UTAPS: An Underlying Topology-aware Peer Selection Algorithm in BitTorrent*

Wei LI[1], Shanzhi CHEN[1, 2], Tao YU[1]
([1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications; [2]China Academy of Telecommunication Technology)
Python.Gozap@gmail.com

Abstract

BitTorrent is one of the most well known peer-to-peer file sharing applications, accounting for a significant proportion of Internet traffic. Current BitTorrent system builds its overlay network by randomly selecting peers, a fact that has the potential to seriously handicap both individual performance and generate a significant amount of cross-ISP traffic. In this paper, we propose a novel peer selection algorithm called UTAPS which could selects peers within small hop counts and low round trip time (RTT) range by utilizing the knowledge of underlying topology. Consequently, the proximities among peers in the P2P overlay network are enhanced. Simulation results show that the UTAPS algorithm can achieve better individual performance in sense of low download time and reduce the traffics which are injected into ISP backbones.

1. Introduction

BitTorrent [1] is a new generation of peer-to-peer file sharing system, and has become the most popular for file sharing. Recent reports of BigChampagne [2] have indicated that there are nearly 7 million BitTorrent online users at the same time in August 2004, and nearly 10 million in August 2005. According to a recent measurement by CacheLogic [3], BitTorrent traffic represents nearly 53% of aP2P traffic and contributes to about 25-30% of all traffic on the Internet. One of the reasons BitTorrent has become so popular is that the sharing is very efficient as allowing downloads to scale well with the size of the downloading population. This efficiency is obtained by breaking up each large file into hundreds or thousands of segments, or pieces, which, once downloaded by a peer, can be shared with its neighbor peers returned from the tracker server while the downloading continues.

Performance studies on BitTorrent systems have drawn a widely attention [4-9]. Many analytical and simulation studies [6-8] have shown that the existing BitTorrent performs near-optimally in terms of uplink bandwidth utilization, and user experienced download time except for under some certain extreme conditions. However, current BitTorrent system sets up connections for data transfers among neighbor peers which are random chosen by the tracker. Although the random peer selection algorithm is simple and resilient to network churn, it could be result in a significant divergence between the P2P overlay network and the underlying IP network. Two nodes are the neighbor peers in the P2P overlay may be in different AS, or even in different ISP's network, hence the connections among peers may be extremely heterogeneous (in sense of delay, loss rate, bandwidth, et al.). The present random peer selection algorithm would not consider either case worse than the other and in fact it has the potential to seriously handicap the individual performance and generate a significant amount of cross-ISP traffic. Therefore, there is demand for new peer selection algorithms that can construct the P2P overlay reflecting the underlying network better.

Through extending and modifying the BitTorrent protocol, we propose a novel peer selection algorithm called *UTAPS* which could select peers by taking the knowledge of underlying topology into consideration. By finding out peers which are close (When we say "closeness" of two peers, there are two meanings involved: first, the hop counts between the two peers are small; second, the connection between the two peers has low RTT), better individual performance and less traffic injected into ISP backbone are achieved for the BitTorrent system.

* This work was supported by the National Natural Science Foundation of China under Grant No.60672086; the National 863 Program of China under Grant No. 2006AA01Z229.

1550-445X/08 $25.00 © 2008 IEEE
DOI 10.1109/AINA.2008.92

The remainder of this paper is organized as follows: Section 2 gives a short review of some prior works in peer selection. Section 3 describes the details of the underlying topology-aware peer selection algorithm. Experiments setup and performance analysis are given in Section 4. Section The conclusion and future work is drawn in the last section.

2. Related work

Generally, the peer selection algorithm in P2P system can be classified into the following three classes: random, end-to-end measurement based, and topology based. The random peer selection strategy is simple and resilient to network churn but cause the extreme heterogeneity in the connections among peers. Therefore, researchers have proposed many new peer selection algorithms [10-13] to improve the performance of current P2P system.

In literature [10], the authors discussed whether there is causal relationship between random peer selection algorithm and near-optimal performance of current BitTorrent. In order to answer the question above, they proposed a simple approach to enhance BitTorrent traffic locality, called biased neighbor selection, in which a peer gets the majority, but not all, of its neighbors from peers within the same ISP. By extensive simulations, that biased neighbor selection, can reduce cross-ISP traffic significantly while keeping the download performance nearly optimal. The biased neighbor selection can be implemented with the help of P2P traffic shaping devices (e.g. Cisco's P-Cube appliances [14]).

By analyzing thousands of Internet peers and conducting trace-based and Internet-based experiments, authors in [11] investigated end-to-end measurement-based techniques including round-trip time (RTT) probing, 10KB TCP probing, and bottleneck bandwidth probing to identify peers. This work reveals that the basic techniques can achieve 40-50% performance improvement. Combining the basic techniques can better identify a high-performance peer.

Literature [12] proposed the methodology of machine learning for the construction of good peer selection decisions from past experience. A decision tree is used to rate peers based on combinations of collected connection information of peers, such as load, bandwidth, and past uploading experience. Then a policy derived by Markov Decision Process is executed to select peers. The peer selection algorithm in [12] is slow due to the learning process and the complexity is also high.

End-to-end measurement-based peer selection algorithms are straightforward, however, the end-to-end probing is very time consuming. The other drawback of the end-to-end technique is that it does not consider shared link among paths, which may become bottlenecks if peers sharing a tight link are selected. In contrast to the end-to-end technique, the topology-aware technique inferring the underlying topology can make a judicious selection by avoiding peers whose paths are sharing a tight link.

The authors in literature [13] focused on peer-to-peer media streaming system and proposed topology-aware peer selection. It infers an approximate topology by network tomography technique (discovery the interior characteristics of network by probing only from its end points) [15] and considers shared network path when selecting peers.

Besides purely based on networking performance, the authors in [16] propose a genetic algorithm based neighbor selection strategy. It increases the content availability to the clients from their immediate neighbors, so the system performance has improved. However, due to the computation complexity, the algorithm in [16] can only work off-line.

3. The UTAPS algorithm

Due to the heterogeneity in the peer-to-peer environment, making a good peer selection is inherently difficult. A simple heuristic to achieve better individual performance is to find peers which are close (i.e. with low RTT and high bandwidth). However, the measurement of bandwidth is difficult. Since high bandwidth is typically placed in local LANs and core networks of ISPs, access links tend to be a bottleneck. As a result, any algorithm that increases the locality of peers provides a simple heuristic.

Inspired by fisheye routing protocol [17] as well as the heuristics described above, we propose a novel peers selection algorithm called underlying topology-aware peer selection algorithm (UTAPS). The basic idea of UTAPS is to select peers which are close based on the inferred underlying topology (the hops and RTTs among routers). There are two main parts of UTAPS: inferring the underlying topology and peer selection based on the inferred underlying topology.

3.1. Underlying topology inferring

The paradigm of inferring the underlying topology is illustrated in Fig.1. We first use network tomography technique to infer the underlying topology. This is a straightforward step in which a tool as *Traceroute* is used. Each time when a new peer joins, the tracker traceroutes the new peer and gets some knowledge (e.g. the router's IP address, the RTT and hops from

some peer to the router) of the underlying IP topology (Fig.1 (a)). With the number of peers increasing, the tracker acquires more accurate information of the underlying IP topology.

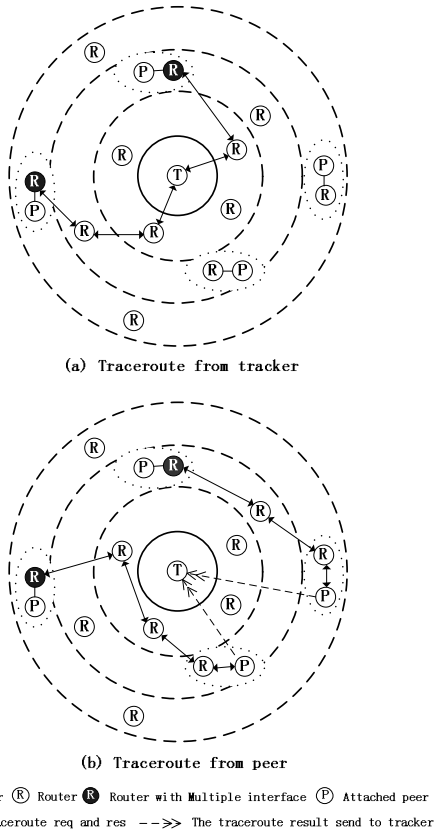

Fig.1. Paradigm of inferring the underlying topology

But, in the underlying IP network, a router may have many interfaces with different IP address; therefore, and by Traceroute, we can find these different IP addresses but can not tell whether they belong to the same router. To solve this problem, the most straightforward way is taking different addresses as different routers. Generally, there are many peers are attached to a single router's interface (subnet). Our peer selection algorithm can work well by returning peers which are close (the peers in the same subnet have a high possibility of being selected). In our UTAPS algorithm, by synthesizing the information from peer tracerouting peer and tracker tracerouting peer, the tracker can tell the accurately relationship between interface and router in most cases. Since the file delivery is directly occurred between peers, peers also traceroute the peers in the peer list returned by the tracker and then report the results to tracker (it will be introduced later) for optimal purpose (Fig.1 (b)). In Fig.1 the nodes in the dash ellipse are peers attached to the same router.

Based on the knowledge of underlying topology, tracker utilizes the topology information from the 'sights' for each router. The sight of a router contains other routers which are within certain hops or RTTs. More specifically, the sight of a router represents what it can 'see' within certain hops or RTTs. For the purpose of storing and searching efficiency, we construct hop-sights within H_{max} (e.g. 5) hops range and RTT-sights within RTT_{thresh} (e.g.100ms) range.

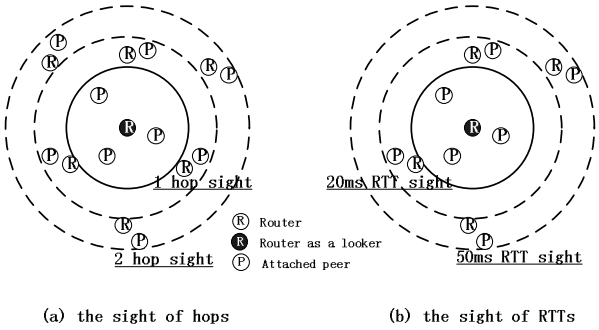

Fig.2. the sight of hops and RTTs in the router as a looker

Fig.2 gives the example of hop-sights (Fig.2 (a)) and RTT-sights (Fig.2 (b)). Since traceroute can return the hop counts and RTT information of routers on the path to peers, it is simple to use these two kinds of information to form the sights of routers.

The pseudo-code of underlying topology inferring algorithm is given in Tab.1. In actual systems, the Traceroute operation is an asynchronous process, but for convenience, we will describe it as a synchronous process in pseudo-code. The algorithms running on peers or tracker is described using rules of the form as below:

receive(*Sender* : *Receiver* : *Message*(*arg*$_1$; …; *arg*$_n$))
action(s) …

The rule describes the event of receiving a message. *Message* at the Receiver and the **action**(s) taken to handle that event. A *Sender* of a message executes the statement **send**(*Sender* : *Receiver* : *Message*(*arg1*; :: ; *argn*)) to send a message to Receiver.

Tab.1. Pseudo-code of underlying topology inferring

```
Tracker Side:
Receive(P, T, JoinReq(Peer))
/* process the join request of peer*/
{
   Res = Traceroute(Peer);
    Update the hop-sights for routers through Res
    Update the RTT-sights for routers through Res;
```

```
}

Receive(P : T : PeerTracerouteResultMsg(MsgList))
/* process the traceroute result message reported from peer*/
{
   for each Msg in MsgList do
   {
      Update the hop-sights for routers through Msg;
      Update the RTT-sights for routers through Msg;
   }
}

Peer Side:
Receive(T : P : GetPeerListRes(PeerList))
/* the peer will traceroute every peer in the peer list, and report
the result to the tracker*/
{
   for each Peer in PeerList
   {
      Msg = Traceroute(Peer);
      add Msg into MsgList;
   }
   Send(P:T:PeerTracerouteResultMsg(MsgList));
}
```

3.2. Peer selection

Different ways or sequences of utilizing the RTT-sights and hop-sights of router can lead to different instances of peer selection algorithms. Hence, we get a family of peer selection algorithms. This family of heuristics for peer selection tries to accelerate the file delivery by reducing RTT (by finding peers with small RTT) and reduce the cross-ISP traffic (by selecting peers within small hops).

Next, we will describe in detail an instance of this family of topology-awareness algorithms and give the pseudo-code in Tab.2. Basically, it is an iteratively searching process in routers' RTT-sights and hop-sights. Let $R_{Hsight}(Q)$ represents the hops of the router Q in router R's hop-sights. Similarly, $R_{Rsight}(Q)$ represents the value of RTT of router P in router R' RTT-sight. We define $R_{Hsight}(R) = 0$, $R_{Rsight}(R) = 0$ and vice versa.

On receiving the request from a peer p, the tracker examines whether the information of the router (e.g. R) which p attached to exists in the topology. (When a new peer joins, it may request peer list immediately. However, the tracker may have not finished the traceroute process and the information of the router that the peer attached to is unavailable). If the information of the router that p attached to is unavailable, the tracker will return the peer list by random strategy.

If the tracker can provide the information of router R, it first selects other peers also attached to router R. If the tracker can not get enough peers as requested, the tracker examines the routers in one hop-sight to see if it can get enough peers to return. The tracker may get more candidate peers than requested and these peers may be attached to different routers, and then it will check these routers in router R's RTT-sights and return peers attached to routers within small RTT range .

If the tracker can not get enough peers it will try to search in wider hop-sights and RTT-sights of router R until it gets enough peers. Generally, if the tracker searches H hop sight of R, where $1 \leqslant H \leqslant H_{max}$, the tracker will select peers attached to a router (e.g. Q) in the router set with the equation defined as below.

$$\sum_{i=1}^{H}((R_{Hsight}(Q)=i)\&\&(Q_{Rsight}(p)\leq RTT_{thresh}-R_{Rsight}(Q))\&\&(Q_{Hsight}(p)=H-i)) \quad \ldots\ldots (1)$$

If the tracker still can not get enough peers after the searching process described above, it will randomly select from the remaining peers.

On peer side, the peer traceroutes the peers returned by the tracker and reports the results as *Traceroute result message* to the tracker. The *Traceroute result massage* contains the hash value of the file to be delivered (peer may join multiple torrents), the IP address and RTT value of the routers on the path to certain peers. As a result, the accuracy of underlying topology is enhanced.

Tab.2. Pseudo-code of peer selection algorithm

```
Tracker Side:
Receive(P : T : GetPeerListReq(Peer))
/* Peer selection based on underlying IP topology*/
{
   /* first, find the router that the peer attached to*/
   if (the router R that the Peer attached to is not available)
   {
      retrieve PeerList through randomly selection;
      Send(T:P:PeerListRes(PeerList));
      return;
   }

   add all peers attached to router R into PeerList;
   if (the count of peers in PeerList is enough)
   {
      Send(T:P:PeerListRes(PeerList));
      return;
   }

   /* then, find some router in wider hop-sights and RTT-sights of
router R*/
   hop = 1;
   /* in our algorithms, Hmax and RTTthresh are the value all can be
negotiated by peer and tracker */
   while (hop < Hmax)
   {
      i = 1;
      while(i < hop)
      {
         find the router Q with the equation (1);
         add the peers attached to router Q into PeerList;
         if (the count of peers in PeerList is enough)
         {
```

```
            Send(T:P:PeerListRes(PeerList));
            return;
          }
          i = i + 1;
        }
        hop = hop + 1;
    }

    if (the count of peers in PeerList is not enough)
    {
        randomly select from the remaining Peers into PeerList;
        Send(T:P:PeerListRes(PeerList));
    }
}
```

4. Performance evaluation

In this section we evaluate the performance of our underlying topology-aware peer selection algorithm. We have a testbed with four PCs and four laptops interconnected by some Linksys switchers. One PC with Linux 2.4.10 operation system runs a tracker. With the help of virtual machine technology, we run three virtual machines with Linux operating system as routers on each PC (the remained three PCs with Windows 2003 operation system). By configuring static routes, we create an IP network consisting of one ISP network and three enterprise networks as illustrated in Fig.3.

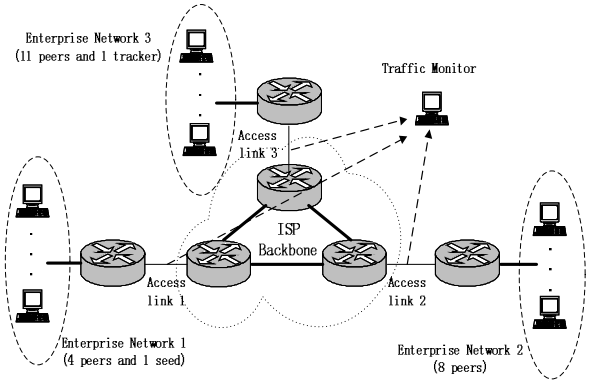

Fig.3. the topology of the testbed

We call the link connects the ISP network and enterprise network 1 as access link 1, similarly, the link connecting the ISP network and enterprise network 2 is called access link 2 and the link connecting the ISP network and enterprise network 3 is called access link 3. Finally, we configure each access link with 50ms delay.

In order to run our experiment with a relatively large number of peers, we configure five IP addresses in each laptop and bind each peer with a particular IP address. The total number of peers is twenty-four (including the seed): fifteen peers running on three laptops and nine running on three PCs with Windows 2003 operation system. There are five peers (include one seed) in enterprise network 1, eight peers in enterprise network 2 and eleven peers in enterprise network 3. The tracker is located at enterprise network 3. The peers arrive in one minute's interval (flash crowd model, since flash crowd generates the highest rate of traffic which is the most challenging for ISPs to handle).The upload speed of peers is 100kB and the upload speed of seed is 500kB.

We implement UTAPS in XBT [18] (an open source implementation of BitTorrent protocol, written in C++). We retain some degree of randomization in peer selection to prevent hot-spots from developing pathologically. As the number of peers in our experiment is small (24 peers), we set the length of peer list as four (the default value is 50). When the tracker receives request from a peer, it first tries to select three peers by UTAPS, and then select the last one randomly.

We make a torrent for a file having a size of 20MB (The size of file to be delivered is not important because we can define different upload speed to make the experiment finish in an appropriate interval) and test the individual performance in sense of download time and ISP backbone resources usage in sense of traffic injecting into ISP backbone.

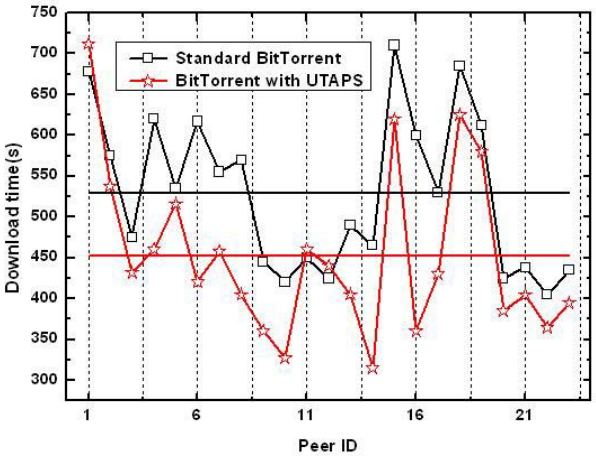

Fig.4. the download time of Standard BitTorrent V.S. BitTorrent with UTAPS

The two solid beeline in Fig.4 show the average download time of peers in our experiment for both standard BitTorrent system and BitTorrent system with UTAPS. We can see that the average download time has been decreased by 14.51% (change from 528.6s of standard BitTorrent to 452.7s of BitTorrent with UTAPS). It means that the enhanced BitTorrent system with UTAPS achieves better individual performance

and user experience than standard BitTorrent system. For a single peer, UTAPS can find neighbor peers with low RTT and tend to avoid low speed access link by find peers within small hops counts; therefore, the connections between Neighbor peers have lager transmission speed than those in standard BitTorrent system.

Fig.4 also plots the download times of every peer in our experiment. With UTAPS BitTorrent enables a greater percentage of peers to download the whole file with shorter delays than standard BitTorrent. Peers with ID 1, ID 11 and ID 12 have a relatively longer download time in BitTorrent system with UTAPS. The reason may be that the tracker hasn't gotten enough knowledge of underlying topology and made a relatively bad peer selection (comparing with the experiment in standard BitTorrent system) with random peer selection strategy when these peers join.

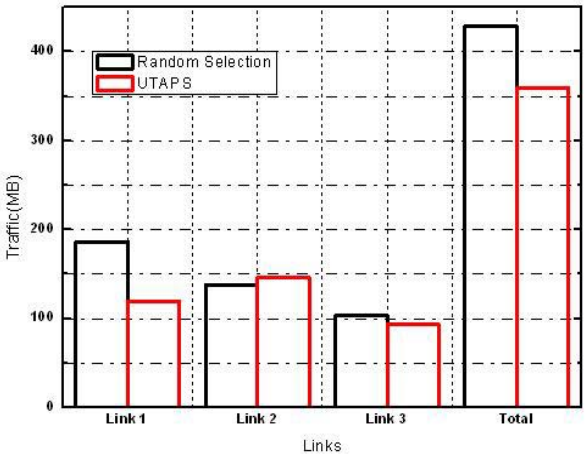

Fig.5. the amount of traffic traversing the access links

Fig.5 shows the total amount of the traffic injecting into the core network through the access link. The usefulness of this metric has two folds: firstly the access link tends to have less bandwidth than in LAN or backbones. If access link can be avoided, then better individual performance can be achieved. Secondly, fewer loads transmitted through the access link means the BitTorrent injects less traffic into ISP network and make an more efficient use of ISP's resources. From table Ⅱ we can see that the traffic injecting into the core network has been reduced due to UTAPS (change from 428.3M to 359.3M). This is because UTAPS try to avoid access link by finding peers within least hops and peers in the same enterprise network have a high possibility of being chosen.

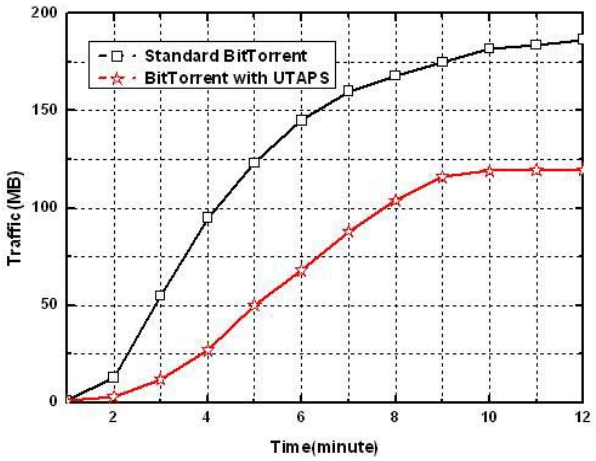

Fig.6. the cumulative traffic of Standard BitTorrent V.S. BitTorrent with UTAPS injected into access link 1

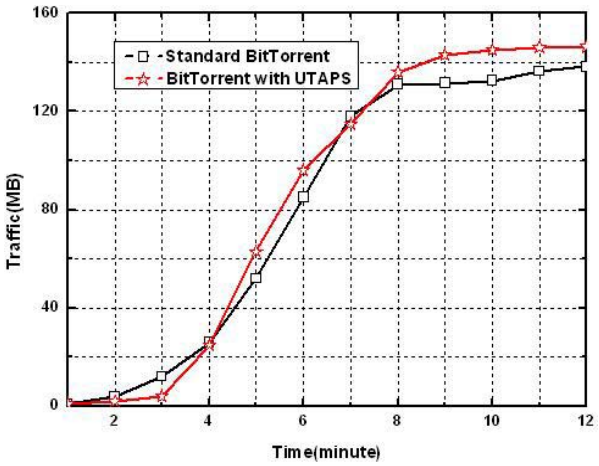

Fig.7. the cumulative traffic of Standard BitTorrent V.S. BitTorrent with UTAPS injected into access link 2

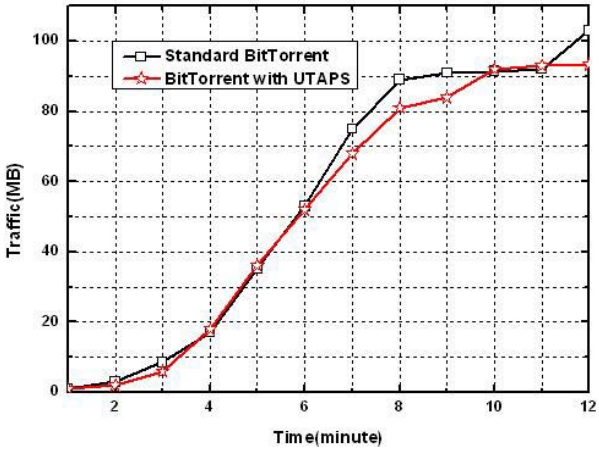

Fig.8. the cumulative traffic of Standard BitTorrent V.S. BitTorrent with UTAPS injected into access link 3

In Fig.6, Fig.7 and Fig.8, the cumulative traffics on the three access links are presented. Fig.6 plots the cumulative traffic on access link 1 and we can see that the traffic is remarkably reduced. Since the seed is place in enterprise network 1, with UTAPS, peers in the enterprise network 1 have little chance to download file fragment from peers out of enterprise network 1. However, with random peer selection, the peers in enterprise network 1 have equal possibility to choose peers outside.

Fig.7 and Fig.8 plot the cumulative traffics on access link 2 and access link 3. In these two figures, the reduction of traffic is not remarkable and there is even a small increase in traffic on access link 2. Since, the peers in enterprise network 2 and 3 have to download from peers (seed) in enterprise network 1 at the beginning of download. The whole download time is relative short and the number of peers is small, therefore, the effect of UTAPS in access link 2 and 3 is not obvious. However, when we consider the total amount of traffic injected into ISP network, the traffic reduction by UTAPS is very obvious.

In our prototype system, the Traceroute operation in UTAPS will lead to some additional overhead which is the fundamental flaw in our system. In our experiment we have not monitor the amount of this overhead separately, but from the comparison results of the traffic injected into the ISP backbone, we believe that the overhead caused by Traceroute operation is in the acceptable range and will not degrade the performance of file downloading.

5. Summery and future work

In this paper, we propose a novel peer selection algorithm called UTAPS which can select peers within the range of low RTT and small hop-counts by utilizing the underlying topology information. Traceroute is used to exploit the underlying topology by both the tracker and peers. The tracker uses the results of tracerouting to form the RTT-sights and hop-sights for all routers it knows. When receiving a request from a peer, UTAPS searches iteratively in the RTT-sights and hop-sights of routers to find pees within a range of low RTT and small hop-counts. Experimental results shows that by replacing the random peer selection algorithms with UTAPS in standard BitTorrent system, we achieve better individual performance in download time and efficient use of resources in ISP's backbone in sense of traffic injected into ISP network.

Our future work will focus on evaluating the performance of different strategies on utilizing the RRT-sights and hop-sights in Internet environment to gain further insight.

6. References

[1] Cohen B. Incentives builds robustness in Bit Torrent. First Workshop on Economics of P2P Systems. Berkeley, CA.2003.

[2] http://www.bigchampagne.com/

[3] CacheLogic. Cachelogic - advanced solutions for peer-to-peer networks. http://www.cachelogic.com.

[4] D. M et al. Can Network Coding Help in P2P Networks? International Symposium on Modeling and. Optimization in Mobile, Ad Hoc and Wireless Networks, Boston, Massachusetts, April 2006.

[5] [P. Felber and E. W. Biersack. Self-scaling networks for content distribution. International Workshop on Self-star Properties in Complex Information Systems, Bertinoro, Italy, May-June 20

[6] A. R. Bharambe, C. Herley, and V. N. Padmanabhan.Analysing and improving bittorrent performance. IEEE Infocom'2006, Barcelona, Spain, April 2006.

[7] A. Bharambe, C. Herley, and V. N. Padmanabhan. Understanding and deconstructing bittorrent performance. SIGMETRICS, Banff, Alberta, Canada, 2005.

[8] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis and modeling of bittorrent-like systems. The Internet Measurement Conference Berkeley, CA, USA, 2005.

[9] Arnaud Legout, Guillaume Urvoy-Keller and Pietro Michiardi. Rarest first and choke algorithms are enough. IMC'2006, Rio de Janeiro, Brazil 2006.

[10] Ruchir Bindal et al. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. IEEE International Conference on Distributed Computing Systems (ICDCS), Lisboan, Portugal ,July 2006

[11] T. S. Eugene Ng et al. Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems. IEEE INFOCOM'03, Orlando, Florida, USA, 2003.

[12] D.S. Bernstein et al. Adaptive Peer Selection.2nd International Workshop on Peer-to-Peer Systems, Berkeley, CA, USA, 2003.

[13] M. Hefeeda et al. PROMISE: Peer-to-Peer Media Streaming Using CollectCast. ACM Multimedia, Berkeley, CA, USA, 2003.

[14] P-Cube. P-cube: IP service control. http://www.pcube.net/indexold.shtml.

[15] M. Coates et al. Internet tomography. IEEE Signal Processing Magazine, 19(3), 2002.

[16] Simon G. M et al. A Genetic-Algorithm-Based Neighbor-Selection Strategy for Hybrid Peer-to-Peer Networks. ICCCN 2004, Chicago USA,2004

[17] Pei G,Gerla M. Fisheye State Routing : A Routing Scheme For Ad Hoc Wireless Networks. IEEE ICC'00, New Orleans, USA, June 2000.

[18] http://xbtt.sourceforge.net/