ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δημιουργία Torrent Client

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γιαλαμάς Νικόλαος-Άδωνις

ΕΠΙΒΛΕΠΟΝΤΕΣ

Σταμούλης Γεώργιος

Θάνος Γεώργιος

Τσαλαπάτα Χαρίκλεια

Βόλος, 2020

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

A Torrent Client Application

DIPLOMA THESIS

Gialamas Nikolaos-Adonis

SUPERVISORS

Stamoulis Georgios

Thanos Georgios

Tsalapata Chariklia

Volos, 2020

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών

Γιαλαμάς Νικόλαος-Άδωνις
8/10/2020

ΠΕΡΙΛΗΨΗ

Οι ταχύτητες με τις οποίες οι online χρήστες μπορούν να “κατεβάσουν” αρχεία μέσω των δικτύων αυξάνονται συνεχώς. Αυτό ευνοεί την συνεχή ανάπτυξη στον τομέα των πρωτοκόλλων διαμοιρασμού αρχείων, ώστε να υπάρξει όσο το δυνατόν καλύτερη και πιο αποτελεσματική χρήση των δικτύων. Το BitTorrent είναι ένα πρωτόκολλο που βοηθά στον διαμοιρασμό μεγάλων αρχείων μεταξύ πλήθους χρηστών, χωρίς να βασίζεται σε έναν server, αλλά στην διάθεση του αρχείου μεταξύ των συμμετεχώντων από όλους προς όλους, δομώντας ένα συνεργατικό δίκτυο. Στην διπλωματική αυτή εργασία θα παρουσιαστεί αναλυτικά το πώς δουλεύει το πρωτόκολλο BitTorrent, χρησιμοποιώντας παράλληλα παραδείγματα από την εφαρμογή ενός torrent client που αναπτύχθηκε κατά την συγγραφή αυτής της διπλωματικής.

ABSTRACT

The speed with which online users can download files over networks is constantly increasing. This favors the continuous development in the field of file sharing protocols, so that there is the best and most efficient use of networks. BitTorrent is a protool that helps share large files amongst a large number of users, without relying on a server, but making the file available to participants from all to all, building a collaborative network. In this thesis will be presented in detail how BitTorrent protocol works, and alongside there will be examples from the application of a torrent client that was developed during the writing of this thesis.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Ενώ η αρχιτεκτονική server – client είναι μια σταθερή και αξιόπιστη λύση για τον διαμοιρασμό αρχείων, η αρχιτεκτονική peer-to-peer γίνεται όλο και πιο δημοφιλής. Η αρχιτεκτονική client – server είναι ένα υπολογιστικό μοντέλο όπου ο server είναι υπεύθυνος για την διαχείριση των πόρων που βρίσκονται διαθέσιμοι, καθώς και την παροχή υπηρεσιών προς τους clients, όπως είναι η διανομή αρχείων. Η αρχιτεκτονική peer-to-peer, ωστόσο, βασίζεται στην ισοτιμία ανάμεσα στους peers. Είναι μια αρχιτεκτονική κατανεμημένης εφαρμογής, όπου οι peers έχουν ίσα προνόμια και ισοδύναμη συμμετοχή στην εφαρμογή.

Συγκρινόμενη με την πιο συχνά συναπαντώμενη αρχιτεκτονική server – client, η αρχιτεκτονική peer-to-peer έχει αρκετά πλεονεκτήματα, όπως είναι η αυξημένη ευρωστία και ο δίκαιος διαμοιρασμός των πόρων, δηλαδή του bandwidth, της υπολογιστικής ισχύς και του αποθηκευτικού χώρου μεταξύ των peers. Αυτά τα χαρακτηριστικά καθιστούν την αρχιτεκτονική peer-to-peer ιδανική στον διαμοιρασμό αρχείων, και δη των μεγάλων αρχείων. Ένας παράδειγμα που αυτή η αρχιτεκτονική έχει φανεί πολύ χρήσιμη και αποτελεσματική εναλλακτική ως προς την server – client είναι το BitTorrent.

1.1 Ιστορική Εξέλιξη

Τα συστήματα peer-to-peer είχαν χρησιμοποιηθεί στο παρελθόν σε διάφορες εφαρμογές. Αυτές επέτρεψαν σε εκατομμύρια χρήστες, μέσω του ίντερνετ, να συνδεθούν μεταξύ τους και να φτιάξουν ομάδες, όπου μέσω της συνεργασίας τους κατάφεραν να δημιουργήσουν μηχανές αναζήτησης, συστήματα αρχείων και εικονικούς υπερυπολογιστές . Το peer-to-peer, παρ' όλ' αυτά άρχισε να γίνεται γνωστό μέσα από εφαρμογές διαμοιρασμού αρχείων, όπως η εφαρμογή Napster, η οποία κυκλοφόρησε αρχικά το 1999 [1].

Ο δημιουργός του πρωτοκόλλου BitTorrent, Bram Cohen, δούλευε στην εταιρία MojoNation. Μέσω του MojoNation χρήστες μπορούσαν να χωρίσουν αρχεία σε κρυπτογραφημένα σε μικρότερα κομμάτια (chunks) και να τα διαμοιράσουν σε υπόλοιπους χρήστες που χρησιμοποιούσαν το ίδιο λογισμικό. Εάν κάποιος ήθελε αυτό το αρχείο, μπορούσε να το κατεβάσει ταυτόχρονα από πολλούς χρήστες [2].

Βασισμένος στην ιδέα αυτή ο Bram Cohen, καθώς άλλα προγράμματα της εποχής, όπως το KaZaA, ήταν πολύ χρονοβόρα στο κατέβασμα ενός μεγάλου αρχείου διότι το αρχείο συνήθως ερχόταν από μια πηγή, εμπνεύστηκε το πρωτόκολλο BitTorrent. Θεώρησε πως μια τέτοια τεχνική είναι τέλεια για μεγάλα αρχεία αφού ο χρήστης θα κατεβάζει το εκάστοτε αρχείο από πολλούς διαφορετικούς διαμοιραστές (peers), αυτούς που έχουν ήδη το αρχείο ή κομμάτια του αρχείου. Έτσι, όσο πιο δημοφιλές είναι ένα αρχείο, τόσο περισσότερους peers θα έχει, μειώνοντας κατά πολύ τον χρόνο download [2].

Το πρωτόκολλο BitTorrent σχεδιάστηκε τον Απρίλιο του 2001 από τον Bram Cohen και κυκλοφόρησε την πρώτη του έκδοση στις 2 Ιουλίου 2001. BitTorrent ονομάστηκε επίσης και το λογισμικό που χρησιμοποιούσε το πρωτόκολλο. Από την κυκλοφορία του, το BitTorrent είναι ένα από τα πιο διαδεδομένα πρωτόκολλα για μεταφορά μεγάλων αρχείων. Τα δίκτυα Peer-To-Peer υπολογίστηκε ότι, τον Φεβρουάριο του 2009, σχετίζονταν με το 43% έως 70% της συνολικής διακίνησης δεδομένων στο διαδίκτυο, ανάλογα με την περιοχή. Τον Φεβρουάριο του 2013, το BitTorrent υπολογίστηκε ότι ήταν υπεύθυνο για το 3.35% του παγκόσμιου bandwidth, την ίδια στιγμή που το συνολικό bandwidth που χρησιμοποιούνταν για διαμοιρασμό

αρχείων ήταν 6%.Το 2019 υπολογίζεται ότι το πρωτόκολλο ήταν υπεύθυνο για το 2.46% του συνολικού downstream traffic και του 27.58% του upstream traffic [2, 3, 4, 5].

1.2 Αντικείμενο της Διπλωματικής

Σκοπός της διπλωματικής αυτής εργασίας είναι να παρουσιάσει στον αναγνώστη μια πλήρη εικόνα του πρωτοκόλλου BitTorrent, καθώς και να παρουσιάσει στον χρήστη τον τρόπο που αυτό δουλεύει μέσω της εφαρμογής Torrent Client που αναπτύχθηκε κατά τη διάρκεια της συγγραφής αυτής της διπλωματικής εργασίας.

1.3 Οργάνωση του τόμου

Στο κεφάλαιο 2 παρουσιάζονται απαραίτητες έννοιες και βασικά εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Στο κεφάλαιο 3 αναλύεται το πρωτόκολλο tracker, ενώ στο κεφάλαιο 4 αναλύεται το πρωτόκολλο peer. Στο κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα.

ΚΕΦΑΛΑΙΟ 2

ΕΙΣΑΓΩΓΙΚΑ ΣΤΟΙΧΕΙΑ – ΕΝΝΟΙΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ

Πριν προχωρήσουμε με την αναλυτική περιγραφή της λειτουργίας του πρωτοκόλλου BitTorrent, θα γίνει μια εκτενής αναφορά στα βασικά στοιχεία πάνω στα οποία έχει χτιστεί, καθώς και θα γίνει μια εισαγωγή στο πρωτόκολλο και στις βασικές αρχές που καθιστούν το πρωτόκολλο λειτουργικό.

2.1 Δικτύωση Peer-to-Peer

Τα δίκτυα peer-to-peer (P2P) περιγράφονται ως μοντέλα δικτύωσης όπου ο κάθε συμμετέχων (peer) έχει τις ίδιες δυνατότητες με τους υπόλοιπους. Κάθε peer από τους συμμετέχοντες μπορεί να ξεκινήσει μια συνεδρία επικοινωνίας. Στόχος του δικτύου αυτού είναι, μέσω του διαμοιρασμού πόρων, όπως είναι η επεξεργαστική ισχύς, ο αποθηκευτικός χόρος και το εύρος ζώνης (bandwidth) να επιτύχουν έναν σκοπό. Κάθε συμμετέχων είναι και καταναλωτής και προμηθευτής, δηλαδή μπορεί να προσφέρει τους πόρους του αλλά και να ζητήσει και να χρησιμοποιήσει τους πόρους των υπόλοιπων συμμετεχόντων [6].

Η χρήση των δικτύων peer-to-peer επεκτείνεται σε πολλούς τομείς, όπως είναι ο διαμοιρασμός αρχείων, οι υπηρεσίες video και audio streaming, τα ψηφιακά κρυπτονομίσματα, μηχανές αναζήτησης και πολλά άλλα. Κύριο χαρακτηριστικό των δικτύων peer-to-peer είναι η σύνδεση χρηστών από όλο τον κόσμο χωρίς αυτή να υπάγεται σε περιορισμούς και λογοκρισία, βοηθώντας στην ελεύθερη διακίνηση ιδεών και αρχείων [1].

Η διακίνηση μεγάλων αρχείων γίνεται συχνά χρησιμοποιώντας δίκτυα peer-to-peer. Ο πιο διαδεδομένος τρόπος είναι η χρήση του πρωτοκόλλου BitTorrent. Πολλές εταιρίες ή projects προτιμούν τα δίκτυα peer-to-peer καθώς η χρήση τους έρχεται με πολλά πλεονεκτήματα:

- **Εύκολος διαμοιρασμός αρχείων**: ένα δίκτυο P2P μπορεί να διαμοιράσει με μεγάλη ταχύτητα αρχεία, ακόμα και σε μεγάλες αποστάσεις.
- **Μειωμένο κόστος**: Είτε δεν χρειάζονται χρήματα για έναν server, είτε το κόστος αυτό είναι πολύ μικρότερο σε σύγκριση με έναν τυπικό server του μοντέλου client-server.
- **Προσαρμοστικότητα**: Ένας νέος peer μπορεί να συμμετάσχει εύκολα στο δίκτυο. Αυτή η προσαρμοστικότητα κάνει τα δίκτυα peer-to-peer πιο ευέλεικτα από ένα δίκτυο client-server.
- **Αξιοπιστία**: Σε ένα δίκτυο peer-to-peer, αν ένας peer πάθει κάποια βλάβη και αποχωρήσει, το δίκτυο θα παραμείνει πιθανότατα λειτουργικό.
- **Υψηλή απόδοση**: Όσο περισσότεροι clients συμμετέχουν στο δίκτυο peer-to-peer, τόσο πιο αποτελεσματικό είναι, καθώς κάθε client συμμετέχει στον διαμοιρασμό του αρχείου. Αντίθετα, σε ένα δίκτυο client-server η απόδοση πέφτει όσο συνδέονται περισσότεροι clients.

2.1.1 Τοπολογίες Peer-to-Peer

Υπάρχουν τρεις διαφορετικές κατηγορίες δικτύων peer-to-peer [1, 7]:

- **Συγκεντρωτικά P2P δίκτυα**:
  Γνωστά και ως “πρώτης γενιάς P2P δίκτυα”, λειτουργούν έχοντας σαν κεντρική οντότητα έναν Index Server στον οποίο αποθηκεύονται οι πληροφορίες για τα περιεχόμενα των καταλόγων που οι συμμετέχοντες επιθυμούν να μοιράζονται. Χρησιμοποιώντας ένα πρόγραμμα-πελάτη, οι χρήστες συνδέονται στον Index Server και αναζητούν τα αρχεία που ψάχνουν. Μετά τον εντοπισμό του αρχείου, ο Index

Server στέλνει τα στοιχεία των peers που έχουν το αρχείο. Ο client επιλέγει τον peer που θέλει να συνεργαστεί και συνδέεται με αυτόν για την μεταφορά του αρχείου (Figure 1).

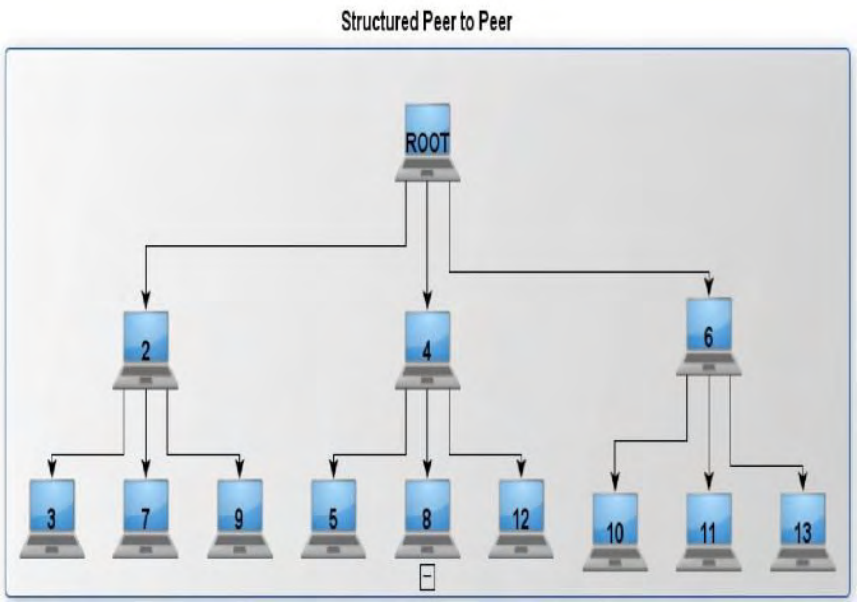

Figure 1: Συγκεντρωτικό δίκτυο peer-to-peer

- **Αποκεντρωτικά P2P δίκτυα**:

  Εδώ κάθε συμμετέχων είναι ταυτόχρονα και client και server. Κατά την αναζήτηση ενός αρχείου, αντί ο client να κάνει την αναζήτησή του σε κάποια κεντρική οντότητα, μπαίνει σε λειτουργία ένας μηχανισμός πλημμύρας (flooding). Το αίτημα περνά στους γειτονικούς peers του client, οι οποίοι με τη σειρά τους το στέλνουν στους δικούς τους γειτονικούς peers. Αυτή η διαδικασία συνεχίζεται μέχρι να εντοπιστεί το ζητούμενο αρχείο. Οι peers που έχουν το ζητούμενο αρχείο, ακολουθώντας την αντίστροφη πορεία, επικοινωνούν με τον client και τον ενημερώνουν. Ο client, αφού λάβει απάντηση από τους peers, επιλέγει αυτόν που θέλει για να του παρέχει το αρχείο. Τα αποκεντρωτικά peer-to-peer δίκτυα είναι πιο αξιόπιστα από τα

συγκεντρωτικά, καθώς δεν υπάρχει κάποια κεντρική οντότητα, με σφάλμα της οποίας καταρρέει όλο το δίκτυο. Παρ' όλ' αυτά, ο μηχανισμός πλημμύρας προκαλεί μεγάλη κίνηση στο δίκτυο, η οποία ρίχνει και την απόδοσή του (Figure 2).

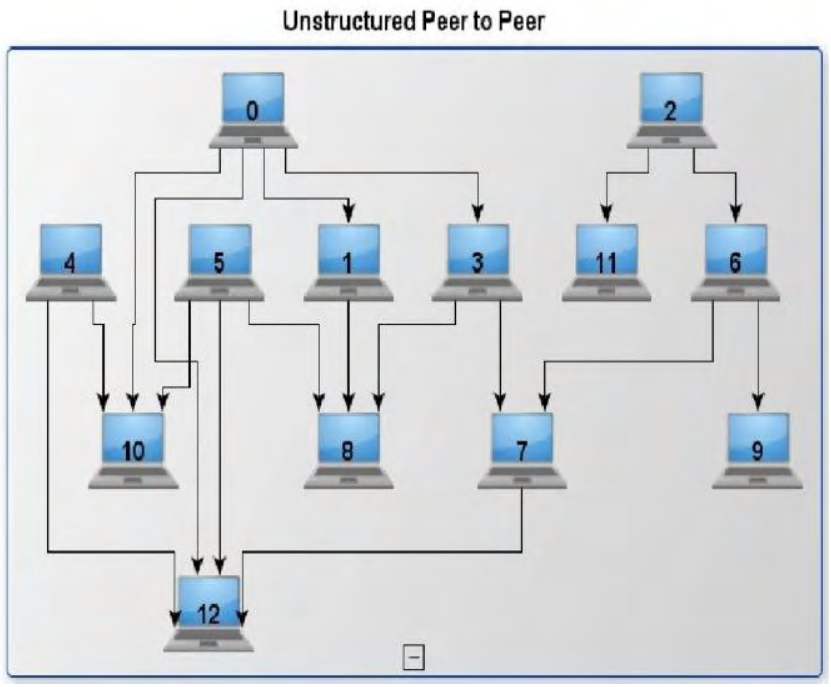

Figure 2: Αποκεντρωτικό δίκτυο peer-to-peer

- **Υβριδικά P2P δίκτυα**:

  Τα υβριδικά peer-to-peer δίκτυα αποτελούνται από έναν συνδυασμό των μοντέλων peer-to-peer και client-server. Διατηρώντας τα πλεονεκτήματα των αποκεντρωτικών και των συγκεντρωτικών δικτύων, διατηρούν την αποτελεσματικότητα της αναζήτησης των δεύτερων και την αξιοπιστία των πρώτων. Υπάρχει ποικιλία σε υβριδικά δίκτυα, αλλά σε όλα υπάρχει ένας κεντρικός server που βοηθά τους peers να βρουν ο ένας τον άλλον. Το πρωτόκολλο BitTorrent υλοποιείται πάνω σε ένα τέτοιο δίκτυο (Figure 3).

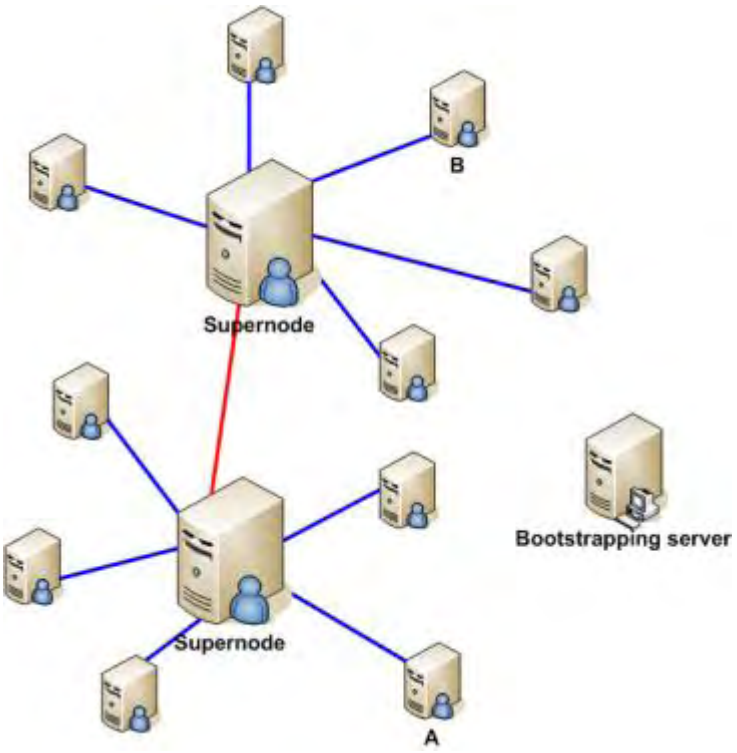

Figure 3: Υβριδικό peer-to-peer δίκτυο

2.2 Πρωτόκολλο BitTorrent

Το πρωτόκολλο BitTorrent είναι ένα πρωτόκολλο επικοινωνίας που υλοποιείται πάνω σε ένα υβριδικό peer-to-peer δίκτυο. Ένας κεντρικός server, γνωστός και ως tracker, διατηρεί διαρκώς μια λίστα με όλους τους συμμετέχοντες. Ο tracker δεν συμμετέχει καθόλου στον διαμοιρασμό του αρχείου, παρά μόνο παρέχει την λίστα με τους συμμετέχοντες. Ο client χρειάζεται ένα πρόγραμμα torrent client για να μπορέσει να συμμετάσχει στο δίκτυο των peers, το σύνολο των οποίων λέγεται swarm [8, 9].

Το πρώτο βήμα για να γίνει ένα αρχείο διαθέσιμο προς διανομή μέσω του BitTorrent είναι να δημιουργηθεί ένα αρχείο με μεταδεδομένα, γνωστό ως αρχείο metainfo ή αρχείο ".torrent". Ο client που θέλει να συνδεθεί με την ομάδα των peers, γνωστή και ως swarm,

χρειάζεται να αποκτήσει αυτό το αρχείο. Συνήθως αυτό γίνεται μέσα από BitTorrent sites που φιλοξενούν αρχεία torrent και χρησιμοποιούνται για την αναζήτησή τους. Ο client μέσω του αρχείου torrent λαμβάνει γνώση για το ποιός είναι ο tracker και ζητά από αυτόν μια λίστα με όλους τους peers που συμμετέχουν στο swarm και διαμοιράζουν το αρχείο. Χρησιμοποιώντας αυτή τη λίστα, ο client πλέον μπορεί να συνδεθεί και να κατεβάσει το αρχείο. Όσα κομμάτια του αρχείου έχει αποκτήσει ο client, μπορεί να τα διαμοιράσει και σε υπόλοιπους peers. Έτσι, κάθε συμμετέχων στο swarm μπορεί και να κατεβάζει το αρχείο αλλά και να συνεισφέρει στον διαμοιρασμό του ανεβάζοντας παράλληλα [8, 9].

2.3 Pieces

Στο πρωτόκολλο BitTorrent το διαμοιραζόμενο αρχείο διαιρείται σε μικρότερα και ίσου μεγέθους κομμάτια, τα οποία ονομάζονται pieces. Το μέγεθός τους είναι συνήθως μεταξύ 64KB και 1MB και καθορίζεται στο αρχείο torrent. Αυτά διαιρούνται περαιτέρω σε μικρότερα κομμάτια, τα blocks, τα οποία είναι και αυτά που ανταλλάσσονται μεταξύ των συμμετεχώντων και το μέγεθός του συνήθως είναι 16KB ή 32KB. Τα blocks αποτελούν τα κομμάτια των δεδομένων που περιέχονται σε ένα μοναδικό μήνυμα, δηλαδή αποτελούν την ανταλλακτική μονάδα του πρωτοκόλλου.

Όταν ενας peer αποκτήσει όλα τα blocks ενός piece, αφού κάνει έλεγχο για την ακεραιότητά του, μπορεί να το διαμοιράσει σε άλλους peers που θα το ζητήσουν από αυτόν [9].

2.3 SHA-1 hashing

Για τον έλεγχο ακεραιότητας ενός piece χρησιμοποιείται μια κρυπτογραφική συνάρτηση κατακερματισμού, η SHA-1 (Secure Hash Algorithm – 1). Αυτή αποτελεί μια μαθηματική συνάρτηση, η οποία δέχεται ώς είσοδο τα bits που αποτελούν το piece και παράγει ένα string μεγέθους 20 bytes. Μετά το κατέβασμα του κάθε piece, ο client παράγει μια τιμή κατακερματισμού μέσω του SHA-1, βάζοντας ως είσοδο το piece, και συγκρίνει την έξοδο με την τιμή που βρίσκεται στο αρχείο torrent. Έτσι ελέγχει αν το piece κατέβηκε σωστά.

Η ίδια συνάρτηση χρησιμοποιείται και κατά την αίτηση που στέλνει ο client στον tracker. Ο client, χρησιμοποιώντας το πεδίο info του αρχείου torrent, παράγει μια τιμή μέσω του SHA-1, την οποία συμπεριλαμβάνει στο αίτημα του. Ο tracker χρησιμοποιεί την τιμή αυτή για να εντοπίσει το αρχείο και να απαντήσει με την λίστα με τους peers [9, 10].

ΚΕΦΑΛΑΙΟ 3

ΕΦΑΡΜΟΓΗ ΤΟΥ ΠΡΩΤΟΚΟΛΛΟΥ TRACKER

Σε αυτό το κεφάλαιο θα γίνει μια παρουσίαση του πρωτοκόλλου tracker, δηλαδή η συνδιαλλαγή μεταξύ torrent client και tracker, και θα παρουσιαστούν σημεία της εφαρμογής torrent client που αναπτύχθηκε.

3.1 Torrent File

Το torrent file, ή αλλιώς metainfo file, είναι ένα αρχείο που περιέχει τις πληροφορίες που χρειάζεται ένας torrent client για να ξεκινήσει να κατεβάζει το αρχείο που είναι προς διαμοιρασμό. Πιο συγκεκριμένα, περιλαμβάνει πληροφορίες για τους φακέλους και τα αρχεία που θα “κατέβουν”, μια λίστα με trackers που έχουν τις πληροφορίες που χρειάζονται για τον εντοπισμό των peers που έχουν αυτό το αρχείο, καθώς και κρυπτογραφικές τιμές κατακερματισμού για την επαλήθευση της ακεραιότητας των αρχείων. Το torrent file έχει την επέκταση “.torrent” στο τέλος του ονόματος του αρχείου και είναι κωδικοποιημένο χρησιμοποιώντας bencoding [9].

3.1.1 Bencoding

Το bencoding είναι ένας τρόπος κωδικοποίησης δεδομένων, και χρησιμοποιείται ώστε να καθορίζει και να οργανώνει τα αρχεία σε μια σύντομη μορφή. Η αποκωδικοποίηση του torrent file παρέχει στον torrent client τις πληροφορίες που χρειάζεται. Υποστηρίζει δεδομένα byte strings, integers, list και dictionaries και η κωδικοποίηση έχει ως ακολούθως [9]:

- **Byte strings**: <το μήκος του string σε base ten ASCII>:<δεδομένα string>
  Για παράδειγμα: “ 4:ball ” συμβολίζει την λέξη ball

- **Integers**: i<ο ακέραιος σε base ten ASCII>e
  Για παράδειγμα: “ i5e “ συμβολίζει τον αριθμό 5

- **Lists**: l<bencoded τιμές>e
  Οι λίστες μπορούν να περιέχουν οποιαδήποτε bencoded δεδομένα, δηλαδή integers, string, dictionaries αλλά και λίστες
  Για παράδειγμα: “ l4:ball3:bowe “ συμβολίζει τη λίστα “ [ball, bow]

- **Dictionaries**: d<bencoded string><bencoded στοιχείο>e
  Το key πρέπει να είναι bencoded string και το value μπορεί να είναι οτιδήποτε bencoded στοιχείο, δηλαδή integer, string, list ή άλλο dictionary.
  Για παράδειγμα: “ d3:dog4:woof3:cat4:meow” και συμβολίζει το dictionary { dog : woof, cat : meow }.

3.1.2 Δομή Αρχείου Torrent

Το αρχείο torrent είναι ένα δομημένο, με κωδικοποίηση bencode, dictionary. Οι τιμές string που περιέχει είναι κωδικοποιημένες σε UTF-8. Στην συνέχεια παρουσιάζεται η δομή του και τα πεδία που περιέχει [9]:

- **announce**: Το url του tracker που συνδέεται ο torrent client για να πάρει τις πληροφορίες που χρειάζεται.

- **announce-list** (προαιρετικά): Πρόκειται για μια λίστα με λίστες που περιλαμβάνει URLs από εναλλακτικούς trackers. Σε περίπτωση που ο torrent client το υποστηρίζει, όπωτος και στη περίπτωση που torrent client που αναπτύχθηκε, και υπάρχει αυτό το πεδίο, ο tracker αγνοεί το πεδίο announce και χρησιμοποιεί αυτό το πεδίο για την ανακάλυψη των trackers.

- **creation date** (προαιρετικά): Η ημερομηνία δημιουργίας το torrent σε μορφή UNIX epoch, δηλαδή σε δευτερόλεπτα (integer) από την 1 Ιανουαρίου 1970 00:00:00 UTC.

- **comment** (προαιρετικά): Σχόλιο του δημιουργού, τύπου string.

- **created by** (προαιρετικά): Το πρόγραμμα (όνομα και έκδοση) που χρησιμοποιήθηκε για την δημιουργία του αρχείου metainfo, τύπου string

- **encoding** (προαιρετικά): Η μορφή της κωδικοποίησης του string που χρησιμοποιήθηκε για να παράγει το πεδίο pieces του πεδίου info, σε μορφή string

- **Info** : Το πεδίο info είναι ένα dictionary που περιλαμβάνει πληροφορίες για τα αρχεία του torrent. Υπάρχουν δυο πιθανές μορφές, μια για τα torrents που περιλαμβάνουν μόνο ένα αρχείο (single file torrents) και μια για τα torrents που περιλαμβάνουν παραπάνω από ένα αρχείο (multi-file torrents).

    - **name**: Στην περίπτωση του singe-file torrent πρόκειται για το όνομα του αρχείου.Στην περίπτωση του multi-file torrent είναι το όνομα του φακέλου που θα αποθηκευτούν τα αρχεία. Και στις δύο περιπτώσεις είναι προαιρετικό.

    - **piece length**: Το μήκος του κάθε piece σε bytes (integer). Συνήθως είναι δύναμη του 2, με πιο διαδεδομένα μεγέθη τα 256 KB, 512KB και 1MB. Κάθε piece έχει το ίδιο μέγεθος με εξαίρεση το τελευταίο piece κάθε αρχείου.

    - **pieces**: string που περιέχει συνεχόμενα όλες τις τιμές hash τύπου SHA1 που αντιστοιχούν σε κάθε piece, με κάθε τιμή να έχει μήκος 20 bytes. Το string αυτό είναι τύπου byte.

    - **private** (προαιρετικά): Όταν είναι 1, ο torrent client μπορεί να πληροφορηθεί για τους υπόλοιπους peers μόνο από τους trackers που αναφέρονται στο αρχείο metainfo. Αν είναι 0 ή δεν έχει οριστεί, ο client μπορεί να χρησιμοποιήσει άλλες μεθόδους για να βρει peers, όπως DHT (distributed hash table) και PEX (peer exchange).

- **length**: Το μήκος του συνολικού αρχείου σε bytes (integer). Περιλαμβάνεται μόνο σε single-file torrents.

- **files**: Το πεδίο αυτό υπάρχει μόνο όταν πρόκειται για multi-file torrents. Πρόκειτα για μια list από dictionaries, με κάθε dictionary να αντιστοιχεί σε ένα αρχείο. Το κάθε dictionary περιλαμβάνει τα εξής πεδία:

  - **length**: Το μήκος του εκάστοτε αρχείου σε bytes (integer)

  - **path**: Μια λίστα που περιλαμβάνει ένα ή περισσότερα strings που μαζί δομούν το path και το όνομα του αρχείου. Κάθε στοιχείο της λίστας αντιστοιχεί είτε σε ένα όνομα φακέλου είτε σε ένα όνομα αρχείου. Για παράδειγμα, για το path για το αρχείο “dir1/dir2/file” η λίστα θα περιλάμβανε 3 stings: “dir1”, “dir2” και “file”. Η λίστα είναι κωδικοποιημένη σαν bencode list και στο παράδειγμα θα είχε την μορφή:
  “l4:dir14:dir24:filee”.

3.2 Tracker

Όταν ο torrent client θέλει να συμμετέχει στο swarm ενός συγκεκριμένου torrent, πρέπει κάπως να πάρει τις πληροφορίες που χρειάζεται, δηλαδή τους peers που διαμοιράζουν το συγκεκριμένο αρχείο, ώστε να αρχίσει να πραγματοποιεί τις απαραίτητες συνδέσεις για να κατεβάσει το αρχείο. Τις απαραίτητες πληροφορίες αυτές τις παρέχει ο tracker. Ο tracker ειναι ένα server που αποκρίνεται σε HTTP GET requests. Δρα σαν διαμεσολαβητής μεταξύ των peers και ο μόνος σκοπός του είναι να παρέχει πληροφορίες για την κατάσταση του swarm για το εκάστοτε torrent. Δεν συμμετέχει στον διαμοιρασμό του αρχείου και δεν περιέχει το διαμοιραζόμενο αρχείο [9, 11].

3.2.1 Tracker Request

Για να ξεκινήσει ο torrent client να κατεβάζει το αρχείο πρέπει να λάβει την λίστα με τους ενεργούς peers από τον tracker. Για να γίνει αυτό, το πρώτο βήμα που κάνει ο torrent client είναι να στείλει ένα GET request στον tracker. Η βάση του URL που θα στείλει ο client το request περιλαμβάνεται είτε στο πεδίο “announce”, είτε στο πεδίο “announce-list” του αρχείου metainfo (.torrent). Στη συνέχεια του βασικού URL προστίθενται οι κατάλληλες παράμετροι, χρησιμοποιώντας τις τυπικές μεθόδους CGI, δηλαδή ένα “?” μετά το announce URL και τις παραμέτρους στη συνέχεια με την μορφή “param=value”, διαχωρισμένες από ένα “&” [9, 11].

Παρακάτω παρατίθενται οι παράμετροι που χρησιμοποιούνται στο URL request [9, 11]:

```
params = {
    'info_hash': torrent.info_hash,
    'peer_id': torrent.peer_id,
    'uploaded': 0,
    'downloaded': 0,
    'port': 6881,
    'left': torrent.total_length,
    'event': 'started'
}
```

Figure 4: Request parameters

- **info_hash**: Για να παραχθεί αυτή η τιμή, κρυπτογραφούμε αρχικά με τον αλγόριθμο SHA-1 το πεδίο info του αρχείου metainfo, όπου παράγεται μια τιμή μήκους 20 bytes. Στη συνέχεια, για να είναι κατάλληλο για αποστολή, εφαρμόζουμε URL encoding.

- **peer_id**: Ένα string, μήκους 20 bytes και URL encoded. Το string αυτό μπορεί να είναι οτιδήποτε και δεν υπάρχουν αυστηρές οδηγίες στο πώς μπορεί να παραχθεί.

Χρησιμοποιείται για την ταυτοποίηση του client. Στον torrent client που αναπτύχθηκε χρησιμοποιούμε ένα timestamp για την παραγωγή του.

```
def generate_peer_id(self):
    seed = str(time.time())
    return hashlib.sha1(seed.encode('utf-8')).digest()
```

Figure 5: παραγωγή peer id

- **port**: Ο αριθμός του port που “ακούει” ο torrent client. Τυπικά χρησιμοποιούνται οι τιμές 6881 έως 6889.

- **uploaded**: Ο συνολικός αριθμός των bytes που έχει “ανεβάσει” ο torrent client, μετρώντας από τη στιγμή που θα στείλει το event “started” στον tracker, σε κωδικοποίηση base ten ASCII.

- **downloaded**: Ο συνολικός αριθμός των bytes που έχει “κατεβάσει” ο torrent client, μετρώντας από τη στιγμή που θα στείλει το event “started” στον tracker, σε κωδικοποίηση base ten ASCII.

- **left**: Ο συνολικός αριθμός των bytes που απομένει για να ολοκληρωθεί το “κατέβασμα” του αρχείου, δηλαδή στο αρχικό request το συνολικό μήκος του αρχείου, σε κωδικοποίηση base ten ASCII.

- **event** (προαιρετικά): Εάν ορίζεται, είναι ένα από τα started, stopped, completed. Αν δεν ορίζεται, πρόκειται για ένα request που γίνεται ανα τακτά χρονικά διαστήματα.
  - *started*: Συμπεριλαμβάνεται στο πρώτο request που θα στείλει ο client στον tracker.
  - *stopped*: Αποστέλλεται όταν ο client αποχωρεί από το swarm.
  - *completed*: Αποστέλλεται όταν ο client ολοκληρώσει το “κατέβασμα” του συνολικού αρχείου.

3.2.2 Tracker Response

Αφού ο torrent client στείλει το πρωταρχικό request στον tracker, ο δεύτερος απαντάει με ένα tracker response, το οποίο είναι ένα bencoded dictionary. Το response περιέχει τα παρακάτω πεδία [9, 11]:

- **failure reason**: Εάν ο tracker στείλει αυτό το πεδίο σημαίνει ότι υπήρξε κάποια αποτυχία και τα υπόλοιπα πεδία παραλείπονται. Το response είναι ένα string σε αναγνώσιμη μορφή, το οποίο περιγράφει τον λόγο για τον οποίο υπήρξε η αποτυχία.

- **interval**: Το χρονικό διάστημα σε δευτερόλεπτα που μπορούν να απέχουν δύο συνεχόμενα requests του torrent client στον tracker ώστε να γίνουν δεκτά κα να επεξεργαστούν.

- **peers**: Το πεδίο αυτό είναι μια λίστα από dictionaries, με καθένα απο αυτά να περιλαμβάνει πληροφορίες για κάθε peer. Περιέχουν τα εξής keys:
  - **peer id**: Το peer id που έχει επιλέξει αυθαίρετα ο peer, όπως αναλύθηκε πιο πάνω.
  - **ip**: Η ip του peer σε μορφή ή IPv4 (τετράδες χωρισμένες με τελεία), είτε IPv6 (δεκαεξαδικές τιμές), είτε DNS (string).
  - **port**: ο αριθμός του port που πραγματοποιεί ο peer την επικοινωνία του (integer).

ΚΕΦΑΛΑΙΟ 4

ΕΦΑΡΜΟΓΗ ΤΟΥ ΠΡΩΤΟΚΟΛΛΟΥ PEER

Το πρωτόκολλο peer χρησιμοποιείται για την ανταλλαγή blocks μεταξύ των peers. Η σύνδεση μεταξύ των peers είναι συμμετρική, δηλαδή τα μηνύματα που ανταλλάσσονται είναι όμοια.Το πρωτόκολλο peer εφαρμόζεται πάνω από το πρωτόκολλο TCP.

4.1 Peer states

Ο client καθ' όλη τη διάρκεια της επικοινωνίας τους peers αποθηκεύει τις εξής καταστάσεις για τον καθένα από τους peers [9, 11]:

- **choked**: Αυτό το πεδίο μας δείχνει αν ο peer δέχεται ή όχι requests από τον client. Όταν ο peer έχει "choked" τον client, αυτό σημαίνει ότι για οποιοδήποτε request στείλει ο client δεν θα λάβει απάντηση. Για να λάβει απάντηση, ο client θα πρέπει να γίνει "unchoked" από τον peer. Οι τιμές που λαμβάνει το πεδίο είναι "1" ή "True" για "choked" και "0" ή "False" για "unchoked".

- **interested**: Αυτό το πεδίο μας δείχνει αν ο απομακρυσμένος peer ενδιαφέρεται για κάποιο piece που έχει ο τοπικός client. Αν είναι "1" ή "True", ο peer πιθανότατα θα στείλει κάποιο request για piece προς τον client. Αν είναι "0" ή "False", αυτό σημαίνει ότι ο peer δεν έχει δείξει κάποιο ενδιαφέρον για κάποιο piece που έχει ο client.

Πρακτικά, ο κάθε peer όπως και ο τοπικός client κρατούν συνεχώς μεταβλητές που καταγράφουν αν είναι "choked" από τον κάθε peer, αν έχουν "choked" κάποιον peer, αν είναι "interested" για κάποιο piece που έχει καποιος peer και αν είναι "interested" οι ίδιοι για κάποιο piece που έχει ο peer [11]:

- *am_choking*: ο τοπικός client έχει τον peer “choked”
- *am_interested*: ο τοπικός client ενδιαφέρεται για τον peer
- *peer_choking*: ο peer έχει τον τοπικό client “choked”
- *peer_interested*: ο peer ενδιαφέρεται για τον client

Η κατάσταση που κρατάει ο client κατά την αρχή της σύνδεσης φαίνεται στην παρακάτω εικόνα:

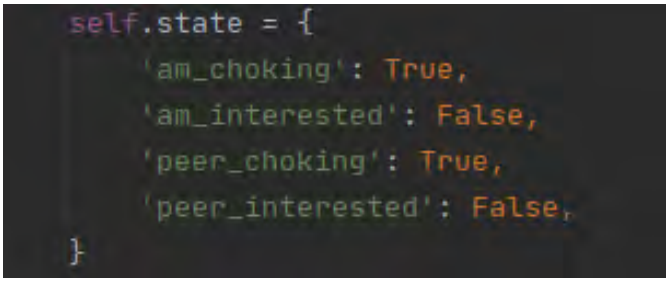

Figure 6: Αρχική κατάσταση client

4.2 Συνδιαλλαγή μεταξύ client και peers

Το πρώτο βήμα που πρέπει να κάνει ο client για να συνδεθεί με έναν peer είναι να δημιουργήσει μια TCP σύνδεση. Αφού δημιουργηθεί η σύνδεση, το επόμενο βήμα είναι να γίνει ανταλλαγή μηνυμάτων τύπου “handshake” και στη συνέχεια, client και peer μπορούν να συνδιαλέγονται μέσω μηνυμάτων [9, 11].

4.2.1 Handshake

Το “handshake” είναι το πρώτο μήνυμα που πρέπει να στείλει ο client προς τον εκάστοτε peer. Με αυτόν τον τρόπο ο torrent client δείχνει την πρόθεσή του να συμμετάσχει στο swarm. Το “handshake” έχει την εξής μορφή:

handshake: <*pstrlen*><*pstr*><*reserved*><*info_hash*><*peer_id*>

Κάθε πεδίο λαμβάνει τις εξής τιμές:

- **pstrlen**: το μήκος του πεδίου “pstr”, μήκους 1 byte.
- **pstr**: string που συμβολίζει το πρωτόκολλο που χρησιμοποιείται, ήτοι “BitTorrent protocol” και μήκους 19 bytes.
- **Reserved**: 8 bytes με τιμή 0 τα οποία έχουν κρατηθεί για μελλοντικές επεκτάσεις του πρωτοκόλλου.
- **Info_hash**: Η τιμή κατακερματισμού του πεδίου info αφού εφαρμοστεί η κρυπτογράφηση SHA-1. Αυτό το πεδίο έχει την ίδια τιμή με το πεδίο info_hash που στέλνει ο client στον tracker κατά το request. Έχει μήκος 20 bytes.
- **peer_id**: Ένα string, μήκους 20 bytes, που χρησιμοποιείται σαν μοναδικό αναγνωριστικό για τον client. Αυτό μπορεί να είναι το ίδιο με το peer_id που στέλνει ο client στον tracker, αλλά όχι πάντα. Μπορεί να επιλεγεί διαφορετικό peer_id για λόγους ανωνυμίας.

Αφού ο client δομήσει και στείλει το “handshake”, θα περιμένει να λάβει ένα αντίστοιχο “handshake” πίσω από τον peer. Το “handshake” του peer θα πρέπει να έχει την ίδια τιμή info_hash με αυτήν που έστειλε ο client. Έτσι εξασφαλίζεται ότι και οι δύο αναφέρονται στο ίδιο αρχείο torrent. Με το πέρας της ανταλλαγής των “handshakes”, ο

client και ο peer έχουν εγκαθιδρύσει τη σύνδεση μεταξύ τους και μπορούν να ανταλλάξουν μηνύματα.

4.3 Μηνύματα

Μπαίνοντας, πλέον, στην ουσία του πρωτοκόλλου BitTorrent, υπάρχουν 11 διαφορετικοί τύποι μηνυμάτων που ανταλλάσσονται μεταξύ των peers. Αυτά έχουν την εξής μορφή: < length prefix >< message id >< payload > . Το length prefix είναι 4 bytes σε μορφή big endian και συμβολίζει το μέγεθος του υπόλοιπου μηνύματος χωρίς να συμπεριλαμβάνεται το μέγεθος του length prefix. Το message id είναι μεγέθους ενός byte και συμβολίζει τον τύπο του μηνύματος. Τέλος, το payload χρησιμοποιείται για περαιτέρω πληροφορίες και το μέγεθός του ποικίλει. Οι 11 τύποι μηνυμάτων είναι οι εξής [9, 11]:

- **keep-alive**: <length prefix = 0000>
  Το μήνυμα keep alive συμβολίζεται με τέσσερα μηδενικά bytes στο πεδίο length prefix, καθώς δεν περιλαμβάνει ούτε message id ούτε payload. Ο κάθε peer, όταν δεν έχει λάβει μήνυμα για κάποιο χρονικό διάστημα (συνήθως δύο λεπτά), μπορεί να κλείσει την σύνδεση με τον client. Το μήνυμα keep-alive αποστέλλεται με σκοπό να διατηρηθεί η σύνδεση σε περίπτωση που ο client δεν έχει στείλει κάποιο άλλο μήνυμα.

- **choke**: <length prefix = 0001><message id = 0>
  Το μήνυμα “choke” αποστέλλεται όταν ο client κάνει “choke” τον peer. Έχει στάνταρ μέγεθος (5 bytes) και δεν περιλαμβάνει payload.

- **unchoke**: <length prefix = 0001><message id = 1>
  Το μήνυμα unchoke αποστέλλεται όταν ο client θέτει τον peer ως unchoked, δηλαδή μπορεί να λαμβάνει και να εξυπηρετεί requests από τον peer. Έχει στάνταρ μέγεθος (5 bytes) και δεν περιλαμβάνει payload.

- **interested**: <length prefix = 0001><message id = 2>

Με το μήνυμα αυτό ο αποστολέας εκφράζει το ενδιαφέρον του για κάποιο piece που έχει ο παραλήπτης. Έχει στάνταρ μέγεθος (5 bytes) και δεν περιλαμβάνει payload.

- **not interested**: <length prefix = 0001><message id = 3>
  Με το μήνυμα αυτό ο αποστολέας δηλώνει ότι δεν ενδιαφέρεται για κάποιο piece που έχει ο παραλήπτης. Έχει στάνταρ μέγεθος (5 bytes) και δεν περιλαμβάνει payload.

- **have**: <length prefix = 0005><message id =4><piece index>
  Το μήνυμα “have” δηλώνει ότι ο αποστολέας έχει κάποιο piece. Με το πεδίο piece index (ακέραιος) υποδηλώνεται ποιό piece είναι αυτό. Συνήθως ο αποστολέας, όταν λάβει κάποιο piece, στέλνει αυτό το μήνυμα σαν επιβεβαίωση ότι το έλαβε.

- **bitfield**: <length prefix = 0001 + bitfield length><message id = 5><bitfield>
  Το bitfield είναι ο τρόπος με τον οποίο απεικονίζονται τα pieces ενός αρχείου που έχει ένας peer. Πρόκειται για μια λίστα από bits, από τα οποία το καθένα συμβολίζει ένα piece του αρχείου και το πρώτο piece έχει την αρίθμηση 0. Αν ο αποστολέας έχει κάποιο piece, τότε το αντίστοιχο bit είναι “1”, ενώ αν δεν το έχει, το bit έχει την τιμή “0”. Το μήνυμα bitfield μπορεί να σταλεί αμέσως μετά το handshake. Το μέγεθός του είναι ανάλογο του μεγέθους του αρχείου και τα περισσευούμενα bits στο τελευταίο byte τίθενται “0”.

- **request**: <length prefix = 0013><message id = 6><index><begin><length>
  Το μήνυμα request αποστέλλεται όταν ο αποστολέας θέλει να ζητήσει κάποιο block από τον παραλήπτη. Αναλύοντας το payload, το index συμβολίζει το piece στο οποίο βρίσκεται το block, δηλαδή το piece index, το begin συμβολίζει το offset του block, δηλαδή την αρχή του και το length συμβολίζει το μέγεθος του block για το οποίο ενδιαφέρεται ο αποστολέας του request.

- **piece**: <length prefix = 0009 + payload><message id = 7><index><begin><block>
  Το μήνυμα piece είναι αυτό που περιέχει το block που έχει ζητήσει ο παραλήπτης. Το μέγεθος του ποικίλει ανάλογα με το μέγεθος του block. Το index και εδώ συμβολίζει το piece του οποίου το block αποτελεί μέρος, το begin συμβολίζει το offset του block μέσα στο piece και το block είναι το ίδιο το block, δηλαδή το κομμάτι των δεδομένων του piece που έχει ζητήσει ο παραλήπτης.

- **cancel**: <length prefix = 0013><message id = 8><index><begin><length>
  Το μήνυμα cancel είναι όμοιο με το μήνυμα request, πέρα από το message id. Το μήνυμα cancel αποστέλλεται όταν ο αποστολέας έχει στείλει ένα μήνυμα request και πιθανότατα έχει λάβει το block που ζητούσε. Με το cancel ο παραλήπτης ενημερώνεται ούτως ώστε να μην στείλει το block που έχει ζητηθεί.

- **port**: <length prefix = 0003><message id = 9><listen port>
  Το μήνυμα port χρησιμοποιείται από clients που υποστηρίζουν Distributed Hash Table (DHT) για να βρουν τους peers, αντί του συμβατικού τρόπου μέσω trackers.

4.4 Ανταλλαγή μηνυμάτων

Η ανταλλαγή μηνυμάτων έχει ως εξής: αφού ο client και ο server έχουν ανταλλάξει handshakes, ο peer στέλνει στέλνει συνήθως ένα μήνυμα “bitfield”, υποδηλώνοντας ποιά pieces του αρχείου έχει ολόκληρα. Εναλλακτικά μπορεί να στείλει μια σειρά μηνυμάτων “have”, ένα για κάθε piece που έχει. Αντίστοιχα και ο client μπορεί να στείλει μηνύματα “bitfield” ή “have” στον peer, αλλά στην περίπτωση που ξεκινά το κατέβασμα του αρχείου μπορεί να παραλείψει την αποστολή τους [9].

Αφού ο client αποθηκεύσει αυτές τις πληροφορίες σε ένα bitfield, επεξεργάζεται και βρίσκει ποια pieces του λείπουν. Το επόμενο βήμα είναι να στείλει ένα μήνυμα interested, όπου θα δηλώσει το ενδιαφέρον του για την λήψη blocks. Σε περίπτωση που ο peer στείλει πίσω ένα μήνυμα “unchoke”, ο client είναι σε θέση να στείλει ένα μήνυμα request με το οποίο θα ζητά συγκεκριμένο block και ο peer απαντά με ένα μήνυμα piece, δηλαδή με το block που ζητήθηκε. Ο peer μπορεί σε οποιοδήποτε σημείο της συνδιάλεξης να στείλει ένα μήνυμα “choke” και να αγνοήσει οποιοδήποτε request λάβει από εκείνο το σημείο κι έπειτα. Αφού τελειώσει το download, ο client μπορεί να στείλει μηνύματα cancel προς όποιον peer έχει στείλει requests, ώστε να τους ενημερώσει ότι το δεν χρειάζεται πια τα blocks που ζήτησε [9].

Η ανταλλαγή αυτών των μηνυμάτων συμβαίνει και προς τις δύο κατευθύνσεις. Εάν ο client έχει ένα μέρος ή ολόκληρο το αρχείο, θα πρέπει να κρατά τις καταστάσεις “choked”, “unchoked”, “interested” και “not interested” για τον καθένα από τους peers που συμμετέχουν. Αν ο peer στείλει ένα μήνυμα “interested”, ο client πρέπει να αποφασίσει αν θα εξυπηρετήσει τα επικείμενα requests του peer, ήτοι αν θα στείλει ή όχι ένα μήνυμα “unchoke”[9].

4.5 Αλγόριθμοι

4.5.1 Αλγόριθμοι request

Για να αποφασίσει ο client τι requests θα στείλει στους peers υπάρχουν τέσσερις διαφορετικές στρατηγικές [12]:

- **Strict Priority**:
  Όταν ο client κατεβάσει ένα block ενός piece, δίνεται προτεραιότητα να ζητηθούν τα υπόλοιπα blocks του piece ώστε αυτό να ολοκληρωθεί, αντί να στείλει request για άλλα blocks άλλων pieces.

- **Random first piece**:
  Στην αρχή του downloading, όταν ο client δεν έχει τίποτα να ανεβάσει, επιλέγεται στην τύχη ένα block για να ζητηθεί. Αυτό συνεχίζεται μέχρι να έχει ένα ολοκληρωμένο piece. Από εκεί κι έπειτα ξεκινάει η στρατηγική rarest piece first.

- **Rarest piece first**:
  Το rarest piece first αποτελεί την πιό διαδεδομένη στρατηγική. Ο client στέλνει requests για τα πιό σπάνια blocks. Κοιτώντας τα bitfields του κάθε peer, τα οποία ανανεώνονται και με μηνύματα have κατά την διάρκεια της συνδιαλλαγής, ο client μπορεί να υπολογίσει ποιά pieces εμφανίζονται πιο σπάνια μεταξύ των peers και να στείλει requests για αυτά. Σε κάθε υλοποίηση αυτής της στρατηγικής παρ’ όλ’ αυτά θα

πρέπει να υπάρχει και μια τυχαιότητα στην επιλογή των πιο σπάνιων pieces, καθώς αν όλοι οι peers ζητούν τα πιο σπάνια pieces ο αλγόριθμος θα προβεί αντιπαραγωγικός.

- **End game**:

  Όταν το κατέβασμα του αρχείου έχει σχεδόν τελειώσει, υπάρχει μια τάση τα τελευταία blocks να κατεβαίνουν αργά. Για να λυθεί αυτό το πρόβλημα, ο client στέλνει μαζικά requests στους peers που έχουν τα blocks που του λείπουν. Όταν κατέβουν και τα τελευταία blocks, ο client στέλνει ένα μήνυμα cancel σε όλους τους peers που είχε στείλει πιο πριν requests.

4.5.2 Choking

Choking ονομάζεται η προσωρινή άρνηση του client να εξυπηρετήσει requests ενός peer. Ο μηχανισμός αυτός υφίσταται για δύο λόγους: για την αποσυμφόρηση του δικτύου, καθώς ο μηχανισμός ελέγχου συμφόρησης του TCP δεν είναι ιδανικός όταν πρόκειται για πολλές παράλληλες συνδέσεις, και για να αποφευχθούν peers οι οποίοι κάνουν download χωρίς να κάνουν upload το αρχείο.

Ένας αλγόριθμος choking θα πρέπει να έχει κάποια συγκεκριμένα χαρακτηριστικά για να είναι αποτελεσματικός. Αρχικά, θα πρέπει να έχει ένα συγκεκριμένο άνω όριο παράλληλων συνδέσεων για να επιτύχει την καλύτερη επίδοση πάνω στο πρωτόκολλο TCP. Θα πρέπει να αποφεύγεται οι peers να γίνονται συχνά choked και unchoked ώστε να δίνεται χρόνος να κριθεί αν ο κάθε peer κάνει upload σε ικανοποιητικό βαθμό και δεν έχει επηρεαστεί ο ρυθμός upload του από τον μηχανισμό αποσυμφόρησης του TCP. Ο μηχανισμός αυτός ονομάζεται “fibrillation” (“μαρμαρυγή”). Επιπρόσθετα, θα πρέπει να δίνεται προτεραιότητα σε peers που κάνουν upload στον client. Τέλος, θα πρέπει να δοκιμάζει μη ενεργές συνδέσεις με peers ώστε να συγκρίνει αν είναι καλύτερες από τις τωρινές. Ο μηχανισμός αυτός ονομάζεται optimistic unchoking.

Στην πράξη συνήθως επιλέγονται τέσσερις peers για να γίνουν unchoke. Οι peers αυτοί επιλέγονται σύμφωνα με τον ρυθμό upload τους, δηλαδή με τον ρυθμό που στέλνουν στον client δεδομένα, και με το αν έχουν εκφράσει ενδιαφέρον προς τον client. Με αυτόν τον τρόπο επιτυγχάνονται μεγαλύτερες ταχύτητες download. Αυτοί οι τέσσερις peers

ονομάζονται downloaders. Οι peers με το καλύτερο ρυθμό upload, σε σύγκριση με τους downloaders, που δεν είναι interested, γίνονται unchoke. Αν γίνουν interested, ο downloader με το χειρότερο ρυθμό upload γίνεται choke. Αν ένας peer έχει ολόκληρο το αρχείο επιλέγει το ποιούς θα κάνει unchoke σύμφωνα με το ρυθμό upload του.

Στον μηχανισμό optimistic unchoking, κάθε 30 δευτερόλεπτα επιλέγεται τυχαία ένας peer και γίνεται unchoke, ανεξάρτητα από τον ρυθμό upload του. Αν αυτός είναι interested, τότε συμπεριλαμβάνεται στους τέσσερις downloaders. Οι peers οι οποίοι συνδέονται τελευταίοι χρονικά έχουν τις τριπλάσιες πιθανότητες να είναι αυτοί που θα επιλεγούν για optimistic unchoking, δίνοντας την ευκαιρία σε peers που μόλις ξεκίνησαν το κατέβασμα του αρχείου να κατεβάσουν κάποιο ολοκληρωμένο piece, ώστε να ξεκινήσουν και αυτοί το upload [12].

4.5.3 Anti-snubbing

Συχνά ένας client μπορεί να γίνει choked από όλους τους peers από τους οποίους έκανε download. Σε αυτήν την περίπτωση ο client θα πρέπει να περιμένει να βρει καινούριους peers μέσω του μηχανισμού του optimistic unchoking. Το πρόβλημα με αυτό είναι ότι το optimistic unchoking συμβαίνει κάθε 30 δευτερόλεπτα, άρα ο client δεν θα λαμβάνει καθόλου δεδομένα για αυτό το χρονικό διάστημα. Για να μην υπάρξει τέτοιο πρόβλημα, όταν κάποιος peer έχει να κάνει upload πάνω από 60 δευτερόλεπτα στον client, ο τελευταίος υποθέτει ότι έχει γίνει “snubbed” (“σνομπάρεται”) από τον peer και σταματάει να του στέλνει δεδομένα, με εξαίρεση αν γίνει optimistic unchoke. Σε αυτήν την περίπτωση, ο client θα ανεβάσει τον αριθμό των optimistic unchoke σε μια προσπάθεια για αναζήτηση καλύτερου uploader [12].

Κεφάλαιο 5

5.1 Συμπεράσματα

Η παρούσα εργασία αποτέλεσε την ανάλυση του πρωτοκόλλου BitTorrent, παρουσιάζοντας το θεωρητικό υπόβαθρο πάνω στο οποίο βασίζεται.

Παρουσιάστηκαν τα πρωτόκολλα peer-to-peer, οι πιθανές τοπολογίες των δικτύων αυτών και η αρχή λειτουργικότητας του BitTorrent, καθώς και σημαντικά κομμάτια που βοηθούν στην πλήρη κατανόηση του.

Παρουσιάστηκε το πρωτόκολλο Tracker, δηλαδή ο τρόπος με τον οποίο ο client συνδέεται με τον tracker για να πάρει την απαραίτητη λίστα με τους peers που χρειάζεται. Αναλύθηκαν τα πεδία του αρχείου torrent και τα πεδία των μηνυμάτων tracker request και tracker reply.

Στη συνέχεια, έγινε λόγος για το πρωτόκολλο peer. Παρουσιάστηκαν οι πιθανές καταστάσεις ενός peer, η αρχική επαφή μεταξύ δύο συμμετεχόντων στο swarm και τα μηνύματα που ανταλλάσουν μεταξύ τους. Τέλος, παρουσιάστηκαν αλγόριθμοι που μπορεί να επιλέξει ο client για να κατεβάσει τα blocks, ο μηχανισμός choking που καθιστά το BitTorrent αξιόπιστο και ο αλγόριθμος anti-snubbing.

5.2 Το μέλλον του BitTorrent

Μετά από 20 και πλέον χρόνια, το BitTorrent συνεχίζει να είναι ένας από τους πιο δημοφιλείς τρόπους για τον διαμοιρασμό μεγάλων αρχείων. Έχει δεχθεί ισχυρό πόλεμο από εταιρίες για πνευματικά δικαιώματα, καθώς το BitTorrent έχει χρησιμοποιηθεί κατά πολύ για παράνομη διανομή ταινιών, τραγουδιών και software. Η παράνομη διανομή υλικού δε φαίνεται να μπορεί να εξαλειφθεί, παρά μόνο να περιοριστεί. Αν γίνει αυτό, το πιο πιθανό είναι το BitTorrent να χάσει πολλούς από τους χρήστες του. Παρ' ολ' αυτά, το BitTorrent θα συνεχίσει να χρησιμοποιείται και να αναπτύσσεται, καθώς θα συνεχίσει, λόγω της αρχιτεκτονικής του, να αποτελεί έναν από τους πιο αξιόπιστους και γρήγορους τρόπους για την διανομή μεγάλων αρχείων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Peer-to-peer, Wikipedia [Online]
https://en.wikipedia.org/wiki/Peer-to-peer

[2] Clive Thompson (2005), "*The Bittorrent Effect*" [Online]
https://www.wired.com/2005/01/bittorrent-2/

[3] Schulze Hendrik, Klaus Mochalski (2009), "I*nternet Study 2008/2009*"

[4] Palo Alto Networks (2013), "Application Usage & Threat Report" [Online]
https://blog.paloaltonetworks.com/app-usage-risk-report-visualization/

[5] Marozzo Fabrizio, Talia Domenico, Trunfio Paolo (2020), "*A Sleep-and-Wake technique for reducing energy consumption in BitTorrent networks*"

[6] Rüdiger Schollmeier (2002), "*A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications"*

[7] Shahriar Mohammadi, Khaje Nasir Toosi University of Technology (2012), "*Anonymous Communication in Peer-to-Peer Networks for Providing more Privacy and Security*"

[8] BitTorrent, Wikipedia [Online]
https://en.wikipedia.org/wiki/BitTorrent

[9] Bram Cohen (2017), "*The BitTorrent Protocol Specification*" [Online]
https://www.bittorrent.org/beps/bep_0003.html

[10] National Institute of Standards and Technology (2015), "*Secure Hash Standard (SHS)*", FIPS Pub 180-4, p.10

[11] Bittorrent protocol specification v.1.0 [Online]
https://wiki.theory.org/index.php/BitTorrentSpecification#Tracker_HTTP.2FHTTPS_Protocol

[12] Bram Cohen (2003), "*Incentives Build Robustness in BitTorrent*"