Traffic Causality Graphs: Profiling Network Applications through Temporal and Spatial Causality of Flows

Hirochika Asai
The University of Tokyo
panda@hongo.wide.ad.jp

Kensuke Fukuda
NII / PRESTO JST
kensuke@nii.ac.jp

Hiroshi Esaki
The University of Tokyo
hiroshi@wide.ad.jp

***Abstract*—Traffic causality graphs (TCGs) are proposed for visualizing and analyzing the temporal and spatial causality of flows to profile network applications without inspecting packet payload. A key idea of TCGs is to focus on the causality of individual flows composed of different application protocols rather than a set of host flows. This idea enables us to analyze temporal interactions between flows, such as the temporal manner of flow generation by identical application programs and interactions between incoming and outgoing flows. We demonstrate the effectiveness of TCGs for profiling network applications in case studies with ground truth datasets. The results show that the simple features of TCGs are discriminative for profiling network applications and that TCGs are also advantageous for profiling application programs, such as user agents of Web browsers and proxies that cannot be classified by existing approaches; this enables us to identify a specific application program that uses the same protocol as other programs. In addition to the TCG features, the visualization of TCGs reveals the causality of each flow, which consequently helps network operators to identify the root causes of other flows, such as malicious ones.**

I. Introduction

Internet usage has become diversified and various network applications are run on the Internet. In this environment, traffic classification is one of the key technologies for IP network management tasks, such as analysis of security incidents, network topology design, and traffic engineering. The simplest traffic classification method is based on the source and destination port numbers of the transport layer (e.g., TCP and UDP) [1]. However, a problem for the port-based method is that port numbers are not statically bound to each application. For example, network applications can use non-standard ports, especially when there are firewall port restrictions. Moreover, some network applications such as peer-to-peer applications may use a random port. Cases such as these make it difficult to classify traffic according to port numbers. Many advanced techniques that do not rely only on port numbers have been proposed for profiling network application traffic. Signature-based traffic classifiers [2], [3], [4], [5], [6], [7] identify applications from network traffic by inspecting packet payloads (i.e., application data). However, packet inspection creates some privacy concerns, and it is difficult to conduct when the data is encrypted. To solve these privacy and encryption problems, statistical approaches [8], [9], [10] have been proposed to classify applications from network traffic. These approaches use statistical properties, such as 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 [11]. An intrinsic approach [12] not relying on signatures or statistical properties checks IP addresses in flows and Web contents found in search engine results corresponding to an IP address to profile end-hosts. However, as the authors mentioned, it cannot profile end-hosts using P2P applications, and applying it to application profiling is difficult because end-hosts, especially end-user hosts, use multiple applications. Other approaches [13], [14], [15], [16], [17] use information on spatial interactions between hosts or flows for traffic classification. However, these approaches do not focus on the causality of flows and cannot easily profile application programs such as 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. Moreover, since these approaches neglect the causality, the root causes of flows cannot be identified.

In summary, the main problem of existing approaches is that they cannot profile application programs well, although application program profiling is important in network operation [18]. The existing approaches do not focus on the temporal order of flows, despite applications generating flows in a certain temporal manner that varies by application type; for example, Web browsers *first* resolve a domain name by DNS and *then* retrieve a content by HTTP. In addition to temporal order of flows, the approaches also ignore interactions between incoming and outgoing flows. For example, a Web proxy partly behaves like a Web client; it resolves a domain name and retrieves content from the original Web server, after receiving an HTTP request. Therefore, the temporal and spatial causality of flows is highly significant for profiling network applications. One practical use of this application program profiling is to identify a specific application program that uses the same protocol as other programs but has security problems.

In this work, we focus on the temporal and spatial causality of individual flows for profiling network applications, without looking at packet payload. Our final goal is to automatically profile application classes and to automatically profile appli-

978-0-9836283-0-9 © 2011 ITC

This paper was peer reviewed by subject matter experts for publication in the Proceedings of ITC 2011

cation programs. We propose *traffic causality graphs (TCGs)* that represent temporal and spatial causality of flows for visualizing and analyzing traffic patterns to profile network applications. We discuss a TCG composition method and a network application traffic profiling approach using TCGs. The main contribution of this paper is to propose a network application traffic profiling approach that enables us to analyze temporal and spatial flow causality. Case study results show the advantage of the proposed approach, which uses the simple features of TCGs, for profiling application programs as well as application classes. In addition to use of the features, we show that the TCG visualization helps network operators to identify the root causes of other flows, e.g., malicious ones.

II. Related works

Several works focus on the interactions between hosts or flows to classify traffic. Iliofotou et al. [15], [16], [19] and Jin et al. [17] have proposed graph-based approaches to profile application's activities. They model the social behavior of hosts by representing hosts and their interactions as vertices and edges in a graph, respectively. In these approaches, temporal activities as well as momentary ones can be analyzed by looking at series of graph snapshots. However, one problem common to these approaches is that they do not focus on interactions between different protocols, such as interactions between DNS and HTTP in Web browsing, because they represent flows by edges, not vertices. Thus, they have difficulty in profiling the activities of applications using multiple protocols. Karagiannis et al. [13], [14] analyzed spatial five-tuple (flow) interactions and they showed that the characteristics of flow 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 with their approach, such as application program profiling, is not possible.

Flow dependency has also been researched. Popa et al. [20] 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 [21]. 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 missed because their approach partitions flows into time windows to search related flows. Namely, temporal flow causality is more useful for profiling network applications if characteristics are extracted from the causality. Unlike their approach, our proposed approach categorizes temporal flow interactions into four relationship types. This categorization has a significant role when we use the interactions as features in TCGs, as shown in §IV-B.

Vladislav et al. [18] investigated signatures of several popular applications, such as Web browsers (Google Chrome, Firefox etc.) and E-mail clients. The motive of their research is similar to ours in terms of focusing on application program profiling. They achieved to extract flow signatures of these applications, and identify them without deep packet inspection by using extracted flow signatures. However, they do not build up any general methods to extract the discriminative flow signatures. We provide more general method and features to profile applications.

III. Traffic Causality Graph Composition

The profiling procedure consists of three steps. The first step is to aggregate packets into flows based on a conventional five-tuple: ⟨`proto`, `srcIP`, `srcPort`, `dstIP`, `dstPort`⟩. This method does not require 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. In this step, we assign the timestamp of the first packet of each flow to the flow. The second step is to compose a TCG from the flows. All flows are represented as vertices in the graph, and then related flows are connected by directed edges. The direction of an edge represents temporal transition. The TCG composition consists of two phases: 1) connecting related flows and 2) reducing edges by heuristics. Finally, we analyze the TCG and profile network applications.

A. Flow causality and visualization of TCGs

In TCGs, vertices and edges represent flows and flow causality, respectively. We first define four types of flow causality relationships to compose TCGs (Fig. 1): 1) communication relationship (CR), 2) propagation relationship (PR), 3) dynamic-port host relationship (DHR), and 4) static-port host relationship (SHR). The first two, CR and PR, are relationships from a flow going to a host (i.e., an IP address) to a flow coming from the host, representing that a flow causes a corresponding flow. CR is a one-to-one relationship from a request to its response (i.e., reverse direction of the request five-tuple). PR is a many-to-many relationship in which one flow propagates information into another flow, such as proxy and relay. The other two, DHR and SHR, are many-to-many relationships between flows coming from an identical host. DHR is the relationship 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. SHR is the relationship 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 relationship of flow causality such as relationship between flows with the same `dstIP` and different `srcIPs`, but we focus on these four relationship types as characteristics representing application behavior.

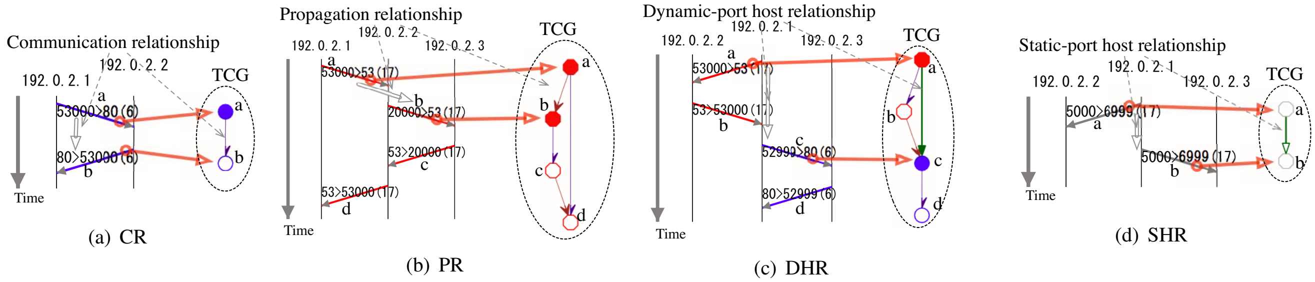


Fig. 1. Examples of four types of flow causality relationships and their TCG visualizations. Each vertex and edge in a TCG represents a flow and relationship between two flows, respectively. Number in parentheses denotes `proto`; i.e., 1 for ICMP, 6 for TCP, and 17 for UDP. Note we treated all flows as outbound for the TCG visualization.

A TCG can be visualized as a directed graph. The shape of vertices represents `proto` and flow direction; we use triangles for outbound ICMP, inverted triangles for inbound ICMP, double circles for outbound TCP, single circles for inbound TCP, double bordered octagons for outbound UDP, and octagons for inbound UDP[1]. For the visualization of edges, we use half arrowheads in indigo, double-headed ones in brown, filled ones in green, and open ones in green for CR, PR, DHR, and SHR, respectively.

Examples of these four types of flow causality relationships with their TCGs are shown in Fig. 1; the details of the algorithm will be introduced in the following subsections. Note that we treat all flows as outbound for the visualization, ignoring the real flow direction. Figure 1(a) shows an example of CR with simple server-client communication through HTTP. The client `192.0.2.1` sends a request to the server `192.0.2.2`, and then the server replies. These flows are related to each other because the response is initiated by the request. Figure 1(b) shows an example of PR with a DNS request and response through a DNS cache server. The client `192.0.2.1` sends a request to the cache server `192.0.2.2`, and then the cache server relays the request to the authoritative server `192.0.2.3`. The flow from the cache server to the authoritative server is caused by the flow from the client (i.e., the original request), so these flows are related. Figure 1(c) shows an example of DHR with single Web page access using DNS and HTTP. The client `192.0.2.1` first resolves the name from the DNS server `192.0.2.2` and then requests the resolved Web server `192.0.2.3`. The flow (Web request) from the client to the Web server depends on the flow (DNS request) from the client to the DNS server because DNS lookup is required before access to the Web server. Figure 1(d) shows an example of SHR with host scan activities from a host with one static source port number. The host `192.0.2.1` scans the host `192.0.2.2`, and then scans the host `192.0.2.3`. This host scanning fixes an identical port through a sequence of scan activities.

Algorithm 1 Get the type of relationship between flows f_1 and f_2

```
procedure getRelationship($f_1$, $f_2$, $\tau$):
 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 COMMUNICATION_RELATIONSHIP
 6: else if dstIP($f_1$) = srcIP($f_2$) then
 7:     return PROPAGATION_RELATIONSHIP
 8: else if srcIP($f_1$) = srcIP($f_2$) and srcPort($f_1$) $\neq$ srcPort($f_2$) then
 9:     return DYNAMIC_PORT_HOST_RELATIONSHIP
10: else if srcIP($f_1$) = srcIP($f_2$) and srcPort($f_1$) = srcPort($f_2$) then
11:     return STATIC_PORT_HOST_RELATIONSHIP
12: else
13:     return Nil
14: end if
end procedure
```

B. Phase 1: Connecting related flows

We compose a TCG from a set of flows with the timestamp of the flow head and five-tuple parameters by connecting related flows according to simple rules. These rules almost exactly correspond to the definitions of flow relationships in §III-A. We use six parameters, i.e., the five elements of the five-tuple and the timestamp, for the TCG composition. 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. The function timestamp(f) returns the start time of the flow f, i.e., the timestamp of the first packet of the flow.

The algorithm to determine the type of relationship between any two flows is 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). The threshold is defined as a global constant value in the algorithm. The threshold works independently of the edge reduction rules explained in §III-C, and it is used to limit the edges generated in order to reduce computational complexity. Then it checks the CR (lines 4–5), PR (lines 6–7), DHR (lines 8–9), and SHR (lines 10–11), in that order. If the input two flows have no relationship, the algorithm returns

[1] It may be difficult to distinguish octagons from circles depending on the resolution. In this case, the shape can be ignored because transport layer protocols are less significant than transport layer ports, which are indicated by color for the visualization.

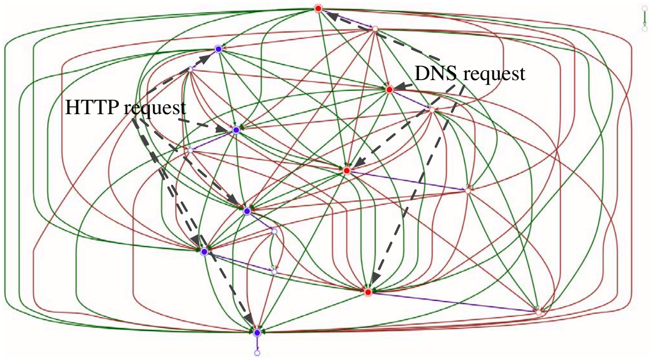


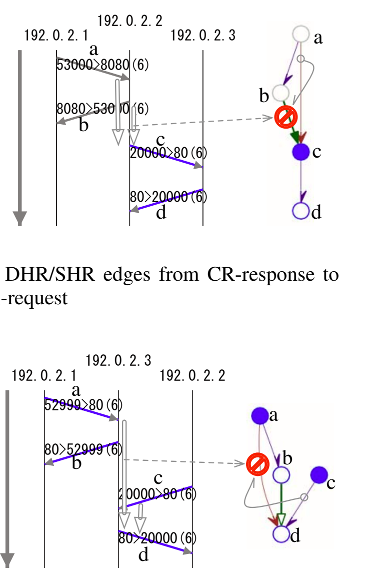
Fig. 2. TCG by Phase 1 without edge reduction (Packet trace: Web access to page http://www.google.com/ by Microsoft Internet Explorer; $\tau = 1$[s])

Nil (lines 12–13). If the algorithm returns a non-Nil value, an edge from f_1 to f_2 labeled with the returned relationship is added to the TCG.

Here, we define the terminology for TCG edges: *CR-request* and *CR-response* are the source and destination vertices of a CR edge, respectively. In the same way, *PR-source*, *PR-destination*, *DHR-source*, *DHR-destination*, *SHR-source* and *SHR-destination* are the source and destination vertices of a PR edge, a DHR edge, an SHR edge, respectively.

C. Phase 2: Reducing edges by heuristics

Figure 2 demonstrates a TCG composed in Phase 1 from a packet trace. Clearly this figure indicates problems with the simple algorithm in Phase 1. Flow causality is difficult to understand from the visualization of this TCG, and consequently, only significant edges should be retained. Algorithm 1 produces TCGs with two problems: 1) too many PR, DHR, and SHR edges, most of which do not represent direct causality, and 2) irrelative edges due to the simplicity of the algorithm. The former problem occurs because PR, DHR, and SHR are many-to-many relationships and an identical host generates several flows within the threshold. We call these edges that do not represent direct causality *tenuous edges*. The latter problem is caused simply because the Phase 1 algorithm is based on simple rules thus generates any possible edges even if some edges indicate indirect causality. For example, CR-responses can also be PR-destinations according to the algorithm, but they should not be because CR-responses are obviously caused by corresponding CR-requests but not by PR-sources. We call these edges that should be removed *irrelative edges*. We remove irrelative edges by looking at neighboring edges. Note that edges that do not represent indirect causality are related (i.e., not irrelative). In addition to these two problems, we may want to remove *insignificant or uninteresting edges*. For example, when we focus on flow causality within a client host, DHR/SHR edges from any CR-response to any CR-response are not of interest because these responses are server activities and can be removed. Therefore, we introduce three heuristic edge reduction rules (ER-Rules) to solve these two problems and to remove insignificant or uninteresting edges.

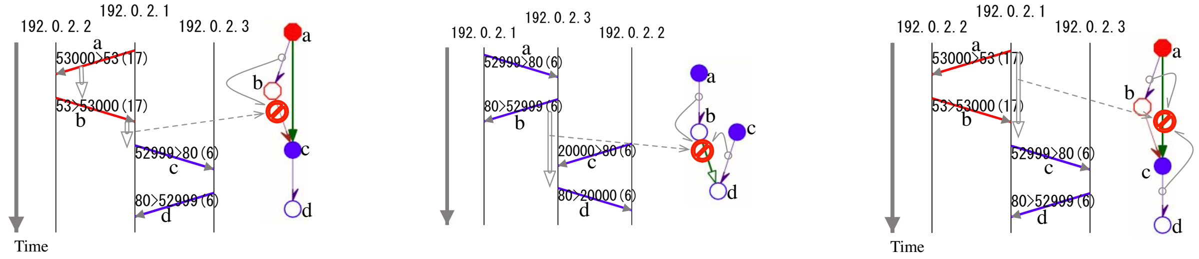


(a) DHR/SHR edges from CR-response to CR-request

(b) PR edges to CR-response and DHR/SHR edges from CR-request to CR-response

Fig. 3. ER-Rule 2: Removing irrelative edges

- ER-Rule 1. *Removing tenuous edges:* To reduce PR, DHR and SHR edges, we simply remove all PR, DHR, and SHR edges except for the temporally closest one for each relationship, i.e., the maximum out-degree for each relationship is one. This rule is based on a heuristic that the temporally closest flows are generated by the same applications and represent direct causality.
- ER-Rule 2. *Removing irrelative edges:* We remove irrelative edges by looking at neighboring edges. DHR/SHR edges from any CR-response to any CR-request should be removed because there must be PR edges as shown in Fig. 3(a) and the CR-responses are not the initiators of the CR-requests, i.e., PR-sources are the initiators of the CR-requests, which are the same as the PR-destinations. DHR/SHR edges from any CR-request to any CR-response and PR edges to any CR-response should also be removed because CR-responses are initiated only by CR-requests, as shown in Fig. 3(b). We keep DHR/SHR edges from a CR-response to another CR-response as a server activity, though they can optionally be removed by ER-Rule 3(b).
- ER-Rule 3. *(a) Removing insignificant edges (PR edges from CR-responses):* We can remove PR edges from any CR-response if we consider that the PR-destination is not initiated by the PR-source but by the *original* CR-request, as shown in Fig. 4(a). *(b) Removing uninteresting edges (server activities):* When we focus on client activities, DHR/SHR edges from any CR-response to any CR-response are not of interest because they represent server activities. Therefore, we can optionally remove them as shown in Fig. 4(b). *(c) Removing uninteresting edges*



(a) PR edges from CR-responses

(b) Server activities

(c) Client activities

Fig. 4. ER-Rule 3: Removing insignificant or uninteresting edges

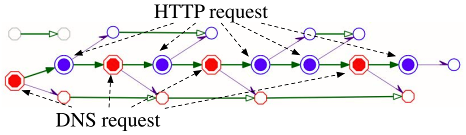


Fig. 5. TCG with Phase 2 edge reduction (Packet trace: same used in Fig. 2; $\tau = 1$[s]; ER-Rules: 1, 2, and 3(a)); color indicates difference of application protocols based on port numbers (red for DNS (UDP/53) and blue for HTTP (HTTP/80)).

(client activities): Likewise, when we focus on server activities, DHR/SHR edges from CR-requests to CR-requests are not of interest and can optionally be removed as shown in Fig. 4(c). Note that the ER-Rule 3(c) is rarely applied because client activities are more significant information on an application's activities.

Figure 5 shows a TCG from the same packet trace used in Fig. 2 after ER-Rules 1, 2, and 3(a) were applied. This figure clearly depicts that the Web browser looks up the domain name just before HTTP access. Thus, applying ER-Rules improves the expressivity of flow causality as well as the visualization. For a detailed analysis, the removed edges may indicate discriminative characteristics of network applications, but we consider that they are less important than the retained edges.

IV. Evaluation

We demonstrate the effectiveness of TCGs for application profiling by using real packet traces. We first visualize TCGs of ground truth packet traces to show the significance of the proposed method because the visualization enables us to intuitively analyze the temporal and spatial causality of flows. We then show the profiling results obtained using simple TCG features. To create ground truth packet traces, we captured packet traces of four Web browsers (Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, and Opera), three P2P file sharing applications (BitTorrent, LimeWire, and Perfect Dark), and one P2P video streaming application (BBbroadcast) at clean-installed operating systems (Windows XP SP3). To create other ground truth packet traces from applications running on the Internet, we also captured actual traffic at a laboratory-level network gateway. We used two traces, SSH brute force attacks and Web proxy, that were manually classified with a server log. We summarize the ground truth packet traces of the application programs and the method for obtaining these traces in TABLE I.

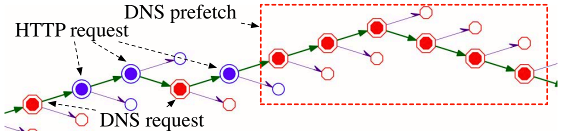


Fig. 7. Part of TCG of actual traffic from/to a host ($\tau = 1$[s]; ER-Rules: 1, 2, and 3(a, b)). Confirmed as Firefox trace by manual inspection.

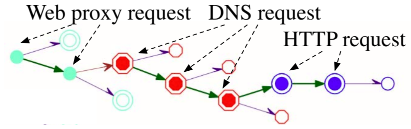


Fig. 8. Connected component of TCG of Web proxy packet trace ($\tau = 1$[s]; ER-Rules: 1, 2, and 3(a, b)).

A. Case studies

We first focus on the characteristics of four major Web browsers. We browsed the same pages with each browser in the same order. Figure 6 shows parts of the TCGs of these browsed packet traces. We used $+\infty$ for the threshold τ and applied ER-Rules 1, 2, and 3(a) to compose these TCGs. From these visualized TCGs, we can see one interesting and discriminative activity that Mozilla Firefox and Google Chrome send DNS queries to resolve domain names before they are actually required, while Internet Explorer and Opera do not. This activity is the so-called *DNS prefetch*, and it is applicable to application program profiling. Figure 7 shows a part of the TCG of actual traffic captured at the laboratory-level gateway. This trace indicated similar Web access activity to a *prefetch-enabled* Web browser. We confirmed that it was Firefox by manual inspection. Thus, the temporal causality of flows is advantageous for profiling application programs, i.e., user agents of Web browsers.

A connected component of the TCG of the Web proxy packet trace is shown in Fig. 8. Unlike the TCGs of Web browsers, there is a PR edge before the DNS request, meaning

TABLE I
GROUND TRUTH PACKET TRACES

Application program	Version	Application class	Trace
Microsoft Internet Explorer	8.9.6991.18702 (Update Versions: 0)	Web browser	Browsed several web pages
Mozilla Firefox	3.6.10	Web browser	Browsed several web pages
Google Chrome	6.0.472.62	Web browser	Browsed several web pages
Opera	10.63 (Build 3516)	Web browser	Browsed several web pages
BitTorrent	7.0 (Build 21591)	P2P file sharing	Launched and downloaded a file
LimeWire	5.5.14	P2P file sharing	Launched and downloaded a file
Perfect Dark	1.0.6	P2P file sharing	Launched and sent queries
BBbroadcast	6.0.11.9232	P2P video stream	Joined and viewed a sample video streaming
SSH brute force attacks	N/A	Actual traffic	Captured at gateway / Manual inspection
Web proxy	N/A	Actual traffic	Captured at gateway / Manual inspection

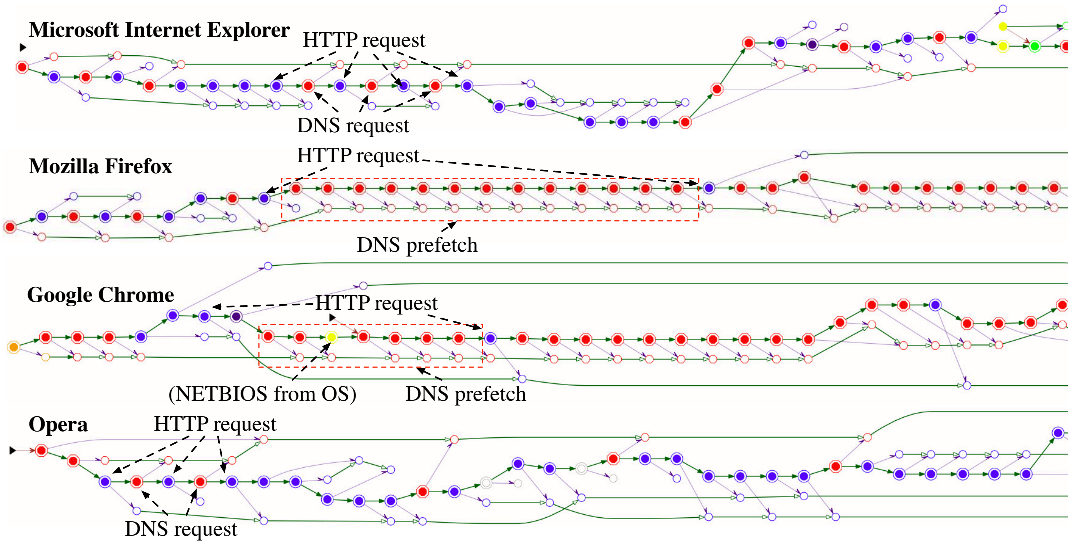


Fig. 6. Parts of TCGs of well-known Web browsers ($\tau = +\infty$; ER-Rules: 1, 2, and 3(a)).

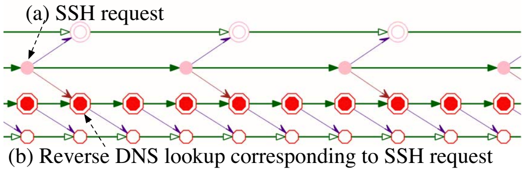


Fig. 9. Part of TCG of SSH brute force attacks ($\tau = 1$[s]; ER-Rules: 1, 2, and 3(a)).

that the sequences of DNS requests and HTTP requests are invoked by Web proxy requests. DNS requests are also invoked in some applications to validate IP addresses or to log host names. For example, HTTP servers look up the host name (i.e., reverse DNS lookup) to store it in a log file. We can also observe this activity in an SSH login procedure because SSH servers validate the host name of SSH clients attempting to login. Figure 9 shows a part of the TCG of SSH brute force attacks. We confirmed that the server resolves the reverse name of the client after receiving SSH login requests. A difference between the Web proxy and SSH brute force attacks is that another flow (i.e., HTTP) follows after DNS lookup in the Web proxy, while SSH brute force attacks generate only DNS flows. An insight from these TCGs is that TCGs can reveal the root causes of flows (e.g., Web proxy request and SSH login request). Identifying the root causes is an important task for network operation. because individual flows do not have discriminative information on whether these flows are proxied/relayed or original ones. For example, flows following a proxy request or an SSH login request are similar to those of a (prefetch-disabled) Web browser or a domain name lookup tool (e.g., nslookup), i.e., normal application programs.

B. Application to traffic classification

To achieve automated profiling of network applications using TCGs, we introduce two simple features: 1) the ratio of the number of PR edges to that of CR edges in a TCG (PR-CR), and 2) the ratio of the number of DHR edges from DNS request to DNS request to that of all DHR edges in a TCG (DNS-DHR). The feature PR-CR represents the activity

TABLE II
Results: Features of TCGs ($\tau = 1$[s]; ER-Rules: 1, 2, and 3(a))

Application program	#Edges	PR-CR	DNS-DHR
Microsoft Internet Explorer	270	0.029	0.077
Mozilla Firefox	739	0.008	0.600
Google Chrome	1161	0.025	0.580
Opera	516	0.020	0.034
BitTorrent	3444	0.595	0.025
LimeWire	4803	0.320	0.058
Perfect Dark	905	0.345	0.000
BBbroadcast	373	0.246	0.006
SSH brute force attacks (60 s)	619	0.259	0.586
Web proxy(5 s)	3668	0.614	0.100

of reactions caused by other flows, such as proxy, relay, and IP address validation with DNS. The feature DNS-DHR represents the activity of consecutive DNS requests, such as DNS prefetch in Web browsers.

We composed TCGs with $\tau = 1$[s] and ER-Rules 1, 2, and 3(a) for each packet trace, and then we calculated the features of the composed TCGs. We show the calculated features in TABLE II as preliminary profiling results. They show that the DNS-DHR of *prefetch-enabled* Web browsers (underlined values) is higher than that of *prefetch-disabled* ones. This is because a prefetch sends DNS queries in clusters, and DNS-DHR performs well for characterizing this activity. The results also show that P2P applications, SSH brute force attacks, and the Web proxy have higher PR-CR than client applications (i.e., Web browsers). Since P2P applications relay queries or contents and communicate with other hosts more frequently than client applications, PR-CR becomes higher. Similarly, SSH brute force attacks and the Web proxy have higher PR-CR than client applications (underlined values). A difference between the SSH brute force attacks and the Web proxy is that the attacks also have higher DNS-DHR, while the Web proxy does not (underlined value). As Fig. 9 confirms, the SSH brute force attacks invoked three DNS queries for one SSH login request, so DNS-DHR became higher. However, since this Web proxy performed like a prefetch-disabled Web browser after proxy requests as shown in Fig. 8, DNS-DHR became smaller.

In summary, the results highlighted that flow causality is effective for profiling network applications. Even the simple features investigated here have potential for profiling applications because they differ by applications. However, more experiments and analysis with datasets are required.

V. Discussion

TCG composition: The ER-Rules may inaccurately remove some PR edges. For example, PR edges of P2P applications may not represent the actual relation. Since P2P applications receive queries from other hosts while also sending queries to other hosts, any two flows are eventually connected depending on the threshold τ. Moreover, since current multitask operating systems may simultaneously run multiple applications, flows from different applications may be inaccurately connected. We believe that the proposed TCG composition method provides extensive information for profiling applications, but the edge reduction algorithm could be improved. To do this, we will consider extending the flow dependency extraction method proposed in Ref. [21] to the TCG composition method in order to divide a TCG into several components. The threshold τ in the edge connection algorithm should also be discussed. This threshold is less important than the edge reduction rules because tenuous edges are removed in the edge reduction procedure independently of the threshold. However, it should be evaluated in the future.

Profiling features: We used two simple features of the TCG profiling results for profiling applications. However, we realize that there are other possible features, such as graph properties, and that each flow also has statistical features, such as flow size and probability distribution of packet inter-arrival time. We can use these additional features to profile applications more accurately. Conversely, we can discard packets except for TCP-SYN packets for TCP if we do not use the other features of TCGs related to flow statistics. Here, we explain an example of the use of additional features. According to the case studies, both the TCG of SSH brute force attacks and that of the Web proxy have PR edges. However, the activity of the SSH brute force attacks is not similar to that of the Web proxy because the contents of the invoked flows in the attacks are not related to the original SSH requests. In contrast, in the Web proxy, the contents of the invoked flows are strongly related to the original requests. We could distinguish these differences by adding statistical features as profiling features because original flows and proxy flows should have similar characteristics.

Restriction: We have focused on edge networks such as company and campus networks, and have assumed that packet traces are captured at the gateway and that there are no in-network proxy servers, such as NAT gateway, DNS cache servers, and Web proxy servers, inside the networks. These in-network proxy servers cause problems with TCGs. For example, if there is a DNS cache server inside the network, DNS flows from hosts in the network will never go through the traffic monitor because these hosts send queries to the DNS cache server. Addressing this problem remains as future work. The applicability of our approach to large networks is also undetermined. In our evaluation, we used two packet traces captured at a single host and at a gateway of a small network ($\sim$/24 IPv4 address space). Figures 7, 8, and 9 were obtained from a trace of the small network, but we have not yet evaluated the proposed approach with traces of large networks. We will evaluate the applicability of our approach to large networks and its scalability in the future. Since the TCG composition algorithm runs in the order of n^2, where n is the number of input flows, the scalability must be evaluated.

DNS for traffic classification: Wu et al. [22] focused on DNS activities to identify the use of P2P applications. Their approach analyzes the characteristics of the control plane protocol (i.e., DNS) rather than data plane protocols (e.g., HTTP and P2P). Unlike other application protocols, DNS usually uses a statically bound port number (i.e., UDP/53) and no other ports can be available in DNS trees on the Internet because authoritative servers listen only on UDP/53 or TCP/53 for DNS queries. We also consider that DNS is one of the best discriminative protocols to use in identifying applications, although the Wu et al. approach focused only on identifying P2P applications. This is why we introduced DNS (i.e., UDP/53) as a feature for TCGs.

Applications of TCGs and impact: TCGs illustrate temporal and spatial communication patterns with flow causality. Our approach is quite different from existing graph-based approaches [15], [16], [17], [19]. These existing approaches profile application traffic at the host level (i.e., IP address level) but not at the application program level (i.e., set of flows level). The authors of Ref. [19] indicate that hosts with multiple application protocols do not have to be considered because only a small number of hosts with multiple application protocols are observed at backbone links. Our main target is edge networks, such as company and campus networks. Thus, we introduce flow causality to take into account application programs that use multiple application protocols. Here, we show an example of applications of TCGs to network management and the impact. In Fig. 8, DNS and HTTP requests are initiated by proxy requests, but it is difficult for network operators to judge whether these DNS and HTTP requests are original requests (i.e., without a proxy) because individual flows do not represent the proxy function. In this case, when network operators detect an anomaly in a HTTP response to the proxy client, network operators possibly identify the cause at the proxy server although the proxy server simply relays the original response. However, TCGs can extract the root cause by profiling application programs with flow causality, i.e., network operators can identify the root cause and the position of the detected anomaly if they know the traffic is a part of Web proxy traffic and proxy servers propagate flows to other hosts. Thus, it is important to profile application programs by flow causality, although existing traffic classifiers do not.

VI. Conclusion

We proposed TCGs for analyzing temporal and spatial flow causality to profile network applications without payload inspection. The TCG composition method with four types of flow causality relationships was discussed, as was the concept of network application profiling. We presented case studies to show the advantages of flow causality for profiling application behavior. The results of these case studies demonstrated the effectiveness of TCGs for profiling network application classes and programs. The main contribution of this paper is that simple features of TCGs are effective for profiling application programs as well as application classes. Existing approaches cannot easily profile application programs. Thus, TCGs are a step towards effective network application profiling. One practical use of application program profiling is to identify a specific application program that uses the same protocol as other programs but has security problems. In addition to the simple TCG features, the visualization of TCGs reveals the causality of each flow, which consequently helps network operators to identify the root causes of other flows.

We will look into other features of TCGs with advanced analysis methods such as pattern matching and graph mining. Although this paper has focused on profiling application classes and programs, we will also evaluate content-related characteristics such as poor and rich content Web sites with TCG features as well as features of other applications.

References

[1] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy, “Transport layer identification of P2P traffic,” in *ACM IMC ’04*, pp. 121–134, 2004.

[2] Sourcefire, Inc., “Snort.” http://www.snort.org/.

[3] J. Levandoski, E. Sommer, and M. Strait, “Application Layer Packet Classifier for Linux.” http://l7-filter.sourceforge.net/.

[4] OpenDPI.org, “OpenDPI.” http://www.opendpi.org/.

[5] S. Sen, O. Spatscheck, and D. Wang, “Accurate, scalable in-network identification of P2P traffic using application signatures,” in *WWW ’04*, pp. 512–521, 2004.

[6] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, “ACAS: Automated construction of application signatures,” in *ACM SIGCOMM MineNet Workshop ’05*, MineNet ’05, pp. 197–202, 2005.

[7] A. W. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” in *PAM ’05*, vol. 3431, pp. 41–54, 2005.

[8] A. W. Moore and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” in *ACM SIGMETRICS ’05*, pp. 50–60, 2005.

[9] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, “Traffic classification through simple statistical fingerprinting,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 5–16, 2007.

[10] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, “Identifying and discriminating between web and peer-to-peer traffic in the network core,” in *WWW ’07*, pp. 883–892, 2007.

[11] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho, “Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures,” in *ACM SIGCOMM LSAD ’07*, pp. 145–152, 2007.

[12] I. Trestian, S. Ranjan, A. Kuzmanovi, and A. Nucci, “Unconstrained endpoint profiling (Googling the Internet),” in *ACM SIGCOMM ’08*, pp. 279–290, 2008.

[13] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “BLINC: Multilevel traffic classification in the dark,” in *ACM SIGCOMM ’05*, pp. 229–240, 2005.

[14] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos, “Profiling the end host,” in *PAM ’07*, vol. 4427, pp. 186–196, 2007.

[15] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, “Network monitoring using traffic dispersion graphs (TDGs),” in *ACM IMC ’07*, pp. 315–320, 2007.

[16] M. Iliofotou, M. Faloutsos, and M. Mitzenmacher, “Exploiting dynamicity in graph-based traffic analysis: Techniques and applications,” in *ACM CoNEXT ’09*, pp. 241–252, 2009.

[17] Y. Jin, E. Sharafuddin, and Z.-L. Zhang, “Unveiling core network-wide communication patterns through application traffic activity graph decomposition,” in *ACM SIGMETRICS ’09*, pp. 49–60, 2009.

[18] V. Perelman, N. Melnikov, and J. Schoenwaelder, “Flow signatures of popular applications,” in *IFIP/IEEE IM 2011*, pp. 9–16, 2011.

[19] M. Iliofotou, B. Gallagher, T. Eliassi-Rad, G. Xie, and M. Faloutsos, “Profiling-by-association: A resilient traffic profiling solution for the Internet backbone,” in *CoNEXT ’10*, p. 12, ACM, 2010.

[20] L. Popa, B.-G. Chun, I. Stoica, J. Chandrashekar, and N. Taft, “Macroscope: End-point approach to networked application dependency discovery,” in *ACM CoNEXT ’09*, pp. 229–240, 2009.

[21] S. Kandula, R. Chandra, and D. Katabi, “What’s going on?: Learning communication rules in edge networks,” in *ACM SIGCOMM ’08*, pp. 87–98, 2008.

[22] H.-S. Wu, N.-F. Huang, and G.-H. Lin, “Identifying the use of data/voice/video-based P2P traffic by DNS-query behavior,” in *IEEE ICC ’09*, p. 5, 2009.