BitTorrent – Peer To Peer File Sharing

CS290F: Networking for Multimedia
Mini PhD Major Area Exam

I) Introduction

Content distribution is an important topic in networking and has been evolving from the start of the Internet. Content distribution began with the simple Client-Server structure in a unicast format. While this is still the cornerstone of the Internet, newer methods of distribution have been developed. These newer methods of distribution are usually developed to reduce the cost of distributing the content, usually in the form of reducing bandwidth restrictions and the bottleneck at the Server. Content Distribution Networks (CDNs) use multicast and anycast to attempt to reduce network resource costs of distributing content. Peer to Peer (P2P) file sharing decentralizes the classical Client-Server structure, removing any bottlenecks on the Server side. BitTorrent, a P2P file sharing protocol, is one of the most popular ways to distribute large amounts of data.

This paper will define Peer to Peer networking then take an in depth look at the BitTorrent protocol. The BitTorrent protocol has been evolving over the years, adding additional features, combating exploitations.

2) Peer to Peer Networking

The P2P structure is a distributed network architecture where node's on the network contribute to the overall system without the need of a centralized controlling server. In the classical Client-Server network structure, Clients are strictly consumers of resources and Servers are strictly producers of resources. Peers in a P2P structure are both consumers and producers of resources. Examples of resources that P2P networks share include processing power, network bandwidth, and disk storage.

P2P file sharing networks are P2P networks that the specific resource that is being produced and consumed by peers is computer files. Peers download files from other peers rather than though a central Sever. This offloads the networking constraints of the Server and spans it across multiple peers.

3) BitTorrent Introduction

"Incentives Build Robustness in BitTorrent," by Bram Cohen, the author of the BitTorrent protocol, was published in the Workshop on Economics of Peer-to-Peer Systems on May 22, 2003. At the time of the publication, the BitTorrent protocol was already implemented and in use. Cohen's paper goes into the basic design and implementation of the first BitTorrent client. Cohen's goal in designing BitTorrent was a peer to peer file distribution system that achieves a higher level of robustness and resource utilization than any other current implementation though a modified tit-for-tat scheme.

A) Publishing Content

Content is made available for download though the BitTorrent protocol initially by the publishing of a *.torrent* file on the web. This small file contains information about the target file that will be downloaded, and most importantly, information about the target file's tracker.

The tracker is responsible for announcing other peers trying to download a file over BitTorrent. The tracker by default returns a random list of peers to a requesting client. Cohen explains that the tracker is the single point of congestion in the BitTorrent implementation.

The target file that will be downloaded is divided into many smaller, approximately a quarter megabyte, files called pieces, which in turn are fragmented into subpieces. The *.torrent* file contains a sequence of SHA1 hash files that allow the verification of a successful downloaded piece.

B) Download Strategies

Cohen introduces a few download strategies that his implementation uses to achieve better download rates. These strategies are actually methods of selecting what piece to attempt to download. The first strategy is to download complete pieces at a time. If the first subpiece is downloaded, then the client requests the other subpieces of that file. This strategy is used in combination with a piece selection strategy.

When a client starts a torrent download and has no pieces itself to upload, it enters the Random First Piece stage. The client must attempt to get any piece possible so it can engage in normal BitTorrent traffic, rather than in just the random un-chocked startup with will be discussed later. During normal downloading, a client downloads pieces in a Rarest First style. More populated pieces tend to have more available bandwidth thus allowing the pieces to be downloaded later with no penelty. Finally, when the download is almost complete, the client enters an End

Game mode where it re-requests downloads of pieces that it is currently downloading in hope to find a faster peer. The client then cancels the slower of the peers.

Cohen explains that if peers only uploaded to peers who provided good download speeds there would be no methods of finding new peers who might provide better bandwidth. Optimistic Unchoking is the random selection of a peer and engaging in unrestricted traffic. This is used to find new peers that might provide good download speeds.

C) Cohen's Paper

Cohen's paper introduces a new protocol that obtained fast and widespread adoption. It fails to however provide background information on a few of it's economic strategies that are used in the development of the protocol. For example, it fails to explain what the Prisoner's Dilemma is and only vaguely relates it to BitTorrent.

Cohen's paper also did not include any experimental or in-depth real word data. The paper includes only one graph that shows the number of peers and seeders over time for a 400 megabyte file. Cohen's paper would have benefited from including and analyzing BitTorrent traffic and relating it to other decentralized file distribution methods, showing where BitTorrent succeeds and the other methods fail.

4) BitTorrent Measurement and Analysis

Pouwelse, Garbacki, Epema, and Sips in their paper "The BitTorrent P2P File-Sharing System: Measurements and Analysis" collect and analyze real world BitTorrent traffic and relate that to other forms of P2P file sharing systems. This paper fills one of the holes in Cohen's paper, but to be fair, came out after BitTorrent became incredibly popular and that type of widespread data was available.

A) Experiment Setup

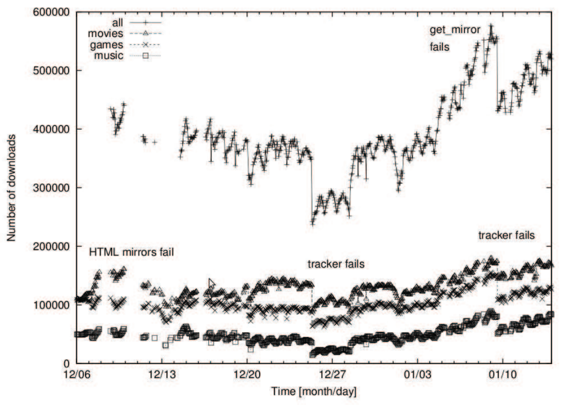
Data was collected over an 8 month period from Suprnova.org, one of the most popular torrent hosting sites. Suprnova uses a mirroring system that balances user requests by redirecting them to the mirrored site. The first part of the experiment uses three scripts. First a script to check the availability and response time of the Suprnova mirrors. Second a script to scrape Suprnova's pages, downloading all torrent files. Finally, a third script to parse the torrent file and check the status of the trackers.

The second part of the experiment was monitoring the peers this was achieved by a script to select a file, a script to contact the tracker to get a list of peers for a file, and a script to contact individual peers downloading a file and get statistics about it.

B) Results

Five different results are reported in the paper: overall system activity, availability, integrity, flashcrowds, and download performance. The paper does a good job explain and using figures to explain the results.

The paper presents a time slice focusing around Christmas 2003 because of it's large variance in number of peers downloading due to numerous failures in mirrors and trackers. Figure 1 showing this time slice is replicated below. This shows the weaknesses in the current implementation of BitTorrent in having a centralized tracker and use of a website like Suprnova. The paper however lacks in presenting an overall system activity from the entire 8 month survey. This would have allowed the identification of the growth of Suprnova and BitTorrent along with other things.



Number of users downloading or seeding on BitTorrent/ Suprnova for one month (Dec'03-Jan'04), from [2]

The availability of mirrors had a direct correlation with the popularity of a download. When the file servers that hosted torrent files went down, it blocked essentially blocked any peer from starting a new download. The availability of peers was also analyzed, showing that peers often did not seed after finishing downloading. The authors identified this problem is due to there is no incentive within the BitTorrent protocol to seed after finishing downloading, nor on the Supernova site or trackers, but provided no possible fixes to the solution.

Integrity of the torrents was mainly controlled by the Suprnova website, outside of the BitTorrent protocol. A combination of message boards and moderators rid any uploaded fakes that the authors tried to post quickly and efficiently. One weak point is that a mirror might not be trustworthy and can inject data into the website or torrent.

When a new file is pushed to the system and gains popularity quickly, this effect is called a flashcrowd. The

BitTorrent system along with Suprnova's structure was easily able to handle the effects of the flashcrowd. Finally, download performance is related to the number of downloaders in a swarm. Download performance is directly related to the popularity of the file, and when the popularity of the file dies off, so does the ability to download the file.

5) Expanding the Protocol

The tracker has clearly been identified as the soul centralized component of the BitTorrent protocol. The tracker was identified as the main contributing factor the the availability of the BitTorrent system in the Suprnova analysis. Decentralizing the BitTorrent protocol would further increase the protocol's robustness.

Two methods of decentralizing of the BitTorrent protocol have been widely adopted. The first and simpler of the two is Peer Exchange (PEX). PEX simply involves two connected peers to exchange their knowledge of other peers in the swarm, increasing the overall connectivity. The second method of decentralizing the BitTorrent protocol is though the use of Distributed Hash Tables (DHTs).

A) Distributed Hash Tables

Crosby and Wallach of Rice University analyze BitTorrent's two Kademlia-Based DHTs. Kademlia is a Distributed Hash Table designed for peer to peer networks. Mainline DHT and Azureus DHT are both based off Kidemlia but are not compatible with each other.

The paper finds many flaws in the current implementations of the Kademlia DHT. It finds that the current implementations of Kademlia are incorrect, causing the Mainline DHT to dead-end it's lookups over 20% of the time and the Azureus DHT nodes reject 50% of the key stores. Some of the problems can be contributed to bugs in the clients. Along with not working correctly, the average lookup time is slow, around one minute.

The paper identifies the weaknesses in the current implementations of Kademlia-DHTs, stemming from incorrectly implementation and poor DHT parameters. While some of these problems can be fixed while maintaining compatibility with the current infrastructure, the paper recommended a incompatible jump to a new DHT implementation that addresses all of the problems laid out. Yet these changes would not solve the inherent problems of security with the DHT.

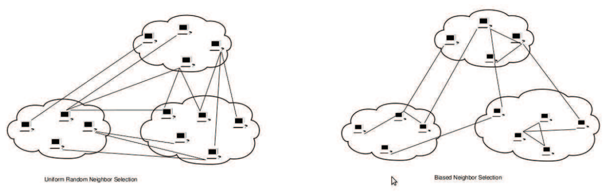
In the end, BitTorrent clients benefit from the current implementation of DHT no matter how flawed it might be. It currently is not ready to replace the centralized tracker, but inefficiently allow peers to join a swarm if the tracker is down. Even if the peer only finds one other peer from the DHT, though PEX the peer can join and participate in the swarm.

6) Improving Traffic Locality

Peer to Peer network traffic has substantially increased over the past few years. Internet Service Providers (ISPs) are starting to throttle, or cut back, bandwidth of P2P traffic that cross over to other ISPs, primarily focusing on BitTorrent traffic. Cross-ISP traffic increases the operating cost of an ISP dramatically, thus limiting it is in the ISP's best interest.

The BitTorrent protocol that Cohen laid out in his paper, and that is currently implemented by all clients, does not discriminate peers, every peer is treated the same. When multiple peers are within an ISP all requesting the same file over BitTorrent, the file is crossed over from other IPS multiple times. To reduce the burden on ISPs, and to avoid the throttling that might be in place, Bindal, Cao, Chan, Medval, Suwala, Bates, and Zhang propose selecting peers locally within the ISP in their paper “Improving Traffic Locality in BitTorrent via Biased Neighbor Selection ”.

The paper states two methods for implementing biased neighbor selection. The first proposed solution is to modify client and trackers. The clients would report to the tracker it's local ISP, then the tracker would respond with a selection of it peers that are local to the ISP. The second proposed solution requires ISPs that use P2P traffic shaping devices to intercept the peer list returned from trackers and substitute external peers with local ones. Figure 1 in the paper, replicated below, visualizes the differences between the current implementation of BitTorrent and how it would look with biased neighbor selection.



Uniform random neighbor selection in the standard BitTorrent versus biased neighbor selection, from [4]

Traffic throttling does not solve the redundancy in cross ISP traffic. Traffic throttling only minor reduces the redundancy and makes the BitTorrent downloads happen at a much slower rate. Biased neighbor selection, on the other hand, does reduce cross ISP traffic redundancy when the amount of external peers used is low.

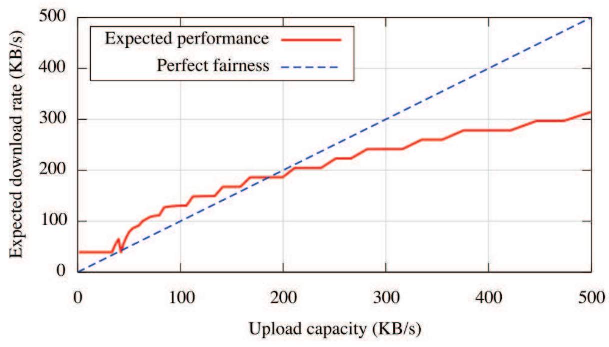
The paper then presents the results of its simulations of clients using biased neighbor selection. It shows that for a throttled ISP setting, small portion of users, around 25%, will not experience as fast downloads as they would have without biased neighbor selection. However, the other 75% of peers will on average experience a decrease in download time. When the ISP is not throttling the download time difference is not as great on either end, and the crossover point was moved up to around 50%.

7) Exploiting the Protocol

BitTorrent, as described in Cohen's paper, attempts to create robustness though incentives. Piatek, Isdal, Anderson, Krishnamurthy, and Venkataramani in their paper “Do incentives build robustness in BitTorrent?” question Cohen's design in whether robustness is actually achieved though the incentives put forth. The authors introduce BitTyrant, a BitTorrent client that uses strategic gaming to increase the client's download speed.

A) BitTyrant

The authors first attempt to locate areas in the BitTorrent client where the client can be selfish and where this is from in the protocol. This is done performance modeling parameterized by real world data. Figure 4 shows the expected download performance as a function of upload capacity, which has been replicated below. It can be seen in this figure that there is not a one to one ratio as a pure tit-for-tat scheme would ensue. This is identified as one of the key exploitable parts in the BitTorrent client, the ratio of upload to download.



Expectation of download performance as a function of upload capacity, from [5]

BitTyrant actively attempts to maximize its performance though three strategies. First, strategically pick peers and upload rates to those peers to maximize the download per unit of upload ratio. Second: strategically maximize the number of active peers until the benefit of adding a new peer is outweighed by the costs of getting replicated data. Third: strategically lower the upload bandwidth to peers as long as that peer continues to upload back, reallocating saved bandwidth to adding an additional peer to the active set.

In addition to the strategic gaming methods described above, BitTyrent implements other methods to cheat current BitTorrent clients. This includes attempting to get peers opportunistic unchoked slot multiple times by reconnecting with a different client identifier. BitTyrent also attempts to connect to older seeding clients that use a seeding algorithm that uploaded to faster downloading clients first in order to maximize the distribution. BitTyrent finally can falsify the pieces it has in order to gain the opportunistic unchoked slot from certain peers.

B) Implementation and Results

The paper then compares BitTyrent to the client it is based off of and most popular in the data analyzed, Azureus. The two clients were capped with upload speeds of 125 kb/s with Azureus set to its default settings. Both clients would join selected swarms with file sizes under 1 gigabyte and their download times would be compared. BitTyrent"s download speed was on average of over 70% faster than that of the Azureus. It is not stated why swarms that are distributing files larger than 1 gigabyte are ignored. BitTorrent was developed in the mind for distribution of large files, and limiting the swarms that are distributing large files seems short-sided.

BitTyrent was also simulated in a swarm of just BitTyrent clients. This simulation showed that if all the clients in a swarm are BitTyrent, download times of clients with lower capacity for uploading suffer, their download times increase. This simulation however does not model real world swarms. The number of clients was static at 350, the file size was set at 5 megabytes with a 128 kb/s seed connection. A more accurate simulation could have easily been modeled by looking at the actual data collected from swarms.

8) Stopping BitTorrent Gamers

Levin, LaCurts, Spring, and Bhattacharjee have taken a closer look at BitTorrent's tit-for-tat structure in their paper “BitTorrent is an auction: Analyzing and improving BitTorrent’s incentives.” In their paper, they clearly state and disprove the common misconception about BitTorrent, focusing mainly on it's perception of using tit-for-tat when it actually models an auction more closely.

A) The Problem

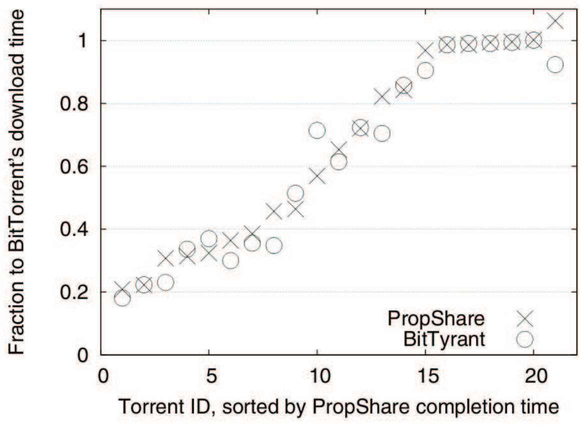
The paper explains that tit-for-tat was originally the idea proposed by Cohen, but to improve download speeds, the specification was relaxed. This relaxation came mainly in the unchoking algorithm used to help find new peers that have fast uploading rates. This relaxation of the tit-for-tat thus created opportunities for exploitation, as explained in the BitTyrant paper. The two most widespread clients that exploit those weaknesses, BitThief and BitTyrant, are introduced.

BitTyrant attempts to find the smallest winning bid to get download bandwidth from a peer by adjusting the uploading bandwidth by small increments. This allows BitTyrant to minimize its upload bandwidth to each peer it downloads from, allowing the saved uploading bandwidth to be put towards downloading from another peer. BitThief attempts to enter as many opportunistic unchoked slots as possible, eliminating the requirement to participate in the swarm.

B) The Solution: PropShare

The paper introduces PropShare, a BitTorrent client that rewards peers with proportional shares of bandwidth. PropsShare which defeats the exploits and preforms on average better than the normal BitTorrent or BitTyrant. BitTorrent's current protocol is not fair: if two peers are connected to another, and one of the two uploads more, that peer does not receive more download bandwidth. PropShare gives out proportional bandwidth to peers that win the audition.

Multiple tests were conducted comparing the average download time of the different protocols. States that PropShare can be introduced into public today, without widespread adoption. Does not go into the affects of PropShare on other normal clients, nor how PropShare preforms with solely PropShare clients. Gives an algorithm for bootstrapping piece exchange, or the starting of peers and building trust between peers.



PropShare and BitTyrant running on live swarms, from [6]

9) Conclusion

BitTorrent's evolution over the years were due to the desire to create a more robust, fast, and distributed peer to peer file sharing network. Identifying the single point of failure, the trackers, brought about the implementation of DHTs. As BitTorrent becomes more popular, ISPs began throttling traffic, which in turn gave way to localizing peers. Once BitTorrent's modified tit-for-tat implementation became exploitable, solutions were developed. BitTorrent is still developing and soon will become a mature, well developed protocol.

References

[1] B. Cohen. Incentives Build Robustness in BitTorrent. In Workshop on Economics of Peer-to-Peer Systems, Berkeley, USA, May 22, 2003.

[2] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips. The BitTorrent P2P File-Sharing System: Measurements and Analysis. In Lecture Notes in Computer Science, Volume 3640, 2005.

[3] S. Crosby and D. Wallach. An Analysis of BitTorrent’s Two Kademlia-Based DHTs . Technical Report TR07-04, Department of Computer Science, Rice University, May 2007.

[4] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang, “Improving Traffic Locality in BitTorrent via Biased Neighbor Selection,” in 26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006), July 2006.

[5] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, A. Venkataramani. Do incentives build robustness in BitTorrent?. In NSDI, 2007

[6] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. BitTorrent is an auction: Analyzing and improving BitTorrent’s incentives. In SIGCOMM Conference on Data Communication, 2008