Modeling Network-Level Impacts of P2P Flows

Márk Jelasity
University of Szeged and HAS
jelasity@inf.u-szeged.hu

Vilmos Bilicki, Miklós Kasza
University of Szeged
bilickiv,kaszam@inf.u-szeged.hu

***Abstract*—It has been clear for a long time that P2P applications represent a large proportion of the load on the network infrastructure. This is why significant research efforts have been devoted to reducing this load, in the form of ISP friendly P2P solutions. These solutions focus on the volume of the traffic as opposed to the number of network flows. At the same time, we are witnessing a great demand for more and more intelligence in the network such as flow based monitoring and application recognition, which have an overhead that depends on the number of flows and not on the volume of the traffic. Besides, the implementation of this intelligence is moving from the access layer towards the distribution and core layers. We show through measurements that the typical devices serving in the different layers of the infrastructure are not sufficiently scalable in terms of the number of flows, and, most importantly, the combined effect of an increase in the access layer bandwidth together with an increase in the P2P (e.g., BitTorrent) population will practically disable the intelligent networking capabilities. Our conclusion is that a novel focus needs to be incorporated into P2P research that concentrates on reducing the number of network flows generated by P2P applications.**

I. Introduction

It has been clear for a long time that P2P applications represent a large proportion of the load on the network infrastructure. This is not only true in terms of traffic volume, but also in terms of the number of flows. According to a recent study P2P traffic has now surpassed web traffic in both senses [1].

Accordingly, the network community has devoted a great effort to dealing with this problem. One approach is to try to filter P2P traffic (e.g., [2], [3]). However, P2P technology can support legitimate applications as well, so an alternative approach is to modify P2P clients so that they respect the interests of the ISP and the network infrastructure in general, without sacrificing performance. Many such ISP-friendly algorithms have been proposed, most of which rely on localization techniques, where the common goal is that traffic should cross fewer links, preferably staying inside a single ISP (e.g., [4], [5]).

These ISP-friendly solutions focus on traffic *volume* and do not consider the problem of the *number of flows*. However, the number of flows is an increasingly important factor. The network is becoming ever more intelligent, providing context-based services, while the applications and the underlying layers are becoming increasingly infrastructure-aware. The main active device vendors are now opening the black boxes and they are starting to provide environments for running third party applications on their active devices [6]. This will make the intelligent or active network possible [7].

The smooth functioning of intelligent services on network devices is, to a large extent, the function of the number of flows. A NAT service or a monitoring service are clear examples. In fact, monitoring all the flows in the backbone has never really been feasible, so sampling techniques need to be applied [8]. However, sampling is not an option for implementing a NAT or a firewall service, or when rare but important traffic needs to be identified such as P2P botnet control traffic [9].

Our contribution is twofold. First, we measure the performance of several stateful services on network devices of different capabilities, and approximate the maximal number of flows that the given device is able to handle. Second, we argue that current P2P protocols, esp. BitTorrent overlays, can easily generate enough flows to render the intelligent network services within a typical autonomous system (AS) unusable at all levels. We also argue that localization approaches do not provide a satisfactory solution to this problem. Instead, P2P protocols need to reduce the number of flows they generate to be able to scale further.

II. Methodology

To tackle the problem outlined above, we need to study two, rather independent, issues. First, we need to understand how many flows are generated by a typical BitTorrent network, and how many of these traverse network devices at different layers of the network. Second, we need to understand how these devices respond to an increasing number of flows, if certain intelligent services—such as NAT, firewall, or monitoring—are operational.

These two problems are rather complex in themselves, and their combination is more so. We had to make a compromise, and we opted for introducing several simplifications that still allow us to make valid, although approximate, conclusions on the scalability of the entire network. Below we give a bird's-eye view of the overall methodology, the details of which can be found in the respective sections.

For the measurement problem, we selected three devices that can be considered typical in the access, distribution, and core layers, respectively. These are Cisco routers of models 2811, 7200 and 6500. All of these devices support NAT, firewall (reflexive ACL), and flow monitoring (NetFlow). Using the setup shown in Figure 1 (see Section III for details), we generated a varying number of flows (spoofing the headers of the packets to simulate a large network) and monitored the packet loss rate and CPU utilization as measures of performance.

To estimate the number of flows, we generated an AS topology using the topology generator IGen [10]. The size

In Proc. 19th PDP, 2011, pp590–594, doi:10.1109/PDP.2011.9

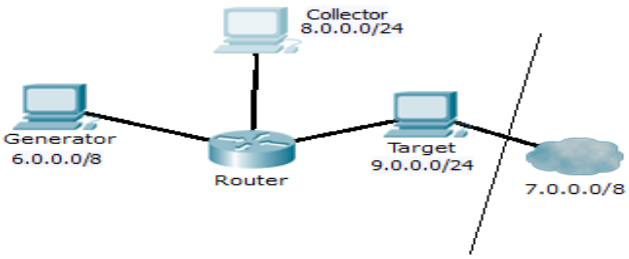

Figure 1. Measurement setup

and topology of this network were set so as to approximate the parameters of the Hungarian academic network Hungarnet. We then simulated different numbers of BitTorrent swarms (clients downloading a single file). The clients participating in a particular swarm are distributed at random over the network. In this setup, based on the parameters of popular BitTorrent clients, we calculated the number of flows that each client generates and mapped these onto the generated physical topology, assuming shortest path routing. From this mapping it is possible to approximate the flow load of each device in the network.

Using this methodology, we can *extrapolate* future BitTorrent popularity and usage by increasing the number of swarms that are present in the network. After measuring the scalability of certain devices, we can also extrapolate the scalability of the network infrastructure in terms of the feasibility of providing intelligent services such as filtering or monitoring.

Our methodology seeks to simulate a complex environment where we are interested in the interaction of a P2P application and the network infrastructure consisting of several layers and different devices. Obviously, the level of realism could be increased at several points, but we believe that the methodology sketched above is suitable for providing a ballpark estimate of the effect of the number of flows. For our purposes this is sufficient, since we are interested in *scalability* as opposed to a fine-grained realistic model of flow dynamics.

III. Measurements

As we said, one of our goals was to test the effect of high density traffic on routers, but with a new aspect in mind, namely the number of network flows. The concern is that when the routers have to track connections, maintain statistics and export them to external collectors, there will be a considerable overhead that depends on the number of flows and not on traffic volume.

To the best of our knowledge, such a flow-centric measurement study has not been conducted for the *stateful* services that we tackle here. One exception is the NetFlow protocol, which was studied by Cisco [11].

A network flow can be defined in many ways. The one we shall adopt here is that it is a unidirectional sequence of packets sharing each of the following properties: source IP address and port, destination IP address and port and the IP protocol. We used IPv4 as the IP protocol, with UDP packets. In this setting, if we spoof the source IP and port for each packet, the router's algorithm will register a new flow for each combination.

Model	features
6500	SUP-720-3B (SR71000 CPU 600 Mhz, Cisco IOS 12.2(33)SXI2a); WS-F6700-DFC3B, WS-F6K-PFC3B; 512 MB DRAM;
7206VXR	NPE 400 (R7000 CPU 350Mhz, Cisco IOS 12.2(28)SB10); 128 MB DRAM
2811	CPU info not released, Cisco IOS 12.4(22)T; 256 MB DRAM

Table I
Devices measured

A. Hardware setup

Figure 1 shows the test setup for the simulation: it consists of a machine used as a traffic generator in a local subnet, and a Cisco router, a target machine, and a monitoring machine in a different subnet.

The generator machine had an IP from the network 6.0.0.0/8. The targets of the packets had an IP from the network 7.0.0.0/8, and they were routed through a PC with an IP from 9.0.0.0/24. We used a loopback device on this PC to simulate an entire 7.0.0.0/8 network range. The router acted as a gateway between these networks, with NAT enabled or disabled depending on the test case.

The source and target machines and the flow collector machine were fixed and they all had a 3.0GHz Intel Pentium IV CPU, 1GB RAM and a Realtek Gigabit Ethernet interface, running Debian Linux 5.0.

The Cisco router was one of the three routers listed in Table I. The 2811 router is designed for small to medium-sized businesses (that is, a maximum of 500 employees). The 7200 model is a more powerful device often used in the distribution layer. The 6500 router is a modular device suitable for deployment in the core layer. Most notably, this is the only device that has TCAM memory (in the PFC3B extension module).

For the tests we used three FastEthernet (100 Mbps) interfaces on the routers. The 2811 model has only two such interfaces: for this reason a switch was added to the configuration during the NetFlow measurements in front of the collector and the target PCs.

B. Traffic generator

In order to simulate a high network load involving a high number of flows, we programmed a traffic generator tool with the capability of creating and sending specially crafted UDP packets. The generated packets have spoofed headers to make the targeted router register a given number of different flows (to simulate the number of clients they serve).

In one experiment, all the packets have the same constant size, and they are generated as quickly as possible, without adding delays. We wrote the application using the C language, for high throughput and speed. When many source IPs are simulated (which is always the case here), the spoofed senders of the packets are picked in a regular fashion, iterating repeatedly over a given range of simulated source addresses.

C. Test parameters

We tested the routers using flows containing packets of 100, 500, or 1000 bytes. The total number of packets sent in one test was always 10,000,000. The number of flows we

# flows	10,000 flows			100,000 flows			1,000,000 flows		
packet size (byte)	100	500	1000	100	500	1000	100	500	1000
6500 - Netflow	100	99.98	100	99.95	99.86	99.93	97.44	96.58	97.00
6500 - Reflexive ACL	99.02	98.98	99.05	86.64	88.07	88.49	5.18	4.97	4.82
6500 - NAT	99.15	99.48	99.54	30.16	30.61	29.90	1.21	1.96	1.13
7200 - Netflow	100	99.00	100	100	100	100	100	100	100
7200 - Reflexive ACL	99.79	99.85	99.93	62.20	59.90	67.07	6.32	5.93	3.14
7200 - NAT	99.93	99.92	100	40.00	10.19	14.88	9.54	1.38	1.68
2811 - Netflow	85.52	96.00	100	99.94	99.90	100	99.90	99.93	100
2811 - Reflexive ACL	96.80	98.83	95.93	87.86	85.44	75.33	11.45	9.85	6.34
2811 - NAT	60.60	70.62	99.44	10.88	8.32	9.93	9.91	1.30	1.24

Table II
MEASUREMENT RESULTS. THE VALUES INDICATE TRANSFERRED PACKETS OUT OF 10,000,000 (%).

# flows	10,000			100,000			1,000,000		
packet size (byte)	100	500	1000	100	500	1000	100	500	1000
6500 - Netflow	18	18	9	80	45	40	80	74	70
6500 - Reflexive ACL	2	3	4	33	25	22	99	99	99
6500 - NAT	5	4	5	96	95	95	99	95	98
7200 - Netflow	23	24	16	22	22	17	20	20	17
7200 - Reflexive ACL	30	45	25	97	81	70	100	97	96
7200 - NAT	60	48	28	96	97	93	100	96	93
2811 - Netflow	90	80	60	70	64	53	62	63	51
2811 - Reflexive ACL	99	96	98	99	92	95	99	99	99
2811 - NAT	99	99	99	99	98	99	99	99	99

Table III
MEASUREMENT RESULTS. THE VALUES INDICATE CPU UTILIZATION DURING THE TEST (%).

discuss in this paper are 10,000, 100,000, and 1,000,000. This means that, for example, in the case of 1,000,000 flows each flow had 10 packets during a test.

All the routers support the four modes of operation that we tested: simple routing, NAT, NetFlow V5 export, and reflexive ACL (firewall). In all the experiments, all the routers performed perfectly (practically all packets were forwarded) for simple routing. In fact, none of the tested routers reached the maximum utilization during the simple routing tests. We do not discuss this case further here.

With NAT enabled, the routers have to translate packet headers and track and maintain basic data about each active connection. This introduces an extra CPU overhead and memory usage that increases with the number of flows.

With flow export enabled the routers maintain flow statistics, aggregate this data and export this information in regular intervals to the flow collector PC. This not only generates extra CPU overhead and memory consumption, but also increases the bandwidth consumption when exporting the NetFlow packets to the collector.

Access control lists (ACLs) implement the rule of a stateful firewall. Here, the communication history is important for decision making. Communication history in most cases means the state of the flow, so we expect similar scalability issues to arise as those for NAT.

D. Results

Tables II and III show the packet loss ratios and CPU utilization for each router and setup combination.

To fully understand the results, the packet arrival rate and the utilized bandwidth need to be discussed as well. In the case of the 100 byte experiments, we were able to send at a bandwidth of around 40-50 Mbps, while for the 1000 byte packets we could reach 80-90 Mbps. At the same time, for the larger packets, the packet arrival rate was of course still slower, despite the higher bandwidth utilization.

Taking this into account, we can explain why there is an improving tendency in CPU utilization as we increase packet size, and sometimes even in the packet loss rate (e.g., in the case of 10,000 flows with the 2811 model): the packet arrival rate decreases with the size of the packets. It should be mentioned here that most of the flows generated by P2P traffic contain very small packets carrying control information.

The 7200 router appears to have a somewhat lower package loss performance with firewalling than the 2811 router. This could be due to the fact that the 2811 router we tested had twice as much DRAM memory.

There are other indications that memory is the key factor in the failure of routers. For example, the 6500 router displays a sudden increase in CPU utilization with NetFlow, at around 100,000 flows and a packet size of 100 bytes (recall that for larger packets the packet arrival rate is lower). At the same point, firewall and NAT performance drops as well. This is very likely due to the TCAM memory being used up.

Overall, there are clear "breaking points" for all the devices. The 2811 router becomes unreliable at 10,000 flows already, when NAT is turned on, while the other two devices start to show unstable behavior at around 100,000 flows. At 1,000,000 flows both the firewall and NAT services become practically unusable.

surprisingly, the flow export service is stable irrespective of the number of flows, showing a constant CPU utilization as well. This could be due to the fact that for NetFlow a much faster memory is used instead of DRAM.

Finally, it should be stressed that the loss of performance is not due to saturating the links of the devices. The actual load we generate is well below 100 Mbps, while, for example, the 6500 router is capable of handling 10 Gbps (as backplane bandwidth) without problems when only routing is used. In each case the failure is due to the inherent requirement of deeper processing and more memory associated with maintaining and filtering flow state.

IV. P2P TRAFFIC

To approximate the number of flows that BitTorrent swarms generate at different points of the network, we first synthesized a physical network topology and subsequently we added different BitTorrent overlay networks.

A. Synthesizing the topology

We modeled the topology after the Hungarnet network. In Hungarnet there are around 700,000 endpoints [12] accessing the Internet through three network layers: core, distribution and access. The core and distribution layers were generated by IGen, a heuristic topology generator [10]. During network generation we used statistical data available on Hungarnet: we generated 600 nodes in the two upper layers, 60 of them in the core layer.

The generation of the access layer is not supported by IGen, so we attached access layer nodes to the nodes in the distribution layer. The number of access nodes attached to a distribution node was heuristically generated as a random number between 30 and 64. The endpoints communicating with each other constitute the fourth layer. We attached 28

nodes to each access node. This way the number of endpoints estimates the 700,000 nodes available in Hungarnet.

B. Modeling BitTorrent swarms

To add a swarm overlay, we assumed that each BitTorrent peer in each swarm has 80 neighbors (which is a typical setting). Connections to neighbors are undirected (bidirectional). Peers maintain a TCP connection to their neighbors, so each peer maintains 80 flows in a swarm. These flows carry only control traffic most of the time. The amount of this traffic is small per flow, but it can be considerable, when hundreds of flows are maintained due to participating in many swarms. According to our own measurements with the Vuze (formerly Azureus) BitTorrent client, each flow generates 1.6 control messages per second on average. This translates to a bandwidth of around 30 to 100 bps per flow.

We also assumed that swarms that coexist in the network are independent. That is, if a peer participates in more than one swarm, then it has 80 neighbors in each swarm that are selected and managed independently.

Swarms are usually much smaller than the total number of 700,000 endpoints in our network. We assumed that the participants of a swarm are assigned to random endpoints, independently for each swarm.

Neighbor selection is a very important component of a swarm model. We used two kinds of neighbor selection methods. The first is *random* selection, where the neighbors of a peer are selected completely at random from the entire swarm. This is the "naive" approach when no optimization is done to minimize the network load.

We also considered a *localized* model, to take into account possible ISP-friendly techniques that also have a side-effect of reducing the number of flows (although their main purpose is reducing traffic that crosses ISP boundaries, or optimizing latency, etc). In this model, after assigning the swarm participants to endpoints, we assign the neighbors to each peer via iterating through the swarm members in increasing distance (hop count) and always picking the first one that has free slots.

C. Experiments

The main goal of the experiments is to measure the number of flows on the various network devices as a function of BitTorrent utilization in the network. We assume that a flow follows a shortest path between its source and destination endpoint, incrementing the flow count on all devices it crosses. To define "BitTorrent utilization", we can either assume a single swarm, the size of which is varied, or multiple swarms, where the size of swarms follows a realistic distribution.

First, let us make an observation: in our network the access layer devices are very difficult to overload with flows. Even if all the endpoints participate in a swarm, the number of flows is still at most $80 \cdot 28 = 2240$. For this reason, we focus on the remaining two layers.

In the case of random neighbor selection, it is easy to see that it does not make any difference whether we increase the number of swarms or we have a single swarm of different sizes: since each flow is placed between two random points, the load on any given device depends only on the overall

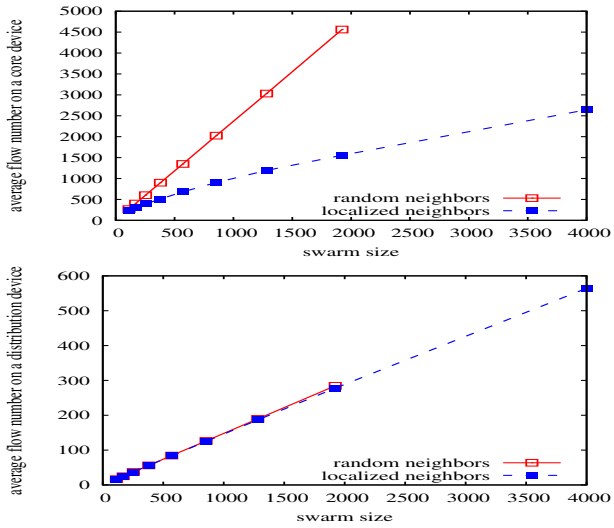

Figure 2. Average flow-load as a function of swarm size (assuming single swarm).

number of flows. In addition, on any given device the expectation of the number of flows scales linearly with the number of peers.

Figure 2 illustrates the load in the random selection scenario. The figure shows the average of the number of flows that have been observed for each relevant device and over 20 independent experiments. Note that the random scenario is measured only up to a swarm size of 2000, since the linear scaling property is analytically evident.

Based on this linear relationship, we reach the critical 100,000 flows (on average) on a core device when the number of peers that participate in a BitTorrent swarm reaches 42,136 (either distributed in many swarms or in a single swarm). This is only 6% of all the endpoints, which is a moderate participation rate.

Let us now move to the case of localized overlays. Here, the size of a swarm matters, because the larger the swarm is, the easier it becomes to find a nearby neighbor. Figure 2 illustrates the scaling of the flow-load as a function of swarm size, assuming there is only a single swarm in the network. It is clear that on the core device there is a significant reduction in the number of flows. However, the distribution layer device experiences the same load, irrespective of localization. The reason is that (with the swarm sizes shown on the plot) there are only a few peers behind each distribution layer device. For example, for a swarm size of 4000, and for our 540 distribution layer devices, each device has around 8 peers behind it on average, so most of the flows have to cross the distribution layer as well.

Now, the question is whether 4000 is indeed a small swarm or a large one? The distribution of swarm sizes in current BitTorrent communities are known (e.g., [13]). The distribution is a power law distribution: $P(\text{swarm size} \geq s) \sim s^{-\alpha}$, where $\alpha \approx 1.1$ and the expectation of swarm size is around 16. Based on these parameters, we get $P(\text{swarm size} \geq 4000) \approx 0.00026$. In other words, in each 10,000 randomly selected swarms only two are larger than

4000 peers on average.

The empirical distribution given in [13] describes the entire Internet, not only a subset of 700,000 endpoints. But the swarm size distribution is scale free, that is, it can be expected to be similar to the global one when restricted to a subset of the Internet. Overall, a swarm of size 4000 can be considered quite large.

Even with localization, the network core is still easily overloaded with flows. If we assume that BitTorrent utilization is modeled as a set of swarms of size 4000, we underestimate the flow load, because in reality most of the swarms are smaller allowing for no possibility for localization. With this assumption, the number of flows is simply a multiplication of the load induced by a single swarm, and the number of swarms. This means—based on the data in Figure 2—that 151,343 peers need to participate in one of 38 swarms of size 4000 on the 700,000 endpoints in order to reach the critical 100,000 flows on the core, which represents around 21% of the endpoints.

D. Discussion

We have seen that core devices experience 100,000 flows on average with 6%, or 21% of the endpoints participating with random, or localized peer selection, respectively, and that the load grows linearly with increasing participation (that can go beyond 100%, since one endpoint can participate in many swarms). Based on our measurements of the control load mentioned earlier, these 100,000 flows will occupy a total of around 5 Mbps bandwidth or more, assuming they carry only control information (overhead). Of course, our measurements are very optimistic, since they ignore non-P2P traffic in the network, as well as non-control P2P flows. What is striking is that the control flows alone can disable key intelligent stateful services on a core device, in our case, the Cisco 6500 router.

As for the distribution layer, the observed average number of flows there is significant as well. Since localization cannot be fully utilized, the number of flows are almost the same for random and localized flows for normal swarm sizes. For these devices, based on the presented data, the 100,000 flow limit is reached roughly when there are 700,000 peers, that is, when all the endpoints participate in one flow on average. This is a large participation, but it is not infeasible, if we project current trends of P2P utilization in several areas including streaming media, TV channels, communication, social networking, and so on.

V. Conclusions

In this paper we argued that the number of flows is a foreseeable major problem for current networks, esp. due to P2P applications, which generate and maintain a very large number of small control flows. We found that traditional plain routing is not affected, because of the availability of specialized hardware, and because of the stateless nature of the task. However, networking devices are becoming ever more intelligent, and the overhead of many of the intelligent stateful services depends on the number of flows. We showed that this overhead is sufficiently large, so that a realistic P2P utilization in a network might cause noticeable problems today, and in future P2P application scenarios it is bound to grow.

Our main conclusion is that if P2P applications intend to be ISP friendly, designers will need to pay more attention to significantly reducing their overhead that is generated by the number of control flows they maintain and manage.

Acknowledgments

M. Jelasity was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences. This work was partially supported by the Future and Emerging Technologies programme FP7-COSI-ICT of the European Commission through project QLectives (grant no.: 231200). We thank Pal Lakatos-Toth for his assistance with the measurements.

References

[1] N. Basher, A. Mahanti, A. Mahanti, C. Williamson, and M. Arlitt, "A comparative analysis of web and peer-to-peer traffic," in *Proc. 17th Intl. World Wide Web Conf. (WWW'08)*. ACM, 2008, pp. 287–296.

[2] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core," in *Proc. 16th Intl. World Wide Web Conf. (WWW'06)*. ACM, 2007, pp. 883–892.

[3] T. T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Comm. Surveys and Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[4] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P users cooperate for improved performance?" *SIGCOMM Comp. Comm. Rev.*, vol. 37, no. 3, p. 40, 2007.

[5] D. Choffnes and F. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," *SIGCOMM Comp. Comm. Rev.*, vol. 38, no. 4, pp. 363–374, 2008.

[6] Cisco Systems, "Cisco Application eXtension Platform Overview," Product documentation, 2008.

[7] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *SIGCOMM Comp. Comm. Rev.*, vol. 37, no. 5, pp. 81–94, 2007.

[8] K. C. Claffy, G. C. Polyzos, and H.-W. Braun, "Application of sampling methodologies to network traffic characterization," *SIGCOMM Comp. Comm. Rev.*, vol. 23, no. 4, pp. 194–203, 1993.

[9] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet detection based on network behavior," in *Botnet Detection: Countering the Largest Security Threat*, ser. Advances in Information Security, W. Lee, C. Wang, and D. Dagon, Eds. Springer, 2008, vol. 36, pp. 1–24.

[10] B. Quoitin, V. Van den Schrieck, P. François, and O. Bonaventure, "IGen: Generation of router-level internet topologies through network design heuristics," in *Proc. 21st Intl. Teletraffic Conf.*, 2009.

[11] Cisco Sytems, "Netflow performance analysis," White Paper, 2007.

[12] Hungarnet, "http://www.hungarnet.hu."

[13] G. Dán and N. Carlsson, "Dynamic swarm management for improved BitTorrent performance," in *Proc. 8th Intl. Workshop on Peer-to-Peer Systems (IPTPS'09)*, 2009.