

Monitoring and Classification of Teletraffic in P2P Environment

Raffaele Bolla, Riccardo Rapuzzi, Michele Sciuto
DIST - Department of Communication, Computer and System Sciences
University of Genoa
Via Opera Pia 13, 16145 Genova, ITALY
raffaele.bolla, riccardo.rapuzzi, michele.sciuto@unige.it

Abstract— Today, a large part of the teletraffic carried by the Internet is generated by file-sharing applications based on the peer-to-peer (P2P) paradigm. This paper considers teletraffic measurements and classification, with an emphasis on techniques which can be used to detect P2P traffic with “conventional” PC hardware, and shows some initial results of such measurements, done at the Telecommunication Networks and Telematics (TNT) Laboratory of DIST-University of Genoa, Italy. The aim is to classify P2P traffic with new traffic measurements, in order to provide a reliable monitoring of the network.

Keywords-component: P2P teletraffic; P2P measurements; file-sharing applications; active and passive monitoring; traffic classification

I. INTRODUCTION

Many recent measurement studies [1][2][3] report that transfer of files still represents the large part of the whole traffic carried by the Internet. Reference [4] outlines that applications based on the P2P technology produce about 60-80% of the overall Internet traffic. The diffusion of residential broadband access is mostly due to the use of bandwidth-intensive P2P applications [5]. A research reported in this paper focuses on analysis of the traffic produced by file-sharing applications, for better understanding how such teletraffic could be effectively measured in real-time in high-speed networks. Our aim is to perform measurements for P2P teletraffic classification, which can provide a reliable monitoring and control of the network.

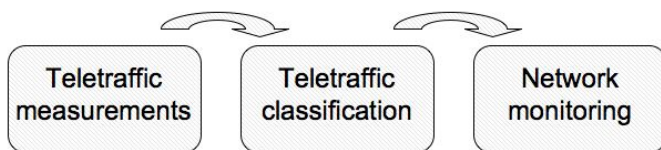


Figure 1. Measuring teletraffic is necessary to classify traffic and then to monitor (and control) the network.

We present some new measurement techniques, developed for detecting P2P traffic. P2P traffic is fast growing, so P2P applications continue to be the killer applications in the Internet. The WWW is being accessed by wider range of users, who frequently are unfamiliar with networking and information technology, but really interested in better and more accessible services. Thus, to satisfy such requirements, it is important to be able to differentiate several classes of traffic in real-time, in order to provide different quality of service for their users. It is well known that in many local area networks, the network

administrators are really interested in filtering P2P traffic out, in order to disable the possibility for the standard users to use it. On the other hand, file-sharing and VoIP applications, based on the P2P paradigm, are becoming very profitable for many ISPs, who are really interested in improving the QoS of P2P applications in order to satisfy the customer requirements. Moreover, today, the Internet carries a large part of P2P traffic in a “hidden” way [6]. Thus, passive monitoring techniques at the flow level, based on filtering of well-known TCP/UDP ports, are not really effective [7]. On the other hand, traffic analysis at the packet level is not easily scalable, resulting in large computational efforts. Owing to this low scalability level, packet monitoring on high-speed links involves only relatively small samples of the entire traffic. Thus, some currently used techniques and tools look at headers of packets and parts of their payload only, or just at only few packets for each connection. An alternative approach is to use specialized hardware, like DAG [8] boards, which are expensive and could be less flexible than software and PC-based solutions. They cannot be, of course, used for classifying encrypted P2P traffic.

We have developed some special monitoring techniques for P2P traffic detection and measurements on high-speed links, using “conventional” PC hardware. These methods can be considered as “hybrid”, i.e. they combine different basic techniques in order to obtain the final results in a scalable way. There are two strictly correlated areas of action: (i) the development of new advanced mechanisms and tools for detecting P2P traffic in high-speed networks, and (ii) conducting measurements with these instruments in operating networks. Both of them are summarized in this work. The reported research activities have been conducted in collaboration with Intel-Research, within the CoMo Project¹. Agreements with CSITA² have been made, in order to monitor activities in operating networks. The paper is organized as follows. Section II and III describe two different teletraffic classification techniques. Section II presents a classification algorithm based on a combination of active and passive monitoring. Section III introduces a hybrid technique, based on a combination of passive monitoring techniques, both at flow and packet level. Future work is presented in Section IV and conclusions are in Section V.

¹ <http://como.intel-research.net/>.

² Centro Servizi Informatici e Telematici di Ateneo –University of Genoa, <http://www.csita.unige.it/>.

II. ACTIVE AND PASSIVE MONITORING COMBINATION FOR CLASSIFYING P2P TRAFFIC

The reported research activities project focus on developing methods for classification of P2P traffic and for discovering the overlay network topology. For this purpose we have applied special techniques, based on active monitoring, which can give us important information about the overlay network. Our work is mainly focused on the design and the development of special crawlers, which can join an overlay network and inject traffic in it, for acquiring and analyzing the acknowledge packets. This approach, combined with a flow level measurement, can be effective for the P2P flow classification in a stub network. We have developed two different crawlers, which take into account specific properties of Gnutella [9]. However, the final goal is to extend our techniques to other protocols. In particular we are interested in Skype [10], which is one of the best candidate for traffic monitoring, as its traffic is encrypted, and as such it cannot be analyzed by using standard sniffing techniques. In this section our investigations of the Gnutella network with two different crawlers are reported. We also discuss how similar studies could be done with the Skype network.

A. The first Gnutella crawler

The Gnutella 0.6 overlay network detection can be provided by using a crawler, a P2P client that can join the overlay network, in order to acquire IP addresses and application port numbers of the largest part of peers in the network. It is worth noting that the overlay network detection process can potentially provide a lot of additional information (besides peers' IP addresses) useful for analyzing and monitoring P2P traffic. It is well known that the Gnutella bootstrapping process is based on contacting a list of active peers (they can be already known or provided by a GWebCache [9]). Gnutella peers flood the received messages to every neighbor, using a TTL equal to 7. At the same time, each peer, that receives a connection message, replies with the relative PONG one, providing information about it and about all the peers it knows at that time. At the end of this procedure a new peer holds the information about a large portion of the overlay network and it maintains updated this information in the following time. Version 0.6 of the Gnutella protocol introduces an overlay hierarchy, in order to reduce the amount of signaling traffic in the network; Gnutella peers are divided into leaves and ultra-peers, and signaling messages are flooded only among ultra-peers, which hold information about their cluster (ultra-peer and attached leaves).

We have modified an existing Gnutella Phex client [11] in such a way that it operates as ultra-peer, continuously collects overlay network data (it generates connecting messages and collects the relative responses) and stores them in a MySQL database. It also supports the X-Try-Ultrapeer header [9], which performs a fast swap of lists of nodes between ultra-peers, enhancing the amount of detected peers. One potential limit of this approach is the injection in the network of the probe traffic generated by the active monitoring client. We

have measured how much traffic our Phex Probe (Phex-P) generates; by limiting the maximum amount of probe traffic we have looked for a relationship between the number of collected IP addresses and the amount of generated traffic. We have observed that the optimal amount of traffic (the minimum one which allows the collection of all IP addresses in a "standard" time) is about 24 Kbps; this amount is also the minimum necessary for enabling the ultra-peer status of the crawler. We can estimate an average number of about 25000 discovered peers every hour. Note that this approach allows ultra-peer traffic to be detected; this fact is related to the specifications of the Gnutella 0.6 protocol, which floods signaling traffic at the first hierarchy level only. Each identified ultra-peer is kept in the database just for 9 hours; we performed this step in order to be sure about the availability of the acquired ultra-peers. The classification technique is based on the acquired information about the overlay network topology, and, at the flow level, on the collected information about every traffic flow is traversing a monitored circuit. The measurement technique is described in the following: (1) Using the Phex-P client presented above, we can collect the IP addresses of the Gnutella ultra-peers and also the application port numbers they reserved to Gnutella application. (2) At the same time we collect NetFlow PDUs [12] from the network device (router) to which the monitored circuit is connected. The raw data exported with NetFlow are organized in a MySQL database, in order to keep trace of every traffic flow carried by the circuit. A specifically developed application (in C++ and PERL) matches the data in the database (collected by NetFlow) and those collected by Phex-P, in order to identify the traffic flows directed to (or originated by) Gnutella ultra-peers. (3) We combine results derived from the crawler, with the information about the flows; the aim consists in finding the Gnutella ultra-peers (identified at the step 1) in the list of flows identified by NetFlow, and available in the MySQL database. When a flow is detected, the corresponding peer, which has an open socket with a Gnutella ultra-peer, is marked as node of the Gnutella overlay network. At this step we know the IP address and the Gnutella port number for this peer. (4) With such information we can detect every Gnutella traffic flow relative to peers of our network, by matching the pairs IP/port that correspond to the peers identified in point 2 above (we have observed that the same port is constantly used by a peer for both signaling and file-sharing traffic); in this way we can measure the Gnutella traffic for each peer. Here we show some sample results, referred to the crawler activity (Figure 2) and to the classification technique presented above. Figure 3 presents one example of traffic classification performed by using the combination of active and passive monitoring; we could validate the classification using a different classification tool such as I7-filter [18].

B. The second Gnutella crawler

The second Gnutella crawler has been implemented on the basis of a modified version of Gnutizen [13], an open source software, appositely developed for providing a minimal framework of functionalities, which can be used for joining

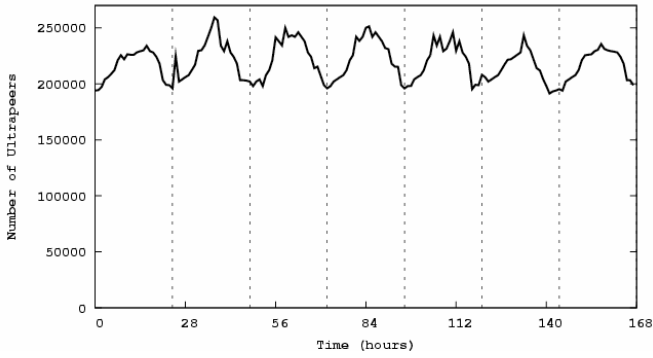


Figure 2. Number of collected ultrapeers by our crawler in the last 9 hours; the behavior is reported over entire week, starting from Monday.

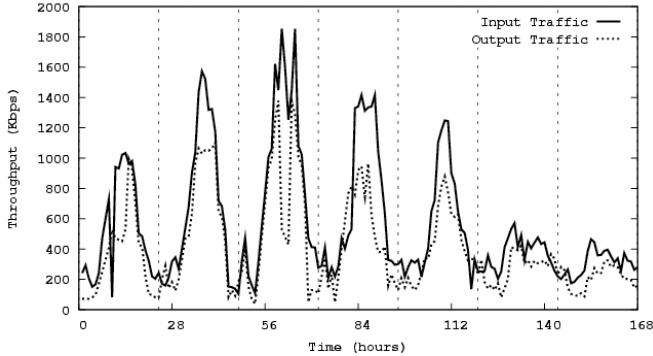


Figure 3. Gnutella throughput measured by using the classification technique based on the Phex-P crawler; the behavior is reported over entire week, starting from Monday.

the Gnutella overlay network, with respect to the Gnutella RFC draft specifications [9]. We have modified the C source code of Gnutizen and, in order to respect our task, we deleted all the parts of code which were not necessary for a crawler, in particular the code parts relative to the file sharing and download procedures; we developed some new source code with the aim of adding some new features to the crawler such as the possibility of running up to 100 parallel threads in the same process, in order to find the greatest number of ultra-peers in the network.

The Gnutizen crawler joins the network in the same way as Phex-P. The relevant information is carried by the PONG messages and by the messages produced by the X-Try-Ultrapeer option, which contain the IP addresses and the application port numbers of the active ultra-peers in the overlay network. Each pair IP/port is appended in a cache file, where the list of active ultra-peers resides. The task of each thread consists in getting in touch with the first available ultra-peer in the list (still unknown to the crawler), realizing, in a continuous cycle, the initialization phase of a Gnutella ultra-peer. When a complete handshake occurs, the complete list of ultra-peers is downloaded by the crawler and appended in the cache file; the process takes up again, contacting all the others ultra-peers in the list, realizing an iterative cycle. In this way it is possible to get a complete snapshot of the overlay network, just starting from one node of the network.

We used a Berkeley database (BDB) [14] as database engine,

in order to store the acquired information, with the aim of performing some statistics on it; the pairs IP/port and a relative timestamp are pushed in the database every hour. In this way we intend to process these acquired results in order to understand how many the discovered ultra-peers are in a particular time interval and also in order to keep a trace of the average lifetime of each ultra-peer. The first analysis of these data has reported that the total number of ultra-peers is approximately constant during the time, as reported in [15]. We decided to move from Phex/Mysql to Gnutizen/BDB (both the software are written in C) with the aim of improving the crawler performances, i.e. minimizing the needed time for acquiring an overlay network snapshot; in fact, with respect to [16], the result accuracy is not independent from the time spent for the acquisition; the more the process is fast, the more the snapshot is accurate.

The classification technique proposed is similar to the previous one, but is based on a more powerful crawler. Figure 4 depicts the number of collected ultra-peers.

C. Classifying Skype traffic with an active crawler

The next step would be to study teletraffic in P2P networks carrying encrypted traffic, such as those used by Skype. In this case, classic classification techniques could be used only at the beginning of each connection (during the handshaking procedures). The Skype protocol is not open and, moreover, all the user traffic and signaling traffic is encrypted. We assume that the use of a crawler can be the best way of traffic investigations in P2P overlay networks of Skype; if combined with a passive monitoring approach, such as the one developed for Gnutella. We can also be able to use an active monitoring technique in order to acquire information about the overlay network, as well as these results can give us relevant information about the protocol's characteristics, which are not well known at the moment. So, we can combine our first challenge, the Skype traffic classification, with an interesting study of the protocol characteristics. Skype provides three different services: VoIP, IM and file-transfer. While little is known about Skype, it is known that it was developed on the basis of the P2P Kazaa protocol [17].

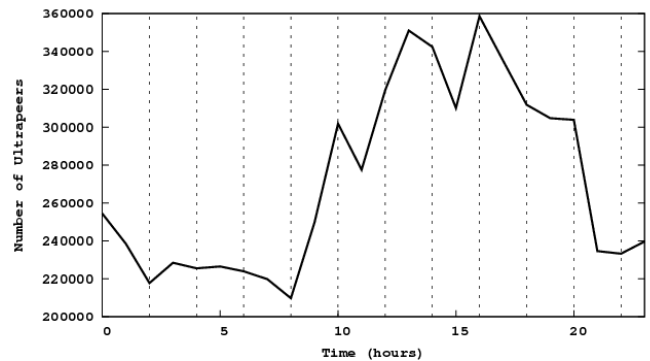


Figure 4. Number of collected ultrapeers by our second crawler in the last 9 hours; the behavior is reported in 24 hours, starting from 12 AM.

The paper [17] is really interesting for studying the protocol characteristics and it is a good starting point for going deeply into the protocol. We intend to use this type of analysis for classification aims, in order to be able to classify the Skype flows in a stub network, using a crawler which can acquire the topology of the overlay network. This approach, as a matter of fact, can be combined with a flow level measurement, performed on the interested circuit, in order to get information about the traffic flows, and then in order to be able to identify which flows are carrying Skype traffic, just considering the tuple information (IP addresses, application port numbers and transport protocol). Using a technique similar to the one presented above for Gnutella protocol, we intend to continuously acquire the list of supernodes in the Internet, in order to look for them in the list of flows traversing our network; in this sense we can classify the Skype traffic, looking for the pairs (IP/port) of the discovered supernodes. The paper [17] explains very well their work, based on a script which can parse and acquire the list of supernodes kept by the crawler; after this process, the script deletes the list from the crawler just letting one supernode. This causes the crawler to download a new list of supernodes, going on in this direction it is possible to acquire a huge list of supernodes of the Skype overlay network. The presence of a supernode can be detected sending the first packet (the same as a client would do) to the supernode in the list, and waiting for the reply. In this way it is also possible to take snapshots of the Skype overlay network.

III. PASSIVE MONITORING AT BOTH FLOW AND PACKET LEVEL FOR CLASSIFYING P2P TRAFFIC

Our second monitoring technique is based on a passive monitoring approach and provides a scalable and powerful way for traffic classification. We decided to use a protocol signature based technique, which, nowadays, seems to be the most accurate way for detecting P2P traffic [7]. Starting from a set of patterns, which can identify each protocol, we perform a search procedure inside the packet payload, with the aim of finding some protocol patterns. The technique is considered very accurate but not easy scalable, considering the analysis of each payload packet. We considered a hybrid approach to the classification, which is focused on the protocol signature but it wants to solve the scalability problem too; we found a good solution in a passive monitoring approach at both flow and packet level. The flow level is used in order to collect information about the flows, using NetFlow as a source of information. At the flow level we can acquire a lot of important information about flows, mostly of them are statistical information, which we want to collect and to analyze in order to perform, in the next future, a real-time classification based on a statistical approach. For every flow we can acquire the duration, the bandwidth, the amount of carried data (both in byte and packet), the inter-arrival time, and the ratio [bytes/pkts]. At the packet level, we can perform a classification of every flow, using a classification engine, based on a pattern matching approach but which could work just considering the first part of every flow, in order to perform a fast classification.

As a beginning, we decided to use I7-filter [18], as classification engine; it is clear that the choice of the classification engine is independent from the structure of the entire software. The data matching between the flow and the packet level analysis is done in the database, so after the flow classification. We intend to study this type of data analysis using an existing tool, with the future aim of developing our own classification software. I7-filter is a very reliable and fast tool for Linux's Netfilter that identifies packets by observing the application layer data. An important aspect, relative to scalability, is that I7-filter works just in kernel space; so we expect it can improve performances with respect to user space sniffing tools (e.g., Ethereal or Tcpdump). I7-filter considers only the first part of each connection, the default amount of data is 2048 bytes³ (or 10 packets). As I7-filter considers just a portion of the entire flow, it cannot measure the whole amount of traffic for the flow; it is important to consider that I7-filter has been designed to operate as a firewall, rather than a passive monitoring instrument. Its main application is such as a border gateway device, e.g. a firewall, which must analyze traffic and take decisions based upon that. We could realize that it is an ideal tool to classify traffic in a scalable way. We configured it as working on an Ethernet interface, in promiscuous mode, not as a transit device, but just as an external classifier.

We performed many measurements in order to evaluate the I7-filter performances. We tuned the portion of data considered by I7-filter, in order to improve the classification rate, minimizing the classification error. We found that, in the case of P2P protocols, like Gnutella, eDonkey2000 and BitTorrent, the best value of *maxdatalen* is 100 bytes. Figure 5 presents the I7-filter performance results we could measure for the P2P protocols of our interest. We verified that the classification error is zero up to rates of 80 Kpps, equivalent to 376.45 Mbps, considering the type of traffic we used for the measures.

The information acquired by I7-filter is matched with those obtained with NetFlow, in this way P2P flows are detected, and the amount of data for each of them is measured. The results of this type of classification are presented in the next Section IV, where we describe how it is possible to use a signature based classification technique for moving to a statistical approach.

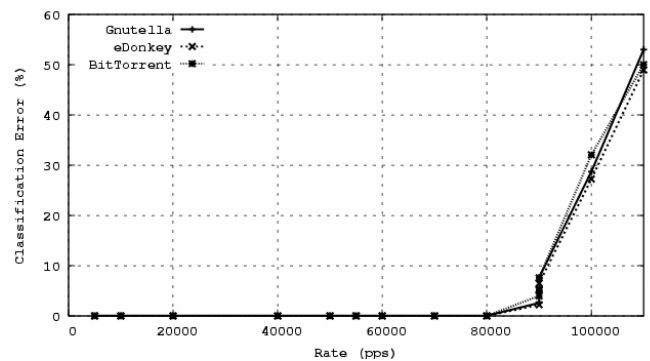


Figure 5. Classification error with *maxdatalen* = 100 bytes; in presence of P2P protocol filters (Gnutella, eDonkey2000 and BitTorrent).

3 The amount of data considered for the classification is known as *maxdatalen*, and can be set as a configuration parameter.

IV. TOWARDS A STATISTICAL CLASSIFICATION

The previous traffic monitoring technique gives us the possibility to classify traffic and to acquire some important statistical information about it. At this step, our intention consists in finding a relationship between the application layer protocol and the measured statistical parameters. In this way we will be able to divide traffic in different classes, each set of statistical parameters will mark out a particular class (protocol) of traffic. Following we present the results of some traffic classification performed for the protocols eDonkey2000 and BitTorrent, which underlines the relevance of some statistical parameters⁴, we are considering for a classification technique based upon them: the average flow inter-arrival time, the average flow size, the average ratio between bytes/packets per flow and the average flow bandwidth. We present also the measures for the number of classified flows, see Figure 6.

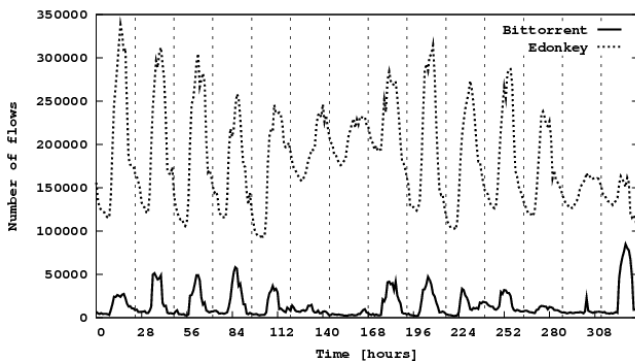


Figure 6. The average number of flows.

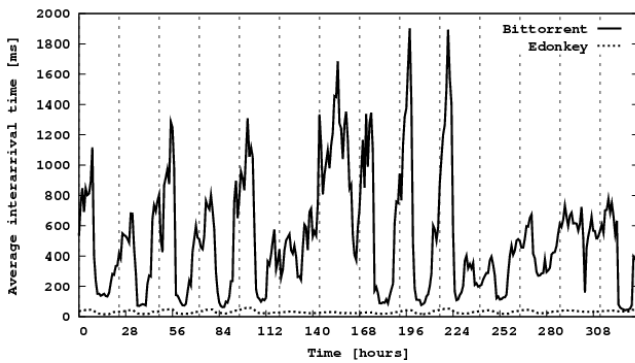


Figure 7. The average inter-arrival time.

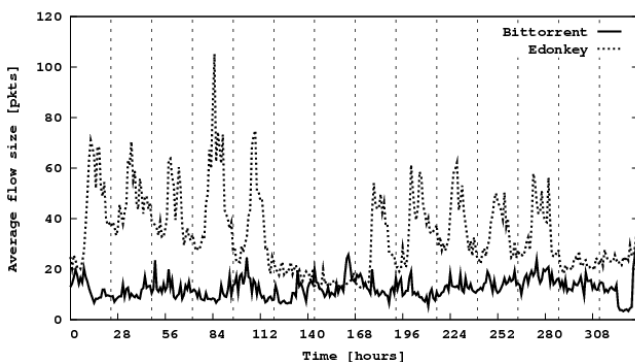


Figure 8. The average flow size.

⁴ The figures report the average values in time periods of one hour.

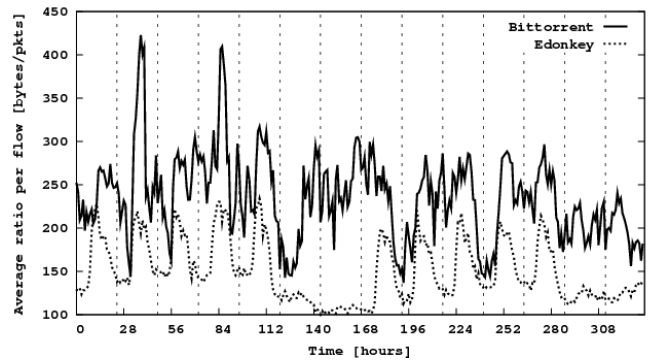


Figure 9. The average ratio between bytes/packets per flow.

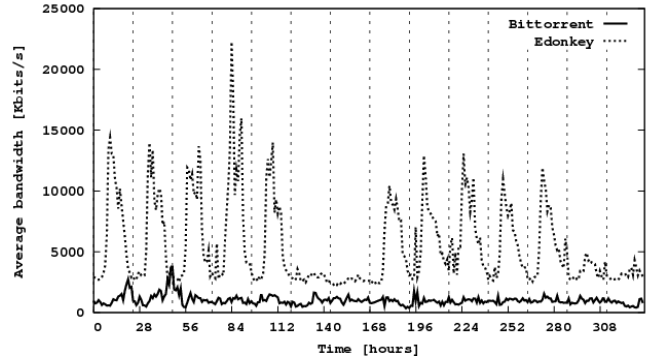


Figure 10. The average bandwidth.

V. CONCLUSIONS

The paper overviews our work in monitoring and classification of P2P traffic. Three different techniques are described. The hybrid techniques have been developed and carefully validated, even we plan to extend our approach to other P2P networks (e.g. Skype). The results presented for eDonkey2000 and BitTorrent in Section IV show that teletraffic characteristics in different P2P networks can significantly differ. Thus, further detailed studies of teletraffic in various P2P environments are needed for their better understanding.

REFERENCES

- [1] N. B. Azzouna, F. Guillemin. "Analysis of ADSL traffic on an IP backbone link", In *Proceedings of the 2003 IEEE Global Telecommunications Conference (GLOBECOM 2003)*, S.Francisco, CA, USA, December 2003, Vol. 22, No. 1.
- [2] M. Ripeanu, I. Foster, A. Iamnitchi. "Mapping the Gnutella Network". *IEEE Internet Computing*, Vol. 6, No. 1, pp. 50-57, January/February 2002.
- [3] S. Saroiu, P. K. Gummadi, S. D. Gribble. "A Measurement Study of Peer-to-Peer File Sharing Systems". In *Proceedings of Multimedia Computing and Networking (SPIE/ACM MMCN'02)*, San Jose, CA, USA, January 2002, pp. 156-170.
- [4] S. Sen, J. Wang. "Analyzing Peer-to-Peer Traffic Across Large Networks". *ACM Transactions on Networking*, Vol. 12, No. 2, pp. 219-232, April 2004.
- [5] L. Plissonneau, J. Costeux, P. Brown. "Analysis of Peer-to-Peer Traffic on ADSL". In *Proceedings of the 6th International Workshop on Passive and Active Network Measurement (PAM 2005)*, Boston, MA, USA, March/April 2005, Vol. 3431, pp. 69-82.

- [6] T. Karagiannis, A. Broido, N. Brownlee, K.C. Claffy, M. Faloutsos. "Is P2P dying or just hiding?" In *Proceedings of the 2004 IEEE Global Telecommunications Conference (GLOBECOM 2004)*, Dallas, TX, USA, November/December 2004.
- [7] S. Sen, O. Spatscheck, D. Wang. "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures". In *Proceeding of the 13th International Conference on World Wide Web (WWW 2004)*, New York, NY, USA, 2004, pp 512-521.
- [8] Endace. Web site, <http://www.endace.com>.
- [9] Gnutella. Web site, <http://rfc-gnutella.sourceforge.net/developer/testing/>.
- [10] Skype. Web site, <http://www.skype.com>.
- [11] Phex. Web site, <http://phex.kouk.de/mambo/>.
- [12] NetFlow. Web site, <http://www.cisco.com>.
- [13] Gnutizen. Web site, <http://gnutizen.sourceforge.net>.
- [14] "Performance Metrics & Benchmarks: Berkeley DB". White Paper, 2005. Available online at <http://dev.sleepycat.com>.
- [15] A. H. Rasti, D. Stutzbach, R. Rejaie. "On the Long-term Evolution of the Two-tier Gnutella Overlay". In *Proceedings of the 9th IEEE Global Internet Symposium*, Barcelona, Spain, April 2006.
- [16] D. Stutzbach, R. Rejaie. "Capturing Accurate Snapshots of the Gnutella Network". In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, March 2005, Vol. 4, pp. 2825-2830.
- [17] S. Guha, N. Daswani, R. Jain. "An Experimental Study of the Skype Peer-to-Peer VoIP System". In *Proceedings of the 5th International Workshop on Peer-to-Peer System*, Santa Barbara, CA, USA, February 2006.
- [18] I7-filter. Web site, <http://i7-filter.sourceforge.net/>.