Bare Demo of IEEEtran.cls for IEEE Conferences

Yuzhou Wangl
School of Electrical and Computer Engineering
University of Waterloo

Email:

Sainan He
School of Electrical and Computer Engineering
University of Waterlo
Waterloo, ON, Canada
Email: s66he@uwaterloo.ca

***Abstract*—The abstract goes here.**

I. Introduction

This demo file is intended to serve as a "starter file" for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

August 26, 2015

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Related works

In order to increase system availability, some BitTorrent protocol extensions have been proposed and deployed. These approaches consider different mechanisms for peers to discover other peers including: multi-tracker, Distributed Hash Table (DHT) protocol and Peer Exchange (PEX) protocol.

The first approach is multi-tracker. To avoid overloading trackers and have backup trackers against failures, it allows two or more trackers to track one same torrent instead only one tracker. Every peer that participates in sharing a file can be tracked by one tracker and is member of one swarm. Multiple swarms tracked by multiple trackers which are associated with one file can coexist in parallel. Multiple trackers improve availability, but the improvement largely comes from a single highly available tracker. The performance of small swarms is sensitive to fluctuations in peer participation. Measurements and analysis have shown that peers in small (less popular) swarms achieve lower throughput on average[2]. In [3], the authors studied the availability of multi-tracker observe the correlated failures of different trackers can reduce the potential improvement from multi-tracker. Besides, the use of multiple trackers can significantly reduce the connectivity of BitTorrent overlay.

DHT

The PEX approach makes use of the communication among peers to share the contact information they have with each other periodically. Though Multiple versions of PEX have been implemented, their main idea is that peers keep their neighbors informed about their current contact list. With its decentralized nature, PEX can help the swarm survive much longer in case of tracker failures, thus increasing the fault tolerance of the system. Unfortunately, using PEX does not eliminate the need for a tracker because the peer need to request the tracker to know at least one other peer. According to the experiments study in [4], PEX could improve the download performance - the average reduction of the download time was measured to be around 7%. As the peer needs to send messages containing contact lists to every other neighbor, a trade-off on the frequency of messages sent must be considered. [4] shows that over 80% of PEX messages have a freshness ratio greater than 0.5, but there exists a large degree of redundancy in PEX messages.

III. Design and implementation

The BitTorrent protocol generally requires the components including tracker, metainfo file containing information about the torrent, original seeder that has the whole content and end user downloaders. Figure 1 illustrates the architecture of our system. The system consists of two main components: metadata server and client.

In our design, we use Apache ZooKeeper to act as a metadata server which provides the information of peer lists of each swarm and metainfo of each file. The top level directories in ZooKeeper consist of /peer and /file nodes. The /peer node is used to hold the hostname and port number information of each peer. The /file nodes maintains the files and swarms. Each file is associated with a descendant node under /file, we store the torrent content in the file nodes data. All peers within one file swarm are registered under that specific file node.

A client can be started either as a seeder or as a leecher with a unique peer ID. If the user has a file to share, he can start the client as a seeder. The client will create a metafile and advertise the file to the ZooKeeper. A node with filename as nodes name is created under /file and the contents of the torrent information include filename, file size, pieces length will be stored in the node. Meanwhile, the peer is registered under both the /peer node and /file/filename node with the peer ID as the nodes name and hostname and port number as the nodes data. If the user wants to download a file, he needs to initiate the client as a leecher with the files name. The client will retrieve the metafile and peer lists associated with the file from the ZooKeeper. The peer is also added to the lists. Then the client can connect to those peers in the returning list to start exchange file pieces. Downloading or uploading multiple files simply involves running multiple client instances.

A. Connection state

A peer must maintain state information for each connection that it has with a remote peer. A peer usually play the roles of both downloader and uploader, Figure 2 shows two finite state machine for these two actions.

B. Deal with churn

The dynamics of peer participation, or churn, are an inherent property of Peer-to-Peer systems and critical for design. In our system, each client spawns a listening thread to accept new connections from peers joining the swarm later. If receive handshake message successfully, current peer will add the new peer into its peer list. To deal with dropping offline peers, as every node is registered in the metadata server, we set a watch for each peer to monitor the file's node they involve in. If any node in the swarm changes, the peer will be informed and pull the updated peer list from the metadata server with dead peers removed.

C. Choking mechanism

Each peer always unchokes a fixed number of other peers which allows TCPs built-in congestion control to reliably saturate upload capacity[1]. This is decided periodically. In our implementation, a scheduled task is executed at a fixed rate to decide the unchoked sets. For simplifying, we calculate the download rates of all other peers that the current peer provide uploadings to, picks k peers who has the fastest rate. If a peer in the unchoked set is choked previously, then the current peer will send unchoke message to that peer.

D. Piece selection

Selecting pieces to download in a good order is very important for transfer efficiency. For example, if all the peers start to download the first piece from the first bit of the bitfield, it results in all peers having same bits of pieces, and can not exchange files with each other. In our design, we adopted the random first piece algorithm. The peer will check its own bitfield, pick the bit with zero randomly and request that index of piece from other peer.

IV. Experiments and result

Our system is tested on ecelinux[1-3].uwaterloo.ca. The ZooKeeper service is on snorkel.uwaterloo.ca. The evaluation focuses on fault tolerance and efficiency. We ran the system with three sizes-small, medium, large- of files of various types including: .jpg, .mp3, .txt. For fault tolerance, as the centralized tracker is already eliminated in our design, we introduced failures to peers. After killing several peers, new peers were started and connect to existing alive peers. When the new peers finish downloading, we examined the downloaded file by comparing with the original file to check the correctness. Efficiency refers to the transfer rate and download duration in terms of ranging swarm size and file size.

A. Evaluation: Fault tolerance

B. Evaluation: Efficiency

References

[1] B. Cohen, *Incentives Build Robustness in BitTorrent* in 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003

[2] D. Menasche, A. Rocha, B. Li, D. Towsley, and A. Venkataramani. *Content Availability and Bundling in Swarming Systems.* in Proc. ACM CoNEXT, Dec. 2009.

[3] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher. *Availability in BitTorrent Systems.* in Proc. IEEE INFOCOM, May 2007.

[4] Wu, Di, Prithula Dhungel, Xiaojun Hei, Chao Zhang, and Keith W. Ross. *Understanding Peer Exchange in BitTorrent Systems.* in Proc. of IEEE Peer-to-Peer Computing (P2P), 2010.