INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE

Advanced Computer Network Project
REPORT
ON

Implementation of BitTorrent

(Peer-to-Peer Networking)

Submitted To:
Dr. P. Sateesh Kumar
Asst. Prof.
Computer Science & Engineering

Submitted By:
Bhupender Singh (18535009)
Jitendra Giri (18535013)
Suresh Togas (18535025)
Tulesh Jaiswal (18535027)
Yogita (18535030)

Table of contents

1 Introduction:

Today, Peer-to-Peer solution is more used for the file distribution. Peer-to-Peer solution provides more advantages as compared to client-server solution. The advantages are :

a) Increased Robustness

b) Resource Providing

i.Bandwidth

ii.Storage Space

iii.Computing Power

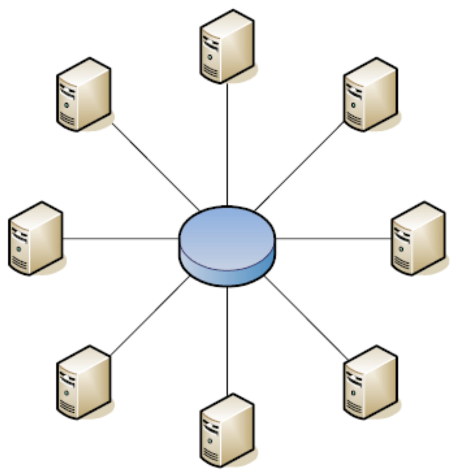
When we are distributing file, especially large one then Robustness and Utilization are important applications. One important Peer-to-Peer solution is BitTorrent which is efficient and reliable.

Here we will discuss first the concept of Peer-to-Peer and after then switch on BitTorrent.

2 Peer-to-Peer Networking :

It is a communication model in which each party have same capability and anyone can initiates the session.

Alternative of client-server solution is Peer-to-Peer which contains a single server and many clients.



Traditional Peer-To-Peer Network Examples:

i.ARPAPET

ii.DNS

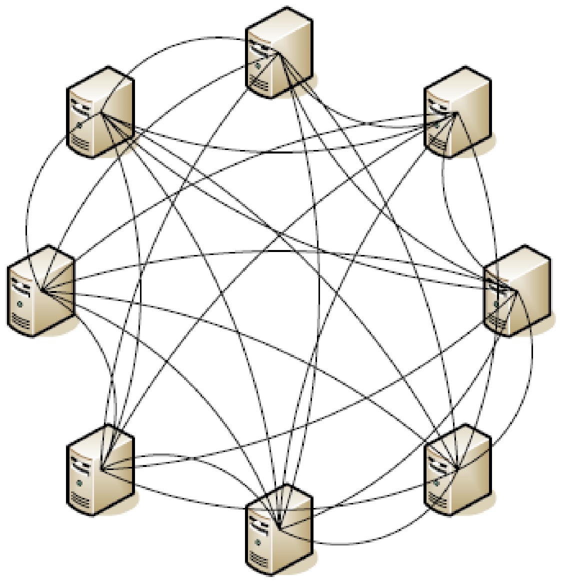
“A distributed network architecture may be called a Peer-to-Peer network, if the participants share a part of their own resources (processing power, storage capacity, network link capacity, printers). These shared resources are necessary to provide the Service and content offered by the network. They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource provider and resource requester.

The Peer-to-Peer network may be pure without having the central entity or can be with central entity.

2.1 Peer-To-Peer Network Topologies:

Pure Peer-to-Peer:

“A distributed network architecture has to be classified as a “Pure” Peer-to-Peer network if any single, arbitrary chosen Terminal Entity can be removed from the network without having the network suffering any loss of network service.”



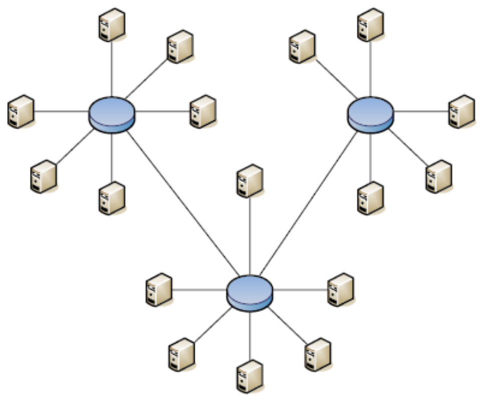
No single point of failure. The peer can be removed, without this having any fatal consequences**.**

Hybrid Peer-to-Peer:

Hybrid network will always include central entity.

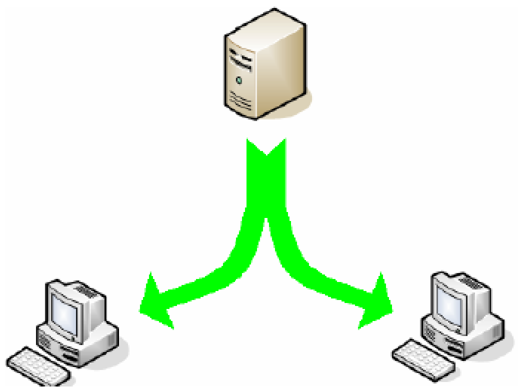
“A distributed network architecture has to be classified as a “Hybrid” Peer-to-Peer network, if central entity is necessary to provide parts of the offered network services.”

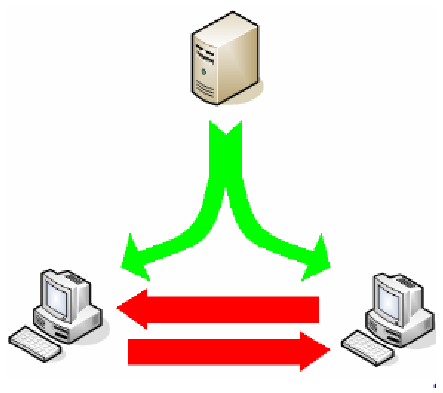
If central hub goes down then some parts of the network suddenly seprated. This makes the Hybrid peer-to-peer network more vulnerable to attacks or failure.



3. Introduction to BitTorrent

It is a technology which makes the distribution of large file simple and less bandwidth consuming. This can be done by utilizing the upload capacity of peersthat are downloading the file.





Shows the Client-Server approach to Download

- Peers download from server simultaneous.
- If upload capacity of server = download capacity of peer
- then download time = 2 times the time if only part one peer is downloading

Shows similar to BitTorrent

- Split the file and send one part to each pear.
- Peers download the missing from each other

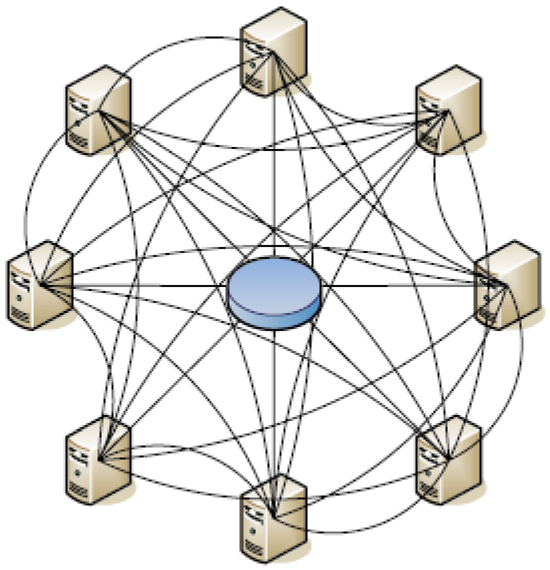
Thus the download and load time on server is reduced. Thus BitTorrent is very useful then the Client-Server.

4. The BitTorrent Architecture

The architecture normally consists of the followiing entities:

a. A torrent file (static metainfo file)
b. A tracker

c. An original downloader (seed)
d. End user downloader (leecher)



BitTorrent in its original form matches the “hybrid” peer-to-peer concept. It’s all about the torrent file, the centralized tracker and the associated swarm of peers. The centralized tracker provides the different entities with an address list over the available peers. These peers will then contact each other to download pieces of the file from each other.

The file using bitTorrent is to first create the metainfo file ie. the "Torrent file". Torrent file consists of

1. Filename
2. Size
3. Hashing information
4. URL of Tracker

- Torrent is created by the free program
- To download or “seed” a file, a BitTorrent client is required.
- Tracker keeps a track of peers who are downloading
- Tracker doesn't have a copy of file.
- Tracker and downloading user exchange information using a simple protocol on the top of HTTP .
- First, User contacts to user and tell about which file it is downloading, port it's listening.
-Tracker gives response to user with a list of other users who are downloading the same file and how to contact them.
- The group of peers that are sharing the same torrent represents a **"swarm"**.
- The original downloader is known as **"Seed"**. Seed is a user that have the complete file.

- The downloading user that have some parts of the file or none are known as **"Leecher"**.
- "Seed" must have to upload atleast one entine copy and stops uploading when a complete file is distributed among other peers.

The original file is cut into pieces of 512KB or 256KB then we can make a torrent file. The SHA-1 hash codes of the pieces are included in the torrent file. Compute SHA-1 hash codes and comparing it to the SHA-1 code of the corresponding piece in the torrent file. It guarantees to the users that they are downloading the real thing.

4.1 Algorithms :

Swapping of file happens one-to-one in many pear-to-peer file sharing protocol. But BitTorrent is able to download from many peers simultaneously. The main goal is to receive complete file as quick as possible. We have to know which peer to download what piece of the file.
Choosing peers to connect to is a two-sided problem. First, we need a way of finding the best sequence of downloading the pieces. This is determined by the piece selection algorithm. But this is not enough. A peer, who has the piece you want, might not let you download it. Strategies for peers not allowing other peers to download from them is known as choking, and concerns resource allocation. These two concepts are examined in the following pages.

4.1.1 The Piece Selection Algorithm

BitTorrent have to select the best sequence of pieces to increase the performance.Never end up with the situation where every peer has all the pieces that are currently available and none of the missing ones.

If the seeder leaves from the network then the user should be able to download from different peers. Thus the main goal is to replicate different pieces on different peers.There are several policies which combined make up the piece selection algorithm.

Sub-Pieces

BitTorrent uses TCP Thus will always transfer data or the transfer rate to reduce because of slow start. The pieces will be broken up into sub-pieces of 16Kb . It always have a typically request of five sub-pieces. A requested sub-piece can be download from many peers.

Policy 1:Strict Policy

When a sub-piece is requested, the remaining sub-pieces for that particular sub-pieces are requested first rather to request another piece. It leads to get a complete piece as quick as possible.

Policy 2: Rarest First

The considered main policy is "**Rarest First**". When a peer selects the next piece to download, it selects the piece which the fewest of their peers have. Considered as good policy because of several reasons:

Spreading the seed:
Rarest first makes sure that only "new" pieces are downloaded from the seed.

Increased download speed:
We have to download rare pieces from the seed so that download speed can be improved.

Enabling uploading:
A rare piece is most wanted by other peers, and by getting a rare piece others will be interested in uploading from you.

Most common last:
It is sensible to leave the most common pieces to the end of the download. Since many peers have it, the probability of being able to download them later is much larger than that of the rare pieces.

Prevent rarest piece missing:
When seed is down at that time the entire pieces must be present within peers thus it will reduce the risk of missing one.

Policy 3: Random First Piece

Once you start downloading, you don't have anything to upload. It is important to get the first piece as fast as possible, and this means that the "rarest first"-policy is not the most efficient. Rare pieces tend to be downloaded slower, because you can download it's sub-pieces from only one (or maybe a few) other peers. As mentioned earlier, multiple peers with the same piece increase the download speed.

Policy 4: Endgame mode

Sometimes a piece might be downloaded from a peer with a slow transfer rate. This can potentially delay the finishing of a download. To prevent this we have the "endgame mode". Remember the pipelining principle, which ensures that we always have a number of requests (for sub-pieces) pending, the number often being set to five. When all the sub-pieces a peer lacks are requested, this request is broadcasted to all peers. This helps us to get the last chunk of the file as quickly as possible.

4.1.2 Resource allocation

No centralized resource allocation exists in BitTorrent. Every peer is responsible for maximizing its download rate. A peer will, naturally, try to download from whoever they can. To decide which peers to upload to, a peer uses a variant of the “tit-for-tat” algorithm**.**

Choking Mechanism

One of BitTorrent’s most powerful idea is the choking mechanism which ensures that nodes cooperate and eliminates the free-rider problem. Choking is a temporary refusal to upload. Each BitTorrent peer always unchokes a fixed number of other peers, so the issue becomes which peers to unchoke.

Optimistic Unchoking

Simply uploading to the peers which provide the best download rate would suffer from having no method of discovering if currently unused connections are better than the ones being used.To fix this, a BitTorrent peer has a single ‘optimistic unchoke’, a peer which is unchoked regardless of the current download rate from it.

Anti-snubbing

When over a minute goes by without getting a single sub-piece from a particular peer, do not upload to it except as an optimistic unchoke. If choked by everyone, increase the number of simultaneous optimistic unchokes to more than one (an exception to the exactly one optimistic unchokerule mentioned earlier), which causes download rates to recover much more quickly when they falter.

Upload Only

when a download is completed we use a new choking algorithm which unchokes the peers with the highest upload rate. This ensures that pieces get uploaded faster and that they get replicated faster. Peers with good upload rates are also probably not being served by others.

4.2 Important improvements

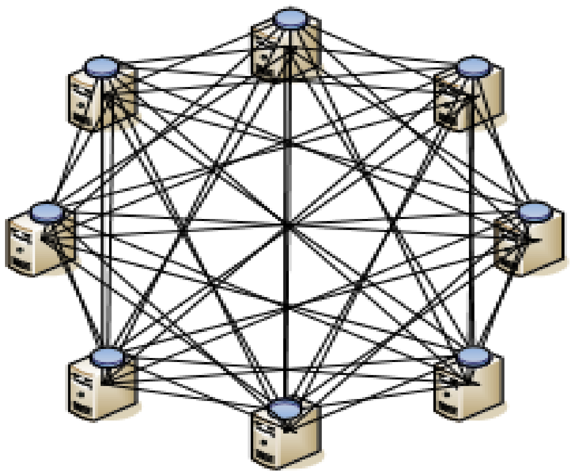
4.2.1 Bulk traffic marking

Bulk traffic marking make it more business-friendly. It is use to shape it traffic so that it is easier to manage it.

Until this improvement the large amount of BitTorrent traffic could easily have great adverse impacts on real-time traffic. The huge amount of data traffic slows down the network. When the BitTorrent file transfers are marked as bulk, almost any standard traffic shaping tool can be used to manage the network traffic.

4.2.2 Decentralized tracker

In a centralized tracker ,the tracker and nodes are like the star topology and tracker is in the middle if tracker goes down then the file will no longer be available because then no way to communicate to each other that's why centralized tracker the Bit Torrent network is not very fault tolerant. One centralized tracker also makes the network vulnerable to denial of service attacks. In May 2005 Cohen release a version 4.1 and the big newsflash was support for a decentralized tracker. In this new version, all have to do is to circulate .torrent-file on a webpage, blog etc., and no tracker arrangement is necessary. Use of a centralized tracker is still supported, and the publisher can choose the mode on its choice.



BitTorrent with a decentralized tracker structure

The tracker is now act like a lightweight tracker because it no more centralized ,and now it is occupied by each and every node .But main problem is how it can possible ,how nodes communicate to each other, what's the criteria and protocol used in communication . The solution is based on distributed hash tables (DHTs). This makes it possible to share files with minimal resources, but with respect to reliability it can't guarantee. Less amount of money is used in decentralized tracker as compared to centralized. Also by using the tracker there is the problem of copyright issues which are mainly happen in TVs and others entertaining rights.By removing the tracker from the web server, the only traffic generated will be the fetching of the .torrent file. Now tracker no longer used to locate users so large amount of bandwidth is also saved .The trackerless version is still in its childhood and the majority of BitTorrent file sharing will probably involve a traditional tracker for quite a while.

Distributed hash tables (DHTs) :

A DHT is a decentralized distributed system. It is used for making highly structured topologies. The system consists of a set of participating nodes and a set of keys. In DHT if a correct key is provided a value is searched for and the combine form of value-key pair is stored in hash tables

,in decentralized system all nodes are free to join and leaving the network . What separates a DHT from an ordinary hash table is that the storage and lookups are distributed among the nodes (machines) in the network. DHT guarantee that if an nodes fails to join then it should not affect the reliability of the BitTorrent . Any DHT protocol emphasizes the following characteristics;

- Decentralization: the system of nodes maintained by themselves without any central coordination.
- Scalability: the system performs efficiently even with thousands or millions of nodes
- Fault tolerance: the system should not be crashed or stop working if nodes continuously joining, leaving or failing.

The DHT structure consists of two parts:

- keyspace partitioning
- Overlay network.

The keyspace partitioning handles the separating of keys among the nodes in the network. The overlay network connects nodes and let them to find the owner of a key. There are many DHT protocols in use. Chord, Pastry, Tapestry and Kademlia are some of them which use use different DHT in their own. These protocols are more similar than they are different. It means if you change some tweaks in some protocol you get another protocol ,we can conclude that they are not much different and all are appropriate into a multidimensional matrix. The Kademlia protocol can be seen as an evolved version of Chord. The Kademlia protocol is used as the BitTorrent client and in the Azureus . There are also several other network that use these protocol.

5. Implementation

Part 1- creation of torrent file at client

- We will develop our own format of .torrent file i.e. ".mtorrent"
- It contains following fields:
    - Tracker UR L1: URL of tracker server1
    - Tracker UR L2: URL of tracker server2
    - File name
    - File size
    - Hash string: This is concatenation of SHA1 hashes of each piece of file
- SHA1 hash of files: we divide file into logical pieces

- Program will create a <filename>.mtorrent file whenever it shares a file on the tracker (signifying that it is a seeder for that file.

Part 2- Torrent tracker

- Tracker will keep track of all the available peer on which the files or its parts reside.

- Implementation of tracker

- Start tracker using:

  <my_tracker_ip>:<my_tracker_port>
  <other_tracker_ip><other_tracker_port>   <seederlist_file><log_file>

  - Seeder list file: list of all available seeder.
  - Both tracker starts separately using above command.

- Tracker handles multiple requests.
- Have "share" functionality for clients via which they notify tracker of existence of file. Client shares file having filename, hashstring of pieces of file, client IP and listen port.

- Have a "seederlist" for client when a file is requested for downloading. This will return list of all available seeders.

- Seeder information should be updated if tracker goes down as soon as back up.

- Sharing a local file:

  - Command: share <localfilepath> <filename>.mtorrent.
  - Creation of .mtorrent file according to above specification.
  - Use "share" functionality of tracker to notify it about shared file.
  - After sharing client should start seeding the file.

- Downloading a remote file:

  - Command: get <path to.mtorrent file> <destination path>
  - Use of "seederlist" to retrieve the peer list from where the file can be downloaded.
  - Seeder list once retrieved is considered as static until the file is downloaded or cancelled downloading.

Part 3 Torrent Clients

- Torrent client should have the following functionalities:
    - Share files to the tracker and generate a corresponding ".mtorrent" file.
    - Retrieve peer information from tracker upon providing a ".mtorrent".
    - Download files from multiple peers simultaneously, same goes for upload.
    - Maintain mapping of paths of files and related .mtorrent files-this should be persistent across runs.

- Implementation Specification-
    - Start client using-
      ./client<CLIENT_IP>:<UPLOAD_PORT><TRACKER_IP_1>:<TRACKER_PORT_1><TRACKER_IP_2>:<TRACKER_PORT_2>:<log_file>
    - When the application starts, the available .mtorrent files and related files will be checked and request to add the client name in respective files on tracker server will be sent.

- The client will have a menu-driven interface to do the following:
    - **Sharing a local file:**
        - Command: share<local file path> <filename>.mtorrent.
        - Our client must download the same file from multiple peers simultaneously.

    - **Sending a downloaded file:** As soon as the file has begun downloading. It should be notified to the tracker and the client should be able to start seeding that file. Seeding multiple files, and the same file to multiple clients is possible.

    - **Start downloads:**
        - Command: show downloads
        - This will list out the files that are downloading and files that have finished downloading.
        - To differentiate, downloading files will have[D] and downloaded files will have[S] before the file name.

    - **Removing a shared file:**
        - Command: remove<filename.mtorrent>
        - This should remove the file information for that file from the tracker and from the persistent file database along with the .mtorrent file but not the actual file.
        - Remove only when the file has been 100% downloaded.

- **Close application:** When the application is closed, a request should be got to the tracker to remove the client from the seeder list of all files it is present in.

6. References

Johnsen et al, 2005, Peer to Peer networking with BitTorrent.

Weblink: http://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf