Automated Classifier Generation for Application-Level Mobile Traffic Identification

Yeongrak Choi
Division of IT Convergence Engineering, POSTECH
Pohang, Korea
dkby@postech.ac.kr

Jae Yoon Chung, Byungchul Park
Department of Computer Science and Engineering, POSTECH
Pohang, Korea
{dejavu94, fates}@postech.ac.kr

James Won-Ki Hong
Division of IT Convergence Engineering, POSTECH
Pohang, Korea
jwkhong@postech.ac.kr

***Abstract*—This paper proposes an automated classifier generation system for application-level mobile traffic identification. The proposed system comprises traffic classifier generation system architecture and mobile traffic measurement agents (mTMAs) that are installed on mobile devices to monitor their application-level traffic. To generate classifiers, our adaptive algorithm selects the lowest cost classifiers from among the available classifier candidates, while assuring acceptable identification accuracy. To verify the efficacy of the proposed identification system, we implemented and deployed it on a campus network. The results of mobile application traffic identification obtained using the system in this deployment are also presented in this paper.**

Keywords—Mobile application traffic identification, mobile traffic measurement and analysis, mobile application classifier, automated classifier generation

I. INTRODUCTION

Application traffic identification is an important step toward understanding network usage, providing high quality network services, deterring malicious attacks, and managing selfish-application traffic. Application-level mobile traffic identification is challenging because of the wide variety of mobile devices and their applications as well as the difference between mobile application and traditional Internet application traffic patterns. Nowadays, the number and the volume of mobile applications are rapidly increasing. Electronics manufacturers are producing various types of mobile devices, such as smartphones and smart tablets, on which people run a variety of applications. According to [1], as of March 2011 there were more than 200,000 Android and 300,000 iPhone applications available and their numbers are increasing every day.

To accurately identify mobile applications at a low cost, we need to investigate which traffic identification methods are suitable for mobile applications by considering their traffic patterns. Although current mobile traffic patterns (high percentage of HTTP traffic and smaller flow size [2][3][4]) are not as complicated as other types of Internet traffic patterns like P2P traffic, some mobile applications (e.g., Skype) can disguise their traffic or use encryption [5][6]. In this paper, we do not specify one traffic identification method for all mobile applications (i.e., we do not adopt a one-size-fits-all approach). We assume that traffic identification methods can be different for different mobile applications. Thus, we define the knowledge for mobile application identification as a classifier that represents key features of each mobile application's traffic.

One difficulty faced in the generation of this knowledge for mobile traffic identification is the collection of ground truth data for mobile applications. To collect these data, we need to install additional measurement agents on target devices [7]. However, mobile devices lack computational processing power and have limited memory compared to PCs and computing servers. These constraints make it difficult to collect ground truth data for mobile application traffic from mobile devices.

This paper focuses on a method of automated generation of classifiers that can effectively identify mobile applications. Our proposed architecture including mobile traffic measurement agents (mTMAs) can generate mobile traffic classifiers automatically and collect ground truth data for validation. Our proposed solution also includes an adaptive algorithm that automatically finds low-cost accurate classifiers by analyzing the available classifier candidates. To confirm the efficacy of our system (and algorithm), we implemented and deployed it on our campus network using mTMAs for the Android OSes.

The remainder of this paper is organized as follows. Section II describes previous research on mobile traffic identification and the categorization of classifiers used in traffic identification. Section III presents our proposed mobile traffic classifier generation architecture and mTMAs. In Section IV, we discuss our proposed adaptive algorithm for automated classifier generation, while Section V validates our proposed system and algorithm. Finally, we summarize our work and discuss some possible future work in Section VI.

This research was supported by World Class University program funded by Ministry of Education, Science and Technology through the National Research Foundation of Korea (R31-10100) and the KCC (Korea Communications Commission), Korea, under the "Novel Study on Highly Manageable Network and Service Architecture for New Generation" support program supervised by the KCA (Korea Communications Agency) (KCA-2011-10921-05003).

978-1-4673-0269-2/12/$31.00 ©2012 IEEE

II. Related Work

A. Analysis of Mobile Traffic

Previous studies measured and analyzed both mobile traffic and the characteristics of mobile networks. *Maier et al.* [2] analyzed hand-held mobile device traffic from residential broadband DSL lines and manually identified mobile traffic based on HTTP user-agent strings. The result of their analysis indicated that iPhone and iPod touch traffic dominate mobile traffic (86–97% of hand-held mobile HTTP traffic and 71–87% of the devices overall) and that HTTP is the most used protocol (80–97% of all hand-held mobile traffic by volume). *Falaki et al.* [3] analyzed smartphone traffic generated by Windows Mobile and Android smartphones. Their application-level analysis included port distribution (HTTP and HTTPS comprised almost 80% of the traffic) and per-process traffic usage (browsing comprised more than half of the traffic: 58.02%). Analysis of campus Wi-Fi network packet traces by *Gember et al.* [4] indicated that more than 90% of mobile application traffic uses HTTP and HTTPS.

The results of these studies indicated that the vast majority of HTTP and HTTPS application protocol traffic is generated by mobile device applications. However, these studies used manual analysis, which involves laborious effort and cannot provide precise or detailed application-level information on mobile traffic. Their protocol classification methods depend on port-matching, which is known to have low accuracy. Another characteristic of mobile traffic is the preponderance of inbound traffic [3][9]. The size of the responses received by most mobile devices are larger than that of requests sent, which implies that most mobile applications run as clients in the client-server architecture.

B. Classifiers in Existing Approaches

Traffic identification approaches in previous studies are categorized into three groups: session-based approaches, content-based approaches, and statistics-based approaches.

Typical session-based approaches include port matching and session behavior modeling methods. Port matching matches the port numbers in each packet or flow with well-known port numbers [10]. It guarantees only 50–70% accuracy in conventional Internet traffic [11]. Behavior modeling matches each traffic graph pattern based on flow information. One such behavior modeling method, BLINC [12], uses graphs and traffic patterns represented by a set of 4-tuples {source IP address, destination IP address, source port number, destination port number} as classifiers. However, although the authors insist that BLINC classifies 80–90% (completeness) of the traffic with more than 95% accuracy, its level of accuracy actually depends on how correctly we model traffic patterns.

Content-based traffic identification deals with signatures of applications. The signature matching method [9] maps signatures to applications or application groups. The signature matching operation, whose comparison cost is high, guarantees high accuracy if there is no flaw in the manual creation of the signatures. Automated methods for creation of signatures have been proposed to reduce the amount of effort extended in signature creation process, which currently requires major manual exertion. Our earlier work, the LASER method [2], is a modified version of the longest common sequence (LCS) algorithm that automatically ascertains the pattern in a packet's payload. The motif-finding algorithm, proposed in [8], automatically generates accurate signatures. However, the cost of operation for both methods is high. Both require a lot of computational power and a large amount of memory.

The statistics-based approach uses classifiers as a group of non-deterministic features. The classification techniques used in this approach convert flows to a predetermined number of clusters and determine classifiers according to the following constraints: port number, number of packets, flow duration, average packet size of a flow, packet inter-arrival time, etc. *Lim et al.* [14], from their analysis of different features and algorithms that can be used to traffic classification, report that this technique performs surprising well in terms of accuracy. However, many applications have yet to be verified on more complex traffic in terms of efficacy and performance (e.g., P2P file sharing).

III. Mobile Traffic Classifier Generation System Architecture

In this section, we present the architecture of our automated classifier generation system. The definitions for various terms used in our system are first presented, after which we discuss our proposed architecture including mTMAs.

A. Definition of Mobile Traffic Classifier

In this paper, we define a *classifier* as a rule for classifying traffic. A rule can express the distinguishing characteristics of the application traffic shown in captured packets or flows such as unique port numbers, a tuple of {IP, Port}, and substrings of a payload. The term *signature* is mostly used to indicate a pattern of bytes in the packet payload [13]. In accordance with this definition, classifiers can be port matching rules, signature matching rules, or even statistics-based traffic identification rules.

On the basis of the definition of classifier, we define a *mobile traffic classifier* as a collection of rules for identifying mobile application traffic. In this paper, we select the following mobile traffic rules as mobile traffic classifiers: "destination IP address" and "port numbers" reflecting client and server architecture in mobile device environments, "HTTP Host" and "HTTP User-agent" formatted fields for HTTP traffic, and the "LCS substrings" [2] focusing on a small amount of non-HTTP mobile traffic. Some types of mobile traffic rules can accurately identify certain mobile applications at a low cost, but cannot identify other mobile applications. For instance, the IP subnet rule can identify a popular instant messaging mobile app called KakaoTalk [23] at a low cost and with 100% accuracy in our work (including encrypted traffic), but it cannot identify mobile web browsers because web browsers can connect to unlimited hosts with different subnets. For mobile web browsers, the "HTTP User-agent" can be involved in a mobile traffic classifier because "HTTP User-agent" characterizes the client application to select suitable operating parameters for the session.

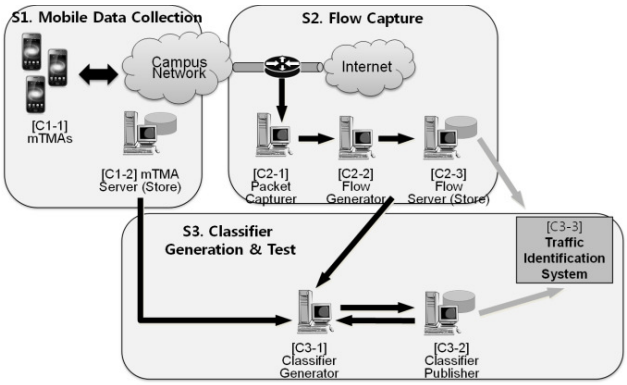

Fig. 1. Overall Mobile Traffic Classifier Generation system

B. Overall System Architecture

Fig. 1 depicts the architecture of our proposed automated classifier generation system. In our proposed system, mTMAs are installed on several mobile devices and collect ground truth, such as which mobile applications are connected to our campus Wi-Fi network and generating traffic. In the proposed architecture, the Mobile Data Collection subsystem (S1) collects application process names and partial packet information, while Flow Capture subsystem (S2) captures all packets using the Internet. The Classifier Generation & Test subsystem (S3) then unifies the information from these two components to generate mobile traffic classifiers.

In our proposed architecture, packet capture is carried out in both subsystems S1 and S2 because of the packet capture loss that occurs when we use the mTMAs. To check the packet capture loss in the mTMAs, we simultaneously captured mobile traffic using mTMAs in S1 and the packet capturer in S2, then compared the number of captured packets. Table I presents the packet loss ratio for various types of mobile applications when we execute an mTMA on a single-core smartphone (Samsung Galaxy S). The total packet loss ratio for all the mobile applications in our experiment was 62.0%, which is very high. Multimedia mobile applications had the highest packet loss, which implies that playing videos requires heavy resources of mobile devices, and mTMAs cannot capture all the packets when the packet flow rate is very high. Therefore, we utilized the data captured in S2 to make the incomplete data captured in S1 complete.

1) The Mobile Data Collection Subsystem (S1)

This subsystem is composed of mTMAs (C1-1) and an mTMA Server (C1-2). The role of this subsystem is to mark application process names on all the flows from mobile devices. First, mTMAs running in mobile devices capture all the packets (the first twenty bytes) using the libpcap library [17]

TABLE I. Packet Loss Ratio in mTMAs

Category	*Mobile Application*	*Packet Loss Ratio*			*Average*	*Standard Deviation*
		1st	*2nd*	*3rd*		
Multimedia	YouTube	72.7%	73.4%	71.2%	72.4%	0.011
Browser	Internet	27.7%	20.5%	18.4%	22.2%	0.049
Messaging & SNS	KakaoTalk, Facebook	58.7%	5.1%	4.3%	22.7%	0.312
(Total)		61.2%	65.1%	59.6%	62.0%	0.028

and store packet information and application process names separately. These results are sent to the mTMA Server periodically. Since mobile devices have low computational processing power, we do not tag the packets with application names or convert captured packets to flows in the mobile devices.

Fig. 2 illustrates the interaction between the mTMAs and the mTMA Server. We use internal socket information in the mobile operating system to obtain application process names for the generated traffic. For TCP protocol packets, we tag only the application names for TCP SYN packets in order to reduce tagging overhead which consumes a lot of resources in mobile devices. Next, the mTMA Server aggregates all the information collected from the mTMAs, matches packet information to application process names, and converts packet-level network information with application names to flow-level information.

2) The Flow Capture Subsystem (S2)

This subsystem is used in our proposed system to capture all the packets with full payload data. First, the Packet Capturer (C2-1) captures the traffic generated from all devices. The Flow Generator (C2-2) then converts packet-level information captured from our campus junctions into flow-level information. Flow-level network information includes a tuple of {source IP address, source port number, destination IP address, destination port number, protocol type}, the flow's start time, the number of packets per flow, the flow duration and the flow size in bytes. This component stores the payload for the first 10 outbound and inbound packets because the first few packets of a flow are enough to guarantee highly accurate traffic identification [18]. We ascertain the start of the flow by analyzing SYN packets and the end of the flow by observing when FIN/RST packets or flow timeout occurs [19]. Finally, the Flow Server (C2-3) manages flow-level traffic data using a database to support fast access from the Classifier Generator (C3-1).

3) The Classifier Generation & Test Subsystem (S3)

The Classifier Generator (C3-1) performs an automated classifier generation process. Inputs to this component are provided as database tables from the mTMA Server (C1-2) and the Flow Server (C2-3). This component first matches application-level information on the mTMA Server and flow-level payload information on the Flow Server using our proposed flow matching algorithm. It first automatically generates classifier candidates by enumerating packet or flow fields from matched results. Then, it checks the accuracy of each available classifier candidate to determine whether it

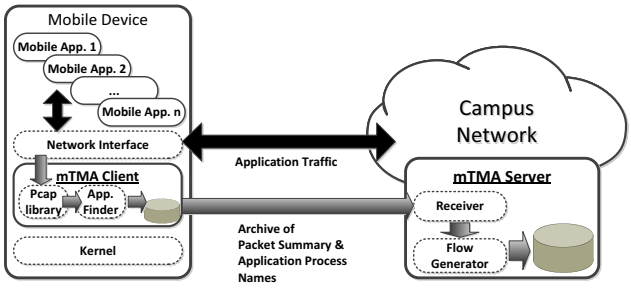

Fig. 2. Interaction between the mTMAs and the mTMA Server

Algorithm 1. Flow Matching for TCP packets

```
00:  procedure TCPFlow_Matching()
01:    for each item in M_subtuple
02:      T ← subset of T_subtuple that matches <dst_ip,
                    dst_port, dst_proto> for item within time period
03:      if # of T for exact matching <pktno, bytes> = 1 then
04:        Insert <appname, dst_ip, dst_port, flow_info, payload> into R
05:      end if
06:      else if # of T for exact matching <pktno> = 1 then
07:        Insert <appname, dst_ip, dst_port, flow_info, payload> into R
08:      end if
09:      else if # of T more or less <pktno> than item=1 then
10:        Insert <appname, dst_ip, dst_port, flow_info, payload> into R
11:      end if
12:      else
13:        Perform Match payload_p and payload for item and T
14:        if # of matched results = 1 then
15:          Insert <appname, dst_ip, dst_port, flow_info, payload>
                                                          into R
16:        end if
17:      end if
18:    end for
19:    return R
20:  end procedure
```

conflicts with other existing classifiers and whether its accuracy is acceptable or not. The automated classifier generation algorithm is required to produce classifiers that identify mobile applications.

The purpose of the Classifier Publisher (C3-2) is to organize the classifiers produced by the Classifier Generator and publish them so as to enable users to utilize the generated classifiers in various ways. Since our proposed system automatically generates classifiers for each mobile application, the generated classifiers may not be optimized to support lower cost of operations. For example, the function one of the payload classifier rules for mobile web browser identification is to match the long string (see Table IV) with the "HTTP User-agent" fields of a payload. However, only "Android 2.3.3", "SHW-M110S", and "GINGERBREAD" substrings in "HTTP User-agent" proved to be meaningful payloads for identifying this mobile application when we analyzed this payload manually. Users check the classifiers generated using this component and they can optimize the produced classifiers.

Finally, the Traffic Identification System (C3-3) uses the produced classifiers to identify mobile applications. This component carries out traffic identification using our generated classifiers. By analyzing the results coming from this component, we can validate the accuracy of our proposed classifiers and understand the usage of various mobile applications across captured packet traces.

IV. Application Classifier Generation Algorithms

In this section, we first explain the flow matching algorithm used in our proposed architecture. We then propose a classifier generation algorithm that adaptively tries and selects classification rules from low cost to higher cost rules.

A. The Flow Matching Algorithm

The flow information collected from the Internet junction can complement the incomplete information in the mTMAs. However, most Wi-Fi networks use network address translator (NAT) functionality [15]. In a NAT environment, the source IP addresses of devices are hidden behind the public IP addresses of Wi-Fi access points (APs), and source port numbers are also assigned randomly. This prevents us from precisely matching flows using 5-tuple information because source addresses and port numbers are useless. To solve this problem, we propose a heuristic matching algorithm to find corresponding mTMA traffic in the Internet junction traffic.

Because of NAT functionality in Wi-Fi APs, only the destination IP address, the destination port number, and protocols can be used to directly compare data in the mTMA Server and Flow Server. We applied a heuristic algorithm that makes use of other flow properties shown in Algorithm 1. For each item of flow information stored in the mTMA Server ($M_{subtuple}$), we try to find the corresponding mobile traffic in the Flow Server data ($T_{subtuple}$) to the extent possible. Our heuristic algorithm divides the matching flow procedures into four cases: (1) Packet and Byte matching (lines 3–5); (2) Packet matching (lines 6–8); (3) More or Less Packet matching (lines 9–11); and (4) Payload matching (lines 12–17).

For each item that contains flow information and an application process name obtained from the mTMAs, the flows that might be matched in the backbone are flows whose start time is almost the same. The algorithm first matches each item from an mTMA to candidates in the backbone using the number of packets and transferred bytes assuming that the number of packets from the mTMAs is reliable. Although packet losses can occur in the mTMA's flow, this algorithm uses the number of packets with confidence at the beginning because matching the number of packets and transferred bytes requires less computational power than matching payload bytes. If no backbone flow or more than one backbone flow is matched to one mTMA flow, then we set the transferred bytes as a non-trust value.

Wi-Fi and wired LANs use different physical layer protocols, and each such protocol can insert different additional trailer bytes for each packet. Therefore, this algorithm only trusts destination IP addresses and destination port numbers, and finds whether there is only one flow in the backbone for a row in the mTMAs. If several candidates match a row in the mTMAs, this algorithm compares the payload data between a row in the mTMAs and candidates in the backbone flows. If there are still two or more candidates per row in the mTMAs, this means that the algorithm cannot determine a matching result. In this case, we infer that several identical mobile devices have generated the same or similar traffic.

B. The Adaptive Classifier Generation Algorithm

The input needed for the classifier generation procedure comes from our flow matching algorithm described in Section IV.A. We divide the flows into HTTP and non-HTTP flows because HTTP traffic comprises a high percentage of mobile traffic. HTTP payloads have "HTTP Host" and "HTTP User-agent" fields that can easily be used to identify mobile applications. For non-HTTP protocol flows, we apply the LCS algorithm [2]. The proposed classifier testing mechanism can then determine classifiers automatically with acceptable accuracy.

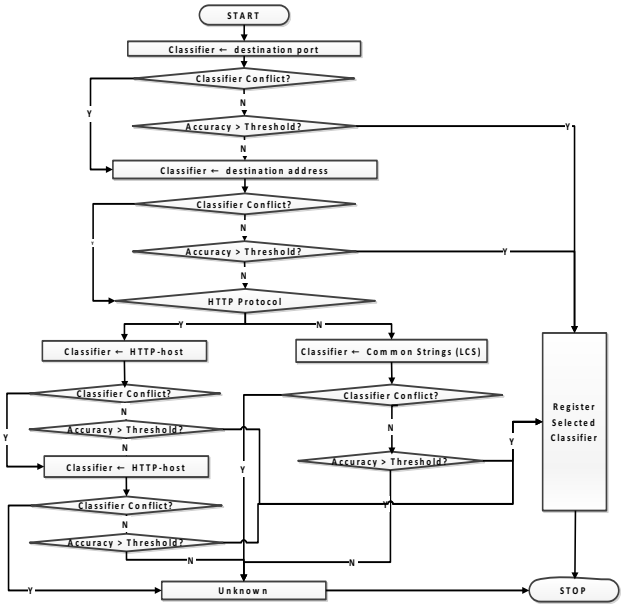

Fig. 3. Flowchart for the Automated Classifier Generation Algorithm

Fig. 3 gives a flowchart for our automated classifier generation algorithm. Our approach to determining classifiers involves selecting low-cost, key rules iteratively for each mobile application, then testing whether the generated classifiers can identify mobile applications while assuring acceptable accuracy. We apply our classifiers in the following rule-based order: destination port numbers, destination IP addresses, subnet of destination IP addresses, "HTTP Host" & "HTTP User-agent" fields for HTTP application protocols, and common payload strings for non-HTTP application protocols. Selected classifier candidates are tested by checking for conflicts and examining the accuracy of the selected classifier candidates. When there is no conflict between the selected classifier candidates and the accuracy is within a pre-determined threshold, we regard the candidates as available classifiers for that mobile application and these classifiers can use traffic identification. When none of the selected classifier candidates identifies a specific mobile application above the accuracy threshold, the proposed algorithm determines that it cannot classify the mobile application.

V. VALIDATION

In this section, we validate our proposed mobile traffic classifier generation system and evaluate the accuracy of the generated classifiers. In the evaluation, we applied our system to our campus network environment.

A. The Mobile Packet Capture Environment

We collected mobile packet traces from the dormitory Internet junction at POSTECH, a university with a dormitory population of approximately 4,000 students and researchers. All dormitory users are connected to the university network by means of wired LAN connections or 1,000 deployed Wi-Fi APs. To avoid any possibility of packet loss, we used a monitoring probe equipped with an optical tap and Endace DAG 4.3GE card [20] to monitor the 500Mbps Ethernet link. Our proposed system utilized this environment to provide payload data in order to generate classifiers and perform traffic identification using captured traces. We captured our campus dormitory mobile traffic using the tcpdump tool [17]. To filter the mobile traffic, we used the IP addresses of the dormitory Wi-Fi APs as filtering rules for the tcpdump tool. Our campus Wi-Fi APs use NAT functionality, which requires the flow matching algorithm in our proposed architecture. Table II illustrates the mobile traffic information used in the validation of our proposed system.

TABLE II. Captured Packet Trace Information

	Date	***Start Time***	***End Time***	***Total Packets***	***Total Volume***
Trace	June 12, 2011	21:03:37	22:02:32	66,578,145	54.78GB

TABLE III. Selected Mobile Applications & Categories

Category	***Defined Mobile Application Name***	***Actual Mobile application***	***Protocol***
Multimedia	Multimedia Player	YouTube, Daum TV Pot, JJangLive	HTTP
	YouTube	YouTube	HTTP
	JJangLive	JJangLive	HTTP, Proprietary
Browser	Web browser	Internet	HTTP, HTTPS
Messaging & SNS	Facebook	Facebook for Android	HTTP, HTTPS, Proprietary
	KakaoTalk	KakaoTalk	HTTP, HTTPS
Market	Android Market	Android Market	HTTP, HTTPS
	T Store	T Store	HTTP, HTTPS

B. The Selected Applications and Generated Classifiers

We focused on four popular categories and a total of nine mobile applications for mobile traffic identification. We implemented mTMAs for the Android platform and chose Android mobile applications. Table III presents the selected applications along with their categories, defined mobile application name, actual mobile application names, and the application protocols used. From the mobile traffic in Table II, our deployed system generated mobile traffic classifiers using our proposed adaptive algorithm. The resulting classifiers generated are presented in Table IV.

The threshold value we used was the number of distinct classifier items in conjunction with accuracy. For example, when we tested the KakaoTalk mobile application, the number of distinct ports detected was two (port 80 and 443), but the accuracy with regards to the port classifier was 259.4%, which means that it over-detected this mobile application. Twelve different IP addresses associated with KakaoTalk traffic were collected, signifying that an IP address classifier cannot be a candidate for the KakaoTalk classifier because of the many number of IP addresses associated with it. Since there were two distinct IP subnets, we tried using IP subnet as a candidate for the classifier. The accuracy was 100%, and the adaptive algorithm thus selected IP subnet as the classifier for the KakaoTalk mobile application.

C. Validation of the Generated Classifiers

We carried out mobile traffic identification using the generated mobile traffic classifiers illustrated in Table IV in

TABLE IV. Generated Mobile Traffic Classifiers

Mobile Application Name	*Classifier Type*	*Classifier*
Media player	HTTP User-Agent	stagefright/1.1 (Linux;Android 2.3.3)
		stagefright/1.1 (Linux;Android 2.3.4)
YouTube	HTTP User-Agent	Android-YouTube/2 (SHW-M110S GINGERBREAD); gzip
JJangLive	IP	110.45.160.213, 110.45.160.215, 211.115.90.136
Web browser	HTTP User-Agent	Mozilla/5.0 (Linux; U; Android 2.3.3; ko-kr; SHW-M110S Build/GINGERBREAD) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
		Mozilla/5.0 (Linux; U; Android 2.3.4; ko-kr; XT800W Build/MIUI) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
Facebook	HTTP Host	facebook.com, ak.fbcdn.net
KakaoTalk	Subnet	110.76.140.0/24, 203.246.172.0/24
Android Market #1 (Download)	HTTP User-Agent	AndroidDownloadManager
Android Market #2 (Search)	HTTP User-Agent	Android-Market/2 (SHW-M110S GINGERBREAD); gzip
T Store	PORT	444, 8205, 9104, 9200, 9401

TABLE V. Mobile Traffic Identification Result using the Generated Classifiers

Mobile Application Name	*Classified traffic (kB)*	*False negative (kB)*	*False positive (kB)*	*Precision (%)*	*Recall (%)*
Media player	369,364	0	0	100.00	100.00
YouTube	5,814	0	0	100.00	100.00
JJangLive	52	155	6	89.18	25.01
Web browser	15,822	109	2,729	85.29	99.32
SearchBox	17	0	0	100.00	99.57
Facebook	750	1,130	80	90.34	39.88
KakaoTalk	6,221	0	0	100.00	100.00
Android Market #1	123,923	0	0	100.00	100.00
Android Market #2	1,870	88	0	100.00	95.50
T Store	91,932	0	490	99.47	100.00
Total mTMA Log Traffic	615,763 kB				

order to check the accuracy of the classifiers generated. Table V shows the mobile traffic identification results in terms of flow size for classified (true positive) traffic, false negative and positive traffic, precision and recall. Precision (1) is the fraction of traffic results that are relevant to the classification and recall (2) is the fraction of traffic results that are relevant to the classifier successfully created. We can calculate the overall Accuracy (3) of our mobile traffic identification. The overall accuracy was 99.23% in our experiment.

$$Precision = \frac{[true\ positive]}{[true\ positive] + [false\ positive]} \quad (1)$$

$$Recall = \frac{[true\ positive]}{[true\ positive] + [false\ negative]} \quad (2)$$

$$Accuracy = \frac{[true\ positive] + [true\ negative]}{all\ traffic\ used\ for\ traffic\ classification} \quad (3)$$

The recall values for the JJangLive (25.01%) and Facebook (39.88%) mobile applications are remarkably low. We surmise that the IP type classifier for the JJangLive mobile application cannot cover all the traffic from those applications. When we analyze the Facebook mobile application traffic, not only HTTP traffic but also HTTPS and its proprietary protocol are captured. We surmise that the generated classifier for the Facebook mobile application also cannot identify Facebook HTTPS or its proprietary protocol traffic. Although the recall values for these two mobile applications are low, the precision is high, which means that these classifiers can identify mobile applications without errors.

Finally, we estimated the population proportion with 95% confidence. We used a confidence interval (5) that indicated the reliability of our estimate. The overall estimated identification accuracy was 99.23 ± 0.0007% (flow size) and 83.85 ± 1.4736% (number of flows).

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} < p < \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\ , where\ z_{\alpha/2} = 1.96 \quad (5)$$

VI. CONCLUSIONS

Although there have been various studies on application traffic identification, to the best of our knowledge, none of the research has investigated accurate mobile application traffic identification. As the number of mobile devices and the volume of traffic they generate are increasing, identifying mobile applications is one important step toward providing stable and seamless mobile services using Wi-Fi or 3G networks. The proposed classifier generation system is focused on overcoming some of the limitations inherent in taking measurements on a mobile device, while the automated classifier generation algorithm is aimed at reducing human effort and providing accurate or detailed application-level information on mobile traffic.

In terms of the present research, the most important future work is to increase the accuracy of the generated classifiers. Other major future work is to develop mTMAs for other mobile device platforms such as iOS, Blackberry, and Windows Phone 7 in order to extract various type of mobile traffic. To decrease the high packet capture loss in mobile devices, we will investigate how we can lower the loss (e.g. dual-core smartphones). Furthermore, we will perform comparative application-level measurement analyses between mobile and non-mobile traffic, or Wi-Fi and 3G network traffic to ascertain the different characteristics of network traffic.

ACKNOWLEDGMENT

We thank Geosung Lee for helping us implement mTMAs for the Android platform.

REFERENCES

[1] Distimo, "The battle for the most content and the emerging tablet market," April 2011, http://www.distimo.com/blog/2011_04_the-battle-for-the-most-content-and-the-emerging-tablet-market/.

[2] G. Maier, F. Schneider, and A. Feldmann, "A First Look at Mobile Hand-held Device Traffic," Passive and Active Measurement, Zurich, Switzerland, April 7–9, 2010, pp. 161–170.

[3] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A First Look at Traffic on Smartphones," Internet Measurement Conference (IMC), 2010, pp. 281–287.

[4] A. Gember, A. Anand, and A. Akella, "A Comparative Study of Handheld and Non-Handheld Traffic in Campus Wi-Fi Networks," Passive and Active Measurement, Atlanta, USA, March 20–22, 2011, pp. 173–183.

[5] E. P. Freire, A. Ziviani, and R. M. Salles, "Detecting Skype Flows in Web Traffic," Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), April 2008, pp. 89–96.

[6] R. Alshammari and A. N. Zincir-Heywood, "Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype," Proc. of the 2009 IEEE Symposium on Computatioinal Intelligence in Security and Defense Applications, July 2009, pp. 1–8.

[7] B. Park, Y. J. Won, M. S. Kim, and J. W. Hong, "Towards Automated Application Signature Genration for Traffic Identification," Proc. of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), April 2008, pp. 160–167.

[8] G. Szabó, Z. Turányi, L. Toka, S. Molnár, and A. Santos, "Automatic Protocol Signature Generation Framework for Deep Packet Inspection," ValueTools 2011, 5th International ICST Conference on Performance Evaluation Methodologies and Tools, Cachan, France, May 16–20, 2011.

[9] Y.J. Won, B.C. Park, S.C. Hong, K.B. Jung, H.T. Ju, and J. W. Hong, "Measurement Analysis of Mobile Data Networks," Passive and Active Measurement Conference, Louvain-la-neuve, Belgium, April 5–6, 2007, pp. 223–227.

[10] IANA, IANA port number list, http://www.iana.org/assignments/port-numbers.

[11] A. Moore and K. Papagiannaki, "Toward the Accurate Identification of Network Applications," Passive and Active Measurements Workshop, Boston, MA, USA, March 31–April 1, 2005, pp. 41–54.

[12] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," ACM SIGCOMM, Philadelphia, PA, August 2005, pp. 229–240.

[13] S. Sen, O. Spatscheck, and D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures," WWW 2004 Conference, pp. 512–521.

[14] Y. Lim, H. Kim, J. Jeong, C. Kim, T. Kwon, and Y. Choi, "Internet Traffic Classification Demystified: On the Sources of the Discriminative Power," ACM SIGCOMM CoNEXT, Philadelphia, PA, December 2010, pp. 1–12.

[15] P. Srisuresh and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," RFC 2663, August 1999.

[16] M. Cotton and L. Vegoda, "Special Use IPv4 Addresses," RFC 5735, January 2010.

[17] Tcpdump & libpcap, http://www.tcpdump.org/.

[18] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape, "PortLoad: taking the best of two worlds in traffic classification", IEEE INFOCOM, 2010, pp. 1–5.

[19] K. C. Claffy, H.-W. Braun, and G. C. Polyzos, "A Parameterizable Methodology for Internet Traffic Flow Profiling," IEEE Journal on Selected Areas in Communications, vol.13, no.8, October 1995, pp. 1481–1494.

[20] Endace, DAG 4.3GE, http://www.endace.com/dag4_3GE.htm.

[21] Google, Youtube, http://www.youtube.com/.

[22] SK Telecom, T store, http://www.tstore.co.kr/.

[23] Kakao, KakaoTalk, http://www.kakao.com/talk/en.

[24] Facebook, Facebook for Android, http://www.facebook.com/android.

[25] Daum, Daum TV Pot, http://mobile.daum.net/web/mobileApp.daum?serviceId=tvpot.

[26] UAJJANG, JJanglive Mobile Application, http://www.jjanglive.com/main.jjang?cmd=mobile.