ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Αξιολόγηση απόδοσης ενός πρότυπου μηχανισμού ανίχνευσης P2P περιεχομένου σε Ασύρματα δίκτυα

Πλευράκη Μαρία
ΑΜ 2246

Εισηγητής: Ευάγγελος Πάλλης

Ηράκλειο, Μάιος 2015

Υπεύθυνη δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και κάθε βοήθεια την οποία είχα για τη προετοιμασίας της, είναι πλήρως αναγνωρισμένη και αναφέρεται στη πτυχιακή εργασία.

Έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες.

Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής του Τ.Ε.Ι. Κρήτης.

Σε όλους όσους με βοήθησαν
στη διάρκεια των σπουδών μου

Ευχαριστίες

Η πτυχιακή αυτή εργασία υλοποιήθηκε στο εργαστήριο ΠΑΣΙΦΑΗ με χρήση της υλικοτεχνικής υποδομής του. Θα ήθελα να ευχαριστήσω όλο το προσωπικό του εργαστηρίου για τη βοήθειά του.

Ιδιαίτερα θα ήθελα να ευχαριστήσω τον Ευάγγελο Πάλλη για το θέμα της πτυχιακής εργασίας, τον Ευάγγελο Μαρκάκη για την υποστήριξή του σε όλη τη διάρκεια της εκπόνησης της πτυχιακής αυτής και το Γιώργο Αλεξίου για τη πολύτιμη βοήθειά του και τις συμβουλές του.

Σύνοψη

Η ευρεία διείσδυση του Διαδικτύου το έχει καθιερώσει σε ένα από τα κύρια μέσα επικοινωνίας. Σε συνδυασμό με την εξάπλωση των κινητών συσκευών, αναγκαία κρίνεται η χρήση εφαρμογών που θα εκμεταλλεύονται στο μέγιστο βαθμό τους διαθέσιμους πόρους των συστημάτων και θα διαθέτουν μηχανισμούς για τη ταχύτερη μετάδοση των δεδομένων. Ιδανική λύση στο πρόβλημα αυτό δίνουν τα peer to peer συστήματα καθώς με φθηνό τρόπο μπορούν να εκμεταλλευτούν τους αχρησιμοποίητους πόρους του και να αυξήσουν την αποδοτικότητά του.

Σε αυτή τη πτυχιακή εργασία θα γίνει η αξιολόγηση ενός πρότυπου μηχανισμού peer to peer εφαρμοσμένος σε ασύρματα δίκτυα. Ο πρότυπος αυτός μηχανισμός δίνει τη δυνατότητα στους κόμβους που τον χρησιμοποιούν, να ανιχνεύουν την ασύρματη λήψη Bittorrent δεδομένων από γειτονικούς τους κόμβους, πέρα της χρήσης του τυπικού Bittorrent μηχανισμού. Οι συμμετέχοντες κόμβοι στο διαμοιρασμό ενός αρχείου μπορούν έτσι να ανιχνεύουν και να λαμβάνουν μέρος των Bittorrent δεδομένων που χρειάζονται απευθείας από τον "αέρα", χωρίς την αποστολή αιτημάτων για λήψη τους προς τους γειτονικούς κόμβους. Ως αποτέλεσμα προκύπτει η πολύ πιο γρήγορη διανομή των δεδομένων στους κόμβους του δικτύου.

Abstract

The pervasiveness of the Internet has established it as one of the main means of communication. In combination with the proliferation of mobile devices, it is necessary to use applications that will exploit to the maximum extent the available resources of the systems and have mechanisms for faster data transmission. The ideal solution to this problem is given by the peer to peer systems as inexpensively can exploit unused system resources and increase their scalability.

In this thesis a prototype peer to peer mechanism implemented in wireless networks will be evaluated. This prototype mechanism enables nodes that use it, to detect the wireless download of Bittorrent data from their neighbour nodes, beyond the use of the default Bittorrent mechanism. The participants in sharing a file nodes, can detect and download part of the Bittorrent data needed directly from the air, without sending requests to download content to their neighbour nodes. The result is a much faster data distribution to network nodes.

Περιεχόμενα

Πίνακας Σχημάτων

“The quitter you become, the more you are able to hear!”

0. Εισαγωγή

0.1 Στόχοι

Στόχος αυτής της πτυχιακής είναι η αξιολόγηση ενός πρότυπου μηχανισμού peer to peer σε ασύρματα δίκτυα. Ο μηχανισμός αυτός αξιοποιεί τα δεδομένα που βρίσκονται στον "αέρα" κατά τη διάρκεια λήψης δεδομένων από τους κόμβους του δικτύου. Αναμένουμε σημαντικές βελτιώσεις στο χρόνο λήψης των δεδομένων, καθώς μέρος τους ανιχνεύεται και λαμβάνεται με τη χρήση λιγότερων αιτημάτων για αποστολή δεδομένων.

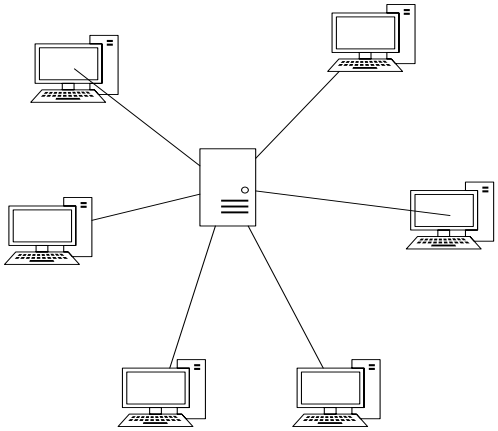
0.2 Δομή

Η πτυχιακή εργασία ξεκινάει αναλύοντας το θεωρητικό υπόβαθρο των peer to peer συστημάτων στο Κεφάλαιο 1 και των ασύρματων δικτύων στο Κεφάλαιο 2, με ιδιαίτερη έμφαση στο πρότυπο 802.11g που χρησιμοποιήθηκε στο πρακτικό μέρος. Ακολουθεί η αναλυτική παρουσίαση του BAP2P μηχανισμού και των λειτουργιών του στο Κεφάλαιο 3. Στο Κεφάλαιο 4 παρουσιάζονται τα αποτελέσματα της αξιολόγησης του μηχανισμού και ο αναλυτικός τρόπος σχεδιασμού και υλοποίησης της προσομοίωσης. Τέλος στο Κεφάλαιο 5 παραθέτονται μελλοντικές υλοποιήσεις που μπορεί να χρησιμοποιηθεί ο πρότυπος μηχανισμός ανίχνευσης peer to peer περιεχομένου.

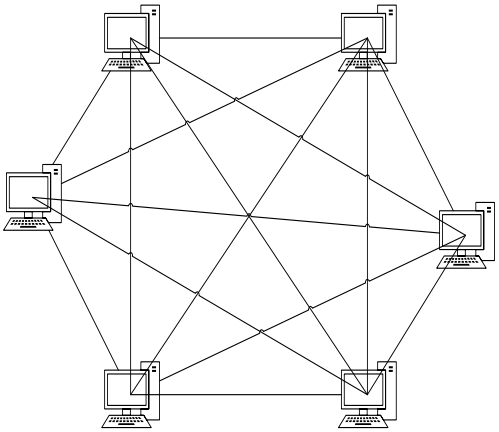
1. Peer to Peer συστήματα

1.1 Εισαγωγή

Τα peer to peer συστήματα αποτελούν ένα κατανεμημένο σύστημα επικοινωνίας. Ένα δίκτυο peer to peer αποτελείται από τουλάχιστον δύο διασυνδεδεμένους κόμβους (peers), χωρίς να είναι απαραίτητη η διαμεσολάβηση κάποιου κεντρικού server. Στόχος των peer to peer συστημάτων είναι το μοίρασμα πόρων μεταξύ των peers. Οι πόροι αυτοί μπορεί να είναι δεδομένα, υπολογιστική ισχύς ή αποθήκευση δεδομένων. Η αρχιτεκτονική επικοινωνίας peer to peer επιμερίζει το φόρτο αποστολής και λήψης στους χρήστες, κάνοντάς τους ταυτόχρονα να ζητάνε αλλά και να προσφέρουν δεδομένα.



Σχήμα 1: Client - Server Μοντέλο



Σχήμα 2: Peer to Peer Μοντέλο

1.2 Χαρακτηριστικά peer to peer συστημάτων

Στα peer to peer συστήματα οι συμμετέχοντες κόμβοι επικοινωνούν απευθείας μεταξύ τους και δρουν ταυτόχρονα ως πελάτες και διακομιστές. Οι ρόλοι κατανέμονται ανάλογα με το τί είναι αποτελεσματικό για το δίκτυο κάθε στιγμή.

Σημαντικά πλεονεκτήματα των peer to peer συστημάτων είναι:

- **Κοινή χρήση πόρων** Η peer to peer αρχιτεκτονική μπορεί να εκμεταλλευτεί πόρους που δε χρησιμοποιούνται.
- **Αυτονομία** Οι peers μπορούν να ενταχθούν ή να φύγουν από το δίκτυο όποτε θέλουν, ενώ μπορούν να διατηρήσουν τον έλεγχο των πόρων τους.
- **Αυτοοργάνωση** Δεν χρειάζεται κεντρικός έλεγχος ή διαχείριση. Μπορούν να προστεθούν νέοι peers εύκολα. Ένα peer to peer δίκτυο μπορεί να λειτουργεί ακόμα και

αν κάποιοι peers δεν λειτουργούν σωστά (ανοχή του συστήματος).

- **Αποφυγή συμφόρησης** Ο φόρτος εργασίας επιμερίζεται σε όλους τους peers. Οι εργασίες ολοκληρώνονται πιο γρήγορα καθώς οι peers ανταλλάσσουν πόρους μεταξύ τους.
- **Επεκτασιμότητα** Νέοι peers μπορούν να ενταχθούν στο δίκτυο εύκολα.
- **Μείωση κόστους** Οι εταιρίες μπορούν να εξοικονομήσουν χρήματα μέσω της βέλτιστης εκμετάλλευσης των πόρων τους.

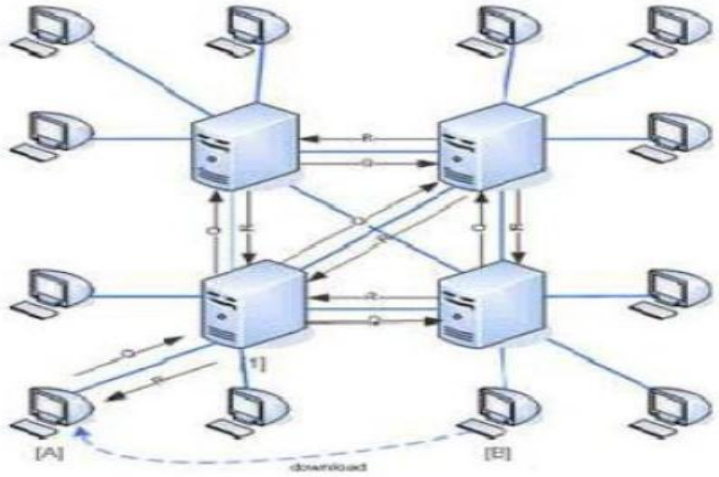
Παρόλο που τα peer to peer συστήματα παρέχουν σημαντικά πλεονεκτήματα, μπορεί να μην αποτελούν τη κατάλληλη υλοποίηση για μερικές εργασίες. Οι αιτίες είναι οι εξής:

- Οι peers είναι πιο ευάλωτοι σε επιθέσεις.
- Είναι δύσκολη η επιβολή standards στο σύστημα.
- Μερικές εργασίες δε μπορούν να μοιραστούν μεταξύ των peers.
- Ένα peer to peer σύστημα δε μπορεί να εγγυηθεί ότι οι πόροι μπορεί να είναι διαθέσιμοι συνεχώς. Κάθε peer μπορεί να φύγει οποιαδήποτε στιγμή από το δίκτυο.
- Είναι δύσκολο να απαλειφθεί ο διαμοιρασμός παράνομου περιεχομένου.
- Δημιουργείται τεράστια κίνηση στο δίκτυο.

1.3 Αρχιτεκτονική peer to peer δικτύων

1.3.1 Κεντρικοποιημένο Σύστημα

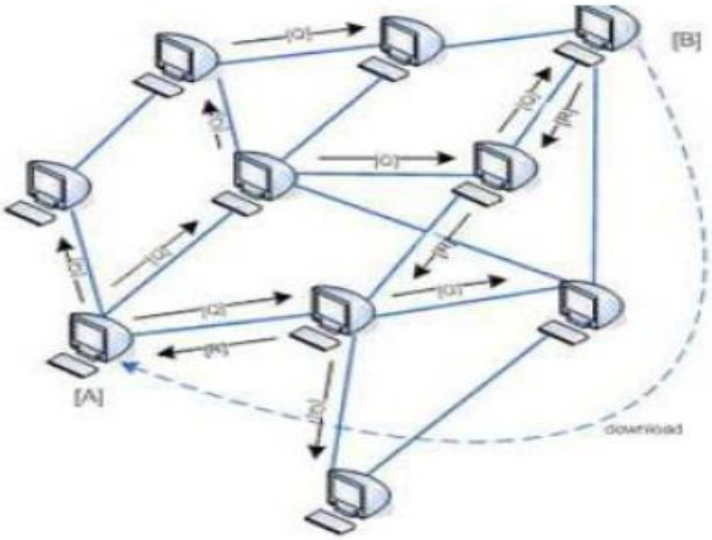
Στο κεντρικοποιημένο peer to peer σύστημα υπάρχει ένας κεντρικός υπολογιστής όπου αποθηκεύονται τα αρχεία που είναι διαθέσιμα προς διαμοιρασμό όπως και οι peers που είναι συνδεδεμένοι σε αυτό. Ο κάθε peer για να συνδεθεί ή να αποσυνδεθεί από το δίκτυο πρέπει να επικοινωνήσει με το κεντρικό υπολογιστή, το ίδιο και για να αναζητήσει κάποιο αρχείο. Ο διαμοιρασμός των αρχείων γίνεται μόνο ανάμεσα στους peers.



Σχήμα 3: Κεντρικοποιημένο Μοντέλο

1.3.2 Αποκεντρικοποιημένο Σύστημα

Στο αποκεντρικοποιημένο σύστημα δεν υπάρχει κάποιος κεντρικός διακομιστής που να διαχειρίζεται τους peers. Οι ίδιοι αυτοοργανώνονται με βάση τοπικές πληροφορίες.



Σχήμα 4: Αποκεντρικοποιημένο Μοντέλο

Τα αποκεντρικοποιημένα συστήματα διακρίνονται σε:

1.3.2.1 Δομημένα

Η σύνδεση των peers διέπεται από συγκεκριμένα κριτήρια και αλγόριθμους που οδηγούν σε συγκεκριμένες τοπολογίες. Οι peers οργανώνονται σε γράφους και ένα κλειδί αντιστοιχίζεται σε κάθε συμμετέχοντα. Η τοποθέτησή τους γίνεται με προκαθορισμένο

τρόπο ώστε να διευκολύνεται η αναζήτηση και να επιτυγχάνεται η κλιμάκωση. Είναι κατάλληλα για εφαρμογές μεγάλης κλίμακας λόγω της υψηλής επεκτασιμότητάς τους και της δεδομένης απόδοσής τους.

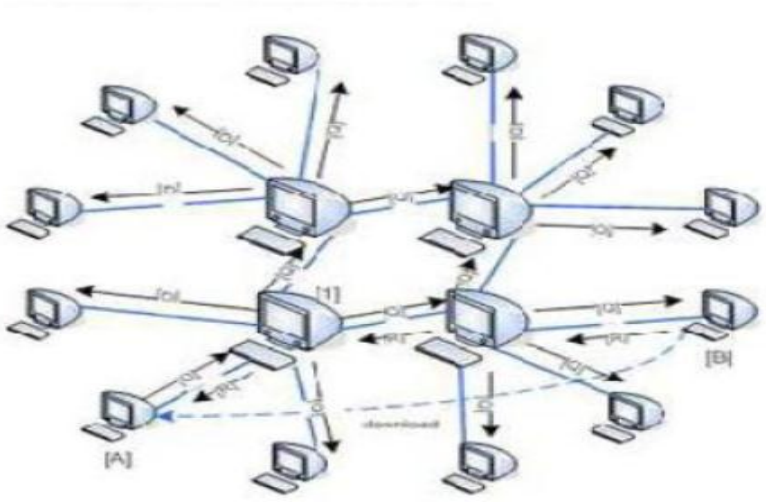
1.3.2.2 Αδόμητα

Δεν υπάρχει συγκεκριμένη δομή ανάμεσα στους peers. Οι peers τοποθετούνται στο δίκτυο χωρίς γνώση της τοπολογίας του. Τα αδόμητα peer to peer συστήματα είναι κατάλληλα σε περιπτώσεις όπου μεγάλο πλήθος κόμβων συμμετέχουν περιοδικά στο δίκτυο, χωρίς αποδοτικούς μηχανισμούς κλιμάκωσης και διαθεσιμότητας.

1.3.3 Υβριδικό Σύστημα

Στο υβριδικό σύστημα ενώνονται οι λειτουργίες του κεντρικοποημένου και του αποκεντρικοποιημένου συστήματος. Ένα σύνηθες σύστημα είναι να υπάρχει ένας / πολλοί κεντρικοί servers - routers ο οποίος θα βοηθάει τους peers να βρουν ο ένας τον άλλον. Παρέχουν ένα κατάλογο διευθύνσεων οι οποίες αναφέρονται από ένα σύνολο δεικτών.

Αυτό το τερματικό δρομολογητής αν είναι δυναμικό μπορεί να χαρακτηριστεί ως peer group manager αλλιώς λειτουργεί ως βάση δεδομένων του δρομολογητή και όλη η διαχείριση γίνεται από τους peers. Πάντως ακόμα και αν οι δρομολογητές ενημερώνουν δυναμικά τους καταλόγους τους εξαρτάται από τον peer αν θα τους χρησιμοποιήσει. Σε αντίθετη περίπτωση γίνεται υποβάθμιση της peer to peer αρχιτεκτονικής καθώς υποβαθμίζεται η αυτοδιαχείριση των peers.



Σχήμα 5: Υβριδικό Μοντέλο

1.4 Bittorrent πρωτόκολλο

Το Bittorrent είναι ένα πρωτόκολλο υλοποίησης peer to peer διαμοιρασμού αρχείων.

Το αρχείο προς διανομή διαιρείται σε μικρότερα κομμάτια, τα pieces. Τα pieces διαιρούνται σε μικρότερα κομμάτια τα blocks. Αν ένα αρχείο ζητείται από πολλούς peers ταυτόχρονα, κάθε peer ζητάει ένα διαφορετικό piece. Κάθε piece που λαμβάνει ο peer, μπορεί έπειτα να το διαθέσει σε άλλους peers (leechers) που το ζητάνε. Όταν ένας peer κατεβάσει ολόκληρο το αρχείο γίνεται seeder.

Το Bittorrent πρωτόκολλο είναι ιδανικό για διαμοιρασμό μεγάλων αρχείων όπου η διαδικασία λήψης διαρκεί πολύ. Η απόδοση ενός συστήματος peer to peer που χρησιμοποιεί το Bittorrent πρωτόκολλο βελτιώνεται όσο μεγαλώνει ο αριθμός των συμμετεχόντων peers καθώς δυσκολεύει η διακοπή του διαμοιρασμού του αρχείου.

Για να ξεκινήσει ο διαμοιρασμός ενός αρχείου, θα πρέπει να δημιουργηθεί ένα .torrent αρχείο το οποίο θα περιέχει metadata για το αρχείο που θα διαμοιραστεί. Είναι ένα binary bencoded αρχείο.

```
{
   'announce': 'http://bttracker.debian.org:6969/announce',
   'info':
   {
      'name': 'debian-503-amd64-CD-1.iso',
      'piece length': 262144,
      'length': 678428672,
      'pieces':
'841ae846bc5b6d7bd6e9aa3dd9e551559c82abc1...d14f1631d776008f83772e
e170c42411618190a4'
   }
}
```

Σχήμα 6: Μορφή .torrent αρχείου

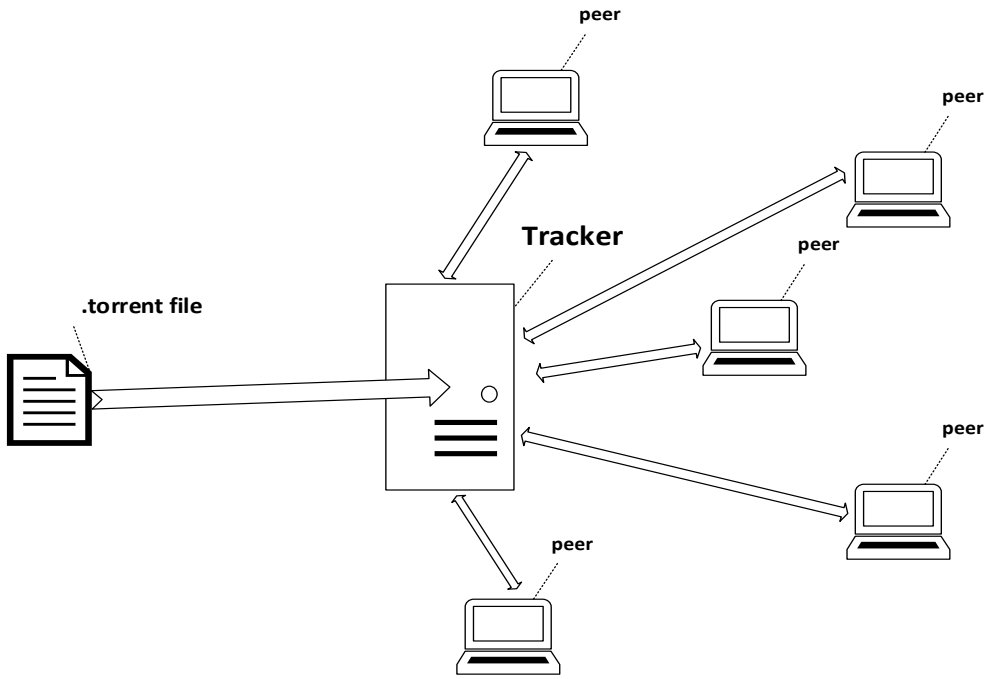
Τα πεδία του .torrent αρχείου είναι:

- **Info**: Dictionary που περιέχει
- **Name**: Το όνομα του αρχείου
- **Announce**: Το Url του tracker
- **Piece Length**: Αριθμός bytes / piece
- **Pieces**: Λίστα hash
- **Length**: Μέγεθος αρχείου σε bytes
- **Files**: Περιεχόμενα αρχεία
- **Path:** Λίστα strings των ονομάτων των υποφακέλων
- **Length:** Μέγεθος του φακέλου σε bytes

Τα .torrent αρχεία φιλοξενούνται σε ένα κεντρικό κόμβο (tracker). Οι peers που θέλουν

να κατεβάσουν το αρχείο, συνδέονται με τον tracker, ανοίγουν το .torrent αρχείο που σχετίζεται με το αρχείο που θέλει να αποκτήσουν με ένα Bittorrent client. Ο tracker δίνει τη λίστα των peers με τους seeders και τους leechers του αρχείου εκείνη τη στιγμή. Το σύνολο των peers που εμπλέκονται στο διαμοιρασμό ενός συγκεκριμένου αρχείου λέγεται swarm. Ανά συγκεκριμένα χρονικά διαστήματα οι peers στο swarm συνδέονται στον tracker για να ενημερώνονται για τη λίστα των peers.

Για να ξεκινήσει η λήψη ενός αρχείου από έναν peer, πρέπει να αποφασιστεί πώς θα ζητήσει τα pieces του αρχείου από τους υπόλοιπους peers στο swarm. Το Bittorrent πρωτόκολλο χρησιμοποιεί μια αρχή που ονομάζεται το *σπανιότερο κομμάτι πρώτα* (rarest piece first). Θα πρέπει να καθοριστεί από τα pieces που δεν έχει ακόμα ο leecher ποιά είναι σπανιότερα στους γειτονικούς peers και να ζητήσει τη λήψη τους πρώτα από τα υπόλοιπα pieces που δεν έχει. Με αυτό το τρόπο τα σπανιότερα pieces διανέμονται με ταχύτατα ώστε να εξισορροπείται η διαθεσιμότητα των pieces ενός αρχείου σε όλους τους peers που το έχουν.



Σχήμα 7: Αρχιτεκτονική Bittorrent πρωτοκόλλου

1.4.1 Bittorrent Handshake

Για να εδραιωθεί μια σύνδεση ανάμεσα στους peers απαραίτητη είναι η ανταλλαγή ενός handshake μηνύματος μεταξύ τους. Το μήνυμα αποτελείται από:

- **protocol name**: Μια συμβολοσειρά για την αναγνώριση του πρωτοκόλλου που χρησιμοποιείται
- **protocol name length**: 1 byte που προσδιορίζει το μέγεθος της συμβολοσειράς του

πεδίου protocol name σε bytes.

- **hash info**: Ένα 20 bytes SHA-1 hash του .torrent αρχείου
- **peer id**: Μια συμβολοσειρά 20 bytes που χρησιμοποιείται ως μοναδικό ID για τον bittorrent client
- **reserved bytes**: 8 bytes που χρησιμοποιούνται για επιπλέον λειτουργίες του πρωτοκόλλου

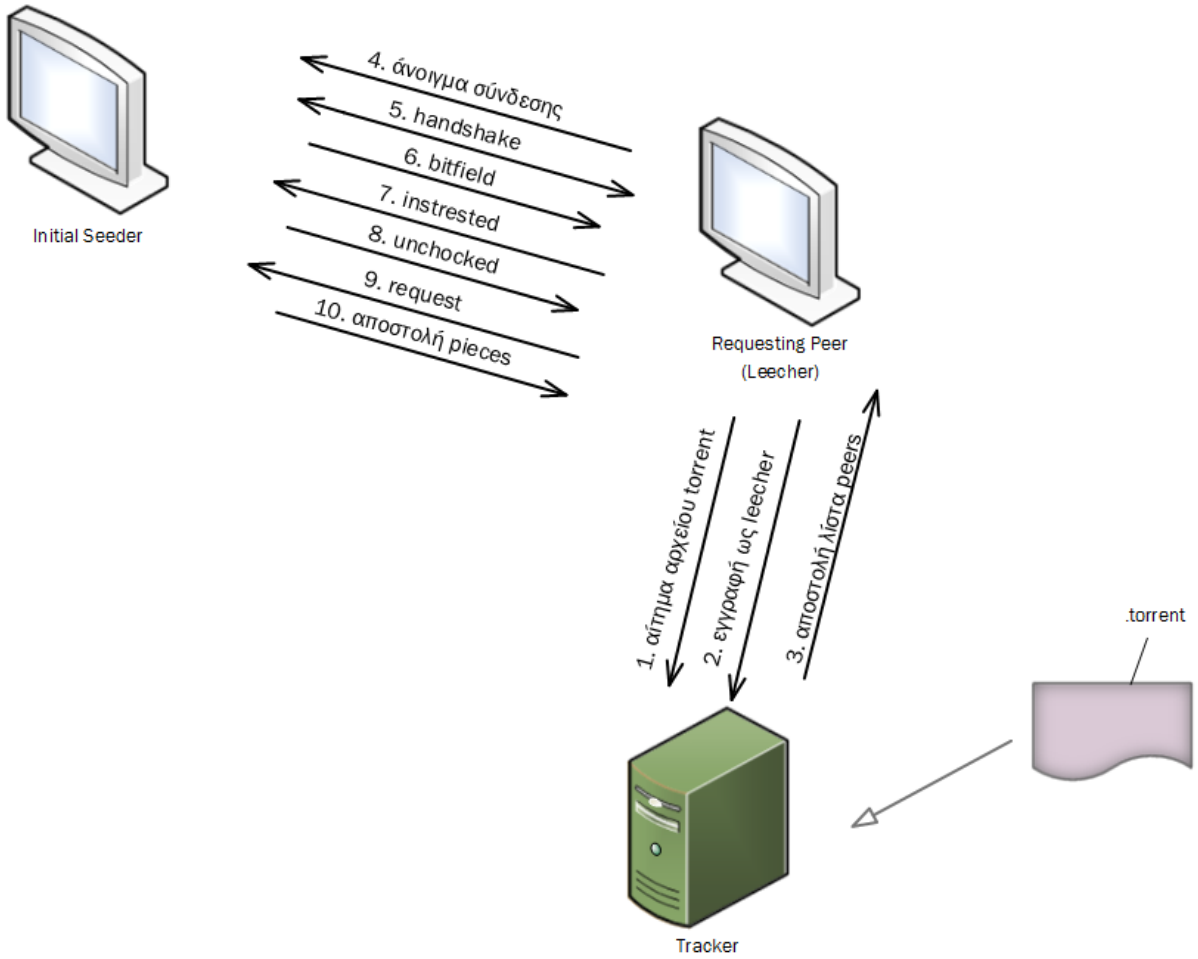
1.4.2 Bittorrent Μηνύματα

Οι peers μπορούν να ανταλλάσσουν μηνύματα μεταξύ τους. Τα μηνύματα αυτά αρχίζουν με ένα πεδίο ακεραίου *bt_msg_t* μεγέθους 4 bytes όπου προσδιορίζεται το μέγεθος του υπόλοιπου μέρους του μηνύματος που ακολουθεί. Στη συνέχεια το πεδίο *bt_type* μεγέθους 1 byte που περιγράφει το τύπο του μηνύματος. Αυτά τα Bittorrent μηνύματα είναι:

- **Choke** Ένας peer στέλνει choke μήνυμα όταν δεν μπορεί να εξυπηρετήσει αιτήματα που δέχεται από άλλους peers για λήψη ή αποστολή blocks.
- **Unchoke** Όταν ο peer μπορεί να εξυπηρετεί αιτήματα αποστολής ή λήψης blocks από άλλους peers αποστέλλει μήνυμα unchoke.
- **Interested** Αποστέλλεται όταν ο peer ζητάει ένα αρχείο .torrent και pieces από αυτό
- **Not interested** Όταν δεν ενδιαφέρεται για ένα αρχείο .torrent ή για pieces του αρχείου αυτού, ο peer στέλνει μήνυμα not interested
- **Have** Πληροφορεί άλλους peers ότι έχει ένα piece ενός .torrent αρχείου.
- **Bitfield** Τα μηνύματα bitfield αποστέλλονται μετά το handshake πληροφορώντας ποια pieces έχει διαθέσιμα ο peer
- **Request** Ένας peer αποστέλλει μήνυμα request όταν ζητά ένα συγκεκριμένο piece ενός αρχείου .torrent. Περιέχει τα πεδία:
    - **Index** Ο αριθμός του piece που ζητήθηκε
    - **Begin** Ο αριθμός του block από το οποίο θα ξεκινήσει
    - **Length** Είναι ο αριθμός των δεδομένων που ζητήθηκαν σε bytes
- **Piece** Χρησιμοποιείται για την αποστολή των blocks. Περιέχει τα πεδία:
    - **Index** Ο αριθμός του piece
    - **Begin** Ο αριθμός του block
    - **Block** Ακολουθία bytes που περιέχουν τα δεδομένα του block
- **Cancel** Αποστέλλεται από έναν peer για την ακύρωση ένος request μηνύματος.
- **Keep alive** Αποτελείται από 4 μηδενικά bytes και αποστέλλεται σε τακτά χρονικά διαστήματα για να παραμένει ενεργεί μια σύνδεση. Όταν σταματήσει να αποστέλλεται αυτό το μήνυμα η σύνδεση θεωρείται ανενεργή.

Η διαδικασία διαμοιρασμού ενός αρχείου μέσω του Bittorrent πρωτοκόλλου ξεκινάει διαλέγοντας 4 τυχαία pieces που θα κατεβάσει πρώτα. Έπειτα ο peer επιλέγει να κατεβάσει πρώτα τα σπάνια κομμάτια του αρχείου, αυτά που έχουν οι λιγότεροι peers, από τους υπόλοιπους peers στο swarm. Αυτό συμβαίνει για να διατηρηθεί η ομοιόμορφη κατανομή των pieces.

Οι συνδέσεις ανάμεσα στους peers είναι συμμετρικές. Μηνύματα και δεδομένα στέλνονται και προς τις 2 κατευθύνσεις. Οι συνδέσεις περιέχουν 2 bits κατάστασης σε κάθε άκρο chocke ή unchoke intrested ή not intrested. Οι συνδέσεις ξεκινάνε πάντα με ένα handshake μήνυμα και από τις δύο πλευρές και έπειτα ο peer που ξεκινάει ως leecher δηλώνει κατάσταση chocked και not intrested. Η μεταφορά δεδομένων ξεκινάει όταν η μια πλευρά είναι σε κατάσταση intrested και η άλλη είναι unchoked. Σε κάθε μεταφορά αρχείου οι λήπτες πρέπει να στέλνουν μερικά συνεχόμενα αιτήματα Piece (pipelining) ώστε να επιτυγχάνεται η καλή απόδοση της TCP μετάδοσης. Τα αιτήματα Piece από τη μεριά του αποστολέα κρατούνται στη μνήμη και όχι σε network number ώστε να απορρίπτονται σε περίπτωση choke κατάστασης.



Σχήμα 8: Διαδικασία Λήψης Bittorrent πρωτοκόλλου

2. Ασύρματα δίκτυα

2.1 Εισαγωγή

Τα ασύρματα δίκτυα επιτρέπουν σε συσκευές να συνδέονται σε δίκτυα ή μεταξύ τους χωρίς την ανάγκη καλωδίωσης. Χρησιμοποιούνται πομποδέκτες που βρίσκονται στις συσκευές αλλά και ένα κοινό σημείο ασύρματης πρόσβασης το Access Point (AP).

Ο όρος WiFi (Wireless Fidelity) χρησιμοποιείται για να προσδιορίσει τα ασύρματα τοπικά δίκτυα με στόχο να προσφέρουν ασύρματη ευρυζωνική σύνδεση σε κάλυψη μικρών εκτάσεων όπως κτήρια, πλατείες κτλ. Βασίζεται στα πρότυπα των IEEE 802.11 με τη χρήση half duplex συνδέσεων, δηλαδή τη χρήση μια από τις δύο διαδρομές κατά τη διάρκεια της σύνδεσης δύο συσκευών. Οι ταχύτητες των WiFi δικτύων κυμαίνονται από 1 εώς 100 Mbps.
Η ευελιξία που παρέχουν τα ασύρματα δίκτυα έχουν δώσει μεγάλη ώθηση στην εξάπλωσή τους, σε συνδυασμό με την εξάπλωση των κινητών συσκευών όπως τα laptops, tablets και smartphones.

Τα πλεονεκτήματα των ασύρματων δικτύων έναντι των ενσύρματων δικτύων συνοψίζονται στα εξής:

- **Κινητικότητα** αφού εξυπηρετούν τη σύνδεση συσκευών χωρίς την ύπαρξη καλωδίων και λύνουν το πρόβλημα σύνδεσης σε δημόσιους ανοικτούς χώρους, κτήρια, συνεδριακά κέντρα.
- **Ταχύτητα υλοποίησης** καθώς δεν χρησιμοποιείται καλωδιακή σύνδεση
- **Χαμηλό κόστος** εγκατάστασης και λειτουργίας.

Τα μειονεκτήματα που καταγράφονται είναι:

- Οι **δυσκολίες κάλυψης** καθώς η ηλεκτρομαγνητική ακτινοβολία εξασθενεί όταν περνάει από υλικά εντός κτηρίων. Στους ανοικτούς χώρους το σήμα διασκορπίζεται, χάνει την ισχύ του και παρουσιάζεται απώλεια διαδρομής καθώς αυξάνεται η απόσταση πομπού – δέκτη.
- Ο **περιορισμός στη χωρητικότητα** καθώς υπάρχουν παρεμβολές ή περιορισμοί στο ασύρματο εύρος ζώνης συχνοτήτων.
- Η **ασφάλεια επικοινωνίας** καθώς απαιτείται προσπάθεια για τη κρυπτογράφηση των δεδομένων που μεταδίδονται ασύρματα καθώς η επικοινωνία είναι εκτεθειμένη στον αέρα και μπορεί να παρακολουθείται από μη εξουσιοδοτημένους χρήστες.
- **Αξιοπιστία** καθώς μπορεί να υπάρχουν αθέμιτες παρεμβολές. Παρατηρείται το φαινόμενο πολλαπλής διάδοσης διαδρομών όταν τα ηλεκτρομαγνητικά κύματα

ανακλώνται σε αντικείμενα και επιφάνειες, φτάνοντας στο δέκτη σε διαφορετικό χρόνο.

2.2 Πρότυπα 802.11

Τα 802.11 πρότυπα αναπτύχθηκαν από τον οργανισμό IEEE. Ξεκίνησαν να σχεδιάζονται το 1990 και χρησιμοποιούν τη συχνότητα των 2.4GHz για την επικοινωνία. Τα πρότυπα περιγράφουν την επικοινωνία του κόμβου με το Access Point και την κατ' επιλογή χρήση κρυπτογράφησης. Καθορίζουν επίσης το φυσικό επίπεδο (PHY) και το επίπεδο ζεύξης (MAC) στη στοίβα του μοντέλου OSI.

IEEE πρότυπα	**Σημειώσεις**
802.11	Πρώτο πρότυπο 1997). Specified the MAC and the original slower frequency - hopping and direct - sequence modulation techniques.
802.11a	Δεύτερο πρότυπο φυσικού επιπέδου (1999)
802.11b	Τρίτο πρότυπο φυσικού επιπέδου (1999)
802.11g	54Mbps, 2.4GHz πρότυπο
802.11n	Υψηλή απόδοση Throughput MIMO (multiple input, multiple output antennas) (Sep 2009)
802.11ac	Πολύ υψηλός Throughput <6GHz; πιθανές βελτιώσεις του 802.11n: καλύτερο σχήμα διαμόρφωσης (αναμένεται αύξηση ρυθμού μετάδοσης ~10%); μεγαλύτερα κανάλια (80 ή ακόμα και 160 MHz), multi user MIMO; (2012)
802.11ad	Πολύ υψηλό Throughput 60 GHz (2013)

Σχήμα 9: Εξέλιξη 802.11 προτύπων

2.2.1 Χαρακτηριστικά

Τα πρότυπα 802.11 προσφέρουν τις παρακάτω υπηρεσίες:

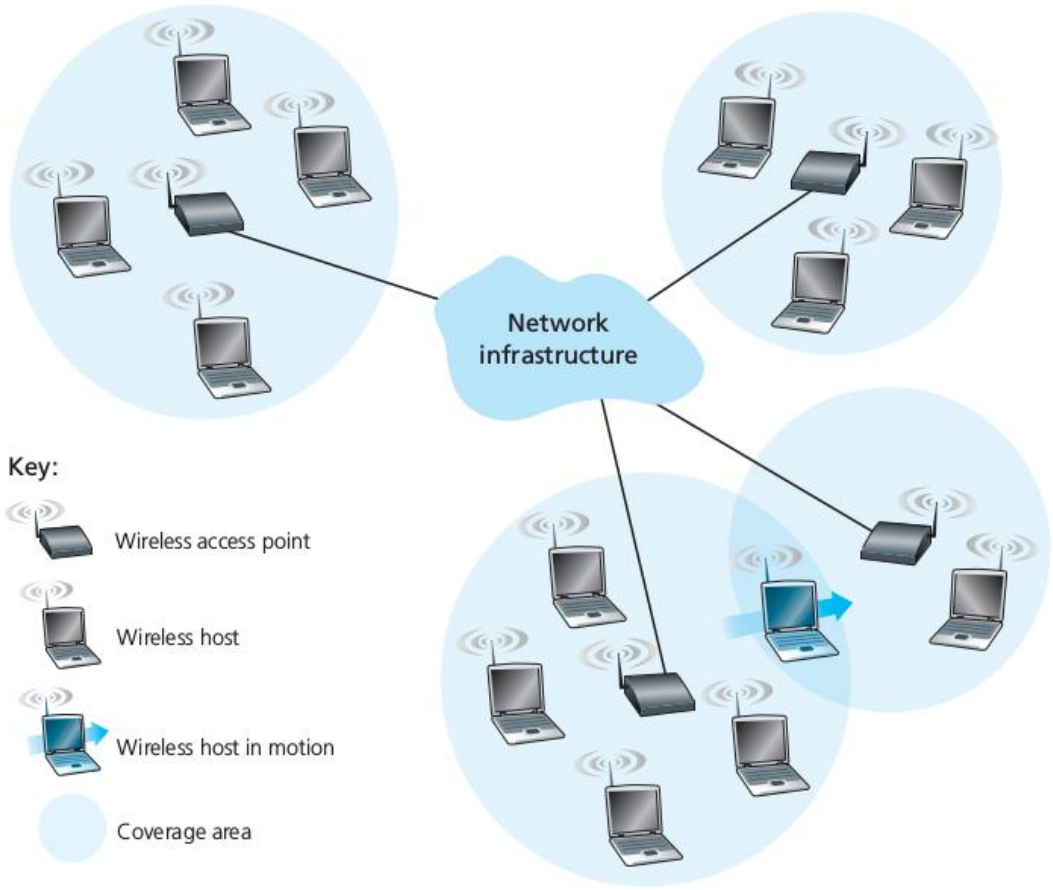
- **Διανομή:** Η υπηρεσία χρησιμοποιείται από κινητούς σταθμούς ως δίκτυο υποδομής για να στείλουν τα δεδομένα τους. Όταν ένα frame γίνεται αποδεκτό από ένα access point χρησιμοποιεί την υπηρεσία διανομής για να παράδοση το frame στο προορισμό του.
- **Διαδραστικότητα:** Είναι μια υπηρεσία που προσφέρεται από το σύστημα διανομής, επιτρέποντας τη σύνδεση μιας μη 802.11 συσκευής στο δίκτυο.
- **Συσχέτιση** Η παράδοση των frames στους κινητούς σταθμούς είναι δυνατή

καθώς οι κινητοί σταθμοί σχετίζονται με τα access points. Έπειτα το σύστημα διανομής μπορεί να χρησιμοποιήσει τις πληροφορίες αυτές για να βρει με ποιο access point σχετίζεται ένας κινητός κόμβος.

- **Επανασυσχέτιση:** Όταν μια κινητή συσκευή μετακινείται στο χώρο μέσα στον οποίο υπάρχει η ασύρματη υπηρεσία, πρέπει να εκτιμηθεί η ισχύ του σήματος και να η πιθανή αλλαγή ανάμεσα στα access points της κινητής συσκευής. Οι επανασυσχετίσεις γίνονται από τους κινητούς σταθμούς και όχι από access points.
- **Αποσυσχέτιση:** Όταν χρησιμοποιείται αυτή η υπηρεσία μια αποθηκευμένη κινητή συσκευή αφαιρείται από το σύστημα διανομής.
- **Αυθεντικοποίηση:** Τα ασύρματα δίκτυα δεν μπορούν να προσφέρουν φυσική ασφάλεια στις συσκευές. Έτσι δημιουργούνται επιπλέον ρουτίνες για την αυθεντικοποίηση των συσκευών. Η αυθεντικοποίηση είναι απαραίτητη για τη συσχέτιση καθώς μόνο οι αυθεντικοποιημένοι χρήστες μπορούν να χρησιμοποιήσουν το ασύρματο δίκτυο.
- **Αποαθεντικοποίηση:** Η υπηρεσία αυτή τερματίζει την αυθεντικοποίηση μιας κινητής συσκευής από το δίκτυο.
- **Ιδιωτικότητα:** Στα ασύρματα δίκτυα προσφέρεται μια προαιρετική υπηρεσία για τη διατήρηση της ιδιωτικότητα κάθε συσκευής. Η υπηρεσία WEP (Wired Equivalent Privacy)
- **MSDU:** Οι κινητοί σταθμοί παρέχουν την υπηρεσία MSDU (mac service data unit) που είναι υπεύθυνη για τη μεταφορά των δεδομένων στο προορισμό τους.

Τα τμήματα που αποτελούν ένα ασύρματο δίκτυο είναι:

- Οι **σταθμοί**, δηλαδή οι συσκευές που διαθέτουν ασύρματο πομποδέκτη και είναι συμβατές με το πρότυπο 802.11. Οι σταθμοί έχουν ισότιμη πρόσβαση στο μέσο διαμορφώνοντας μια ομότιμη σύνδεση.
- Το **σημείο πρόσβασης** (Access Point) είναι ένας κεντρικός κόμβος που αναλαμβάνει τον έλεγχο πρόσβασης στο κοινό μέσο. Όλοι οι σταθμοί του δικτύου συνδέονται ασύρματα σε αυτή τη συσκευή.
- Οι **δικτυακές ομάδες** όπου εντάσσονται οι συσκευές προκειμένου να υπάρξει επικοινωνία μεταξύ τους.



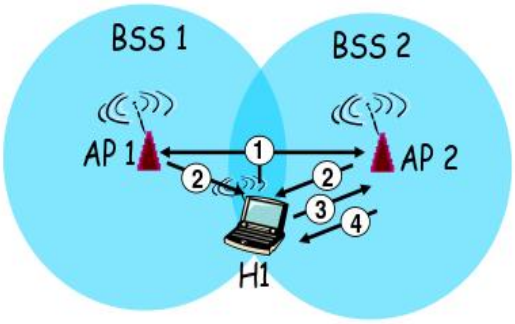
Σχήμα 10: Στοιχεία ασύρματου δικτύου

Το θεμελιώδες δομικό στοιχείο της αρχιτεκτονικής του 802.11 είναι το βασικό σύνολο υπηρεσιών (BSS). Το BSS περιέχει ένα ή περισσότερους ασύρματους κόμβους και ένα κεντρικό σταθμό βάσης (AP). Κάθε ασύρματος σταθμός περιέχει μια MAC διεύθυνση μεγέθους 6 bytes στο firmware της κάρτας δικτύου του. Με την εγκατάσταση ενός AP εκχωρείται ένα αναγνωριστικό υπηρεσιών SSID (Service Set Identifier).

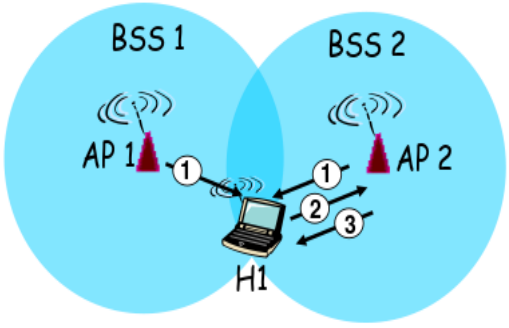
Στη προσπάθεια σύνδεσης ενός κόμβου σε ένα BSS περιλαμβάνει δύο τεχνικές την active scanning και passive scanning. Με το active scanning ο κόμβος στέλνει ένα μήνυμα το probe request στο οποίο τα υπάρχοντα AP οφείλουν να απαντήσουν με ένα probe response μήνυμα δηλώντας τη παρουσία τους και τα SSID τους. Κατά τη διάρκεια του probe request αποστέλλεται σε όλα τα κανάλια για εύρεση AP.

Με τη μέθοδο passive scanning το κάθε AP στέλνει περιοδικά beacon frames το οποίο το καθένα περιέχει το SSID και το MAC του AP. Ο ασύρματος κόμβος σαρώνει τα κανάλια επικοινωνίας ψάχνοντας τα beacon frames για να αναγνωριστούν τα κοντινά AP. Έπειτα ο ασύρματος κόμβος επιλέγει σε ποιο AP να συνδεθεί στέλνοντας ένα πλαίσιο αίτησης

συσχέτισης σε αυτό.



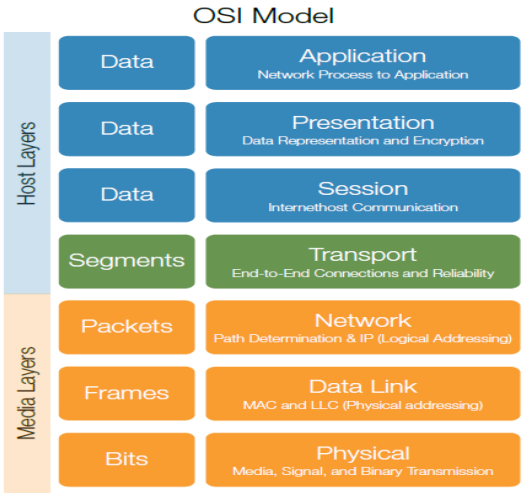
Σχήμα 11: Active scanning ασύρματου κόμβου



Σχήμα 12: Passive scanning ασύρματου κόμβου

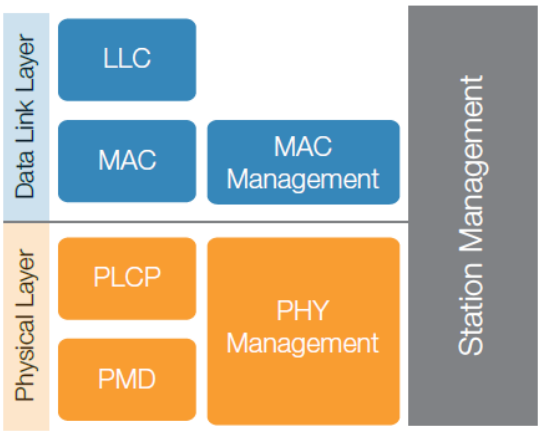
2.2.2 Αρχιτεκτονική 802.11

Για να περιγραφεί πώς η πληροφορία μετακινείται δικτυακά από ένα κόμβο σε ένα άλλο χρησιμοποιείται το μοντέλο OSI, που ορίζει 7 ιεραρχικά επίπεδα. Σε κάθε επίπεδο ορίζεται ο σκοπός του και τα πρωτόκολλα που χρησιμοποιούνται σε αυτό.



Σχήμα 12: Μοντέλο OSI

Το πρότυπο 802.11 ορίζει τις προδιαγραφές για τα δυο κατώτερα επίπεδα του μοντέλου OSI το Φυσικό επίπεδο (PHY) και το επίπεδο Ζεύξης (MAC).



Σχήμα 13: PHY και MAC επίπεδα στο 802.11 πρότυπο

2.2.2.1 Φυσικό Επίπεδο PHY

Το φυσικό επίπεδο ορίζει τη σχέση μεταξύ της συσκευής και του μέσου μετάδοσης. Οι κύριες λειτουργίες του είναι:

- Καθιέρωση και έναρξη της σύνδεσης στο μέσο επικοινωνίας
- Συμμετοχή σε διαδικασίες που διαμοιράζουν τους πόρους της σύνδεσης σε πολλούς χρήστες
- Κατάλληλη διαμόρφωση των δεδομένων και των σημάτων απόκρισης που μεταφέρονται στο κανάλι επικοινωνίας

Επίπεδο PLCP: Είναι το επίπεδο προσαρμογής όπου προσαρμόζονται πακέτα για διαφορετικές τεχνολογίες φυσικών επιπέδων.

Επίπεδο PMD: Καθορίζει τις τεχνικές κωδικοποίησης και διαμόρφωσης

Επίπεδο Διαχείρισης PHY: Αναλαμβάνει το συντονισμό του καναλιού.

Το **επίπεδο Διαχείρισης Σταθμού** αναλαμβάνει το συντονισμό του PHY και MAC επιπέδου.

2.2.2.2 Επίπεδο ζεύξης MAC

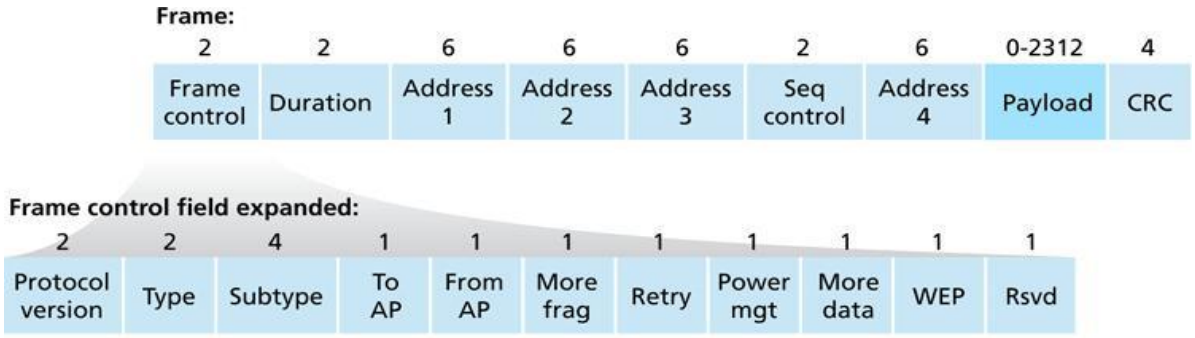
Το επίπεδο ζεύξης ορίζει τα διαδικαστικά και λειτουργικά μέσα για τη μεταφορά δεδομένων μεταξύ οντοτήτων. Γίνεται επίσης ανίχνευση και διόρθωση λαθών που τυχόν συμβαίνουν στο Φυσικό επίπεδο.

Επίπεδο διαχείρισης MAC: Ορίζει τη διαχείριση ενέργειας, την ασφάλειας και τη περιαγωγή.

Επίπεδο LLC: Παρέχει μηχανισμούς πολυπλεξίας ώστε τα διάφορα πρωτόκολλα δικτύου να συνυπάρχουν στο δίκτυο.

Επίπεδο MAC: Ορίζει τους μηχανισμούς πρόσβασης και τη μορφή των πακέτων.

Πιο αναλυτικά το πλαίσιο MAC περιέχει αρκετά πεδία που αφορούν την ασύρματη ζεύξη



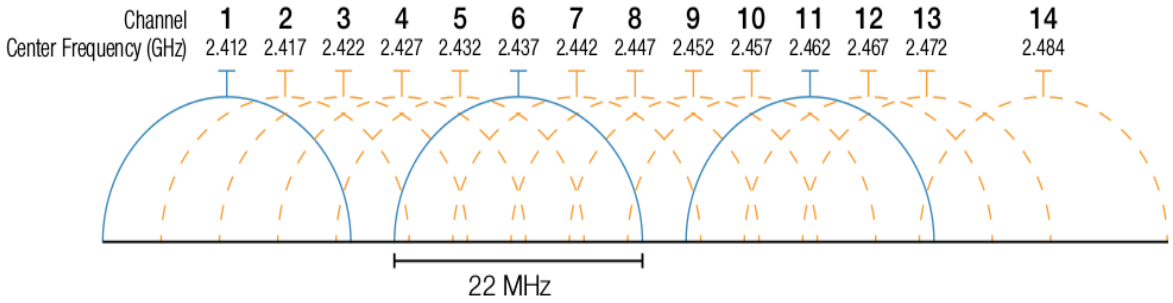
Σχήμα 13: Πλαίσιο 802.11

- Το **payload** (ωφέλιμο φορτίο) αποτελείται από ένα IP datagram ή ένα πακέτο IP. Το πεδίο επιτρέπεται να έχει 2312 bytes συνήθως λιγότερα από 1500 bytes.
- Το **CRC** (πεδίο κυκλικού ελέγχου πλεονασμού) 4 bytes ώστε ο δέκτης να μπορεί να ανιχνεύει τα σφάλματα στο ληφθέν πλαίσιο.
- Τα **Address** πεδία (πεδία διευθύνσεων) (4) περιέχει μια διεύθυνση MAC 6 bytes. Αποδεικνύεται ότι απαιτούνται 3 πεδία διευθύνσεων για τη μεταφορά datagram του επιπέδου δικτύου, ασύρματα μέσω AP. Το 4ο πεδίο χρησιμοποιείται μόνο όταν στέλνονται πακέτα ανάμεσα σε AP σε adhoc επίπεδο. Η Address 1 είναι η MAC διεύθυνση του ασύρματου σταθμού που πρόκειται να λάβει το πλαίσιο Address 2 είναι η MAC διεύθυνση του σταθμού που μεταδίδει το πλαίσιο. Η 3η Address περιέχει τη MAC διεύθυνση του δρομολογητή που βρίσκεται στο BSS.
- Το πεδίο **Seq control** επιτρέπει στο δέκτη του να αναγνωρίσει αν ένα μεταδιδόμενο πλαίσιο είναι νέο ή είναι αναμετάδοση ενός προηγούμενου. Αναμετάδοση πλαισίου μπορεί να συμβεί εάν η επιβεβαίωση του χαθεί και έτσι θα σταλούν πολλαπλά αντίγραφα του.
- Το πεδίο **Duration** δείχνει το χρονικό διάστημα για τον οποίο είχε δεσμευτεί το κανάλι στο οποίο γίνεται η μετάδοση από ένα κόμβο. Ο χρόνος αυτός υπολογίζεται σύμφωνα με το χρονικό διάστημα που απαιτείται να μεταφερθεί ένα πλαίσιο 802.11 μαζί με τη γνωστοποίησή του.
- Το **Frame control field** περιλαμβάνει πολλά υποπεδία. Το μέγεθός τους μετριέται σε bits.
- Το Protocol version περιγράφει την έκδοση 802.11 πρωτοκόλλου που χρησιμοποιείται

- Τα **Type** και **Subtype** τύπου χρησιμοποιούνται για τη διάκριση των πλαισίων συσχέτισης RTS και CTS
- Τα **To AP** και **From AP** αντίστοιχες διευθύνσεις
- Το **More Fragments** δηλώνει αν ακολουθούν άλλα τμήματα του πλαισίου όπως δεδομένα ή πλαίσια τύπου διαχείρισης.
- Το **Retry** δηλώνει αν το πλαίσιο προέρχεται από αναμετάδοση.
- Το **Power Management** υποδηλώνει αν ο σταθμός από όπου προέρχεται το πλαίσιο είναι σε ενεργή κατάσταση ή σε λειτουργία εξοικονόμησης ενέργειας.
- Το **More Data** υποδηλώνει σε ένα σταθμό που είναι σε λειτουργία εξοικονόμησης ενέργειας ότι ακολουθούν και άλλα πλαίσια.
- Το **WEP** χρησιμοποιείται για να δηλώσει αν υπάρχει κρυπτογράφηση ή όχι
- Το **RSVD** δηλώνει ότι όλα τα πακέτα πρέπει να επεξεργαστούν στη σειρά.

2.3 802.11g πρότυπο

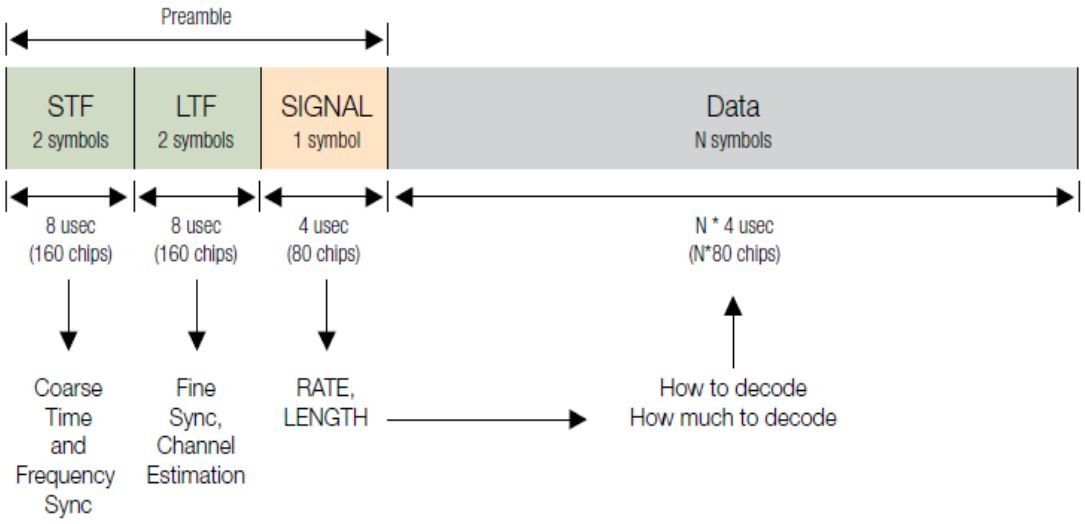
Το 2003 εγκρίθηκε η λειτουργία του 802.11g. Έχει γίνει το κυρίαρχο πρότυπο στη WiFi τεχνολογία. Χρησιμοποιεί τη μπάντα των 2.4GHz με μέγιστο ρυθμό μετάδοσης τα 54Mbp/s.



Σχήμα 14: 802.11g band

Στο 802.11g τα πακέτα δεδομένων αποτελούνται από δύο μέρη:

- **Κεφαλίδα:** Η κεφαλίδα ενημερώνει τους παραλήπτες ότι ακολουθεί η μετάδοση δεδομένων ώστε να συγχρονιστούν με τη μετάδοση.
- **Ωφέλιμο Φορτίο:** Είναι τα πραγματικά δεδομένα που θέλουμε να μεταφερθούν. Μπορεί να βρίσκονται σε μέγεθος από 64 bytes εώς 1500 bytes.



Σχήμα 15: Πακέτο 802.11g

Το 802.11g εκπέμπει στην ίδια μπάντα συχνοτήτων με το 802.11b, γι' αυτό υπάρχει η ανάγκη να συνυπάρχουν χωρίς προβλήματα. Στο 802.11g χρησιμοποιείται ένα εκτεταμένο εύρος φυσικού επιπέδου (ERP), υπάρχουν δηλαδή 4 φυσικά επίπεδα 2 υποχρεωτικά και 2 προαιρετικά.

Physical layer	Supported rates (Mb/s)	PLCP preamble + header delay		PLCP preamble + header length	
		Long	Short	Long	Short
ERP-DSSS (mandatory)	1, 2, 5.5, 11	192 µs	96 µs	192 bits	120 bits
ERP-OFDM (mandatory)	6, 9, 12, 18, 24, 36, 48, 54	20 µs		40 bits[1]	
ERP-PBCC (optional)	1, 2, 5.5, 11, 22, 33	192 µs	96 µs	192 bits	120 bits
DSSS-OFDM (optional)	6, 9, 12, 18, 24, 36, 48, 54	192 µs	96 µs	192 bits	120 bits

Σχήμα 16: Εκτεταμένο εύρος PHY 802.11g

Το ERP – DSSS / CCK χρησιμοποιείται όταν μια συσκευή 802.11g επικοινωνεί με μία συσκευή 802.11b. Το ERP – OFDM χρησιμοποιείται στην επικοινωνία συσκευών 802.11g. Το DSSS – OFDM είναι μαι υβριδική διαμόρφωση όπου η DSSS διαμορφώνει

τη κεφαλίδα, ενώ το OFDM το ωφέλιμο φορτίο. Το ERP – PBCC είναι μια διαμόρφωση μονού φέροντος που κωδικοποιεί το ωφέλιμο φορτίο με ένα 256 δυαδικό πακέτο για ρυθμό μετάδοσης 22 -33 Mbps.

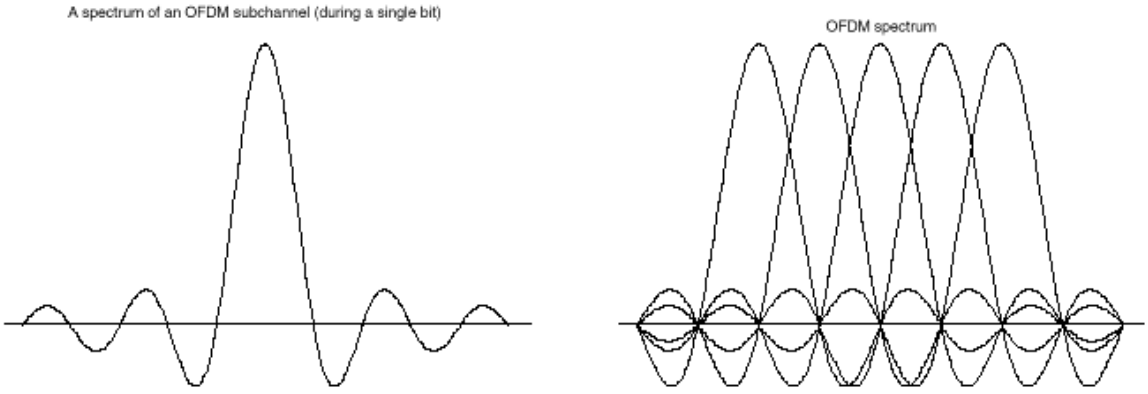
2.3.1 OFDM

Η OFDM είναι μια τεχνική μετάδοσης σήματος. Η OFDM (ορθογώνια πολυπλεξία με διαίρεση συχνότητας) μοιράζει το διαθέσιμο φάσμα καναλιού σε πολλά υποκανάλια και κάνει ταυτόχρονη αποστολή πολλών διαμορφωμένων σημάτων στις γειτονικές αυτές συχνότητες. Τα σήματα αυτά εκπέμπονται σε παραπλήσιες συχνότητες χωρίς να παρενοχλούν το ένα το άλλο καθώς αυτά τα σήματα είναι μεταξύ τους ορθογώνια.

Αν το σήμα κάποιου υποκαναλιού καταστραφεί είναι δυνατόν με τη κατάλληλη κωδικοποίηση να δημιουργηθεί ξανά με δεδομένα από τα υπόλοιπα κανάλια. Έτσι αντιμετωπίζεται η παραμόρφωση από τη διάδοση πολλαπλών διαδρομών που δημιουργείται σε αστικά περιβάλλοντα λόγω την ύπαρξη εμποδίων (κτήρια, ανάκλαση) που εμποδίζουν την άμεση οπτική επαφή. Διαφορετική στιγμή άφιξης σημάτων.

Στην OFDM διαμόρφωση οι απώλειες αυτές μειώνονται δραστικά. Το κάθε bit της εκπομπής έχει μεγαλύτερη διάρκεια και τα εξ ανακλάσεως σήματα φτάνουν με καθυστέρηση μικρότερη όμως της διάρκειας εκπομπής του κάθε bit και έτσι δε επηρεάζονται τα γειτονικά bits.

Κάθε μετάδοση bit σχηματίζει μια διαδοχική σειρά τετραγωνικών παλμών οι οποίοι διαμορφώνουν το φορέα ραδιοσήματος ώστε να δημιουργείται ένα ημιτονοειδές συμμετρικό φάσμα στον άξονα της συχνότητας. Τα σημεία μηδενικής συχνότητας εμφανίζονται σε πολλαπλάσια απόσταση από τη διάρκεια του παλμού (bitrate). Πρόσθετοι φορείς μπορούν να τοποθετηθούν στα σημεία μηδενικής συχνότητας ώστε να δημιουργούν την ελάχιστη παρεμβολή στο γειτονικό φορέα. Οι φορείς με αυτό το τρόπο τοποθετούνται σε φάση 90ο μοιρών.



Σχήμα 17: Ορθογώνια κανάλια στην OFDM διαμόρφωση

Μια επιπλέον βελτίωση του OFDM είναι η χρήση μικρών νεκρών διαστημάτων ανάμεσα στη μετάδοση των bits. Ο φορέας μετά τη διαμόρφωσή του από ένα bit επιστρέφει και παραμένει για ένα διάστημα σε κατάσταση μη διαμόρφωσης. Το μικρό αυτό νεκρό σημείο επιλέγεται να είναι μεγαλύτερο από τη καθυστέρηση των σημάτων που φτάνουν μέσω ανάκλασης. Έτσι οι παρεμβολές λόγω ανακλάσεων εκμηδενίζεται, ενώ έχει κόστος μια μικρή μείωση του ρυθμού μετάδοσης. Η μετάδοση σε κοντινές συχνότητες επιτυγχάνει μεγαλύτερους ρυθμούς μετάδοσης.

2.3.2 DSSS

Η τεχνολογία μετάδοσης ευρέως φάσματος DSSS κάθε bit κωδικοποιείται με ένα τρόπο που είναι γνωστός μόνο στο αποστολέα και το δέκτη και εκπέμπεται στ όλες τις διαθέσιμες συχνότητες. Έτσι αν ένας μη εξουσιοδοτημένος χρήστης αποκτήσει πρόσβαση στα δεδομένα πολύ δύσκολα θα μπορέσει να αναγνωρίσει το πραγματικό περιεχόμενο. Το DSSS χρησιμοποιεί τη τεχνική διαμόρφωσης DB-PSK για μετάδοση δεδομένων σε ταχύτητα 1Mbps και DQ-PSK για ταχύτητα 2 -11Mbps.

Τα DSSS συστήματα κατανέμουν το εκπεμπόμενο σύστημα σε ένα μεγάλο εύρος συχνοτήτων διαμορφώνοντας το αρχικό σήμα με ένα μεγαλύτερου εύρους ζώνης σήμα. Για τη διασπορά αυτή χρησιμοποιείται ένα σήμα από 11 bit που ονομάζεται chipping code και πολλαπλασιάζεται με κάθε bit του σήματος προς διαμόρφωση. Έπειτα κάθε bit που μεταδίδεται παράγει μια σειρά 11 ψηφίων που ονομάζονται chips.

Ο λόγος chips/bits ονομάζεται λόγος διασποράς και χαρακτηρίζει το βαθμό διασποράς στη ζώνη συχνοτήτων. Το πρότυπο ορίζει ως τιμή τα 11 chips/bit. Όσο μεγαλύτερο είναι ο αριθμός των chips τόσο ανθεκτικότερη σε σφάλματα είναι η μετάδοση αλλά και τόσο μεγαλύτερο φάσμα συχνοτήτων απαιτείται. Στο δέκτη γίνεται η αντίστροφη μετατροπή των 11 chips σε 1 bit για την εξαγωγή του αρχικού σήματος.

Μεγάλο πλεονέκτημα του DSSS είναι ότι η διασπορά του σήματος σε ένα ευρύτερο φάσμα το κάνει ανθεκτικότερο σε παρεμβολές και βοηθάει το δέκτη να αναγνωρίσει το σήμα ακόμα και αν είναι ασθενές ή μέσω θορυβώδους περιβάλλοντος.

2.3.3 CCK

Η CCK τεχνική (Complementary code keying) διαμορφώνει δεδομένα σε ρυθμούς μετάδοσης 5.5 Mbps και 11 Mbps. Χρησιμοποιείται ένα σήμα 8 chips/bit για ρυθμό μετάδοσης 5.5 Mbps και ένα σήμα 4 chips/bit για ρυθμό μετάδοσης 11Mb/s, ώστε να πολλαπλασιάζεται με κάθε bit του σήματος που διαμορφώνει.

Η χρήση μικρότερου αριθμού chips επιτυγχάνει μεγαλύτερους ρυθμούς μετάδοσης στο διαθέσιμο εύρος ζώνης.

2.3.4 PBCC

Η PBCC τεχνική (Packet Binary Convolutional Coding) διαμορφώνει δεδομένα με ρυθμούς μετάδοσης 22Mbps και 33Mbps. Η κωδικοποίηση PBCC χρησιμοποιεί 1/2 ρυθμό διαμόρφωσης παράγοντας έτσι δύο εξόδους bits για κάθε ένα bit εισόδου.

Η έξοδος της PBCC διαμορφώνεται με QPSK διαμόρφωση για ρυθμό μετάδοσης 22Mbps και με PBSK για διαμόρφωση ρυθμού μετάδοσης 33Mbps. Το ωφέλιμο φορτίο κωδικοποιείται με μια ακολουθία 256 bits.

2.3.5 CSMA / CA

Όταν το ασύρματο δίκτυο αποτελείται από πολλούς κόμβους που θέλουν να μεταδίδουν δεδομένα ταυτόχρονα πάνω στο ίδιο κανάλι, χρειάζεται ένα πρωτόκολλο πολλαπλής πρόσβασης ώστε να συντονίζει τις μεταδόσεις. Τα ασύρματα δίκτυα χρησιμοποιούν τη μέθοδο πολλαπλής πρόσβασης CSMA / CA (Carrier Sense Multiple Access/ Collision Avoidance) ώστε να λαμβάνονται μέτρα για την αποφυγή της σύγκρουσης και να επιβεβαιώνεται η ορθή λήψης των δεδομένων από το δέκτη.

Ο κάθε κόμβος ακούει το κανάλι πριν τη μετάδοση των δεδομένων για να ελέγξει αν κάποιος άλλος κόμβος μεταδίδει ή όχι. Πριν η συσκευή εκπέμψει περιμένει ένα τυχαίο διάστημα, το οποίο καθορίζεται από μια ψευδοτυχαία γεννήτρια καθυστέρησης. Έτσι αποφεύγεται η σύγκρουση όταν πολλοί σταθμοί θέλουν να εκπέμψουν μόλις ελευθερωθεί το κανάλι.

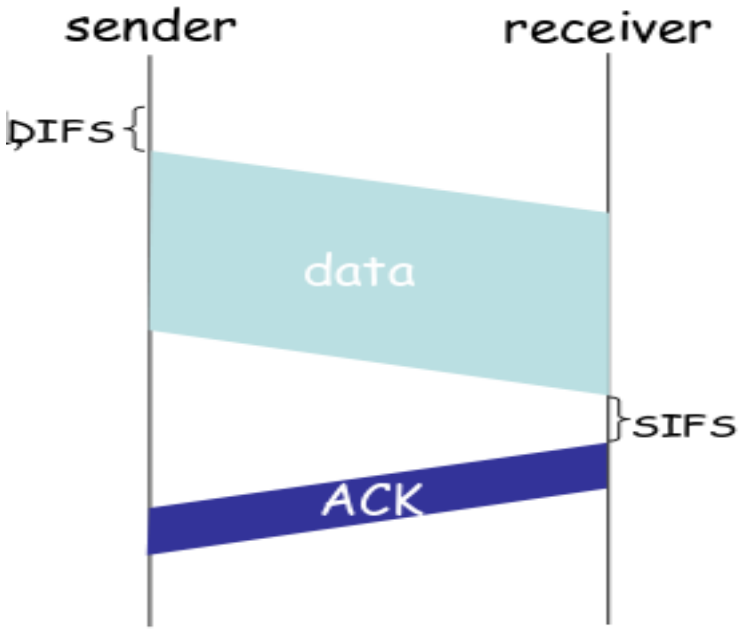
Υπάρχει επιπλέον ένας πρόσθετος χρόνος που πρέπει να περιμένει μια συσκευή για να εκπέμψει μετά την απελευθέρωση του καναλιού και ονομάζεται απόσταση πλαισίων IFS (InterFrame Spacing). Ο χρόνος αυτός εξαρτάται από το τύπο του 802.11 πλαισίου που πρόκειται να σταλεί και το τύπο διαμόρφωσης. Οι καθυστερήσεις IFS χωρίζονται σε:

- **SIFS (Short IFS)** Οι μικρότερες καθυστερήσεις που προορίζονται για πακέτα ελέγχου.
- **DIFS (Distributed IFS)** που προορίζονται για πλαίσια δεδομένων
- **EIFS (Extended IFS**) Προορίζονται για περιπτώσεις επανεκπομπής πλαισίων που συμβαίνουν μετά από σφάλμα στη μετάδοση και έχουν μεγάλη καθυστέρηση.

Όταν ένας κόμβος βρει το κανάλι αδρανές μεταδίδει το πλαίσιό του μετά από μια καθυστέρηση DIFS. Αλλιώς επιλέγει μια τυχαία τιμή και μετρά αντίστροφα από αυτή τη τιμή και εφόσον το κανάλι είναι αδρανές. Για όσο το κανάλι είναι απασχολημένο αυτή η τιμή περιμένει παγιωμένη. Όταν η τιμή αυτή φτάσει στο μηδέν ο σταθμός εκπέμπει όλο το πλαίσιο και περιμένει μια γνωστοποίηση (ACK).

Όταν γίνει επιτυχώς η λήψη του ζητηθέντος πλαισίου, ένα ACK στέλνεται από το λήπτη

για να μην θεωρηθεί ότι το πακέτο χάθηκε και ζητήθηκε αναμετάδοση. Αν δεν ληφθεί ACK ο κόμβος που έστειλε το πλαίσιο επιλέγει μια μεγαλύτερη τιμή καθυστέρησης και ακολουθεί την ίδια διαδικασία αποστολής πλαισίου.



Σχήμα 18: Επιβεβαιώσεις επιπέδου ζεύξης στο 802.11

3. Μηχανισμός Broadcast Aware Peer to Peer (BAP2P)

3.1 Εισαγωγή

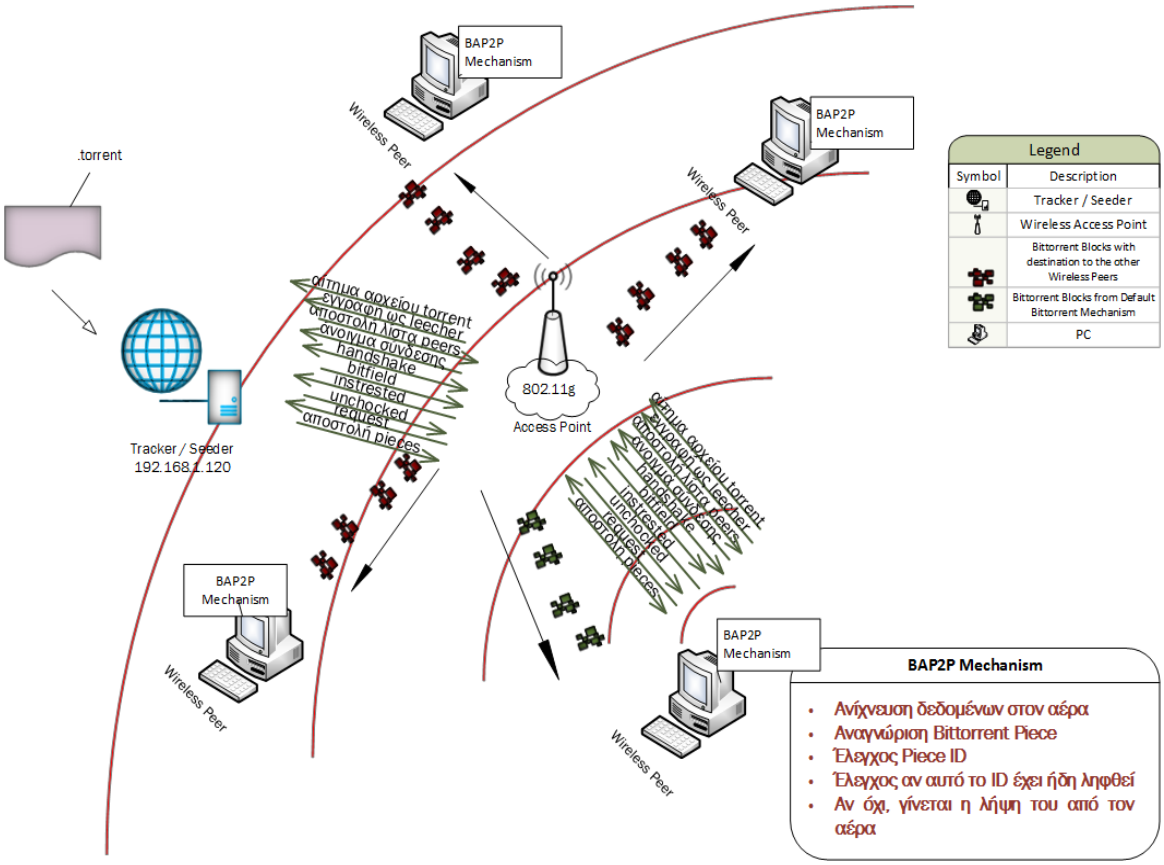
Σκοπός του μηχανισμού BAP2P είναι η καλύτερη εκμετάλλευση των πόρων του ασύρματου δικτύου. Ο μηχανισμός εγκαθιστάτε στους κόμβους του δικτύου που θα πραγματοποιήσουν τη λήψη του αρχείου.

Ο BAP2P ανιχνεύει την Bittorrent κίνηση που κυκλοφορεί στον "αέρα" του ασύρματου δικτύου η οποία ξεκινά να δημιουργείται από έναν τυπικό Bittorrent μηχανισμό. Όταν γίνεται αίτημα για ένα Piece από ένα peer και ξεκινήσει η λήψη του, οι γειτονικοί peers μπορούν να ανιχνεύσουν τη λήψη του συγκεκριμένου αυτού piece μέσω του αντίστοιχου μηνύματος Bittorrent, Piece.

Υπενθυμίζουμε ότι τα Piece μηνύματα χρησιμοποιούνται για την αποστολή των blocks και περιέχουν τα παρακάτω πεδία:

- **Index** Ο αριθμός του piece
- **Begin** Ο αριθμός του block
- **Block** Ακολουθία bytes που περιέχουν τα δεδομένα του block

Ο peer ελέγχει αν έχει ήδη λάβει αυτό το piece κοιτάζοντας τον αριθμό του piece (index) και τον αριθμό του block (Begin). Αν όχι το λαμβάνουν από τον αέρα όπως και ο peer που το ζήτησε αρχικά.

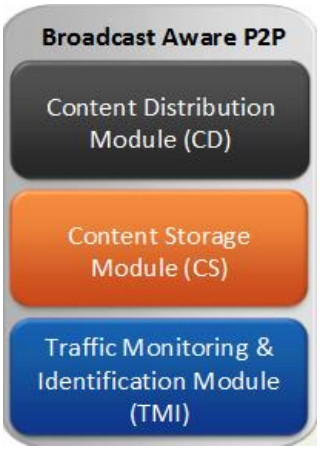


Σχήμα 19: Τρόπος λειτουργίας μηχανισμού BAP2P

3.2 Αρχιτεκτονική BAP2P

Αποτελείται από 3 τμήματα:

- **Traffic Monitoring & Identification Module (TMI)**: Ανιχνεύει την Bittorrent κίνηση
- **Content Storage Module (CS)**: Αποθηκεύει και διαχειρίζεται το περιεχόμενο
- **Content Distribution Module (CD)**: Διανέμει το περιεχόμενο στους peers



Σχήμα 20: Στοίβα μηχανισμού BAP2P

3.2.1 Traffic Monitoring and Identification

Ανιχνεύει τη Bittorrent κίνηση που γίνεται στον αέρα του ασύρματου δικτύου. Εντοπίζει τα Handshake και Piece μηνύματα και τα προωθεί στο Content Storage.

3.2.2 Content Storage

Διαχειρίζεται τα δεδομένα που ανιχνεύει ο TMI και είναι υπεύθυνος για την αποθήκευσή τους. Δέχεται ως είσοδο τα Handshake και Piece μηνύματα και εξάγει πληροφορίες από τα πακέτα αυτά για κάθε ένα από αυτά τα συγκεκριμένα μηνύματα.

Από το μήνυμα Handshake μπορεί να αναγνωρίσει σε ποίο αρχείο αναφέρονται τα δεδομένα της σύνδεσης, καταγράφοντας το Hash Info, την IP και τις θύρες που χρησιμοποιούν αντίστοιχα αποστολέας και παραλήπτης.

Το Piece μήνυμα είναι αυτά που περιέχουν τα δεδομένα του Bittorrent αρχείου. Υπάρχει στο μήνυμα επίσης το Piece Index και το Piece Length που βοηθάνε στη ταξινόμηση και αποθήκευση των δεδομένων που περνάνε από το μηχανισμό.

Δεν μπορούμε να αναγνωρίσουμε μέσω του piece μηνύματος σε ποιό αρχείο αναφέρεται το κάθε piece μήνυμα, γι΄ αυτό είναι απαραίτητη η καταγραφή των hashshake μηνυμάτων.

3.2.3 Content Distribution

Περιέχει τις βασικές λειτουργίες ενός Bittorrent Client και μπορεί να διαθέσει στους peers το αρχείο Bittorrent που διαμοιράζεται από το μηχανισμό. Κάνει broadcast ένα μήνυμα HAVE σε όλους τους peers της γειτονιάς τους κάθε φορά που αποθηκεύει ένα block και γίνεται αυτό διαθέσιμο από το μηχανισμό. Έτσι οι peers μπορούν να γνωρίζουν ότι το συγκεκριμένο block είναι διαθέσιμο από τον αέρα μέσω του BAP2P μηχανισμού.

4. Αξιολόγηση μηχανισμού

4.1 Εισαγωγή

Η αξιολόγηση του μηχανισμού BAP2P πραγματοποιήθηκε σε πραγματικές συνθήκες ασύρματου τοπικού δικτύου. Χρησιμοποιήθηκε η αρχιτεκτονική του Bittorrent πρωτοκόλλου peer to peer με 4 ασύρματους peers που "τρέχουν" τον BAP2P μηχανισμό και τον τυπικό Bittorrent μηχανισμό, ένας τοπικός tracker που λειτουργεί και ως seeder και ένα AP που χρησιμοποιεί το πρωτόκολλο 802.11g. Στόχος σε όλη την υλοποίηση είναι η ανεμπόδιστη και χωρίς οποιοδήποτε περιορισμό μεταφορά των δεδομένων ανάμεσα στο τοπικό δίκτυο τόσο από πλευράς software αλλά και hardware ώστε ο πρότυπος μηχανισμός BAP2P να αποδώσει το μέγιστο δυνατό αποτέλεσμα.

4.2 Λογισμικό υλοποίησης

Ο σκοπός των εργαλείων που χρησιμοποιήθηκαν είναι για τη συνολική αξιολόγηση και εγκυρότητα των μετρήσεων που ελήφθησαν. Το λογισμικό που χρησιμοποιήθηκε κατά τη διάρκεια της υλοποίησης είναι:

4.2.1 XBT Tracker

Είναι ένας Bittorrent tracker αναπτυγμένος σε C++. Διαθέτει μόνο backend διαχείριση με τη χρήση της MySQL. Χρησιμοποιήθηκε ώστε να φιλοξενηθεί εκεί το .torrent αρχείο που αντιστοιχεί στο αρχείο προς διαμοιρασμό.

4.2.2 Vuze

Το Vuze είναι ένας Bittorrent client αναπτυγμένος σε JAVA. Χρησιμοποιήθηκε από τον αρχικό seeder ώστε να ξεκινήσει η λήψη του αρχείου από τους υπόλοιπους peers μέσω του μηχανισμού BAP2P.

4.2.2 Kali Linux OS

Είναι μια διανομή Linux βασισμένη στο Debian λειτουργικό. Είναι προσανατολισμένο για να πραγματοποιεί δοκιμές διείσδυσης και διαθέτει αρκετά εργαλεία για τη πραγματοποίηση τέτοιον ενεργειών.

4.2.3 Ubuntu Linux OS

Είναι μια διανομή Linux βασισμένη στο Debian λειτουργικό. Δημιουργήθηκε για να παρέχει τη πιο εύκολη στη χρήση διανομή Linux.

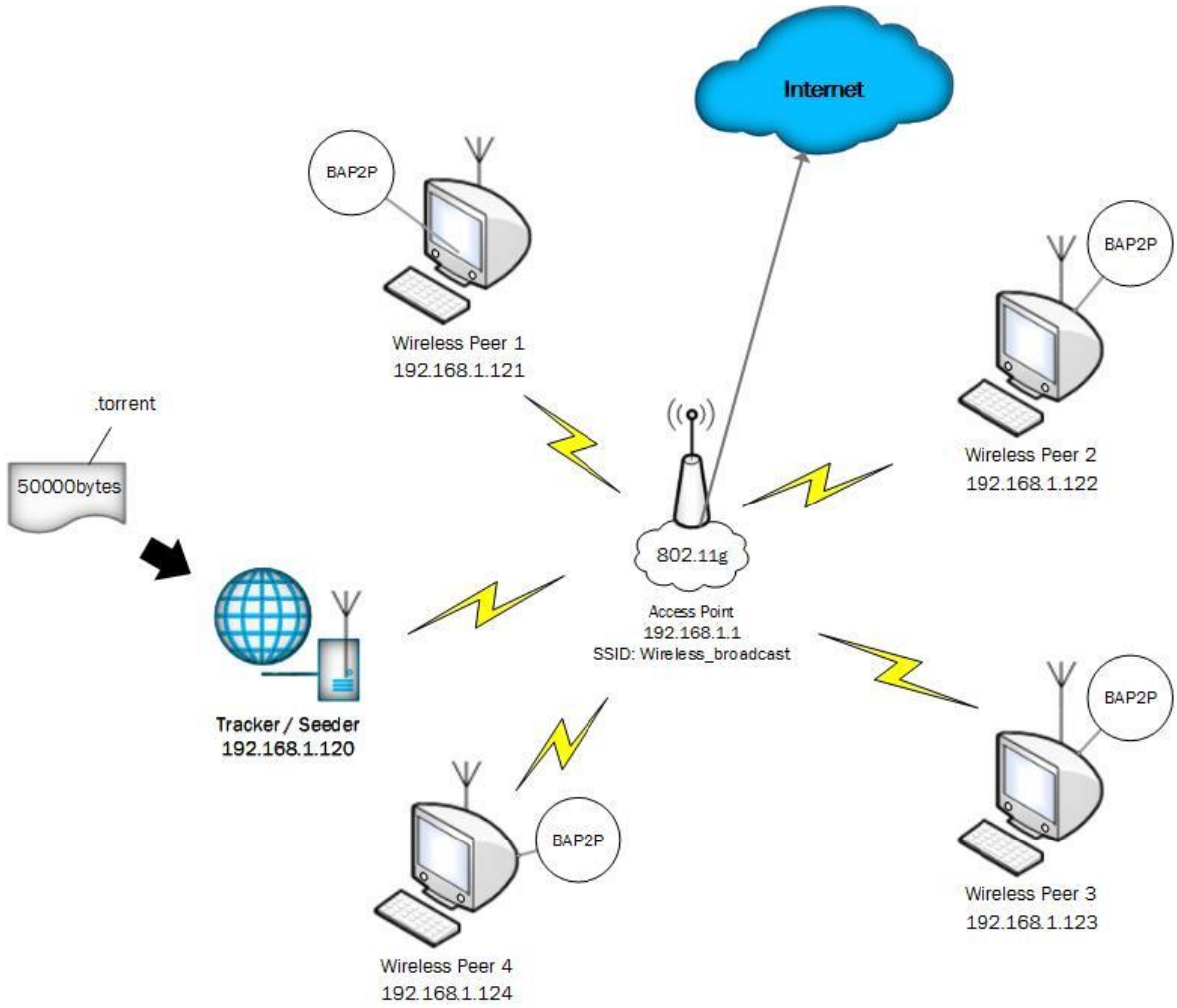
4.2.4 Iperf

Το Iperf είναι ένα εργαλείο δημιουργίας δικτυακής κίνησης TCP ή UDP γραμμένο στη γλώσσα προγραμματισμού C. Μπορεί να χρησιμοποιηθεί για τη μέτρηση της απόδοσης του εύρους δικτύου.

4.2.5 Tcpdump

Το Tcpdump είναι ένα εργαλείο ανάλυσης δικτυακής κίνησης. Μπορεί να προβάλει οποιαδήποτε δικτυακή κίνηση δέχεται ο υπολογιστής στον οποίο τρέχει. Εκτελείται από τη γραμμή εντολών του υπολογιστή.

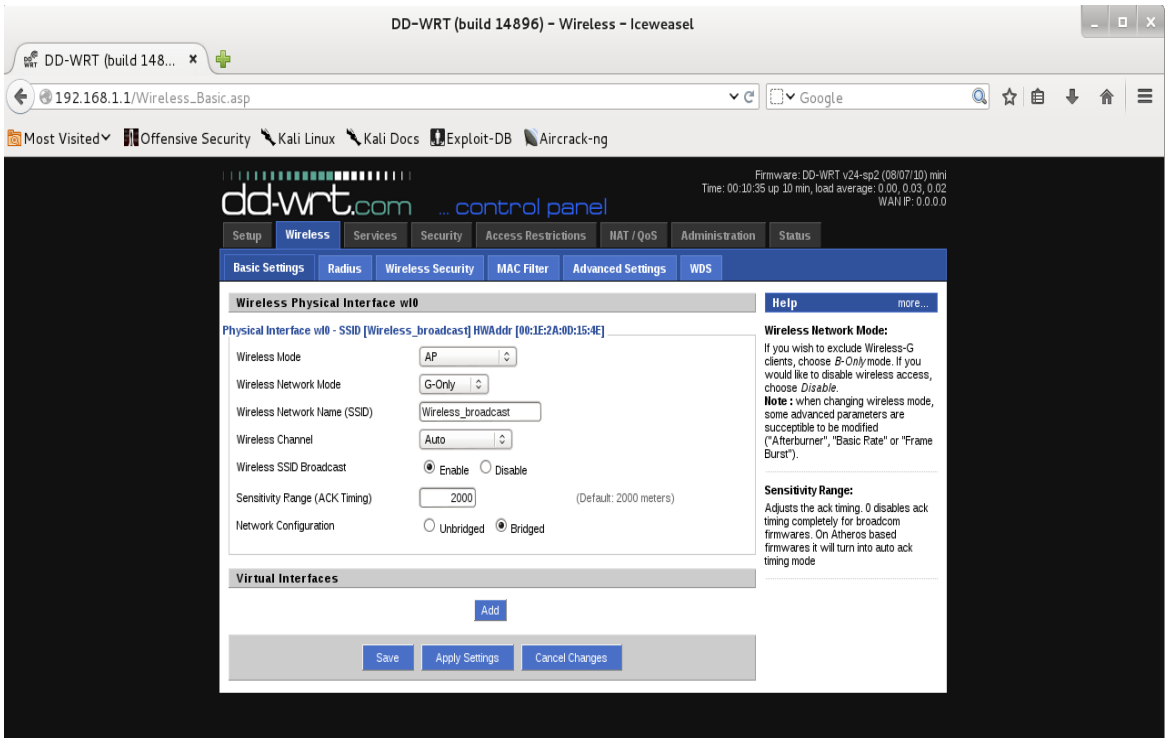
4.3 Περιγραφή υλοποίησης



Σχήμα 21: Τοπολογία δικτύου

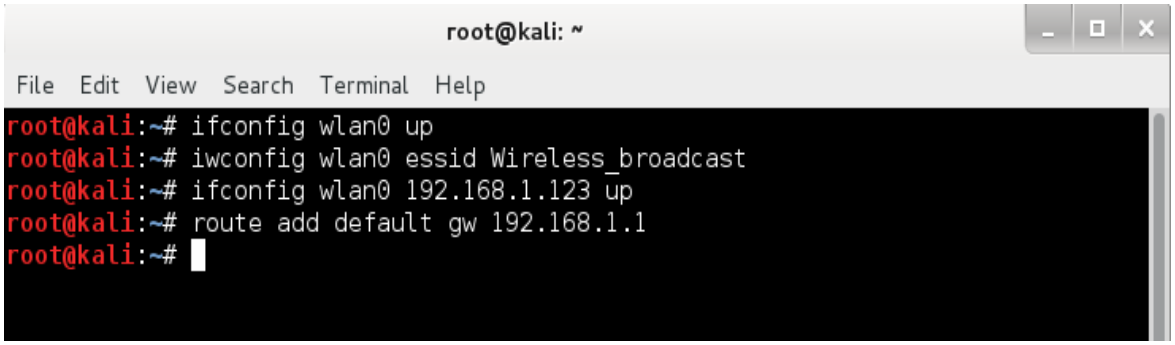
Χρησιμοποιήθηκαν σταθεροί υπολογιστές με ενσωματωμένες ασύρματης εμβέλειας κάρτες δικτύου με chipset της Marvel Technology Group Ltd 88E8001 Gigabit Controller (rev 14). Πάνω στις κάρτες δικτύου εφαρμόστηκαν κεραίες ενίσχυσης ώστε να αυξηθεί η σταθερότητα του σήματος και η ταχύτητας σύνδεσης . Οι υπολογιστές έτρεξαν με το λειτουργικό Kali Linux.

Ως AP χρησιμοποιήθηκε μια ασύρματη συσκευή της Netgear WNR834B v2. Ως SSID ορίστηκε το Wireless_broadcast και ως πρότυπο το 802.11g για τη λειτουργία του καθώς με αυτό το πρότυπο λειτουργεί ο κεντρικός router του εργαστηρίου συντελέστηκε το πείραμα. Στο AP απενεργοποιήθηκε η WEP ασφάλεια για να έχουμε ανεμπόδιστη μεταφορά των δεδομένων.



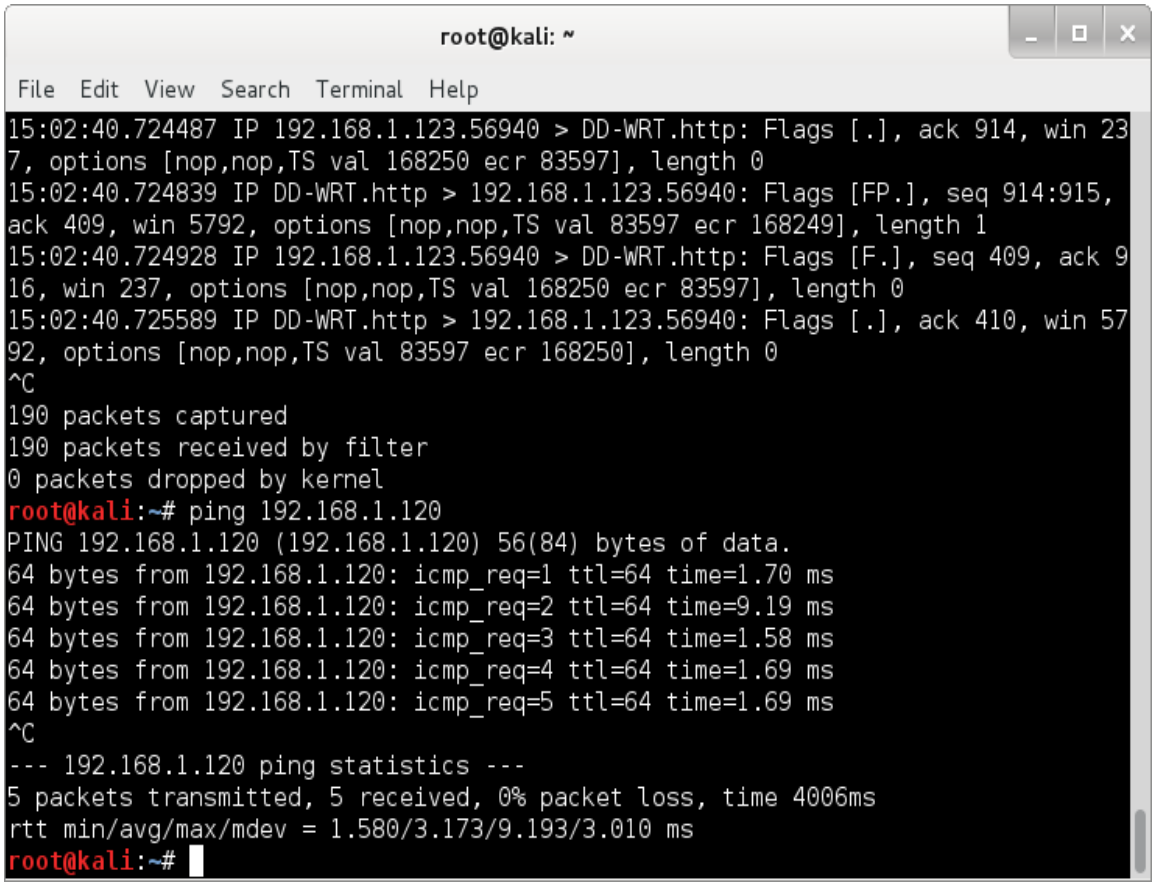
Σχήμα 22: Ρύθμιση Access Point

Κατά τη διάρκεια σύνδεσης των υπολογιστών με το AP μέσω του Network Manager (το λογισμικό που διαχειρίζεται τις δικτυακές συνδέσεις στα Linux OS), παρατηρήθηκαν συχνές αποσυνδέσεις από το AP και προσπάθεια σύνδεσης των υπολογιστών με άλλα ασύρματα δίκτυα που βρίσκονται στο πεδίο κάλυψης. Έτσι απενεργοποιήθηκε το γραφικό περιβάλλον του Network Manager και οι υπολογιστές συνδέονται στο AP μέσω του Terminal στο wlan0 interface με τις ακόλουθες εντολές.



Σχήμα 23: Σύνδεση υπολογιστών με το AP

Έπειτα ελέγχουμε αν οι υπολογιστές επικοινωνούν μεταξύ τους ώστε να έχουμε επιτυχημένη λειτουργία του BAP2P μηχανισμού που θα χρησιμοποιήσουμε αργότερα. Εκτελούμε την εντολή ping στο Terminal ώστε να εντοπιστεί η διαθεσιμότητα του πόρου από το κάθε υπολογιστή προς τους υπόλοιπους. Το αποτέλεσμα της εντολής ping ονομάζεται και χρόνος μετάδοσης μετ' επιστροφή (RTT). Το RTT πρακτικά είναι η χρονική διάρκεια που απαιτείται για να μεταδοθεί ένα σήμα από τον αποστολέα στον παραλήπτη, συν την επιβεβαίωση της λήψης του σήματος από τον παραλήπτη.



Σχήμα 24: Στιγμιότυπο εκτέλεσης ping

Τα αποτελέσματα RTT που μετρήθηκαν για το κάθε ένα συμμετέχοντα κόμβο προς το gateway του AP είναι τα εξής:

	Seeder/ Tracker	Wireless Peer 1	Wireless Peer 2	Wireless Peer 3	Wireless Peer 4
RTT min	1.130 ms	0.764ms	0.796 ms	0.814 ms	0.780 ms
RTT avg	1.624 ms	0.848ms	0.857 ms	0.850 ms	0.866 ms
RTT max	3.039 ms	2.016ms	1.806 ms	0.859 ms	1.801 ms
RTT mdev	0.500 ms	0.284ms	0.233 ms	0.028 ms	0.293 ms

Σχήμα 25: RTT από τους peers προς AP Gateway

Έχοντας συνδέσει επιτυχώς τους υπολογιστές στο AP και έχουμε ελέγξει την επιτυχή επικοινωνία μεταξύ τους θα πρέπει να καταγράψουμε το bandwidth του δικτύου με τη χρήση των δικτυακών εργαλείων tcpdump και iperf. Έπειτα θα συγκρίνουμε τα αποτελέσματα με αυτά που θα προκύψουν από τις μετρήσεις του τυπικού μηχανισμού Bittorrent και του μηχανισμού BAP2P.

Συνδέουμε το AP eth to eth με έναν υπολογιστή (client) και έπειτα ενεργοποιούμε ένα άλλο υπολογιστή ως server. Ξεκινάμε το tcpdump σε client και server ώστε να ανιχνεύσουμε τη κίνηση που θα δημιουργηθεί έπειτα με το iperf από το server.
Στον server γράφουμε

```
tcpdump -w /root/Desktop/sAP -i wlan0 tcp port 5001 -n -vv
```

και στον client

```
tcpdump -w /home/mary/Desktop/cAP -i eth0 tcp port 5001 -n
-vv
```

Η κίνηση που θα ανιχνευθεί θα αποθηκεύεται στα αρχεία sAP και cAP αντίστοιχα

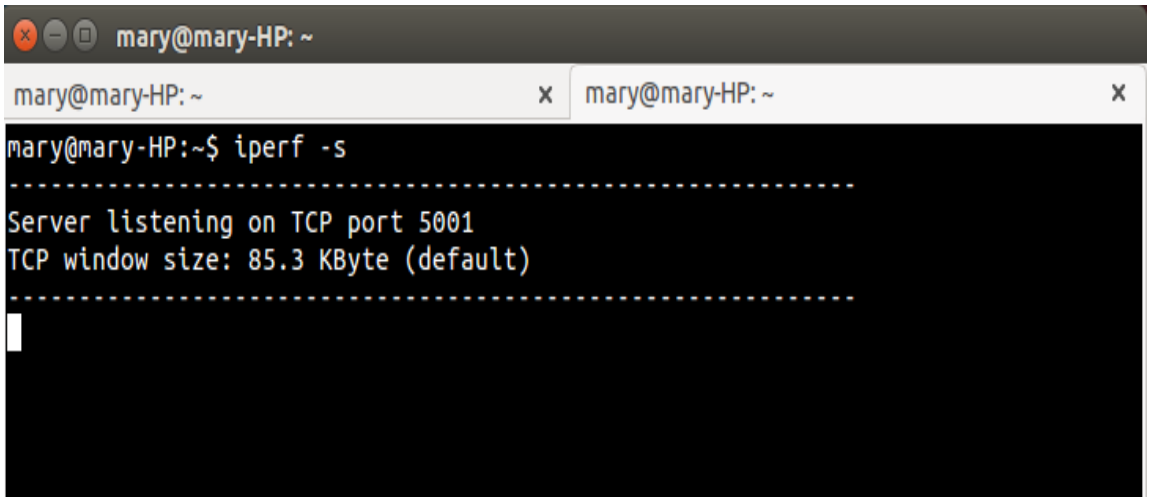
Τώρα μπορούμε να δημιουργήσουμε τη δικτυακή κίνηση και να εξάγουμε τα δεδομένα. Θα μεταφέρουμε το αρχείο που θα χρησιμοποιήσουμε στη εκτέλεση του μηχανισμού BAP2P.
Στον server γράφουμε

```
iperf -c 192.168.1.120 -F /root/Desktop/50MBytes.rar.rar
```

και στον client

```
iperf -s
```

Την ίδια διαδικασία ακολουθούμε για όλους τους υπολογιστές.



Σχήμα 26: Στιγμιότυπο εκτέλεσης iperf

Τα αποτελέσματα που καταγράφηκαν είναι:

	Seeder/ Tracker	Wireless Peer 1	Wireless Peer 2	Wireless Peer 3	Wireless Peer 4
Bandwidth (Mbps)	8,91	10,6	10,2	10,6	10,3

Έπειτα προχωράμε στην εγκατάσταση και ρύθμιση του tracker. Ο tracker θα βοηθήσει τους peer / υπολογιστές του δικτύου που χρησιμοποιούν το Bittorrent πρωτόκολλο, να επικοινωνούν μεταξύ τους. Στην υλοποίηση αυτή χρησιμοποιούμε τον opensource tracker XBT_Tracker τον οποίο και εγκαθιστούμε σε υπολογιστή με λειτουργικό Ubuntu. Για την εγκατάσταση προαπαιτούνται τα πακέτα που πρέπει να εγκατασταθούν και αυτά στο λειτουργικό:

- boost-devel
- mysql-devel
- gcc-c++
- subversion

Έπειτα ξεκινάμε την εγκατάσταση του tracker. Μετά την εγκατάσταση θα δημιουργηθεί ένας φάκελος xbt.

```
svn co http://xbt.googlecode.com/svn/trunk/xbt/misc
xbt/misc
```

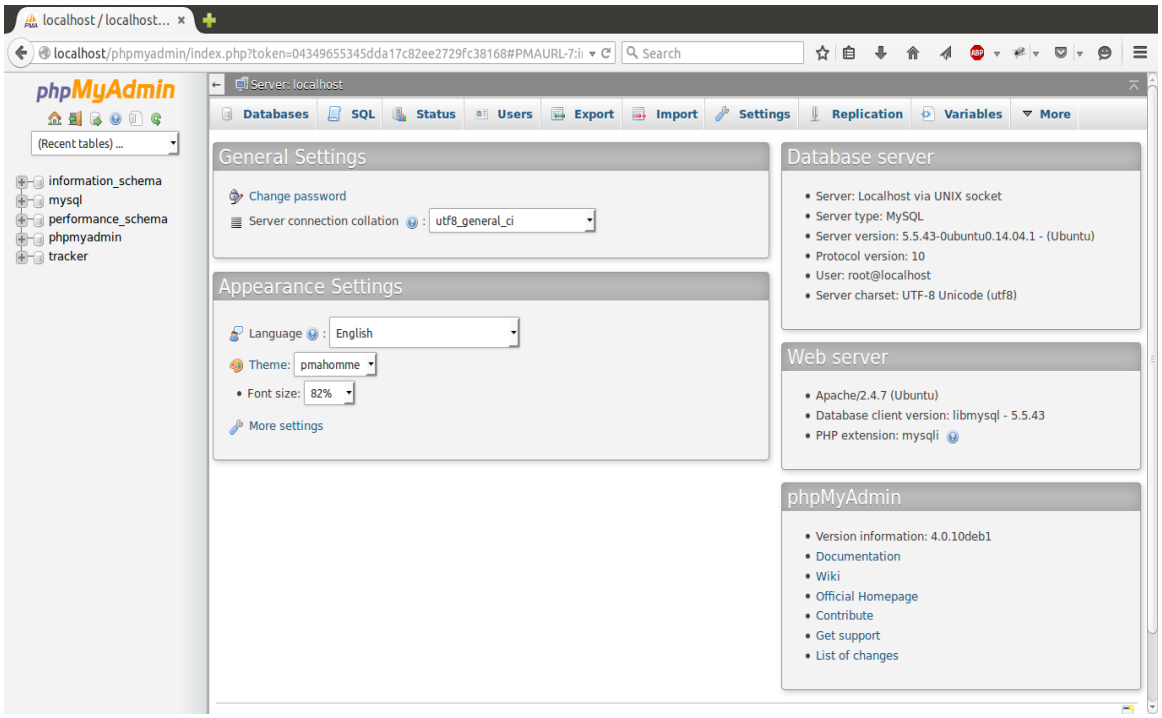
```
svn co http://xbt.googlecode.com/svn/trunk/xbt/Tracker
xbt/Tracker
```

```
cd xbt/Tracker
```

./make.sh

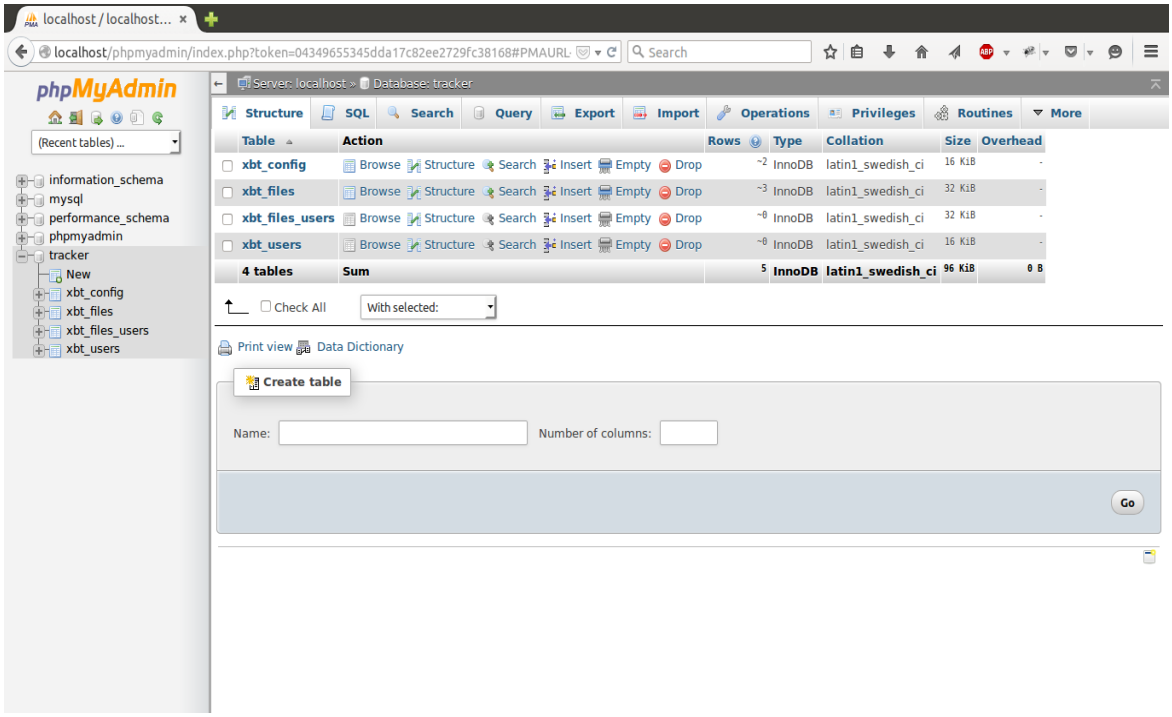
Αν θέλουμε να μετακινήσουμε σε άλλο directory τον tracker, αρκεί να αντιγράψουμε το binary αρχείο Tracker και το αρχείο xbt_tracker.conf.default από το φάκελο xbt στο επιθυμητό directory. Μετά την αντιγραφή πρέπει να μετονομάσουμε το αρχείο xbt_tracker.conf.default σε xbt_tracker.conf.

Έπειτα πρέπει να δημιουργηθεί η βάση δεδομένων του tracker. Για να δημιουργηθεί τοπικά η mysql βάση που χρειαζόμαστε πρέπει να εγκαταστήσουμε πρώτα το εργαλείο διαχείρισής της. Θα χρησιμοποιήσουμε το phpmyadmin ώστε να δημιουργήσουμε, να ρυθμίσουμε και να διαχειριστούμε τη βάση αυτή. Το phpmyadmin προϋποθέτει την εγκατάσταση των Apache, MySQL, PHP για να λειτουργήσει. Μετά την εγκατάστασή του μπαίνουμε στο localhost από το browser μας στο περιβάλλον διαχείρισης.



Σχήμα 27: Περιβάλλον διαχείρισης XBT Tracker

Για να δημιουργήσουμε τη βάση που επιθυμούμε γράφουμε πηγαίνουμε στο tab Databases → Create Database δίνουμε το όνομα tracker → Create.



Σχήμα 28: Οι βάσεις mysql του XBT Tracker

Μετά τη δημιουργία της πατάμε πάνω στη βάση πηγαίνουμε στο tab sql και βάζουμε το παρακάτω mysql query:

```
CREATE USER 'xbt_tracker'@'localhost' IDENTIFIED BY
'my_tracker_password';
GRANT USAGE ON * . * TO 'xbt_tracker'@'localhost'
IDENTIFIED BY 'my_tracker_password';
CREATE DATABASE IF NOT EXISTS `xbt_tracker` ;
GRANT ALL PRIVILEGES ON `xbt_tracker` . * TO
'xbt_tracker'@'localhost'
```

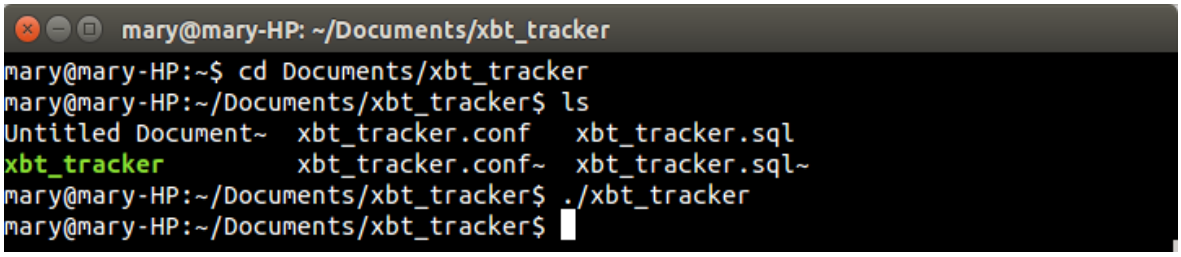
Για να εισάγουμε τη default mysql στη βάση γράφουμε το παρακάτω στο terminal αλλάζοντας κατάλληλα τα directories.

mysql -uxbt_tracker -pmy_tracker_password xbt_tracker < xbt_tracker.sql

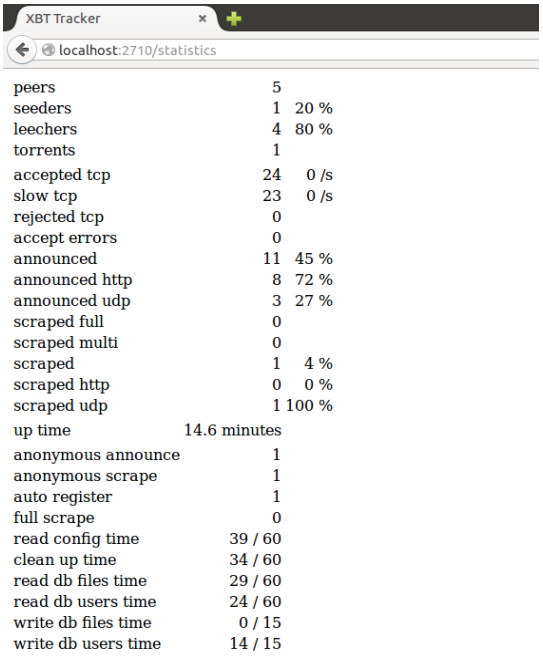
Αφού γεμίσουμε τη βάση μας ρυθμίζουμε τον tracker ανοίγοντας το αρχείο xbt_tracker.conf και γράφοντας:

```
mysql_host = localhost
mysql_user =  xbt_tracker
mysql_password =  my_tracker_password
mysql_database = tracker
```

Τώρα μπορούμε να ξεκινήσουμε τον tracker από το directory εγκατάστασης γράφοντας `./xbt_tracker`

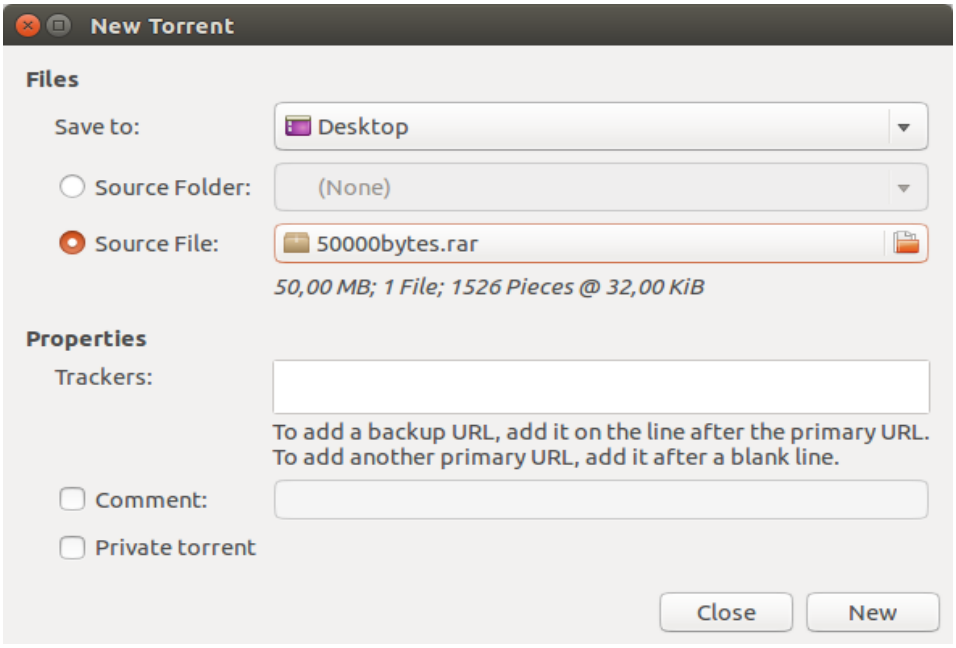


Σχήμα 29: Εκκίνηση XBT Tracker



Σχήμα 30: Στατιστικά χρήσης XBT Tracker

Το αρχείο που χρησιμοποιήθηκε για το διαμοιρασμό είναι ένα συμπιεσμένο αρχείο τύπου .rar με μέγεθος 50 Mbytes (50000000 bytes). Δημιουργούμε το αρχείο torrent για αυτό το αρχείο ώστε να μπορεί μέσω αυτού να διανεμηθεί σε όλους τους peers. Με τη βοήθεια του client Transmission ξεκινάμε τη διαδικασία δημιουργίας.



Σχήμα 31: Δημιουργία αρχείου .torrent

Το torrent αρχείο αυτό είναι χωρισμένο σε 1526 pieces μεγέθους 32Kib. Το κάθε piece αποτελείται από 2 blocks των 16Kib, δηλαδή 3052 blocks για όλο το αρχείο.

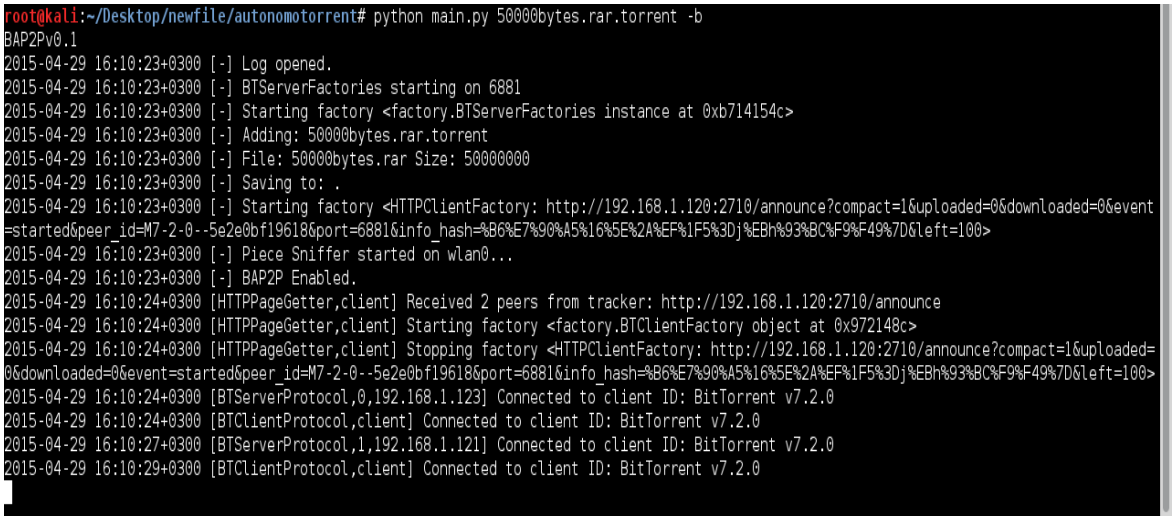
Τοποθετούμε το BAP2P μηχανισμό σε κάθε υπολογιστή και ξεκινάμε την αξιολόγηση υλοποιώντας μετρήσεις πρώτα με τον τυπικό Bittorrent μηχανισμό και έπειτα με το πρότυπο μηχανισμό BAP2P.

Σε κάθε υπολογιστή γράφουμε στο Terminal
`python main.py 50000bytes.rar.torrent`
για να ενεργοποιήσουμε τον Default Bittorrent μηχανισμό

```
root@kali:~/Desktop/newfile/autonomotorrent# python main.py 50000bytes.rar.torrent
BAP2Pv0.1
2015-04-29 15:58:49+0300 [-] Log opened.
2015-04-29 15:58:49+0300 [-] BTServerFactories starting on 6881
2015-04-29 15:58:49+0300 [-] Starting factory <factory.BTServerFactories instance at 0xb714254c>
2015-04-29 15:58:49+0300 [-] Adding: 50000bytes.rar.torrent
2015-04-29 15:58:49+0300 [-] File: 50000bytes.rar Size: 50000000
2015-04-29 15:58:49+0300 [-] Saving to: .
2015-04-29 15:58:49+0300 [-] Starting factory <HTTPClientFactory: http://192.168.1.120:2710/announce?compact=1&uploaded=0&downloaded=0&event
=started&peer_id=M7-2-0--25fcd9518e18&port=6881&info_hash=%B6%E7%90%A5%16%5E%2A%EF%1F5%3Dj%EBh%93%BC%F9%F49%7D&left=100>
2015-04-29 15:58:49+0300 [-] Piece Sniffer started on wlan0...
2015-04-29 15:58:49+0300 [-] BAP2P Disabled.
2015-04-29 15:58:49+0300 [HTTPPageGetter,client] Received 2 peers from tracker: http://192.168.1.120:2710/announce
2015-04-29 15:58:49+0300 [HTTPPageGetter,client] Starting factory <factory.BTClientFactory object at 0x9bff48c>
2015-04-29 15:58:49+0300 [HTTPPageGetter,client] Stopping factory <HTTPClientFactory: http://192.168.1.120:2710/announce?compact=1&uploaded=
0&downloaded=0&event=started&peer_id=M7-2-0--25fcd9518e18&port=6881&info_hash=%B6%E7%90%A5%16%5E%2A%EF%1F5%3Dj%EBh%93%BC%F9%F49%7D&left=100>
2015-04-29 15:58:49+0300 [BTServerProtocol,0,192.168.1.123] Connected to client ID: BitTorrent v7.2.0
2015-04-29 15:58:49+0300 [BTClientProtocol,client] Connected to client ID: BitTorrent v7.2.0
2015-04-29 15:58:50+0300 [BTServerProtocol,1,192.168.1.121] Connected to client ID: BitTorrent v7.2.0
2015-04-29 15:58:54+0300 [BTClientProtocol,client] Connected to client ID: BitTorrent v7.2.0
```

Σχήμα 32: Εκκίνηση Default Bittorrent μηχανισμού

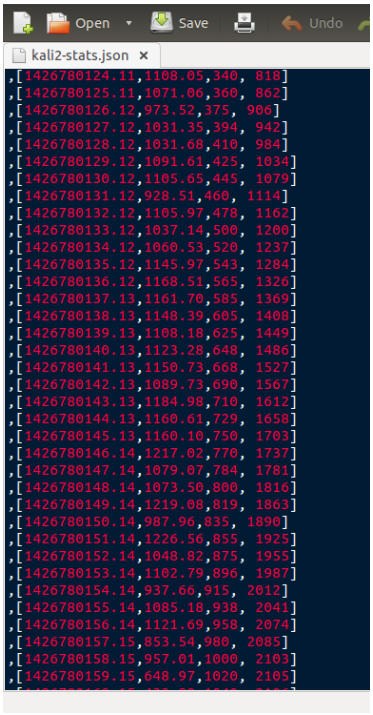
`python main.py 50000bytes.rar.torrent -b`
για να ενεργοποιήσουμε τον BAP2P μηχανισμό



Σχήμα 33: Εκκίνηση BAP2P μηχανισμού

Περιμένουμε να συνδεθούν οι υπολογιστές με τον tracker και ξεκινάμε το torrent από τον tracker / seeder για να ξεκινήσει ο διαμοιρασμός.

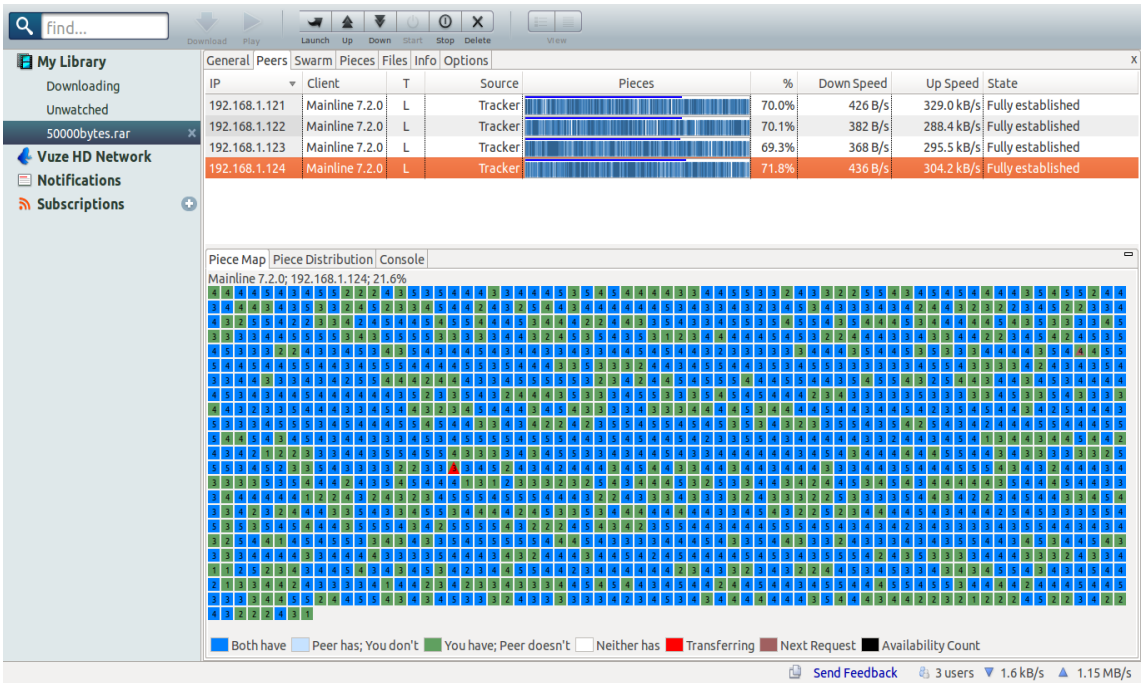
Εφόσον ολοκληρώσουν όλοι οι peer τη λήψη του αρχείου κλείνουμε τους μηχανισμούς και τα αποτελέσματα αποθηκεύονται σε ένα .json αρχείο στο root του μηχανισμού. Το .json αρχείο αποθηκεύει στατιστικά για κάθε χρονικό σημείο της λειτουργίας των μηχανισμών. Τα στατιστικά αυτά είναι το timestamp της στιγμής της λήψης μέτρησης το bandwidth, ο αριθμός των ληφθέντων blocks από το default bittorrent μηχανισμό και ο αριθμός των blocks που ελήφθησαν από τον αέρα.



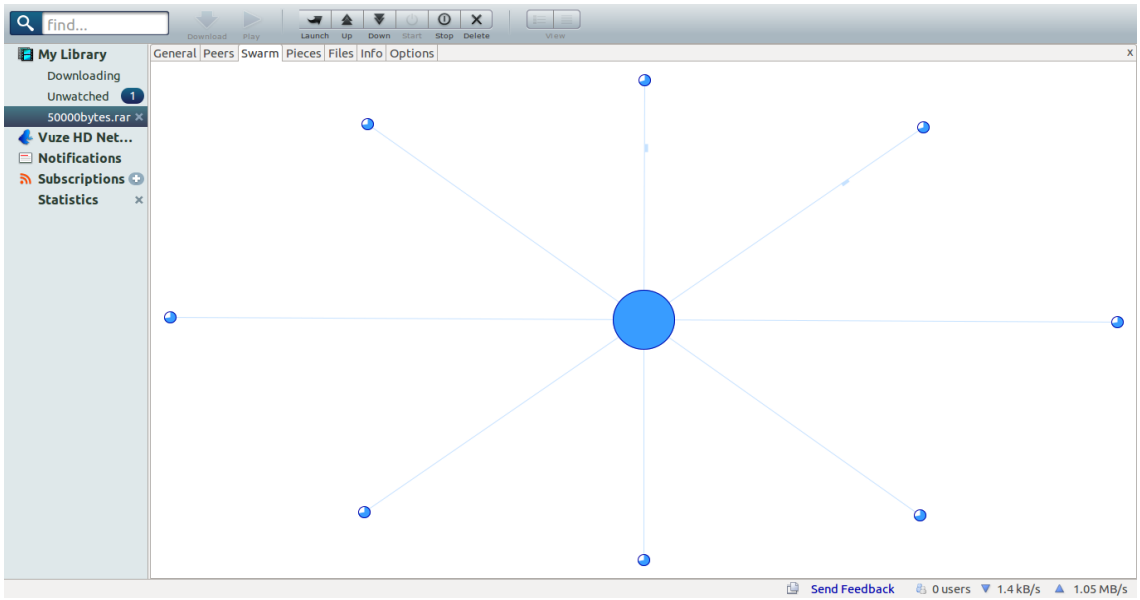
Σχήμα 34: Στιγμιότυπο εξαγωγής αποτελεσμάτων στο .json αρχείο

4.4 Αποτελέσματα μετρήσεων

Το πείραμα σχεδιάστηκε για να επαληθεύσει την ορθή χρήση του μηχανισμού BAP2P στα ασύρματα δίκτυα κάτω από πραγματικές συνθήκες. Για την αξιολόγηση του μηχανισμού πραγματοποιείται η σύγκρισή του με ένα τυπικό μηχανισμό Bittorrent.

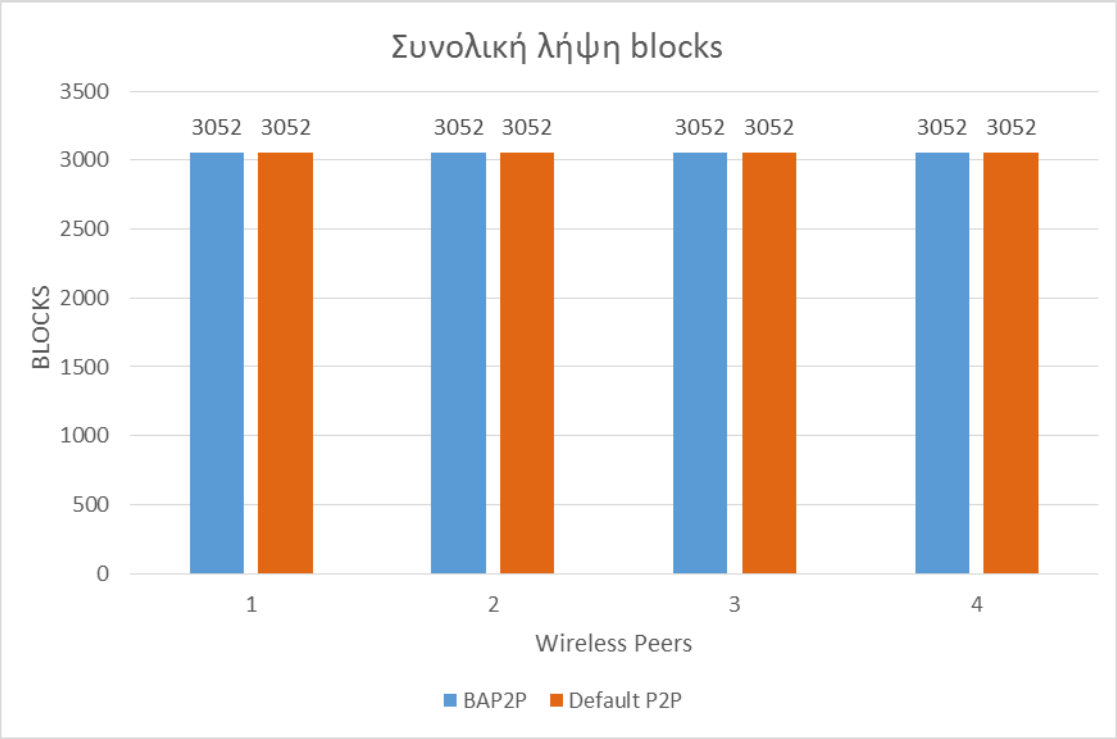


Σχήμα 35: Στιγμιότυπο διαμοιρασμού περιεχομένου



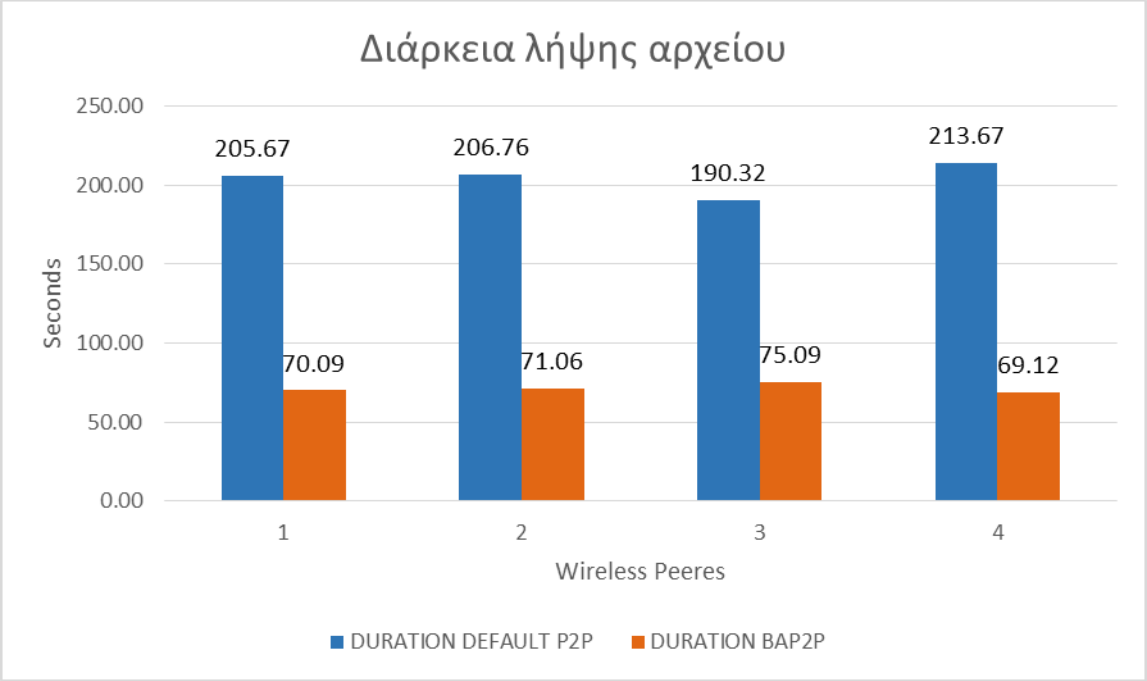
Σχήμα 36: Διαμόρφωση swarm κατά τη λήψη του αρχείου. Εμφανίζονται διπλάσιοι οι leechers καθώς σε κάθε peer τρέχουν ουσιαστικά δύο μηχανισμοί, ο τυπικός Bittorrent και ο BAP2P

Στο πρώτο σχήμα παρουσιάζεται ο αριθμός επιτυχούς λήψης των blocks του αρχείου με τον τυπικό Bittorrent μηχανισμό και με τον μηχανισμό BAP2P. Υπενθυμίζουμε εδώ ότι το αρχείο που διαμοιράζεται αποτελείται από 1526 pieces μεγέθους 32Kib. Το κάθε piece αποτελείται από 2 blocks των 16Kib, δηλαδή 3052 blocks για όλο το αρχείο. Με τη χρήση του τυπικού Bittorrent μηχανισμού παρατηρείται η επιτυχής λήψη όλων των blocks του αρχείου. Ο μηχανισμός BAP2P καταφέρνει επίσης να εξασφαλίσει τη σωστή λήψη όλων των blocks από τους peers χωρίς να υπάρχει καμία απώλεια.

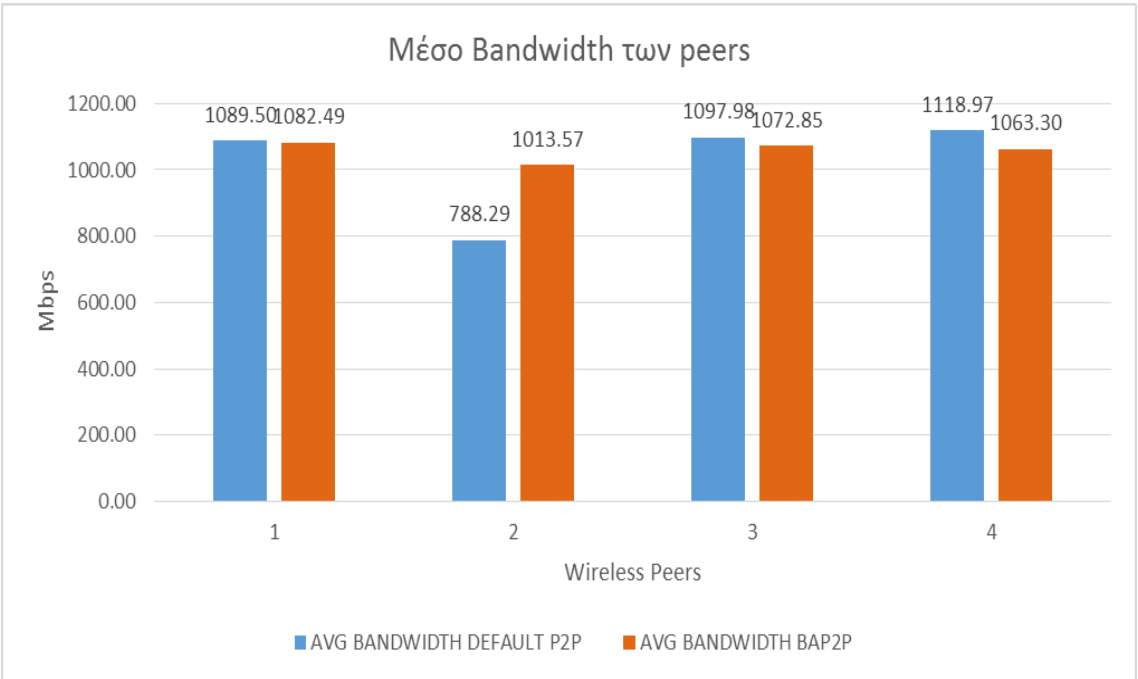


Σχήμα 37: Συνολική λήψη blocks πριν και μετά την ενεργοποίηση του μηχανισμού BAP2P

Στο επόμενο σχήμα βλέπουμε το χρόνο ολοκλήρωσης λήψης του αρχείου από τους 4 ασύρματους peers του δικτύου. Η χρήση του μηχανισμού BAP2P πέτυχε σημαντική μείωση του χρόνου ολοκλήρωσης στο 1/3 του τυπικού Bittorrent μηχανισμού.



Σχήμα 38: Διάρκεια λήψης αρχείου από τους peers

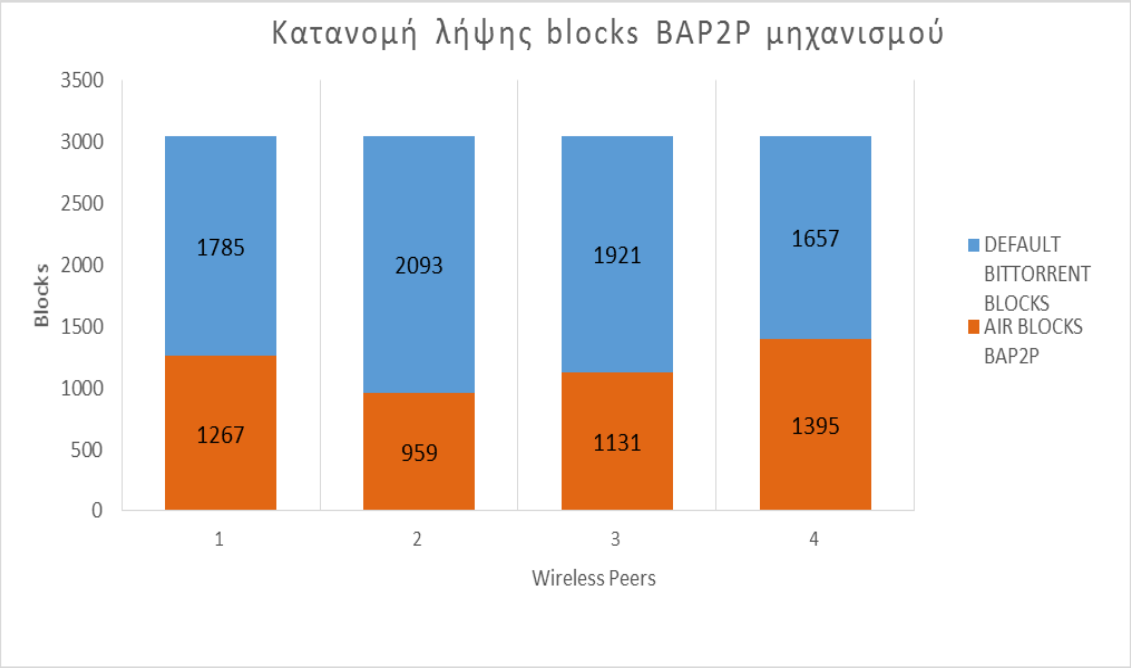


Σχήμα 39: Μέση χρήση bandwidth από τους peers

Στο παραπάνω σχήμα απεικονίζεται το μέσο bandwidth κάθε peer κατά τη διάρκεια της μεταφοράς του αρχείου στο δίκτυο. Οι τιμές κατά τη χρήση του τυπικού Bittorrent

μηχανισμού δείχνουν σχεδόν ίδιες με αυτές κατά τη χρήση του BAP2P μηχανισμού.

Στο επόμενο σχήμα παρατηρούμε τη κατανομή λήψης του αρχείου έχοντας ενεργοποιημένο το μηχανισμό BAP2P. Εδώ μπορούμε να διακρίνουμε ποιά blocks κατάφεραν οι peers να πάρουν από τον αέρα και ποιά μέσω του τυπικού Bittorrent μηχανισμού. Η λήψη του αρχείου ξεκινάει με τον seeder να μεταδίδει τα πρώτα πακέτα με τον τυπικό Bittorrent μηχανισμό και έπειτα να αναζητά ο κάθε peer τα block που δεν έχει από τον αέρα, ανιχνεύοντας τη λήψη τους από τους άλλους γειτονικούς peers. Αν το piece που ζητάει δεν το βρει στον αέρα, το κατεβάζει με το τυπικό Bittorrent μηχανισμό.



Σχήμα 40: Αριθμός blocks που ελήφθησαν από τον τυπικό Bittorrent μηχανισμό και από τον αέρα

Μετά το πέρας της εξαγωγής των αποτελεσμάτων της χρήσης του πρότυπου μηχανισμού BAP2P ως προς τον τυπικό Bittorrent μηχανισμό, μπορούμε να εξάγουμε τα εξής συμπεράσματα.

Ο μηχανισμός καταφέρνει να λειτουργεί αποτελεσματικά πάνω σε ένα ασύρματο δίκτυο. Επιτυγχάνει τους στόχους του καθώς διαμοιράζει το περιεχόμενο επιτυχώς και ταχύτερα

στους κόμβους του δικτύου. Δεν υπάρχει απώλεια δεδομένων του αρχείου, καθώς κατά τη χρήση του BAP2P μηχανισμού εναλλάσεται επιτυχώς με τον τυπικό Bittorrent μηχανισμό για την ολοκλήρωση της διανομής του αρχείου.

4.4. Μελλοντικές βελτιώσεις

Ο μηχανισμός BAP2P θα μπορούσε να δοκιμαστεί σε πολυπλοκότερα και ταχύτερα δίκτυα κινητών συσκευών όπως τα MANET και LTE δίκτυα.
Ο μηχανισμός της ασύρματης υλοποίησης του BAP2P μπορεί να μεταφερθεί επίσης σε ένα cloud σύστημα. Εκεί θα μπορούν να δημιουργηθούν περισσότεροι peers και να ερευνηθεί σε μεγαλύτερο εύρος η απόδοση του μηχανισμού

Βιβλιογραφία

1. Evangelos Markakis, Charalampos Skiannis, Anargiros Sideris, George Alexiou, Evangelos Palis, *A broadcast aware P2P mechanism for improving BitTorrent content delivery* (2014), Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)
2. Philippas Tsigas, Bittorrent: *A technical description of the Bittorent protocol*, URL: http://www.cse.chalmers.se/~tsigas/Courses/DCDSeminar/Files/BitTorrent.pdf
3. Evangelos Markakis, *Peer to peer constellations in broadcasting environments* (2014), Πανεπιστήμιο Αιγαίου Σχολή Θετικών Επιστημών. Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων, URL: http://phdtheses.ekt.gr/eadd/handle/10442/34935
4. Bram Cohen, *The bittorrent protocol specification* (2008), URL: http://www.bittorrent.org/beps/bep_0003.html
5. Ross Lee Graham, *International Conference on Peer-to-Peer Computing (2001),* URL http://www.ida.liu.se/conferences/p2p/p2p2001
6. Rodrigo Rodrigues, Peter Druschel *Peer-to-Peer Systems* Communications of the ACM, Vol. 53 No. 10, Pages 72-82 *(2010) URL: http://cacm.acm.org/magazines/2010/10/99498-peer-to-peer-systems/fulltext*
7. G. Camarillo, *Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability,* IAB (2009) URL: https://tools.ietf.org/html/rfc5694
8. Chibuike Muoh, *A Tutorial on Gnutella, Bittorrent and Freenet Protocols* (May 2006) URL: http://medianet.kent.edu/surveys/IAD06S-P2PArchitectures-chibuike/P2P%20App.%20Survey%20Paper.htm
9. Karthik Tamilmani, *The BitTorrent File Sharing Network - How does it work?,* URL: http://alexmohr.com/bittorrent/btworking.html
10. James F. Kurose, Keith W. Ross *Δικτύωση Υπολογιστών προσέγγιση από πάνω προς τα κάτω* (2008) 4η έκδοση, Εκδόσεις Μ. Γκιούρδας
11. Άρης Αλεξόπουλος, Γιώργος Λαλογιάννης *Τηλεπικοινωνίες και Δίκτυα Υπολογιστών* (2012), 8η έκδοση
12. Matthew S. Gast, 802.11 *Wireless Networks: The Definitive Guide* (2005), O'Reilly Media
13. Alan Holt, Chi-Yu Huang *802.11 Wireless Networks Security and Analysis* (2010), Springer-Verlag London Limited
14. Ian Poole, *IEEE 802.11g Wi-Fi Tutorial, URL: http://www.radio-electronics.com/info/wireless/wi-fi/ieee-802-11g.php*

15. Cisco Systems Inc., *Capacity, coverage and deployment considerations for IEEE 802.11g* (2009)
URL:https://www.cisco.com/application/pdf/en/us/guest/products/ps430/c1244/ccmigration_09186a00801d61a3.pdf

16. National Instruments, *WLAN - 802.11 a,b,g and n* (2013)
URL: http://www.ni.com/tutorial/7131/en/

17. Dimitris Vassis, George Kormentzas, Angelos Rouskas, Ilias Maglogiannis *The IEEE 802.11g Standard for High Data Rate WLANs,* University of the Aegean (2005)
URL: http://pamvotis.org/vassis/VKRM_Final.pdf

18. University of Ediburgh, 8*02.11 Wireless LANs aka WiFi*,
URL: http://downloads.specknet.org/DKACN/CN-MKM-Lect9.pdf

19. Humboldt University Berlin, *802.11 Network Structures*,
URL: https://sarwiki.informatik.hu-berlin.de/802.11_Network_Structures

20. *FORENSIC INSIGHT: DIGITAL FORENSICS COMMUNITY IN KOREA*,
URL: http://forensicinsight.org/wp-content/uploads/2013/02/F-INSIGHT-BitTorrent-Protocol.pdf?ckattempt=2

21. Autonomotorrent, URL: https://github.com/abhijeeth/AutonomoTorrent

22. Vuze, URL: http://www.vuze.com

23. Transmission, URL: http://www.transmissionbt.com/

24. XBT Tracker, URL: http://visigod.com/xbt-tracker

25. Kali Linux, URL: https://www.kali.org/

Παράρτημα Α

piecesniffer.py

```
import pcapy
import dpkt
import logging
from btcacher import PieceManager
from threading import Thread
import socket
import re
import binascii
from twisted.python import log
from iptools import get_ip_address
#from iptools import BAP2P_ENABLE
import stats
block_re =
re.compile('0000400907(?P<index>\w{8})(?P<offset>\w{8})')
my_ip = get_ip_address('wlan0')
BAP2P_ENABLE = False


class PieceSniffer(Thread):
    def __init__(self, dev='wlan0',
bap2p_enable=BAP2P_ENABLE):
        Thread.__init__(self)




        #self.expr = 'udp and dst port %s' % port
        self.expr = 'udp or tcp'
```

```
        #self.expr = 'tcp'
        self.maxlen = 65535  # max size of packet to
capture
        self.promiscuous = True  # promiscuous mode?
        self.read_timeout = 1000  # in milliseconds
        self.max_pkts = -1  # number of packets to capture;
-1 => no limit
        self.active = True
        self.p = pcapy.open_live(dev, self.maxlen,
self.promiscuous, self.read_timeout)
        self.p.setfilter(self.expr)
        log.msg('Piece Sniffer started on %s...' % dev)

        global BAP2P_ENABLE
        BAP2P_ENABLE = bap2p_enable
        if not BAP2P_ENABLE:
            log.msg('BAP2P Disabled.')
        else:
            log.msg('BAP2P Enabled.')
        #logging.info('Piece Sniffer started...')
        #logging.debug('Pcap Filter:  \"%s\"' % self.expr)


    # @staticmethod
    # def dvbt_block_stats(src, dst):
    #     target_src_subset = '172.16.'
    #     target_dst_subset = '172.16.%s0.' %
HOSTNAME_NUMBER
    #
    #     if src.startswith(target_src_subset) and
dst.startswith(target_dst_subset):
    #         if str(dst).endswith('.6'): stats.dvbt_in_cl1
```

```
+= 1
    #         elif str(dst).endswith('.14'):
stats.dvbt_in_cl2 += 1
    #         elif str(dst).endswith('.22'):
stats.dvbt_in_cl3 += 1
    #         elif str(dst).endswith('.30'):
stats.dvbt_in_cl4 += 1
    #         elif str(dst).endswith('.38'):
stats.dvbt_in_cl5 += 1


    @staticmethod
    def cb(hdr, data):
       global BAP2P_ENABLE
        if not data:
            return
        eth = dpkt.ethernet.Ethernet(str(data))
        #ip validation
        if eth.type != dpkt.ethernet.ETH_TYPE_IP:
            return




        ip = eth.data
        try:
            if ip.p != dpkt.ip.IP_PROTO_UDP and ip.p !=
dpkt.ip.IP_PROTO_TCP:
                return
        except:
            return
        #tcp udp validation
        #if ip.p != dpkt.ip.IP_PROTO_UDP and ip.p !=
```

```
dpkt.ip.IP_PROTO_TCP:
        # return
        src_ip = socket.inet_ntoa(ip.src)
        dst_ip = socket.inet_ntoa(ip.dst)
        if ip.p == dpkt.ip.IP_PROTO_TCP:
            tcp = ip.data
            try:
                pass
                stats.bitrate_current += len(tcp.data)
            except:
                return
            for m in
block_re.finditer(binascii.hexlify(tcp.data)):
                # stats.blocks_count += 1
                # if dst_ip != my_ip:

                #     stats.other_blocks_count +=1
                block = m.groups()
                block_index = int(block[0], 16)
                block_offset = int(block[1], 16)
                #if not BAP2P_ENABLE:
                ####################################
                # SENARIO 1 (DEFAULT P2P)
                ####################################
                    #log.msg('BAP2P Disabled.')
                    #if dst_ip == my_ip:
                    #    stats.blocks_count += 1
                     #
PieceManager.add_block(block_index, block_offset, '')
                #else:
```

```
                ###################################
                # SENARIO 2 (BAP2P)
                ###################################
                    #log.msg('BAP2P Enabled.')
                    #PieceManager.add_block(block_index,
block_offset, '')
                    #stats.blocks_count += 1
                if dst_ip == my_ip:
                    stats.blocks_count += 1
                else:
                    if not
PieceManager.do_i_have_block(block_index, block_offset):

                        stats.other_blocks_count +=1
                PieceManager.add_block(block_index,
block_offset, '')
                #log.msg('block %s %s' % (block_index,
block_offset))
                #print 'block %s %s' %  (block_index,
block_offset)
                #PieceSniffer.dvbt_block_stats(src_ip,
dst_ip)
                PieceManager.print_stats()
                #PieceManager.print_stats()


    def stop(self):
        log.msg('Piece Sniffer stopped...')
        #logging.info('Piece Sniffer stopped...')
        self.active = False
    def run(self):
        while self.active:
```

```
self.p.dispatch(0, PieceSniffer.cb)
```

FileManager.py

```
"""
"""
import os
import hashlib
from twisted.python import log
from twisted.internet import reactor, defer
from bitfield import Bitfield
from tools import sleep
from btcacher import PieceManager


class BTFileError (Exception) :
    pass
class BTHashTestError (Exception):
    pass
class BTFile:
    def __init__(self, metainfo, index, saveDir):
        fileinfo = metainfo.files[index]
        piece_len = metainfo.piece_length
        self.fileInfo = fileinfo
        self.path = os.path.join(saveDir, fileinfo['path'])
        self.length = fileinfo['length']
        self.piece_len = piece_len
        self.abs_pos0, self.abs_pos1 =
fileinfo['pos_range']
        self.fd = None
        idx0, ext = divmod(self.abs_pos0, self.piece_len)
        self.idx0_piece = idx0
```

```
        idx1, ext = divmod(self.abs_pos1, self.piece_len)
        self.idx1_piece = idx1+1 if ext else idx1
        h, t = os.path.split(self.path)
        if not os.path.exists(h):
            os.makedirs(h)

    def __str__(self):
        return u'piece=[{},{}) size={:,d} "{}"
'.format(self.idx0_piece, self.idx1_piece, self.length,
os.path.split(self.path)[1]).encode('gb2312')

    def __getIntersection(self, index, beg, data_len):
        # p0,p1,f0,f1 absolute position in files
        p0 = index * self.piece_len + beg
        p1 = p0 + data_len
        f0, f1 = self.abs_pos0, self.abs_pos1
        # intersect sub piece
        pf0 = max(p0, f0)
        pf1 = min(p1, f1)
        # pb,pe relative positioin in piece
        pb = pf0 - p0
        pe = pf1 - p0
        # fb,fe relative position in current file
        fb = pf0 - f0
        fe = pf1 - f0
        return (pb, pe), (fb, fe)

    def write(self, index, beg, data):
        (pb,pe), (fb,fe) = self.__getIntersection(index,
beg, len(data))
        if pb >= pe :
```

```
            raise BTFileError("index isn't in this file")

        my_data = data[pb:pe]
        if self.fd is None :
            if os.path.exists(self.path) :
                length = os.path.getsize(self.path)
                if length != self.length:
                    raise BTFileError(u'old file size is
error: {}'.format(self.path))
                fd = open(self.path, 'rb+')
            else :
                fd = open(self.path, 'wb+')
                fd.truncate(self.length)
            self.fd = fd
        self.fd.seek(fb)
        self.fd.write(my_data)
        return pb, len(my_data)

    def read(self, index, beg, data_len):
        (pb,pe), (fb,fe) = self.__getIntersection(index,
beg, data_len)
        if pb >= pe :
            raise BTFileError("index isn't in this file")
        if self.fd is None:
            try:
                self.fd = open(self.path, 'rb+')
            except IOError as error:
                raise BTFileError(str(error))
        self.fd.seek(fb)
        data = self.fd.read(fe-fb)
```

```
        return pb, data



    def close(self):
        if self.fd :
            self.fd.close()
    def __getitem__(self, idx):
        return self.read(idx, 0, self.piece_len)
    def __setitem__(self, idx, data):
        self.write(idx, 0, data)
    def __iter__(self) :
        for idx in xrange(self.idx0_piece, self.idx1_piece)
:
            yield idx, self[idx]
    def __len__(self) :
        return self.idx1_piece - self.idx0_piece
    def __contains__(self, idx) :
        return self.idx0_piece <= idx < self.idx1_piece


class BTFiles :
    def __init__(self, metainfo, saveDir,
selectedFileIndex=None):
        if selectedFileIndex is None :
            selectedFileIndex = range(len(metainfo.files))
        selectedFileIndex.sort()
        self.metainfo = metainfo
        self.saveDir = saveDir
        self.totalSize = metainfo.total_length
        self.pieceNum = metainfo.pieces_size
```

```
        self.pieceLength = metainfo.piece_length
        self.hashArray = metainfo.pieces_hash



        self.files = []
        for i in selectedFileIndex :
            self.files.append(BTFile(metainfo, i, saveDir))

    def doHashTest(self, idx, data):
        return hashlib.sha1(data).digest() ==
self.hashArray[idx]
    # def getBitfield(self) :
    #     bfNeed = Bitfield(self.pieceNum)
    #     for f in self.files :
    #         for i in xrange(f.idx0_piece, f.idx1_piece) :
    #             bfNeed[i] = 1
    #
    #     bfHave = Bitfield(self.pieceNum)
    #     for i in xrange(self.pieceNum):
    #         try :
    #             ds = self[i]
    #             if len(ds) == 1:
    #                 beg, dat = ds[0]
    #                 if self.doHashTest(i, dat):
    #                     bfHave[i] = 1
    #                     bfNeed[i] = 0
    #         except BTFileError as error :
    #             pass
    #
```

```
#     return bfHave, bfNeed



def getBitfield(self) :
    bfNeed = Bitfield(self.pieceNum)
    for f in self.files :
        for i in xrange(f.idx0_piece, f.idx1_piece) :
            bfNeed[i] = 1

    bfHave = Bitfield(self.pieceNum)
    for i in xrange(self.pieceNum):
        try :
            ds = self[i]
            if len(ds) == 1:
                beg, dat = ds[0]
                if self.doHashTest(i, dat):
                    if PieceManager.do_i_have(i):
                         bfHave[i] = 1
                         bfNeed[i] = 0
                    else:
                         bfHave[i] = 0
                         bfNeed[i] = 1
                # if self.doHashTest(i, dat):
                #     bfHave[i] = 1
                #     bfNeed[i] = 0
        except BTFileError as error :
            pass
    #custom need
    #for x in range(0, len(bfNeed)):
```

```
        #    bfNeed[x] = 1
        return bfHave, bfNeed



    def write(self, idx, data) :
        ds = [f.write(idx,0,data) for f in self.files if
idx in f]
        if len(ds) <= 1 :
            return ds
        else
            _ds = ds[0:1]
            for d in ds[1:] :
                beg0, len0 = _ds[-1]
                beg1, len1 = d
                assert beg0+len0 <= beg1
                if beg0+len0==beg1:
                    _ds[-1] = beg0, len0+len1
                else:
                    _ds.append(d)
            return _ds


    def __getitem__(self, idx) :
        ds = []
        for f in self.files:
            if idx in f:
                try:
                    ds.append(f[idx])
                except BTFileError as error:
                    pass
```

```
    if len(ds) <=1 :
        return ds
    else :
        _ds = ds[0:1]
        for d in ds[1:] :
            beg0, dat0 = _ds[-1]
            beg1, dat1 = d
            assert beg0+len(dat0) <= beg1
            if beg0+len(dat0)==beg1:
                _ds[-1] = beg0, dat0+dat1
            else:
                _ds.append(d)
        return _ds

def __setitem__(self, idx, data) :
    for f in self.files:
        if idx in f :
            f[idx] = data


def __iter__(self):
    for idx in xrange(len(self)) :
        yield idx, self[idx]

def __contains__(self, idx) :
    return any(idx in f for f in self.files)
```

```
    def __len__(self):
        return self.pieceNum

    def __str__(self):
        return '\n'.join(str(f) for f in self.files)

class BTFileManager :
    '''
    '''
    slice_size = 2**14
    def __init__(self, btm):
        self.btm = btm
        self.config = btm.config
        metainfo = self.config.metainfo
        self.download_list = self.config.downloadList
        self.metainfo = metainfo
        self.piece_length = metainfo.piece_length
        self.pieceNum = metainfo.pieces_size
        self.btfiles = BTFiles(metainfo,
self.btm.app.save_dir, self.config.downloadList)
        self.bitfieldHave, self.bitfieldNeed =
self.btfiles.getBitfield()
        log.msg("Saving to:
{0}".format(self.btm.app.save_dir))
        self.buffer_reserved = {}
        self.buffer_max_size = 100 * 2**20 /
self.piece_length

    def __update_bitfield(self):
```

```
        self.bitfieldHave, self.bitfieldNeed =
self.btfiles.getBitfield()
        reactor.callLater(0.01, self.__update_bitfield)

        #print self.bitfieldHave

    def start(self) :
        self.status = 'started'
        self.buffer = {}
        self.buffer_record = []
        self.buffer_dirty = {}
        reactor.callLater(10, self.deamon_write)
        reactor.callLater(10, self.deamon_read)
        reactor.callLater(0.01, self.__update_bitfield)

    def stop(self) :
        for idx, data in self.buffer_dirty.iteritems():
            self.write(idx, data)
        self.buffer_dirty.clear()
        self.buffer.clear()
        del self.buffer_record[:]
        self.status = 'stopped'

    @defer.inlineCallbacks

    def deamon_write(self):
        while self.status == 'started':
            self.__thread_write()
            yield sleep(10)
```

```
def __thread_write(self):

    if not hasattr(self, '__thread_write_status') :
        self.__thread_write_status = 'stopped'
    if self.__thread_write_status == 'running' :
        return
    if not self.buffer_dirty :
        return

    bfd = self.buffer_dirty.copy()

    def call_in_thread():
        # Writing to disk
        for idx in sorted(bfd.keys()) :
            data = bfd[idx]
            self.write(idx, data)
        reactor.callFromThread(call_from_thread)

    def call_from_thread():
        self.__thread_write_status = 'stopped'
        for idx, data in bfd.iteritems() :
            if data is self.buffer_dirty[idx] :
                del self.buffer_dirty[idx]

    if self.__thread_write_status == 'stopped' :
        self.__thread_write_status = 'running'
        reactor.callInThread(call_in_thread)
```

```
    @defer.inlineCallbacks

    def deamon_read(self):
        while self.status == 'started':
            size = len(self.buffer)
            if size > self.buffer_max_size :
                remove_count = size - self.buffer_max_size
                remove_count += self.buffer_max_size / 5
                for idx in
self.buffer_record[:remove_count] :
                    del self.buffer[idx]
                del self.buffer_record[:remove_count]
            yield sleep(10)



    def readPiece(self, index) :
        if not (0 <= index < self.pieceNum) :
            raise BTFileError('index is out of range')
        if not self.bitfieldHave[index] :
            raise BTFileError('index is not downloaded')

        if index in self.buffer :
            data = self.buffer[index]
            self.buffer_record.remove(index)
            self.buffer_record.append(index)
            return data
        else:
            for idx in [index-1, index, index+1] :
                if 0 <= idx < self.pieceNum and idx not in
```

```
self.buffer :
                    data = self.read(idx)

                    assert data
                    self.buffer[idx] = data
                    self.buffer_record.append(idx)
            data = self.readPiece(index)
            return data



    def writePiece(self, index, piece) :
        if not (0 <= index < self.pieceNum) :
            raise BTFileError('index is out of range')
        if not self.bitfieldNeed[index] :
            raise BTFileError('index is not need')
        if not self.btfiles.doHashTest(index, piece):
            raise BTHashTestError()
        else:
            self.bitfieldHave[index] = 1
            self.bitfieldNeed[index] = 0
            if index in self.buffer :
                self.buffer[index] = piece

            self.buffer_dirty[index] = piece
            return True

    def read(self, index):
        if index in self.buffer_dirty:
            return self.buffer_dirty[index]
```

```
    elif index in self.buffer_reserved :

        return self.buffer_reserved[index]
    data_list = self.btfiles[index]

    if len(data_list) == 1 :
        assert data_list[0][0] == 0
        return data_list[0][1]
    else:
        assert False
        return data_list



def write(self, index, data) :
    ds = self.btfiles.write(index, data)
    if len(ds) > 1 :
        self.buffer_reserved[index] = data
    elif not ds :
        assert False

def __iter__(self):
    return self.btfiles.__iter__()
```

PieceManager.py

```
"""
"""
import hashlib
import random
from twisted.python import log
from bitfield import Bitfield
from twisted.internet import reactor
from FileManager import BTFileManager, BTFileError,
BTHashTestError
from btcacher import PieceManager

class BTPieceManager:
    slice_size = 2**14
    def __init__(self, btm):
        self.btm = btm
        self.metainfo = btm.metainfo
        self.connectionManager = btm.connectionManager
        self.btfiles = BTFileManager(btm)
        self.bitfield = self.btfiles.bitfieldHave
        metainfo = self.metainfo
        self.piece_length = metainfo.piece_length
        self.pieces_size = metainfo.pieces_size
        self.pieces_hash = metainfo.pieces_hash
        self.buffer = {}
        self.bfNeed = self.btfiles.bitfieldNeed
        self.pieceDownload = {} # [idx]: [todo], [doing],
[done]
```

```
        self.pieceTodo = {}
        self.pieceDoing = {}
        self.pieceDone = {}

    def __update_bitfield(self):
        self.bitfield = self.btfiles.bitfieldHave
        self.bfNeed = self.btfiles.bitfieldNeed
        reactor.callLater(0.01, self.__update_bitfield)
        # print self.bitfield

    def start(self) :
        self.btfiles.start()
        reactor.callLater(0.01, self.__update_bitfield)

    def stop(self) :
        self.btfiles.stop()

    def do_slice(self, beg, end):
        slice_list = []
        r = range(beg, end, self.slice_size)
        for beg in r[:-1] :
            slice_list.append((beg, self.slice_size))
        slice_list.append((r[-1], end-r[-1]))
        return slice_list

    def __getPieceSlice(self, idx):
        if idx == self.pieces_size-1:
            return self.do_slice(0,
self.metainfo.last_piece_length)
        else:
```

```
        return self.do_slice(0, self.piece_length)

def amInterested(self, idx):
    if type(idx) is Bitfield:
        try:
            for i in (self.bfNeed & idx):
                return True
            else:
                return False
        except TypeError:
            return False
    else:
        return idx in self.bfNeed

def doIHave(self, index):
    return self.bitfield[index]



def getMorePieceTask(self, peer_bf, num_task=5):
    num_task=5
    if num_task == 0:
        return None
    tasks = []
    #print peer_bf
    #print self.bfNeed

    # random_idx = random.randrange(len(peer_bf))
    # if self.bfNeed[random_idx] == 0:
    #     for x in range(random_idx, len(peer_bf)):
    #         if self.bfNeed[x] == 1:
```

```
#             random_idx = x
#             break
#
# if self.bfNeed[random_idx] == 0:
#     for x in range(0, len(peer_bf)):
#         if self.bfNeed[x] == 1:
#             random_idx = x
#             break
#

#print "random >>: %s" % random_idx

#for idx in (peer_bf & self.bfNeed) :
#idx = random.randrange(len(peer_bf))
while not self.torIsCompleted():
    idx = random.randrange(len(peer_bf)-1)
    if PieceManager.do_i_have(idx):
        for x in range(idx, len(peer_bf)-1):
            if not PieceManager.do_i_have(x):
                idx = x
                break

    if PieceManager.do_i_have(idx):
        for x in range(0, len(peer_bf)-1):
            if not PieceManager.do_i_have(x):
                idx = x
                break

#for idx in (peer_bf & self.bfNeed) :
    #print "REAL >> %s - %s" % (idx,
```

```
PieceManager.do_i_have(idx))
            #print self.bfNeed

            while True:
                task = self.getPieceTask(idx)
                if not task :
                    break
                tasks.append(task)
                if len(tasks) == num_task:
                    return tasks

    def torIsCompleted(self):
        for x in range(0, len(self.bfNeed)-1):
            if not PieceManager.do_i_have(x):
                return False
        return True

    def getPieceTask(self, idx):
        assert idx in self.bfNeed
        if idx not in self.pieceDownload:
            slice_list = self.__getPieceSlice(idx)
            self.pieceDownload[idx] = [slice_list, [], []]

        task_to_do, task_doing, task_done =
self.pieceDownload[idx]
        if not task_to_do:
            return None
        my_task = task_to_do[0]
        del task_to_do[0]
        task_doing.append(my_task)
```

```
        return idx, my_task


    def failedPieceTask(self, idx, task):
        task_to_do, task_doing, task_done =
self.pieceDownload[idx]
        assert task in task_doing
        task_doing.remove(task)
        task_to_do.append(task)


    def finishPieceTask(self, idx, task, data):
        task_to_do, task_doing, task_done =
self.pieceDownload[idx]
        assert task in task_doing
        task_doing.remove(task)
        task_done.append((task, data))

        if not task_to_do and not task_doing :
            task_done.sort(key=lambda x : x[0][0])
            data = ''.join(d for t, d in task_done)

            try:
                pass
                #self.btfiles.writePiece(idx, data)
                #self.bitfield[idx] = 1
                #self.bfNeed[idx] = 0

            except BTHashTestError as error:
                # sha1 error ~ corrupt piece
                del self.pieceDownload[idx]
                if idx == self.pieces_size-1:
```

```
                self.do_slice_tail()

        else:
            self.connectionManager.broadcastHave(idx)

def getPieceData(self, index, beg, length) :
    if not self.doIHave(index) :
        return None
    piece = self.btfiles.readPiece(index)
    if piece :
        return piece[beg:(beg+length)]
    else :
        return None
```