Bare Demo of IEEEtran.cls for IEEE Conferences

Yuzhou Wang
School of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada
Email: y2345wan@uwaterloo.ca

Sainan He
School of Electrical and Computer Engineering
University of Waterloo
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. Terminology overview

A. BitTorrent

BitTorrent is a peer-to-peer(P2P) file sharing mechanism, which is designed to facilitate large file transfers and make it more efficient among multiple peers. A large file is divided into blocks with same size except the last block. Based on this assumption, when peers download blocks from the server, they also upload to each other concurrently. Thus, the system achieves selfscaling as its serving capability grows with peers increasing. The most significant role in the system is called tracker, which keeps track of peers to maintain lists of alive peers, those are corresponding to a certain file, either download or upload it. Since peers send messages periodically to and new one also contact with the tracker, the lists will always keep updated. The contact information of the tracker and detail of a large file is stored in a related metainfo file known as .torrent. A peer is any BitTorrent client which takes part in a download task. It can play two roles in the system, seeder and leecher. Any peer which has and upload a complete file to those seek to download is a seeder. In contrast, those who send a request to the tracker for downloading and get a response with the list mentioned before to contact with other peers is a leecher. However, it will upload the blocks of file it owns to other leechers. There are three main policies in this scalable file distribution protocol, Local Rarest First(LRF), Tit-for-Tat(TFT) and Choking policy. As a result of that LRF policy enables peers to choose rarest blocks preferentially rather than at random to download, the system avoids a circumstance where it is likely to take some time for peers to find blocks with few replicas and slow down the download rate. TFT policy is proposed to solve free-riding problem in which some leechers are selfish by only downloading without serving others. The aim of it is to achieve a cooperation that upload bandwidth is exchange with download bandwidth. This strategy is complied by fair trading, whereby peers prefer allocating upload slots within threshold to those also send data at a fast rate in return. As to Choking policy, it is raised to assist TFT to control the number of active connections among peers. Download rate and upload rate are used respectively to determine which remote peer to be choked related to whether the peer has a complete copy of the file or not. These strategies allow BitTorrent to use bandwidth between peers (i.e., perpendicular bandwidth) effectively and handle flash crowds well[1].

III. 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.

It is obvious that limited tracker bandwidth will make an effect on upload/download rate of peers. In order to reduce this impact and accelerate building connection between peers, Andrew and Arvid[1] exploit a trackerless structure using

Distributed Hash Table(DHT) protocol. In this protocol, each peer has a DHT node, which maintains contact information of closest nodes/peers in a routing table. In another word, a peer can play a role of tracker, and thus locating those peers related with its requesting file by itself, which simplifies the bootstrapping in the original mechanism. For determining the closeness, they use a XOR-based distance metric in Kademlia[2]. As this novel topology uses parallel, asynchronous queries to update routing table, nodes have enough information so that flexible route queries are guaranteed. When in a fault-prone situation, this system has provable great performance in consistency.

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.

IV. 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.

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[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.