Graph-based P2P Traffic Classification at the Internet Backbone

Marios Iliofotou*, Hyun-chul Kim†, Michalis Faloutsos*,
Michael Mitzenmacher§, Prashanth Pappu¶, and George Varghese‡
* University of California, Riverside
† CAIDA and Seoul National University
‡ University of California, San Diego
§ Harvard University
¶ Conviva, Inc.

***Abstract*—Monitoring network traffic and classifying applications are essential functions for network administrators. In this paper, we consider the use of Traffic Dispersion Graphs (TDGs) to classify network traffic. Given a set of flows, a TDG is a graph with an edge between any two IP addresses that communicate; thus TDGs capture network-wide interactions. Using TDGs, we develop an application classification framework dubbed Graption (*Grap*h-based classifica*tion*). Our framework provides a systematic way to harness the power of network-wide behavior, flow-level characteristics, and data mining techniques. As a proof of concept, we instantiate our framework to detect P2P applications, and show that it can identify P2P traffic with recall and precision greater than 90% in backbone traces, which are particularly challenging for other methods.**

I. Introduction

An important task when monitoring and managing large networks is classifying flows according to the application that generates them. Such information can be utilized for network planning and design, QoS and traffic shaping, and security. In particular, detecting P2P traffic is a potentially important problem for ISPs that want to manage such traffic, and for specific groups such as the entertainment industry in legal and copyright disputes. Detecting P2P traffic also has particular interest since it represents a large portion of the Internet traffic, with more than 40% of the overall volume in some networks [11].

Most current application classification methods can be naturally categorized according to their level of observation: payload-based signature-matching methods [16], [14], flow-level statistical approaches [6], [18], or host-level methods, such as BLINC [13], [24]. Each existing approach has its own pros and cons, and no single method clearly emerges as a winner. Relevant problems that need to be considered include identifying applications that are new, and thus without a known profile; operating at backbone links [2], [13]; and detecting applications that intentionally alter their behavior. Flow-level and payload-based approaches require per application training and will thus not detect traffic from emerging protocols. Host-based approaches can detect traffic from new protocols [13], but have weak performance when applied at the backbone [2]. In addition, most tools including BLINC [13] (which has 28 parameters) require fine-tuning and careful selection of parameters [2]. We discuss the limitations of previous methods in more detail in §IV.

In this paper, we use the network-wide behavior of an application to assist in classifying its traffic. To model this behavior, we use graphs where each node is an IP address, and each edge represents a type of interaction between two nodes. We use the term **Traffic Dispersion Graph** or **TDG** to refer to such a graph [10]. While we recognize that some previous efforts [3], [5] have used graphs to detect worm activity, they have not explored the full capabilities of TDGs for application classification.

We propose a classification framework, dubbed **Graption**, as a systematic way to combine network-wide behavior and flow-level characteristics. Graption first *groups* flows using flow-level features, in an unsupervised and agnostic way, i.e., without using application-specific knowledge. It then uses TDGs to *classify* each group of flows. As a proof of concept, we instantiate our framework and develop a P2P detection method, which we call "*Graption-P2P*". Compared to other methods, Graption-P2P is easy to configure and requires very little a priori knowledge (mainly a few intuitive parameters).

The experimental part of our paper shows that:

- Graption-P2P identifies over 90% of P2P traffic with precision greater than 95% in backbone traces.
- Graption-P2P performs better than BLINC in P2P identification at the backbone. For example, Graption-P2P identifies 95% of BitTorrent traffic while BLINC identifies only 25%.
- Even a single backbone link contains enough information to generate TDGs that can be used to classify traffic. In addition, TDGs of the same application seem fairly consistent across different times and locations.

The rest of the paper is organized as follows. In §II we define TDGs, and identify TDG-based metrics that differentiate between applications. In §III we present the Graption framework and our instantiation, Graption-P2P. In §IV we discuss related work. In §V we conclude the paper.

II. Traffic Dispersion Graphs

Definition. Throughout this paper, we assume that packets can be grouped into flows using the standard 5-tuple {`srcIP,`

Name	***Date/Time***	***Duration***	***Flows***
TR-PAY1	2004-04-21/17:59	1 hour	38,808,604
TR-PAY2	2004-04-21/19:00	1 hour	37,612,752
TR-ABIL	2002-09 /(N/A)	1 month	2,057,729

TABLE I
SET OF BACKBONE TRACES FROM THE COOPERATIVE ASSOCIATION FOR INTERNET DATA ANALYSIS (CAIDA). STATISTICS FOR THE TR-ABIL TRACE, ARE REPORTED ONLY FOR THE FIRST FIVE-MINUTE INTERVAL.

`srcPort, dstIP, dstPort, protocol`}. Given a group of flows S, collected over a fixed-length time interval, we define the corresponding TDG to be a directed graph $G(V, E)$, where the set of nodes V corresponds to the set of IP addresses in S, and there is a link $(u, v) \in E$ from u to v if there is a flow $f \in S$ between them.

In this paper, we consider bidirectional flows. We define a TCP flow to start on the first packet with the `SYN` flag set (referred to as the `SYN`-packet), so that the initiator and the recipient of the flow are defined for the purposes of direction. For UDP flows, direction is decided upon the first packet of the flow.

Data Set. To study TDGs, we use three backbone traces from a Tier-1 ISP and the Abilene (Internet2) network. These traces are summarized in Table I. All data are IP anonymized and contain traffic from both directions of the link. The TR-PAY1 and TR-PAY2[1] traces were collected from an OC48 link of a commercial US Tier-1 ISP at the Palo Alto Internet eXchange (PAIX). The TR-ABIL trace is a publicly available data set collected from the Abilene (Internet2) academic network connecting Indianapolis with Kansas City. The Abilene trace consists of five randomly selected five-minute samples taken every day for one month, and covers both day and night hours as well as weekdays and weekends.

Ground Truth. We used a Payload-based Classifier (PC) to establish the ground truth of flows for the TR-PAY1 and TR-PAY2 traces. Both traces contain up to 16 bytes of payload in each packet, thereby allowing the labeling of flows using the signature matching techniques described in [2], [13]. Running the PC over the TR-PAY1 and TR-PAY2 traces we find 14% of the traffic to be P2P, 28% Web, 6% DNS, and the rest to belong to other applications, such as Email, FTP, NTP, SNMP, etc. For our study, we remove the 2% of traffic that remained unclassified and the 28% that contained no payload.

A. Identifying P2P TDGs

Identifying the right metrics to compare graph structures is a challenging question that arises in many disciplines [17]. Our approach is to consider several graph metrics, each capturing a potentially useful characteristic, until a set of metrics is found that distinguishes the target graphs.

To select an appropriate set of metrics, we generate a large number of TDGs using all our traces (Table I), thus observing TDGs over two different locations at the backbone. For the TR-PAY1 and TR-PAY2 traces, we use the payload-based classifier (PC) in order to select which flows belong to each TDG. Since the TR-ABIL trace does not have any payload information, we use port numbers [2] to assign flows to applications. We can use port numbers for the TR-ABIL trace since it was collected in 2002 where most P2P applications used their default port numbers [7], [12]. We only use the TR-ABIL trace to verify our TDG observations over a second location in the backbone and we do not use it in the final evaluation of our classifier. By using the month-long TR-ABIL trace, we can study the consistency of TDGs over different times of the day and over weekdays and weekends.

We observe TDGs over 5-minute intervals. This interval length gives good classification results and stability of TDG metrics over time. For each TDG we generate a diverse set of metrics. Our metrics capture various aspects of TDGs including the degree distribution, degree correlations, connected components, and distance distribution. For additional details about these metrics we refer the reader to [9], [17].

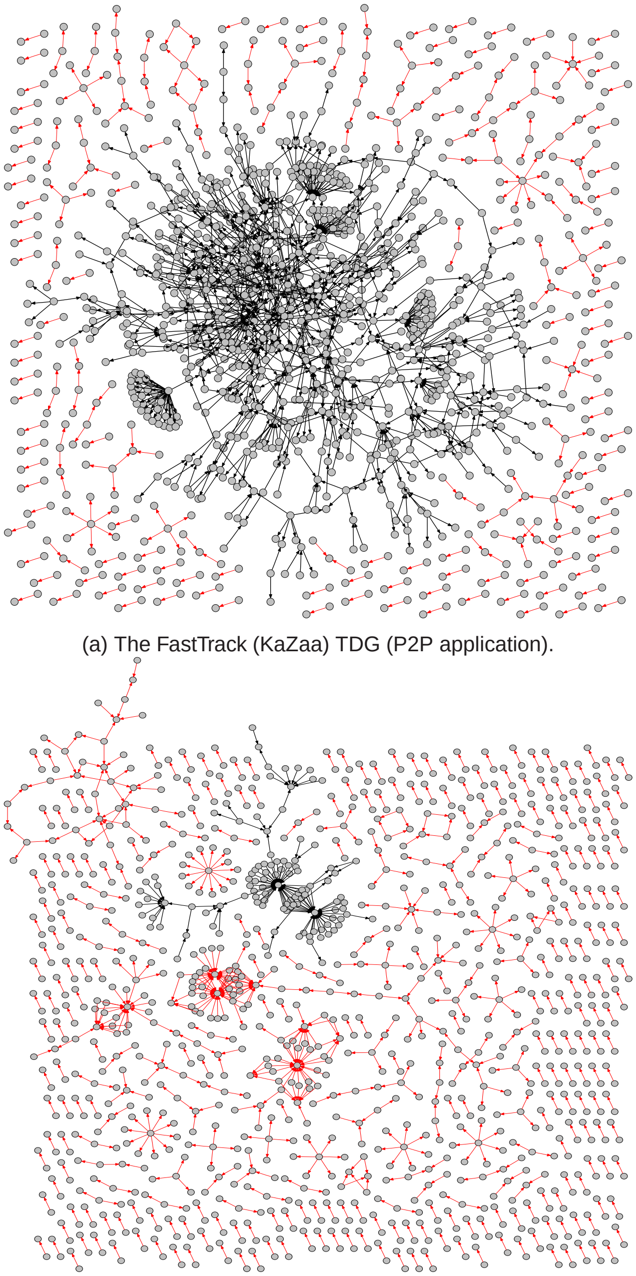
To select the right set of metrics we use various graph visualizations and trial and error. Finding a less ad hoc approach is beyond the scope of this work. Two TDG visualization examples are shown in Figure 1. We see that FastTrack (P2P) has a denser graph than HTTPS, or a higher **average degree**, where the average node degree $\bar{k}$ is given by $\bar{k} = 2|E|/|V|$.

We utilize two other metrics that capture the directionality of the edges in the graph and the distances between nodes. The directionality is useful since we know that pure clients only initiate traffic, pure servers should never initiate traffic, and that some P2P nodes play both roles. To capture this quantitatively, we define **InO** to be the percentage of nodes in the graph that have both incoming and outgoing edges.

The distance between two nodes is defined as the length of their shortest path in the graph. The diameter of a graph is defined as the maximum distance between all pairs of nodes, which is sensitive as a metric [17]. For a more robust metric, we use the **effective diameter** (EDiam), which we define as the 90-th percentile of all pairwise distances in the graph.

From our measurements, we empirically derive the following two rules for detecting P2P activity. **Rule 1**: $\bar{k} > 2.8$ and $InO > 1\%$; **Rule 2**: $InO > 1\%$ and $EDiam > 11$. With these simple rules, we can correctly identify *all P2P TDGs from both backbone locations* (Abilene backbone and Tier-1 ISP). Intuitively, P2P hosts need to be connected with a large set of peers in order to perform tasks such as answering content queries and sharing files, which can explain the higher average degree compared to client-server applications. An additional characteristic of P2P applications is the duality of roles, with many hosts acting both as client and server. The duality of roles is in turn captured by the high InO value. We further speculate that the decentralized architecture of some P2P applications (such as BitTorrrent), can explain the high diameters in some P2P TDGs. Additional speculations on why these three metrics effectively capture P2P behavior is provided in [9] and are omitted due to space limitations.

[1] The authors thank CAIDA for providing this set of traffic traces. Additional information for these traces can be found in the DatCat, Internet Measurement Data Catalog [26], indexed under the label “PAIX”.



(a) The FastTrack (KaZaa) TDG (P2P application).

(b) The HTTPS TDG (client-server application).

Fig. 1. Two TDG visualization contrasting a P2P with a client-server application. Largest component is with bold edges.

Distinguishing collaborative applications from P2P: Some well-known applications other than P2P exhibit collaborative behavior, such as DNS and SMTP. This is not surprising since in these applications servers communicate with each other and with other clients (high $\bar{k}$), and servers act both as clients and servers (high InO). This is exactly what our metrics are set out to detect. It has been reported recently [2] that port numbers are fairly accurate in identifying such legacy applications, although they fail to identify P2P and other applications with dynamic use of port numbers. Therefore, one could use legacy ports to pinpoint and isolate such collaborative applications and then use graph metrics on the remaining traffic. In addition to port inspection, we can also examine the payload of a flow in order to verify that it follows the expected application-layer interactions. As a future work, our goal is to select metrics that can further help to separate between collaborative applications (e.g., DNS) and P2P. We discuss similar topics again is §III-C.

We do not claim that our thresholds are universal, but our measurements suggest that small adjustments to these simple parameters allow our methodology to work on different backbone links. Furthermore, the three thresholds (InO, EDiam, and average degree) are observed to remain stable over time.

III. The Graption Framework

The Graption framework consists of the following three steps.

Step 1. Flow Isolation. The input is network traffic in the form of flows as defined in §II. The goal of this first optional step is to utilize external information to isolate any flows that can already be classified. This knowledge could be based on payload signatures, port numbers, or IP address (e.g., exclude flows from a particular domain such as `google.com`).

Step 2. Flow Grouping. We use similarity at the flow and packet level to group flows. The definition of similarity is flexible in our proposed methodology. We can use flow statistics (duration, packet sizes, etc.) or payload if this is available. Eventually, the output of this step is a set of groups with each group ideally containing flows from a single application (e.g., Gnutella, NTP, etc.). However, at this step, the exact application of each group is not known.

Step 3. Group Classifier. For each group of flows, we construct a TDG. Next, we quantify each TDG using various metrics. The classifier uses these metrics to identify the application for each group of flows. For the classification decision, we use a set of rules which in general depend on the focus of the study.

Although this paper focuses on P2P detection, Graption can be used for general application classification by choosing metrics and parameters appropriately. We next describe how we specialized Graption to detect P2P traffic (Graption-P2P).

A. Implementation Details of Graption-P2P

Step 1. This is an optional step in our methodology. Experiments without this step are discussed later in the section. Recent work [2] suggests that port-based classification works very well for legacy applications, as legacy applications use their default ports and tunneling of P2P at such ports is not very common. Thus, in this study, we isolated flows with port 80 for Web, port 53 for DNS, and port 25 for SMTP. These applications turn out to be about 65% of the total number of flows. In our traces, the proportion of P2P actually using one of these ports is as low as 0.1%.

Step 2. To implement flow grouping we use the fact that application-level headers are likely to recur across flows from the same application. Therefore, payload similarity can be used to group flows. In Graption-P2P, we only use the first sixteen bytes from each flow. As we show, sixteen bytes are sufficient

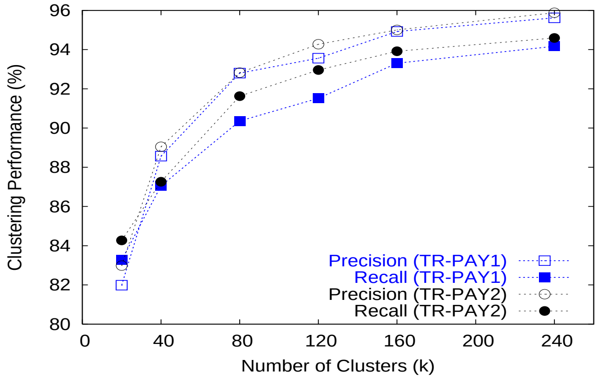


Fig. 2. Evaluating K-means.

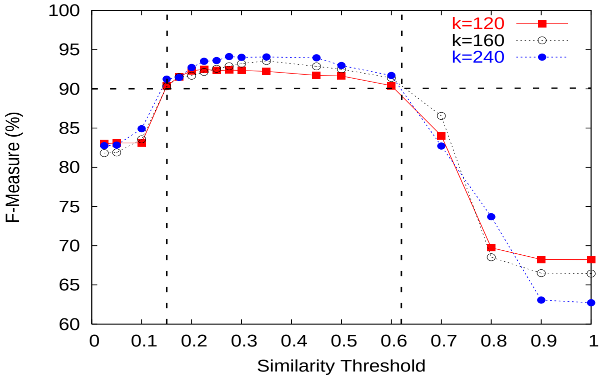


Fig. 3. Graption-P2P achieves $> 90\%$ F-Measure over a large range of similarity thresholds and number of clusters (k).

to give very good classification results. This observation agrees with findings in [16], [14]. Even though we use the payload bytes, our grouping is agnostic to application semantics, as each byte is considered as a single independent categorical feature. We consider each byte as a single categorical feature in the range $\{0, 1, ..., 255\}$.

The flow grouping step comprises two substeps: cluster formation and cluster merging.

a. Forming Clusters. Given the set of discriminating features, the next step is to cluster "similar" flows together. We use the term **cluster** to describe the outcome of an initial grouping using the selected features. Clusters may be merged in the next function of this step to form **groups**, which produces the final output.

Feature-based clustering is a well-defined statistical data mining problem. For this task we used the popular **K-means** algorithm [22]. This algorithm has been commonly used for unsupervised clustering of network flows [6], [16], with very good results and low computational cost. K-means operates with a single parameter that selects the number of final clusters (k). As we show later in our evaluation, our classifier gives very good results over a large range of k.

The similarity between two flows is measured by the *Hamming* [22] distance calculated over the 16 categorical features (i.e., the payload bytes). Even though more involved similarity measures such as edit-distance (also known as Levenshtein distance) exist, Hamming distance has been used successfully before [8] and performs very well in our application.

b. Cluster merging. During clustering, it is likely that the same application generates multiple clusters. For example, many P2P protocols exhibit a variety of interaction patterns, such as queries (UDP flows) and file transfers (TCP flows), each with significantly different flow and packet characteristics [12].

This motivates mergings clusters that we expect to belong to the same application into groups. This grouping provides a more complete view of the application and aides in understanding the structure of the P2P protocol, as we show in §III-B.

Cluster merging cannot be based on the chosen set of flow-level features that were already used to create the clusters originally. Instead, in the case of a P2P protocol, it is natural to assume that the TDGs corresponding to each cluster of the same protocol would share a large number of common nodes (IP addresses).

Based on these observations, we use an Agglomerative (Hierarchical) Clustering Algorithm that recursively merges clusters with significant similarity in IP addresses. We used the following metric to calculate similarity between clusters: $Sim(C_1, C_2)$ = (Number of flows having their source or destination IPs present in both clusters) / (The number of flows of the smaller cluster). The cluster merging process starts by hierarchically merging clusters with high similarity and stops when the similarity between all new cluster pairs is below a **similarity threshold (ST)**. As we show later in our evaluation, our classifier gives very good results over a large range of similarity thresholds.

Step 3. The outcome of the previous step is a set of groups of flows, with each group consisting of flows that we hope stem from a single application. In order to classify each group, we generate a TDG on the group in the same way as described in §II. Each group yields a TDG that can be summarized using graph metrics. To identify P2P TDGs, we used the rules extracted from §II-A. When a group is labeled as P2P then all the flows of that group are classified as P2P flows.

B. *Evaluating Graption-P2P*

To evaluate Graption-P2P, we use traces TR-PAY1 and TR-PAY2, where we have the ground truth using the payload classifier (§II). We compute the True Positives, False Positives, and False Negatives. The True Positives (TP) measures how many instances of a given class are correctly classified; the False Positives (FP) measures how many instances of other classes are confused with a given class; and the False Negatives (FN) measures the number of misclassified instances of a class. In our comparisons, we used the following standard metrics: **Precision** (P), defined as $P = TP/(TP + FP)$; **Recall** (R), defined as $R = TP/(TP + FN)$; and the **F-Measure** [22], defined as $F = 2P \cdot R/(P + R)$, combining Precision and Recall.

We first test the ability of K-means to generate clusters with flows from a single application. After forming the clusters

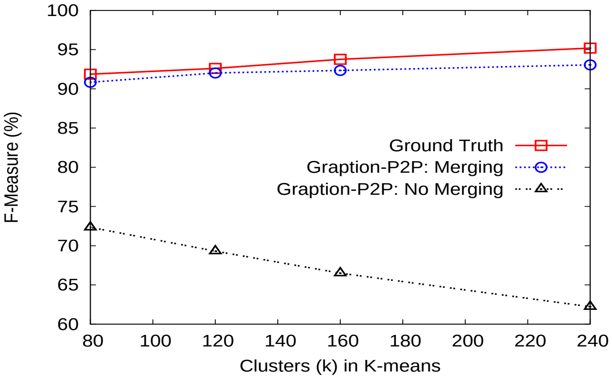


Fig. 4. Graption-P2P with and without cluster merging. Results are also compared with cluster labeling based on ground truth.

with K-means, we use the ground truth and label each cluster as belonging to the application with the majority of flows. All the flows of a cluster are then classified to belong to this dominant application. The P and R of K-means as we increase k for both traces are shown in Figure 2. We observe that with sufficiently large k (> 120) we achieve very good results with P and R above 90%.

Using Graption-P2P, we achieve high F-Measure over a range of values of k (K-means) and similarity thresholds (ST). We show this in Figure 3, where we vary the ST from 0.01 to 1 and use a sufficiently large k (see Figure 2). All experiments are averaged over each disjoint 5 minute interval of both traces. Intuitively, by using a very large ST, the clusters of an application are not grouped together, which results to TDGs that are harder to classify as P2P. On the other hand, with a very small ST, clusters belonging to different applications are merged together leading to poorer classification performance. The results in Figure 3 show that we achieve good classification performance $(> 90\%$ F-Measure), over a large range of similarity thresholds and number of clusters (k).

In Figure 4, we compare our approach with labeling each cluster using the ground truth (i.e. without merging any clusters and labeling each cluster based on the dominant application). Intuitively, for a given clustering of flows, the ground truth shows the best that any cluster labeling mechanism can achieve. For merging, we use an ST of 0.5. From Figure 4, we see that Graption-P2P deviates only slightly from labeling clusters using the ground truth. In the same plot, we also compare Graption-P2P without the cluster merging step, highlighting the benefit of merging clusters of the same application together.

Using a ST of 0.5 and $k = 160$, Graption-P2P achieves above 90% Recall and above 95% Precision over all disjoint 5 minute intervals for both traces. To apply Graption-P2P to other backbone link, the same selection processed can be repeated to adjust the values of ST and k. Our experiments show that the classification performance can degrade with a very bad choice of parameters. However, as shown in Figures 2, 3, and 4, for reasonable choices for k and ST, our method provides very good results.

C. Discussion

Comparison with BLINC [13]. We used BLINC to classify traffic over both TR-PAY1 and TR-PAY2 traces. BLINC was optimized after several trial and error efforts to achieve its best accuracy over these traces, as described in [2]. The Recall and Precision for BLINC are 84% and 89% respectively. In particular, BLINC has significantly lower performance for some P2P applications. For example, Graption-P2P detects 95% of BitTorrent traffic, while BLINC detects only 25%! Our experiments suggest that BLINC and possible other hostbased approaches work well when applied at the edge, where a large fraction of host flows are observed and hence enough evidence is collected to profile each node. However, this is not always true for backbone monitoring points which can explain BLINC's lower performance. These observations are also supported by findings in [2].

Other Configurations. We have tested Graption-P2P without using payload under the assumption that payload is encrypted. For grouping flows we used packet size information (i.e. min, max, and the size of the first five packets) and protocol (UDP or TCP). Our method performed comparably well with R and P above 88% in both traces. We also experimented without using Flow Isolation. To achieve good results (P, R $>$ 85%) we had to increase k (> 300) in K-means, which made cluster merging more challenging. More details are omitted due to space limitations. Evaluating Graption with other configurations and different data mining clustering algorithms is included in our future work.

Enhancing Isolation. To improve isolation we can enforce payload inspection in addition to port-based filtering. For example, we can test all DNS flows at port 53 to see if they also have a DNS payload signature or if another protocol is tunneling its traffic under the DNS port. If payload is encrypted, then we can choose to use flow-level feature such as packet sizes [2] or white-listed IP addresses [21].

IV. Related Work

Traffic Classification. As an alternative to port-based methods, some works used payload [16], [14]. Other approaches use Machine Learning (ML) methods to classify traffic using flow features (e.g, packet sizes). For an exhaustive list and comparison of ML methods we refer the reader to [18] and [2]. Our work has more in common with unsupervised data mining methods which group similar flows together. All previous methods [15], [6] require manual labeling of clusters. Our work bridges this gap by providing a method to automatically label clusters of flows based on their network-wide behavior.

In BLINC [13], the authors characterize the connection patterns (e.g., if it behaves like using P2P) of a single host at the Transport Layer and use these patterns to label the flows of each host. BLINC uses graph models called graphlets to model a host's connection patterns using port and IP cardinalities. Unlike TDGs, graphlets do not represent *network-wide* host interaction. In some sense, TDGs represent a further level of aggregation, by aggregating across hosts as well. Thus it is perhaps fair to say that while BLINC hints at the benefit of analyzing the node's interaction at the "social" level, it ultimately follows a different path that focuses on the behavior of individual nodes. As we show, our approach performs better than BLINC in our backbone traces.

Similar to BLINC, other host-based method [1], [23] target the identification of P2P users inside a university campus (i.e., network edge). Unlike Graption, in [1], [23], [13] they do not use *network-wide* host interaction. In [4], the authors use a port-based method to identify P2P users, using their temporal appearance and connection patterns in a trace.

The most recent host profiling method is by Trestian et al. [21]. They used readily available information from the Web to classify traffic using the Google search engine. They show very good results for classifying flows for legacy application, but their results are not promising for P2P detection because of the dynamic nature of P2P IP hosts. Our method can be used to complement the work in [21].

Worm Detection. Graphs have been used for detecting worm activities within enterprise networks [5]. Their main goal was to detect the tree-like communication structure of worm propagation. This characteristic of worms was also used for post-mortem trace analysis (for the identification of the source of a worm outbreak, the so-called patient zero) using backbone traces [25]. More recent studies use graph techniques to detect hit-list worms within an enterprise network, based on the observation that an attacker will alter the connected components in the network [3].

Measurements. Statistical methods are used in [24] for automating the profiling of network hosts and ports numbers. The connectivity behavior and habits of users within enterprise networks is the focus of many papers, including [20]. In [19], the authors study P2P overlays using passive measurements, but target mainly the profiling of P2P hosts. None of the above papers targets P2P detection.

V. Summary and Conclusions

The underlying theme of our work is to go beyond just monitoring individual packets, flows, or hosts, to *also* monitoring the interactions of a group of hosts. Towards this end, we introduce TDGs and show their potential to generate novel tools for traffic classification. Based on TDGs, we developed Graption, a graph-based framework, which we then specialize (Graption-P2P) for the detection of P2P applications. We show that Graption-P2P classifies more than 90% of P2P traffic with above 95% precision when tested on real-world backbone traces.

Acknowledgments

This work was supported by NSF grant NETS-0721889, a Cisco Systems, Inc. URP grant, the IT R&D program [2007-F-038-02, Fundamental Technologies for the Future Internet], and the ITRC support program [IITA-2009-C1090-0902-0006] of MKE/IITA. The SDSC's TeraGrid and compute resources are supported by the NSF grant CONMI CRI-0551542. Michael Mitzenmacher was supported in part by NSF grant CNS-0721491 and research grant from Cisco Systems, Inc. Support for CAIDA's Internet traces is provided by the National Science Foundation, the US Department of Homeland Security, CAIDA Members, and the DatCat system.

References

[1] Barlett, G., Heidemann, J., and Papadopoulos, C. Inherent Behaviors of On-line Detection of Peer-to-Peer File Sharing. In *IEEE Global Internet* (2007).

[2] Hyun-chul Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., and Lee, K. Internet Traffic Classificatoin Demystified: Myths, Caveats, and the Best Practices. In *ACM CoNEXT* (2008).

[3] Collins, M. P., and Reiter, M. K. Hit-List Worm Detection and Bot Identification in Large Networks Using Protocol Graphs. In *RAID* (2007).

[4] Constantinou, F., and Mavrommatis, P. Identifying known and unknown peer-to-peer traffic. In *IEEE NCA* (2006).

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

[6] Erman, J., Arlitt, M., and Mahanti, A. Traffic classification using clustering algorithms. In *ACM MineNet* (2006).

[7] Gerber, A., Houle, J., Nguyen, H., Roughan, M., and Sen, S. P2P, the Gorilla in the Cable. In *NCTA* (2003).

[8] Haffner, P., Sen, S., Spatscheck, O., and Wang, D. ACAS: automated construction of application signatures. In *ACM MineNet* (2005).

[9] Iliofotou, M., Pappu, P., Faloutsos, M., Mitzenmacher, M., Kim, H., and Varghese, G. Graption: Automated Detection of P2P Applications in the Internet Backbone. Tech. rep., UC Riverside, 2008.

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

[11] IPOQUE. Internet study 2007 file hosters like rapidshare and streaming services like youtube. 2007.

[12] Karagiannis, T., Broido, A., Brownlee, N., kc claffy, and Faloutsos, M. Is p2p dying or just hiding? In *IEEE GLOBECOM* (2004).

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

[14] Ma, J., Levchenko, K., Kreibich, C., Savage, S., and Voelker, G. M. Unexpected means of protocol inference. In *ACM IMC* (2006).

[15] McGregor, A., Hall, M., Lorier, P., and Brunskill, J. Flow Clustering Using Machine Learning Techniques. In *PAM* (2004).

[16] Moore, A., and Papagiannaki, K. Toward the accurate identification of network applications. In *PAM* (2005).

[17] Newman, M. E. J. The structure and function of complex networks. *SIAM Review 45* (2003), 167.

[18] Nguyen, T. T., and Armitage, G. A survey of techniques for Internet traffic classification using machine learning. *IEEE Communications Surveys and Tutorials 4th edition* (March 2008).

[19] Sen, S., and Wang, J. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Transaction on Networking 12*, 2 (2004), 219–232.

[20] Tan, G., Poletto, M., Guttag, J., and Kaashoek, F. Role classification of hosts within enterprise networks based on connection patterns. In *USENIX ATC* (2003).

[21] Trestian, I., Ranjan, S., Kuzmanovic, A., and Nucci, A. Unconstrained endpoint profiling (Googling the Internet). In *ACM SIGCOMM* (2008).

[22] Witten, I. H., and Frank, E. *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2005.

[23] W. John and S. Tafvelin. Heuristics to classify Internet Backbone Traffic based on Connection Patterns. In *ICOIN*, 2008.

[24] Xu, K., Zhang, Z., and Bhattacharyya, S. Profiling Internet backbone traffic: Behavior models and applications. In *ACM SIGCOMM* (2005).

[25] Y. Xie et al. Forensic analysis of epidemic attacks in federated networks. In *IEEE ICNP*, 2006.

[26] DatCat, Internet Measurement Data Catalog. `http://www.datcat.org`.