ΠΑΝΕΠΙΣΤΗΜΕΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ
ΒΟΛΟΣ 2013

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ :

ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΠΡΩΤΟΚΟΛΟΥ BITTORRENT ΣΤΟΝ ΠΡΟΣΟΜΟΙΩΤΗ OMNET++

IMPLEMENTATION OF TE BITTORRENT PROTOCOL IN THE OMNET++ SIMULATOR

ΜΙΜΙΔΗΣ-ΚΕΝΤΗΣ ΑΓΓΕΛΟΣ

Επιβλέπων καθηγητής Αργυρίου Αντώνιος, λέκτορας Π.Θ.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους εξής ανθρώπους για την συνεισφορά τους στην έως τώρα πορεία μου.

- Την οικογένεια μου
- Τους φίλους μου
- Τους καθηγητές μου για τις γνώσεις που μου έδωσαν και ιδιαίτερα τον κύριο Αργυρίου για την βοήθειά του στην ολοκλήρωση αυτής της πτυχιακής

ΠΕΡΙΕΧΟΜΕΝΑ

Εισαγωγή

Θέμα της πτυχιακής είναι η μελέτη, κατανόηση και υλοποίηση του γνωστού peer to peer πρωτοκόλλου BitTorrent. Για την καλύτερη κάλυψη του θέματος ακολουθείτε η εξής δομή.

Αρχικά γίνεται μια περιγραφή πάνω στην τεχνολογία peer to peer. Μετά παρουσιάζεται αναλυτικά η λειτουργία του BitTorrent όπως αυτή περιγράφεται στο επίσημο site. Στην συνέχεια γίνεται μια αναφορά στις τεχνικές της προσομοίωσης δικτύων (τι εξυπηρετούν, τι εργαλεία υπάρχουν διαθέσιμα) και τέλος γίνεται μια αναλυτική περιγραφή της υλοποίησης στον Omnet++ του πρωτοκόλλου BitTorrent, καθώς και η παράθεση αποτελεσμάτων από πειράματα που έγιναν σε αυτήν με σκοπό τόσο τον έλεγχο της ορθότητάς της όσο και για την καλύτερη κατανόηση του πρωτοκόλλου.

PEER-TO-PEER ΣΥΣΤΗΜΑΤΑ

Ένα peer-to-peer σύστημα διασυνδέει μεταξύ τους πολλούς τελικούς χρήστες σε μια κατανεμημένη αρχιτεκτονική δικτύου στο οποίο ο κάθε χρήστης συνεισφέρει ισότιμα. Με το που ένας peer εισέρχεται στο σύστημα διαθέτει μέρος των διαθέσιμων πόρων του στους υπόλοιπους χρήστες σε αντίθεση με το μοντέλο client-server στο οποίο όλοι οι χρήστες χρησιμοποιούν μόνο τους διαθέσιμους πόρους του/των server.

Τα peer-to-peer συστήματα δημιουργούν ένα overlay δίκτυο στο επίπεδο εφαρμογής το οποίο "πατάει" πάνω στο ήδη υπάρχων φυσικό δίκτυο. Ανάλογα με τον τρόπο σύνδεσης μεταξύ των peer τα συστήματα αυτά χωρίζονται σε δομημένα (structured) και μη-δομημένα (unstructured).

Σε ένα δομημένο peer-to-peer δίκτυο οι peer ακολουθούν ένα ακριβές πρωτόκολλο το οποίο περιγράφει την συμπεριφορά του κάθε peer. Αυτό διασφαλίζει ότι οποιοσδήποτε peer μπορεί εύκολα να συνδεθεί με τον peer που διαθέτει το επιθυμητό αρχείο. Αυτό επιτυγχάνεται συνήθως μέσω ενός DHT πίνακα στον οποίο διευκρινίζεται ποιος peer είναι υπεύθυνος για πιο αρχείο.

Στα αδόμητα δίκτυα οι peer αναζητούν τα επιθυμητά αρχεία μέσω διάφορων τεχνικών όπως το flooding, στην οποία ο κάθε peer στέλνει queries σε όλους τους peer του δικτύου για το αν διαθέτουν το αρχείο. Τα αδόμητα αυτά δίκτυα χωρίζονται σε τρεις κατηγορίες τα κεντρικοποιημένα, τα αποκεντροποιημένα και τα υβριδικά. Σε ένα κεντρικοποιημένο P2P δίκτυο υπάρχει μια κεντρική οντότητα (server) η οποία είναι υπεύθυνη να ενημερώνει τους peer για το πώς θα συνδεθούν με τους υπόλοιπους καθώς και το ποια αρχεία αυτοί διαθέτουν. Χαρακτηριστικά παραδείγματα αδόμητων κεντρικοποιημένων P2P συστημάτων αποτελούν τα πρωτόκολλα BitTorrent και Napster. Η κύρια αδυναμία αυτού του τύπου δικτύων είναι ότι διαθέτουν ένα κεντρικό σημείο το οποίο μπορεί είτε να αποτύχει και να οδηγήσει σε πλήρη κατάρρευση του δικτύου, είτε να έχει πολύ μεγάλη κίνηση η οποία θα οδηγήσει σε bottleneck στο συγκεκριμένο σημείο.

Στα αποκεντροποιημένα συστήματα όλοι οι κόμβοι του δικτύου είναι ισότιμοι,

αυτό σημαίνει ότι όταν ένας peer θέλει να συνδεθεί με το δίκτυο πρέπει να βρει και να συνδεθεί αρχικά με έναν peer και ο οποίος θα του δώσει τις απαραίτητες πληροφορίες (IP, port) ώστε να συνδεθεί με τους υπόλοιπους. Σε περίπτωση που ένας peer θέλει να αναζητήσει ένα αρχείο στέλνει queries στους peer με τους οποίους είναι συνδεμένος και αυτοί με τη σειρά τους είναι υπεύθυνοι να τα προωθήσουν στους δικούς τους γειτονικούς peers. Σε περίπτωση που κάποιος peer λάβει ένα query για κάποιο αρχείο που διαθέτει τότε στέλνει πίσω μια ανάλογη απάντηση η οποία και ακολουθεί την αντίθετη διαδρομή του query. Αφού λάβει τις όποιες απαντήσεις ο peer επιλέγει έναν από τους peers που απάντησαν και αφού δημιουργήσει μεταξύ τους μια TCP σύνδεση, κατεβάζει το επιθυμητό αρχείο απ' ευθείας από τον απομακρυσμένο peer. Παράδειγμα τέτοιου δικτύου είναι το Gnutella. Η κύρια αδυναμία αυτού του τύπου δικτύων είναι ότι δεν κλιμακώνουν καλά όταν το δίκτυο μεγαλώνει πολύ. Αυτό συμβαίνει γιατί μεγαλώνει παράλληλα και το μέγεθος της κίνησης των queries με αποτέλεσμα να καταναλώνεται σημαντικό μέρος του διαθέσιμου bandwidth μόνο για το "overhead" του πρωτοκόλλου.

Στα υβριδικά δίκτυα για να συνδεθεί ένας peer στο δίκτυο πρέπει να επικοινωνήσει αρχικά με έναν super-peer ο οποίος έχει μεγάλο διαθέσιμο bandwidth και να τον ενημερώσει για το ποια αρχεία έχει στη διάθεση του, από εκεί και πέρα ο συγκεκριμένος super-peer είναι υπεύθυνος για αυτόν. Με αυτή την αρχιτεκτονική δικτύου τα queries δεν είναι ανάγκη να γίνονται flood σε όλο το δίκτυο αλλά μόνο μεταξύ των super-peer οι οποίοι μπορούν να διαχειριστούν την μεγάλη κίνηση που διέρχεται από αυτούς. Με αυτόν τον τρόπο λύνεται επιτυχώς το θέμα της κλιμάκωσης σε μεγάλα δίκτυα, αλλά η συγκεκριμένη αρχιτεκτονική είναι ιδιαίτερα σύνθετη και είναι δύσκολη στην συντήρησή της. Παράδειγμα τέτοιου δικτύου αποτελεί το Kazaa.

The BitTorrent Protocol

Το Bit-Torrent είναι ένα peer-to-peer πρωτόκολλο για τον διαμοιρασμό (μεγάλων) αρχείων μεταξύ πολλαπλών peers οι οποίοι βρίσκονται πάνω σε αναξιόπιστα δίκτυα. Υλοποιήθηκε από τον Bram Cohen τον Απρίλιο του 2001 με την πρώτη έκδοση να διατίθεται τον Ιούλιο του ίδιου έτους.

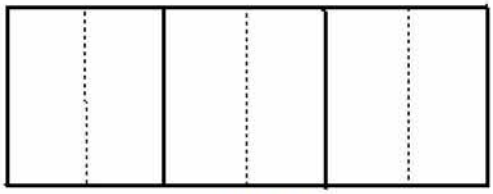
Λόγω της απόκεντροποιημένης φύσης του (όσον αφορά τον καθ' αυτό διαμοιρασμό του αρχείου) διαχειρίζεται πιο ορθολογικά το διαθέσιμο bandwidth του δικτύου, καθώς χρησιμοποιεί το upload bandwidth όλων των κόμβων που συμμετέχουν στη μεταφορά του αρχείου, σε αντίθεση με το κλασικό μοντέλο client-server που περιορίζεται μόνο στο upload του/των εκάστοτε server (bottleneck). Επίσης όσο περισσότεροι peer ολοκληρώνουν το αρχείο ή τμήματα αυτού, οπότε και συμμετέχουν με τη σειρά τους στην διαδικασία διαμοιρασμού, τόσο μειώνετε και η απαίτηση ο αρχικός peer να διαθέτει ισχυρούς πόρους από πλευράς δικτύου ή hardware. Τέλος το πρωτόκολλο είναι αρκετά ανεκτικό σε προβλήματα συστήματος καθώς η σημαντικότητα κάθε κόμβου ξεχωριστά είναι σχετικά μικρή, εξαιρουμένου φυσικά του tracker και του αρχικού seeder, οι οποίοι όμως καλούνται να διαχειριστούν μόνο ένα μικρό κομμάτι της συνολικής κίνησης καθώς τη μερίδα του λέοντος αναλαμβάνουν τα swarms των peer.

Τα παραπάνω αυτά χαρακτηριστικά έκαναν το συγκεκριμένο πρωτόκολλο ένα από τα ποιο διαδεδομένα πρωτόκολλα διαδικτύου τα τελευταία χρόνια όσον αφορά τον διαμοιρασμό αρχείων. Ακόμα και υπηρεσίες όπως το Facebook και το Twitter χρησιμοποιούν το συγκεκριμένο πρωτόκολλο για να μεταφέρουν τα updates στους server που διαθέτουν. Το μεγαλύτερο κομμάτι κίνησης που αφορά το πρωτόκολλο BitTorrent αφορά κυρίως αρχεία βίντεο και ήχου τα οποία διαμοιράζονται μεταξύ χρηστών του διαδικτύου. Πολλά από αυτά τα αρχεία μοιράζονται όμως παράνομα καθώς αποτελούν αντίγραφα αρχείων τα οποία προστατεύονται από άδειες πνευματικού περιεχομένου. Ουσιαστικά το BitTorrent αποτέλεσε συνέχεια των προηγούμενων πρωτοκόλλων peer-to-peer όπως το Gnutella.

Ακολουθεί η απαραίτητη επεξήγηση κάποιων σημαντικών όρων για την

κατανόηση της λειτουργίας του πρωτοκόλλου :

- **Tracker** : Αποτελεί ουσιαστικά ένα server που είναι υπεύθυνος για να οργανώσει όσους ενδιαφέρονται για το ίδιο αρχείο σε μια ομάδα (swarm). Αρκετά συχνά διατηρεί και τα αρχεία .torrent χωρίς αυτό να είναι πάντως απαραίτητο.
- **Peer** :Είναι ο τελικός χρήστης του πρωτοκόλλου που επιθυμεί να κατεβάσει ή να μοιραστεί ένα αρχείο.Ανάλογα με την λειτουργία που επιτελεί χωρίζεται σε leecher και seeder.
- **Seeder** : Οποιοσδήποτε peer έχει ολόκληρο το επιθυμητό αρχείο και συνεχίζει να το διαμοιράζει στους υπόλοιπους peers. Για την ομαλή λειτουργία του πρωτοκόλλου είναι απαραίτητη η ύπαρξη τουλάχιστον ενός seeder στο σύστημα. .
- **Leecher**: Οποιοσδήποτε peer δεν έχει ολοκληρώσει ακόμα το download του αρχείου. Συμμετέχει όμως στο upload με όση πληροφορία διαθέτει.
- **Client**: Μία εφαρμογή που υλοποιεί το πρωτόκολλο στην πλευρά του peer. Υπάρχει πληθώρα τέτοιων εφαρμογών όπως τα μTorrent, Azureus (Windows) και Transmission (Linux).
- **Αρχείο torrent**: Ένα αρχείο κειμένου που βοηθά τον client να επικοινωνήσει επιτυχώς με τον tracker, παρέχοντάς του τις απαραίτητες πληροφορίες που χρειάζεται. Οι πληροφορίες που περιέχει καθώς και ο τρόπος που είναι δομημένες ορίζονται αυστηρά από το πρωτόκολλο.
- **Swarm** : Μια ομάδα από peers στην οποία συμμετέχει όποιος ενδιαφέρεται να κατεβάσει ένα αρχείο. Τη λίστα με αυτούς τους peer διανέμει ο tracker.
- **Piece** : Κάθε αρχείο χωρίζεται σε έναν αριθμό piece ίδιου μεγέθους. Όταν ένας peer ολοκληρώσει (και επαληθεύσει) ένα piece μπορεί να αρχίσει να το διαμοιράζει.
- **Block** : Κάθε piece με τη σειρά του χωρίζεται και αυτό σε ένα αριθμό blocks ίδιου μεγέθους. Τα blocks είναι το κομμάτι πληροφορίας που μπορεί να αποσταλεί και να ληφθεί μέσω των μηνυμάτων response.



ΑΡΧΕΙΟ
με πλήρεις γραμμές παρουσιάζονται τα piece του αρχείου
με διακεκομμένες παρουσιάζονται τα blocks του αρχείου

Σε γενικές γραμμές το πρωτόκολλο λειτουργεί ως εξής :

- Ο peer κατεβάζει ένα αρχείο με κατάληξη .torrent, αυτό αποτελεί ουσιαστικά ένα text αρχείο με τις απαραίτητες πληροφορίες για να ξεκινήσει η διαδικασία download του επιθυμητού αρχείου.
- Εν συνεχεία χρησιμοποιώντας έναν bit-Torrent client (bitTorrent, uTorrent,Transmission etc) ανοίγει το .torrent αρχείο που κατέβασε. Το αρχείο αυτό περιλαμβάνει τις εξής πληροφορίες.
    i. Την διεύθυνση του tracker (IP address).
    ii. Την SHA1 τιμή που αντιστοιχεί στο συγκεκριμένο torrent.
    iii. Τον τρόπο με τον οποίο πρέπει να χωρίσει ο client το αρχείο σε pieces και blocks, καθώς και πώς να επαληθεύσει στο τέλος το αρχείο που κατέβασε.
- **(Εικόνα 1)** Με τις πληροφορίες που περιλαμβάνει αυτό το αρχείο, ο client επικοινωνεί με τον tracker και τον ενημερώνει ότι επιθυμεί να συμμετέχει στην διαδικασία διαμοιρασμού του εν λόγω αρχείου. Πιο συγκεκριμένα στέλνει ένα HTTP GET request στην διεύθυνση του tracker περιλαμβάνοντας τις εξής πληροφορίες.
    i. Την SHA1 τιμή του αρχείου που επιθυμεί (ΑΠΑΡΑΙΤΗΤΟ)
    ii. Ένα string ID μεγέθους 20 byte το οποίο δημιουργεί ο ίδιος ο peer με κάποια συνάρτηση παραγωγής τυχαίων strings.Το string αυτό θα χρησιμεύει για να αναγνωρίζεται ο peer μέσα στο swarm στο οποίο θα συμμετέχει (ΑΠΑΡΑΙΤΗΤΟ).
        - Παρότι υπάρχει η πιθανότητα το συγκεκριμένο ID να μην είναι μοναδικό, αυτή είναι πολύ μικρή λόγω κυρίως του μεγέθους του

string.

iii. Τον αριθμό της port στην οποία ακούει ο peer. Το πρωτόκολλο δεν θέτει περιορισμό στο ποια πρέπει να είναι αυτή οπότε ο peer μπορεί να αναθέσει μια αυθαίρετα (ΑΠΑΡΑΙΤΗΤΟ).

iv. Τον αριθμό τον bytes τα οποία έχει κάνει έως τώρα upload ο peer (ΑΠΑΡΑΙΤΗΤΟ).

v. Τον αριθμό τον bytes τα οποία έχει κάνει έως τώρα download ο peer (ΑΠΑΡΑΙΤΗΤΟ).

vi. Τον αριθμό των bytes τα οποία υπολείπονται από τον peer για να ολοκληρώσει το αρχείο (ΑΠΑΡΑΙΤΗΤΟ).

vii. Την IP διεύθυνση του client (ΠΡΟΑΙΡΕΤΙΚΟ).

viii. Τον αριθμό των peer πού επιθυμεί ώστε να δημιουργήσει το swarm με το οποίο θα επικοινωνεί (ΠΡΟΑΙΡΕΤΙΚΟ).

ix. Μία εκ των παρακάτω τιμών ώστε ο tracker να μπορεί να καταλάβει τον λόγω για τον οποίο στάλθηκε το μήνυμα (ΠΡΟΑΙΡΕΤΙΚΟ).

- Started : Με αυτό ο peer δηλώνει ότι στέλνει το πρώτο του GET request στον tracker.
- Stopped : Ο peer δηλώνει ότι θα κλείσει τις συνδέσεις του.
- Completed : Ο peer ολοκλήρωσε επιτυχώς το αρχείο
- Αν δεν περιλαμβάνετε συγκεκριμένη τιμή τότε το μήνυμα θεωρείται αυτομάτως σαν περιοδικό update προς τον tracker με σκοπό να τον ενημερώσει για την τωρινή κατάσταση καθώς και να αποφύγει τον περιοδικό καθαρισμό της λίστας των peers από την πλευρά του tracker.

- **(Εικόνα 2)** Ο tracker αφού λάβει την αίτηση συμμετοχής από τον client τον εγγράφει στην λίστα των peer και του αποστέλλει ένα αρχείο με τα εξής πεδία πληροφορίας.

    i. Την χρονική περίοδο μέσα στην οποία ο peer πρέπει να ανανεώνει την εγγραφή του στον tracker (ΑΠΑΡΑΙΤΗΤΟ).

    ii. Μία λίστα με τις IP διευθύνσεις τα string Ids καθώς και τις ports

ενός αριθμού ενεργών peers (ΑΠΑΡΑΙΤΗΤΟ).

iii. Τον αριθμό των peers που κατεβάζουν το αρχείο (leechers) καθώς και τον αριθμό αυτών που το έχουν ολοκληρώσει (seeders) (ΠΡΟΑΙΡΕΤΙΚΟ).

iv. Σε περίπτωση αποτυχίας ο tracker ενημερώνει τον client περιλαμβάνοντας τον λόγω της αποτυχίας. Σε περίπτωση που αυτό το πεδίο δεν είναι κενό, τότε πρέπει να είναι κενά όλα τα υπόλοιπα (ΠΡΟΑΙΡΕΤΙΚΟ).

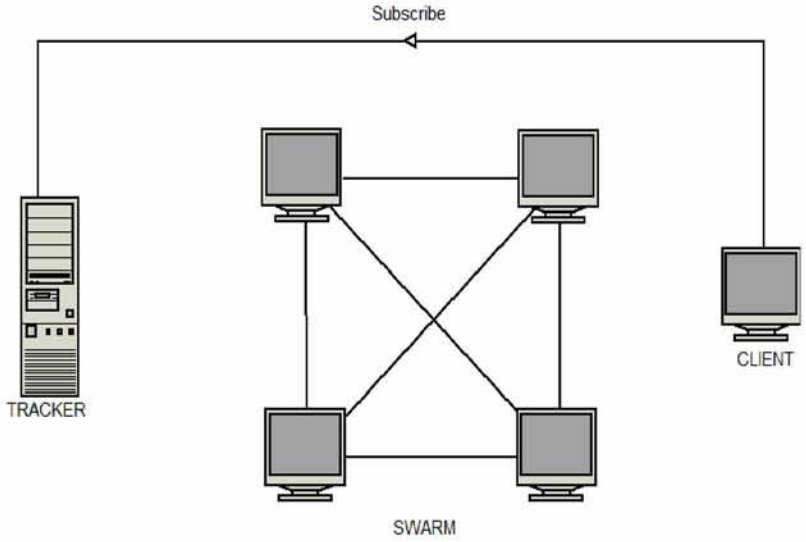
- Αφού λάβει την λίστα, ο client πρέπει να επικοινωνήσει με τους peers που βρίσκονται στη λίστα που έλαβε ώστε να δημιουργήσει τις απαραίτητες TCP συνδέσεις για την μεταφορά του αρχείου.
- **(Εικόνες 3,4)** Το πρώτο βήμα της peer-to-peer επικοινωνίας είναι η σύναψη "χειραψίας" με όλους τους peers που απέστειλε ο tracker. Αυτό επιτυγχάνετε με την αποστολή (στον απομακρυσμένο peer) και λήψη (από τον απομακρυσμένο peer) handshake μηνυμάτων τα οποία περιλαμβάνουν την SHA1 τιμή που αντιστοιχεί στο torrent καθώς και το μοναδικό αναγνωριστικό string του peer.
- Αφού ολοκληρωθεί η χειραψία μεταξύ δυο peer τότε θεωρείτε πώς υπάρχει σύνδεση μεταξύ των δυο peers η οποία μπορεί κάθε στιγμή να βρίσκεται στις εξής καταστάσεις.
    i. Interested/Not Interested : Με το να θέσει ένας peer την δική του πλευρά της σύνδεσης ως interested δηλώνει ότι επιθυμεί να κατεβάσει κάποια κομμάτια του αρχείο από τον peer στην άλλη άκρη της TCP σύνδεσης. Αντιθέτως αν τη θέσει not interested τότε δηλώνει πως δεν ενδιαφέρεται για κανένα από τα κομμάτια του άλλου peer.
    ii. Chocked/Unchocked : Αν μια πλευρά της σύνδεσης είναι chocked τότε ο αντίστοιχος peer δηλώνει πώς στην παρούσα κατάσταση δεν διατίθεται να αποστείλει δεδομένα του αρχείου στον άλλο peer.
- **(Εικόνα 5)** Κάθε σύνδεση μεταξύ δυο peer ξεκινά ως Chocked/Not Interested και από τις δύο πλευρές καθώς δεν γνωρίζει ακόμα η μία πια κομμάτια του αρχείου

διαθέτει η άλλη. Το πρόβλημα αυτό μπορεί να ξεπεραστεί με δυο τρόπους, είτε με την αποστολή/λήψη ενός μηνύματος bitfield είτε ενός μηνύματος have.

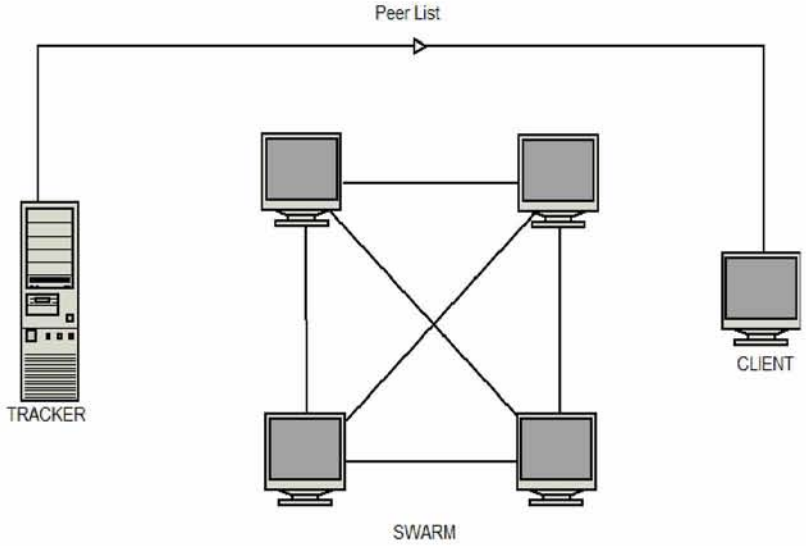
i. Bitfield : Αποστέλνεται συνήθως με την ολοκλήρωση της χειραψίας μεταξύ δυο peer και αποτελεί ουσιαστικά ένα δυαδικό αριθμό μεγέθους ίσο με τον αριθμό των pieces του αρχείου, όπου το 0 αντιπροσωπεύει ένα piece το οποίο δεν έχει διαθέσιμο ο peer και αντίστοιχα το 1 αντιπροσωπεύει ένα piece το οποίο ο peer έχει κατεβάσει άλλα και επαληθεύσει επιτυχώς. **(Εικόνα 6)**
ii. Have : Αποστέλνεται προς όλους τους peers κάθε φορά που ο συγκεκριμένος peer κατεβάσει και επαληθεύσει ένα piece του αρχείου και περιλαμβάνει τον αριθμό του συγκεκριμένου piece. **(Εικόνα 7)**

- Με το που λάβει ένας peer ένα από τα παραπάνω δυο μηνύματα ελέγχει αν τον ενδιαφέρει κάποιο piece που διαθέτει ο peer στην άλλη άκρη της σύνδεσης, και ανάλογα στέλνει ή όχι ένα μήνυμα interested προς τον συγκεκριμένο peer με σκοπό να ενεργοποιήσει την μεταξύ τους σύνδεση. **(Εικόνα 8)**
- Αν ένας peer λάβει ένα μήνυμα interested τότε αν διαθέτει ελεύθερα Upload slots στέλνει πίσω ένα μήνυμα Unchocked. **(Εικόνα 9)**
- Αν ένας peer λάβει μήνυμα Unchocked τότε είναι ελεύθερος να ζητήσει τα κομμάτια του αρχείου που τον ενδιαφέρουν. Αυτό γίνεται μέσω ενός μηνύματος request στο οποίο ο peer διευκρινίζει πιο block ποιου piece τον ενδιαφέρει. Ο τρόπος με τον οποίο ο peer επιλέγει ποιο κομμάτι του αρχείου θα κατεβάσει κάθε στιγμή δεν διευκρινίζεται από το πρωτόκολλο οπότε αυτό αποφασίζεται από τον εκάστοτε client. **(Εικόνα 10)**
- Αφού λάβει ένα μήνυμα request ο peer αποστέλλει τα ζητούμενα δεδομένα στον άλλο peer, αναφέροντας και ο ίδιος τον αριθμό piece-block στον οποίο αναφέρονται τα συγκεκριμένα δεδομένα . **(Εικόνα 11)**
- Σε περίπτωση που ο peer ο οποίος ζητάει τα δεδομένα, τα λάβει μέσω ενός άλλου peer τότε αποστέλλει στους υπόλοιπους peers από τους οποίους το ζήτησε ένα μήνυμα cancel.

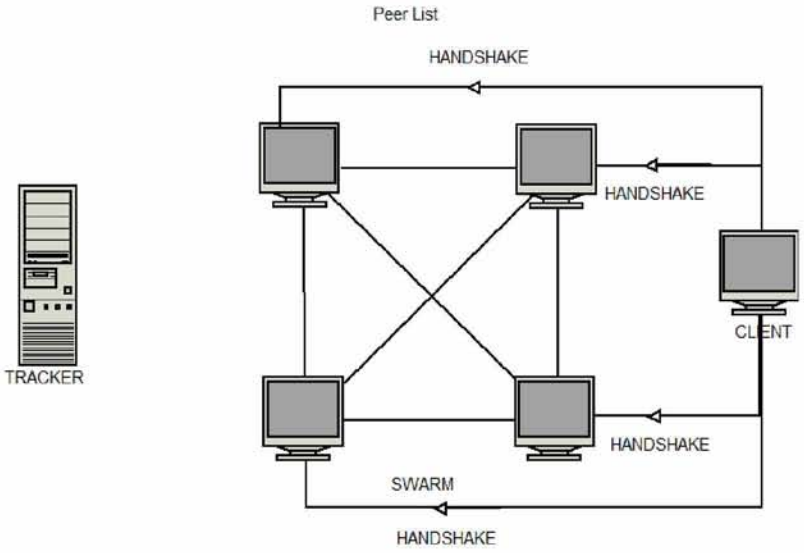
Ακολουθεί μια απλοποιημένη διαγραμματική αναπαράσταση της παραπάνω περιγραφής.



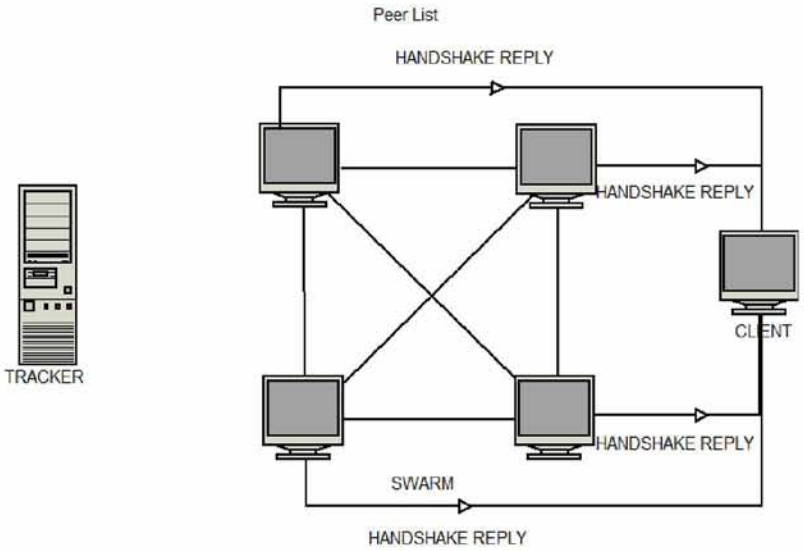
Εικόνα 1



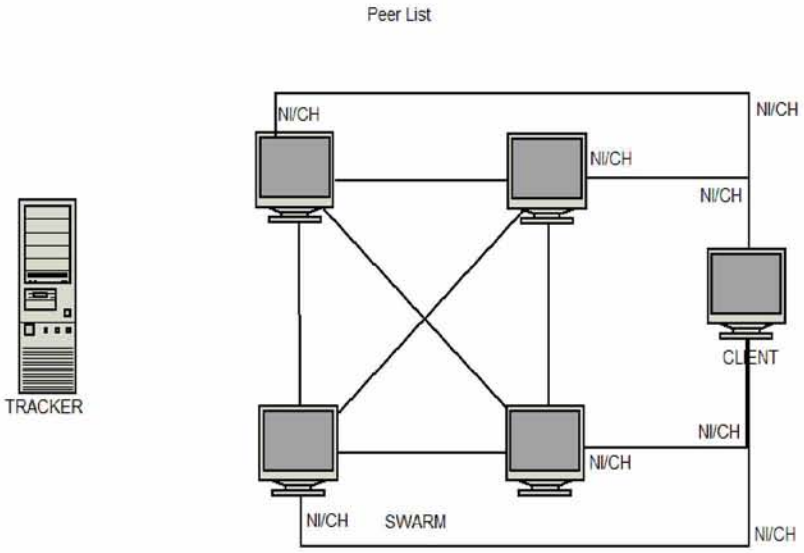
Εικόνα 2



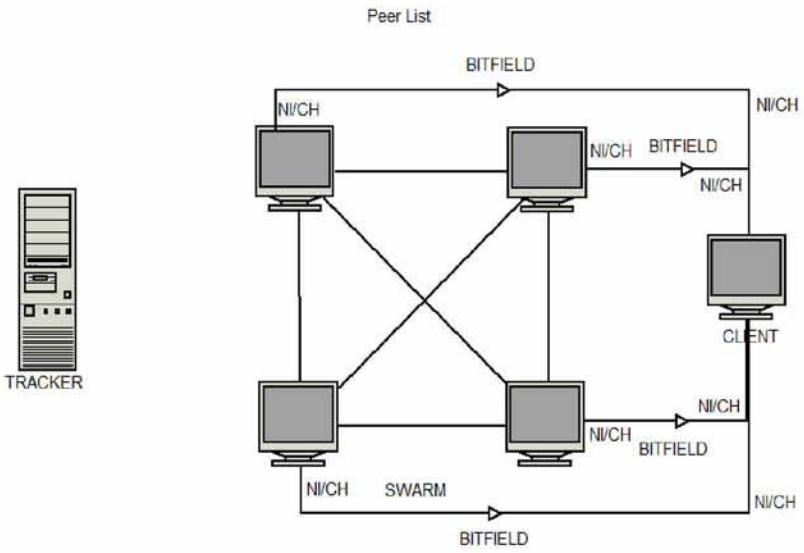
Εικόνα 3



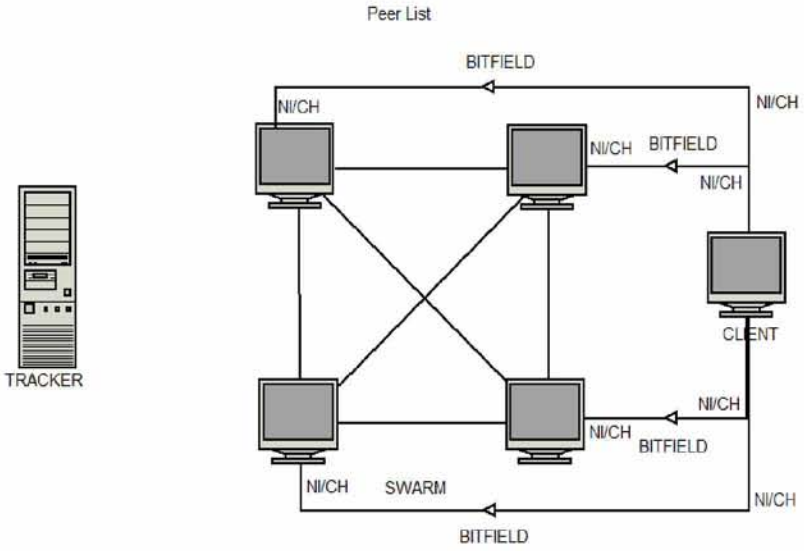
Εικόνα 4



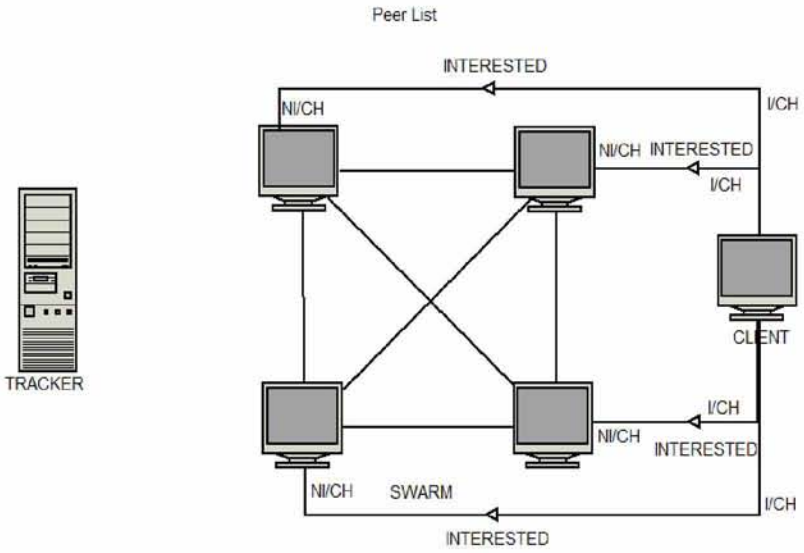
Εικόνα 5



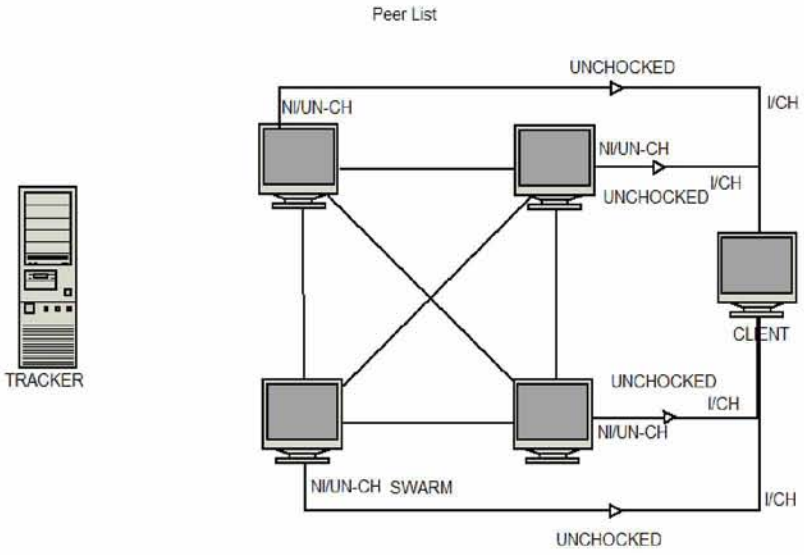
Εικόνα 6



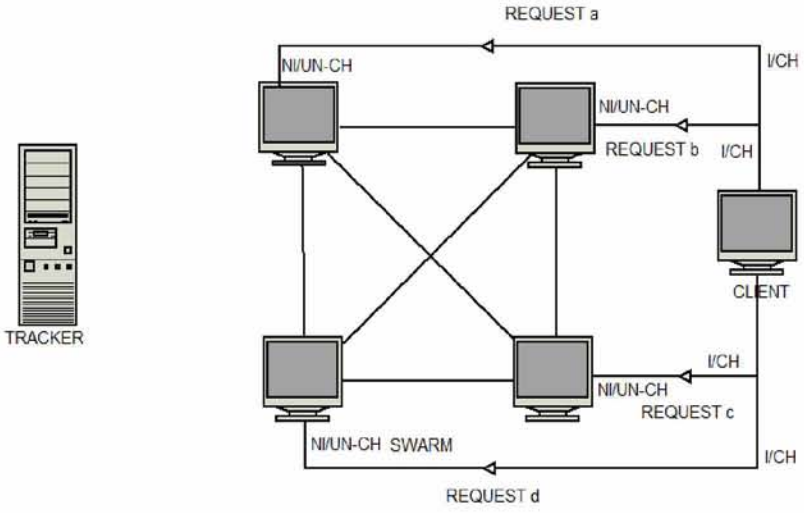
Εικόνα 7



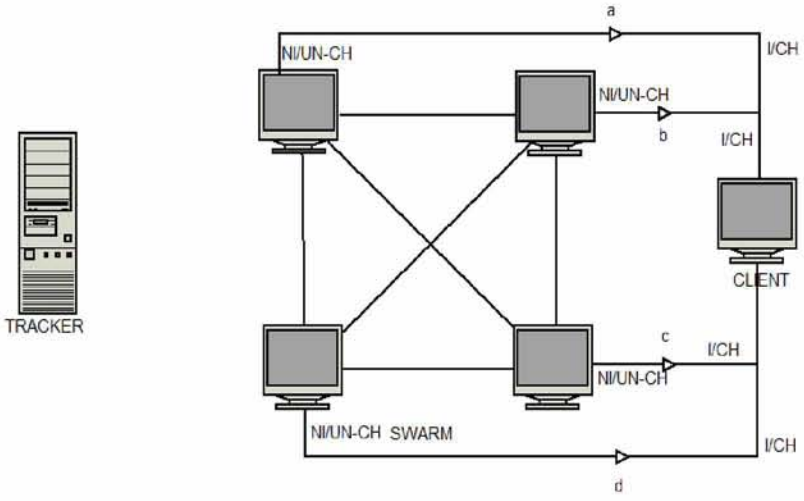
Εικόνα 8



Εικόνα 9



Εικόνα 10


TRACKER
a
NI/UN-CH
I/CH
NI/UN-CH
b
I/CH
CLIENT
c
I/CH
NI/UN-CH
NI/UN-CH SWARM
I/CH
d

Εικόνα 11

ΠΡΟΣΟΜΟΙΩΣΗ ΔΙΚΤΥΩΝ

Ένας προσομοιωτής δικτύου αποτελεί ουσιαστικά ένα software ή hardware το οποίο μοντελοποιεί τη συμπεριφορά ενός δικτύου χωρίς να είναι απαραίτητη η φυσική ύπαρξη αυτού. Χρησιμοποιούνται κυρίως από μηχανικούς δικτύων και ερευνητές γιατί τους δίνουν τη δυνατότητα να δοκιμάσουν πειράματα τα οποία θα ήταν πολύ δύσκολο και ακριβό να γίνουν στο φυσικό επίπεδο. Κύριες χρήσεις είναι η μελέτη της συμπεριφοράς νέων πρωτοκόλλων, είτε ήδη υπαρχόντων στα οποία έχουν γίνει αλλαγές με στόχο την βελτίωση της απόδοσης. Η μελέτη συμπεριφοράς ενός πρωτοκόλλου απαιτεί συνήθως πολύ μεγάλο αριθμό κόμβων ώστε να θεωρηθεί επαρκής, πόρους στους οποίους πολύ σπάνια έχει κάποιος πρόσβαση, ενώ με τη χρήση προσομοιωτών η ανάγκη για πόρους μειώνετε δραματικά.

Οι προσομοιωτές αυτοί χωρίζονται σε δύο κατηγορίες. Τους discrete event simulators και τους Markov-Chain based simulators. Οι πιο συνηθισμένοι είναι οι discrete event simulators, σε αυτούς κάθε γεγονός εκτελείτε σε μια προκαθορισμένη σειρά με κάποια γεγονότα να οδηγούν στη δημιουργία κάποιων άλλων (όπως για παράδειγμα η αποστολή ενός πακέτου μέσω TCP οδηγεί στην αποστολή ενός ACK πακέτου από τον παραλήπτη). Στους Markov-Chain simulators οι οποίοι βασίζονται στην θεωρία ουρών, δεν διατηρείτε κάποια λίστα με τα γεγονότα που θα υπάρξουν στο μέλλον , αλλά η προσομοίωση γίνεται μέσω μεταβάσεων χωρίς μνήμη, αν και αυτός ο τρόπος προσομοίωσης είναι πιο γρήγορος είναι και λιγότερο ακριβής και ελαστικός από τον discrete-event.

Οι πιο διαδεδομένοι προσομοιωτές αυτή τη στιγμή είναι οι εξής:

- ns-2
- ns-3
- OMNET++

Οι προσομοιωτές ns (network simulator) είναι της κατηγορίας discrete event και αποτελούν ελεύθερο λογισμικό (GNU GPLv2 license for research, development).Ο ns-2 δημιουργήθηκε το 1996-1997 σαν συνέχεια του ns-1 (1995) είναι γραμμένος κυρίως σε C++ αλλά και με την oTcl μια αντικειμενοστραφή scripting language η

οποία χρησιμοποιούνταν κυρίως για τις καθ' αυτό προσομοιώσεις. Ο ns-3 δημιουργήθηκε το 2006 χωρίς όμως να υποστηρίζει κανένα πρωτόκολλο που είχε φτιαχτεί για τον ns-2 (δηλαδή χωρίς backwards compatibility), γράφτηκε και αυτός σε C++ ενώ οι προσομοιώσεις υποστηρίζουν C++ και Python. Τα κυρίως μειονεκτήματα με τους προσομοιωτές ns είναι η έλλειψη γραφικού περιβάλλοντος η οποία κάνει δύσκολη την εξοικείωση με το προσομοιωτή, η έλλειψη σταθερών αποτελεσμάτων λόγω των συνεχών αλλαγών στα υπάρχοντα πρωτόκολλα και τέλος το γεγονός ότι με την μετάβαση στον ns-3 χάθηκε η υποστήριξη για ένα πολύ μεγάλο αριθμό πρωτοκόλλων που υπήρχαν στον ns-2.

Ο προσομοιωτής OMNET έχει μεγαλύτερη γκάμα εφαρμογών καθώς μπορεί να χρησιμοποιηθεί και για άλλες προσομοιώσεις εκτός από δικτύων, αν και χρησιμοποιείτε κατά κόρων για δίκτυα. Έχει ένα δικό του IDE το οποίο βασίστηκε στο Eclipse ενώ διαθέτει και ένα ξεχωριστό GUI για την γραφική παρουσίαση των προσομοιώσεων (Tkenv). Οι τοπολογίες δικτύων ορίζονται μέσω της NED topology description language. Αν και από μόνος του δεν υποστηρίζει αρχικά πολλά πρωτόκολλα , έχει μια μεγάλη γκάμα εξωτερικών frameworks όπως το INET που συμπληρώνουν την λειτουργικότητα του Omnet++ με έτοιμες υλοποιήσεις πρωτοκόλλων.

Προσωπικά αφού ασχολήθηκα και με τους δύο προσομοιωτές (ns omnet++) βρήκα τον Omnet++ πιο προσιτό για τα πρώτα βήματα στον τομέα της προσομοίωσης δικτύων κυρίως λόγω του πολύ χρήσιμου GUI για τα τρεξίματα των προσομοιώσεων αλλά και της μεγαλύτερης και πιο ενεργής κοινότητας χρηστών. Έτσι αποφάσισα να ολοκληρώσω το προγραμματιστικό κομμάτι της πτυχιακής εργασίας μου σε αυτόν τον προσομοιωτή.

ΥΛΟΠΟΙΗΣΗ ΤΟΥ BITTORRENT ΣΤΟΝ OMNET++

Για την μελέτη του πρωτοκόλλου επιλέχθηκε η προσομοίωση έναντι της μαθηματικής μοντελοποίησης καθώς πιστεύω δίνει την ευκαιρία για μια ποιο βαθιά κατανόηση του πρωτοκόλλου καθώς και την δυνατότητα hands-on πειραματισμού με ρεαλιστικά σενάρια.

Η υλοποίηση του πρωτοκόλλου Bit-torrent που έκανα αφορά τη ραχοκοκαλιά του κανονικού πρωτοκόλλου καθώς δεν υλοποιήθηκαν κάποιες μικρές λεπτομέρειες οι οποίες δεν είχαν ουσιαστική επίδραση στην απόδοση του πρωτοκόλλου όταν αυτό έτρεχε σε προσομοιωτή. Τέτοιο παράδειγμα αποτελούν τα περιοδικά keep Alive μηνύματα τα οποία στέλνουν κανονικά οι peers στον tracker για να παραμείνουν στο swarm το οποίο βρίσκονται.

Τα κομμάτια τα οποία υλοποιήθηκαν όπως ορίζει το πρωτόκολλο είναι τα εξής

1. Η εγγραφή του κάθε peer στον tracker η "διεύθυνση" του οποίου θεωρείται εξ αρχής γνωστή.
2. Η απάντηση του tracker που ενημερώνει παράλληλα τους peers για τις "διευθύνσεις" των υπόλοιπων.
3. Η διαδικασία των χειραψιών μεταξύ των peers.
4. Η αποστολή των μηνυμάτων have και bitField.
5. Η διαδικασία choke-unchoke interested uninterested στις συνδέσεις μεταξύ των peers.
6. Τα μηνύματα request response για τα εκάστοτε blocks του αρχείου.

Όσον αφορά κάποιες επιπλέον λεπτομέρειες για την υλοποίηση αξίζουν να σημειωθούν οι εξής.

1. Στη υλοποίηση δεν χρησιμοποιούνται real life πρωτόκολλα δρομολόγησης της κίνησης μέσα στην τοπολογία. Αλλά γίνεται χρήση της built in shortest path δρομολόγησης που προσφέρει το Omnet++.
2. Επίσης δεν γίνεται χρήση του πρωτοκόλλου TCP καθώς οι συνδέσεις μεταξύ των κόμβων δεν χάνουν μηνύματα. Κατά την διάρκεια της υλοποίησης

προσπάθησα να δημιουργήσω ένα μηχανισμό drop και retransmit στα μηνύματα αλλά συνάντησα αρκετές δυσκολίες και τον άφησα ανεκπλήρωτο. Παρόλα αυτά έχω βάλει το απαραίτητο TCP overhead στα μηνύματα που αποστέλλονται για να μην γίνει παραποίηση της απόδοσης του πρωτοκόλλου.

3. Όσον αφορά την διαδικασία του download των pieces, καθώς αυτή δεν ορίζεται από το πρωτόκολλο, ακολούθησα την στρατηγική που ακολουθούν οι κλασικοί clients. Για την επιλογή του κομματιού χρησιμοποιούνται 2 αλγόριθμοι, ο random piece και ο rarest first. Κάθε peer ξεκινά εφαρμόζοντας τον random και σε κάποια στιγμή (ορίζεται από το πρόγραμμα) αλλάζει στον rarest first. Όσον αφορά τώρα την επιλογή του peer από όπου θα γίνει το request αυτή γίνεται random μεταξύ όλων των peer που μπορούν να το διαθέσουν.

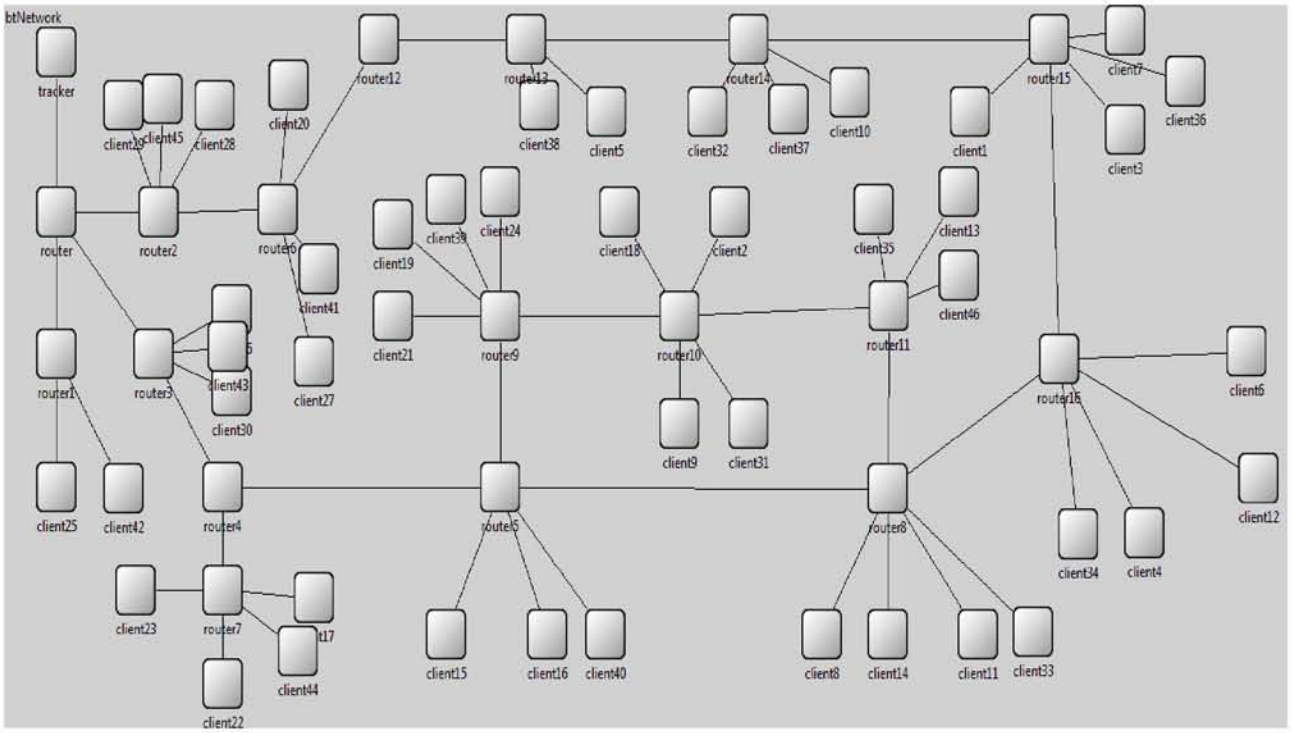
Για τον έλεγχο εγκυρότητας του προγράμματος κάθε γεγονός που συμβαίνει, όπως οι αποστολές-λήψεις μηνυμάτων καταγράφονται με κατάλληλα μηνύματα log.

Παρακάτω ακολουθεί ένα παράδειγμα εξόδου του προγράμματος με τις εξής παραμέτρους εισόδου

1. 46 Peers
2. 1 Seeder
3. 1 Torrent
4. 25 pieces των 5 blocks
5. Random Piece στην αρχή και Rarest First αργότερα όσον αφορά το piece selection

Για λόγους οικονομίας χώρου παραθέτω και εξηγώ μόνο κομμάτια του συνολικού log

Αρχικά παρουσιάζεται η τοπολογία που χρησιμοποιήθηκε για την πλειονότητα των πειραμάτων.



Η τοπολογία όπως φαίνεται περιέχει 46 peers έναν tracker και κάποιους routers οι οποίοι εφαρμόζουν την διαδικασία δρομολόγησης των πακέτων στο δίκτυο. Οι συνδέσεις μεταξύ των κόμβων έχουν τα εξής χαρακτηριστικά.

1. Router to Router connection : delay = 100 μs
   bandwidth = 100 Mbps
2. Router to Peer connection : delay = 100 μs
   bandwidth = 6 Mbps
3. Router to Tracker connection : delay = 100 μs
   bandwidth = 100 Mbps

Όπως θα συνέβαινε σε ένα ρεαλιστικό σενάριο οι router και ο tracker έχουν αναλογικά ισχυρό bandwidth σε σχέση με τους peers ώστε να διαχειρίζονται την κίνηση που περνάει από αυτούς.

- **Στην αρχή του log γίνεται η αρχικοποίηση των κόμβων με τις απαραίτητες παραμέτρους καθώς επίσης παρουσιάζεται και η τοπολογία.**

Topology Description
The topology has 64 nodes
Node i=0 is btNetwork.tracker
 It has 1 conns to other nodes
 and 1 conns from other nodes
 Connections to other modules are:
 btNetwork.router through gate port$o[0]
Node i=1 is btNetwork.client1
 It has 1 conns to other nodes
 and 1 conns from other nodes
 Connections to other modules are:
 btNetwork.router15 through gate port$o[0]
Node i=2 is btNetwork.client2
 It has 1 conns to other nodes
 and 1 conns from other nodes
 Connections to other modules are:
 btNetwork.router10 through gate port$o[0]
...
...
Node i=62 is btNetwork.router15
 It has 6 conns to other nodes
 and 6 conns from other nodes
 Connections to other modules are:
 btNetwork.router14 through gate port$o[0]
 btNetwork.client3 through gate port$o[1]
 btNetwork.client7 through gate port$o[2]
 btNetwork.router16 through gate port$o[3]
 btNetwork.client1 through gate port$o[4]
 btNetwork.client36 through gate port$o[5]
Node i=63 is btNetwork.router16
 It has 6 conns to other nodes
 and 6 conns from other nodes
 Connections to other modules are:
 btNetwork.router15 through gate port$o[0]
 btNetwork.router8 through gate port$o[1]
 btNetwork.client6 through gate port$o[2]
 btNetwork.client4 through gate port$o[3]
 btNetwork.client12 through gate port$o[4]
 btNetwork.client34 through gate port$o[5]

Tracker: Initialisation successful
Initializing module btNetwork.client1, stage 0
CLIENT1: Initialisation successful with id ppKlPRA7GzII5qcxoaDu
CLIENT1: Sent a request message to the tracker for torrent with id= 1
Initializing module btNetwork.client2, stage 0
CLIENT2: Initialisation successful with id IYZSxdzi3KcrXKsgkLFZ
CLIENT2: Sent a request message to the tracker for torrent with id= 1
...
...
Initializing module btNetwork.client46, stage 0
CLIENT46: Initialisation successful with id 4RJuwMe4ZYnSxAOohHrh
CLIENT46: Sent a request message to the tracker for torrent with id= 1

- **Εν συνεχεία ακολουθεί η εγγραφή των peers στον tracker, η ενημέρωση των peers από τον tracker και ξεκινά η διαδικασία χειραψιών μεταξύ των peers.**

Tracker: Received a request message for torrent with id=1 from peer with id= CwXy7u5rBTyXcO0gapsl
TRACKER: Added torrent with id=1 to the database
TRACKER: Added a new peer with id=CwXy7u5rBTyXcO0gapsl and ip= 25 to torrent's 1 database
Tracker: Received a request message for torrent with id=1 from peer with id= DTvQOYKSLnhSK4stTD38
TRACKER: Added a new peer with id=DTvQOYKSLnhSK4stTD38 and ip= 26 to torrent's 1 database
Tracker: Received a request message for torrent with id=1 from peer with id= vXU4BdClualTfYrOexlX
TRACKER: Added a new peer with id=vXU4BdClualTfYrOexlX and ip= 28 to torrent's 1 database
...
...
Tracker: Received a request message for torrent with id=1 from peer with id= pC6TIQ4lsICPmXxyeda7
TRACKER: Added a new peer with id=pC6TIQ4lsICPmXxyeda7 and ip= 39 to torrent's 1 database

CLIENT25: Received an update message from the tracker for torrent with id= 1
CLIENT25: PeerList Updated
CLIENT25: Received an update message from the tracker for torrent with id= 1
CLIENT25: PeerList Updated
CLIENT25: Handshake Initiated for torrent with id= 1with peer 26
CLIENT25: Received an update message from the tracker for torrent with id= 1
...
...
CLIENT25: Handshake Initiated for torrent with id= 1with peer 45
CLIENT25: Received an update message from the tracker for torrent with id= 1
CLIENT25: PeerList Updated
CLIENT26: Handshake Initiated for torrent with id= 1with peer 45
CLIENT26: Received an update message from the tracker for torrent with id= 1
CLIENT26: PeerList Updated
CLIENT30: Handshake Initiated for torrent with id= 1with peer 45
CLIENT30: Received an update message from the tracker for torrent with id= 1
CLIENT30: PeerList Updated
CLIENT42: Handshake Initiated for torrent with id= 1with peer 45
CLIENT42: Received an update message from the tracker for torrent with id= 1
CLIENT42: PeerList Updated
CLIENT43: Handshake Initiated for torrent with id= 1with peer 45
CLIENT43: Received an update message from the tracker for torrent with id= 1
CLIENT43: PeerList Updated
CLIENT28: Handshake Initiated for torrent with id= 1with peer 45
CLIENT28: Received an update message from the tracker for torrent with id= 1
CLIENT28: PeerList Updated

- **Όταν γίνει η λήψη ενός μηνύματος χειραψίας εμφανίζεται το εξής.**

CLIENT25: Handshake Request Received for torrent with id= 1 from peer 42
CLIENT25: Handshake was a reply
Connection between nodes 25 and 42 is active
CLIENT: 25 sent a bit-field message to client 42
CLIENT26: Handshake Request Received for torrent with id= 1 from peer 20
CLIENT26: Handshake was a reply
Connection between nodes 26 and 20 is active
CLIENT: 26 sent a bit-field message to client 20

- **Μετά ακολουθούν τα μηνύματα interested-unchoce και request**

Client 30 sent an "interested" message to client 31
Client 17 sent an "interested" message to client 31
Client 43 sent an "interested" message to client 31
Client 22 sent an "interested" message to client 31
Client 23 sent an "interested" message to client 31
Client 44 sent an "interested" message to client 31
Client 25 sent an "interested" message to client 31
Client 29 sent an "interested" message to client 31
CLIENT 31 un-chocked client 30
Client 42 sent an "interested" message to client 31
Client 45 sent an "interested" message to client 31
CLIENT 31 un-chocked client 17
CLIENT 31 un-chocked client 43
CLIENT 31 un-chocked client 22
CLIENT 31 un-chocked client 23
CLIENT 31 un-chocked client 44
CLIENT 31 un-chocked client 25
CLIENT 31 un-chocked client 29
CLIENT 31 un-chocked client 42
CLIENT 31 un-chocked client 45
CLIENT18: Requested block (RP) 0 of piece 17 for torrent 1 from peer 1
CLIENT18: Requested block (RP) 1 of piece 17 for torrent 1 from peer 1
CLIENT18: Requested block (RP) 2 of piece 17 for torrent 1 from peer 1
CLIENT18: Requested block (RP) 3 of piece 17 for torrent 1 from peer 1
CLIENT18: Requested block (RP) 4 of piece 17 for torrent 1 from peer 1

- **Όταν ένας peer λάβει το τελευταίο block ενός piece στέλνει μηνύματα have.**

CLIENT18: Received block 2 of piece 12 for torrent 1 from client 7
CLIENT 18 has finished piece 12
CLIENT 18 sent a have message to client 25
CLIENT 18 sent a have message to client 26
CLIENT 18 sent a have message to client 28
CLIENT 18 sent a have message to client 29
CLIENT 18 sent a have message to client 30

- **Όταν ένας peer ολοκληρώσει το αρχείο τυπώνεται κατάλληλο μήνυμα και ο peer συνεχίζει να συμμετέχει ως seeder.**

CLIENT 39 finished at 3.79978

- **Όταν όλοι οι peers ολοκληρώσουν το αρχείο το πρόγραμμα τερματίζει ανακοινώνοντας τον συνολικό χρόνο τρεξίματος.**

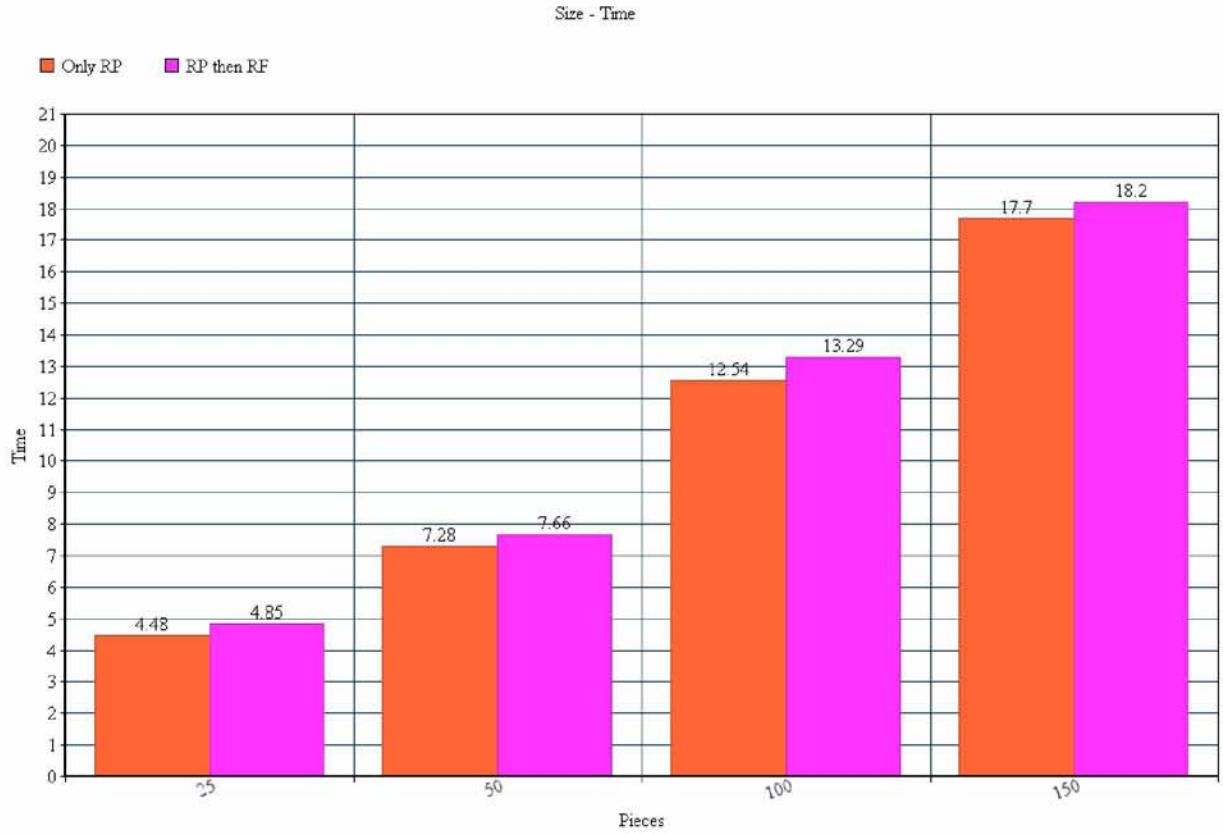
:

Στα πειράματα που έκανα πάνω στην υλοποίηση, επικεντρώθηκα στο πώς αλλαγές σε συγκεκριμένες παραμέτρους έχουν επίδραση στον χρόνο ολοκλήρωσης της προσομοίωσης. Οι παράμετροι με τις οποίες πειραματίστηκα ήταν

1. Ο αριθμός των peers.
2. Ο αριθμός των seeders.
3. Το διαθέσιμο bandwidth των peers.
4. Ο αλγόριθμος επιλογής κομματιού.
5. Το μέγεθος του αρχείου.

Στις επόμενες σελίδες ακολουθούν σχηματικά τα αποτελέσματα που έλαβα από τα πειράματα. Βάσει των αποτελεσμάτων που έλαβα κατέληξα και σε κάποια χρήσιμα συμπεράσματα γύρω από τη λειτουργία του πρωτοκόλλου.

Πείραμα Πρώτο: Σχέση μεταξύ μεγέθους αρχείου και συνολικού χρόνου υλοποίησης



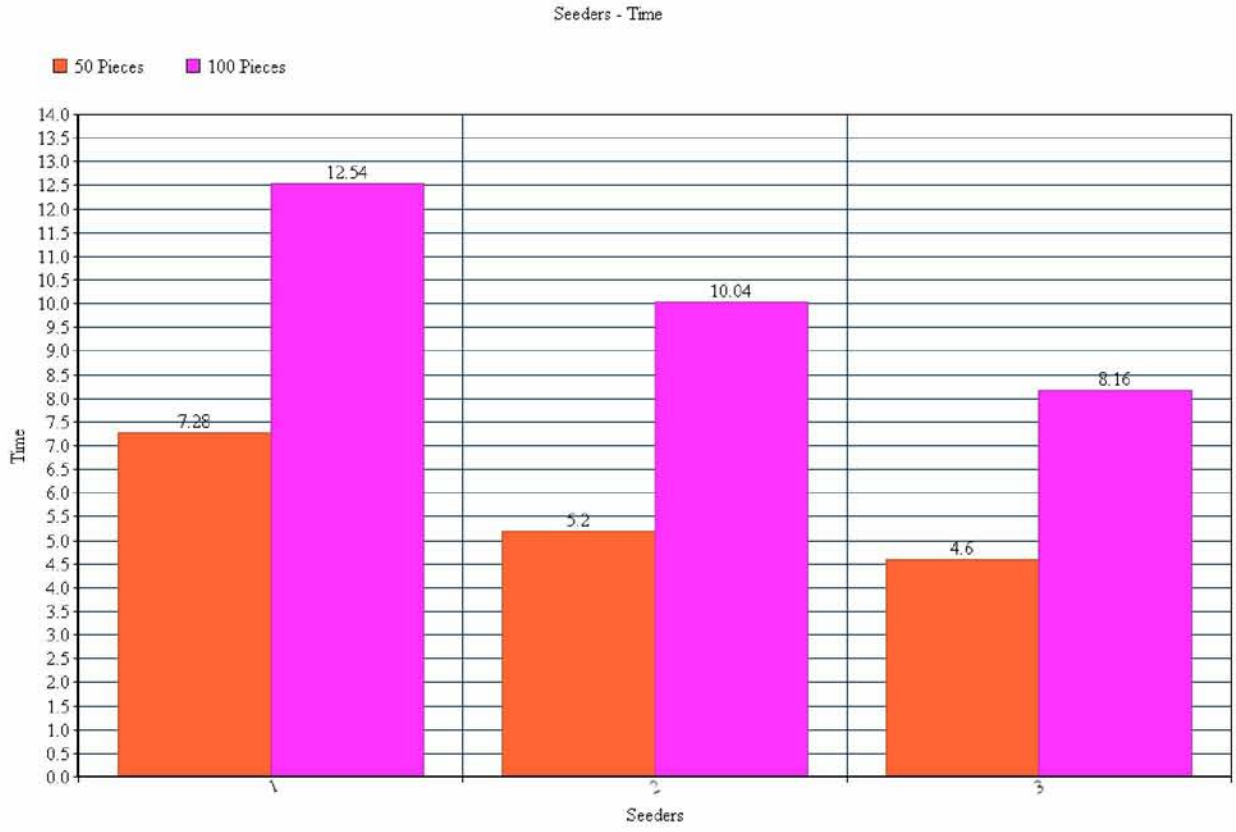
Παραπάνω φαίνεται καθαρά η σχετικά γραμμική κλιμάκωση του πρωτοκόλλου στον χρόνο καθώς μεγαλώνει το μέγεθος του αρχείου. Επίσης φαίνεται πώς και οι δύο μέθοδοι επιλογής piece έχουν σχετικά παρόμοια συμπεριφορά. Αν και ο αλγόριθμος rarest first είναι κάπως πιο αργός διασφαλίζει ότι αν φύγουν κάποιοι peers από το σύστημα τότε τα κομμάτια που αυτοί διαθέτουν θα είναι διαθέσιμα και από αλλού.

Η καθυστέρηση που εμφανίζει ο αλγόριθμος οφείλεται (κατά πάσα πιθανότητα) στο bottleneck που δημιουργείτε όταν ζητάνε όλοι οι peers το ίδιο piece από έναν seeder. Η θετική συνεισφορά του αλγορίθμου rarest first δεν είναι δυνατό να καταγραφεί από την παρούσα υλοποίηση καθώς αυτή δεν υποστηρίζει τυχαία είσοδο-έξοδο peer κατά τη διάρκεια της προσομοίωσης.

Στο συγκεκριμένο πείραμα έλαβα υπόψιν τον συνολικό χρόνο του

προγράμματος, αλλά και ο μέσος χρόνος για κάθε peer έδινε παρόμοια αποτελέσματα.

Πείραμα Δεύτερο: Σχέση μεταξύ αριθμού initial seeders και συνολικού χρόνου υλοποίησης



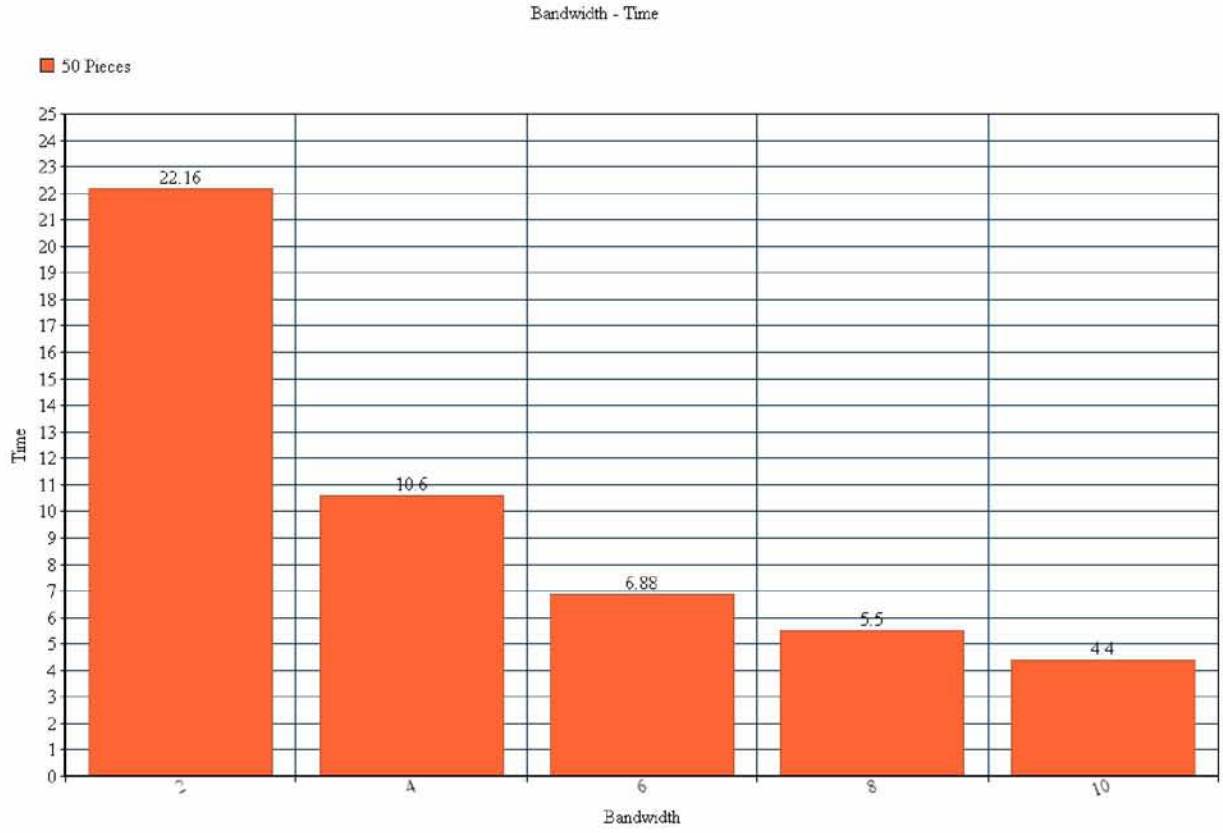
Από το παραπάνω διάγραμμα μπορούμε να βγάλουμε ακόμα ένα ενδιαφέρον συμπέρασμα για το πρωτόκολλο bittorrent. Φαίνεται ξεκάθαρα η βελτίωση στην απόδοση του πρωτοκόλλου όταν αυξάνουμε τον αριθμό των αρχικών seeders. Αυτή η βελτίωση συμβαίνει γιατί μειώνεται το φαινόμενο του bottleneck που συμβαίνει στην αρχή μέχρι να ομογενοποιηθεί κάπως το σύστημα.

Γενικότερα η ανάγκη για μια καλή αναλογία seeders-leechers είναι ένα από τα σημαντικότερα προβλήματα του Bittorrent. Ιδιαίτερα αν αναλογιστεί κανείς ότι συνήθως το download bandwidth είναι συνήθως μια κλίμακα μεγαλύτερο από το upload bandwidth αυτό το πρόβλημα γίνεται πιο έντονο. Μια λύση που έχει δοθεί

είναι η χρήση των seedbox δηλαδή seeders που διαθέτουν πολύ μεγάλο upload bandwidth συγκριτικά με αυτό ενός απλού χρήστη και οι οποίοι καλούνται να καλύψουν το παραπάνω χάσμα.

Όπως και στο πρώτο πείραμα έχω λάβει υπόψιν τους συνολικούς χρόνους υλοποίησης.

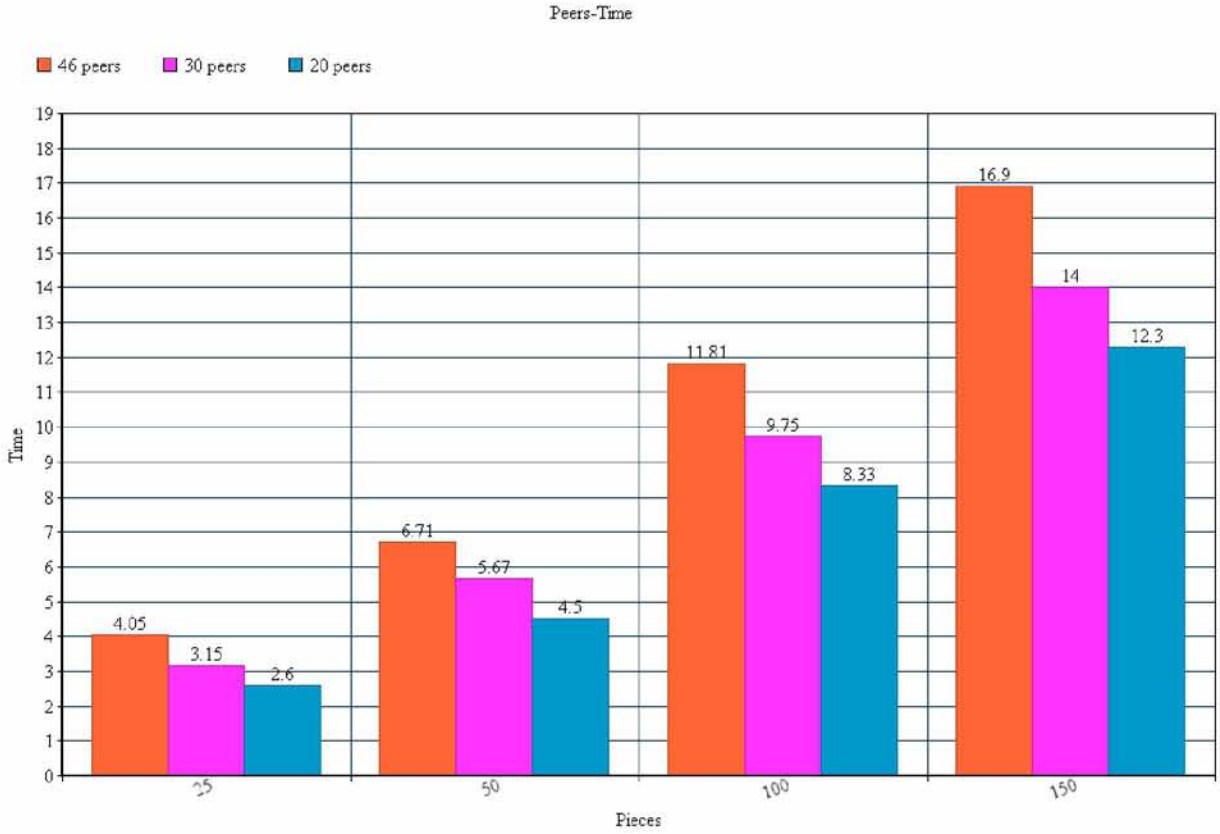
Πείραμα Τρίτο : Σχέση μεταξύ bandwidth και συνολικού χρόνου υλοποίησης



Στο συγκεκριμένο πείραμα φαίνεται η συμπεριφορά του πρωτοκόλλου για διαφορετικά bandwidth (2,4,6,8,10 Mbps) στους peers. Τα bandwidth στους routers και στον tracker μένουν σταθερά καθώς σε ένα ρεαλιστικό σενάριο αυτά είναι αρκετά μεγάλα και δεν επηρεάζουν την συμπεριφορά του πρωτοκόλλου. Αυτό που παρατηρείται είναι και πάλι η γραμμική συμπεριφορά του πρωτοκόλλου. Αξίζει να αναφερθεί πώς στη υλοποίηση που έκανα δεν γίνεται διαχωρισμός σε upload και

download bandwidth αλλά θεωρούνται ως ένα κοινό κανάλι.

Πείραμα Τέταρτο: Σχέση μεταξύ αριθμού peer και μέσου χρόνου υλοποίησης



Το συγκεκριμένο πείραμα απαιτούσε κάθε φορά τη δημιουργία μια νέας τοπολογίας για να δώσει τα απαιτούμενα αποτελέσματα. Αυτό βέβαια περιείχε τον κίνδυνο οι διαφορετικές τοπολογίες να είχαν τελείως διαφορετικά χαρακτηριστικά, τα οποία στο τέλος θα επηρέαζαν τα αποτελέσματα. Για να λύσω αυτό το πρόβλημα ακολούθησα την εξής πρακτική. Έκανα αυξομειώσεις στον αριθμό των peers που συνδέονταν σε κάθε router ώστε να διατηρώ παρόμοια κάθε φορά την τοπολογία. Δηλαδή αν ήθελα να αυξήσω τον αριθμό τον peers προσέθετα τον ίδιο αριθμό peer σε κάθε router και ομοίως αν ήθελα να μειώσω τους συνολικούς peers αφαιρούσα από κάθε router τον ίδιο αριθμό peer. Για μια ποιο ολοκληρωμένη προσέγγιση επανέλαβα το πείραμα και για διάφορα μεγέθη αρχείου(25-50-100-150 pieces).Εδώ έχω λάβει υπόψιν τον μέσο χρόνο υλοποίησης για κάθε peer.

Στο παραπάνω πείραμα φαίνεται ξεκάθαρα πώς αν και ο αριθμός τον peer

αυξάνεται σε μεγάλο βαθμό, ο χρόνος μέσος χρόνος ολοκλήρωσης για κάθε peer αυξάνεται αναλογικά πολύ λιγότερο. Αυτό συμβαίνει γιατί στο bittorrent όλοι οι peer συμμετέχουν και στο upload και στο download ταυτόχρονα. Το συμπέρασμα που μπορούμε με ασφάλεια να βγάλουμε είναι πως, το bittorent δεν έχει κανένα πρόβλημα κλιμάκωσης όσον αφορά των αριθμό των peer που συμμετέχουν στην διαδικασία. Η όποια επιπλέον καθυστέρηση προκύπτει οφείλεται κυρίως στο bottleneck που παρουσιάζεται στην αρχή.

Τοπολογίες που χρησιμοποιήθηκαν

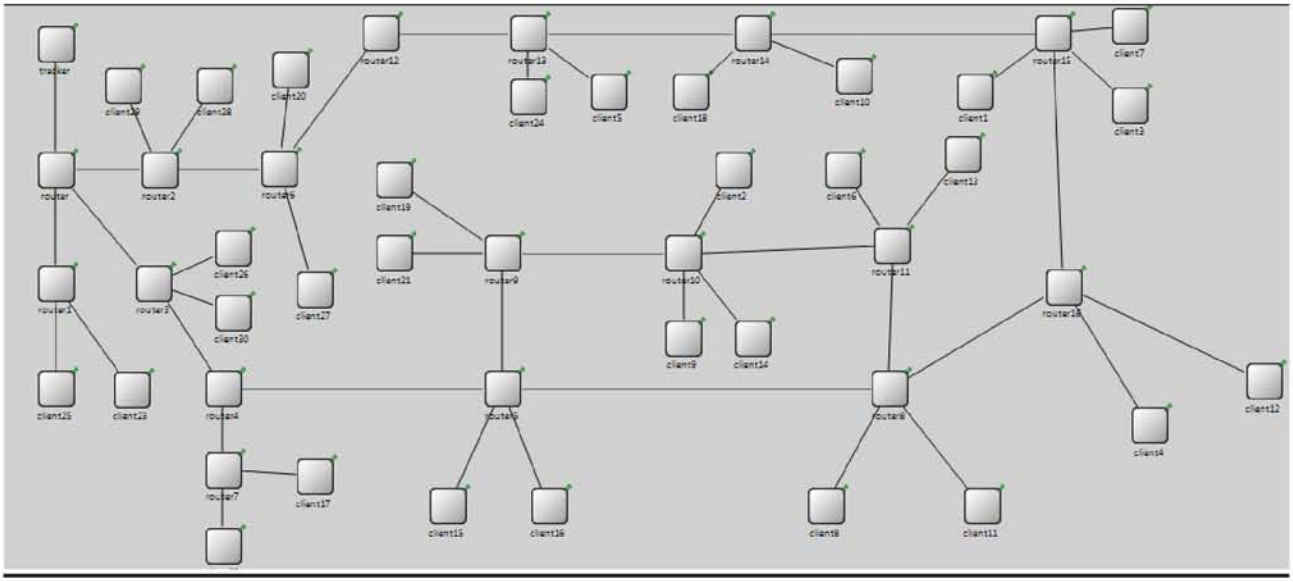
Παρακάτω παραθέτω τις τοπολογίες που χρησιμοποιήθηκαν για τα παραπάνω πειράματα.

Τοπολογία 1 : 30 peers

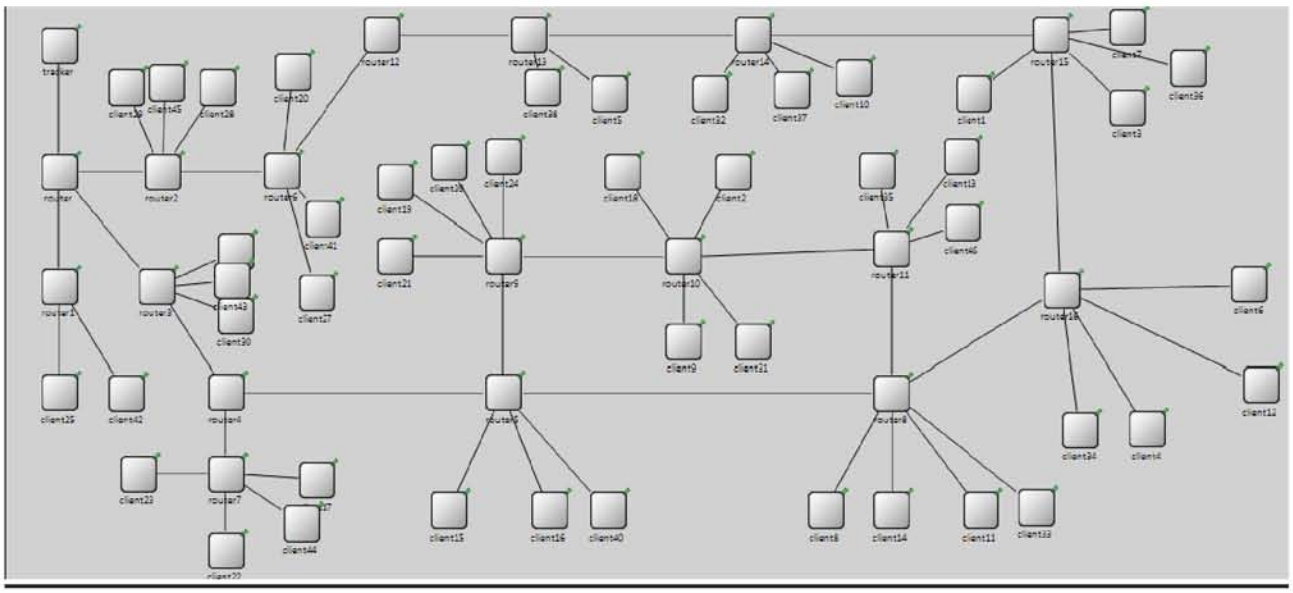
Τοπολογία 2 : 46 peers

Τοπολογία 3 : 20 peers

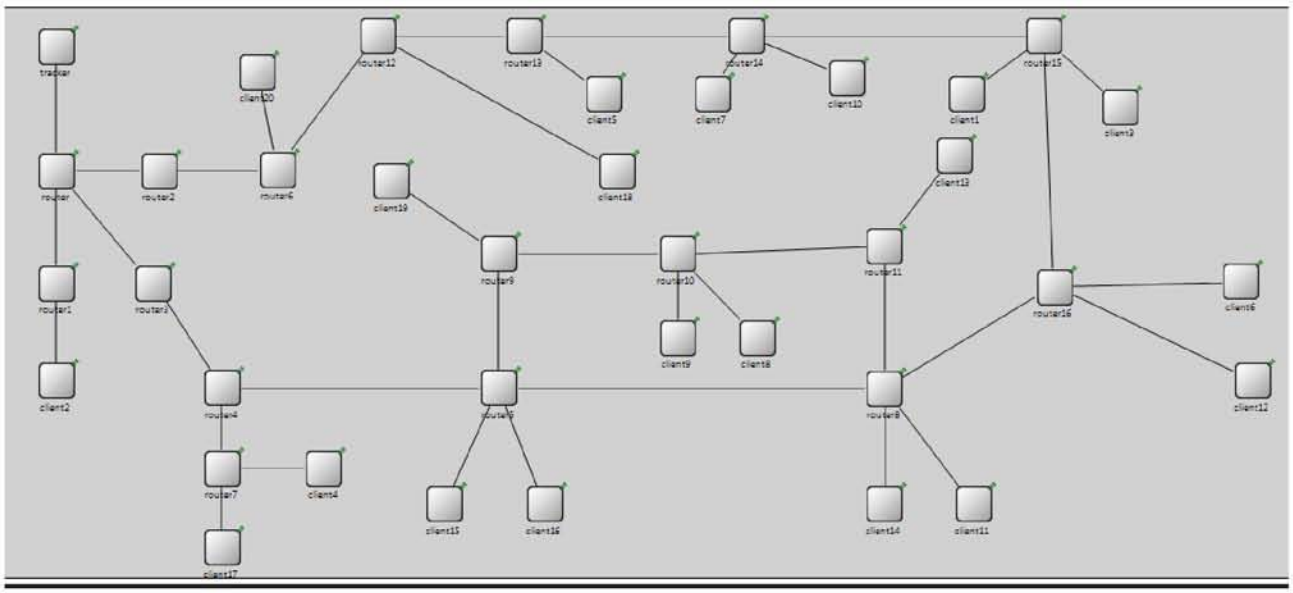
Όπως έχει προαναφερθεί στην δημιουργία των τοπολογιών δόθηκε προσοχή ώστε η αλλαγή στον αριθμό των peer να μην αλλάζει την μορφή της τοπολογίας αλλά μόνο να την κλιμακώνει.



Τοπολογία 1



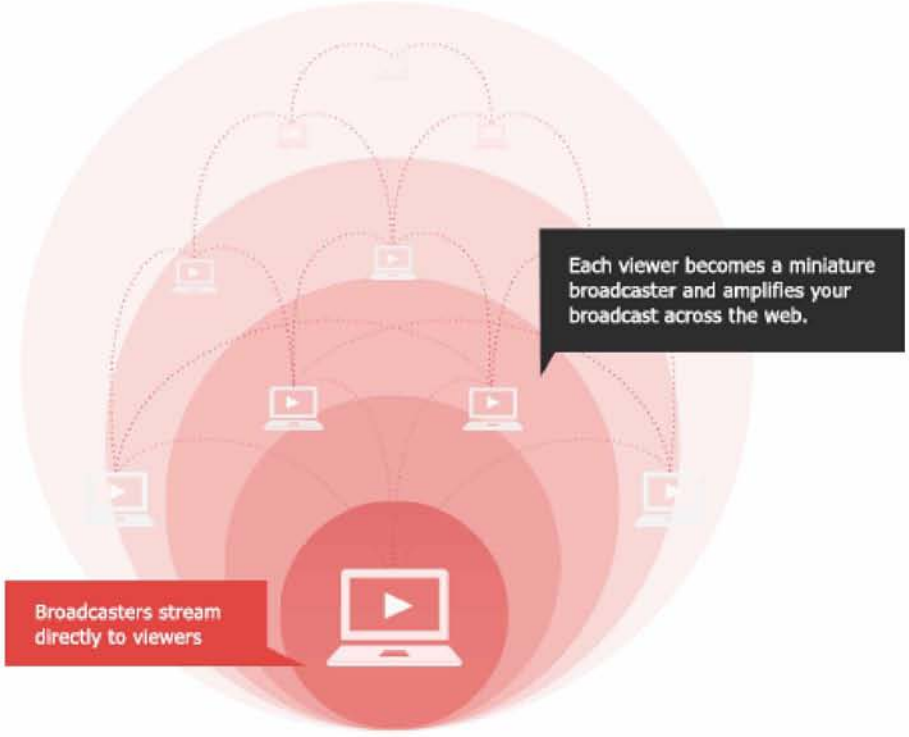
Τοπολογία 2



Τοπολογία 3

Συμπεράσματα

Μέσα από το σύνολο των παραπάνω πειραμάτων γίνεται ξεκάθαρο για ποιο λόγο το BitTorrent είναι ένα από τα ποιο διαδεδομένα πρωτόκολλα όσον αφορά την κίνηση στο διαδίκτυο. Κυρίως λόγο της καλής του κλιμάκωσης σε σχέση με μέγεθος αρχείου, αριθμό χρηστών και bandwidth αλλά και λόγος της ελευθερίας που δίνει στον τελικό χρήστη να επιλέξει ο ίδιος πώς θα επιλέξει piece και peer. Λόγω της μεγάλης του επιτυχίας στοιχεία αυτού του πρωτοκόλλου έχουν χρησιμοποιηθεί και στο έως τώρα beta πρωτόκολλο BitTorrent live το οποίο δίνει τη δυνατότητα στους χρήστες να κάνουν live streaming χωρίς να είναι αναγκαία η ύπαρξη ισχυρού δικτύου από μεριάς τους. Αυτό επιτυγχάνετε γιατί όπως και στο BitTorrent κάθε τελικός χρήστης συμμετέχει στο δίκτυο αξιοποιώντας το δικό του Upload bandwidth.Δηλαδή όπως φαίνεται στο παρακάτω σχήμα κάθε χρήστης κάνει παράλληλα και stream το περιεχόμενο που παρακολουθεί.



Επίλογος

Τέλος πρέπει να αναφερθεί πως η υλοποίηση η οποία έγινε δεν αποτελεί σε καμία περίπτωση ακριβές αντίγραφο του πρωτοκόλλου BitTorrent αλλά μια απλοποιημένη προσέγγιση σε αυτό. Αυτό έγινε κυρίως λόγο της τεράστιάς δυναμικότητας και τυχαιότητας του πρωτοκόλλου. Σε μια πραγματική κατάσταση τόσο ο αριθμός των peer όσο και η συμπεριφορά τους αλλάζουν συνεχώς κάτι τέτοιο όμως δεν μπορεί να υλοποιηθεί εύκολα σε έναν προσομοιωτή. Έτσι επιλέχθηκε μια ποιο γραμμική προσέγγιση στην συμπεριφορά των χρηστών. Δυστυχώς με αυτή την μέθοδο υλοποίησης, ήταν αδύνατο να μελετηθούν οι τεχνικές chocking/unchocking που γίνονται με σκοπό να δώσουν κίνητρα στους χρήστες ώστε να διαθέσουν το upload capacity τους. Παρόλες όμως τις αδυναμίες η συγκεκριμένη υλοποίηση μπορεί να δώσει πολύ ενδιαφέροντα συμπεράσματα γύρω από νέες τεχνικές piece και peer selection.

Πηγές-εργαλεία

1. http://www.bittorrent.org/beps/bep_0003.html
2. http://jonas.nitro.dk/bittorrent/bittorrent-rfc.html
3. Improving the BitTorrent Protocol Using Different Incentive Techniques by Rafit Izhak-Ratzin
4. http://www.omnetpp.org/doc/omnetpp/manual/usman.html#sec136
5. https://groups.google.com/forum/#!forum/omnetpp
6. http://www.nsnam.org/
7. http://en.wikipedia.org/wiki/Network_simulation
8. http://www.onlinecharttool.com/
9. http://live.bittorrent.com/