TnT: Transparent Network Tracker for P2P applications

Shakir James and Patrick Crowley
Applied Research Laboratory
Department of Computer Science and Engineering
Washington University in St. Louis
{sjames, pcrowley}@wustl.edu

ABSTRACT
Peer-to-peer (P2P) applications are voracious bandwidth consumers, and ISPs have no effective options for curbing their bandwidth consumption. The Transparent Network Tracker (TnT) is a network device that fills this void: it identifies and monitors P2P traffic on the wire to support applications that control it. In this paper, we describe our TnT prototype and an associated ISP tracker application.

Categories and Subject Descriptors
C.2.3 [**Computer–Communication Networks**]: Network Operations

General Terms
Management, Design, Experimentation.

Keywords
P2P, cross-ISP traffic.

1. INTRODUCTION
Peer-to-peer (P2P) applications are expensive and economical. For Internet Service Providers (ISPs), they increase network costs because peers randomly *share*–i.e., upload as well as download–data. Yet, for content providers, P2P applications are cheap because they scale without dedicated servers. Not surprisingly, P2P traffic now accounts for about 50% of Internet traffic [IP09].

ISPs have failed to control P2P data traffic. They installed network devices to detect and limit traffic, but developers used random ports and encryption to bypass them [CH08]. Instead of data traffic, TnT detects plaintext, control messages on known ports. It forwards these messages, which reveal content identifiers and peer addresses [BE10], to applications that control traffic.

We use a network processor (NP) to build a high-performance prototype of TnT for *BitTorrent*, which is the most popular P2P protocol [IP09]. On an experimental platform that includes tens of NP routers and hundreds of hosts [ONL], we use our prototype to forward messages to an ISP tracker application that aims to reduce costly, cross ISP traffic.

ANCS'10, October 25-26, 2010, La Jolla, CA, USA.

2. BACKGROUND
2.1 BitTorrent
The BitTorrent protocol uses a *metainfo file*, which describes the content file; *client* applications, which support the protocol; and a *tracker*, which servers as an index of peers. The metainfo file, obtained from a website, contains a hash of the content file and the tracker's URL. Clients use the metainfo file to request peers from the tracker and exchange pieces of the file with these peers.

2.2 Motivating Application
A peer randomly connects to other peers. This random peer-selection ignores ISPs' economics such as routing policy and peering agreements [XI08]. Even when local peers have 50%-90% of the data, peers download it externally and increase costly, cross-ISP traffic [KA05]. The tracker, a peer-discovery service, also selects a random set of peer addresses.

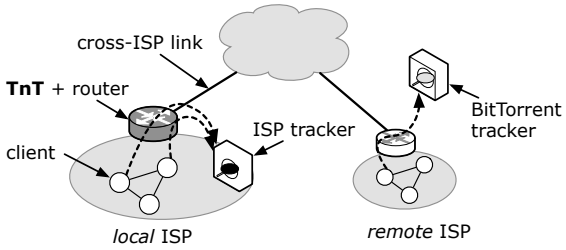


Figure 1. TnT redirects queries to the ISP tracker.

The ISP tracker, however, selects more local peers. And, TnT allows the ISP tracker to work without protocol changes. This is important because BitTorrent is not a standard protocol [BI06]. Figure 1 shows that TnT redirects tracker queries to the ISP tracker, which forges the network headers and responds to clients. A flag in its response tells developers the message came from their ISP, so they may set a flag in their query to bypass TnT.

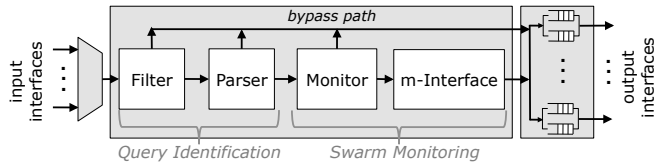
3. DESIGN


Figure 2. TnT identifies and monitors P2P traffic to support traffic control applications.

TnT identifies tracker queries to monitor swarm sizes. Figure 2 shows the system's components. The *Filter* uses port scanning

and string matching to detect tracker queries, and the *Parser* uses nondeterministic finite automata to extract values from them. The *Monitor* uses a hash table to track the growth of swarms, and the *m-Interface* forwards queries from relatively large swarms to a traffic-control application.

4. IMPLEMENTATION

We use Intel's IXP NP to build a high-performance prototype [IXP]. Each of the 128 threads passes control to one another in round robin order when it stalls on a memory access. This hides memory-access latency and keeps the NP busy. Besides multithreading, we use the IXP's hardware accelerators: the Hash unit for hash functions, cyclic redundancy check (CRC) unit for checksums, and, TCAM to associative array lookups. Using our NP prototype, we extend ONL's NP-router.

5. PRELIMINARY RESULTS

We aim to verify that TnT works without client support. To that end, we perform ONL experiments with different BitTorrent clients: the *Mainline* or the first client, a Python application; CTorrent, a C++ application [CT06]; Vuze, a Java application [VU09]; and *Mixed*, an equal mix of the three clients.

To determine the benefit of the applications that TnT supports, we measure the cross-ISP traffic of BitTorrent (BT) with and without the ISP tracker (+ISP). We construct a tree topology with two virtual ISPs (Figure 1), install TnT on the "local" ISP's edge NP-router, and run the ISP tracker on a host connected to this router. Our script creates a random data file, starts a "remote" client with the complete 1 GB file, then immediately starts 24 clients (12 "local") without the file in random order.

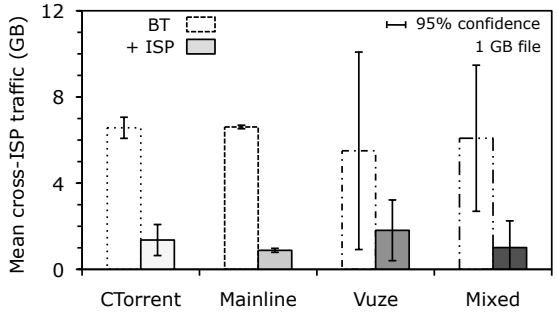


Figure 3. Cross-ISP traffic and client type.

Figure 3[1] shows the mean cross-ISP, traffic measurement of three repetitions. TnT works transparently and enables the ISP tracker to reduce cross-ISP traffic. The ISP tracker provides about an eightfold reduction in cross-ISP traffic for each client except Vuze. The overlapping confidence intervals for Vuze indicate that cross-traffic are the same at 95% confidence. Since Vuze also uses a distributed hash table (DHT) to find peers, this result is not surprising.

[1] Figure 3 and Table 1 also appear in [JA10] that focuses on the ISP Oracle (tracker) instead of TnT.

The ISP tracker also does not hurt application performance. It appears to maintain client download times. Table 1 shows that the differences in download times are negligible.

Table 1. Download times (s) for different clients1.

Client	50th percentile		95th percentile	
	BT	ISP	BT	ISP
CTorrent	154	157	162	161
Mainline	186	182	192	188
Vuze	159	166	165	173
Mixed	156	160	190	174

6. CONCLUSION

TnT provides ISPs autonomous options for reducing P2P network costs. Our measurements on real systems show that TnT works without protocol changes and allows the ISP tracker application to reduce costly, cross-ISP traffic for most clients by a factor of eight. Avenues for future study include identification of DHT control messages, field tests, a detailed evaluation of TnT's scalability, and support for other P2P protocols.

7. REFERENCES

[BE10] S. L. Blond, A. Legout, F. Lefessant, W. Dabbous, M. A. Kaafar, "Spying the World from your Laptop - Identifying and Profiling Content Providers and Big Downloaders in BitTorrent," in *LEET*, 2010.

[BI06] BitTorrent Specification Wiki, Bittorrent Protocol Specification v1.0. http://wiki.theory.org/BitTorrentSpecification

[CH08] D. Choffnes, and F. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," in *SIGCOMM*, 2008.

[CT06] CTorrent v1.4 DEVEL. http://ctorrent.sourceforge.net/

[IP09] Ipoque, "Internet Study 2008-2009," 2009. http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009

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

[JA10] S. James and P. Crowley, "IMP: ISP Managed Peer-to-peer" in *P2P*, 2010.

[KA05] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. "Should internet service providers fear peer-assisted content distribution?" in *IMC*, 2005.

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

[VU09] Vuze 4.2.0.4 BitTorrent client, formerly Azureus. http://azureus.sourceforge.net/

[XI08] H. Xie, R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider portal for (P2P) applications," in *SIGCOMM*, 2008.