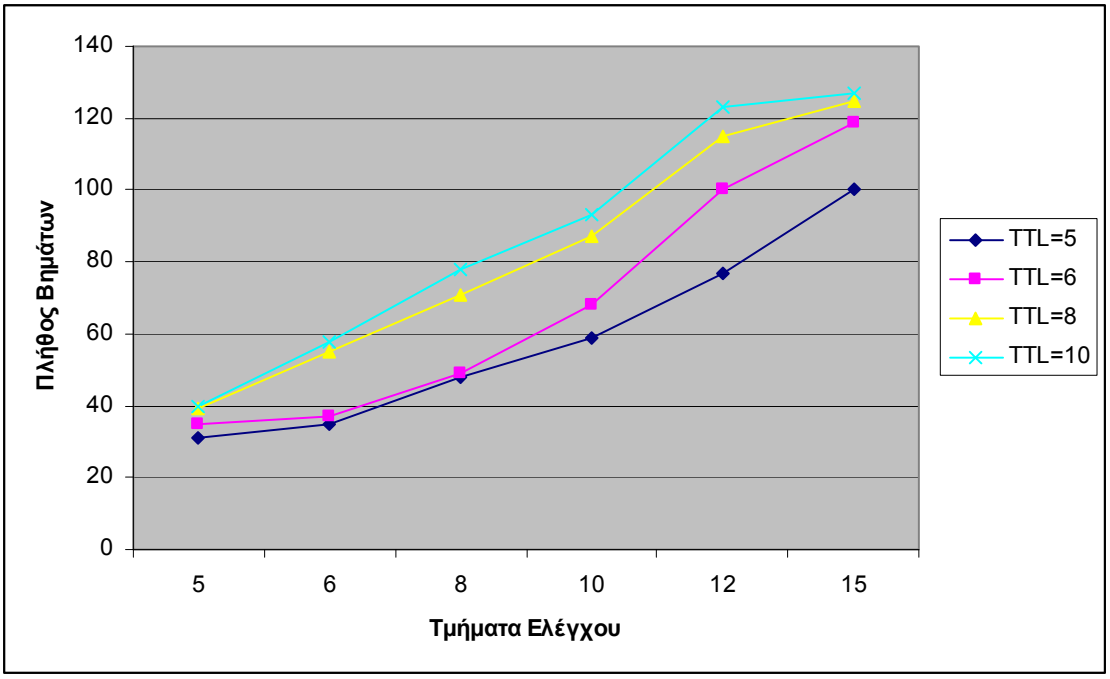
Σχήμα 5.6. Μέγεθος αποθήκευσης για διαφορετικό αριθμό τμημάτων δεδομένων

Από το Σχήμα 5.6 παρατηρούμε ότι αν χωρίσουμε ένα έγγραφο σε περισσότερα τμήματα, τα επιπρόσθετα τμήματα ελέγχου που αποθηκεύονται στο δίκτυο έχουν μικρότερο μέγεθος. Για παράδειγμα, χωρίζοντας το έγγραφο που χρησιμοποιήσαμε (10KB) σε 5 τμήματα, κάθε τμήμα που βρίσκεται στο δίκτυο έχει μέγεθος 2KB. Αν όμως το χωρίσουμε σε 3 τμήματα, κάθε τμήμα έχει μέγεθος 3.33 KB. Συνεπώς, καταλήγουμε στο ότι το κόστος αποθήκευσης κάθε τμήματος ελέγχου στο δίκτυο, είναι αντιστρόφως ανάλογο με το πλήθος των τμημάτων δεδομένων στα οποία χωρίζουμε το έγγραφο. Αν και αυτό είναι σχετικά προφανές, δε σημαίνει ωστόσο ότι πάντα είναι προτιμότερο να χωρίζουμε ένα έγγραφο σε πολλά τμήματα. Αυτό διότι πρέπει να λάβουμε υπόψιν ότι όσο περισσότερα είναι τα τμήματα που μπορούν να δώσουν το έγγραφο, τόσο περισσότερα είναι και τα μηνύματα που πρέπει να στείλει ένας κόμβος, προκειμένου να τα ανακτήσει.

Πείραμα 7

Στο πείραμα αυτό στόχος είναι να μελετήσουμε το κόστος αναζήτησης, σε πλήθος μηνυμάτων, για να ανακτήσει ένας κόμβος κάποιο αρχείο. Τα χαρακτηριστικά του δικτύου που έχουν σημασία για το πείραμα 7 είναι τα ίδια με αυτά του Πίνακα 5.3. Το TTL που χρησιμοποιείται εδώ, υποδηλώνει το μέγιστο βάθος κόμβων στο οποίο μπορεί να φτάσει η αναζήτηση των τμημάτων ελέγχου. Για διαφορετικά TTL και

λήθος τμημάτων ελέγχου που ανακτώνται, πήραμε τα αποτελέσματα του Σχήματος 5.7.



Σχήμα 5.7. Κόστος αναζήτησης τμημάτων για διαφορετικά TTL

Από το Σχήμα 5.7 συμπεραίνουμε ότι με την αύξηση των τμημάτων ελέγχου, αυξάνονται ανάλογα και τα βήματα που πραγματοποιούνται για την ανάκτησή τους. Επιπλέον, αυξάνοντας το TTL παρατηρούμε ότι για μικρότερο πλήθος τμημάτων ο αριθμός των βημάτων είναι περίπου ίδιος. Αυτό διότι όταν τα τμήματα είναι αποθηκευμένα σε κοντινούς κόμβους, για μικρά και μεγάλα TTL θα ακολουθηθεί η ίδια πορεία για την ανάκτησή τους. Όσο αυξάνεται το πλήθος των τμημάτων (όπου η διασπορά τους είναι μεγαλύτερη) για μεγαλύτερα TTL ψάχνουμε μεγαλύτερο εύρος κόμβων. Ενδέχεται λοιπόν, να ψάξουμε σε μεγαλύτερο βάθος για την ανάκτηση κάποιου τμήματος. Είναι προφανές, ότι για μεγαλύτερα TTL το ποσοστό επιτυχίας ανάκτησης είναι σίγουρα μεγαλύτερο από ότι για μικρά TTL. Όταν συνεπώς έχουμε μικρό TTL, για να έχουμε επιτυχία σημαίνει ότι τα τμήματα βρίσκονται σε κοντινούς κόμβους. Για περισσότερα τμήματα, τα μικρά TTL φαίνεται να ευνοούνται, ωστόσο τα ποσοστά επιτυχίας είναι μικρότερα όπως θα δούμε σε επόμενο πείραμα.

Τα συμπεράσματα που εξάγαμε από τα προηγούμενα πειράματα, όσον αφορά τη λειτουργία της μεθόδου των LT κωδικών, παρουσιάζονται συνοπτικά στον πίνακα 5.4.

Πίνακας 5.4. Συμπεράσματα για τη μέθοδο των LT κωδικών

1. Πετυχαίνουμε καλύτερη συμπεριφορά της Robust Soliton κατανομής λαμβάνοντας μεγάλες τιμές για το c (π.χ. 0.5)και μικρότερες για το δ (π.χ. 0.03)
2. Ο χρόνος κωδικοποίησης της μεθόδου εξαρτάται από το πλήθος των τμημάτων ελέγχου που δημιουργούνται.
3. Ο χρόνος κωδικοποίησης αυξάνεται γραμμικά με την αύξηση στη δημιουργία τμημάτων ελέγχου και είναι ικανοποιητικά χαμηλός.
4. Ο χρόνος αποκωδικοποίησης έχει την ίδια μορφή με το χρόνο κωδικοποίησης. Είναι λίγο μεγαλύτερος, διότι απαιτείται επιπλέον χρόνος για την αφαίρεση ακμών μετά από μια επιτυχία και για την εύρεση του επόμενου κόμβου βαθμού 1.
5. Το κόστος αποθήκευσης και αναζήτησης αυξάνεται ανάλογα με το πλήθος των τμημάτων ελέγχου που παράγονται. (σε πλήθος βημάτων)
Το επιπρόσθετο κόστος αποθήκευσης (σε χώρο) κάποιου τμήματος ελέγχου, εξαρτάται από το πλήθος των τμημάτων δεδομένων που δημιουργούνται κατά την κωδικοποίηση. Ωστόσο, το κόστος αυτό είναι πολύ μικρότερο από ολόκληρο το έγγραφο και αυξάνει κατά πολύ τη διαθεσιμότητα.

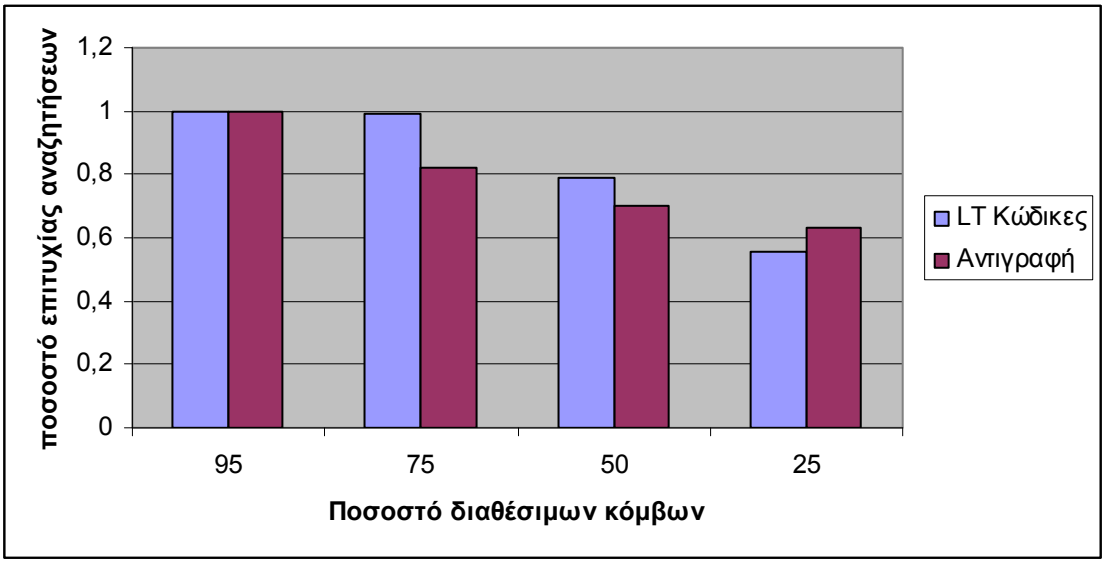
5.3 Πειράματα στο Δίκτυο Ομότιμων

Στην παρούσα Ενότητα παρουσιάζουμε πειράματα που έγιναν στο Δίκτυο Ομότιμων που υλοποιήθηκε. Το Δίκτυο αυτό διαχειρίζεται XML έγγραφα, τα οποία κωδικοποιούνται με τους LT Κωδικούς και αποθηκεύονται σε διάφορους κόμβους. Οι ιδιότητες καθώς και οι διαδικασίες κωδικοποίησης, αποθήκευσης αλλά και ανάκτησης ενός XML εγγράφου, έχουν αναλυθεί στα Κεφάλαια 3 και 4. Εδώ, στόχος είναι να διαπιστώσουμε αν ωφελεί και σε ποιους τομείς η Κωδικοποίηση Απαλοιφής. Ως μέτρο σύγκρισης, χρησιμοποιήσαμε τη μέθοδο της Αντιγραφής στο Δίκτυο και βλέπουμε τις επιμέρους διαφορές των δύο μεθόδων μεταξύ τους.

Πείραμα 8

Στο πείραμα 8 θεωρούμε ότι έχουμε αποτυχία κάποιων κόμβων του δικτύου. Ως αποτυχία ενός κόμβου ορίζουμε την κατάσταση στην οποία ο κόμβος βρίσκεται εκτός Δικτύου. Συνεπώς και τα δεδομένα που κρατά στη λίστα του δεν είναι διαθέσιμα για προσπέλαση. Θέτουμε διαφορετικά ποσοστά κόμβων που αποτυγχάνουν και συγκεκριμένα χρησιμοποιούμε το 95% των κόμβων του Δικτύου, το 75% αλλά και το 25% από αυτούς. Θεωρούμε ότι το Δίκτυο περιλαμβάνει 100 κόμβους και τα XML έγγραφα που χρησιμοποιούνται έχουν τάξη μεγέθους 3KB. Επιπλέον, τα τμήματα δεδομένων είναι 3 και τα παραγόμενα τμήματα ελέγχου είναι 6.

Για την υλοποίηση του πειράματος αυτού θεωρούμε ότι το δίκτυο είναι ιδιαίτερα δυναμικό. Παίρνουμε κάποιες περιπτώσεις όπου το ποσοστό των κόμβων που δεν βρίσκονται στο δίκτυο κυμαίνεται από 5% έως 75%. Το πείραμα, λοιπόν, δείχνει το ποσοστό επιτυχίας στην ανάκτηση ενός εγγράφου XML για διάφορες τιμές αποτυχιών, με διαφορετική διαθεσιμότητα κόμβων στο δίκτυο. Τα αποτελέσματα φαίνονται στο Σχήμα 5.8.



Σχήμα 5.8. Επιτυχία αναζητήσεων για διαφορετικά ποσοστά διαθεσιμότητας κόμβων

Το πείραμα αυτό πραγματοποιήθηκε τόσο για τη μέθοδο των LT κωδικών, όσο και τη μέθοδο της Αντιγραφής ως μέτρο σύγκρισης. Από το Σχήμα 5.8 προκύπτουν διάφορα συμπεράσματα. Στη μέθοδο της Αντιγραφής, η Ανοχή Αποτυχιών επιτυγχάνεται αν , για όσους κόμβους αποτύχουν στο Δίκτυο, δημιουργηθούν αντίστοιχο πλήθος αντίγραφα του αρχείου, τα οποία αποθηκεύονται στους κόμβους που υπάρχουν σε αυτό. Για παράδειγμα, μια Ανοχή με τιμή 3 σημαίνει ότι πρέπει να δημιουργηθούν 3 αντίγραφα του αρχείου τα οποία να είναι διαθέσιμα στο Δίκτυο. Για την Κωδικοποίηση Απαλοιφής με σχήμα m από n (3 από 6 στην περίπτωσή μας) η πιθανότητα επιτυχούς ανάκτησης ενός εγγράφου υπολογίζεται από την ακόλουθη εξίσωση:

$P_F = 1 - \sum_{i=m+1}^{n} (P_D)^i$, όπου P_F είναι η πιθανότητα ανάκτησης του εγγράφου και P_D είναι η πιθανότητα ένας κόμβος να έχει αποτύχει.

Για να πετύχει η Κωδικοποίηση Απαλοιφής, θα πρέπει να βρίσκονται κάθε στιγμή στο δίκτυο οποιοιδήποτε m κόμβοι, οι οποίοι θα περιέχουν κάποιο τμήμα ελέγχου. Προφανώς, η πιθανότητα επιτυχούς ανάκτησης ενός εγγράφου ισούται με ένα μείον την πιθανότητα να έχουν απομείνει λιγότεροι από m κόμβοι στο δίκτυο.

Παρατηρούμε ότι όταν το ποσοστό των διαθέσιμων κόμβων είναι στο 95%, οι δύο μέθοδοι έχουν μεγάλη απόδοση και υπάρχει μεγάλη πιθανότητα (που αγγίζει το 99.99%) να ανακτηθεί το ζητούμενο έγγραφο. Η Κωδικοποίηση Απαλοιφής δίνει λίγο μεγαλύτερη πιθανότητα (0.99999 έναντι 0.99 της Αντιγραφής), διότι η διαθεσιμότητα του εγγράφου μέσω των τμημάτων ελέγχου είναι μεγαλύτερη από αυτή μέσω της μεθόδου της Αντιγραφής. Ωστόσο, οι δύο μέθοδοι έχουν παραπλήσιο ποσοστό επιτυχίας. Αυτό είναι λογικό, διότι θα υπάρχουν σχεδόν σίγουρα κόμβοι οι οποίοι να διαθέτουν κάποιο αντίγραφο του αρχείου το οποίο μπορεί να ανακτηθεί κατά την Αντιγραφή, αλλά και υπάρχουν αρκετοί συνδυασμοί από m κόμβους που μπορούν να δώσουν το απαραίτητο πλήθος από τμήματα ελέγχου, ώστε να επιτύχει και η Κωδικοποίηση Απαλοιφής.

Όταν το ποσοστό των διαθέσιμων κόμβων πέφτει στο 75% (δηλαδή έχουμε 25% κόμβους που απέτυχαν) η Κωδικοποίηση Απαλοιφής υπερτερεί της Αντιγραφής. Αυτό

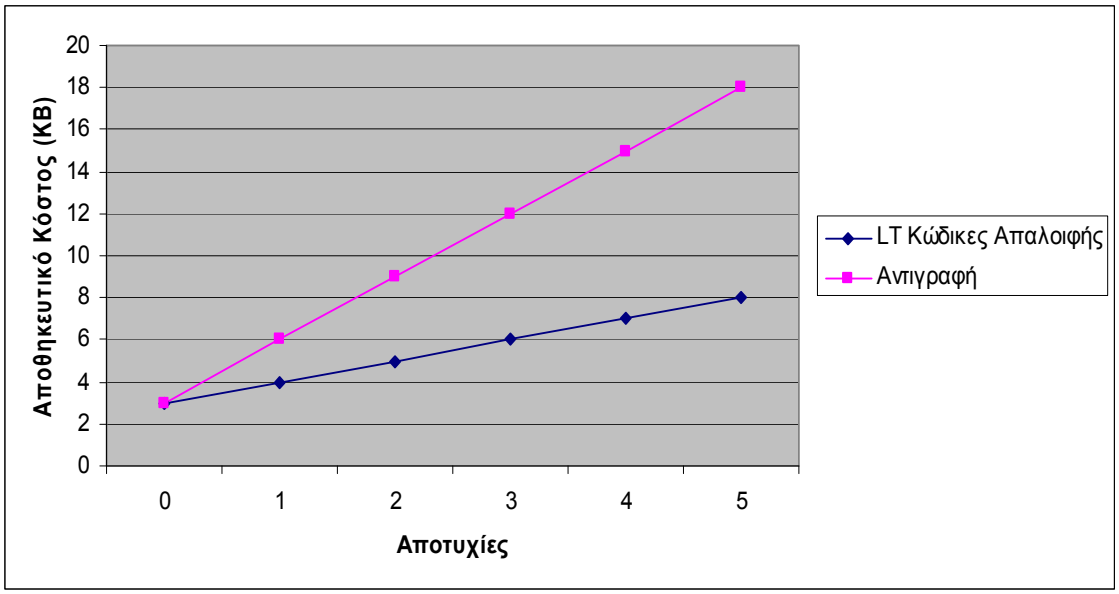
οφείλεται στο γεγονός ότι, σύμφωνα με την Αντιγραφή, κάποια αντίγραφα έχουν αρκετά μεγάλη πιθανότητα να βρίσκονται αποθηκευμένα στους κόμβους που έχουν αποτύχει. Αυτό είναι εις βάρος της διαθεσιμότητας του εγγράφου, κάτι που αντικατοπτρίζεται από το (ελαφρώς) μειωμένο ποσοστό επιτυχών αναζητήσεων για το έγγραφο. Η Κωδικοποίηση Απαλοιφής, από την άλλη, έχει καλύτερη συμπεριφορά, διότι ο συνδυασμός από m διαφορετικούς κόμβους που να υπάρχουν στο δίκτυο και να μπορούν να δώσουν κάποιο τμήμα ελέγχου είναι σαφώς μεγαλύτερος από το πλήθος αυτούσιων αντιγράφων που έχουν μείνει στο δίκτυο. Δηλαδή, η πιθανότητα στο 25% των κόμβων που έχουν αποτύχει, να υπάρχουν αρκετοί κόμβοι που να διαθέτουν Τμήμα ελέγχου που ζητάμε, είναι μικρότερη από την αντίστοιχη πιθανότητα να υπάρχει ένα ή περισσότερα αντίγραφα στους κόμβους αυτούς.

Ωστόσο, όταν το ποσοστό των κόμβων που βρίσκονται στο Δίκτυο είναι πολύ μικρό (αποτυχία κόμβων στο 75%) η μέθοδος της Αντιγραφής λειτουργεί καλύτερα από την Κωδικοποίηση Απαλοιφής. Αυτό οφείλεται στο γεγονός ότι για να πετύχει η Κωδικοποίηση Απαλοιφής απαιτούνται τουλάχιστον m κόμβοι που να έχουν ζητούμενα τμήματα ελέγχου. Αντιθέτως, η Αντιγραφή απαιτεί τουλάχιστον ένα κόμβο να υπάρχει που να διαθέτει αντίγραφο του εγγράφου. Όταν λοιπόν οι διαθέσιμοι κόμβοι είναι πολύ λίγοι, η ύπαρξη συνδυασμών m διαφορετικών κόμβων που να δίνουν τα απαραίτητα τμήματα ελέγχου έχει σχετικά μικρή πιθανότητα. Η ύπαρξη, ωστόσο, ενός ή περισσοτέρων κόμβων που να έχουν αντίγραφο του εγγράφου είναι πιο πιθανή. Συνεπώς, η διαθεσιμότητα του εγγράφου μέσω της Αντιγραφής συγκεντρώνει μεγαλύτερες πιθανότητες. Βλέπουμε, λοιπόν, ότι για σχετικά μικρότερα ποσοστά αποτυχιών, η Κωδικοποίηση Απαλοιφής συμπεριφέρεται καλύτερα από την Αντιγραφή. Το αντίθετο συμβαίνει όταν τα ποσοστά αποτυχιών των κόμβων αυξάνονται σημαντικά.

Πείραμα 9

Ένα άλλο πείραμα που παρουσιάζει ενδιαφέρον είναι το κόστος αποθήκευσης που επιβάλλεται στο δίκτυο κατά τη δημιουργία αντιγράφων των αρχείων. Συγκεκριμένα, το πείραμα πραγματοποιήθηκε στις μεθόδους Αντιγραφής και LT Κωδικών. Υπολογίσαμε το επιπλέον κόστος που επιβάλλουν οι δύο αυτές μέθοδοι στο δίκτυο,

ώστε να διατηρήσει το δίκτυο τα ίδια επίπεδα διαθεσιμότητας των αρχείων όταν έχουμε αποτυχίες κόμβων διαφορετικού πλήθους. Στο πείραμά μας χρησιμοποιήθηκε ένα XML έγγραφο μεγέθους 3KB, ενώ το κάθε τμήμα ελέγχου που προκύπτει από την Κωδικοποίηση Απαλοιφής έχει μέγεθος 1KB. Τα αποτελέσματα φαίνονται στην ακόλουθη παράσταση (Σχήμα 5.9):



Σχήμα 5.9 Επιπρόσθετο Αποθηκευτικό Κόστος σε σχέση με διαφορετικό πλήθος Αποτυχιών

Κατά την Αντιγραφή, για να διατηρήσουμε το αρχείο στα ίδια επίπεδα διαθεσιμότητας, θα πρέπει να δημιουργούμε τέτοιο πλήθος αντιγράφων του αρχείων όσο και το πλήθος των κόμβων που αποχωρούν από το δίκτυο. Αυτό διότι αν φύγει κάποιος κόμβος ο οποίος κρατά ένα αντίγραφο στη βάση του, τότε η διαθεσιμότητα μειώνεται κατά ένα. Συνεπώς χρειαζόμαστε ένα επιπλέον αντίγραφο για να αντισταθμίσουμε αυτό το γεγονός. Αν δηλαδή αποτύχει ένας κόμβος, πρέπει να δημιουργήσουμε ένα αντίγραφο των 3KB σε μέγεθος. Αν πέσουν δύο κόμβοι χρειαζόμαστε δύο αντίγραφα, δηλαδή 6KB και το κόστος αυξάνεται, όσο αυξάνονται οι αποτυχίες κόμβων.

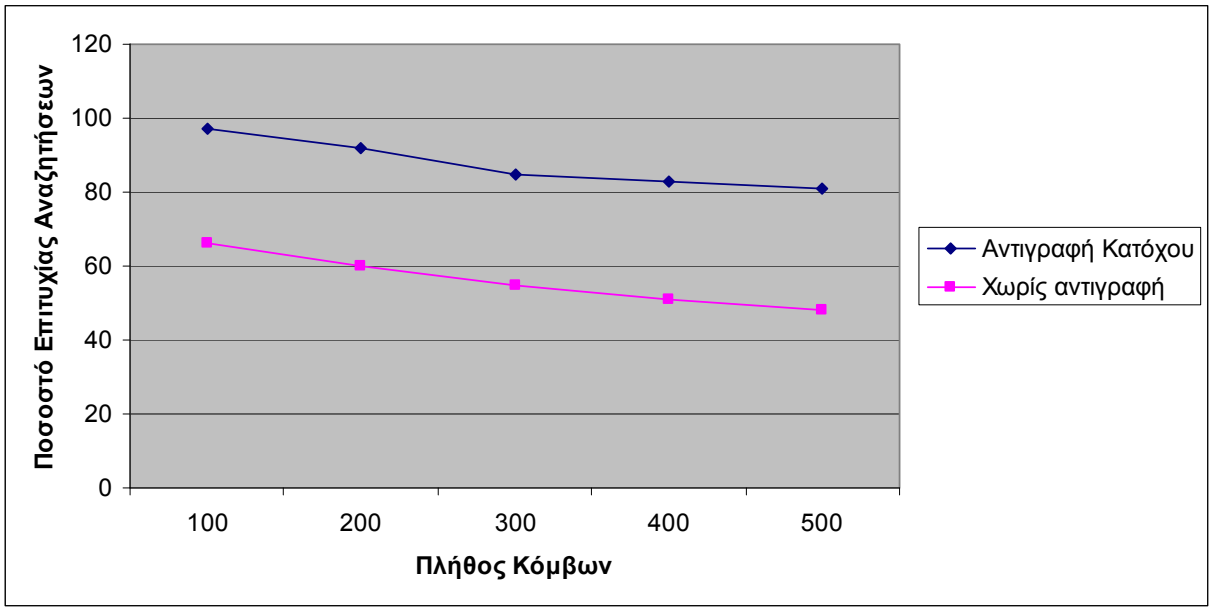
Στην Κωδικοποίηση Απαλοιφής, αντιθέτως, το έγγραφο χωρίστηκε σε 3 τμήματα δεδομένων και κωδικοποιήθηκε σε έξι τμήματα ελέγχου, μεγέθους 1KB. Αυτό σημαίνει ότι κάθε κόμβος κρατά ένα μόνο τμήμα ελέγχου. Αν, λοιπόν, αποχωρήσει

κάποιος κόμβος που ενδεχομένως κρατά τμήμα του αρχείου, τότε απλώς χρειαζόμαστε ένα επιπλέον τμήμα ελέγχου για να διατηρήσουμε το ίδιο επίπεδο διαθεσιμότητας του εγγράφου. Χρειαζόμαστε, δηλαδή, 1KB επιπλέον στο δίκτυο και όχι αντίγραφο ολόκληρου του εγγράφου.

Από την παραπάνω ανάλυση (Σχήμα 5.9) καταλήγουμε στο συμπέρασμα ότι η μέθοδος της Αντιγραφής επιφέρει μεγάλο κόστος στη διατήρηση της διαθεσιμότητας όταν έχουμε αποτυχίες κόμβων. Η Κωδικοποίηση Απαλοιφής συμπεριφέρεται πολύ καλύτερα, διότι το επιπρόσθετο κόστος είναι πολύ μικρότερο, κατά την αποτυχία κόμβων. Στην περίπτωση του πειράματος που πραγματοποιήθηκε, η Κωδικοποίηση Απαλοιφής χρειάζεται περίπου το 1/3 του αποθηκευτικού χώρου που χρειάζεται η Αντιγραφή για να διατηρηθεί η διαθεσιμότητα στα ίδια επίπεδα. Αυτό, σε ένα δυναμικό δίκτυο, όπως είναι τα δίκτυα Ομότιμων, είναι ιδιαίτερα σημαντικό.

Πείραμα 10

Στο ακόλουθο πείραμα υπολογίσαμε το ποσοστό επιτυχίας στην αναζήτηση του XML εγγράφου το οποίο έχει κωδικοποιηθεί με την LT μέθοδο, για διαφορετικό πλήθος κόμβων που συμμετέχουν στο δίκτυο. Για κάθε πλήθος κόμβων κάναμε αναζήτηση 5 φορές για το αρχείο και πήραμε το μέσο ποσοστό επιτυχίας των αναζητήσεων. Στη μία περίπτωση δε δημιουργούνταν νέα αντίγραφα των τμημάτων ελέγχου του εγγράφου έπειτα από μία επιτυχή αναζήτηση. Στη δεύτερη περίπτωση, μετά από μια πετυχημένη αναζήτηση, αποθηκεύονταν αντίγραφα των τμημάτων ελέγχου που ανακτήθηκαν με βάση την αντιγραφή κατόχου που αναλύσαμε στο προηγούμενο κεφάλαιο (Ενότητα 4.5.2). Το TTL που θέσαμε για την αναζήτηση είναι δέκα (10), δηλαδή το έγγραφο αναζητείται μέχρι και σε βάθος δέκα από τον κόμβο που το ζήτησε. Τα αποτελέσματα φαίνονται στο διάγραμμα που ακολουθεί:

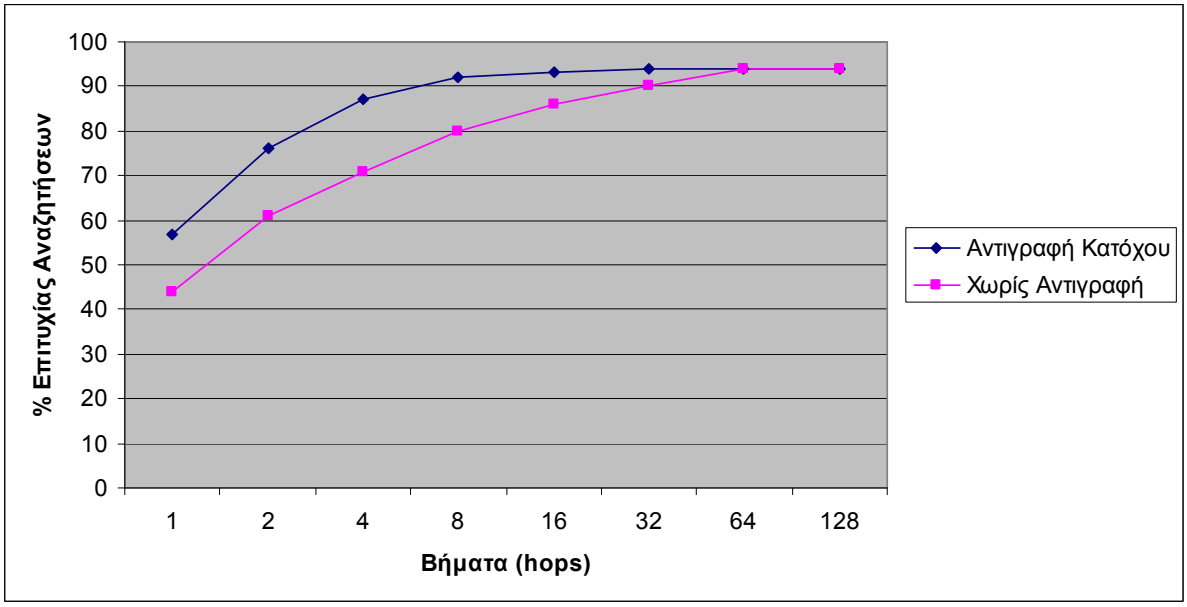


Σχήμα 5.10 Ποσοστό Επιτυχίας Αναζητήσεων Χωρίς Αντιγραφή και με Αντιγραφή Κατόχου

Παρατηρούμε στο Σχήμα 5.10 ότι με την Αντιγραφή Κατόχου τα ποσοστά επιτυχίας είναι σαφώς μεγαλύτερα σε σύγκριση με την περίπτωση όπου δεν δημιουργούμε επιπλέον αντίγραφα. Ατό είναι λογικό διότι με την αντιγραφή κατόχου αυξάνουμε τη διαθεσιμότητα του εγγράφου, αφού πλέον υπάρχουν περισσότερα τμήματα που αποκωδικοποιούν το έγγραφο. Επίσης, περισσότεροι είναι και οι κόμβοι που περιέχουν ένα τέτοιο τμήμα, συνεπώς η βιωσιμότητα του αρχείου είναι επίσης μεγαλύτερη. Βέβαια, αυξάνοντας το πλήθος των κόμβων παρατηρούμε μια μείωση στα ποσοστά επιτυχίας, που ωστόσο παραμένουν υψηλά. Αυτό συμβαίνει διότι η διασπορά των κόμβων, άρα και των τμημάτων, είναι μεγαλύτερη και συνεπώς η πιθανότητα εύρεσης του κατάλληλου πλήθους από αυτά μειώνεται όσο αυξάνονται οι κόμβοι.

Πείραμα 11

Στο συγκεκριμένο πείραμα υπολογίσαμε το ποσοστό επιτυχίας στην αναζήτηση του εγγράφου αυξάνοντας τα βήματα (hops) σε κάθε αναζήτηση. Τα βήματα συμβολίζουν το πλήθος των κόμβων στους οποίους φτάνει το μήνυμα αναζήτησης μέχρι να επέλθει η επιτυχία. Όπως και στο προηγούμενο πείραμα, έτσι και σε αυτό, συγκρίνουμε τη διαδικασία Αντιγραφής Κατόχου και χωρίς δημιουργία Αντιγράφων κατά την επιτυχία. Τα αποτελέσματα παρουσιάζονται παρακάτω (Σχήμα 5.11):



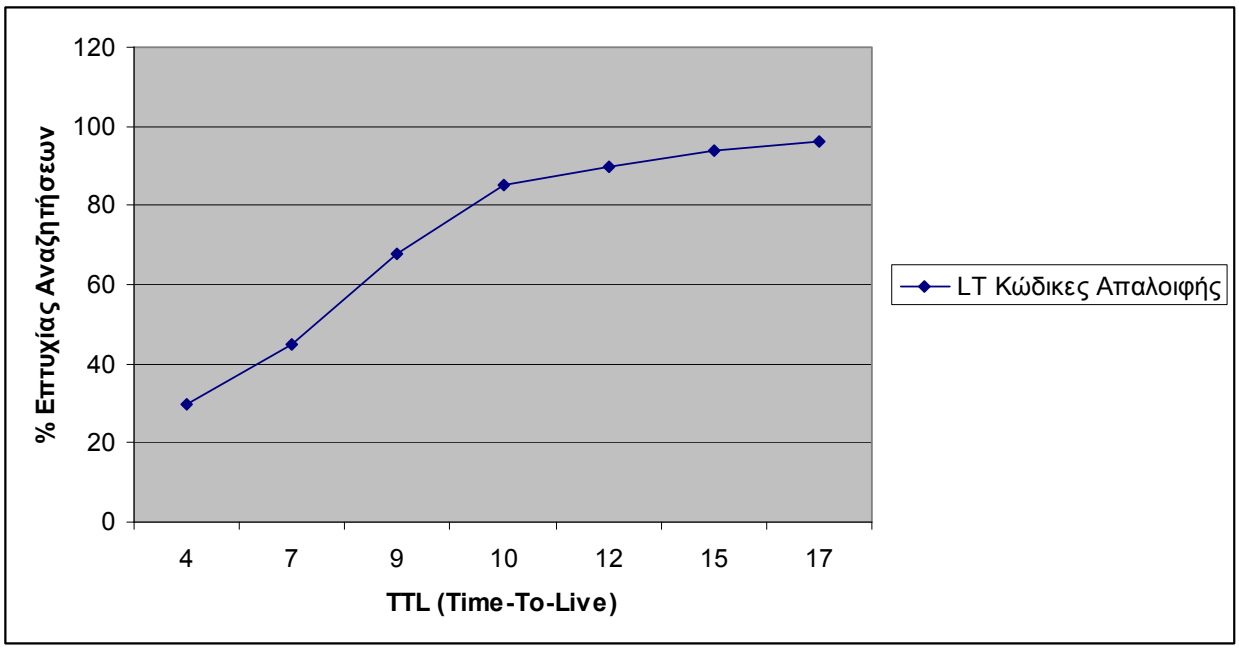
Σχήμα 5.11 Ποσοστό Επιτυχίας Αναζήτησης άναλογα με το πλήθος των Βημάτων Χωρίς Αντιγραφή και με Αντιγραφή Κατόχου

Από τα αποτελέσματα που πήραμε παρατηρούμε ότι με την Αντιγραφή Κατόχου, τα αποτελέσματα είναι καλύτερα. Αυτό οφείλεται στο γεγονός ότι δημιουργώντας αντίγραφα των τμημάτων ελέγχου, περισσότεροι κόμβοι περιέχουν από ένα τέτοιο τμήμα. Η αναζήτηση, λοιπόν, τέτοιων κόμβων γίνεται πιο εύκολη. Άρα, υπάρχει μεγαλύτερη πιθανότητα κάποιος κόμβος με τμήμα ελέγχου του εγγράφου να βρίσκεται κοντά σε αυτόν που κάνει την αναζήτηση. Συνεπώς και τα βήματα που απαιτούνται για την εύρεση αυτού του κόμβου είναι λιγότερα. Αυτό βοηθά αρκετά στην αποφυγή μεγάλου φόρτου μηνυμάτων στο δίκτυο, κάτι ιδιαίτερα σημαντικό για την καλύτερη λειτουργία του.

Επιπλέον, παρατηρούμε ότι για μεγάλο πλήθος βημάτων (πάνω από 30) οι δύο μέθοδοι συγκλίνουν στο ποσοστό επιτυχημένων αναζητήσεων (πάνω από 90%). Αυτό είναι λογικό διότι η Κωδικοποίηση Απαλοιφής δίνει μεγάλη διαθεσιμότητα για το αρχείο (οποιαδήποτε τρία τμήματα από τα έξι που βρίσκονται αρχικά στο δίκτυο). Συνεπώς, όταν μεγαλώνει τόσο το πλήθος των κόμβων που ελέγχει η αναζήτηση, είναι σχεδόν σίγουρο ότι θα βρεθούν οι ζητούμενοι κόμβοι, είτε δημιουργήσουμε επιπρόσθετα αντίγραφα τμημάτων ελέγχου, είτε όχι.

Πείραμα 12

Στο συγκεκριμένο πείραμα υπολογίζουμε το ποσοστό επιτυχών αναζητήσεων του αρχείου για διαφορετικά TTL. Είπαμε στο προηγούμενο κεφάλαιο ότι σε κάθε μήνυμα αναζήτησης θέτουμε ένα όριο βάθους (βημάτων) στο οποίο διαρκεί το μήνυμα. Αν ξεπεραστεί αυτό το όριο χωρίς να έχει βρεθεί κόμβος που να έχει τμήμα ελέγχου που αποκωδικοποιεί το έγγραφο, η αναζήτηση αποτυγχάνει. Είναι ενδιαφέρον να δούμε πώς διαμορφώνονται οι αναζητήσεις αυξάνοντας το TTL των μηνυμάτων αναζήτησης. Παίρνουμε τα κάτωθι αποτελέσματα (Σχήμα 5.12):



Σχήμα 5.12 Ποσοστό επιτυχίας αναζήτησης για διάφορα TTL

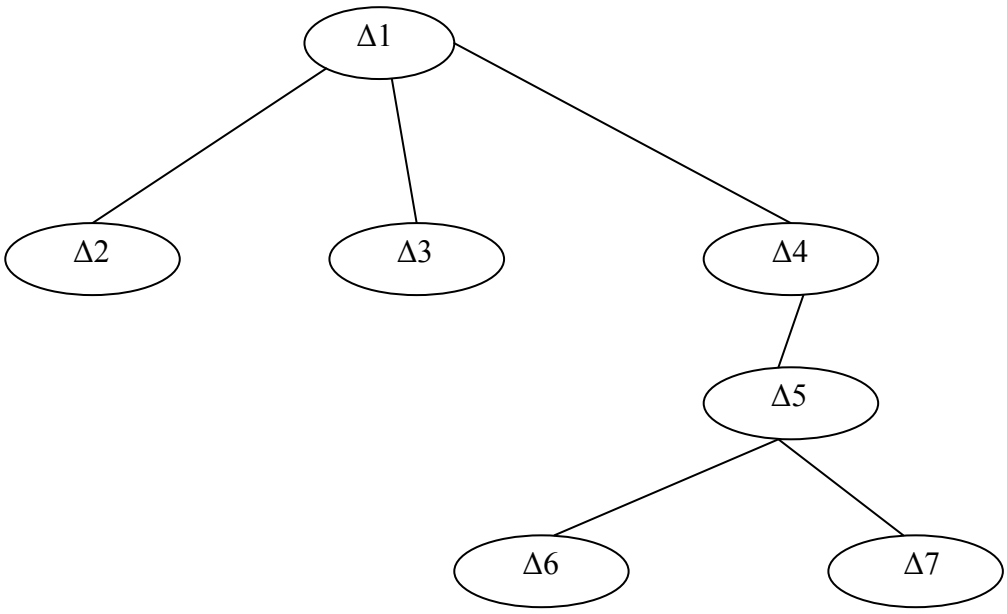
Παρατηρούμε από το παραπάνω Σχήμα ότι αυξάνοντας το TTL πετυχαίνουμε καλύτερα αποτελέσματα στην αναζήτηση του εγγράφου. Συγκεκριμένα, για TTL περίπου δέκα, το ποσοστό επιτυχίας πλησιάζει το 90%. Το γεγονός αυτό εξηγείται διότι αυξάνοντας το TTL, αυξάνεται το βάθος των κόμβων στο οποίο ψάχνουμε για να βρούμε κατάλληλο τμήμα ελέγχου. Αυτό σημαίνει ότι η αναζήτηση επεκτείνεται σε περισσότερους κόμβους, επομένως είναι λογικό αφού ψάχνουμε περισσότερους κόμβους, η πιθανότητα να βρούμε τον κατάλληλο κόμβο αυξάνεται ανάλογα. Ωστόσο,

αξίζει να σημειώσουμε ότι πολλές φορές δεν επιθυμούμε να έχουμε μεγάλο TTL διότι αυξάνεται ο φόρτος μηνυμάτων στους κόμβους του δικτύου, κάτι που σε παράλληλες αναζητήσεις εγγράφων, ενδέχεται να έχει αρνητικές συνέπειες για το δίκτυο. Σε ένα δίκτυο 100 κόμβων, ένα TTL κοντά στο 10 είναι αποδεκτό, διότι παίρνουμε πολύ καλά ποσοστά επιτυχίας, κάνοντας αναζήτηση σε ένα σχετικά μικρό ποσοστό κόμβων.

Πείραμα 13

Στο πείραμα 13 εξετάζουμε τη συμπεριφορά της μεθόδου των LT κωδικών σε ένα XML έγγραφο. Όπως είπαμε στο κεφάλαιο 4, ένα XML έγγραφο αποτελείται από πολλά στοιχεία (elements), τα οποία περιέχουν πληροφορία η οποία δεν εξαρτάται από τα υπόλοιπα δεδομένα του εγγράφου, αλλά μπορεί να αποτελέσει μια ξεχωριστή οντότητα. Συνεπώς, κάποιος κόμβος ο οποίος επιθυμεί να ανακτήσει μέρος της πληροφορίας που περιέχει ένα XML έγγραφο, δεν είναι ωφέλιμο να υποχρεούται να ανακτήσει ολόκληρη την πληροφορία άλλα το τμήμα που ζητά.

Έστω ότι διαθέτουμε το δέντρο του Σχήματος 5.13, το οποίο ορίζει κάποιο XML έγγραφο.



Σχήμα 5.13 Δέντρο XML εγγράφου

Στο πείραμα αυτό εξετάζουμε την περίπτωση όπου ένας κόμβος αναζητά την πληροφορία που περιέχεται στο στοιχείο <Δ4> του εγγράφου. Παίρνουμε δύο ενδεχόμενα για την αναζήτηση. Το ένα ενδεχόμενο είναι να μην έχει γίνει κατάτμηση του εγγράφου σε υποδέντρα του. Στην περίπτωση αυτή η αναζήτηση γίνεται με βάση το file-id του εγγράφου, με στόχο να ανακτηθεί ολόκληρο το αρχείο ώστε η αναζήτηση να είναι επιτυχής. Στο δεύτερο ενδεχόμενο θεωρούμε ότι έχει γίνει κατάτμηση του εγγράφου στα υποδέντρα του και ότι το υποδέντρο που υποδηλώνει το Δ4 βρίσκεται αυτόνομα αποθηκευμένο στο δίκτυο. Στην περίπτωση αυτή το υποδέντρο αυτό χωρίζεται σε m τμήματα και κωδικοποιείται εκ νέου σε n τμήματα ελέγχου. Κάθε τμήμα ελέγχου έχει επιπλέον id το path-id που χαρακτηρίζει το υποδέντρο. Συνεπώς, η αναζήτηση γίνεται με βάση αυτό το path-id.

Τα δύο αυτά ενδεχόμενα τα εξετάζουμε ως προς το ποσοστό επιτυχίας της αναζήτησης για διάφορα TTL. Επίσης υπολογίζουμε το κόστος αναζήτησης των τμημάτων σε πλήθος απαιτούμενων βημάτων μέχρι την ολοκλήρωσή της.

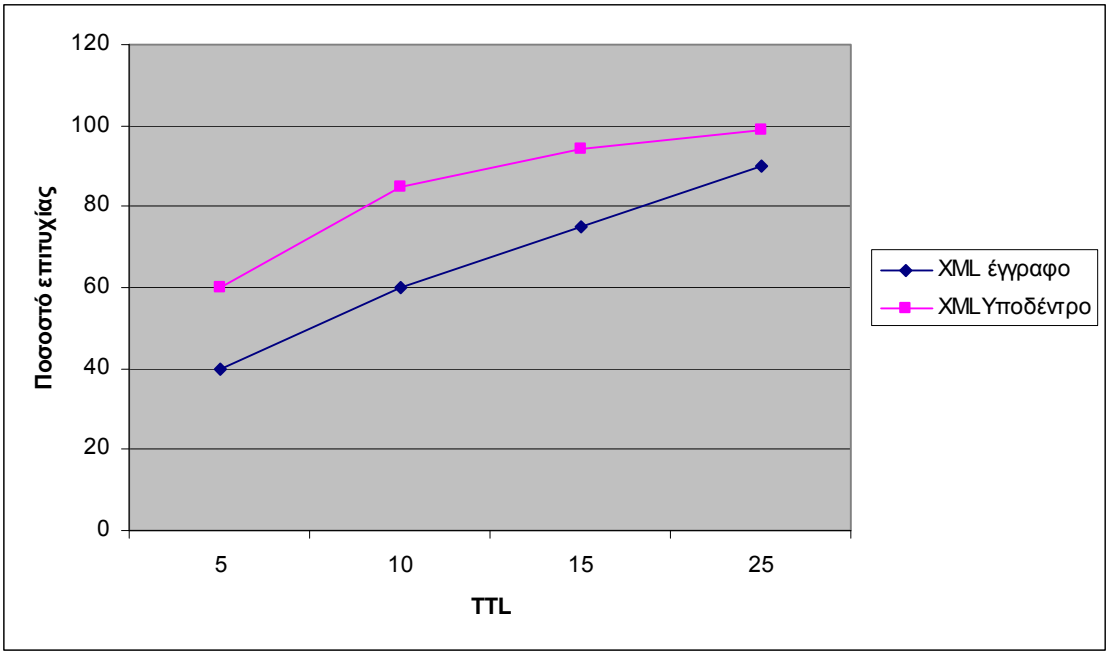
Για το συγκεκριμένο πείραμα προσομοιώσαμε ένα δίκτυο που αποτελείται από 100 κόμβους, με 5 γείτονες να αντιστοιχούν σε κάθε κόμβο. Επιπλέον, το μέγεθος του αρχείου είναι 3KB, ενώ το μέγεθος του υποδέντρου είναι 1KB. Χωρίσαμε τα αρχεία σε 3 τμήματα δεδομένων και τα κωδικοποιήσαμε σε 6 τμήματα ελέγχου το καθένα. Στην περίπτωση ολόκληρου του αρχείου , κάθε τμήμα έχει μέγεθος 1KB (3KB/3), ενώ στη δεύτερη περίπτωση όπου αναζητούμε συγκεκριμένα το υποδέντρο, το μέγεθος κάθε τμήματος ελέγχου είναι 333bytes (1KB/3). Με βάση προηγούμενο πείραμα, παρατηρούμε ότι αν λάβουμε επιπρόσθετα τμήματα (από τα 3 που απαιτούνται) της τάξης του $O(\sqrt{k} * \ln^2(\frac{k}{\delta}))$, έχουμε σχεδόν 100% επιτυχία στην αποκωδικοποίηση. Στην περίπτωση τριών τμημάτων ελέγχου που κωδικοποιούνται σε έξι τμήματα ελέγχου, αρκεί να λάβουμε τέσσερα από τα έξι τμήματα ελέγχου για να έχουμε επιτυχία ανάκτησης. Για το πείραμα θεωρούμε ότι έχουμε ίσο χώρο στη διάθεσή μας από το δίκτυο για την αποθήκευση των τμημάτων στις δύο περιπτώσεις. Αν θεωρήσουμε ότι ο χώρος ισούται με το διπλάσιο του μεγέθους του εγγράφου (δηλαδή 6KB), τότε συμπεραίνουμε ότι μπορούμε να αποθηκεύσουμε τα 6 τμήματα ελέγχου

του εγγράφου (6*1KB=6KB), αλλά μπορούμε να αποθηκεύσουμε 18 τμήματα ελέγχου εφόσον έχουμε αυτόνομο το υποδέντρο (18*333 bytes= 6 KB) Τα δεδομένα εισόδου παρουσιάζονται συγκεντρωτικά στον πίνακα 5.5.

Πίνακας 5.5 Δεδομένα εισόδου για το πείραμα 13

Πλήθος κόμβων	100
Πλήθος γειτόνων κάθε κόμβου	5
Μέγεθος τμήματος αρχείου	1KB
Μέγεθος τμήματος υποδέντρου	333bytes
Πλήθος τμημάτων δεδομένων	3
Πλήθος τμημάτων ελέγχου	6
Πλήθος ζητούμενων τμημάτων ελέγχου	4

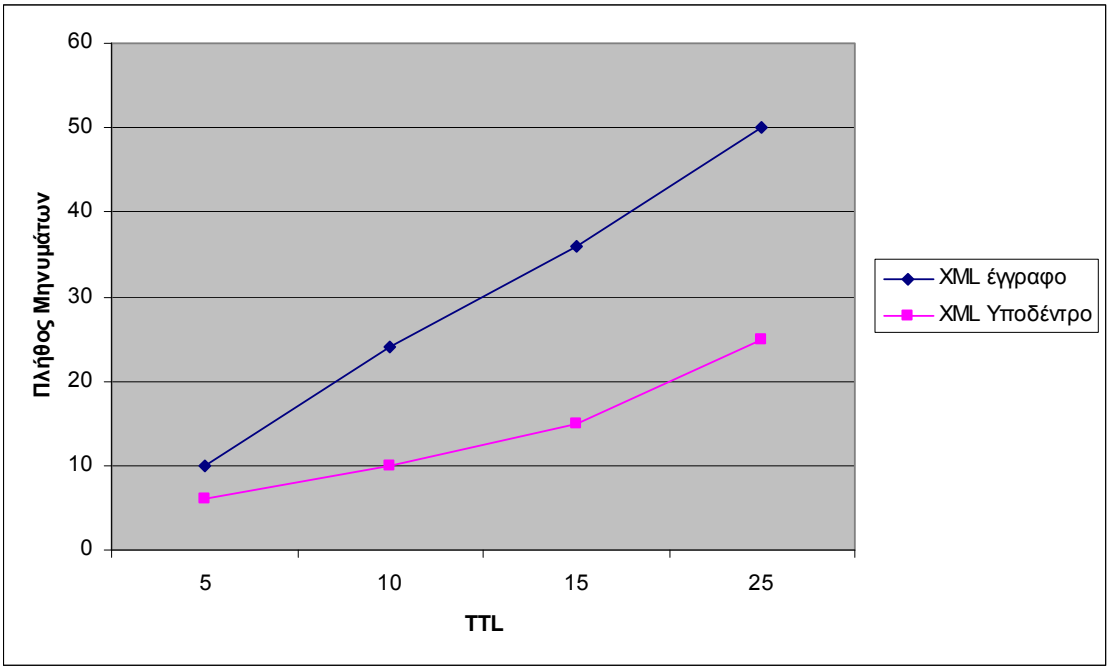
Αρχικά εξετάζουμε τα ποσοστά επιτυχίας αναζήτησης του δεδομένου note3 για τις δύο περιπτώσεις, με βάση διαφορετικού αριθμού TTL, δηλαδή για διαφορετικά βάθη αναζήτησης στο δίκτυο. Παίρνουμε τα αποτελέσματα του Σχήματος 5.14.



Σχήμα 5.14 Ποσοστό επιτυχίας για διαφορετικά TTL του υποδέντρου <Δ4>

Από το Σχήμα 5.14 προκύπτει ότι σε κάθε TTL, η κατάτμηση και κωδικοποίηση ενός υποδέντρου του εγγράφου αποδίδει πολύ καλύτερα από το ενδεχόμενο μη κατάτμησής του. Αυτό είναι προφανές διότι όταν έχουμε τον ίδιο χώρο στη διάθεσή μας, τα τμήματα που μπορούμε να αποθηκεύσουμε είναι τριπλάσια από την περίπτωση όπου δεν έχουμε κατάτμηση. Η διαθεσιμότητα του συγκεκριμένου υποσυνόλου αυξάνεται σημαντικά, κάτι που δείχνει ότι είναι επικερδές να έχουμε αυτόνομο ένα υποδέντρο το οποίο περιέχει πληροφορία που ζητείται συχνά. Η πρόταση αυτή ενισχύεται και από το γεγονός ότι στην πρώτη περίπτωση πρέπει να ανακτήσουμε 4 τμήματα μεγέθους 1KB το καθένα (συνολικά 4KB), ενώ στη δεύτερη περίπτωση χρειαζόμαστε 4 τμήματα μεγέθους 333 bytes το καθένα (συνολικά 1,32KB). Βλέπουμε , λοιπόν, ότι χωρίς κατάτμηση, αναγκαζόμαστε να λάβουμε περιττή πληροφορία, επιβαρύνοντας το δίκτυο για μεταφορά δεδομένων που, στην ουσία, δε χρειαζόμαστε. Με την κατάτμηση, αντιθέτως, μπορούμε να ανακτήσουμε μόνο τη χρήσιμη πληροφορία, μειώνοντας το φόρτο στο δίκτυο.

Επίσης, εξετάσαμε τις δύο περιπτώσεις ως προς το κόστος μιας επιτυχούς αναζήτησης σε πλήθος απαιτούμενων βημάτων. Τα αποτελέσματα φαίνονται στο Σχήμα 5.15.



Σχήμα 5.15 Κόστος αναζήτησης σε πλήθος μηνύματων με κατάτμηση και χωρίς κατάτμηση

Από το Σχήμα 5.15 προκύπτει ότι το κόστος αναζήτησης σε πλήθος μηνυμάτων είναι σαφώς μικρότερο στην περίπτωση όπου έχουμε κατάτμηση του εγγράφου σε υποδέντρο. Μάλιστα, για μεγάλα TTL το κέρδος αγγίζει το 50%. Αυτό ισχύει διότι με την κατάτμηση, τα τμήματα ελέγχου που είναι αποθηκευμένα είναι περισσότερα και συνεπώς έχουμε μεγαλύτερη διαθεσιμότητα του δεδομένου όταν είναι αυτόνομο, παρά όταν αποτελεί υποσύνολο του αρχικού εγγράφου. Άρα, η πιθανότητα να βρεθούν χρήσιμα τμήματα σε λιγότερα βήματα είναι πολύ μεγαλύτερη από το να ψάχνουμε για λιγότερα τμήματα στο ίδιο δίκτυο.

Επιπλέον, παρατηρούμε ότι αυξάνοντας το TTL αυξάνεται και το συνολικό πλήθος των βημάτων και στις δύο περιπτώσεις. Αυτό δεν πρέπει να μας εκπλήσσει, διότι για μικρά TTL, τα ποσοστά επιτυχίας είναι μικρότερα από ότι για μεγαλύτερα TTL. Στα μικρά TTL, τα τμήματα θα πρέπει να είναι αποθηκευμένα σε σχετικά κοντινούς κόμβους, ώστε να είναι επιτυχής η αναζήτηση. Στα μεγάλα TTL, μπορεί το συνολικό πλήθος των βημάτων να είναι μεγαλύτερο, ωστόσο αντίστοιχα μεγάλο είναι και το ποσοστό να επιτύχει η αναζήτηση, καθώς δίνεται το δικαίωμα να αναζητηθούν περισσότεροι κόμβοι για τα ζητούμενα τμήματα.

5.4 Γενικά Συμπεράσματα

Τα πειράματα που πραγματοποιήθηκαν παραπάνω μας έδωσαν χρήσιμα συμπεράσματα για τη μέθοδο της Κωδικοποίησης Απαλοιφής ως μέσον για τη διατήρηση αντιγράφων αρχείων και τους LT Κώδικες Απαλοιφής ειδικότερα. Τα συμπεράσματα αυτά αφορούν αρχικά στην ίδια τη μέθοδο και συγκεκριμένα στους αλγορίθμους κωδικοποίησης και αποκωδικοποίησης. Παρατηρούμε ότι λαμβάνοντας το ίδιο πλήθος τμήματα ελέγχου όσα και τα τμήματα δεδομένων στα οποία χωρίζεται το έγγραφο, παίρνουμε πολύ καλά ποσοστά επιτυχούς αποκωδικοποίησης. Ωστόσο, όπως εξηγήσαμε στο Κεφάλαιο 2, οι LT κώδικες βασίζονται σε πιθανοτικούς αλγορίθμους για την κωδικοποίηση και αποκωδικοποίηση, Για το λόγο αυτό ενδέχεται να χρειάζονται λίγο παραπάνω τμήματα ελέγχου, ώστε να διασφαλίσουμε με πολύ μεγάλη πιθανότητα ότι το έγγραφο αποκωδικοποιείται με επιτυχία. Το επιπλέον πλήθος εκτιμάται στα $O(\sqrt{k})$ τμήματα (όπου k τα τμήματα δεδομένων), δηλαδή για

τρία τμήματα δεδομένων, λαμβάνοντας πέντε τμήματα ελέγχου πετυχαίνουμε επιτυχή αποκωδικοποίηση με ποσοστό πάνω από 90 %.

Επίσης, όσον αφορά την ίδια τη μέθοδο, παρατηρούμε ότι οι χρόνοι κωδικοποίησης και αποκωδικοποίησης είναι αρκετά χαμηλοί, κάτι που ωθεί στην καλύτερη συμπεριφορα των LT Κωδικών. Αυτό οφείλεται στο γεγονός ότι οι βασικές πράξεις που χρησιμοποιούνται τόσο στην κωδικοποίηση, όσο και στην αποκωδικοποίηση, είναι πράξεις τύπου XOR, οι οποίες δεν είναι απιτητικές πράξεις και ολοκληρώνονται πολύ γρήγορα. Αυτό είναι ένα πλεονέκτημα της μεθόδου των LT Κωδικών έναντι άλλων μεθόδων Κωδικοποίησης Απαλοιφής, όπως η Reed Solomon μέθοδος. Η μέθοδος αυτή χρησιμοποιεί πράξεις πολλαπλασιασμού πινάκων και αντιστροφής πινάκων, οι οποίες είναι ιδιαίτερα χρονοβόρες. Σε ένα δίκτυο ομότιμων, μάλιστα, όπου οι κόμβοι εφαρμόζουν την κωδικοποίηση και αποκωδικοποίηση, η μικρή διάρκεια των διαδικασιών αυτών βοηθά στην εν γένει καλύτερη συμπεριφορά των κόμβων.

Πέρα από τα συμπεράσματα για τη μέθοδο, χρήσιμες είναι και οι πληροφορίες που λαμβάνουμε για τη συμπεριφορά και απόδοση της μεθόδου σε ένα δίκτυο ομότιμων. Για καλύτερη κατανόηση των πλεονεκτημάτων της Κωδικοποίησης Απαλοιφής, τη συγκρίναμε με τη μέθοδο της Αντιγραφής (Replication), μιας ιδιαίτερα διαδεδομένης μεθόδου διατήρησης αντιγράφων που χρησιμοποιείται σε τέτοια δίκτυα. Παρατηρούμε, λοιπόν, ότι η διαθεσιμότητα που έχει ένα αρχείο με την Κωδικοποίηση Απαλοιφής είναι πολύ μεγαλύτερη και με μικρότερο αποθηκευτικό κόστος, από ότι η μέθοδος της Αντιγραφής. Συγκεκριμένα, με Κωδικοποίηση ενός εγγράφου τριών τμημάτων δεδομένων σε έξι τμήματα ελέγχου, με επιπρόσθετο αποθηκευτικό κόστος ίσο με δύο, η διαθεσιμότητα φτάνει στο $\binom{6}{3}$. Με το ίδιο αποθηκευτικό κόστος, Η Αντιγραφή μας δίνει διαθεσιμότητα μόλις δύο.

Επιπλέον, η Κωδικοποίηση Απαλοιφής έχει καλύτερη συμπεριφορά και κατά την αποχώρηση κόμβων από την Αντιγραφή. Για μικρά και μεγαλύτερα ποσοστά αποτυχιών, η Κωδικοποίηση Απαλοιφής μας δίνει πολύ καλύτερα ποσοστά επιτυχημένων αναζητήσεων των εγγράφων. Για αποτυχίες 25 και 50 %, τα ποσοστά επιτυχίας αγγίζουν το 90 %, ενώ η Αντιγραφή το 75 % περίπου. Μόνο σε ακραίες

περιπτώσεις, όπου έχουμε αποτυχία κόμβων πάνω από 70 %, η Αντιγραφή υπερτερεί της Κωδικοποίησης Απαλοιφής. Αυτό διότι για την Κωδικοποίηση Απαλοιφής χρειαζόμαστε αρκετούς κόμβους να είναι ταυτόχρονα διαθέσιμοι, ενώ για την Αντιγραφή αρκεί ένας ο οποίος να κρατά αντίγραφο του εγγράφου. Ακόμη, για να επαναφέρουμε το δίκτυο σε σταθερό επίπεδο λειτουργίας, όσον αφορά τη διαθεσιμότητα των αρχείων, όταν αποτυγχάνει κάποιος κόμβος, το επιπρόσθετο αποθηκευτικό κόστος είναι σαφώς μεγαλύτερο κατά την Αντιγραφή, περίπου τριπλάσιο σε σύγκριση με την Κωδικοποίηση Απαλοιφής.

Τέλος, παρατηρήσαμε ότι αν εφαρμόσουμε την αντιγραφή Κατόχου στους LT Κώδικες, επιτυγχάνουμε ακόμη μεγαλύτερη αύξηση στη διαθεσιμότητα ενός εγγράφου. Συγκεκριμένα, βλέπουμε ότι αυξάνοντας το συνολικό πλήθος των κόμβων του δικτύου, με την Αντιγραφή Κατόχου πετυχαίνουμε καλύτερα αποτελέσματα στην επιτυχή ανάκτηση ενός εγγράφου, περίπου 95 % για 100 κόμβους. Χωρίς τη μέθοδο αυτή, το ποσοστό πέφτει (περίπου 70 %). Επιπλέον, με την Αντιγραφή Κατόχου, το βάθος το δίκτυο, στο οποίο ψάχνουμε για Τμήματα Ελέγχου, είναι μικρότερο από την περίπτωση χωρίς Αντιγραφή. Αυτό βοηθά στο να μη χρειάζονται πάρα πολλά μηνύματα να φτάνουν στους κόμβους για την ανάκτηση των Τμημάτων, βελτιώνοντας το φόρτο που συσσωρεύεται στο δίκτυο.

ΚΕΦΑΛΑΙΟ 6. ΕΠΙΛΟΓΟΣ - ΜΕΛΛΟΝΤΙΚΕΣ ΠΡΟΕΚΤΑΣΕΙΣ

Στην παρούσα εργασία ασχοληθήκαμε με μία νέα μέθοδο διατήρησης αντιγράφων ενός αρχείου σε ένα Δίκτυο Ομότιμων. Η μεθοδος αυτή ονομάζεται Κωδικοποίηση Απαλοιφής. Κατά την Κωδικοποίηση Απαλοιφής ένα αρχείο χωρίζεται αρχικά σε m Τμήματα Δεδομένων και στη συνέχεια κωδικοποιείται σε n (n>m) νέα τμήματα, τα Τμήματα Ελέγχου. Το βασικό χαρακτηριστικό της μεθόδου αυτής είναι ότι μπορούμε να κατασκευάσουμε το αρχικό αρχείο λαμβάνοντας οποιαδήποτε m από τα n τμήματα που υπάρχουν. Αυτό είναι ιδιαίτερα κερδοφόρο σε ένα Δίκτυο Ομότιμων, καθώς μπορούμε να αποθηκεύσουμε τα Τμήματα Ελέγχου σε n διαφορετικούς κόμβους. Μόνο αν αποχωρήσουν από το δίκτυο περισσότεροι από n-m κόμβοι, το αρχείο γίνεται μη διαθέσιμο. Προφανώς, το ενδεχόμενο αυτό συγκεντρώνει μικρές πιθανότητες. Επιπλέον, η μέθοδος της Κωδικοποίησης Απαλοιφής δεν επιβαρύνει με μεγάλο επιπρόσθετο αποθηκευτικό κόστος το δίκτυο. Συγκεκριμένα, επιβαρύνεται κατά παράγοντα n-m.

Αρχικά, στο Κεφάλαιο 2, έγινε μια εισαγωγή στην έννοια της Κωδικοποίησης Απαλοιφής. Αναλύθηκε η μέθοδος, καθώς και τα πλεονεκτήματα και μειονεκτήματά της. Στη συνέχεια επικεντρωθήκαμε στη μέθοδο των Luby-Transform (LT) Κωδικών Απαλοιφής. Οι LT Κώδικες αποτελούν μια ειδική κατηγορία Κωδικών Απαλοιφής, συγκεκριμένα ανήκουν στην κατηγορία των Κωδικών Πηγής. Περιγράφηκαν οι Αλγόριθμοι Κωδικοποίησης και Αποκωδικοποίησης της μεθόδου και εκτιμήθηκε η απόδοση και συμπεριφορά τους. Επιπλέον, αναλύθηκαν και κάποια τεχνικά χαρακτηριστικά της μεθόδου, όπως η συνάρτηση κατανομής που πρέπει να χρησιμοποιηθεί ώστε να επιλεχθούν οι κατάλληλες παράμετροι (π.χ. πλήθος γειτόνων κάθε Τμήματος Δεδομένων) για την καλύτερη επίδοση των Αλγορίθμων αυτών.

Πειραματικά, η καταλληλότερη συνάρτηση κατανομής είναι η Robust Soliton Κατανομή.

Στο Κεφάλαιο 3 περιγράψαμε το Δίκτυο Ομότιμων που υλοποιήθηκε στην παρούσα εργασία (αδόμητο, μη κεντρικοποιημένο). Εισάγαμε τη μέθοδο των LT Κωδικών ως τη μέθοδο διατήρησης αντιγράφων των εγγράφων του δικτύου αυτού. Περιγράψαμε τα χαρακτηριστικά των κόμβων που συμμετέχουν στο δίκτυο και χρησιμοποιούν τη συγκεκριμένη μέθοδο. Επίσης περιγράφηκε η διαδικασία εισόδου ενός νέου έγγραφου στο δίκτυο και τα μηνύματα που αποστέλλονται για την αποθήκευση των Τμημάτων Ελέγχου στους κόμβους του. Επιπλέον, αναλύθηκε και η διαδικασία αναζήτησης και ανάκτησης ενός εγγράφου από κάποιο κόμβο που το ζητά. Στο τέλος του κεφαλαίου ασχοληθήκαμε με μια άλλη μέθοδο δημιουργίας αντιγράφων σε ένα δίκτυο Ομότιμων, η οποία χρησιμοποιείται ευρέως. Η μέθοδος αυτή ονομάζεται Αντιγραφή (Replication) και δημιουργεί αυτούσια αντίγραφα ολόκληρου του εγγράφου, τα οποία αποθηκεύονται σε διάφορους κόμβους του Δικτύου. Περιγράφηκαν επίσης διάφορες τεχνικές που χρησιμοποιούνται κατά την Αντιγραφή (π.χ. Αντιγραφή Κατόχου, Αντιγραφή Μονοπατιού).

Στο Κεφάλαιο 4 ασχοληθήκαμε με τα XML έγγραφα. Περιγράφηκε η δομή ενός τέτοιου έγγραφου, καθώς και η δενδρική απεικόνιση που μπορεί να αποκτήσει. Στη συνέχεια του Κεφαλαίου αναλύθηκε η διαδικασία κατάτμησης ενός XML δέντρου σε μικρότερα υπο-δέντρα, τα οποία μπορούν να χρησιμοποιηθούν αυτόνομα για την εξυπηρέτηση αναζητήσεων για πληροφορία που περιέχεται μόνο σε αυτά. Ακόμη, περιγράψαμε την εισαγωγή XML εγγράφων στο Δίκτυο Ομότιμων που υλοποιήσαμε και την εφαρμογή της LT Κωδικοποίησης Απαλοιφής σε τέτοια έγγραφα. Περιγράψαμε το πώς μεταβάλλονται τα χαρακτηριστικά των κόμβων και των δεδομένων που κρατούν για να διαχειρίζονται XML έγγραφα. Αναλύθηκαν τα μηνύματα και η διαδικασία εισαγωγής ενός νέου XML εγγράφου στο Δίκτυο, καθώς και η αναζήτηση και ανάκτηση ενός τέτοιου εγγράφου από κάποιο κόμβο. Επιπλέον, παρουσιάστηκαν τα διαφορετικά είδη επερωτήσεων που μπορεί να κάνει κάποιος κόμβος του Δικτύου για την ανάκτηση ενός τέτοιου εγγράφου.

Καταλήγοντας, στο Κεφάλαιο 5, πραγματοποιήθηκαν ορισμένα πειράματα στη μέθοδο της Κωδικοποίησης Απαλοιφής που χρησιμοποιήθηκε, καθώς και στη συμπεριφορά της μέσα στο Δίκτυο Ομότιμων. Επίσης, τα πειράματα αφορούσαν και σύγκριση μεταξύ της μεθόδου των LT Κωδικών με τη μέθοδο της Αντιγραφής, ώστε να συμπεράνουμε τα επιμέρους πλεονεκτήματα και μειονεκτήματα κάθε μιας από αυτές. Παρατηρήσαμε ότι η LT μέθοδος έχει μεγάλη επιτυχία στην αποκωδικοποίηση αν ληφθούν επιπρόσθετα τμήματα ελέγχου της τάξης του k+k/2 (όπου k τα τμήματα δεδομένων). Επίσης είδαμε ότι οι χρόνοι Κωδικοποίησης και Αποκωδικοποίησης είναι πολύ καλοί. Ακόμη, από τη σύγκριση με τη μέθοδο της Αντιγραφής, παρατηρήσαμε ότι οι LT Κώδικες Απαλοιφής δίνουν μεγαλύτερη διαθεσιμότητα και με μικρότερο αποθηκευτικό κόστος από την Αντιγραφή. Επιπλέον, λειτουργούν καλύτερα σε αποχωρήσεις κόμβων, επαναφέροντας το δίκτυο στην κανονική του κατάσταση με πολύ μικρότερο επιπρόσθετο αποθηκευτικό κόστος. Η Αντιγραφή αποδίδει καλύτερα στην περίπτωση όπου έχουμε πολύ μεγάλη αποτυχία κόμβων (πάνω από 70 %), διότι χρειάζεται ένας κόμβος που να διαθέτει αντίγραφο ενός εγγράφου για την επιτυχή ανάκτησή του, ενώ για τους LT Κώδικες χρειάζονται m κόμβοι.

Ως μελλοντική προέκταση στην παρούσα εργασία προτείνεται η χρήση διαφορετικής μεθόδου Κωδικοποίησης Απαλοιφής ως μέσον δημιουργίας αντιγράφων ενός εγγράφου. Μια τέτοια μέθοδος θα μπορούσαν να είναι οι Raptor Κώδικες, μια ακόμη νεότερη κατηγορία Κωδικών Πηγής. Επιπλέον, θα μπορούσαν να δημιουργηθούν οι κατάλληλες συνθήκες σε ένα Δίκτυο Ομότιμων για on the fly δημιουργία Τμημάτων Ελέγχου, σε περίπτωση που είναι αδύνατη η ανάκτηση ενός εγγράφου με τα ήδη υπάρχοντα Τμήματα Ελέγχου. Ακόμη, θα μπορούσε να ερευνηθεί η συμπεριφορά των Κωδικών Απαλοιφής σε δομημένα Δίκτυα Ομότιμων, όπως τα Δίκτυα με την παρουσία Υπερ-κόμβου (super-peer) ή με τοπολογία δακτυλίου (π.χ. CHORD). Με δεδομένη την καλή συμπεριφορά της μεθόδου, ενδείκνυται η έρευνα της απόδοσης της μεθόδου και σε άλλα δίκτυα πλην των δικτύων Ομότιμων, όπως τα συστήματα Πελάτη- Εξυπηρετητή (Client Server) ή Γεωγραφικά Συστήματα.

ΑΝΑΦΟΡΕΣ

ΑΝΑΦΟΡΕΣ

[1] Amin Shokrollahi. “LDPC Codes: An Introduction”, Digital Fountain, Inc., April 2, 2003

[2] Robert G. Gallager, “ Low-Density Parity-Check Codes”, 1963

[3] Wolfgang Brauneis and Hilmar Linder. “A New Class of Erasure Codes and its Application to Scalable Multicast Content Delivery”, Department of Scientific Computing

[4] Hakim Weatherspoon and John D. Kubiatowicz. “Erasure Coding vs. Replication: A Quantitative Comparison”, Computer Science Division University of California, Berkeley

[5] Rodrigo Rodrigues1 and Barbara Liskov. “High Availability in DHTs: Erasure Coding vs. Replication”, INESC-ID / Instituto Superior T_ecnico, Lisbon, Portugal
MIT Computer Science and Arti_cial Intelligence Laboratory, Cambridge MA, USA

[6] DAN WANG, LI PIN*y*, Xiao YU HU, *and* Xin MEI WANG, “Fast Decoding Algorithm for Low-Density Parity-Check Codes”, IEICE TRANS. COMMUN.

[7] Xiao–Yu Hu, Evangelos Eleftheriou, Dieter–Michael Arnold, and Ajay Dholakia, “Efficient Implementations of the Sum-Product Algorithm for Decoding LDPC Codes”, IBM Research, Zurich Research Laboratory, CH-8803 R¨uschlikon, Switzerland

[8] James S. Plank and Michael G. Thomason. “On the Practical Use of LDPC Erasure Codes for Distributed Storage Applications”, Department of Computer Science
University of Tennessee, September 23, 2003

[9] David J.C. MacKay. “FOUNTAIN CODES”, Cavendish Laboratory, University of Cambridge

[10] M. Luby, “LT-codes,” in Proceedings of the ACM Symposium on Foundations of Computer Science (FOCS), 2002.

[11] Chris Harrelson, Lawrence Ip, Wei Wang. “Limited Randomness LT Codes”, October 2002

[12] Frank Uyeda, Huaxia Xia, Andrew A. Chien, “
Evaluation of a High Performance Erasure Code Implementation

[13] Esa Hyytia, Tuomas Tirronen, Jorma Virtamo. “Optimizing the Degree Distribution of LT Codes with an Importance Sampling Approach”, June 2006

[14] Byers, J., Luby, M., Mitzenmacher, M., and Rege, A. “A digital fountain approach to reliable distribution of bulk data”, Proc. ACM SIGCOMM’98, 2–4 September 1998

[15] Amin Shokrollahi. “Raptor Codes”, in Proceedings of the ACM Symposium on Foundations of Computer Science (FOCS), 2004

[16] Simon S. Woo and Michael K. Cheng. “Prioritized LT Codes”, California Institute of Technology, Pasadena, 2003

[17] Sam Roweis. “Erasure (Deletion) Channels & Digital Fountain Codes”, November 22, 2006

[18] Shingo Ata, Yoshihiro Gotoh, Masayuki Murata. “Replication Strategies in Peer-to-Peer Services over Power-Law Overlay Networks”, APNOMS 2003

[19] Qin Lv , Pei Cao, Edith Cohen, Kai Li , Scott Shenker. “Search and Replication in Unstructured Peer-to-Peer Networks”, International Computer Science Institute (ICSI), 2002

[20] Beverly Yang Hector, Garcia-Molina. “Designing a Super-Peer Network”, Computer Science Department, Stanford University, 2001

[21] Zheng Zhang, Qiao Lian. “Reperasure: Replication Protocol using Erasure-Code in Peer-to-Peer Storage Network]”, IEEE, 2002

[22] W. K. Lin, D. M. Chiu, Y. B. Lee, “Erasure Code Replication Revisited”. Peer-to-Peer Computing 2004: 90-97

[23] George Nychis, Argyro Andreou, Deepti Chheda, and Alexander Giamas. “Analysis of Erasure Coding in a Peer to Peer Backup System”, Information Networking Institute, 2003

[24] http://www.hispread.net

[25] http://www.pdl.cmu.edu/Pasis

ΠΑΡΑΡΤΗΜΑ

ΠΑΡΑΡΤΗΜΑ

Στο παράρτημα αυτό παραθέτουμε ορισμένα τμήματα κώδικα τα οποία υλοποιούν αλγορίθμους που περιγράψαμε σε προηγούμενα κεφάλαια.

Κλάσεις που χρησιμοποιούνται στη LT κωδικοποίηση και αποκωδικοποίηση

```
'tmhmata dedomenwn sta opoia xwrizoume to arxeio
Public Class datanode
    Public id As Integer
    Public data As String
    Public degree As Integer
    Public nb(degree)

End Class


'kwdikopoihmena tmhmata pou dhmiourgountai
Public Class checknode
    Public degree As Integer
    Public nb(degree)
    Public data As String
    Public id As Integer

End Class
```

LT κωδικοποίηση

```
'Anagnwsh arxeiou C:\file.txt
        Dim oRead As System.IO.StreamReader
        oRead = IO.File.OpenText("C:\file.xml")

        Dim intSingleChar As Integer
        Dim count As Integer 'h count ypodhlwnei to plhthos twv bits
tou arxeiou
        count = 0
```

```
        While oRead.Peek <> -1
            count = count + 1

            intSingleChar = oRead.Read()

        End While

        'o cSingleChar apouhkevei ta bits tou eggrafou
        Dim cSingleChar(count) As String


        count = 0
        oRead.Close()
        oRead = IO.File.OpenText("C:\file.xml")
        While oRead.Peek <> -1


            intSingleChar = oRead.Read()

            ' Convert the integer value into a character

            cSingleChar(count) = Chr(intSingleChar)

            count = count + 1
        End While



        'H metavlhth size2 dhlwnei to plhthos twn bits kathe
kommatiou pou epeksergazomaste
        Dim size2 As Integer
        Size2 = 50 'timh ths size2

        Dim s As String
'H metavlhth bits dhlwnei to plhthos twn tmhmatwn dedomenwn
        Dim bits, a As Integer

        bits = Math.Ceiling(count / size2)

'H metavlhth piece dhlwnei to plhthos twn tmhmatwn dedomenwn

        Dim piece, kom, b, g, h As Integer
        piece = Math.Ceiling(cSingleChar.Length / size2)


'Ypologizoume to megethos kathe kommatiou tou arxeiou. Profanws ola
ta kommatia exoun to idio megethos ektos apo to teleytaio pou
endexomenws exei mikrotero megethos
        Dim size(piece) As Integer

        For a = 1 To piece - 1
            size(a - 1) = size2

        Next
       size(piece - 1) = cSingleChar.Length - (piece - 1) * size2 - 1
        h = size(piece - 1)

        bits = size2
```

```
        Dim elem(bits) As peer
        Dim i, k, t, p As Integer
        Dim nbstr As String

        Dim checkbits, checknodes As Integer
        checkbits = Math.Ceiling((piece + sqrt(piece))) 'plithos
check pou ftiaxnoume
        checknodes = checkbits

        Dim d As String
        Dim databits As Integer

        databits = bits 'to plithos twn bits toy arxikou dedomenou


        Dim data(databits) As String
        Dim data2(piece - 1) As String


        Dim data3 As String
        data3 = Nothing
        Dim origin As String
        origin = Nothing
Line11:


        For kom = 1 To piece


            nbstr = Nothing

            For i = 0 To size(kom - 1) - 1
                data(i) = cSingleChar(i + size2 * (kom - 1))


            Next
            If kom = piece Then
                For i = size(kom - 1) To size2 - 1
                    data(i) = " "
                                    Next
                size(piece - 1) = size2
                Console.WriteLine("teleytaio :" & size(piece - 1))
            End If
            data2(kom - 1) = final
            data3 = data3 & data2(kom - 1)

        Next

        Dim sel(piece) As Integer
        For kom = 1 To piece
            Dim rand As New Random
            Dim check(checkbits) As checknode
            Dim degree As Integer
            Dim abs(piece) As Integer
            For i = 0 To piece - 1
                abs(i) = 1
            Next
```

```
            For i = 0 To checkbits - 1

                For t = 0 To piece - 1
                    If abs(t) = 0 And t = piece - 1 Then
                        checkbits = i
                        GoTo line13
                    ElseIf abs(t) = 1 Then
                        GoTo line5


                    End If
                Next
line5:
                'H epilogh vathmou kathe check node ginetai mesw ths
                'Robust Soliton Distribution
                degree = robust(0.5, 0.003, piece)

                check(i) = New checknode
                check(i).degree = degree
                ReDim Preserve check(i).nb(check(i).degree - 1)

                'kathe check node exei ena ID
                check(i).id = i
                For p = 0 To piece - 1
                    sel(p) = 1
                Next

                'Kathe check node exei kapoious geitones pou
dhlwnontai apo ton pinaka check(i).nb()

                For k = 0 To degree - 1
Line2:
                    p = rand.Next(piece)

      'Den theloume dyo check nodes vathmou 1 na exoun ton idio
geitona
                    If degree = 1 And abs(p) = 0 Then

                        GoTo Line2
                    End If

                    If sel(p) = 0 Then
                        GoTo Line2
                    End If
line12:

                    check(i).nb(k) = p
                    sel(p) = 0
                Next
                If degree = 1 Then
                    check(i).data = data2(check(i).nb(0))
                    abs(check(i).nb(0)) = 0
                Else
                    check(i).data = data2(check(i).nb(0))

'H timh kathe check node isoutai me to XOR twn data nodes me tous
opoious syndeetai
                    For k = 1 To degree - 1
                        For g = 0 To check(i).data.Length - 1
```

```
check(i).data.Insert(g,String.Concat(Asc(check(i).data.Chars(g)) Xor
Asc(data2(check(i).nb(k)).Chars(g))))
                        Next
                    Next

                End If


            Next
```

LT αποκωδικοποίηση

```
            Dim original(piece - 1) As String
            Dim pick = 0, u, j, x, y As Integer


            pick = piece + Math.Ceiling(sqrt(piece)) 'Epilogh
plhthous check nodes pou tha xrhsimopoihsoume


            For i = 0 To pick - 1
                Dim q As Long

 'Vriskoume check node vathmou 1 kai thetoume thn timh tou geitonikou
  data node ish me thn timh tou check node

                If check(i).degree = 1 Then
                    original(check(i).nb(0)) = check(i).data

                    check(i).degree = 0


                    For u = 0 To pick - 1

                        If check(u).degree <> 0 Then

                            For j = 0 To check(u).degree - 1

 'apomakrynoume to apokvdikopoihmeno data node apo ola ta check nodes
  me ta opoia syndeetai
                    If check(u).nb(j) = check(i).nb(0) Then

                       For g = 0 To check(u).data.Length - 1

check(u).data.Insert(g, String.Concat(Asc(check(u).data.Chars(g))
Xor Asc(check(i).data.Chars(g))))
                        Next


        'afou apomakrynoume to tmhma, o pinakas geitonwn (akmwn)
meiwnetai kata ena stoixeio (ta epomena stoixeia metatopizontai mia
thesh aristera
                    For q = j To UBound(check(u).nb) - 1
                    check(u).nb(q) = check(u).nb(q + 1)
                    Next q
```

```
ReDim Preserve check(u).nb(UBound(check(u).nb) - 1)

      'meiwnoume to vathmo tou check node kata 1
check(u).degree = check(u).degree - 1


               End If

            Next

       End If

     Next

     i = 0
   End If
 Next
```

Κλάση μηνύματος

```
Public Class Message
    Public type As String
    Public qid As Integer
    Public tuple As Tuple
    Public nodeorigin As Integer
    Public hops As Integer
    Public ttl As Integer
    Public seen As List(Of Integer)
    Public mes As List(Of Integer)
    Public nextnode()
    Public walkers
    Public success As Boolean = False



'h setquery thetei erwthma gia thn pleiada me id=tupler apo ton komvo
me id =origin
    Public Function setquery(ByVal tupler As String, ByVal origin As
Integer)
        tuple.id = tupler

        origin = nodeorigin
        hops = hops
        walkers = walkers
        seen.Add(origin)
    End Function


'h update prosthetei ton komvo me id=i th lista twn komvwn pou exoun
prospelastei apo to mhnyma
    Public Function update(ByVal i As Integer)
```

```
        seen.Add(i)
    End Function


    'epistrefei True otan petyxei h anazhthsh enos tmhmatos

    Public Function successful()
        success = True
        Return success
    End Function


'h forw proothei to mhnyma m ston komvo i
    Public Function forw(ByVal m As Message, ByVal i As Node)
        Dim k, l As Integer

        seen = New List(Of Integer)
        l = seen.Count

        For k = l To l + i.nei.Length - 1

            If m.seen.Contains(i.nei(k - l)) = False Then
                m.update(i.nei(k - l))
            End If

      Next

    End Function

    'h receive apothikevei thn pleiada me id=s ston komvo g

    Public Function receive(ByVal s As String, ByVal g As Node)


        Dim i As Integer

        For i = 0 To g.data.Count - 1
            If g.data(i).id = s Then
                Return True

            End If
        Next
        Return False



    End Function
End Class
```

Κλάση κόμβου που συμμετέχει στο δίκτυο

```
Public Class Node

    Public neig As Integer
    Public nodes As Integer
    Public myid As Integer
    Public nei(100) As Integer
    Public data As List(Of Tuple)
    Public mes As List(Of String)
    Public nextmes As List(Of String)
    Public queries As List(Of String)

'arxikopoihsh twn stoixeiwn tou komvou pou thn kalei
    Public Function node(ByVal id As Integer, ByVal nodes1 As
Integer, ByVal neigh As Integer)
        myid = id
        nodes = nodes1
        neig = neigh
        data = New List(Of Tuple)
        mes = New List(Of String)
        nextmes = New List(Of String)
        queries = New List(Of String)

        Dim i As Integer

        For i = 0 To neig - 1

            nei(i) = -1

        Next


    End Function


'h setdata apothikevei thn pleiada me id=s ston komvo
    Public Function setdata(ByVal s As String)
        'data = New List(Of Tuple)
        Dim tuple As Tuple
        If tuple.id = s Then
            data.Add(tuple)
                    End If

    End Function


'h exists elegxei an o komvos periexei pleiada me id=s kai epistrefei
True h False se antitheth periptwsh
    Public Function exists(ByVal s As String)
        Dim tuple As Tuple
        While tuple.id = s
            If data.Contains(tuple) Then
                Return True
            Else
                Return False
            End If
        End While
```

```
    End Function

'h addnei prosthetei ena komvo pou den exei epilegei hdh, sth lista
geitonwn tou komvou pou thn kalei
Public Function addnei(ByVal neig As Integer, ByVal neiid As Integer)


        For i = 0 To neig - 1
            '


            If nei(i) = -1 Then
                nei(i) = neiid
                Exit Function
            End If

        Next
    End Function


'h hasnei elegxei an o komvos me id=k yparxei sth lista geitonwn tou
komvou pou thn kalei

    Public Function hasnei(ByVal k As Integer)
        For i = 0 To nei.Length - 1
            If nei(i) = k Then
                Return True

            End If
        Next
        Return False
    End Function


'h receive apothikevei thn pleiada pou zhtatai apo to mhnyma mes ston
komvo pou thn kalei

    Public Function receive(ByVal mes As Message)

        data.Add(mes.tuple)
    End Function


'h send stelnei ston komvo me id=id thn pleiada me id=tuple_id
    Public Function send(ByVal id As Integer, ByVal tuple_id As
Integer)
        Dim n As Node
        Dim t As Tuple

        If t.id = tuple_id Then
            Return t
        End If
    End Function


End Class
```

Δημιουργία γειτόνων ενός εισαγόμενου κόμβου

```
'h numnodes periexei to plhthos twn komvwn sto diktyo
Dim numnodes As Integer = 100

'h neig periexei to plhthos twn geitonwn kathe komvou
        Dim neig As Integer = 6

        Dim nodes(numnodes) As Node
        Dim tuples(numnodes) As Tuple

        Dim result As Boolean


        For i = 0 To numnodes
            nodes(i) = New Node

'Arxikopoihsh
            nodes(i).node(i, numnodes, neig)


        Next

            Array.Resize(nodes(i).nei,neig)

        Dim n As Integer = 0
        For t = 0 To numnodes

'epilegoume tous geitones kathe komvou
            Console.WriteLine(" ")
            Console.WriteLine("geitones tou nnode" & t & " :")
            For i = 0 To nodes(t).nei.Length - 1

line1:
                k = 0


                Randomize()
                k = CInt(Int((numnodes * Rnd()) + 1))




                If nodes(t).nei.Contains(-1) = False Then

                    Console.WriteLine(" ")
                    Console.WriteLine("geitones tou node" & t & " :")
                    For j = 0 To nodes(t).nei.Length - 1
                        Console.Write(nodes(t).nei(j) & ", ")

                    Next
                    Console.WriteLine(" ")

                    GoTo line2
                End If

'otan enas komvos apokta ena geitona, aytomatws ginetai geitonas kai
tou allou komvou
                If nodes(t).hasnei(k) = False And t <> k Then
```

```
                    If nodes(k).nei.Contains(-1) And
nodes(t).nei.Contains(-1) Then


                        nodes(t).addnei(nodes(t).nei.Length, k)
                        nodes(k).addnei(nodes(k).nei.Length, t)
                        Console.Write(nodes(t).nei(i) & ", ")
                    Else

                        GoTo line1

                    End If


                    GoTo line1
                End If


            Next

line2:
        Next
line3:

        For i = 0 To numnodes - 1

            For z = 0 To nodes(i).nei.Length - 1
                If nodes(i).nei(z) = -1 Then

                    Array.Resize(nodes(i).nei, z - 1)
                    If z = 0 Then
                        Array.Resize(nodes(i).nei, 1)
                        Randomize()
                        k = CInt(Int((numnodes * Rnd()) + 1))
                        nodes(i).addnei(nodes(i).nei.Length, k)

                    End If
                    For e = 0 To numnodes - 1
                  If nodes(e).nei.Contains(nodes(i).myid) = True Then
                  nodes(i).addnei(nodes(i).nei.Length, nodes(e).myid)
                            Console.WriteLine(" ")
                    Console.WriteLine("geitones tou node" & i & " :")
                            For d = 0 To nodes(i).nei.Length - 1
                                Console.Write(nodes(i).nei(d) & ", ")
                            Next
                            Console.WriteLine(" ")
                        End If
                    Next
                    Exit For
                End If
                'Exit For
            Next

            If nodes(i).nei.Contains(-1) = True Then
                                GoTo line3
            End If
        Next
```

Αποθήκευση τμημάτων στο δίκτυο

```
'to m einai to mhnyma apothikeyshs
Dim m As Message
        m = New Message
        m.message()
        m.setquery("id", 0)

        Dim b As Integer = 0
        Dim ttl, f, h As Integer

'thetoume ttl apothikeyshs
        m.ttl = 10
        b = m.ttl
        f = 0


'o messenger einai o komvos pou kanei thn apothikeysh
 Dim messenger As Node
        messenger = nodes(0)
        Randomize()
        f = CInt(Int(((messenger.nei.Length - 1) * Rnd()) + 1 - 1))
        m.mes = New List(Of Integer)

        Dim piece As Integer = kom

        For v = 0 To piece - 1

            m.mes = New List(Of Integer)
            m.ttl = b
            messenger = nodes(0)
            For k = 0 To b - 1
line4:
                h = h + 1
                If m.forward(m.ttl, "id", messenger.nei(f)) = True
Then
   Console.WriteLine("PROOTHISH STON KOMVO : " & messenger.nei(f) &
", " & m.ttl)


'kathe komvos elegxei an to ttl=0

If m.ttl = 0 Then


 If nodes(messenger.nei(f)).data.Contains(tuples.id="id") = True Then

 'an o komvos exei hdh thn pleiada, ayksanei to ttl kata 1 kai
proothei to mhnyma se diko tou geitona

 m.ttl = m.ttl + 1

  f = CInt(Int(((messenger.nei.Length - 1) * Rnd()) + 1 - 1))

 GoTo line4
```

```
      Else
'alliws o komvos apothikeyei thn pleiada
nodes(messenger.nei(f)).setdata("id")


       End If

  End If

'an to ttl den einai 0, tote messenger ginetai o geitonas ston opoio
exei ftasei to mhnyma kai synexizei thn proothish tou
messenger = nodes(messenger.nei(f))

 f = f + 1

Else
          f = CInt(Int(((messenger.nei.Length - 1) * Rnd()) + 1 - 1))
                    GoTo line4

                End If
          f = CInt(Int(((messenger.nei.Length - 1) * Rnd()) + 1 - 1))
            Next
        Next
```

Αναζήτηση τμημάτων στο δίκτυο

```
f = CInt(Int(((numnodes - 1) * Rnd()) + 1))

        Dim receiver As New Node
        m = New Message
        m.message()

'thetoume query gia pleiades me sygkekrimeno id
        m.setquery("id", 0)


        ttl2 = 10
        p = 0
'h count sto telos tha prepei na isoutai me to plhthos twn tmhmatwn
pou theloume
        count = 0
        r = 0

'o receiver einai o komvos pou kanei thn anazhthsh
        receiver = nodes(arxh)
        Console.WriteLine("KSEKINA ANAZHTHSH O KOMVOS " &
receiver.myid)
        For i = 0 To piece - -1
```

```
'epilegoume ena tyxaio geitona na steiloume to mhnyma
      x = CInt(Int(((receiver.nei.Length - 1) * Rnd()) + 1 - 1))

            For k = 0 To ttl2 - 1
                Console.WriteLine("TO MHNYMA PAEI STON KOMVO " &
receiver.nei(x))

'an kapoios komvos exei zhtoumenh pleiada, th stelnei ston arxiko
komvo
                If m.receive("id", nodes(receiver.nei(x))) = True
Then

'ayksanoume thn count kathws vrethike tmhma
                    count = count + 1
                    r = r + 1
   Console.WriteLine("VRETHIKE KOMMATI STON: " & receiver.nei(x))

receiver.data.add(receiver.nei(x).send(receiver.id,"id"))


                    p = p + r
                    r = 0

nodes(receiver.nei(x)).data.Remove(nodes(receiver.nei(x)).data(0))
                    Exit For
                Else
                    r = r + 1
'alliws o komvos stelnei to mhnyma se geitona tou
                    receiver = nodes(receiver.nei(x))
           x = CInt(Int(((receiver.nei.Length - 1) * Rnd()) + 1 - 1))
                End If


            Next
        Next



     'an to count einai ayto pou theloume exoume epityxia alliws
apotyxia
      If count = piece - 1 Then
            Console.WriteLine("EPITYXHS ANAKTHSH!!!


        Else
            Console.WriteLine("APOTYXHS ANAKTHSH")

        End If
```