Network Monitoring using Traffic Dispersion Graphs (TDGs)

Marios Iliofotou
UC Riverside
marios@cs.ucr.edu

Prashanth Pappu
Rinera Networks
prashanth@rinera.com

Michalis Faloutsos
UC Riverside
michalis@cs.ucr.edu

Michael Mitzenmacher
Harvard University
michaelm@harvard.edu

Sumeet Singh
Cisco Systems, Inc.
sumeets@cisco.com

George Varghese
UC San Diego
varghese@cs.ucsd.edu

ABSTRACT

Monitoring network traffic and detecting unwanted applications has become a challenging problem, since many applications obfuscate their traffic using unregistered port numbers or payload encryption. Apart from some notable exceptions, most traffic monitoring tools use two types of approaches: (a) keeping traffic statistics such as packet sizes and inter-arrivals, flow counts, byte volumes, etc., or (b) analyzing packet content. In this paper, we propose the use of Traffic Dispersion Graphs (TDGs) as a way to monitor, analyze, and visualize network traffic. TDGs model the *social behavior* of hosts ("who talks to whom"), where the edges can be defined to represent different interactions (e.g. the exchange of a certain number or type of packets). With the introduction of TDGs, we are able to harness a wealth of tools and graph modeling techniques from a diverse set of disciplines.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network monitoring; C.2.5 [**Local and Wide-Area Networks**]: Internet

General Terms

Measurement, Experimentation, Security

Keywords

Behavioral Approach, Hosts' Connection Graphs, Network Monitoring, Network Traffic Visualization

1. INTRODUCTION

The fundamental problem that motivates this paper is the need for better methods and tools to monitor networks. The network could be a large ISP such as Sprint, or an enterprise network such as the Walmart network. In such environments we want to be able to detect abnormal phenomena such as: (a) new applications that "abuse" legacy ports, (b) malicious activity such as worm spreads and port scanning. These forces suggest the need for a more fundamental *behavioral* approach to characterize traffic in the face of obfuscation.

Current monitoring and application classification methods can be grouped by their level of observation: (a) packet level, such as signature-based application detection [5] and methods using the well known port numbers, (b) flow level statistical techniques [7], [9], [12], and (c) host level, such as host-profiling approaches [6]. In both the case of malcode and P2P, the standard approaches using *content signatures* seem destined to fail in the face of encryption (e.g., for P2P traffic) and polymorphism (e.g., for worms).

In this work, we propose a different way of looking at network traffic that focuses on network-wide interactions of hosts (as seen at a router). We use the term **Traffic Dispersion Graph** or **TDG** to refer to this graph, which intuitively shows which hosts interact. We argue that there is a wealth of information embedded in a TDG. For example, a popular website will have a large in-degree, while P2P hosts will be tightly connected. Despite what may appear at first, the definition of TDGs is non-trivial, as it hinges on how we define an edge. An edge can represent the exchange of at least one packet, but it can also be the exchange of one or more TCP SYN packet, or more than, say, five packets of any type. In other words, a TDG can represent a particular type of interaction, which gives them significant descriptive power, as we discuss later in detail.

We note here that TDGs can be seen as the natural next step in the progression of packet, flow, and host level **aggregation**. This is because a flow aggregates a set of packets, a host aggregates a set of flows originating and terminating at the host (e.g., BLINC operates at this level [6]), and a graph aggregates a group of hosts. In other words, we can analyze the "social interaction" of network hosts as a whole, which leads to a graph where each node is an IP address, and each directed edge represents an interaction between two nodes. Studying the aggregation properties of TDGs and how they can help in providing compact traffic representation is included in our future directions.

Our main goal is to propose TDGs as a different way of modeling traffic behavior, and show that they: (a) have characteristic structure and provide visualizations that can distinguish the nature of some applications, (b) describe traffic along a new "dimension", the network-wide social behavior, which complements traffic characterization at the packet, flow and host levels, and (c) represent a new "type" of graphs, by showing that even though they have similar characteristics they are not yet another power-law graph.

2. RELATED WORK

The profiling of "social" behavior of hosts was studied in BLINC [6] for the purpose of traffic classification. BLINC, however, focuses only at the host level. In BLINC, the notion of a "graphlet" models a *single* host's flow patterns. By contrast, our work on TDGs uses *network-wide* graphs. In some sense, a TDG is an aggregation of the graphlets of all hosts in a network for a particular key. To simplify the aggregation, TDGs use edges only between source and destination IP addresses unlike "graphlets" [6] which also have edges to nodes that represent port numbers.

Apart from some notable exceptions, TDGs have not been used for network analysis and monitoring. The first work using TDG-like graphs for intrusion detection appeared in 1999 [2], but there has not been recent follow up. Recently, work by Ellis uses graph-based techniques to detect worm outbreaks within enterprise network environments [4]. These efforts only focus on worm detection, by trying to capture the tree-like structure of self-propagating code.

In addition, companies like Mazu and Arbor Networks utilize graph-based techniques. Tan et al. [11] in collaboration with Mazu show how similarity between hosts, based on their connection patterns, can be used to group network users into related roles. In contrast to grouping hosts and calculating user similarity, the present work mainly focuses on characterizing the structure found in connection graphs of hosts based on a key (e.g., a common TCP or UDP port number).

Graph based techniques are also used by Aiello et al. [1] in the field of Communities of Interest (CoI). Similar to [11], CoI research focuses on extracting communities and modeling "normal behavior" of hosts. Deviation from normal behavior can then be used to trigger an alarm. Moreover, the authors in [1] and [11] do not use the graph metrics we employ in this work and mainly focus on enterprise network environments.

Finally, TDGs as defined here are significantly different than trust propagation networks [13]. The problem there is to identify intruders in social networks, which represent trust relationships (and not an exchange of packets), as discussed in SybilGuard [13].

3. TRAFFIC DISPERSION GRAPHS

Definition: A traffic dispersion graph is a graphical representation of the various interactions ("who talks to whom") of a group of nodes. In IP networks, a node of the TDG corresponds to an entity with a distinct IP address and the graph captures the exchange of packets between various sender and destination nodes. Moreover, a TDG, by definition, is a graph that evolves both in time and space as various nodes interact with each other. Hence, the edges in a TDG have an implicit temporal relation showing the order in which the corresponding node-interactions were observed in the network. This also means that a given *static* TDG has an associated time interval over which it evolved.

Edge Filtering: One of the fundamental questions in using TDGs is the definition of an edge. This basic question can be answered in many different ways depending on the goal of the study. We start with the observation that the edges in a TDG are directed because there is a clear definition of sender and receiver in every data packet. In general, the directed edges in a TDG can be used to identify the initiator of the interaction between a pair of nodes. As we will later see, directed edges in a TDG are very useful in identifying various node behaviors and also in establishing their causal relationship. However, we could choose to consider undirected edges, which will enable us to use the more extensively studied graph metrics for undirected graphs, as discussed in later sections.

Besides direction, it is important to define what kind or level of interaction between a pair of nodes should be translated into an edge in the TDG. We call this process *Edge Filtering.* One simple edge filter is to add an edge (u, v) between nodes u and v when the first packet is sent from u to v in the interval of observation. Once an edge is added, the filter ignores any further packets sent from u to v. We call this edge filter as the Edge on First Packet (EFP), and is mainly used for translating UDP flows between nodes into TDG edges.

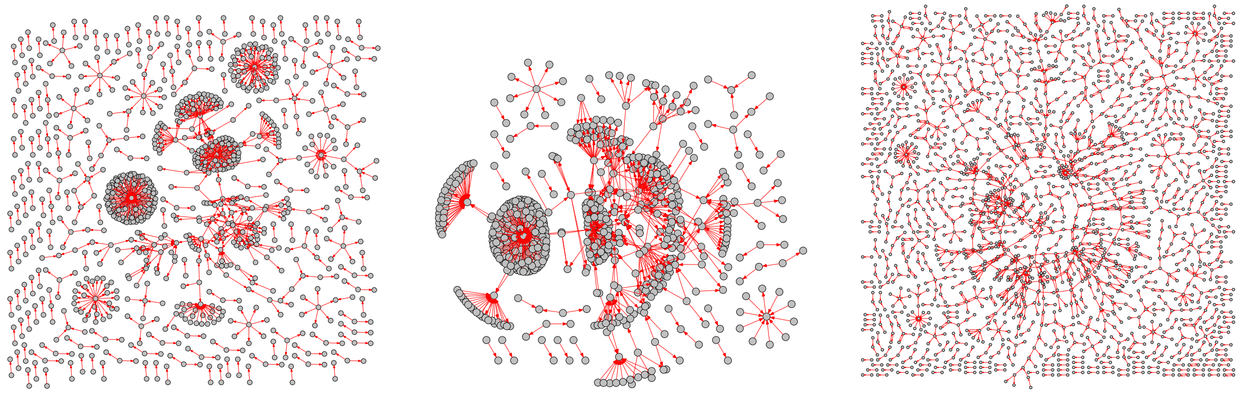
Unlike UDP flows, TCP flows have an explicit definition of initiation of interaction between two nodes. For TCP flows, we can choose to add a directed edge (u, v) between two nodes u and v when the first SYN packet is sent from u to v. We call this filter as the Edge on First SYN Packet (EFSP) filter. While the EFSP filter is applicable only to TCP flows, it can accurately determine the initiator of the interactions between a pair of nodes.

In addition to this basic edge filtering, we can enrich the definition of what constitutes an edge by imposing "stricter" rules that capture different aspects of the interaction. For example, we can have filters for "allowing" an edge between a pair of nodes based on: **(a)** the number of packets/bytes exchanged, **(b)** the type and sequence of packets (e.g., TCP three-way handshake), **(c)** the transport protocol used (TCP, UDP, ICMP etc.), **(d)** the application based on port number or port number range (e.g., Port Number 80, or Port Range 6881-6889), and finally **(e)** looking at properties of the content, such as payload size or by using deep packet inspection.

TDG Formation: In this paper we focus on port-based TDGs using the **(d)** filter type (as defined above). Throughout the paper and unless stated otherwise, when the legacy application for a port uses TCP, we use the EFSP edge filter on the corresponding destination port (e.g., TCP Port 25 for SMTP). When we examine UDP interactions, we use the EFP edge filter on the destination port of interest (e.g., UDP Port 53 for DNS). For ease of presentation, we will refer to each port-based TDG using the name of the dominant or well-known application under that port. For example, the HTTP TDGs is formed by using as edges all the TCP SYN packets that have as destination port the number 80.

Since we use edge filtering by port number, the TDGs capture aspects of any application that uses these ports. We are fully aware that many non-standard applications, such as P2P traffic, use standard ports such as Port 80 [6]. However, port-based filtering is consistent with our use of TDGs as a monitoring tool. For example, if at some point traffic at TCP Port 80 appears significantly different, it could be: (a) a new benign or malicious application tunneling its traffic under that port, or (b) a change in the behavior of the traditional application.

Network Traffic Traces: To study and analyze TDGs, we use a variety of publicly available, real-traffic traces listed in Table 1. These traces are non-sampled, IP anoymized, and include up to layer-4 headers with no payload. We verified our findings with many other traces provided by the same online sources (see *Online Sources* in Table 1), but the results are not shown here due to space limitations. All traces



(a) HTTP (30 sec) (b) DNS (5 sec) (c) WinMX P2P (15 sec)

Figure 1: TDG visualizations, using the GraphViz tool, for various communities of hosts from backbone packet traces.

Name	*Date/Time*	*Online Source*	*Duration*	*Unique IPs*	*Unique Dst. Port Numbers*	*5-tuple Flows*	*Mbps(Kpkt/s)*
WIDE	2006-03-03/13:00	tracer.csl.sony.co.jp	2 h	1,041,622	TCP=62,463 UDP=64,727	4,670,259	31.0(9.7)
ABIL	2004-06-01/17:30	www.nlanr.net	1 h	6,520,862	TCP=65,536 UDP=63,560	23,623,601	1,726.2(226.6)
OC48	2003-01-15/10:00	www.caida.org	1 h	3,463,366	TCP=65,536 UDP=62,369	22,109,673	590.1(128.1)

Table 1: Our set of publicly available backbone traces from **WIDE**, CAIDA (**OC48**) and the Abilene Network (**ABIL**).

used in this paper (Table 1) are captured on a single bidirectional link and hence reveal only the node interactions that cross the monitoring point. If TDGs are implemented on a firewall or router, they will most likely see exactly that. Therefore, the analysis of TDGs derived from one point of observation is probably appropriate when considering practical deployment. Clearly, the TDGs formed by traces taken at a single point are inherently *bipartite*. We realize, of course, that observing *all* the network interactions, say within an enterprise network, would provide an even more complete view of the network.

TDG Visualization: Traditionally, visualization of traffic in monitoring tools has largely been limited to visualizing measures of traffic volumes on a per flow basis. By contrast, we show that TDGs lend themselves to simple graphical visualizations of interaction patterns. For example, the graphs in Fig. 1 show a simple set of TDGs, where we filter the edges for distinct Port Numbers. The observation intervals for these graphs were chosen so as to capture good visual details for each TDG. Studying these TDGs (Fig. 1), we can quickly reach the following conclusions, which we have corroborated with a number of similar visualizations:

(i) TDGs are not a single family of graphs. We can see that TDGs have significant visual and structural differences, which we quantify with graph metrics in Table 2. This characteristic is what gives TDGs descriptive power, as we discuss later in more detail.

(ii) TDGs capture many interesting patterns of node interactions. We can identify several distinctive structures and patterns in TDGs, which are indicative of the behavior of different applications.

Node degrees - The degrees of various nodes and their connectivity in a TDG helps us in visually determining the type of relationship between the nodes. In general, the TDGs corresponding to protocols with a prevalence of client-server interactions, such as DNS (Fig. 1(b)) and HTTP (Fig. 1(a)), are dominated by a few high degree nodes whereas the TDG of the popular peer-to-peer application WinMX has many similar degree nodes (Fig. 1(c)).

Node roles - In many TDGs, the role of a node can be inferred from the direction of its edges (not easily distinguishable at this visualization scale). For example, Fig. 1(a) presents an HTTP TDG. This makes it easy to spot Web servers for example, since they are "pointed to" by edges (non zero in-degree). Also, most hosts have either zero in-degree or zero out-degree, indicating that they act either solely as a server or solely as a client. Interestingly, however, there are a few nodes with both non-zero in-degree and out-degree, which can correspond to HTTP proxies or Web caching systems.

Node chains - Long undirected chains of nodes are very common in TDGs of peer-to-peer applications like WinMX shown in Fig. 1(c) and are mostly non-existent in the TDGs of other applications. This shows that a node u can be linked to node v via a significant number of hops (intermediate IP hosts).

Node communities and their sizes - There are many disjoint components in each of the TDGs. This is in contrast to many other types of graphs such as the Internet topology or the web-page graph [10]. Also, as we will see later on, the number of connected components and the size of the largest component varies a lot across different TDGs. For example, note the differences in the distribution of the sizes of various components in the HTTP TDG which has many small subgraphs.

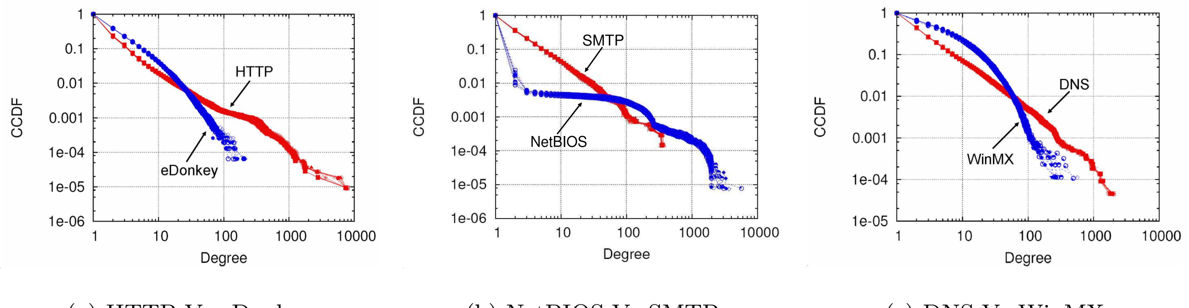
Discussion: Although our goal here is to visually examine the various properties of TDGs, good visualization methods have their own value. In fact, effective visualization and human monitoring can often be a more viable alternative to complicated automated methods for anomaly detection. While visualization is useful by itself, if TDGs are to be used for application monitoring, it is important to translate visual intuition into quantitative measures.

4. QUANTIFYING TDG PROPERTIES

In this section, we present a series of fundamental graph metrics computed over real-traffic TDGs. A set of experimental results is summarized in Table 2. Graphs are calculated over 12 consecutive, disjoint, 300-second-long observa-

TDG	*Nodes*	*Edges*	*Avg. Degree*	*InO(%)*	*OnlyIn(%)*	*MDR*	*GCC(%)*	*MAX Depth*
OC48-HTTP	109,090(1,432)	138,301(874)	2.536(0.023)	0.09(0.01)	40.76(1.49)	0.070(0.002)	61.24(2.19)	2(0)
OC48-SMTP	6,913(76)	9,799(146)	2.835(0.027)	3.41(0.21)	45.70(0.87)	0.051(0.002)	79.92(0.96)	2.9(0.57)
OC48-DNS	22,025(384)	52,126(1109)	4.733(0.039)	11.00(0.21)	36.43(0.39)	0.085(0.004)	97.99(0.23)	3.5(0.53)
OC48-WinMX	8,890(225)	33,593(599)	7.560(0.171)	28.68(0.56)	35.39(1.17)	0.035(0.012)	98.99(0.16)	3.9(0.57)
ABIL-HTTP	30,308(3204)	30,196(3,168)	1.993(0.014)	0.48(0.10)	76.79(2.49)	0.082(0.033)	36.08(9.88)	2.2(0.42)
ABIL-SMTP	7,191(162)	8,512(312)	2.367(0.049)	7.03(0.43)	56.02(1.07)	0.026(0.007)	79.58(1.52)	3.1(0.32)
ABIL-DNS	21,307(428)	91,851(4,987)	8.619(0.366)	24.12(2.13)	24.62(1.58)	0.101(0.002)	97.53(0.18)	3.8(0.63)
ABIL-Blubster	1,640(85)	16,106(506)	19.667(0.768)	52.96(4.08)	41.37(2.82)	0.218(0.024)	99.49(0.01)	4.3(0.48)
WIDE-HTTP	10,922(10,512)	12,590(10,675)	2.389(0.102)	0.22(0.08)	71.96(7.96)	0.155(0.230)	57.46(13.88)	2(0)
WIDE-SMTP	2,242(61)	3,061(203)	2.732(0.148)	4.52(0.49)	55.31(1.40)	0.152(0.036)	91.04(1.71)	2.4(0.5)
WIDE-DNS	9,830(321)	18,799(613)	3.825(0.026)	6.99(0.24)	40.95(0.55)	0.432(0.006)	98.85(0.20)	2.3(0.7)
WIDE-NetBIOS	3,486(887)	3,475(892)	1.993(0.028)	0.02(0.04)	97.82(0.55)	0.081(0.023)	9.74(2.76)	1.1(0.3)

Table 2: Measured features (averages) for TDGs generated within a 300 sec time window. Values in parenthesis provide the standard deviation for the measured quantity after generating each TDG twelve times; each for every 300 seconds-long disjoint interval of an one-hour-long traces (12x300sec=1hour). *Info:* Blubster(P2P) TDG: UDP Port 41170, WinMX(P2P) TDG: UDP Port 6257.



(a) HTTP Vs eDonkey (b) NetBIOS Vs SMTP (c) DNS Vs WinMX

Figure 2: Empirical Complementary Cumulative Distribution Functions (CCDF), $P(X > x)$, of the degrees of various TDGs from trace OC48. Stability of measured distributions, across disjoint intervals (one curve for every 300 sec), is shown by the multiple overlapping curves.

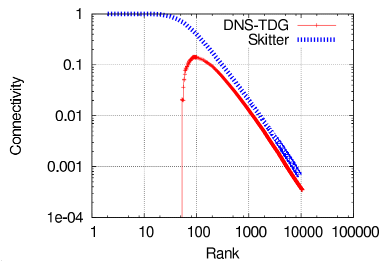


Figure 3: Rich Club Connectivity metric of a 300 sec DNS TDG (WIDE trace). By contrast, the rich club connectivity of the AS-Level Internet topology graph from Skitter (CAIDA).

tion intervals, which corresponds to an hour of monitored traffic for each trace. The values in the table are averaged over the 12 intervals and values in parenthesis provide the standard deviation of each metric, and which is typically small. The small standard deviations suggests that: (a) the TDG properties seem very stable over the duration of observation, and (b) that 300 seconds is a reasonable interval of observing the formation of TDGs. We now discuss the different graph metrics and their importance.

Average Degree: This metric is calculated by counting both in-coming and out-going edges, hence is the degree of a host if we ignore the directivity. Therefore, it indicates the popularity of a host, i.e., how many distinct IP nodes a host interacted over the observation interval. The average degree is a first approach into quantifying the coarseness of a graph; graphs with high average degrees tend to be tightly connected [8]. As expected, this metric is high in P2P TDGs.

Max Degree Ratio (MDR): MDR is the maximum degree in the graph normalized by the number of nodes in the graph minus one (i.e., the maximum *possible* degree of a node in the graph). Intuitively, high MRD suggests the presence of a dominant node in the graph e.g., a busy server. As shown in Table 2, DNS always has a very high degree node indicating the presence of a dominant DNS server. However, high MDR can also reveal the existence of malicious activity. For example, in the HTTP TDG of the WIDE backbone trace, the relatively high MDR is due to the presence of malicious heavy scanning activity in the trace.

Directionality: This metric captures the percentage of nodes in the graph that have: (a) only in-degree (sinks), (b) only out-degree (sources), or (c) both in- and out-degree (InO). Nodes with InO are both initiators and "receptors" of communications, and hence act both as *clients and servers*. For example, in the OC48 trace, we see that HTTP has practically zero percent of such nodes (pure client server application), while the P2P protocols WinMX (UDP Port 6257) and Blubster (UDP Port 41170) have 28% and 53%, respectively, of nodes with "dual" role. The importance of directionality is also shown for the NetBIOS (UDP Port 137) where we have address-space scan activity. In this TDG, there is a large amount of nodes with only in-degree (IP address being scanned). As we can see from Table 2, this behavior is unique for NetBIOS.

Size of Giant Connected Component (GCC): We use the term "component" to refer to a maximally connected subgraph. By definition, there must be an *undirected* path from each node u to each other node v in a component. GCC is the largest connected component in a traffic dispersion graph. We will report this quantity as percentage of the total number of nodes in the TDG. Intuitively we expect densely connected TDG communities, as commonly found in P2P protocols, DNS and network-gaming overlays, to have a large connected component that concentrates the majority (usually, >90%) of participating hosts. Such components are in general not present in typical client server applications,

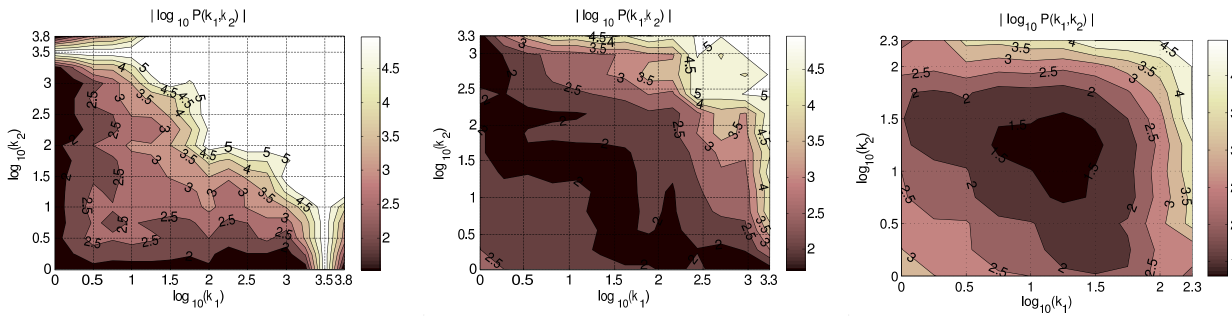

(a) HTTP (client-server). (b) DNS (client-server & P2P). (c) WinMX (P2P).

Figure 4: Joint Degree Distribution(JDD): $P(k_1, k_2)$ gives the probability of a randomly selected edge to connect nodes of degree k_1 and k_2. Same-colored areas in the contours give regions where an edge is equal likely to exists. Dark colored region shows areas with the higher concentration of edges (higher probability P), and white color denotes regions with no edges. Data are from the OC48 trace.

such as HTTP (Table 2). Moreover, the total number of such connected components is an important quantity as we will see later.

Depth: For this metric, we consider the dynamic (temporal) nature of the graph, and we consider edges and nodes in the order of first appearance in an online fashion. We attempt to capture the "spread" of the communication (e.g., u talks to v, then v talks to q etc.). For calculating depth, the first time we see a directed edge (u, v), if we have not seen node u before we give it a depth of zero. If v is a new node in the graph then $depth(v) = depth(u)+1$. The importance of this metric lies also in its ability to identify worm propagation, which naturally leads to high depths [4].

Node Degree Distribution: The degree distribution of any TDG can be represented by its corresponding marginal in-degree and out-degree distributions. In this work, we use the degree distribution of the undirected graph in the same way as we discussed for average degree. We present data from the OC48 trace with similar findings for all other traces. For each TDG, twelve distributions are generated, one for each disjoint 300-second-long interval of the trace. The empirical degree distributions of Fig. 2 indicate that: (a) as with most Internet data [3], many TDGs (e.g., HTTP, DNS, NetBIOS) exhibit highly skewed distributions, and that (b) our measured quantities are stable over the intervals of observation, since the empirical distributions from multiple disjoint intervals of the trace are very close together. It is interesting to observe the distinct behavior of NetBIOS where the majority of nodes have degree of one (nodes being scanned, with only a single in-edge), compared to the degree distributions of the other protocols.

In Fig. 2, the corresponding average degrees and their standard deviations[1] are: 2.48 with $\sigma = 32.96$ for HTTP, 4.83, with $\sigma = 27.87$ for DNS and 2.01 with $\sigma = 29.57$ for NetBIOS. For typical exponentially distributed data the Coefficient of Variation ($CV(X) = \sqrt{Var(X)}/E[X]$) [3] is close to 1, for large enough samples. However, the CV for the HTTP, DNS and NetBIOS TDGs are 13.3, 5.8, 14.7 respectively, indicating a significant level of variation from the average degree. These results are for the first 300 secs of the OC48 trace, with very similar findings for all other intervals. Even though this is not a strict rule for all traces we investigated, the distributions of HTTP and DNS shown in Fig. 2 can be well described by a power law relationship of the form $P(X > x) = 1 - P(X \le x) \cong c \cdot x^{-\alpha}$, with exponents $\alpha = 1.10$ for HTTP and $\alpha = 1.27$ for DNS and goodness-of-fit $R^2 = 0.98, 0.99$ respectively. For completeness, we report that the CV for WinMX, eDonkey (TCP Port 4662) and SMTP are 1.6, 1.8 and 3.5 respectively. Clearly, even though not exponentially distributed, the degrees of the two P2P protocols exhibit much smaller variability.

Rich Club Connectivity (RCC) metric: The rich club connectivity is typically analyzed with the following procedure [10]. We sort nodes in the order of decreasing degree (x-axis in the plot) which we call the **degree rank** of the node. Then, to calculate the connectivity at k, we consider the group of nodes from rank 1 to rank k and we compute the number of edges that exist between these nodes, over the maximum possible number of edges between them (i.e., when they form a perfect clique). In Fig. 3, we plot the RCC for a typical TDG from the first 300 sec of the WIDE trace.

In contrast to many structured graphs with power-law degree distributions, such as the AS-level Internet topology (Fig. 3), in TDGs we have *no clustering of highest degree nodes*. In other words, top ranked nodes (e.g., large DNS servers) are not observed to exchange packets with each other. We confirmed this finding in a number of other TDGs (e.g., for HTTP, SMTP, etc.) and with many other traces. Some notable exceptions are few P2P protocol TDGs (e.g., Blubster P2P of Table 2) where high degree nodes are connected. The above observations support our statement that TDGs appear to be different in this respect from many other structured power-law graphs.

Joint Degree Distribution (JDD): JDD goes one step further than the degree distribution and gives the probability $P(k_1, k_2)$ of a randomly selected edge to connect nodes of degrees k_1 and k_2. Mahadevan et al. [8] emphasize the ability of this metric to fully capture the characteristics of a large family of graphs (e.g., the AS-Level Internet topology). The JDD for three TDGs derived from the first 300 sec interval of the OC48 trace are shown in Fig. 4. The contour plots have logarithmic x- and y-axis, as well as probabilities $P(k_1, k_2)$. High probabilities therefore appear as small values of $|\log_{10} P(k_1, k_2)|$. While not all traces give identical JDDs for the same edge filter, they lead to similar observations as shown here.

Fig. 4(a) graphically illustrates the JDD of a traditional client-server application such as HTTP. As expected, the re-

[1]The Standard Deviation(σ) calculated here captures the variability of the degrees found in a single TDG and is different than the σ of the average degree across multiple TDGs as given in Table 2.

gion with the higher concentration of edges (darker region) is for low k_2 with high k_1 (and vice versa due to symmetry, $P(k_1, k_2) = P(k_2, k_1)$). The concentration of edges gradually decreases as we jointly increase k_1 and k_2. The white colored region at the top right corner indicates the zero probability of high degree nodes to be directly connected with each other. This finding is also supported by the RCC metric (Fig. 3). In general with TDGs, the majority of edges connect nodes of high degree with nodes of low degree (i.e., dissasortive networks [10]). On the other hand, Fig. 4(c) quantifies what we originally observed in the P2P (WinMX) visualization of Fig. 1(c), where average degree nodes are connected with each other. This is shown with the darker colors of the contour plot placed in the middle of the graph, illustrating the prevalence of edges connecting "medium" degree nodes. Not surprisingly, the DNS TDG (Fig. 4(b)) shows both signs of P2P (e.g., communication among some DNS servers) as well as client-server interactions (i.e., low degree clients connected with high degree DNS servers).

5. CONCLUSIONS AND FUTURE WORK

Two essential features in network monitoring tools dealing with vast amounts of network data are *aggregation* and *the ability to spot patterns.* TDGs represent a natural extension of previous approaches that have aggregated at the packet, flow, and host levels by aggregating across nodes. The aggregation across nodes also reveals patterns of social interaction across nodes that are specific to applications. These interaction patterns or graph structures can then be used to visually and quantitatively monitor existing applications and potentially detect concealed applications and malcode.

Assuming that not many diverse application use the same port number, port-based TDGs can be used in order to identify the type of application utilizing a given port. We envisage such a system working as follows. First, given any type of edge filter (e.g., a port number) we first construct the TDG. Next, using graph metrics, we identify the nature of the application on that port (e.g., if is a client-server, peer-to-peer, or malware application). The filter selection can be: (a) extracted automatically, triggered by an anomalous behavior, such as a burst of traffic at some port, or by a heavy hitter detection system (e.g., monitor the TDGs on the ports with the most flows, bytes, packets, etc.), or (b) given *a priori* by the network administrator (e.g., monitor the Web, Email, and DNS TDGs etc.). In the second scenario, deviations from "normal" graph metrics can be used to trigger an alarm.

Preliminary results show that simple visualizations including the total number of connected components and the GCC for various protocols, can be used to infer the nature of applications. For example, Fig. 5 shows a scatter plot for the top ten most active ports (in number of flows) for the OC48 trace. Interestingly the ports corresponding to well known P2P applications (two ports for Soribada, one for WinMX and two for eDonkey) show relatively small number of components and one large GCC. In addition, it is encouraging to see the stability in this metric since points corresponding to multiple disjoint 300 sec intervals are clustered closely together.

While this paper introduces the idea of TDGs as a potentially promising network monitoring tool we fully realize that much work needs to be done. Our future directions include **(a)** designing efficient algorithms and thresholds that can be used to identify applications from a given TDG, **(b)** contrasting the behavior of TDGs when deployed at an access link as well as at the central router of an enterprise network (i.e., monitoring a large portion of the traffic), **(c)** experimenting with different edge filters, e.g., that capture the frequency of an edge's appearance (weight of the edge), **(d)** deploying a working system that provides real-time TDGs and traffic characterization at a live deployment.

Acknowledgments

The authors thank Flavio Bonomi and Ronak Desai, from Cisco Systems, Inc., for their support throughout this work. Finally, we would like to thank our anonymous reviewers for their constructive comments and suggestions. Support for this work was provided by a Cisco URP grant.

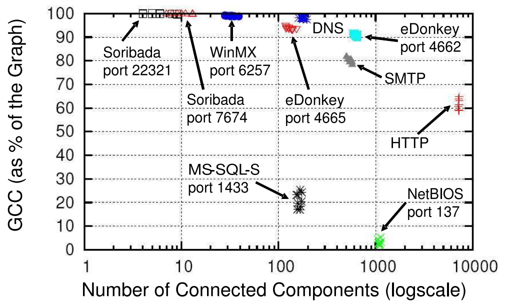


Figure 5: Scatter plot: Size of largest connected component versus the number of connected components per destination-port number for multiple intervals of the OC48 trace.

6. REFERENCES

[1] W. Aiello, C. Kalmanek, P. McDaniel, S. Sen, O. Spatscheck, and J. Merwe. Analysis of Communities of Interest in Data Networks. In *Passive and Active Measurement Conference (PAM)*, 2005.

[2] S. Cheung et al. The Design of GrIDS: A Graph-Based Intrusion Detection System. *UCD TR-CSE-99-2*, 1999.

[3] M. Crovella and B. Krishnamurthy. *Internet Measurement: Infrastructure, Traffic and Applications.* John Wiley and Sons, Inc, 2006.

[4] D. Ellis, J. Aiken, A. McLeod, and D. Keppler. Graph-based Worm Detection on Operational Enterprise Networks. *Technical Report MITRE Corporation*, 2006.

[5] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. ACAS: Automated Construction of Application Signatures. In *ACM SIGCOMM MineNet Workshop*, 2005.

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

[7] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker. Unexpected means of Protocol Inference. In *ACM Internet Measurement Conference (IMC)*, 2006.

[8] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat. Systematic Topology Analysis and Generation Using Degree Correlations . In *ACM SIGCOMM*, 2006.

[9] A. Moore and D. Zuev. Internet Traffic Classification using Bayesian Analysis Techniques. In *ACM SIGMETRICS*, 2005.

[10] M. Newman, A. Barabasi, and D. J. Watt. *The Structure and Dynamics of Networks.* Princeton Press, 2006.

[11] 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.

[12] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *ACM SIGCOMM*, 2005.

[13] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: Defending Against Sybil Attacks via Social Networks. In *ACM SIGCOMM*, 2006.