Network Traffic Analysis Using Principal Component Graphs

Harsha Sai Thota
Indian Institute of Technology
Guwahati, Assam, India
harsha.sai@iitg.ernet.in

V. Vijaya Saradhi
Indian Institute of Technology
Guwahati, Assam, India
saradhi@iitg.ernet.in

T. Venkatesh
Indian Institute of Technology
Guwahati, Assam, India
t.venkat@iitg.ernet.in

ABSTRACT

Graph-based techniques for monitoring network traffic and traffic classification have gained widespread attention due to their power of visualization and ability to detect anomalous behavior. In this work, we construct generic traffic activity graphs (TAGs) to represent packet-level traces and propose a method based on principal component analysis (PCA) of TAGs for traffic classification. The technique proposed in this work overcomes the issues with TAG construction related to ambiguity in edge definition, interval of trace collected for graph construction, and the time and location of trace collection. We construct a **set of TAGs** corresponding to traces from different time intervals and locations, and analyze the collection of TAGs by using PCA on the **set of TAGs** to identify the applications associated with traffic flows. Results on traffic classification show that the accuracy is very high and the technique is portable across different traces and therefore is useful to develop better graph-based traffic analysis tools.

General Terms

Principal component analysis; Graph Mining; Network Traffic Classification

Keywords

1. INTRODUCTION

Due to an increasing number of applications in the Internet and the complex interactions among them, monitoring network traffic and understanding communication patterns among the hosts in a network has attracted significant interest in the recent past [7]. Understanding network-wide communication patterns is critical for several reasons, from traffic classification to network planning and anomaly detection. In the recent times, representing the interaction between the hosts in a network as traffic activity graphs (TAGs), is observed to help not only in visualization of communication patterns of different applications but also in identification of applications from packet-level interactions [8]. Recent studies based on TAGs (sometimes also called traffic dispersion graphs (TDGs)), showed that representing host-level interactions in the form of graphs can help to identify behavioral characteristics of hosts in the network and identify anomalous communication patterns [6].

Traditionally, traffic classification (or identification) is done by associating the applications with a set of well-known port numbers. Rapid proliferation of a new set of applications and masquerading the port numbers for firewall traversal make port-based traffic classification unreliable [4]. Payload examination, sometimes called deep packet inspection that looks for unique strings called *signatures* inside the packet content is an alternate technique for classification. However, this technique is very slow to operate online, has privacy concerns, and does not work when the packet content is encrypted. These problems with the traditional techniques led to the development of machine learning techniques (both supervised and unsupervised) which identify the applications based on some discriminant criteria based on statistical distributions of various packet-level features [13]. However, these techniques require selection and extraction of features for each data set separately, and for most of the applications it is not easy to determine the discriminant features [4]. All the above techniques, ignore network associations that can provide valuable information about the host behavior and the application-specific communication. In recent times, approaches that take into account social interaction of hosts have shown to improve the accuracy of traffic classification and are shown to be robust to obfuscation of ports, modification of traffic features, and encryption of payload [6, 5].

TAGs are very useful to visualize and analyse the interaction of the hosts in a network. In an application-specific TAG, the nodes of the graph represent the hosts in the network and the edges represent the interaction between them with a specific application. For example, a TAG used to study HTTP traffic is a graph that has edges representing only HTTP interactions among the hosts in a network. It is shown that the application-specific TAGs have some characteristic features that help to distinguish between different applications [6]. Based on graph properties such as degree distribution and the size of giant connected component, TAGs are used to identify the type of application and detect anomalous interactions in the network. By constructing a TAG for each port or a range of ports and then analyzing the graph properties, the application using these ports can be identified given the graph properties of known applications.

Problems in TAG-based analysis: Though TAGs or

TDGs are shown to be useful in visualizing and monitoring the social behavior of hosts in the network, there are several issues with the construction and use of TAGs as a network monitoring tool. Some key issues in the construction of TAGs are:

1. **Edge Definition:** The first issue is that of defining the edge used to represent the interaction among the hosts. An edge in a TAG could be based on a filter such as the number of packets/bytes exchanged, port number used for interaction, the transport protocol used, or a combination of these [6]. Each edge definition gives rise to different graph representations for the network interactions and analyzing these graphs is a difficult task in network monitoring. Often, a TAG is constructed for each application and the edges do not carry any information about the number of packets/bytes exchanged (i.e., the edges are not weighted) so that the statistical feature information is lost in a TAG representation.

2. **Time Interval:** Another issue with TAG construction is that of the time interval used for trace collection and thus graph construction. Like many other complex network graphs, TAGs evolve over time and their degree distributions change with the time of observation [14]. As the time interval of observation increases, new interactions may be noticed and existing interactions may be terminated. This changes the number of edges with time and the graph properties change over time. However, identifying the time interval necessary to build a TAG that can throw some light on the application characteristics is not easy. Selecting a large time interval for graph construction leads to large graphs and many finer details are lost. At the same time, if the time interval is very small the graphs are sparse providing very little information about the interactions over time.

3. **Portability:** The hour of the day during which the trace is collected also affects the constructed TAG. TAGs constructed in peak traffic hours have different characteristics from those constructed in off-peak hours. The number of flows from different applications is also different during the peak and the off-peak hours [12]. Hence the analysis of TAGs might be influenced by the hour at which the trace is collected. Similarly, the graph metrics of TAGs change with the packet trace data used for construction. For example, two TAGs for DNS application constructed based on the trace collected at two different locations have a completely different average degree [6]. The degree distribution and other graph properties are also significantly different for TAGs constructed for different locations. This affects the portability of graph-based techniques for traffic analysis since, the information learned from a trace cannot be used for another trace.

In the present work, we address all the above shortcomings in TAG-based tools for traffic analysis by obtaining constructing **a set of generic TAGs**. We construct a single TAG for different applications and port numbers. The TAGs are constructed with both equal observation intervals and unequal observation intervals. The TAGs are constructed from traces collected at different hours of the day and at different locations. These mixed TAGs are all placed in the same set and a longitudinal approach for analysis is taken. Further, the TAGs used in this work also include the number of bytes transferred between the hosts as a weight on the edges.

The central idea in constructing a generic set of TAGs is that the application associated with a flow can be identified by observing the variance in the bytes transferred between the hosts across different time intervals and locations [12]. We apply principal component analysis (PCA) on the set of TAGs (graphs) to capture the variance in the number of bytes across different interactions to identify the application. Note that each TAG is a graph and PCAs work on the set of graphs instead of traditional vector data. Resulting principal components are called **principal component graphs** (PCGs) that capture variance across TAGs. We analyze the PCGs to identify the application associated with the edges of the graph. Key contributions of this work are:

1. Addressed **all** the issues in application-specific TAG construction by analyzing a mixed set of TAGs.

2. Addressed the problem of portability of traffic analysis tools by applying PCA on a set of mixed TAGs; the resulting PCGs are able to clearly identify the application flows in traces collected at different locations and times.

3. Proposed a method for traffic classification using PCA applied on TAGs.

4. Proposed a method to interpret the PCGs as a unified view of the set of graphs.

This paper is organized as follows: Section 2 discusses the construction of mixed set of TAGs. PCA on graphs is discussed in Section 3 along with the proposed method of edge classification and graph construction of PCGs. Demonstration of the proposed technique on public network traces is presented Section 4. Review of the literature on application of PCA for traffic classification and application of PCA to different data types is given in Section 5. Section 6 summarizes the work.

2. CONSTRUCTION OF SET OF GENERIC TAGS

We consider two public traces from (i) trans-Pacific 150 Mbps line (WIDE) collected on different dates and (ii) CAIDA 1 Gbps commercial backbone links (equinix-chicago and equinix-sanjose). Table 1 gives the total number of unique IP addresses, the number of flows, and the number of packets in each trace. For constructing the TAGs only the flows from HTTP, HTTPS, DNS and AIM-Video (a peer to peer application) are considered. Though the experiments were conducted on other applications such as Yahoo! Messenger and MSN Messenger, results for them are not presented since they are similar. Table 2 shows the total number of unique IP addresses (nodes in a TAG) and the number of flows for each application (edges in a TAG).

TAG construction: For each trace, a unique IP address is represented as a node in the TAG. Two nodes u and v are connected by an edge when there is a flow that uses one of the ports corresponding to the application of interest, namely HTTP (port 80), HTTPS (port 443), DNS (port

Application		Port Number	# Unique IPs	# Flows
WIDE 07-May-2009 @14:00	HTTP	80	28K	56K
	HTTPS	443	3K	4K
	DNS	53	89K	248K
	AIM Video	1024 - 5000	270K	338K
WIDE 13-April-2010 @04:00	HTTP	80	44K	79K
	HTTPS	443	4K	4K
	DNS	53	60K	234K
	AIM Video	1024 - 5000	215K	258K
WIDE 13-April-2010 @10:45	HTTP	80	74K	125K
	HTTPS	443	4K	7K
	DNS	53	48K	194K
	AIM Video	1024 - 5000	251K	334K
CAIDA 16-April-2009	HTTP	80	217K	300K
	HTTPS	443	65K	78K
	DNS	53	33K	73K
	AIM Video	1024 - 5000	418K	470K
CAIDA 21-Jan-2010	HTTP	80	91K	122K
	HTTPS	443	7K	5K
	DNS	53	37K	110K
	AIM Video	1024 - 5000	264K	267K

Table 2: Application Composition in Each Trace

Trace	Date	# IPs	# Flows
WIDE	07-May-2009	966K	4M
WIDE	13-April-2010	626K	2M
WIDE	13-April-2010	576K	2M
CAIDA	21-January-2010	923K	2M
CAIDA	16-April-2009	1M	2M

Table 1: Traces Used for Experimentation

53) or AIM Video (port range 1024 to 5000). Note that, the edge filter is only to reduce the number of edges in the TAG and we do not use the port number in traffic identification anywhere in the proposed technique. To verify the accuracy of the classification we use port number since the traces are not tagged with any other ground truth. In addition to this simple edge definition, we also use the total number of bytes transferred between nodes u and v during the time interval as a weight on the edge (u, v).

Set of TAGs: As mentioned in the Section 1, instead of using single TAG for analysis, a mixed set of TAGs is constructed for analyzing the variance across the TAGs used to represent different application interactions. The set of TAGs are constructed in different settings as described below.

1. **Single Trace:** Each trace in the table 1 is partitioned into groups using either equal time intervals or unequal time intervals.

   Each group comprises of as many TAGs as the number of applications considered. For example, a 900 seconds WIDE trace with 2 applications say, HTTP and DNS is divided into 4 groups using same time interval of 225 seconds. In each group, 2 TAGs are constructed one for HTTP and one for DNS. A total of 8 TAGs are thus constructed when 4 time intervals and 2 applications are considered. This set of 8 TAGs includes a mixture of TAGs characterizing two different applications and traces from 4 times.

   Using the same trace one can obtain 4 groups each constructed for unequal intervals say 150 seconds, 200 seconds, 250 seconds and 300 seconds and with 2 applications in each. In this case, first group consists of 2 TAGs, HTTP and DNS, constructed with an observation time interval of 150 seconds. Similarly second group consists of 2 TAGs constructed with a time interval of 200 seconds and so on. This set of 8 TAGs includes a mixture of TAGs with varying time intervals and two different applications.

2. **Multiple Traces:** Procedure described above is employed for constructing TAGs across two traces collected from two different locations. For example, when a 900 seconds WIDE and 60 seconds CAIDA traces are considered for TAG construction, both traces with flows from 2 applications are partitioned using 4 equal time intervals, then 8 TAGs are obtained such that each group consists of TAGs from both WIDE and CAIDA. This case takes considers different edge definitions, observation periods, and trace collection points while constructing TAGs. Similarly the two traces are divided into 4 groups with unequal time intervals each consisting of 2 application TAGs resulting in a total of 8 TAGs.

3. PCA ON SET OF GRAPHS

PCA: Given a data set having a certain coordinate system, principal component analysis (PCA) obtains a new coordinate system. Axes of the new coordinate system are orthonormal to each other and are referred to as principal components (PC). Each of the PC captures maximum variance when the original data is projected onto these PCs. First PC captures maximum variance in the data when the data is projected onto the first PC. Second PC captures the second largest variance when the data is projected onto second PC and so on. Note that any point in the data set can be expressed as a weighted linear combination of the obtained PCs. This process of expressing the data point through the PCs is known as reconstruction. A data point is reconstructed by considering a few PCs instead of considering all of the PCs leading to reduced dimensionality of the new coordinate system.

PCA on TAGs: Each TAG in a set of TAGs is represented in the form of a matrix, say G. Element G_{ij} gives the total number of bytes transferred from node i to node j in the interval of observation. The vectorized form of G is considered as a vector in one dimension. Thus there would as many dimensions as the cardinality of the TAG set. Total number of data points in this representation is equal to the number of elements in the TAG.

PCA is applied on *vectorized form of the TAG* set to obtain PCGs. Covariance matrix on the set of TAGs is obtained. PCA is applied on the resulting covariance matrix. Note that [3] has shown that the covariance matrix is a positive semidefinite matrix which is necessary to solve the eigenvalue problem.

After applying PCA, we obtain as many principal components as the number of TAGs in the set. Each principal axis is termed as a PCG which gives an unified view of the set of TAGs constructed across time and location. The PCGs capture the variance of the bytes transferred over the edges of the graphs constructed across different time intervals. The key idea in including the number of bytes pertaining to each application flow on the TAG edges is that variance of the number of bytes across different time intervals or across different traces for any given application will be similar and the PCGs capture this variance.

Classifying edges of each PCG: Let the set of TAGs be: $\{G^1, G^2, G^3, \cdots, G^d\}$; where d = number of groups $\times$ number of applications. Let the constructed PCGs be $\{PCG^1, PCG^2, PCG^3, \cdots, PCG^d\}$. Let the size of each graph be $n \times n$; where n is the number of unique IP addresses and let an edge (i, j) in G^q be denoted by G^q_{ij} which represents the number of bytes transferred in a time interval. Note that each G^q, by edge definition, stand for an application. Let $c(.)$ be a function which takes as argument a TAG and return the application that TAG signifies. For each $q \in \{1, 2, \cdots, d\}$ and when $G^q_{ij} > 0$, the contribution of the closest PCG is obtained as:

$$indx = arg \min_{\ell}(G^q_{ij} - |PCG^{\ell}_{ij}|).$$

Then the edge (i, j) in the PCG^{indx}_{ij} is assigned to application $c(G^q)$ as

$$PCG_{ij}^{arg \min_{\ell}(G^q_{ij} - |PCG^{\ell}_{ij}|)} = c(G^q) \qquad (1)$$

We explain the above edge classification through an example. Let a given trace be divided into 4 groups. In each group, we construct 2 TAGs corresponding to applications HTTP and AIM-Video. We then have a total of 8 TAGs. Let G^1 though G^4 represent HTTP TAG and let G^5 through G^8 represent AIM-Video. Table 3 shows 8 TAGs for edge (1, 2) and corresponding PCGs. G^1_{12} carries 1934 bytes and the PCG that has closest contribution corresponding to these bytes is the PCG^3 with a contribution of -1950.03. Therefore PCG^3_{12} is classified as $c(G^1)$ which is HTTP. This procedure is continued for all $q = 1, 2, \cdots, d$.

Principal Components as Graphs: The obtained PCs are orthogonal and are of unit norm. A threshold, set as $\frac{1}{\lambda}$ where λ is the variance captured on that PCG, is applied to each element of the PCG to round every element to 0 or 1. Once the PC is rounded, the vectorized PC is converted back to adjacency matrix form for a graph. This graph is subjected to further inference.

4. EXPERIMENTAL RESULTS

1. **Single Trace:** We have considered third trace in table 2 and the pairs of applications considered are: (HTTP, DNS), (HTTP, HTTPS), (HTTPS, DNS) and (HTTP, AIM-Video). The trace is divided into 4 equal groups and a total of 8 TAGs are formed for each pair of applications.

   Variance Across TAGs: Figures 1 and 2 show the variance within a TAG and across every edge of TAGs. That is variance on edge (i, j) across G^1 through G^d. For every pair of applications, it is observed that HTTP graph has the highest variance compared to any other application while DNS graph has the lowest variance. This observation is independent of time duration and edge definition. For the case when applications (HTTP, DNS) are involved, we note that first 4 PCGs capture HTTP traffic and last 4 PCGs capture DNS traffic. A similar observation is made for applications (HTTP, HTTPS), (HTTPS, DNS) and (HTTP, AIM-Video).

   Importance of smallest variance PCGs: Though the contribution of low-variance PCGs towards reconstructing original graphs is small, one cannot discard these PCGs unlike in the case of traditional PCs. Reason for this is that every PCG captures a unified view of the TAGs and applications having small variance in the number of bytes are potentially captured in the lower order PCGs. In case of (HTTP, DNS) application traffic, if we decide to discard last 4 PCGs, then we lose information about DNS completely.

   Classification of PCGs: Each edge in the PCG is classified as per the procedure explained in section 3. In the case of (HTTP, AIM-Video) applications, we observe from figures 3 and 4 that $1^{st}, 4^{th}, 5^{th}$ and 6^{th} PCGs capture HTTP traffic and $2^{nd}, 3^{rd}, 7^{th}$ and 8^{th} PCGs capture the AIM-Video traffic. Note that AIM-Video application variance is very close to that of HTTP traffic and the PCGs are able to classify these two applications 1 and 2. Table 4 shows total number of flows in each graph from HTTP and AIM-Video applications and how many of these flows are captured in the PCGs. This table suggests that G^1 contains 2115 flows related to HTTP and the majority of the flows are captured in PCG^8 which has small variance. This suggests that in the time interval when G^1 graph constructed, there is not much variance in the HTTP flows. Remaining 109 flows are misclassified into other PCGs. The last column denotes the percentage of edges having HTTP traffic in PCG^8.

2. **Multiple Traces:** We have considered four traces namely first to fourth from table 2 and the pairs of applications considered are (HTTP, DNS) and (HTTP, HTTPS). The trace is partitioned into 4 equal groups

TAG Set							
HTTP TAG				AIM-Video TAG			
G^1	G^2	G^3	G^4	G^5	G^6	G^7	G^8
1934	1444	0	768	0	96	0	0

PCG Set							
PCG^1	PCG^2	PCG^3	PCG^4	PCG^5	PCG^6	PCG^7	PCG^8
0.35	-392.70	-1950.03	1374.92	-36.01	-753.37	89.98	-3.32

Table 3: Example bytes transferred from node 1 to node 2 in TAG set and its corresponding edge (1,2) in PCG set; As contribution of edge PCG^3_{12} close to G^1_{12}, according to (1) PCG^3_{12} belongs HTTP. Similarly PCG^4_{12} belongs to HTTP and PCG^7_{12} belongs to AIM-VIDEO applications.

	PCG^1	PCG^2	PCG^3	PCG^4	PCG^5	PCG^6	PCG^7	PCG^8	Accuracy
$G^1 : 2115$	9	16	15	13	12	23	21	2006	97.49
$G^2 : 1644$	10	15	23	28	20	23	1496	29	95.38
$G^3 : 1805$	28	50	17	1637	11	29	15	18	95.96
$G^4 : 1986$	1852	18	15	20	17	32	15	17	95.92
$G^5 : 2588$	26	27	2359	30	24	46	32	44	94.36
$G^6 : 2688$	32	45	23	26	2473	38	20	31	95.31
$G^7 : 3151$	25	32	27	30	38	2956	17	26	96.38
$G^8 : 2514$	40	2240	28	51	39	55	21	40	93.83

Table 4: Confusion Matrix: First column: Total number of flows from each TAG. Rest of the columns the number of flows captured by respective PCG.

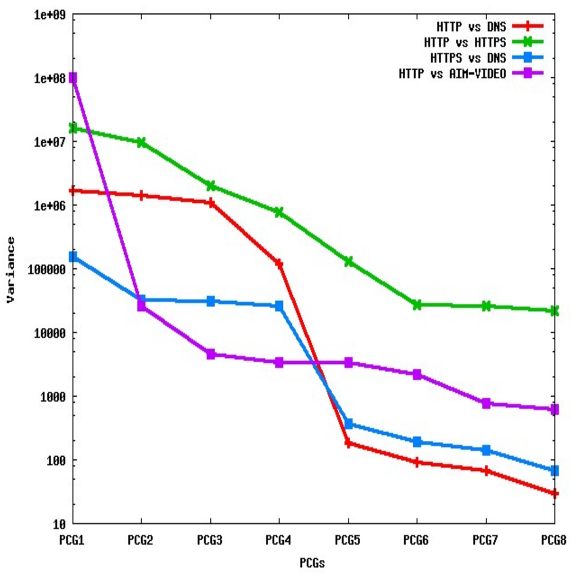

Figure 1: Byte variance across TAGs: Single Trace, Same Time Interval

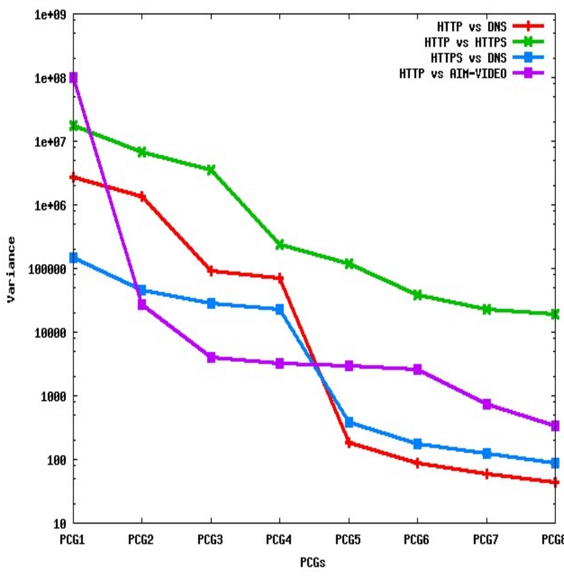

Figure 2: Byte variance across TAGs: Single Trace, Varying Time Intervals

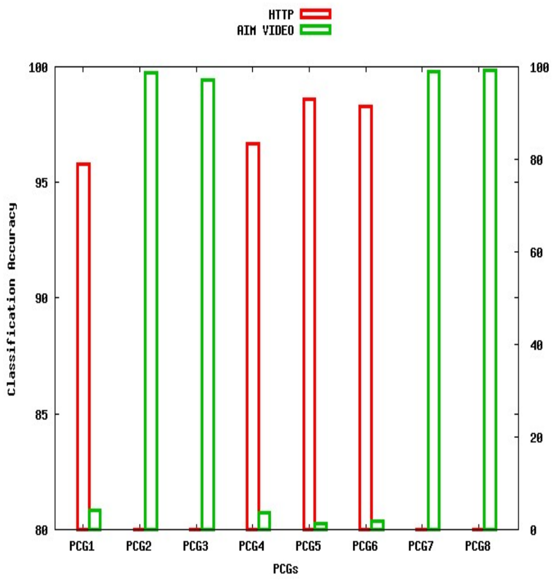

Figure 3: PCG Traffic Analysis: Single Trace, Same Time Interval

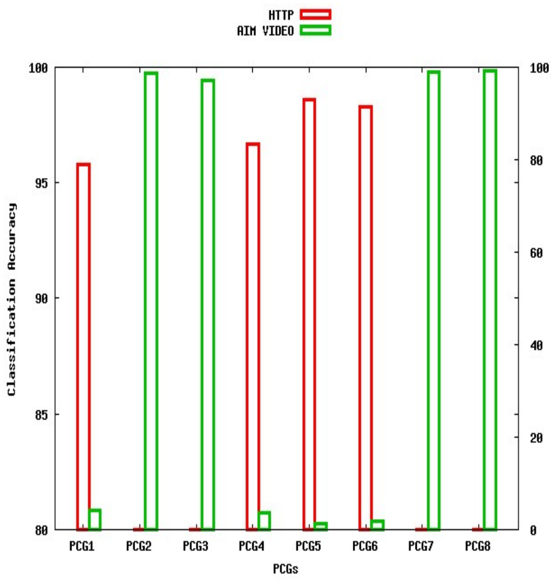

Figure 4: PCG Traffic Analysis: Single Trace, Varying Time Intervals

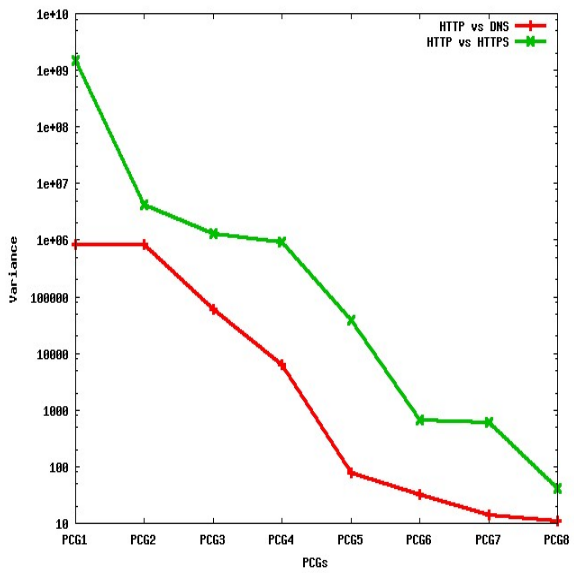

Figure 5: Byte variance across TAGs: Multiple Traces, Same Time Interval

(with equal time intervals) and a total of 8 TAGs are formed for each pair of applications. Variance across TAGs for HTTP is significantly high compared to DNS or HTTPS.

Variance Across TAGs: Variances of TAGs are shown in figure 5. For the case when applications (HTTP, HTTPS) are involved, the HTTP traffic variance is significantly high and is captured in the first PCG. Last PCG (i.e 8^{th} PCG) captures least variance and corresponds to HTTPS traffic. We note that first 4 PCGs capture HTTP traffic and last 4 PCGs capture HTTPS traffic. A similar observation is made for PCGs involving applications (HTTP, DNS).

Classification of PCGs: A clear separability of HTTP and DNS, and HTTP and HTTPS, is observed from the PCGs obtained. Since the volume of bytes transferred is significantly different in the two traces, the PCGs are able to clearly separate out applications corresponding to each edge accurately. The results are depicted in figure 6. To the best of our knowledge this is the first time such a mixture of flows from different trace collection points were considered together for analysis and inference.

3. **PCGs a Close Look as Graphs:** As explained in section 3, every PCG is interpreted back as a graph. In the case of single trace and considering two applications (HTTP and DNS) the 4^{th} TAG and the PCG which captured this graph are plotted using open access visualization tools and are given in figure 8. We note that majority of edges belonging to 4^{th} TAG are captured in the PCG. Apart from the edges in 4^{th} TAG, this PCG captures edges which are from other TAGs as well (the peripheral nodes and edges in PCG).

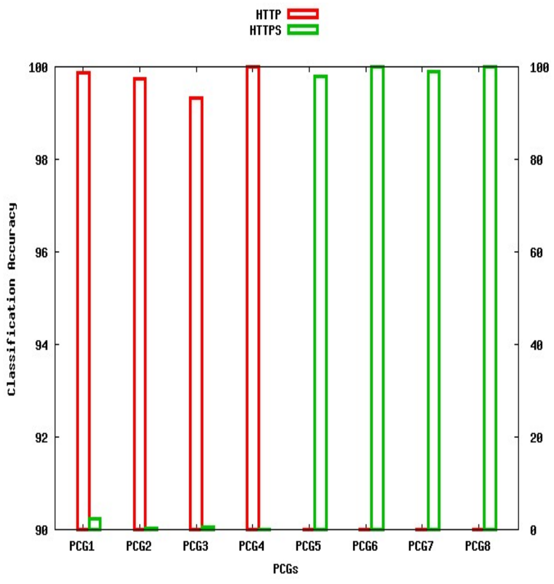

Figure 6: PCG Traffic Analysis: Multiple Traces, Same Time Intervals

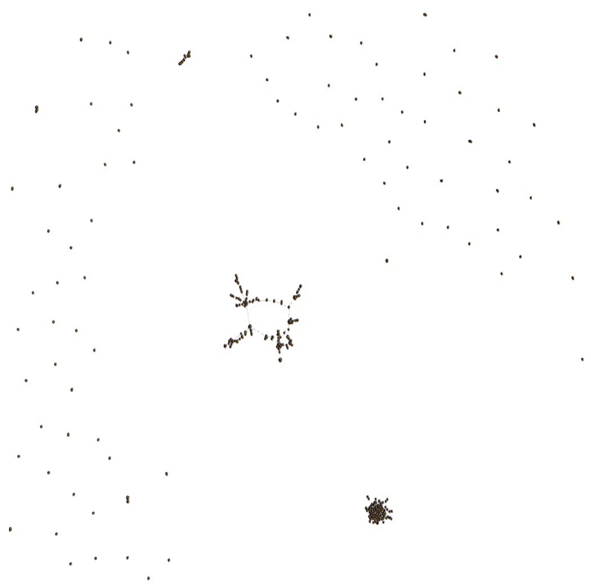

Figure 7: HTTP-TAG

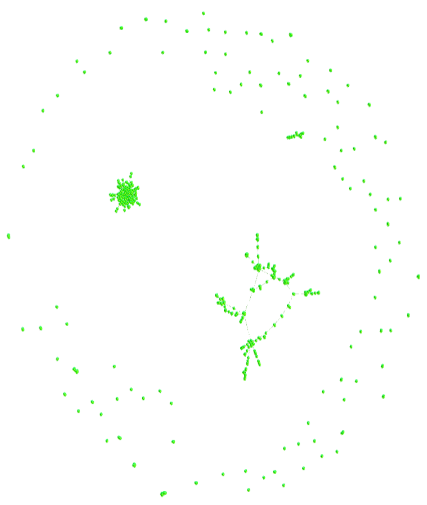

Figure 8: HTTP-PCG

5. RELATED WORK

In general, PCA is applied on vector data. PCA is also applied for special types of data such as time series data [9], discrete sets [9], trees [1] and graphs [3]. In particular, [1] considered blood vessels whose structure correspond to trees is considered and PCs for tree structures is developed. An interesting relation between human age and blood vessel structure is identified. In [3], PCA on set of graphs is applied where a graph in the set is a concept graph. Nodes in the concept graph are concept words. Relation between two concepts stand for an edge in the concept graph. [3] identified the structure of the first PC's and relation between the concepts. In the present work, we apply PCA on set of graphs constructed from the network traffic data and analyze the resulting PCs which we term as principal component graphs (PCGs).

PCA found diverse applications in the network traffic analysis literature. In [11], two distinct backbone network's complete origin destination (OD) flow time series data is analyzed through PCA. The key contribution is in analyzing the complete OD flow. Though the number of OD flows are large, authors identify that majority of the network traffic variability is captured by a few eigenflows (PCs). Through the eigenflows, the network traffic structure is identified through periodic, short lived and noisy flows. In [10, 2], PCA is applied to identify anomalies in the network traffic data. Daniela *et al.* discuss the problems that arise when PCA is applied for anomaly detection, reasons for the identified problems and incorporate time information while applying PCA.

There is a large literature on network traffic analysis from traffic classification to anomaly detection. An interested reader may look at the surveys for various techniques that exist for traffic classification based on machine learning and otherwise [13, 4]. As mentioned earlier, traditional traffic classification techniques are based on either port-based detection, payload inspection, statistical analysis of packet-level features, and host behavior in a network. In the recent times, using graphs to represent the network-level interactions of hosts is gaining popularity for network monitoring, traffic analysis, and traffic classification [6, 5]. TAGs help not only in visualization of network interactions but also in effectively identifying the signatures of different applications in terms of the host-level interactions. It is observed that TAGs can serve as a powerful tool for traffic analysis and the techniques based on these are robust against port masquerading or obfuscation and polymorphic blending of features [5].

6. SUMMARY

In this work, we developed a technique that uses PCA on a set of generic TAGs for network traffic analysis and classification. The generic TAG set includes application flows from different locations and time intervals of observation which overcomes the issues related to portability in the previous approaches based on TAGs. A unified view of the TAG set is obtained through PCGs. PCGs are subject to further inference yielding encouraging results from the perspective of network flow classification. Experimental results demonstrated that the proposed technique is highly accurate in traffic flow classification and is also portable across traces and time of observation. The applicability of PCGs for traffic analysis thoroughly analyzed in this work and we hope that this would lead to development of graph-based approaches for network analysis.

7. REFERENCES

[1] B. Aydin, G. Pataki, H. Wang, E. Bullitt, and J. S. Marron. A principal component analysis for trees. *The Annals of Applied Statistics*, 3(4):1597–1615, 2009.

[2] D. Brauckhoff, K. Salamatian, and M. May. Applying PCA for traffic anomaly detection: Problems and solutions. In *IEEE INFOCOM*, 2009.

[3] C. Butts and K. Carley. Multivariate methods for inter-structural analysis. Technical report, Department of Social and Decision Sciences, Center for Computational Analysis of Social and Organizational Systems, Carnegie Mellon University, 2001.

[4] A. Dainotti, A. Pescape, and K. Claffy. Issues and future directions in traffic classification. *IEEE Network Magazine*, pages 35–40, January 2012.

[5] M. Iliofotou, B. Gallagher, T. Eliassi-Rad, G. Xie, and M. Faloutsos. Profiling-by-association: A resilient traffic profiling solution for the Internet backbone. In *Proceedings of the ACM CoNEXT*, 2010.

[6] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Sing, and G. Varghese. Network monitoring using traffic dispersion graphs (TDGs). In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007.

[7] Y. Jin, P. Haffner, S. Sen, and Z.-L. Zhang. Can't see forest through the trees? understanding mixed network traffic graphs from application class distribution. In *Proceedings of the Workshop on MLG*, 2011.

[8] Y. Jin, E. Sharafuddin, and Z.-L. Zhang. Unveiling core network-wide communication patterns through application traffic activity graph decomposition. In *Proceedings of the ACM SIGMETRICS*, 2009.

[9] I. T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 2010.

[10] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, pages 219–230, New York, NY, USA, 2004. ACM.

[11] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczykp, and N. Taft. Structural analysis of network traffic flows. In *SIGMETRICS*, 2004.

[12] Y.-D. Lin, C.-N. Lu, Y.-C. Lai, W.-H. Peng, and P.-C. Lin. Application classification using packet size distribution and port association. *Journal of Network and Computer Applications*, 32(5):1023–1030, 2009.

[13] T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Commun. Surveys Tuts.*, 10(4):56–76, Oct. 2008.

[14] X. Wu, K. Yu, and X. Wang. On the growth of Internet application flows: A complex network perspective. In *Proceedings of the IEEE INFOCOM*, 2011.