Synoptic Graphlet: Bridging the Gap between Supervised and Unsupervised Profiling of Host-level Network Traffic

Yosuke Himura, Kensuke Fukuda, *Member, IEEE,* Kenjiro Cho, *Member, IEEE,* Pierre Borgnat, *Member, IEEE,* Patrice Abry, *Fellow, IEEE,* and Hiroshi Esaki, *Member, IEEE,*

***Abstract*—End-host profiling by analyzing network traffic comes out as a major stake in traffic engineering. Graphlet constitutes an efficient and common framework for interpreting host behaviors, which essentially consists of a visual representation as a graph. However, graphlet analyses face the issues of choosing between supervised and unsupervised approaches. The former can analyze a priori defined behaviors but is blind to undefined classes, while the latter can discover new behaviors at the cost of difficult a posteriori interpretation. This work aims at bridging the gap between the two. First, to handle unknown classes, unsupervised clustering is originally revisited by extracting a set of graphlet-inspired attributes for each host. Second, to recover interpretability for each resulting cluster, a *synoptic graphlet*, defined as a visual graphlet obtained by mapping from a cluster, is newly developed. Comparisons against supervised graphlet-based, port-based, and payload-based classifiers with two datasets demonstrate the effectiveness of the unsupervised clustering of graphlets and the relevance of the a posteriori interpretation through synoptic graphlets. This development is further complemented by studying *evolutionary tree* of synoptic graphlets, which quantifies the growth of graphlets when increasing the number of inspected packets per host.**

***Index Terms*—Internet traffic analysis; Unsupervised host profiling; Microscopic graph evolution; Visualization**

I. Introduction

An essential task in network traffic engineering stems from host-level traffic analyses, where the behavior of a host is characterized based on traffic (i.e., packet sequence) generated from the host. Host-level traffic analyses enable to find users of specific applications for the purpose of traffic control, to identify malicious or victim hosts for security, and to understand the trend of network usage for network design and management. Flow analysis, which also constitutes an important networking stake, can be fruitfully complemented by host profiling (e.g., by breaking down host behaviors into flow characteristics).

Numerous attempts have been made to develop statistical methods for host profiling. Such methods aim at overcoming packet encryption, encapsulation, use of dynamic ports, or dataset without payload – situations that impair the classical approaches relying on payload inspection [26, 17, 24] and port-based rules [4]. The most recently proposed ones are based on heuristic rules [15], statistical classification procedures [27, 18, 22], Google database [28], or macroscopic graph structure [30, 13, 10, 11].

In particular, an effective yet heuristic approach to host profiling is based on *graphlets* [15, 14, 19]. A graphlet is a detailed description of host communication patterns as a graph, as illustrated in Figure 1. For each flow, the 5-tuple defining it (proto, srcIP, dstIP, srcPort, dstPort) gives a set of attributes ($A_1, A2, \ldots$) and the communication pattern of a host is the union, for all flows, of edges connecting nodes associated to flow's attributes. This leads to diverse visual shapes of graphlets depending on the host's flows. The graphlet representation facilitates the intuitive analysis of differences and resemblances among host behaviors, whereas conventional approaches directly handle numerical values of statistical features, which are difficult to interpret.

However, as for any host-profiling approach, the use of graphlets faces the classical issue of choosing supervised versus unsupervised procedures. Supervised approaches rely on a priori determined classes or models of graphlets [15], predefined by human experts in a necessarily limited number, and these approaches cannot substantially classify new or unknown host behaviors. Unsupervised approaches are adaptive insofar as the data directly define the output classes of graphlets and can discover behaviors never observed before. These approaches, however, potentially produce clusters composed of a large number of numerical features that cannot receive easy meaningful or useful interpretation.

The present work aims at bridging the gap between the two types of approaches. The main idea for this is the combination of two techniques: To avoid the limitation of supervised approach, we use an unsupervised clustering of graphlets that is able to capture previously unknown classes; To ease the difficulties of unsupervised approach, the resulting clusters are re-visualized into *synoptic graphlets* that allow us to interpret the clusters obtained. Our approach is evaluated with two large datasets of traffic collected on two different links (Sec. III). The article is organized along three contributions.

First, the classical problem of supervised classification is revisited (Sec. IV). This investigation comprises two aspects: a list of graphlet-based features is proposed to quantify in a relevant way the visual graphlet shape associated with each host; An unsupervised clustering method is applied to these

Yosuke Himura and Hiroshi Esaki are with The University of Tokyo.
Kensuke Fukuda is with National Institute of Informatics and PRESTO, JST
Kenjiro Cho is with Internet Initiative Japan (IIJ) and Keio University.
Patrice Abry and Pierre Borgnat are with CNRS and École Normale Supérieure de Lyon (ENS de Lyon), Laboratoire de Physique.
Work supported by CNRS/JSPS Grants "Benchmarking and statistical analysis tools for modern Internet and sensor network traffics," 2010 and 2012.

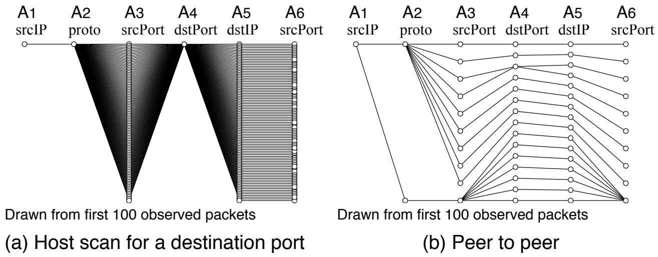


Fig. 1. Examples of graphlets. Traffic from a single source host is represented as a graph connecting attributes such as proto, srcPort, dstPort, and dstIP.

features to yield classification in terms of graphlet shapes. Comparisons against a supervised graphlet-based classifier (BLINC [15]), a port-based one, and a payload-based one allow us to check that most clusters match well-known host behaviors. This result shows that our method makes it possible to discover unknown graphlets, which avoids the problem faced by supervised approaches.

Second, the issue of automatically providing interpretation of the output of the unsupervised clustering is addressed (Sec. V). We solve the *inverse problem* of reconstructing a *synoptic graphlet*, defined as a graphlet inferred from each obtained cluster, by using an original mapping of the cluster attributes (cluster centroid) into a graphlet. Our development of synoptic graphlets shows that an interpretable meaning can be associated automatically to each cluster without any a priori expertise. The effectiveness of synoptic graphlets, which successfully provide interpretability for unsupervised approaches as shown in this work, ends up in bridging the gap between supervised and unsupervised methods.

Third, the nature of host behavior is further studied via synoptic graphlets (Sec. VI). The use of synoptic graphlets is expanded to creating an *evolutionary tree*, which explores the visually intuitive growth of a set of synoptic graphlets as a function of the number P of inspected packets per host. This study is useful in integrating host-level traffic characteristics of different P in an interpretable manner, and in quantifying the order of magnitude P beyond which further increase does not lead to substantially more relevant host profiling, i.e., how many packets P we need to profile hosts.

II. Preliminaries

Before turning to the method itself and the datasets used in the next sections, we recall the definition of graphlets in the context of Internet traffic and discuss related work. Then we propose an overview of our approach.

A. Graphlet

A *graphlet* is defined as a graph having the following characteristics in the context of network communication: (1) the graph is composed of several columns $(A_1, A_2, \ldots)$ of nodes, where each column represents one attribute of packets or flows, (2) a node (vertex) in a column is a unique instance of the attribute, and (3) there is an edge between two nodes of neighboring columns if at least one packet/flow has the two corresponding attributes. Columns of a graphlet are usually related to flow attributes (5-tuple): proto (protocol number), srcIP (source IP address), dstIP (destination IP address), srcPort (source port number), and dstPort (destination port number), which are specified in the header field of every packet.

Figure 1 illustrates two manually annotated examples of graphlets drawn with $P = 100$ packets per source host. Figure 1(a) shows that the source host, which is represented as the single node in srcIP column, sends packets to a specific destination port of many destination hosts (almost one packet per flow); This suggests that the source host is a malicious scanner aiming to find hosts running a vulnerable application corresponding to the port. Figure 1(b) displays a host communicating with several hosts without any specific source/destination port, and hence this host is a peer-to-peer user (not server or client). As shown in these examples, a strong merit of graphlets is the visual interpretability of host characteristics as compared to examining a large number of raw packet traces or directly handling a set of numerical statistics.

We draw a graphlet from host-level traffic. Here, host-level traffic is defined as the sequence of packets sent from the host; Headers in those packets contain source IP addresses equivalent to the host's address. Note that this measurement does not necessarily capture initiation of communication (e.g., TCP hand shake). Each graphlet is drawn from a certain number of observed packets P sent from each host.

The graphlet we use is composed of six columns $A_1, \ldots, A_6$, which represent srcIP-proto-srcPort-dstPort-dstIP-srcPort[1]. The order of columns is different from the original definition [15]. We consider that srcIP-srcPort-dstPort-dstIP should be more comprehensive, because it clarifies the activity of computer processes inside end-hosts (IP-Port pairs) and network-wide inter-process communication among hosts (srcPort-dstPort pairs). We place srcPort at the right side again to capture the relation between dstIP and srcPort (inspired by [14]). Since we draw one graphlet per source host, there is only one point in the left column (srcIP).

B. Related work and open issues

Here the standpoints of the graphlet-based works and of this work are presented in the context of network traffic classification conducted over the course of a decade.

Many statistics-based methods for traffic analyses have been proposed to classify flows and host characteristics by means of supervised and unsupervised methods. These studies have made use of various supervised machine learning methods such as nearest neighbors [16, 8, 20], Bayesian statistics-based techniques [23, 16, 25, 20], decision tree [16, 25, 20], Support Vector Machine (SVM) [16, 20], or even natural language processing on Google search results [28]. The others have leveraged unsupervised ones including K-means clustering [18, 1, 7], or hierarchical clustering [18]. Both the approaches have been applied to traffic features from various aspects – packet sizes only [1, 8], combinations of packet sizes, flow

[1] We define 'pseudo' source and destination ports for ICMP to be $srcPort = dstPort = icmp\_code$ in order to consistently draw graphlets.

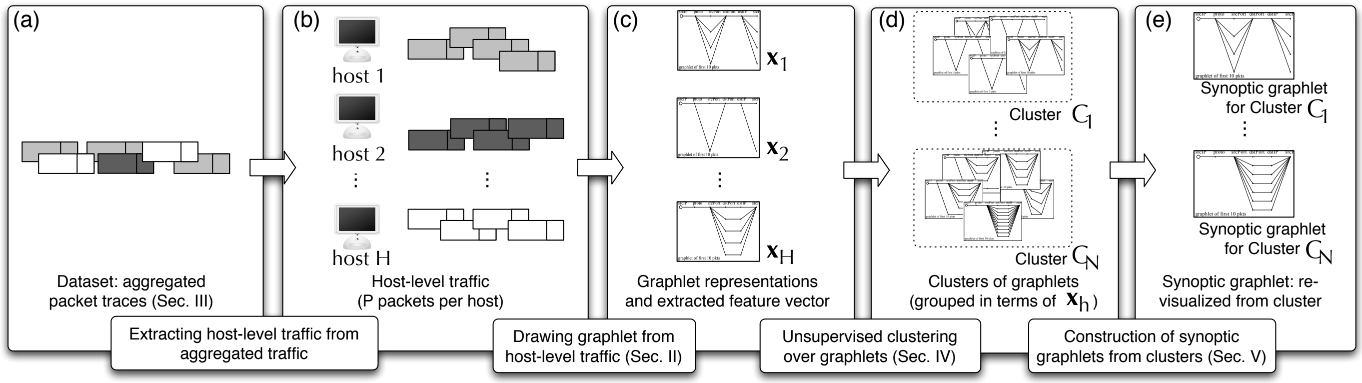


Fig. 2. Overview of our approach.

sizes, inter-arrival times, flow durations etc. [23, 7, 25, 20], and/or entropy regarding the number of related hosts/ports [18, 30]. Those statistics-based methods are capable of overcoming packet encryption, encapsulation, use of dynamic ports, or dataset without payload, which are limitations on conventional approaches relying on payload inspection [26, 17, 24] and port-based rules [4].

Several recent studies particularly focused on large-scale host-to-host connections [27, 22, 13, 10, 11, 29], the use of which promisingly enables to visualize how hosts communicate with one another and enables to find groups of hosts communicating with each other. These works leverage existing graph-based analytical capabilities such as feature extraction regarding complex networks [10], community mining techniques [11], or block identification in communication (adjacency) matrix [13, 29].

Different from those previous works, the approach described here focuses on graphlets – detailed aspects of host behaviors including the usage of protocols and source/destination ports. The use of graphlets has been motivated by their visual interpretability (as shown before), and has been conducted in a few works. For example, Karagiannis *et al.* perform supervised classification of flows based on graphlet models pre-determined by human experts [15]. Other works characterize graphlet-based host behaviors in unsupervised manners as follows: Karagiannis *et al.* discuss in-degrees and out-degrees of nodes and average degrees of graphlets in [14], and focus on manual finding of typical graphlets as well as on time transition of those features; In [6], Dewaele *et al.* classify hosts, making use of various features (some of them inspired by graphlets) applied to an unsupervised clustering technique.

To overcome the various limitations of supervised/unsupervised approaches that were discussed previously, and in contrast to previous works, the present article aims at bridging the gap between the two analytical approaches on graphlets by proposing a new framework for graphlet manipulation.

C. Overview of our approach

The three contributions of this work are: (1) the automation of finding typical graphlets via unsupervised clustering in an interpretable manner, (2) a method to re-visualize graphlets from clustering results, and (3) an analysis on evolution of typical graphlet shapes while increasing the number of packets per graphlet, which is complementary to analyses on time-transition of graphlet features. Each contribution is an important step of our method. Steps (1) and (2) are depicted in Figure 2 and step (3) is represented in Figure 9. Our method is organized as follows.

As a preprocessing step, aggregated traffic traces are first computed. (Figure 2(a)). The traffic is measured in a backbone link and composed of packets sent from hundreds of thousands of hosts (Sec. III). We identify per-host traffic (Figure 2(b)) according to the source IP addresses specified in the packets, and draw graphlets from the first P measured packets sent from each host (Figure 2(c)).

Step (1): An unsupervised clustering over graphlets is conducted to find typical graphlets (Sec. IV). A numerical feature vector $\mathbf{x_h}$, which represents shape-based characteristics of a graphlet, is extracted from the graphlet of P packets sent from host h. The set of feature vectors $\mathbf{x_1}, \ldots, \mathbf{x_H}$, representing a set of H hosts, is used for hierarchical clustering to produce clusters of hosts $C_1, \ldots, C_N$ (Figure 2(d)). Cluster C_i consists of hosts that are similar in terms of their feature vector in the feature space. For each cluster, we obtain the components of a representative feature vector $\mathbf{x}$ which will be converted to graphlets in the next step.

Step (2): Resulting clusters are visualized to recover interpretability (Sec. V). Since unsupervised clustering handles numerical features and thus loses visual information of graphlets, we re-visualize a representative graphlet associated with each cluster (Figure 2(e)). The reproduced graphlet, called synoptic graphlet, is derived from the feature vector $\mathbf{x}$ of the centroid of a cluster. We develop an original method to re-visualize synoptic graphlets in a deterministic manner, since conventional probabilistic ways of graph rewiring are not suitable for highly-structured graphlets.

Step (3): Additionally, the evolutionary nature of synoptic graphlets is studied (Sec. VI). The key observation is that our knowledge of hosts may evolve as P increases from 1 to larger numbers. To study the evolution of the associated synoptic graphlets, we build an evolutionary tree of synoptic graphlets that evolve from the single-line graphlet (the only existing shape for $P = 1$) to the diversity of synoptic graphlets.

This evolutionary tree is obtained by combining the clustering results of increasing P (see Figure 9). It provides intuitive understanding of the divergences and convergences in the growth of host characteristics as P increases.

III. Datasets

This section describes first the two datasets used for the validation of the proposed method, and, second, discusses how combining three different and classical traffic classifiers produces surrogates for real traffic ground-truth.

A. Traffic traces

We analyze traffic traces stored in the MAWI repository [21, 3] and traces measured at Keio University (used in [16] as Keio-I and Keio-II). MAWI traffic [21, 3] is measured on a transpacific IPv4 link between the U.S. and Japan for 15 minutes everyday. The public repository removes packet payloads, while the private repository retains payloads, up to the first 96 bytes. Results are reported here based on 12 MAWI traces collected once a month (on the 14th) in 2008. Keio traces used here are those presented in [16] and measured for 30 minutes, for two different days in 2006, on a bi-directional edge link in a campus of Keio University. Packet payloads up to 96 bytes were also preserved. We first removed the packets related to protocols other than TCP, UDP, and ICMP.

In the results reported below, we use the source hosts[2] sending at least 1000 packets for MAWI trace (respectively, 100 packets for Keio trace). This choice balances the trade-off between (a) having a lower reliability when hosts do not exchange enough packets, and (b) not keeping enough hosts when the required number of packet is too high. It has been checked that this arbitrary choice is not crucial; Results (e.g., the evolution of the number of clusters) similar to those obtained with $500 < Q < 1000$ were drawn with $200 < Q < 500$, or with $100 < Q < 200$ for MAWI traces, where Q denotes the number of packets sent by a host (observed in a trace). We will quantitatively evaluate the differences of results regarding the choice of Q in the future.

Each of the 12 MAWI traces contains about 1,700 analyzed hosts, yielding approximately a number of analyzed hosts $H = 20,000$ in total for the 12 traces, and the 2 Keio traces contain about $H = 10,000$ hosts in total (H is the number of analyzed hosts). Those analyzed hosts for MAWI data account for 1.1% (19K out of 1.7M) regarding the number of hosts, 86% regarding the number of packets (207M out of 239M), and 93% regarding the number of bytes (1.43T out of 1.52T).

B. Pseudo ground-truth generators

Traffic analysis methods generally have to be evaluated with dataset annotated from ground-truth. A crucial issue raised in the recent literature, however, lies in designing a procedure to obtain ground-truth on actual traffic traces. Most of researches indeed have regarded ground-truth as the labels put by a single payload-based packet classifier. However a lot of packets are labeled as unknown by payload classifiers (as exhibited in this paper). Also, payload-based methods do not necessarily produce correct outputs. To improve the ground-truth coverage and accuracy, we carefully create three sets of pseudo ground-truth from different methods detailed here.

(a) Reverse BLINC. BLINC was originally proposed in [15] and extended to Reverse BLINC in [16], which is now state-of-the-art. BLINC profiles a pair of a source address and a port, and once the pair is matched with one of the heuristic rules based on the graphlet models, all pairs connected to that pair are classified. We used the default setting of Reverse BLINC as in [16]. BLINC's classification framework is WWW, CHAT, DNS, FTP, MAIL, P2P, SCAN, and UNKN (unknown). Since this classifier reports classification results as flow records, we need to convert them into a host-level database. For each source host, we collect a set of flows generated from the host and select the category (except for UNKN) that is the most frequent among the flows. For example, if ten flows from a host are classified into three DNS, one WWW, and six UNKN, then the type of the host is identified as DNS.

(b) Port-based classifier. We use another classifier, which was originally developed in [5] and also used in [2, 9]. This tool inspects a set of packets sent from a host, considering port numbers, TCP flags, and the number of higher/lower source/destination ports and destination addresses. The classification categories are WWWS (web server), WWWC (web client), SCAN, FLOOD (flooding attacker), DNS, MAIL, OTHERS, and UNKN [9]. This tool reports host-level classification results by itself.

(c) Payload classifier. We also use the payload-based classifier developed in [16] [3]. This classifier inspects the payload string of each packet by comparing it with its signature database. The classification categories we select are WWW, DNS, MAIL, FTP, SSH, P2P, STREAM, CHAT, FAILED (when the packets have no payload), OTHERS (minor flows such as games, nntp, smb, and snmp), and UNKN. Since this tool also generates outputs in the form of flow tables, we merge them into host-level reports by the same means used to aggregate outputs from Reverse BLINC.

The hosts annotations given by the three classifiers of different perspectives are used to evaluate the unsupervised analysis on graphlets that is presented in the next section.

IV. Unsupervised graphlet analysis

We detail the first step of the method, which is an unsupervised classifier for typical behaviors of hosts that does not rely on predefined models. However, it will still allow us afterwards to provide visual interpretation of the behaviors found.

[2] It should also be meaningful to analyze destination hosts; With this analysis, for instance, we will be able to capture hosts receiving lots of attack packets. As a first step, we selected to analyze source hosts because of the easier interpretation of results; Packets sent from a host can be well explained by the application of the host, compared to packets received by a host.

[3] In our preliminary experiment, we examined l7-filter [17] and found that the tool generated rather unreliable outputs because of loose payload signatures that are represented as regular expressions with a few bytes. Also, we found that OpenDPI [24] produced mostly unknown reports because it uses strict rules.

TABLE I
NOTATIONS FOR GRAPHLET DESCRIPTION. AN ATTRIBUTE HAS TWO DIFFERENT DEGREE DISTRIBUTIONS BASED ON DIRECTION (E.G., A_2 IS SEPARATED INTO $2:1$ AND $2:3$). SEE SEC. II-A FOR DETAILS.

A_i	i-th column (or attribute) of graphlets (from left to right)
$v_{k,i}$	Node (vertex) in A_i
$i:j$	Direction from A_i to A_j ($j = i \pm 1$)
$d_{k,i:j}$	In/out-degree of node $v_{k,i}$: in-degree for $i:i-1$ (left half of $v_{k,i}$) and out-degree for $i:i+1$ (right half of $v_{k,i}$)
$D_{i:j}$	Empirical distribution of in/out-degrees in A_i

TABLE II
NOTATIONS FOR GRAPHLET CLUSTERING.

$\mathbf{x_h}$	Host h's graphlet feature vector, composed of the five degree-based features (Figure 3)
Dim	Dimension of $\mathbf{x_h}$ (44-dimensional for 6 columns)
H	Number of hosts analyzed
P	Number of packets per graphlet
C_i	Cluster of label i obtained
N	Total number of clusters obtained
θ	Distance-based threshold for clustering

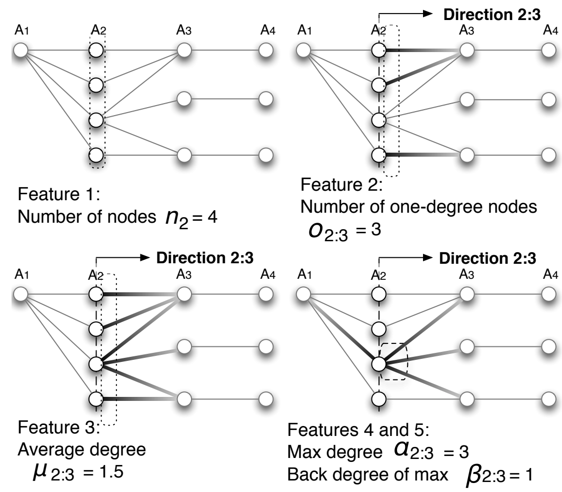


Fig. 3. Shape-based features for a graphlet (i.e., behavior of a host).

A. Methodology for unsupervised graphlet analysis

1) Extracting shape-based features from graphlets: We first extract numerical feature values from graphlets, because visual graphlets cannot be used directly as input to conventional statistical methods (except for image processing). We choose afterwards several types of features related to shapes. We note $\mathbf{x_h}$ the feature vector for the graphlet of host h.

Notations on graphlets. We denote the six attributes (srcIP-…-srcPort) as column $A_1, \ldots, A_6$. In column A_i, the total number of nodes is n_i, and nodes are $v_{1,i}, \ldots, v_{n_i,i}$. We define $i:j$ as the direction from A_i to A_j, which is used to define the in-degree and out-degree of nodes in column A_i ($j = i+1$ or $i-1$). The in-degree of node $v_{k,i}$ is defined on direction $i : i-1$ as $d_{k,i:i-1}$, namely, $d_{k,i:i-1}$ is the number of nodes in A_{i-1} that are connected to node $v_{k,i}$ in A_i. The out-degree is similarly defined on direction $i : i+1$ as $d_{k,i:i+1}$. As a consequence, node $v_{k,i}$ is characterized by the pair of the in-degree and out-degree $(d_{k,i:i-1}, d_{k,i:i+1})$. We define the array of in/out degrees for direction $i:j$ as $D_{i:j} = (d_{1,i:j}, \ldots, d_{n_i,i:j})$ where n_i is the number of nodes in column A_i. $D_{i:j}$ gives the empirical distribution measured from an observed graphlet. Table I summarizes these notations.

Feature extraction. The proposed features are based on five types of shape-related information, described formally as follows and visually in Figure 3 (the relevance of the features is discussed later).

(1) n_i is the number of nodes in column A_i. Note that it is equal to the size of arrays $D_{i:i+1}$ and $D_{i:i-1}$. (6 columns)
(2) $o_{i:j} = \sum_{d_{k,i:j} \in D_{i:j}} I(d_{k,i:j} = 1)$, where $I(\cdot)$ is the indicator function, is the number of nodes that have degree 1 in direction $i:j$ (with $j = i \pm 1$). (10 directions)
(3) $\mu_{i:j} = \frac{1}{n_i} \sum_{d_{k,i:j} \in D_{i:j}} d_{k,i:j}$ is the average degree of direction $i:j$. (10 directions)
(4) $\alpha_{i:j} = \max_{d_{k,i:j} \in D_{i:j}} \{d_{k,i:j}\}$ is the maximum degree of direction $i:j$. (10 directions)
(5) $\beta_{i:i+1} = d_{k,i:i-1}$, where $k = \arg\max_l \{d_{l,i:i+1}\}$ is, for the node having maximum degree in $i:i+1$ (i.e., Feature 4), its degree in the backward direction $i:i-1$. If more than one node has the maximum degree for Feature 4, the pair with the highest degree is selected from among the candidates. A similar definition holds for the reverse direction $\beta_{i:i-1}$. (8 directions, since the edge columns have degree for only one direction)

As a result, from the graphlet for host h, we obtain a feature vector $\mathbf{x_h} = (x_{h,1}, \ldots, x_{h,44}) = (n_1, \ldots, n_6,\ o_{1:2}, \ldots, o_{6:5},\ \mu_{1:2}, \ldots, \mu_{6:5},\ \alpha_{1:2}, \ldots, \alpha_{6:5},\ \beta_{2:1}, \ldots, \beta_{5:6})$ of dimension of $Dim = 44$ $(= 6 + 10 + 10 + 10 + 8)$. We examine packet traces or flow lists (input) to compute these features (output). The index $i:j$ is omitted when not needed.

Examples. Figure 3 shows an example of features. For direction $2:3$, there are four nodes ($n_2 = 4$) and three nodes of one-degree ($o_{2:3} = 3$), and the average degree is 1.5 ($\mu_{2:3} = 1.5$). The second bottom node has the highest degree of three ($\alpha_{2:3} = 3$) and the degree of the node for the other direction is one ($\beta_{2:3} = 1$).

Practical meanings. Even though these features are selected from the viewpoint of graphlet re-visualization (Sec. V), a few of them can also be interpreted as traffic characteristics in a practical sense. n_i is the number of unique instances of the flow attribute (e.g., the number of destination addresses). $\mu_{i:j}$ and $\alpha_{i:j}$ are respectively the average and maximum number of unique flows of an instance of the attribute among all the instances.

Relevance of features. The selection of the five types of features is empirically motivated by two objectives: (*i*) the expected ability to obtain relevant clustering results because a few of the features are already well-known and well-studied [6] and (*ii*) the ability to re-visualize graphlets from the resulting clusters as explained in Sec. V. Also, the relative importance of the five types of features is evaluated by a feature selection method in Sec. IV-B4. Macroscopic degree-related features such as betweenness, the assortativity coefficient, or eigenvalues, are not used because graphlets are microscopic and highly structured. We only use graph-based features to evaluate the interpretability of graphlet clustering results, although there are many other well-studied features such as TCP flag, packet size, and flow size. Such features and ours are not exclusive but complementary; Using both

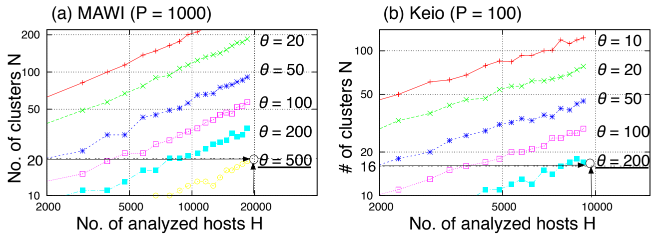


Fig. 4. Clustering threshold θ characterized by the dependency on the number of analyzed hosts H and the number of resulting clusters N.

types would enhance host profiling schemes.

2) Applying graphlet features to unsupervised clustering: Here, we establish a method to find typical host behaviors in terms of graphlet shapes. At a high-level view, a set of hosts $\mathbf{x_1}, \ldots, \mathbf{x_H}$ are grouped into clusters $C_1, \ldots, C_N$ (clusters are disjoint sets of the hosts). Table II lists the notations used for the graphlet clustering.

Feature normalization. Each feature value $x_{h,i}$ from feature vector $\mathbf{x_h}$ is mapped onto a log space as $\log_{10}(x_{h,i}+1)$. For the features related to the ID of the transport protocol, the possible ranges of the values are adjusted to the other features (i.e., addresses and ports) as follows: $\log_{10}(P\frac{x_{h,i}}{\min(3,P)}+1)$, where P is the number of analyzed packets to be drawn as a graphlet, and the value 3 stems from the number of analyzed protocols (TCP, UDP, and ICMP). Hence, this type of feature is distributed into $[0, \log_{10}(P+1)]$ as well as the other features for any P. This normalization onto the log space is motivated by our empirical observation that graphlet shapes can be logarithmically well characterized; For example, by inspecting graphlet shapes with changing P, we observed that difference in graphlet shapes between P = 10 and P = 20 was intuitively similarly significant to P = 100 and P =200 (rather than P =100 and P =110).

Unsupervised clustering. Unsupervised clustering finds groups of hosts that are similar in terms of feature values by analyzing the H hosts $\mathbf{x_1}, \ldots, \mathbf{x_H}$. The hierarchical clustering [18] with Ward's method is used, as it is known to outperform other methods (e.g., single-linkage method). The similarity between a pair of clusters (C_i, C_j) is defined as a merging cost: $\Gamma(C_i, C_j) = E(C_i \cup C_j) - E(C_i) - E(C_j)$, with $E(C_i) = \sum_{h \in C_i}(\gamma(\mathbf{x_h}, \mathbf{c_i}))^2$ the intra cluster variance in Cluster C_i, the Euclidean distance $\gamma(\mathbf{x}, \mathbf{y})$ between vectors $\mathbf{x}$ and $\mathbf{y}$, and the average feature vector $\mathbf{c_i}$ of all hosts in C_i. The distance-based threshold θ is used to separate clusters in this feature space. The clustering produces a set of N clusters $C_1, \ldots, C_N$, depending only on θ (each host is included in a single cluster only). The selection of θ is discussed in Sec. IV-B.

Motivation for distance-based threshold instead of number-based one. The distance-based threshold θ is preferable compared to cluster-number-based thresholds (such as the one for the K-means technique). This is because a consistent value of θ can be used for any P, which mitigates the burden of parameter tuning in analyses with several Ps as performed in Sec. VI. Number-based thresholds would have to be appropriately tuned through trial-and-error independently for each P, as the number of typical clusters for each P cannot be known. The consistent use of a single threshold over different Ps is empirically enabled by the normalization of the feature spaces as $[0, \log_{10} P]$, because distance between two clusters of typical behaviors will remain mostly the same for different Ps. Instead, conventional normalization into $[0, 1]$ would induce clusters with different behaviors at larger P to be located closer, requiring θ to be decreased.

Computational load. We used *hcluster* methods in the *amap* R-library. Approximately 1.5 GB memory was required for about $H = 20,000$ instances of $Dim(\mathbf{x}) = 44$ dimensional vectors. It took around 2.4 minutes with a 2.8 GHz Intel Core 2 Duo CPU with 4GB memory. By performing the clustering with changing H, we empirically confirmed that time and space complexities were both $O(H^2)$.

B. Results: finding typical patterns of host behaviors

1) Threshold selection: The distance-based threshold θ eventually determines the number of extracted clusters N according to a conventional trade-off: a too high θ misses a number of typical host behaviors, while a too low θ produces redundant clusters (i.e., different clusters having similar compositions). By changing the value of θ, we inspected the list of synoptic graphlets (representative graphlets for resulting clusters – details are defined in Sec. V) to identify whether there are redundant clusters (having same shape of synoptic graphlets) and new types of clusters (which cannot be found by large θ). We experimentally found that thresholds that balance this trade-off well are $\theta = 500$ with the MAWI traces (about $H = 20,000$ hosts) for $P = 1000$, producing approximately $N = 20$ clusters, and $\theta = 250$ with the Keio traces (about $H = 10,000$ hosts) for $P = 100$, resulting in $N = 16$ clusters. This trade-off has been manually inspected, because it is quite difficult to computationally identify redundancy of clusters in terms of the shapes of graphlets, which are one of our major focus and are enumerated in Sec. V.

Figure 4 addresses the characteristics of θ by showing its relationship to the number of analyzed hosts H and the number of clusters N obtained from (a) MAWI (for $P = 1000$) and (b) Keio (for $P = 100$). Each set of analyzed hosts was selected from a random sample of the total number of original hosts by changing the sampling rate. This figure suggests referential values of θ for each dataset to obtain a certain number of clusters that balances the trade-off well for any H.

We note that this value of θ can be consistently used for other P, and this is the reason why we do not directly use the number-based threshold. Since θ is based on the distance in the feature space, we can compare the clustering outputs from various P with a single consistent criteria. For example, smaller P might lead to fewer numbers of clustering with regard to the feature space. We confirmed that the value of θ is consistently appropriate for other P as shown in Sec. VI.

2) Typical patterns of host behaviors: Table III shows the clustering result, with $H = 20,000$ hosts at $P = 1000$ of MAWI data, obtained from a comparison between the graphlet clustering and the three classifiers, i.e., Reverse BLINC (R-BLINC), port-based classifier (Port), and payload-based classifier (Payload). This table displays the total number of hosts

TABLE III

RESULTING CLUSTERS (MAWI WITH $P = 1000$) COMPARED WITH THREE CLASSIFIERS: REVERSE BLINC (R-BLINC), PORT-BASED CLASSIFIER (PORT), AND PAYLOAD-BASED CLASSIFIER (PAYLOAD). $N = 20$ CLUSTERS ARE OBTAINED FROM THE ANALYZED $H = 20,000$ HOSTS WITH THE SELECTED THRESHOLD $\theta = 500$.

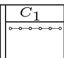
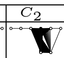
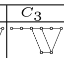
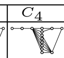
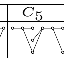
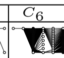
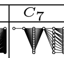
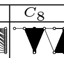
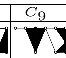
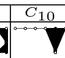
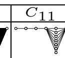
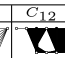
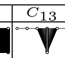
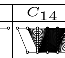
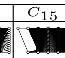
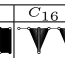
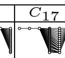
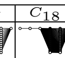
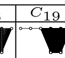
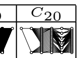
			C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	C_{20}
																						
			4594	1252	986	1283	1526	1427	1134	274	715	451	297	152	950	961	613	652	964	690	221	305
R-BLINC	WEB	10199	1612	672	348	991	825	1361	1031	9	249	5		13	885	233	199	539	681	527	7	12
	DNS	1131	12	54	35	6	50	17	25	208	4	62	56	49	1	216	156	1	38	46	92	3
	MAIL	721	17	21	8	21	25	34	39		7	2		8	16	189	183	17	44	81		9
	P2P	1660	572	14	18	22	222	3	14	33	11	17	3	10	5	285	52	14	107	21	61	176
	SCAN	191										191										
	FTP	253	33				95				5				1	8		17	1			93
	CHAT	24				1													16		7	
	UNKN	5268	2348	491	577	242	309	12	25	24	439	174	238	72	42	30	23	64	77	15	54	12
Port	WWWS	6538	1553	670	710	1178	101	1	6			1	2		892	154	9		709	538	14	
	WWWC	5457	1337			4	722	1351	987	9	255	1	1	19		35	202	532				2
	DNS	807	8	51	33	2	5	5	10	180	3	53	43	32	1	114	74		40	50	103	
	MAIL	632	27	15	5	14	24	33	39		7			11	16	151	162	12	27	84		5
	P2P	361	74	5	3	6	16	2	4			1				160	24		51	2	12	1
	SSH	645	18	466	2	3	13	1	1		2			17	14	86	5	4	3			10
	SCAN	620				2	3	1		1	6	389	19	64		38	30			1	64	2
	FLOOD	709	66	1	111	3	3	5	2	66	173	4	224		4	6	15					26
	PROXY	113	1				5	3			85				2	2	1	12				2
	FTP	129	49	6	3		25		4		4			1	2	7		5	1			22
	OTHER	121	46	4	12	3	19	3	10		3	1			1	4	2				13	
	UNKN	3315	1415	34	107	68	590	22	71	18	177	1	8	8	18	204	89	87	133	15	15	235
Payload	WEB	11392	2620	627	671	1143	810	1316	1003	4	225		1	11	858	177	183	519	694	522	6	2
	MAIL	648	29	15	6	24	24	33	42		8			2	15	169	144	12	38	83		4
	DNS	1171	14	50	33	6	53	14	25	241	4	56	54	52	1	201	169	3	40	50	104	1
	P2P	430	309	2	20	7	68								1	16	4	2				1
	SSH	1023	30	505	12	39	50	29	28		33			16	62	97	10	37	54	20		1
	FAILED	504	128	5	11	7	54	16	7	9	18	21		66	1	29	40	7	2	5	69	9
	FTP	300	28		1	1	124				6				1	12	1	18	1			107
	CHAT	152	3	12	1		3	2			2					18			110			1
	STREAM	107	41		18	3	43				1					1						
	OTHER	106	29	32	8	2	11		1		4			1	3	5	8			1		1
	UNKN	3614	1363	4	205	51	286	17	28	20	414	374	242	4	8	236	54	54	25	9	42	178
#packets to stop separation			500	1000	1000	1000	1000	1000	1000	50	1000	20	50	1000	1000	1000	1000	1000	1000	1000	500	1000

TABLE IV

GRAPHLET FEATURES EVALUATED BY FCBF.

MAWI		Keio	
feature	$SU_{i,c}$	feature	$SU_{i,c}$
$o_{i:j}$ of srcPort → dstPort	0.51	$o_{i:j}$ of srcPort → dstPort	0.57
$o_{i:j}$ of dstPort → srcPort	0.48	$o_{i:j}$ of dstPort → srcPort	0.50
$o_{i:j}$ of dstIP → dstPort	0.39	$o_{i:j}$ of dstIP → srcPort	0.41
$o_{i:j}$ of dstIP → srcPort	0.39	$o_{i:j}$ of dstIP → dstPort	0.40
$\mu_{i:j}$ of dstIP → srcPort	0.36	n_i of proto	0.06
$\beta_{i:j}$ of dstIP → srcPort	0.34		
$\mu_{i:j}$ of dstPort → dstIP	0.31		
n_i of proto	0.10		

in each category and each cell shows the number of hosts in the intersection between two classes of two classifiers. The first row of the column headings is auto-generated labels. The second row shows graphlets re-visualized from clusters (Sec. V), and the bottom row is discussed in Sec. VI.

The sparseness of Table III indicates that each cluster mostly corresponds to a type of host behavior. For instance, C_6 (containing 1427 hosts) is characterized by one typical category because most of the hosts are labeled as a category of each classifier: 1361 hosts as WEB by R-BLINC, 1351 hosts as WWWC by Port, and 1316 hosts as WEB by Payload. In addition, the overall similarity among the results from the three classifiers cross-validates their effectiveness.

Clusters can show the typical host behaviors hidden in a single category. WEB of R-BLINC, for example, is separated into a few clusters, reflecting the different behaviors of web hosts such as server (C_2, C_3, C_4, C_{13}, C_{17}), client (C_5, C_6, C_7, C_{16}), and P2P user (C_{14}) as suggested by WWWS, WWWC, and P2P of Port, respectively. Moreover, the WWWC (web client) category of Port is clustered into a few groups, and a plausible reason for this is that there are a few typical behaviors of web clients based on the usage of web such as large-file transfer, web browsing, and ajax-based activity. Also, the MAIL category of Port shows the behaviors of only server (C_{18}), only client (C_5, C_6, C_7, C_9) or both server and client (C_{14}). This observation can also be validated by the other categories in the same cluster (e.g., P2P of Port in C_{14}).

In particular, the ability to cluster unknown data is an advantage of the unsupervised approaches. Our clustering method provides key information to profile hosts that R-BLINC classifies as UNKN[4] by separating these hosts into different categories. For example, C_3 separates 577 UNKNs of R-BLINC from the totally 5268 UNKNs of the classifier, and we can speculate that most of the 577 UNKNs are web servers as most hosts in the cluster are classified as web servers (e.g., C_3 mainly consists of 348 WEB hosts labeled by R-BLINC other than the 577 UNKN hosts). The same is true for other UNKNs of the three classifiers. Thus, the results of the classifiers and of our approach complement each other.

The effectiveness of a connection pattern-based approach can also be complementarily improved by port- and payload-based approaches. One notable example is C_1, which contains the most of UNKN hosts from R-BLINC. The port and payload classifiers both indicate that this cluster is mainly related to web server and client hosts. Actually, for the 2348 UNKN hosts in C_1, our additional inspection found that 1150

[4] We provide an example of UNKN hosts labeled by R-BLINC by examining Cluster C_4, which consists of 1283 hosts. This cluster consists of mainly WEB hosts as suggested by the three classifiers and the shape of synoptic graphlet. As mentioned above, this synoptic graphlet can be mapped with BLINC's original WEB graphlet, but this cluster contains 242 UNKN hosts classified by R-BLINC. A plausible reason of the UNKN hosts is as follows. As one of the classification rules, R-BLINC considers WEB hosts to follow "#dstPort - #dstIP $> a$", where a is one of the 28 thresholds and its value with our default setting is $a = 4$. The average and standard deviation of "#dstPort - #dstIP" are 8.12 ± 4.82 for WEB hosts of R-BLINC inside C_4 (991 hosts), and are 3.61 ± 2.39 for UNKN hosts of R-BLINC inside C_4 (242 hosts), which does not follow the above-mentioned R-BLINC's classification rule for WEB.

hosts are classified as web server or client by both the port- and payload-based classifiers; This suggests that such cross-validation would reduce the UNKN classification. Another example is that 1404 hosts out of the 1612 WEB hosts for R-BLINC in C_1 are identified as web server or client as well by both the port- and payload-based classifiers, which indicates those hosts can be considered as web-related ones with high 'plausibility.'

3) Inter-cluster distance: We examined the distribution of clusters in the feature space by using the inter-cluster distance metric: $dist(C_i, C_j)$ defined as $\frac{1}{Dim}||\mathbf{c_i} - \mathbf{c_j}||$, where c_i is the centroid vector for C_i. We define $mindist(C_i) = \min_j dist(Ci, Cj)$. The average and standard deviation of $mindist(C_i)$ is 6.63 ± 4.50, with minimum $mindist(C_1) = dist(C_1, C_3) = 0.56$ and maximum $mindist(C_{10}) = dist(C_{10}, C_{11}) = 26.7$ in the log space. This means that the clusters are not uniformly distributed. Our observation was that graphlets with low number of flows (e.g., C_1, C_3, C_5, C_4) have low $dist$ between each other, i.e., they are densely distributed yet clustered due to the high number of hosts; Whereas, high $dist$ derives from graphlets with high number of flows (e.g., $dist(C_6, C_7)$, $dist(C_{10}, C_{11})$) having similar shape but different typical number of flows.

4) Dominant features: Here we extend the discussion by evaluating which out of the $Dim = 44$ features significantly contributed to the $N = 20$ obtained clusters (Table III). For this evaluation, we use Fast Correlation-Based Filter (FCBF) [31, 23, 16], a feature ranking and selection method. We note that FCBF is used only for evaluating the relative contribution of the features to the clustering results and is not used for other parts of this work.

FCBF selects the most effective and smallest set of features with respect to symmetric uncertainty (SU) $\in [0, 1]$, which measures a form of correlation between two random variables: $SU_{X,Y} = 2\frac{H(X)-H(X|Y)}{H(X)+H(Y)}$, where $H(\cdot)$ is the information-theoretical entropy and $H(\cdot|\cdot)$ is the conditional entropy. $SU_{i,c}$ is the correlation between feature i and clusters (SU against clusters), and $SU_{i,j}$ is that between features i and j (SU against features). A higher $SU_{i,c}$ means that feature i contributes to detecting one or more clusters, whereas a higher $SU_{i,j}$ indicates that joint use of features i and j is redundant. The method first removes irrelevant features (having low $SU_{i,c}$) and then excludes redundant features (having higher $SU_{i,j}$ than $SU_{i,c}$).

Table IV lists the selected features showing their SU against clusters for MAWI and Keio data: $N = 20$ clusters for MAWI with $P = 1,000$, and $N = 16$ clusters for Keio with $P = 100$. The features selected by FCBF are mainly $o_{i:j}$ (the number of one-degree nodes), and this result suggests that this type of feature is more relevant and less redundant than the other features. Our interpretation is that $o_{i:j}$ represents well a part of the graphlet (i.e., the area between $i : j$ and $j : i$) in term of its shape (e.g., a square (parallel line(s) between columns), or a triangle (a knot on a column)) and of its number of lines (i.e., visual complexity) (e.g., one line, a few lines, or many lines). These are basic characteristics of the behavior of hosts, and the features $o_{i:j}$ represent such characteristics better than the other features used here. Figure 1 shows examples for $o_{i:j}$. Square shapes such as the area between A_5 and A_6 in Figure 1(b) occur when both the values of $o_{i:i+1}$ and $o_{i+1:i}$ are high. On the other hand, triangle shapes such as the area between A_4 and A_5 in the figure appear when one of $o_{i:i+1}$ and $o_{i+1:i}$ is quite low (e.g., zero or one). In particular, $o_{i:j}$ between srcPort and dstPort contributes significantly to the clustering (1st and 2nd ranks in Table IV). The relation between the ports represents the detailed behavior of inter-process communication, which is an important aspect of networking.

Even though other features also have discriminative power, such features are not part of the best set of features. For example, we observe that n for srcPort has $SU_{i,c} = 0.43$, and $\alpha_{i:j}$ of srcPort to dstPort has $SU_{i,c} = 0.41$ for MAWI data, indicating that these features are also useful. These features, however, were removed because of their high correlation with corresponding $o_{i:j}$ (e.g., a higher $o_{i:j}$ will be provided by a higher n_i). It means that they have similar but weaker effect on the clustering compared to $o_{i:j}$. In other words, $o_{i:j}$ is a good approximation of the shapes of graphlets. Even so, the other features are also necessary for inferring synoptic graphlets (see next section), and this is why we keep all the features.

V. SYNOPTIC GRAPHLET

According to the unsupervised procedure described in Sec. IV, graphlets associated with hosts are clustered with respect to their feature vectors. Now, as an inverse problem aiming at associating each cluster with a representative graphlet, as sketched in Figure 5, we propose a method to construct a *synoptic graphlet* from the feature vector representing a cluster.

A. Synoptic graphlet: construction

An original mapping from a feature vector into a set of bipartite graphs that constitute a graphlet is detailed here and illustrated in Figure 5. This mapping is applied to the feature vector of the cluster centroid. We will address the motivation to use synoptic graphlets instead of centroid-nearest graphlets at the end of this subsection.

Median centroid. Recalling that the feature vector of host h was defined as $\mathbf{x_h} = (x_{h,1}, \ldots, x_{h,Dim})$, let us define $\mathbf{c_k} = (c_{k,1}, \ldots, c_{k,Dim})$ as the centroid features of Cluster C_k, where the $\frac{|C_k|}{2}$-th largest value of $x_{h,i}$ among $h \in C_k$ is selected as the median feature $c_{k,i}$ [5][6].

(1) Considering a graphlet as a set of bipartite graphs. To infer a graphlet from the centroid features of a cluster, we construct a graphlet as a set of bipartite graphs. A_1 and A_2 are a disjoint set of a bipartite graph, A_2 and A_3 are another, and so on. In other words, we break down the

[5] As an example, for n_2, if a cluster contains 100 hosts, the 50th largest value in n_2 is chosen as the median (x_i and x_j ($i \neq j$) do not necessarily derive from the same host).

[6] We note that statistics other than the median could be chosen as a representative. We also tried to use average as representative, but average is not robust to outlier features, and more critically taking the averages lead to decimal values, which are difficult to deal with for graph rewiring. The Dim-dimensional median features are converted from a log scale into a linear scale by inverting the normalization function defined in Sec. IV-A2.

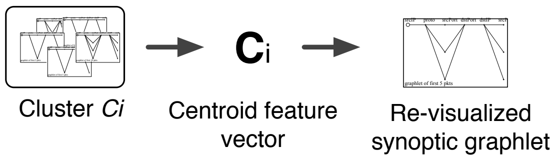


Fig. 5. Synoptic graphlet. Graphlets obtained from hosts are clustered. In turn, each cluster is associated with a representative a posteriori synoptic graphlet. The second row of Table III displays the synoptic graphlets re-visualized from the actual clusters of hosts.

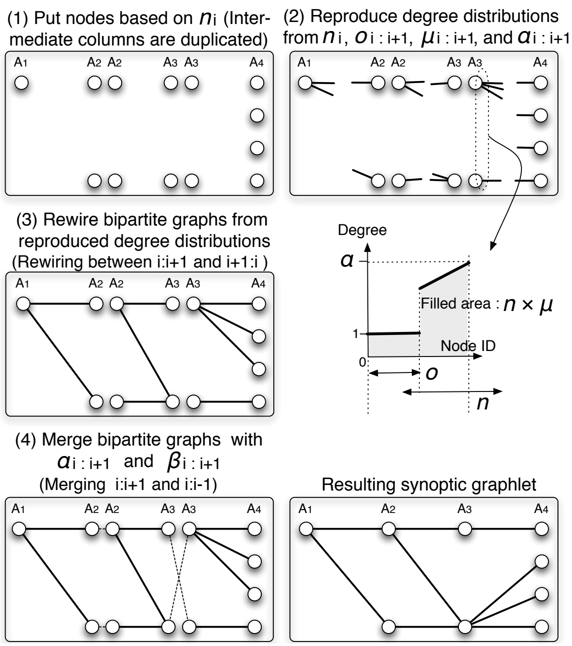


Fig. 6. Procedure of re-visualizing synoptic graphlets. A synoptic graphlet of a cluster is reproduced from the graphlet features of the cluster centroid. Graphlet features are defined in Sec. IV-A1 and Figure 3.

graphlet reproduction problem into (a) reproducing the degree distributions of each bipartite graph, (b) rewiring each bipartite graph based on the degree distributions, and (c) merging neighboring bipartite graphs.

(2) Reproducing degree distributions. From a feature vector, we build the degree distribution of direction $i : j$ ($j = i + 1$ or $i - 1$), denoted as $\hat{D}_{i:j} = (d_1, \ldots, d_n)$ where n is the total number of nodes as defined in Sec. IV-A1 ("$i : j$" is omitted from $d_{k,n_{i:j}}$ and $n_{i:j}$ for brevity). We first consider the one-degree nodes as follows: $d_n = d_{n-1} = d_{n-o+1} = 1$. If all the nodes have degree of one (i.e., $n = o$), this procedure ends; Otherwise we rebuild the remaining part of the degree distribution. We define the number of remaining nodes ζ and the remaining degrees ξ as $\zeta = n - o$ and $\xi = \mu \times n - 1 \times o$. The degrees are estimated as follows: $d_1 = \alpha, d_2 = \alpha - \Delta, \ldots, d_\zeta = \alpha - (\zeta - 1) \times \Delta$, where $\Delta = \frac{2}{\zeta-1}\ (\alpha - \frac{\xi}{\zeta})$, which satisfies $\xi = d_1 + \ldots + d_\zeta$. This process to distribute the remaining degrees to the remaining nodes is based on the usual appearances of graphlets (e.g., some 'knot' nodes, only one, etc.).

(3) Rewiring bipartite graphs. A bipartite graph is generated from $\hat{D}_{i:i+1}$ and $\hat{D}_{i+1:i}$ computed above. Nodes of higher degrees of A_i are connected with those of lower degrees of A_{i+1}, which reflects an emprical traffic characteristics (one-to-many connection rather than two-to-many). An example of this characteristics is server-client behavior, where (a) a source port is connected with several destination hosts and also (b) a destination host is associated with a set of several destination ports, which are not related to other hosts. By defining $i : i+1$ as r (right) and $i+1 : i$ as l (left), we connect $v_{1,r}$ with $v_{n_l,l}, \ldots, v_{(n_l-d_{1,r}-1),l}$, and then connect $v_{2,r}$ with $v_{k,l}, \ldots, v_{(k-d_{2,r}-1),l}$, where k is the largest label of nodes that have degree remaining after the previous connections. We iterate this connection procedure until $v_{n_r,r}$ is dealt with and consequently obtain a bipartite graph.

(4) Merging bipartite graphs into a synoptic graphlet. A synoptic graphlet is then drawn by combining each pair of neighboring bipartite graphs. We additionally define the direction: $i : i+1$ as f (forward) and $i : i-1$ as b (backward). The two directions have different degree distributions with the same number of nodes: $\hat{D}_f$ and $\hat{D}_b$, and a pair $(d_{k,f}, d_{l,b})$ is merged into a node $v_{m,i}$, where k, l, and m are determined as follows. We first compute the degree correlation between $\hat{D}_f$ and $\hat{D}_b$, which we define as $\gamma = (\alpha_f - \alpha_b) \times (\beta_f - \beta_b)$, with $\alpha_{i:j}$ and $\beta_{i:j}$ of the centroid features. If the correlation is positive ($\gamma \geq 0$), we combine the nodes in the same order of degree value: $v_{1,i} = (d_{1,f}, d_{1,b}), \ldots, v_{n,i} = (d_{n,f}, d_{n,b})$. Conversely, for $\gamma < 0$, the combination order is reversed: $v_{1,i} = (d_{1,f}, d_{n,b}), \ldots, v_{n,i} = (d_{n,f}, d_{1,b})$.

Synoptic versus centroid graphlets. Instead of synoptic graphlets, centroids may have been selected as cluster representative. For clusters with very large number of flows, both choices likely yield close representatives, however, centroids suffer from a number of disadvantages: (i) Centroid graphlets may show a very large variability (hence lacking robustness) for clusters with small number of flows, while synoptic graphlets are less dependent on the actual number of flow per host, because it is regenerated from all the representative features of a cluster; (ii) Centroid graphlets not necessarily result into the typical representative of the cluster. The centroid may occasionally correspond to a specific behavior, even when many of its Dim features are close to the median, to the contrary of synoptic graphlets that somehow make the visualization/interpretation step independent from the classification phase (in a semi-supervised spirit) [7]; Therefore synoptic graphlets should be more effective tools to represent what actually happens in the feature space and thus to profile and interpret host behaviors. More detailed comparisons between centroid and synoptic graphlets are beyond the scope of the

[7]We briefly compared the synoptic graphlet and centroid-nearest graphlet for each cluster. Approximately 80% of clusters produced intuitively similar shapes of the two kinds of graphlets; This is plausibly due to the well-tuned threshold θ and enough number of hosts inside a cluster. Such correspondence between the two kinds implicitly validates the overall procedure of rewiring graphlets. We also found the differences in shapes of the two kinds. For example, in Cluster C_2, there were differences in the #nodes in the dstIP column between the corresponding synoptic graphlet and centroid-derived graphlet; Indeed, the collapse in the shape of the synoptic graphlet (Table III) indicates that this graphlet does not represent per-host behavior well but rather represents an aggregated view. We manually inspected the composition of C_2 and found that this cluster contained two types of typical host behaviors. This graphlet suggests that it would be meaningful to further separate C_2 into different clusters.

present contribution and will be discussed elsewhere.

B. Synoptic graphlet: interpretation

The second row of the column headings in Table III shows the synoptic graphlets, re-visualized from the $N = 20$ clusters presented in Sec. IV-B (larger versions are displayed in Figure 9).

Effectiveness of synoptic graphlets. One of the advantages of synoptic graphlets is the ability to construct an intuitive understanding of clustering results. The "complexity" of the shapes of synoptic graphlets meaningfully represents the intensity of flows. For example, a graphlet of many lines is derived from the use of many flows, indicating that the corresponding host uses an application for many peers and/or many ports (e.g., DNS and MAIL are the categories of the many-lines graphlets such as C_{15}). In addition, the number of nodes for each column A_i is also meaningful. For instance, if A_3 (srcPort) has only a few nodes, then the corresponding host can be speculated to be a server (e.g., C_3 is mainly labeled as WWWS by Port).

BLINC models validity. Most of the synoptic graphlets in Table III correspond to most of the BLINC graphlets[8] (listed in [15]), and thus our result validates the intuitions behind the BLINC series. An exception, though, is pointed out by C_{11}; Most hosts are identified as UNKN by R-BLINC, whereas they are mainly identified as FLOOD by Port (probably because of a large amount of SYN packets and few targeted hosts). On the other hand, some clusters having similar shape of synoptic graphlets consist of similar breakdown such as C_{17} and C_{18}. As implied by the different number of lines in the shapes of synoptic graphlets for the two clusters (Table III), this result indicates two typical number of flows of graphlets, which might not easily be found by applying untuned heuristic rules.

One-flow graphlets. C_1 represents synoptic graphlets composed of one flow (4594 in total – about 25% among the analyzed hosts), and the three classifiers unfortunately identify many of them as UNKN. This kind of isolated communication has been observed in prior studies [12, 10, 13] as well. Although one-flow graphlets are classified into various application categories as the three classifiers point out, the one-line shape itself reveals the important information that $P = 1000$ packets from a single host constitute only one flow. In other words, a one-flow graphlet possibly implies large file transfer, because we do not observe any control flows or the other flows. This plausible interpretation is supported by the finding that many of these hosts identified by the three classifiers are web or P2P users, which are occasionally used for host-to-host large-file transfer in some cases.

[8]For example, the synoptic graphlet of C_4 can be mapped with WEB graphlet (shown in Figure 5(d) of the original paper), because the two graphlets commonly represent one srcPort, and several dstPort, and a few dstIP nodes. The relevance of this mapping is supported by the fact that R-BLINC classifies most of the hosts in C_4 as related to WEB. For another example, the synoptic graphlet of C_8 can be related to DNS graphlet (shown in Figure 5(g) of the original), because the two graphlets commonly represent many srcPort, and one dstPort, and a few dstIP nodes. The original graphlet represents both client-side and server-side behavior in a single figure, yet the graphlet of C_8 can be mapped with client-side one. The relevance of this mapping is supported by the fact that R-BLINC classifies most of the hosts in C_8 as related to DNS.

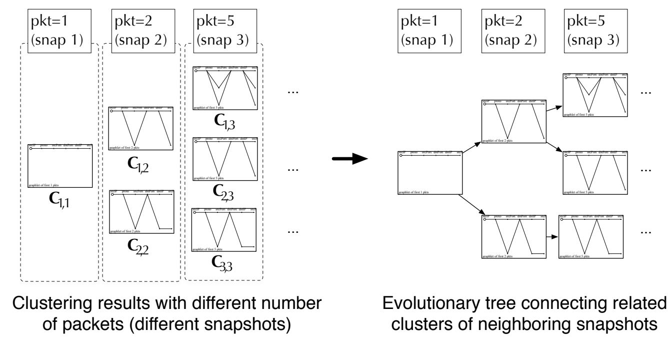


Fig. 7. Creation of evolutionary tree.

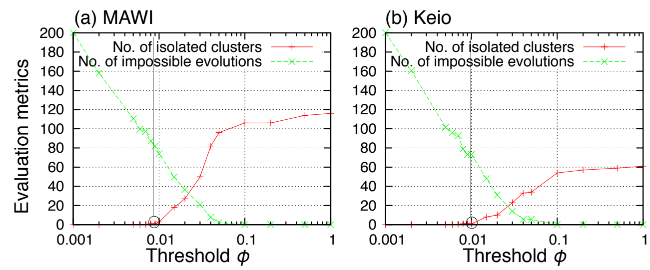


Fig. 8. Characteristics of the threshold for evolutionary tree ϕ.

In summary, synoptic graphlets are effective for an intuitive and visual understanding of the clustering output, and the comparison result indicates the relevance of the overall idea of BLINC, while alleviating the difficulty of manually setting appropriate rules and parameters.

VI. Evolutionary nature of host-level traffic

Let us further discuss the effectiveness of the new method by introducing *evolutionary tree* of synoptic graphlets, which provides a way to understand the evolution of information about host behaviors when the number P of analyzed packets increases. To achieve this, we analyze the same set of $H = 20,000$ hosts by changing the value of P. This tree can also answer the question "how many packets P do we need to find all typical patterns?" and "how accurately hosts can be profiled with a given P?."

A. Evolutionary tree: creation

Snapshot. The next key question in the assessment of synoptic graphlets is raised by the choice of the number P of packets that need to be involved in graphlet construction to find all typical patterns and thus permit accurate host profiling. This is addressed via the concept of synoptic graphlet *evolutionary tree* that characterizes host behavior profiling evolution when P increases. For example, a single packet (thus a single flow) produces a single-line graphlet, whereas two packets may result either in a single line if they belong to the same flow or in two lines sharing nodes and edges if they share common attributes. Any graphlet may hence evolve from an identical single-line shape towards a complex pattern as P increases. An evolutionary tree is thus obtained from combining different *snapshots* s, i.e., graphlets obtained from different values of P_s [9]. For MAWI data, $P = 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000$ for snapshots $s = 1, \ldots, 10$; for Keio data, $P = 1, 2, 5, 10, 20, 50, 100$ for $s = 1, \ldots, 7$.

Tree creation. Let C_{s,N_s} denote the set of N_s clusters obtained at snapshot s (i.e., from P_s packets). For each s, C_{s,N_s} is obtained with a value of the sole threshold θ that remains constant and does not depend on P_s. Thus, θ serves as distance basis in the feature space, and hence does not determine a priori the number of clusters, which permits to compare clustering outputs obtained with different P. The evolutionary tree is created from a single criteria, relying on a threshold ϕ: if the number of hosts in $C_{s,i} \cap C_{s+1,j}$ is larger than $\phi \times H$ (H being the total number of analyzed hosts), the two clusters $C_{s,i}$ and $C_{s+1,j}$ are connected by an edge, which materializes that the typical behavior $C_{s,i}$ at snapshot s tends to evolve into $C_{s+1,j}$ at $s+1$. Finally, an evolutionary tree provides an intuitive overview of the behavioral growth of hosts (cf. e.g., Figure 9).

Threshold. Setting the threshold ϕ, which determines whether neighboring clusters are connected or not, results from the following trade-off: Too high ϕ may yield 'isolated' clusters, not connected to any other clusters on any neighboring snapshot; Too low ϕ may yield many 'impossible' evolutions in graphlet shapes. For example, for some synoptic graphlets, α might be reduced from s to $s+1$ because of the changes in the set of hosts within a cluster, despite the fact that this never occurs in the evolution of the graphlet of a single host. Therefore, the connection between Cluster i at snapshot s and Cluster j at $s+1$ is declared *impossible*, if either of parameters n, μ, and α is reduced. Figure 8 illustrates the trade-off, plotting the number of isolated clusters and that of impossible evolutions as a function of ϕ. Empirically, the threshold is set to $\phi = 0.0077$ (i.e., about 150 hosts) for MAWI data, and to $\phi = 0.0070$ (i.e., about 70 hosts) for Keio data, which maintain no isolated cluster and a low number of impossible evolutions.

B. Evolutionary tree: interpretation

1) Intuition from evolutionary tree – visual analysis: **Global view.** Figure 9 depicts the resulting evolutionary tree for MAWI data ($H = 20,000$, $\theta = 500$, cf. Sec. IV-B1, $\phi = 0.77\%$, cf. Sec. VI-A). Synoptic graphlets at snapshot s are shown in the s-th column, and related synoptic graphlets (from successive snapshots) are linked with arrows. The synoptic graphlets at $s = 10$ correspond to the evaluations presented in Secs. IV-B and V.

Figure 9 thus provides an intuitive and comprehensive overview of the evolution of typical host behaviors, from $P = 1$ (origin of graphlets) to large P, permitting interpretation of graphlet changes with P. Interestingly, clusters do not only separate but also merge as P increases. This suggests that there exist different evolution footprints, even when hosts are clustered into a same group at a given snapshot. Evolutionary trees thus enhance the profiling by providing richer information.

[9]It should also be interesting to analyze this by increasing the number of flows per host (instead of increasing the number of packets P). However, we have to elaborate on the appropriate way to deal with hosts with low number of flows (e.g., 25% of hosts have only one flow).

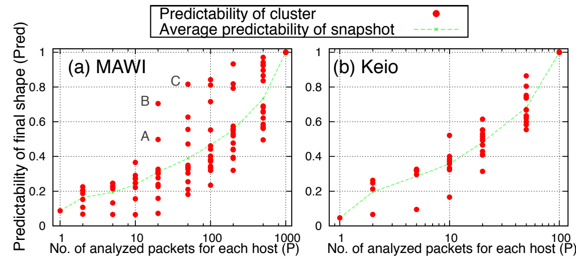


Fig. 10. Predictability of evolution as a function of P.

Early stages. For $P = 1$, by nature, there is only one-flow graphlets. For $P = 2$, although there are theoretically $2^4 = 16$ possible graphlets (combination of four attributes: proto, srcPort, dstPort, and dstIP), only 7 are actually observed. Although some graphlets are actually different from the seven synoptic graphlets and have different transitions, these are not typical, and hence do not appear in the figure. Such minor graphlets could be found by finer-grained clustering, with lower θ.

Late stages. The final forms of graphlets become apparent in the late stages. For example, one-flow graphlet A is destined to mostly remain one-flow, after $P = 20$, as indicated by the abrupt increase in predictability discussed in Sec. VI-B2. Other examples are provided by synoptic graphlets B and C, prominent at $P = 20$ and 50, respectively. They are mainly related to scanning activities, which thus indicates that $P = 20$ is large enough to permit separation of scanners from other activities. As a whole, the total number of clusters at $P = 1000$ remains quasi-unchanged compared to that at $P = 100$. Thus, $P = 100$ can be considered as the reference number of packets required for accurately discovering typical host behaviors. Also, this result implies that $P = 100$ provides some longitudinal stationarity of aggregated view of host behaviors. The bottom row in Table III lists each stage at which each Cluster $C_{S,i}$ stops its evolution along the tree (i.e., becomes predictable).

Keio data case. Similar results were obtained for Keio data, but for the fact that one-flow graphlets continue to evolve at $P = 50$. We interpret that the stagnation of one-flow graphlets for MAWI could stem from the partial view of the traffic, measured at the backbone link, whereas Keio traffic is measured at an edge router.

2) Predictability in evolution – quantitative analysis: To complement the understanding of synoptic graphlet evolution, the evolution predictability of a given host in the tree is quantified. Let us define $P(C_{s_2,j}|C_{s_1,i}) = \frac{|C_{s_2,j} \cap C_{s_1,i}|}{|C_{s_1,i}|}$, which measures the probability that hosts in Cluster i at snapshot s_1 ($C_{s_1,i}$) evolves into Cluster j at s_2 ($C_{s_2,j}$). We define the predictability of Cluster $C_{s,i}$ as $Pred(C_{s,i}) = 1 + \frac{1}{\log_{10} N_S} \sum_{j=1}^{N_S} P(C_{S,j}|C_{s,i}) \times \log_{10} P(C_{S,j}|C_{s,i})$, where S is the final snapshot and N_S the corresponding number of clusters. $Pred(C_{s,i})$ is hence a normalized entropy that

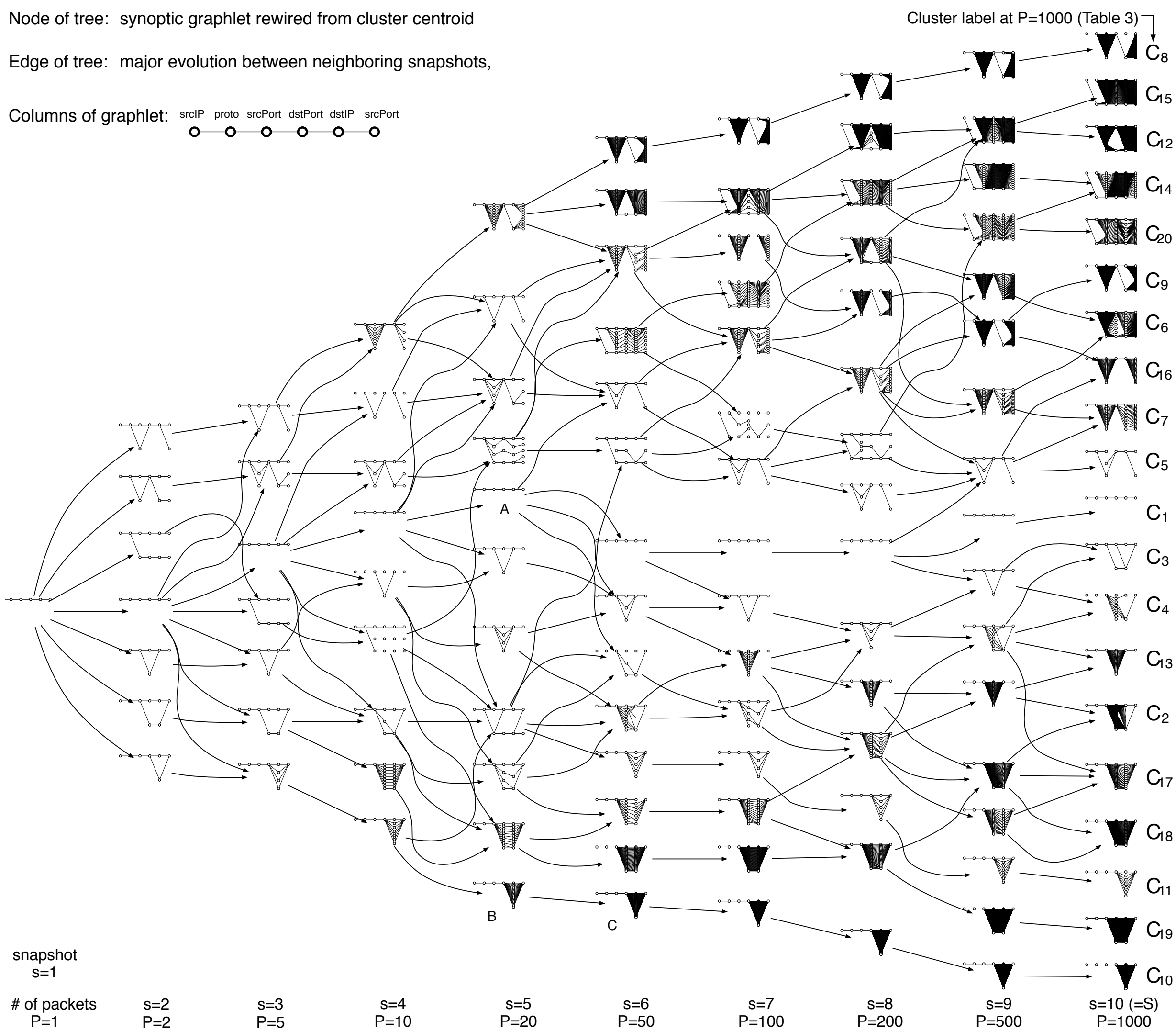


Fig. 9. Evolutionary tree of synoptic graphlets as a function of P (or s).

characterizes the dispersion of transition probabilities. Thus, if $C_{s,i}$ grows only to $C_{S,1}$ then $Pred(C_{s,i}) = 1$, whereas if $C_{s,i}$ can evolve into any future shapes with equal probability then $Pred(C_{s,i}) = 0$. Note that this predictability is computed considering all possible evolutions (i.e., $\phi = 0$).

Figure 10 displays the predictabilities of all clusters $Pred(C_{s,i})$ as a function of P (or s) for MAWI and Keio. Each dot stands for a synoptic graphlet (i.e., a cluster), for a given snapshot. The dashed line represents the transition in the average predictability and shows that the predictability is approximately linear with $\log P$ (Pearson's correlation coefficient is 0.95). The predictability at $P = 1$ is almost 0, which suggests that the corresponding origin of a graphlet can evolve into any final graphlet. Conversely, this predictability becomes higher with higher P. In addition, predictabilities for some $C_{s,i}$ abruptly become higher than for others, which indicates the end of the evolution for that synoptic graphlet, as shown by Points A and B at $P = 20$ and C at $P = 50$, (that correspond to synoptic graphlets A, B, and C in Figure 9). The high predictability value for these synoptic graphlets at low snapshots confirms the observation made from the evolutionary tree that the future of these graphlets is early set and hence that can be easily distinguished with fewer packets than other types of graphlets.

VII. Discussion

Revisiting BLINC. The results presented in Secs. IV-B and V validate the concepts at work in BLINC, as most of the auto-generated synoptic graphlets can be related to empirically defined BLINC graphlet models [15]. However, such heuristic model-based approaches face the potential difficulties in (a) designing appropriate rules as indicated by the observed unknown clusters and in (b) determining the relevant values of thresholds for accurate classification as partially implied by a prior work [16], which conducted a number of trials to determine appropriate parameters. Instead, unsupervised approaches can potentially uncover new types of applications with the tuning of only a very limited number of threshold levels. In addition, an advantage of our approach should be to avoid the assumption that the traffic of one host should be mostly explained by a single application[10].

Traffic characteristic evolution when increasing the number of analyzed packets. Sec. VI showed that the method requires around 100 packets to classify hosts. This is larger than the findings of a few previous works. For instance, the work reported in [1] showed that major TCP flows can be identified on bi-directional links from their size and direction, by examining only the first four or five packets (after the handshake) in a connection. Other works [25, 8, 20] also claimed such an ability. The present work, however, deals with more general assumptions about traffic: uni-directional links, legitimate as well as anomalous and unknown traffic, a few protocols besides TCP, not certainty of observing the first packets of flows. In this context, the need to collect a larger amount of information to predict traffic characteristics does not come as a surprise. Moreover, our work is to profile hosts, not only identifying the application in a TCP connection.

Limitations. (a) The degree-based features used here do not include relations among non-neighboring columns such as A_1 and A_3. (b) In addition, real graphlets are not as clean as rewired ones, because they include packets unrelated to the main behaviors of the hosts. Features could be weighted to remove such noise, e.g., the width of edges and the radius of nodes could be set based on the number of packets. (c) In some cases, host behaviors may result from two dominant kinds of applications, e.g., a host serving both mail and DNS, or a NAT gateway with a web client and a P2P user. Such a host cannot easily be profiled. (d) In general, synoptic graphlets only provide shape information; Although such information provides meaningful insight into host behaviors as shown throughout this work, it is still difficult to identify the exact application names used by hosts. If we want to identify them, it would be helpful to put port numbers in the graphlet figure or to cross-compare with classifiers based on port numbers, payloads, IP addresses [28], packet sizes [1, 8], and so on.

Application to supervised approaches. A potential application is to create a reliable dataset of known flows, that then an approach like that of Iliofotou et al. [11] could use. This is because our experiment found graphlets corresponding to a single application (say C_{13} in table III); Such graphlets could be known signatures for any supervised methods. This approach is better than just using a signature generator (say a payload-based classifier), as any classifier will have some misclassification; The clustering scheme presented here can group highly inter-related flows that can be characterized as learning data of enhanced reliability.

Application to unsupervised approaches. Another use case of our method is to help researchers (or network administrators) to interpret the results of unsupervised clustering over graphlets. In general, interpretation of resulting clusters should require to examine a lot of numerical features ($\mathbf{x_h}$), as a prior work [6] does, which becomes significantly difficult as the dimension increases. On the other hand, the use of synoptic graphlet supports such interpretation by converting those features in a single intuitive figure. For example, if a synoptic graphlet for Cluster C contains only a single node in the column for srcPort and the node has several edges, one can easily interpret that C is mostly composed of server hosts (similarly, that for dstPort implies that C is related to client hosts). With such assistance in interpretation, operators will efficiently notice and understand the emergence of new types of application usages (e.g., malicious hosts, P2P software users, or rapid increase in web clients) appearing as new clusters in the monitored link.

VIII. Concluding remarks

The main issue of the present work was the trade-off in choosing between supervised and unsupervised approaches to end-host profiling. The former is comprehensive but is blind to undefined classes, while the latter can uncover unknown pattens of behavior at the sacrifice of interpretability. We aimed to bridge the gap between the two in the present work. The proposed method was designed to perform unsupervised clustering for finding undefined classes and to re-visualize the resulting clusters as synoptic graphlets for providing interpretability. We compared the method against a graphlet-based state-of-the-art classifier (BLINC) as well as against a classical port-based inspector and a payload-based one, by applying these methods to two sets of actual traffic traces measured at different locations. The proposed method spontaneously generated synoptic graphlets that are typical in their shape, which validates the graphlet models heuristically pre-defined in earlier works. Also, for methodological study of the improvements brought to host profiling, this work demonstrated how to extend beyond a simple classification to the production of an evolutionary tree by increasing the number of observed packets per host. The entire procedure requires only a few threshold to be tuned while the state-of-the-art method needs many. The new achievements in this contribution are as follows: (a) an unsupervised clustering applied to graphlet shape-based characteristics, which is further significantly extended to (b) a visualization-oriented auto-enumeration of typical host behaviors generated from actual data, successfully resulting in validating the relevance of past works, and (c) an analysis on evolutionary characteristics of the growth of host behaviors both in visual and quantitative manners, which is useful in understanding the evolutionary nature of host behaviors.

Acknowledgement. We thank Young J. Won (IIJ), Guillaume Dewaele (ENS de Lyon), and Romain Fontugne (Sokendai/NII) for their invaluable advice. We also thank Hyunchul Kim (Seoul National University), Thomas Karagiannis (Microsoft Research), and Dhiman Barman (Juniper Networks) for providing access to the payload-based classifier and BLINC.

[10]To show the non-negligible amount of application mixture of a host, we quantify the degree of this for a host h as $p_{max}(h) = \max_a \frac{\#flow(h,a)}{\#flow(h)}$, where a is an application (except for UNKN), $\#flow(h)$ is the total number of h's flows identified as a certain application (except for UNKN) by the payload classifier, $\#flow(h,a)$ is the number of h's flows identified as application a by the classifier. In other words, $p_{max}(h)$ is the fraction of most dominant application in terms of #flows. For the result for $H = 20,000$ hosts in the 12 MAWI traces, we found that bottom 10% of hosts have $p_{max}(h) < 75\%$, bottom 20% have $p_{max}(h) < 95\%$, and bottom 25% have $p_{max}(h) < 99\%$ (i.e., remaining 75% of hosts are mostly characterized by a single application).

References

[1] L. Bernaille, R. Teixeira, and K. Salamatian. Early Application Identification. *ACM CoNEXT 2006*, p. 12, 2006.

[2] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho. Seven Years and One Day: Sketching the Evolution of Internet Traffic. *IEEE INFOCOM 2009*, pp. 711–719, 2009.

[3] K. Cho, K. Mitsuya, and A. Kato. Traffic Data Reposity at the WIDE Project. *USENIX 2000 FREENIX Track*, p. 8, 2000.

[4] CAIDA The CoralReef Project. http://www.caida.org/tools/measurement/coralreef/.

[5] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho. Extracting Hidden Anomalies using Sketch and Non Gaussian Multiresolution Statistical Detection Procedure. *ACM SIGCOMM LSAD'07*, pp. 145–152, 2007.

[6] G. Dewaele, Y. Himura, P. Borgnat, K. Fukuda, P. Abry, O. Michel, R. Fontugne, K. Cho, and H. Esaki. Unsupervised host behavior classification from connection patterns. *International Journal of Network Management, Vol.10 Issue 5*, pp. 317–337, 2010.

[7] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification Using Clustering Algorithms. *ACM SIGCOMM'06 MINENET*, pp. 281–286, 2006.

[8] V. C. Espanol, P. B. Ros, M. S. Simó, A. Dainotti, W. D. Donato, and A. Pescapé. K-dimensional trees for continuous traffic classification. *TMA 2010*, p. 14, 2010.

[9] Y. Himura, K. Fukuda, P. Abry, K. Cho, and H. Esaki. Characterization of Host-Level Application Traffic with Multi-Scale Gamma Model. *IEICE Transactions on Communications*, E93-B(11):3048–3057, 2010.

[10] M. Iliofotou, M. Faloutsos, and M. Mitzenmacher. Exploiting Dynamicity in Graph-based Traffic Analysis: Techniques and Applications. *ACM CoNEXT 2009*, pp. 241–252, 2009.

[11] M. Iliofotou, B. Gallagher, T. E.-Rad, G. Xie, and M. V. Faloutsos. Profiling-by-Association: A Resilient Traffic Profiling Solution for the Internet Backbone. *ACM CoNEXT 2010*, p. 12, 2010.

[12] M. Iliofotou, H. C. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese. Graph-based P2P Traffic Classification at the Internet Backbone. *IEEE Global Internet Symposium 2009*, p. 6, 2009.

[13] Y. Jin, E. Sharafuddin, and Z. L. Zhang. Unveiling Core Network-Wide Communication Patterns through Application Traffic Activity Graph Decomposition. *ACM SIGMETRICS'09*, pp. 49–60, 2009.

[14] T. Karagiannis, K. Papagianakki, N. Taft, and M. Faloutsos. Profiling the End Host. *PAM 2007*, pp. 186–196, 2007.

[15] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. *ACM SIGCOMM'05*, pp. 229–240, 2005.

[16] H. Kim, kc claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Y. Lee. Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices. *ACM CoNEXT 2008*, p. 12, 2008.

[17] l7-filter. http://l7-filter.sourceforge.net.

[18] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. *ACM SIGCOMM'05*, pp. 217–228, 2005.

[19] S. Lee, H. Kim, D. Barman, S. Lee, C. Kim, and T. T. Kwon. NeTraMark: A Network Traffic Classification Benchmark. *ACM SIGCOMM CCR Vol.41 No.1*, pp. 23–30, 2010.

[20] Y. Lim, H. Kim, J. Jeong, C. Kim, T. T. Kwon, and Y. Choi. Internet Traffic Classification Demystified: On the Sources of the Discriminative Power. *ACM CoNEXT 2010*, p. 12, 2010.

[21] MAWI Working Group Traffic Archive. http://mawi.wide.ad.jp/mawi/.

[22] J. McHugh, R. McLeod, and V. Nagaonkar. Passive network forensics: behavioural classification of network hosts based on connection patterns. *ACM SIGOPS Operating Systems Review, Vol.42, No.3*, pp. 99–111, 2008.

[23] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. *ACM SIGMETRICS'05*, pp. 50–60, 2005.

[24] OpenDPI. http://opendpi.org/.

[25] M. Pietrzyk, J. L. Costeux, G. Urvoy-Keller, and T. En-Najjary. Challenging statistical classification for operational usage: the ADSL case. *ACM IMC'09*, pp. 122–135, 2009.

[26] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. *USENIX LISA'99*, pp. 229–238, 1999.

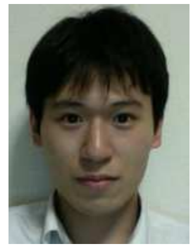
[27] G. Tan, M. Poletto, J. Guttag, and F. Kaashoek. Role Classification of Hosts within Enterprise Networks Based on Connection Patterns. *2003 USENIX Annual Technical Conference*, pp. 15–28, 2003.

[28] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci. Unconstrained Endpoint Profiling (Googling the Internet). *ACM SIGCOMM'08*, pp. 279–290, 2008.

[29] K. Xu, F. Wang, and L. Gu. Network-Aware Behavior Clustering of Internet End Hosts. *IEEE INFOCOM 2011*, pp. 2078–2086, 2011.

[30] K. Xu, Z. L. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. *ACM SIGCOMM'05*, pp. 169–180, 2005.

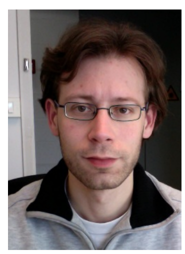
[31] L. Yu and H. Liu. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. *International Conference on Machine Leaning (ICML-03)*, pp. 856–863, 2003.



Yosuke Himura is a Master course student in Department of Information and Communication Engineering, Graduate School of Information Science and Technology, the University of Tokyo. His research interests include statistical analysis of Internet traffic.

Kensuke Fukuda is an associate professor at the National Institute of Informatics (NII) and is a researcher, PRESTO, JST. He received his Ph.D degree in computer science from Keio University in 1999. He worked in NTT laboratories from 1999 to 2005, and joined NII in 2006. His current research interests are Internet traffic measurement and analysis, intelligent network control architectures, and the scientific aspects of networks. In 2002, he was a visiting scholar at Boston University.

Kenjiro Cho is Deputy Research Director at Internet Initiative Japan, Inc. He is also an adjunct professor at Keio University and Japan Advanced Institute of Science and Technology, and a board member of the WIDE project. He received the B.S. degree in electronic engineering from Kobe University, the M.Eng. degree in computer science from Cornell University, and the Ph.D. degree in media and governance from Keio University. His current research interests include traffic measurement and management, and operating system support for networking.



Pierre Borgnat received the Professeur-Agrégé de Sciences Physiques degree in 97, a Ms. Sc. in Physics in 99 and defended a Ph.D. degree in Physics and Signal Processing in 2002. In 2003-2004, he spent one year in the Signal and Image Processing group of the IRS, IST (Lisbon, Portugal). Since October 2004, he has been a full-time CNRS researcher with the Laboratoire de Physique, ENS de Lyon. His research interests are in statistical signal processing of non-stationary processes and scaling phenomena for complex systems. He is also working on Internet traffic measurements and modeling, and in analysis and modeling of dynamical complex networks.

Patrice Abry received the degree of Professeur-Agrégé in Physics, in 1989 and completed a Ph.D in Physics and Signal Processing, at Ecole Normale Supérieure de Lyon and Université Lyon I, in 1994. He received the AFCET-MESR-CNRS prize for best Ph.D in Signal Processing for the years 93-94. He is a CNRS researcher since 1995 and became CNRS Research Director in 2005 and IEEE Fellow in 2012. His research interests and include wavelet-based analysis and modeling of scale invariance phenomena and related topics (self-similarity and multifractal) and range from theory to applications.

Hiroshi Esaki received Ph.D from the University of Tokyo, Japan, in 1998. In 1987, he joined Research and Development Center, Toshiba Corporation. From 1990 to 1991, he has been at Applied Research Laboratory of Bellcore Inc., New Jersey, as a residential researcher. From 1994 to 1996, he has been at Center for Telecommunication Research of Columbia University in New York. From 1998, he has served as a professor at the University of Tokyo, and as a board member of WIDE Project. Currently, he is executive director of IPv6 promotion council, vice president of JPNIC, IPv6 Forum Fellow, board member of WIDE Project and Board of Trustee of Internet Society.