Performance Evaluation of the Advanced Network Tracker for BitTorrent

Shakir James, sjames@wustl.edu (A project report written under the guidance of Prof. Raj Jain)

Download

Abstract:

BitTorrent, a popular Peer-to-Peer (P2P) protocol, has recently engendered considerable controversy. Many P2P applications implement a variant of the protocol due to its cost-effective scalability. However, for Internet Service Providers (ISPs) the protocol is notorious for its high network cost. To reduce these costs, ISPs have deployed network devices to detect and throttle BitTorrent traffic. In response, application developers have begun encrypting the protocol's traffic to avoid ISP-detection. These actions perpetuate a "cat and mouse" game between ISPs and developers. Recent proposals to reduce BitTorrent's network cost involve either changing the protocol or deploying ISP-managed services. These proposals require broad adoption by developers to significantly benefit ISPs. In this paper, we propose a solution that is universally applicable to all BitTorrent clients and provides savings for ISPs out of the box.

We have developed a high performance network device, called the Advanced Network Tracker (ANT), which reduces BitTorrent network cost without changing the protocol. Moreover, ANT requires no explicit communication between client applications and ISPs. It inspects packets on the wire to identify BitTorrent traffic to discover local peers sharing similar files. To reduce network costs, ANT localizes traffic to an ISP's network by informing local peers about each other. In this work, we implement a working prototype of ANT using Network Processors (NP) for high throughput packet processing. In addition, we evaluate our implementation on the Open Network Lab's [ONL] NP-based programmable routers. Our evaluation shows ANT has a considerable impact on cross-ISP traffic. And equally important, it has a negligible impact on client performance.

Keywords:

Network architecture, network processors, traffic management, performance evaluation, peer-to-peer (P2P), BitTorrent

Table of Contents

---

1. Introduction

Peer-to-Peer (P2P) systems promise cost-effective file distribution for content providers on the Internet. In a P2P system, each peer downloads as well as uploads data to other peers. Thus, P2P systems are self-scaling because their bandwidth capacity increases as the number of peers increase. Compared to the client-server model or content distribution networks, P2P systems are cheaper for content providers since they do not require an investment in dedicated servers. Furthermore, P2P systems are better able to recover from failures since each peer holds some piece of the file. This paper focuses on the BitTorrent P2P protocol. Researchers have shown that the BitTorrent protocol is self-scaling [Yang04][Qiu04], robust [Legout06], and provides near optimal performance in terms of peer download time [Bharambe06].

Although the BitTorrent protocol provides cheap scalability for content providers, it is expensive for Internet Service Providers (ISPs). The protocol is oblivious to the underlying network topology, so its traffic traverse routes without concern for ISP economics. In fact, the protocol increases traffic on costly, cross-ISP link [Bindal06]. At the same time, P2P applications like BitTorrent increase revenue for ISPs because they encourage more broadband subscriptions. As a result, ISPs are left in a quandary: throttle BitTorrent traffic to reduce network cost at the risk of losing revenue from broadband subscriptions. ISPs have deployed traffic shaping devices [P-Cube][Sandvine] to "throttle" or limit BitTorrent traffic. Although the legality of traffic shaping is debatable, the practice degrades P2P application performance and can turn away potential customers.

Instead of reducing BitTorrent network cost, traffic-shaping devices have sparked an arms race between BitTorrent developers and ISPs. To evade identification by the first generation of these devices [P-Cube], some BitTorrent applications added header encryption to peer-to-peer messages. To counter, ISPs introduced new devices [Sandvine] that targeted other protocol (peer-to-tracker) messages. Recently there have been signs of an armistice, as ISPs have begun supporting initiatives to cooperate with developers to reduce BitTorrent network cost [Xie08]. Some authors [Bindal06][Xie08][Choffnes08] have proposed changes to the de facto protocol. Unfortunately, these collaborative initiatives have not been widely adopted.

As a new solution, we propose a network device, called the Advanced Network Tracker (ANT), which ISPs install on their network beside edge routers. What separates ANT from other proposals, is that ANT works with all BitTorrent clients and requires neither protocol changes nor direct application-ISP communication. Thus, ANT's novel contribution is solving ISPs' P2P quandary without requiring any extensions to the protocol. To reduce network costs, ANT keeps BitTorrent traffic within an ISP's local network.

We have implemented ANT on Intel's Internet eXchange Architecture (IXA), which uses a Network Processor (NP) called the IXP [IXP] for high performance and programmability. NPs are a programmable alternative to Application Specific Integrated Circuits (ASIC) for rapid development of high performance network devices. We used the specialized hardware units on the IXP, such as the hash unit, a Ternary Content ddressable Memory (TCAM), and high performance memory interfaces to enable high performance packet processing. We also used the hardware multithreading capability of the IXP's Micro-Engines (ME) to hide the latency of accessing of chip memory. To evaluate our implementation, we deployed ANT on the Open Network Lab's [ONL] IXP-based routers. Our evaluation shows that ANT has a meaningful impact on cross-ISP link utilization and an insignificant impact on client download time.

The remainder of the paper is organized as follows. Section 2 provides a brief background on relevant aspects of the BitTorrent protocol. Section 3 describes the design of ANT. Section 4 discusses interesting aspects of ANT's implementation on the IXP 2800, and on ONL in particular. Our evaluation results are presented in Section 5. In Section 6, we survey related work. We consider future work in Section 7 and conclude in Section 8.

---

2. Background

BitTorrent is an extremely popular P2P file distribution protocol that is not formally standardized. The original BitTorrent client, called the mainline client, was open-source. Other clients interoperated with the original client and each other by conforming to aspects of the "protocol" gleaned from its source code. A de facto standard consisting of dominant protocol features eventually emerged [Cohen08][BitTorrent]. However, many clients make non-standard extensions to the protocol such as Ono [Choffnes08], peer ISP-cache support, peer exchange protocol, and multitorrents. Interestingly, the mainline client is no longer open source and is now owned by a company called BitTorrent. Nonetheless, other existing clients have no incentive to adopt any new, official protocol changes. ANT is based on aspects of the de facto protocol, so it works with all clients.

Although many different clients exist, at a high level, they all follow the same basic steps when downloading a file. To begin downloading a file, a peer procures a metainfo file with a .torrent extension from a website. Among other fields the metainfo file contains the infohash field that uniquely describes the file and the Uniform Resource Locator (URL) that represents the address of the tracker, a server that aids in peer discovery. Using the metainfo file, the peer queries the tracker's URL with the infohash as a parameter. In its request, the peer also includes the TCP port number it is listening on for incoming peer connections. Then the tracker responds with a random list of IP addresses and port numbers of other peers that are sharing the file identified by the infohash. The peer attempts to connect to these other peers to download pieces of the file. Finally, once a peer obtains a piece of the file, it uploads that piece to other peers that need it. In BitTorrent jargon, peers sharing the same file are connected to the same torrent or participating in the same swarm. Also, a peer that has the complete file is called a seed and other peers without the complete file are called leechers.

As might be expected, the de facto protocol relates to the basic operation of the protocol. It specifies the content of the torrent file and the formats of the peer-tracker and peer-peer messages [Cohen08]. It also makes recommendations for a peer selection strategy, but it only mandates that the strategy selected "should" work with the strategy used by the mainline client and other peers using a similar strategy. ANT operates on the peer-tracker control messages. The control messages use the Hypertext Transfer Protocol (HTTP), which is layered above the Transmission Control Protocol (TCP). A peer's HTTP GET request embeds parameters in the URL, so ANT can capture parameters such as the infohash from a single peer-tracker request. The tracker responds to a peer requests with a plain text document containing a list of peer addresses and port numbers.

3. Design

ANT's primary objective is to reduce the network costs of the BitTorrent protocol for ISPs. We also considered the following goals in our design of ANT:

- Portability: It is essential that our solution work with all BitTorrent clients. To achieve this goal, our design only uses features of the de facto protocol. Therefore, we do not require any new additions to the protocol. As a result, ANT works out of the box when deployed in an ISP's network.
- Performance: We require ANT to scan network packets at line rate. For high-throughput packet processing, we designed thread safe data structures that are amenable to parallel execution. Moreover, we designed ANT to be modular by separating components that require high performance from slower components via a simple interface. Thus, it is easy to optimize the design for higher performance by reallocating processing resources and using buffers.
- Privacy: To allay users' fears that ISPs may be monitoring their P2P activities, neither BitTorrent clients nor application trackers should exchange information with an ISP-managed service. ANT should not explicitly communicate with user applications. Moreover, all data used by ANT should be gleaned from protocol messages.

First, we cover the overall system architecture of ANT. Then we provide succinct details on each component.

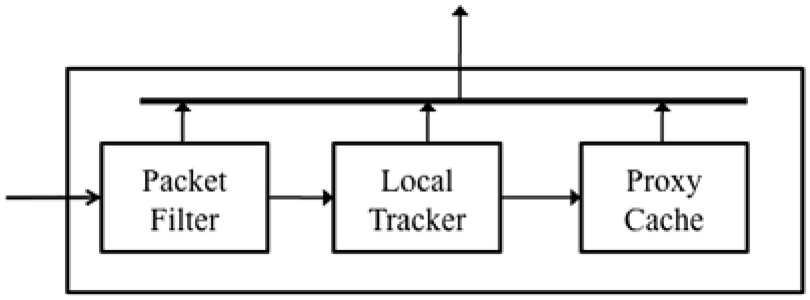

Figure 1. ANT system overview

```
PROCEDURE ant() {
  pass = packet_filter(info_hash, peer_address);

  IF (pass == 0) { // confirmed peer request
    pass = local_tracker(info_hash, peer_address);

    IF (pass == 0) { // new, local peer
     drop_packet(); // drop original peer request
      proxy_cache();
    }
  }

  IF (pass == 1) { // pass through
    send__packet();
  }
}
```

Figure 2. ANT algorithm

3.1 Component Architecture

ISPs will deploy ANT alongside edge routers to reduce BitTorrent traffic on costly routes to other autonomous ISPs. Figure 1 shows the three major components of ANT, and Figure 2 describes the ANT algorithm. First, the Packet Filter scans packets on the wire to identify peer-to-tracker protocol messages. Then the Local Tracker keeps track of the local peers connected to a particular torrent. Finally, the Proxy Cache responds to new, local peers in lieu of an application tracker. Unlike the application tracker's response, ANT's response contains only addresses of other local peers. By informing intra-ISP peers of each other, ANT encourages local peering that reduces costly, cross-ISP traffic.

3.2 Packet Filter

The Packet Filter identifies peer requests to trackers. It uses deep packet inspection to match data in packet header and payload. It uses a three-stage filtering process to distinguish peer requests from other packets. First, it probes the Internet Protocol (IP) header of the packet to single out TCP packets. (Recall that peer-tracker messages use the HTTP protocol that runs on top of TCP.) Since BitTorrent trackers usually listen on TCP port 6969, a test for this port number is used as the second level criterion. In the TCP payload, the start of the HTTP message body contains a static character string: "GET /announce?infohash" representing a request to a tracker. Thus, a test for the presence of this string is used as a final criterion for marking a packet as a peer request to a tracker. After identifying a request, the infohash and the peer's TCP port number are extracted from the packet payload using a simple Discrete Finite Automata (DFA). The peer's IP address (from the IP header) and TCP port number (from the TCP payload) are collectively referred to as the peer address. If a packet is identified as a peer request, the peer address, infohash, and a reference to the packet are passed to the Network Tracker. Otherwise, the packet is allowed to pass through.

3.3 Local Tracker

Using the infohash and peer address, the Local Tracker keeps track of torrents on an ISP local network. Figure 3 shows the data structure, called the tracker table. A hash lookup, using the infohash as a key, determines the appropriate hash table entry for the peer address. (It assumed that all peer addresses are local.) Each table entry consists of a bloom filter [Broder04] that stores the addresses of known peers for a torrent. The bloom filter is a fixed size, bit array that tests whether a peer address is a member of the known set of peers. Similar to a hash table, the filter's search time is O(1). However, the filter uses less space than a hash table at the cost of the possibility of false positives. We ensure, by design, a very low (less than 0.5%) false positive probability. A full description of the bloom filter is beyond the scope of this report. If the peer address was previously unknown then the Local Tracker passes the peer address and a reference to the original packet to the Proxy Cache. Otherwise, the packet is passed along to its destination.

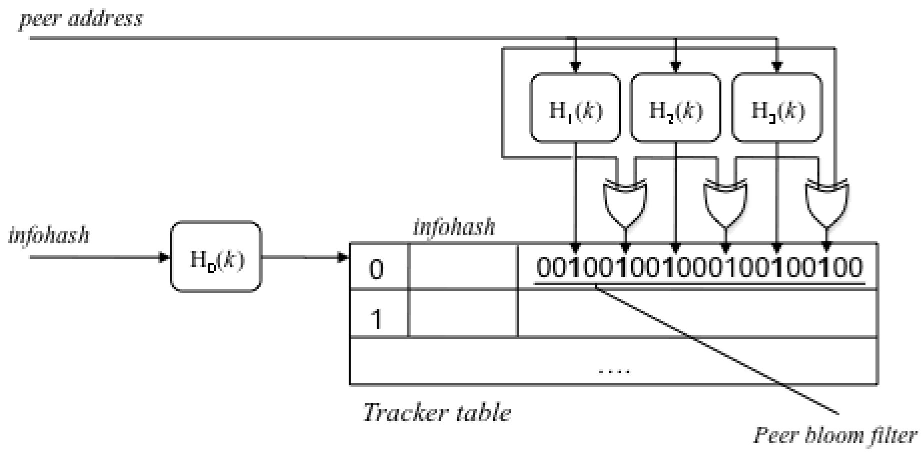

Figure 3. Local Tracker data structure

3.4 Proxy Cache

A packet that arrives at the Proxy Cache has been identified as a peer request (by Packet Inspection) from a new, local peer (by the Local Tracker). The Proxy Cache masquerades as application-level tracker by responding to the new peer in lieu of the application tracker. First, it adds the peer address to a torrent's byte string. The byte string consists of a list of peer address formatted according to the protocol specifications [BitTorrent]. Then it drops the packet carrying the original peer request. Lastly, using the byte string and fields in the original packet IP and TCP headers, it sends a "tracker" response to the peer that consists of only local peers. Since local peers are usually on higher capacity routes than external ones and most implementations of the BitTorrent peer selection algorithm favor peers on high capacity routes [Cohen08], the Proxy effectively biases the peer selection algorithm toward local peers. This "biased neighbor selection" technique is not novel. Other work [Bindal06][Xie08][Choffnes08] use some form of this biased peer selection method to improve locality and locality enhancing schemes for BitTorrent that have been shown to be beneficial to ISPs by [Karagiannis05].

In summary, ANT consists of three components: Packet Filter, Local Tracker, and Proxy Cache. The Packet Filter identifies peer requests to trackers. Next, the Local Tracker determines if the peer is local and previously unknown. Lastly, if the peer is local and unknown, the Proxy Cache responds to the peer with a list of other known local peers sharing the same file.

---

4. Implementation

To identify peer requests, keep track local peers, and respond to peers, ANT needs to process packets at line rate. For high performance packet processing, we implemented our design on a Network Processor (NP). We chose Intel's NP, the IXP, because the research community is developing publicly accessible, programmable testbeds based on the IXP [Turner07][Wiseman08]. Although there are other commercially available NPs [Cavium][RMI], we are unaware of similar publicly accessible platforms based on them. Furthermore, [Turner07] showed that NP based frameworks outperform those based on general-purpose processors such as Click [Kohler00]. Thus, we chose to implement our prototype on the IXP. In the following subsections, we give a brief overview of NPs in general. Then we provide some background on ONL, a publicly available NP testbed, and its NP-based router. Finally, we briefly cover ANT implementation as an ONL plugin.

4.1 Network Processors

In general, NPs such as the IXP include multiple processing cores, high bandwidth memory interfaces, and other hardware accelerators that make them more amendable to high-performance, programmable network devices. [Turner07] [Wiseman08] cover the general architecture of the IXP and its high performance features. In this paper, we only focus on the IXP's features that are relevant to our design. As shown in Figure 4, the IXP includes an XScale processor and 16 Mirco-Engines (MEs). The XScale is an ARM-based control processor that handles IP control packets such as Internet Control Message Protocol (ICMP) and Address Resolution Protocol (ARP) packets. The bulk of packet processing is done by the MEs. Four types of memories are available: ME-local memory, scratch memory, QDR SRAM, and DRAM. Each memory type has a different size and access latency. DRAM is the slowest but offers the most capacity. On the other end of the spectrum, local memory offers lowest latency but has the least capacity. An additional hallmark of NPs is the presence of hardware accelerators that speed up common networking tasks such as a Hash Unit for hash functions, Cyclic Redundancy Check (CRC) unit for CRC checksums, and a TCAM for associate array lookups. Currently, our implementation targets the 16-core IXP2800, but we plan to migrate to the 40-core version when it is made available in 2009 [Neugebauer08].

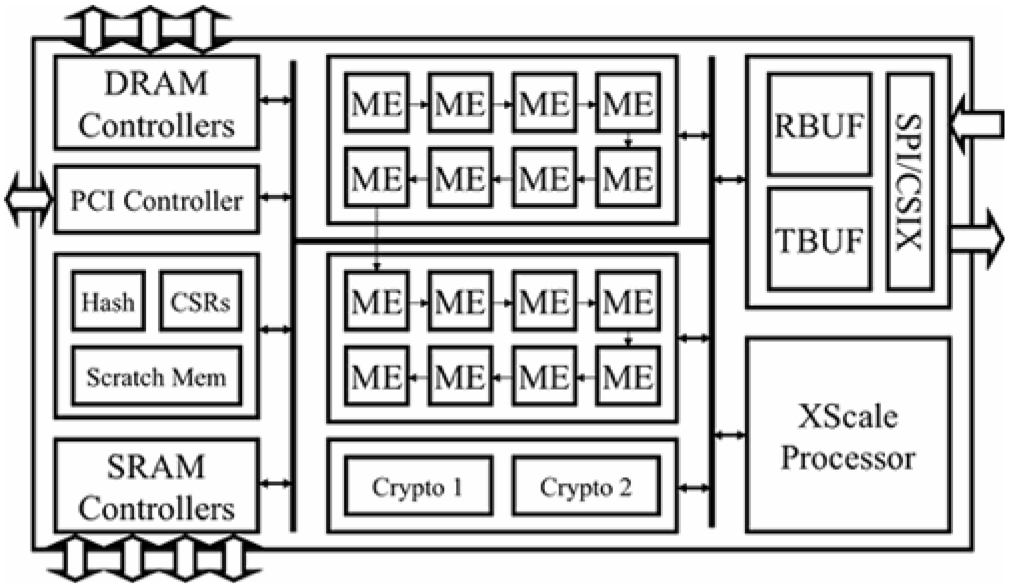

Figure 4. IXP Block Diagram

4.2 Open Network Lab

We deploy ANT on Open Network Lab (ONL), which is a publicly available IXP-based network test-bed that shares many features of PlanetLab's NP router [Turner07]. As a result of the similarity between the two NP platforms, we can easily port ANT to PlanetLab , a larger scale testbed, once [Turner07] becomes available. ONL's Remote Laboratory Interface (RLI) provides a Graphical User Interface for constructing various network topologies. Its NP routers (NPR) are based on the IXP 2800. Figure 5 shows the software structure and data flow of the NPR. As depicted in the figure, five ME are allotted to plugins. User code can be dynamically loaded on to these plugin MEs using the RLI. ANT runs as a plugin on the NPR as discussed in Section 4.4. Additionally, the platform provides a plugin Application Programming Interface (API) to simplify development. The API provides a library of helper functions that hide the complexities of performing common operations on packet data. Some examples of helper functions include functions for reading packet headers from DRAM into local memory, calculating packet checksums, and incrementing status counters.

4.3 ONL Network Processor Router

It is also helpful to understand the typical life-cycle of a packet in the NPR. Figure 5 shows the flow of packets through the NPR. First, the Receive block (Rx) places an incoming packet into DRAM and passes a reference to the packet onto the Multiplexer block (Mux). Then Mux initializes the packet metadata in a specific segment of SRAM and passes the reference to the Parse, Lookup, and Copy (PLC) block. PLC uses fields in the packet header to form a lookup key, which is used to match user-defined filters or specific routes stored in the TCAM. (Users can add filters to direct packets to particular queues, output ports, or plugins.) Based on the lookup result, PLC passes the packet reference to ither the Queue Manager (QM) block, the XScale, or the plugins. The QM places the reference on one of the per-port queues and uses a simple Deficit Round Robin (DRR) scheduling algorithm to select a reference to pass to the Header Format (HF) block. The HF block adds necessary Ethernet header fields and sends it to the Transmit block (Tx). Tx simply reads the referenced packet data from DRAM and sends it out on the external link. Finally, a Stats block (Stats) keep track of system counters that can be used for debugging. [Wiseman08]

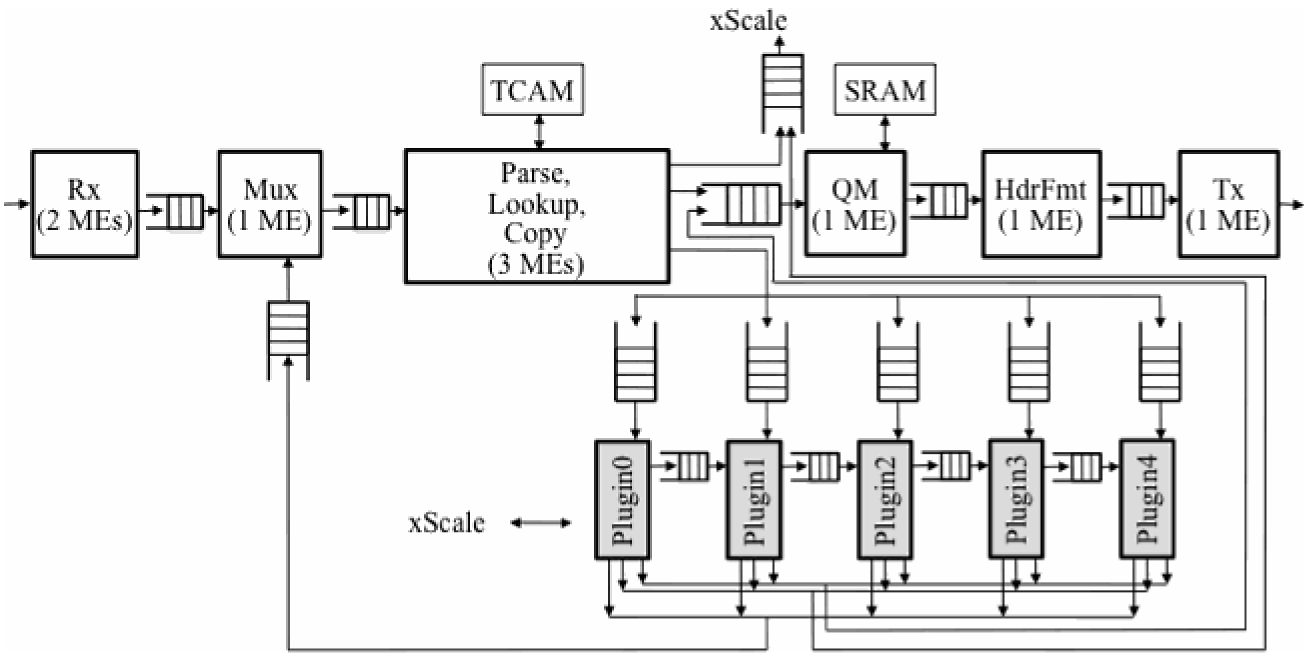

Figure 5. ONL NP router (NPR) software architecture

4.4 ANT Plugin

Using the plugin API, we implemented ANT as an ONL plugin. In the RLI, we configure a TCAM filter to direct TCP packets destined to the known tracker port to our plugin. Our plugin processes packets and forwards them to Mux, so they can pass through PLC and be sent to the appropriate output port. ANT is implemented on one of the available plugin ME. In ANT, packet-processing times vary from one packet to the next. Some packets never get processed at all if they are not BitTorrent peer requests. Other packets may cause ANT to add entries to its data structures. Since the time it takes to process a packet varies, the unordered-thread execution model is most appropriate. In this model, threads take turns executing in a round-robin manner. Each thread explicitly passes control to another thread using hardware signals when it blocks waiting on a memory access. Since the processor usually is kept busy executing some thread, this effectively hides the memory latency. Furthermore, by virtue of the round-robin thread scheduling, packets in this model are processed in arrival order. ANT also uses the ME CRC to perform fast hash lookups.

To recap, ANT is implemented on a NP. NPs facilitate rapid development of high-performance network devices. We chose Intel's NP, the IXP, because an IXP-based testbed, called ONL, is publicly available. ONL's NP routers, NPRs, can be dynamically loaded with user code via the plugin environment. ANT is implement as an ONL plugin.

---

5. Evaluation

To evaluate the performance of our implementation, we first state our goals and define the system boundary. Then we select the metrics based on the system services. From a list of parameters that affect performance, we select the factors that will be studied. After selecting the factors, we choose appropriate workloads. We design experiments based on the metrics and factors that offer maximum information for minimal effort. Finally, we interpret the data and present the results.

5.1 Goal and Metrics

The goal of our evaluation is to compare the cross-ISP traffic of standard application tracker to ANT. To this end, the system under test is the cross-ISP link. The system consists a torrent with intra-ISP peers and inter-ISP peers connected via a cross-ISP link. To ensure that all peers receive some amount of data via the cross-ISP link, we place a seed on one end of the cross-ISP link and all leechers on the other end. All other effects of components external to the system were minimized.

To determine the performance criteria or metrics, we consider the system's services. The primary function of the cross-ISP link is to transfer data from one ISP to the other. In other words, the cross-ISP link will transfer data from the seed to the leechers. Thus, an appropriate metric is link utilization, which we define the ratio of the number of bytes transferred to the total file size. For example, for a 1MB file, if 1MB of data is transfer over the cross-ISP link then the link utilization is 1. To measure ANT's impact on client performance, we include client download time as our second metric. Due to resource constraints, we assume that all peers, both the seed and leechers, remain connected to the torrent even after downloading the complete file. We further assume error free operation. Under these conditions, the rate at which the service is performed and the resources consumed by the service will be compared. (We do not consider delay and availability of packets sent due to resource constraints.) Thus, two performance metrics are of interest: throughput and link utilization.

5.2 Factors

After determining the metrics, we decide on the factors by considering the parameters that affect the performance cross-ISP link and the workload. The parameters that affect the cross-ISP link, the system, are fixed since they depend on the physical medium used to transmit bits. The workload parameters include:

- Operating system running on end hosts
- Memory of end hosts
- CPU of end hosts
- Speed of host's Ethernet interface
- BitTorrent application
- Number of leechers
- Number of seeders
- Number of torrents
- Limit on upload and download rate of leechers (set on client software)
- Limit on upload rate of seeders (set on client software)
- File size
- Other load on the cross-ISP link

From these workload parameters we select the following key factors to study:

- Type of tracker: application tracker and ANT
- Number of leechers: 8, 16, 32, ..., 512, 1024
- Type of peer: university node (2Mb/s up and 2Mb/s down), cable modem (128Kb/s up and 1Mb/s down)
- Upload rate of seeders: 128Kb/s, 256Kb/s, 384 Kb/s
- File size: 1MB, 8MB, 16MB, ... , 512MB

Resource availability and our sponsors' interest dictated the choice of parameters. Upload and download rates were chosen to model the asymmetric link capacities of "cable modem" users (128/s upload and 1Mb/s) and the symmetric links model the so-called "university nodes" (2Mb/s upload and download) [Bindal06]. The seeder upload rate was limited to 1, 2, and 4 times that of cable model download rates [Bindal06]. All other parameters were fixed. Thus, our result will only be valid for one physical memory, end host, operating, and

BitTorrent client. (Each host on ONL is a 1 GHz AMD Opteron machine with 512 MB RAM and 1 Gb/s Ethernet interface. The hosts run Linux version 2.6.21. Using version 4.0.2 of the mainline BitTorrent client.) Our experiments have exclusive access to the network testbed, so other network load is negligible.

5.3 Evaluation Technique and Experimental Design

Now that we have selected the metrics and factors, we decide on an evaluation technique. A prototype of ANT is currently available, so measurement is the obvious choice for one of our evaluation techniques. The ONL testbed will be used to instrument our prototype. Using the ONL's RLI, we can construct different network topologies and monitor performance metrics such as link traffic. The accuracy of the RLI's monitoring tools has been extensively verified.

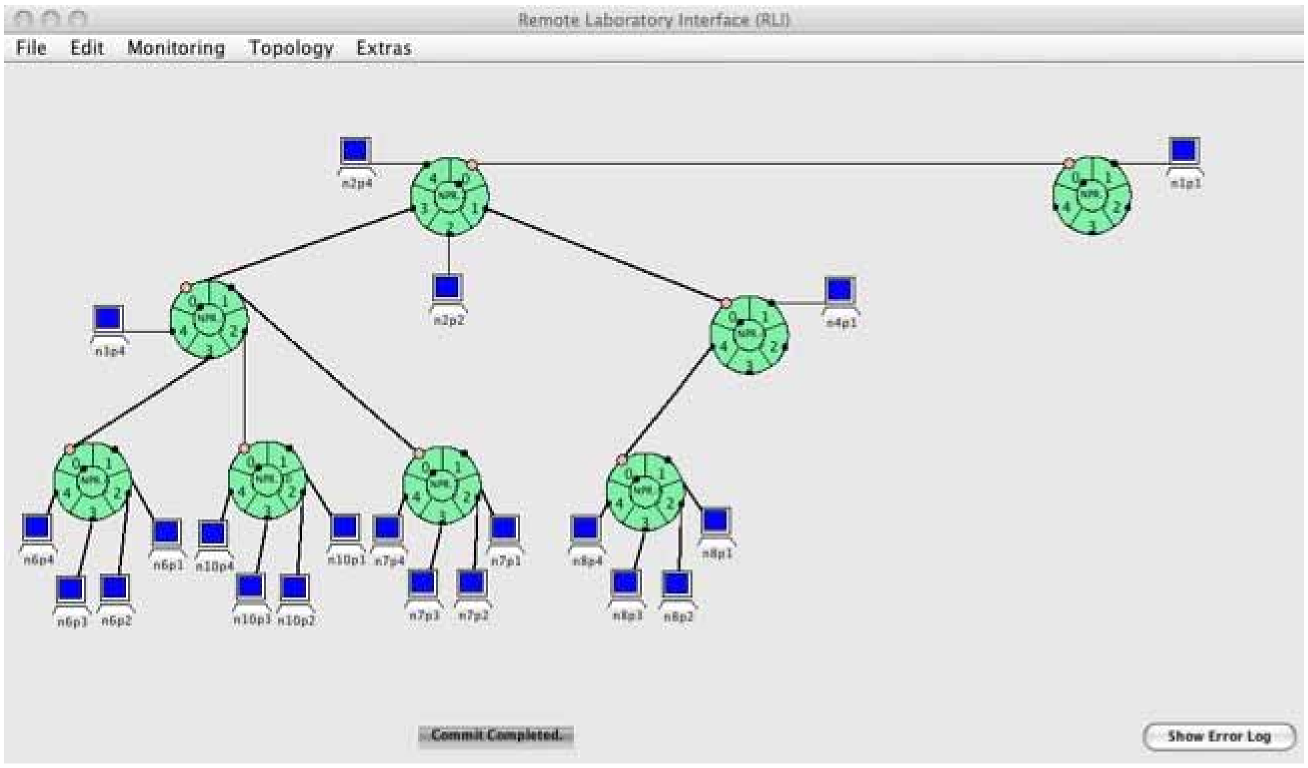

Figure 6. Evaluation topology (in ONL's Remote Labratory interface)

Our evaluation technique dictates our workload choice. For measurement, we use scripts to generate representative workloads. Due to resource constraints, we only consider a "flash crowd" workload, which is representative of a situation where a popular file is made available for the first time [Bindal06]. A flash crowd workload is the most taxing situation for the cross-ISP link. To model a flash crowd, our scripts start the seed then launch all leechers a few seconds after.

Finally, we design an experiment to generate maximum information with minimum effort. We select a 5^{k-p}experimental design k = 1, p = 2. This reduces the number of experiments by from 32 to 16. Table 7 shows the factors and levels in the design. The 2^{5-1} design uses the generator $I = ABCDE$. Based on the results of this investigation we determine which factors account for the majority of variation.

Symbol	Factor	Level -1	Level 1
A	Tracker	Standard	ANT
B	Leechers	8	19[1]

C	Peer	University[2]	Cable[3]
D	Seed	128 Kb/s up	384 Kb/s up
E	Filesize	1 MB	8 MB

[1] Limit of 19 is due to ONL resource constraints
[2] Real upload and download rate (in hardware) is 1.366Mbps
[3] Real download rate (in hardware) is 2.732Mbps

Table 7. Factors and Levels in the Experimental Design

5.4 Results

Based on the level assignments for the five factors, the measured link utilizations or the flash crowd workload is shown in Table 8. The effects and the percentage of variation explained for link utilization and download time are shown in Table 9.

Exp. #	*A*	*B*	*C*	*D*	*E*	LU	DL
1	-1	-1	-1	-1	1	1.54	788.13
2	1	-1	-1	-1	-1	1.32	74.55
3	-1	1	-1	-1	-1	5.94	373.98
4	1	1	-1	-1	1	1.65	824.56
5	-1	-1	1	-1	-1	4.05	248.22
6	1	-1	1	-1	1	1.60	862.62
7	-1	1	1	-1	1	2.53	1189.69
8	1	1	1	-1	-1	2.70	36.92
9	-1	-1	-1	1	-1	3.09	72.87
10	1	-1	-1	1	1	1.11	182.42
11	-1	1	-1	1	1	1.48	259.06
12	1	1	-1	1	-1	1.57	36.92
13	-1	-1	1	1	1	2.44	407.85
14	1	-1	1	1	-1	1.85	245.87
15	-1	1	1	1	-1	4.62	119.14
16	1	1	1	1	1	4.01	767.68

Table 8. Measured link utilization (LU) and download time (DT) for the Tracker study.

Confounded Effects		**Link Utilization**		**Download Time**	
1	**2**	**Estimate**	**% VAR**	**Estimate**	**% VAR**
I	*ABCDE*	2.59		405.65	
A	*BCDE*	-0.62	20.8	-26.71	0.6
B	*ACDE*	0.47	12.0	45.34	1.7
C	*ABDE*	0.38	7.9	79.09	5.1
D	*ABCE*	-0.07	0.3	-144.18	16.9
AB	*CDE*	0.04	0.1	-7.76	0.0
AC	*BDE*	0.18	1.8	20.24	0.3

AD	*BCE*	0.23	2.9	73.46	4.4
BC	*ADE*	0.02	0.0	-1.73	0.0
BD	*ACE*	-0.07	0.3	-11.12	0.1
CD	*ABE*	0.33	5.8	44.57	1.6
DE	*ABC*	0.29	4.5	-111.82	10.1
CE	*ABD*	0.22	2.6	67.61	3.7
BE	*ACD*	-0.10	0.5	54.66	2.4
AE	*BCD*	0.66	24.0	25.78	0.5
E	*ABCD*	-0.55	16.4	254.60	52.6

Table 9. Effects and Variation Explain in the Tracker Comparison Study

Looking the percentage of variation explained in Table 9, the effects that impact link utilization are the type of tracker (*A*), file size (*E*), and *AE*. This is promising because it means ANT may have a significant impact on link utilization. Furthermore, for download time, the type of tracker (*A*) accounts for less than 1% of the variation. This also shows promise because it means that ANT may not have a significant impact on client performance. Not surprisingly, the important factors for download time are file size (*E*) and seed upload rate. File size, type of peer, and seed account for approximately 80% of variation in download time.

To sum up, we defined our system under test ad the cross-ISP link, selected link utilization and download time as metrics, and settled on a list of five factors to study at various levels. To use of time efficiently, we designed 25-1 experimental design to isolate the factors that account for the most variation.

6. Related Work

There are many proposals to reduce BitTorrent network costs [Bindal06] [Xie08] [Choffnes08] [Aggarwal07]. In general, these proposals either require explicit communication with an ISP-managed service or significant changes to the BitTorrent protocol. All previous proposals use an application layer technique to bias the selection of local peers. [Bindal06] shows a minimal deterioration in client performance (download time), and the others [Xie08] [Choffnes08] [Aggarwal07] show improvement. Unlike other locality enhancing proposals, ANT requires no changes to the protocol, and thus works with all clients. Our local "tracker" solution also merges duplicate torrents. Duplicate torrents are disjoint swarms; i.e. distinct groups of peers sharing the same file but connected to different application trackers. [Neglia07] showed that duplicate torrents are common. Therefore, ANT provides greater opportunity to exploit locality than some application-level solutions such as [Bindal06].

7. Future Work

A significant amount of work remains to be done to evaluate ANT's benefits and performance. First, we must address the obvious question: are our results applicable to real world swarms? We believe that by finding the average size of a typical BitTorrent swarm and re-running our experiments with this average swarm size will make our proposal more convincing. Additionally, an analytical model needs to be developed to explain our measurement results. ANT's network locality enhancing technique will be compared to other application schemes such as [Bindal06]. (It will also be beneficial to measure the benefits of ANT with different types of BitTorrent clients.) In terms of performance, ANT's impact on tracker response time was omitted. We will consider error metrics such as packet retransmissions. In the future, we hope to deploy ANT on PlanetLab's

NP platform [Turner07] when it becomes available. This will validate our results on a larger scale.

---

7. Concluding Remarks

In this paper, we have demonstrated a high-performance, network-based solution to the ISPs' quandary of controlling BitTorrent network costs. We presented a solution that is universally applicable to all BitTorrent clients because it depends only on features of the de facto protocol and hence requires no client extensions. This shows that ISPs can deploy devices that constructively alter BitTorrent behavior as opposed to devices that throttle or block BitTorrent traffic. Furthermore, our device does not require broad adoption in the BitTorrent developer community to provide savings for ISPs. To date, we are unaware of any other solution that can make this claim. Our evaluation shows that our device has a significant impact on cross-ISP link utilization and a minimal impact on client performance.

Acknowledgements

We are grateful to the staff of the Applied Research Lab at Washington University in St. Louis for helping us work through our many implementation bugs.

References

[ONL] Open Network Lab. http://www.onl.wust.edu, 2008. http://www.onl.wust.edu

[Yang04] Yang, X. and G de Veciana, "Service capacity of peer to peer networks," In Proc. of IEEE INFOCOM, 2004. http://www.ieee-infocom.org/2004/Papers/46_3.PDF

[Qiu04] Qiu, D. and R. Srikant. "Modeling and performance analysis of BitTorrent-like peer-to-peer networks." In Proc. of ACM SIGCOMM, 2004. http://portal.acm.org/citation.cfm?id=1015508

[Legout06] Legout, A., G. Urvoy-Keller, and P. Michiardi. "Rarest First and Choke Algorithms are Enough." In Proc. of Internet Measurement Conference (IMC), 2006. http://portal.acm.org/citation.cfm?id=1177106

[Bharambe06] Bharambe, A., C. Herley, and V. Padmanabhan, "Analyzing and Improving a BitTorrent Networks Performance Mechanisms," In Proc. of IEEE INFOCOM, 2006. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4146981

[Bindal06] Bindal, R., P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," In Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS), 2006. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01648853

[Broder04] Broder, A. and Mitzenmacher, A. "Network Applications of Bloom Filters: A Survey," Internet Mathematics, 2004, Vol. 1, No. 4: pages 485-509. http://www.eecs.harvard.edu/~michaelm/postscripts/im2005b.pdf

[P-Cube] P-Cube. P-Cube: Ip Service Control. http://www.p-cube.net/indexold.shtml

[Sandvine] Sandvine. Sandvine: Intelligent broadband network management. http://www.sandvine.com

[Xie08] Xie, H., R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider portal for (P2P) applications," In Proc. of ACM SIGCOMM, 2008. http://portal.acm.org/citation.cfm?id=1402999

[Choffnes08] Choffnes, D., and F. Bustamante, "Taming the torrent: a practical approach to reducing cross- isp traffic in peer-to-peer systems," In Proc. of ACM SIGCOMM, 2008. http://ccr.sigcomm.org/online/?q=node/403

[IXP] Intel IXP 2xxx Product Line of Network Processors. http://www.intel.com/design/network/products/npfamily/ixp2xxx.htm

[Cohen08] Cohen, B. The BitTorrent Protocol Specification, Feb 2008. http://www.bittorrent.org/beps/bep_0003.html

[BitTorrent] BitTorrent Specification Wiki, Bittorrent Protocol Specification v1.0. http://wiki.theory.org/BitTorrentSpecification

[Karagiannis05] Karagiannis, T., P. Rodriguez, and K. Papagiannaki. "Should internet service providers fear peer-assisted content distribution?" In Proc. of Internet Measurement Conference (IMC), October 2005.http://portal.acm.org/citation.cfm?id=1251092

[Turner07] Turner, J., P. Crowley, J. DeHart, A. Freestone, B. Heller, F. Kuhns, S. Kumar, J. Lockwood, J. Lu, M. Wilson, C. Wiseman and D. Zar. "Supercharging PlanetLab - a High Performance, Multi-Application, Overlay Network Platform," In Proc. of SIGCOMM, 2007. applications," In Proc. of ACM SIGCOMM, 2008. http://portal.acm.org/citation.cfm?id=1282391

[Wiseman08] Wiseman, C., J. Turner, et al., "A Remotely Accessible Network Processor-Based Router for Network Experimentation," ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2008. http://www.arl.wustl.edu/~scj1/papers/wiseman08.pdf

[Cavium] Cavium Networks. OCTEON CN38XX/CN36XX Multi-Core MIPS64 Based SoC Processors. http://www.cavium.com/OCTEON_CN38XX_CN36XX.html

[RMI] RMI: XLR Family of Thread Processors. http://www.razamicro.com/products/xlr.htm

[Kohler00] Kohler, E., R. Morris, B. Chen, J. Jannotti, and M. Frans Kaashoek. "The Click Modular Router," ACM Transactions on Computer Systems 18(3), August 2000, pages 263-297. http://pdos.csail.mit.edu/papers/click:tocs00/paper.pdf

[Aggarwal07] Aggarwal, V., A. Feldmann, and C. Scheideler,"Can ISPS and P2P users cooperate for improved performance?" In Proc. of SIGCOMM, 2007. http://portal.acm.org/citation.cfm?id=1273449

[Neglia07] Neglia, G., G. Reina, Z. Honggang, D. Towsley, A. enkataramani, and J. Danaher, "Availability in BitTorrent Systems," In Proc. of INFOCOM, 2007. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4215838

[Neugebauer08] Neugebauer, R. "Network Topology Offload," In Proc. of Xen Summit, http://www.xen.org/xensummit/xensummit_summer_2008.html, Boston, 2008.

List of Acronyms

ANT	Advanced Network Tracker
API	Application Programming Interface
ARP	Address Resolution Protocol
ASIC	Application Specific Integrated Circuits
CRC	Cyclic Redundancy Check
DFA	Discrete Finite Automata
DHT	Distribute Hash Table

DPI	Deep Packet Inspection
DRAM	Dynamic Random Access Memory
DRR	Deficit Round Robin
DTS	Deli Ticket Server
Gb	Gigibit
HF	Header Format
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISP	Internet Service Provider
IXA	Intel's Internet eXchange Architecture
IXP	IXA's network Processor
ME	Micro-Engines
Mux	Multiplexer block
NP	Network Processor
NPR	Network Processor router
P2P	Peer-to-Peer
PLC	Parse, Lookup, and Copy
QDR	Quad Data Rate
QM	Queue Manager
RLI	Remote Laboratory Interface
Rx	Receive block
SRAM	Static random access memory
TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol
Tx	Transmit block
URL	Uniform Resource Locator
