Exploring Graph-based Network Traffic Monitoring

Marios Iliofotou
Department of Computer Science
University of California, Riverside
Email: marios@cs.ucr.edu

***Abstract*—Monitoring network traffic and classifying applications are essential functions for network administrators. These tasks are becoming increasingly challenging since (a) many applications obfuscate their traffic using nonstandard ports, and (b) new applications constantly appear. This suggests the need for a behavioral-based approach, where the detector looks for fundamental behaviors of the application that are both intrinsic to the application and distinct from normal traffic. Identifying intrinsic behaviors makes it difficult for application writers to disguise such behaviors without defeating the very purpose of the application. In this paper, we propose a graph-based representation of network traffic which captures the network-wide interactions of applications. In these graphs, nodes are individual IP address and edges between nodes represent particular communications. For example, an edge might represent the exchange of a single packet, or the exchange of at least ten packets of any type. We call such graphs "Traffic Dispersion Graphs" or TDGs [3]. As a proof of concept we show that our proposed graph-based classifier out-perfoms BLINC [4] in detecting P2P traffic on backbone links. Our results are very promising, showing that TDGs can provide the basis for the next generation of network monitoring tools.**

I. Introduction

Modeling network traffic as a graph has recently attracted significant attention. Initial studies, have only explored the capabilities of TDGs for the detection of worm activity [1] and the grouping of related hosts in enterprise networks [5]. In addition, prior efforts only focused on TDGs extracted from a single network. The main goals of this work is to explore the full potential of TDGs over a large set of applications, over long time duration, and different locations of observation.

In more detail, this work comprises of two thrusts. First, we target the selection of: (a) the best methods to create and use TDGs in practice, and (b) the best metrics to model and summarize TDGs. Towards this end we experimented with a large set of real-world network traces collected from a diverse set of network links, some being of extended duration ranging from 2000 to 2008. Second, using the TDG profile of well known applications, we design a P2P classification framework [2], which is a systematic methodology for classifying network traffic using TDGs.

II. TDGs: Practical Considerations

The definition of what constitutes an edge in the graph is flexible and can change depending on the application at hand. We refer to the processes of selecting which nodes are connected in a TDG as *Edge Filtering*. The edges in TDGs can be annotated to store information regarding the interaction between hosts. Such information can be the number of flows/bytes/packets exchanged between two IPs.

In the basic case, an edge can represent the exchange of a single packet between two IPs. Other edge filters can collect traffic that belongs to a particular application. For example, under the assumption that all flows (defined using the 5-tuple {`srcIP,srcPort,dstIP,dstPort,Proto`}) with port number 53 belong to the DNS application, an edge filter using this port can be used to form the DNS TDG. Similarly, we can choose filters to generate TDGs for a large range of applications, e.g., SMTP, Gnutella, Bittorrent, Web, etc. A filter can potentially utilize all attributes of a flow, such as: (a) the payload, (b) port numbers, and (c) packet sizes and packet timing (e.g., inter packet gap) information.

To model and summarize TDGs we experimented with a large set of both standard and novel metrics. We can group these metrics in two groups: (a) static metrics that capture the structure of the graph, and (b) dynamic metrics that can describe the changes of TDGs in time. Static graph metrics provide statistical summaries of TDGs. Such metrics include the degree distribution, degree correlation, distances, etc. Preliminary results of this study can be found in [3]. Dynamic metrics introduce time as a parameter in the study of TDGs. For example, a dynamic metric can quantify how much the graph has changed over the past two hours. An in depth study of dynamic properties is part of our ongoing work.

We studied a large set of metrics (static, and dynamic) over a large set of real-world traces. We here summarize some key observations. The graphs formed within five-minute intervals generate graphs with stable graph structure, compared to graphs generated over shorter intervals. Such five-minute long graphs can allow the separation between different classes of applications, as we show in Section III in more detail. Another interesting observation is that TDGs show periodic and predictable behavior over the length of a day and over weekends. Finally, even though TDGs can have quantitatively different metric values over different locations, their structure is distinctive enough to lead to qualitatively similar observations.

III. Graph-based P2P Traffic Classification

The key idea in our graph-based classifier, is that TDGs generated by different applications have characteristic structure that we can use to distinguish between them. For example, in Figure 1 we show the Gnutella (P2P) and Web TDGs. Clearly, the two graphs even visually look different. To select the best

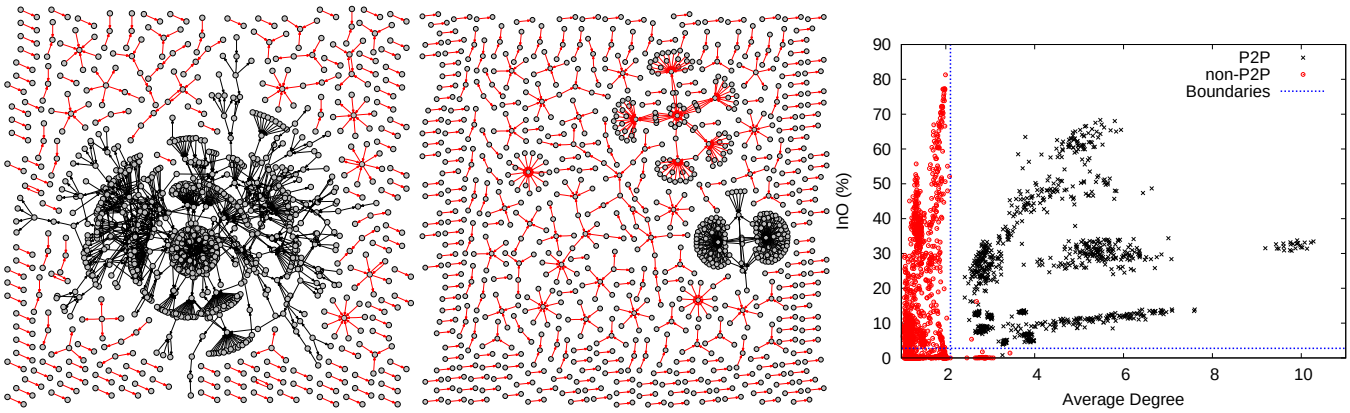

Fig. 1. **Left:** The Gnutella TDG (P2P). **Center:** The Web TDG (non-P2P). **Right:** Scatter plot showing how few metrics can distinguish between TDG classes. The InO metric captures the percentage of nodes that have both in-coming and out-going connections. The average degree, is the average number of neighbors of a node in the graph.

metrics to model P2P TDGs, we used standard data mining (feature selection) methods and traces from a diverse set of links. The data set comprise of three Tier-1 ISPs and five backbone links from Abilene Internet2 backbone. We observed that few static graph metrics are robust in time and space so they can be used to distinguish between applications. An example is illustrated in Figure 1 (Right), where two metrics are suffice to distinguish between the majority of P2P and non-P2P TDGs.

Even though some non-P2P applications have graphs that look like P2P, we observed that using dynamic graph metrics we can still distinguish between the two. For example, DNS TDGs are similar in structure with P2P applications (e.g., FastTrack, Gnutella, etc.), but they are very different in how the graphs change over time. In particular, DNS TDGs are more stable and change less frequent than P2P TDGs.

Our proposed methodology consists of three steps. At step one, we use data mining methods (K-means) to cluster flows into groups based on their similarity. For example, similar flows can be those with a common application-layer header. Intuitively, similar flows will belong to the same application. Therefore, the final output of this step will be groups of flows where each group has flows from a single application. At the second step, a TDG is formed for each group. Using the empirical profile of P2P TDGs (Figure 1), our method classifies each group as P2P or non-P2P. Finally, as an optional step, we can use statistical methods to extract a distinctive signature for each application. For example, such signature can be a regular expression (regex) matching part of the payload. All three steps can be performed offline.

The evaluation on backbone traces shows that our method can detect over 90% of P2P traffic with over 95% accuracy. Detailed results for the main P2P protocols are presented in Figure 2. By comparing our method with BLINC [4], we observe that our approach detects overall 7% more P2P traffic with 6% higher accuracy on the same link. BLINC has significantly lower performance for some P2P applications. For example, we detect 95% of BitTorrent traffic, while BLINC detects only 25%!

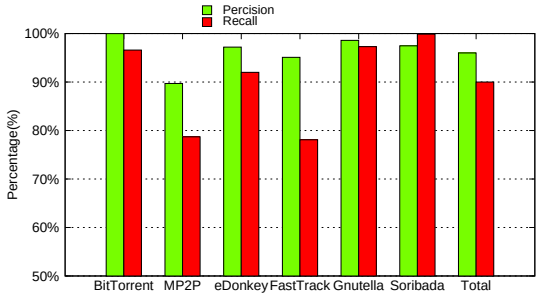

Fig. 2. Classification precision and recall for the six main P2P protocols. Recall, is the percentage of traffic from each class that was detected as P2P. Precision denotes the percentage of times we classify a flow as, e.g., Gnutella, and the flow is actually Gnutella.

IV. Summary & Conclusions

Even though modeling network traffic as a graph has been proposed before [1], [5], we are the first to study TDGs in depth and explore their full potential. To the best of our knowledge, our graph-based approach is the first method to use the network-wide behavior of applications in order to classify network traffic. We believe TDGs will provide another useful tool in the arsenal of traffic analysis and monitoring techniques.

Acknowledgments

The author would like to thank all his collaborators: Michalis Faloutsos, George Varghese, Michael Mitzenmacher, Prashanth Pappu, Sumeet Singh, Hyun-chul Kim, and CAIDA.

References

[1] D. Ellis, J. Aiken, K. Attwood, and S. Tenarglia. A Behavioral Approach to Worm Detection. In *ACM CCS WORM*, 2004.

[2] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, H. Kim, and G. Varghese. Graption: Automated Detection of P2P Applications in the Internet Backbone. Technical report, UC Rverside, 2008.

[3] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese. Network Monitoring Using Traffic Dispersion Graphs (TDGs). In *ACM IMC*, 2007.

[4] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multi-level Traffic Classification in the Dark. In *ACM SIGCOMM*, 2005.

[5] G. Tan, M. Poletto, J. Guttag, and F. Kaashoek. Role classification of hosts within enterprise networks based on connection patterns. In *USENIX Annual Technical Conference*, 2003.