Leveraging BitTorrent for End Host Measurements

Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson

Department of Computer Science and Engineering
University of Washington, Seattle, WA, 98195

Abstract. Traditional methods of conducting measurements to end hosts require sending unexpected packets to measurement targets. Although existing techniques can ascertain end host characteristics accurately, their use in large-scale measurement studies is hindered by the fact that unexpected traffic can trigger alarms in common intrusion detection systems, often resulting in complaints from administrators. We describe *BitProbes*, a measurement system that works around this challenge. By coordinated participation in the popular peer-to-peer BitTorrent system, BitProbes is able to unobtrusively measure bandwidth capacity, latency, and topology information for $\sim$500,000 end hosts per week from only eight vantage points at the University of Washington. To date, our measurements have not generated a single complaint in spite of their wide coverage.

1 Introduction

Detailed estimates of Internet path properties allow applications to improve performance. Content distribution networks (CDNs) such as Coral [1], CoDeeN [2] and Akamai [3] use topology information to redirect clients to mirrors that provide the best performance. Peer-to-peer services such as Skype [4] and BitTorrent [5] can use knowledge of both the core and the edge to optimize peer selection, improving end-user performance [6]. Overlay services such as RON [7] optimize routes based on metrics such as loss rate, latency, or bandwidth capacity to allow applications to select routes based on specific needs.

Although valuable, large-scale measurement of edge characteristics is encumbered by uncooperative or even hostile hosts. End hosts are often firewalled, making them unresponsive to active probing and ruling out many of the tools commonly used for mapping the Internet core. Further, administrators frequently mistake measurement probes for intrusion attempts, raising alarms in intrusion detection systems (IDSs). To avoid these problems, researchers have been forced to take alternate approaches when measuring end hosts. Systems such as PlanetSeer [8] take an *opportunistic* approach to end host measurement. PlanetSeer infers link properties by passively monitoring existing TCP connections between a centralized content provider and end hosts. By piggybacking measurements on existing, expected traffic, an opportunistic approach allows measurement of end hosts without triggering alarms in IDSs. Although successful, these systems are limited by the popularity of the service offered and require content providers to instrument servers with custom measurement software. This method has the additional drawback that the majority of data is transferred from the server to the end host, making accurate measurements of the end hosts upload capacity impossible.

Drawing inspiration from these, we consider an alternative platform for attracting measurement targets: the BitTorrent peer-to-peer system. BitTorrent has several desirable features for such a platform. It is extremely popular, suggesting wide coverage. Also, its normal operation involves sending a large number of data packets over TCP connections in both directions, making those connections amenable to inspection by measurement tools.

We develop a tool aimed at leveraging BitTorrent for large-scale Internet measurements and make the following contributions:

- We present the design, implementation, and evaluation of *BitProbes*, a system that performs large scale measurements to end hosts using BitTorrent.
- BitProbes discovers ∼500,000 unique measurement targets in a single week, an order of magnitude more than previous systems.
- By using features of the BitTorrent protocol, BitProbes attracts traffic from end hosts that it uses to measure upload capacity, latency, and network topology. To the best of our knowledge, it is the first system that elicits TCP streams from non-webservers.
- We determine the bandwidth distribution for end hosts, updating a previous study of Gnutella users collected in 2002 [9].
- BitProbes collects comprehensive logs of BitTorrent protocol traffic and client behavior. We make anonymized versions of these logs public and present a surprising immediate fallout from their analysis in Sect. 4.5. Specifically, peer capacity is an uncertain predictor of performance in BitTorrent.

2 BitTorrent Overview

BitTorrent is a peer-to-peer file distribution tool that has seen a surge in popularity in recent years [10]. In this section, we describe the features of the BitTorrent protocol that make it amenable to end host measurements. A complete description of the BitTorrent protocol is beyond the scope of this paper, but is readily available [5, 11].

For the purpose of Internet measurement, two parts of the protocol specification are relevant: how peers discover one another and exchange control information and when peers are permitted to send and receive file data.

2.1 Data and Availability Messages

BitTorrent is a request driven protocol. To initially connect to a swarm, a client first contacts a centralized coordinator called the *tracker*. The tracker maintains a list of currently active peers and provides a random subset to clients upon request. The client will then initiate TCP connections with the peers returned by the tracker. Clients then request small pieces (typically 64–512 KB) of the complete file from directly connected peers that possess them. When the download of any one small piece completes, a client is required to notify its connected peers that it has new data available for relay via a `have` message. By serving these small pieces of the whole file, peers assist in its distribution before completing the download themselves.

In addition to per-piece updates between directly connected peers, availability information is also exchanged after the handshake between two newly connected peers via a `BitField` message. This message allows newly connected peers to efficiently update each other's views of pieces available for request.

By monitoring the `BitField` and `have` messages a peers broadcasts, it is possible to infer the download rate of that peer by multiplying the rate at which `have` messages are sent by the known size of each piece. We use this technique in the instrumented BitTorrent client of BitProbes, enabling measurement of not only the properties of end hosts but also information regarding BitTorrent application level behavior.

2.2 Choking, Unchoking, and Optimistic Unchokes.

A noted feature of BitTorrent is its support for robust incentives for contribution, achieved via a tit-for-tat (TFT) reciprocation policy. Informally, a peer preferentially sends data to those peers that have recently sent it data. This has the effect that the more a peer contributes to the swarm, the faster the download rate of that peer will be. Before a peer Q is permitted to request data from a peer P, Q must receive an `unchoked` message from P, meaning that its requests are permitted and will be serviced. If, later, P notices that Q is sending data more slowly than other peers, P will send a `choke` message to Q, indicating that it can no longer make requests.

The problem with a pure TFT approach is that it provides no reason for an existing client to unchoke a newly joined peer as that peer does not have any pieces with which to reciprocate. To work around this difficulty and bootstrap new users, the BitTorrent protocol includes so-called *optimistic unchokes*. Every 30 seconds, each client randomly selects a member of its peer set to unchoke. This serves two purposes; it helps to bootstrap new peers so that they can contribute their resources. Also, it allows each peer to search the set of available peers for those willing to reciprocate with greater capacity. For the purpose of end-host measurements, optimistic unchokes are crucial.

3 BitProbes Design and Implementation

BitProbes provides a platform for conducting large-scale measurements of end hosts. By layering a measurement infrastructure on top of BitTorrent swarms associated with popular files, BitProbes leverages the willing participation of end-users to quickly discover measurement target. This section discusses the BitProbes architecture and the challenges associated with layering a measurement infrastructure on BitTorrent.

3.1 Attracting Traffic

The key to achieving broad coverage of end hosts is making sure that the BitTorrent swarms that BitProbes targets have a large number of participants. To this end, we crawl several popular websites that aggregate swarm connection information and user statistics. We rank these by total users and assign the most popular to measurement nodes running our instrumented BitTorrent client. We do not store or serve any content

obtained. This allows BitProbes to connect to the full range of popular swarms since the risk of distributing copyrighted material is eliminated.

BitProbes relies exclusively on optimistic unchokes to induce measurement targets to willingly send large packets filled with piece data. As a result, the behavior of the BitProbes client is optimized to maximize optimistic unchokes. First, we increase the size of the directly connected peer set. Most BitTorrent implementations maintain 50–100 simultaneous connections per swarm. To increase the likely number of optimistic unchokes, BitProbes increases this limit to 1000. The actual connection count is limited by the number of users in each torrent, but a few hundred directly connected peers per torrent is not uncommon. Second, we connect to many peers briefly. During our measurements, we observed that many BitTorrent client implementations quickly unchoke new connections for a single round without reciprocation, but rarely thereafter. Fortunately BitProbes does not require a large amount of data to be able to make measurements. To limit the resources consumed, both in the swarm and for the nodes running BitProbes, we will disconnect a peer after receiving 2 MB of data.

A challenge is that the trackers from which BitProbes receives candidate peers often specify a minimum time between requests. This time is usually 10 minutes. To increase the number of peers to which each measurements node can connect, BitProbes maintains a *shadow tracker* to share peer information among measurement vantage points. Each measurement node relays all peer information obtained from trackers to the single shadow tracker. The shadow tracker is queried every minute by all measurement nodes to increase their lists of candidate targets. In addition to increasing the number of candidate targets for each measurement node, the shadow tracker also minimizes probing by preferentially selecting only those targets yet to be measured.

3.2 Performing Measurements

BitProbes is designed to provide a wide ranging set of measurement targets to any tool that operates on timed packets from TCP connections. Each measurement node runs a packet logging tool that records packets from BitTorrent TCP connections and passes them to a given measurement tool. Many of the available techniques for transparently measuring properties of Internet paths require an open TCP connection between the target and the measuring node.

To evaluate the feasibility of our approach, the current BitProbes implementation focuses on upload capacity estimation using the MultiQ tool. MultiQ is known to be reasonably accurate, providing 85% of measurements within 10% of actual capacity [12]. This accuracy comes at a price, however, as MultiQ requires traces from a "significant flow" to provide a prediction. Fortunately, the large data packets obtained from BitTorrent connections are sufficient. We use the `libpcap` library to record kernel-level timestamps of incoming packets for accuracy. The arrival times for MTU-sized packets are then supplied to the MultiQ application, which reports the estimated upload capacity. MultiQ is sometimes able to discover the capacity of multiple bottlenecks between the measurement node and the end-host. We assume that the lowest capacity measured is the access link capacity of the end host, even though it also can be due to a bottleneck within the network or the capacity of the measurement node. It is therefore important that the measurement nodes have a high capacity connection. To measure per-hop link

latencies and route information, we are currently experimenting with a technique similar to that used by TCP-Sidecar [13]. By injecting extra packets with the IP record route option and varying TTL into an existing TCP stream, the probe packets appear as legitimate TCP packets both to the end host being probed and to any intermediate firewalls. Although our extension of BitProbes to include this technique is ongoing, it serves as an example of another measurement technique that can be integrated into the BitProbes framework.

3.3 Analysis of BitTorrent Protocol Messages

In addition to end host behavior, layering our measurement infrastructure on BitTorrent allows us to collect data regarding the BitTorrent protocol itself and its behavior in the wild. For instance, we record trace data of `have` and `BitField` messages, allowing us to infer download rates of peers. Correlating this data with capacity measurements provides insight into the effectiveness of BitTorrent's TFT incentive mechanism. We discuss this in Section 4.5.

The trace logs are aggregated at a centralized database for easy access and a global view of swarm behavior. In addition to capacity measurements, we also log all protocol messages as well as tracker responses, peer arrivals and departures, and crawled swarm file sizes and popularity. We make these anonymized versions of logs public.

4 Results

This section discusses initial results obtained with the BitProbes prototype. We show that BitProbes quickly discovers a wide ranging set of measurement targets and that BitTorrent's opportunistic unchokes can furnish MultiQ with enough large packets to provide capacity measurements.

Our measurement testbed includes eight machines running at the University of Washington. Each of these were running 40 instances of the modified BitTorrent client. The results presented were collected between September 2nd and September 9th, 2006. Torrent aggregation sites[1] were crawled once every 12 hours to obtain candidate swarms.

4.1 Rate of Connections

BitProbes provides measurement targets in two forms. First, it collects a list of valid candidate target IP addresses and ports on which those IPs expect to receive TCP traffic. Then, if an IP has no firewall blocking, or if port forwarding is properly set up, we directly connect to the peer. Otherwise, in the case of a peer P behind a NAT, we are able to conduct measurements to P only when P receives our BitProbes client IP from the tracker and decides to connect to it. In practice, this occurs frequently and is therefore a significant benefit to BitProbes, since many end hosts use NATs. In this section, we consider the rate at which new candidate IPs are discovered, independently of whether those connections resulted in successful measurements or not. The rate at

[1] The websites used were http://thepiratebay.org and http://www.mininova.org.

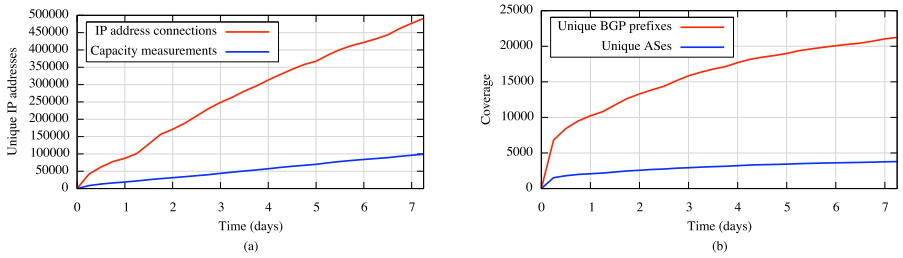

Fig. 1. (a): Number of connections and capacity measurements to unique IP addresses. (b): Increasing AS and BGP prefix coverage.

which new candidates are discovered is an optimistic measure of BitProbes coverage. First, it gives an upper bound on the number of possible measurements BitProbes can provide when running on our current testbed. Second, it indicates how quickly BitProbes exhausts the pool of available swarms and candidate peers made available from the swarm aggregation sites crawled.

Figure 1(a) gives the number of distinct candidate IPs observed over the duration of our trace. During our week-long trace, the measurement nodes connected to or received connections from roughly 500,000 unique IP addresses. Because the rate of new connections is roughly constant, we conclude that the pool of end hosts participating in swarms listed on our crawled sites is significantly larger than the number that can be covered by our prototype system in a week. These results are promising given our plans for a wider deployment. By adding more measurement nodes to BitProbes, we expect to increase the number and coverage of measurements significantly.

As we log data for longer time periods, we expect an eventual decrease in the rate at which new end hosts are discovered. However, we are only crawling only two BitTorrent aggregation sites, and we have already discovered more popular BitTorrent swarm aggregation sites containing an order of magnitude more available swarms than sites we use currently. Further, crawling such sites is straightforward since RSS [14] feeds of available swarms are often provided.

4.2 Rate of Capacity Measurements

The raw number of connections provides an optimistic upper bound on the number of useful measurements possible with BitProbes. Providing concrete results, however, requires integrating an existing measurement tool into the BitProbes framework. We use the MultiQ tool for capacity measurements [12].

For MultiQ to infer the upload capacity of an end host, a TCP flow with a significant number of packets has to be sent by the end host to the measurement node. The packets are then analyzed by MultiQ in an attempt to provide a capacity measurement. Notably, MultiQ will fail rather than return a potentially erroneous measurement if the trace does not meet its correctness criteria. We provide MultiQ with all packet traces having more than 100 MTU sized packets. During our trace, significant flows from 176,487 unique

IP addresses were received. In 96,080 of these cases (54%), MultiQ reported an upload capacity estimate.

As shown in Fig. 1(a), the number of measurements increases steadily throughout our measurement period, providing upload capacity measurements of roughly 100,000 unique IPs in total, meaning that ∼20% of discovered end hosts provide targets amenable to measurement. In contrast to the upper bound provided by the total number of connections, these results provide a more conservative estimate of the likely number of targets that BitProbes can provide to measurement tools. To measure capacity, MultiQ requires specific sanity checks on the number and sizing of packets, requirements that are more stringent than what other tools might need. Further, obtaining TCP packet traces requires that end hosts peer with and optimistically unchoke BitProbes clients, while other techniques (such as TCP-Sidecar) might not require a packet trace.

4.3 Network coverage

Having confirmed that BitProbes is capable of connecting to and collecting measurements from many thousands of distinct targets, we next turn to whether those targets cover the administrative and prefix diversity of the Internet. We chose to build on the popularity of BitTorrent because we expected its popularity to be reflected in wide coverage. In this section, we examine whether we achieve this.

During our week-long measurement period, BitProbes discovered candidate hosts in 21,032 unique BGP prefixes. A BGP prefix corresponds to an entry in the global BGP routing table. As seen in Fig. 1(b), the rate of new prefix observations diminishes over the course of the trace. Almost half of the total number of prefixes observed were discovered during the first 24-hour interval. Even though the swarms joined have a large number of end hosts associated with them, many of these hosts are in a limited subsection of the Internet at large. We suspect that this bias is due primarily to our limited selection of torrent aggregation sites, which cater primarily to English speaking audiences in the US and Europe.

Coverage of BGP Autonomous Systems (ASes) exhibits properties similar to those of BGP prefixes. Connections to a total of 3,763 ASes were observed, with roughly half being discovered in the first 12 hours of operation. Because AS coverage and BGP prefix coverage are correlated, we expect our efforts at increasing prefix coverage by diversifying our crawling to also increase AS coverage. Since one AS often corresponds to one Internet Service Provider (ISP) or POP, higher coverage again requires crawling swarms of global or diverse regional interest. However, the activity and size of ASes varies significantly. The bias in our coverage at the AS level reflects skew in size. Because we do not preferentially probe targets based on AS, we will simply have more measurements to popular ASes and fewer to those less popular. Nevertheless, our current sources of candidate swarms are sufficient to cover ∼20% of total ASes in one week[2].

[2] We estimate the total number of ASes by counting unique observed AS numbers from merged RouteViews [15] and RIPE [16] traces

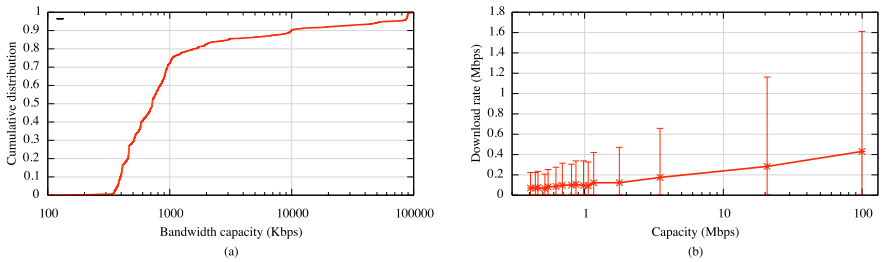

Fig. 2. (a): Upload capacity distribution of BitTorrent users. (b): The average download throughput received by our modified client from peers in a given capacity range. Although average throughput is correlated with upload capacity, no conclusive throughput prediction can be made.

4.4 Capacity Distribution

Knowing the capacity distribution for end hosts on the Internet is a crucial building in the designing distributed systems. This section reports on the observed upload capacity distribution recorded during our evaluation of BitProbes, given in Fig. 2(a). Since the measurements are of hosts running BitTorrent, we point out that these results are not completely general. For instance, users with dial-up connectivity are unlikely to participate in the distribution of multi-gigabyte files via BitTorrent.

A majority (70%) of hosts have an upload capacity between 350 Kbps and 1 Mbps. Only 10% of hosts have an upload capacity of 10 Mbps or more. However, the 5% of hosts with bandwidth capacities between 55 Mbps and 110 Mbps contribute 64% of the available resources, suggesting that successfully incorporating the resources of the high capacity clients is crucial for achieving high utilization in peer-to-peer systems. We make this capacity distribution public along with our BitTorrent trace data.

4.5 Capacity as a Performance Predictor in BitTorrent

This section discusses the applicability of our quick capacity estimation framework to BitTorrent itself. As discussed in Sect. 2.2, BitTorrent employs a TFT reciprocation strategy to encourage peers to contribute resources to the swarm. Although TFT tends to correlate contribution and performance, studies have suggested that in practice it often results in unfair peerings for high capacity users [17]. One suggested method for avoiding this unfairness has been to use quick bandwidth estimation techniques as a basis for selecting peering relationships in BitTorrent. In this section, we evaluate the ability of MultiQ's capacity estimates to predict observed upload rates from those peers from which we receive data.

Figure 2(b) gives the average of observed upload rates at which we receive piece data from peers as a function of the measured upload capacity of peers in a given capacity range. Error bars show the 5^{th} and 95^{th} percentiles of observed download rates of our modified client.

Because BitTorrent shares available upload capacity among several TCP connections simultaneously, we do not expect observed download rates as seen by a single peer to match capacity directly. However, for quick bandwidth estimation to select peers with good performance, the rank ordering of estimated bandwidth and observed performance should be correlated. This is not the case in BitTorrent today, with observed download rates varying quite a bit for for a given capacity (shown in Fig. 2(b)). From the perspective of a single peer in a single swarm, it is impossible to determine as to how many TCP connections a particular client has, whether the client's user has imposed application-level bandwidth limits, or whether the user is participating in multiple swarms simultaneously, all of which will cause actual transfer rate to be much lower than raw capacity. Still, average performance tends to correlate with estimated capacity, suggesting that although quick bandwidth estimation cannot serve as a replacement for ranking peers by observed performance, it may provide early guidance for selecting peers in the absence of other information.

5 Related Work

The opportunistic measurement strategy of BitProbes is broadly similar to that of PlanetSeer [8], which monitor users of the CoDeeN CDN [2] to detect failures on the Internet. By monitoring existing TCP connections to CoDeeN users, it detects when a large number of clients simultaneously disconnect. After such events, PlanetSeer probes the remote nodes from different vantage points. If an endpoint is reachable from some vantage points but not others, it is recorded as a route abnormality. Using CoDeeN as a data source, PlanetSeer observes between 9,000–12,000 clients per day. Sherwood and Spring [13] note that when monitoring the CoDeeN network for a week, 22,428 unique IP addresses were seen using hundreds of available PlanetLab nodes as measurement hosts. In contrast, BitProbes observes 500,000 connections to unique IP addresses in a week from only eight vantage points.

The TCP-Sidecar project shares our goal of avoiding complaints from system administrators [13]. While BitProbes focuses on providing candidate targets to generic measurement tools, TCP-Sidecar is designed to construct an accurate router-level topology of the Internet using a modified traceroute tool. Since traceroutes can cause IDS alarms, modified traceroute packets are embedded into existing TCP connections. TCP-Sidecar use TCP connections from two sources: 1) passive monitoring of CoDeeN, resulting in measurements to 22,428 unique end-hosts per week and 2) downloading `robots.txt` files from web-servers, providing 166,745 unique IP addresses. Both of these sources of candidate targets are limited, in contrast, including TCP-Sidecar's topology measurement techniques in the BitProbes framework may dramatically increase their coverage and effectiveness.

Casado et. al. [18] examine unconventional sources of traffic including spam, worms, and automated scans, yielding a potential coverage of several hundred thousand IP addresses. The authors point out that CAIDA received probes from 359,000 infected servers during the first Code Red outbreak. Using tools similar to those of BitProbes, their system can infer link and path properties to the traffic sources. The number of significant flows from unique sources was limited, however, with only 2,269 unique

sources during a 24 hour period. In comparison, BitProbes attracted 176,487 flows of more than 100 MTU size packets from unique sources during one week of operation. On the whole, BitTorrent appears to be a more fruitful source of candidate targets than other sources previously considered.

6 Conclusion

This paper describes BitProbes, a measurement system for end hosts on the Internet. BitProbes attracts connections from close to half a million unique end-hosts per week by leveraging the popularity of BitTorrent peer-to-peer filesharing swarms. The observed connections are used to infer link latency, topology, and capacity. All measurements are performed unobtrusively, avoiding probes that might trigger IDS alarms. Prototype results demonstrate that BitProbes provides close to an order of magnitude more candidate connections than previous systems relying on opportunistic measurement, suggesting the leverage of attracting measurement candidates via BitTorrent.

References

1. Freedman, M.J., Freudenthal, E., Mazières, D.: Democratizing content publication with Coral. In: NSDI. (2004)
2. Wang, L., Park, K., Pang, R., Pai, V.S., Peterson, L.L.: Reliability and security in the CoDeeN content distribution network. In: USENIX. (2004)
3. Akamai Inc: Akamai: The trusted choice for online buisniess. http://www.akamai.com (2006)
4. Skype: The whole world can talk for free. http://www.skype.com (2006)
5. Cohen, B.: Incentives build robustness in BitTorrent. In: P2PEcon. (2003)
6. Madhyastha, H.V., Isdal, T., Piatek, M., Dixon, C., Anderson, T., Krishnamurthy, A., Venkataramani, A.: iPlane: An information plane for distributed services. In: OSDI. (2006)
7. Anderson, D.G., Balakrishnan, H., Kaawhoek, M.F., Morris, R.: Resilient Overlay Networks. In: SOSP. (2001)
8. Zhao, M., Zhang, C., Pai, V., Peterson, L., Wang, R.: PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In: OSDI. (2004)
9. Saroiu, S., Gummadi, P., Gribble, S.: Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments (2002)
10. Parker, A.: The true picture of p2p filesharing. http://www.cachelogic.com/home/pages/studies/2004_01.php (2004)
11. Cohen, B.: BitTorrent Protocol Specifications v1.0. http://www.bittorrent.org/protocol.html (2002)
12. Katti, S., Katabi, D., Blake, C., Kohler, E., Strauss, J.: MultiQ: Automated detection of multiple bottleneck capacities along a path. In: IMC. (2004)
13. Sherwood, R., Spring, N.: Touring the Internet in a TCP Sidecar. In: IMC. (2006)
14. RSS Advisory Board: Really Simple Syndication: RSS 2.0 Specification. http://www.rssboard.org/rss-specification (2006)
15. Meyer, D.: RouteViews. http://www.routeviews.org (2005)
16. RIPE NCC: Routing Information Service. http://www.ripe.net/ris/ (2006)
17. Bharambe, A., Herley, C., Padmanabhan, V.: Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In: IEEE INFOCOM. (2006)
18. Casado, M., Garfinkel, T., Cui, W., Paxson, V., Savage, S.: Opportunistic measurement: Extracting insight from spurious traffic. In: HotNets. (2005)