INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2014; **24**: 289–303
Published online 5 June 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nem.1865

Network application profiling with traffic causality graphs

Hirochika Asai,[1,*,†] Kensuke Fukuda,[2] Patrice Abry,[3] Pierre Borgnat[3] and Hiroshi Esaki[1]

[1]*Graduate School of Information Science and Technology, University of Tokyo, Japan*
[2]*NII, Tokyo, Japan*
[3]*CNRS, Ecole Normale Supérieure de Lyon, Lyon, France*

SUMMARY

A network application profiling framework is proposed that is based on traffic causality graphs (TCGs), representing temporal and spatial causality of flows to identify application programs. The proposed framework consists of three modules: the feature vector space construction using discriminative patterns extracted from TCGs by a graph-mining algorithm; a feature vector supervised learning procedure in the constructed vector space; and an application identification program using a similarity measure in the feature vector space. Accuracy of the proposed framework for application identification is evaluated, making use of ground truth packet traces from seven peer-to-peer (P2P) application programs. It is demonstrated that this framework achieves an overall 90.0% accuracy in application identification. Contributions are twofold: (1) using a graph-mining algorithm, the proposed framework enables automatic extraction of discriminative patterns serving as identification features; 2) high accuracy in application identification is achieved, notably for P2P applications that are more difficult to identify because of their using random ports and potential communication encryption. Copyright © 2014 John Wiley & Sons, Ltd

Received 31 October 2013; Revised 26 March 2014; Accepted 10 April 2014

KEY WORDS: Network application profiling; Identification; Traffic causality graph; Graph mining

1. INTRODUCTION

The Internet has become a fundamental infrastructure for modern society and industry in the last two decades, and a large variety of applications and services have been developed and deployed for it. In this context, traffic classification is one of the key technologies for IP network management tasks, such as analysis of security incidents, anomaly detection and traffic engineering. The simplest traffic classification method is based on the source and destination port numbers of the transport layer (i.e. TCP and UDP) [1]. However, one of the problems with port-based methods is that port numbers are not statically bound to an application. For example, some network applications can use non-standard ports, especially under firewall port restrictions. Moreover, some network applications such as peer-to-peer (P2P) applications may use a random port to pass firewall port restrictions and to obfuscate detection. These cases significantly impair traffic classification based on port numbers.

Numerous advanced techniques not relying on the sole port numbers have been proposed for profiling network application traffic. Signature-based traffic classifiers [2–8] identify applications from network traffic by inspecting packet payloads (i.e. application data). However, it is difficult to apply it to encrypted data, and packet inspection raises privacy concerns. To solve these encryption and privacy problems, statistical approaches [9–11] have been proposed to classify applications from network traffic. These approaches use statistical properties of applications, such as

[*]Correspondence to: Hirochika Asai, Graduate School of Information Science and Technology, University of Tokyo, Japan.
[†]E-mail: panda@hongo.wide.ad.jp

the probability distribution of packet inter-arrival time and of packet size, instead of packet payload inspection. These properties are useful for detecting anomalies in network flows, and consequently they have also been used in anomaly detection methods [12]. An intrinsic approach [13] not relying on signatures or statistical properties checks IP addresses in flows and web content found in search engine results corresponding to an IP address to profile end-hosts. However, as mentioned by authors, it cannot profile end-hosts using P2P applications, and applying it to application profiling is difficult because end-hosts, especially end-user hosts, involve multiple applications. Other approaches [14–19] use information on spatial interactions between hosts or flows for traffic classification. However, these approaches cannot profile application programs such as user agents of web browsers/proxies without payload inspection, though they might succeed in profiling certain application classes, such as the web browsing and P2P file-sharing classes. Traffic causality graphs (TCGs) that represent temporal and spatial causality of flows have been proposed to profile network application programs without looking at packet payload [20]. we demonstrated previously [20] that TCGs were applicable to network application program profiling with case studies. However, automated profiling with TCGs has not been accomplished because the features used were empirically discovered and selected and they were too simple and not sufficient to permit systematic application identification.

In this article, a framework is proposed to achieve automated network application profiling by extending the TCG-based profiling approach. After having reviewed related works (Section 2), we leverage a graph-mining algorithm to extract discriminative patterns (also referred to as substructures) from TCGs and to construct a feature vector space for profiling. We also make use of a similarity measure between two TCG feature vectors in the feature vector space to identify applications (Section 3). The accuracy of the proposed framework for application identification is evaluated based on ground truth packet traces from seven P2P application programs. We demonstrate that this framework achieves 90.0% accuracy in application identification with the similarity measure in the feature vector space constructed from discriminative substructures extracted by the proposed graph-mining algorithm (Section 4). The contributions of this article are twofold: (1) the proposed framework enables automatic extraction of discriminative substructures as features using a graph-mining algorithm; (2) the proposed framework achieves high accuracy in application program identification for P2P applications that are more difficult to identify because they use random ports and they may encrypt their communication. Section 5 provides details on issues raised by the proposed procedure.

2. RELATED WORKS

Past work focuses on interactions between hosts or flows in traffic classification. Iliofotou *et al.* [16,17,21] and Jin *et al.* [18] have proposed graph-based approaches to profile an application's activities. They model the social behavior of hosts by representing hosts and their interactions as vertices and edges, respectively, in a graph. However, one problem common to these approaches is that, because they represent flows by edges, not vertices, they do not focus on interactions between different protocols, such as interactions between DNS and HTTP in web browsing. Thus they have difficulty in profiling the activities of applications using multiple protocols. Karagiannis *et al.* [14,15] analyzed transport layer interactions and showed that the characteristics of transport layer interactions could be used to identify the application classes of end-hosts such as web, P2P application and attack classes. However, the temporal interactions between flows were ignored, and more detailed profiling, i.e. application program profiling, is not possible from such an approach.

Flow dependency has also been well studied. Popa *et al.* [22] proposed an approach to identify network application dependencies by using process IDs on operating systems as well as packet traces. However, this approach requires a process monitor to be installed at each end-host, so deployment in some networks, such as a campus guest network, is difficult because each end-host is owned and administered by each user. Kandula et al. focused on flow dependencies to construct communication rules for an edge network [23]. Their focus was similar to ours in terms of flow interactions, and they extracted significant communication patterns. However, temporal information, such as the order of consecutive flows, was not considered in their approach because it partitions flows into time windows to search related flows. Temporal flow causality is crucial for profiling network applications; therefore causality-based features should improve classification performance. In debugging of distributed systems, temporal causality of messages such as remote procedure calls (RPCs) was investigated to find performance bottlenecks of systems [24,25]. However, since the problems addressed there differ from whose targeted here, they cannot be directly applied to application profiling. Unlike such approaches, the approach proposed here categorizes temporal flow interactions into four causality types for application profiling, which plays a significant role when using interactions to design TCG-based features, as shown in Asai *et al.* [20].

Vladislav *et al.* [26] investigated signatures of several popular applications, such as web browsers (Google Chrome, Firefox etc.) and E-mail clients. The goals of the present contribution is thus similar to theirs, focusing on application program profiling. They achieved flow signatures extraction and managed to identify applications by using these flow-extracted signatures, while avoiding recourse to deep packet inspection. However, they did not build up any general method to extract discriminative flow signatures. The present contribution provides a more general method to profile network applications, with automated systematic feature extraction.

TCGs have been proposed to represent temporal and spatial causality of flows for traffic profiling [20]. In Asai *et al.* [20], we defined and discussed a TCG composition method. We also demonstrated that TCGs were useful for network application program profiling with case studies. These case studies relying on ground truth packet traces demonstrated the advantage of TCG-based profiling, which makes use of simple TCG features for profiling both application programs as well as application classes. Moreover, the dynamics of TCG features obtained from time-sliced traces rather than the features calculated from a single snapshot of a trace represented the characteristics of network applications. In addition, it is shown that TCG representation helps network operators to identify the root causes of other flows, e.g. malicious ones. However, the results of network application profiling also showed that automated TCG profiling based solely on simple features has not been accomplished because such features are empirically discovered and selected and they prove insufficient for systematic application identification.

3. NETWORK APPLICATION PROFILING WITH TCGS

The profiling procedure with TCGs generally consists of three steps, as illustrated in Figure 1. The first step amounts to aggregating packets into flows based on the conventional five-tuple: ⟨`proto`, `srcIP`, `srcPort`, `dstIP`, `dstPort`⟩. This aggregation method does not use the packet payload; instead it uses the transport layer header. Note that each direction of a flow is processed as a different flow, e.g. one bidirectional TCP connection is represented as two flows. The first packet of each flow timestamp is assigned to the five-tuple. The second step consists in constructing the TCG from the flows. In the graph, all flows are represented as vertices labeled by five-tuple properties. Related flows are connected by directed edges, labeled by the type of flow causality. The direction of an edge represents temporal transition. Finally, TCG analysis leads to network application profiling. In this article, we construct a feature vector space for the discriminative substructures extracted by a graph-mining algorithm. We also perform supervised learning for the feature vectors of application programs from ground truth datasets. We then identify application programs making use of a similarity measure between the feature vector from an unknown application program and learned feature vectors.

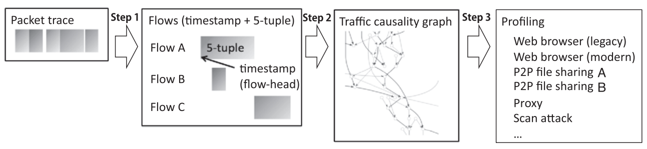


Figure 1. TCG-based profiling procedure. The first step obtains flows by aggregating packets based on the conventional five-tuple. The second step composes a traffic causality graph from the flows. The third step is profiling with the analysis of the traffic causality graph

3.1. TCG construction

Vertices and edges of a TCG represent flows labeled by five-tuple properties and causality type. Four types of causality are used: (1) communication causality (CC); (2) propagation causality (PC); (3) dynamic-port host causality (DHC); and (4) static-port host causality (SHC). The first two—CC and PC—are causality between a flow going to a host (i.e. an IP address) and a flow coming from the host, representing that a flow causes a corresponding flow. CC is a one-to-one relationship from a request to its response (i.e. reverse direction of the request five-tuple). PC is a many-to-many relationship in which one or more flows propagate information into one or more other flows, such as proxy, relay and forwarding. The other two—DHC and SHC—are many-to-many relationships between flows coming from an identical host. DHC is the causality between flows with the same `srcIP` *but* a different `srcPort`; e.g. web browsers create multiple connections to an identical server with different source port numbers. SHC is the causality between flows with the same `srcIP` *and* `srcPort`, e.g. some port scanners use an identical `srcPort` for a sequence of the port scan procedure, and a server uses a static source port for responses. Note that there are hundreds of other possible types of flow causality such as causality between flows with the same `dstIP` and different `srcIPs`, but we focus on these four causality types as characteristics well representing application behaviors.

Let us now briefly explain how to construct a TCG from raw data. The details of the procedure are available in Asai *et al.* [20]. TCG construction consists of two phases: (1) connecting related flows with labeled directed edges; and (2) reducing edges by heuristics.

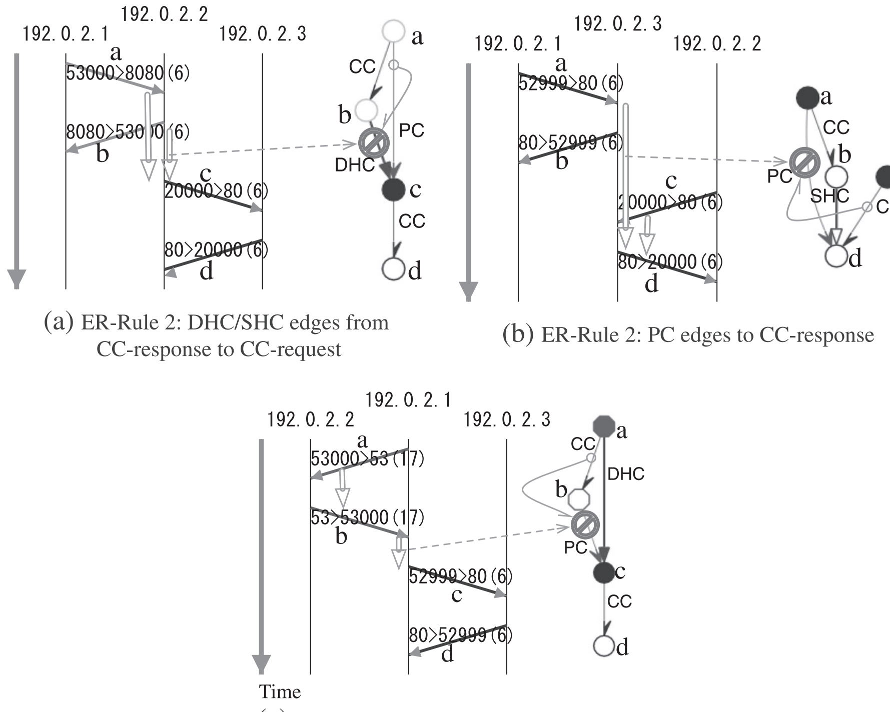
First, TCG is constructed from a set of flows with the timestamp of the flow head and five-tuple parameters by connecting related flows according to simple rules. Let the functions proto(f), srcip(f), srcport(f), dstip(f), and dstport(f) return `proto`, `srcIP`, `srcPort`, `dstIP`, and `dstPort` of the flow f, respectively, and the function timestamp(f) return the start time of flow f, i.e. the timestamp of the first packet of the flow. The algorithm to determine the type of causality between any two flows is defined in Algorithm 1. Note that non-consecutive flows are also processed. Since temporally distant flows can be considered as *not related*, this algorithm first checks a threshold τ (lines 1–3), set to a global and constant value. The threshold is used independently from the edge reduction rules explained in the next paragraph, and is used to limit the number of generated edges, in order to reduce computational complexity. The threshold is also not to connect flows with a long interval, which are not temporally related. It then checks the CC (lines 4–5), PC (lines 6–7), DHC (lines 8–9) and SHC (lines 10–11), in that order. If the input two flows have no causality, the algorithm returns `Nil` (lines 12–13). If the algorithm returns a non-`Nil` value, an edge from f_1 to f_2 labeled with the returned causality is added to the TCG.

Algorithm 1. Get the type of causality between flows f_1 and f_2 with threshold τ

```
procedure getCausalityType($f_1$, $f_2$, $\tau$):
 1: if timestamp($f_2$) − timestamp($f_1$) > $\tau$ then
 2:     return Nil
 3: end if
 4: if proto($f_1$) = proto($f_2$) and srcip($f_1$) = dstip($f_2$) and srcport($f_1$) =
    dstport($f_2$) and dstip($f_1$) = srcip($f_2$) and dstport($f_1$) = srcport($f_2$) then
 5:     return CC
 6: else if dstip($f_1$) = srcip($f_2$) then
 7:     return PC
 8: else if srcip($f_1$) = srcip($f_2$) and srcport($f_1$) $\neq$ srcport($f_2$) then
 9:     return DHC
10: else if srcip($f_1$) = srcip($f_2$) and srcport($f_1$) = srcport($f_2$) then
11:     return SHC
12: else
13:     return Nil
14: end if
end procedure
```

The number of edges is then reduced using heuristics aimed at retaining only significant edges. Three heuristic edge reduction rules (ER-Rules) are defined to remove *tenuous edges*, *irrelative edges* and *insignificant edges*, relying on the following terminology: *CC-request* and *CC-response* are the source and destination vertices of a CC edge, respectively. In the same way, *PC-source*, *PC-destination*, *DHC-source*, *DHC-destination*, *SHC-source* and *SHC-destination* are the source and destination vertices of a PC edge, a DHC edge and an SHC edge, respectively.

- ER-Rule 1. *Removing tenuous edges:* To reduce PC, DHC and SHC edges, we simply remove all PC, DHC and SHC edges except for the temporally closest one for each causality, i.e. the maximum out-degree of a vertex for each causality is one. This rule is based on the observation that the temporally closest flows are generated by the same applications and thus represent direct causality.
- ER-Rule 2. *Removing irrelative edges:* We remove irrelative edges by looking at neighboring edges. DHC/SHC edges from any CC-response to any CC-request should be removed because there must be PC edges and the CC-responses are not the initiators of the CC-requests, i.e. PC-sources are the initiators of the CC-requests, which are the same as the PC-destinations. For example, in Figure 2(a), a DHC edge from a CC-responses *b* to a CC-request *c* is removed. DHC/SHC edges from any CC-request to any CC-response and PC edges to any CC-response should also be removed because CC-responses are initiated only by CC-requests. For example, in Figure 2(b), a PC edge to a CC-response *d* is removed. We keep DHC/SHC edges from a CC-response to another CC-response as a server activity.
- ER-Rule 3. *Removing insignificant edges (PC edges from CC-responses):* We remove PC edges from any CC-response. For example, in Figure 2(c), a PC edge from a CC-response *b*. Note that this is equivalent to ER-Rule 3(a) in Asai *et al.* [20]. We refer to it as ER-Rule 3 for simplicity in this article because we do not use ER-Rule 3(b) and (c) in Asai *et al.* [20].



(a) ER-Rule 2: DHC/SHC edges from CC-response to CC-request

(b) ER-Rule 2: PC edges to CC-response

(c) ER-Rule 3: PC edges from CC-responses

Figure 2. ER-Rule 2 and 3: removing irrelative edges and insignificant edges. (a) ER-Rule 2: DHC/SHC edges from CC-response to CC-request. (b) ER-Rule 2: PC edges to CC-response. (c) ER-Rule 3: PC edges from CC-responses. (For the visualization of edges, we use half arrowheads, double-headed ones, filled ones and open ones for CC, PC, DHC and SHC, respectively)

As an example of TCGs, a TCG composed from a packet trace for web page access to `http://www.google.com/` by Microsoft Internet Explorer is illustrated in Figure 3. We use $\tau = 1$[s] for the threshold in Algorithm 1. As the result of ER-Rules, the number of edges are reduced from 98 to 23.

3.2. *Automated network application profiling framework*

In this section, we propose a framework to achieve automated network application profiling with TCGs. The proposed framework consists of three steps. The first step is to extract discriminative substructures from TCGs and to construct a feature vector space. It relies on a graph-mining algorithm to extract discriminative substructures from TCGs. The second step performs a (naive) supervised learning of the feature vectors in the constructed vector space, tagged with application names. The third step consists in identifying the application corresponding to an input TCG, by using a similarity measure between two TCG feature vectors.

3.2.1. *Feature vector space construction using extracted discriminative substructures by graph mining*

Since TCGs are generalized as 'labeled directed graphs', a graph-mining approach is straightforwardly used to extract discriminative substructures from TCGs. In graph mining, the minimum description length (MDL) approach, which tries to minimize the description length, an index explaining the amount of information of a graph compressed by a substructure has been widely used. We leverage an MDL-based graph-mining algorithm 'Subdue' [27] to extract discriminative substructures from TCGs.

Let us now summarize the Subdue algorithm description length procedure. Let V, E and L be the sets of vertices, of edges and of labels (for both vertices and edges), respectively. The Subdue algorithm defines the description length (DL) of a labeled directed graph $G = (V, E)$ as follows:

$$\mathrm{DL}(G) := h_v + h_e + h_r$$
$$\text{s.t.} \begin{cases} h_v = \log_2(\|V\|) + \|V\| \log_2(\|L\|) \\ h_e = \|E\| \log_2(1 + \|L\|) + (K+1)\log_2(m) \\ h_r = (\|V\| + 1)\log_2(b+1) + \sum_{i=1}^{\|V\|} \log_2 \binom{\|V\|}{k_i} \end{cases}$$

where $\|V\|$, $\|E\|$ and $\|L\|$ denote the number of vertices, edges and labels, respectively. Parameter m is the maximum out-degree of the graph, and b and K are defined in equations (1) and (2), respectively, where $A = (a_{ij})$ is the adjacency matrix of graph G and $k_i = \sum_{j=1}^{\|V\|} a_{ij}$ is the strength of node i:

$$b := \max_i k_i \tag{1}$$

$$K := \sum_{i=1}^{\|V\|} k_i \tag{2}$$

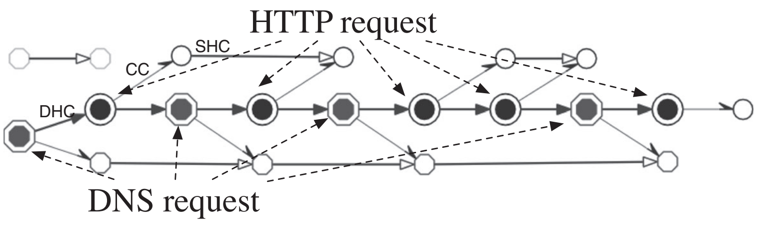


Figure 3. Example of TCG constructed from a google web page access trace

The simplest MDL algorithm searches a substructure that minimizes the description length of the graph compressed by it, i.e. it minimizes $\mathrm{DL}(G|S)$, where $G|S$ represents graph G compressed by substructure S. The Subdue algorithm extends this basic MDL algorithm by defining the feature quantity $f(S, G)$ of substructure S to graph G in equation (3):

$$f(S,G) = \frac{\mathrm{DL}(G)}{\mathrm{DL}(S) + \mathrm{DL}(G|S)} \tag{3}$$

A great feature $f(S, G)$ indicates that substructure S is more representative of graph G; i.e. the substructure explains the graph better. In other words, a smaller $\mathrm{DL}(G|S)$ stems from substructures S with smaller description lengths. Indeed, substructures with smaller description length that minimize $\mathrm{DL}(G|S)$ are considered more discriminative because they frequently appear in the graph. Note that the Subdue algorithm searches for a substructure that *maximizes* the feature quantity, although it is based on the *minimum* description length approach, because the description length of a graph compressed by a substructure constitutes the denominator of the feature quantity definition (3).

The Subdue algorithm searches the best substructures that maximize the feature quantity $f(S, G)$ by making use of a beam search strategy. The beam search strategy explores substructures by expanding an edge to the promising set of subgraphs, starting from single-node subgraphs. After expanding an edge, a limited number of subgraphs with larger feature quantity are chosen as the promising set of subgraphs for the next procedure of the beam search. The limited number of subgraphs for the promising set is referred to as the *width* of the beam search. Note that the beam search equals the breadth-first search when the width of the beam search is large enough. The Subdue algorithm relies on five parameters: (1) *beamwidth*: the width of the beam search; (2) *limit*: a limit of the rounds (i.e. amount of computation) in the beam search; (3) *maxsize*: the maximum number of vertices for an extracted substructure; (4) *minsize*: the minimum number of vertices for an extracted substructure; and (5) *maxsubs*: the maximum number of the substructures to be returned.

A feature vector space is constructed to embed the extracted substructures of TCGs. Let $\mathbb{S} = \{S_1, \ldots, S_n\}$ be the set of substructures extracted from one or more than one TCGs, where the number of the substructures in $\mathbb{S}$ is n, the feature vector $\boldsymbol{f}$ for a given TCG G_x is defined in equation (4):

$$\boldsymbol{f}(G_x) := [f(S_1, G_x), \ldots, f(S_n, G_x)] \tag{4}$$

Here, we use the same feature quantity as used in the Subdue algorithm to construct the feature vector space because it also represents how frequently the substructure appears in the graph. Note that the set of extracted substructures $\mathbb{S}$ may contain duplicate substructures (i.e. the actual dimension of the vector space is less than n) but we do not eliminate duplicates because it does not lead to significant problems in application profiling. This is further discussed in Section 5.

3.2.2. *Similarity measure between feature vectors for application identification*

We use the cosine similarity between two feature vectors to measure the similarity between two TCGs. The cosine similarity is commonly used for a similarity measure in the research area of natural language processing [28]. A feature of cosine similarity, that it is independent of the magnitude of vectors, is beneficial in measuring the similarity between feature vectors of TCGs because the magnitude of feature vectors varies by the size of TCGs. The cosine similarity (sim) between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined by equation (5):

$$\mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}) := \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|} \tag{5}$$

where $\boldsymbol{u} \cdot \boldsymbol{v}$ denotes the inner product of $\boldsymbol{u}$ and $\boldsymbol{v}$ in the vector space $\mathbb{R}^n$, and $\|\boldsymbol{v}\|$ denotes the ℓ^2-norm of the vector $\boldsymbol{v}$.

Let us now assume that TCG learned vectors are available from supervised learning achieved on packet traces with available ground truths. The detailed learning procedures are discussed in Section 5. Application identification is achieved by searching the learned feature vector that is most similar to the feature vector computed from a target input TCG. A naive identification algorithm is used to find the learned feature vector which attains the maximum cosine similarity to the feature vector of the input TCG.

4. PERFORMANCE EVALUATION

Accuracy of the application identification procedure is now evaluated. To that end, a leave-one-out cross-validation procedure and a k-fold cross-validation using ground truth packet traces as learning and validation datasets are conducted.

4.1. *Experimental setup*

To create ground truth packet traces for the evaluation of automated application traffic profiling, we captured packet traces of five P2P file sharing applications (BitTorrent, Cabos, Winnyp, Share, and Perfect Dark), of one P2P video streaming application (BBbroadcast), and of one voice over IP and chat application (Skype) on a clean-installed operating system (Windows 7). Note that we focus on P2P applications since they are difficult to identify because they use random ports and they may support encrypted communication. Indeed, four of these seven P2P applications—Winnyp, Share, Perfect Dark and Skype—use encryption by default. Three traces are captured for each application program, and split each trace into 10-minute subtraces. For example, the duration of trace 1 of Skype is equal to or larger than 350 minutes but less than 360 minutes and then split into 35 subtraces. The ground truth packet traces of the application programs and the number of 10-minute subtraces for each application program are summarized in Table 1. The total number of 10-minute subtraces is 831. We construct TCGs with $\tau = 1$[s] and ER-Rules 1, 2 and 3 for each packet trace. The vertices of TCGs are labeled with the protocol number in the five-tuple of flows; i.e. TCP, UDP and ICMP.

The Subdue algorithm relies on five parameters, as described in Section 3.2.1. In this experiment, we use 128, half the value of the number of edges in the input graph, 6, 1 and 50, for *beamwidth*, *limit*, *maxsize*, *minsize* and *maxsubs*, respectively. Among these parameters, *beamwidth* is the only parameter subject to optimization in the beam search strategy. The other parameters can be set if we use the breadth-first search strategy instead of the beam search strategy. Here, we discuss *beamwidth* to demonstrate that the value 128 for *beamwidth* is large enough to extract the best substructures appropriately. The feature quantities of extracted substructures and the number of extracted substructures are computed as functions of the *beamwidth*, for every subtrace from the ground truth datasets, and shown in Figure 4. It shows that both the feature quantities and the number of extracted substructures

Table 1. Ground truth packet traces for automated application traffic profiling

		No. of 10-minute subtraces		
Application program	Abbr.	(Trace 1)	(Trace 2)	(Trace 3)
BitTorrent	bt	4	04	7
Cabos	cabos	13	16	278
Winnyp	winny	4	4	5
Share	share	45	22	36
Perfect Dark	pd	8	24	4
BBbroadcast	bbb	4	9	26
Skype	skype	35	264	19

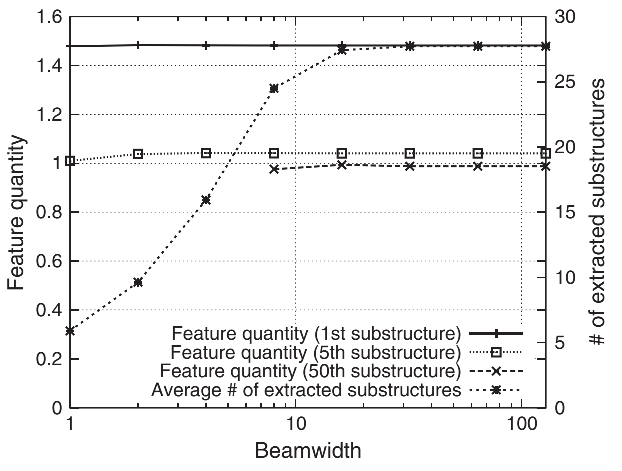


Figure 4. Average feature quantity of extracted substructures and the number of extracted substructures as functions of the *beamwidth*

saturate when *beamwidth* is equal to or larger than 32, thus showing that the best substructures are appropriately extracted. Note that the feature quantity of 50th substructure displayed in this figure is most strongly affected by the *beamwidth* since it is the last one when we choose 50 for *maxsubs*. We conjecture that the value 128 for *beamwidth* is large enough to achieve results (i.e. substructures) that approximately match those obtained from the breadth-first search strategy. Note that 128 is the value that we expected to be large enough before this discussion on the *beamwidth* and used for preliminary experiments, so we use this specific value.

Since the number of 10-minute subtraces varies strongly among ground truth traces, we use at most 24 subtraces for each trace for this experiment. This limitation yields 291 subtraces in total. We first extract the best substructures from these 10-minute subtraces. The total number of extracted substructures is 10 779. We then construct a feature vector space using these 10 779 extracted substructures. Note that this limitation also aims at reducing the computational time because the subgraph isomorphism problem used in the Subdue algorithm is NP-complete and thus requires a huge amount of time to conduct this evaluation. Computational complexity is further discussed in Section 5.

4.2. Feature vectors and identification results

The feature vectors are computed according to equation (4) from the TCG of each 10-minute subtrace. We first demonstrate that the similarity between feature vectors is effective for application identification. Figure 5 displays the feature vectors of each 10-minute subtrace in the constructed feature vector space. Note that each column of the matrix represents a feature vector of a subtrace. Only the best 5 substructures are used to construct the feature vector space to draw this figure. This limitation reduces the number of elements of the feature vector space to 1443 from 10 779, in accordance with typical high-resolution printing of 600 dpi. This figure illustrates that feature vectors computed from traces associated with the same application display similar patterns. It is indeed not surprising that the feature quantity corresponding to substructures extracted from one same subtrace is large. Yet horizontal stripe patterns among subtraces related to the same application can also be observed. Note that the cosine similarity is independent of the magnitude of the feature vectors, and consequently the absolute feature quantities of horizontal stripes (e.g. horizontal lines through subtraces) are meaningless, but horizontal stripe patterns from the bottom of the figure to the top are significant. This demonstrates that subtraces from the same application display similar patterns in the feature vector space. The similarity between feature vectors can thus be used for the application program identification.

Application identification performance is then evaluated. To that end, two types of cross-validations are conducted: the leave-one-out cross-validation and the k-fold cross-validation. In the leave-one-out cross-validation, we use one 10-minute subtrace for validation and the other subtraces for learning. In

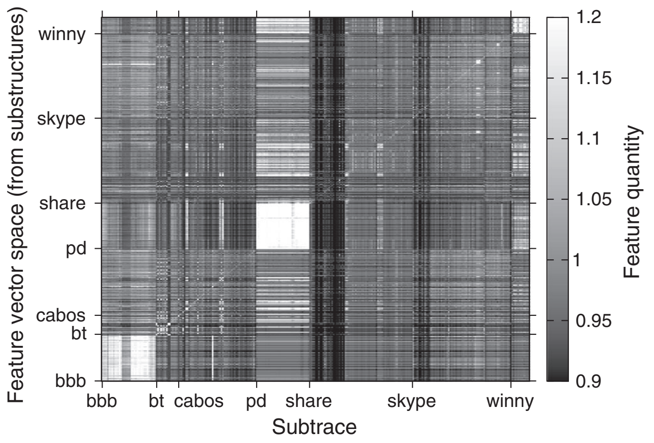


Figure 5. Feature vectors of each 10-minute subtrace for the best 5 substructures: horizontal and vertical axes represent the 10-minute subtraces and corresponding substructures in the constructed feature vector space, respectively. For readability, only the best 5 substructures in the feature vector space are used to draw this figure

Table 2. Confusion matrix by the leave-one-out cross-validation

Input\output	bbb	bt	cabos	share	skype	pd	winny
bbb	37	0	0	0	0	0	0
bt	0	13	1	0	1	0	0
cabos	1	2	48	1	1	0	0
share	0	0	2	61	6	0	1
skype	0	0	7	2	58	0	0
pd	0	0	0	0	0	36	0
winny	0	0	1	3	0	0	9

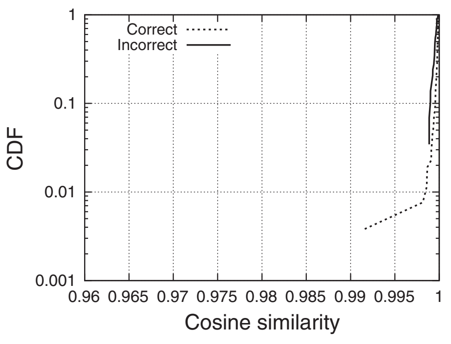
Table 3. Confusion matrix by the k-fold cross-validation

Input\output	bbb	bt	cabos	share	skype	pd	winny
bbb	31	0	6	0	0	0	0
bt	0	10	3	0	2	0	0
cabos	1	1	47	1	3	0	0
share	1	0	20	39	7	0	3
skype	0	0	34	0	33	0	0
pd	0	0	0	0	0	36	0
winny	0	0	1	2	0	0	10

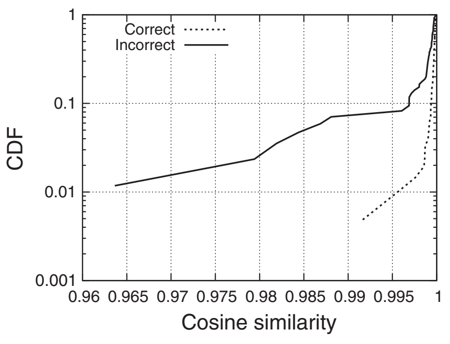
the k-fold cross-validation, we use 10-minute subtraces belonging to one of three traces for validation for each application and the other subtraces for learning. Thus we obtain k=3 partitions for the k-fold cross-validation.

Table 2 summarizes the confusion matrix of the leave-one-out cross-validation. The diagonal elements in this table indicate correct application identification, while off-diagonal terms quantify miss-classifications, thus showing a 90.0% accuracy in application identification.

Similarly, Table 3 displays the confusion matrix of the k-fold cross-validation. The proposed framework achieves a 70.8% accuracy in application identification. This accuracy is lower than the accuracy in the leave-one-out cross-validation because the number of feature vectors for learning is smaller due to the different learning dataset selection of cross-validation, and consequently significant feature vectors are not found for some subtraces. As shown in Figure 6, the cosine similarity values corresponding



(a) Leave-one-out cross-validation



(b) K-fold cross-validation

Figure 6. Cumulative distribution functions (CDFs) of cosine similarity by the identification result: in the k-fold cross-validation, the cosine similarity of incorrect identification is distributed at smaller values than that of correct identification. This characteristic is not obvious in the leave-one-out cross-validation but in the k-fold cross-validation. This is because the number of the learned feature vectors in the k-fold cross-validation is smaller than that in the leave-one-out cross-validation, and the accuracy of application identification is lower. (a) Leave-one-out cross-validation. (b) k-fold cross-validation

to incorrect identifications in the k-fold cross-validation are observed to be distributed across smaller values than those for correct identifications, as opposed to what can be seen in the leave-one-out cross-validation. If the similarity of incorrect identifications is distributed almost at 1.0, the feature vector space constructed from the learning dataset is not good for the identification because it is able to find a similar vector from the learned vectors but it is that of a different program. On the other hand, if the similarity of incorrect identifications is distributed at smaller values, the learning is not enough because it does not find any similar vectors from the learned feature vectors. Therefore, the proposed framework cannot find feature vectors from the learning set that are similar to the target set because the number of learned feature vectors is not large enough in the k-fold cross-validation. Table 2 also shows another reason for this lower accuracy in the k-fold cross-validation: a significant number (39%) of incorrect identifications stem from Skype and Share identified as Cabos. This is because the subtraces of these applications have similar feature vectors. Figure 5 confirms that analysis, showing similar horizontal stripe patterns shared by these applications. To avoid these incorrect identifications, the quality of the learned feature vectors should be improved by collecting more datasets for learning. Another possible solution is to improve the extraction of discriminative substructures or identification algorithm (e.g. similarity measure) to appropriately identify these applications.

5. DISCUSSION

Let us now further discuss some issues raised in the previous sections.

5.1. TCG composition

Due to multitasking of current operating systems, flows from different applications may be inaccurately connected during the TCG composition. A case study in the prior work [20] showed that a TCG component of a web browser could be composed from a traffic trace captured at a gateway router, containing traffic from other applications and other hosts. However, inaccurate connections during TCG composition potentially occur. To mitigate these inaccurate connections, we will consider extending the flow dependency extraction method proposed in Kandula *et al.* [23] to the TCG composition in order to divide a TCG into several components. The threshold τ in the edge connection algorithm should also be discussed. A shorter threshold possibly mitigates the inaccurate connections due to multitasking but causes a split in a TCG. The threshold τ and the accuracy and computational complexity should be evaluated in the future.

5.2. *Complexity of vertex labeling*

We labeled vertices with the protocol number in the five-tuple in Section 3.2. The vertex labeling can be extended, and properties of flows as well as the other fields in the five-tuple can be used for the vertex labeling. For example, the statistical properties of flows that have been used in traffic classification methods [9–11] can be added to vertex labeling parameters. However, these extension might increase the complexity of the methodology. This is one reason why we adopt vertex labeling based on conventional five-tuple.

5.3. *Computational complexity*

Two bottlenecks exist in the proposed framework in terms of the computational complexity. The first bottleneck is the TCG construction method. The computational complexity of Algorithm 1 is $O(\|V\|^2)$, where $\|V\|$ is the number of vertices (i.e. flows) if the threshold τ is not set. This computational complexity can be reduced by setting an appropriate threshold τ. The other bottleneck is the feature quantity computation that is used both in the Subdue algorithm and application identification. The subgraph isomorphism problem used in the feature quantity computation is NP-complete. This computational complexity can be reduced by the following two methods: (1) reducing the size of the graph with the threshold τ; and (2) reducing the size of the subgraph with the parameter *maxsize* of the Subdue algorithm. For the Subdue algorithm, the number of the feature quantity to compute can be reduced by tuning the parameters *beamwidth*, *limit* and *maxsubs*. However, this parameter tuning might affect the accuracy of the profiling. Therefore, tuning of these parameters is required to be carefully carried out so as not to degrade the accuracy and not to increase the computational complexity.

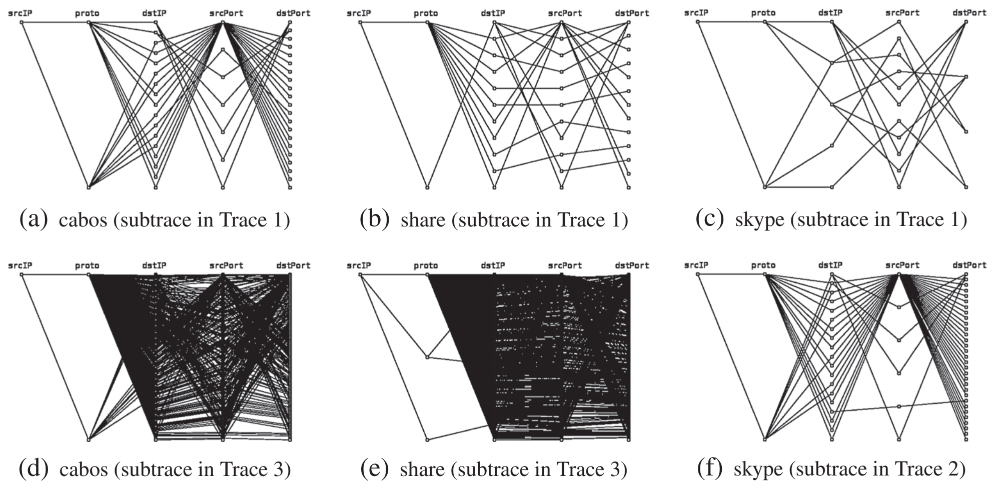
5.4. *Advantages against existing approach based on spatial interactions between flows*

Spatial interactions between flows are used for traffic classification in BLINC [14]. This approach does not focus on the causality of flows and cannot easily profile application programs without payload inspection, though they might succeed in profiling certain application classes such as P2P file sharing. The graphlets[1] proposed in BLINC of six ground truth packet subtraces of three applications used in Section 3.2.1. (Table 1) are drawn in Figure 7, which sheds light on the difficulty in profiling application programs with simple rules on graphlets as proposed in Karagiannis *et al.* [14]: shapes of the graphlets between different applications, Cabos (a) and Skype (f) are similar, and the shapes of the graphlets between the same application, Skype (c) and (f), are not similar. An advanced analysis of graphlets [19] has been proposed but it still does not focus on the application programs. Note that graph mining or other advanced techniques might be also be applied to graphlet-based application program profiling but this has not yet been investigated.

5.5. *Graph-mining algorithm and duplicate substructures in the feature vector space construction*

We adopted the Subdue algorithm based on the MDL approach for graph mining. In the Subdue algorithm, its graph-mining procedure to search discriminative substructures is based on complete matching, and the similarity between substructures is not taken into account, extending substructure isomorphism. Other graph-mining algorithms that take into account similarity might improve the definition of description length and the profiling performance, though it would definitely further increase computational complexity. Moreover, duplicate substructures are not eliminated in the feature vector space because they do not yield significant issues for application profiling. However, profiling performance will be improved by taking into account similarity.

[1]*The term 'graphlet' here is different from another definition by Pržulj et al.* [29,30] *used in the bioinformatics research area.*



(a) cabos (subtrace in Trace 1) (b) share (subtrace in Trace 1) (c) skype (subtrace in Trace 1)

(d) cabos (subtrace in Trace 3) (e) share (subtrace in Trace 3) (f) skype (subtrace in Trace 2)

Figure 7. Graphlets of application programs: (a) Cabos (subtrace in trace 1); (b) Share (subtrace in trace 1); (c) Skype (subtrace in trace 1); (d) Cabos (subtrace in trace 3); (e) Share (subtrace in trace 3); (f) Skype (subtrace in trace 2)

5.6. *Learning of feature vectors*

We use the same environment and one version for each application both for learning of the feature vectors and for identification. However, different versions of application, underlying environments or topologies possibly result in different flow patterns, as mentioned in device-fingerprinting works [31,32]. The impact on the profiling induced by these differences is to be evaluated. One possibility to be tolerant of this impact is unsupervised learning. We use supervised learning to tag the feature vectors with the name of application programs, in Section 3.2. However, the proposed framework can adopt unsupervised learning by using similarity as achieved in the research area of natural language processing [28], though the tag of the feature vectors (i.e. exact application name) cannot be annotated. We will examine several clustering techniques and apply them to feature vectors (e.g. Figure 5) to achieve unsupervised learning and classification for a more advanced network application traffic-profiling framework.

6. CONCLUSION

We propose a network application profiling framework adopting TCGs that represent temporal and spatial causality of flows to identify application programs. We leverage a graph-mining algorithm to extract discriminative substructures from TCGs as features for network application profiling. It overcomes the limitations stemming from manually selected simple features that we used in an earlier work. We evaluated the accuracy of the proposed framework in the application program identification with ground truth packet traces of seven P2P application programs, and demonstrated that this framework achieved 90.0% accuracy in application identification.

The contributions of this manuscript are that the proposed framework enables us to automatically extract discriminative substructures as features using a graph-mining algorithm, and that the proposed framework achieves high accuracy in application program identification for P2P applications that are difficult to identify because they used random ports and might encrypt communication.

REFERENCES

1. Karagiannis T, Broido A, Faloutsos M, claffy K. Transport layer identification of P2P traffic, In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement: IMC'04*, Taormina, Sicily, Italy, 2004; 121–134.
2. Sourcefire. Inc. Snort. Available: http://www.snort.org/ [9 May 2014].

3. Levandoski J, Sommer E, Strait M. *Application layer packet classifier for Linux*. Available: http://l7-filter.sourceforge.net/ [9 May 2014].
4. ntop. nDPI: open and extensible GPLv3 deep packet inspection library. Available: http://www.ntop.org/products/ndpi/ [9 May 2014].
5. Sen S, Spatscheck O, Wang D. Accurate, scalable in-network identification of P2P traffic using application signatures, In *Proceedings of the 13th International Conference on World Wide Web: WWW'04*, Manhattan, NY, USA, ACM, 2004; 512–521.
6. Haffner P, Sen S, Spatscheck O, Wang D. ACAS: automated construction of application signatures, In *Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data: MineNet'05*, Philadelphia, PA, USA, 2005; 197–202.
7. Moore AW, Papagiannaki K. Toward the accurate identification of network applications. In *Passive and Active Network Measurement*, Lecture Notes in Computer Science, vol. 3431 Springer: Berlin, 2005, pp. 41–54.
8. Alcock S, Nelson R. Libprotoident: traffic classification using lightweight packet inspection. *Technical Report*, 2013.
9. Moore AW, Zuev D. Internet traffic classification using Bayesian analysis techniques, In *SIGMETRICS'05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Banff, Alberta, Canada, 2005; 50–60.
10. Crotti M, Dusi M, Gringoli F, Salgarelli L. Traffic classification through simple statistical fingerprinting, *SIGCOMM Computer Communication Review* 2007; **37**(1): 5–16.
11. Erman J, Mahanti A, Arlitt M, Williamson C. Identifying and discriminating between web and peer-to-peer traffic in the network core, In *Proceedings of the 16th International Conference on World Wide Web, WWW'07*, Banff, Alberta, Canada, ACM, 2007; 883–892.
12. Dewaele G, Fukuda K, Borgnat P, Abry P, Cho K. Extracting hidden anomalies using sketch and non Gaussian multiresolution statistical detection procedures, In *LSAD'07: Proceedings of the 2007 Workshop on Large Scale Attack Defense*, Kyoto, Japan, ACM, 2007; 145–152.
13. Trestian I, Ranjan S, Kuzmanovi A, Nucci A. Unconstrained endpoint profiling (googling the Internet), In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, Seattle, WA, USA, 2008; 279–290.
14. Karagiannis T, Papagiannaki K, Faloutsos M. BLINC: multilevel traffic classification in the dark, In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications: SIGCOMM'05*, Philadelphia, PA, USA, ACM, 2005; 229–240.
15. Karagiannis T, Papagiannaki K, Taft N, Faloutsos M. Profiling the end host. In *Passive and Active Network Measurement*, Lecture Notes in Computer Science, vol. 4427 Springer: Berlin, 2007, pp. 186–196.
16. Iliofotou M, Pappu P, Faloutsos M, Mitzenmacher M, Singh S, Varghese G. Network monitoring using traffic dispersion graphs (TDGs), In *IMC'07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, San Diego, CA, USA, 2007; 315–320.
17. Iliofotou M, Faloutsos M, Mitzenmacher M. Exploiting dynamicity in graph-based traffic analysis: techniques and applications, In *CoNEXT'09: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, Rome, Italy, ACM, 2009; 241–252.
18. Jin Y, Sharafuddin E, Zhang ZL. Unveiling core network-wide communication patterns through application traffic activity graph decomposition, In *SIGMETRICS'09: Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems*, Seattle, WA, USA, ACM, 2009; 49–60.
19. Himura Y, Fukuda K, Cho K, Borgnat P, Abry P, Esaki H. Synoptic graphlet: bridging the gap between supervised and unsupervised profiling of host-level network traffic, *IEEE/ACM Transactions on Networking* 2013; **21**(4): 1284–1297.
20. Asai H, Fukuda K, Esaki H. Traffic causality graphs: profiling network applications through temporal and spatial causality of flows, In *Proceedings of the 23rd International Teletraffic Congress: ITC'11*, San Francisco, CA, USA, 2011; 95–102.
21. Iliofotou M, Gallagher B, Eliassi-Rad T, Xie G, Faloutsos M. Profiling-by-association: a resilient traffic profiling solution for the Internet backbone, In *Proceedings of the 6th International Conference, Co-NEXT'10*, Philadelphia, PA, USA, ACM, 2010; 2:1–2:12.
22. Popa L, Chun BG, Stoica I, Chandrashekar J, Taft N. Macroscope: end-point approach to networked application dependency discovery, In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT'09*, Rome, Italy, 2009; 229–240.
23. Kandula S, Chandra R, Katabi D. Whats going on? Learning communication rules in edge networks, In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, Seattle, WA, USA, 2008; 87–98.
24. Aguilera MK, Mogul JC, Wiener JL, Reynolds P, Muthitacharoen A. Performance debugging for distributed systems of black boxes, In *Proceedings of the 19th ACM Symposium on Operating Systems Principles, SOSP'03*, Bolton Landing, NY, USA, 2003; 74–89.
25. Reynolds P, Wiener JL, Mogul JC, Aguilera MK, Vahdat A. WAP5: black-box performance debugging for wide-area systems, In *Proceedings of the 15th International Conference on World Wide Web: WWW'06*, Edinburgh, Scotland, 2006; 347–356.
26. Perelman V, Melnikov N, Schoenwaelder J. Flow signatures of popular applications, In *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management*, Dublin, Ireland, 2011; 9–16.
27. Holder LB, Cook DJ, Djoko S. Substructure discovery in the subdue system, In *Workshop on Knowledge Discovery in Databases*, Seattle, WA, USA, 1994; 169–180.
28. Tan PN, Steinbach M, Kumar V. *Introduction to Data Mining*, Pearson Education India: Delhi, 2007.
29. Pržulj N, Corneil D, Jurisica I. Modeling interactome: scale-free or geometric? *Bioinformatics* 2004; **20**(18): 3508–3515.
30. Pržulj N. Biological network comparison using graphlet degree distribution, *Bioinformatics* 2007; **23**(2): e177–e183.

31. Scholz H. *SIP stack fingerprinting and stack difference attacks (2006)*. Available: http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Scholz.pdf [9 May 2014].
32. François J, Abdelnur H, State R, Festor O. Automated behavioral fingerprinting. In *Recent Advances in Intrusion Detection*, Kirda E, Jha S, Balzarotti D (eds.), Lecture Notes in Computer Science, vol. 5758 Springer: Berlin, 2009, pp. 182–201.

AUTHORS' BIOGRAPHIES

Hirochika Asai is an assistant professor at the University of Tokyo. He received his Ph.D degree in information science and technology from the University of Tokyo in 2013. His current research interests are Internet traffic measurement and analysis, operations and management of infrastructure and platforms in cloud computing, and high-performance network operating systems.

Kensuke Fukuda is an associate professor at the National Institute of Informatics (NII). 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. He was a visiting scholar at Boston University in 2002, and also was a researcher of PRESTO JST (Sakigake) in 2008–2012. His current research interests are Internet traffic measurement and analysis, intelligent network control architectures, and scientific aspects of networks.

Patrice Abry received the degree of Professeur-Agrégé de Sciences Physiques, in 1989 at Ecole Normale Supérieure de Cachan and completed a Ph.D in Physics and Signal Processing Université Claude-Bernard Lyon I, in 1994. Since 1995, he is a permanent CNRS researcher, at the Physics Dept. of Ecole Normale Supérieure de Lyon where he is now a CNRS Senior Researcher. He received the AFCET-MESR-CNRS prize for best Ph.D in Signal Processing for the years 1993–94. He was elected IEEE Fellow in 2011. He authored of a book 'Ondelettes et Turbulences', published in 1997, by Diderot (Paris, France) and is also the coeditor of a book 'Scaling, Fractals and Wavelets', ISTE, UK, 2009 (French Edition, 2002). His current research interests include wavelet-based analysis and modeling of scaling phenomena and related topics (self-similarity, stable processes, multi-fractal, 1/f processes, long-range dependence, local regularity of processes, infinitely divisible cascades, departures from exact scale invariance, ...). Hydrodynamic turbulence and the analysis and the modeling of computer network teletraffic are the main applications under current investigation. He also recently got involved in the analysis of heart rate variability and brain activity fluctuations.

Pierre Borgnat is Researcher at CNRS in Signal Processing, at theLaboratoire de Physique, ENS de Lyon, France. He was born in France in 1974, was received at the École Normale Supérieure de Lyon in 1994 and as the Professeur Agrégé in Physical Sciences in 1997.He got a Ph.D. degree in physics and signal processing in 2002. He worked in 2003 in the Signal and Image Processing group of the IRS, IST (Lisbon, Portugal). Since October 2004, he has been a full-time CNRS researcher. His research interests are in statistical signal processing of non-stationary processes (time-frequency, time warping, test of stationarity, EMD, ...), of scaling phenomena (time-scale, wavelets)and in processing of graph signals and complex networks. He works on many applications of these signal processing methods, among them for networking,Internet traffic modeling and measurements, and their applications (traffic classification, anomaly identification, ...), for fluid mechanics, for analysis of social data, or for transportation studies.

Hiroshi Esaki received the B.E. and M.E. degrees from Kyushu University, Fukuoka, Japan, in 1985 and 1987, respectively. And, he received Ph.D from University of Tokyo, Japan, in 1998. In 1987, he joined Research and Development Center, Toshiba Corporeation, where he engaged in the research of ATM systems. From 1990 to 1991, he has been at Applied Research Laboratory of Bellcore Inc., New Jersey (USA), as a residential researcher. From 1994 to 1996, he has been at CTR (Center for Telecommunication Research) of Columbia University in New York (USA). During his staying at Columbia University, he has proposed the CSR architecture, that is the origin of MPLS (Multi-Protocol Label Switching), to the IETF and to the ATM Forum. From 1996 to 1998, he has conducted the CSR project in Toshiba, as a chief architect. Since 1998, he has served as a professor at the University of Tokyo, and as director of WIDE Project (HYPERLINK "http://www.wide.ad.jp" www.wide.ad.jp). He is a fellow of IPv6 Forum, a vice president of JPNIC and a Board of Trustee for ISOC (Internet Society).