2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing

An Effective Network-Aware Peer Selection Algorithm in BitTorrent

Fenglin Qin[†*], Ju Liu[†], Lina Zheng[†] and Liansheng Ge[*]
[†]*School of Information Science and Engineering, Shandong University, Jinan, 250100, China*
[*]*Network and Information Center, Shandong University, Jinan, 250100, China*
E-mail: {qfl, juliu, zhenglina, lsge}@sdu.edu.cn

***Abstract*—This paper presents an effective network-aware peer selection algorithm to alleviate the topology mismatch issue in BitTorrent. By utilizing the characteristic of hierarchical network topology, the Tracker in BitTorrent classifies the peers into three clusters: local, intra-ISP and inter-ISP clusters based on peers' traceroute results, and then chooses close peers within the first two clusters. Thus, the closeness among peers in BitTorrent can be enhanced. Experimental results show that, compared with standard BitTorrent, the proposed algorithm can achieve better download performance and greatly reduce both ISP's backbone traffic and the cross-ISP traffic.**

I. Introduction

BitTorrent [1] is emerging as the most popular peer-to-peer (P2P) system for large-scale file sharing and content distribution, and has been widely adopted by various Internet content providers. However, current implementations of BitTorrent ignore the underlying network topology and randomly select peers among all Internet service providers (ISPs). While random peer selection is simple and resilient to peer churn, it leads to a significant mismatch between the logical overlay and the underlying network. This mismatch issue not only degrades download performance, but also brings about a great amount of cross-ISP traffic. As a result, ISPs are reduced to throttling or even blocking the BitTorrent traffic [2].

In recent years, a number of network-aware peer selection techniques have been put forward to deal with the mismatch issue in BitTorrent. Bindal et al. [3] proposed the idea of ISP-aided biased neighbor selection in BitTorrent. ISPs can employ P2P traffic shaping devices (e.g., P-Cube [4]) that would bias peer selection towards nodes in the same ISP to improve traffic locality. P4P [5] also attempts to address the mismatch issue using portal servers (named *iTracker*) in ISPs. Despite having shown to be effective in reducing ISP's costs without affecting peers' performance, ISP-aided techniques require deploying additional devices in participating ISPs. UTAPS [6] is an underlying topology-aware peer selection algorithm which considers peers' traceroute results. In UTAPS, peers can choose the neighbors within certain hop count of routers (denoted as H_{max}) and low RRT ranges. However, UTAPS is coarse-grained as the hop threshold H_{max} is presumed. T2MC [7] is another topology mismatch reduction algorithm based on traceroute results, but T2MC is also coarse-grained as it only classifies the peers into two "remote" and "close" groups. Moreover, T2MC is specifically designed for DHT-based P2P systems.

To address these limitations, we propose a simple and effective network-aware peer selection algorithm. By utilizing the characteristic of hierarchical network topology, the Tracker in BitTorrent classifies the peers into three clusters: local, intra-ISP and inter-ISP clusters based on peers' traceroute results and then chooses close peers within the first two clusters. In summary, our approach has two important features. First, in contrast to the ISP-aided techniques, we take a different approach that tries to cluster close peers without ISPs' collaboration. Second, the proposed algorithm avoids the presumed hop threshold in [6] and improves the accuracy of peer clustering in [7].

The rest of this paper is organized as follows. We first present the proposed network-aware peer selection algorithm in Section II and then evaluate its performance in Section III. Finally, we conclude the whole paper with Section IV.

II. The Proposed Network-Aware Peer Selection Algorithm

The proposed algorithm is inspired by several basic characteristics of the Internet. First, although end users can join or leave the Internet with high churn rate, the core routers are usually more stable and therefore the underlying network topology is stable. Second, the Internet, at least in China, is split into a number of ISPs (e.g., there are four major ISPs, including China Telecom, China Unicom, China Mobile, and CERNET, in China). And each ISP, as shown in Figure 1(a), presents an explicit hierarchical network topology that consists of three network layers: local PoP (Point of Presence) network layer, intra-ISP network layer and inter-ISP network layer. Correspondingly, the edge routers between different network layers, namely local edge router and ISP edge router, are connected with each other by three types of link models: local link, intra-ISP (backbone) link and inter-ISP link.

Generally, the inter-ISP link has a huger latency than the backbone link, and the backbone link in turn has a much higher latency than the local link. Figure 1(b) presents an example of path information by an end user in CERNET traceroutes to a counterpart in China Telecom. As seen from Figure 1(b), the latency on the backbone link (R3– R4) is much higher than that on local link (R1–R3), and the

978-0-7695-3762-7/09 $26.00 © 2009 IEEE
DOI 10.1109/IIH-MSP.2009.90

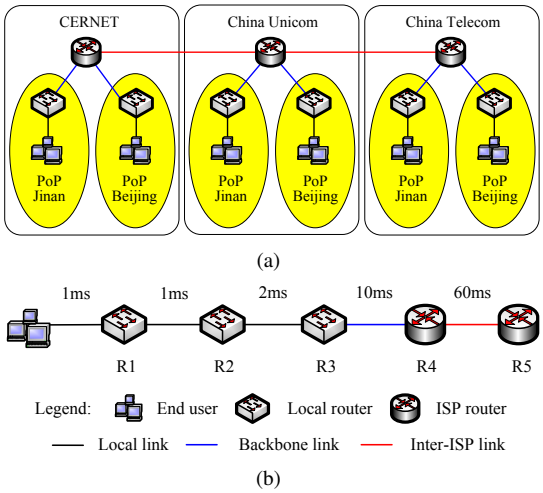

Figure 1. Illustrations of basic characteristics of the Internet. (a) An example of the hierarchical network topology, (b) An example of the traceroute results

inter-ISP link (R4–R5) also presents a huger latency than the backbone link. It usually means that these hops cross different network layers or different ISP ranges (i.e., cross local edge routers or ISP edge routers). How to find these latency leaps is the core to differentiate the local, intra-ISP, and inter-ISP clusters in our approach.

A. Peer clustering with traceroute

In standard BitTorrent protocol [8], when a new peer enters the system, it first announces itself to the Tracker to request a subset of peers to communicate with. Thus, we can extend and modify the peer to traceroute the Tracker simultaneously and then keep track of the results into a vector of <IP, Hop, Latency> tuples, where IP and Hop are the IP address and the hop of every router in the traceroute path, respectively, and Latency is the link delay between this router and its previous hop. Next, the peer reports the traceroute results to the Tracker.

Upon receiving the traceroute results, the Tracker uses the Latency attributes in the tuples to classify the routers into three clusters: local, intra-ISP, and inter-ISP clusters. The router classification method can use the k-means classification algorithm [9] with $k = 3$. A higher-level description of the router classification algorithm is listed in Figure 2 and is summarized as follows. The Tracker first chooses the tuples with minimum, average and maximum Latency attributes as centroids for three initial clusters. After calculating the absolute distance between the Latency attribute of each tuple with the centroid, the Tracker associates the tuple to the cluster with smallest absolute distance value. Then the Tracker calculates the centroid and the variance of each new cluster and repeats the above procedures until the intra-cluster variance is smaller than the threshold ϵ. Finally, three router clusters, namely local, intra-ISP, and inter-ISP clusters, are formed.

```
procedure GetRouterClusters (routers[1...routerno])
localcentroid ← min{routers[1...routerno].Latency}
intracentroid ← avg{routers[1...routerno].Latency}
intercentroid ← max{routers[1...routerno].Latency}
oldE ← -1
E ← 0
while E−oldE ≥ ϵ do
  for all i ← 0 in routers[1...routerno] do
    dlocal ← |localcentroid−routers[i].Latency|
    dintra ← |intracentroid−routers[i].Latency|
    dinter ← |intercentroid−routers[i].Latency|
    if dlocal=min{dlocal,dintra,dinter} then
      locallist.add{routers[i]}
    if dintra=min{dlocal,dintra,dinter} then
      intralist.add{routers[i]}
    if dinter=min{dlocal,dintra,dinter} then
      interlist.add{routers[i]}
  localcentroid ← the average of the locallist
  intracentroid ← the average of the intralist
  intercentroid ← the average of the interlist
  localE ← the variance of the localist
  intraE ← the variance of the intralist
  interE ← the variance of the interlist
  oldE ← E
  E ← localE+intraE+interE
List ← new Vector[3]
List[0] ← localist
List[1] ← intralist
List[2] ← interlist
return List[0...2]
```

Figure 2. Pseudo code of router classification algorithm

In the end, the Tracker chooses the router with minimum Hop attribute in the intra-ISP cluster as the local edge router, and the router with minimum Hop attribute in the inter-ISP cluster as the ISP edge router. Thus, the peers located behind the local edge router would be grouped together to form the local cluster, similarly, the other peers behind the ISP edge router form the intra-ISP cluster, and the remaining peers form the inter-ISP cluster. The more peers enter the system, the more accuracy the Tracker will have on the peer clustering.

B. Peer selection

In standard BitTorrent protocol [8], the Tracker responds with a random peer list when it receives the request from a peer. Based on the peer clustering results in the above subsection, we propose an improved peer selection algorithm on the Tracker side to return a biased peer list which consists of k close peers and $n-k$ remote peers, where n is the length of the returned peer list (the default is 50).

Figure 3 describes the pseudo code of the improved peer selection algorithm and we summarize it as follows. On receiving the request from a peer p, the Tracker first examines whether the information of p's edge routers is available. If unavailable (e.g., when the peer p is newly joined, the Tracker may not have received its traceroute results, so the information of p's edge routers is unavailable), the Tracker will return a random peer list as peer p may

```
procedure PeerSelectionOnTrackerSide (peer p)
if the information of p's edge routers is unavailable then
    add n random peers to the peer list
else
    localnodes ← the count of peers in p's local cluster
    intranodes ← the count of peers in p's intra-ISP cluster
    internodes ← the count of peers in p's inter-ISP cluster
    if localnodes ≥ k then
        add k random peers from the local cluster into the peer list
        if intranodes ≥ n − k then
            add n − k random peers from the intra-ISP cluster into the
            peer list
        else
            add all peers from the intra-ISP cluster into the peer list
            add n − k − intranodes random peers from the inter-ISP
            cluster into the peer list
    else
        add all peers in p's local cluster to the peer list
        if intranodes ≥ k − localnodes then
            add k − localnodes random peers from the intra-ISP cluster
            into the peer list
        else
            add all peers from the intra-ISP cluster into the peer list
            add n − k − intranodes random peers from the inter-ISP
            cluster into the peer list
return the peer list
```

Figure 3. Pseudo code of the improved peer selection algorithm on the Tracker side

demand a peer list immediately. If the Tracker can provide the information of p's edge routers, the peer selection is basically an iterative searching process. Specifically, the Tracker first examines whether there are enough peers in p's local cluster. If the Tracker can not get enough peers it will try to search peers in the intra-ISP cluster, and the peers in the inter-ISP cluster in turn if it still can not get enough peers after the searching process described above.

III. Performance evaluation

In this section we evaluate the performance of the proposed network-aware peer selection algorithm by comparing with standard BitTorrent in terms of peers' average download time and the traffic amount injected into both ISP's backbone link and the inter-ISP link.

We build a testbed using the similar approach in [6] with four Dell servers and five PCs connected to each other with 100Mbps switched ethernet network. As illustrated in Figure 4, three servers act as the ISP edge routers (R1, R2, and R3) and the last one as the local edge server (R4). By configuring static routes, we create an IP network that consists of three ISPs, among which ISP1 has one PoP network with two local networks (local network 1 and 2), and ISP2 and ISP3 each has one local network (local network 3 and 4, respectively). Based on a well known profile [10], we configure the local links (e.g., local link 1) with delay 20ms, ,the backbone links (e.g., backbone link 1) with delay 100ms, and the inter-ISP links (e.g., inter-ISP link 1) with delay 300ms.

In addition, we configure six IP addresses on each of four

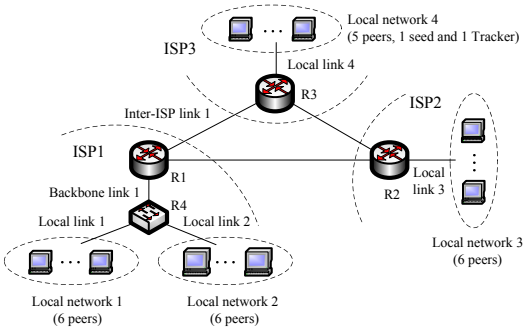

Figure 4. The topology of the testbed

PCs with virtualization technology and bind each peer with a unique IP address, and thus we have 24 peers (including the seed). For simplicity we place equal number of peers in every local network. The last PC runs the Tracker and is placed together with the seed in ISP3. Further, we assume that peers arrive in one minute's interval to simulate the flash crowd model that is most challenging for ISPs to handle.

We implement the proposed network-aware peer selection algorithm in XBT [11] (referred to as network-aware BitTorrent thereafter), and we define the peer's download speed 768Kbps and upload speed 384Kbps. As the number of peers in our experiment is small, we set the length of the peer list returned by the Tracker is five which consist of four close peers and one remote peer. Finally, we make a torrent for a file of 20MB to be delivered (the size of the file is not important as we can define different upload speeds to finish the experiment in an appropriate period).

Figure 5 gives a comparison of the average download time of the peers in different local networks between standard BitTorrent and network-aware BitTorrent. We can find that network-aware BitTorrent has a relatively shorter average download time than standard BitTorrent. In particular, the average download time in local network 1 reduces nearly 13.94% (from 612.21sec to 526.32sec) and in local network 3 the average download time reduces about 5.2% (from 561.56sec to 532.2sec). The main reason is that as network-aware BitTorrent enables a greater percentage of peers to download the file with low-delay peers within the same local networks, it decreases user perceived file download time and therefore achieves better download performance than standard BitTorrent. Moreover, the reason for more noticeable decrease in the average download time in local network 1 is that the proposed algorithm classifies the peers from local network 1 and 2 in the same PoP network into a large close cluster, thus the peers can achieve a higher download speed from close neighbors within the same PoP network.

Figure 6 depicts the amount of traffic on the backbone link 1, local link 3 and 4 that are injected into the inter-ISP links. We can find that network-aware BitTorrent generates

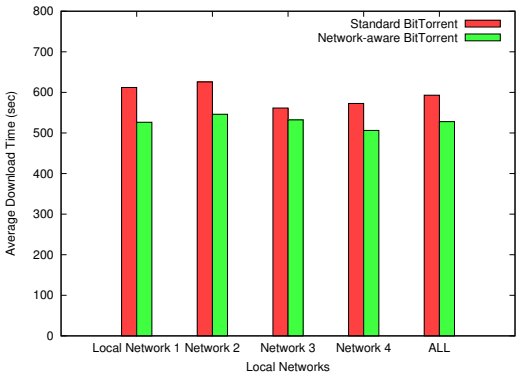

Figure 5. Average download time of the peers in different local networks

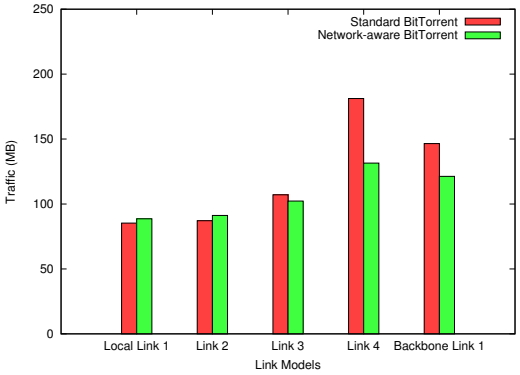

Figure 6. Traffic amount on different links

less cross-ISP traffic than standard BitTorrent. As shown in Figure 6, the total cross-ISP traffic reduces from 434.8MB to 355.1MB with the proposed peer selection algorithm. This metric has two-fold meanings. First, network-aware BitTorrent makes a more efficient use of ISP's resources because it incurs less cross-ISP traffic. Second, as ISP's backbone link and the inter-ISP link tend to have higher delays than the local link, if they can be avoided, BitTorrent will achieve better download performance. Figure 6 also shows that although traffic amount on backbone link 1 decreases, traffic amount on local link 1 and 2 experiences small increases. The main reason may be that as the peers in local network 1 and 2 are grouped into a larger close cluster with the proposed algorithm, they have more chances to download from each other than from peers outside of the local PoP network. It means that the traffic on ISP's backbone link can also be reduced with the proposed algorithm.

As indicated in previous studies (e.g., in [6]), the traceroute operation in the proposed algorithm will unnecessarily incur some additional overhead, which is an inherent limitation of our approach. From the comparison results of traffic amount on different link models in Figure 6, we believe that the overhead is within an acceptable range and will not degrade the download performance of BitTorrent.

IV. Conclusion and Future Work

In this paper, we presented an effective network-aware peer selection algorithm to alleviate the topology mismatch issue in BitTorrent, which groups the peers into local, intra-ISP, and inter-ISP clusters based on peers' traceroute results and then chooses close peers within the first two clusters. Experimental results demonstrate the effectiveness of the proposed algorithm in improving download performance as well as reducing both ISP's backbone traffic and cross-ISP traffic.

For future work, we aim to perform large-scale experiments in actual networks to further evaluate the performance of network-aware BitTorrent.

Acknowledgments

This work is supported in part by Natural Science Foundation of China (No. 60572105, 60872024), and Natural Science Foundation of Shandong Province (No. Y2007G04, Y2007G42, Q2008G03), and Cultivation Fund of the Key Scientific and Technical Innovation Project (No. 708059), Ministry of Education of China.

References

[1] BitTorrent. (2009). [Online]. Available: http://www.bittorrent.com/

[2] J. Reimer. (2006) ISPs fight against encrypted BitTorrent downloading. [Online]. Available: http://arstechnica.com/old/content/2006/08/7638.ars/

[3] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in BitTorrent via biased neighbor selection," in *ICDCS'06*, 2006.

[4] P-Cube. P-cube: IP service control.

[5] H. Xie, R. Y. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: provider portal for applications," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 351–362, 2008.

[6] W. Li, S. Chen, and T. Yu, "UTAPS: An underlying topology-aware peer selection algorithm in BitTorrent," in *AINA'08*, 2008, pp. 539–545.

[7] G. Shi, Y. Long, J. Chen, H. Gong, and H. Zhang, "T2MC: A peer-to-peer mismatch reduction technique by traceroute and 2-means classification algorithm," *Networking 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, vol. 4982, pp. 366–374, 2008.

[8] BitTorrent protocol specification v1.0. [Online]. Available: http://wiki.theory.org/BitTorrentSpecification/

[9] V. Faber, "Clustering and the continuous k-means algorithm," *Los Alamos Science*, vol. 22, pp. 138–144, 1994.

[10] Qwest Internet network service level agreement. [Online]. Available: http://www.qwest.com/legal/docs/Qwest_Internet_Network_SLA_102103.pdf

[11] XBT. (2006). [Online]. Available: http://xbt.sourceforge.net