ネットワークアプリケーション特論レポート (BitTorrent)

Kwame Ackah Bohulu　1631133

26-02-2017

1 Introduction

In the usual case, when a file is made available for download using HTTP, the cost for uploading the file is borne entirely by the host Machine. There is the naeed to make file hosting affordable when multiple file downloaders are involved. BitTorrent, which is a file distribution system acheives this by distributing pieces of the files to all downloader involved. Implementations before BitTorrent have prooved to be inefficient due to logistical and robustness issues. These include high churn rate and fairness issues just to mention a few. BitTorrent is able do deal with these issues in ways that will be discussed in this document.

2 Technical Framework

2.1 Publishing Content

To publish content for download, one needs to begin a BitTorrent deployment. This involves placing a static .torrent extension file on an ordinary web server. The .torrent file contains the name, length and hashing information for the file to be downloaded. It also contains the url of a tracker, which is supposed to help downloader find each other.

A simple protocol which is layered over HTTP is used for communication between the tracker and the downloader. Downloaders provide information about the file its downloading, listening port,etc whiles the tracker returns a list of contact info of peers downloading the same file. With this information, the downloaders are able to find each other. To make files available for download, a seed(A downloader which already has the complete file) needs to be started. Compared with the web server and the tracker, the bandwidth requirement for the seed is very huge, since it is required to send out a complete copy of the original file.

2.2 Publishing Content

After the trackers provide information to the downloader about other downloaders downloading the same file, the logistical problem of keeping track of which downloader(peer) has what is left entirely to the downloaders. To do this, BitTorrent divides the file into 0.256 Mb pieces and the information about what piece each downloader has is shared with everyone else.

SHA1 hashes of all the pieces are included in the .torrent file and used for checking data integrity. Peers report on which peice they have only after the hash has been checked. it should be noted that the connection established between peers is full duplex and they continuously download from the peers from which they can. Ofcourse, there are cases where the pieces that a peer wants is unavailable at the moment. Letting peers make known to other peers what they have reliably utilizes all available upload capacity and requires little bandwidth overhead(10%)

2.3 Pipelining

It is necessary to avoid delay between pieces between pieces being sent, as this results in very bad data rates. To deal with this, BitTorrent futher breaks pieces into 16 Kb sub-pieces and typically piplelines 5 requests at a go.So everytime a sub-piece is received a new request is sent and pipelined. The amount of data to pipeline has been selected as a value which can reliably saturate most connections

2.4 Piece Selection

To choose pieces in a good and ensure good performance, piece selection algorithms are very necessary. This subsection lists a number of piece selection algorithms used by BitTorrent.

Strict Priority

To ensure that the download of a particular piece is quickly completed, once a sub-piece from a particular piece is requested no other sub-pieces from other pieces are requested till completion.

Rarest First

In 'rarest first', pieces which very few downloaders have are selected to be downloaded first and the more common ones are left for later download. This method has many advantages. In the case where the upload capacity of the seed is very low as compared to other downloaders, rarest find makes sure that only the rarest pieces are downloaded from the seed and made available to other peers. And since the other peers can see the pieces owned by each other the required piece can be downloaded from other peers with a better upload capacity. Also in the case where the original seed is eventually taken down, the risk of downloads never being completed is reduced using rarest first.

Random First Piece

When downloading is initialized, rarest first is not used. This is because rare pieces are usually found on a single peer and would be downloaded slower compared to common pieces found on may peers making it possible to download sub-pieces from many sources. Therefore a piece is selected randomly to be downloaded and once it is done, rarest first is used.

Endgame Mode

Pieces that are requested from a peer with a very slow transfer rate can cause download completion to be delayed. The prevent this, once all sub-pieces which a peer does not have are actively being requested it sends requests for all sub-pieces to all peers. Redundant sub-pieces are discarded. Since the endgame period is short, the end of the file is always downloaded quickly and in practical terms not much bandwidth is wasted.

3 Choking Algorithm

In BitTorrent, resource allocation is handled by the peers. They do this by downloading from available peers and by the tit-for-tat algorithm decide which peers to upload to. If a peer doesnt want to upload to a particular peer, it 'chokes' it. Choking is a temporal refusal to upload. This can be done while downloding is still ongoing. A good choking algorithm should use all available resources, provide reasonably consistent download rates for all peers, and be somewhat resistant to peers only downloading and not uploading.

3.1 Pareto Efficiency

Pareto efficiency means that no two counterparties can make an exchange and both be happier. BitTorrent's choking algorithm tends to achieve this by using a version of tit-for-tat. In this version, peers upload to peers who also upload to them. The makes available multiple active connections transferring in both directions. Occasionaly, unutilized connections are aslo uploaded to see if better rates could be found using them.

3.2 BitTorrent's Choking Algorithm

Technically, each BitTorrent peer has to unchoke a default number of 4 peers. The issue that arises is to decide which peers to unchoke, which is based only on current download rate. It is not easy to meaningfully calculate current download rate, so the current implementation calculates it using a 20 second rolling average. Resources may be wasted by rapid choking and unchoking of peers. This is avoided by the peers recalculating the peers they want to choke every 10 seconds and maintain the current state untill the next 10 second period is up.

3.3 Optimistic Unchoking

If uploading of pieces is done to only peers which currently have the best download rates, it is impossible to find out if the data rate of previously unused connections have improved over time or are better than those currently being used. To deal with this issue, a BitTorrent peer has a single 'optimistic unchoke' carried out regardless of the current download rate. The peer to be unchoked is rotated every 30 seconds, which is enough time for the upload to get to full capacity, the download to reciprocate, and the download to get to full capacity.

3.4 Anti-snubbing

In the case where a peer is choked by all peers it was previouusly downloading from(snubbed), poor download rates will continue untill the optimistic unchoke finds better peers. This is dealt with by not uploading to a peer after it has not received a single piece from if after a minute has gone by, exception being in the case of optimistic unchoke. This frequently causes more than one optimistic unchoke, causing download rates to recover quickly when they falter.

3.5 Upload Only

Once peer is done downloading, it switches to preferring peers which it has better upload rates to since it cannot use its download rate to determine who to download to. This makes it possible to utilize all available upload capacities.

Reference [1]Bram Cohen. Incentives Build Robustness in BitTorrent. In The First Workshop on the Economics of Peer-to-Peer Systems ,May 22, 2003