New Heuristics for Node and Flow Detection in eDonkey-based Services

Rafael A. Rodríguez-Gómez, Gabriel Maciá-Fernández, Pedro García-Teodoro
Department of Signal Theory, Telematics and Communications,
CITIC - ETSIIT, University of Granada
Granada, Spain
rodgom@ugr.es, gmacia@ugr.es, pgteodor@ugr.es

***Abstract*—The development and use of applications based on peer-to-peer (P2P) networks have exponentially grown in the last years. In fact, the traffic volume generated by these applications supposes almost the 80% of all the network bandwith nowadays. For this reason, the interest of Internet Service Providers (ISPs) for classifying this large amount of traffic has also grown in a considerable manner. In this context, the present paper describes two detection algorithms for eDonkey services. The first one is aimed to detect eDonkey flows. It is based on the hypothesis that clients that begin connections are in charge of sending the information. The second algorithm has been developed to detect nodes that generate eDonkey traffic. It is based on the hypothesis that the up-rate of these nodes follows a constant pattern along the time. Both detection algorithms have been proved in three different groups of network traces. As a result, our detection hypothesis is checked. Additionally, the experiments carried out show that the proposed algorithms have a high classification rate and a low false positive rate.**

Keywords-Traffic classification; flow detection; node detection; P2P; eDonkey.

I. INTRODUCTION

The development and use of applications based on P2P networks have exponentially grown in the last years. Nowadays, several examples can be found: eMule or uTorrent, as file sharing applications, Skype, as voice over IP application, and Spotify, as audio flow sharing.

Traffic generated by P2P applications consumes around 80% of all the network bandwith [1]. This enormous use of available bandwith requires the allocation of a considerable amount of resources to guarantee the quality of the provided services.

The ability of classifying the P2P traffic is a key issue to ISPs, as they are forced to increase the maintenance operations due to this excessive growth in the use of P2P services [2].

Traffic classification methods can be divided in three approaches: *(i)* based on port, *(ii)* based on packet content, and *(iii)* based on flow features. Current P2P applications can receive connections in any port and encrypt the content of its messages. This feature makes the classifications based on port and packet content difficult.

The classification methods suggested in this paper are based on flow features, and are used to detect eDonkey traffic, a communication protocol of P2P networks mainly used in file sharing applications such as eMule or aMule.

Two new detection heuristics are proposed: *node detection based on up-rate*, and *flow detection based on inversion of download direction*.

Node detection based on up-rate relies on the next two assumptions: *(i)* users of eDonkey-based applications limit the up-rate, and *(ii)* the up-rate is approximately constant around this limit established by the user.

The proposed flow detection method assumes that those nodes with eDonkey traffic that, under certain conditions, establish a connection will send the majority of the information. This behavior is radically opposite to the common client-server paradigm, in which clients establish connections and servers send the required information.

The rest of the paper is organized as follow: In Section II, some relevant papers in the field of traffic classification are presented, the novelty of the present contribution being remarked. In Section III, some specially relevant concepts regarding eDonkey are exposed. The proposed heuristics to detect nodes and flows in eDonkey-based services are detailed in Section IV, presenting Section V the experimental framework considered. In Section VI, the experimental results obtained are shown and discussed. Finally, in Section VII, principal conclusions of this work are drawn.

II. RELATED WORK

In the specialized literature, three kinds of classification methods can be found: *(i)* based on well-known ports, *(ii)* based on packet content, and *(iii)* based on flow features. T. Karagiannis et al. [3] assure that classification methods based on well-known ports are not valid to detect P2P traffic nowadays. On the other hand, methods based on packet content imply legal issues related to privacy and thus, their field of application is enormously reduced.

There exists a huge amount of papers proposing classification methods based on flow features analysis. A relevant example is BLINC [4], a classification tool based on the assumption that a host can be associated with an application responsible to generate the majority of its traffic volume. In the same manner as BLINC, we also propose here a node classification method.

 ISBN: 978-1-61208-173-1

Another possibility is to classify only a subgroup of existing protocols. This is the most common approximation in the field of P2P classification. An example of this we present the work of T. Karagiannis et al. [3], this is the first work that tries to classify encrypted P2P traffic in random ports without inspecting packet payload. The classification is based on connection patterns of P2P networks. Our classification methods also detect P2P encrypted traffic in random ports.

Xu et al. [5] propose a method to identify P2P traffic based on the data transfer behavior of P2P applications. The authors assure that the downloaded data by a node will be subsequently uploaded to another node in the network. Thus, flows that download and upload the same data blocks will be identified as P2P flows. The heuristics proposed in the present paper are also based on the data transfer behavior of P2P applications. As we will show in Section IV there exist several differences with respect to the work of Xu et al.

Finally, there exist some targeted at classifying a single protocol. As an example, Bonfiglio et al. [6] detect real time Skype traffic exploiting the randomness introduced at the bit level by the encryption process to detect Skypes fingerprint from the packet framing structure. In our case, the classification also aims at classifying only the eDonkey protocol.

III. General concepts of the eDonkey protocol

The eDonkey protocol was designed to communicate nodes belonging to a hybrid P2P network composed by server and client nodes. Servers are in charge of giving access to the network and managing the information distribution in a similar manner to a dictionary, *i.e.*, they store the correspondence between resources and the nodes sharing them. On the other hand, clients are the nodes that share data, and they store the resources of the network.

In the following, a brief description of eDonkey-based communications, with special interest in the aspects used in the present paper, is provided. To do this in a structured manner, we explain the process followed by a client to download a resource from the network.

A. Client to server connection

The first step to download a resource from the network is to connect to an eDonkey server. This connection can be divided in two steps: *(i)* TCP connection from client to server, and *(ii)* server challenge. To carry out this challenge the server also tries to establish a TCP connection with the client. This connection allows to discover if the client is able to receive connections from other clients of the eDonkey network. If the challenge is passed, the client will receive an identification called *high ID*; otherwise, it will receive a *low ID*. Thus, a high ID client can directly receive connections from other clients in the network, while a low ID client can not.

Once a client has accessed to an eDonkey server, it can look for resources by describing them with certain key words. The requested server will respond with a list of related resources. Subsequently the client requests one or several of them to be downloaded, and finally he will receive a response with a list of clients that share the requested resource.

B. Client to client connection

It is necessary to carry out client to client connections in order to download any resource from the network. However, a client with a low ID can not accept connections. To solve this, there exists a procedure in the eDonkey protocol called `callback`. If a client wants to connect to a low ID node, it has to send a `callback` message to the corresponding eDonkey server. This server will resend, through a previously established connection, the `callback` to the low ID node who will begin a new connection. This mechanism does not solve the case in which both nodes are low ID because none of them are able to accept external connections.

C. Downloading of a shared file

Downloading a shared file in eDonkey protocol consist of two steps: *(i)* entering the reception queue, and *(ii)* starting the download. Firstly, a client A requests a file to a client B. B answers by sending the position in which this request is stored in its queue. Secondly, when this request reaches a position in the queue to be served, B sends a message to A indicating this new state so that the download process starts.

The most common situation in a download process is that a request from client A has to wait a considerable time to be served. Thus, after a fixed time (around 40 seconds in our experiments) client B closes the connection with client A. B will establish a new connection when the request from A is able to be served. As detailed in Section IV, this behavior will be used in our flow detection method.

IV. Detection heuristics

Two detection methods of eDonkey protocol are now proposed: node detection based on up-rate and flow detection based on inversion of download direction. These methods can work together: firstly, a node detection indicates those nodes that generate eDonkey traffic and, subsequently flow detection determines specifically eDonkey file sharing flows of the detected node.

The detection heuristics used in both methods are explained in the following.

A. eDonkey flows detection

The first heuristic is aimed to detect eDonkey flows based on the hypothesis that clients that begin connections also send the majority of the data over that connection. As explained in Section III, this is the most common situation in any download process of a shared file in eDonkey.

 ISBN: 978-1-61208-173-1

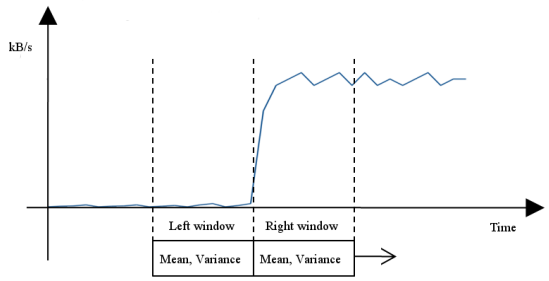

Figure 1. Calculation of Kullback-Leibler distance in time.

In client-server applications, servers usually send the majority of the data after a connection is started by a client. This behavior is reverse to that of eDonkey protocol, and this is the reason because we propose to use the next hypothesis to detect eDonkey flows:

Hypothesis 1. *eDonkey flows are those in which clients who begin the connection send substantially more information than they receive.*

Note that this heuristic is only valid for file sharing flows, and not in the case of signaling flows. File sharing flows are specially relevant because they are the principal responsible of congesting the bandwidth of a network.

The proposed detection method has been developed to be executed over offline traces. eDonkey flows are those where the number of bytes sent by clients (who begin the connection) are greater than the number of bytes received plus a threshold, $Thres_B$. This threshold is experimentally determined by means of a study of the size distributions in file sharing flows of the eDonkey protocol.

The selection of the threshold to be used in this method is not critical, as we argue in Section VI, because the difference between sent and received bytes in file sharing flows is really noticeable.

B. eDonkey nodes detection

Here we expose a method to detect nodes which are generators of eDonkey traffic.

The proposed method is based on the assumption that users of P2P file sharing applications usually limit their up-rate. This is due to the fact that these applications consume the majority of the upload bandwidth, and consequently users that use them without a limitation in the up-rate suffer a decrease in the quality of their normal Web browsing.

The mentioned constant rate supposes a differentiating feature allowing the detection of these nodes. In conclusion, the proposed hypothesis is defined as follow:

Hypothesis 2. *Nodes generators of eDonkey traffic are those whose up-rate is quasi-constant.*

To apply the previous hypothesis it is necessary to detect a quasi-constant level in the up-rate of a node. We take as a reference the work of J. Ramirez et al. [7], in which they use the Kullback-Leibler (KL) divergence to detect voice activity in audio signals. This detection is aimed to determine the specific instant at which an evaluated audio signal changes from only noise to contain human speech, or vice versa. Authors use KL divergence to detect changes in mean and variance of audio signals. In our case the KL divergence is used to detect the absence of significant changes in the up-rate of a node (quasi-constant up-rate).

The KL divergence can be described as an indicator of the similarity between two probability distributions. In the case of two Gaussian distributions p_L and p_R is defined as (taken from [7]):

$$H(p_L||p_R) = \frac{1}{2}[log(\frac{\sigma_R^2}{\sigma_L^2}) - 1 + \frac{\sigma_L^2}{\sigma_R^2} + \frac{(\mu_L - \mu_R)^2}{\sigma_R^2}] \quad (1)$$

where σ_R y σ_L represent standard deviations of p_R y p_L, and μ_R y μ_L their means.

The KL divergence is not symmetric, and thus we will use the KL "distance" $\rho_{I,D} = H(p_R||p_L) + H(p_L||p_R)$. In the case of Gaussian distributions it is defined as (taken from [7]):

$$\rho_{L,R} = \frac{1}{2}\left[\frac{\sigma_L^2}{\sigma_R^2} + \frac{\sigma_R^2}{\sigma_L^2} - 2 + (\mu_L - \mu_R)^2(\frac{1}{\sigma_L^2} + \frac{1}{\sigma_R^2})\right] \quad (2)$$

The proposed detection method can be described as follows (Algorithm 1). Firstly, up-rate values are calculated every t seconds. A median filter [8] of length N is applied to the resulting values. This filter takes a window of length N and sorts some values extracting the central one.

Secondly, the means and variances of the filter values are calculated by means of two consecutive windows (v_I y v_D) of length N (see Figure 1). The Gaussian distributions represented by these means and variances should be very similar if there exists a quasi-constant up-rate. Therefore, the KL "distance" (Equation (2)) computed from these means and variances should be minor than $Thres_{KL}$ to represent a constant up-rate. If a constant rate is detected, our method will classify the corresponding node as an eDonkey traffic generator.

V. Experimental framework description

Three groups of network traces have been used to carry out the experimentation related to the proposed methods of eDonkey traffic detection. In the following, the principal features of these traces are exposed.

- *Controlled environment traces (CE).* The traffic generated by 5 users during 72 hours were collected. In this period, they used aMule version 2.2.6 and shared the same 10 files. The eDonkey server to which they connected was *se-Master Server 1*. They limited the up-rate of aMule to 30kB/s. All of them used their PCs

 ISBN: 978-1-61208-173-1

Algorithm 1 Node detection

```
for node = 0 while node < num_nodes do
  for i = 0 while i < len(up_rate_node) do
    rate_filt_node ⇐ median_filt(up_rate_node, N)
    v_I ⇐ rate_filt_node[i : i + N]
    v_D ⇐ rate_filt_node[i + N + 1 : i + 2N + 1]
    ρ_{L,R}[i] ⇐ Equation(2)
    if ρ_{L,R}[i] < Thres_KL AND μ_R different to 0 then
      return eDonkey node
    else
      return Not eDonkey node
    end if
    i ⇐ i + 1
  end for
  node ⇐ node + 1
end for
```

and Internet connection without restrictions. Every user generated around 19,000 connections of the eDonkey protocol and more than 7,000 of other protocols like DNS, HTTP, SSH and SMTP.

- *HTTP server traces (HS).* This collection of traces represents the traffic generated by an HTTP server from an European University during 7 days. The server is Apache version 2.2.0 and receives a mean of 8,971 connections per day.
- *University trunk traces (UT).* All the traffic of a trunk from a Middle East University during 48 hours composes these traces. There are around 73,000 IPs, 300 millions of packets transmitted, and the most common protocols that appear are: Bittorrent, HTTP, DNS, SSL, and FTP. After the analysis of the entire database by means of a packet inspector application (`OpenDPI` [9]), very few eDonkey packets have been detected. This is due to the fact that the P2P file sharing applications used in this Middle East University are based on Bittorrent protocol instead of on eDonkey.

VI. Experimental results

The experimentation carried out for both detection methods is focused on a double aim: *(i)* To study if the presented hypotheses are valid through the evaluation of detection and false negative rates obtained in CE traces, and *(ii)* to analyze if these hypotheses present low false positive rates for other protocols through HS and UT traces.

Following, the obtained results for flow and node detection methods are analyzed.

A. eDonkey flows detection

First of all, to execute the eDonkey flow detection method it is necessary to determine the value of threshold the $Thres_B$, which is the maximum difference allowed between the number of the sent and received bytes to consider the evaluated node as not generator of eDonkey traffic. The results of a study of the detection and false positive rates in function of $Thres_B$ in the three groups of traces indicate that there exists a wide range (between 5 and 25kB) to select this threshold in which the success of the method is very similar. Specifically, the threshold selected is 10kB.

Table I
Detection rate of flow detection method in UT traces.

	Detection rate	Detected flows	Total flows
BitTorrent	0.0256	854	33,304
HTTP	0.01691	50,795	3,003,161
FTP	0.01423	35	2,460
SSL	0.01244	2,808	225,685
IRC	0.00213	7	3,281
Oscar	0.00079	2	2,528
DNS	0.00001	8	1,508,413
Mail_POP	0.00000	0	5,208
All	0.01139	54,509	4,784,040

In the present experiment another assumption has been applied. An eDonkey node download or upload files from or to more than one peer simultaneously. Thus, we can suppose that eDonkey flows coincide temporally with other eDonkey flows.

CE traces contain 37,089 file sharing flows of eDonkey protocol. 28,016 of them have been detected, which implies a detection rate of 77.53%. None of the flows belonging to other protocols different to eDonkey have been detected (0% of false positives). Finally, there exists a considerable false negative rate: 22.47%. This is mainly due to two factors:

1) *Low ID in some of the ends.* Nodes with low ID can not accept connections from other peers of eDonkey network and, for this reason, they always begin the connections, independently of being server or receiver of the information. This situation is not valid in the proposed detection hypothesis.
2) *Service without an intermediate close of connection.* A request of a resource could be served without an intermediate close of the initial connection (explained in Section III). In this case, the number of bytes received are greater than the sent.

Therefore, the proposed detection method is able to detect file sharing flows of eDonkey protocol between high ID nodes and with an intermediate close of connection (the most common situation, as we mentioned in Section III).

The results obtained from HS traces show that none of the 62,798 HTTP flows have been detected as an eDonkey flow, which represents a 0% of false positive rate.

Flows from UT have been labeled through the execution of a modification of `OpenDPI`, an application that performs deep packet inspection [9]. We take this labels as ground truth and compare the results of our detection method with it obtaining the results shown in Table I.

The principal contribution in false positve rate corresponds to bitTorrent protocol. This protocol is also used in

 ISBN: 978-1-61208-173-1

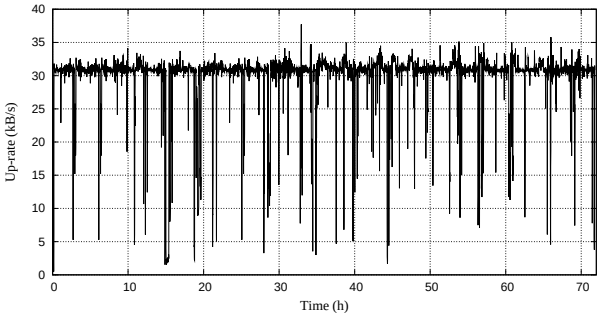

Figure 2. Typical up-rate of a monitored node from CE traces.

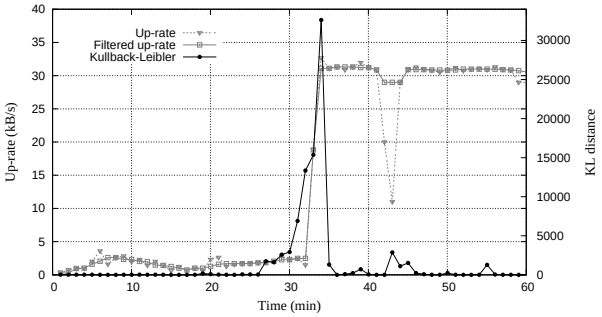

Figure 3. Evolution of Kullback-Leibler distance in one hour of CE traces.

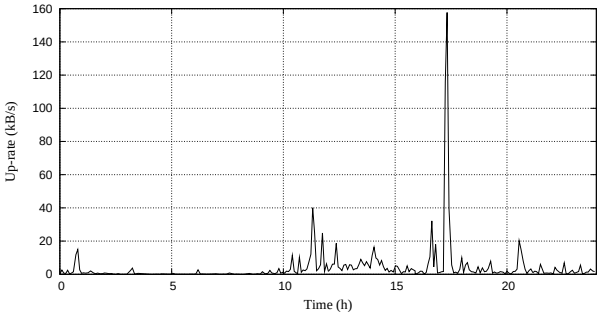

Figure 4. Up-rate of HTTP server (HS trace) during 24 hours.

P2P file sharing applications. However, a principal difference with eDonkey is that bitTorrent flows are bidirectional, so both peers send information to the other in the same flow and it can occur that clients that begin a connection send more than they receive. In the same manner, FTP flows were detected because clients frequently send more data than servers do.

Finally, it is remarkable the false positive rate associated to the HTTP protocol, because none of the flows in HS traces were detected while UT traces represent a 1.69%. After a detailed study we conclude that these false positives are mainly caused by three factors: *(i)* high length of cookie and URL in HTTP GET messages, *(ii)* very short server responses, 304 (Not Modified), and *(iii)* HTTP POST sending a big amount of data. This is the case of using the Web 2.0 philosophy, but it has no significant influence in our detection method nowadays.

B. eDonkey nodes detection

The second detection hypothesis is validated with CE traces, as we can see in Figure 2, in which the up-rate of a monitored node is shown. During the 72 hours of traces there exists a quasi-constant behavior around 30kB/s, the limit fixed in the experiment.

The up-rate suffers several descents of low duration. These descents correspond to instants at which the monitored client stops sending data to a peer in order to begin the transmission with another one. The churn [10] (independent arrival and departure by peers) is extremely high in P2P networks, so the mentioned situation is frequent.

In CE traces, a 38.7% of flows belong to other protocols different to eDonkey, the most common being DNS, HTTP, SSH and SMTP. This is represented in Figure 2 as instants at which the up-rate limit fixed in the experiment is exceeded. So, these instants are due to additional network activity to eDonkey traffic.

In Figure 3, the up-rate of a user in CE traces during the first hour of the capture is shown. The up-rate presents two well defined sections: near to zero, and around 30kB/s. During the first 30 minutes the up-rate is near to zero because only a few nodes know the existence of the monitored peer. After that, we can observe the mentioned constant behavior around 30kB/s. The up-rate filtered is also presented in the same figure, which allows us to appreciate the suppression of values that extremely deviate from the expected ones.

Finally, the KL distance is also presented in this figure and is divided in three parts: two sections near zero, separated by a clear increase of the KL distance. KL distances near zero represent the constant up-rate sections, and a relative maximum of the KL distance indicates a change in the up-rate distribution. The second section of the near zero KL distance will be identified as generated by an eDonkey node because it presents a constant up-rate different to zero.

Additionally, a study using all the network traces has been carried out to select a value for the threshold $Thres_{KL}$, and a length of the window, N, for the median filter and the KL distance. There exists a wide range of $Thres_{KL}$ values that present a high detection rate in CE traces and a low false positive one in UT and HS traces. This range is $[10^3, 10^4]$ and the value for $Thres_{KL}$ finally selected was 8,000. On the other hand, the selected value for N was 5 minutes because this is the minimum burst of constant traffic to be detected as generated by an eDonkey node. Bursts of traffic from eDonkey nodes with a duration less than 5 minutes are not interesting in this work, because we try to detect nodes with a substantial consumption of bandwidth network.

The execution of nodes detection method in CE traces indicates that these nodes are classified as generators of eDonkey traffic during a 86.10% of the monitored time. The rest of the time is mainly constituted of instants at which clients stop to share with certain client and consequently

 ISBN: 978-1-61208-173-1

their up-rates will not be constant.

Finally, the up-rate of HTTP server has also been analyzed. As we see in Figure 4, the up-rate do not present a constant behavior, so this node was classified as eDonkey generator only a 1.687% of the total monitored time (168 hours in total).

VII. Conclusions

In this paper, two methods to detect eDonkey traffic without inspecting packet payload have been proposed: *(i)* an eDonkey flow detection and, *(ii)* an eDonkey node detection.

The experimental results obtained allow us to conclude that the proposed detection hypotheses are acceptable in the case of eDonkey protocol. Moreover, both detection methods present a high classification rate and a low number of false positives. Finally, we specify that the eDonkey flow detection process is valid for file sharing flows from eDonkey nodes with high ID.

Additionally, in a near future we plan to raise two work lines:

- To combine the information obtained by both methods in order to increase the detection rate and to reduce false positive rate.
- To explore the possibility of detecting other P2P protocols used in file sharing applications, *e.g*, Kademlia or BitTorrent, through the execution of the second method (eDonkey flow detection). We think that this method could detect protocols used in P2P file sharing applications because they saturate the up-rate of users and that implies the necessity of a limitation.

Acknowledgement

This work has been partially supported by Spanish MICINN under project TEC2008-06663-C03-02.

References

[1] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A Survey on Internet Traffic Identification," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 37–52, Aug. 2009.

[2] A. Feldmann, "A possibility for ISP and P2P collaboration," in *Broadband Communications, Networks and Systems, 2008. BROADNETS 2008. 5th International Conference on*, Sep. 2008, p. 239.

[3] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ser. IMC '04. New York, NY, USA: ACM, 2004, pp. 121–134.

[4] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '05, vol. 35, no. 4. New York, NY, USA: ACM, 2005, pp. 229–240.

[5] K. Xu, M. Zhang, M. Ye, D. M. Chiu, and J. Wu, "Identify P2P traffic by inspecting data transfer behavior," *Computer Communications*, vol. 33, no. 10, pp. 1141–1150, Jun. 2010.

[6] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing skype traffic: when randomness plays with you," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '07. New York, NY, USA: ACM, 2007, pp. 37–48.

[7] J. Ramirez, J. Segura, C. Benitez, A. de la Torre, and A. Rubio, "A new kullback-leibler vad for speech recognition in noise," *Signal Processing Letters, IEEE*, vol. 11, no. 2, pp. 266–269, Feb. 2004.

[8] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proceedings of the IEEE*, vol. 78, no. 4, pp. 678–689, Apr. 1990.

[9] Opendpi. http://www.opendpi.org/ [Last accessed in September 26, 2011].

[10] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ser. IMC '06, 2006, pp. 189–202.

 ISBN: 978-1-61208-173-1