Towards a Dynamic File Bundling System for Large-scale Content Distribution

Song Zhang[†], Niklas Carlsson[§], Derek Eager[‡], Zongpeng Li[†], Anirban Mahanti[SS]
[†]University of Calgary, Canada
[§] Linköping, Sweden
[‡] University of Saskatchewan, Canada
[SS] NICTA, Australia

***Abstract*—Peer-assisted content delivery systems can provide scalable download service for popular files. For mildly popular content, however, these systems are less helpful in offloading servers as the request rate for less popular files may not enable formation of self-sustaining torrents (where the entire content of the file is available among the peers themselves). As there typically is a long tail of mildly popular content, with a high aggregate demand, a large fraction of the file requests must still be handled by servers, and is not off-loadable to peers. Bundling approaches have been proposed where peers are requested to download content which they may not otherwise be interested in order to "inflate" the popularity of less popular files. We present the design and implementation of a dynamic bundling system, in which a large number of files may be bundled to form a *super bundle*. From this super bundle, smaller individual bundles, consisting of a small set of files, can dynamically be assigned to individual users. Our system has the capability to dynamically adjust the number of downloaders of each file, thus allowing popularity inflation to be optimized according to current file popularities.**

I. Introduction

In Internet content delivery, a highly scalable approach is to harness the upload bandwidth of the clients to offload the original content source. With such a peer-assisted approach, each file is typically split into many small pieces that can be downloaded from different peers. Such techniques are flexible, and scalable in that they can effectively serve popular content. However, characterization studies of file popularity [1]–[5] have shown that there typically is a long tail of mildly popular content, and content in the long tail can not be efficiently served by such techniques. The request rates of the files in the tail are not sufficient for the corresponding torrents to be self-sustaining (e.g., [6], [7]). These contents must instead be served primarily using server (or seed) bandwidth.

Menasche *et al.* [8] show that the content availability, specifically the set of peers that can provide upload bandwidth to other peers downloading the same content, can be significantly improved using *bundling*. With bundling, a set of contents are grouped (bundled) into a single file for download. Peers download a bundle that contains both desired file(s) along with some other files that make up the bundle. This approach inflates the aggregated popularity of mildly popular files, and helps increase content availability.

Bundling can be either static [8] or dynamic [7]. With *static* bundling, a pre-determined set of files are grouped together by the publisher, and every peer participates by downloading the entire bundle. With *dynamic* bundling, peers may be assigned complementary content (files or parts of files) to download at the time they decide to download a particular file. Dynamic bundling has the advantage that wasted downloads can be easily avoided in cases when the content is popular and popularity inflation is not necessary.

We present the design and implementation of a dynamic bundling system, where a server may host a large number of files. From this set of files, individual bundles consisting of a subset of the files are dynamically assigned to peers. This flexibility allows the system to dynamically adjust which files each peer should assist with. Our system can leverage tracker information about peer participation in the different files to make informed decisions about which files a new peer should serve in addition to its requested file. This set of files constitute its *individual bundle*.

Our implementation of the system uses the mainline BitTorrent source code, and builds upon the current BitTorrent specifications. We discuss the challenges in using the existing BitTorrent source code, as well as a solution that employs incremental changes to BitTorrent. Our solution entails creation of *super bundles* consisting of a large fixed set of files. This allows easy referencing within each bundle, yet enables clients to be assigned individual bundles from within each super bundle.

The primary modifications to the mainline client include the addition of a bundle file selection mechanism at the trackers, a generalized piece selection rule, and optimizations to the piece disk writing function. In particular, we perform the following modifications to the BitTorrent source:

- Revise the peer-to-tracker communication and how the tracker handles incoming HTTP requests from the peers.
- Modify the piece selection rule to give priority to pieces from the requested file and to ensure that each peer only expresses interest in pieces from its assigned bundle.
- Change the piece writing sequence to disk, to accommodate for the fact that each peer only downloads pieces from its assigned bundle, rather than the entire super bundle.

Our system is implemented and evaluated on PlanetLab.

In future work, we will present performance results for both steady-state and time-varying scenarios, verifying the operation of our design and examining the performance of example policies for dynamic selection of bundles. The full version of this work will also include a detailed description of some important but subtle design issues encountered. Due to lack of space many such issues are omitted from this short paper. Our work is novel in that no previous work has developed a working dynamic bundling system.

II. System Design Overview

We have three primary design goals. First, as with other bundle proposals, our system should allow clients to assist in the delivery of files that they do not themselves request. Second, the number of changes to the BitTorrent client should be kept at a minimum and allow for incremental deployment. Finally, our system should be flexible such that the system can dynamically adjust which file(s) each peer should assist with.

Our dynamic bundle system groups a large number of files into *super bundles*. From within a super bundle, clients can request to download individual files. When they do so, the tracker also requests that they download additional files. By keeping track of the peers currently downloading each file within a super bundle, the tracker can dynamically assign smaller *individual bundles* to each client, consisting of a small subset of the files in the super bundle. Using information available at the trackers, this allows our system to take into account current content popularities and adjust the relative file download frequencies.

To make informed decisions about which sub-contents (or files) a new peer should download in addition to its requested file, our system must make some modifications to the information maintained by the trackers and the information exchanged between peers and trackers. For efficiency reasons, we must also modify some additional BitTorrent components. We modify the mainline BitTorrent source code, and make use of current BitTorrent specifications. The primary modifications carried out include communication protocol revisions, the addition of a bundle file selection mechanism at the trackers, a generalized piece selection rule, and optimizations to the piece disk writing function, as outlined below:

- The communication protocol between the local[1] peer and the tracker needs to be revised, so that the tracker can learn the initial file that the local peer is interested in downloading and return the suggested supplementary bundle file to the local peer according to the bundle selection policy of the tracker.
- The tracker mechanism needs to be revised. A tracker bundle selection policy will be applied to choose an appropriate supplementary bundle file for the local peer.
- The piece selection policy needs to be revised as peers now only download particular files that are part of their assigned bundle. We refer to the new piece selection policy as the *greedy* policy. Here 'greedy' refers to the fact that a peer only requests and downloads the file pieces that belong to its bundle.
- The piece disk writing algorithm needs to be revised as well. In particular, we modify the piece priority in the disk writing algorithm, so that the pieces of interest can be written to the disk with high priority.

The general principle of the design is to utilize the current mainline BitTorrent structure and specifications and make revisions as necessary. The design supplements the current system and does not affect the mainline BitTorrent system mechanism. The following sections briefly discuss the aforementioned implementation issues.

III. Peer-to-Tracker Communication

There are two types of communication protocols used in BitTorrent: peer-to-tracker and peer-to-peer. Here we consider the peer-to-tracker communictaion. The local peer and the tracker communicate through the HTTP protocol. The local peer sends information such as torrent infohash and listening ports to the tracker. The tracker updates its peer information with the incoming message, and returns the peer set to the local peer. The local peer periodically communicates with the tracker so that the peer information on both sides is up-to-date.

In the HTTP communication between the local peer and the tracker, there are four types of messages defined by the key parameter "event" in the mainline BitTorrent system: `started`, `no event`, `completed`, and `stopped`. The client sends a `started` message to the tracker when it begins to download, a `completed` message when it finishes downloading the file, and a `stopped` message when it is gracefully shutting down. The `no event` message is simply used to periodically update the tracker during the download. These four message types correspond to different event processing modules on the tracker, which respond to the messages as appropriate. The tracker also relies on these messages for updating its management information for the entire swarm.

Bundle negotiation: We introduce a new HTTP message type, the `bundle negotiation` message, to be used by a local peer to express interest in files, and to ask the tracker for supplementary bundle file(s). The modified tracker uses such information (together with information on peers currently downloading the different parts of the bundle) to determine which additional files the peer should download.

Figure 1 illustrates the message flow between the local peer and the tracker in our system. In the current design, the local peer sends a new `bundlenegotiation` message (message 1 in Figure 1) to the tracker before it sends the `started` message. When the tracker learns the local peer's desired file in the `bundle negotiation` message, it will use the bundle selection policy to check whether a bundle should be formed. If a bundle can be formed, the tracker will return the supplementary bundle file to the local peer (message 2). The local peer will send a `started` message (message 3) after receiving `bundle negotiation` response from the tracker. In our system, communication begins with a new

[1]We use "local" peer to refer to the peer on which our BitTorrent software is running.

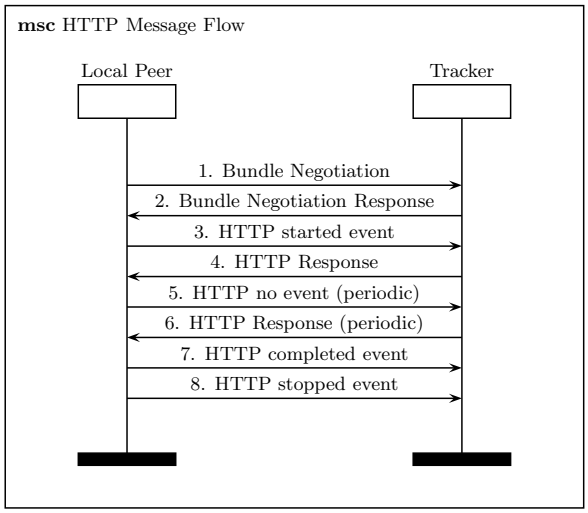

Fig. 1: HTTP message flow for the dynamic bundling system.

`bundle negotiation` message and response (messages 1 and 2) before the normal `started` message and response (messages 3 and 4).

Bundle selection: A bundle selection policy is needed to determine the constituent files of each dynamically formed bundle. The tracker applies this policy to select the supplementary bundle file(s) during the bundle negotiation stage. To facilitate dynamic bundling, the tracker obtains the mapping of the file names and the corresponding number of peers in the system. A new status file is created on the tracker to record such information. The tracker can obtain the number of peers currently downloading a specific file from the new status file. The new status file is updated when `started` messages and `stopped` messages are received by the tracker.

IV. Peer-to-Peer Comunication

The local peer uses the Peer Wire Protocol (PWP) to communicate with the peers in its peer set. First, the local peer establishes TCP connections with each peer in its peer set. Once a TCP connection is established to a neighbor and a PWP handshake has occurred successfully, file pieces can be uploaded and downloaded over the same connections.

Tit-for-tat unchoking: For efficiency reasons, at each point in time, the local peer typically only uploads pieces to a subset of its peer set. These peers are said to be unchoked. To provide incentive to peers downloading other content, BitTorrent's rate-based tit-for-tat unchoke policy is used to determine which connection can be used to upload data at each point in time. More specifically, we apply the policy on the entire bundle (rather than on peers downloading each separate file). This ensures high piece sharing efficiency.

Piece selection policy: As each peer has an individual bundle, and therefore is interested in pieces of different files, we employ a piece index filter for its piece selection policy. The peer only requests and downloads pieces belonging to files in its bundle.

Our implementation modifies the piece selection policy, used to determine the next piece to request for download. It was found necessary to consider carefully the cases in which the local peer receives a HAVE message from a remote peer when choked or unchoked by the other peer.

V. Piece Disk Writing Algorithm

The piece disk writing algorithm is used to increase the disk space utilization. It tries to avoid holes in the allocated disk space. This algorithm allows BitTorrent to achieve compact file space growth on the disk. Unfortunately, the holes structure in BitTorrent may cause issues for the dynamic bundling system, when a peer only requests and downloads the bundled file pieces. The bundled file pieces will be allocated on the spaces with the sequence of the holes list but the peer never downloads other pieces in the torrent. The default piece disk writing algorithm will result in slow download times for the bundled file pieces or the pieces can never be rearranged to their right places. Our new system design includes a modified "holes" list, to avoid performance degradation caused by BitTorrent's piece disk writing algorithm.

VI. Conclusions

In this short paper, we describe the design and implementation of a dynamic bundling system. We create super bundles of many files and let users pick which files they want and request that they also help out in the sharing of one or more additional files within this bundle. The files a user downloads in addition to its file of interest are determined dynamically. Dynamic allocation of individual bundles allows the system to inflate swarms based on measured demand for files and the desired tradeoff (for example between server bandwidth and download latency). In future work we will present empirical studies on the system's effectiveness for solving the content popularity problem, as well as detailed descriptions of some of the important but subtle design issues encountered.

References

[1] G. Dan and N. Carlsson, "Power-law Revisited: A Large Scale Measurement Study of P2P Content Popularity," in *Proc. IPTPS*, San Jose, CA, Apr. 2010.

[2] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, Modeling and Analysis of a Peer-to-Peer File-Sharing Workload," in *Proc. ACM SOSP*, Bolton Landing, USA, Oct. 2003.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, New York, NY, Mar 1999.

[4] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in *Proc. IMC*, San Diego, CA, Oct. 2007.

[5] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti, "Characterizing Web-based Video Sharing Workloads," *ACM Transactions on the Web (to appear)*.

[6] D. S. Menasché, A. A. A. Rocha, E. A. de Souza e Silva, R. M. Leão, D. Towsley, and A. Venkataramani, "Estimating self-sustainability in peer-to-peer swarming systems," in *Proc. IFIP Performance '10*, Namur, Belgium, Nov. 2010, pp. 1243–1258.

[7] N. Carlsson, D. L. Eager, and A. Mahanti, "Using torrent inflation to efficiently serve the long tail in peer-assisted content delivery systems," in *Proc. IFIP/TC6 Networking*, Chennai, India, May 2010.

[8] D. S. Menasche, A. A. A. Rocha, B. Li, D. Towsley, and A. V. Taramani, "Content availability and bundling in swarming systems," in *Proc. ACM CoNEXT*, Rome, Italy, Dec. 2009.