BitTorrent and Attacks against It

Magnus Jensen

4. BitTorrent and Attacks against It

- BitTorrent and Attacks against It

    - Motivation
    - Terms
        - Tracker
        - Seeds
        - Leechers
        - Swarm
        - .torrent
    - The BitTorrent protocol
        - Available pieces
        - Downloading
        - Piece strategy
    - Attacking BitTorrent
        - Harming the swarm
        - Taking advantage

1. Motivation

Great, so welcome to Piracy 101!

Kidding a side, BitTorrent is an easy way to share (very) massive files.

- is properly the most succesfull p2p system in the world
- some estimate it to be half of all internet traffic

2. Terms

BitTorrent is based on a number of different participants and actors, so let's quickly make clear how the different aspects of it.

2.1. Tracker

BitTorrent is a semi decentralised system, with a single centralised component for discovery of other peers; this is called the tracker.

It is the ever old question of P2P networks, "how do join the network, and then discover and talk to other peers?" This is what the tracker does, it help peers come in contact with each other. Doing

so, is done completely random, among available peers.

> – **Tracker**
> – Help peers come in contact, at random
> – Never shares torrent data

This is the only thing that the tracker does, meaning, that it never shares any of the data that the torrent provides access to.

The tracker is thereby only to connect with other peers, and gets out of the way after this step. But as always, there is an exception to the rule. Some networks use privat trackers, where one have to be logged in. These networks often keep tracks of ones upload and download ratio, to reward good behaviour and punish bad. Thereby, this is a job for the tracker, why it keeps track of such.

2.2. Seeds

If a peer has the whole file in the storage, it is named as a Seeder, well I guess because it can seed the file, meaning that it can share every bit of the file.

> – **Seeder**
> – Has the whole file
> – Can share the whole file

Therefor, it is a seeder which start the whole process, when it connects to the tracker, to make its content available for others to acces.

When other peers has fully downloaded the file, they too will become seeders; where it is common practice to leave the torrent client running for others to become seeders.

2.3. Leechers

Leechers are the peers which do not have the whole file yet, and is still downloading it.

> – **Leecher**
> – Downloads a torrent
> – Uploads the bits it got

But even though the leecher is download, doesn't mean that it should not upload; why it just upload those bits which it already has.

2.4. Swarm

The swarm is pretty simple; it is the union of the seeders and the leechers for a given torrent.

> $$Swarm = leechers \cup seeders$$

But being that swarms often can be very large, one would only connect to a subset of it, called ones *peer set*, which is assigned by the tracker.

2.5. .torrent

A .torrent file describes a given torrent. For a file to become a torrent, it gets broken down, often into thousand of parts, and then it is the job of the .torrent file to allow us where to find each one.

> - **.torrent**
> - describes a torrent
> - trackers, "file" locations

Information about trackers, and other kinds of meta data is also part of the file.

The file is not part of the BitTorrent system it self, and is shared traditionally, like simply downloading a file from the web of distributed via email.

More technically, the .torrent file contains a *bencoded* python dictionary, containing at minimum the keys: 'announce' and 'info'.

> - **Keys**
> - `announce` is the url of the tracker
> - `info` is another dict with the following keys
> - `- name` of the file
> - `- piece length` the length of the pieces which the file is broken into
> - `- pieces` pieces is a string containing the SHA1 hashs of all the pieces
> - `- length` and at least the total length of the shared file

Though it is also possible to share files, where the ‘length’ key is replaced by a ‘files’ key, containing information about the files.

3. The BitTorrent protocol

Let’s now dive deeper down into the protocol of BitTorrent.

When retrieving a list of roughly 50 peers from the tracker, the new peer will connect to about 30 of them, called its neighbours, over TCP.

3.1. Available pieces

This is where the new peer will send a *bitfield* message to its new neighbours.

> Bitfield: 0b1101111111

The bitfield is a space efficient representation of the pieces of the file which the peer got of the file. Though if the new peer does not have any part of the file, it will not be send.

Being that the bitfields are zero-indexed, this means that here, the second piece of the file is missing.

Once the piece has been obtained and its SHA1 is guaranteed, it sends a *have* message to its neighbours, letting them know its available at this peer.

3.2. Downloading

Downloading in the BitTorrent protocol is made to expect peers to contribute, that means that it is build as a tit-for-tat kind of manner, if you wanna download you have to upload.

In the protocol, it means, that you cant just go ahead and download from the other peers, they will need to grant you permission of each piece first. This is a multi-part process.

Now that the peer is ready to download, it starts out by indicating to a peer that it is interested in download a particular piece.

> choke/unchoke = no/yes to download

Now its up to the peer if we are allowed to download or not, referred to as being unchoked or choked. This is a way for the peers to make the game fair. If a peer does not contribute, meaning that we are not able to download from it, we can choose to choke those peers, punishing them for not sharing and thereby trying to make them share. This is why we should be downloading from peers which are interested in our pieces. Though choking is reconsidered every ten second ish.

But what if everybody is choked? That could happen. Well this is where we pass the blinds around as in poker. Every 30 seconds or so, one or more peers will get *optimisticly unchoked*, why it will download and thereby should unchoke those which is download from. Stopping these kinds of deadlocks.

Now, choking only makes sense for leechers, since seeders should at all time be sharing, upping there ratio.

3.3. Piece strategy

Great, so quickly this last thing about the protocol: which pieces to downloading.

You could do a *random first* or a *rarest first*, kind of strategy, and I think those are pretty self explanatory.

Though, when joining and having nothing to share, one relies on *optimistic unchoke* to be able to download and one should just be happy to download something, why *random first* should be used, and then later shift over to *rarest first* when you have something others are interested in.

~~Der er også video streaming torrent ting, popcorntime, hvor man nok burde arbejde med en strategi der hedder: "send mig det næste jeg har brug for"~~

4. Attacking BitTorrent

Great so having discussed BitTorrent, lets now discuss how to attack it. Yes. This is war.

4.1. Harming the swarm

The first kind of attacks we will be looking at, means to harm the swarm; where we wish to make it difficult for other peers to download a specific file.

Now why would we wanna do something like that? Well, it could be that we were a movie production company, where one of our movies got leaked on the internet, and because we have limitless ressources, we wanted to stop the sharing of our movie!

This could be pretty simple, as we could have a lot of peers claim to have a piece, and when people try to download it, we simply send them a bad piece.

> Claim to have piece, share bad.

Well that works! But it takes a lot.

Lets look more generally on some attacks on a swarm, starting with a Sybil attack of *piece lying*.

> - **Piece lying**
> - `join with peers` as it is a Sybil attack, we start by joining with a bunch of peers
> - `rare first strategy` now .. we wanna use the rare first strategy.. mmm.. I wonder what a collection of malicuis peers could do with this strategy?
> - `claim rare pieces` now we wanna lie about having the rare pieces, making them appear not rare. In this way, when the peers having the actual rare pieces leave, the pieces will become extinct, which means the swarm has failed.
> - `any interested?` it could be that some peers use the random strategy, and in that case and they ask for a piece we lie about, we simply choke them

We could also make an Eclipse attack, where we aim to isolate a peer from other peers.

> - **Eclipse attack**
> - `Join with peers` again we need to join with a lot of peers
> - `connect to real peer` then we seek to find a real peer to connect with
> - `isolate!` and if so, we tell the other malicious peers about this peer, and we all try to connect to him! Doing so, we are able to isolate the peer, and then we can decide what happens next.

4.2. Taking advantage

We can also attack BitTorrent in other ways, actually in less aggressive and less destructive ways - what if we just wanna be selfish?

What we really wanna do is get a free ride.

> - **Optimistic unchocke**
> - Slow

This could easily be done using the optimistic unchock, and then just download small bits of data whenever a peer randomly unchocks us, and in return never upload anything. But because of the very tit-for-tat nature of bittorent, this is a slow process. Can we do it faster?

Yes!

> - **Sybill download!**
> - Split bandwidth
> - Act as several small peers

We can simply split our bandwith up, acting as several smaller peers, and then we will theoretically be able to download faster, because we constantly across all our Sybil peers get optimistic unchoked. This is also an attack which is easy to stop though, since all of our peers come from the same ip, it could be that only one peer is allowed at a time from an ip.

One could also be using a strategic client, such as BitTyrrant, which does the minimal needed to stay unchoked.

> - **BitTyrrant**
> - Strategic client
> - min. work for unchoked
> - `vary active set size` it turns out that there is a relationsship between how big ones active set size is and how much one is able to download with minimal work, so bittyrrant first of all varies the active set size
> - keep track of HAVE messages

BitTyrrant does so by keeping track of the HAVE messages posted by the other peers; and because the piece size is known to all, it can begin to estimate their bandwith. And knowing these thing, BitTyrrant

can begin to estimate how much a peer needs upload for us to be able to download from it; and who and how many it should download from and upload to.