Passive OS Fingerprinting by DNS Traffic Analysis

Takashi MATSUNAKA[†], Akira YAMADA[‡] and Ayumu KUBOTA[†]
[†]KDDI R&D Laboratories Inc.
Saitama, Japan
{ta-matsunaka, kubota}@kddilabs.jp
[‡]KDDI CORPORATION
Tokyo, Japan
ai-yamada@kddi.com

***Abstract*—Network administrators want to determine which services and applications are most frequently used, which and how many devices and operating systems (OSs) are used, and when and where the highest peak of network traffic is to overcome the massive traffic demand. However, it is hard to recognize the situation in large and complicated networks. It requires massive additional monitoring nodes or systems and large volumes of traffic data analysis. Moreover, in the case of using NAT or tethering, the number of IP addresses used does not coincide with the number of devices because IP addresses is shared with devices in the behind of NAT-boxes or tethering devices.**

In this paper, we propose a new passive OS fingerprinting method which requires analyzing only DNS traffic. The method utilizes characteristics on DNS queries that each OS sends DNS queries related to specific domains, and each OS sends these queries with specific patterns of time interval between them. The method can estimate the number of devices with each OS from the number of queries by utilizing the characteristics of the time interval patterns. The method considers the likelihood of irregular events that some queries are sent less than regular time intervals, and some other queries are sent more than regular time intervals. According to our examination on our intra-network, some results of our estimation method are close to the results of DHCP fingerprinting.

Keywords-Passive OS fingerprinting; Traffic analysis

I. INTRODUCTION

In recent years, data traffic has increased explosively due to the increase in the number and use of smartphones. To overcome the massive traffic demand, network administrators must perceive the status of their networks to ensure stable network service. Network administrators want to determine which services and applications are most frequently used, which devices and operating systems (OSs) are used, and when and where the highest peak of network traffic is. In particular, the most important and useful factor is to recognize the trend in the distribution of operating systems in use in terms of network management. According to Ericson's report [1], different OSs have different trends in traffic volume. Additionally, different OSs have different applications installed. However, it is hard to perceive the status since the network is more complicated due to the diverse access networks (wire (e.g. FTTH (Fiber To The Home), xDSL (Digital Subscriber Line)) or wireless networks e.g. cellular, WLAN (Wireless Local Area)), traffic off-loading from mobile networks to fixed ones (e.g. via WLAN (Wireless Local Area Network)), and tethering by smartphones or mobile routers.

Previous works studied ways to infer network status. In [11,12], they profiled user activities or classified the traffic on the network. In [2,3], they studied ways to detect an OS (OS fingerprinting) by monitoring traffic on the network. For example, OS fingerprinting is realized by using characteristics in the TCP/IP header [2], fields in the DHCP packets [3], and the HTTP header. Some works took another approach to actively detect an OS by sending or injecting configured packets to the target hosts or TCP/IP sessions [4,6]. Another work used a hybrid approach [10] that combined passive approaches with active ones. However, these works are unrealistic for large, complicated networks in terms of storage and computational cost. These works, except for [3], force network administrators to deploy massive additional monitors or systems on their networks and to analyze large volumes of traffic data to profile all activities. The works utilizing DHCP packets [3] cannot extract additional information related to user activities. Some works of reducing the monitoring nodes for network management and monitoring are to adapt dynamic networks, such as virtual networks, the Internet, or sensor networks by selecting appropriate nodes [7,8,9]. However, these works also have the deployment issue; these works need to improve or replace existing network devices or nodes to add new functions. These works also require large volumes of data, making it difficult to extract information on network status in terms of not only the volume of traffic but also services and application trends. Furthermore, all previous works have difficulty in estimating the number of devices with each OS in the case that devices are located in the behind of NAT (Network Address Transform) boxes or tethering devices, or devices move across access networks by traffic off-loading from the cellular network to fixed one via WLAN, where a device is assigned with different IP addresses by access networks it uses, or a device shares an IP address with other devices.

To overcome such a difficulty, we focus on DNS traffic as a tool to be aware of the situation on a network. Analyzing DNS traffic results in a substantial amount of useful information about the status of the network, such as popular services and applications among users and daily traffic trends. Furthermore, it allows us to presume upcoming traffic since a user's device first sends a query to a DNS server to resolve the IP address of a service provider. Moreover, we argue that we can effectively realize awareness of the network status by simply monitoring DNS-related traffic without additional systems or monitoring points. Then, this is a suitable and realistic solution for large and complicated networks.

In this paper, we propose a new passive OS fingerprinting method by analyzing DNS traffic. The method utilizes characteristics on DNS queries: each OS has specific queries for domains to which other OSs send no query, and each OS has characteristics on the time interval distribution in sending the OS-specific domain queries. The method can estimate the number of OSs from the number of specific DNS queries. In order to realize our method, we derive characteristics regarding DNS traffic by analyzing DNS queries from each OS. Our analysis shows that each OS has two important characteristics on DNS queries described above. We also devise a method for estimating the number of OSs from the number of queries by utilizing the characteristics. For the estimation, we derive an estimation equation which utilizes the characteristics of specific DNS queries and also considers the irregular time interval case that some queries are sent less than regular time intervals, and some other queries are sent more than regular time intervals. In this paper, we provide the results of our analysis against DNS queries from the Android OS and the characteristics of the queries. Furthermore, this paper shows the results of our examination on our intra-network for estimating the number of OSs by using our estimation. Some results show that our method is a close estimation of the results of DHCP fingerprinting.

A. Contribution and Outline of this Paper

In this paper, we propose a new passive OS fingerprinting method using DNS traffic. We demonstrate, for example in the case of the Android OS, characteristics for OS fingerprinting derived from DNS-related traffic analysis. We derive a method for estimating the number of OSs by using the characteristics and considering the likelihood of irregular events: sending queries much less than the regular time interval and sending queries much more frequently. We demonstrate the results of our examination of the estimation on our intra-network.

The outline of this paper is as follows. We describe the works related to our study in Section II. We summarize our proposal for the estimation in Section III. We introduce the results of our DNS traffic analysis with the Android OS and the equation for estimating the number of OS devices in Section IV. We introduce our examination of our estimation by using DNS traffic on our intra-network in Section V, and conclude this paper in Section VI.

II. RELATED WORKS

There are some works of OS fingerprinting. In [2], Zalewski uses a passive approach by monitoring differences in the TCP/IP headers, TTL (Time To Live), and MSS (Maximum Segment Size) to distinguish OSs. In the HTTP headers, the User-Agent field has information about the web browsers as well as the OSs of the users. In [5], Shah tries to distinguish HTTP server software and the OS by using information included in the HTTP responses. However, these works are not feasible on large, complicated networks, since these works need to establish traffic monitoring equipment at all network borders and requires the filtering of usable information from high volumes of captured traffic data. Moreover, especially in [2], it does not work in the case of tethering. In this case, some fields in the TCP/IP headers are usually rewritten. In [3], Kollmann uses DHCP-related packets for passive OS fingerprinting. He uses the time difference between retransmission frames or DHCP fields, such as Secs. However, there is no information about the services or applications that users enjoy in the DHCP frames. So, an additional system is needed to gather information from another traffic analysis to that from DHCP frames.

There are other works of active OS fingerprinting. In [6], Lyon uses the network scanning tool, Nmap. This tool has a remote OS fingerprinting function. Nmap sends probe packets to the target devices and monitors the response. The application then determines the OS of the target from the response packets. In [10], Gagnon takes a hybrid approach that combines the passive approaches with active approaches to increase the accuracy of OS fingerprinting. However, the method does not work when the target devices are located behind network devices, such as a firewall or NAT box. In such cases, the application is unable to send probe packets to the targets. Some works have been studied to overcome the NAT-like situations. In [13], Beverly used a passive approach to classify the traffic derived from NAT hosts with other hosts by using a naïve Bayesian classifier for the characteristic values in the TCP/IP header fields. In [4], Schulz enabled active OS fingerprinting in the tethering environment by injecting ICMP (Internet Control Message Protocol) error packets into the target client's TCP session. However, this approach required an additional system to monitor all clients networking and, especially in [4], to inject ICMP packets at the right time. Therefore, the approach is unfeasible with large, complicated networks.

Other works were studied to profile user activities by analyzing traffic. In [11], Xu classified Internet backbone traffic into clusters (servers/services, heavy hitter hosts, scans/exploits) with source/destination IP addresses. This approach is unrealistic for large networks because of the need to analyze the volume of traffic data to profile all activities in terms of storage and computational cost. Furthermore, there is a problem with the deployment of monitors to obtain all traffic data on a large network. In [12], Zhang tried to infer online user activities (browsing, online game, video, etc.) by analyzing MAC-level traffic on a wireless LAN and extracting the feature of data/control/management frames (data rate, frame interval time, etc.). This approach specialized in wireless LAN traffic but had a monitor deployment issue.

III. DESCRIPTION OF PROPOSED METHOD

Figure 1 shows our assumption of the network environment for passive OS fingerprinting. There are some access networks (cellular, FTTH, etc.) on the whole network, and each device can connect to any access network. There is a (set of) DNS server on a core network. Whichever access network a device connects to, a device sends a query to the same DNS server. We also assume that there are some devices that connect to an access network through another device, such as tethering-enabled ones or NAT-boxes. This

implies that an IP address is not used only by a certain device; it is shared by some devices.

The outline of our proposal for the estimation of the number of OSs from DNS traffic is as follows:

1. (In the experimental environment) Gather DNS traffic from each mobile OS device.
2. Extract characteristics from DNS traffic: queries for a specific domain, a specific pattern of queries (time interval between each query, query flow to complete a name resolution tasks).
3. Make a signature from the extracted characteristics.
4. (In the service network) Gather DNS traffic and estimate the number of OSs from the traffic data by using the signature.

The following sections show the result of our analysis and examination of the estimation about the Android OS as an example of proof of our proposal. In Section IV, we denote an example of our analysis for extracting characteristics from DNS traffic. Section IV shows the results with the Android OS as an example of our analysis. Through the analysis, we found characteristics for the estimation: (A) the Android OS has specific domain names for which any other OSs sends no query regularly (denoted in Section IV-A); (B) the Android OS has specific query flows where first it sends an AAAA record query, then sends an A record, and after receiving a response, it sends a PTR query for one of the IP addresses in the response (denoted in Section IV-A); and (C) the Android OS has specific patterns of queries involved with time intervals between queries (denoted in Section IV-B). We represent a general model of the signature as an equation with each parameter that represents a characteristic (time interval pattern) used for the estimation of the number of OSs (denoted in Section IV-C). In Section V, we show the results of our examination of the estimation of the number of Android OS on our intra-network.

IV. Result of our DNS traffic Analysis

To extract signatures for passive OS fingerprinting, we capture DNS-related traffic from mobile devices with each OS left without any operation and the captured traffic. We use four smartphone devices, two of which have Android 2.3, and the others have different OSs. All devices access the Internet using wireless LAN. All devices are configured to allow auto-update of applications and enable GPS functions with the other configurations set to the default. All devices have some applications installed by default. DNS-related traffic evolved from devices is captured on the DNS server on our intra-network.

A. OS-specific DNS query

We extract domain names for which any other OSs sends no query. Table I shows an example of domain names for which only the Android OS sends queries. According to Table I, the Android OS has specific domain names for which any other OSs send no query. Moreover, it is notable that when the Android OS sends a query to *clients.android.google.com* (or some other *google.com* subdomains), the OS first sends a query for AAAA records. Then, the OS sends a query for A records. Less than 200 milliseconds later, after a response has arrived, the Android OS sends a query for the PTR record of an IP address, which is in response to the A record query.

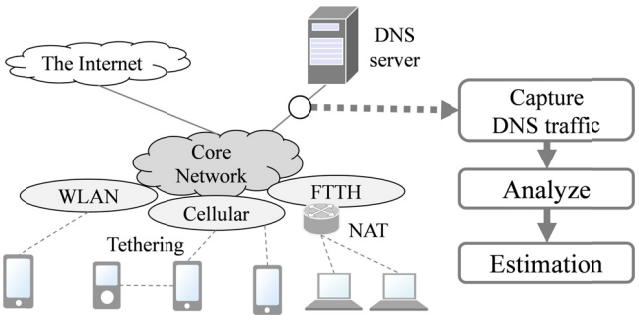


Figure 1. Our assumption of the network environment

Android OS also sends queries to **.pool.ntp.org* regularly. The domain is likely to be queried by other OSs by setting the domain as an NTP (Network Time Protocol) server domain. However, according to our extra examination by other OSs (Linux OS, Windows™), the OSs send a query only once when configuring an NTP server and send no query whenever the OSs send NTP-related traffic. The OSs take time intervals over one day because of TTL (Time To Live) value of the DNS server of *ntp.org* domain as far as we observed. It differs from the behavior of Android OS, which sends queries at about 14,400 seconds interval.

B. Interval time pattern of DNS queries

We then analyze the interval time between DNS queries for OS-specific domain. Figure 2 shows query time for *clients.android.google.com* A record from the base time when the first query (query number is 0) is evolved. Device 1 and 2 have the same Android OS version 2.3, but these are produced by different vendors. According to Figure 2, queries are often evolved at the same time with query number 0 every day. Sometimes, there is no query within a day or there are some queries at different times in a day. Device 2 sends more queries than Device 1 in 30 days. After sending a query, Device 2 sends a query after less than 3 seconds again. Moreover, Device 2 sometimes sends a query again after more than an hour (3,600 seconds).

Figure 3 shows frequency distribution of queries for *clients.android.google.com*. At Device 1, 21.4% of queries take time intervals near 86,400 seconds (from 84,600 to 88,200 seconds). 42.9% of queries take intervals over one day (more than 88,200 seconds). At Device 2, 8.7% of queries take time intervals near 86,400 seconds (from 84,600 to 88,200 seconds). A total of 13.0% of queries take intervals over one day. Moreover, Figure 2(b) shows specific characteristics Device 2 only owns, 33.0% of queries take

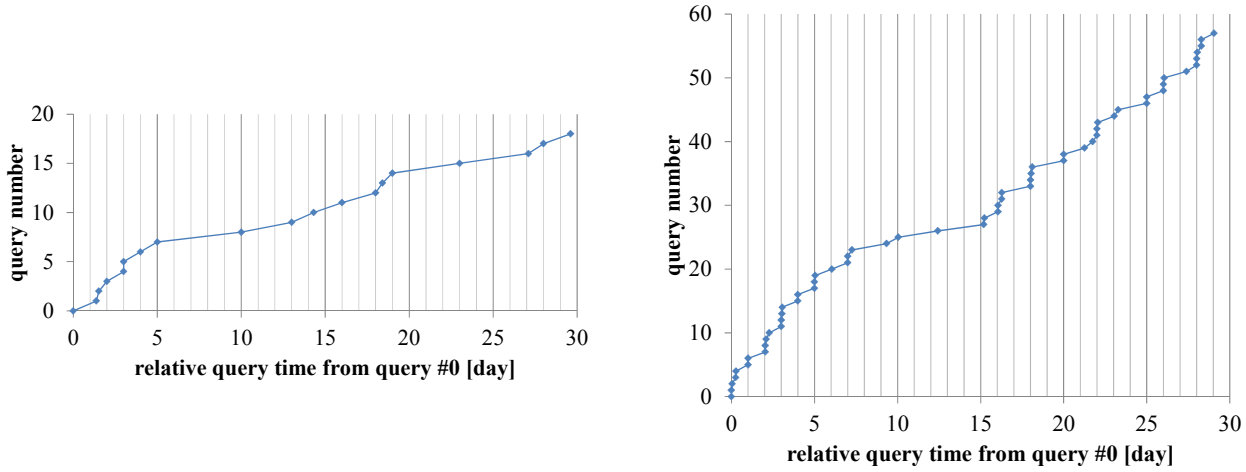
TABLE I. Examples of OS-specific DNS query (Android OS)

domain name	clients.google.com, *.pool.ntp.org, mtalk.google.com

interval less than 1,800 seconds (most of these queries send at less than 3 seconds interval) and 16.5% of queries at less than 5,400 seconds (in fact, these queries' intervals take from 3,600 to 4,000 seconds). Device 2 also takes intervals near 82,800 seconds (from 81,000 to 82,800 seconds) at 6.1% of queries. This appears that if a previous query takes intervals near 3,600 seconds, the next interval is near 82,800 seconds in order to adjust the query time at the same time in a day.

Through our analysis described above, we summarize characteristics on DNS queries for *clients.android.google.com* as follows:

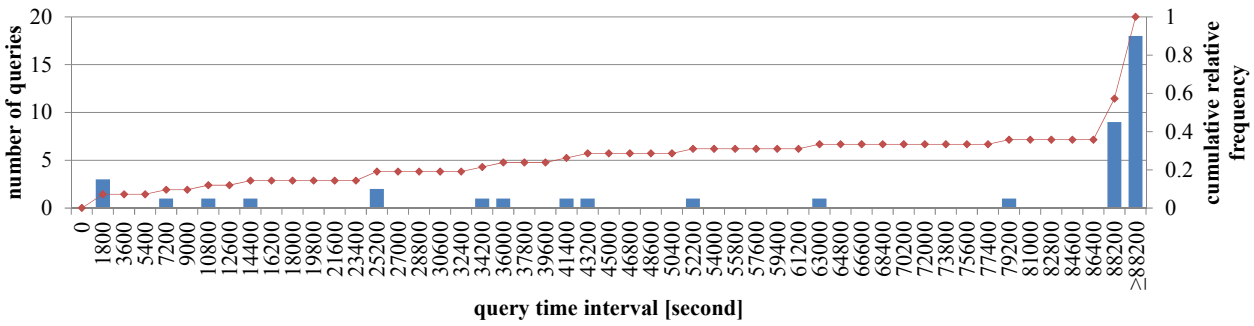
- After a query for the A record of *clients.android.google.com*, the Android OS sends a PTR query for an IP address, which is in response to an A record query at an interval of less than 200 milliseconds.



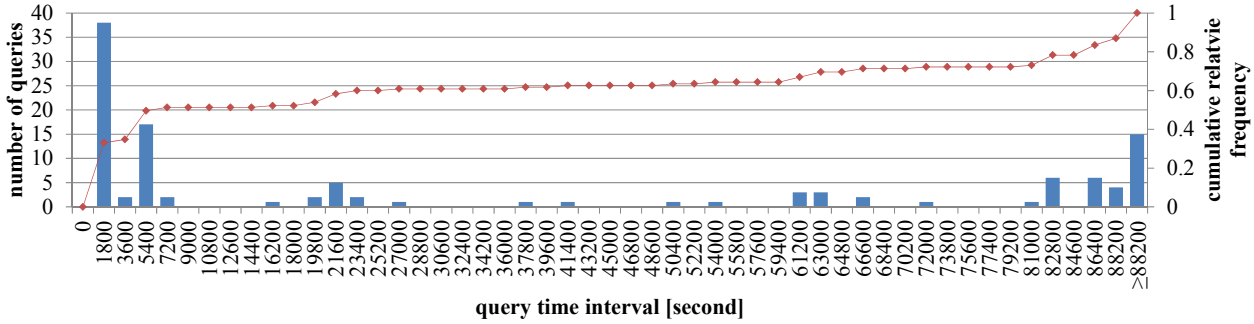
(a) Device 1

(b) Device 2

Figure 2. DNS query evolved time (domain name: *clients.android.google.com*, days: 30)



(a) Device 1

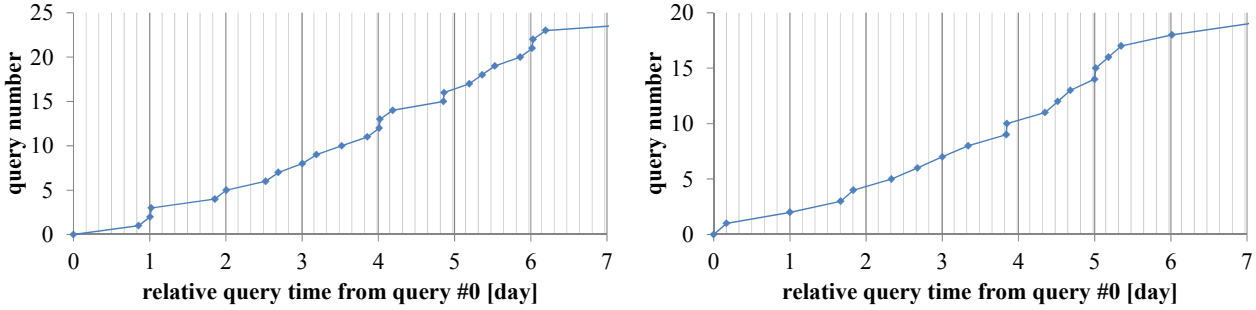


(b) Device 2

Figure 3. Query time interval (domain name: *clients.android.google.com*, days: 60)

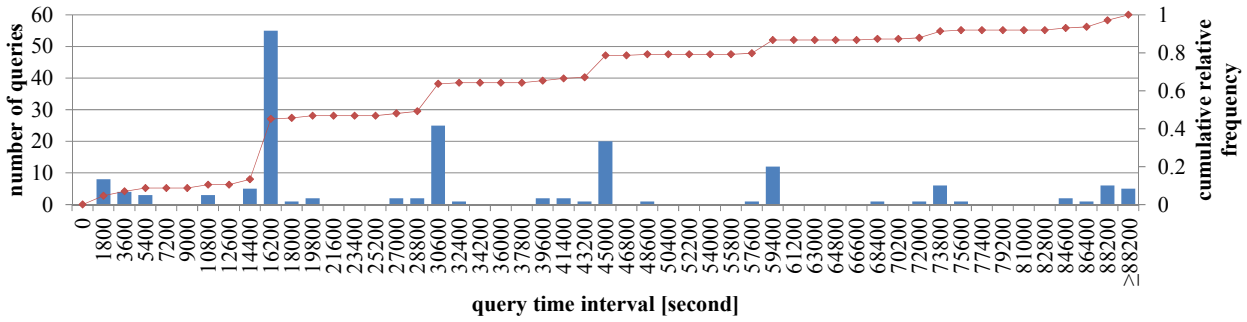
- Android OS often sends queries for *clients.android.google.com* at the same time every day. However, the Android OS sometimes sends no query in a day (42.9% of Device 1 queries, 13.0% of Device 2 queries).
- The Android OS sometimes sends queries for *clients.android.google.com* at different times from the regular time in a day. Some devices have a specific pattern for the different time (e.g. Device 2 sends the queries at intervals near 3,600 seconds (16.5%) or less than 3 seconds (33.0%)).

We analyze another query domain, **.pool.ntp.org*. Figure 4 shows query time for **.pool.ntp.org* A record from the base time when the first query is evolved. In Figure 4, vertical axes between days are drawn at 14,400 seconds. According to Figure 5, queries are often evolved at time intervals of multiples of 14,400 seconds (4 hours). Figure 5 shows frequency distribution of queries for **.pool.ntp.org*. Most of queries are sent at the time intervals of near the multiples of 14,400 seconds, 78.0% of Device 1 queries and 78.5% of Device 2 queries. Some queries take intervals less than 7,200 seconds, 8.7% of Device 1 and 9.3% of Device 2. These queries appear to be for the alignment of the timing of sending queries. Some queries take intervals over one day, 2.9% of Device 1 queries and 2.3% of Device 2 queries. Through our analysis described above, we summarize characteristics of DNS queries for

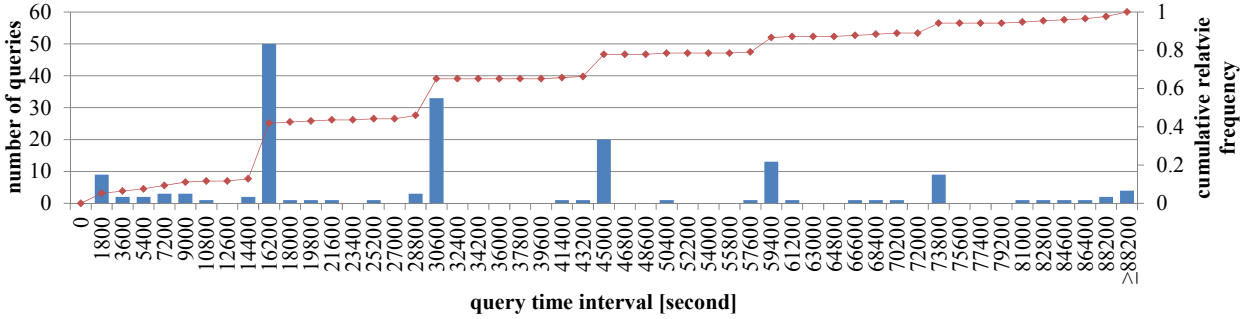


(a) Device 1 (b) Device 2

Figure 4. DNS query evolved time (domain name: **.pool.ntp.org*, days: 7)



(a) Device 1



(b) Device 2

Figure 5. Query time interval (domain name: **.pool.ntp.org*, days: 60)

clients.android.google.com as follows:

- The Android OS often sends queries for **.pool.ntp.org* at multiples of 14,400 seconds. However, the Android OS sometimes sends queries over a day (2.9% of Device 1 queries, 2.3% of Device 2 queries).
- The Android OS sometimes sends queries for **.pool.ntp.org* at less than 7,200 seconds (8.7% of Device 1 queries, 9.3% of Device 2' queries) for perhaps timing alignments.

C. Estimating the number of OSs

To estimate the number of OSs, we consider the characteristics described above: (A) regularly, query time intervals have a cyclic nature, (B) irregularly, some queries are sent less than regular time intervals, and (C) some other queries are sent more than regular time intervals. Furthermore, for estimating the number of OSs, we have to consider how to estimate the number of OSs using the data captured during the less than the regular cyclic time interval. This means that there are some OS devices that do not send queries during the captured time interval, and we estimate the number of such devices by using the captured data that includes no query sent from the devices. In this section, we first introduce how to estimate the number of OSs using the data captured during the less than regular cyclic time interval. Then, we introduce how to consider the irregular characteristics described above. For the purpose of the following explanation, Figure 6 is an example of the situation of the following explanations. Table II summarizes the notations we use in the following explanations.

First, we introduce the estimation equation which utilizing the regular cyclic nature of queries (A). Let the cyclic interval time for a domain d be T_d, and the interval time for capturing traffic data be $T_q (< T_d)$. A probability p_{d,T_q} that an OS device sends a query for domain d in the capture interval T_q satisfies $p_{d,T_q} = T_q/T_d$. Therefore, let the number of queries for domain d in the interval T_q be N_{d,T_q}, if all queries are sent at the cyclic interval T_d, the number of OSs OS_i, N_{OS_i}, satisfies $N_{OS_i} \cdot p_{d,T_q} = N_{d,T_q}$. So, N_{OS_i} can estimate by the following equation,

$$N_{OS_i} = \frac{N_{d,T_q}}{p_{d,T_q}} = N_{d,T_q} \frac{T_d}{T_q}. \quad (1)$$

For example, in Figure 6, the number of queries for domain d, N_{d,T_q}, is 3 (query $Q_{1,0}$, $Q_{1,1}$, $Q_{1,2}$ and $Q_{2,0}$). If T_q satisfies $T_q = 1/2 \cdot T_d$, the number of OS devices is estimated that $N_{OS_i} = 3/(1/2) = 6$.

Then, we introduce how to consider the irregular characteristics (B). In the queries in the captured data during T_q, there are the queries that are sent by the same OS device. So, we should remove such duplicated queries from the number N_{d,T_q} before the estimation of N_{OS_i} by the equation (1).

First, we consider the irregular characteristic (B) to the estimation equation (1). Let N^1_{d,T_q} be the number of OS devices that send only one query in the capture interval T_q, N^2_{d,T_q} be the number of OS devices that send more than one query in the capture interval T_q. Let μ^{L}_d be the mean of the number of queries sent by OS devices, which send more than one query in the capture interval T_q, in the capture interval T_q. The number of queries in the capture interval T_q, N_{d,T_q}, is denoted as $N_{d,T_q} = N^1_{d,T_q} + (\mu^{\mathrm{L}}_d - 1) \cdot N^2_{d,T_q}$. N^2_{d,T_q} satisfies $N^2_{d,T_d} = N^1_{d,T_d} \cdot p^{\mathrm{L}}_{d,T_q}$, where p^{L}_{d,T_q} is the probability that an OS device sends a query for domain d at less than the capture interval T_q. So, the number of devices that send only one query N^1_{d,T_q} is denoted as $N^1_{d,T_d} = N_{d,T_q}/(1 + p^{\mathrm{L}}_{d,T_q}(\mu^{\mathrm{L}}_d - 1))$. Therefore, the equation (1) is revised as,

TABLE II. NOTATIONS

T_d	Cyclic interval time for a domain d
T_q	Interval time for capturing traffic data
N_{d,T_q}	The number of queries for domain d that are sent in the interval T_q
N^1_{d,T_q}	The number of OS devices that sends only one query for domain d during the interval T_q
N^2_{d,T_q}	The number of OS devices that sends more than one query for domain d during the interval T_q
μ^{L}_d	Mean of the number of queries sent by OS devices in the interval T_q, that send more than one query in the interval T_q
p_{d,T_q}	Probability that a OS device sends queries for domain d in the interval T_q
p^{L}_{d,T_q}	Probability that a OS device sends queries for domain d at less than the interval T_q
p^{O}_{d,T_d}	Probability that a OS device sends queries for domain d over the cyclic interval T_d
N_{OS_i}	The number of OS devices only that send at least one query in the cyclic interval T_d
$N^{\mathrm{A}}_{OS_i}$	The number of all OS devices

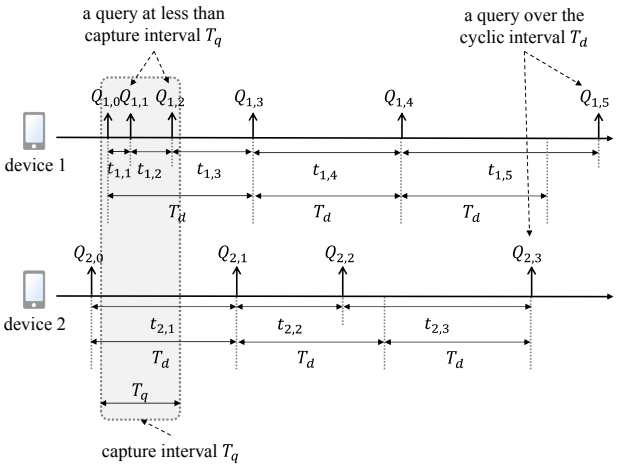


Figure 6. An example of the estimation situation

$$N_{OS_i} = \frac{N^1_{d,T_q}}{p_{d,T_q}} = \frac{N_{d,T_q}}{(T_q/T_d)(1+p^{\mathrm{L}}_{d,T_q}(\mu^{\mathrm{L}}_d-1))}. \quad (2)$$

In Figure 6, the probability that an OS device sends a query at less than the capture interval T_q, p^{L}_{d,T_q}, is $2/6 = 1/3$ derived from Device 1 pattern (queries that sent at less than the interval T_q is $Q_{1,1}$ and $Q_{1,2}$). The mean of the number of queries that is sent in the capture interval T_q, μ^{L}_d, is 3 derived from the Device 1 pattern. So, the number of OS devices is estimated as $N_{OS_i} = \frac{3}{(1/2)(1+1/3\cdot(3-1))} = 18/5$.

Then, we consider the irregular characteristic (C) to the equation (2). N_{OS_i} in the equation (2) denotes the number of OS devices that send at least one query in the cyclic time interval T_d. However, according to characteristic (C), there are some OS devices that send no query over the cyclic time interval. So, let p^{O}_{d,T_d} be the probability that an OS device sends a query over the cyclic time interval T_d, and the estimated number of all OS devices $N^{\mathrm{A}}_{OS_i}$ is denoted as follows, $N^{\mathrm{A}}_{OS_i} = N_{OS_i}/(1-p^{\mathrm{O}}_{d,T_d})$. Therefore, the equation (2) is revised as,

$$N^{\mathrm{A}}_{OS_i} = \frac{N_{OS_i}}{1-p^{\mathrm{O}}_{d,T_d}} = \frac{N^1_{d,T_q}}{p_{d,T_q}(1-p^{\mathrm{O}}_{d,T_d})}$$

TABLE III. PARAMETERS FOR THE EQUATION (3)

(A) DOMAIN: *ANDROID.CLIENTS.GOOGLE.COM*

		Device 1			Device 2		
T_d	T_q	p^{L}_{d,T_q}	p^{O}_{d,T_d}	μ^{L}_d	p^{L}_{d,T_q}	p^{O}_{d,T_d}	μ^{L}_d
86400	86400	0.357	0.429	2.00	0.783	0.130	2.86
	43200	0.262	0.429	2.00	0.626	0.130	2.34
	21600	0.143	0.429	2.00	0.539	0.130	2.05
	10800	0.095	0.429	2.00	0.513	0.130	2.05
	5400	0.071	0.429	2.00	0.348	0.130	2.00

(B) DOMAIN: **.POOL.NTP.ORG*

		Device 1			Device 2		
T_d	T_q	p^{L}_{d,T_q}	p^{O}_{d,T_d}	μ^{L}_d	p^{L}_{d,T_q}	p^{O}_{d,T_d}	μ^{L}_d
14400	14400	0.104	0.549	2.00	0.116	0.581	2.00
	7200	0.087	0.549	2.00	0.076	0.581	2.00
	3600	0.046	0.549	2.00	0.052	0.581	2.00

TABLE IV. VALUES OF THE NUMBER OF QUERIES IN EACH CAPTURED TIME INTERVAL
(DOMAIN: *ANDROID.CLIENTS.GOOGLE.COM*)

T_q	86400	43200	21600	10800	5400
N_{d,T_q}	16.0	8.58	4.61	2.29	1.14

TABLE V. VALUES OF THE NUMBER OF QUERIES IN EACH CAPTURED TIME INTERVAL
(DOMAIN: **.POOL.NTP.ORG*)

T_q	14400	7200	3600
N_{d,T_q}	1.50	0.73	0.35

$$= \frac{N_{d,T_q}}{(T_q/T_d)(1+p^{\mathrm{L}}_{d,T_q,}(\mu^{\mathrm{L}}_d-1))(1-p^{\mathrm{O}}_{d,T_d})}. \quad (3)$$

In Figure 6, the probability that an OS device sends a query over the cyclic time interval T_d, p^{O}_{d,T_d}, is 1/6 derived from Device 1 pattern (queries that sent over the cyclic time interval T_d is $Q_{1,5}$). So, the number of OS devices is estimated as $N^{\mathrm{A}}_{OS_i} = \frac{3}{(1/2)(1+1/3\cdot(3-1))(1-1/6)} = 108/25$.

V. EXAMINATION IN OUT INTRA-NETWORK

We examine our estimation equation (3) by estimating the number of Android OSs on our intra-network. We capture the DNS traffic data and DHCP-related traffic in our intra-network. DHCP traffic is used for DHCP fingerprinting [3] to compare the estimation result by using DNS traffic. In this examination, we use two DNS queries that are for *android.clients.google.com* and **.pool.ntp.org*. We derive each parameter in the equation (3) from our DNS traffic analysis described in Section III-B. Table III summarizes the parameters for the equation (3) related to queries for *android.clients.google.com* and **.pool.ntp.org*, respectively.

Figure 7 shows the difference in the estimation results by using queries for *clients.android.google.com* with the captured time interval T_q and parameters from device 1 analysis and device 2. We derive a number of queries, N_{d,T_q} in the equation (3) from the captured DNS traffic data during one day. Each value of N_{d,T_q} related to the captured time interval T_q is shown in Table IV. Each value of N_{d,T_q} is derived by calculating an average of the number of queries in each interval where the start time is shifted hour by hour. The dashed line in Figure 7 indicates the result of the DHCP fingerprinting, which estimates that 8 OS devices exist in the network. According to Figure 7, the results from the Device 2 parameters are closer to the DHCP fingerprinting result. Therefore, Device 2 parameters are more suitable for the characteristics of Android OS queries. Device 1 parameters derive worse estimation results since the probability that an OS device sends a query over the cyclic time interval p^{O}_{d,T_d} is much higher than Device 2 due to device specific characteristics or irregularly factors. Figure 7 also indicates the feature that the longer the captured time interval T_q, the closer the estimation results are to the DHCP fingerprinting result.

Figure 8 shows the estimation results by using queries for **.pool.ntp.org*. Each value of N_{d,T_q} is shown in Table V. According to Figure 8, both estimation numbers of OS devices are less than the DHCP fingerprinting result. It is because some Android OS devices send no query for **.pool.ntp.org* by default, and we presume that there are some Android OS devices that are set to choose another domain or method for time synchronization in the network. Our extra observation shows that 2 devices of 5 devices send no query for that domain. If we consider the rate of such devices to the estimation, the results of the estimation become closer to the DHCP result.

VI. CONCLUSION

In this paper, we study ways to passive OS fingerprinting from the analysis of DNS traffic and we derive a method to estimate the number of OSs in the network.

We first reveal characteristics to determine OSs from DNS traffic by analyzing DNS queries from each OS. Each OS, especially the Android OS, has useful characteristics for the estimation, each OS has specific domains to which other OSs send no query, and each OS has characteristic time interval distributions in sending the OS-specific domain queries. Our analysis also shows that the OS-specific domain queries are sometimes sent irregular time intervals; some queries are sent less than regular time intervals, and some other queries are sent more than regular time intervals.

We then propose a method for estimating the number of OSs from a number of specific DNS queries in a captured DNS traffic data during a time interval. We derive an equation for estimating the number of OSs, which considers not only the cyclic nature of queries for specific domains but also the irregular time interval cases described above.

Finally, we provide the results of our examination on our intra-network. Some results show our estimation method can result in close estimation number of OSs to the results of DHCP fingerprinting. The result indicates that the accuracy of our estimation method depends on the parameters for the equation. In the case of using DNS queries to *clients.android.google.com*, we can obtain closer estimation number of OSs by using the parameters which are derived from our DNS traffic analysis regarding Device 2 than ones regarding Device 1. Furthermore, the results also reveal the feature that the shorter the data captured time interval, the worse the precision of the estimation. Additional methods should be studied to raise the precision of the estimation from captured data with shorter time intervals and to derive the adequate parameters that correctly describe OS characteristics.

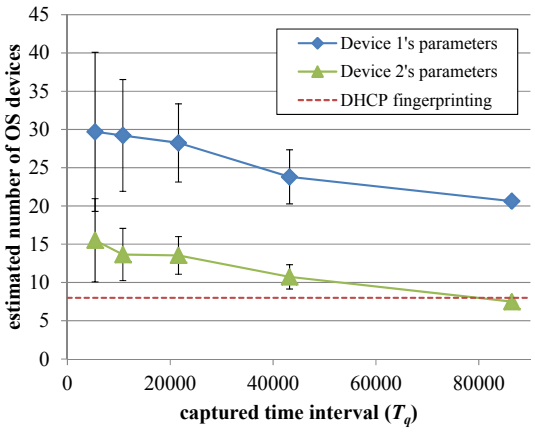


Figure 7. Estimation results with queries for *clients.android.google.com*

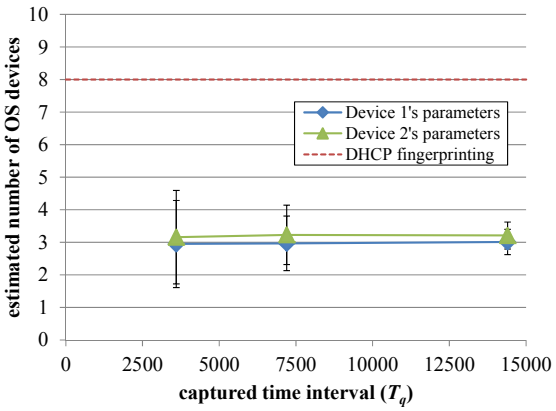


Figure 8. Estimation results with queries for **.pool.ntp.org*

ACKNOWLEDGMENT

We would like to thank Mr. Yamashita from KDDI R&D Laboratories Inc. for sharing his works.

REFERENCES

[1] Ericsson, "Traffic and Market Report", Available at http://www.ericsson.com/res/docs/2012/traffic_and_market_report_june_2012.pdf, Jun. 2012.

[2] M. Zalewski, "p0f v3",Available at http://lcamtuf.coredump.cx/p0f3/.

[3] E. Kollmann, "Chatter on the Wire: A look at DHCP traffic", Available at http://myweb.cableone.net/xnih/download/Chatter-DHCP.pdf, 2007.

[4] S. Schulz, A. Sadeghi, M. Zhdanova, H. A. Mustafa, W. Xu and V. Varadharajan, "Tetherway: A Framework for Tethering Camouflage", Proc. ACM Wireless Network Security (WISEC 2012), pp. 149-160, 2012.

[5] S. Shah, "HTTP Fingerprinting and Advanced Assessment Techniques", Blackhat 2003 USA, Available at http://www.blackhat.com/presentations/bh-usa-03/bh-us-03-shah/bh-us-03-shah.ppt, 2003.

[6] G. F. Lyon., "Remote OS Detection via TCP/IP stack fingerprinting", Available at http://nmap.org/book/osdetect.html, 2011.

[7] C. Popi, O. Festor, "A Scheme for Dynamic Monitoring and Logging of Topology Information in Wireless Mesh Networks", Proc. IEEE Network Operations and Management Symposium (NOMS 2008), pp. 759-762, 2008.

[8] D. Tuncer, M. Charalambides, G. Pavlou and N. Wang, "DACoRM: A Coordinated, Decentralized and Adaptive Network Resource Management Scheme", Proc. IEEE Network Operations and Management Symposium (NOMS 2011), pp. 417-425, 2011.

[9] R. G. Clegg, S. Clayman, G. Pavlou, L. Mamatas and A. Galis, "On the selection of management/monitoring nodes in highly dynamic networks", IEEE Trans. on Computers, vol. 99, pp. 1-15, Mar., 2012.

[10] F. Gagnon and B. Esfandiari, "A Hybrid Approach to Operating System Discovery Based on Diagnosis Theory", Proc. IEEE Network Operations and Management Symposium (NOMS 2012), pp. 860-865, 2012.

[11] K. Xui, Z. Zhang and S. Bhattacharyya, "Profiling Internet Backbone Traffic: Behavior Models and Applications", Proc. ACM SIGCOMM 2005, pp. 169-180, 2005.

[12] F. Zhang, W. He, X. Liu and P. G. Bridges, "Inferring Users' Online Activities Through Traffic Analysis", Proc. ACM conference on Wireless Network Security (WISEC 2011), pp. 59-70, 2011.

[13] R. Beverly, "A Robust Classifier of Passive TCP/IP Fingerprinting", Proc. Workshop Passive and Active Network Measurement (PAM 2004), pp. 158-167, 2004.