ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Σύστημα peer-to-peer για αποτελεσματική μεταφορά αρχείων

Ναπολέων Πάπιστας

Μεταπτυχιακή Εργασία Ειδίκευσης

Επιβλέπων Καθηγητής
Στράτος Πάσχος, Λέκτορας

Ιωάννινα, Ιούνιος 2003

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

Εὐχαριστίες

Θεωρώ χρέος μου να ευχαριστήσω ορισμένους ανθρώπους που με βοήθησαν σημαντικά στην ολοκλήρωση της εργασίας αυτής, είτε σε τεχνικά, είτε σε επιστημονικά σημεία.

Κατ' αρχάς οφείλω ένα ευχαριστώ στον κ. Στράτο Πάσχο για τη βοήθεια του, τόσο στην επιλογή της εργασίας αυτής, όσο και σε κάποια δύσκολα σημεία αυτής. Ευχαριστώ επίσης και τους κ.κ Βασίλειο Δημακόπουλο και Γεώργιο Μανή, ως μέλη της επιτροπής εξέτασης της εργασίας.

Επίσης θα ήθελα να ευχαριστήσω τους Κυριάκο Κορίτσογλου, Βασίλη Ράδη και Δημήτρη Τσάκαλη, για τις ιδέες τους σε ορισμένα ζητήματα, τη συμμετοχή τους στη διαδικασία ελέγχου της παρούσας εργασίας καθώς και για την συμπαράστασή τους την περίοδο που επιτελούσα την εργασία αυτή.

Θα ήταν παράλειψη να μην ευχαριστήσω τους γονείς μου, Δημήτριο και Μαργαρίτα για την ηθική αλλά και υλική υποστήριξη που μου παρείχαν καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Σχήματα

Κεφάλαιο 1

Εισαγωγή

1.1 Βασικές έννοιες

Τα συστήματα peer-to-peer για μεταφορά αρχείων έχουν εμφανιστεί τα τελευταία χρόνια στην κοινότητα του Internet και έχουν γνωρίσει μια τρομερή άνθηση. Συνεχώς εμφανίζονται νέα τέτοια συστήματα με όλο και πιο προηγμένες λειτουργίες και σχεδόν όλοι όσοι ασχολούνται με τους ηλεκτρονικούς υπολογιστές (επαγγελματικά ή ερασιτεχνικά) έχουν χρησιμοποιήσει κάποιο. Ενδεικτικά αναφέρουμε κάποια ονόματα, τα οποία σίγουρα ακούγονται γνωστά σε όλα τα μέλη της κοινωνίας των υπολογιστών: Napster, Audiogalaxy, Kazaa, FreeNet, eDonkey, Overnet.

Ένα σύστημα peer-to-peer ουσιαστικά επεκτείνει το γνωστό μοντέλο επικοινωνίας πελάτη-εξυπηρέτη (*client-server model*) με τέτοιο τρόπο ώστε οι πελάτες να επικοινωνούν και μεταξύ τους. Ο ρόλος του εξυπηρέτη, όπου υπάρχει, περιορίζεται κυρίως στο να γνωστοποιεί την ύπαρξη του κάθε πελάτη στους υπόλοιπους, δηλαδή να υποδεικνύει στον κάθε πελάτη ποιος ή ποιοι άλλοι πελάτες μπορούν να τον εξυπηρετήσουν. Η αντιστοιχία φαίνεται στο σχήμα 1.1.

Ας πάρουμε για παράδειγμα την περίπτωση που μας ενδιαφέρει, τη μεταφορά αρχείων. Χωρίς τη χρήση ενός συστήματος peer-to-peer, κάποιοι χρήστες που θέλουν να ανταλλάξουν μερικά αρχεία θα έπρεπε να δημιουργήσουν π.χ. έναν FTP-server ή web-server, ο καθένας να τοποθετήσει εκεί τα αρχεία που θέλει να μοιραστεί και ύστερα να κατεβάσει τα αρχεία που τοποθέτησαν οι άλλοι και τα οποία χρειάζεται.

Χρησιμοποιώντας ένα απλούστατο σύστημα peer-to-peer για μεταφορά αρχείων, τα πράγματα διευκολύνονται κατά πολύ. Αρκεί να δημιουργηθεί ένας εξυπηρέτης του συστήματος (ή να χρησιμοποιηθεί ένας ήδη υπάρχων, αν πρόκειται για γνωστό σύστημα) και ο κάθε χρήστης να εκτελέσει το πρόγραμμα-πελάτη του συστήματος. Η μεταφορά των αρχείων θα γίνεται από πελάτη σε πελάτη με απ' ευθείας σύνδεσή

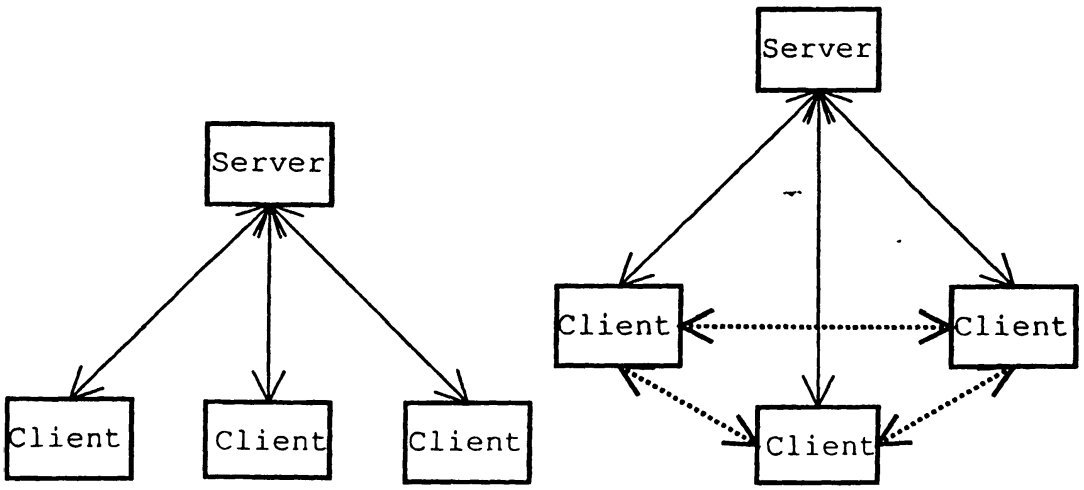
τους (peer-to-peer) και ο φόρτος του εξυπηρέτη θα ελαττωθεί σημαντικά.

1.2 Χαρακτηριστικά των συστημάτων peer-to-peer

Αν θελήσουμε να μελετήσουμε ένα σύστημα peer-to-peer πιο στενά, θα παρατηρήσουμε ότι λειτουργεί ένα επίπεδο πάνω από το Internet, δηλαδή οι κόμβοι[1] που συμμετέχουν στο σύστημα πρέπει να είναι ήδη συνδεδεμένοι στο διαδίκτυο. Δημιουργείται δηλαδή ένα "ιδεατό δίκτυο" (*virtual network*) ή "δίκτυο επικάλυψης" (*overlay network*) στο οποίο συμμετέχουν οι κόμβοι του συστήματος, το οποίο εμείς θα ονομάζουμε "δίκτυο του συστήματος" ή απλά "δίκτυο".

Προφανώς, αφού οι κόμβοι ενός τέτοιου δικτύου είναι συνδεδεμένοι στο διαδίκτυο, θα χρησιμοποιούν για την επικοινωνία τους τη γνωστή οικογένεια πρωτοκόλλων TCP/IP. Τα συστήματα peer-to-peer χρησιμοποιούν αυτά τα πρωτόκολλα για την επικοινωνία μεταξύ των κόμβων τους. Συνήθως, οι ερωτήσεις προς τον εξυπηρέτη ή προς άλλους πελάτες χρησιμοποιούν το πρωτόκολλο UDP ενώ η μεταφορά των αρχείων γίνεται με TCP. Γενικότερα, όπου υπάρχει η έννοια της "σύνδεσης", π.χ. συνδέομαι σε έναν εξυπηρέτη ή συνδέομαι σε έναν πελάτη για να κατεβάσω ένα αρχείο, χρησιμοποιείται συνήθως το TCP, ενώ όταν χρειάζεται να αποσταλλεί ένα πακέτο ενημέρωσης ή ερώτησης μόνο του, χρησιμοποιείται το UDP.

[1] πελάτες και εξυπηρέτες



(a) Μοντέλο πελάτη-εξυπηρέτη (b) Σύστημα peer-to-peer

Σχήμα 1.1: Αντιστοιχία μοντέλου πελάτη-εξυπηρέτη και συστήματος peer-to-peer

Ένα άλλο χαρακτηριστικό που κάνει τα συστήματα peer-to-peer να ξεχωρίζουν είναι το γεγονός ότι στο δίκτυο ενός τέτοιου συστήματος μπορούν να συνδέονται και να αποσυνδέονται κόμβοι ενώ το δίκτυο λειτουργεί (*on the fly*). Όταν ένας κόμβος συνδέεται, μπορεί να υπάρχουν ήδη άλλοι κόμβοι που συμμετέχουν στο δίκτυο, πολλές μάλιστα φορές *επιβάλλεται* να υπάρχουν άλλοι κόμβοι στο δίκτυο για να βοηθήσουν τον νέο να συνδεθεί. Η σύνδεση του νέου κόμβου γίνεται χωρίς να χρειαστεί να σταματήσει η λειτουργία του δικτύου και χωρίς να επηρεάζονται άμεσα οι υπόλοιποι κόμβοι. Ομοίως και κατά την αποσύδεση ενός κόμβου, ομαλή[2] ή όχι[3], το δίκτυο συνεχίζει να λειτουργεί κανονικά. Οι μόνοι κόμβοι που επηρεάζονται είναι αυτοί οι οποίοι κατέβαζαν κάποιο αρχείο από τον κόμβο που αποσυνδέθηκε, αλλά και πάλι ανάλογα με το σύστημα μπορεί να συνεχίσουν το κατέβασμα από άλλον κόμβο.

Αυτό είναι δυνατόν σε πολλά από τα σημερινά συστήματα peer-to-peer για μεταφορά αρχείων, δηλαδή να συνεχίζεται το κατέβασμα ενός αρχείου που διεκόπη για διάφορους λόγους. Πολλά συστήματα χωρίζουν κάθε αρχείο σε κομμάτια συγκεκριμένου μεγέθους (*chunks*) και μπορούν να κατεβάζουν κομμάτια του ίδιου αρχείου από διαφορετικούς χρήστες. Έτσι, ελαχιστοποιείται ο κίνδυνος να μην μπορούμε να κατεβάσουμε ένα αρχείο όταν αποσυνδεθεί ένας χρήστης. Το κομμάτι που κατεβάζαμε από αυτον, μπορεί να το έχουν κι άλλοι χρήστες και να το ζητήσουμε απ' αυτούς. Για να είναι αυτό δυνατό, πρέπει κάθε αρχείο να χαρακτηρίζεται με μοναδικό τρόπο, έτσι ώστε να μπορούμε να διαπιστώσουμε αν όντως πολλοί χρήστες έχουν το ίδιο αρχείο. Συνήθως για το σκοπό αυτό χρησιμοποιούνται οι hash-functions md4 και md5, που περιγράφονται στο παράρτημα Α. Αλλά ακόμα και στην περίπτωση που δεν έχει κανένας άλλος το συγκεκριμένο κομμάτι, σχεδόν σε όλα τα σημερινά συστήματα, κρατούνται τα δεδομένα (π.χ. διεύθυνση IP, όνομα χρήστη, κ.ά.) του κόμβου που το είχε και όταν αυτός ξανασυνδεθεί θα του το ζητήσουν άμεσα, χωρίς μεσολάβηση του χρήστη.

Τέλος, πολλά από τα σημερινά συστήματα peer-to-peer, χρησιμοποιούν τις απ' ευθείας συνδέσεις μεταξύ των πελατών τους και για διαφορετικές λειτουργίες εκτός από την μεταφορά αρχείων. Για παράδειγμα, κάποια συστήματα υποστηρίζουν και συνομιλία πραγματικού χρόνου (*chatting*) ή ανταλλαγή μηνυμάτων (*messaging*) μεταξύ των χρηστών τους, ενώ κάποια άλλα προσφέρουν τη δυνατότητα για προσπέλαση των αρχείων άλλων χρηστών (*browsing*).

[2] ο χρήστης κλείνει το πρόγραμμα ή τον υπολογιστή του

[3] αστοχία του δικτύου ή του υπολογιστή

1.3 Σκοπός της εργασίας

Στα πλαίσια της εργασίας αυτής σχεδιάστηκε ένα σύστημα peer-to-peer για ανταλλαγή αρχείων. Καθορίστηκαν οι λειτουργίες που πρέπει να επιτελεί το σύστημα έτσι ώστε η μεταφορά των αρχείων να είναι αποδοτική. Ορίστηκαν οι απαραίτητες δομές που θα χρειαστούν για τη σωστή λειτουργία του αλλά και θα διευκολύνουν την υλοποίησή του και τέλος καθορίστηκε και σχεδιάστηκε το πρωτόκολλο επικοινωνίας μεταξύ των κόμβων που συμμετέχουν στο σύστημα, δηλαδή οι ενέργειες που γίνονται από κάθε κόμβο και τα πακέτα που ανταλλάσσονται έτσι ώστε να εκτελούνται σωστά και αποδοτικά οι λειτουργίες που ορίσαμε.

1.4 Δομή της εργασίας

Αφού ορίσαμε τι είναι ένα σύστημα peer-to-peer και αναφέραμε τα βασικά χαρακτηριστικά του, στο επόμενο κεφάλαιο (κεφ. 2) θα μελετήσουμε μερικά από τα πιο διαδεδομένα σήμερα συστήματα peer-to-peer για ανταλλαγή αρχείων. Στο κεφάλαιο 3 θα κάνουμε μία γενική περιγραφή του συστήματος που σχεδιάσαμε και των λειτουργιών του, ενώ στο κεφάλαιο 4 θα δούμε κάποιες απαραίτητες δομές και αρχεία που θα χρειαστούν στο σύστημά μας. Στο κεφάλαιο 5 θα περιγράψουμε λεπτομερώς το πρωτόκολλο επικοινωνίας μεταξύ των κόμβων, θα δούμε δηλαδή τα πακέτα που ανταλλάσσονται και τις ενέργειες που κάνει κάθε κόμβος όταν λάβει κάποιο πακέτο. Στο κεφάλαιο 6 θα παρουσιάσουμε κάποια προβλήματα που προέκυψαν κατά τη διάρκεια του σχεδιασμού και θα θέσουμε τις βάσεις για μελλοντική δουλειά. Τέλος, στο παράρτημα Α περιγράφονται οι hash-functions md4 και md5, που χρησιμοποιούνται αρκετά στα συστήματα peer-to-peer για ανταλλαγή αρχείων.

Κεφάλαιο 2

Γνωστά συστήματα peer-to-peer

2.1 Napster

Το Napster[5] ήταν το πρώτο διαδεδομένο σύστημα peer-to-peer για μεταφορά αρχείων και ουσιαστικά πυροδότησε την μεγάλη ανάπτυξη των συστημάτων αυτών. Εμφανίστηκε στο διαδίκτυο γύρω στα τέλη της δεκαετίας του '90 και απέκτησε αμέσως τεράστια δημοτικότητα. Επέτρεπε στους χρήστες του να μοιράζονται μουσικά αρχεία mp3 και ξεσήκωσε θύελλα διαμαρτυριών σχετικά με τα πνευματικά δικαιώματα (*copyright*) των αρχείων αυτών. Ο δημιουργός του Shawn Fenning δέχτηκε μηνύσεις, έγιναν κάποιες δίκες και τελικά το 2000 το Napster αγοράστηκε από πολυεθνική εταιρεία και έγινε συνδρομητικό, θέτοντας έτσι τέλος στην πολυετή κυριαρχία του στο χώρο των συστημάτων peer-to-peer.

Περνώντας σε πιο λεπτομερή περιγραφή του Napster, πρέπει να τονίσουμε ότι χρησιμοποιούσε πολλούς εξυπηρέτες στους οποίους συνδεόταν οι χρήστες, αλλά *μόνο* σε έναν κάθε φορά. Πιο συγκεκριμένα, καθένας μπορούσε να κατεβάσει το πρόγραμμα-εξυπηρέτη για διάφορα λειτουργικά συστήματα και να το τρέξει, δημιουργώντας έτσι έναν νέο εξυπηρέτη. Επίσης, προγράμματα-πελάτες για το Napster είχαν δημιουργηθεί πάρα πολλά, για κάθε λειτουργικό σύστημα, με γραφικό περιβάλλον ή χωρίς.

Η λειτουργία του Napster ήταν σχετικά απλή. Αρχικά, κάθε κόμβος συνδεόταν σε έναν εξυπηρέτη, στον οποίο έστελνε μια λίστα με όλα τα αρχεία που ήθελε να μοιραστεί. Ο εξυπηρέτης, δηλαδή, λαμβάνοντας μια τέτοια λίστα από κάθε πελάτη που συνδεόταν σ' αυτόν, είχε μια τεράστια λίστα με αρχεία και τους κατόχους τους. Όταν ένας κόμβος ήθελε να βρει ένα αρχείο, έστελνε στον εξυπηρέτη την αναζήτησή του και αυτός, αφού έψαχνε στη λίστα του, του επέστρεφε ποιοι κόμβοι είχαν αρχεία που ταιριάζουν με τη συνθήκη αναζήτησης. Ο χρήστης τότε επέλεγε ποιο αρχείο

ήθελε να κατεβάσει και από ποιον και επικοινωνούσε απ' ευθείας (peer-to-peer) με τον κόμβο αυτόν.

Από άποψη τεχνικών χαρακτηριστικών και λειτουργικότητας, το Napster μπορεί να θεωρηθεί πρωτόγονο σε σχέση με τα σημερινά εξελιγμένα συστήματα peer-to-peer. Ο κάθε χρήστης μπορούσε να κατεβάσει ένα αρχείο *μόνο* από έναν άλλο χρήστη. Δεν μπορούσε δηλαδή να κατεβάζει ταυτόχρονα κομμάτια του ίδιου αρχείου από πολλούς χρήστες, κάτι που συμβαίνει σχεδόν με όλα τα σημερινά συστήματα peer-to-peer. Ως συνέπεια αυτού, δεν υπήρχε ανάγκη το κάθε αρχείο να χαρακτηρίζεται με μοναδικό τρόπο[1], αφού δεν υπήρχε η ανάγκη να πιστοποιηθεί ότι δύο ή περισσότεροι χρήστες είχαν το ίδιο αρχείο.

Επίσης, το Napster δεν υποστήριζε τη συνέχιση ενός κατεβάσματος που διεκόπη ανώμαλα, από κάποια αστοχία του δικτύου ή επειδή ένας κόμβος αποσυνδέθηκε από το δίκτυο. Αν συνέβαινε αυτό, ο χρήστης έπρεπε να ξανααναζητήσει το αρχείο και να ξεκινήσει να το κατεβάζει πάλι από την αρχή. Δεν ήταν βέβαιο ότι μετά από μία αποσύνδεση (ομαλή ή όχι) ο κόμβος θα συνδεόταν ξανά στον ίδιο εξυπηρέτη οπότε υπήρχε η περίπτωση κάποιος να αποσυνδεθεί, να ξανασυνδεθεί και να μην βρίσκει το αρχείο που κατέβαζε προηγουμένως, πολύ απλά επειδή συνδέθηκε σε διαφορετικό εξυπηρέτη και κανένας άλλος κόμβος που είναι συνδεδεμένος στον ίδιο εξυπηρέτη δεν το έχει.

Παρ' όλα τα μειονεκτήματα που είχε το Napster, η μεταφορά των αρχείων ήταν σχετικά γρήγορη και αποτελεσματική. Έτσι, το Napster βοήθησε σημαντικά στην καλύτερη και στενότερη επικοινωνία της δικτυακής κοινότητας και πολλά άλλα συστήματα το μιμήθηκαν για να καταλήξουμε στον σημερινό κατακλυσμό από συστήματα peer-to-peer και να λέμε ότι μπορούμε να βρούμε εύκολα στο διαδίκτυο σχεδόν οποιοδήποτε αρχείο χρειαζόμαστε.

2.2 eDonkey 2000

Το eDonkey 2000[1], όπως λέει και το όνομά του, εμφανίστηκε το 2000 και έχει καταλήξει σήμερα να είναι ένα από τα πιο διαδεδομένα συστήματα peer-to-peer για ανταλλαγή αρχείων. Επιτρέπει την ανταλλαγή αρχείων οποιουδήποτε τύπου (μουσικά, βίντεο, προγράμματα, κ.ά.), αλλά ταυτόχρονα προσφέρει και υπηρεσίες συνομιλίας μέσω του δικτύου IRC. Για το δίκτυο του eDonkey έχουν γραφεί αρκετά προγράμματα πελάτες, με πιο γνωστό το eMule.

Στο δίκτυο του eDonkey υπάρχουν πολλοί εξυπηρέτες, γιατί, όπως και στο Napster, ο καθένας μπορεί να δημιουργήσει έναν, αρκεί να κατεβάσει και να εκτελέσει

[1] π.χ. με μία hash-function

το αντίστοιχο πρόγραμμα. Οι εξυπηρέτες αυτοί όμως, αντίθετα από το Napster, μπορούν και επικοινωνούν μεταξύ τους, ανταλλάσσοντας πακέτα UDP για να ενημερώσει ο καθένας τους υπόλοιπους ότι λειτουργεί. Κάθε πελάτης συνδέεται με TCP σύνδεση σε έναν εξυπηρέτη και τον θεωρεί ως "βασικό εξυπηρέτη". Κατά τη σύνδεση, κάθε πελάτης στέλνει στον βασικό εξυπηρέτη του μια λίστα με τα αρχεία που μοιράζεται, έτσι ώστε κάθε εξυπηρέτης να δημιουργήσει μία μεγάλη λίστα με τα αρχεία που έχουν οι πελάτες του.

Όταν κάποιος πελάτης θέλει να αναζητήσει ένα αρχείο, στέλνει την αίτησή του στο βασικό εξυπηρέτη του, ο οποίος ελέγχει τη λίστα του και στέλνει πίσω στον πελάτη ώς απάντηση όλα τα αρχεία που ταιριάζουν με τη συνθήκη της αναζήτησης. Υπάρχει ακόμα η επιλογή ο πελάτης να στείλει την αναζήτησή του και σε άλλους εξυπηρέτες πέραν του βασικού, για να βρει περισσότερα αποτελέσματα. Αυτό γίνεται με πακέτα UDP για να ελαττωθεί ο φόρτος στους εξυπηρέτες.

Από τη στιγμή που ένας χρήστης επιλέξει να κατεβάσει ένα αρχείο το eDonkey προσπαθεί να ανακαλύψει όσους περισσότερους χρήστες μπορεί που έχουν αυτό το αρχείο. Αυτό επιτυγχάνεται ρωτώντας πρώτα τον βασικό εξυπηρέτη του για το ποιοι πελάτες του έχουν το αρχείο και έπειτα επεκτείνοντας την ερώτηση και σε άλλους εξυπηρέτες. Αφού δημιουργήσει έτσι μία λίστα με το ποιοι έχουν το αρχείο, αρχίζει και ζητάει απ' τον καθένα και ένα διαφορετικό κομμάτι του αρχείου.

Για να έχει νόημα η παραπάνω λειτουργία, πρέπει να πληρούνται δύο προϋποθέσεις: πρώτον να μπορεί ένα αρχείο να χωριστεί σε κομμάτια σταθερού μεγέθους και δεύτερον να μπορεί ένα αρχείο να χαρακτηριστεί με μοναδικό τρόπο, έτσι ώστε να μπορεί ο χρήστης να καταλάβει ποιοι *πραγματικά* το έχουν. Πραγματικά, το eDonkey χωρίζει τα αρχεία σε κομμάτια (chunks) μεγέθους 9MB, έτσι ώστε να μπορεί κάποιος να κατεβάσει κάθε κομμάτι από διαφορετικό χρήστη για να επισπεύσει τη διαδικασία κατεβάσματος ενός μεγάλου αρχείου. Στην πραγματικότητα, μπορεί να κατεβάσει από διαφορετικούς χρήστες και κομμάτια του ίδιου chunk. Όσον αφορά το μοναδικό χαρακτηρισμό του αρχείου, το eDonkey χρησιμοποιεί ως "ταυτότητα" κάθε αρχείου το μεγέθους το και του md4 του, μιας και δύο αρχεία με ακριβώς ίδιο μέγεθος είναι μάλλον απίθανο να έχουν το ίδιο md4.

Ακόμα, το eDonkey υποστηρίζει τη συνέχιση ενός κατεβάσματος που διεκόπη, είτε από τον ίδιο χρήστη[2], αν δεν έχει κανένας άλλος το αρχείο, είτε από διαφορετικούς χρήστες που το έχουν.

Τέλος, το eDonkey χρησιμοποιεί ένα σύστημα "πιστωτικών μονάδων", έτσι ώστε κάποιος που προσφέρει πολλά αρχεία στο δίκτυο να μπορεί να κατεβάζει πιο άνετα. Πιο συγκεκριμένα, όταν ένας χρήστης Α κατεβάζει ένα αρχείο από τον χρήστη Β, ο Β παίρνει κάποιες μονάδες και έτσι όταν με τη σειρά του θα επιχειρήσει να κατεβάσει

[2]όταν ξανασυνδεθεί στο δίκτυο

ένα αρχείο από τον Α θα εξυπηρετηθεί με προτεραιότητα έναντι άλλων χρηστών που δεν μοιράστηκαν αρχεία.

2.3 Kazaa

Το Kazaa[4] είναι ένα ακόμα πολύ διαδεδομένο σύστημα peer-to-peer για ανταλλαγή αρχείων. Όπως και το eDonkey, επιτρέπει στους χρήστες του να ανταλλάσσουν αρχεία κάθε τύπου. Επίσης, προσφέρει στους χρήστες του τη δυνατότητα να επικοινωνούν ανταλλάσσοντας μικρά μηνύματα, αλλά και τη δυνατότητα να βλέπουν όλα τα αρχεία που μοιράζεται κάποιος χρήστης (browsing). Τέλος, στο πρόγραμμα-πελάτη του Kazaa είναι ενσωματωμένος ένας player/viewer[3] για να μπορεί ο χρήστης να βλέπει τα αρχεία που κατεβάζει, ακόμα κι αν δεν έχουν κατεβεί ολόκληρα.

Από τεχνικής άποψης, το Kazaa το αναφέρουμε εδώ γιατί χρησιμοποιεί μια ελαφρώς διαφορετική αρχιτεκτονική από τη συνηθισμένη των συστημάτων peer-to-peer, δηλαδή το μοντέλο πελάτη-εξυπηρέτη. Οι πελάτες χωρίζονται σε δύο κατηγορίες: απλούς πελάτες και *"supernodes"*, που είναι οι πελάτες με πιο ισχυρούς υπολογιστές και καλύτερες συνδέσεις δικτύου. Κάθε πελάτης είναι εν δυνάμει supernode, εκτός αν δηλώσει μέσω μίας επιλογής στο πρόγραμμα ότι δεν το επιθυμεί.

Κατά τη σύνδεση, κάθε πελάτης ρωτάει έναν κεντρικό εξυπηρέτη ποιο είναι το κοντινότερό του supernode και συνδέεται σε αυτό με TCP σύνδεση. Κατά την αναζήτηση ενός αρχείου, ο πελάτης στέλνει την αίτησή του στο supernode που είναι συνδεδεμένος και αυτό την προωθεί αρχικά στους άλλους πελάτες που τυχόν είναι συνδεδεμένοι πάνω του και έπειτα σε άλλα, γειτονικά supernodes, και συνεχίζεται η προώθηση με τον ίδιο τρόπο. Η απάντηση στην αίτηση θα επιστρέψει από το ίδιο μονοπάτι και μετά θα γίνει η peer-to-peer σύνδεση μεταξύ των δύο (ή περισσοτέρων) πελατών για να κατεβεί το αρχείο.

Το Kazaa, όπως όλα τα σημερινά συστήματα peer-to-peer, προσφέρει και τη δυνατότητα να κατεβάσει κάποιος ένα αρχείο από πολλούς κόμβους, χομμάτι-χομμάτι, αλλά και τη δυνατότητα να συνεχιστεί ένα κατέβασμα που διεκόπη. Αν διεκόπη επειδή ο κόμβος που κατέβαζε αποσυνδέθηκε, η συνέχιση γίνεται αυτόματα την επόμενη φορά που θα συνδεθεί. Αν διεκόπη γιατί αποσυνδέθηκε ο κόμβος από τον οποίον κατεβάζε, τότε, για να βρει κάποιον άλλο που έχει το ίδιο αρχείο και να συνεχίσει το κατέβασμα από αυτόν, χρειάζεται να δώσει την εντολή ο χρήστης του προγράμματος.

Για την μοναδικότητα του αρχείου χρησιμοποιείται κι εδώ η hash-function md4 σε συνδυασμό με το μέγεθός του , ενώ δεν υπάρχει σταθερό μέγεθος χομματιού στο

[3] Ουσιαστικά, πρόκειται για ένα plug-in για το Microsoft Media Player των Windows

οποίο να χωρίζονται τα αρχεία.

Ακόμα, το Kazaa υποστηρίζει ένα απλό σύστημα πιστωτικών μονάδων. Όσο πιο πολλά αρχεία μοιράζεσαι με το σύστημα, δηλαδή όσο πιο πολλά αρχεία κατεβάζουν οι άλλοι χρήστες από σένα, τόσο ανεβαίνει το επίπεδο των μονάδων σου. Αντίθετα, όσα περισσότερα αρχεία κατεβάζεις εσύ από το σύστημα, τόσο ελαττώνονται οι μονάδες σου. Επίσης, μπορεί το κάθε αρχείο να "βαθμολογηθεί" από τους κατόχους του ως προς την ποιότητά του. Αρχεία που είναι χαρακτηρισμένα άριστα, επηρεάζουν περισσότερο τις πιστωτικές μονάδες κάθε κόμβου, θετικά ή αρνητικά. Στην πράξη πάντως, δεν παρατηρείται σημαντική διαφορά ανάμεσα σε κόμβους με υψηλό όριο μονάδων και σε κόμβους με χαμηλό όριο.

2.4 FreeNet

Το FreeNet[2][8] είναι ένα σύστημα peer-to-peer το οποίο είναι αρκετά διαφορετικό από όσα έχουμε αναφέρει μέχρι τώρα, και από άποψη λειτουργικότητας και από άποψη αρχιτεκτονικής. Αντί για την απλή ανταλλαγή αρχείων, το FreeNet λειτουργεί περισσότερο σαν ένα κατανεμημένο σύστημα αποθήκευσης, δηλαδή χρησιμοποιεί τον χώρο που του προσφέρει κάθε κόμβος στο δίσκο του για να δημιουργήσει ένα ιδεατό σύστημα αρχείων.[7]

Από άποψη αρχιτεκτονικής, το FreeNet δεν χρησιμοποιεί καθόλου εξυπηρέτες, δηλαδή όλοι οι κόμβοι του δικτύου του είναι ισότιμοι και εκτελούν τις ίδιες λειτουργίες. Για να συνδεθεί ένας κόμβος στο δίκτυο, αρκεί να ανακαλύψει τη διεύθυνση ενός ή περισσοτέρων κόμβων που είναι ήδη συνδεδεμένοι στο δίκτυο και μετά να ανταλλάξουν κάποια μηνύματα.

Στο FreeNet, σε κάθε αρχείο που συμμετέχει αντιστοιχεί ένα κλειδί, το οποίο υπολογίζεται με τη hash-function SHA-1. Επίσης, κάθε κόμβος που συμμετέχει στο δίκτυο διατηρεί μία λίστα με κόμβους που ξέρει και με τα κλειδιά των αρχείων που έχει ο καθένας. Έτσι, όταν ένας κόμβος θέλει να αποθηκεύσει ένα αρχείο στο σύστημα, θα στείλει ένα μήνυμα εισαγωγής προς τον κόμβο που κατέχει το "κοντινότερο" κλειδί με αυτό του αρχείου, για κάποια ορισμένη έννοια της απόστασης. Η διαδικασία θα συνεχιστεί με αυτόν τον τρόπο, μέχρι να βρεθεί ένας κόμβος που έχει ήδη το αρχείο, οπότε δεν θα αποθηκευτεί ξανά στο σύστημα, ή να περάσει ο "χρόνος ζωής" (*time-to-live, TTL*) που έχει το πακέτο εισαγωγής, οπότε το αρχείο θα αποθηκευθεί στον τελευταίο κόμβο που έφτασε το πακέτο, και ενδεχομένως σε κάποιους άλλους κόμβους που βρίσκονται στη διαδρομή μεταξύ του κόμβου αυτού και του κόμβου που έκανε την αίτηση εισαγωγής.

Η διαδικασία αναζήτησης είναι εντελώς όμοια, δηλαδή στέλνεται το πακέτο προς

τον κόμβο που έχει το κοντινότερο κλειδί με αυτό του αρχείου, και συνεχίζει με τον ίδιο τρόπο. Η διαφορά με άλλα συστήματα peer-to-peer χωρίς εξυπηρέτη είναι ότι τα δεδομένα του αρχείου δεν επιστρέφουν απ' ευθείας στον κόμβο που το ζήτησε αλλά το αρχείο μεταφέρεται μέσω της ίδιας διαδρομής που έκανε η αίτηση αναζήτησης. Αποθηκεύεται μάλιστα και στους ενδιάμεσους κόμβους (*caching*), έτσι ώστε μια επόμενη αναζήτηση να μπορεί να εκπληρωθεί γρηγορότερα.

Είναι φανερό ότι με αυτό τον τρόπο ενδιάμεσης αποθήκευσης, κάποτε δεν θα υπάρχει αρκετός χώρος για την εισαγωγή νέων αρχείων στο σύστημα, οπότε έπρεπε να προβλεβθεί ένας μηχανισμός απομάκρυνσης αρχείων. Έτσι, κάθε κόμβος που πρέπει να αποθηκεύσει ένα αρχείο, είτε λόγω caching είτε λόγω εισαγωγής, αν δεν έχει αρκετό χώρο στο δίσκο του για να το αποθηκεύσει θα πρέπει να διαγράψει ένα ή περισσότερα αρχεία μέχρι να δημιουργηθεί ο απαραίτητος χώρος. Τα αρχεία που θα διαγράψει θα είναι αυτά για τα οποία δέχτηκε τις λιγότερες αιτήσεις αναζήτησης, έτσι ώστε τα δημοφιλέστερα αρχεία να υπάρχουν σε περισσότερα αντίτυπα στο σύστημα.

Κατά τη δημιουργία του FreeNet, το κυρίαρχο σημείο στο οποίο δόθηκε έμφαση ήταν η ασφάλεια και η ανωνυμία. Για το λόγο αυτό, κατά τη διαδικασία εισαγωγής, αναζήτησης ή αποστολής δεδομένων, οι κόμβοι που δέχονται τα μηνύματα και τα προωθούν δεν είναι *ποτέ* σίγουροι για τον πραγματικό αποστολέα, δηλαδή τον κόμβο που έστειλε πρώτος το μήνυμα, ούτε για τον πραγματικό παραλήπτη του μηνύματος. Ακόμα και ο δεύτερος κόμβος π.χ. σε μία αναζήτηση, δηλαδή ο κόμβος που λαμβάνει το μήνυμα από τον κόμβο που εκκίνησε την αναζήτηση, δεν είναι σίγουρος για το αν ο κόμβος αυτός που του έστειλε το μήνυμα κάνει την αναζήτηση για λογαριασμό του ή απλά προωθεί μια αναζήτηση κάποιου άλλου.

Επίσης, για λόγους ασφαλείας, οι επικοινωνίες μεταξύ των κόμβων του δικτύου είναι κρυπτογραφημένες, ενώ εναθρρύνονται οι χρήστες να κρυπτογραφούν τα αρχεία προτού τα εισαγάγουν στο σύστημα, έτσι ώστε ο κόμβος στον οποίο θα αποθηκευτούν τελικά να μην γνωρίζει τι είναι αποθηκευμένο στο σκληρό του δίσκο[4].

2.5 OverNet

Το OverNet[6] είναι ένα σύστημα peer-to-peer χωρίς εξυπηρέτη για ανταλλαγή αρχείων, το οποίο είναι βασισμένο πάνω στο σύστημα "Kademlia"[9]. Κι εδώ, όπως και στο FreeNet, όλοι οι κόμβοι είναι ισοδύναμοι και εκτελούν τις ίδιες λειτουργίες.

Σχετικά με τις λειτουργίες και τις διευκολύνσεις που προσφέρει το OverNet, αυτές είναι ίδιες με του eDonkey που αναφέραμε στην ενότητα 2.2, αφού και τα δύο συστήματα είναι κατασκευασμένα από την ίδια ομάδα προγραμματιστών. Έτσι, για

[4] π.χ. για την αποφυγή νομικών προβλημάτων

το μοναδικό χαρακτηρισμό ενός αρχείου χρησιμοποιείται η hash-function md4, τα αρχεία χωρίζονται σε κομμάτια 9MB, ενώ φυσικά υπάρχει η δυνατότητα να κατεβάζει κάποιος ένα αρχείο από πολλούς χρήστες ταυτόχρονα και η δυνατότητα για συνέχιση ενός κατεβάσματος που διεκόπη.

Για τη σύνδεση ενός κόμβου στο δίκτυο του OverNet είναι απαραίτητο ο κόμβος αυτός να γνωρίζει τη διεύθυνση IP και το port ενός άλλου κόμβου που συμμετέχει ήδη στο δίκτυο. Συνήθως, τον κόμβος αυτό τον βρίσκει από κάποιο chatroom ή κάποια σελίδα στο διαδίκτυο που έχει δημιουργηθεί για το σκοπό αυτό. Αφού λοιπόν βρει τα στοιχεία του άλλου κόμβου, ανταλλάσσουν κάποια μηνύματα και ο κόμβος αυτός τον ενημερώνει για άλλους κόμβους που γνωρίζει, κατόπιν επικοινωνεί με αυτούς για να τον ενημερώσουν και για άλλους, κ.ο.κ, έτσι ώστε ο νέος κόμβος να αποκτήσει γνώση για μια "γειτονιά" του δικτύου. Η διαδικασία αυτή επαναλαμβάνεται όταν ο κόμβος αυτός χάσει όλες τις επαφές του με τους κόμβους που έμαθε, π.χ. αν αποσυνδεθούν όλοι ή αν υπάρξει κάποιο πρόβλημα στη σύνδεση.

Όσον αφορά τη διαδικασία τοποθέτησης ενός αρχείου στο σύστημα και την αναζήτηση, το OverNet μοιάζει περισσότερο με το FreeNet. Κάθε αρχείο που συμμετέχει στο σύστημα έχει ένα *μοναδικό* κλειδί μήκους 160 bit, που υπολογίζεται με τη hash-function SHA-1. Επίσης, κάθε κόμβος έχει και αυτός ένα κλειδί 160 bit που υπολογίζεται με τον ίδιο τρόπο. Έτσι, για κάθε αρχείο είναι υπεύθυνος ένας κόμβος, αυτός του οποίου το κλειδί είναι το κοντινότερο στο κλειδί του αρχείου. Για τον ορισμό της απόστασης μεταξύ δύο κλειδιών, το Kademlia χρησιμοποιεί τη λογική πράξη αποκλειστικού-ή (XOR) μεταξύ των κλειδιών. Τέλος, κάθε κόμβος διατηρεί μία λίστα με γειτονικούς του κόμβους (*k-buckets*) και τα κλειδιά τους, για τις ανάγκες της δρομολόγησης των αιτήσεων που θα λαμβάνει.

Με τον τρόπο αυτό, κατά την εισαγωγή ενός αρχείου στο σύστημα, ο κόμβος που θέλει τα το μοιραστεί με τους υπόλοιπους θα κάνει μία αναζήτηση για να βρει τους k κόμβους που έχουν τα κοντινότερα κλειδιά με αυτό του αρχείου. Το k είναι παράμετρος του πρωτοκόλλου, που επιλέγεται τέτοια ώστε οποιοιδήποτε τυχαίοι k κόμβοι να έχουν μηδαμινή πιθανότητα να αποσυνδεθούν μέσα σε μία ώρα, για παράδειγμα k=*20*. Η αναζήτηση αυτή γίνεται μέσω των γειτόνων του που έχουν τα κοντινότερα κλειδιά με του αρχείου, αυτοί την προωθούν στους δικούς τους γείτονες, και ούτω καθ' εξής μέχρι να βρεθούν k κόμβοι. Έπειτα, ο αρχικός κόμβος θα στείλει μία αίτηση εισαγωγής στους k αυτούς κόμβους για να αποθηκεύσουν το αρχείο.

Η αναζήτηση ενός αρχείου γίνεται με τον ίδιο ακριβώς τρόπο, δηλαδή ο κόμβος που αναζητά ένα αρχείο θα αναζητήσει πρώτα τους k "κοντινότερους" κόμβους με το αρχείο, με τον τρόπο που αναφέραμε. Αν κάποιος απ' αυτούς όντως έχει το αρχείο, ή καλύτερα το κομμάτι του αρχείου που ζητάει ο αρχικός κόμβος, θα του το στείλει απ' ευθείας (peer-to-peer).

2.6 Gnutella

Το Gnutella[3] είναι ένα πρωτόκολλο για κατανεμημένες αναζητήσεις αρχείων, πάνω στο οποίο βασίζονται πολλά από τα σημερινά συστήματα peer-to-peer για ανταλλαγή αρχείων, όπως τα BearShare, Morpheus, LimeWare κ.ά. Πρωτοεμφανίστηκε στις αρχές του 2000 και διαδόθηκε πολύ γρήγορα διότι ήταν με τη μορφή λογισμικού ελεύθερου κώδικα (*open source software*) και ο καθένας μπορούσε να επέμβει και να προσθέσει τα χαρακτηριστικά που επιθυμούσε.

Από τεχνικης άποψης, το δίκτυο του Gnutella δεν έχει εξυπηρέτες. Όλοι οι κόμβοι λειτουργούν και σαν πελάτες και σαν εξυπηρέτες ταυτόχρονα και λέγονται "*servents*". Για τη σύνδεση στο δίκτυο του Gnutella ένας κόμβος χρειάζεται να ξέρει έναν άλλο κόμβο που είναι ήδη συνδεδεμένος. Αυτό συνήθως γίνεται με βάση μία λίστα με γνωστούς κόμβους που διατηρεί κάθε κόμβος και που την ενημερώνει συνεχώς μέσω της επικοινωνίας με άλλους κόμβους.

Οι αιτήσεις στο Gnutella είναι δύο ειδών: αιτήσεις για αναζήτηση κάποιου αρχείου βασισμένη στο όνομά του και αιτήσεις για να ανακαλυφθούν άλλοι κόμβοι του δικτύου. Οι αιτήσεις μεταδίδονται μέσω των γειτονικών κόμβων, όπως γίνεται και σε άλλα συστήματα χωρίς εξυπηρέτη που είδαμε. Η διαφορά είναι ότι οι αιτήσεις στέλνονται ταυτόχρονα προς όλους τους γείτονες ενός κόμβου (και όχι μόνο σε κάποιους που βασίζονται σε μία έννοια απόστασης, όπως στο OverNet και το FreeNet), οι οποίοι τις προωθούν σε όλους τους δικούς τους γείτονες και ούτω καθ' εξής. Η διαδικασία αυτή λέγεται "πλημμύρα" (*flooding*). Η προώθηση των αιτήσεων γίνεται μέχρι να εξαντληθεί ο χρόνος ζωής τους, δηλαδή μέχρι να γίνει ένας ορισμένος αριθμός βημάτων (συνήθως 7).

Όταν ένας κόμβος δεχθεί μία αίτηση για αναζήτηση και έχει κάποιο ή κάποια αρχεία που ταιριάζουν, θα πρέπει να απαντήσει στην αναζήτηση. Η απάντηση αυτή αποστέλλεται στον αρχικό κόμβο που έκανε την αναζήτηση μέσα από το ίδιο μονοπάτι που έφτασε η αναζήτηση στον κόμβο, δηλαδή θα περάσει από τους ίδιους κόμβους με αντίστροφη φορά.

Όταν ένας κόμβος δεχθεί μία αίτηση για κόμβους, τότε αν θέλει θα απαντήσει στέλνοντας πίσω πληροφορίες για τον εαυτό του ή για άλλους κόμβους που γνωρίζει. Μπορεί να στείλει και περισσότερες από μία απαντήσεις, οι οποίες όμως θα σταλούν πίσω στον κόμβο που έκανε την αίτηση μέσα από το ίδιο μονοπάτι, όπως και στην προηγούμενη παρίπτωση. Οι αιτήσεις αυτές έχουν νόημα για να κατασκευάσει κάθε κόμβος μία λίστα με άλλους κόμβους στους οποίους αφ' ενός θα στέλνει αναζητήσεις και αφ' ετέρου μπορεί να τους χρησιμοποιήσει για σύνδεση κάποια επόμενη φορά.

Τέλος, η μεταφορά ενός αρχείου γίνεται με τη βοήθεια του πρωτοκόλλου HTTP, από τον κόμβο που το έχει κατ' ευθείαν στον κόμβο που το ζήτησε (peer-to-peer).

2.7 Αλλα πρωτόκολλα

2.7.1 Chord

Το Chord[11] είναι ένα πρωτόκολλο για να αντιστοιχεί ένα κλειδί σε έναν κόμβο και μπορεί να αποδειχθεί πολύ χρήσιμο σε ένα σύστημα peer-to-peer. Αν σκεφτούμε να αντιστοιχίσουμε ένα κλειδί σε κάθε αρχείο μπορούμε να χρησιμοποιήσουμε ένα πρωτόκολλο σαν το Chord για να βρούμε αρχικά σε ποιον κόμβο θα αποθηκευτεί το αρχείο και στη συνέχεια να το αναζητήσουμε.

Το Chord αναθέτει στα δεδομένα[5] και στους κόμβους που συμμετέχουν στο δίκτυο από ένα κλειδί μήκους m bits, το οποίο προκύπτει από μια hash-function, όπως η SHA-1. Το μήκος m πρέπει να είναι αρκετά μεγάλο έτσι ώστε η πιθανότητα δύο δεδομένα ή κόμβοι να έχουν το ίδιο κλειδί να είναι αμελητέα. Στη συνέχεια, τα κλειδιά ταξινομούνται σε έναν κύκλο *modulo* 2^m και κάθε δεδομένο αποθηκεύεται στον κόμβο του οποίου το κλειδί είναι στην ίδια ή σε επόμενη θέση στον κύκλο από το κλειδί του δεδομένου.

Με τη λογική αυτή, όταν ένας νέος κόμβος προστεθεί στο δίκτυο, κάποια δεδομένα που είχαν αντιστοιχηθεί στον επόμενο κόμβο στον κύκλο θα πρέπει να ανατεθούν στον νέο κόμβο. Ομοίως, αν ένας κόμβος αποχωρήσει, τα δεδομένα που διατηρούσε θα πρέπει να μεταφερθούν στον επόμενο κόμβο στον κύκλο. Με μεγάλη πιθανότητα, όταν ο εισέλθει ή αποχωρήσει ο N-οστός κόμβος του δικτύου, μόνο ένα τμήμα μεγέθους $O(1/N)$ των δεδομένων θα χρειαστεί να μετακινηθεί.

Για να μπορεί να γίνει μια αναζήτηση στο Chord, ο κάθε κόμβος θα πρέπει να ξέρει μόνο τον επόμενό του στον κύκλο. Έτσι, όταν ένας κόμβος θελήσει ένα δεδομένο θα στείλει στον επόμενό του μία αίτηση με το κλειδί του δεδομένου, αυτός θα το στείλει στον δικό του επόμενο, μέχρι να φτάσει η αίτηση σε έναν κόμβο με κλειδί μεγαλύτερο του δεδομένου. Ο κόμβος αυτός τότε είναι ο κάτοχος του δεδομένου. Για την βελτιστοποίηση της αναζήτησης, οι κόμβοι του Chord διατηρούν και κάποιες επιπλέον πληροφορίες σε έναν πίνακα δρομολόγησης.

2.7.2 CAN

Το CAN (*Content-Addressable Network*)[10] είναι επίσης ένα πρωτόκολλο για αντιστοίχιση κλειδιών, δηλαδή δεδομένων, σε κόμβους. Το CAN χρησιμοποιεί ένα χώρο καρτεσιανών συντεταγμένων d-διαστάσεων σε ένα d-δακτύλιο (*d-torus*). Κάθε στιγμή ο χώρος των συντεταγμένων μοιράζεται δυναμικά ολόκληρος στους κόμβους, έτσι ώστε κάθε κόμβος να έχει τη δική του ανεξάρτητη περιοχή και να μη

[5] στην περίπτωσή μας αρχεία ή κομμάτια αρχείων

μένει κάποια περιοχή του χώρου ακάλυπτη.

Κάθε δεδομένο αντιστοιχίζεται σε ένα κλειδί και με τη σειρά του κάθε κλειδί αντιστοιχίζεται σε ένα σημείο του d-διάστατου χώρου με τη βοήθεια μιας hash-function. Έτσι, το δεδομένο αποθηκεύεται στον κόμβο στην περιοχή του οποίου ανήκει το σημείο που αντιστοιχεί στο κλειδί του. Ομοίως, για την αναζήτηση ενός δεδομένου, αρκεί να βρούμε τον κόμβο που έχει το σημείο του δεδομένου στην περιοχή του. Έτσι, όλες οι λειτουργίες ανάγονται σε εύρεση ενός σημείου στο d-διάστατο χώρο.

Η αποθήκευση ενός δεδομένου στο σύστημα ή η αναζήτησή του από έναν κόμβο γίνεται με αιτήσεις προς τους γείτονες του κόμβου αυτού, που τις προωθούν στους δικούς του γείτονες κλπ. Κάθε κόμβος στο CAN διατηρεί έναν πίνακα με τους κόμβους που κατέχουν γειτονικές περιοχές με αυτόν ως προς d-1 διαστάσεις. Έτσι, κατά την προώθηση ενός μηνύματος αποθήκευσης ή αναζήτησης, το προωθεί απλά στον κόμβο που έχει συντεταγμένες πιο κοντά στις συντεταγμένες του σημείου που αντιστοιχεί το αρχείο.

Κατά τη σύνδεση ενός κόμβου, ο κόμβος αυτός επιλέγει ένα τυχαίο σημείο στο χώρο, βρίσκει τον κόμβο στην περιοχή του οποίου βρίσκεται αυτό και κατόπιν η περιοχή αυτή διαιρείται στο μέσο κατά μία διάσταση και ανατίθεται στο νέο κόμβο. Είναι προφανές ότι και όλα τα δεδομένα που τα σημεία τους βρίσκονται στη μισή αυτή περιοχή θα μεταφερθούν στο νέο κόμβο. Κατά την αποχώρηση ενός κόμβου, η περιοχή που διατηρούσε, καθώς και τα δεδομένα της ανατίθενται στον πιο γειτονικό του κόμβο.

Κεφάλαιο 3

Περιγραφή του συστήματος

3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα κάνουμε μια γενική περιγραφή του συστήματος που σχεδιάσαμε, το οποίο είναι ένα σύστημα peer-to-peer για ανταλλαγή αρχείων. Τα βασικά χαρακτηριστικά που προσπαθήσαμε να έχει το σύστημα είναι τα εξής:

- Να μην υπάρχει εξυπηρέτης, ούτε κάποιοι κόμβοι να έχουν κάποια "ειδική" λειτουργία. Όλοι οι κόμβοι να είναι ισότιμοι.
- Να μπορεί να κατεβάζει ένας κόμβος ένα αρχείο από πολλούς άλλους που το έχουν, διαφορετικό κομμάτι από τον καθένα. Γι' αυτό χρειάζεται να χαρακτηρίζεται κάθε αρχείο μοναδικά, με χρήση της **hash-function md5**, και να υπάρχει ένα σταθερό μέγεθος κομματιού, το οποίο ορίζεται στα 10MB.
- Να υπάρχει η δυνατότητα για συνέχιση ενός κατεβάσματος που διεκόπη από οποιονδήποτε λόγο, από τον ίδιο κόμβο ή από άλλους, χωρίς την επέμβαση του χρήστη.
- Ένα αρχείο να είναι διαθέσιμο για τους υπόλοιπους κόμβους ακόμα κι αν ο κόμβος που το έχει δεν το έχει ολόκληρο (π.χ. το κατεβάζει ακόμα).

Για να μπορεί το σύστημα να μας προσφέρει αυτά που απαιτούμε, θα πρέπει να εκτελεί διάφορες βασικές λειτουργίες, όπως η σύνδεση ενός κόμβου στο δίκτυο, η αναζήτηση ενός αρχείου, η διαδικασία μεταφοράς του (κατέβασμα) από διάφορους κόμβους καθώς και η αποσύνδεση ενός κόμβου από το δίκτυο. Στις παρακάτω ενότητες θα δώσουμε μια γενική περιγραφή για το πώς εκτελούνται οι λειτουργίες αυτές στο σύστημά μας.

3.2 Σύνδεση

Η διαδικασία της σύνδεσης ενός κόμβου στο δίκτυο του συστήματος μας είναι πολύ σημαντική, διότι αν δεν πραγματοποιηθεί σωστά η σύνδεση, δεν θα μπορεί ο κόμβος να λειτουργήσει κανονικά. Επειδή το σύστημά μας δεν έχει εξυπηρέτες, η σύνδεση γίνεται με τη βοήθεια ενός άλλου κόμβου που είναι ήδη συνδεδεμένος στο δίκτυό μας. Προς το παρόν, υποθέτουμε ότι έχουμε μία διαδικασία εύρεσης[1] ενός τέτοιου κόμβου (*boot process*), η οποία πληροί τις εξής συνθήκες:

1. Πάντα θα μας επιστρέψει έναν άλλο κόμβο που βρίσκεται ήδη στο δίκτυο.

2. Όλοι οι κόμβοι που είναι ήδη στο δίκτυο έχουν την ίδια πιθανότητα να επιστραφούν ως αποτέλεσμα.

Μετά την εύρεση ενός άλλου κόμβου που μετέχει στο δίκτυο, οι δύο κόμβοι ανταλλάσσουν κάποια πακέτα για να επιτευχθεί η σύνδεση του νέου κόμβου. Αρχικά, ο νέος κόμβος στέλνει μία αίτηση για σύνδεση στον υπάρχοντα κόμβο, ο οποίος του απαντάει στέλνοντάς του πληροφορίες (π.χ. διεύθυνση IP, ports που χρησιμοποιούν) για τους γείτονές του (ή μέχρι ένα όριο γειτόνων, άν έχει πολλούς), δηλαδή για άλλους κόμβους που γνωρίζει[2]. Ακόμα, προσθέτει τον κόμβο αυτό στην λίστα με γειτονικούς κόμβους που γνωρίζει.

Αν η λίστα αυτή είναι γεμάτη, δηλαδή έχει ξεπεράσει έναν αριθμό γειτόνων που έχουμε θέσει, ο κόμβος στέλνει πίσω στον νέο κόμβο που έκανε την αίτηση σύνδεσης ένα πακέτο απόρριψης. Ο νέος κόμβος τότε εκτελεί ξανά τη boot-process μέχρι να βρει έναν κόμβο που να τον δεχτεί.

Το επόμενο βήμα για να ολοκληρωθεί επιτυχώς η σύνδεση είναι να στείλει ο νέος κόμβος από ένα πακέτο στον καθένα από τους κόμβους που του γνωστοποίησε ο αρχικός, για να τους ενημερώσει για την ύπαρξή του και να τον τοποθετήσουν στις λίστες τους. Ο νέος κόμβος καταρτίζει τη δική του λίστα με τον αρχικό κόμβο και τους κόμβους αυτούς. Αλλοι κόμβοι θα προστεθούν όταν κάποιος κόμβος συνδεθεί πάνω του ή πάνω σε κάποιον γείτονά του.

[1] Για τη διεκόλυνση της υλοποίησης, μπορεί να χρησιμοποιηθεί μια στατική λίστα με διευθύνσεις άλλων κόμβων που υπάρχουν πάντα στο δίκτυο

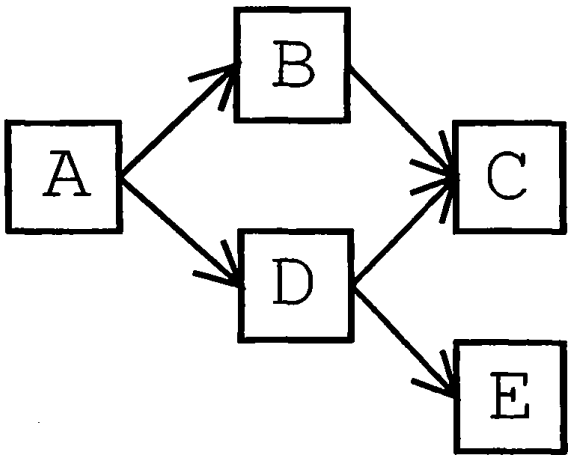
[2] π.χ που έχουν συνδεθεί σ' αυτόν ή που έχει συνδεθεί σ' αυτούς ο ίδιος

3.3 Αναζήτηση αρχείου

3.3.1 Με βάση το όνομα του αρχείου

Όταν ένας κόμβος θελήσει να αναζητήσει ένα αρχείο, θα πρέπει να στείλει ένα πακέτο αναζήτησης στους γείτονές του. Η συνθήκη αναζήτησης που θα περιέχεται στο πακέτο μπορεί να είναι το πλήρες όνομα του αρχείου, αν το γνωρίζει ο χρήστης, ή ένα τμήμα του ονόματος. Αυτοί, αν δεν έχουν κάποιο αρχείο που να ταιριάζει, θα προωθήσουν την αναζήτηση στους δικούς τους γείτονες, αυτοί στους δικούς τους κ.ο.κ. Όποιος κόμβος έχει κάποιο αρχείο που ταιριάζει με την αναζήτηση, αφού προωθήσει την αναζήτηση στους γείτονές του για να βρεθούν και άλλα αρχεία θα απαντήσει απ' ευθείας στον κόμβο που εκκίνησε την αναζήτηση με το όνομα του αρχείου ή των αρχείων που ταίριαξαν, καθώς και με κάποια χαρακτηριστικά τους (π.χ. μέγεθος, md5). Είναι προφανές ότι για να γίνει αυτό θα πρέπει μαζί με την συνθήκη αναζήτησης να στέλνει και κάποια προσωπικά του στοιχεία (π.χ. διεύθυνση IP, όνομα χρήστη).

Επίσης, κάθε πακέτο αναζήτησης θα πρέπει να περιέχει έναν αναγνωριστικό αριθμό αναζήτησης (*search-ID*), έτσι ώστε να μπορεί ένας κόμβος να ελέγξει αν απάντησε ήδη στην αναζήτηση αυτή ή αν την προώθησε. Λόγω της αρχιτεκτονικής του συστήματος, είναι πιθανόν μια αναζήτηση να περάσει πολλές φορές από τον ίδιο κόμβο (όπως φαίνεται και στο σχήμα 3.1), οπότε αυτός ο μηχανισμός ασφαλείας είναι απαραίτητος.



Σχήμα 3.1: Ο κόμβος C δέχεται την ίδια ερώτηση από δύο διαφορετικούς κόμβους

Στην περίπτωση που ένας κόμβος λάβει περισσότερες από μία φορές ένα πακέτο αναζήτησης, απαντάει ή το προωθεί μόνο την πρώτη φορά που θα το λάβει. Τις επόμενες φορές το αγνοεί. Ο κόμβος που έστειλε την αναζήτηση θα πρέπει να

μετρήσει πόσος χρόνος πέρασε από τη στιγμή που την έστειλε και αν ο χρόνος αυτός περάσει κάποιο όριο (*timeout*) να θεωρήσει την αναζήτηση ανεπιτυχή, αφού κατά πάσα πιθανότητα δεν έχει κανένας κόμβος το αρχείο που ζητάει και απλά οι υπόλοιποι κόμβοι στέλνουν την αναζήτηση ο ένας στο άλλον.

Τέλος, το πακέτο που περιέχει την αναζήτση θα πρέπει να περιέχει και ένα πεδίο "χρόνου ζωής" (*Time-to-live, TTL*) ή καλύτερα "βημάτων ζωής" (*Hops-to-live, HTL*), έτσι ώστε να μη διαδίδεται η αναζήτηση επ' άπειρον, αν το δίκτυό μας είναι αρκετά μεγάλο. Κάθε κόμβος που θα δέχεται δηλαδή το πακέτο της αναζήτησης για πρώτη φορά, θα ελέγχει την τιμή αυτή και αν δεν έχει μηδενιστεί, θα το προωθεί αφού πρώτα την ελαττώσει κατά μία μονάδα. Στην αντίθετη περίπτωση θα αγνοεί το πακέτο. Η ποσότητα αυτή θα πρέπει να είναι τέτοια ώστε να μην υπάρχει πιθανότητα ο κόμβος που εκκίνησε την αναζήτηση να έχει ξεπεράσει το χρονικό όριο αναμονής απάντησης, δηλαδή να έχει θεωρήσει την έρευνα ανεπιτυχή, και ταυτόχρονα να κυκλοφορούν πακέτα της αναζήτησης αυτής στο δίκτυο.

3.3.2 Με βάση το md5 του αρχείου

Ενδέχεται ένας κόμβος να χρειαστεί να αναζητήσει ένα αρχείο, όχι με βάση το όνομά του όπως είδαμε, αλλά με βάση το md5 του. Αυτό συμβαίνει σε περίπτωση που ο κόμβος χρειάζεται ένα *συγκεκριμένο* αρχείο, δηλαδή όταν ήδη κατεβάζει το αρχείο αυτό και διακοπεί το κατέβασμα για διάφορους λόγους (π.χ. αν σταματήσει η λειτουργία του κόμβου ή διακοπεί η σύνδεση). Η αναζήτηση αυτή γίνεται με τον ίδιο τρόπο που γίνεται και η αναζήτηση με βάση το όνομα, δηλαδή με πακέτα που στέλνονται μέσω των γειτονικών κόμβων, απλά αντί να περιέχουν μία συνθήκη αναζήτησης, περιέχουν το md5 του ζητούμενου αρχείου.

Και στην περίπτωση αυτή, η απάντηση του κόμβου που έχει το αρχείο θα αποσταλλεί απ' ευθείας στον κόμβο που έκανε την ερώτηση. Εδώ πρόκειται για θετική απάντηση για ένα μόνο αρχείο, αντίθετα με την προηγούμενη περίπτωση που σε μία συνθήκη αναζήτησης μπορούσαν να ταιριάξουν περισσότερα αρχεία. Ο κόμβος που κάνει την ερώτηση, την κάνει γιατί ενδιαφέρεται να κατεβάσει κομμάτια του συγκεκριμένου αρχείου. Για το λόγο αυτό, ο κόμβος που απαντάει ότι έχει το αρχείο θα στέλνει μαζί με την απάντηση και το ποια κομμάτια του αρχείου διαθέτει, για να δει ο αρχικός κόμβος αν τον βολεύει να κατεβάσει απ' αυτόν.

3.4 Μεταφορά αρχείου

3.4.1 Ερώτηση για κομμάτια

Ας υποθέσουμε ότι η αναζήτηση που έκανε ένας κόμβος ήταν επιτυχής και ότι έχει βρει το αρχείο που θέλει να κατεβάσει. Ας υποθέσουμε επίσης ότι το αρχείο αυτό το έχουν περισσότεροι από ένας χρήστες. Ο κόμβος τότε θα στείλει στον καθένα μια ερώτηση για το ποια κομμάτια (chunks) του αρχείου έχει και με βάση τις απαντήσεις τους θα δημιουργήσει μια λίστα από κομμάτια και ποιος έχει το καθένα. Το επόμενο βήμα είναι να επιλέξει με ποιον τρόπο θα κατεβάσει το αρχείο.

3.4.2 Πολιτικές Κατεβάσματος

Από τη στιγμή που ένας κόμβος έχει βρει ποιοι έχουν το αρχείο που ζητάει και ποια κομμάτια έχει ο καθένας, υπάρχουν πολλοι διαφορετικοί τρόποι για να το ζητήσει και να το κατεβάσει κομμάτι-κομμάτι. Δεν έχουμε καταλήξει ακόμα ως προς το ποιος είναι ο ενδεδειγμένος για να είναι το κατέβασμα πιο αποδοτικό.

Αρχικά, μπορεί ο κόμβος που θέλει να κατεβάσει το αρχείο να ζητήσει τυχαία κάποιο κομμάτι (διαφορετικό) από κάθε κόμβο που έχει το αρχείο. Όταν κάποιο από αυτά κατέβει εντελώς, μπορεί να ζητήσει και δεύτερο από τον ίδιο κόμβο. Ακόμα, αν κάποιος κόμβος αρνηθεί να δώσει ένα κομμάτι ή τον τοποθετήσει στην ουρά προς εξυπηρέτηση, μπορεί να ζητήσει το ίδιο κομμάτι και από άλλο κόμβο.

Επίσης, μπορεί ο κόμβος να ταξινομήσει τα κομμάτια του αρχείου που θέλει ως προς τη σπανιότητά τους, δηλαδή να τοποθετήσει πρώτο αυτό που το έχουν οι λιγότεροι κόμβοι, μετά το αμέσως λιγότερο σπάνιο, κτλ. Έτσι, μπορεί να ζητάει πρώτα τα πιο σπάνια κομμάτια από τους κόμβους που τα έχουν κι έπειτα τα λιγότερο σπάνια. Με τον τρόπο αυτό εξασφαλίζεται το ότι τα σπάνια κομμάτια ενός αρχείου θα διαδοθούν πιο γρήγορα, και τελικά θα πάψουν να είναι τόσο σπάνια. Δεν μας εγγυάται η μέθοδος αυτή την ταχύτητα όμως, γιατί είναι πολύ πιθανόν οι κόμβοι που έχουν τα σπάνια κομμάτια να τα δίνουν ταυτόχρονα σε πολλούς άλλους κόμβους, οπότε να ελαττώνεται το διαθέσιμο bandwidth για μεταφορά από τους κόμβους αυτούς.

Ακόμα, με βάση την προηγούμενη ταξινόμηση των κομματιών ως προς τη σπανιότητα, μπορεί ο κόμβος να ζητήσει πρώτα τα πιο διαδεδομένα κομμάτια, αυτά δηλαδή που έχει πολλές επιλογές ως προς τον κόμβο από τον οποίο θα τα κατεβάσει. Έτσι, αφ' ενός εγγυώμαστε ότι τα κομμάτια αυτά θα κατεβούν αρκετά γρήγορα, αφ' ετέρου μπορούμε να ελπίζουμε ότι μέχρι να τα κατεβάσει ο κόμβος τα διαδεδομένα κομμάτια του αρχείου, τα σπάνια κομμάτια θα έχουν διαδοθεί λίγο περισσότερο

(π.χ. σους κόμβους που έχουν τα πιο διαδεδομένα) και θα είναι πιο εύκολο να τα κατεβάσει αργότερα.

Τέλος, θα μπορούσε ο κόμβος που θέλει να κατεβάσει ένα αρχείο να ταξινομεί τους κόμβους που το διαθέτουν ως προς την ταχύτητα της μεταξύ τους σύνδεσης - π.χ. ως προς το χρόνο απάντησης σε ένα ping ή ως προς το χρόνο απάντησης στην ερώτηση για τα κομμάτια (ενότητα 3.4.1) - και έπειτα να ζητάει κομμάτια του αρχείου αρχικά από τους πιο γρήγορους κόμβους. Βεβαίως, μπορούν να γίνουν και συνδυασμοί των τεχνικών αυτών, δηλαδή π.χ. ο κόμβος να ζητάει τα πιο διαδεδομένα κομμάτια από τους πιο γρήγορους κόμβους

3.4.3 Αίτηση κατεβάσματος

Ανεξάρτητα από το ποια πολιτική κατεβάσματος θα ακολουθηθεί, ο κόμβος που θέλει να κατεβάσει ένα αρχείο, ή πιο σωστά ένα κομμάτι ενός αρχείου, από έναν άλλο κόμβο, θα πρέπει να του στείλει μία αίτηση, δηλώνοντας ποιο κομμάτι θέλει και από ποιο αρχείο.

Ο κόμβος που θα λάβει την αίτηση αυτή, έχει τις εξής επιλογές:

1. Να μπορεί να στείλει το κομμάτι αμέσως, οπότε στέλνει ένα μικρό πακέτο θετικής απάντησης και ύστερα αρχίζει να στέλνει τα δεδομένα του κομματιού.

2. Να μην μπορεί να στείλει το κομμάτι αμέσως διότι εξυπηρετεί άλλους κόμβους, αλλά να υπάρχει κενή θέση στην ουρά αναμονής που διατηρεί, με αιτήσεις προς εξυπηρέτηση. Στην περίπτωση αυτή, στέλνει ως απάντηση τη θέση της ουράς στην οποία τοποθετήθηκε η αίτηση του κόμβου.

3. Να μην μπορεί να εξυπηρετήσει την αίτηση αμέσως και η ουρά αναμονής να είναι γεμάτη, οπότε στέλνει αρνητική απάντηση και ο κόμβος που έκανε την αίτηση πρέπει να ζητήσει το κομμάτι από αλλού ή να περιμένει και να το ξαναζητήσει αργότερα από τον ίδιο κόμβο, μήπως και άδειασε μία θέση στην ουρά αναμονής.

3.5 Αποσύνδεση

3.5.1 Ομαλή αποσύνδεση

Όταν ένας κόμβος χρειάζεται να αποσυνδεθεί ομαλά από το δίκτυο, δηλαδή όταν ο χρήστης αποφασίσει να τερματίσει κανονικά το πρόγραμμα, πρέπει να ενημερώσει τους γειτονές του για την αποχώρησή του έτσι ώστε να μην του προωθούν πακέτα

αναζητήσεων τα οποία δεν θα φτάνουν ποτέ στον προορισμό τους. Αυτό μπορεί να γίνει απλούστατα με ένα πακέτο στο οποίο θα δηλώνει τα στοιχεία του (π.χ. διεύθυνση) και την πρόθεσή του να αποχωρήσει.

Οι κόμβοι που θα λάβουν τις ειδοποιήσεις αυτές, θα πρέπει να απομακρύνουν τον κόμβο που τις έστειλε από τις λίστες με τους γείτονές τους. Σε περίπτωση που ο αριθμός των γειτόνων στη λίστα κάποιου απ' αυτούς ελαττωθεί κάτω από ένα ορισμένο όριο που έχουμε θέσει ανάλογα με το μέγεθος του δικτύου, θα πρέπει ο κόμβος αυτός να ανακαλύψει νέους γείτονες ρωτώντας τους ήδη υπάρχοντες, αν είναι αρκετοί ή επανεκτελώντας τη διαδικασία σύνδεσής του στο δίκτυο (boot process).

3.5.2 Μη ομαλή αποσύνδεση

Ένας κόμβος που αποσυνδέεται με μη ομαλό τρόπο από το σύστημα, π.χ. αν πάθει κάποια βλάβη ο υπολογιστής ή η σύνδεση με το δίκτυο, θα πρέπει να εντοπιστεί και να απομακρυνθεί από τις λίστες όσων κόμβων τον έχουν ως γείτονα. Επειδή ο κόμβος που αποσυνδέεται ανώμαλα δεν έχει χρόνο να ειδοποιήσει για την αποσύνδεσή του, θα πρέπει να χρησιμοποιηθεί κάποια μέθοδος ανίχνευσης τέτοιων περιπτώσεων.

Προτείνουμε λοιπόν την απλούστερη δυνατή μέθοδο, δηλαδή κάθε κόμβος στη λίστα γειτόνων που διατηρεί να προσθέσει για κάθε γείτονα, εκτός από τα στοιχεία του, και ένα πεδίο που θα μετράει το χρόνο από την τελευταία επικοινωνία που είχαν οι δύο κόμβοι. Παράλληλα, σε τακτά χρονικά διαστήματα να ελέγχει τους γείτονές του με ένα PING αν λειτουργούν κανονικά και να ενημερώνει το πεδίο αυτό για τον καθένα. Το πεδίο θα ενημερώνεται επίσης όταν κάποιος γείτονας του στείλει ένα πακέτο αναζήτησης. Με τον τρόπο αυτό, αν το πεδίο για κάποιον κόμβο ξεπεράσει ένα ορισμένο χρονικό περιθώριο που έχουμε θέσει, ο κόμβος να θεωρείται ανενεργός και να απομακρύνεται από τη λίστα.

3.6 Επιπλέον χαρακτηριστικά

Τέλος, το πρόγραμμά μας θα θέλαμε να υποστηρίζει τη δημιουργία ομάδων από χρήστες που να μπορούν να ανταλλάσσουν αρχεία μόνο μέσα στην ομάδα, δηλαδή να υπάρχουν στον κάθε κόμβο αρχεία διαθέσιμα μόνο στα μέλη της ομάδας που ανήκει ο χρήστης αυτός.

Αυτό μπορεί να επιτευχθεί αν ο κάθε χρήστης διατηρεί στη λίστα με τα αρχεία που μοιράζεται ένα πεδίο για κάθε αρχείο που να δηλώνει αν το αρχείο αυτό το μοιράζεται με όλο το σύστημα ή μόνο με μία συγκεκριμένη ομάδα. Ακόμα, κάθε χρήστης πρέπει να έχει ένα αναγνωριστικό "όνομα χρήστη" (*username*), για να

μπορεί να ξεχωρίζει από τους υπόλοιπους. Τέλος, θα πρέπει να διατηρεί μία λίστα ή ένα αρχείο για κάθε ομάδα που συμμετέχει ο χρήστης που θα περιλαμβάνει τα μέλη της ομάδας, για παράδειγμα το όνομα του κάθε χρήστη και τη διεύθυνσή του (αν είναι μοναδική) ή τις διευθύνσεις που τυχόν χρησιμοποιεί.

Προς το παρόν, έχει προβλεφθεί να αποστέλλονται αυτά τα στοιχεία κάποιου χρήστη όταν αυτός κάνει μια αναζήτηση ενός αρχείου, έτσι ώστε ο κόμβος που θα έχει το αρχείο να μπορεί να ελέγξει αν ο χρήστης αυτός έχει δικαίωμα να το κατεβάσει. Αν ναι, θα του απαντήσει θετικά, αν όχι, θα απαντήσει ότι δεν έχει το αρχείο. Αυτό που δεν έχει σχεδιαστεί ακόμα είναι μία διαδικασία για να εισάγονται νέοι χρήστες στις ομάδες και βασιζόμαστε στο ότι οι χρήστες πριν την εκκίνηση του προγράμματός μας θα έχουν καταρτίσει ήδη τις λίστες των ομάδων τους με κάποιο τρόπο.

Κεφάλαιο 4

Απαιτούμενες δομές

Προτού αρχίσουμε να περιγράφουμε λεπτομερώς το πρωτόκολλο που χρησιμοποιεί το σύστημά μας για την επικοινωνία μεταξύ των κόμβων του, θεωρούμε χρήσιμο να αναφέρουμε κάποια στοιχεία που απαιτούνται για τη σωστή λειτουργία των πρωτοκόλλων. Γι' αυτό στις επόμενες ενότητες θα μελετήσουμε τι δομές (π.χ. λίστες) που πρέπει να έχει κάθε κόμβος για να διατηρεί διάφορα δεδομένα, τι αρχεία, προσωρινά και μη, χρειάζονται και πόσα ports πρέπει να δεσμευτούν.

Με τον τρόπο αυτό θα έχουμε αφ' ενός μια εικόνα για το ποιες θα είναι σε γενικές γραμμές οι απαιτήσεις του συστήματός μας (π.χ. σε μνήμη, χώρο στο δίσκο), και αφ' ετέρου θα μπορέσουμε να διευκολυνθούμε όταν στο επόμενο κεφάλαιο θα περιγράψουμε το πρωτόκολλο, αφού θα έχουμε ήδη ορίσει τα στοιχεία που απαιτούνται.

4.1 Δομές

4.1.1 Λίστες γειτόνων

Αρχικά, όπως είδαμε στην ενότητα 3.2 για τη σύνδεση ενός κόμβου, κάθε κόμβος πρέπει να διατηρεί μία δομή με τους γείτονές του. Η δομή αυτή θα είναι μία απλή λίστα, για να είναι δυναμική, να μην καταλαμβάνει δηλαδή σταθερό χώρο στη μνήμη αλλά να αυξομειώνεται ανάλογα με το πόσους γείτονες έχει ο κόμβος. Είναι φυσικό να μην θέλουμε αυτή η λίστα να μεγαλώνει επ' άπειρον, οπότε, όπως αναφέραμε, θα πρέπει να υπάρχει ένα άνω όριο μεγέθους της, δηλαδή ένα άνω όριο γειτόνων που θα έχει ο κάθε κόμβος.

Κάθε μέλος της λίστας θα διατηρεί τις εξής πληροφορίες για κάθε γείτονα:

- Τη διεύθυνση IP του κόμβου.

- Τα ports που χρησιμοποιεί για να δέχεται αιτήσεις και ενημερώσεις (Ενότητα 4.3).

- Έναν αριθμό που αντιπροσωπεύει το χρόνο που ο κόμβος αυτός είναι ανενεργός, δηλαδή το χρόνο που πέρασε από την τελευταία επικοινωνία που είχαμε με τον κόμβο. Αν ο χρόνος αυτός ξεπεράσει μία ορισμένη τιμή, θεωρούμε τον κόμβο ανενεργό και τον αφαιρούμε από τη λίστα.

Για καλύτερη απόδοση του συστήματος, η λίστα αυτή που αναφέρουμε είναι προτιμότερο να χωριστεί σε δύο. Η πρώτη λίστα θα περιέχει τους "άμεσους" γείτονες του κόμβου, δηλαδή τον κόμβο στον οποίο συνδέθηκαμε και τους κόμβους οι οποίοι συνδέθηκαν πάνω μας, ενώ η δεύτερη θα περιέχει τους υπόλοιπους γείτονες, δηλαδή αυτούς για τους οποίους μας ενημέρωσε ο αρχικός κόμβος και αυτούς που τυχόν συνδέθηκαν σε κάποιον που είναι συνδεδεμένος πάνω μας. Εναλλακτικά, μπορούμε να πούμε ότι στην πρώτη λίστα αποθηκεύονται οι κόμβοι που απέχουν από μας ένα βήμα, ενώ στη δεύτερη αυτοί που απέχουν δύο βήματα. Θα ονομάσουμε τις λίστες αυτές *n1* και *n2* αντίστοιχα.

4.1.2 Λίστες αναζητήσεων

Θα χρειαστούν ακόμα κάποιες δομές για τη διαχείριση των αναζητήσεων που έχει κάνει ένας κόμβος, καθώς και των αναζητήσεων που έχει εξυπηρετήσει, δηλάδή προωθήσει. Είναι απαραίτητο ένας κόμβος να θυμάται για ένα λογικό χρονικό διάστημα ποιες αναζητήσεις έχει κάνει, έτσι ώστε αν έρθει ένα καθυστερημένο πακέτο για κάποια απ' αυτές να το δεχθεί. Επίσης θα πρέπει να θυμάται και τις αναζητήσεις που έχει εξυπηρετήσει για να μην να εξυπηρετήσει την ίδια αίτηση περισσότερες φορές, που όπως είδαμε μπορεί να του ξαναέρθει.

Και στις δύο λίστες αυτές θα πρέπει για κάθε αναζήτηση να διατηρούνται το search-ID της και ένα πεδίο που θα υποδηλώνει το χρόνο από τη στιγμή που εστάλη ή εξυπηρετήθηκε η αίτηση. Προφανώς, όταν περάσει αρκετός χρόνος από την αποστολή ή προώθηση της αίτησης, αυτή θα απομακρύνεται από τη λίστα. Καλό θα ήταν οι λίστες αυτές να είναι ταξινομημένες ως προς το χρόνο, έτσι ώστε η αναζήτηση και απομάκρυνση των εγγραφών που ξεπέρασαν το χρονικό όριο να είναι πιο εύκολη.

4.1.3 Ουρά αναμονής

Τέλος, κάθε κόμβος θα πρέπει να διατηρεί μία δομή με τους κόμβους που περιμένουν να εξυπηρετηθούν, δηλάδή που περιμένουν να τους στείλει κάποιο κομμάτι ενός αρχείου. Όπως είδαμε στην ενότητα 3.4.3 μία τέτοια δομή είναι απαραίτητη

γιατί ένας κόμβος δεν μπορεί να εξυπηρετεί πολλούς κόμβους ταυτόχρονα, οπότε εξυπηρετεί κάποιους μέχρι ένα όριο που έχουμε θέσει και όσοι κόμβοι από κει και πέρα θέλουν να κατεβάσουν κάτι απ' αυτόν θα τοποθετούνται στη δομή αυτή.

Είναι προφανές ότι η δομή αυτή θα είναι μία ουρά, έτσι ώστε οι κόμβοι που κάνουν πρώτοι χρονικά μία αίτηση για ένα κομμάτι αρχείου να τοποθετούνται πρώτοι και συνεπώς να εξυπηρετούνται πρώτοι. Η ουρά αυτή δεν μπορεί να εκτείνεται απεριόριστα αλλά θα πρέπει να έχει ένα ορισμένο μέγεθος. Το μέγεθος αυτό μπορεί να ορίζεται από το χρήστη, ανάλογα με τη διάθεση που έχει να μοιραστεί αρχεία, ή μπορεί να υπολογίζεται αυτόματα από το πρόγραμμα, βάσει υπολογισμών που κάνει για την απόδοση του υπολογιστή και την ταχύτητα της δικτυακής σύνδεσης που υπάρχει.

Για κάθε κόμβο που περιμένει στην ουρά θα πρέπει να κρατάμε:

- Την διεύθυνση IP του κόμβου που περιμένει το αρχείο και το port στο οποίο περιμένει τα δεδομένα.
- Το μέγεθος και το md5 του αρχείου, που το χαρακτηρίζουν με μοναδικό τρόπο.
- Τα κομμάτια του αρχείου που έχει ζητήσει ο κόμβος.

4.2 Αρχεία

Εκτός από τις παραπάνω δομές, το σύστημα θα αποθηκεύει ένα μέρος των χρήσιμων πληροφοριών σε κάποια αρχεία, είτε γιατί πρέπει να τις ξαναβρεί την επόμενη φορά που θα εκκινήσει, είτε γιατί ο όγκος τους είναι αρκετά μεγάλος για να αποθηκευτεί στη μνήμη. Τα αρχεία αυτά είναι δύο τύπων: ένα αρχείο για να αποθηκευτούν τα δεδομένα για τα αρχεία που μοιράζεται ο κόμβος αυτός και από ένα αρχείο για κάθε ενεργό κατέβασμα που έχει ο κόμβος, δηλαδή για κάθε αρχείο που κατεβάζει κάθε στιγμή.

Το πρώτο αρχείο θα δημιουργείται κάθε φορά που εκτελείται το πρόγραμμα, διαβάζοντας τους καταλόγους που έχει δηλώσει ο χρήστης ότι θέλει να μοιραστεί. Θα περιέχει μία εγγραφή για κάθε αρχείο που μοιράζεται, είτε είναι ολόκληρο είτε όχι. Η εγγραφή ενός αρχείου θα περιλαμβάνει το όνομά του, το μέγεθός του, το md5 του και τα κομμάτια του που διαθέτει ο κόμβος.

Είναι φανερό, ότι αφού θέλουμε (ενότητα 3.1) ένα αρχείο να είναι διαθέσιμο ακόμα κι αν δεν έχει κατεβεί ολόκληρο, ότι το αρχείο με τις πληροφορίες για τα μοιραζόμενα αρχεία θα πρέπει να ανανεώνεται κάθε φορά που κατεβαίνει ένα κομ-

μάτι ενός αρχείου, έτσι ώστε το κομμάτι αυτό να γίνεται αμέσως διαθέσιμο στους υπόλοιπους κόμβους.

Όσον αφορά τα αρχεία με τις περιγραφές για τα ενεργά κατεβάσματα, θα δημιουργούνται κάθε φορά που αρχίζει ένα κατέβασμα και θα διαγράφονται όταν το κατέβασμα τελειώσει, δηλαδή όταν κατεβεί ολόκληρο το αρχείο. Ο λόγος για τον οποίο τα χρειαζόμαστε είναι αφ' ενός για να ξέρουμε ποια κομμάτια έχουμε από το ημιτελές αρχείο και αφ' ετέρου για να ξέρουμε κάθε στιγμή από ποιον κατεβάζουμε κάθε κομμάτι, έτσι ώστε αν διακοπεί το κατέβασμα να μπορούμε πιθανώς να το συνεχίσουμε χωρίς νέα αναζήτηση.

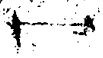
Έτσι, για κάθε αρχείο που κατεβαίνει δημιουργείται ένα προσωρινό αρχείο που περιέχει τα χαρακτηριστικά του αρχείου (μέγεθος και md5), ποια κομμάτια του αρχείου έχουμε ολόκληρα (απλή αναφορά, όχι τα δεδομένα) και ποια κατεβαίνουν τη στιγμή αυτή. Επίσης, για κάθε κομμάτι που κατεβαίνει θα έχει τη διεύθυνση του κόμβου από τον οποίο κατεβάζουμε. Έτσι, σε περίπτωση που διακοπεί το κατέβασμα από κάποιο πρόβλημα του υπολογιστή μας ή του δικτύου να ξέρουμε τουλάχιστον ένα κόμβο που το έχει και να έχουμε μια πιθανότητα να το κατεβάσουμε χωρίς περεταίρω αναζήτηση (αν ο κόμβος αυτός λειτουργεί όταν θελήσουμε να συνεχίσουμε). Το αρχείο αυτό θα ανανεώνεται κάθε φορά που κατεβεί ένα κομμάτι, ώστε να δηλωθεί ολόκληρο, και κάθε φορά που αρχίζει να κατεβαίνει ένα κομμάτι, ώστε να αποθηκευθούν τα δεδομένα του κόμβου απ' τον οποίο κατεβάζουμε.

4.3 Ports

Τα ports που χρειάζεται να δεσμεύσει κάθε κόμβος είναι τα εξής:

1. Ένα port στο οποίο θα δέχεται συνδέσεις νέων κόμβων με πακέτα TCP.
2. Ένα port για να τον ενημερώνουν οι γειτονικοί κόμβοι για την ύπαρξή τους ή την αποσύνδεσή τους με πακέτα UDP.
3. Ένα port για να δέχεται αιτήσεις αναζήτησης από τους γείτονές του με πακέτα UDP.
4. Ένα port για να δέχεται τις απαντήσεις στις δικές του αναζητήσεις με πακέτα UDP.
5. Ένα port για να λαμβάνει κομμάτια αρχείων μέσω σύνδεσης TCP.
6. Ένα port για να δέχεται συνδέσεις TCP σε περίπτωση που έχει πολλά αρχεία που ταιριάζουν σε μια αναζήτηση.

Περισσότερες λεπτομέρειες για τη χρήση του κάθε port θα δούμε στο επόμενο κεφάλαιο, όπου θα περιγράψουμε τα πρωτόκολλα.



Κεφάλαιο 5

Περιγραφή του πρωτοκόλλου επικοινωνίας

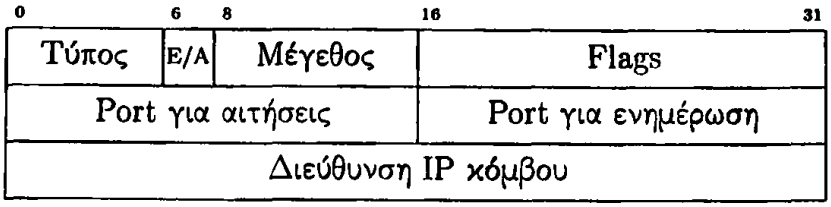
5.1 Σύνδεση νέου κόμβου

Ας δούμε με μεγαλύτερη λεπτομέρεια τη διαδικασία σύνδεσης ενός νέου κόμβου που περιγράψαμε στην ενότητα 3.2. Υποθέτουμε ότι ο κόμβος που θέλει να συνδεθεί έχει ήδη εκτελέσει την boot process του και έχει βρει έναν άλλο κόμβο που είναι συνδεδεμένος στο δίκτυό μας. Τότε ακολουθούν τα εξής βήματα:

Βήμα 1: Ο νέος κόμβος αποστέλλει στον υπάρχοντα κόμβο ένα πακέτο TCP που περιέχει:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (σύνδεση), με τιμή 000001 (=1).
- 2 bits που δηλώνουν το είδος του πακέτου (αίτηση), με τιμή 01 (=1).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 2 bytes για flags, επειδή το ίδιο πακέτο μπορεί να χρησιμοποιηθεί και σε άλλες περιπτώσεις, με τιμή στην περίπτωση αυτή 0.
- 2 bytes το port στο οποίο ο νέος κόμβος θα δέχεται αιτήσεις αναζήτησης.
- 2 bytes το port στο οποίο ο νέος κόμβος θα δέχεται ενημέρωση από τους γείτονές του.
- 4 bytes που περιέχουν τη διεύθυνση IP του νέου κόμβου, για να τον προσθέσει ο ήδη συνδεδεμένος κόμβος στη λίστα με τους γείτονές του.

Το πακέτο αυτό φαίνεται στο σχήμα 5.1.

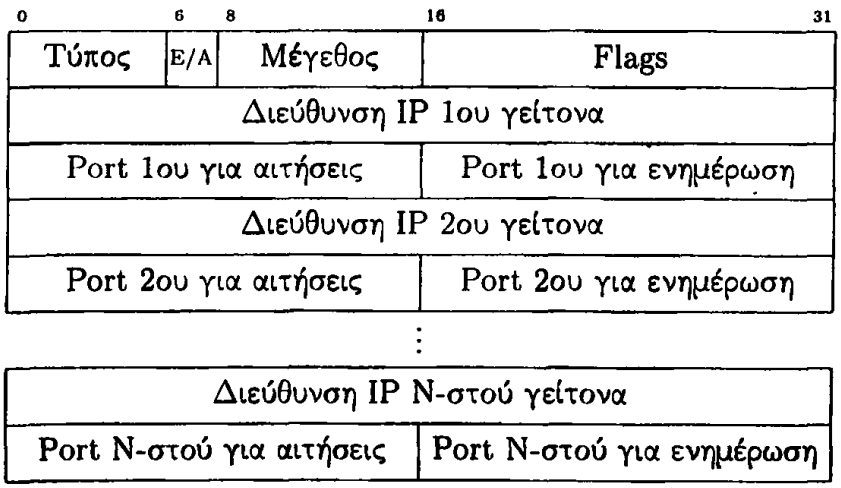


Σχήμα 5.1: Πακέτο αίτησης για σύνδεση νέου κόμβου

Βήμα 2: Ο κόμβος που είναι συνδεδεμένος ήδη στο δίκτυο, όταν λάβει το πακέτο αυτό, αρχικά θα πρέπει να ελέγξει τον αριθμό των γειτόνων του στη λίστα *n1* με τους άμεσους γείτονές του. Αν δεν έχει φτάσει στο όριο που έχει τεθεί, κατασκευάζει το πακέτο TCP του σχήματος 5.2, το οποίο περιέχει:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (σύνδεση), με τιμή 000001 (=1).
- 2 bits που δηλώνουν το είδος του πακέτου (απάντηση), με τιμή 10 (=2).
- 1 byte το μέγεθος του πακέτου, πάντα σε τετράδες bytes (2N+1).
- 2 bytes για flags.
- 8N bytes, 4 για τη διεύθυνση IP του κάθε γείτονα και 4 για τα ports του (2 για το port αιτήσεων και 2 για το port ενημέρωσης)

και το αποστέλλει στο νέο κόμβο.



Σχήμα 5.2: Πακέτο απάντησης σε αίτηση σύνδεσης νέου κόμβου

Στο πακέτο αυτό ο κόμβος αποστέλλει στον νέο κόμβο πληροφορίες για N γείτονές του και μάλιστα για άμεσους γείτονές του (της λίστας *n1*). Το N είναι επιλεγμένο έτσι ώστε να μην μεγαλώνει υπερβολικά ο αριθμός των γειτόνων κάθε κόμβου. Αν ο κόμβος δεν έχει N γείτονες στη λίστα *n1*, τότε αποστέλλει λιγότερους (όσους έχει). Αν έχει περισσότερους, επιλέγει τυχαία N απ' αυτούς, με την ίδια πιθανότητα.

Κατόπιν, ο κόμβος αυτός προσθέτει τον νέο κόμβο στη λίστα *n1* με τους άμεσους γείτονές του.

Αν η λίστα *n1* είναι γεμάτη, δηλαδή έχει φτάσει στο όριο που έχουμε θέσει, ο κόμβος στέλνει πίσω στο νέο κόμβο το ίδιο πακέτο (σχήμα 5.2), χωρίς τις πληροφορίες για τους γείτονες (δηλαδή μόνο τα πρώτα 4 bytes), δηλώνοντας κατάλληλα στο πεδίο "flags" ότι η αίτηση για σύνδεση απορρίπτεται. Αν $flags = 0$ τότε η αίτηση έγινε δεκτή, αν $flags = 1$ τότε η αίτηση απορρίπτεται. Βέβαια, χωρίς σημαντική αλλαγή του πρωτοκόλλου, θα μπορούσε να στέλνει πίσω και τους γείτονές του, έτσι ώστε ο νέος κόμβος να δοκιμάσει να συνδεθεί σε έναν από αυτούς και να μην χρειαστεί να επανεκτελέσει τη boot process.

Βήμα 3: Ο νέος κόμβος δέχεται το πακέτο που του έστειλε ο ήδη συνδεδεμένος κόμβος και ελέγχει τα flags για να δει αν έγινε δεκτή η αίτησή του. Αν όχι, επανεκτελεί τη boot process, βρίσκει έναν άλλο κόμβο και ξαναρχίζει τη διαδικασία από το Βήμα 1.

Αν η αίτηση έγινε δεκτή, τοποθετεί τον ήδη συνδεδεμένο κόμβο στη λίστα *n1* με τους άμεσους γείτονές του και τους υπόλοιπους κόμβους, τους οποίους του γνωστοποίησε ο αρχικός στη λίστα *n2*, αφού ουσιαστικά απέχουν δύο βήματα απ' αυτόν.

Επιπλέον, στέλνει στον καθένα τους ένα UDP πακέτο με περιεχόμενα σαν αυτά του πακέτου του σχήματος 5.1, θέτοντας όμως στο πεδίο "flags" την τιμή 1 για να δηλώσει ότι δεν πρόκειται για αίτηση άμεσης σύνδεσης αλλά για ενημέρωση ότι υπάρχει. Ο λόγος που στην αρχική σύνδεση χρησιμοποιούμε σύνδεση TCP ενώ εδώ πακέτα UDP είναι ότι αφ' ενός στην αρχική θέλουμε να είμαστε σίγουροι ότι το πακέτο θα φτάσει στον προορισμό του και αφ' ετέρου η απάντηση που περιμένουμε μπορεί να έχει μέγεθος μεγαλύτερο από αυτό που δέχεται ένα πακέτο UDP (512 bytes).

Βήμα 4: Οι κόμβοι που θα λάβουν αυτή την ενημέρωση, θα τοποθετήσουν το νέο κόμβο στη λίστα *n2* τους, αφού δεν πρόκειται για άμεσο γείτονα (δεν συνδέ-

θηκε πάνω τους, αλλά πάνω σε γειτονά τους), αν υπάρχει διαθέσιμος χώρος. Αν δεν υπάρχει, απλά αγνοούν το πακέτο.

5.2 Αποσύνδεση κόμβου

Κατά την ομαλή αποσύνδεση ενός κόμβου, ο κόμβος πρέπει να στείλει ειδοποιήσεις στους γείτονές του έτσι ώστε να τον διαγράψουν από τις λίστες τους. Το πακέτο που θα στείλει αρκεί να περιέχει τα στοιχεία του κόμβου, δηλάδή τη διεύθυνσή του και τα ports στα οποία "ακούει". Θα στείλει δηλαδή ένα UDP πακέτο όμοιο με αυτό που είχε στείλει για την ενημέρωση των γειτόνων του, που φαίνεται στο σχήμα 5.1, δηλώνοντας απλά ότι πρόκειται για αποσύνδεση δηλαδή βάζοντας στο πεδίο "flags" την τιμή 2. Το πακέτο αυτό θα σταλεί σε όλους τους γείτονές του, άμεσους και μη.

Ο κόμβος που θα λάβει τώρα ένα τέτοιο πακέτο που δηλώνει ότι ένας γείτονάς του αποχώρησε από το δίκτυο, θα ενεργήσει ανάλογα με ένα από τα τρια ενδεχόμενα:

1. Ο κόμβος που αποχώρησε να μην ήταν δηλωμένος ως γείτονας του κόμβου αυτού, απλούστατα διότι όταν τον ενημέρωσε για την ύπαρξή του οι λίστες του κόμβου ήταν γεμάτες και αγνόησε το πακέτο. Στην περίπτωση αυτή αγνοεί και αυτό το πακέτο.

2. Ο κόμβος που αποχώρησε να ήταν δηλωμένος ώς μη άμεσος γείτονας του κόμβου αυτού, δηλαδή στη λίστα *n2*. Στην περίπτωση αυτή απλά απομακρύνει τον κόμβο από τη λίστα του.

3. Ο κόμβος που αποχώρησε να ήταν δηλωμένος ώς άμεσος γείτονας του κόμβου αυτού, δηλαδή στη λίστα *n1*. Στην περίπτωση αυτή απομακρύνει τον κόμβο από τη λίστα του και ύστερα επιλέγει έναν κόμβο από τη λίστα *n2* και τον προσθέτει στη *n1*, έτσι ώστε ο αριθμός των άμεσων γειτόνων του να μην ελαττωθεί. Αν δεν υπάρχουν διαθέσιμοι κόμβοι στη *n2* και ο αριθμός των γειτόνων της *n1* είναι πολύ μικρός, επανεκτελεί τη boot process του έτσι ώστε να βρει κάποιους γείτονες για να μπορεί να λειτουργεί κανονικά.

Οι περιπτώσεις (2) και (3) μπορούν και πρέπει να εφαρμοστούν και όταν διαπιστώσει ότι κάποιος γείτονάς του (άμεσος ή μη) έχει πάψει να λειτουργεί με μη ομαλό τρόπο. Θα πρέπει πάλι να τον αφαιρέσει από την αντίστοιχη λίστα και να προβεί στις αντίστοιχες ενέργειες.

5.3 Αναζήτηση αρχείου με βάση το όνομά του

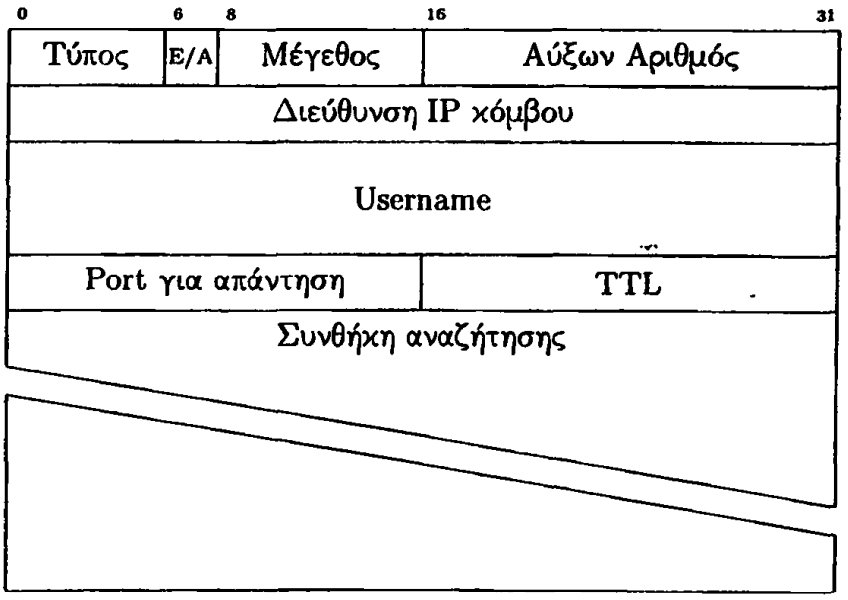
Στην περίπτωση που ένας κόμβος θελήσει να αναζητήσει ένα αρχείο με βάση το όνομά του ή ένα τμήμα του ονόματος του, θα στείλει όπως είδαμε ένα UDP πακέτο (σχήμα 5.3) αναζήτησης στους άμεσους γείτονές του, το οποίο θα περιέχει τα εξής δεδομένα:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (αναζήτηση με βάση το όνομα), με τιμή 000010 (=2).
- 2 bits που δηλώνουν το είδος του πακέτου (ερώτηση), με τιμή 01 (=1).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 2 bytes με έναν αύξοντα αριθμό, ο οποίος σε συνδυασμό με την ακόλουθη διεύθυνση IP και το username δημιουργούν ένα μοναδικό search-ID.
- 4 bytes που περιέχουν τη διεύθυνση IP του κόμβου που εκκίνησε την αναζήτηση.
- 8 bytes το όνομα του χρήστη (username) που κάνει την αναζήτηση.
- 2 bytes το port στο οποίο περιμένει απάντηση ο κόμβος που εκκίνησε την αναζήτηση.
- 2 bytes με το χρόνο ζωής του πακέτου (TTL).
- τη συνθήκη αναζήτησης (μεταβλητού μεγέθους), η οποία είναι μία συμβολοσειρά (string) που τερματίζει με '\0'. Αν δεν συμπληρώνονται ακριβώς οι τετράδες bytes, τοποθετούμε στο τέλος για συμπλήρωμα 1 έως 3 '\0' επιπλέον. Η συνθήκη μπορεί να είναι το όνομα του αρχείου ή ένα τμήμα του ονόματος.

Το search-ID βλέπουμε ότι δημιουργείται από τη σύνθεση τριών πεδίων, της διεύθυνσης IP του κόμβου, που τον χαρακτηρίζει μοναδικά, το όνομα του χρήστη, επειδή ενδέχεται στον ίδιο κόμβο να εκτελούνται πολλά αντίτυπα του προγράμματος, το καθένα από διαφορετικό χρήστη, και ενός αύξοντα αριθμού μεγέθους 2 bytes (0-65535), για να ξεχωρίζουν μεταξύ τους διαδοχικές αναζητήσεις του ίδιου χρήστη. Μετά από 65535 αναζητήσεις που κάνει ένας κόμβος, ο αριθμός ξαναρχίζει απ' το 0 αλλά θεωρούμε μάλλον απίθανο να έχουν γίνει 65535 αναζητήσεις και να υπάρχει ακόμα κάποιο πακέτο που να αναφέρεται στην πρώτη αναζήτηση. Εξάλλου, το TTL μας εξασφαλίζει το γεγονός αυτό, αν είναι επιλεγμένο κατάλληλα.

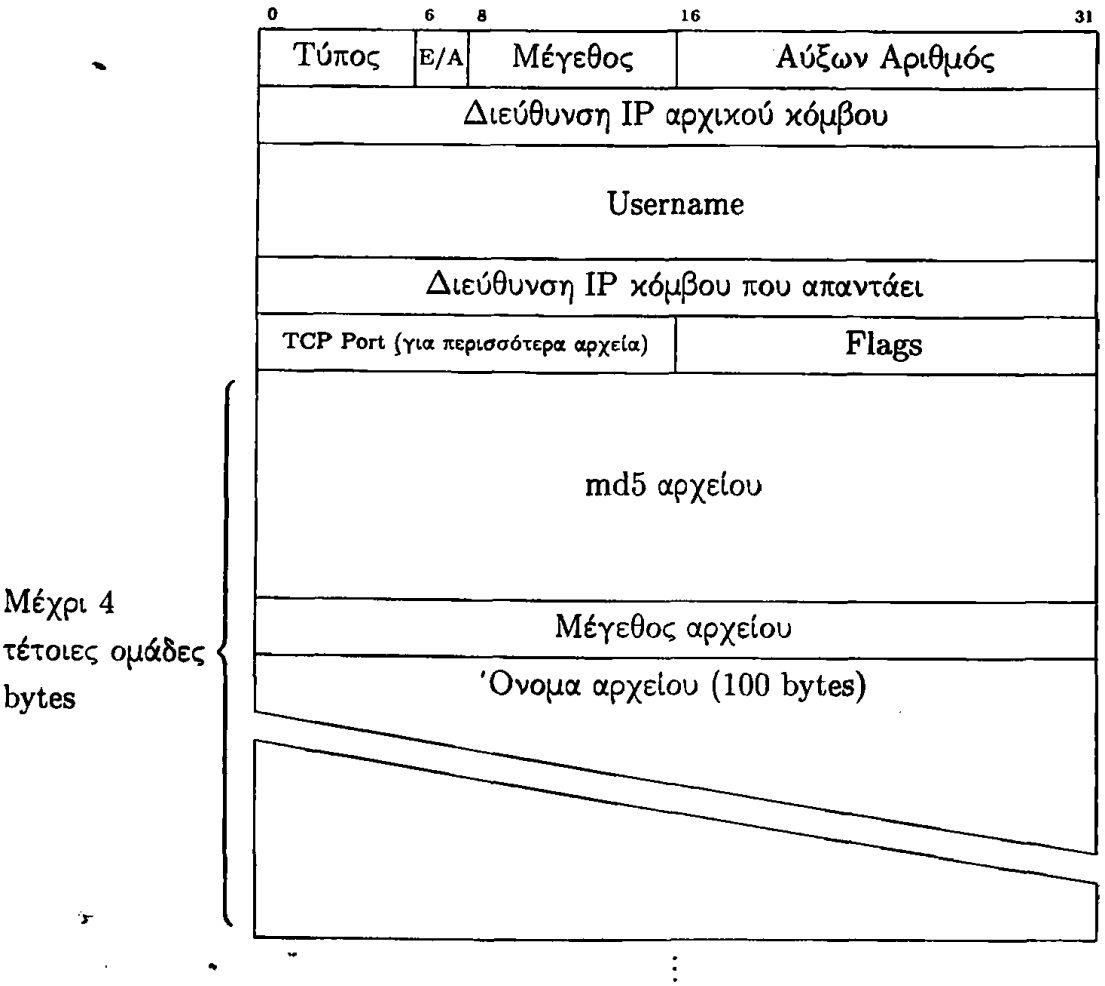
Ένας κόμβος που θα δεχτεί ένα τέτοιο πακέτο αναζήτησης, θα ελέγξει πρώτα το search-ID του για να δει αν έχει ήδη επεξεργαστεί την αναζήτηση αυτή. Σε θετική περίπτωση θα αγνοήσει το πακέτο ενώ σε αρνητική θα ελέγξει το αρχείο που διατηρεί με πληροφορίες για τα αρχεία που μοιράζεται (ενότητα 4.2) για να δει αν κάποιο από αυτά ταιριάζει με την αναζήτηση. Αν ναι, θα στείλει στον κόμβο που εκκίνησε την αναζήτηση ένα UDP πακέτο σαν αυτό του σχήματος 5.4 που θα περιέχει:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (αναζήτηση με βάση το όνομα), με τιμή 000010 (=2).
- 2 bits που δηλώνουν το είδος του πακέτου (απάντηση), με τιμή 10 (=2).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 2 bytes τον αύξοντα αριθμό του πακέτου ερώτησης.
- 4 bytes που περιέχουν τη διεύθυνση IP του κόμβου που εκκίνησε την αναζήτηση, για να σχηματιστεί το search-ID.
- 8 bytes το όνομα του χρήστη (username) που κάνει την αναζήτηση.
- 4 bytes που περιέχουν τη δική του διεύθυνση IP.
- 2 bytes το TCP port στο οποίο θα γίνει η σύνδεση αν υπάρχουν περισσότερα αρχεία.



Σχήμα 5.3: Πακέτο αναζήτησης με βάση το όνομα ενός αρχείου

- 2 bytes για flags (π.χ. αν υπάρχουν πολλά αρχεία που ταιριάζουν, οπότε θα χρειαστεί TCP σύνδεση).
- μέχρι 4 ομάδες από τα παρακάτω:
  - 16 bytes το *μοναδικό* md5 του αρχείου.
  - 4 bytes το μέγεθος του αρχείου.
  - 100 bytes για το όνομα του αρχείου. Αν χρησιμοποιούνται λιγότερα, τότε τοποθετούμε στο τέλος για συμπλήρωμα όσα '\0' χρειάζονται.



Σχήμα 5.4: Πακέτο απάντησης σε αναζήτηση με βάση το όνομα ενός αρχείου

Με το πακέτο αυτό, ο κόμβος μπορεί να απαντήσει το πολύ για τέσσερα (4) αρχεία που έχει και ταιριάζουν με την αναζήτηση. Αυτό γίνεται γιατί το μέγεθος

ενός πακέτου UDP δεν μπορεί να υπερβαίνει τα 512 bytes. Έτσι, σε περίπτωση που ο κόμβος έχει περισσότερα από 4 αρχεία που ταιριάζουν δεν στέλνει ολόκληρο το πακέτο, αλλά μόνο τα 24 πρώτα bytes (6 τετράδες), δηλώνοντας στο πεδίο "flags" ότι υπάρχουν πολλά αρχεία και ταυτόχρονα το port στο οποίο περιμένει TCP σύνδεση για να δώσει τις λεπτομέρειές τους. Το πεδίο "flags" θα έχει τιμή ίση με 0 αν δεν βρέθηκαν περισσότερα από 4 αρχεία και τιμή ίση με 1 αλλιώς.

Ο κόμβος που θα λάβει απάντηση ότι υπάρχουν πολλά αρχεία, θα συνδεθεί με TCP σύνδεση στον κόμβο που απαντάει και θα του στείλει την ίδια αναζήτηση (σχήμα 5.3). Ο άλλος κόμβος τότε θα του απαντήσει με ένα πακέτο TCP όμοιο με αυτό του σχήματος 5.4 αλλά χωρίς τον περιορισμό των τεσσάρων αρχείων. Μπορεί δηλαδή να του στείλει πληροφορίες για όλα τα αρχεία που έχει και ταιριάζουν με την αναζήτηση.

Σε κάθε περίπτωση πάντως, δηλαδή είτε ο κόμβος έχει περισσότερα από 4 αρχεία που να ταιριάζουν, είτε λιγότερα, είτε κανένα, θα πρέπει να προωθήσει την αίτηση και στους δικούς του γείτονες έτσι ώστε να βρεθούν κι άλλοι κόμβοι που να έχουν αρχεία για να διευκολυνθεί το κατέβασμα. Θα ελέγξει λοιπόν το TTL του αρχικού πακέτου και αν δεν έχει μηδενιστεί θα προωθήσει το πακέτο στους άμεσους γείτονές του ελαττώνοντας το TTL κατά ένα. Αν έχει μηδενιστεί σημαίνει ότι η αναζήτηση δεν μπορεί να συνεχίσει και άρα δεν θα προωθήσει το πακέτο.

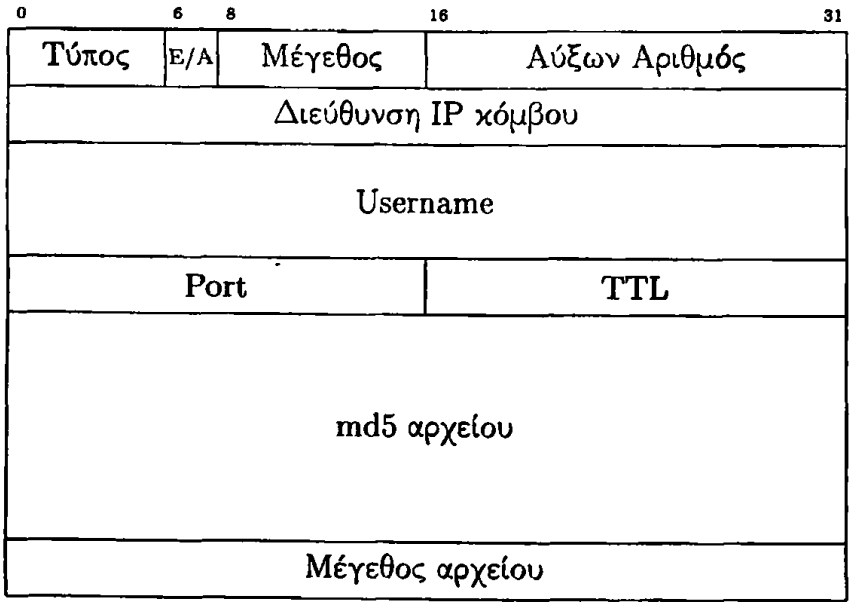
5.4 Αναζήτηση αρχείου με βάση το md5 του

Είδαμε στην ενότητα 3.3.2 ότι είναι δυνατόν ένας κόμβος να θελήσει να αναζητήσει κάποιο αρχείο με βάση το md5 του. Αυτό είναι πιθανόν όταν διακοπεί το κατέβασμα ενός αρχείου και ο κόμβος θελήσει να βρει ποιοι κόμβοι έχουν το *συγκεκριμένο* αρχείο για να συνεχίσει το κατέβασμα απ' αυτούς.

Η αναζήτηση γίνεται με τον ίδιο τρόπο που γίνεται και η αναζήτηση με βάση το όνομα ενός αρχείου, δηλαδή με UDP πακέτα που στέλνονται μέσω των άμεσων γειτόνων. Το μόνο που αλλάζει είναι τα περιεχόμενα του πακέτου αναζήτησης (σχήμα 5.5), τα οποία είναι τα εξής:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (αναζήτηση με βάση το md5), με τιμή 000100 (=4).
- 2 bits που δηλώνουν το είδος του πακέτου (ερώτηση), με τιμή 01 (=1).
- 1 byte το μέγεθος του πακέτου, σε τετράδες bytes.
- 2 bytes με έναν αύξοντα αριθμό, για το search-ID.

- 4 bytes που περιέχουν τη διεύθυνση IP του κόμβου που εκκίνησε την αναζήτηση.
- 8 bytes το όνομα του χρήστη (username) που κάνει την αναζήτηση.
- 2 bytes το port στο οποίο περιμένει απάντηση ο κόμβος που εκκίνησε την αναζήτηση.
- 2 bytes ο χρόνος ζωής του πακέτου (TTL).
- 16 bytes το md5 του ζητούμενου αρχείου.
- 4 bytes το μέγεθος του ζητούμενου αρχείου.



Σχήμα 5.5: Πακέτο αναζήτησης με βάση το md5

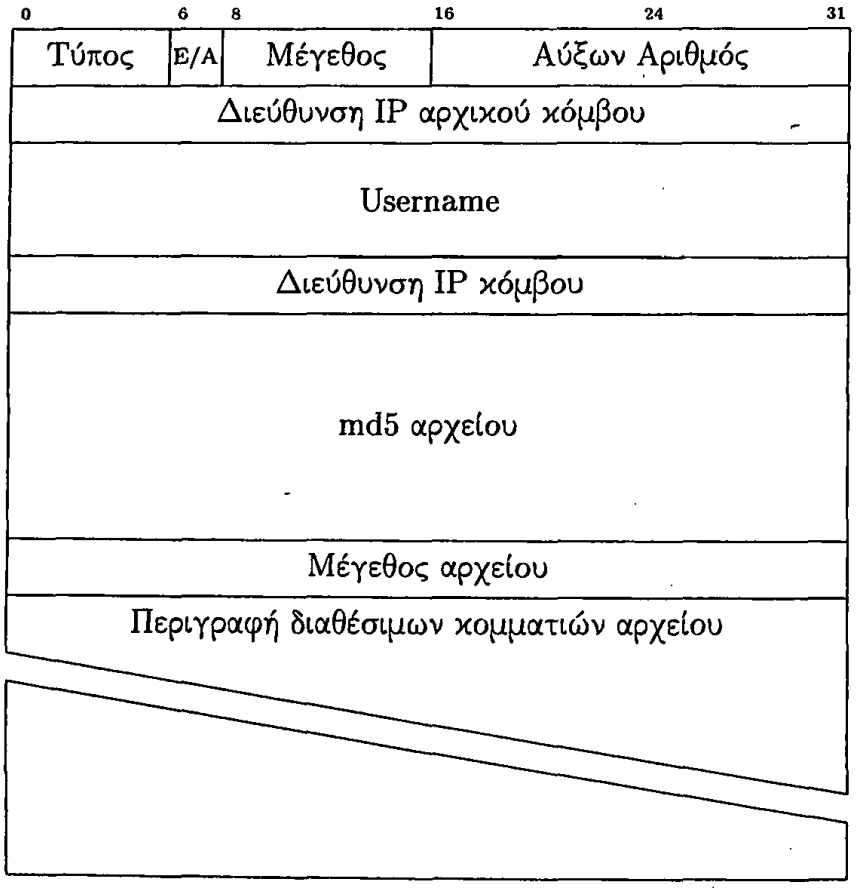
Όπως και στον άλλο τύπο αναζήτησης, ο κόμβος που θα λάβει ένα τέτοιο πακέτο θα ελέγξει αρχικά το search-ID για να διαπιστώσει αν έχει επεξεργαστεί την αναζήτηση προηγουμένως. Έπειτα θα κοιτάξει αν έχει κάποιο αρχείο με το συγκεκριμένο md5 και μέγεθος και θα απαντήσει στον κόμβο που έκανε την αναζήτηση. Τέλος, θα προωθήσει το πακέτο στους άμεσους γείτονές του αφού ελαττώσει κατά ένα το TTL, αν φυσικά αυτό δεν έχει μηδενιστεί. Αλλιώς δεν θα προωθήσει το πακέτο.

Το σημείο στο οποίο διαφοροποιείται σημαντικά η αναζήτηση αυτή από την προηγούμενη είναι η απάντηση του κόμβου που θα βρει το αρχείο με τα συγκεκριμένα χαρακτηριστικά (md5 και μέγεθος). Στην αναζήτηση με βάση το όνομα, μπορεί να

βρεθούν πολλά διαφορετικά αρχεία που να ταιριάζουν με τη συνθήκη και να επιλέξει ο χρήστης ποιο ή ποια θα κατεβάσει. Η αναζήτηση όμως με βάση το md5 γίνεται με σκοπό το άμεσο κατέβασμα του συγκεκριμένου αρχείου, οπότε είναι σκόπιμο στην απάντηση να αναφέρεται και το ποια κομμάτια του συγκεκριμένου αρχείου έχει ο κόμβος.

Το UDP πακέτο απάντησης, που φαίνεται στο σχήμα 5.6, περιέχει τα ακόλουθα δεδομένα:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (αναζήτηση με βάση το md5), με τιμή 000100 (=4).
- 2 bits που δηλώνουν το είδος του πακέτου (απάντηση), με τιμή 10 (=2).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 2 bytes τον αύξοντα αριθμό του πακέτου ερώτησης.
- 4 bytes που περιέχουν τη διεύθυνση IP του κόμβου που εκκίνησε την αναζήτηση, για να σχηματιστεί το search-ID.
- 8 bytes το όνομα του χρήστη (username) που κάνει την αναζήτηση.
- 4 bytes που περιέχουν τη διεύθυνση IP του συγκεκριμένου κόμβου που απαντάει.
- 16 bytes το md5 του ζητούμενου αρχείου.
- 4 bytes το μέγεθος του ζητούμενου αρχείου.
- την περιγραφή των κομματιών (μεταβλητό μέγεθος) του ζητούμενου αρχείου που έχει ο κόμβος. Για κάθε κομμάτι χρησιμοποιείται ένα bit, το οποίο έχει τιμή 1 αν ο κόμβος έχει το κομμάτι, ή τιμή 0 αν δεν το έχει. Εφόσον ξέρουμε το μέγεθος του αρχείου και το ορισμένο εξ αρχής μέγεθος του κάθε κομματιού, μπορούμε να υπολογίσουμε σε πόσα κομμάτια θα χωριστεί το αρχείο και άρα πόσα bits θα χρειαστούν για την περιγραφή τους. Αν τα απαιτούμενα bits δεν συμπληρώνουν ακριβώς τετράδες από bytes, προσθέτουμε στο τέλος όσα μηδενικά bits χρειάζονται.



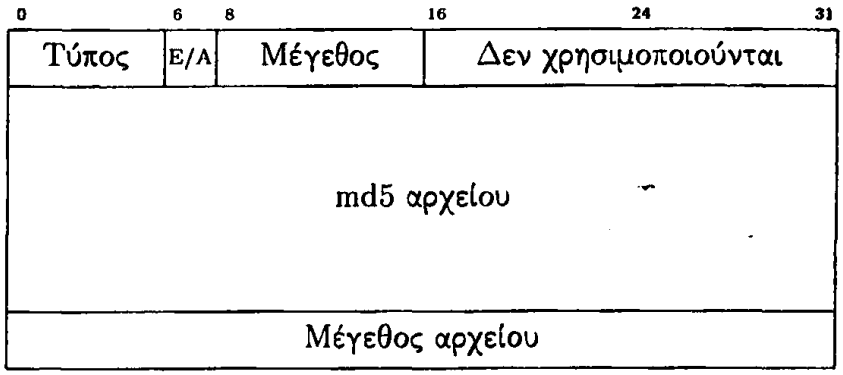
Σχήμα 5.6: Πακέτο απάντησης σε αναζήτηση με βάση το md5

5.5 Ερώτηση για κομμάτια αρχείου

Μετά από μία επιτυχή αναζήτηση με βάση το όνομα ενός αρχείου, ο κόμβος που την έκανε θα πρέπει να έχει δημιουργήσει, με βάση τις απαντήσεις που έλαβε, μία λίστα με αρχεία που ταιριάζουν με τη συνθήκη της αναζήτησης και τους κόμβους που έχουν το καθένα απ' αυτά. Αφού αποφασίσει ποιο αρχείο τον ενδιαφέρει και θέλει να κατεβάσει, πρέπει να βρει πώς κατανέμεται το αρχείο αυτό ανάμεσα στους κόμβους που δήλωσαν ότι το έχουν, δηλαδή να βρει ποιοι κόμβοι το έχουν ολόκληρο, ποιοι έχουν κάποια κομμάτια και ποια είναι αυτά.

Για να γίνει αυτό θα πρέπει να συνδεθεί με TCP σύνδεση στον καθένα απ' τους κόμβους που έχουν το αρχείο και να στείλει ένα πακέτο ερώτησης για το ποια κομμάτια έχει (σχήμα 5.7). Το πακέτο αυτό περιέχει απλά:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (ερώτηση για διαθέσιμα κομμάτια), με τιμή 001000 (=8).
- 2 bits που δηλώνουν το είδος του πακέτου (ερώτηση), με τιμή 01 (=1).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 2 bytes κενά, για συμπλήρωμα.
- 16 bytes το md5 του αρχείου που τον ενδιαφέρει.
- 4 bytes το μέγεθος του αρχείου.



Σχήμα 5.7: Πακέτο ερώτησης για διαθέσιμα κομμάτια αρχείου

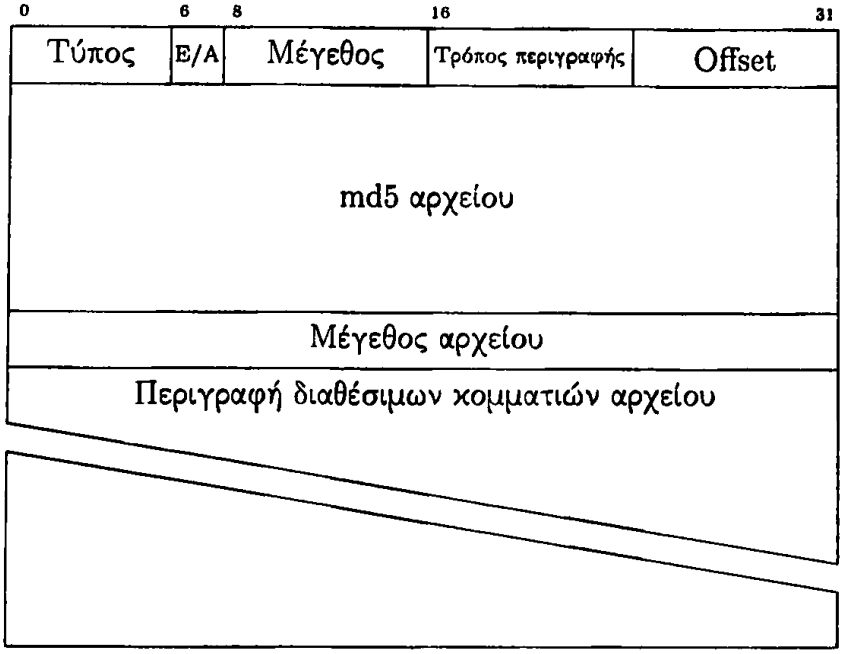
Ο κόμβος που θα λάβει το πακέτο αυτό αρχικά θα ελέγξει το αρχείο που διατηρεί με πληροφορίες για τα αρχεία που μοιράζεται για να δει ποια κομμάτια του αρχείου με τα συγκεκριμένα χαρακτηριστικά (md5 και μέγεθος) έχει. Κατόπιν θα κατασκευάσει

και θα αποστείλει στον κόμβο που έκανε την ερώτηση ένα TCP πακέτο (σχήμα 5.8) που θα περιέχει:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (ερώτηση για διαθέσιμα κομμάτια), με τιμή 001000 (=8).
- 2 bits που δηλώνουν το είδος του πακέτου (απάντηση), με τιμή 10 (=2).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 1 byte που υποδηλώνει τον τρόπο με τον οποίο περιγράφονται τα διαθέσιμα κομμάτια του αρχείου (με αριθμούς ή με bits). Αν το πεδίο έχει την τιμή 0, τότε η περιγραφή γίνεται με bits, αν έχει την τιμή 1 η περιγραφή γίνεται με αριθμούς
- 1 byte για "offset".
- 16 bytes το md5 του αρχείου.
- 4 bytes το μέγεθος του αρχείου.
- την περιγραφή των κομματιών (μεταβλητό μέγεθος) του ζητούμενου αρχείου που έχει ο κόμβος.

Θα πρέπει να εξηγήσουμε ότι η περιγραφή των διαθέσιμων κομματιών του αρχείου δεν γίνεται απαραίτητα με τον τρόπο που περιγράψαμε στο πακέτο απάντησης της ενότητας 5.4 (σχήμα 5.6), δηλαδή με ένα bit για κάθε κομμάτι. Αν το αρχείο είναι μεγάλο και ο κόμβος διαθέτει λίγα κομμάτια σε σχέση με το μέγεθός του, τότε μία περιγραφή με bits θα χρησιμοποιούσε πολλά bits που τα περισσότερα θα ήταν 0. Οπότε, για να μικρύνει το μέγεθος του πακέτου και να βελτιωθεί η επικοινωνία, δίνουμε τη δυνατότητα σε έναν κόμβο να μπορεί να αναφέρει ακριβώς τα κομμάτια που διαθέτει, π.χ. το κομμάτι υπ' αριθμόν 12.

Έτσι, αν στο πεδίο "Τρόπος περιγραφής" είναι δηλωμένο ότι θα γίνεται περιγραφή με bits, τότε ισχύουν αυτά που αναφέραμε στην ενότητα 5.4 σχετικά με τον συγκεκριμένο τρόπο περιγραφής. Αν όμως είναι δηλωμένο ότι θα γίνει περιγραφή με αριθμούς, τότε για κάθε κομμάτι του αρχείου χρησιμοποιείται ένας ακέραιος αριθμός 2 bytes που δηλώνει τον αύξοντα αριθμό του κομματιού. Αν δεν συμπληρώνονται ακριβώς οι τετράδες bytes του πακέτου, μπορεί να χρειαστεί να συμπληρώσουμε 2 κενά bytes (με τιμή 0) στο τέλος του πακέτου, οπότε τοποθετούμε την τιμή 1 στο πεδίο "Offset" για να το δηλώσουμε. Αλλιώς το πεδίο έχει τιμή 0.



Σχήμα 5.8: Πακέτο απάντησης για διαθέσιμα κομμάτια αρχείου

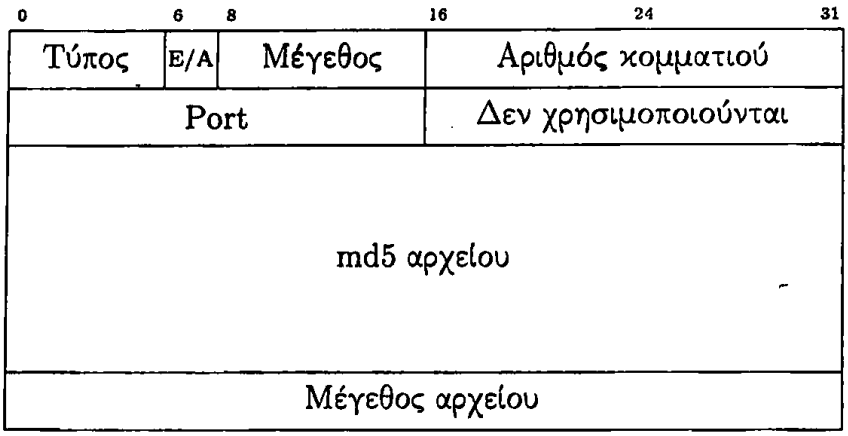
5.6 Αίτηση για κατέβασμα αρχείου

Αφού ένας κόμβος κάνει τις απαραίτητες ερωτήσεις για τα διαθέσιμα κομμάτια του αρχείου που θέλει να κατεβάσει σε όλους τους κόμβους που το έχουν, είναι σε θέση να καταρτίσει μία λίστα από κομμάτια και ποιοι κόμβοι έχουν το καθένα και άρα μπορεί να επιλέξει την πολιτική με την οποία θα κατεβάσει το αρχείο, δηλαδή από ποιον κόμβο θα ζητήσει το κάθε κομμάτι ώστε να συμπληρώσει το αρχείο.

Όταν θελήσει να ζητήσει ένα κομμάτι από έναν κόμβο, του στέλνει το TCP πακέτο του σχήματος 5.9 που περιέχει:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (αίτηση για κατέβασμα), με τιμή 010000 (=16).
- 2 bits που δηλώνουν το είδος του πακέτου (ερώτηση), με τιμή 01 (=1).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 2 bytes που δηλώνουν τον αριθμό του κομματιού το οποίο ζητείται.
- 2 bytes για το TCP port στο οποίο περιμένει να δεχτεί το αρχείο.
- 2 bytes κενά, για συμπλήρωμα.

- 16 bytes το md5 του ζητούμενου αρχείου.
- 4 bytes το μέγεθος του ζητούμενου αρχείου.



Σχήμα 5.9: Πακέτο αίτησης για κατέβασμα αρχείου

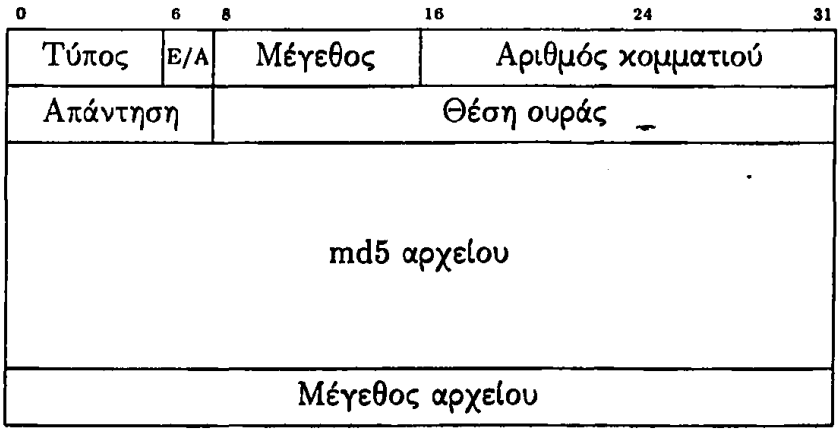
Ο κόμβος που θα λάβει ένα τέτοιο πακέτο, θα ελέγξει αν μπορεί να εξυπηρετήσει την αίτηση αμέσως, δηλαδή αν δεν στέλνει ήδη αρχεία σε έναν αριθμό χρηστών. Αλλιώς, θα ελέγξει αν η ουρά αναμονής έχει κενές θέσεις για να τοποθετήσει εκεί την αίτηση ή να την απορρίψει. Θα απαντήσει με ένα πακέτο TCP (σχήμα 5.10) που θα περιέχει:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (αίτηση για κατέβασμα), με τιμή 010000 (=16).
- 2 bits που δηλώνουν το είδος του πακέτου (απάντηση), με τιμή 10 (=2).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 2 bytes που δηλώνουν τον αριθμό του κομματιού το οποίο ζητείται.
- 1 byte για την απάντησή του, δηλαδή "ναι" (τιμή 0),"όχι" (τιμή 1) ή "τοποθετήθηκε στη ουρά" (τιμή 2).
- 3 bytes για τη θέση της ουράς στην οποία τοποθετήθηκε η αίτηση.
- 16 bytes το md5 του ζητούμενου αρχείου.
- 4 bytes το μέγεθος του αρχείου.

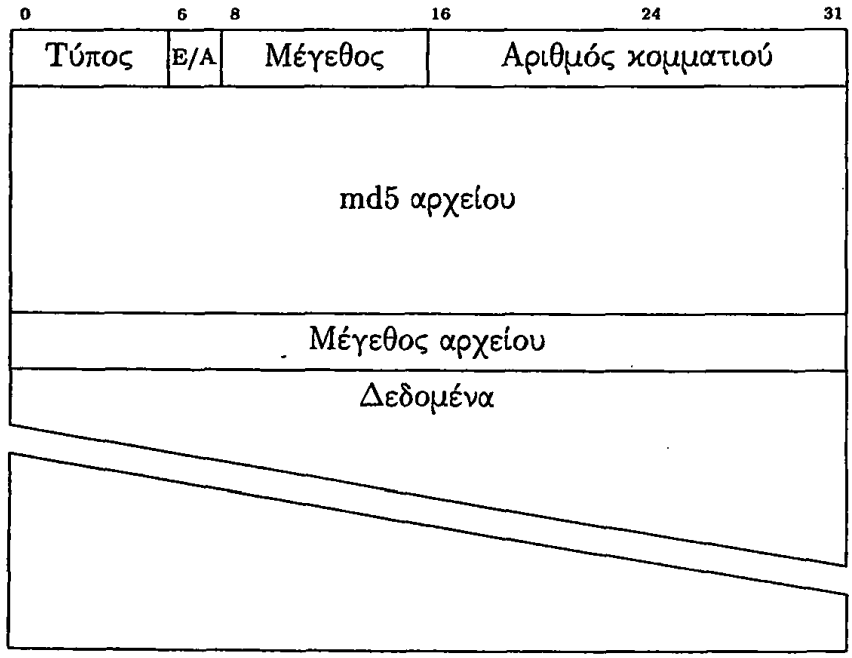
5.7 Μεταφορά αρχείου

Τέλος, όταν κάποιος κόμβος θέλει να στείλει σε έναν άλλο κόμβο ένα κομμάτι ενός αρχείου το κάνει με τη βοήθεια του πακέτου του σχήματος 5.11, το οποίο έχει για δεδομένα:

- 6 bits που δηλώνουν τον τύπο της επικοινωνίας (μεταφορά αρχείου), με τιμή 100000 (=32).
- 2 bits που δηλώνουν το είδος του πακέτου, με τιμή 01 (=1).
- 1 byte το μέγεθος του πακέτου σε τετράδες bytes.
- 2 bytes τον αριθμό του ζητούμενου κομματιού.
- 16 bytes το md5 του αρχείου.
- 4 bytes το μέγεθος του αρχείου.
- τα δεδομένα του κομματιού (σταθερού μεγέθους).



Σχήμα 5.10: Πακέτο απάντησης σε αίτηση για κατέβασμα αρχείου



Σχήμα 5.11: Πακέτο μεταφοράς αρχείου


ΠΑΝΕΠΙΣΤΗΜΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ ΙΩΑΝΝΙΝΩΝ

Κεφάλαιο 6

Προβλήματα - Μελλοντική δουλειά

Στο κεφάλαιο αυτό θα μελετήσουμε κάποιες ενέργειες που πρέπει να γίνουν έτσι ώστε να διευκολυνθεί η υλοποίηση του συστήματος που σχεδιάσαμε, αλλά θα θέσουμε και τις βάσεις για μελλοντική δουλειά.

Το πρώτο σημείο στο οποίο πρέπει να εστιάσουμε είναι η διαδικασία εκκίνησης (boot process). Είπαμε ότι θέλουμε να βρίσκει έναν κόμβο που είναι ήδη συνδεδεμένος στο δίκτυο του συστήματός μας έτσι ώστε ένας νέος κόμβος να μπορεί να συνδεθεί πάνω του. Ακόμα καλύτερο θα ήταν να βρίσκει αρκετούς κόμβους που είναι συνδεδεμένοι και μετά ο νέος κόμβος να επιλέγει τυχαία σε ποιον απ' όλους θα συνδεθεί, έτσι ώστε να υπάρχει μια σχετική ισοκατανομή των νέων κόμβων πάνω στους υπάρχοντες.

Για τις ανάγκες της υλοποίησης βέβαια, η boot process μπορεί να ανατικατασκαθεί από μία λίστα με γνωστούς κόμβους (π.χ. κάποιοι φίλοι του χρήστη) ή μια σελίδα στο web που να αναφέρει τα στοιχεία διάφορων κόμβων που λειτουργούν, όπως γίνεται στα ήδη υπάρχοντα συστήματα peer-to-peer χωρίς εξυπηρέτη (FreeNet, OverNet). Πάντως αξίζει να γίνει κάποια έρευνα στον τομέα αυτον, καθώς δεν έχει δοθεί λύση από κανένα άλλο peer-to-peer σύστημα.

Ένα πρόβλημα που προέκυψε κατά το σχεδιασμό του συστήματος και δεν στάθηκε δυνατό να αντιμετωπιστεί είναι το κατά πόσον μπορεί ένας κόμβος να εξαπατήσει έναν άλλο και κατά πόσο μπορεί να διαδοθεί μέσω του συστήματος ένα "λανθασμένο" αρχείο. Είναι δηλαδή πιθανόν κατά τη διάρκεια της μεταφοράς ενός κομματιού ενός αρχείου να υπάρξει ένα πρόβλημα στο δίκτυο ή ένας κακόβουλος χρήστης και να αλλοιωθούν με κάποιον τρόπο τα δεδομένα του κομματιού αυτού. Για να το διαπιστώσει αυτό ο κόμβος που δέχεται το κομμάτι θα πρέπει να περιμένει μέχρι να κατεβεί ολόκληρο το αρχείο (όλα τα κομμάτια του), να υπολογίσει το md5 του και να δει αν ταιριάζει με αυτό που ήδη έχει μάθει από τον κόμβο που έστελνε τα

δεδομένα. Αν έχει αλλοιωθεί έστω και ένα bit του αρχείου, τα δύο md5 θα είναι διαφορετικά.

Ας υποθέσουμε τώρα ότι μέχρι να κατεβεί ολόκληρο το αρχείο, ο κόμβος έχει ήδη στείλει το αλλοιωμένο κομμάτι και σε άλλους κόμβους, έχει αρχίσει δηλαδή να διαδίδεται στο δίκτυο κάτι που δεν είναι σωστό. Ακόμα, όταν ο κόμβος θα διαπιστώσει ότι το αρχείο που έλαβε είναι αλλοιωμένο, δεν θα είναι σε θέση να διαπιστώσει σε ποιο κομμάτι υπήρξε η αλλοίωση και από ποιον κόμβο το κατέβασε. Πρέπει δηλαδή να μπορεί ένας κόμβος να ελέγξει το αρχείο κομμάτι-κομμάτι για να διαπιστώσει που υπάρχει το πρόβλημα και να ξανακατεβάσει μόνο ένα συγκεκριμένο κομμάτι και όχι όλο το αρχείο.

Μία λύση στο πρόβλημα αυτό που χρησιμοποιεί το eDonkey, το οποίο περιγράψαμε στην ενότητα 2.2, είναι να υπολογίζεται το md4 *καθενός* κομματιού του αρχείου και έπειτα το md4 του συνολικού αρχείου. Έτσι, μπορεί κάθε κόμβος να ελέγχει κάθε κομμάτι που κατεβάζει, τη στιγμή που θα κατέβει ολόκληρο, χωρίς να περιμένει να κατέβει όλο το αρχείο.

Ένα άλλο ζήτημα που πρέπει να λυθεί έτσι ώστε να υλοποιηθεί σωστά το σύστημα είναι να καθοριστούν οι παράμετροι του δικτύου μας, δηλαδή τα μεγέθη από τις διάφορες δομές που διατηρούν οι κόμβοι (λίστες γειτόνων, ουρά αναμονής), διάφορα χρονικά όρια (π.χ. για το πότε μια αναζήτηση θεωρείται ανεπιτυχής) καθώς και διάφορες παράμετροι των πακέτων που στέλνονται (TTL, αριθμός γειτόνων N). Έχουμε αναφέρει ότι οι δομές αυτές αφ' ενός δεν μπορούν να μεγαλώνουν επ' άπειρον γιατί θα έχουμε πρόβλημα με τη διαθέσιμη μνήμη και αφ' ετέρου δεν είναι δυνατόν οι λίστες με τους γείτονες να αδειάσουν κάτω από ένα όριο γιατί ο κόμβος δεν θα μπορεί να κάνει αποτελεσματικές και γρήγορες αναζητήσεις.

Ο βέλτιστος τρόπος να καθοριστούν τα μεγέθη αυτά είναι να δημιουργηθεί και να εκτελεστεί μία προσομοίωση του συστήματος που σχεδιάσαμε, στην οποία θα δοκιμαστούν διάφορες τιμές για το καθένα απ' αυτά. Έτσι, αφού γίνουν οι απαραίτητες μετρήσεις για την αποδοτικότητα του συστήματος, δηλαδή την αποδοτικότητα των αναζητήσεων και των μεταφορών αρχείων, και οι κατάλληλες συγκρίσεις μεταξύ τους, να είμαστε σε θέση να επιλέξουμε τον βέλτιστο συνδυασμό παραμέτρων.

Ακόμα, αφού οριστούν τα μεγέθη αυτά, θα είναι λογικό να δούμε ποια θα είναι η βέλτιστη πολιτική κατεβάσματος ενός αρχείου, δηλαδή με ποιον τρόπο πρέπει ένας κόμβος να ζητάει τα κομμάτια ενός αρχείου που θέλει να κατεβάσει από τους διάφορους κόμβους που τα διαθέτουν. Πιστεύουμε ότι κι εδώ θα μας βοηθήσει η προσομοίωση του συστήματος.

Τέλος, στο σύστημά μας μπορούν να προστεθούν επιπλέον χαρακτηριστικά, όπως οι ομάδες χρηστών που αναφέραμε ή η δυνατότητα να προσπελάσει κάποιος όλα τα αρχεία που μοιράζεται ένας κόμβος. Να δίνει για παράδειγμα το όνομα ενός

χρήστη, να γίνεται μία αναζήτηση για να βρεθεί ο κόμβος στον οποίο βρίσκεται ο χρήστης αυτός και μετά να γίνεται μία TCP σύνδεση μεταξύ των δύο κόμβων για να εμφανιστούν όλες οι πληροφορίες για τα αρχεία που ο χρήστης αυτός μοιράζεται.

Το σύστημά μας έχει σχεδιαστεί με τέτοιο τρόπο ώστε μικρές προσθήκες ή βελτιώσεις, όπως αυτές που αναφέραμε να μπορούν να γίνουν σχετικά εύκολα και χωρίς να χρειαστούν αλλαγές στις ήδη υπάρχουσες λειτουργίες του.

Παράρτημα Α

Οι hash-functions md4 και md5

Οι δύο αυτοί αλγόριθμοι δημιουργήθηκαν από τον Ron Rivest, πρώτα ο md4 και ως βελείωσή του ο md5. Πρόκειται για δύο hash-functions που επιδρούν πάνω σε ακολουθίες από bits οποιουδήποτε μεγέθους και παράγουν ένα αποτέλεσμα μήκους 128 bits τέτοιο ώστε να ισχύουν οι εξής ιδιότητες:

1. Δύο διαφορετικές ακολουθίες να δίνουν διαφορετικά αποτελέσματα (Αν και υπάρχει ελάχιστη πιθανότητα αυτό να συμβαίνει).

2. Να μην είναι δυνατόν δοθέντος του αποτελέσματος να βρθεί η ακολουθία που το παρήγαγε.

Θα περιγράψουμε παρακάτω τον md5 και θα αναφέρουμε τις διαφορές του από τον md4.

Αρχικά, τοποθετείται ένα συμπλήρωμα στην ακολουθία των bits της μορφής 1000...0 (δηλαδή ένα bit με τιμή 1 και τα υπόλοιπα με τιμή 0) έτσι ώστε να έχει συνολικό μήκος 448 *modulo* 512 και στο τέλος προστίθενται 64 bits που αντιπροσωπεύουν το αρχικό μέγεθος της ακολουθίας. Κατόπιν, η ακολουθία χωρίζεται σε τμήματα των 512 bits τα οποία υπόκεινται στην ακόλουθη επεξεργασία.

Η επεξεργασία αυτή είναι ουσιαστικά μια συμπίεση των bits της αρχικής ακολουθίας. Για το κάθε κομμάτι των 512 bits γίνονται τέσσερεις γύροι επεξεργασίας, με παρόμοια δομή αλλά που χρησιμοποιούν διαφορετικές λογικές συναρτήσεις πάνω στα bits του κομματιού. Ο κάθε γύρος έχει 16 βήματα και ουσιαστικά ανακατεύει τα bits του κομματιού με τυχαίο τρόπο, προσθέτοντας και κάποιες αρχικές σταθερές. Στο κάθε βήμα από τα 16, προστίθεται και το αποτέλεσμα του προηγούμενου βήματος.

Τελικά, αφού γίνει η παραπάνω επεξεργασία για κάθε κομμάτι 512 bits της αρχικής ακολουθίας και μετά από διάφορες προσθέσεις, παράγεται μία ακολουθία

από 128 bits η οποία είναι το αποτέλεσμα του αλγορίθμου.

Ο md4 διαφέρει από τον md5 στα εξής σημεία:

- Έχει τρεις γύρους των 16 βημάτων αντί για 4, και συνεπώς τρεις λογικές συναρτήσεις.
- Το αποτέλεσμα ενός βήματος δεν προστίθεται στον υπολογισμό του επόμενου.
- Χρησιμοποιούνται οι ίδιες σταθερές για καθένα από τα 16 βήματα κάθε γύρου, ενώ στον md5 χρησιμοποιούνται διφορετικές για κάθε βήμα.

Θεωρητικά, ο md5 είναι πιο ασφαλής, δηλαδή υπάρχει μικρότερη πιθανότητα να βρεθούν δύο ακολουθίες που να δίνουν το αποτέλεσμα και επίσης είναι πολύ πιο δύσκολο να βρεθεί ποια ακολουθία παρήγαγε ένα δοθέν αποτέλεσμα.

Συμπερασματικά μπορούμε να πούμε ότι οι hash-functions αυτές μπορούν να χαρακτηρίσουν με μοναδικό τρόπο ένα αρχείο και να πιστοποιήσουν την αυθεντικότητά του και για το λόγο αυτό χρησιμοποιούνται συχνά στα συστήματα peer-to-peer για ανταλλαγή αρχείων.

Βιβλιογραφία

[1] edonkey 2000. http://www.edonkey2000.com.

[2] The freenet project. http://www.freenetproject.org.

[3] Gnutella. http://www.gnutella.com.

[4] Kazaa media desktop. http://www.kazaa.com.

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

[6] Overnet. http://www.overnet.com.

[7] Ian Clarke, Scott G. Miller, Theodore W. Hong, Oskar Sandberg, and Brandon Wiley. Protecting free expression online with freenet. *IEEE Internet Computing*, pages 40–49, Jan.-Feb. 2002.

[8] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, pages 311–320, Berkeley, CA, USA, July 2000.

[9] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, USA, Mar. 2002.

[10] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings ACM SIGCOMM 2001*, San Diego, CA, USA, Aug. 2001.

[11] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings ACM SIGCOMM 2001*, San Diego, CA, USA, Aug. 2001.