An Automated Classifier Generation System for Application-Level Mobile Traffic Identification

2011 정 라

Master Thesis

An Automated Classifier Generation System for Application-Level Mobile Traffic Identification

Yeongrak Choi (최 영 락)

Division of IT Convergence Engineering (Autonomics)

Pohang University of Science and Technology

2011

어플리케이션 모바일 트래픽 식별을 위한 자동화된 분류자 생성 시스템

An Automated Classifier Generation System for Application-Level Mobile Traffic Identification

An Automated Classifier Generation System for Application-Level Mobile Traffic Identification

by
Yeongrak Choi

Division of IT Convergence Engineering (Autonomics)
Pohang University of Science and Technology

A dissertation submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Division of IT Convergence Engineering (Autonomics).

Pohang, Korea
June 20, 2011

Approved by

Academic Advisor: James Won-Ki Hong

An Automated Classifier Generation System for Application-Level Mobile Traffic Identification

Yeongrak Choi

The undersigned have examined this thesis/dissertation and hereby certify that it is worthy of acceptance for a master's degree from POSTECH.

June 20, 2011

Committee Chair	James Won-Ki Hong	(Seal)
Member	Jong Kim	(Seal)
Member	Young-Joo Suh	(Seal)

MITCE 20091163 최영락 Yeongrak Choi, “An Automated Classifier Generation System for Application-Level Mobile Traffic Identification”, “어플리케이션 모바일 트래픽 식별을 위한 자동화된 분류자 생성 시스템”
Division of IT Convergence Engineering (Autonomics), 2011, 49P, Advisor: James Won-Ki Hong, Text in English.

ABSTRACT

As mobile devices and smartphones have become increasingly popular in recent years, there are now a growing number of mobile applications that require access to the Internet. Network administrators need to be able to identify application-level mobile traffic in order to deal with this mobile ‘big-bang’, but doing so is a challenge because of the rapid increase in the number of mobile applications and the volume of mobile application traffic. To deal with diverse mobile applications with minimal effort, we need a highly accurate automated method to identify mobile applications.

This thesis investigates the architecture of an automated system to generate classifiers. These classifiers represent key features of each mobile application’s traffic. This architecture includes mobile traffic measurement agents (mTMAs) which are installed on mobile devices and monitor their application-level traffic. The proposed mTMAs are designed to work well even on mobile devices that have low computation processing power and minimal memory. The proposed flow matching shows how the information collected by mTMAs can be converted into effective input for a classifier generation system.

Finally, this thesis proposes an algorithm to automatically find each mobile application’s classifiers. Using this proposed algorithm, we extract several mobile traffic classifiers from our campus network traffic. The experiment results show the feasibility of our proposed method with acceptable accuracy.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Application traffic identification is an important step toward understanding network usage, providing high quality network services, and managing selfish-application traffic. Identifying mobile applications from the Internet traffic is challenging because of the variety of mobile devices and their applications and the difference between mobile application and traditional Internet application traffic patterns. Nowadays, the number of mobile applications is increasing rapidly. Many electronics manufacturers are producing various types of mobile devices such as mobile phones, PDAs (personal digital assistants) and tablets and people are using a variety of applications with these mobile devices. According to [1], there are more than 200,000 Android and about 300,000 iPhone available applications as of March, 2011. The traffic patterns generated by these mobile applications have specific characteristics such as a smaller flow size and relatively high percentage of HTTP traffic [2]. Moreover, most mobile applications are based on client-server architecture, rather than P2P architecture.

The signature-based traffic identification approach provides reliable accuracy with considerable manual effort. To minimize this effort and deal with a higher number of applications, a number of studies have proposed automated application signature generation methods [3][4][5]. However, these methods require high computation and memory cost. Another difficulty in mobile traffic identification is that it is more difficult to collect ground truth data for mobile applications. To collect ground truth for traffic identification, we need to install additional measurement agents on target devices [3][6]. Mobile devices lack computation processing power and have limited memory and these constraints made it difficult to collect ground data for mobile applications.

1.2 Research Goals

Mobile application traffic identification is a procedure to find the application that is

responsible for mobile traffic using packets or flows as the unit of measurement. This thesis proposes an automated method for finding classifiers which can effectively identify mobile applications based on Internet traffic. To automate the identification of mobile applications, we need to collect ground truth traffic with application names from mobile devices and we need an effective algorithm to generate classifiers. Hopefully, this work will suggest a good starting point to prepare network administrators for the steady increase in mobile devices and their traffic.

1.3 Proposed Method

This thesis proposes a way of modifying session-based and content-based approaches in order to generate classifiers automatically and to effectively identify mobile applications. Our proposed classifiers reflect the characteristics of mobile traffic and show better accuracy in identifying mobile applications. Moreover, our proposed classifiers are more optimized to identify mobile applications than existing approaches.

An automated method for generating classifiers must be able to select from all possible classifier candidates and an algorithm to collect ground truth traffic. We define the available classifiers and optimize them for the mobile traffic environment. Our proposed architecture automatically collects ground truth traffic using mobile traffic measurement agents (mTMA) to support light-weight processing on mobile devices. Using our proposed architecture, we can generate classifiers for each mobile application.

We validate our classifiers by collecting real traffic from our campus backbone traffic. We calculate the accuracy of our classifiers by comparing them with the ground truth traffic collected by our mTMAs.

1.4 Thesis Outline

The structure of this thesis is as follows:

Section 2 describes related work on analyzing mobile traffic and compares classifiers from existing approaches. The overall architecture with mobile traffic measurement

agents (mTMA) is discussed in Section 3. Section 4 describes the procedures needed to generate application classifiers from flow matching of backbone traffic and mobile traffic and to find classifiers with the proposed algorithm. Section 5 discusses the validation of this new approach. Finally, Section 6 presents the thesis conclusions in together with suggested further work and summary of the thesis' contributions.

2 Related Work

This section outlines existing work on the analysis of mobile traffic and on classifiers used in existing traffic classification methods. This section first describes state of the art research on the analysis of mobile traffic. Then, we present existing traffic classification approaches such as session-based, content-based and statistics approaches with their classifiers, cost of operation and accuracy.

2.1 Analysis of Mobile Traffic

Previous studies have presented the measurement and analysis of mobile traffic with the characteristics of mobile networks.

Won et al. compared the mobile data traffic trace of a CDMA network [7] with wired Internet traffic. A few of the characteristics of mobile traffic are the small sized packets (more than 85% of the total packets are under 100 bytes), preponderance of inbound traffic (85:15 in packet counts and 91:9 in byte counts), short session length (90% of data traffic within 10-20 seconds), and high retransmission ratio (almost 80%). However, the authors did not describe application-level mobile traffic characteristics, and the appearance of new types of mobile devices such as smartphones might change the traffic characteristics that they discovered.

Maier et al. analyzed hand-held mobile device traffic from residential broadband DSL lines [8]. The authors observed more than 20,000 DSL lines over a period of 11 months and they manually identified mobile traffic based on HTTP user-agent strings and default IP TTL values. Their analysis results show that iPhones and iPod touch dominated mobile traffic (86-97% of hand-held mobile HTTP traffic and 71-87% of the devices), HTTP was the most used protocol in TCP (80-97% of all hand-held mobile bytes), and the most popular application was Apple's browser Safari (up to 62%). However, their analysis methods cannot guarantee accurate identification because these methods require cumbersome manual work and have a limited number of analysis categories.

Falaki et al. performed smartphone traffic analysis using traffic logs from 43 users across two smartphone platforms [2]. Windows Mobile platform applications logged packet level traces and Android platform applications periodically recorded the number of bytes sent and received by each process. Based on log results, their application-level analysis includes the port distribution result (HTTP & HTTPS make up almost 80% of the traffic) and per-process traffic usage result (browsing makes up more than half the traffic: 58.02%). Although the authors revealed various characteristics of smartphone traffic (transfer size, retransmission rate, round-trip-time, and the interaction with radio power) in detail, their port-matching method cannot identify applications with high accuracy [15] and per-process traffic usage cannot provide packet-level or flow-level analysis results.

Gember et al. studied packet traces from campus Wi-Fi networks, with 3 days of traffic for 32,278 handheld and non-handheld devices [9]. Their application protocol analysis shows that more than 90% of handheld traffic is HTTP and HTTPS, and their traffic reveals that video traffic is more than twice as common on handheld devices (40%) than on non-handheld devices (17%). Moreover, the handheld web traffic shows lower HTTP host diversity (the top 10 handheld hosts account for 74% of handheld data, whereas 42% of non-handheld data) and greater intra-user content similarity. However, the method they use to identify handheld devices requires them to collect each device's MAC address, which is only possible with access to all Wi-Fi APs. Furthermore, their classification result is based on application-layer protocol types, which provides abstract information on mobile traffic dynamics.

In summary, previous studies have demonstrated that the vast majority of HTTP and HTTPS application protocol traffic is generated from mobile device applications. However, the manual analysis required by these methods involves laborious effort and cannot provide accurate or detailed application-level information on mobile traffic because these protocol classification methods depend on port-matching, which provides conceptual knowledge related to the applications but guarantees low accuracy. Moreover, most mobile devices retrieve larger size of responses than smaller size of requests, which can imply that most mobile applications run as client in the client-server architecture.

2.2 Classifiers in Existing Approaches

This section compares classifiers used in three traffic identification approaches: session-based, content-based and statistics approaches. Each approach is explained in detail with its cost of operation and accuracy.

2.2.1 Session-based Traffic Identification

Well-known Port Matching: This method uses TCP and UDP port numbers as classifiers to identify applications. The Internet Assigned Numbers Authority (IANA) maintains the official assignments of port numbers for the use of applications [10]. The cost of operation for this method is low because to extract this classifier we only need to parse the TCP/IP header formats and match the port numbers to identify applications. However, more complex network conditions such as the deployment of a firewall, use of VLAN or VPN, or allocation of ephemeral ports by P2P applications prohibit applications from using the official port numbers assigned by IANA. *Moore et al.* [15] dispute that port matching guarantees only 50-70% of accuracy in the current Internet traffic, which is low.

Session Behavior Modeling: BLINC [16] models the behavior of traffic patterns using graphs and each unique traffic pattern is represented by set of 4-tuples {source IP address, destination IP address, source port number, destination port number} as classifiers. The cost of the BLINC classification method is not as high as content-based approaches because it just performs the matching of each traffic graph pattern. Although the authors insist that BLINC classifies 80-90% of the traffic with more than 95% accuracy, its accuracy depends on the difficulty of modeling correct traffic patterns. It accompanies adequate operation costs such as limited portion of manual payload inspection.

2.2.2 Content-based Traffic Identification

Signature Matching: The signature matching method [11] pre-defines distinct

classifiers as a subnet of payload strings with static or dynamic positions called signatures. These signatures are mapped to applications or application groups, and then the method identifies applications by comparing signatures with payload data in a set of packets or flows. Although its operation requires the comparison overhead of matching strings, it guarantees high accuracy if there is no flaw in the manual creation of signatures. However, this creation process requires a major human effort, so **Automated Creation of Signatures** methods have been proposed to minimize this effort. Signatures can be a classifier to identify application protocols. Some packet monitoring and analysis tools, such as Wireshark [18], offer a signature matching function and can identify a few application protocols such as FTP, TELNET, SMTP and HTTP.

Automated Signature Generation: As more applications use proprietary protocols, it becomes increasingly difficult to manually generate signatures for various applications. Several novel algorithms have been proposed to automatically generate accurate signatures without application knowledge. The LASER method [3] is based on the modification of the LCS (longest common sequence) algorithm and it automatically determines a pattern in the packet's payload. With some constraints (e.g. minimal substring length) and refinements (e.g. elimination of trivial strings), a set of payload strings is determined to classify applications. The adaptive merging algorithm [4] and motif finding algorithm [5] were proposed to automatically generate more accurate signatures. Although these algorithms can minimize the manual effort, their cost of operation is high as automatically generating signatures requires a lot of computational power and large memory space.

2.2.3 Statistics-based Traffic Identification

Rather than session-based and content-based identification, a statistics-based approach uses classifiers as a group of non-deterministic features. Methods using this approach first construct statistical model information related to traffic data in order to determine its features. Classification techniques in this approach, such as Supervised or Un-supervised Machine Learning [13] and Statistical Signature [12], convert flows to a predetermined

number of clusters according to the following constraints: Port number, number of packets, flow duration, average packet size of a flow, packet inter-arrival time and more. *Lim et al* analyzed different features and algorithms that can be used to traffic classification [17] and the authors claim that this technique performs surprising well in terms of accuracy. However, many applications have yet to be verified on more complex traffic (e.g. P2P file sharing).

Table 1. Categorization of Traffic Identification Approaches and Classifiers

Category	*Approach*	*Classifiers*	*Cost of Operation*	*Accuracy*
Session-based	Well-known Port Matching [IANA]	Port	Low	Low
	Session Behavior Modeling [Karagiannis, SIGCOMM'05]	{srcIP, dstIP, srcPort, dstPort}	Medium	Medium
Content-based	Signature Matching [Sen, WWW'04]	Payload strings	High	High
	Automated Generation of Signatures [Park, NOMS'08]	Payload strings	High	High
Statistics-based	Group of features [Lim. CoNext'10]	Ports, number of packets, transferred bytes, duration	High	Medium
Proposed method	Automated choice of lower-cost classifiers optimized for mobile traffic identification	dstPort, dstIP, HTTP-host, HTTP-useragent (or common strings)	Medium	High

The last entry in Table 1 represents the proposed identification method in this thesis - an automated approach for choosing lower-cost classifiers optimized for mobile traffic identification.

3 Proposed Architecture

3.1 Introduction

Before explaining mTMAs and the proposed architecture, we first define what we mean by mobile applications and classifiers. Then we describe the target mobile applications and the environment for collecting packet traces.

3.1.1 Definitions

Mobile Application: We define a mobile application as a software process developed for mobile devices that requires network connectivity such as Wi-Fi or 3G networks and generates distinguishable network traffic to fulfill the overall service requirements. Although two different mobile applications A and B use the same application protocols such as HTTP and HTTPS, we can differentiate A from B if the A and B's traffic have different characteristics such as different destination port numbers and different payload strings. As explained in Section 2.1, most mobile applications use HTTP and HTTPS protocols and access mobile web content. Some mobile applications show different traffic characteristics although they use the same application protocols. For example, both the Android [23] web browser and Facebook mobile application [29] use the HTTP application protocol, but web browser HTTP traffic has distinct 'HTTP User-agent' payload strings representing Android OS code names such as "FROYO" (Android OS Version 2.2) and "GINGERBREAD", and Facebook HTTP traffic has "facebook.com" or "ak.fbcdn.net" payload strings in 'HTTP Host'. Some mobile applications are dependent on the mobile device platform. Our definition is slightly different from the ones used in most mobile application stores such as the Apple App Store [24] Android Market [25] and T Store [27]. For instance, Android YouTube [26], Daum TV Pot [30] and JJangLive [31] mobile applications support real-time multimedia videos. However, they actually call Android a 'mediaserver' process and this process retrieves videos using Wi-Fi or 3G

networks generating distinct traffic characteristics. According to our definition, ‘mediaserver’ is a mobile application for watching multimedia videos and we regard other traffic such as video information from YouTube as coming from another mobile application.

Classifier: We define a classifier as a rule for classifying Internet traffic. A rule can express the distinguishing characteristics of an application. These characteristics can be observed in captured packets or flows and include 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 that are present in the packet payload [11]. According to our definition, classifiers can be port matching, signature matching, or even statistics-based traffic identification rules. This thesis investigates low-cost classifiers to complicated classifiers to check whether or not a classifier can effectively identify mobile applications.

Mobile Traffic Classifier: With above-mentioned definitions, we define a mobile traffic classifier as a collection of rules for identifying mobile application traffic. Specific mobile traffic classifiers can identify certain mobile applications with low-cost and high accuracy, but cannot identify other mobile applications. For instance, the IP subnet classifier can identify KakaoTalk [28] with low-cost and 100% accuracy in our work, but cannot identify mobile web browsers because web browsers can connect to unlimited hosts with different subnets. For mobile web browsers, the ‘HTTP User-agent’ classifier shows high accuracy with more costly operations than the IP subnet classifier. In this thesis, we select the following low-cost classifiers: destination IP address and port numbers reflecting client and server architecture in mobile devices environment, ‘HTTP Host’ and ‘HTTP User-agent’ formatted fields for HTTP traffic, and the LCS (longest common subsequence) [3] focusing on a small amount of non-HTTP mobile traffic.

3.1.2 Choice of Applications

We focus on four popular categories and a total of nine mobile applications for traffic identification. We chose Android mobile applications because we implemented a version of mTMAs for this platform, and we chose highly popular mobile applications in the Android platform. Table 2 illustrates chosen applications with their categories, process names, actual mobile application names, used application protocols and port numbers.

Table 2. Target Mobile Applications

Category	*Defined Mobile Application Name*	*Process name*	*Actual Mobile application*	*Protocol*	*Port (Monitored)*
Multimedia	Multimedia Player	/system/bin/ mediaserver	YouTube, Daum TV Pot, Jjanglive	HTTP	80, 8080
	YouTube	com.google. android.youtube	Youtube	HTTP	80
	JJangLive	com.uajjang. android	JJangLive	HTTP, Proprietary	80, 11120, 5220, 15151, 1109
Browser	Web browser	com.android. browser	Internet	HTTP, HTTPS	80, 443, 8080
	Searchbox	com.google. android. googlequick searchbox	(attached to other mobile applications)	HTTP	80
Messaging & SNS	Facebook	com.facebook .katana	Facebook for Android	HTTP, HTTPS, Proprietary	80, 443, 5222
	KakaoTalk	com.kakao.talk	KakaoTalk	HTTPS	443
Market	Android Market	android.process .media	Android Market	HTTP	80
		com.android .vending		HTTP HTTPS	80, 443
	T Store	com.skt.skaf. A000Z00040	T Store	HTTP HTTPS	444, 8205, 9200, 443, 9401, 9104, 20000

3.1.3 Mobile Packet Capture Environment

We collected mobile packet traces from dormitory Internet junctions at POSTECH, a university with a dormitory population of about 4,000 students, researchers and staff. All dormitory users are connected to the university network using wired LAN connections or 1,000 deployed Wi-Fi APs. To avoid any possible packet loss, we used a monitoring probe equipped with an optical tap and Endace DAG 4.3GE card [35] 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 [32]. To filter the mobile traffic, we applied the IP addresses of dormitory Wi-Fi APs as filtering rules for the tcpdump tool.

Our campus Wi-Fi APs use NAT (Network Address Translator) functionality [19][20], which is the process of mapping one real IP address to other IP address(es) while routing traffic to hosts. In our campus network environment, mobile devices connected to Wi-Fi APs are assigned to IP addresses for private networks [21], whereas the monitoring probe only observes the IP addresses of Wi-Fi APs. To generate classifiers in our proposed system, the *mix* of application-level packet information from mTMAs and payload data from capture traffic should be provided concurrently. However, NAT functionality prevents us from identifying a captured flow, regardless of whether the flow is generated from a single mobile device.

Since we captured traffic from our campus Wi-Fi APs, we also observed network traffic from non-mobile devices such as notebook computers that connected to our campus Wi-Fi APs. An OS fingerprinting approach is one way of distinguishing mobile traffic from the *mix* of mobile and non-mobile traffic [22]. This approach is based on the differences of an OS's default TCP option parameters shown in the TCP SYN and SYN-ACK packets. To obtain each OS fingerprint, we purposely generated distinguishable mobile traffic by using different devices and connecting to unknown and distinguishable web pages, and we extracted OS fingerprints from that traffic. Table 3 is the captured

trace of our campus Wi-Fi APs used for generating classifiers with mTMAs, and Figure 1 illustrates the distribution of major mobile and non-mobile operating systems (Windows, Mac OS, Android, iOS, and Blackberry) from the trace.

Table 3. Captured Packet Trace Information

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

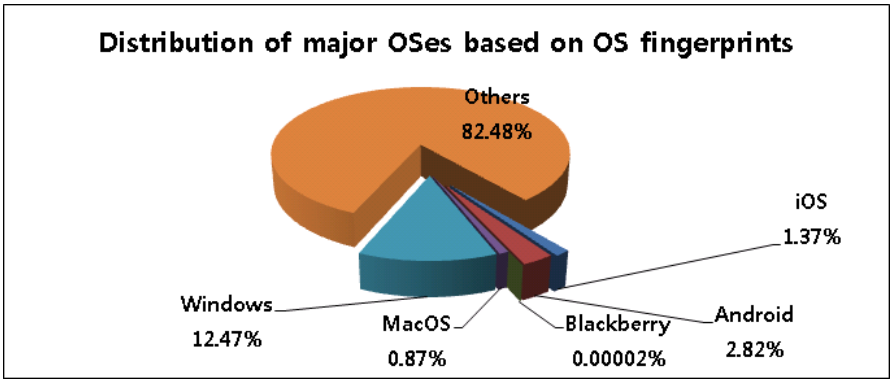

Figure 1. Distribution of Major Operating Systems from the Trace

The OS fingerprinting approach identifies the OS on a device and infers the type of each device. However, this approach cannot be applied to our proposed system because it needs exact matching of application-level packet information in mTMAs and traffic data captured in the backbone network. In this thesis, we propose a heuristic flow matching algorithm to overcome this limitation. The details of this proposed algorithm are described in Section 4.

3.2 Limitations of Traffic Analysis using Mobile Devices

The tcpdump tool and libpcap libraries [32] were developed to enable users to collect

packet-level and flow-level network traffic information, such as source and destination IP addresses and port numbers, total sent and received bytes, and the duration of each packet or flow. However, these solutions do not focus on application-level information such as which application process is sending and receiving packets or flows. To obtain application-level network information from mobile devices, we need to find a mechanism which relates network-level information with application process names, and sends collected information to a server called a mTMA server. Figure 2 illustrates the interaction between the mTMA program and mTMA server. The mTMA program captures a minimal summary of packet information with application process names, and the archive of this information is periodically transferred to the mTMA server to minimize the effect on network traffic by mTMA programs.

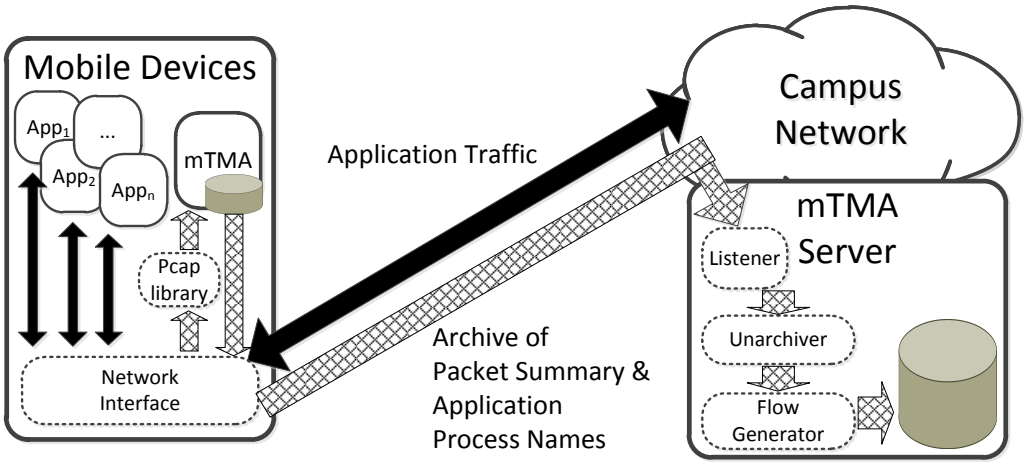

Figure 2. Interaction with mTMA Client and Server

In the Android platform, the netstat command can obtain application process names using socket information managed by the Android OS. The netstat command collects information rather slowly, and it cannot find application process names if the application process has already been terminated. To obtain accurate packet information and application process names, we developed an Android native application, and this native application collaborates with a service process and mTMA server.

To check the accuracy of mTMAs, we first compared the number of captured flows from mobile devices and dormitory Internet junctions to check whether the mTMAs captured all the packets in a flow. Packet loss can be calculated by comparing the number of captured packets in the backbone and in an mTMA program (1). We applied this calculation to validate our proposed flow matching algorithm which will be explained in Section 4.1.

$$Packet\ Loss\ = 1 - \frac{[\#\ of\ captured\ packets\ in\ mTMA]}{[\#\ of\ total\ captured\ packets\ in\ router]} \quad (1)$$

Table 4 illustrates the packet loss for three different trials using three types of mobile applications: multimedia, web browser and SNS (Social Network Service) and we match application names for all packets. The total packet loss radio for all the mobile applications in our experiment was 61.96%, which is very high. When we observe the packet loss rate for different categories, multimedia mobile applications had highest packet loss. We infer that playing videos requires heavy resources from mobile devices, and mTMAs cannot capture all the packets when there are a large number of many packets in a short time. Web browser mobile applications have the second highest packet loss rate, which supports our argument because SNS and messaging mobile applications send fewer packets than web browsers over the same time period.

Table 4. Packet Loss Ratio for Different Types of Applications

Category	*Application*	*Packet Loss Ratio*			*Average*	*Standard Deviation*
		1st	*2nd*	*3rd*		
Multimedia	YouTube	72.70%	73.42%	71.18%	72.43%	0.011
Browser	Internet	27.68%	20.47%	18.40%	22.18%	0.049
Messaging & SNS	KakaoTalk, Facebook	58.72%	5.06%	4.32%	22.70%	0.312
(Total)		61.15%	65.10%	59.63%	61.96%	0.028

Second, we analyzed the packet loss rate resulted from finding application names in mTMAs. All packets are captured from mTMAs and we want to find application names of those packets to know application-level information of packets. We compared two

different types of mTMAs: one finds application names of all captured packets and the other finds application names of only TCP SYN packets. We played the same YouTube video five times for each type of mTMAs and the result is shown in Figure 3. The average packet loss rate of the first type of mTMAs (matching all packets) is 29.79%. On the other hand, the average packet loss rate of the second type of mTMAs (matching only TCP SYN packets) is 0.58%. Although this packet loss rate is very low, there is still some loss of packets. We conclude that matching application names for all packets requires substantial overhead in mobile devices. We also conclude that flow information captured from mTMAs such as the number of packets in a flow and the total transferred bytes cannot be trusted any more. To obtain trusted flow information, the proposed system architecture utilizes backbone flow data, which will be explained in the next section.

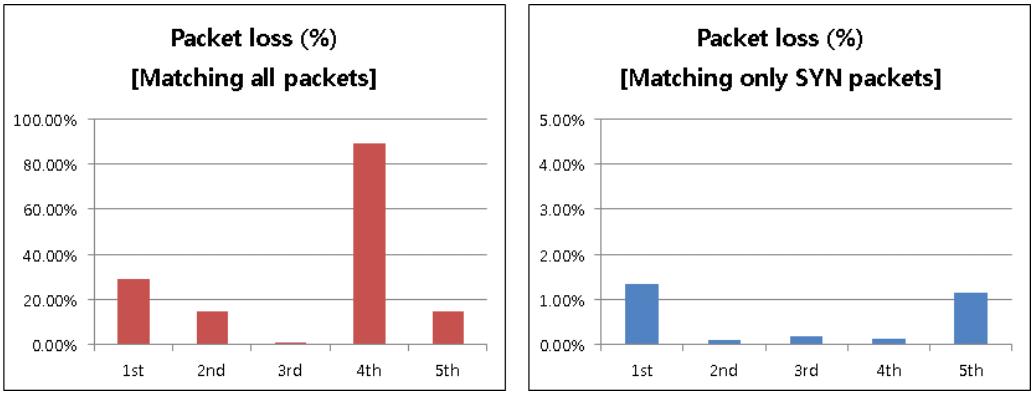

Figure 3. Comparison of Packet Loss: Matching all Packets vs. SYN Packets

3.3 Overall Architecture

In our proposed architecture, mobile traffic measure agents (mTMAs) are installed on mobile devices, and obtain application-level information such as which mobile applications are connected to our campus Wi-Fi network and generating traffic. Our proposed system runs within this architecture and utilizes the proposed flow matching

and automated classifier generation algorithms. The details of these algorithms will be explained in Section 4. This section addresses the details for each component and module of the system architecture illustrated in Figure 4.

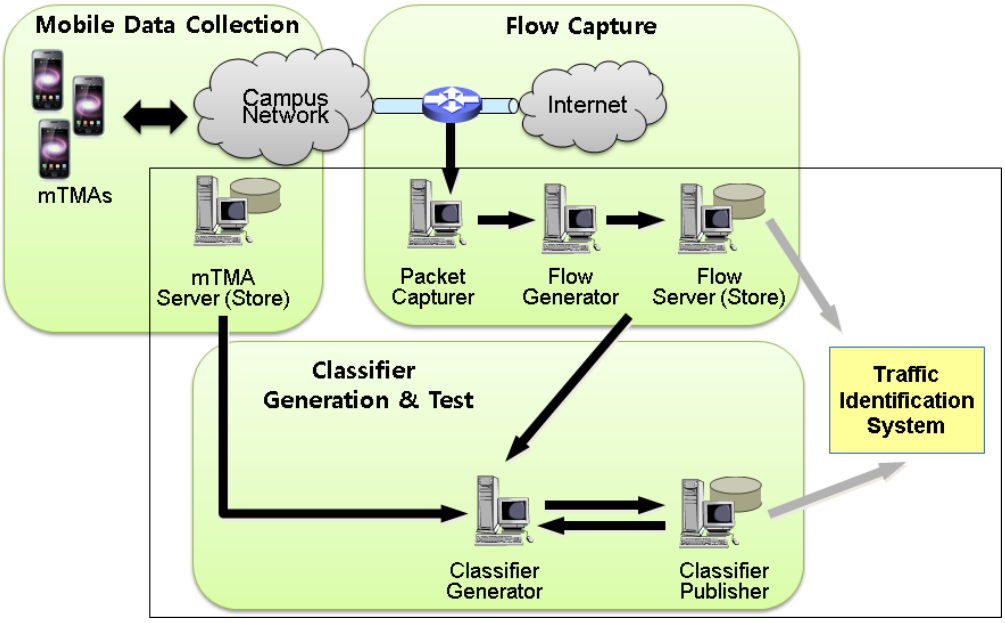

Figure 4. Proposed Automated Classifier Generation System Architecture

3.3.1 Mobile Data Collection

This component is composed of mTMAs and an mTMA Server. The role of the mTMAs is to collect application-level information from mobile devices. We developed an mTMA program that runs on the Android platform, and this agent program captures all the packets using libpcap and stores packet information and application process names separately. These results will be sent to the mTMA Server Store periodically. Since mobile devices have weak computation processing power, the programs we have developed do not match packets to application names or convert captured packets to flows in mobile devices.

Figure 5 illustrates the internal structure of the mTMAs implemented on the Android

platform. An mTMA service process monitors the activity of mobile applications without any interference from users and another native process observes the the names of mobile application processes and information related to their traffic. An mTMA is able to run only rooted Android devices because such collection using libpcap requires root privileges. On the Android platform, only native processes can require root privileges. Not even while general or service processes can do so. This is why the mTMA program is composed of a separate ‘mTMA service’ process and ‘mTMA native’ process.

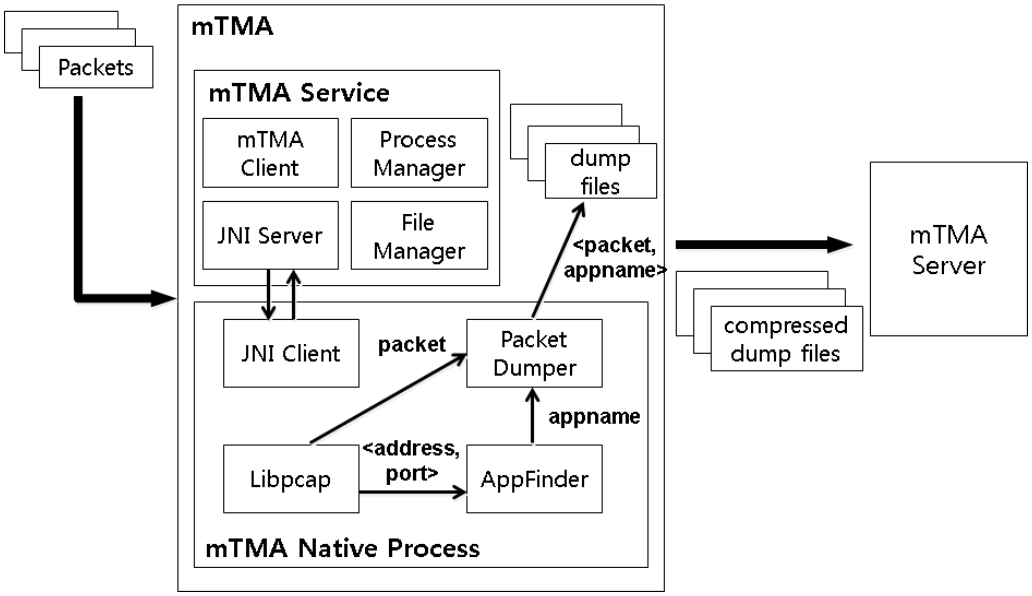

Figure 5. Internal Structure of mTMA in Android Platform

The ‘mTMA service’ process automatically executes the ‘mTMA native’ process when it starts. The mTMA client provides users with administrative user interfaces related to the mTMA program. To communicate between the ‘mTMA service’ and ‘mTMA native’ processes, we use JNI (Java Native Interface) server and client interfaces. When the ‘mTMA native’ process successfully stores captured packets and the corresponding application process names as one or more dump files, it sends the name of the dump files to the ‘mTMA service’ process using JNI. Then, the file manager in the ‘mTMA service’

manages the dump files and sends compressed dump files to the mTMA server periodically. Unlike non-mobile devices, mobile devices change network connectivity frequently, including Wi-Fi handovers and the change from Wi-Fi to 3G and vice versa. When these activities occur in mobile devices, the ‘mTMA native’ process is automatically terminated because the targeted monitoring network interface has gone offline. The process manager in the ‘mTMA service’ process monitors the network connectivity of a targeted mobile device and executes the ‘mTMA native’ process when it is not running.

When a mobile application processes generate network packets, the libpcap library in a ‘mTMA native’ process captures the generated network packets and appfinder in Figure 5 finds the applications’ names using a tuple of <source IP address, source port number, destination IP address, destination port number, protocol type>. For TCP protocol packets, we only find application names for TCP SYN packets because finding application names for all network packets requires too many mobile device resources and the limited resources of mobile devices means that the libpcap library cannot capture all of the packets.

Then, the mTMA Server aggregates all the information collected from mTMAs. This module matches packet information to application process names, and converts packet-level network information with application names to flow-level information. Finally, converted flow-level information is stored in a database to provide inputs for the flow matching algorithm in the Classifier Generator module. The internal mechanisms for generating flows from packets are the same as those for the Flow Generator.

3.3.2 Flow Capture

1) **Packet Capturer:** As explained in 3.1.3, backbone traffic is captured through a monitoring probe equipped with an optical tap and Endace DAG 4.3GE card. The responsibility of this module is to capture packets using the tcpdump tool. This module should have enough disk space to store all packet header information and payload bytes for the desired duration.

2) **Flow Generator:** The Flow Generator 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, flow size, flow duration and the flow's payload in bytes. Although the Packet Capturer fully captures the full packet payload, this module only stores the payload for the first 10 outbound and inbound packets because the first few packets of a flow guarantee highly accurate traffic identification [33]. We determined the flow start by observing SYN packets and the flow end by observing when FIN/RST packets were generated or when there was no packet of the same tuple for 64 seconds [34] for TCP flows.
3) **Flow Server:** This module manages flow-level traffic data using a DBMS (Database Management System). Each trace is managed by one or more tables and indexes are made in each database table to support fast access from the Classifier Generator.

3.3.3 Classifier Generation & Test

1) **Classifier Generator:** Automated classifier generation is performed in this module. Inputs of this module are provided as database tables from the mTMA Server and Flow Server. This module first matches application-level information on the mTMA Server and flow-level payload information on the Flow Server using the proposed flow matching algorithm. This module automatically generates candidates for classifiers by extracting useful features from matched results. Then, this module checks the accuracy of each candidate for classifiers whether it conflicts to other existing classifiers or its accuracy is acceptable or not. This algorithm for automated classifier generation finally produces classifiers which identify mobile applications.
2) **Classifier Publisher:** Since our proposed system automatically generates classifiers for each mobile application, generated classifiers may not be optimized

to support lower cost of operations. For example, one of the payload classifier rules for mobile web browser identification is to match the ‘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’ string with the ‘HTTP User-agent’ fields of a payload. However, only ‘Android 2.3.3’, ‘SHW-M110S’, and ‘GINGERBREAD’ substrings in ‘HTTP User-agent’ are proved to be meaningful payload for identifying this mobile application when we analyze this payload manually. To optimize the produced classifiers, we need to make our proposed classifiers public and receive feedback from other experts. The purpose of this module is to organize the produced classifiers from the Classifier Generator and publish them on a public homepage so as to enable users to give feedback on the produced classifiers.

3) **Traffic Classification System:** Producing classifiers would be pointless if they could not be used in a traffic classification system. This module runs traffic classification using our generated classifiers. By analyzing the result of this module, we can check the accuracy of our proposed classifiers and observe the distribution of various mobile applications across captured packet traces.

4 Application Classifier Generation Algorithms

To implement the proposed architecture, finding traffic data generated from mTMAs in backbone is the key to success of generating classifiers. However, the incomplete flow information of mTMAs and NAT functionality in Wi-Fi APs make it difficult to find traffic data exactly. To solve this problem, Section 4.1 proposes a heuristic matching algorithm to find mTMA traffic in the backbone trace. To test the accuracy of possible classifiers and ground truth data, we can use the results of the heuristic matching algorithm as inputs into the proposed automated classifier generation algorithm. In Section 4.2, we will explain the procedures and algorithms used to select and determine classifiers with examples related to a few mobile applications.

4.1 Finding mTMA Traffic in Backbone Trace

In the previous section, we proved that packet information collected from mTMAs cannot show how many packets were generated and transferred using Wi-Fi. Moreover, it is difficult for mobile devices to capture the full payload of all packets using mTMAs because of mobile devices' weak computation processing power and small memory space. To find precise packet information and of the full payload, the proposed architecture matches flows captured from mTMAs with those captured from campus junctions. However, NAT functionality prevents us from exactly matching packets from mTMAs and the backbone. This section shows how we applied flow matching using the proposed heuristic algorithms.

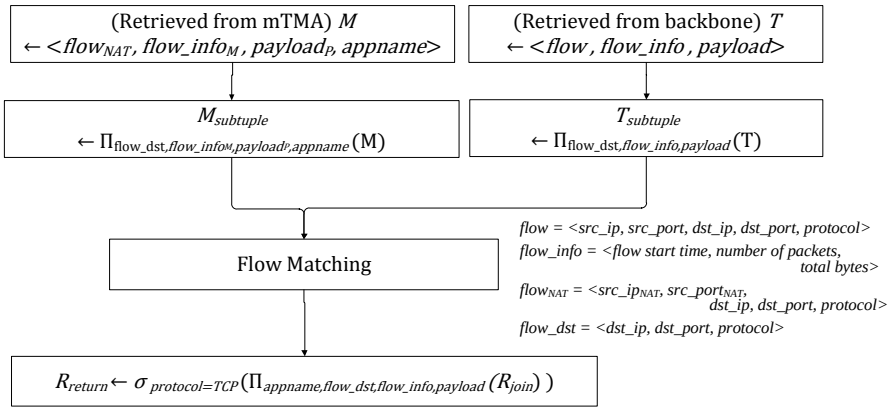

Figure 6. Procedures of Flow Matching in Classifier Generator

Figure 6 shows the overall procedures of flow matching in Classifier Generator module. Because of NAT functionality in Wi-Fi APs, only destination IP address, the destination port number and protocol type are extracted from a tuple of flow: <source IP address, source port number, destination IP address, destination port number, protocol type> in both mTMA and backbone data. mTMA data also contains application names and a small part of the payload for each flow. A small number of payload strings will be matched with payload data of flows in the backbone. After the flow matching algorithm, each row in the matched results is composed of a destination IP address, a destination port, an application name, and payload data from backbone.

Note that in this thesis, flow matching algorithms would be sufficient to generate classifiers for mobile applications. The proposed classifier generation should be automated, which means that the execution should not be slow. The proposed flow matching algorithm is used to recover inaccurate values collected from mTMAs such as the number of packets in a flow. The result of this algorithm will be provided as an input for automated classifier generation. Therefore, we do not require that all flows generated from mTMAs be 100% matched to flows in the backbone. Instead of designing flow matching algorithms with results that perfectly represent mobile device traffic captured from mTMAs, we propose a heuristic flow matching algorithm.

```
procedure TCPFlow_Matching ()
   for each item in M_subtuple
      T ← subnet of T_subtuple that matches
            <dst_ip, dst_port, dst_protocol> for item within time period
      if # of T for exactly matching <pktno, bytes> = 1 then
         Insert <appname,dst_ip,dst_port,flow_info,payload> into R
      end if
      else if # of T for exactly matching <pktno> = 1 then
         Insert <appname,dst_ip,dst_port,flow_info,payload> into R
      end if
      else if # of T for more or less <pktno> than item = 1 then
         Insert <appname,dst_ip,dst_port,flow_info,payload> into R
      end if
      else
         Perform Match payload_P and payload for item and T
         if # of matched results = 1 then
            Insert <appname,dst_ip,dst_port,flow_info,payload> into R
      end if
   end for
   return R
end procedure
```

Figure 7. Proposed Heuristic Flow Matching Algorithm

Our proposed flow matching algorithm is shown in Figure 7. This algorithm mainly depends on the number of packets and transferred bytes in a flow, which are non-deterministic values. For each row which contains flow information and an application process name from mTMAs, the flows that might be matched in the backbone are flows whose start time is almost the same to the row. Since the measurement point of an mTMA and an Internet junction are different, the flows would appear to start at different times, but this difference would not be substantial. We assume that an Internet junction on our campus can send out a packet within a few seconds. The algorithm first matches each row from an mTMA to candidates in the backbone using the number of packets and transferred bytes assuming that the number of packets from mTMAs is reliable. Although packet losses can occur in 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. Then, 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 mTMAs, this algorithm compares payload data between a row in mTMAs and candidates in backbone flows. Note that mTMAs only capture the first 20 bytes per packet in a flow not to make heavy overhead in mobile devices. If there is still more than one candidate for a row in mTMAs, this means the algorithm cannot determine a matching result. In this case, we infer that several identical mobile devices have generated the same or similar traffic. Table 5 shows found traffic in the backbone using our proposed flow matching algorithm.

Table 5. Traffic Found for each Mobile Application

Category	*Application Name*	*Application Process Name*	*# of flows*	*# of packets*	*Flow size*
Multimedia	Multimedia Player	/system/bin/mediaserver	62	405,468	378,228,999
	YouTube	com.google.android.youtube	55	8,288	5,953,064
	JJangLive	com.uajjang.android	30	555	211,090
Browser	Web Browser	com.android.browser	1,107	29,138	16,313,238
	SearchBox	com.google.android.googlequicksearchbox	10	104	17,015
Messaging & SNS	Facebook	com.facebook.katana	334	6,139	1,925,338
	KakaoTalk	com.kakao.talk	133	9,347	6,370,146
Market	Android Market	android.process.media	52	134,200	126,897,330
		com.android.vending	30	3,718	2,005,156
	T Store	com.skt.skaf.A000Z00040	291	99,117	94,138,064
(others)			284	10,194	6,038,118

4.2 Automated Classifier Generation

Figure 8 illustrates the overall procedure of classifier generation for mobile applications in the Classifier Generator module. The input needed for the classifier generation procedure comes from our flow matching algorithm described in Section 4.1. We divided flows into HTTP and non-HTTP flows because mobile traffic has a high percentage of HTTP traffic and HTTP payloads have 'HTTP Host' and 'HTTP User-agent' fields which can easily identify mobile applications. For non-HTTP protocol flows, we applied an LCS (longest common sequence) algorithm [3] grouping destination port numbers. Then, the proposed classifier testing mechanism can determine classifiers automatically with acceptable accuracy.

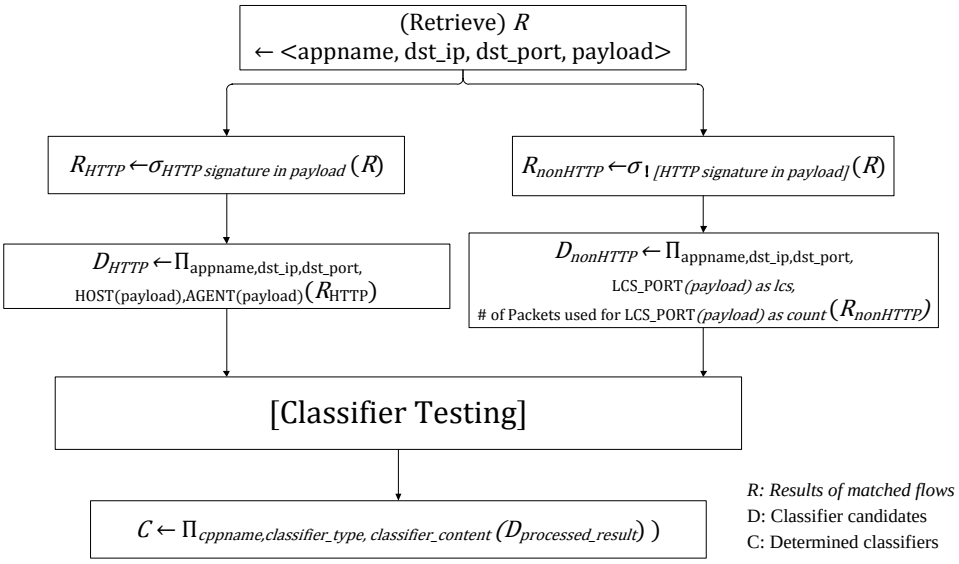

Figure 8. Classifier Generation Procedure for Mobile Applications

Our approach to determining classifiers involves selecting low-cost and key classifiers iteratively for each mobile application and testing whether the selected classifiers can identify mobile applications with acceptable accuracy. Each of these classifiers is based on the characteristics of mobile traffic. They were developed by comparing the cost of existing approaches and then selecting the most efficient ones.. We apply our classifiers in

the following 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 selected classifier candidates. When there is no conflict between 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. Figure 9 shows a flow chart of automated classifier decision algorithm. When none of selected classifier candidates can identify a specific mobile application within the accuracy threshold, the proposed algorithm determines that it cannot classify the mobile application.

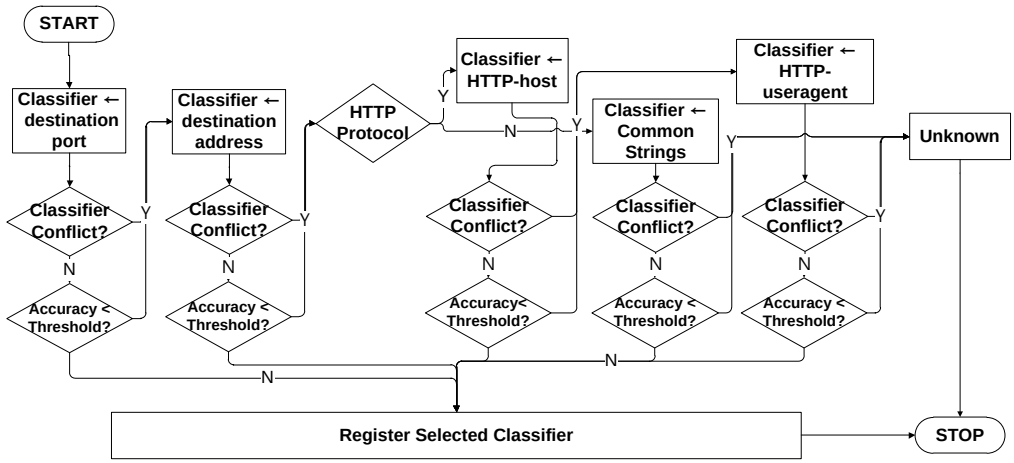

Figure 9. Flow Chart of Automated Classifier Decision Algorithm

Using the four examples of ‘Multimedia player’, ‘Web browser’, ‘KakaoTalk’, and ‘Facebook’ mobile applications, we show how the proposed classifier generation procedure and automated classifier decision algorithm determine classifiers for target mobile applications.

4.2.1 Case 1: Multimedia Player

Table 6. Information on Determining 'Multimedia Player' Classifier

Multimedia Player	# of distinct port number (80, 8080)	2
	# of distinct IP addresses	24
	# of distinct IP subnets	9
	# of distinct HTTP host strings	18
	# of distinct HTTP user-agent strings	2

On the Android platform, 'Multimedia Player' runs a system process that plays all videos excluding Flash videos and animated GIFs. This process runs even when other mobile applications such as YouTube, Daum TV Pot and JJangLive are running. Table 6 shows the number of distinct port numbers, IP addresses, IP subnets, and 'HTTP Host' & 'HTTP User-agent' fields when we played several multimedia videos using several different mobile applications. The proposed automated classifier generation algorithm tests the low-cost classifiers first and then continues to the high-cost classifiers using the following steps:

- The number of distinct port numbers is small; that is, destination port numbers can be a candidate for the 'Multimedia Player' classifier.
  - The ground truth data contains 62 flows for 'Multimedia Player'. However, when we test destination port numbers as a potential classifier, 1,721 flows are classified, which is higher than ground truth data. In this case, the algorithm detects 2776% more flows than those contained in the ground truth data and we argue that these port numbers should not be used as a classifier for this mobile application.
- There are more than 24 distinct IP addresses, and IP addresses are difficult to use to classify 'Multimedia Player'. The number of distinct IP subnets and 'HTTP Hosts' is rather big, which means that they cannot be used to classify 'Multimedia Player' either.
- The number of distinct values of the 'HTTP user-agent' field is only 2, so it can

be a candidate for the ‘Multimedia Player’ classifier.

- The ground truth data for HTTP traffic contains 62 flows and this classifier identifies exactly 62 flows, which is the same. We can determine that two ‘HTTP user-agent’ strings can be an effective classifier for this mobile application.

4.2.2 Case 2: Web Browser

Table 7. Information on Determining ‘Web Browser’ Classifier

Web Browser	# of distinct port number (80, 443, 8080)	3
	# of distinct IP addresses	187
	# of distinct IP subnets	84
	# of distinct HTTP host strings	65
	# of distinct HTTP user-agent strings	6

Table 7 shows the number of distinct port numbers, IP addresses, IP subnets, and ‘HTTP Host’ & ‘HTTP User-agent’ fields for the HTTP protocol ‘Web Browser’ mobile application flows within the target traffic. With this information, the proposed automated classifier generation algorithm runs as follows:

- The number of distinct port numbers is small; that is, destination port numbers can be a candidate for the ‘Web Browser’ classifier.
  - The ground truth data contains 1,107 flows for ‘Web Browser’. However, when we test destination port numbers as a potential classifier, 2,066 flows are classified, which is higher than the ground truth data. In this case, the calculated accuracy is 186.63% and we argue that using these port numbers as a classifier over-detects this mobile application, which is a conflict.
  - Therefore, port numbers should not be used as a classifier for ‘Web Browser’.
- The number of distinct IP addresses is too large, which means that IP addresses

cannot be a candidate for the ‘Web Browser’ classifier.

- The number of distinct IP subnets is also too large, which means that IP subnets cannot be a candidate for the ‘Web Browser’ classifier.
- The number of distinct values of the ‘HTTP host’ field is too large, which means that the ‘HTTP host’ cannot be a candidate for the ‘Web Browser’ classifier.
- The number of distinct values of the ‘HTTP user-agent’ field is only 6, so it can be a candidate for the ‘Web Browser’ classifier.
    - The ground truth data for HTTP traffic contains 1,069 flows and this classifier identifies 1,240 flows. Thus, its calculated accuracy is 116.0%. This value is acceptable for identifying ‘Web Browser’, so the ‘HTTP user-agent’ can be a classifier for this mobile application.

4.2.3 Case 3: KakaoTalk

Table 8. Information on Determining ‘KakaoTalk’ Classifier

KakaoTalk	# of distinct port number (443)	1
	# of distinct IP addresses	12
	# of distinct IP subnets	2

Table 8 shows the number of distinct port numbers, IP addresses, and IP subnets for non-HTTP protocol ‘KakaoTalk’ mobile application flows within the target traffic. Unlike ‘Web Browser’, KakaoTalk does not use HTTP protocol traffic but uses encrypted traffic using the HTTPS protocol. With this information, the proposed automated classifier generation algorithm runs as follows:

- There is only one distinct port number; that is, the destination port number can be a candidate for the ‘KakaoTalk’ classifier.
    - The ground truth data contains 133 flows for ‘KakaoTalk’. However, when we test destination port numbers as a potential classifier, 345 flows are classified, which is higher than ground truth data. In this case, the calculated accuracy value is 259.4% and we argue that using this port number as a

classifier over-detects this mobile application, which is a conflict. Therefore, port numbers should not be used as a classifier for ‘KakaoTalk’.

- There are 12 different IP addresses, which means that IP addresses cannot be a candidate for the ‘KakaoTalk’ classifier.
- Since there are two distinct IP subnets, we test IP subnets as a candidate for the ‘KakaoTalk’ classifier. Then, exactly 133 flows are classified, which is the same as the ground truth data. Therefore, IP subnets can be a classifier for the ‘KakaoTalk’ mobile application.

4.2.4 Case 4: Facebook

Table 9. Information on Determining ‘Facebook’ Classifier

Facebook	# of distinct port number (80, 443, 5222)	3
	# of distinct IP addresses	39
	# of distinct IP subnets	10
	# of distinct ‘HTTP Host’ strings	2
	# of distinct ‘HTTP User-agent’ strings	3
	Ratio of HTTP traffic	39.33%
	Ratio of non-HTTP traffic	60.67%

Table 9 shows the number of distinct port numbers, IP addresses, and IP subnets for non-HTTP protocol ‘Facebook’ application flows within the target traffic. Unlike ‘Web Browser’, ‘Facebook’ does not use HTTP protocol traffic but uses encrypted traffic using the HTTPS protocol. With this information, proposed automated classifier generation algorithm runs as follows:

- The number of distinct port number is small; that is, destination port numbers can be a candidate for the ‘Facebook’ classifier.
  - The ground truth data contains 334 flows for ‘Facebook’. However, when we test destination port numbers as a potential classifier, 2,046 flows are classified, which is much higher than the ground truth data. In this case, the calculated accuracy value is more than 600% and we argue that using port

numbers as a classifier over-detects this mobile application, which is a conflict. Therefore, port numbers should not be used as a classifier for the 'Facebook' mobile application.

- There are more than 30 distinct IP addresses, which means that IP addresses cannot be a candidate for the 'Facebook' classifier.
- It is difficult to determine whether IP subnets are a suitable classifier for this application because there are 10 distinct IP subnets, which is a large number
- There are only 2 distinct values for the 'HTTP host' field, so this field can be a candidate for the 'Facebook' classifier.
  - The ground truth data for HTTP traffic contains 186 flows, and this classifier identifies 235 flows, which calculated accuracy value is 126%. This level of accuracy is acceptable for identifying HTTP mobile traffic for 'Facebook', so 'HTTP Host' can be a classifier for this mobile application.
- For non-HTTP traffic, we tested whether or not extracted common payload strings can classify the 'Facebook' mobile application. There are 147 ground truth flows for this application, but 2,396 flows are classified using common payload strings, which is too large. The proposed algorithm cannot determine a classifier for non-HTTP traffic.

5 Validation

The section validates our proposed mTMA traffic extract algorithm and evaluates the accuracy of the automated system we have developed to generate classifiers. In this evaluation, we apply our system to our campus network environment.

5.1 Validation of mTMA Traffic Extract Algorithm

As shown in Figure 7, the proposed heuristic flow matching algorithm finds mTMA traffic which destination IP address, destination port number, and protocol types are the same as backbone traffic. Then, packet numbers in a flow, total transferred bytes in a flow and payload data are used. An mTMA collects only the first few bytes of each packet's payload and this information also allows us to locate mTMA traffic more accurately in the backbone trace. Table 10 shows the overall rate of flow matching that uses capturedtraces. Although proposed algorithm cannot find all of the flows located on mobile devices, the rate may be sufficient to provide input and ground data for the other algorithm proposed in Section 4.2. Payload inspection is a costly operation, but when payload matching is used to find mobile device traffic in the backbone trace, it increases the rate of flow matching.

Table 10. Overall Flow Matching Rate using Captured Trace

	Matching with payload	*Matching without payload*
# of matched flows (total: 2,692)	2,257	2,396
Ratio	83.84%	89.00%

Table 11 shows each flow matching ratio for different mobile applications. In this case, the proposed algorithm does not match a high percentage of flows from 'Web Browser' and 'KakaoTalk' with the ground truth data. From this experiment result, we assume that most mobile device users use these two mobile applications often and that they connect to

a similar set of mobile web pages. We also assume that the HTTP requests from one mobile web browser are the same as those from a browser running on another mobile device. On the other hand, the 'KakaoTalk' mobile application uses encrypted network traffic based on the HTTPS application protocol. The encrypted flows from this application show similar flow size or number of packets in a flow. Also, there would only be a few encrypted payload bytes for messaging and SNS, which is difficult to find meaningful payload data. To check these two assumptions, we calculated the coverage of mTMA traffic for each of the steps of our proposed algorithm and for each mobile application. The result of these calculations is presented in Table 12.

Table 11. The Effect of Payload Flow Matching: Number of Flows and Rate

Category	*Application Name*	*# of generated flows*	*Matching without payload*		*Matching payload*	
			# of flows	*rate*	*# of flows*	*rate*
Multimedia	Multimedia player	62	60	96.77%	62	100.00%
	YouTube	56	55	98.21%	55	98.21%
	JJangLive	30	30	100.00%	30	100.00%
Browser	Web Browser	1357	1038	76.49%	1107	81.58%
	SearchBox	10	10	100.00%	10	100.00%
Messaging & SNS	Facebook	345	329	95.36%	334	96.81%
	KakaoTalk	155	111	71.61%	133	85.81%
Market	Android Market	54	52	96.30%	52	96.30%
	T Store	30	28	93.33%	30	100.00%
	(Other)	293	274	93.52%	291	99.32%
		300	270	90.00%	292	97.33%

The proposed algorithm utilizes features from the lowest cost ones to the most complex ones. First, it considers the number of packets and bytes transferred in a flow, and payload data. For 'Web Browser' and 'KakaoTalk', matching the number of packets is not powerful way, and matching the number of packets and flow size do not strengthen the matching rate in Table 12. However, there is low matching rate of payload inspection for 'Web Browser', and 'KakaoTalk' shows the highest rate of inspecting payload bytes. This

fact supports our two assumptions because the number of packets and total transferred bytes in a flow are nondeterministic values and multiple different message flows may show the same features if the encrypted message format is a fixed message format. A small difference in the number of payload bytes between two flows from the 'KakaoTalk' mobile application is captured through payload inspection step in our proposed algorithm.

Table 12. Coverage of mTMA Traffic in Each Step

Category	*Application Name*	*Packet number & Flow size*	*Packet number (exact->more->less)*	*Payload inspection*
Multimedia	Multimedia player	11.29%	25.81% → 88.71%	0.00%
	YouTube	0.00%	80.36% → 98.21%	0.00%
	JJangLive	0.00%	96.67% → 100%	3.33%
Browser	Web Browser	6.48%	66.32% → 71.55%	3.54%
	SearchBox	0.00%	100%	0.00%
Messaging & SNS	Facebook	19.42%	73.62% → 76.81%	0.58%
	KakaoTalk	7.10%	33.55% → 65.81%	12.90%
Market	Android Market	9.26%	50.00% → 90.48%	0.00%
	T Store	9.90%	66.21% → 86.69%	2.73%
	(others)	10.33%	78.33% → 94.33%	2.67%

5.2 Validation of Extracted Classifiers

Table 13 shows the mobile application identification result when we use classifiers extracted by our proposed classifier generation algorithm. With the exception of Facebook mobile application, the proposed automated classifier generation algorithm created classifiers with high coverage of ground-truth traffic data for all mobile applications. It is estimated that more than 60% of the non-HTTP traffic is used in Facebook mobile application (Section 4.2.4) and it is difficult for our algorithm to extract meaningful common payload bytes from this non-HTTP Facebook traffic. Another low-coverage result is related to the 'com.android.vending' process used in the Android Market mobile application. The value is 53.33% for flow numbers and 95.50% for

transferred bytes. 14 un-classified (46.67%) flows of this process are 94.50% of transferred bytes.

Table 13. Coverage of Generated Classifiers for Mobile Application Identification

Mobile Application Name	*Number of flows*			*Flow size*		
	Ground truth	*Classified*	*Rate*	*Ground truth*	*Classified*	*Rate*
Media player	62	59	95.16%	378,228,999	361,731,404	95.64%
YouTube	55	47	85.45%	5,953,064	5,866,006	98.54%
JJangLive	30	30	100.00%	211,090	211,090	100.00%
Web browser	1,107	1,072	96.84%	16,313,238	16,219,363	99.42%
SearchBox	10	9	90.00%	17,015	16,941	99.57%
Facebook	334	180	53.89%	1,925,338	745,304	38.71%
KakaoTalk	133	133	100.00%	6,370,146	6,370,146	100.00%
Android Market #1	52	52	100.00%	126,897,330	126,897,330	100.00%
Android Market #2	30	16	53.33%	2,005,156	1,914,903	95.50%
T Store	291	291	100.00%	94,138,064	94,138,064	100.00%

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

$$Recall = \frac{[true\ positive]}{[true\ positive] + [false\ negative]} \quad (3)$$

$$Accuracy = \frac{[true\ positive] + [true\ negative]}{all\ traffic\ used\ for\ traffic\ classification} \quad (4)$$

We calculated the accuracy of generated classifiers in terms of precision and recall. Precision (2) is the fraction of traffic results that are relevant to the classification and recall (3) is the fraction of traffic results that are relevant to the classifier successfully created. Accuracy (4) of traffic identification includes true positive and true negative traffic from all traffic used for identifying the targeted mobile application(s). Table 14 shows the number of flows and Table 15 shows flow size for true positive, true negative, false positive and false negative traffic and the calculated precision, recall and accuracy. The over accuracy of traffic identification using automatically generated classifiers is 76.20% for the number of flows and 93.84% for flow size, which is acceptable.

Table 14. Precision, Recall and Accuracy (Number of Flows)

Mobile Application Name	*True positive*	*False negative*	*True negative*	*False positive*	*Precision (%)*	*Recall (%)*	*Accuracy (%)*
Media player	62	0	2334	0	100.00	100.00	100.00
YouTube	55	0	2341	0	100.00	100.00	100.00
JJangLive	25	5	2364	2	92.59	83.33	99.71
Web browser	1068	39	1125	164	86.69	96.48	91.53
SearchBox	9	1	2386	0	100.00	90.00	99.96
Facebook	182	152	2060	2	98.91	54.49	93.57
KakaoTalk	133	0	2263	0	100.00	100.00	100.00
Android Market #1	52	0	2344	0	100.00	100.00	100.00
Android Market #2	16	14	2366	0	100.00	53.33%	99.42
T Store	291	0	2097	8	97.32	100.00	99.67
(Total)	1,893	211	116	176	91.49	89.97	**83.85**

Table 15. Precision, Recall and Accuracy (Flow Size)

Mobile Application Name	*True positive (KB)*	*False negative (KB)*	*True negative (KB)*	*False positive (KB)*	*Precision (%)*	*Recall (%)*	*Accuracy (%)*
Media player	369,364	0	253,809	0	100.00	100.00	100.00
YouTube	5,814	0	617,360	0	100.00	100.00	100.00
JJangLive	52	155	622,961	6	89.18	25.01	99.97
Web browser	15,822	109	604,513	2,729	85.29	99.32	99.54
SearchBox	17	0	623,156	0	100.00	99.57	100.00
Facebook	750	1,130	621,213	80	90.34	39.88	99.81
KakaoTalk	6,221	0	616,952	0	100.00	100.00	100.00
Android Market #1	123,923	0	499,250	0	100.00	100.00	100.00
Android Market #2	1,870	88	621,215	0	100.00	95.50	99.99
T Store	91,932	0	530,751	490	99.47	100.00	99.92
(Total)	615,763	1,482	2,622	3,306	99.47	99.76	**99.23**

We estimate the population proportion with 95% confidence. The confidence interval is (5). The overall estimated identification accuracy is 83.85±1.4736%. Table 16 shows the confidence interval for mobile applications.

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

Table 16. Estimating Confidence for Mobile Application Identification Result

Category	*Mobile Application Name*	*Confidence (95%)*
Multimedia	Media player	100.00±0.0000%
	YouTube	100.00±0.0000%
	JJangLive	99.71±0.2161%
Browser	Web browser	91.53±1.1151%
	SearchBox	99.96±0.0818%
Messaging & SNS	Facebook	93.57±0.9820%
	KakaoTalk	100.00±0.0000%
Market	Android Market	99.42±0.3052%
	T Store	99.67±0.2310%

Finally, we performed mobile application-level traffic classification using the generated classifiers (Appendix A). The trace from one day (April 16th, 2011), *trace1*, was captured with the entire packet payload. Figure 10 shows the relative volume of mobile application traffic over 24 hours. Mobile web browser traffic dominates in target mobile applications and multimedia player traffic increases rapidly in the evening and around midnight. Web browser traffic is generally spread evenly over the full day. In addition, users usually search for and download mobile applications through Android Market or T Store at lunch time or around midnight.

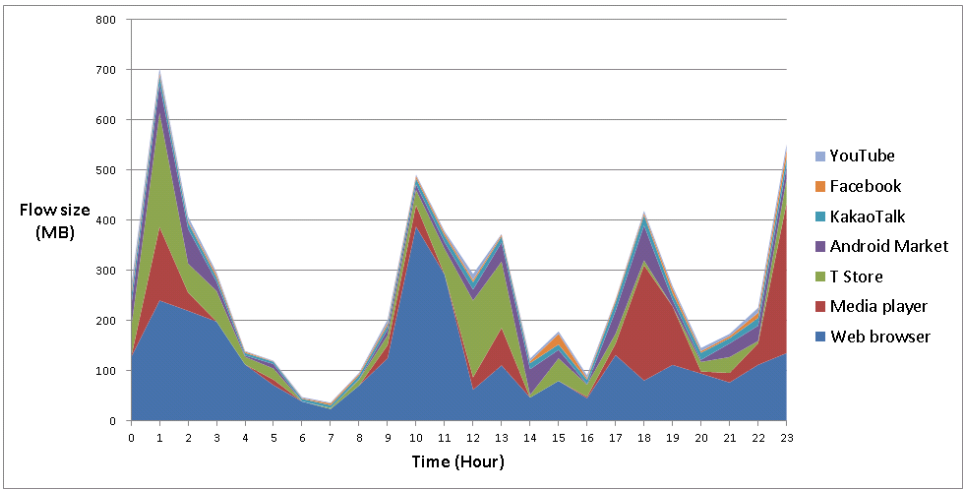

Figure 10. Proportion of Mobile Application Traffic Volume over 24 Hours

6 Conclusions

6.1 Thesis Summary

Although there have been various studies on application traffic identification, to our knowledge, none of research has investigated accurate mobile application identification. As the number of mobile devices and the volume of traffic they causes 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 focused on overcoming some limitations of taking measurements on a mobile device and the automated classifier generation algorithm aimed at reducing human effort and providing accurate or detailed application-level information on mobile traffic.

Section 1 presented an introduction to the thesis and the goal for the research. Identifying mobile applications in mobile networks is challenging because there are packet losses when we capture packets in mobile devices and the majority of Wi-Fi APs have mostly NAT functions.

Section 2 surveyed studies on mobile traffic analysis and the three main types of application traffic identification algorithms. In order to select candidates for mobile traffic classifiers, we also provided results of mobile traffic analysis and discussed the advantages and shortcomings of each type of identification algorithm.

Section 3 provided a number of definitions to ensure that key terms were used clearly and consistently throughout the thesis. This section also described the architecture of the automated classifier generation system using mTMAs. The proposed architecture utilizes flow information from the captured traffic in the backbone.

Section 4 proposed two algorithms related to proposed architecture. One algorithm finds mTMA traffic in the backbone by using from low-cost to costly information. The other algorithm generates classifiers by automatically selecting low-cost classifiers and testing whether or not they can identify the targeted mobile application. In the validation process, we checked the matching rate for the mTMA traffic find algorithm and analyzed

precision, recall, over accuracy and estimated confidence values for generated classifiers. The rest of this chapter summarizes the contributions of this thesis and future work.

6.2 Thesis Contribution

This thesis makes a number of the following contributions.

The proposed architecture can overcome the limitations imposed by mobile devices and NAT functionality. The proposed heuristic matching algorithm runs well even when NAT functionality prevents it from matching flows in backbone traffic and mobile traffic and when mobile devices collect inaccurate flow information because of limited computation processing power and memory space. In addition, proposed algorithm corrects inaccurate information collected from mobile devices using flow information captured from the Internet backbone.

The proposed automated classifier generation algorithm tries to find the lowest-cost classifiers possible, and it is expected to decrease human effort involved in manually identifying the characteristics of targeted mobile application traffic.

Finally, this thesis applied the generated classifiers to demonstrate that the proposed system can also be used to measure and analyze mobile traffic captured in a router or in a backbone.

6.3 Future Work

In terms of the present research, the most important future task is to increase the accuracy of the generated classifiers. Unfortunately, encrypted mobile traffic cannot be identified using the proposed system. To increase the accuracy, we now consider the followings: First, we will consider the combination of applied classifiers in this thesis. Second, we will find other candidate of classifiers for mobile traffic. Although this thesis does not focus on statistics-based classifiers, it would be worth to investigate whether or not statistics-based classifiers perform mobile traffic identification with high accuracy.

Third, finding more accurate threshold values in the automated classifier generation algorithm can improve this algorithm with higher accuracy.

Other main future work is to develop mTMAs for other mobile device platforms such as iOS, Blackberry and Windows Phone 7 to extract various type of mobile traffic. Furthermore, we will perform comparative application-level measurement analysis between mobile and non-mobile traffic, or Wi-Fi and 3G network traffic to find different characteristics of network traffic.

References

[1] Distimo, “Distimo: The battle for the most content and the emerging tablet market”, April, 2011.

[2] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin, "A First Look at Traffic on Smartphones", Internet Measurement Conference (IMC), 2010.

[3] Byung-Chul Park, Young J. Won, Myung-Sup Kim, and James Won-Ki 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.

[4] Mingjiang Ye, Ke Xu, Jianping Wu, Hu Po, "AutoSig-Automatically Generating Signatures for Applications," cit, vol. 2, pp.104-109, 2009 Ninth IEEE International Conference on Computer and Information Technology, 2009.

[5] G. Szabo, Z. Turanyi, L. Toka, S. Molnar, A. Santos, “Automatic Protocol Signature Generation Framework for Deep Packet Inspection”, ValueTools 2011.

[6] G. Szabo, D. Orincsay, S. Malomsoky, I. Szabo, “On the Validation of Traffic Classification Algorithms”, PAM 2008.

[7] 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, Apr. 5-6, 2007.

[8] G. Maier, F. Schneider, and A. Feldmann, “A First Look at Mobile Hand-held Device Traffic”, Passive and Active Measurement, Zurich, Switzerland, Apr. 7-9, 2010.

[9] 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, Mar. 20-22, 2011.

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

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

[12] M. Roughan, S. Sen, and O. Spatscheck, "Class of Service Mapping for QoS: A Statistical Signature based Approach to IP Traffic Classification", Internet Measurement Conference, Taormina, Sicily, Italy, October 25-27, 2004.

[13] A. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques", ACM SIGMETRICS, Banff, Canada, June 2005.

[14] W. Charles, M. Fabian, M. Gerald, "HMM Profiles for Network Traffic Classification (Extended Abstract)", In Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, Washington, DC, USA, October 29, 2004.

[15] 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.

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

[17] 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, Dec. 2010.

[18] Wireshark, http://www.wireshark.org.

[19] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)", RFC

1631, May 1994.

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

[21] M. Cotton and L. Vegoda, “Special Use IPv4 Addresses”, RFC 5735, January 2010.

[22] 이현신, 김명섭, “HTTP 트래픽을 이용한 운영체제 시그니처 자동 생성에 관한 연구”, 2011 통신망 운용관리 학술대회 (KNOM Conference), 2011년 4월 21일-22일.

[23] Google, Android, http://www.android.com/.

[24] Apple, iPhone App Store, http://www.apple.com/iphone/apps-for-iphone/.

[25] Google, Android Market, https://market.android.com/.

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

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

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

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

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

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

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

[33] "PortLoad: taking the best of two worlds in traffic classification," Giuseppe Aceto, Alberto Dainotti, Walter de Donato, Antonio Pescape, IEEE INFOCOM, 2010.

[34] 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, Oct. 1995, pp. 1481-1494.

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

[36] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015", Feb. 1, 2011.

Appendix A.

Mobile Application Process Name	Protocol	Classifier Type	Classifier	Note
/system/bin/mediaserver	TCP	HTTP User-Agent	stagefright/1.1 (Linux;Android 2.3.3)	Multimedia player in Android
			stagefright/1.1 (Linux;Android 2.3.4)	
android.process.media	TCP	HTTP User-Agent	AndroidDownloadManager	Android Market Process
com.android.browser	TCP	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	Android Web Browser
			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	
			Mozilla/5.0 (Linux; U; Android 2.2.1; ko-kr; SHW-M110S Build/FROYO) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1	
com.android.vending	TCP	HTTP User-Agent	Android-Market/2 (SHW-M110S GINGERBREAD); gzip	Android Market Process

			Android-Market/2 (umts_sholes MIUI); gzip	
com.clov4r.android.nil	TCP	PORT	8060	
com.cubeflux.news	TCP	IP	220.73.138.230	News
			220.73.138.234	
com.dropbox.android	TCP	IP	174.36.30.71	File Sharing
com.estsoft.alyac	TCP	IP	218.153.8.225	Vaccine
com.facebook.katana	TCP	HTTP Host	facebook.com	Facebook for Android
			ak.fbcdn.net	
com.google.android.youtube	TCP	HTTP User-Agent	Android-YouTube/2 (SHW-M110S GINGERBREAD); gzip	YouTube Mobile Application in Android (Multimedia Files are not played using this process.)
			Android-YouTube/2 (umts_sholes MIUI); gzip	
			Android-YouTube/2	
			Apache-HttpClient/UNAVAILABLE (java 1.4)	
com.kakao.talk	TCP	Subnet	110.76.140.0/24	Messaging & SNS mobile application
			203.246.172.0/24	
com.sec.android.widgetapp.infoalarm	TCP	IP	66.114.50.13	
com.skt.pdf.appmerser	TCP	IP	113.217.232.12	
com.skt.skaf.A000Z00040	TCP	PORT	444, 8205, 9104, 9200, 9401	T Store
com.uajjang.android	TCP	IP	110.45.160.213	JJangLive
			110.45.160.215	
			211.115.90.136	
com.weathernews.Weather	TCP	IP	210.116.104.240	Weather

com.whatsapp	TCP	IP	50.22.199.44 50.22.227.224	Messaging & SNS
net.daum.android.air	TCP	IP	61.111.62.174 114.108.158.104	Daum TV Pot

요 약

최근, 스마트폰, 타블렛 등과 같이 다양한 종류의 모바일 장비들이 대중화되면서, 인터넷 접근을 필요로 하는 모바일 어플리케이션의 숫자 또한 증가하고 있다. 네트워크 관리자들은 '모바일 빅뱅' 시대를 대비하기 위해서 어플리케이션 레벨에서 모바일 트래픽을 식별하고자 한다. 그러나 모바일 어플리케이션 숫자의 증가 및 이에 따른 모바일 어플리케이션 트래픽 양의 급격한 증가로 인해 모바일 트래픽을 식별하는 데 있어 많은 어려움이 있다. 보다 적은 노력으로 다양한 모바일 어플리케이션들을 식별하기 위해서는 모바일 어플리케이션들을 높은 정확도를 가지고 자동으로 식별 가능한 방법이 필요하다.

본 논문은 모바일 트래픽 식별에 사용되는 분류자를 자동으로 생성하는 시스템에 대한 아키텍처를 설명한다. 해당 분류자들은 각 모바일 어플리케이션이 생성한 트래픽에 대해 핵심이 되는 특징들을 표현한다. 제안하는 아키텍처는 모바일 장비에 설치되어 어플리케이션 레벨의 모바일 트래픽을 모니터링하는 mTMA (mobile Traffic Measurement Agents) 프로그램을 포함한다. 제안하는 mTMA는 모바일 장비들이 낮은 처리 속도 및 적은 메모리 양을 지니고 있어 어플리케이션 정보에 따른 네트워크 레벨의 플로우 정보를 정확하게 수집하지 못한다. 이를 보완하기 위해 Backbone에서 수집한 플로우 정보를 기반으로 mTMA에서의 불완전한 플로우 정보를 보완하는 알고리즘을 제안하였으며, 해당 결과는 분류자를 자동으로 생성하는 알고리즘의 입력 값 및 정확도를 확인하기 위한 정답지로 동시에 사용된다.

또한, 본 논문에서는 모바일 어플리케이션에 대한 분류자를 자동으로 찾기 위해 비용이 적게 드는 분류 방법 점차적으로 적용하면서 정확도 및 다른 분류자와의 충돌을 자동으로 확인하면서 분류자를 결정하는 알고리즘을 제안하였다. 제안하는 알고리즘을 실행하여, POSTECH 학교 네트워크 트래픽으로부터 많이 사용되는 모바일 어플리케이션에 대한 분류자를 찾아내었다. 검증 과정을 통해 해당 분류자가 적용 가능한지를 확인하였으며, 생성한 분류자들을 학교 네트워크 트래픽에 적용하여 학교 네트워크에서 모바일 어플리케이션에 따른 트래픽 특성을 분석해 보았다.

감사의 글

먼저 제가 연구실 생활을 할 수 있도록 저를 받아주시고 2년 동안 부족한 저를 아껴주시고 지도해 주신 홍원기 교수님께 진심으로 감사드립니다. 인생의 선배로서 들려주신 교수님의 조언을 통해 큰 힘을 낼 수 있었고, 앞으로도 계속 최선을 다하여 열심히 하고자 합니다. 또한 바쁘신 와중에도 깊은 관심과 정성으로 논문 심사를 맡아 주신 김종 교수님과 서영주 교수님께도 감사의 인사를 드립니다. Also, I am very thankful to Prof. John Strassner. I hope he will recover from a sinus infection as soon as possible.

DPNM 연구실에서 함께 생활한 식구들의 도움과 응원이 없었다면 지금의 제가 없었을 것입니다. 곧 있으면 포항을 떠나시는 듬직하신 준명이형, 현재는 대구에서 열심히 연구 중이신 성철이형, 영원한 2분반 선배 병철이형, 랩장이시면서 체력 코치까지 친절하게 해 주시는 성수형, 우리 연구실에서 가장 모범적이신 신석이형, 그리고 시끌벅쩍 재윤이, 반면 조용하지만 재미있는 혁수, 입학 동기면서 박사까지 함께 할 술 친구 건이, 이쁜 아름이, 시크한 윤선이, 그리고 앞으로도 함께 할 태현이, Yongfeng, Do, 태열이, 모두 정말 감사합니다. 앞으로도 같은 DPNM 연구실 선후배 동기로서 함께하여 모두 보다 나은 결실을 이루고자 열심히 하겠습니다. 석사 생활 중 짧지만 연구실 생활을 함께 하였던 영준이형과 진이형도 많이 생각납니다. 또한, 연구실 행정을 신경 써 주시는 혜정 누나께도 깊은 감사의 인사를 드리고자 합니다.

무엇보다 병철이형과 재윤이가 아니었다면 본 석사 논문을 완성할 수 없었을 것입니다. 모바일 트래픽 분야에서 논문의 방향을 정할 때, 고민할 때, 방황할 때 저에게 큰 힘이 되어 주었던 병철이형, 그리고 트래픽 분야에서 모르는 것이 있을 때마다 언제나 같이 고민해 주던 재윤이의 고생 덕에 작은 결실을 맺을 수 있었습니다. 그리고 mTMA 개발을 하느라 고생 많았던 학부 과정 거성이에게도 큰 고마움을 느낍니다.

석사 생활 중 많은 힘이 되었던 방돌이 종훈이와 제3의 방돌이인 병일이, 여전히 학교에서 함께 생활하고 있는 2분반 동기들, 멀리 떨어져 있지만 같이 재미있게 고민하는 NoTag 멤버들, 부족한 ITCE 학생 대표와 함께 새로운

ITCE 학과에서 함께 생활하고 있는 ITCE 입학 동기들 및 학과 선생님들께도 감사의 인사를 드리고 싶습니다.

자주 연락 드리지 못하지만 언제나 저를 사랑해 주시고 믿어 주시는 아버지, 어머니, 이 지면을 빌어 정말 감사와 함께 죄송하다는 말을 전하고자 합니다. 비록 앞으로도 몇 년 더 떨어져 생활하겠지만, 열심히 하여 부모님 은혜에 보답하고 효도하고자 노력하겠습니다. 그리고 나 대신 부모님과 많은 생활하고 있는 남동생에게, 형이 항상 함께 생활하지 못해 지금은 미안하지만 서로 서로 열심히 하자꾸나.

이상 제가 언급하지 못하였지만, 본 지면을 작성하면서 어렸을 적부터 대학 학부에서의 생활, 3년 간의 대전 애니솔루션 및 서울 애니파크에서의 생활, 그리고 1년간의 연구 참여 및 2년 간의 석사 생활 동안 인연을 맺었던 수많은 분들이 한 분 한 분 생각납니다. 여러 일들이 있었지만 지난 과거를 발판으로 삼아 더욱 열심히 하여 은혜에 보답할 줄 아는 사람이 되겠습니다.

이 력 서

성　　명 : 최 영 락

생년월일 : 1983년 12월 28일

출 생 지 : 충북 청주시

주　　소 : 경북 포항시 남구 효자동 산 31 포항공과대학교 RIST 4-4405

학 력

2002.3 ~ 2009.8 포항공과대학교 컴퓨터공학과 (B.S.)

2009.9 ~ 2011.8 포항공과대학교 정보전자융합공학부 (M.S.)

경 력

2005.9 ~ 2007.2 ㈜ 애니솔루션, Developer

2007.3 ~ 2008.8 ㈜ 애니파크, Server Developer

학 술 활 동

International Conference Papers

1. Yeongrak Choi, Jian Li, Yoonseon Han, John Strassner and James Won-Ki Hong, "Towards a Context-Aware Information Model for Provisioning & Managing Virtual Resources and Services", Lecture Notes in Computer Science, Vol. 6473, Modelling Autonomic Communication Environments, 5th International Workshop on Modelling Autonomic Communication Environments (MACE 2010), Niagara Falls, Canada, Oct. 28, 2010, pp. 100-112.

Domestic Journal Papers

1. 최영락, 리건, 홍원기, "가상화와 데이터 관리 기술들에 대한 확장성 및 신축성 평가," KNOM Review, vol. 12, no. 1, Dec. 2009, pp. 40-48.

Domestic Conference Papers

1. 최영락, 리건, 한윤선, 홍원기, "가상 지원 및 서비스 관리를 위한 상황 인지 정보 모델", KNOM Conference 2011, Pohang, Korea, Apr. 21-22, 2011.
2. 정재윤, 최영락, 박병철, 홍원기, "모바일 트래픽의 모니터링 및 분석", KNOM Conference 2011, Pohang, Korea, Apr. 21-22, 2011.
3. 리건, 최영락, 홍원기 "OSGi 지능형 홈 네트워크 서비스를 위한 효율적인 자원 관리 방안", 한국통신학회 하계학술대회, 제주, 2010년 6월 23-25일, 2010.
4. 최영락, 강준명, 리건, John Strassner, 홍원기, "클라우드 컴퓨팅 기반의 통합 대중 교통 수단 검색 시스템", 한국통신학회 하계학술대회, 제주, 2010년 6월 23-25일, 2010.

연 구 활 동

1. HiMang(Highly Manageable Network and Service Architecture for New Generation) 원천 연구, ETRI, 2010 ~ 2011.
2. 시맨틱 기반의 VoD 추천 시스템, KT, 2010.