Fine-grained Traffic Classification based on Functional Separation

2012

B PARK

Doctoral Thesis

Fine-grained Traffic Classification based on Functional Separation

Byungchul Park (박 병 철)

Division of Electrical and Computer Engineering
(Computer Science and Engineering)
Pohang University of Science and Technology
2012

기능 구분을 통한 세분화된 응용 트래픽 분류

Fine-grained Traffic Classification based on Functional Separation

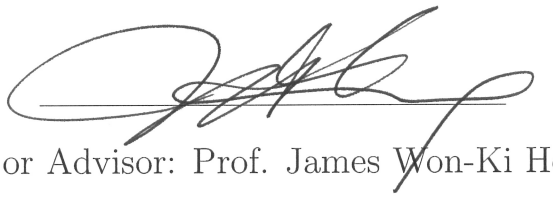
Fine-grained Traffic Classification based on Functional Separation

by

Byungchul Park
Division of Electrical and Computer Engineering
(Computer Science and Engineering)
Pohang University of Science and Technology

A thesis submitted to the faculty of Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Division of Electrical and Computer Engineering (Computer Science and Engineering).

Pohang, Korea
December 16, 2011
Approved by



Major Advisor: Prof. James Won-Ki Hong

Fine-grained Traffic Classification based on Functional Separation

Byungchul Park

The undersigned have examined this dissertation and hereby certify that it is worthy of acceptance for a doctoral degree from POSTECH.

December 16, 2011

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

DECE 20065383 Byungchul Park, 박 병 철, Fine-grained Traffic Classification based on Functional Separation. 기능 구분을 통한 세분화된 응용 트래픽 분류, Division of Electrical and Computer Engineering (Computer Science and Engineering), 2012, 125P, Advisors: James Won-Ki Hong. Text in English.

ABSTRACT

The most critical and representative issue that exists in today's Internet traffic classification is achieving a high-level of accuracy and completeness. A decade of research on traffic classification has provided various methodologies to investigate the traffic composition in data communication networks. Many variants or combinations of such methodologies have been continuously introduced to improve the classification accuracy and efficiency. However, due to the fast-changing nature of the Internet traffic, it is extremely difficult for any method to achieve 100% accuracy and completeness. This indicates that traffic classification research still has a lot of room for improvement.

Another issue in traffic classification is the analysis of traffic classification results. Previous studies have discussed various classification methodologies. Yet, the level of classification details is often subject to identifying protocols or applications in use. The main causes of low accuracy and completeness are new types of network applications and their complex characteristics. In recent years, the majority of traffic classification studies have focused on detecting major applications such as P2P and streaming applications, which occupy most of the Internet traffic. Furthermore, detecting P2P or streaming also has concentrated on classifying main functions that generate a great portion of traffic workload.

In this context, this thesis proposes a new traffic classification scheme called a fine-grained traffic classification based on the analysis of existing classification

methodologies. Our unique traffic scheme can increase traffic classification accuracy and completeness by reducing the amount of undetected traffic and provide more in-depth classification results for various analyses which are unable to be achieved by current traffic classification schemes and methods. The key to the fine-grained traffic classification is classifying network flows into different functional groups based on origin functions in an application. To achieve this, we also propose functional separation method. By applying this method we are able to detect different types of traffic generated by a single application according to their functionalities. The fine-grained traffic classification based on functional separation will potentially increase the amount of information that can be obtained by traffic classification and lay a cornerstone in the foundation of applying traffic classification in user-behavior analysis.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter I

INTRODUCTION

This chapter provides a brief introduction to Internet application traffic classification. The motivation and problems with current approaches to Internet application traffic classification are illustrated in the outline of our proposed approach.

1.1 Background

Network traffic classification is one of the most essential steps to understanding the current network status. As an important part of network operations and management, the traffic classification statistics are widely utilized for various network management purposes such as network planning, network usage reporting, and usage-based charging. Traffic classification is also highly relevant to Quality of Service (QoS) and Service Level Agreement (SLA) monitoring. For example, a network operator can guarantee a certain level of QoS and minimize SLA violations by assigning various types of application traffic to a proper class of service. Moreover,

traffic classification plays an important role in user behavior analysis, which has been attracting increasing attention among researchers as well as in the private sector in the recent years. It benefits service providers by helping them comprehend the behavior of their customers and provide more personalized services to achieve greater customer satisfaction.

As the Internet perpetually evolves in complexity and scope, its traffic characteristics also continue to change in terms of composition and volume. Accordingly, in response to the evolution of the Internet, the research community has presented and explored various methodologies for investigating the traffic composition in data communication networks. Despite these efforts, however, the high-speed evolution and the dynamic nature of the Internet prove it very difficult to cope with new characteristics of the Internet and its traffic. As a result, traffic classification remains an unsolved challenge. In the early days of the Internet, traffic classification was often motivated by certain prevailing protocols in networks and relied on the well-known TCP/UDP port numbers. However, nowadays this well-known port mapping strategy can no longer guarantee the accuracy of results [1], as the evolution of the Internet has brought and will continue to bring new traffic characteristics including unfamiliar traffic composition, volume, and application trends.

In contrast to its early days when most of the Internet traffic consisted of traditional network protocols such as HTTP, TELNET, SMTP, FTP, and SNMP, a significant portion of today's Internet traffic comprises various new types of traffic including peer-to-peer (P2P), voice-over-internet protocol (VoIP), and multimedia traffic. In particular, P2P application is often combined with many different obfuscation strategies, such as ephemeral port allocation, to avoid detection and filtering.

A popular communication application, Skype, for instance, eludes detection by payload encryption or plain-text ciphers. Therefore, there has always been a call for more sophisticated and advanced methodologies to classify application traffic. It has driven the research community to constantly search for and produce solutions.

1.2 Motivation and Problem Statements

The most critical and representative issue that exists in today's Internet traffic classification is achieving a high-level of accuracy and completeness. Accuracy in this context is how correctly the fraction of traffic is classified according to its original applications. There are many metrics such as False Positive (FP), False Negative (FN), precision, and recall that can be used to evaluate the accuracy of classification methodologies or systems. Completeness, on the other hand, is the percentage of the traffic classified by a certain classification methodology. Completeness of traffic classification is mainly related to unknown traffic patterns and has correlation with the FN ratio.

It is highly desirable to maximize both accuracy and completeness during traffic classification, and many variants of the established methodologies have been continuously introduced to improve the accuracy, completeness, and efficiency of classification. However, as aforementioned, it is extremely difficult for any method to claim 100% of accuracy and completeness due to the fast-changing nature of Internet traffic. There is still a lot of room for improvement in traffic classification.

The main causes of low accuracy and completeness are new types of network applications and their complex characteristics. Traffic classification functions as a

filtering process, using predefined filters or classification models (i.e., port numbers, applications signatures, connection patterns, statistical models); hence, the limited number of predefined filters results in the low level of completeness. Undoubtedly, constructing filters for every network application can solve the completeness problem, but it is extremely difficult or impossible in a practical sense due to the large number of applications that is still growing rapidly.

Another cause of low accuracy and completeness is the emergence and evolution of the newer generations of applications like P2P and multimedia applications. A significant number of these applications employ mystification techniques, such as ephemeral/random port allocation, data encryption, and adopting proprietary protocols, in order to avoid detection or filtering. Moreover, functionalities that are incorporated into a single application are also becoming more intricate. A case in point is Microsoft's MSN messenger application. Old versions of MSN (prior to version 3) had only supported simple functions in a limited range such as plain text messaging, e-mail notification (Hotmail), and contact lists update. On the other hand, the latest version of MSN now provides numerous advanced functions including video/voice call, file transfer, mobile games, and remote assistance, just to name a few. The ever-increasing complexity and multiplicity of applications preclude detecting a complete set of traffic generated even from a single application.

In general, the current studies on traffic classification mainly grapple with detecting major applications including P2P and streaming applications that occupy most of today's Internet traffic. Furthermore, detecting P2P or streaming itself is also heavily weighted toward classifying a few main functions of the major applications (e.g., file transfer in P2P) that generate the greatest volume of traffic workload.

Nevertheless, the applications simultaneously generate all different kinds of traffic besides the prominent ones of high-volume traffic. As a result, the lack of interest in minor traffic classes and consequential neglect of them inevitably undermine the accuracy and completeness of classification, especially in face of newly emerging network applications.

Another issue in traffic classification is the analysis of traffic classification results. Many previous studies have suggested various classification methodologies (e.g., well-known port number matching, payload contents analysis, machine learning, etc.), from which even more variants have been derived. However, it is extremely difficult for any method to claim 100% accuracy due to fast-changing and dynamic nature of the Internet traffic. The classification accuracy is also questionable since there is often no ground truth dataset available. In another respect, each research aims at different levels of classification. Some only had a coarse classification goal such as classifying traffic protocol or application type; while others had more detailed classification goal such as identifying the exact application name. Therefore, it is often unfair to cross-compare each classification method in terms of accuracy. To overcome this issue, it is more important to investigate how we can provide more meaningful information with such limited traffic classification results rather than to struggling to overimprove 1 or 2% of classification accuracy.

In this context, we propose a new approach based on functional separation and fine-grained traffic classification in order to achieve a higher level of classification accuracy and completeness. The functional separation method identifies and sorts out different types of traffic generated by a single application according to their functionalities. By doing so, it reduces the amount of undetected traffic and corre-

spondingly increases the completeness of traffic classification. In other words, the outputs of this process are various groups of traffic, neatly categorized and labeled in accordance with their functions. Then, classification filters are extracted out of the respective traffic groups in preparation of the next stage, the fine-grained traffic classification.

The fine-grained traffic classification further classifies the diverse traffic groups, which are the outcomes from the previous stage of functional separation. Until now, previous studies have shown discrepancies between their goals and standards for classification. One may aim for general classification and be only concerned with notions such as whether it is transaction-oriented, bulk-transfer, or peer-to-peer file sharing. On the other hand, one may have a finer-grained classification goal and attempt to identify the exact application represented by the traffic. We believe that our new traffic classification scheme enables more in-depth analysis on traffic classification results. While top n protocol or application analysis is possible with other schemes, our method allows for new analysis categories like the average of browsing time taken to initialize a file download or the most popular functions among Internet users. It is also a tool to analyze user behavior and to design future Internet-based applications. When applied to web traffic, it is also possible to analyze the most popular function of websites The new possibilities allowed by our new approach will broaden the realm of the traffic classification research beyond that of network administrative oriented studies to user context dependent research.

1.3 Research Approach

Considering the problems with previous studies and their methods, as aforementioned, we have developed a new solution for traffic classification. In this section, we explain the solution to the critical problems of current traffic classification schemes mentioned in previous section and methodologies.

In regards to both problems, accuracy and completeness problem and lack of information problem, we suggest a new method to classify the current Internet traffic. First, we begin with reviewing current traffic classification schemes from the classification level point of view and establishing a common understanding of the traffic classification level.Many of previous traffic classification studies do not have a shared definition, if any, on their traffic classification level. We categorize existing traffic classification works into four different levels. Considering the classification levels and their limitations, we define a new traffic classification scheme called a fine-grained traffic classification. The main idea of this fine-grained traffic classification comes from the fact that there exist some differences among flows belonging to a certain application according to their functionalities.

The key to the fine-grained traffic classification is discriminating network flows based on their origin functions in an application. To resolve this issue, we developed a method called a functional separation method, which classifies flows into several functional traffic groups. This functional separation method is composed of three crucial steps. The functional separation method first requires a cleaned traffic trace as its input data. The cleaned traffic trace here refers to network traffic that only belongs to the target application. In order to obtain the cleaned traffic trace, we

developed a packet of dump agents for Windows and Android. The dump agents can collect a certain type of application's traffic using port allocation and process information, which can also be obtained from the operating system.

The first step of functional separation, Port-Relation Grouping (PRG), identifies the dependency on allocated port numbers and aggregates flows according to the dependency. Then follows Contents-Relation Grouping (CRG) that measures the content similarity among flows and assorts them again based on the results of their content similarity. Finally, Contents-Relation Decomposition (CRD) divides the flow groups composed in the previous steps. This decomposition step is essential to eliminate any uncertainty caused by misclassified flow groups and discover more detailed functions.

Outputs of this functional separation process are a number of flow groups. We developed a traffic classification filter generation method to apply functional separation results to actual traffic classification. The filter generation step extracts common functional signatures from each flow group.

To validate the proposed methods, we have run tests by actually classifying traffic from various types of real applications using the fine-grained traffic classification filters. We also compared our algorithms with existing signature-based classification frameworks and a clustering algorithm to prove the higher accuracy and efficiency acquired by the proposed methods. Our research result can be used for improve the traffic classification accuracy and completeness. The fine-grained traffic classification might enrich the amount of information which can be obtained by traffic classification and can be a beginning of user-behavior analysis based on traffic classification.

1.4 Thesis Organization

The organization of this thesis is as follows. Chapter II reviews the existing literature on the topic of traffic classification. In particular, we present existing traffic classification works from two different perspectives. In other words, we categorize existing traffic classification work according to classification methods and classification levels. We also clarify the scope and objectives of our work. In Chapter III, we explain the proposed fine-grained traffic classification scheme and functional separation method in detail. Chapter IV describes the traffic classification filter generation technique. We describe the validation of the proposed method in Chapter V. The validation includes results of the accuracy evaluation and comparison with unsupervised machine learning algorithm. Finally, Chapter VI concludes the thesis with a brief summary and suggestion for future work as a further extension of this research.

Chapter II

RELATED WORK

In this chapter, we provide an overview of application traffic identification techniques and classify them into the following categories: 1) Port-based, 2) Payload-based, 3) Host-behavior-based, and 4) Statistical traffic classification approaches. We also make a note on the trend of traffic classification techniques. We then describe different types of traffic classification research according to the level of classification required and analysis capability. Finally, we explain the scope of this thesis by explaining the architectures of traffic classification systems.

2.1 Application Traffic Classification Approaches

As the Internet and its application evolve, application traffic classification have utilized various methodologies gradually to deal with obstacles related to traffic classification accuracy, processing speed and many other aspects of traffic classification. In this section we explain four different traffic classification approaches and explain

the trends of traffic classification techniques.

2.1.1 Port-based Approach

Most traditional traffic classification methods rely on the inspection of transport layer port numbers. Many traditional applications utilize a well-known port number to initiate and maintain their network connection. Therefore, the application can be inferred by looking up the packet's target port number registered in the Internet Assigned Number Authority (IANA) port list [2]. For example, Web traffic is associated with TCP port 80 and DNS is associated with TCP port 53.

The main advantage of port-based approach is the simplicity of the port matching technique. Due to the simple port-mapping strategy, port-based classification shows the best performance in terms of classification speed and extensibility. Adding new application port data into a classifier's database enables the traffic classification system to identify new applications.

However, this approach has limitations in terms of classification accuracy. Many applications use port numbers that are not registered in IANA port lists and many P2P applications allocate multiple port numbers dynamically. As a consequence, it is hard to match a certain port number with an application. Further, some applications allocate well-known port numbers (e.g., TCP port 80) intentionally to hide their traffic from detection or filtering. In this case, the traffic classified as Web traffic is not actual Web traffic. Moore *et al.* [1] asserted that the accuracy of port-based identification is no more than 50∼70%. Madhukar *et al.* [3] also showed that port-based analysis is ineffective for classifying P2P applications: 30∼70% of the Internet traffic is classified as "unknown."

Yet, the port-based approach is still a popular solution at the Internet backbone because of the high volume of traffic and limited computing resources for traffic classification. Moreover, the port-based approach is efficient for identifying general trends of application usage due to its simplicity. Borgnat *et al.* [4] have conducted a longitudinal traffic characterization of trans-Pacific backbone links. Their empirical findings showed that a significant portion of P2P traffic is replaced by multimedia and Web traffic. Although they may have aggregated the P2P traffic with other unknown protocols, they also utilized well-known port numbers to analyze the traffic data.

2.1.2 Payload-based Approach

This approach has been proposed to remove any uncertainty of the port-based classification approach [5, 6]. Theoretically, a complete protocol parsing is the most accurate solution for traffic classification. However, many applications utilize proprietary protocols to avoid public disclosure. In addition, some applications adopt well-known application protocol as parts of their application layer protocol. Another problem is the complexity of protocol parsing, which requires great amounts of computing resources and is not suitable for real-time analysis of backbone traffic.

As an alternative, the research community has responded by investigating a classification scheme capable of identifying applications with predefined byte patterns called signatures. A signature is a portion of the payload data that is static and distinguishable for applications, which can be described as a sequence of strings or hex values. For example, Web traffic contains the string "HTTP" or "GET" and BitTorrent traffic contains the string "0x13BitTorrent protocol" within its payloads.

The traffic classifier can identify certain traffic data by examining whether target traffic data contain the signature or not.

Once a set of unique payload signatures is available for an application, this approach produces extremely accurate classification. Signatures have been manually extracted by network administrators or security experts. The signature extraction process requires preceding protocol semantic analysis or empirical packet payload inspection for pattern recognition. Sen *et al.* [6] have generated the signatures of a few P2P applications by analyzing the application-layer protocols; their signatures could reduce FP and FN to 5% of the total bytes. The protocol semantic analysis could improve the accuracy of signatures, but signature generation with protocol analysis has the same problem as complete protocol parsing. This approach is only feasible when the application protocol description is publicly available. Human decision-making in such a process results in a slow response time to deal with new applications. The quality of signatures also varies due to the level of expertise. It is also an issue to find feasible signatures that deliver acceptable accuracy against the traffic in an asymmetric routing environment (e.g., an ISP's backbone links). Thus, researchers have proposed automated ways to extract application signatures to ease the manual signature generation process [7, 8, 9].

Although the payload-based approach removes dependency on fixed port numbers and increases the traffic classification accuracy, it still has some drawbacks. Inspecting payload data is a computing-resource-intensive process. It requires huge amounts of computing resources in the traffic classification system due to the complexity and processing load. Furthermore, extracting signature from the encrypted or tunneled traffic is difficult or impossible. In some limited instances, the signature

can be extracted from the encrypted traffic. Ehlert *et al.* [10] detected the hexadecimal patterns in Skype (v.1.5, v.2.0) packet traces during the initial communication setup phase. However, it is getting hard to inspect payload data because an increasing number of emerging network applications are eluding detection by packet payload encryption or by the use of plain-text ciphers. Finally, privacy legislation related to payload inspection is also a problem of the payload-based approach.

2.1.3 Host-behavior-based Approach

The host-behavior-based approach was proposed to classify traffic based on social interaction observable even with encrypted payloads [11, 12, 13]. For example, Karagiannis *et al.* [11] developed a general method (BLINC) for identifying applications. This method uses various heuristics and interconnection patterns exhibited by groups of nodes to identify services. BLINC classifies hosts by capturing the fundamental patterns of their behavior at the transport layer. The captured profile of a host, in terms of the destinations and ports with which it communicates, is represented by special graphs call graphlets. Then, the BLINC identifies applications or services by comparing the captured profile with predefined host behavior signatures (also described by graphlets). The key advantage of BLINC is flexibility; no additional information of application, such as port number, is required. Using a given set of pattern models, the authors claim that BLINC can identify application traffic with more than 90% accuracy. However, the level of classification is limited to identifying application types not exact application names.

FRM [14] handles each distinct flow-a collection record of packet header information-rather than packets themselves. This is based on an enhanced port and session

behavior mapping strategy that was developed by POSTECH in 2004. The idea of grouping flows using session patterns is quite similar to BLINC; however, port information is used to decide the actual name of an application.

Recently, Iliofotou *et al.* [12, 15] also proposed a graph-based representation of network traffic that captures the network-wide interactions of applications. Traffic dispersion graphs (TDGs) represent an individual IP address as a node and particular communications as edges between nodes.

The idea behind BLINC and TDGs is to investigate the communication pattern generated by a host and extract behavioral patterns that may represent distinct activities or applications.

Similar approaches have been adopted in network research area. Kim *et al.* [13] proposed a technique for traffic anomaly detection by analyzing correlations of destination IP addresses in outgoing traffic at an egress router. By observing the traffic and correlating it to the previous normal states of traffic, it may be possible to evaluate whether the current traffic is behaving in an anomalous manner.

2.1.4 Statistical (Machine Learning-based) Approach

Since Machine Learning (ML) was first utilized for Internet flow classification in the context of intrusion detection in 1994 [16], recent studies have applied ML algorithms to classify network traffic. Various types of learning algorithms such as supervised, unsupervised, reinforcement, and hybrid learning algorithms are used to build classifying models [17, 18, 19, 20, 21, 22, 23].

Generally speaking, ML takes input in the form of a dataset of instances. An instance refers to an individual, independent example of the dataset. Each instance

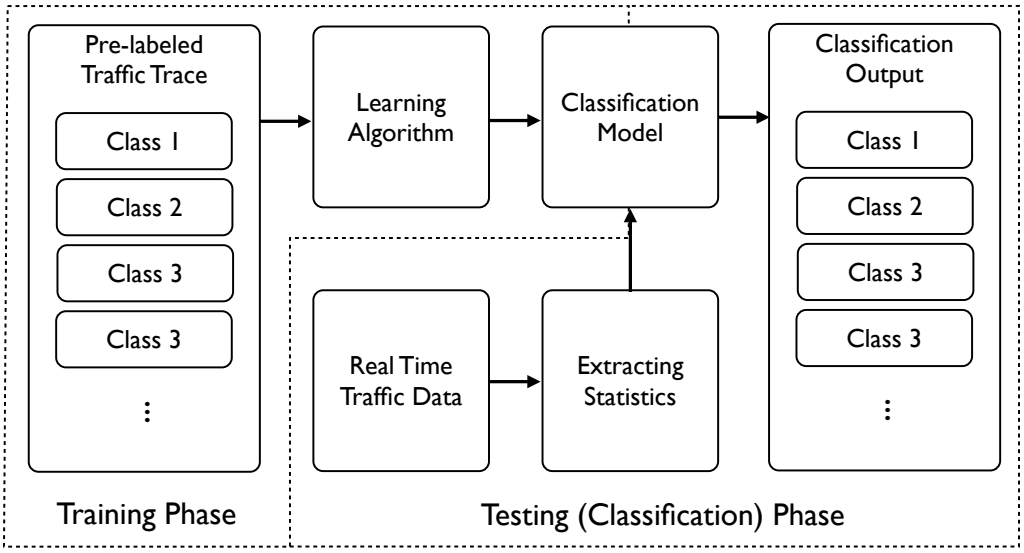


Figure II.1: Traffic classification process using supervised ML

is characterized by the values of its features that measure different characteristics of the instance. The dataset is ultimately presented as a matrix of instances versus features. In traffic classification area, most ML-based classification methods exploit connection-related statistical information-including connection duration, inter-packet arrival time, and packet size-as a feature set regardless of its type; thus, these methods can also be used with encrypted traffic. The idea underlying this approach is that traffic at the network layer has statistical properties that are unique for certain applications and enable each different application's traffic to be distinguished from another.

ML algorithms, utilized for traffic classification, can be categorized into supervised learning (or classification) and unsupervised learning (or clustering). Supervised learning creates knowledge structures (e.g., traffic classifier) that support the

task of classifying new instances (e.g., new traffic data) into pre-defined classes (e.g., applications or services). For supervised algorithms, the class of each traffic flow must be known before learning. Figure II.1 illustrates the overall process of traffic classification using supervised learning. A classification model is built using a training set of example instances that represent each class (training phase). The model is then able to predict class membership for new instances by examining the feature values of the unknown traffic data set (testing phase). There exist a number of supervised learning algorithms and each algorithm differs in the way the classification model is constructed and what optimization algorithm is used for a good modeling.

In contrast, unsupervised learning does not require a training data set. Unsupervised algorithms group traffic flows into different clusters according to similarities in the feature values. These clusters are not pre-defined and the algorithm itself determines their number and statistical nature. While supervised learning focuses on the relationship between input and output data, clustering focuses on finding patterns in the input dataset.

Erman *et al.* [20] compared the performance of three different unsupervised algorithms (KMeans, DBSCAN, and Autoclass) to classify application traffic data, and other researchers used various supervised learning algorithms [24]. Each researcher observed that the algorithms show good performance for traffic classification in terms of classification accuracy (maximum: 92∼93%). Nguyen et al. [25] surveyed and reviewed the state-of-the-art approaches to ML-based IP traffic classification. There has been much research on feature selection [26, 27], and finding an effective feature selection for traffic classification helps enhance the accuracy of the classifiers. Some

approaches have focused on using features that are based on aggregate flow statistics such as average packet size [18, 19, 22, 23, 28, 29]. Other approaches have used per-packet statistics such as individual packet sizes and inter-arrival times [22, 29].

2.1.5 Trend of Classification Approaches

In this thesis, we have analyzed the papers published between 1994 and 2009, starting with traffic classification and measurement papers, papers written by the same set of authors, and references cited therein. Figure II.2 shows the trend in classification approaches over time based on the number of publications. These statistics were calculated based on papers published between 2001 and 2009 (the values derived from 1994 to 2000 were omitted due to their small contributions). The proportion of classification approaches shows the trend in classification approaches better than the absolute number for each approach. As mentioned before, the port-based approach was dominant early on (beginning of the 2000s) and its proportion decreased as time passed due to its various limitations. However, it is remarkable that this port-based approach has been utilized continuously because of the high volume of traffic and limited computing resources for traffic classification. We believe that the port-based approach is still efficient for identifying general trends in application usage owing to its simplicity.

Since Napster in 1996, many more P2P applications have been and continue to be introduced. The growth in user popularity and traffic volume of P2P applications has been explosive. Due to the little-known port allocation scheme of many P2P applications, the research community started to focus on packet payload content rather than port number to identify network traffic. DPI examines the pay-

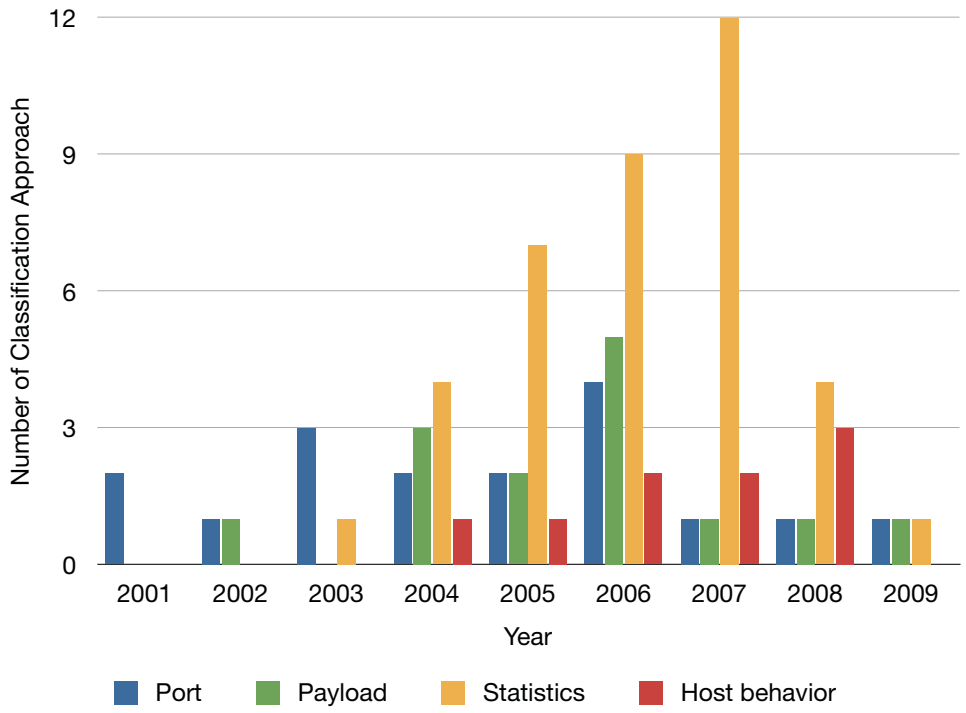


Figure II.2: Trend of classification approaches based on number of publications

load data of a packet to identify network traffic. This approach is also popular in the IDS and intrusion prevention system (IPS). Its high accuracy enables advanced network management, including traffic classification, security function, and service management. Even though the absolute number and proportion of payload-based approaches have been decreasing over time because of the privacy legislation related to payload inspection, the majority of previous work demonstrated that the signature-based approach was the most reliable one available for accuracy. Even some statistical approaches (machine-learning approaches) used signatures (or manual payload inspection) as ground truths for validation [5, 24, 29, 30, 31, 32].

Due to the privacy legislation issues, the research community tried to develop new

Table II.1: Comparison of different classification approaches

	Accuracy	Computational Cost		Extensibility	Handling Encryption
		Classifier Generation	Matching		
Port-based	Low	Low	Low	High	No
Payload-based	High	High	High	High	No
Host-behavior-based	High	Medium	High	High	Yes
Statistics (ML-based)	High	High	Medium	Low	Yes

classification approaches that do not require any payload data. Since the machine-learning technique was adopted in the traffic classification area, many papers using machine learning have been published; this statistical approach constitutes the largest portion of classification approaches.

Table II.1 summarizes comparison between four different traffic classification approaches in the abstract. Even though this summarization cannot characterize individual traffic classification research, it shows general characteristics of each classification approach.

2.2 Level of Application Traffic Classification

The previous studies have discussed various classification methodologies (e.g., well-known port matching, payload content analysis, and ML). Steadily over time, many variants of such methodologies have been introduced to improve the classification accuracy and efficiency. However, it is extremely difficult for any method to achieve 100% accuracy due to the fast-changing and dynamic nature of Internet traffic. In another respect, each study aims at different levels of classification. Some only have a coarse classification goal, such as classifying traffic protocols or application

types, while others have more detailed classification goals, such as identifying the exact application name. Therefore, it is often unfair to compare each classification method in terms of accuracy. This section describes different types of traffic classification research according to the level of classification requirements and analysis capability rather than techniques used in traffic classification. We categorize traffic classification studies into following categories: Application protocol breakdown scheme, Traffic clustering scheme, Application-type breakdown scheme, Application breakdown scheme, and others.

2.2.1 Application Protocol Breakdown Scheme

Traffic classification is a process of identifying network traffic based on the features that can be passively observed in the traffic. The features and classification results may vary according to specific classification requirements and analysis needs. Early on, traffic classification was performed as a part of traffic characterization work, and was often motivated by the dominance of certain protocols in the network. Several studies [33, 34] analyzed the packet and byte distributions regarding transport and application layer protocols. TCP/UDP port numbers map to a well-known TCP/UDP protocol. The protocol breakdown scheme shows a rough estimation of the traffic composition and is still a popular solution at the Internet backbone because of its high traffic volumes and limited computing resources for traffic analysis [8, 20, 22, 34, 35, 36, 37].

Borgnat *et al.* [4] have conducted a longitudinal traffic characterization of trans-Pacific backbone links. Their empirical findings showed that a significant portion of P2P traffic is replaced by multimedia and Web traffic. Although they may have

aggregated the P2P traffic with other unknown protocols, they also utilized well-known port numbers for protocol breakdown.

2.2.2 Traffic Clustering Scheme

A straightforward classification, relying on IP protocol and port information, cannot provide in-depth classification of traffic within a single protocol where similar traffic types use different protocols. This scheme reflects traffic workload characteristics rather than the protocol composition [17, 38, 19, 39]. McGregor *et al.* [17] proposed a machine-learning-based classification methodology that can break the traffic down into clusters: bulk transfers, small transactions, and multiple transactions. This allows us to understand the major types of traffic in a network.

2.2.3 Application-type Breakdown Scheme

BLINC [11] is a connection-pattern-based classification method. The idea behind BLINC is to investigate the communication pattern generated by host and extract behavioral patterns that may represent distinct activities or applications. It categorizes network traffic according to application type rather than a specific application name, such as Web, game, chat, P2P, DNS, FTP, streaming, mail, and attack activities. This scheme resides between the former two schemes [1, 18, 40, 41, 42, 43, 44, 45, 46, 47]. The application-type breakdown scheme helps network administrators or operators to understand dominant application types rather than specific application names or protocols.

2.2.4 Application Breakdown Scheme

The dominance of P2P networking in the Internet has had a huge influence on traffic classification research and led to more sophisticated heuristics. In this context, many researchers have focused on identifying the exact application represented by the traffic [5, 6, 7, 48, 49, 50, 51, 52, 53, 54, 55]. Discovering byte signatures [6] has been a popular solution. Regardless of its proven accuracy, the signature-based solution possesses high processing overhead and privacy-breaching issues because it requires a packet header and payload analysis. Recently, machine learning techniques which use statistical information of the transport layer [24] is introduced to overcome privacy legislation related to packet payload inspection. It focuses on the fact that different applications have different communication patterns (behaviors). Moreover, Szabó *et al.* [56] introduced combinations of these existing methods in order to balance between the level of classification completeness and accuracy. All these efforts are presented to classify network traffic according to the name of application in use.

The dominance of P2P networking in the Internet has had a huge influence on traffic classification research and led to more sophisticated heuristics. In this context, many researchers have focused on identifying the exact application represented by the traffic [5, 6, 7, 48, 49, 50, 51, 52, 53, 54, 55]. Discovering byte signatures [6] has been a popular solution. Regardless of its proven accuracy, the signature-based solution possesses high processing overhead and privacy-breaching issues because it requires a packet header and payload analysis. Recently, machine-learning techniques that use statistical information of the transport layer [35] were introduced to

overcome privacy legislation related to packet payload inspection. The techniques focus on the fact that different applications have different communication patterns (behaviors). Moreover, Szabó *et al.* [56] introduced combinations of these existing methods in order to balance the level of classification completeness and accuracy. All these efforts are presented to classify network traffic according to the name of the application in use.

2.2.5 Other Classification Levels

Some other research use multiple classification levels simultaneously to classify traffic. Most common combination is protocol breakdown and application breakdown [3, 29, 31, 32, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71]. Other combinations are [protocol breakdown and application-type breakdown][72], [traffic clustering and application protocol breakdown][19, 39], and [application-type breakdown and application breakdown][73]. Some other research focus on classification of network anomalies [9, 74, 75, 76, 77, 78, 79, 80, 81]. More details on how each research classifies network traffic will be provided as appendix later.

Other studies use multiple classification levels simultaneously to classify traffic. The most common combination is protocol breakdown and application breakdown [3, 29, 32, 57, 58, 59, 60, 61, 62, 63, 31, 64, 65, 66, 67, 68, 69, 70, 71]. Other combinations are as follows: protocol breakdown and application-type breakdown [72], traffic clustering and application protocol breakdown [19, 39], and application-type breakdown and application breakdown [73]. Some other studies have focused on the classification of network anomalies [74, 75, 76, 77, 78, 9, 79, 80, 81]. More details on how each research classifies network traffic will be provided in an appendix

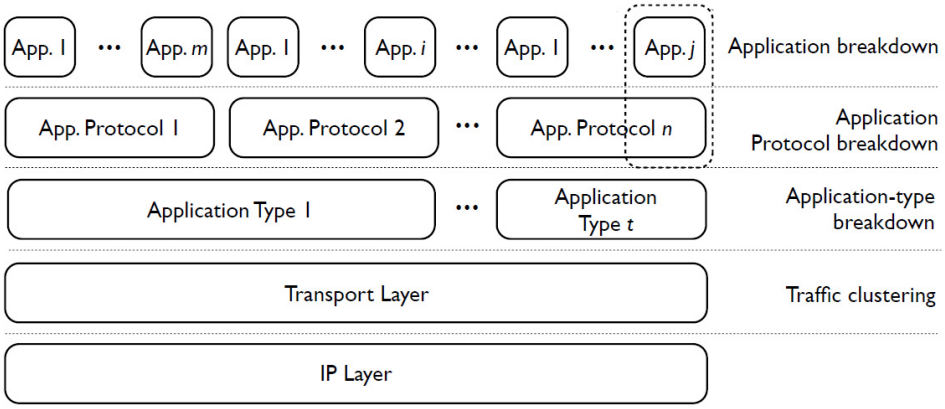


Figure II.3: Traffic classification schemes according to different classification levels

to be included later.

2.2.6 Comparison

Figure II.3 shows different traffic classification schemes according to their classification level. The application protocol breakdown scheme presents network traffic into different protocols rather than application types or names. For example, all ftp traffic is classified under the ftp protocol group although there are many distinct ftp client programs since all clients employ the same ftp protocol for data transfer. The traffic clustering scheme was proposed in different perspective to traffic classification. While the application protocol breakdown focuses on identifying certain protocol, the clustering scheme can capture common characteristics shared among the distinct applications using a single or multiple protocols. In addition, the application breakdown scheme can provide more detailed classification results, especially for P2P applications. It would classify distinct application names even if the corresponding traffic is generated from the same protocol. For example, there are many

descendant applications which use the BitTorrent protocol. While the application protocol breakdown scheme cannot distinguish the traffic generated by different BitTorrent clients, the application breakdown scheme can classify the traffic according to the exact client name represented by the traffic.

The application-type breakdown scheme resides between the traffic clustering and application protocol breakdown schemes in terms of classification level. It characterizes the traffic based on connection pattern or host profiles and classifies into various application-types, such as Web, game, chat, P2P, streaming, mail, and security attack activities. One application-type can be a superset of both application and application protocol.

2.3 Scope and Objectives

2.3.1 RTFM/IPFIX Architecture

Many traffic monitoring and analysis systems follows the Real-time Traffic Measurement (RTFM) architecture [82] which was proposed by the IETF working group and it is a logical monitoring structure for different types of networks. The RTFM architecture is composed of four entities that communicate by means of a suitable protocol which is not explicitly specified in the definition: Meter, Meter Reader, Manager, and Analysis Application.

- **Manager**: A traffic measurement manager is an application which configures 'meter' entities and controls 'meter reader' entities. It uses the data requirements of analysis applications to determine the appropriate configurations for each meter, and the proper operation of each meter reader. It may well be

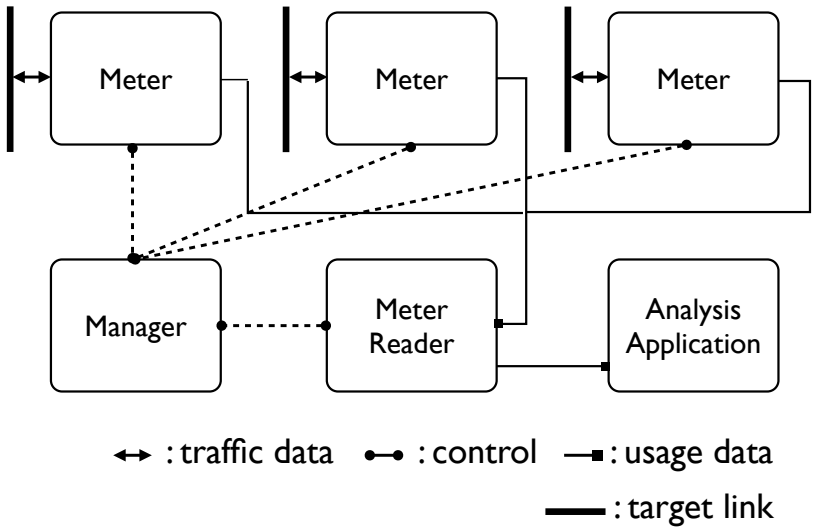


Figure II.4: RTFM architecture: relationships between RTFM entities

convenient to combine the functions of meter reader and manager within a single network entity.

- **Meter**: Meters are placed at measurement points determined by network operator. Each meter selectively records network activity as directed by its configuration settings. It can also aggregate, transform and further process the recorded activity before the data is stored. The processed and stored results are called the 'usage data'.

- **Meter Reader**: A meter reader reliably transports usage data from meters so that it is available to analysis applications.

- **Analysis Application**: An analysis application processes the usage data so as to provide information and reports which are useful for network engineering and management purposes.

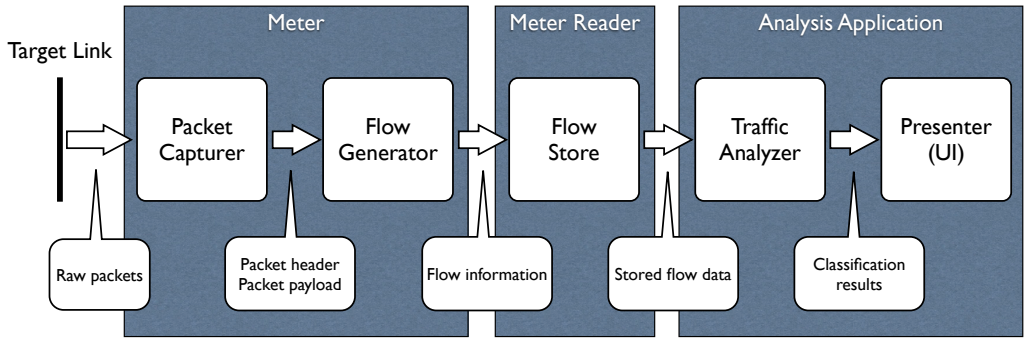


Figure II.5: A typical Internet application traffic classification system

Figure II.4 illustrates relationships between entities of RTFM architecture. There is an open-source implementation of the RTFM architecture for network traffic flow measurement called NeTraMet [83, 84]. To improve the compatibility with high-speed network links, many traffic monitoring systems have proposed improved architecture for monitoring network traffic such as IPFIX [85].

2.3.2 Architecture of Traffic Classification System

Many traffic monitoring systems follow the RTFM architecture. Figure II.5 illustrates the design of a typical Internet application traffic classification system which is influenced by RTFM architecture.

Although detailed design of each traffic classification system differs according to their specific classification goal and implementation issues, many traffic classification systems such as nTop [86, 87] and CoralReef [88] have common components in terms of functionalities as follows.

- **Packet Capturer**: A packet capturer collects the entire raw packets passing

through the network link by using the port mirroring function supported by switches and routers or network splitters.

- **Flow Generator**: A flow generator aggregates raw packet trace into flows. A flow is a sequence of packets which shares same 5-tuple header values; source IP address, destination IP address, protocol, source port, and destination port. Even though there are various definitions about the network flows such as Cisco's NetFlow [89] and Juniper's Jflow, their common goal is aggregating raw packet trace to deal with high-speed network traffic.

- **Flow Store**: Generated flows are stored in a flow store so that it is available to traffic analyzer.

- **Traffic Analyzer**: A traffic analyzer queries the flow data stored in a flow store according to the various analysis scopes. For traffic classification, the traffic analyzer matches stored flows with traffic classification rules for filters. Each classification system has their own knowledge based to classify network traffic into different classes.

- **Presenter**: A presenter shows traffic analysis result with graphical user interface.

Each component has lots of room for improvements in terms of performance. Table II.2 describes what kinds of improvements are possible with each component.

For the packet capturing, there are various efforts to handle high-speed network links. University of Waikato [90] developed DAG technology which is a combination of hardware design (using FPGA technology) and software (a software driver layer)

Table II.2: Possible improvements of traffic classification system

	Methodology	Effects on Performance
Packet Capturer	High performance packet capturing devices Reducing packet dropping ratio	Handling high-speed network link
Flow Generator	Flow sampling Support different flow formats	Handling high-speed network link Improvement of flexibility
Flow Store	Optimize database operations	Improvement of analysis speed Efficient data reduction
Traffic Analyzer	New classification methodology Matching algorithm	Improvement of accuracy and completeness Improvement of classification speed
Presenter	Data visualization	Efficient reports Effective in data presentation

and Endace Measurement Systems have developed commercial products of DAG technology named DAG cards which are the ultimate in network packet capture interface cards. They guarantee 100% packet capture on any network regardless of packet size, interface type or network load [91].

For traffic analyzer and presenter, many of previous studies related to traffic classification proposed various techniques to classify network traffic accurately and efficiently. Traffic analyzers utilize their own methodologies to detect target application traffic. Previous studies described in Section 2.1 and Section 2.2 have proposed new classification methodologies or new types of traffic classifiers.

In this thesis, we will propose a new traffic classification methodology from the viewpoint of traffic analyzer. More specifically, our methodology aims to provide an accurate way to build knowledge structure for traffic classification. The performance improvements for packet capturing and flow generation are beyond the scope of this thesis.

Chapter III

FINE-GRAINED TRAFFIC CLASSIFICATION AND FUNCTIONAL SEPARATION

In this chapter, we propose a solution for the aforementioned problem of accuracy and completeness in traffic classification. More specifically, we propose a scheme called fine-grained traffic classification that can provide more in-depth information about traffic classification results. The key to this fine-grained traffic classification scheme is that the methodology can classify various traffic generated by a single application.

We first will describe the characteristics of current Internet applications and traffic classification methodologies. Then, we will describe the details of our fine-grained traffic classification scheme and its main algorithm, a functional separation methodology. The proposed methodology can detect different types of traffic gen-

erated by a single application according to their functionalities, and we expect that it will increase accuracy and completeness of traffic classification by reducing the amount of undetected traffic.

3.1 Fine-Grained Traffic Classification

This section explicates the concept and algorithm of the fine-grained traffic classification and its significance by analyzing the characteristics of current Internet applications and traffic classification schemes.

3.1.1 Characteristics of Current Internet Traffic and Applications

Compared to traditional network applications,Internet applications nowadays are becoming more complex as they generate various types of traffic. Traditional applications, such as web, e-mail, and ftp, tend to communicate with a single or just a few hosts for their service completion while using fixed ports most of the time. On the other hand, however, newly emerged Internet applications use multiple sessions with a dynamic and ephemeral port allocation strategy, which makes traffic classification more difficult.

Figure III.1 illustrates the number of open connections over time when a host downloads a file using a client application named BitTorrent. Although the exact number of connections varies according to the condition of BitTorrent swarms, in general, a large number of connections are established simultaneously. It stands true for not only BitTorrent but also other P2P applications such as eDongkey, as most of them keep multiple sessions to download contents from other peers.

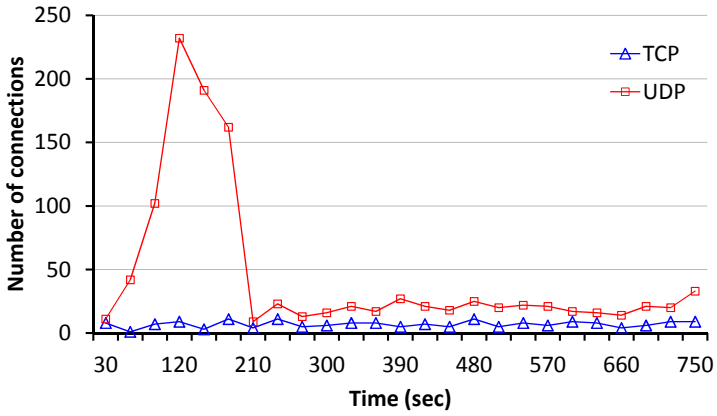
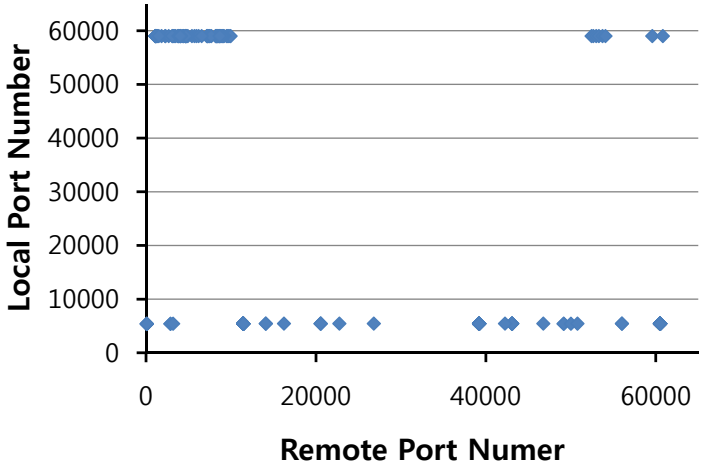


Figure III.1: Number of concurrent network connections over time

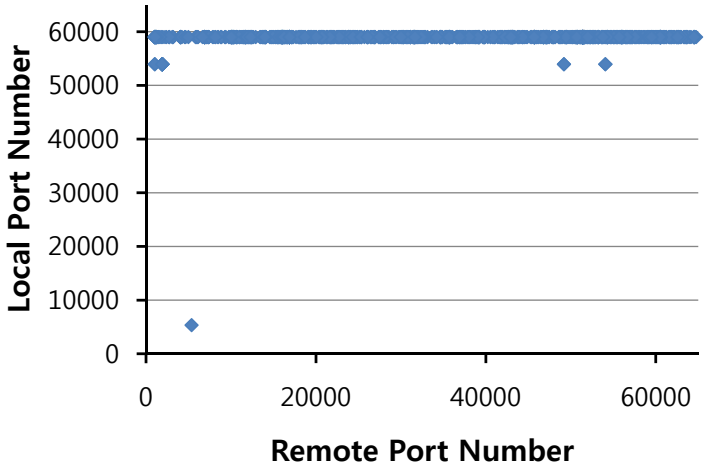
Figure III.2 shows the port allocation behavior of BitTorrent. X-axis indicates the number of remote ports and y-axis the number of local ports. An individual session is represented as a dot. As the figure shows, local port numbers are concentrated in a small range whereas remote port numbers are broadly distributed. Considering the fact that a P2P client acts both as a client and server, we can conclude that a BitTorrent client allocates dynamic port numbers and that it makes application traffic identification tricky as a result.

Another characteristic of recent Internet applications is their multi-functionalities. Newly emerged applications are usually equipped with various functionalities in addition to their main ones while traditional applications have only one or a small number of functionalities.

A case in point is Microsoft's MSN messenger application. Old versions of MSN (prior to version 3) had only supported simple functions in a limited range such as plain text messaging, e-mail notification (Hotmail), and contact lists update. On the



(a) TCP port allocation



(b) UDP port allocation

Figure III.2: BitTorrent's port allocation behavior

other hand, the latest version of MSN now provides numerous advanced functions including video/voice call, file transfer, mobile games, and remote assistance, just to name a few. The ever-increasing complexity and multiplicity of applications preclude

detecting a complete set of traffic generated even from a single application.

A more serious problem is that these types of newly emerged applications and the traffic they generate Largely dominate today's internet. Results from Internet traffic characterization research support this phenomenon. A piece of research work on analyzing P2P traffic shows that P2P traffic now accounts for 50∼70% of the total Internet traffic [92]. Kim *et al.* [69] also corroborate that P2P applications generate a substantial volume in enterprise networks. Some long term studies on traffic characterization make it more clear. Claffy *et al.* [93] analyzes a long term growth of NSFNET from 1988 to 1993. During those days, FTP and Mail accounted for about half of the growing traffic volume until web traffic became the majority. In 2009, a Japanese research group [4] conducted a study of Internet traffic characterization during the period of 2001∼2008. Their empirical observation suggests that a significant portion of P2P traffic is now being replaced by multimedia and Web traffic.

3.1.2 Significance of Fine-Grained Traffic Classification

The significance of fine-grained traffic classification can be found in the characteristics of current Internet applications and their traffic described in Section 3.1.1. In general, the current studies on traffic classification mainly grapple with detecting major applications including P2P and streaming applications that occupy most of today's Internet traffic. Furthermore, detecting P2P or streaming itself is also heavily weighted toward classifying a few main functions of the major applications (e.g., file transfer in P2P) that generate the greatest volume of traffic workload. Nevertheless, the applications simultaneously generate all different kinds of traffic

besides the prominent ones of high-volume traffic. As a result, the lack of interest in minor traffic classes and consequential neglect of them inevitably undermine the accuracy and completeness of classification, especially in face of newly emerging network applications.

As an example, we have conducted a very simple experiment. We have measured amount of traffic generated by a P2P client when downloading a small size of music files. The total traffic volume generated by the P2P client was approximately 45.9 Mbytes; yet, the actual volume of downloaded files was 24.1 Mbytes, which is less than 53% of the generated traffic in total. It implies that ancillary functionalities (except the actual downloading) of the P2P client generate a significant portion of the traffic.

Figure III.3 shows the relationship between the fine-grained traffic classification and other traffic classification schemes described in Section 2.2. The fine-grained traffic classification can classify various types of traffic derived from a single application. As shown in Figure III.3, a single application typically has several functions, and each function triggers a unique traffic characteristic. By identifying various types of functional traffic, we can reduce undetected or unclassified traffic in comparison to other traffic classification schemes. In addition to this, the fine-grained traffic classification enriches the quality of information achieved by classifying traffic.. In other words, this new traffic classification scheme can provide more in-depth traffic classification results. While top n protocol or application analysis is possible with other schemes, our fine-grained traffic classification allows new approaches to data analysis using innovative categories such as the average browsing time to initialize a file download and the most popular functions currently in use among users.

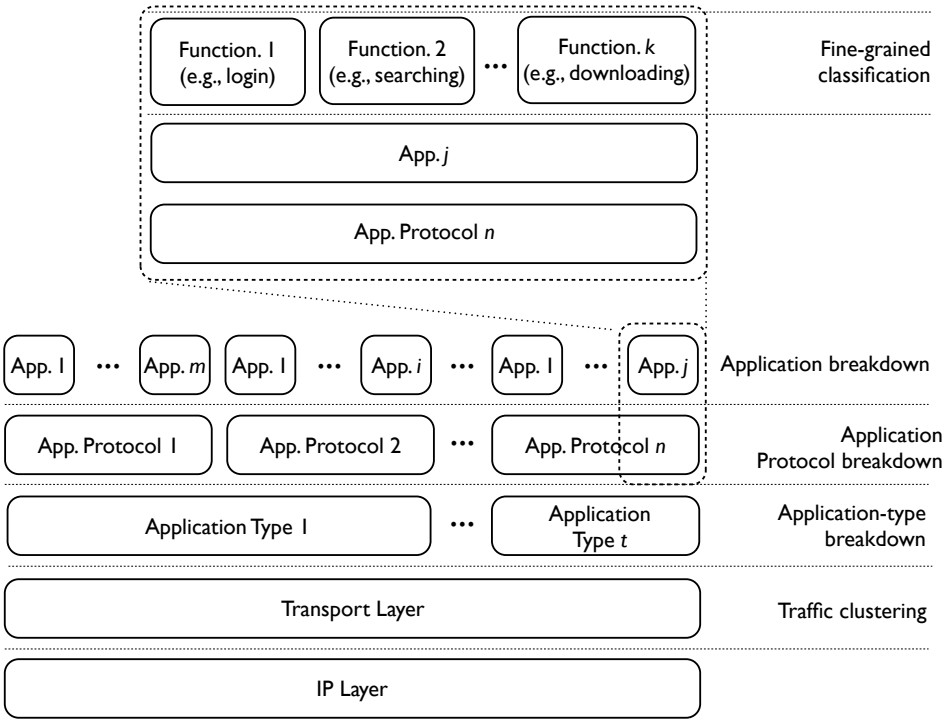


Figure III.3: Relationship between fine-grained traffic classification and other traffic classification schemes

Therefore, it can also serve as a tool to analyze user behavior and design future applications in the Internet. When it applies to Web traffic, analyzing the most popular function of the Web site (e.g., Facebook) is also possible. This will extend the traffic classification research from network administrative oriented research to user context dependent research.

3.1.3 Methodology for Fine-Grained Traffic Classification

A key to the fine-grained traffic classification is how to categorize single application traffic into different traffic groups. It does not vary significantly from the central

problem of traditional traffic classification except for the degree of detail in classification. Accordingly, various existing methods, most of which use a classifier per application or protocol can be applied to this classification scheme.

We simplify the fine-grained traffic classification problem as shown below. We first build arbitrary classifiers (e.g., application signature, connection behavior model, statistical model, and etc.) per application where each classifier corresponds to a distinct function in the application. Among many classifiers, we select signature as the classifier because a large number of previous works have demonstrated that a signature-based approach is by far the most reliable in terms of accuracy. Even some statistical approaches used signature as ground truth for validation [24]. In addition, it is convenient to apply new fine-grained signatures into existing traffic classification systems and commercial traffic shapers as intrusion detection devices utilize application signatures for their classification.

Our methodology for fine-grained traffic classification consists of three parts: 1) input data collection, 2) discrimination of various functionalities in a single applications traffic, 3) traffic classification filter extraction, and 4) traffic classification using fine-grained signatures.

Figure III.4 illustrates the workflow of fine-grained traffic classification process. This process is consists of includes both an offline and online process. The offline process is for to build a knowledge structure for online traffic classification system while the online process is the actual traffic classification process with a traffic classification system. As described in Section 2.3.2, an online traffic classification system has an traffic analyzer, and it has its own knowledge base for classifying or identifying application traffic. This thesis focuses on the offline process that is to

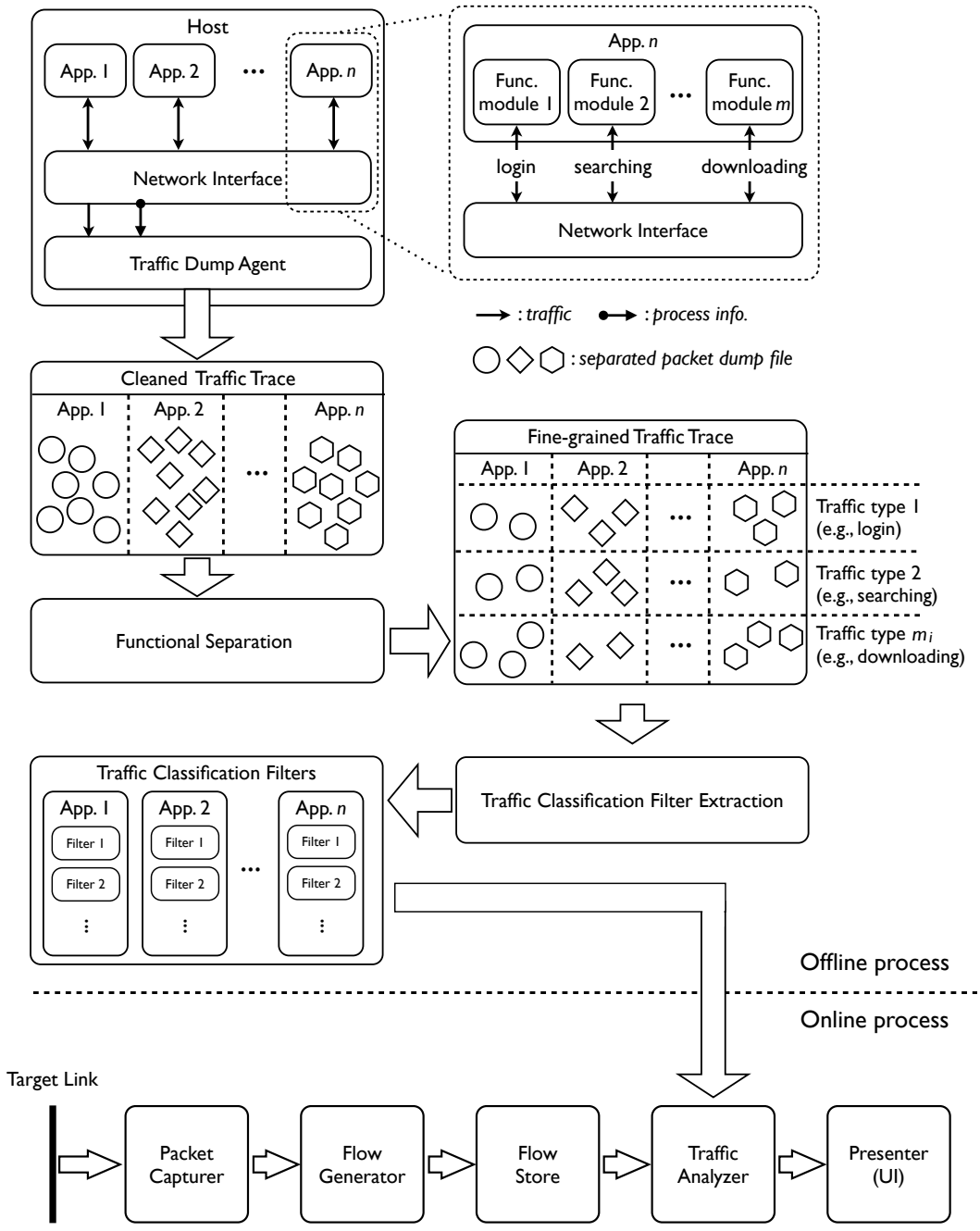


Figure III.4: Abstraction of fine-grained traffic classification process

build fine-grained traffic classification filters. Once fine-grained traffic classification filters are built, the filters can be applied to existing traffic classification systems with little modification in the system. small amount of system modifications.

The offline process starts with input data collection. The functional separation algorithm requires a collection of cleaned packet as its input data. Cleaned raw packets refer to the packets belonging to the target application only. We have developed a continuous packet dump agent using libpcap library [94] to collect the packet trace for every running process in the OS. The collecting agent divides the sanitized packets according to each flow and stores them in a separate packet dump files tagged with the origin process name. It is important to keep the datasets separate according to each flow and its process name because it can reduce unnecessary flow comparison overhead in functional separation step. If the given dataset is a mixture of many different applications, it may be difficult to discover common patterns in an application. Such a design decision is necessary to guarantee efficiency and accuracy of the functional separation; thus, we remove any uncertainty of traffic being fed to the functional separation algorithm.

When n different applications are running on a host, each application executes m_i different functional modules (the subscript i indicates that the value of m_i differs from one application to another) and each module in an application generates different types of traffic. The traffic dump agent continuously monitors the network interface and captures all traffic data passing through the network interface. The agent aggregates the traffic data into flows and stores each flow in a separate file. Every flow is tagged with the application or process name acquired from the OS. The stored n groups of cleaned traffic, labeled with an application name, are fed into the

functional separation. The functional separation classifies the cleaned traffic into m_i subcategories according to the flow type. As a result of this functional separation, we can get $n \times m_i$ groups of flow data. Each group of flow data is used as input data for traffic classification filter extraction. In this case, a number of filter extraction's output is $n \times m_i$ and a single application can have at the most m_i signatures. Most of the prior research on signature-based traffic classification designates a single signature per application. However, this may lead to an increase in the false negative ratio as well. We will discuss the fine-grained traffic classification process more in detail in the following sections.

3.2 Cleaned Data Collection

In this section, we describe our cleaned data collecting agent called Traffic Measurement Agent (TMA). We have developed different versions of data collecting agent programs which run on Windows based hosts and Andoroid based smart phones. We believe that these collecting agent applications not only for data collecting but also for other functions such as generating ground truth data used for the for validation of traffic classification systems.

3.2.1 Traffic Measurement Agent

If the target application traffic blends into other traffic, the possibility of finding reliable traffic groups in a single application will decrease. The role of the Traffic Measurement Agent (TMA) is to collect a packet trace without any undesired traffic and generated traffic summary log data. The agent program captures all the packets

passing through the network interface and filter out undesired packets based on socket allocation information obtained from its operating system.

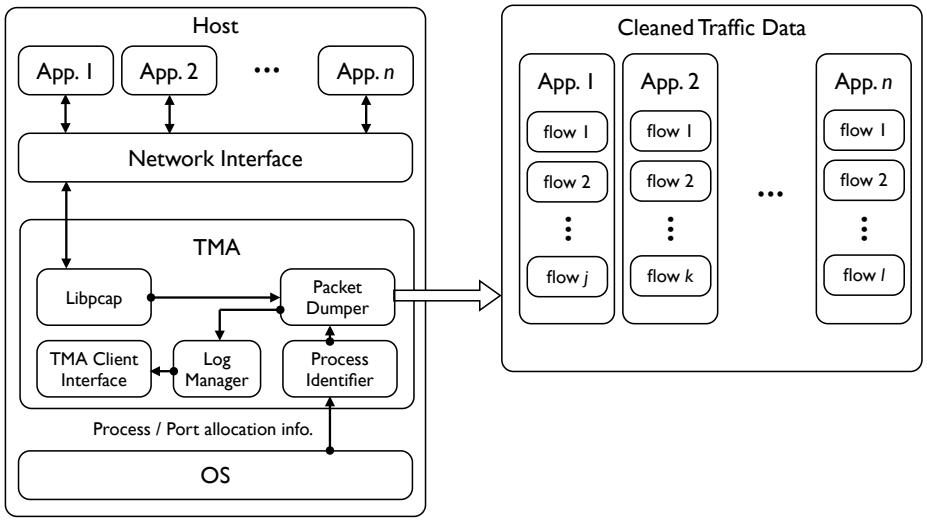


Figure III.5: Internal structure of TMA

Figure III.5 illustrates the internal structure of the TMA. The TMA runs on a Windows-based host and interacts with a network interface and operating system. The agent captures packets from the network interface using libpcap packet capture library [94]. Before dumping the packet data into a file, the agent extracts packet header information and passes it onto a process identifier.

The process identifier obtains TCP/UDP allocation status from the operating system. A Windows operating systems provides a family of APIs to retrieves a table that contains a list of TCP and UDP endpoints available to the application. The process identifier retrieves a process name responsible for a certain port number and returns this information to the packet dumper.

The packet dumper aggregates packet traces into flows and stores each flow in a

separate file. While storing the packets, the process name is used as a key. Therefore, each stored trace file only contains packets of the selected application.

The TMA also generates the traffic summary log of cleaned traffic data using a log manager. The log is composed of lines of traffic summary according to each corresponding process. The log data is sent to a server via TMA client interface. The log data collected in the TMA server can be later used as a ground truth for FP and FN verification.

3.2.2 Mobile Traffic Measurement Agent

The mobile traffic collecting agent called Mobile Traffic Measurement Agent (mTMA) runs on Android based smart phones. Due to the differences between Windows and Android, implementation details of mTMA and TMA are different. However, an overall structure of mTMA is similar to that of the TMA as described in the previous subsection. Smart phones have limited computing resources compared to usual desktop or laptop computers. Storing every packet can be a heavy burden to smart phones. To mitigate the overhead, therefore, we removed the packet dumper from the mTMA to an extra device specifically designed for packet dumping.

We developed a separated dump agent that is capable of acting as a wireless access point. This dump agent operates as a network bridge. When a smart phone is connected to the access point, the dump agent captures all packet passing through the agent and stores them into a temporal trace file. Then a trace filterer filters out undesirable traffic based on traffic summary log sent from the mTMA in the smart phone. The output file is exactly same with the cleaned traffic trace stored by the TMA.

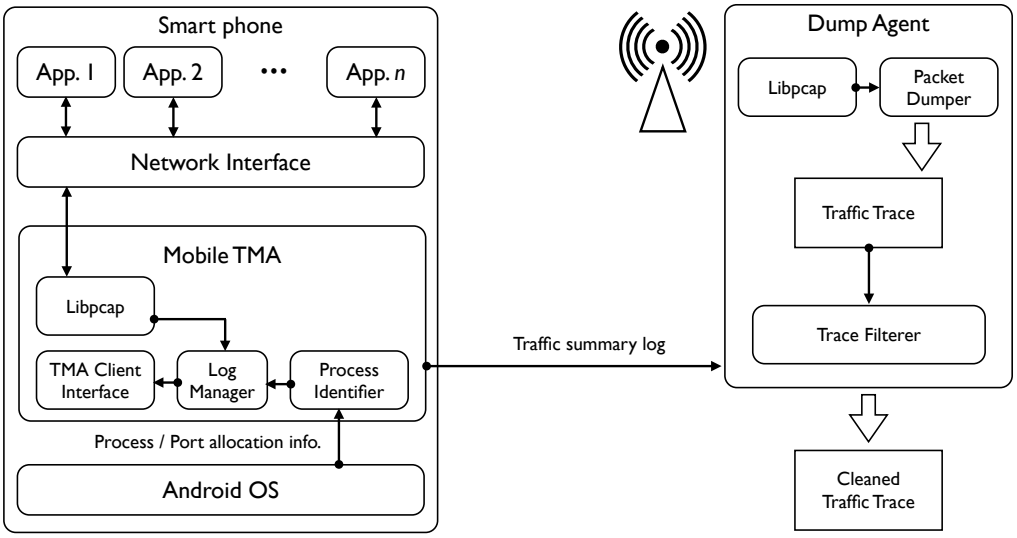


Figure III.6: Internal structure of mTMA and dump agent

3.2.3 Ground Truth Data

The TMA server-client architecture is one way to obtain a set of ground truth data for validation of the traffic classification. Both TMA and mTMA have a TMA client interface inside of them. The TMA client interface is the communication channel between TMA (or mTMA) and TMA server. A TMA transfers traffic summary logs generated by the log manager to a TMA server periodically to minimize the effect on network traffic by TMA programs.

Figure III.7 shows the usage of TMA server and client. The TMA log data are generated from a campus network's edge client and transmitted to the TMA log sever. Then the server keeps the accumulated log data, which can be used to determine the overall accuracy of the traffic classified by a traffic classification system. This TMA approach can also be applied to various other classification

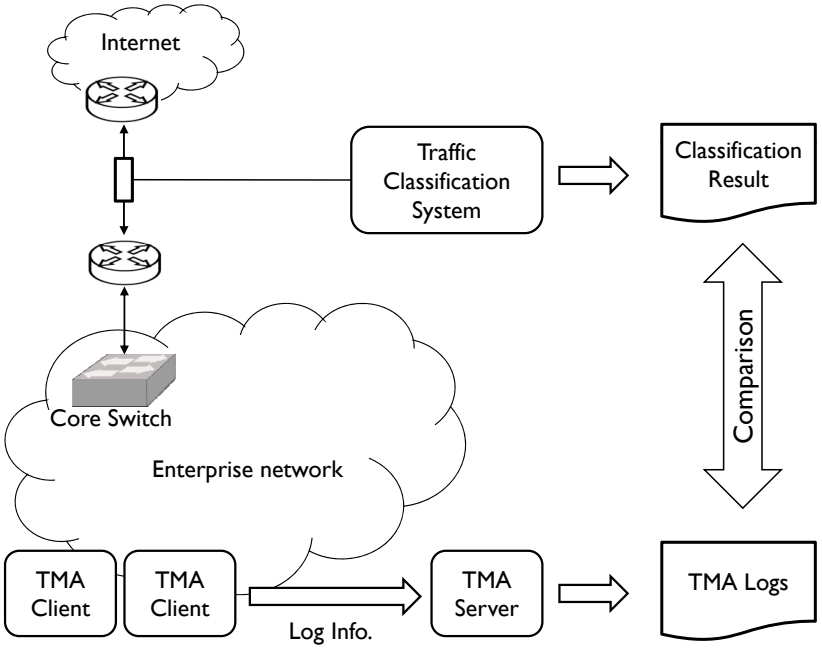


Figure III.7: Validation using TMA clients and TMA server

methods attempting to measure accuracy.

3.3 Functional Separation

In this section, we present our functional separation algorithm, which is the core of the entire fine-grained traffic classification. Functional separation (FS) aims to classify individual flows according to their functionalities in an application. The main idea of the FS reflects the fact that there exist common characteristics among flows that belong to the same functionality. We adopt the TCP/UDP port number and payload contents as our grouping criteria. For example, a BitTorrent client allocates a number of TCP/UDP port dynamically and simultaneously. Although

assigned port numbers are not predictable, the sessions that share the same port number in a certain time slot can be grouped into a same functional group. We also examine actual packet contents so that we can assort flows based on the similarity of application protocol layers even if same functional sessions allocate different port numbers.

Figure III.8 illustrates the entire FS process. It consists of three different grouping/decomposition steps. The solid rectangles describe how the traffic data is organized after each grouping/decomposition steps, and the dotted rectangles indicate the information used in each step. The role of each grouping/decomposition step is as follows.

1. Port-Relation Grouping (PRG) means assorting flows based on the dependency of assigned port numbers. In this step, the flow information is generated from the captured clean traffic trace according to their 5-tuple information: source IP address, destination IP address, source port, destination port, and protocol. During the PRG step, we treat the port numbers as indexes without any function-related information. For example, we groups flows which use TCP port 21 into same PRG group, but we do not assume in any way that the use of TCP port 21 signifies ftp control traffic.

2. Contents-Relation Grouping (CRG) means measuring the similarity between different Port-Relation groups. The CRG process utilizes payload contents and communication pattern of each Port-Relation groups. Based on the payload contents, we can capture the common byte pattern of application protocol. We also examine communication patterns in terms of the number of source/desti-

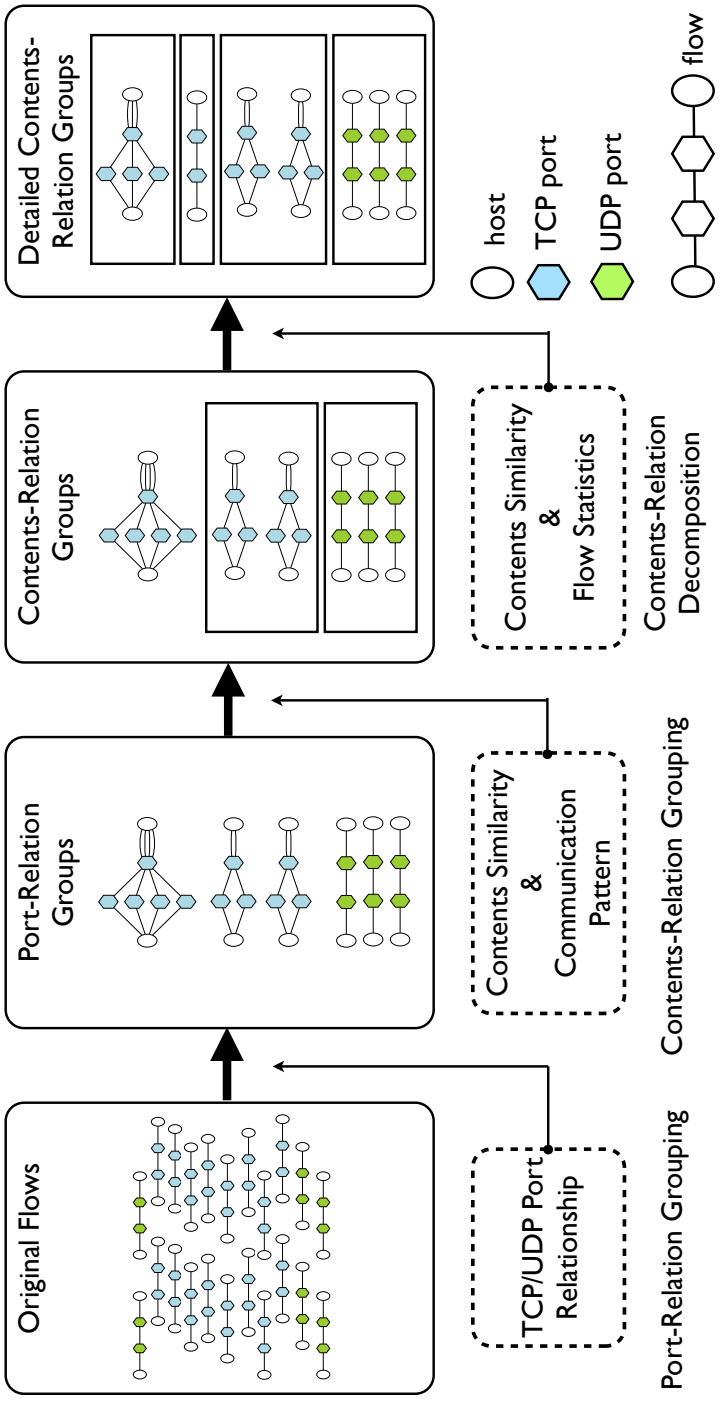


Figure III.8: Overall process of functional separation

nation ports. It can reduce unnecessary flow comparison overhead in contents examination.

3. Contents-Relation Decomposition (CRD) means dividing Contents-Relations groups based on the contents similarity again. In PRG step, we assort flows according to their port numbers not actual contents of application protocols. Therefore, original Port-Relation group might have different functional flows, which allocate the same port number by chance.

Our proposed method for functional separation normally works on both TCP and UDP flows. For the functional separation step, we do not consider unintentional packet drops when we manipulate flows. In case of a general traffic monitoring and measurement system which located in the middle of a certain network link, unintentional packet drops occurs when the system is exposed to an asymmetric routing environment or when the system has limited capacity for capturing packets. However, the offline processes (data collection, functional separation, and classification filter extraction) of the fine-grained traffic classification works on an end host. In this case, a TCP protocol provides reliable and ordered delivery of a stream of bytes as the nature of their design choices. The following subsections will explain each functional separation steps in detail.

3.3.1 Port-Relation Grouping

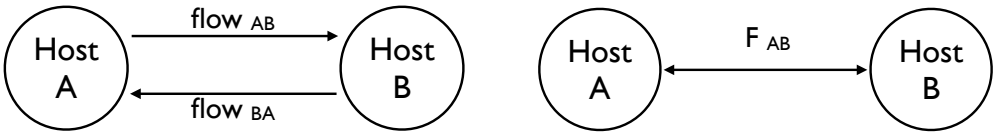
The Port-Relation Grouping (PRG) classifies individual flows according to the dependencies on port numbers. The PRG begins with very simple assumptions.

Assumptions

1. Packets that occur in the close time interval and share the same 5-tuple (source IP address, source port, destination IP address, destination port, and protocol) had originated from the same functionality.

2. Reverse packets (displacement of 5-tuple information, protocol must be the same) in the close time interval ($\leq$ 1 minute) belong to the same functionality.

These assumptions follow the fundamentals of the concept of flow. The packets belonging to these categories can be treated as being from the identical functionality without loss of generality.

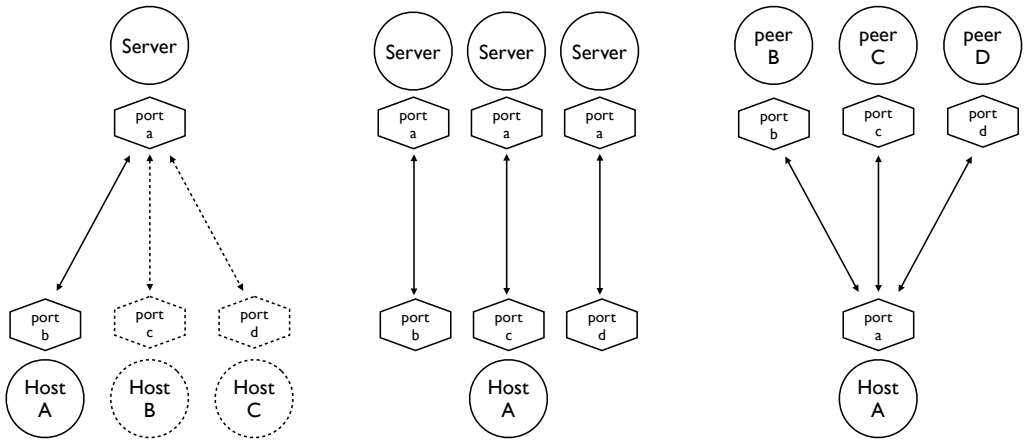


(a) Flows (b) Bidirectional flow

Figure III.9: Bidirectional flow diagram

Figure III.9 illustrates these assumptions in a diagram of flows. When a host A tries to establish a TCP session with host B, a SYN packet is sent from A to B and this packet initiates a flow ($flow_{AB}$). Based on the TCP 3-way handshaking operation, B sends a SYN-ACK packet back to A and this packet initiates a reverse flow ($flow_{BA}$). These two flows form a TCP connection and they can be considered as a same group of functionality. For the sake of simplicity, we treat a flow and its reverse flow as a bidirectional flow.

Figure III.10 describes a host's connection behavior and how the PRG classifies flows into certain RG groups based on these connection behaviors. Figure III.10 (a)



(a) Server/Client (b) Sever/Client (c) Peer-to-Peer

Figure III.10: Connection behavior of a host

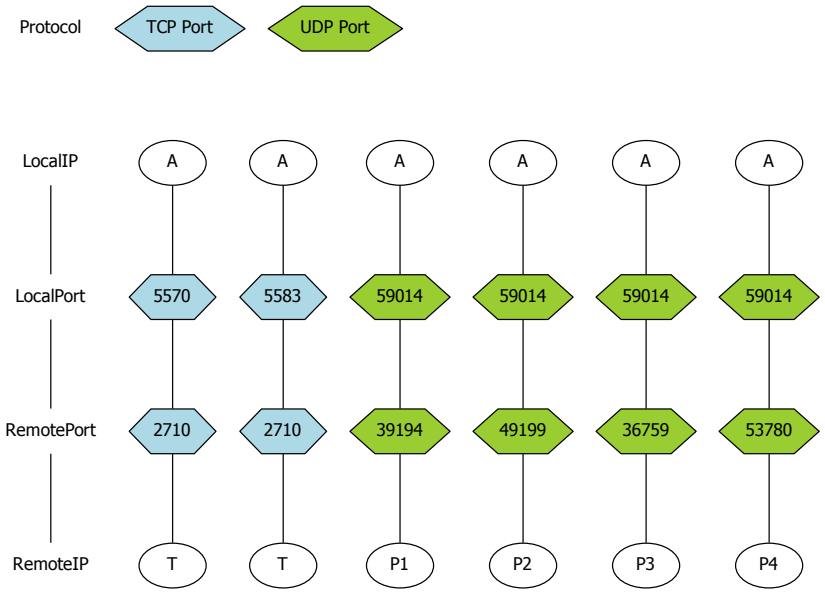
illustrates connections between a server and three clients. In TCP connection, a server opens a listening port (port a) and waits for clients' connection requests. We can say that connections are generated from a same functionality if different clients connect to a server using the server listening port even if the clients allocate different ports (b, c, d). However, our input traffic data is collected from the client side (host A), and we cannot compare connections generated from other clients. Thus, we focus on different connection behaviors occurring on the client. In Figure III.10 (b), the host A connects to three different servers using a common server listening port number. An example of this connection behavior can be easily found in a Web browser. When a browser connects to a Web page, the Web page may consist of various contents, which may be located in different servers. In this situation, the client establishes extra connections to receive contents from different servers and those connections utilize a common server listening port unless other port numbers

are assigned explicitly. Figure III.10 (c) demonstrates the connection behavior of host A when the host runs a P2P application. Unlike a server-client model, a host acts both as a server and client. When the host acts as a client, the connection behavior can be treated as a normal server-client model described in (b). When a host acts as a sever, the host opens a listening port and waits other peers' connection requests. Therefore, the flows that have a same local port number can be grouped into one class.

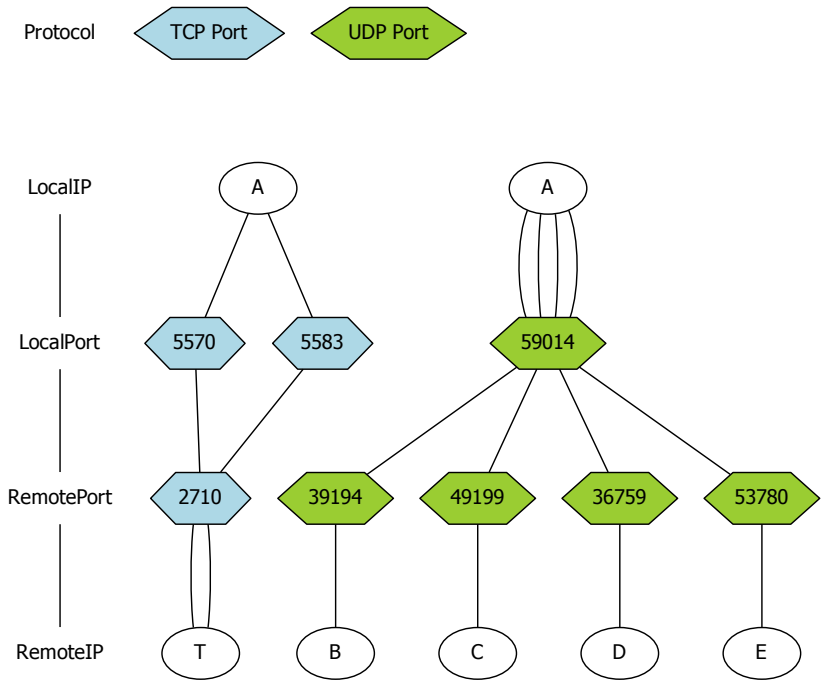
Algorithm 1 describes the PRG process. A flow $f_{(sip,dip,sport,dport,proto)}$ is a sequence of packets with the same 5-tuple header values: source IP address, destination IP, protocol number, source port, and destination port. One TCP or UDP connection consists of two different flows. Therefore, a flow f_a and its reverse flow f_b belong to a same bidirectional flow and same PR group (line 5∼11). Because the input data is collected from a host, a bidirectional flow can be described as $F_{(localip,localport,remoteport,remoteip,proto)}$.

In a single execution of an application, different functionalities of the application create multiple flows concurrently. Even though a port number that is to be allocated is determined in a run time, flows that allocate a same port (either local or remote) simultaneously belong to the same PR group (line 12∼22). The best instance of this is the behavior of a P2P client. A P2P client is required to allocate a port in order to upload data when the client joins a P2P network. Although the port number is not predictable, once the client allocates a port, every flow established using the port is now designated for uploading data.

Figure III.11 depicts an example of PRG on a real BitTorrent traffic. Sample traffic shown in this example is captured when a BitTorrent client downloads a



(a) Bidirectional flows



(b) Port-Relation groups

Figure III.11: An example of Port-Relation Grouping on BitTorrent traffic

Algorithm 1: Pseudo algorithm for Port-Relation Grouping

```
procedure FlowGrouping
input  : Flows, Flow = {f_1, f_2, ..., f_n}
output: Port-relation groups, PR
1  begin
2      PR = {[ ], [ ], ..., [ ]}                    // initialize port-relation group
3      F = {[ ], [ ], ..., [ ]}                     // initialize bidirectional flow set
4      for 1 ≤ i ≤ n do
5          for i ≤ j ≤ n do
6              if f_i = reverse(f_j) then
7                  F ← f_i,f_j
8              end
9          end
10     end
11     for 1 ≤ i ≤ |F| do
12         if |PR| = 0 and i = 1 then
13             PR_0 ← C_i
14         end
15         else
16             for 1 ≤ i ≤ |PR| do
17                 if localPort(PR_j) = localPort(C_i) or
                   remotePort(PR_j) = remotePort(C_i) then
18                     PR_j ← C_i
19                 end
20             end
21         end
22     end
23 end
```

file, and the figures only contain some parts of the traffic for simplicity's sake. A BitTorrent client uses both TCP and UDP flows when it downloads a file. The TCP flows are used for getting the file's hash value from a tracker while UDP flows are for actual downloading. Figure III.11 (a) describes the bidirectional flows (aggregation of a flow and its reverse flow) and Figure III.11 (b) describes the PR groups after

flows are assorted based on either their local or remote port numbers. A connected graph represents a PR group. In this example, the output of PRG is two PR Groups.

3.3.2 Contents-Relation Grouping

When some flows allocate different port numbers, the PRG cannot classify them into a same group even if they actually belong to the same functional group. A case in point is the behavior of a P2P client that allocates random ports. A P2P client connects to different peers at the same time and each flow might allocate a random port. Therefore, it is impossible to place all flows into one functional group using PRG even if those flows are established for a same purpose (e.g., searching or downloading). To solve this shortfall of PRG algorithm, we have developed a method called Contents-Relation Grouping (CRG). The main role of CRG is to connect the preliminary PR groups according to their inter-dependencies in terms of contents similarity.

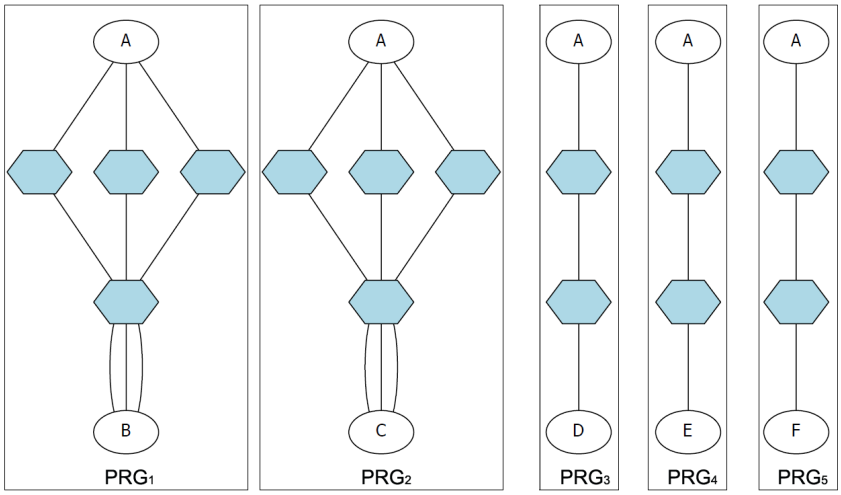


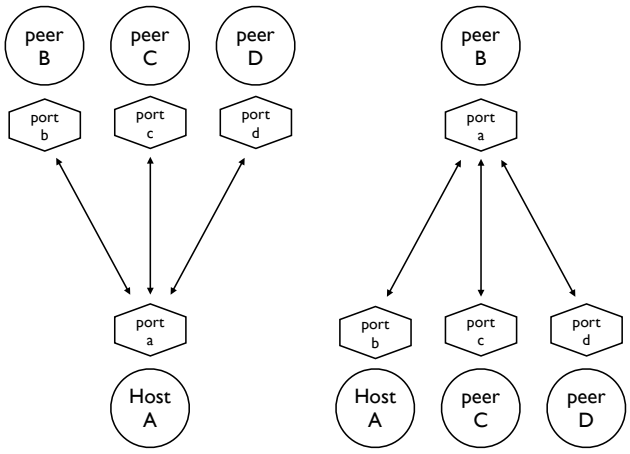
Figure III.12: An example of connection patterns

To combine inter-related PR groups, we measure the degree of contents similarity among PR groups. If the similarity metric between two PR groups exceeds a certain threshold value, then the PR groups are merged into one CR group. When we compare two different PR groups, we take into consideration the connection patterns of these PR groups. We define the connection pattern of a PR group as {1:number of local port:number of remote port:number of destinations}. The first number 1 indicates the number of source hosts. In our functional separation, we collect data from a selected host, which make the value always remain one. Figure III.12 shows an example of connection patterns of PR groups. PRG_1 and PRG_2 have $1:3:1:1$ pattern while PRG_3, PRG_4, and PRG_5 have $1:1:1:1$ pattern.

If different PR groups are generated by a same functionality, they are likely to have common connection pattern. Thus, we measure the contents similarity of PR groups that have same connection patterns (i.e.,$\{PRG_1, PRG_2\}$, $\{PRG_3, PRG_4, PRG_5\}$).

In addition to the connection patterns, we also consider the relationship between a local port number and remote port number of bidirectional flows. In the PRG, bidirectional flows are grouped according to a common local or emote port number. A P2P host acts both as a client and server as illustrated in Figure III.13.

In Figure III.13 (a) the host A operates as a server. Peers connect to the host A through A's listening port a. Thus, the bidirectional flow F_{AB}, F_{AC}, and F_{AD} form a PR group, and their common local port number is a. If the host A operates as a client (Figure III.13 (b)), a bidirectional flow F_{AB} forms a PR group. If the protocol used in both cases is identical, then these PR groups should be merged. However, the PRG cannot group them due to their different port numbers. To solve this problem, CRG compares two different PR groups to see whether their local port



(a) Server mode (b) Client mode

Figure III.13: Connection behavior of a P2P host A

number and remote port number is identical.

In order to compare the contents similarity, we adopt a document retrieval technique [95] which is one of the main research area in NLP filed. The key concept of document retrieval is that the similarity between documents can be measured by the frequency of keywords appearing in the documents. We have defined several key terms used when applying document similarity to traffic classification. This subsection provides our payload vector conversion, vector comparison, and flow comparison methodologies.

Payload vector conversion.

To represent network traffic as a text document, we use vector space modeling (VSM). VSM is an algebraic model that represents text documents as vectors. The objective of document retrieval is to find a subset of documents from a set of stored

text documents D which satisfy certain information requests or queries Q. Provided that a document space consists of documents D_i, each will be identified by one or more index terms T_j; the terms may be weighted according to their importance [96].

A typical way to determine the significance of a term is to measure the occurrence of the term T_j. When t different index terms are presented in document D_i, each document D_i is represented by a t-dimensional term-frequency vector $D_i = (d_{i1}, d_{i2}, \cdots, d_{ij})$ where d_{ij} representing the frequency of the j-th term. While text documents are composed of terms (words) which are units of language with functions as a principal carrier of meaning, a packet does not have basic units containing a certain meaning. We have defined the *term* of a payload as follows address this issue

Definition III.1 *A term is a payload data within a i-bytes sliding window where the position of the sliding window can be $1, 2, \cdots, n-i+1$ with n bytes payload. The size of the term set is $2^{8\times i}$, and the length of a term is i.*

If the length of word i is too short, the word cannot reflect the sequence of the byte patterns in the payload. In this case, we cannot recognize the differences among permutations of byte patterns, such as "0x01 0x02 0x03" and "0x03 0x01 0x02". If the word length is too long, the number of whole representative words increases exponentially. With definition III.1, a packet can be represented as a term-frequency vector called *payload vector*.

Definition III.2 *When w_i is the occurrence of the i-th term that appears repeatedly in a payload, the payload vector is*

$$Payload\ Vector = [w_1 w_2 \cdots w_n]^T, \tag{III.1}$$

where n is the size of whole representative term set.

We set the sliding window size i to 2 because it is the simplest case for representing the order of content in payloads. When the term size is 2byte, the size of all terms is 2^{16}. Therefore, the payload vector is represented as a 2^{16}-dimensional term-frequency vector.

Payload vector comparison.

Once packets are converted into vectors, the degree of similarity between packets can be calculated by measuring the distance between vectors. We use Jaccard similarity as a distance metric. In our previous work [97], we compared three different similarity metrics: Jaccard similarity, Cosine similarity, and RBF, and Jaccard similarity showed the best performance without using any sophisticated techniques. The Jaccard similarity $J(X, Y)$ draw word sets from the comparison instances to and use them evaluate similarity. $J(X, Y)$ is defined as the size of the intersection of the word sets divided by the size of the union of the sample sets X and Y:

$$J(X, Y) = \frac{|X \bigcap Y|}{|X \bigcup Y|} \tag{III.2}$$

One of the advantages of using Jaccard similarity instead of Euclidean distance is that the similarity value can be normalized, which enable the similarity to be

calculated by the dot product, and it approaches one if the vectors are similar and zero otherwise. Our payload vectors have very high dimensionality, so statistically they are very sensitive. If two payload vectors are generated by different functionalities, then the contents of each payload consist of distinct word (or binary) sequences and their vectors are also very different. Because most signatures of the application traffic are a small portion of the payload data, we may ignore the other part of the payloads signatures which is actually arbitrary binary data.

Flow similarity comparison.

Formula V.3 defines the payload flow matrix (PFM). The i-th row of a PFM is the payload vector of i-th packet in the flow. PFM is a $k \times n$ matrix, where k is the number of packets and n is the dimension of payload vectors.

Definition III.3 *Payload flow matrix (PFM) is*

$$PMF = [\vec{p_1}, \vec{p_2}, \cdots, \vec{p_k}]^T, \tag{III.3}$$

where $\overrightarrow{p_i}$ is payload vector defined in Definition III.2.

The similarity score between PFMs can be calculated by simply adding up the packet similarity values.

Definition III.4 *Similarity Score* $= \sum_{i=1}^{k} J(p_i, p_i') \cdot \alpha$, *where* p_i *and* p_i' *are* i*-th packet of first and second flow accordingly.*

Algorithm 2 describes the grouping process of CRG. CRG procedure reads bidirectional flows and groups them into CR group based on their similarity scores. If

Algorithm 2: Pseudo algorithm for CRG using similarity

procedure Contents-Relation Grouping
input : Bidirectional flows, $Flow = \{F_1, F_2, \cdots, F_n\}$
output: Flow groups, $FG = \{fg_1, fg_2, \cdots, fg_m\}$

```
1  begin
2      FG = {[ ], [ ], ··· , [ ]}                          // initialize flow group
3      for 1 ≤ i ≤ n do
4          if i = 1 then
5              FG[0] ← F_i
6          end
7          else
8              M = PFM(F_i)
9              for 1 ≤ j ≤ number of flow group do
10                 Similarity{} ← SimilarityScore(FG[j], M)
11             end
12             if Max(Similarity) ≥ threshold then
13                 FG[Max index] ← F_i
14             end
15             else
16                 FG ← F_i                                 // create a new flow group
17             end
18         end
19     end
20 end
```

CR group set is empty, the first flow F_1 creates a new flow group $FG[0]$ (line 4∼6). Otherwise, input flow is compared with existing CR groups and inserted into a CR group with a maximum flow similarity score (line 12∼14). When the maximum similarity score is less than a certain threshold value, a new CR group is created and the bidirectional flow F_i becomes a member of the new flow group (line 16).

Algorithm 3: Pseudo algorithm for CRD using similarity

procedure Contents-Relation Decomposition
input : CR groups, $CR = \{CR_1, CR_2, \cdots, CR_n\}$
output: DCR groups, $DCR = \{DCR_1, DCR_2, \cdots, DCR_m\}$

```
1  begin
2      FG = {[ ], [ ], ··· , [ ]}                          // initialize flow group
3      for 1 ≤ i ≤ n do
4          if isInRPGroup(CR_i) then
5              FG ← Decompose(CR_i)
6          end
7      end
8      n ← |FG|
9      DCR = {[ ], [ ], ··· , [ ]}                          // initialize DCR group
10     for 1 ≤ i ≤ n do
11         if i = 1 then
12             DCR[0] ← F_i
13         end
14         else
15             M = PFM(F_i)
16             for 1 ≤ j ≤ |FG| do
17                 Similarity{} ← SimilarityScore(DCR[j], M)
18             end
19             if Max(Similarity) ≥ threshold then
20                 DCR[Max index] ← F_i
21             end
22             else
23                 DCR ← F_i                                // create a new DCR group
24             end
25         end
26     end
27 end
```

3.3.3 Contents-Relation Decomposition

In the CRG step, only PR groups with same connection patterns or same representative ports are compared and have a chance to be merged into CR groups. Otherwise,

PR groups form independent CR groups on their own. In other words, these CR groups are created without any contents examination.

The PRG procedure classifies flows according to the dependency of assigned port numbers. Such a grouping process of PRG works effectively when a functionality is mapped to a single port number. However, the PRG cannot perceive the different functional traffic generated by a single port number. As a result, a PR group represented by TCP port 1863 may have different functionalities such as getting buddy lists, messaging, etc. The Contents-Relation Decomposition (CRD) process compensates this kind of PRG's defect.

The CRD discriminates different functionalities in a CR group which consists of a PR group based on contents similarity. The same calculation method used to measure contents similarity in Section 3.3.2 can be utilized again, only in the opposite direction. On the contrary to CRG, if similarity metric between two flows in a CR groups does not exceed a certain threshold value, then the CR group is partitioned into different CRD groups. Algorithm 3 describes the decomposition process of the CRD. First, the procedure examines every CR group to check whether a CR group is composed with a single PR group. If a CR groups is formed with a single PR group, the CR group is disassembled into individual bidirectional flows (line 4∼6). If a CRD group set is empty, the first flow F_1 creates a new DCR group $DCR[0]$ (line 12∼14). Otherwise, an input flow is compared with already existing CRD groups and inserted into a CRD group with a maximum flow similarity score (line 19∼21).

Figure III.14 shows an example of the functional separation processes including RPG, CRG, and CRD on a MSN traffic. As mentioned above, MSN messenger use TCP 1863 port to communicate with MSN servers. Bidirectional flows F_1, F_2, F_3,

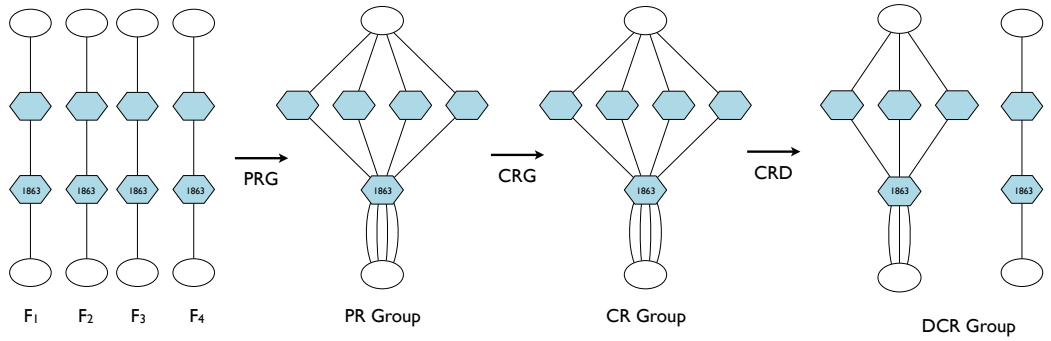


Figure III.14: An example of the functional separation on MSN traffic

and F_4 use TCP 1863 as their remote port number in common. These flows are grouped into a same PR group based on PRG. The PR group has 1:4:1:1 connection pattern and its representative port number is TCP 1863. If there is no other PR group which has 1:4:1:1 connection pattern or TCP 1863 as its representative port, the PR group is not examined and simply forms a CR group without any change. If F_1, F_2, and F_3 are generated by same functionality and F_4 is generated by another functionality, the results of CRD are two different DCR group as $\{F_1, F_2, F_3\}$ and $\{F_4\}$.

Chapter IV

TRAFFIC CLASSIFICATION FILTER EXTRACTION

In this chapter, we propose an automated traffic classification filter extraction methodology. There are various types of traffic classification filters such as well-known port, session behavior, payload signature, and ML-based classifiers. Among various types of traffic classification filters, we focus on generating payload signatures due to its high accuracy and applicability. Traditionally, Internet applications have been identified by using predefined well-known ports with questionable accuracy. An alternative approach, application layer signature mapping, involves the exhaustive search of reliable signatures but with more promising accuracy. With a prior protocol knowledge, the signature generation can guarantee a high accuracy. As more applications use proprietary protocols, it becomes increasingly difficult to obtain an accurate signature while avoiding time-consuming and manual signature generation process. This chapter proposes an automated approach for generating application-level sig-

nature, the LASER algorithm, which does not need to be preceded by an analysis of application protocols.

4.1 Signature Formats

In this section, we show the commonly used signature formats for worms and redefine the signature formats prior to describing our signature extraction approach.

4.1.1 Anomaly Signature Format

Newsome *et al.* [98] categorized the signature format for polymorphic worms, which refer to the worms that vary their payload on every infection attempt. Their definition does not completely cover the traffic other than worms; these signatures are simply sequences of substrings. The followings are the signature classification for polymorphic worm:

1. **Conjunction signature**: A signature that consists of a set of substrings (or tokens), and matches a payload if all tokens in the set are found in it, in any order.

2. **Token-subsequence signature**: A signature that consists of an ordered set of tokens.

3. **Bayes signature**: A signature that consists of a set of tokens, each of which is associated with a score, and an overall threshold.

Brumley *et al.* [99] also proposed several representations of worm signatures: Turing machine signatures, symbolic constraint signatures, and regular expression

signatures. However, there exists tradeoff between the expression power and matching efficiency. It is not suitable for real-world applications, especially when matching against large number of application signatures.

4.1.2 Innocuous Application Signature Formats

Signature-based identification research that proceeded up to now defined their own signature formats and developed solutions corresponding to their formats to analyze application traffic. We categorize several established signature formats as follows:

- **Common string with fixed offset**: defines the signature as string or hex values that appear on a specified offset of packet payload. In most cases, the offset is the beginning of packet's payload (e.g., eDonkey '0xe3' at offset 0).
- **Common string with variable offset**: this format is almost the same as the former except for the offset value. In common string with fixed offset, the offset of signature is constant; however in this case, common string can appear at any position in payload.
- **Sequence of common substrings**: different from the former two formats, it defines the signature as an order of some substrings that appear on the payload. It means that one string or hex value cannot be a signature, since when one string is considered as a signature, the accuracy cannot be guaranteed.
- **Behavioral signature**: depends on statistical characteristic of application traffic such as packet inter-arrival time, minimum packet size, maximum packet size, etc. Machine learning techniques based on various measurement values

are widely studied. This format was not considered in our work since our goal is to achieve automated signature generation by inspecting packet data.

Karagiannis *et al.* [5] used the common string with fixed offset for their work. However, when used for signature generation, the accuracy of signature can decrease, since the signature is generated without an analysis of application protocols. For example, if the protocol field's length is variable, then the offset is variable as well. Thus, such problem is solved by using common string with variable string format. Since many applications use HTTP protocol for convenience these days, 'GET' or 'HTTP' string on payload are frequently found. However, those strings cannot distinguish applications; hence for the signature formats mentioned prior to this, those strings cannot have any meaning as a signature. Sen *et al.* [6] described how to increase the signature accuracy by using a sequence of common substrings format. If we use this format, 'HTTP' or 'GET' can be treated as a part of signature.

Taken altogether, the sequence of common substrings format is a superset and the most accurate format among all signature formats except for the behavioral signature. Therefore, our approach uses a sequence of common substrings.

4.2 Signature Extraction Process

For the payload signature extraction, we adopted the Longest Common Subsequence (LCS) algorithm [100], which is one of the classical computer science problem and the basis of diff (a file comparison program that outputs the differences between two files). The LCS also has applications in bioinformatics such as DNA sequence matching. DNA sequence matching calculates common subsequences and measures

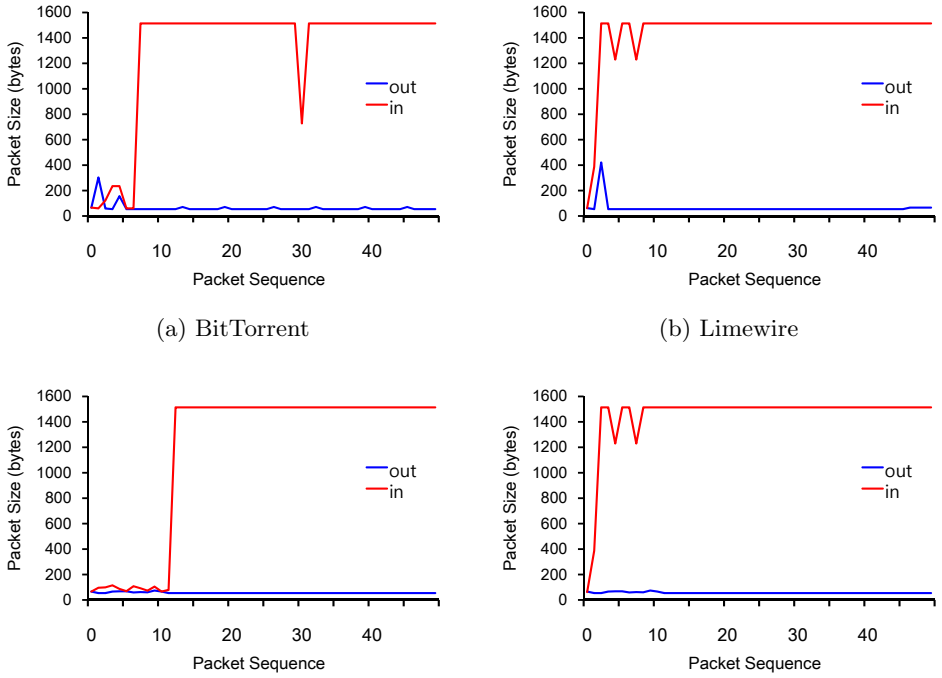
the similarity between DNA of different organisms. The basic idea of extracting common strings from packet payloads is analogous to DNA sequence matching; however, there are fundamental differences between DNA and packet payload, such as a basic unit of string composition. Thus, we modified LCS algorithm suitable for signature generation. Packet payloads substitute for DNA sequence, and the LASER algorithm extracts common substrings as a signature from the traffic generated by the same application.

Because the application signature extraction deals with a larger number of strings being compared, some modifications on LCS are needed. We describe the following constraints as our modification to LCS. Note that, we are also interested in all possible common substrings that can be found along with the longest one.

4.2.1 Constraints for Signature Extraction

Number of packets per flow A flow is a collection of packets that share the identical source IP, destination IP, source port, destination port, and protocol. It is impossible and unnecessary to exploit all packets in flows, especially conducting an expensive deep packet inspection. Sen *et al.* [6] also observed that the concrete signature exists in the initial few packet of the flow. Thus, we limited the number of packets for the efficiency of the algorithm.

Minimum substring length The generated signature consists of the sequence of substrings. The computational cost of signature matching is proportional to the length of the signatures [101]. From this point of view, shorter substrings are desirable for the efficiency of signature matching process. Nevertheless, the length of



(a) BitTorrent (b) Limewire

(c) FTP (d) Fileguri

Figure IV.1: Packet size distribution of first one hundred packets

substrings in a signature reflects their significance as a reliable signature for accurate classification. In order to avoid any trivial signatures, a minimum bound for the substring length should be considered as a constraint for the LASER algorithm. For example, if a single character is determined as one of the common substrings during the signature generation process, it would be difficult to conclude that it was an actual signature unless it appears repeatedly in the fixed offset position. In fact, the applications that employ the HTTP protocol contain the frequent use of '/' in its packets. By having this length constraint, we prevent these single and multi-positioned characters from involving in the sequence of common strings.

Packet size comparison As explained previously, it increases the possibility of finding a reliable signature if the packets are grouped in a similar purposes (e.g., signaling traffic and downloading traffic) and traffic characteristics. One of the fundamental characteristics for deciding the closeness between the packets is a packet size. While the sanitized flow and fine-grained flow are dealing with characteristics of flows, the packet size comparison deals with characteristics of individual packets. Figure IV.1 illustrates the packet size of first 100 packets on each session for four different applications when initializing the file download: (a) BitTorrent, (b) Limewire, (c) FTP, and (d) Fileguri. Packets are separated into in and out flows.

In and out flows indicate traffic from server to client and vice versa. Each session consists of two flows called in and out flow. In flow is a collection of packets which are sent from server to client and out flow is a collection of packets which are sent from client to server. Most of inbound flow packets are actual downloading packet (with MTU) and most of outbound flow packets are TCP ACK (54 bytes). Although the packet size distributions of each application are little bit different from each other, a first few packets are relatively smaller than that of actual downloading in common and these packets are generated to initiate the connection handshake, such as signaling phase to download a file. The volume of transmitted data at the initial phase is relatively smaller than that of actual downloading. A concrete signature exists only in the initial few packets. Thus, the comparison between the small handshake packets and the large downloading packets is undesirable while generating a reliable signature.

Consequently, the LASER relies on the comparison of packets within the similar range of packet sizes. For example, LimeWire, a popular Gnutella application, shows

that its signaling traffic – connection control, search, etc. – are dealt with relatively light packets of average 390 bytes [102]. However, the average packet size for the traffic containing the actual download is 1514 bytes. These two types of traffic are less likely to share any signatures between them because their role is different in the application.

Algorithm 4: Singature generation using LASER part.1

procedure Applying constrains for signature generation)
input : Fine-grained traffic trace (DCR groups),
$DCR = \{DCR_1, DCR_2, \cdots, DCR_n\}$
output: Application signature, $Sig = \{Sig_1, Sig_2, \cdots, Sig_n\}$

```
1  begin
2      for 1 ≤ i ≤ n do
3          flow[ ] ← DecomposeToFlow(DCR_i)
4          while 0 ≤ i ≤ Packet number constraint do
5              while 0 ≤ j ≤ Packet number constraint do
6                  if |flow_1[i].packet_size − flow_2[j].packet_size| <Packet size
                   constraint then
7                      result LCS ← LASER(flow_1[i], flow_2[j]) LCS Pool [ ] ←
                       append(result LCS)
8                  end
9                  j ← j + 1
10             end
11             i ← i + 1
12         end
13         Signature ← select the longest LCS from LCS Pool
14         forall the remaining flows in flow[ ] do
15             f ← select one from the rest of flows result LCS ←
               LASER(Signature, f)
16         end
17         Sig_i ← select the longest LCS from LCS Pool
18     end
19     return Sig
20 end
```

Algorithm 5: Singature generation using LASER part.2

procedure LASER (Building matrices)
input : Two byte stream, $Stream_A, Stream_B$
output: Longest common subsequences, LCS

```
1  begin
2      Stream_A[m, ..., 1] ← ReverseByteStream(Stream_A[1, ..., m])
3      Stream_B[n, ..., 1] ← ReverseByteStream(Stream_B[1, ..., n])
4      m ← |Stream_A|
5      n ← |Stream_B|
6      Matrix_A[m][n]
7      Matrix_B[m][n]
8      while 0 ≤ i ≤ m do
9          while 0 ≤ jn do
10             if i = 0 or j = 0 then
11                 Matrix_A[i][j] ← 0
12             end
13             else if Stream_A[i] = Stream_B[j] then
14                 Matrix_A[i][j] ← Matrix_A[i − 1][j − 1] + 1 Matrix_B[i][j] ←
                   Diagonal
15             end
16             else
17                 Matrix_A[i][j] ← max(Matrix_A[i][j − i], Matrix_B[i − 1][j])
18                 if Matrix_A[i][j] ≠ Matrix_A[i][j − 1] then
19                     Matrix_B[i][j] ← Up
20                 end
21                 else
22                     Matrix_B[i][j] ← Left
23                 end
24             end
25         end
26     end
27     resultLCS ← trackMatrix(Matrix_A, Matrix_B)
28     return result LCS
29 end
```

4.2.2 LASER Algorithm

Algorithm 4, 5, 6, and 7 describe the overall signature generation process and the LASER algorithm. The LASER algorithm takes fine-grained traffic trace (described

Algorithm 6: Singature generation using LASER part.3

procedure LASER (Tracking matrices)
input : Two matrices, $Matrix_A, Matrix_B$
output: Longest common subsequences, LCS

```
1  begin
2      i ← m − 1
3      j ← n − 1
4      while Matrix_B[i][j] ≠ 0 do
5          if Matrix_B[i][j] =Left then
6              j ← j − 1
7          end
8          else if Matrix_B[i][j] = Up then
9              i ← i − 1
10         end
11         else if Matrix_B[i][j] = Diagonal then
12             Substring ← Stream_A[i]
13         end
14         if Matrix_B[i − 1][j − 1] ≠Diagonal then
15             Substring ← append break point character i ← i − 1 j ← j − 1
16         end
17     end
18     while tokenizing substring based on break point do
19         if |token| >minimum substring length constraint then
20             result LCS ← append token
21         end
22     end
23     return result LCS
24 end
```

in Chapter III) as its input data and discovers common patterns shared by flows. The fine-grained traffic trace is consists of DCR groups which are traffic grouped according to application and its functionalities. The signature generation process begins with decomposition of DCR group (Algorithm 4: line 2). The target DCR group is decomposed into flows which are a collection of packets sharing the identical

Algorithm 7: Signature refinement process of LASER

procedure Signature refinement process
input : Flows, $Flow[flow_1, \cdots, flow_k]$
output: Final application signature, Signature

```
1  begin
2      i, j, l ← 1
3      Candidate Signature_i ← sigGeneration(flow_j, flow_{j+1})
4      while do
5          i ← i + 1
6          j ← j + 2
7          Candidate Signature_i ← sigGeneration(flow_j, Candidate
           Signature_{i−1})
8          if Candidate Signature_i = Candidate Signature_{i−1} then
9              break
10         end
11     end
12     Signature ← Candidate Signature_i
13     return Signature
14 end
```

5-tuple information.

Initially, the two distinct fine-grained flows – $flow_1$ and $flow_2$ – are used as input to the first iteration. The inputs to the LASER algorithm itself are two distinct byte streams of packet payloads that belong to $flow_1$ and $flow_2$ (Algorithm 4: line 7). The packet size constraint (Algorithm 4: line 6) and the number of packets per flow constraint (Algorithm 4: line 5∼6) are applied ahead of LASER procedure which is actual patter extraction process based on LCS problem (Algorithm 5 and 6).

Figure IV.2 illustrates the step after the packet selection based on the packet size constraint and the number of packets per flow constraint. It shows two byte streams from LimeWire. The candidate signature in the figure shows the preliminary signature as the following: *'HTTP 200 OK Server : LimeWire/ Content-type : **

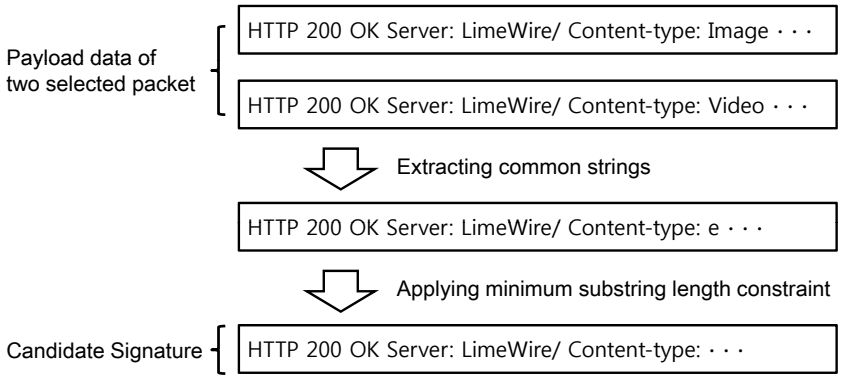


Figure IV.2: An example of applying minimum substring length constraint

*e *'*. However, the substring *'e'* is too short to have any meaningful decision. We remove this ambiguous substring from the current signature according to the minimum substring length constraint (Algorithm 6: line 19∼21). The candidate signature after the first iteration is defective because its sample set is small – only two flows (up to line 15). More details on generating the final signature will be covered in the signature refinement process. In addition, the substring like, *'HTTP 200 OK'* is often engaged by many other applications using HTTP protocol. Its lone usage is insufficient to be a unique signature. When it comes as a part of common sequences, it can be a meaningful substring.

Once a candidate signature is obtained, signature refinement process is required. This is a refining process of eliminating trivial strings (or hexadecimal values) from the candidate signatures by iterating through the collected flows. Trivial strings imply any string that is not qualified for concrete signature, like replaceable information in the payload (e.g., IP address, object referrer in URL, etc.) In other words, the strings do not appear in the application protocol format. The candidate signatures

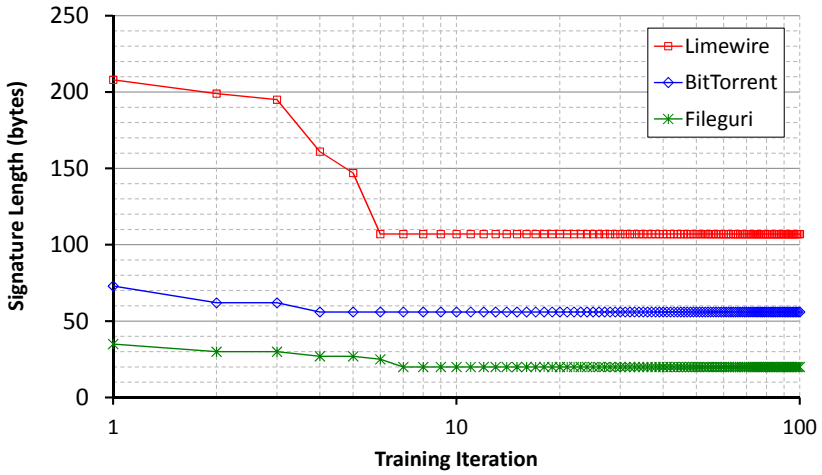


Figure IV.3: Candidate signature length according to signature refining iteration

after the first iteration using the two flows are fed back to the LASER procedure along with another flow (Algorithm 4: line 15∼20). A new set of substrings after each iteration is added to or removed from the pool of candidate signatures. The iteration continues until the numbers of candidate signatures are fixed or all the flows are iterated. Ultimately, as more training steps are processed, the signature accuracy increases. The signature refinement process can be simply expressed as Algorithm 7.

Figure IV.3 shows the signature refinement process as the number of iteration increases. It indicates a decrease in length of candidate signature as refining iteration continues. After the certain point, the length of candidate signatures remains still which implies an iteration termination point in the proposed algorithm. The number of refining iteration to have the final signature is below 10 on all three applications. Surprisingly, a small amount of traffic input is needed.

4.3 Comparison with Manually Searched Signatures

To confirm the validity of the LASER algorithm, we have compared signatures generated by LASER with manually searched signatures by semantic analysis of application protocols.

We have selected three popular P2P applications – LimeWire, BitTorrent, and Fileguri – based on their popularity and current availability of the concrete signatures to validate our LASER algorithm. LimeWire is well-known P2P application worldwide that uses the Gnutella protocol. BitTorrent is also a popular file sharing application for large files, such as movies. Both applications are popular choices for previous signature-based identification research [5, 6]. Sen *et al.* [6] have generated the signatures of a few P2P applications including LimeWire and BitTorrent by analyzing the application-layer protocols; their signatures could reduce FP and FN to 5% of the total bytes. Finally, Fileguri is a regional P2P application. Won *et al.* [103] presented the manually generated signature of this application.

Table IV.1 presents the signatures generated by the following three distinct generation methodologies: Packet analysis, protocol analysis, and our proposed method. Packet analysis extracts a signature from raw packets by inspecting every individual packet without a prior knowledge of the application protocol structure and behavior. Meanwhile, protocol analysis decides the signatures based on the corresponding protocol semantics which often requires protocol reverse engineering. When the application protocol is publicly available, we assumed that protocol analysis is the most accurate method for signature generation. In fact, the previous research [6] claims that they achieve over 99% signature accuracy.

Table IV.1: Comparison of the generated signatures by packet analysis, protocol analysis, and LASER

	LimeWire	BitTorrent	Fileguri
Packet Analysis	"GNUT"	"0x13Bit"	"Freechal"
Protocol Semantic Analysis	"GET" or "HTTP" followed by "User-Agent: Limewire" or "UserAgent: Limewire" or "Server: Limewire"	"0x13BitTorrent protocol"	N/A
Automated Signature Generation by LASER	Sequence of 10 substrings	Sequence of 1 substring	Sequence of 6 substrings
	"LimeWire", "Content-Type:", "Content-Length:", "X-Gnutella-Content-URN", "run:sha:1", "X-Alt", "X-Create-Time:", "X-Features:", "X-Thex-URI"	"0x13BitTorrent protocol"	"HTTP", "Freechal P2P", "User-Type:", "P2PErrorCode:", "Content-Length:", "Content-Type:", "Last-Modified"

For the LASER algorithm, the following constraint conditions are applied. The number of initial packets to compare is set to 10 according to [104]. We also cross-verify this count by measuring packet size distribution in Section IV. The minimum substring length is set to 3. These constraints are adjustable depending on the type of applications.

- **LimeWire**: The signature found after packet analysis is “GNUT” which is relatively short. Its sole use in traffic identification cannot guarantee the identification correctness due to the possible signature collision with non-LimeWire packets. The signatures by protocol analysis are compared to those by the LASER algorithm, which shows that they share one clear common substring – “Limewire”. The substrings that appear only in the list of LASER signatures, such as “XGnutella-Content-URN” are clearly one of the Gnutella protocol’s

fields.

- **BitTorrent**: The packet analyzed signature is shorter than the protocol analyzed signature, but the LASER signature finds the identical signature from protocol analysis.

- **Fileguri**: This application uses undisclosed proprietary protocol; hence, protocol analyzed signature is currently unavailable. It is discovered that the LASER signatures are also a super set of the packet analyzed signatures, which is similar to the LimeWire's case.

We observe that the LASER signatures are either identical or close to the signatures from the rest of the methods. In fact, some sets of strings, which were spotted only in the LASER signature, were actually a part of the original application protocol format. This leads to the conclusion that the LASER algorithms can effectively extract the common subsequence strings. We claim that the accuracy of LASER signatures is guaranteed in the term of closeness of string appearance.

Chapter V

Evaluation

In this chapter, we describe the traffic classification using the fine-grained traffic classification based on functional separation. First, we present the functional separation result on different application traffic. We also present the classification result in terms of classification accuracy and compare the accuracy with conventional Deep Packet Inspection (DPI) based application traffic classification methods. Finally, the comparison with an unsupervised machine learning algorithm is presented.

5.1 Traffic Classification using Functional Separation

In this section, we present a traffic classification result using functional separation. First, we describe the functional separation results on a number of applications which have different characteristics in terms of application category and operational architecture. We also present the fine-grained traffic classification results, which utilize the output of the functional separation, in terms of traffic classification accu-

racy. To validate the proposed methods, we compare our methods with two different conventional DPI-based application traffic classification methods.

5.1.1 Target Applications

We have selected fifteen network applications based on their popularity in network and application types. Selected applications include both Internet applications and mobile applications. Application types of selected applications include P2P (file sharing), messenger, Web storage (file hosting service), video/music streaming, and online games.

Table V.1 describes the selected applications and the amount of input traffic data for functional separation. We collected traffic data for functional separation using the TMA and the mTMA described in Section 3.2.

5.1.2 Functional Separation Results

Functional separation is the part of the process that generates arbitrary classifiers (e.g. application signature, connection behavior model, statistical model, etc.) for every application where each classifier corresponds to a distinct function in the application. Even though the amount of the input data is small, it is enough to characterize different functions in a certain application. We applied our functional separation algorithm to the input data.

Table V.2 show the results of functional separation. Since there is no ground truth from the perspective of the application's functionality, we manually analyzed flows in each DCR group and labeled each DCR group with functionality. This labeling process will be utilized for analyzing usage patterns of an application later in

Table V.1: List of selected applications

	Application Type	Name	Operation		Data Set	
			P2P	SC	Bytes	# of flow
Wired Internet Applications	P2P (File sharing)	BitTorrent	o		789,588 K	372
		LimeWire	o		76,296 K	5,071
		Fileguri	o	o	78,768 K	11,044
	Messenger	MSN	o	o	81,420 K	224
		NateOn	o	o	306,388 K	140
	Web Storage (File hosting service)	DropBox		o	12,228 K	64
		uCloud		o	95,176 K	32
	Video Streaming	Gom	o	o	176,192 K	1,044
		PotPlayer	o	o	64,320 K	240
	Online Game	Starcraft	o	o	4,118 K	584
		Starcraft 2		o	48,815 K	98
Mobile Applications	Web Storage (File hosting service)	DropBox		o	16,898 K	44
	Video Streaming	TVPot		o	82,502 K	422
	Music Streaming	Bugs		o	23,523 K	51
	Messenger	NateOn		o	1,690 K	154

SC: Server-Client

the actual traffic classification step. Even if correct labeling is impossible, functional separation will help to increase accuracy and completeness.

The number of DCR group fields only counts the labeled group and other groups are considered as misclassified groups. We also considered a DCR group as a misclassified group when we cannot label its exact functionality. For example, we cannot label passive FTP sessions without any protocol semantic analysis. Although the FTP sessions are grouped together, this group is regarded as a misclassified group. The grouping ratios indicate the proportion of labeled traffic to entire traffic trace

Table V.2: Functional separation result

	Application	# of DCR Groups	Grouping Ratio		Details on DCR Groups
			Byte	Flow	
Wired Internet Applications	BitTorrent	5	97.85%	99.98%	Tracker connection, Download (UDP), Download (TCP), Service discovery, DHT management
	LimeWire	5	99.51%	56.18%	Search, Download, Peer info. exchange, uPnP discovery, DNS query (MDNS)
	Fileguri	7	100%	100%	Authentication, Advertisement, Peer Info, Search, Download, Connection management (UDP), Connection management (TCP)
	MSN	6	96.75%	99.99%	Advertisement, Authentication, Messaging, Remote assistant, Voice chat control, Voice chat
	NateOn	7	100%	100%	Advertisement, Authentication, Messaging, File transfer, Voice chat control, Voice chat, Remote assistant
	DropBox	4	100%	100%	Download, Host discovery, Authentication, Net cache
	uCloud	3	99.89%	75%	Authentication, File transfer, Version management
	Gom	4	99.99%	99.96%	Advertisement, Contents browsing, Streaming, Streaming management
	PotPlayer	6	99.90%	89.71%	Streaming (Broadcast), Streaming (VOD), Service discovery, NTP, Contents browsing, Chat
	Stacraft	2	94.81%	96.92%	Battle net server connection(authentication + update + game list), Game data
	Stacraft 2	4	99.99%	94.12%	Game data, Battle net news, Battle net images, Authentication
Mobile Applications	DropBox	3	100%	100%	Download, Authentication, DNS query
	TVPot	4	100%	100%	Advertisement, Contents browsing, Streaming, DNS query
	Bugs	6	100%	100%	Top music chart, MV image, Album image, Music Stream, Menu browsing, Authentication
	NateOn	3	100%	100%	Advertisement, Messaging, Text Chat

in terms of total bytes and number of flows respectively.

While examining the misclassified traffic separately, we made an observation

that may account for misclassification. Some misclassified flows did not have any application level packet payload except for TCP/IP headers. Some flows even had only one packet for the TCP SYN flag. Our CRG algorithm utilizes application-level packet payloads for the grouping; thus, flow generated under the circumstance where the application only exchanges the packets for TCP connection only containing TCP flags, cannot be grouped with other PR groups due to the lack of payloads data.

Classification accuracy can be described as either flow accuracy or byte accuracy. The former indicates the proportion of correctly classified flow counts out of the entire traffic data set, and the latter indicates how many bytes of traffic are correctly classified by the classification algorithm. Most recent studies have focused on flow accuracy. However, Erman *et al.* [105] argued that byte accuracy is crucial for evaluating the accuracy of traffic classification algorithms. Due to the "elephants and mice phenomenon" [39], the majority of the flows on the Internet are small and occupy only a small portion of the total bytes and packets, while the majority of the traffic bytes are the result of a small number of large flows. The authors analyzed a six-month-period traffic data set and, in terms of bytes, the top 1% of flows accounted for over 73% of the traffic and the top 5% of flows accounted for 83% of the traffic. This traffic trace results in 99.9% flow accuracy but only 54% byte accuracy when their classifier was applied to their experiment. This imbalanced accuracy result shows the importance of byte accuracy: byte accuracy must also be used when evaluating the accuracy of traffic classification algorithms. Therefore, we measured our grouping ratio in terms of both byte and flow. The grouping ratio indicates how the functional separation distinguished each functionality. Even though the flow grouping ratios of LimeWire and uCloud is 56.18% and 75% respectively, the byte

grouping ratio is higher than 94% for every application. Low flow grouping ratio in some applications was mainly caused by the flows which do not have any payload contents. However, these flows have a relatively small size in comparison to the total bytes and rarely affects the byte grouping ratio. Although the test cases were simple and limited, we could confirm that the functional separation has reasonable accuracy.

5.1.3 Traffic Classification Results

For the classification filter extraction, we used the LASER algorithm described in Chapter IV which can generate an application signature automatically. This filter extraction (or signature generation) process and traffic classification using extracted traffic classification filters depend on the DPI approach. We selected the DPI-based approach for a clear reason. The majority of previous works demonstrated that the signature-based approach was the most reliable approach available for accuracy. Even some statistical approaches (machine learning approaches) used signature (or manual payload inspection) as ground truth for validation [5, 24, 29, 30, 31, 42].

As the results of the functional separation, we can get n different traffic group, DCR group, per application. Each group of DCR is used as input data for the application signature extraction process. In this case, an application has at most n signatures. Most of the prior research on signature-based traffic classification used a single signature per application. However, this may lead to an increase in the false negative ratio and completeness as described in Section 1.2.

Even though we could not label some DCR groups in the functional separation step (grouping ratio in Table V.2), the classification filter extraction was also applied

to the unlabeled groups. In some cases, we could not get any concrete signatures for certain DCR groups so we made simple heuristics to detect the function group based on flow statistics and relationships, and subnet information which generates traffic corresponding to the DCR group as an alternative classifier. For example, BitTorrent changed their protocol to utilize UDP protocol for downloading. For backward comparability, it still supports TCP downloading protocol. The signaling for downloads are sent only once via UDP. If a corresponding peer does not reply to the downloading request, the client connects to the peer again via TCP. However, the download request is not issued again. In this circumstance, we considered TCP connections established within a small time frame right after the UDP download request as BitTorrent's download traffic.

Table V.3 shows the traffic identified by our proposed method. The accuracy is calculated as follows:

$$Byte\ Accuracy = \frac{\sum total\ bytes\ of\ TP\ flows}{total\ bytes\ of\ all\ flows} \tag{V.1}$$

$$Flow\ Accuracy = \frac{\sum number\ of\ TP\ flows}{number\ of\ all\ flows} \tag{V.2}$$

Except MSN (78.87%) and Starcraft (75.05%), byte accuracy of all applications is higher than 91.39% and the highest byte accuracy reaches up to 100%. In the case of flow accuracy, the flow accuracy of every application is lower than byte accuracy similar to the functional separation result. This phenomenon can be explained by the "elephants and mice phenomenon" described in the previous subsection.

Figure V.1 shows the cumulative probability distributions in terms of flow size

Table V.3: Classification results: accuracy of proposed method

Type	Name	Classification Accuracy	
		Byte	Flow
Wired Internet Application	BitTorrent	91.39%	65.78%
	LimeWire	99.78%	89.15%
	Fileguri	99.84%	99.49%
	MSN	78.87%	87.50%
	NateOn	100%	100%
	DropBox	100%	100%
	uCloud	100%	100%
	Gom	99.79%	97.31%
	PotPlayer	99.90%	81.36%
	Starcraft	75.05%	91.84%
	Starcraft2	99.98%	88.23%
Mobile Application	DropBox	99.97%	54.55%
	TVPot	100%	100%
	Bugs	100%	100%
	NateOn	99.90%	60.87%

(in byte) of three different applications. In the case of BitTorrent and LimeWire, flows which have less than 1,000 bytes occupy about 80% of the total flows number. However, their total contribution to the total traffic volume is very low. Figure V.2 supports this fact. This figure illustrates how much of total traffic is generated by larger flows. More than 90% of total traffic is caused by the top 1% of flows. These large flows are usually related to file downloads. Therefore, the byte accuracy is rarely affected even if our method cannot detect some mice flows.

We manually analyzed traffic which was not identified by our method and found some observations on that. Some applications use very well known protocols as part of their protocol operations. An example is the SSDP protocol used by BitTorrent.

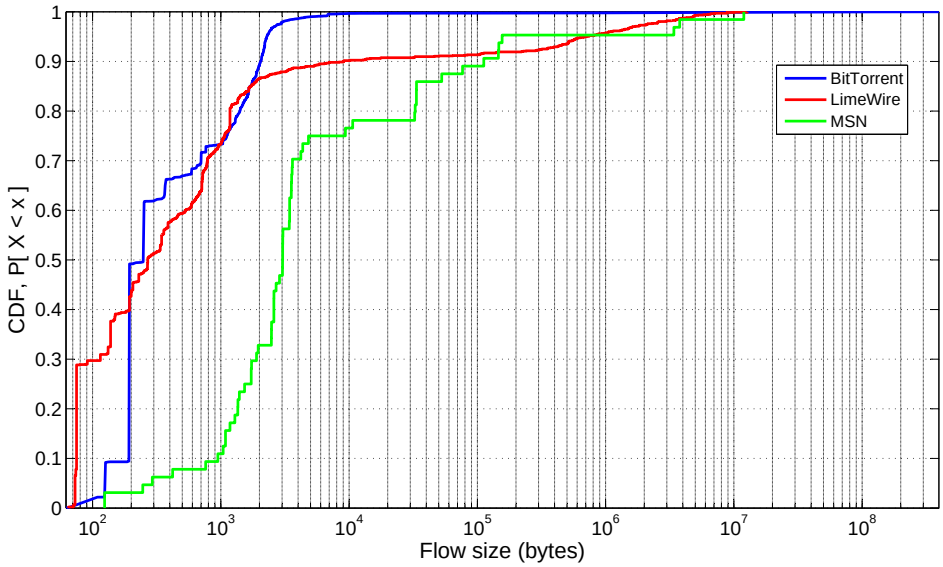
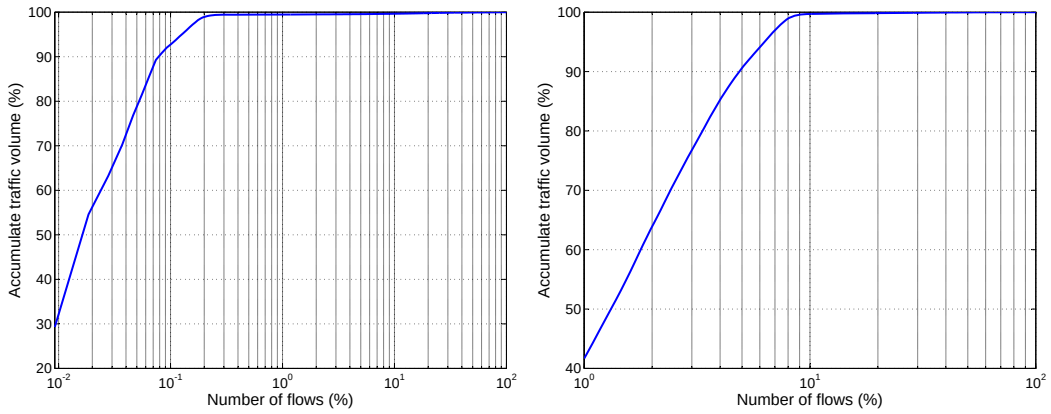


Figure V.1: CDF of flows generated from three different applications

BitTorrent clients use the SSDP protocol for service discovery however SSDP traffic does not have any BitTorrent specific features in their payloads.

Another reason was the flows do not have any payload data which was not included in the DCR group in Table V.2. While this traffic data degrades the flow identification ratio, the byte identification ratio is barely affected due to the small byte size of the flow. The main cause of low byte accuracy of MSN was the SIP protocol used in its voice chat. The flow for voice chat used the SIP protocol and there were no MSN specific contents in its payload. Even though MSN creates one SIP session for voice chat, the size of the flow is relatively bigger than other flows. Therefore, misclassifying a large flow degrades byte accuracy not flow accuracy.



(a) BitTorrent (b) LimeWire

Figure V.2: Contribution of top n% of flows in traffic volume

5.1.4 Comparison with Conventional DPI solutions

We have compared our methodology with two different conventional DPI-based traffic classification methods; L7-filter and OpenDPI.

- **L7-filter** [106] is most widely used for connection-based DPI in Linux. L7-filter uses the GNU Regular Expression engine to match application signatures with application layer data in the packet payloads. The current version supports 113 different application protocols.

- **OpenDPI** [107] is a software library designed to classify Internet traffic. OpenDPI is derived from the commercial PACE product [108]. In addition to signature matching, OpenDPI incorporates connection behavior and statistical analysis. The current version supports 101 different protocols.

Table V.4 shows the traffic identified by our method, L7-filter and OpenDPI. Due to the limited applications supported by L7-filter and OpenDPI, BitTorrent,

Table V.4: Accuracy of proposed method vs. L7-filter and OpenDPI

Application	Identification Ratio (Accuracy)					
	Proposed method		L7-filter		OpenDPI	
	Byte	Flow	Byte	Flow	Byte	Flow
BitTorrent	91.39%	65.78%	16.57%	19.02%	16.44%	47.98%
LimeWire	99.78%	89.15%	97.85%	9.98%	99.73%	46.39%
MSN	78.87%	87.50%	17.64%	4.69%	17.64%	7.81%

LimeWire, and MSN were used in the comparison. The proposed methodology shows the highest identification ratio among three approaches. L7-filter and OpenDPI show poor accuracy (16∼18% and 4∼48% for byte accuracy and flow accuracy respectively). Even though the difference between L7-filter and OpenDPI in terms of accuracy is negligible, OpenDPI shows better performance.

We analyzed classification results of OpenDPI to show the difference of our fine-grained traffic classification scheme. Table V.5 shows the classification results on BitTorrent, LimeWire, and MSN. OpenDPI classified each application traffic into different application protocol groups. Except for unknown traffic, the classification result was correct. In other words, BitTorrent actually utilizes HTTP protocol for discovering content and LimeWire utilizes MDNS protocol for discovering peers. The problem is that there is no means of identifying the origin application of those protocols even though they were generated by BitTorrent or LimeWire clients.

Newer generations of applications tend to adopt multiple application protocols and support various functions (Figure V.3). OpenDPI classifies application traffic only into application protocol layers not higher layers. It causes low classification ratio in application layers. Our fine-grained traffic classification based on functional separation can detects a certain application's traffic regardless of application

Table V.5: Classification results of OpenDPI

Application	Classified Protocol	Proportion (%)	
		Byte	Flow
BitTorrent	unknown	83.54	90.50
	HTTP	0.02	0.60
	BitTorrent	16.44	8.89
LimeWire	unknown	1.38	52.95
	HTTP	0.01	0.51
	MDNS	0.00	0.07
	SSDP	0.00	0.07
	Gnutella	98.60	46.39
MSN	unknown	0.63	6.76
	HTTP	0.45	45.95
	Flash	17.64	6.76
	MSN	17.64	6.76
	STUN	0.02	5.41
	RTP	18.77	1.35
	RDP	59.41	1.35
	SSL	0.78	22.97

protocol. In addition to this, the proposed method classifies application traffic in functional layers. It enriches the information obtained from traffic classification results.

Figure V.4 shows the functional decomposition of traffic. Each color indicates a function in a single application. As mentioned repeatedly, our proposed method can classify different traffic types within a single application according to functionalities. Taking this advantage, we can analyze the usage pattern of an application and also use it as a tool to analyze user behavior and design future applications on the Internet. From a network operator perspective, this can be applied to QoS management for different traffic classes or bandwidth restriction on different traffic classes with in an application.

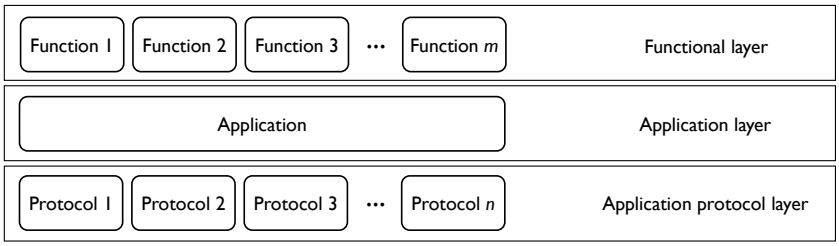


Figure V.3: An application from the perspective of layer

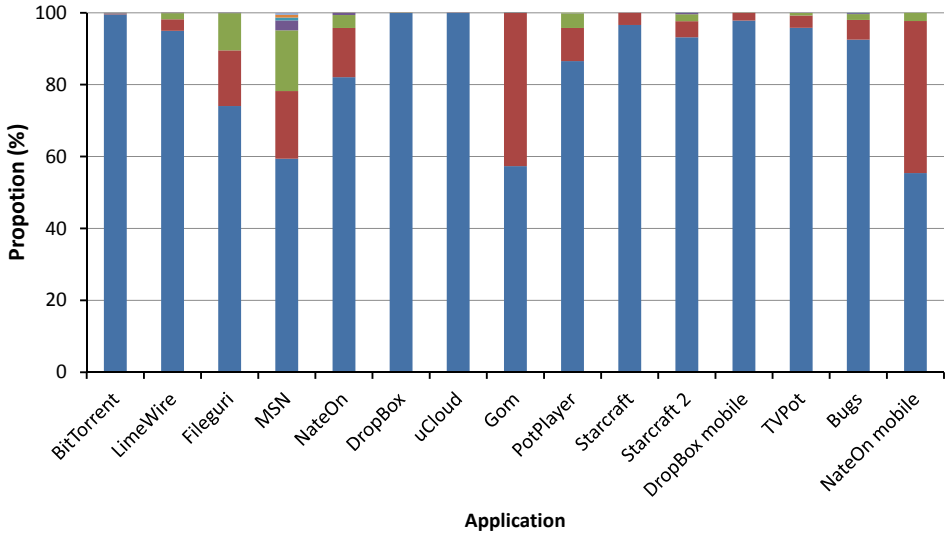


Figure V.4: Traffic composition of each functionality

The comparison shows that the proposed method outperforms the conventional signature based classification methods. As mentioned in Section 1.2, current traffic classification studies have concentrated on classifying main functions (e.g., file transfer in P2P) which generate a great amount of workload in terms of traffic volume. Thus, the conventional signature based classification methods miss a fairly large portion of application traffic data generated by other functions. According to our manual analysis on these results, both L7-filter and OpenDPI can detects only file

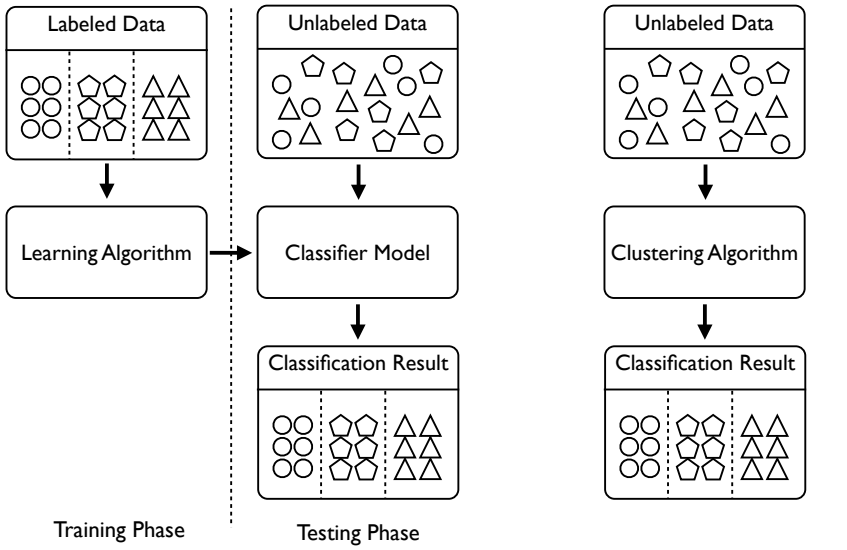
transfers and chat traffic for MSN and some part of signaling traffic for BitTorrent. Although the experiment was limited, we could confirm that the proposed method can increase accuracy and completeness.

5.2 Comparison with Machine Learning Algorithm

In this section, we compare the functional separation and Machine Learning (ML) approaches to show the superiority of functional separation in terms of finding the differences of various traffic characteristics according to functionalities. We applied a clustering algorithm to Fileguri and NateOn which have the most complex functionalities among selected applications. Each application has seven different functions (Table V.2).

As briefly mentioned in Section 2.1.4, ML approaches can be categorized into supervised learning (or classification) and unsupervised learning (or clustering). Figure V.5 explains the difference between supervised learning and unsupervised learning.

Supervised learning requires a training phase to grasp the interrelationship between various features and classes. Training requires a pre-labeled dataset. For example, flows should be labeled with their original application for traffic classification. For this reason, supervised learning is useful for the identification of a particular function of an application. Unsupervised learning, on the other hand, does not require a training phase or pre-labeled dataset. Unsupervised learning automatically discovers the nature of different classes (clusters) in the dataset without any prior guidance.



(a) Supervised learning (b) Unsupervised learning

Figure V.5: Supervised learning vs. Unsupervised learning

Considering the functional separation problem which we need to solve (classifying different traffic generated by different functionalities within an single application), functional separation is quite close to unsupervised learning because prior knowledge on functionalities is not available. In other words, the number of functionalities in different applications is not predefined and it is impossible to characterize different functional traffic groups. Consequently, we compare our functional separation methodology with unsupervised learning algorithms.

Table V.6: Traffic classification research using clustering and their feature set

Title	ML Algorithms	Features	Traffic Composition
Flow clustering using machine learning techniques [18]	Expectation Maximization	Packet length statistics Inter-arrival statistics Byte counts Connection Duration Idle time	HTTP, SMTP, FTP, NTP, IMAP, DNS, etc
Automated traffic classification and application identification using machine learning [71]	AutoClass (Bayesian clustering)	Packet length statistics Inter-arrival statistics Flow size (bytes) Flow duration	Half-Life, Napster, AOL, HTTP, DNS, SMTP, Telnet, FTP
Traffic classification on the fly [33]	SimpleKMeans	Packet length of the first few packets	eDonkey, FTP, HTTP, KaZaA, NTP, POP3, SMTP, SSH, HTTPS, POP3
Identifying and discriminating between web and peer-to-peer traffic in the network core [44]	K-Means	Total number of packets Mean packet length Mean payload length Flow duration Flow size (bytes) Mean inter-arrival time	Web, P2P, FTP, Others
Traffic classification using clustering algorithms [21]	K-Means DBSCAN AutoClass	Total number of packets Mean packet length Mean payload length Flow size (bytes) Mean inter-arrival time	HTTP, P2P, SMTP, IMAP, POP3, MSSQL, Others

* statistics includes mean/min/max/sd

5.2.1 Feature Selection

Candidate Features

For the feature selection, we have analyzed previous research on traffic classification using unsupervised learning. Table V.6 describes a selection of traffic classification research using clustering algorithms and their feature set. We have selected candidate features that were commonly used in previous works. The feature set includes various flow statistics such as packet length, packet inter-arrival time, flow duration, etc. Bernaille *et al.* [32] have used packet length of the first few packets as their feature set. The results show that more than 80% of total flows are correctly identified and the highest accuracy reached up to 99%. Although this feature set is not universal, we have included it on the candidate features due to the high classification ratio. The candidate features are listed in Table V.7.

Table V.7: List of candidate features

Index	Feature	Index	Feature
1	Protocol	10	Maximum packet inter-arrival time
2	Number of packets	11	Average payload size
3	Flow size in bytes	12	Minimum payload size
4	Flow duration	13	Maximum payload size
5	Average packet size	14	Size of 1st packet in the flow
6	Minimum packet size	15	Size of 2nd packet in the flow
7	Maximum packet size	16	Size of 3rd packet in the flow
8	Average packet inter-arrival time	17	Size of 4th packet in the flow
9	Minimum packet inter-arrival time	18	Size of 5th packet in the flow

Feature Selection Algorithm

Classification accuracy of ML-based traffic classification techniques is directly affected by its features and the effectiveness of a feature set varies depending on specific classification problems. Therefore, it is very crucial to find an appropriate feature set for a given problem.

Feature selection is a very crucial process for selecting a subset of relevant features for building robust learning models [109, 110]. There has been much research on feature selection [26, 27], and finding an effective feature selection for traffic classification helps enhance the accuracy of the classifiers.

We have measured the impacts of each candidate feature using a feature selection algorithm to select the final feature set. Among various feature selection algorithms, we have used the Relief algorithm to finalize the feature set for functional separation.

Relief algorithm Relief is an instance based feature ranking algorithm introduced by Kira *et al.* [111] and improved by Kononenko [112]. Relief algorithm has been extensively researched and it is well known as the most successful feature selection methods for classification [113]. The Relief family of algorithms identifies the importance of features based on the distance of nearest hits and nearest misses. Nearest hits denote data points within the same class and nearest misses refer to data points from different classes. Relief algorithms are known to be efficient for high dimensional data. They work well even in circumstances when there is no linear relationship among features.

Algorithm 8: Relief algorithm for identifying discriminating features

procedure Relief
input : m training vectors of n attributes and the class label for each vector
output: the weigh vector $\vec{w}$ ($= < w_1 \cdots w_n >$)

```
1  begin
2      for 1 ≤ i ≤ n do
3          w_i ← 0
4      end
5      foreach data point x⃗ do
6          NH ← nearest hit
7          NM ← nearest miss
8          for 1 ≤ j ≤ n do
9              w_j = w_j + dist(x^(j), NM^(j)(x⃗)) - dist(x^(j), NH^(j)(x⃗))
10         end
11     end
12 end
```

The Relief weighs can be calculated by following formula:

$$w_i = w_i + dist(x^{(i)}, NM^{(i)}(\vec{x})) - dist(x^{(i)}, NH^{(i)}(\vec{x})) \tag{V.3}$$

$x^{(i)}$ denotes the i^{th} feature of a data point $\vec{x}$. $NM^{(i)}(\vec{x})$ and $NH^{(i)}(\vec{x})$ indicate i^{th} feature of nearest hit and nearest miss respectively. Algorithm 8 shows the Relief algorithm. The function *dist* computes the difference between the values of feature for two instances. For discrete attributes, the difference is either 1 (the values are different) or 0 (the values are the same), while for continuous attributes the difference is the actual difference normalized to the interval [0,1].

Selected Feature Set

We selected the final feature set by applying the Relief algorithm to candidate features of two different applications which belong to different application types. The results are described in Table V.8.

Table V.8: Weights of features calculated by Relief algorithm

	Fileguri	NateOn
Protocol	0.000+0.000	0.351+0.004
Number of packets	0.085+0.007	0.015+0.003
Flow size in bytes	0.050+0.009	0.003+0.003
Flow duration	0.242+0.031	0.115+0.013
Average packet size	0.152+0.011	0.018+0.006
Minimum packet size	0.193+0.010	0.253+0.007
Maximum packet size	0.305+0.008	0.030+0.010
Average packet inter-arrival time	0.129+0.019	0.027+0.015
Minimum packet inter-arrival time	0.085+0.027	0.042+0.009
Maximum packet inter-arrival time	0.180+0.023	0.038+0.015
Average payload size	0.152+0.010	0.018+0.006
Minimum payload size	0.000+0.000	0.351+0.004
Maximum payload size	0.304+0.008	0.029+0.010
Size of 1st packet in the flow	0.128+0.028	0.248+0.011
Size of 2nd packet in the flow	0.086+0.011	0.003+0.001
Size of 3rd packet in the flow	0.332+0.016	0.108+0.012
Size of 4th packet in the flow	0.301+0.018	0.242+0.018
Size of 5th packet in the flow	0.317+0.019	0.248+0.017

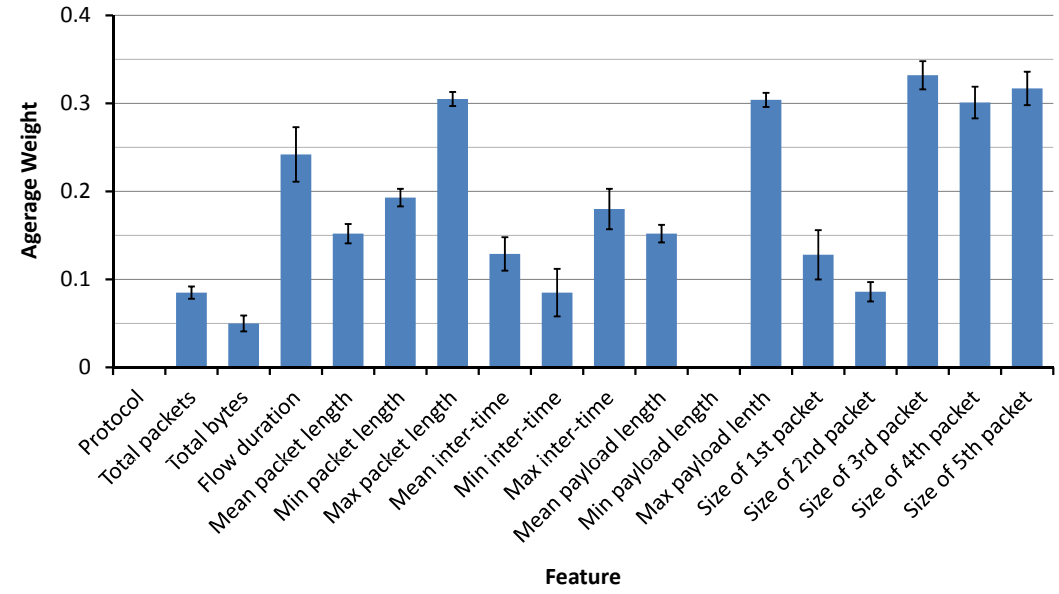
Weights of each feature fluctuate significantly from application to application. For example, protocol has the lowest weight (0) when identifying functionalities of Fileguri traffic while it has the highest weights (0.351) when identifying functionalities of NateOn. Figure V.6 shows the different feature weights of two different

applications. Every feature besides the protocol has different importance for different applications. Therefore, it is difficult to select features which are commonly efficient for every dataset (application traffic).

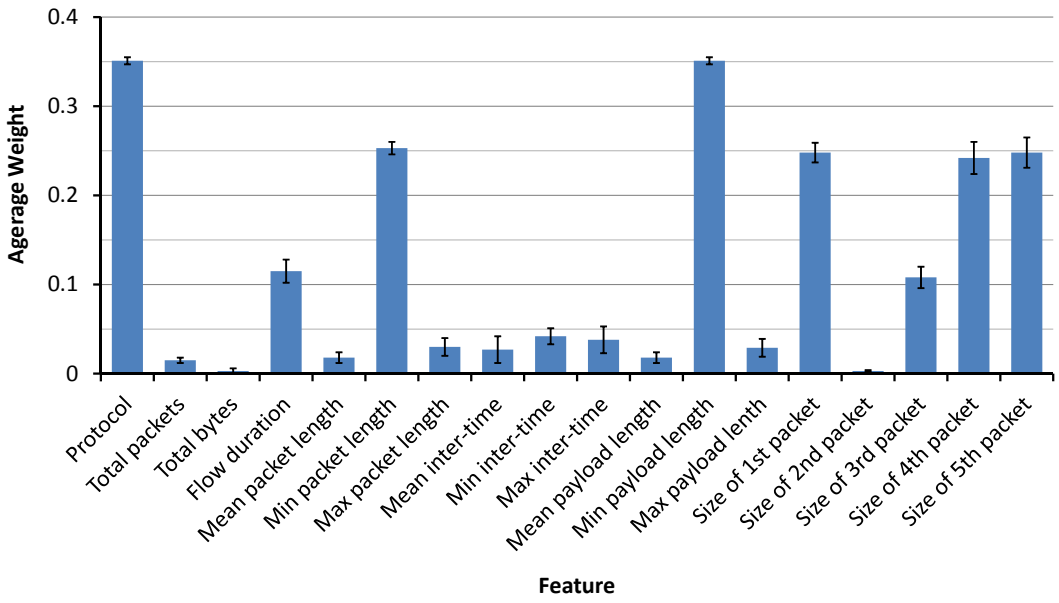
We have selected a different feature set for each application to resolve this problem. We have removed features with a weight value of less than 0.1 (Table V.9).

Table V.9: Final feature set of each application

	Fileguri	NateOn
Protocol		o
Number of packets		
Flow size in bytes		
Flow duration	o	o
Average packet size	o	o
Minimum packet size	o	o
Maximum packet size	o	
Average packet inter-arrival time	o	
Minimum packet inter-arrival time	o	
Maximum packet inter-arrival time	o	
Average payload size	o	
Minimum payload size		o
Maximum payload size	o	
Size of 1st packet in the flow	o	o
Size of 2nd packet in the flow		
Size of 3rd packet in the flow	o	o
Size of 4th packet in the flow	o	o
Size of 5th packet in the flow	o	o



(a) Fileguri



(b) NateOn

Figure V.6: Average weight of each feature

5.2.2 Clustering Algorithms

Previous traffic classification works based on clustering algorithms have utilized five different clustering algorithms; Expectation Maximization (EM), AutoClass, SimpleKMeans, K-Means, and DBSCAN (Table V.6). The DBSCAN algorithm was used in clustering for functional separation among these algorithms due to the characteristics of the clustering algorithms and the functional separation problem. Other clustering algorithms require the number of clusters to be clustered as an input parameter. However, the functional separation starts without any information on the number of clusters (functionalities in an application). Therefore, we compared our method to the DBSCAN which automatically determine the number of clusters.

DBSCAN Algorithm DBSCAN algorithm [114] is a density-based clustering algorithm. This algorithm finds a number of clusters starting from the estimated density distribution of corresponding nodes. DBSCAN is known as one of the most common clustering algorithms. In DBSCAN, an object p is directly density-reachable from an object q if both objects are located within a given distance ϵ. If p is surrounded by sufficiently many points objects which are closer than ϵ in terms of distance, p and those objects are considered as a cluster. An object p is density-reachable from q if the object p is within the ϵ-neighborhood of an object r which is directly density-reachable or density-reachable from q. As mentioned previously, the main advantage of the DBSCAN algorithms is that this clustering algorithm does not require a specific number of clusters in the data. It makes DBSCAN suitable for clustering application traffic when the number of traffic type of is unknown.

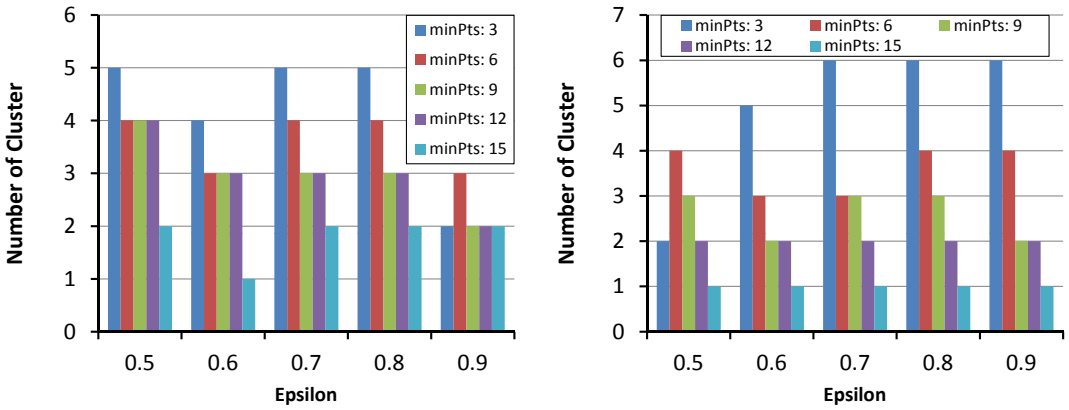
5.2.3 Clustering Results

DBSCAN has two input parameters; ϵ and *minPts*. Each parameter denotes the radius of the ϵ distance and minimum number of data object required in an ϵ distance range respectively. We varied these parameters and the best combination of (ϵ, *minPts*). The values for minPts were tested between 3 and 15. The ϵ distance was tested from 0.1 to 0.9.

Before analyzing traditional accuracy metrics such as *precision* and *recall*, we first examined the number of clusters. Figure V.7 shows the results of the DBSCAN in terms of the number of output clusters. The numbers of labeled classes in input data was seven for both applications. Because multiple clusters can be mapped to a functional group, some cases that the number of clusters is more than the original classes are acceptable considering the characteristic of the functional separation problem. The DBSCAN algorithm determines the number of clusters automatically; however it does not work well on discriminating different types of traffic within a single application. The number of output clusters is less than the original number of classes in every execution with different parameters. These results imply that the clustering algorithm cannot detect the clear boundaries of different traffic characteristics using statistical features.

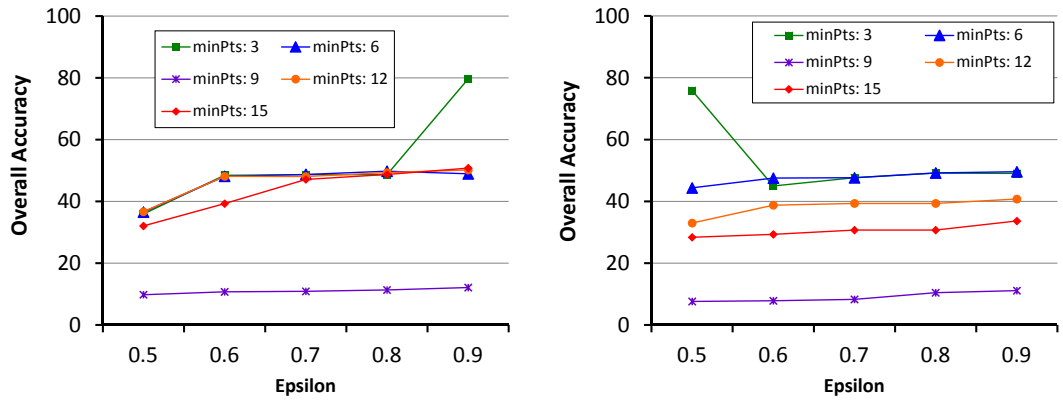
In the case of NateOn, we could get close values to the original number of classes by adjusting the parameters. However, the overall accuracy was not acceptable.

Figure V.8 shows the overall accuracy (unclustered data instances were considered as mis-clustered) of clustering. The highest accuracy was about 80% in both applications. The cause of such low accuracies was the unclustered data instances



(a) Fileguri (b) NateOn

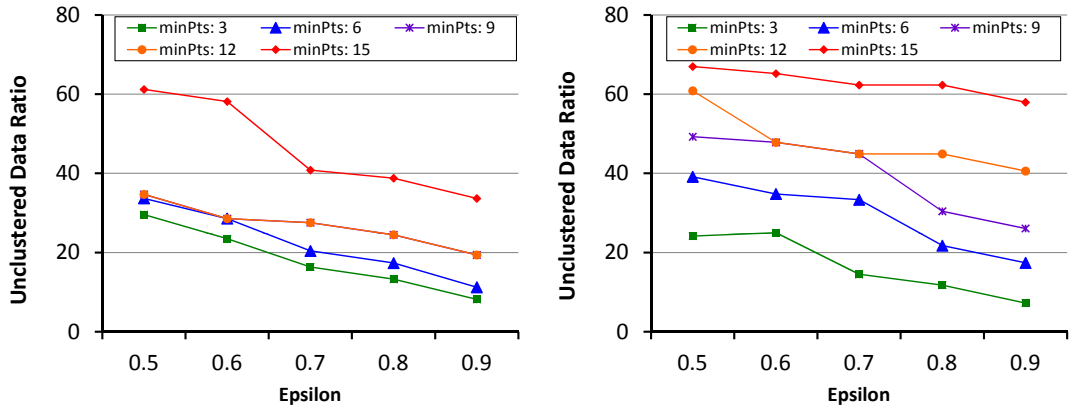
Figure V.7: Number of clusters using DBSCAN



(a) Fileguri (b) NateOn

Figure V.8: Clustering accuracy of clustered data instances

described in Figure V.9. The DBSCAN algorithm can label noise data. This is one of the strengths of the DBSCAN algorithm when applied to general clustering problems. However, these noise data instances can be considered as mis-clustered data in functional separation. In the worst case, the ratio of the unclustered data



(a) Fileguri (b) NateOn

Figure V.9: Unclustered data ratio

instances reached up to 61.22% and 67.21% for each application. It means that the accuracy of clustering results is less than 40%.

We need to maximize the number of clusters close to the real number of classes and minimize the mis-clustered ratio to apply clustering algorithms to functional separation problem. However, we could not get any proper results. Experimental results of this section show that the traditional clustering algorithm is not suitable for identifying different traffic characteristics of a single application.

5.3 Use Cases of Fine-grained Traffic Classification

This section provides some use cases of fine-grained traffic classification. The fine-grained traffic classification scheme can be utilized for user behavior analysis and other analyses which are unable to be achieved by current traffic classification schemes. Although use cases provided in this section are very simple and limited,

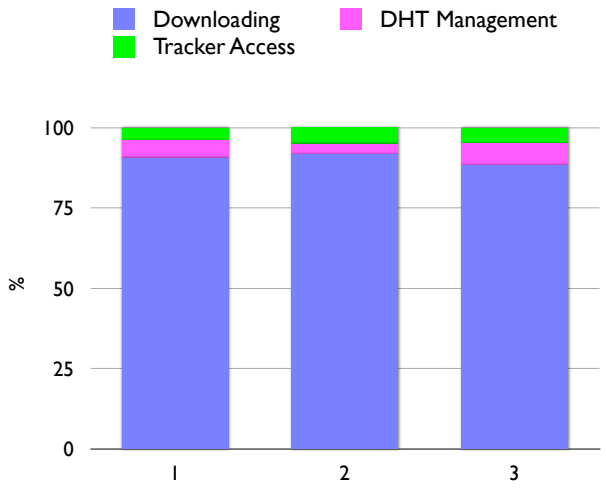


Figure V.10: BitTorrent's workloads according to functionality

we believe that the fine-grained traffic classification has a much wider application.

User behavior analysis As an example of user behavior analysis, we can analyze the average search counts when a user initializes downloading using P2P applications. The ratio of searching to downloading in terms of transaction number was 56,392:1. We have empirically confirmed that the Fileguri client generated about 6,000 TCP transactions in a single keyword search. Thus, we conclude that a Fileguri user performs about 9.398 searches on average before downloading from the P2P network. The goal of this simple analysis is to provide the average searching counts of users.

Workload analysis according to functions Most of wired Internet access applies flat rate billing. However, Internet access for smartphones applies usage-based billing. According to a report, there are more than 200,000 Android and about

300,000 iPhone applications available as of March 2011 and they are increasing continuously everyday [115]. In this situation, the amount of traffic generated by an application is a crucial issue from the perspective of billing. As an example, we have analyzed BitTorrent's workloads when a user is downloading files. Figure V.10 illustrates BitTorrent's workloads according its functionalities. About 10 to 15% of traffic was not generated by actual downloading but by other functionalities. Although such traffic is essential for downloading, it is undesired by users. If we analyze different applications' workloads and compare them, we can provide the proportion of unwanted traffic of each application to users and this information is useful for users to select an application considering their data plan or billing scheme.

Chapter VI

CONCLUSIONS AND FUTURE WORK

This chapter summarizes the overall contents of the thesis and lists a set of contributions this thesis achieved. Furthermore, suggested areas for future work are also discussed.

6.1 Summary

The most critical and representative issue that exists in today's Internet traffic classification is achieving a high-level of accuracy and completeness. Accuracy in this context is how correctly the fraction of traffic is classified according to its original applications. There are many metrics such as False Positive (FP), False Negative (FN), precision, and recall that can be used to evaluate the accuracy of classification methodologies or systems. Completeness, on the other hand, is the percentage of

the traffic classified by a certain classification methodology. Completeness of traffic classification is mainly related to unknown traffic patterns and has correlation with the FN ratio. It is highly desirable to maximize both accuracy and completeness during traffic classification. However, it is extremely difficult for any method to claim 100% of accuracy and completeness due to the fast-changing nature of Internet traffic. There is still a lot of room for improvement in traffic classification

Another issue in traffic classification is the analysis of traffic classification results. Many previous studies have suggested various classification methodologies (e.g., well-known port number matching, payload contents analysis, machine learning, etc.), from which even more variants have been derived. However, it is extremely difficult for any method to claim 100% accuracy due to the fast-changing and dynamic nature of Internet traffic. The classification accuracy is also questionable since there is often no ground truth dataset available. In another respect, all research aims at different levels of classification. Some only had a coarse classification goal such as classifying traffic protocol or application type; while others had more detailed classification goal such as identifying the exact application name. Therefore, it is often unfair to cross-compare each classification method in terms of accuracy. To overcome this issue, it is more important to investigate how we can provide more meaningful information with such limited traffic classification results rather than to struggling to overimprove 1 or 2% of classification accuracy.

In regards to both problems, accuracy and completeness problem and lack of information problem, we have defined a new traffic classification scheme called a fine-grained traffic classification based on the analysis of existing classification methodologies. This scheme allows classifying traffic according to the functionalities in an

application. The main idea of this fine-grained traffic classification comes from the fact that there exist some differences among flows belonging to a certain application according to their functionalities. The key to the fine-grained traffic classification is discriminating among network flows based on their original functions in an application. To resolve this issue, we have developed a method called the functional separation method, which classifies flows into several functional traffic groups.

The first step of functional separation, Port-Relation Grouping (PRG), identifies the dependency on allocated port numbers and aggregates flows according to the dependency. Then follows Contents-Relation Grouping (CRG) that measures the content similarity among flows and assorts them again based on the results of their content similarity. Finally, Contents-Relation Decomposition (CRD) divides the flow groups composed in the previous steps. This decomposition step is essential to eliminate any uncertainty caused by misclassified flow groups and discover more detailed functions.

To validate the proposed methods, we have run tests by actually classifying traffic from various types of real applications using fine-grained traffic classification filters. We have also compared our algorithms with existing DPI-based classification frameworks and a clustering algorithm to prove the higher accuracy and efficiency acquired by the proposed methods. Our method have shown better performance than other traffic classification approaches in terms of classification accuracy. In comparison with the DBSCAN clustering algorithm, we have shown that the clustering algorithm is not suitable for the functional separation problem.

Our unique traffic scheme can increase traffic classification accuracy and completeness by reducing the amount of undetected traffic and provide more in-depth

classification results for various analyses which are unable to be achieved by current traffic classification schemes and methods. The key to the fine-grained traffic classification is classifying network flows into different functional groups based on origin functions in an application. To achieve this, we also proposed the functional separation method. By applying this method we are able to detect different types of traffic generated by a single application according to their functionalities. The fine-grained traffic classification based on functional separation will potentially increase the amount of information that can be obtained by traffic classification and lay a cornerstone in the foundation of applying traffic classification in user-behavior analysis.

6.2 Contributions

The followings are key contributions that are expected from this thesis.

First, we provided an overview of the existing traffic classification approaches and comprehensive survey of traffic classification studies from the perspective of classification approaches and classification level. Although Internet traffic classification has became one of the major research areas in network management fields, there was no concrete taxonomy of traffic classification. We have categorized existing studies into different traffic classification levels. This survey is the first attempt to developing a taxonomy of traffic classification based on classification levels.

Second, we proposed a new traffic classification scheme. The absence of an advanced classification scheme leads to the development of a new classification scheme which can classify various types of traffic generated by a single application. The pro-

posed traffic classification scheme and method for functional separation in a single application improve traffic classification accuracy and completeness.

Third, our proposed traffic classification scheme can provide more in-depth classification results for analyzing user contexts. The fine-grained traffic classification enriches the amount of information which can be obtained by traffic classification. For the evaluation, we have provided use cases of the fine-grained traffic classification in terms of user behavior analysis and other analyses which are unable to be achieved by current traffic classification schemes and methods.

Finally, we validated the proposed methodologies by applying them to real-world application traffic traces. We also compared the proposed method with conventional traffic classification DPI-based traffic classification methods. In addition to this, we showed that it is difficult to resolve the functional separation problem with traditional clustering algorithms even though the characteristics of functional separation are similar to clustering problems.

6.3 Future Work

Improving accuracy and complete in traffic classification and applying traffic classification results into actual network management purposes are critical research challenges on current Internet traffic classification areas. In this thesis, we handled these two problems and proposed solutions. The following are a list of future work.

Our functional separation method has a limitation. In our method, we discriminated different functionalities among traffic flows generated by a single application. Each discriminated traffic group is labeled with its functionality. The labeling pro-

cess is achieved by manual inspections. We consider enhancing the labeling process of the functional separation method. It will be impossible to identify the purposes of network flows automatically; it has a lot of room for improvement.

Traffic classification filters applied in this thesis are based on payload signatures which require deep packet inspection. We plan to analyze the flexibility of our approach by applying different classification methodologies such as statistical approaches instead of payload signatures.

In this thesis, we have focused on developing a method to increase traffic classification accuracy and completeness. The validation result shows that the proposed method outperforms other traffic classification methods. A large collection of knowledge base for traffic classification is crucial in order to building an accurate traffic classification system. We plan to increase the number of applications and share the classification filter database within the measurement research community in order to verify further on the classification accuracy and completeness.

The number of mobile devices and the amount of mobile traffic are increasing rapidly. Network traffic generated by mobile devices is increasing at this very moment along with the number of smartphone users and mobile services (or applications). Characteristics of mobile traffic are known to be different from normal wired traffic. Mobile traffic has smaller size in terms of flow duration and total bytes. Furthermore, HTTP protocol is the most used application layer protocol in mobile traffic. Considering these characteristics, we plan to develop a lightweight functional separation algorithm for mobile traffic.

Finally, we also plan to conduct further research on user behavior analysis based on fine-grained traffic classification. User behavior has gained considerable atten-

tions recently. Traffic classification can be utilized by service providers to comprehend the behavior of their customers so they can utilize the information to provide personalized services to increase their customers' satisfaction.

REFERENCES

[1] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proceeding of Passive and Active Network Measurement (PAM)*, (Boston, MA, USA), pp. 41–54, March 2005.

[2] IANA, "Iana port numbers list." `http://www.iana.org/assignments/port-numbers`. [Online; accessed 08-October-2011].

[3] A. Madhukar and C. Williamson, "A longitudinal study of p2p traffic classification," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on*, pp. 179 – 188, sept. 2006.

[4] P. Borgnat, G. Dewael, K. Fukuda, P. Abry, and K. Cho, "Seven years and one day: Sketching the evolution of internet traffic," in *Proceeding of IEEE INFOCOM 2009*, (Rio de Janeiro, Brazil), pp. 711–719, April 2009.

[5] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of p2p traffic," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC)*, (Taormina, Sicily, Italy), pp. 121–134, October 2004.

[6] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proceedings of the 13th international conference on World Wide Web (WWW)*, (New York, NY, USA), pp. 512–521, May 2004.

[7] B.-C. Park, Y. Won, M.-S. Kim, and J. Hong, "Towards automated application signature generation for traffic identification," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pp. 160 –167, april 2008.

[8] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "Acas: automated construction of application signatures," in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, MineNet '05, (New York, NY, USA), pp. 197–202, ACM, 2005.

[9] H.-A. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection," in *roceedings of the 13th Usenix Security Symposium*, (San Diego, CA, UASA), pp. 271–286, August 2004.

[10] S. Ehlert, S. Petgang, and T. Magedanz, "Analysis and signature of skype voip session traffic," tech. rep., Franunhofer FOKUS, NGNISKYPE-06b, Berlin, Germany, July, 2006.

[11] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *Proceedings of the ACM SIGCOMM 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, (Philadelphia, PA, USA), pp. 229–240, August 2005.

[12] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network monitoring using traffic dispersion graphs," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC)*, (San Diego, CA, USA), pp. 315–320, October 2007.

[13] S. S. Kim and A. Reddy, "Image-based anomaly detection technique: Algorithm, implementation and effectiveness," *Selected Areas in Communications, IEEE Journal on*, vol. 24, pp. 1942 –1954, oct. 2006.

[14] M. sup Kim, Y. J. Won, and J. W. ki Hong, "Application-level traffic monitoring and an analysis on ip networks," *ETRI Journal*, vol. 27, pp. 22–42, 2005.

[15] M. Iliofotou, "Exploring graph-based network traffic monitoring," in *Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, INFOCOM'09, (Piscataway, NJ, USA), pp. 353–354, IEEE Press, 2009.

[16] J. Frank, "Machine learning and intrusion detection: Current and future directions," in *the National 17th Computer Security Conference*, (Washington,D.C), October 1994.

[17] A. Mcgregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *Proceedings of the 5th international workshop Passive and Active Network Measurement*, pp. 205–214, 2004.

[18] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '05, (New York, NY, USA), pp. 50–60, ACM, 2005.

[19] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, (New York, NY, USA), pp. 135–148, ACM, 2004.

[20] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the ACM SIGCOMM Workshop on Mining Network Data (MineNet)*, (Barcelona, Spain), pp. 281–286, August 2006.

[21] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Transactions on Neural Networks*, vol. 18(1), pp. 223–239, 2007.

[22] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM SIGCOMM Computer Communication Review*, vol. 37(1), pp. 5–16, 2007.

[23] N. Williams, S. Z, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *SIGCOMM Computer Communication Review*, vol. 30, pp. 7–15, 2006.

[24] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, (Madrid, Spain), pp. 1–12, December 2008.

[25] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys Tutorials, IEEE*, vol. 10, pp. 56 –76, quarter 2008.

[26] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[27] M. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 15, pp. 1437 – 1447, nov.-dec. 2003.

[28] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Semi-supervised network traffic classification," in *Proceedings of the 2007 ACM*

SIGMETRICS international conference on Measurement and modeling of computer systems, SIGMETRICS '07, (New York, NY, USA), pp. 369–370, ACM, 2007.

[29] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06, (New York, NY, USA), pp. 6:1–6:12, ACM, 2006.

[30] Y.-s. Lim, H.-c. Kim, J. Jeong, C.-k. Kim, T. T. Kwon, and Y. Choi, "Internet traffic classification demystified: on the sources of the discriminative power," in *Proceedings of the 6th International COnference*, Co-NEXT '10, (New York, NY, USA), pp. 9:1–9:12, ACM, 2010.

[31] A. Dainotti, W. de Donato, A. Pescape, and P. Salvo Rossi, "Classification of network traffic via packet-level hidden markov models," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pp. 1 –5, 30 2008-dec. 4 2008.

[32] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 23–26, April 2006.

[33] K. Thompson, G. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *Network, IEEE*, vol. 11, pp. 10 –23, nov/dec 1997.

[34] D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy, "The coralreef software suite as a tool for system and network administrators," in *Proceedings of the 15th USENIX conference on System administration*, (San Diego, California), pp. 133–144, December 2001.

[35] K. Claffy, "Internet traffic characterization," *Ph.D. Thesis*, 1994.

[36] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, (New York, NY, USA), pp. 313–326, ACM, 2006.

[37] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "Observing slow crustal movement in residential user traffic," in *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, (New York, NY, USA), pp. 12:1–12:12, ACM, 2008.

[38] A. Soule, K. Salamatia, N. Taft, R. Emilion, and K. Papagiannaki, "Flow classification by histograms: or how to go on safari in the internet," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, pp. 49–60, June 2004.

[39] K.-c. Lan and J. Heidemann, "A measurement study of correlations of internet flow characteristics," *Comput. Netw.*, vol. 50, pp. 46–62, January 2006.

[40] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, G. Varghese, and H. Kim, "Graption: Automated detection of p2p applications using traffic dispersion graphs," tech. rep., UC Riverside, UCR-CS-2008-06080, June, 2008.

[41] D. Zuev and A. W. Moore, "Traffic classification using a statistical approach," *Passive and Active Network Measurement*, vol. 3431, no. 3, pp. 321–324, 2005.

[42] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Perform. Eval.*, vol. 64, pp. 1194–1213, October 2007.

[43] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core," in *Proceedings of the 16th international conference on World Wide Web*, WWW '07, (New York, NY, USA), pp. 883–892, ACM, 2007.

[44] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network traffic analysis using traffic dispersion graphs (tdgs): Techniques and hardware implementation," tech. rep., UC Riverside, UCR-CS-2007-05001, 2007.

[45] W. John and S. Tafvelin, "Heuristics to classify internet backbone traffic based on connection patterns," in *Information Networking, 2008. ICOIN 2008. International Conference on*, pp. 1 –5, jan. 2008.

[46] W. John, S. Tafvelin, and T. Olovsson, "Trends and differences in connection-behavior within classes of internet backbone traffic," in *Proceedings of the 9th international conference on Passive and active network measurement*, PAM'08, (Berlin, Heidelberg), pp. 192–201, Springer-Verlag, 2008.

[47] M. Pietrzyk, J.-L. Costeux, G. Urvoy-Keller, and T. En-Najjary, "Challenging statistical classification for operational usage: the adsl case," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, (New York, NY, USA), pp. 122–135, ACM, 2009.

[48] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is p2p dying or just hiding? [p2p traffic measurement]," in *Proceeding of IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 3, (Dallas, Texa, USA), pp. 1532–1538, November 2004.

[49] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Inherent behaviors for on-line detection of peer-to-peer file sharing," in *IEEE Global Internet Symposium, 2007*, pp. 55 –60, may 2007.

[50] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing skype traffic: when randomness plays with you," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 37–48, August 2007.

[51] F. Constantinou and P. Mavrommatis, "Identifying known and unknown peer-to-peer traffic," in *Network Computing and Applications, 2006. NCA 2006. Fifth IEEE International Symposium on*, pp. 93 –102, july 2006.

[52] L. Plissonneau, J.-L. Costeux, and P. Brown, "Analysis of peer-to-peer traffic on adsl.," in *Passive and Active Measurement Workshop*, (Boston, MA, USA), pp. 69–82, March 2005.

[53] G. Bartlett, J. Heidemann, C. Papadopoulos, and J. Pepin, "Estimating p2p traffic volume at usc," tech. rep., USC, ISI-TR-645, 2007.

[54] N. Duffield and C. Lund, "Predicting resource usage and estimation accuracy in an ip flow measurement collection infrastructure," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, (New York, NY, USA), pp. 179–191, ACM, 2003.

[55] K. Suh, D. R. Figueiredo, J. Kurose, and D. Towsley, "Characterizing and detecting skype-relayed traffic," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, (Barcelona, Spain, April), pp. 1 –12, april 2006.

[56] G. Szabo, I. Szabo, and D. Orincsay, "Accurate traffic classification," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pp. 1 –8, june 2007.

[57] L. Bernaille, A. Soule, I. Akodjenou, and K. Salamatian, "Blind application recognition through behavioral classification," tech. rep., LIP6, 2005.

[58] L. Bernaille and R. Teixeira, "Early recognition of encrypted applications," in *Proceedings of the 8th international conference on Passive and active network measurement*, PAM'07, (Berlin, Heidelberg), pp. 165–175, Springer-Verlag, 2007.

[59] D. Antoniades, M. Polychronakis, S. Antonatos, E. P. Markatos, S. Ubik, and A. Oslebo, "Appmon: An application for accurate per application network traffic characterisation," in *IST Broadband Europe Conference*, 2006.

[60] S. Zander, T. T. T. Nguyen, and G. J. Armitage, "Self-learning ip traffic classification based on statistical flow characteristics.," in *PAM* (C. Dovrolis, ed.), vol. 3431 of *Lecture Notes in Computer Science*, pp. 325–328, Springer, 2005.

[61] M. Dusi, F. Gringoli, and L. Salgarelli, "A preliminary look at the privacy of ssh tunnels," in *Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on*, pp. 1 –7, aug. 2008.

[62] M. Perenyi, T. Dang, A. Gefferth, and S. Monlnar, "Identification and analysis of peer-to-peer traffic," *Journal of Communications*, vol. 1, pp. 36–46, 2006.

[63] J. Erman, A. Mahanti, and M. Arlitt, "Qrp05-4: Internet traffic identification using machine learning," in *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pp. 1 –6, 27 2006-dec. 1 2006.

[64] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, (New York, NY, USA), pp. 90–102, ACM, 2009.

[65] N. Azzouna and F. Guillemin, "Analysis of adsl traffic on an ip backbone link," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, vol. 7, pp. 3742 – 3746 vol.7, dec. 2003.

[66] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "The impact and implications of the growth in residential user-to-user traffic," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, (New York, NY, USA), pp. 207–218, ACM, 2006.

[67] H. T. Marques, Nt., L. C. D. Rocha, P. H. C. Guerra, J. M. Almeida, W. Meira, Jr., and V. A. F. Almeida, "Characterizing broadband user behavior," in *Proceedings of the 2004 ACM workshop on Next-generation residential broadband challenges*, NRBC '04, (New York, NY, USA), pp. 11–18, ACM, 2004.

[68] A. Balachandran, G. M. Voelker, P. Bahl, and P. V. Rangan, "Characterizing user behavior and network performance in a public wireless lan," in *Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '02, (New York, NY, USA), pp. 195–205, ACM, 2002.

[69] M.-S. Kim, Y. J. Won, and J. W. Hong, "Characteristic analysis of internet traffic from the perspective of flows," *Computer Communications*, vol. 29, no. 10, pp. 1639–1652, 2006. Monitoring and Measurements of IP Networks.

[70] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pp. 250 –257, nov. 2005.

[71] M. Tai, S. Ata, and I. Oka, "Fast, accurate, and lightweight real-time traffic identification method based on flow statistics," in *Proceedings of the 8th international conference on Passive and active network measurement*, PAM'07, (Berlin, Heidelberg), pp. 255–259, Springer-Verlag, 2007.

[72] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting," *Comput. Netw.*, vol. 53, pp. 81–97, January 2009.

[73] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of internet chat systems," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, (New York, NY, USA), pp. 51–64, ACM, 2003.

[74] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*, LISA '99, (Berkeley, CA, USA), pp. 229–238, USENIX Association, 1999.

[75] M. P. Collins and M. K. Reiter, "Hit-list worm detection and bot identification in large networks using protocol graphs," in *Proceedings of the 10th international conference on Recent advances in intrusion detection*, RAID'07, (Berlin, Heidelberg), pp. 276–295, Springer-Verlag, 2007.

[76] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '05, (New York, NY, USA), pp. 217–228, ACM, 2005.

[77] W. John and S. Tafvelin, "Analysis of internet backbone traffic and header anomalies observed," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, (New York, NY, USA), pp. 111–116, ACM, 2007.

[78] Z. Li, M. Sanghi, Y. Chen, M.-Y. Kao, and B. Chavez, "Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience," in *Security and Privacy, 2006 IEEE Symposium on*, pp. 15 pp. –47, may 2006.

[79] E. P. Markatos, S. Antonatos, M. Polychronakis, and K. G. Anagnostakis, "Exclusion-based signature matching for intrusion detection," in *In Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN*, (Cambridge, MA, USA), pp. 146–152, ACTA Press, 2002.

[80] P. Chhabra, A. John, and H. Saran, "Pisa: Automatic extraction of traffic signatures.," in *International Conference in Networking*, (Waterloo, Canada), pp. 730–742, May 2005.

[81] C.-T. Huang, S. Thareja, and Y.-J. Shin, "Wavelet-based real time detection of network traffic anomalies," in *Securecomm and Workshops, 2006*, pp. 1 –7, 28 2006-sept. 1 2006.

[82] N. Brownlee, C. Mills, and G. Ruth, "Traffic flow measurement: Architecture." IETF RFC 2722, October 1999.

[83] N. Brownlee, "Traffic flow measurement: Experiences with netramet." IETF RFC 2123, March 1997.

[84] N. Brownlee, "Nettramet distribution." `http://www.caida.org/tools/measurement/netramet/dist.xml`. [Online; accessed 08-October-2011].

[85] G. Sadasvian, N. Brownlee, B. Claise, and J. Quittek, "Architecture for ip flow information export." IETF RFC 5470, March 2009.

[86] L. Deri, R. Carbone, and S. Suin, "Monitoring networks using ntop," in *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pp. 199–212, 2001.

[87] L. Deri, "ntop." `http://www.ntop.org/`. [Online; accessed 08-October-2011].

[88] CAIDA, "Coralreef." `http://www.caida.org/tools/measurement/coralreef/`. [Online; accessed 08-October-2011].

[89] Cisco, "Cisco ios netflow version 9." IETF RFC 3954, October 2004.

[90] U. of Waikato, "Network research group." `http://www.endace.com/endace-dag-high-speed-packet-capture-cards.html`. [Online; accessed 08-October-2011].

[91] Endace, "Endace dag cards." `http://www.endace.com/endace-dag-high-speed-packet-capture-cards.html`. [Online; accessed 08-October-2011].

[92] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *Networking, IEEE/ACM Transactions on*, vol. 12, pp. 219 – 232, april 2004.

[93] K. C. Claffy, H.-W. Braun, and G. C. Polyzos, "Tracking long-term growth of the nsfnet," *Commun. ACM*, vol. 37, pp. 34–45, August 1994.

[94] "Libpcap library." `http://www.tcpdump.org/`. [Online; accessed 08-October-2011].

[95] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," tech. rep., Cornell University, Ithaca, NY, USA, 1987.

[96] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[97] J. Y. Chung, B. Park, Y. J. Won, J. Strassner, and J. W. ki Hong, "An effective similarity metric for application traffic classification," in *Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010)*,, (Osaka, Japan), pp. 286–292, May 2010.

[98] J. Newsome, B. Karp, and D. X. Song, "Polygraph: Automatically generating signatures for polymorphic worms.," in *Proceedings of the IEEE Symposium on Security and Privacy*, (Oakland, CA, USA), pp. 226–241, May 2005.

[99] D. Brumley, J. Newsome, D. Song, H. Wang, and S. Jha, "Towards automatic generation of vulnerability-based signatures," in *In Proceedings of the IEEE Symposium on Security and Privacy*, (Oakland, CA, USA), pp. 2–16, May 2006.

[100] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press, 2nd ed., 2001.

[101] J. Newsome and D. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, (San Diego, CA, USA), February 2005.

[102] "The gnutella protocol specification." `http://rfc-gnutella.sourceforge.net/developer/stable/index.html`. [Online; accessed 08-October-2011].

[103] Y. Won, B.-C. Park, H.-T. Ju, M.-S. Kim, and J. Hong, "A hybrid approach for accurate application traffic identification," in *End-to-End Monitoring Techniques and Services, 2006 4th IEEE/IFIP Workshop on*, pp. 1 – 8, april 2006.

[104] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, (New York, NY, USA), pp. 314–329, ACM, 2003.

[105] J. Erman, A. Mahanti, and M. Arlitt, "Byte me: a case for byte accuracy in traffic classification," in *Proceedings of the 3rd annual ACM workshop on Mining network data*, MineNet '07, (New York, NY, USA), pp. 35–38, ACM, 2007.

[106] "L7-filter." `http://l7-filter.clearfoundation.com/`. [Online; accessed 08-October-2011].

[107] "Opendpi." `http://www.opendpi.org/`. [Online; accessed 08-November-2011].

[108] "Ipoque." `http://www.ipoque.com/`. [Online; accessed 08-November-2011].

[109] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, pp. 131–156, 1997.

[110] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[111] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of the ninth international workshop on Machine learning*, ML92, (San Francisco, CA, USA), pp. 249–256, Morgan Kaufmann Publishers Inc., 1992.

[112] I. Kononenko, "Estimating attributes: analysis and extensions of relief," in *Proceedings of the European conference on machine learning on Machine Learning*, (Secaucus, NJ, USA), pp. 171–182, Springer-Verlag New York, Inc., 1994.

[113] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Mach. Learn.*, vol. 53, pp. 23–69, October 2003.

[114] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data Min. Knowl. Discov.*, vol. 2, pp. 169–194, June 1998.

[115] DISTIMO, "The battle for the most content and the emerging tablet market." `http://www.distimo.com`. [Online; accessed 08-November-2011].

Acknowledgements

사랑하는 저의 부모님께 이 논문을 바칩니다.

Curriculum Vitae

Byungchul Park

Birthday	Feb. 1, 1983
Address	**Contact**
RIST Building 4, Rm. 4405, POSTECH SAN 31, HYOJA-DONG, NUM GU POHANG, KOREA, 790-784	Office : +82-54-279-5641 Cell : +82-11-9441-1141 Fax : +82-54-279-5699
Web	**Email**
http://home.postech.ac.kr/~fates	fates@postech.ac.kr

Education

Mar. 2006 – Feb. 2012 : Ph.D. in Computer Science and Engineering, POSTECH, Korea

Mar. 2001 – Feb. 2006 : B.S. in Computer Science and Engineering, POSTECH, Korea

Experience

Sept. 2008 – Mar. 2009 : Research Intern, Wireless & Network group, Microsoft Research Asia, Beijing, China

Publications: International Journal/Magazine Papers

1. Byungchul Park, Young J. Won, and Jame Won-Ki Hong, “Toward Fine-grained Traffic Classification”, IEEE Communications Magazine, vol. 49, Issue 7, July, 2011. pp. 104-111 **(SCI)**

2. Young J. Won, Mi-Jung Choi, Byungchul Park, James W. Hong, and John Strassner, "A Novel Approach for Failure Recognition in IP-Based Industrial Control Networks and Systems", Journal of Network and Systems Management (JNSM). Accepted to appear **(SCIE)**

Publications: International Conference/Workshop Papers

1. Yeongrak Choi, Jae Yoon Chung, Byungchul Park, and James Won-Ki Hong, “Automated Classifier Generation for Application Level Mobile Traffic Classification”, the 13th IEEE/IFIP Network Operations and Managment Symposium (NOMS 2012), accepted to appear.
2. Jae Yoon Chung, Yeongrak Choi, Byungchul Park, and James Won-Ki Hong, “Measurement Analysis of Mobile Traffic in Enterprise Networks”, 13th Asia-Pacific Network Operations and Management Symposium (APNOMS 2011), Taipei, Taiwan, Sep. 21-23, 2011, pp. 1--4.
3. Jae Yoon Chung, Byungchul Park, Young J. Won, John Strassner, and James W. Hong, “An Effective Similarity Metric for Application Traffic Classification”, the 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), Osaka, Japan, Apr. 19-23, 2010, pp. 286--292.
4. Seong-Cheol Hong, Jin Kim, Byungchul Park, Young J. Won, and James W. Hong, “Traffic Growth Analysis over Three Years in Enterprise Networks”, 15th Asia-Pacific Conference on Communications (APCC 2009), Shanghai, China, Oct. 2009, pp. 896--899.
5. Jae Yoon Chung, Byungchul Park, Young J. Won, John Strassner, and James W. Hong, “Traffic Classification Based on Flow Similarity”, 9th IEEE International Workshop on IP Operations and Management (IPOM 2009), Venice, Italy, Oct. 2009, pp. 56--77.
6. Byungchul Park, Young J. Won, Hwanjo Yum and James Won-Ki Hong, “Fault Detection in IP-Based Process Control Networks using Data Mining Technique”, 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), New York, USA, Jun. 2009, pp. 211--217.
7. Byungchul Park, Young J. Won, Mi-jung Choi, Myung-Sup Kim, and James W. Hong, “Empirical Analysis of Application-level Traffic Classification using Supervised Machine

Learning", 11th Asia-Pacific Network Operations and Management Symposium (APNOMS 2008), Beijing, China, Oct. 2008, pp. 474--477.

8. Byung-Chul Park, Young J. Won, Myung-Sup Kim, and James Won-Ki Hong. "Towards Automated Application Signature Generation for Traffic Identification", 11th IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), Salvador, Brazil, April 2008, pp. 160--167.

9. Young J. Won, Byung-Chul Park, Mi-jung Choi, James W. Hong, Hee-Won Lee, Chan-Kyu Hwang, Jae-Hyoung Yoo, "End-User IPTV Traffic Measurement of Residential Broadband Access Networks", 6th IEEE International Workshop on End-to-End Monitoring Techniques and Services (E2EMON 2008), Salvador, Brazil, April 2008, pp. 95--100.

10. Young J. Won, Byung-Chul Park, Mi-Jung Choi, and James Won-Ki Hong. "Service-based Charging Scheme for Mobile Data Networks", 1st KICS International Conference, Yanbian, China, Aug. 23-25, 2007.

11. Young J. Won, B.C. Park, S.C. Hong, K.B. Jung, H.T. Ju, James W. Hong, "Measurement Analysis of Mobile Data Networks", Passive and Active Measurement Conference (PAM 2007), Louvain-la-neuve, Belgium, April 5-6, 2007, pp. 223--227.

12. Young Joon Won, Byung-Chul Park, Myug Sup Kim, Hong-Tek Ju, and James Won-ki Hong, "A Hybrid Approach for Accurate Application Traffic Identification", IEEE/IFIP E2EMON, Vancouver, Canada, April 3, 2006, pp. 1--8.

Domestic Journal/Conference Papers

1 journal & 9 **conference papers** (in Korean), available upon request

Research/Project Experience

- Title: “Design and Implementation of Mobile Traffic Classification Methodology” – Microsoft Research Asia (2011-)

This research project aims at developing a novel application-level traffic classification methodology to monitor and analyze mobile traffic. The key challenges are the different traffic characteristics of mobile traffic compared to traditional Internet traffic and the limited computing resources of mobile devices. I role in this project is developing methodologies for discrimination of mobile traffic from the mixture of mobile and non-mobile traffic and a new application-level mobile traffic classification.

- Title: “Highly Manageable Network and Service Architecture for Next Generation (HiMang)” – Electronics and Telecommunications Research Institute (ETRI) (2010-)

HiMang is a part of 3-year-long government funded project. This research project is developing a novel autonomic and cognitive approach to providing a highly manageable network and service management architecture for current as well as future networks. It is based on using an innovative knowledge representation methodology that unifies disparate knowledge source and greatly improves learning, decision-making, and reasoning in management systems. My work involves investigating traffic monitoring methodologies for HiMang architecture.

- Title: “IT Convergence for Ubiquitous Autonomic Systems” – Ministry of Education, Science, and Technology, Korea (2009-)

This is part of the research work proposed under our World Class University grant from the Korean government. My work involves investigating how autonomic mechanisms can be applied to manage new ubiquitous computing systems that use bio-informatics, nano-technologies, and networking technologies for building ubiquitous computing applications (called ubiquitous health and ubiquitous environment applications in Korea).

- Title: “Collect, Analyze, and Share for Future Internet (CASFI) – Manageability of Future

Internet" – Ministry of Knowledge Economy, Korea (2008-)

CASFI is a 5-year-long government funded project that focuses on high-precision network performance measurement and analysis. This project is developing new performance measurement and analysis mechanisms for the current Internet; this will be used to gain insight to develop better manageability approaches for the future Internet. I have focused on investigating different manageability challenges for the Future Internet. For example, I have developed an automated signature generation system and similarity-based traffic classification algorithms. I also have given a talk at the CAIDA-WIDE-CASFI annual joint workshop since 2008.

- Title: "Data Center Networking" – Microsoft Research Asia Internal project (2008-)

In this project, we are working on several aspects of data center networking (DCN), including DCN architecture, DCN infrastructure consolidating, DCN measurement and management, DCN resource management and congestion control, traffic engineering and power control, and application and OS support in DCN. I role in this project was developing a method for measuring and monitoring data center internal traffic. I was also involved in detecting performance bottleneck of a data center.

- Title: "Fault Detection & Prediction for Industrial Control Networks" – POSCO (2008-2009)

This work was a follow-on project from the above fault monitoring project of POSCO. I extended the capabilities of their existing fault diagnosis system to include fault prediction and adaptive decision.

- Title: "Load testing for network-based home robots via simulation," – Korea Telecom (2007-2008)

A network-based intelligent service robot, called a ubiquitous robotic companion (URC), has been introduced recently. URC robots access services, content, and application software via the Internet and provide users with intelligent services in diverse fields such as education, entertainment, health care, etc. The quality of the URC service is dependent on the performance of the URC robot server. Korea Telecom (KT) initiated a pilot project on URCs as a national project and started a

URC robot service as a trial business from 2005. They are planning to deploy one million network-based robots to households. I was responsible for designing and implementing a network-based load testing simulation system which evaluates the URC robot management server performance and analyzes the performance evaluation results.

- prior to the deployment of the URC robots by testing the URC robot servers.Title: “End-User IPTV traffic modeling and analysis” – POSTECH Internal Project (2008)

Offering IPTV to broadband access subscribers is a key challenge as well as a prospective revenue source for ISPs. Despite of its growing interest, no comprehensive study has presented the traffic details of realworld commercial IPTV services yet. We have measured commercial IPTV traffic via four different residential broadband access networks, namely xDSL, Cable, FTTB, and FTTH. Based on our observations on IPTV traffic characteristics, we have formulated the representation for bandwidth demand in IPTV VoD services. Using this formula, we simulate the proposed model on the dedicated links where the IPTV server farm supports 200 active viewers.

- Title: “Triple Play Services Traffic Impact Analysis on Broadband Access Networks – Korea Telecom (2007)

Korea Telecom provides full scale triple- and quadruple-play services over IP networks. This project focused on traffic impact analysis on broadband access networks (e.g., xDSL, FTTH) that used large-scale triple play services. It involved traffic measurement and analysis of each of the services provided, along with analyzing how the user utilized each service. I was responsible with providing analysis results and projection models for these services.

- Title: “Remote Fault Monitoring System for Industrial Control Networks” – POSCO (2006-2008)

This project was funded by POSCO, the second largest iron and steel manufacturer in the world. POSCO operates many different types of specialized IP networks for their manufacturing process and various business-specific uses. I was involved in developing an advanced fault monitoring system for a set of their Industrial Control Networks (ICN); this system was modular and

independent of any one specific ICN. We also developed a set of related analysis techniques for handling different types of ICN-specific alarms and faults.

- Title: “Traffic Analysis and Application-specific Billing Systems for Commercial Mobile Data Networks” – nTelia (2006)

This was a confidential research project for a large Korean vendor. The project goal developed sophisticated billing schemes for Internet users over 2G and 2.5G cellular phone networks. It involved real-time packet inspection for improved classification and billing by using advanced traffic characteristic analysis in mobile data networks.