NeTraMark: A Network Traffic Classification Benchmark

Suchul Lee
Seoul National University
sclee@popeye.snu.ac.kr

Hyun-chul Kim
Seoul National University
hkim@mmlab.snu.ac.kr

Dhiman Barman
Juniper Networks
tenida@gmail.com

Sungryoul Lee
Seoul National University
srlee@popeye.snu.ac.kr

Chong-kwon Kim
Seoul National University
ckim@snu.ac.kr

Ted "Taekyoung" Kwon
Seoul National University
tk@mmlab.snu.ac.kr

ABSTRACT

Recent research on Internet traffic classification has produced a number of approaches for distinguishing types of traffic. However, a rigorous comparison of such proposed algorithms still remains a challenge, since every proposal considers a different benchmark for its experimental evaluation. A lack of clear consensus on an objective and scientific way for comparing results has made researchers uncertain of fundamental as well as relative contributions and limitations of each proposal. In response to the growing necessity for an objective method of comparing traffic classifiers and to shed light on scientifically grounded traffic classification research, we introduce an Internet traffic classification benchmark tool, NeTraMark. Based on six design guidelines (Comparability, Reproducibility, Efficiency, Extensibility, Synergy, and Flexibility/Ease-of-use), NeTraMark is the first Internet traffic classification benchmark where eleven different state-of-the-art traffic classifiers are integrated. NeTraMark allows researchers and practitioners to easily extend it with new classification algorithms and compare them with other built-in classifiers, in terms of three categories of performance metrics: per-whole-trace flow accuracy, per-application flow accuracy, and computational performance.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network monitoring

General Terms

Design, Management, Measurement, Performance

Keywords

Benchmark, Traffic classification

1. INTRODUCTION

Traffic classification has gained substantial attention within the Internet research and operations communities. Traffic classification is used to measure, understand and predict patterns and trends of network resource usage. Specific usages include network provisioning, capacity planning, protecting networks against security threats, etc. All the issues over the appropriate use and pricing of the Internet, such as (i) the tussle between file sharing user communities and intellectual property representatives like RIAA (Recording Industry Association of America) and MPAA (Moving Picture Association of America), (ii) the continuous struggle between malicious hackers and Internet security companies, and (iii) the network neutrality debate [22] between network service providers and content/application service providers, boil down to the ability of network operators to know what kind of traffic goes over their network [6].

Until the fall of 2002, when P2P file sharing applications like KaZaA started using dynamic ports, traffic classification had largely relied on the use of transport layer port numbers. As increasingly popular applications like P2P file sharing attempt to evade their identity by dynamically assigning ports and/or masquerading into the well-known ports of other applications, port-based classification has become less reliable [14]. As a more accurate and reliable approach, Deep Packet (Payload) Inspection (DPI) technique looks at the packet payloads to classify traffic as many applications write their signatures in the first few bytes in the payload. While DPI has been found very accurate once given a set of unique payload signatures for corresponding applications, it fails to work on encrypted traffic and entails significant privacy and legal concerns, which often preclude access to payload data. Worse, some applications like Gnutella (and its variants) have evolved to get around DPI as well by using variable length padding[1] [18].

The research community has responded by developing classification techniques capable of inferring application-specific communication and/or statistical patterns without inspection of packet payloads, most of which are categorized into (i) host-level communication behavior-based approaches represented by BLINC [13] and Traffic Dispersion Graphs (TDG) [11, 12], which inspect "social interaction" for classification, and (ii) statistical approaches where data mining techniques are applied on a variety of traffic flow features such as packet size, inter-packet arrival time, etc [15, 16, 17, 20].

Despite a variety of algorithms proposed for traffic classification, mutual comparison of them and the quantification of benefits of one approach over another still remain a challenge, since every paper considers different performance metrics, application categories, codes, datasets (typically locally collected since there is few publicly available trace data to use as a benchmark [9, 6]), and even ground-truth for its experimental evaluation [6, 9, 19]. This lack of an objective and head-to-head comparison/benchmark framework has left researchers and practitioners little basis for consensus on what approach to use when and where, and how could an approach be improved upon (or extended) to

[1]Erman *et al.*'s measurements indicate that 400 payload bytes of each packet is required to identify 90% of the Gnutella flows using payload signatures [18].

identify a specific type of and/or all the existing network applications [6].

In response to the growing necessity for a traffic classification benchmark with which researchers and operators can compare different approaches in an objective way, we present an extensible Internet traffic classification benchmark framework, named NeTraMark[2]. We believe NeTraMark's design principles and implementation lay a foundation on which scientifically grounded (i.e., comparable and reproducible) traffic classification research will be enabled.

We highlight the main contributions from this paper:

1) We identify six key requirements (Comparability, Reproducibility, Efficiency, Extensibility, Synergy, and Flexibility/Ease-of-use) for a traffic classification benchmark, inspired from the previous work and experiences of Salgarelli *et al.* [9] and Kim *et al.* [6]. A benchmark tool fulfilling these requirements should help researchers and operators (i) understand fundamental contributions and limitations of existing approaches as well as new proposals of their own, and (ii) choose what approach(es) to use for a specific purpose in a given environment (e.g., a single approach vs. multiple combined approaches, backbone vs. edge link, target applications or threats, etc.)

2) We present NeTraMark. As it is designed and implemented on the basis of the above six requirements, the capabilities of NeTraMark are not limited to the comparison of multiple traffic classification approaches. It also enables users to achieve various classification needs in the field of network monitoring and operation; for example, a user might be more interested in finding heavy P2P traffic sources while others might be interested in detecting malicious traffic. Similarly, different users might focus on different performance metrics.

NeTraMark has the following three key features.

First, it incorporates a rich set of eleven state-of-the-art traffic classification approaches in its classification engine. The suite of classifiers includes: (i) the payload-based classifier, `crl_pay`, which has been developed and widely used by the Internet research community[3] [6, 13, 7, 12], (ii) the graph-based classifiers, BLINC and Traffic Dispersion Graphs, (iii) the CoralReef's [1] ports-applications matching database (i.e., ports-based classifier), (iv) the seven most commonly used machine learning algorithms such as C4.5 Decision Tree, Naive Bayes, Naive Bayes Kernel Estimation, Bayesian Networks, k-Nearest Neighbors, Neural Networks and Support Vector Machines [6], and (v) a combined weight-based classifier which takes classification results from some or all of the aforementioned classifiers and performs a weighted voting process to derive a single best classification result[4].

NeTraMark enables users to perform a thorough evaluation of these traffic classification algorithms, based on six performance metrics: per-trace accuracy (Overall Accuracy), per-application accuracy (Precision, Recall, and F-Measure), and computational performance (Learning Time and Classification Time). To the best of our knowledge, this is the first work that has undertaken this challenging task of integrating the state-of-the-art classification engines.

Second, NeTraMark allows researchers and practitioners to extend it with new classification algorithms easily and compare their performance with that of the eleven existing ones. Users can combine multiple classifiers to test for possible synergies (i.e., more accurate and complete results).

Third, NeTraMark has adapted the interactive visualization module of BLINC and TDG graphs (which capture transport layer communication and social behavior of hosts as shown in Fig. 3 and Fig. 4). This helps users further examine and identify new P2P or malicious traffic flows whose classification results are left "unknown" with the incorporated classifiers.

3) We make the the NeTraMark source code available[5] for researchers or practitioners interested in validating or extending this benchmark tool. We believe that NeTraMark will facilitate both the research and operation communities pursuing the direction of comparable and reproducible traffic classification research, to address the one of the foremost challenges in this field: effectively comparing between the many proposed approaches [19].

The rest of this paper proceeds as follows. We first review the requirements for a traffic classification benchmark in Section 2. Section 3 describes the architecture design, implementation, and usage examples of NeTraMark. Section 5 concludes the paper.

2. REQUIREMENTS ANALYSIS

NeTraMark has been developed based on six key requirements as discussed in this section.

Comparability: A benchmark tool should facilitate users to evaluate candidate algorithms on the same performance metrics and traces [9]. It is also desirable that the benchmark tool retains the definitions of application classes/categories of different classification methods, yet in a comparable way [9]. This requirement helps users quantify improvements of one approach over others.

Reproducibility: To verify the results obtained by different research groups, classification approaches and their results must be reproducible within the framework of a benchmark tool. Often results from different groups have not been reproducible as the codes and datasets used in their experimental evaluation are not shared nor publicly available [9, 19, 6]. Note that reproducibility is also important to stimulate adoption of a traffic classification algorithm.

Efficiency: Traffic classification involves processing of voluminous datasets which is both time-consuming and resource intensive. Efficiency in running a benchmark tool is a key to experimental productivity.

Extensibility: Researchers and practitioners should be able to easily extend the benchmark's features. In particular, users should be able to incorporate new algorithms and compare them with existing ones.

Synergy: Every traffic classification method has its own strengths and weaknesses. For example, payload-based classifiers are not applicable to encrypted traffic data. Careful combinations of classifiers may be synergistic and result in more accurate and complete classifications [6, 8]. A traffic classification benchmark should allow users to test and obtain synergy by combining multiple classification methods on a given dataset such that a combined classifier (i) outper-

[2]NeTraMark was demonstrated in the IEEE INFOCOM Demo Session, San Diego, CA, March 2010.

[3]The `crl_pay` and BLINC codes have been distributed to > 30 universities and research institutes.

[4]The combined classifier may not always perform the best.

[5]http://popeye.snu.ac.kr/∼sclee/NeTraMark, on a per-request basis, for research purposes only.

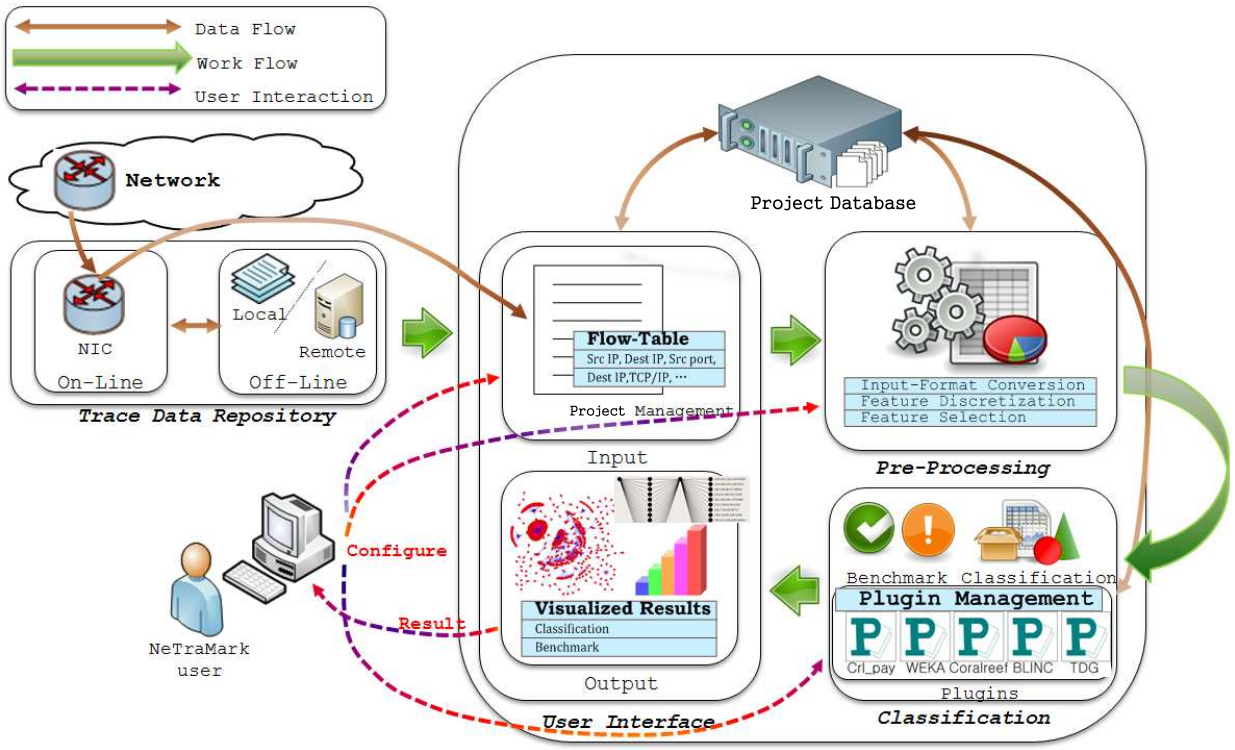

Figure 1: Block diagram of the NeTraMark software suite

forms every individual method, and (ii) performs tasks that are not supported by any single method (e.g., further investigation on "Unknown"-labeled traffic with multiple approaches.)

Flexibility/Ease-of-use: a traffic classification benchmark tool should allow users to configure various traffic classifiers in a variety of settings in a flexible and easy way, either interactively or via batch. Different elements such as pre-processing of data, selection of flow features, data format conversion, deciding on ground-truth reference should be automated and/or pipe-lined whenever necessary.

In the next section, we present the architecture and implementation of NeTraMark meeting these six requirements.

3. NETRAMARK

3.1 Performance Metrics

Using the same and appropriate metrics to evaluate the performance of traffic classification algorithms constitutes the basis of a benchmark. To (i) avoid the coarseness of evaluating performance over an entire trace rather than for each application contained in it [6] and (ii) eliminate the bias known as the "class imbalance problem" [19], NeTraMark assesses the performance of each algorithm on a per-application basis using the following four metrics: *overall accuracy, precision, recall, and F-Measure*. For a given application class, the number of correctly classified flows[6] is referred to as the True Positives (TP). False Positives (FP) is the number of flows falsely ascribed to a given application. False Negatives (FN) is the number of flows from a given application that are falsely labeled as another application.

•*Overall accuracy* is the ratio of the sum of all True Positives to the sum of all the True Positives and False Positives for all classes. We apply this metric to measure the accuracy of a classifier on the whole trace set. The following three metrics are used to evaluate the quality of classification results for each application class.

•*Precision* $= TP/(TP + FP)$, or the percentage of flows that are properly attributed to a given application.

•*Recall* $= TP/(TP + FN)$, or the percentage of flows in an application class that are correctly identified.

•*F-Measure*, a widely-used metric in information retrieval and classification [21], considers both precision and recall in a single metric by taking their harmonic mean: *2 × Precision × Recall / (Precision + Recall)*. NeTraMark uses this metric to compare and rank the per-application performance of classifiers.

In addition, NeTraMark also evaluates the computational performance of traffic classifiers, using the following two metrics[7]:

•*Classification time* is the time spent for a classifier to perform classification operations on a given data set.

[6]Currently, NeTraMark does not adopt "byte accuracy", which is also an important measure for a traffic classifier's performance [19]. It will be incorporated in the next release of NeTraMark.

[7]NeTraMark evaluates the computational performance of concrete implementations plugged into its own platform, not the theoretical complexity of the algorithms because (i) all the plugged-in codes are real implementations developed and used in previous traffic classification efforts [6, 1, 15, 13, 11, 2, 17, 20, 16], and (ii) this approach yields tangible performance numbers for comparisons [20]. Optimized implementations would likely yield faster learning and classification speeds for all classifiers.

Table 1: Application Categories

Category	Application/protocol
Web	HTTP, HTTPS
P2P	FastTrack, eDonkey, BitTorrent, Ares, Gnutella, WinMX, OpenNap, MP2P, SoulSeek, Direct Connect, GoBoogy, Soribada, PeerEnabler, Napster Blubster, FileBEE, FileGuri, FilePia IMESH, ROMNET, HotLine, Waste
FTP	FTP
DNS	DNS
Mail/News	BIFF, SMTP, POP, IMAP, IDENTD, NNTP
Streaming	MMS(WMP), Real, Quicktime, Shoutcast, Vbrick Streaming, Logitech Video IM Backbone Radio, PointCast, ABACast
Network Operation	Netbios, SMB, SNMP, NTP SpamAssasin, GoToMyPc, RIP ICMP, BGP, Bootp, Traceroute
Encryption	SSH, SSL, Kerberos, IPSec, ISAKMP
Games	Quake, HalfLife, Age of Empires, DOOM Battle field Vietnam, WOW, Star Sieze Everquest, Startcraft, Asherons, HALO
Chat	AIM, IRC, MSN Messenger, Yahoo messenger IChat, QNext, MS Netmeet, PGPFone, TALK
Attack	Address scans, Port scans
Unknown	-

•*Learning time* is the time spent for an (supervised machine learning) algorithm to build a classification model using a given training data set.

A flow is defined on its 5-tuple (source IP address, destination IP address, transport protocol, source port, destination port) with a timeout of 64 seconds [5].

3.2 Application Categories

Another issue to consider when comparing multiple classifiers is that the definition of application classes varies across the classifiers [10, 9, 6]. The scope and goal of different methods render them incomparable in some cases. For example, BLINC does not distinguish between *BitTorrent* and *Gnutella* as different P2P applications and clubs them under P2P application, while CoralReef can identify individual P2P applications.

We use the definition of *group* classes and assign each application into a relevant group as in [10, 6, 1]. This is to address the "Comparability" requirement, which allows us to (i) compare a technique that classifies traffic into specific applications with another classifying into group classes, and (ii) do a higher-level (group-class level) performance comparison. By default, NeTraMark is configured to use application classes as shown in Table 1. The default categorization has been derived from those of the default component classifiers: CoralReef, BLINC, `crl_pay`, etc. Users can change and update the default application categories by modifying the NeTraMark source code, as new applications may appear over time or different categorizations may be more appropriate in different contexts [9]. For example, the "Network Operation" category may include different application protocols or may be further divided into a few subcategories when needed.

3.3 Architecture and Implementation

We present the architecture overview of NeTraMark in Fig. 1, which consists of five parts: (1) Graphical and command-line user interface for interactive and batch processing, (2) Data preprocessing for traffic flow feature selection, feature discretization, input data format conversion and other related tasks, (3) Traffic Data Repository from/to which datasets are retrieved and stored, (4) PostgreSQL-based [3] Project Database where classification results are stored in a fast and efficient manner, for later retrieval and re-use, and (5) Classification engine, currently comprised of eleven state-of-the-art traffic classification methods: the payload-based classifier `crl_pay`, BLINC (including the Reverse BLINC module [6]), Traffic Dispersion Graph (TDG), CoralReef's ports-based classification DB (the up-to-date version 3.8 [1]), and the seven most often-used machine learning algorithms from WEKA[2]. Moreover, NeTraMark also provides a combined weight-based classification method, which takes prediction results from the other classifiers and performs a weighted voting process to derive a single best classification.

3.3.1 Project Management

NeTraMark operates on voluminous trace data with multiple classification algorithms, rendering a benchmarking process both time-consuming and resource-intensive. In order to achieve efficiency in running several benchmarking experiments and avoid unnecessary repetition of the same experiments, NeTraMark stores classification benchmark results in "Project" DB.

Each project consists of four components: (1) ***Task***: a set of classifiers selected by users for benchmarking, (2) ***Timestamp***: the (UNIX) system time at which the project started, (3) ***Flow-table***: a collection of flows (generated with 5 min. interval by default) with the user-selected flow features and classification results of all the tested classifiers, (4) ***Benchmark results***: accuracy and efficiency results.

Due to the sheer amount of trace data and processing involved, NeTraMark adopts the PostgreSQL [3] Database Management System (DBMS), for fast and efficient indexing and retrieval of flow records as well as classification results.

The Project Management (PM) module provides the following advantages:

(1) *Reusability*: The PM allows users to store all the classification and benchmark results from an experiment as a project. Users can reload the saved project later to review the results, and then simply extend it with any new classification algorithms to compare and/or datasets to add, without repeating the same experiments already done and stored in the project.

(2) *Extension to a Distributed NeTraMark System*: As stored project or their sub-components can be shared and collected from remote locations, NeTraMark can be used to build a distributed traffic classification and benchmark system, in which different (or the same set of) classifiers run on different datasets (typically locally collected from different locations [9]) over different servers as necessary. This extension is left for future work.

3.3.2 User Interface

NeTraMark's User Interface (UI) provides users with various knobs to configure benchmarking experiments and visualize the results. Upon startup, users can establish a *project*, a basic operational unit of NeTraMark's interactive and batch processing, by selecting a set of classifiers to evaluate, traffic flow features to use and/or discretize, and datasets against which the selected classifiers will be tested (and trained, if necessary). Users also can adjust the weights of those classifiers to obtain combined classification

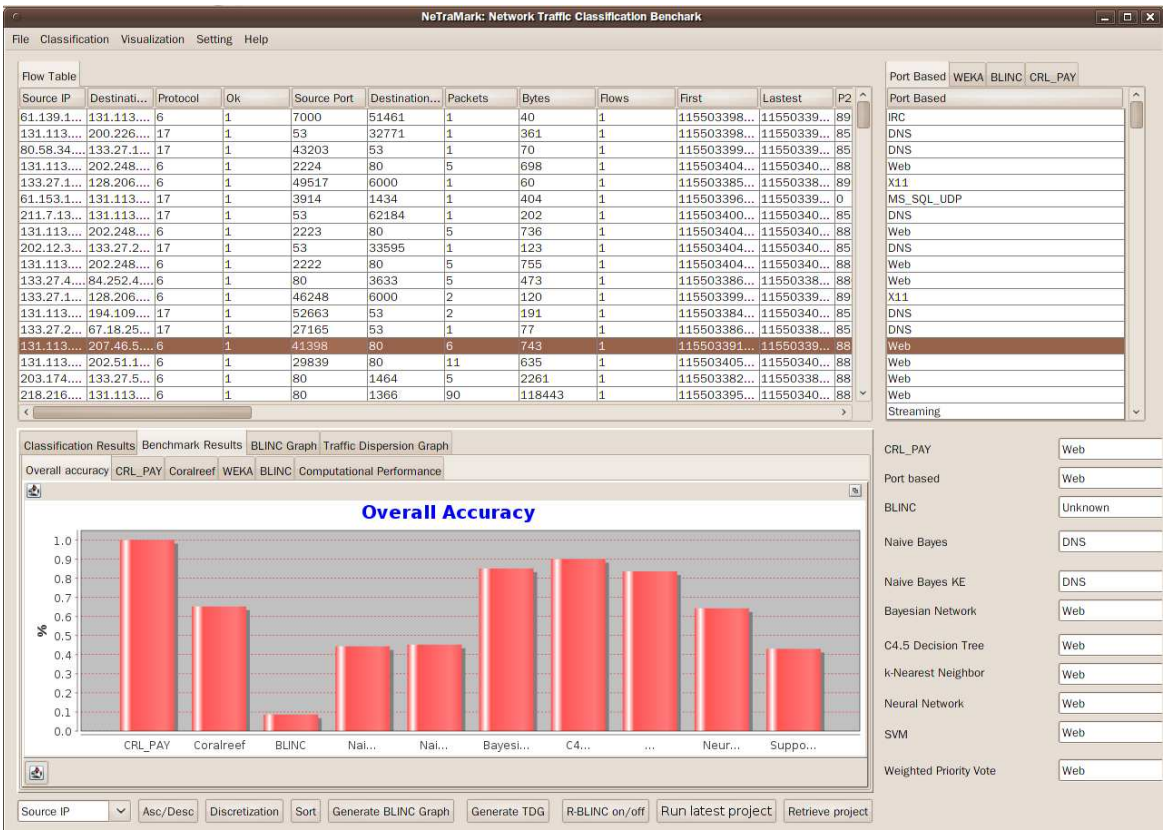

Figure 2: The graphical NeTraMark user interface with an example benchmark result.

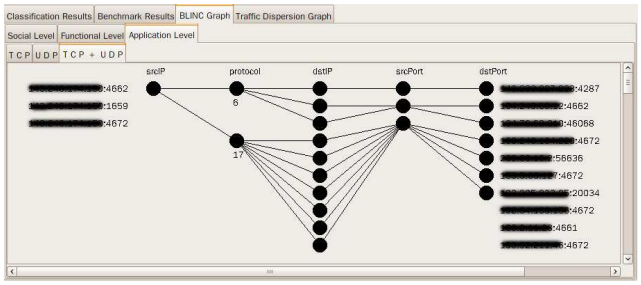

Figure 3: BLINC Visualization (The BLINC Graph tab in the bottom-left panel): a behavior graph of a P2P host.

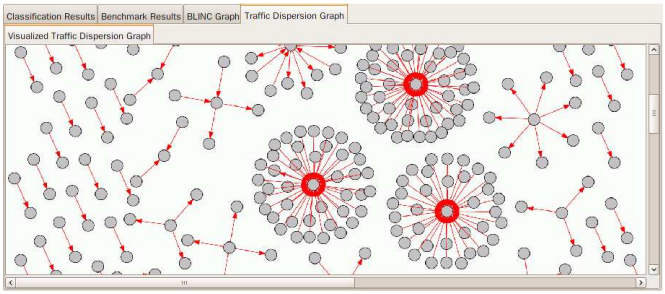

Figure 4: TDG Visualization (the TDG tab on the bottom-left panel): A TDG graph of "Unknown" flows.

results. Once users are done with all the configuration processes and finish running the experiment, they can re-load and re-run the most recent experiment by clicking "Run the latest project". This knob reprieves users from repeating tedious manual configuration process, particularly when the same experiments have to be repeated on multiple traces.

Fig. 2 shows a screenshot of the graphical UI, comprised of four different panels with different statistics and views.

(i) Top-left panel displays flow records (of the first 5 minutes' interval by default) consisting of 38 unidirectional flow features most of which were inspired from the 248 bidirectional features used in [16] and the 22 bidirectional features in [20]. These 38 unidirectional features have proved useful and more than expressive enough to distinguish among different applications [6, 7]. The 38 flow features are: protocol, source and destination ports, the number of packets, transferred bytes, the number of packets without Layer 4 (TCP/UDP) payload, start time, end time, duration, average packet throughput and byte throughput, max/min/average/standard deviation of packet sizes and inter-arrival times, the number of TCP packets with FIN, SYN, RSTS, PUSH, ACK, URG (Urgent), CWE (Congestion Window Reduced), and ECE (Explicit Congestion Notification Echo) flags set (all zero for UDP packets), the size of the first ten packets, and payload classification results provided by the `crl_pay` classifier. Users can select and/or discretize specific features before running classification experiments. (ii) Top-right and (iii) Bottom-right panels display two types of classification

Table 2: Classifier class definition

```
public class Classifier_plugin {
 boolean enabled;
 String name;
 String version;
 String description;
 boolean external_or_internal;
 String execution;
 Object[] parameters;
}
```

Table 3: An instance of a classifier (Naive Bayes)

```
enabled = true; // Use this classifier plugin
name = ''Naive_Bayes'';
version = ''3.7.2'';
description = ''Naive Bayes is the simplest probabilistic
classifier...'';
external_or_internal = false; // external code (non-Java code)
execution = ''weka.classifiers.bayes.NaiveBayes'';
// library information
parameters[0] = ''-D''; // execution arguments
```

results. The former shows per-classifier view (a table of application labels generated by a single classifier[8] and the latter shows per-flow view (classification results of different classifiers for the single highlighted flow). (iv) Bottom-left panel displays performance statistics such as application breakdown obtained from each classifier, performance benchmarking results (as in Fig. 2), as well as host-behavior graphs generated by BLINC (Fig. 3) and TDG (Fig. 4).

As such, this four-panel user interface enables users to easily obtain, compare, and visualize the classification results as well as performance of different classifiers and figure out underlying traffic mix in a dataset.

3.3.3 Classifier Plugins

In the current implementation of NeTraMark, each classification module is plugged in via a Java class, whose members include: (i) name of the corresponding classifier, (ii) list of parameters/options for the classifier (which can be configured through the user interface), and (iii) link to the executable codes (for non-Java codes such as BLINC, TDG, crl pay, etc.) or library (for Java codes like WEKA), etc. Table 2 and 3 show the class definition and an instantiation of Naive Bayes classifier plugin, respectively. Users can add new classification plugins written in any user-preferred programming languages or modify existing ones by adding new classes or reconstructing existing ones in the NeTraMark source code. To help plugin developers, detailed documentation on how to add or modify classification plugins as well as sample reference codes are distributed with NeTraMark.

3.3.4 Ground Truth

Establishing a comparison reference point (i.e., ground truth) also forms the basis on which to compare performance of different traffic classifiers. As a flexible traffic classification benchmark framework, NeTraMark allows users to choose any plugged-in classifier (including a combined weight-based classifier) to establish ground truth, while its default setting is to use the payload-based `crl_pay` classifier for the purpose. This flexibility (and extensibility together) helps to update and maintain NeTraMark with the most accurate and complete ground truth available, as users can add newly developed classification techniques that generate such results.

[8]When the "WEKA" tab is clicked, sub-tabs of the seven machine learning algorithms appear.

3.3.5 Combination of multiple classifiers

A traffic classifier typically consists of multiple fingerprints designed to identify specific target applications, thus each approach often has its own strengths and weaknesses in classifying a particular application. When a classifier has a fundamental weakness in identifying particular types of traffic, integrating aspects of other techniques can help [6]. Motivated by the above observations, NeTraMark supports the combination of multiple classifiers by providing a weight-based combined classifier, whose per-application weight values (i.e., confidence level of the prediction) can be either given manually by users or automatically derived from per-application performance history in the n latest experiments.

For example, suppose that classifiers A, B, and C identify a flow F as of Web, Web, and BitTorrent, respectively, while the average F-Measures of these classifiers have been set to 0.6 on Web, 0.5 on Web, and 0.9 on BitTorrent. In this case, our combined classifier may identify the flow F as of Web or BitTorrent depending on which combination algorithm is set to use: the majority votes, the best F-Measured one, or hybrid which somehow takes into account both of them. The current implementation of NeTraMark supports the two simplest combination rules only, the majority votes and the best F-Measured classifier. More complicated strategies like hybrid ones can be developed and added by users.

3.3.6 System Requirements

The current implementation of the NeTraMark framework is written in Java using NetBeans IDE 6.7, while its classifier plugins are in C++ (`crl_pay` and BLINC), Java (machine learning algorithms from WEKA), and Perl (TDG and CoralReef's ports-based classification module). NeTraMark currently supports Linux platforms and requires the CoralReef [1] software suite, as the `crl_pay` and ports-based classifier rely on it. It supports `pcap` and `DAG` trace files.

We tested the current implementation on a desktop PC with a 2.4 GHz Intel Core 2 Quad Kentzfield Q6600 CPU, 4 GB of memory and Ubuntu 9.10 (kernel ver. 2.6.31-22-generic), and confirmed that it successfully performed benchmark experiments with max 2 GB memory usage, where eleven classifiers are tested against two multi-gigabyte traces collected at a university campus network. The average utilization and the number of flows per 5 minutes of the traces were around 75 Mbps and 92-158 K, respectively. The memory usage depends on the number of flows processed and put into the project DB in a time interval (5 minutes by default) rather than the volume of traffic. We estimate that around 10-12 GB of memory will be required to process a backbone trace whose average number of flows per time interval is $>$ 1 M, though more optimized implementations would likely consume less memory to run, which we leave as future work.

3.4 Usage Examples

In this subsection, we briefly present two examples of using NeTraMark in an operational setting: (i) identifying heavy hitters, and (ii) addressing "unknown" flows with interactive visualization of BLINC and TDG graphs. A more complete outline of uses can be found in the NeTraMark documentation.

ISPs or network operators are often interested in finding hosts or flows which consume most of network resources (e.g., heavy hitters, scanners, spammers, etc.). NeTraMark's user interface makes it easy and convenient for users to quickly identify heavy hitters by providing "Sort" and "Asc/Desc" buttons in the bottom panel (Fig. 2); when they are clicked, flow records are sorted accordingly based on the columns corresponding to the number of packets or bytes transferred.

We also implemented the interactive visualization module of BLINC and TDG graphs in the NeTraMark software suite, so as to allow users to further examine and identify traffic flows whose classification results are "unknown" with the incorporated classification methods. Since BLINC and TDG graphs capture the transport layer communication and social behavior of application protocols, the visualized graphs may provide users with hints in identifying previously "unknown" applications and protocols that fall under the graph-modeled types of BLINC and TDG (e.g., a new P2P protocol or malicious scanning behavior) [13, 11].

Fig. 4. shows a TDG graph of flow-records that were ascribed as *unknown* or *unclassified* by the payload-based and ports-based classifiers, with which users may further identify a few to several nodes with malicious activity such as worm spreads and scanning. Fig. 3 shows an example BLINC graph of a P2P application (*eMule*), which exactly matches with the typical communication behavior graph of P2P applications as explained in [13].

4. COMPARISON WITH RELATED WORK

Recently, Dainotti *et al.* introduced a novel community-oriented traffic classification platform called *Traffic Identification Engine (TIE)*, which is the only work both comparable and complementary to NeTraMark, to the best of our knowledge. TIE's design choices focused on comparison of different classification approaches (i.e., comparability and extensibility), multiple classification (i.e., synergy of combining multiple classifiers' results), and real-time online classification [10]. The current implementation of TIE supports two classification plugins: the ports-based classifier of CoralReef and the payload-based L7-filter classifier [4], with a few more payload-based and flow features-based classification plugins still under development [10].

In contrast, NeTraMark (i) incorporates a richer set of eleven state-of-the-art classification techniques, including host-behavior based classification techniques and their visualization modules as well as flow features-based classifiers in addition to ports and payload-based ones, and (ii) was designed with more emphasis on reproducibility, efficiency, flexibility and ease-of use than real-time classification of online traffic, as online classification does not seem to be of a top priority particularly when benchmarking > 10 classifiers on a broad range of multi-giga/terabytes trace sets collected from different geographic locations with diverse link characteristics and application traffic mix. As a consequence of our design choices, NeTraMark adopts PostgreSQL DBMS back-end for efficient management and reuse of benchmark projects and results, and user-friendly GUI that allows users to easily configure, run, store, and repeat benchmark experiments and visualize the results, none of which are supported by TIE.

5. CONCLUSIONS

We presented an Internet traffic classification benchmark tool, NeTraMark. Based on the six key design guidelines (Comparability, Reproducibility, Efficiency, Extensibility, Synergy, and Flexibility/Ease-of-use), NeTraMark is the first Internet traffic classification benchmark where eleven state-of-the-art traffic classifiers are integrated. NeTraMark also allows researchers and practitioners to easily extend it with new classification algorithms and compare them with other built-in classifiers, in terms of the three categories of performance metrics: Per-whole-trace Accuracy, Per-application Accuracy, and Computational Performance.

In the near future, we plan to extend NeTraMark to support (i) online traffic classification as well as (ii) distributed traffic classification and benchmarking. This will allow network operators and researchers to monitor, collect, and classify traffic and compare the classification results and performance from multiple locations. (iii) We are also developing and integrating an automated application payload signature detection and generation module like LASER [23], which will be available in the next release of NeTraMark.

Releasing the source code to researchers and practitioners,[9] we believe that NeTraMark will lay a foundation for a meaningful traffic classification benchmark and facilitate both the research and operation communities pursuing the direction of scientifically-grounded traffic classification research.

6. ACKNOWLEDGEMENT

We are grateful to Thomas Karagiannis, Marios Iliofotou, Michalis Faloutsos, and Konstantina Papagiannaki for sharing their BLINC and TDG codes with us. We owe special thanks to the editor and anonymous reviewers for their thorough and helpful feedback, which was tremendously vital to significantly improve the contents and presentation of this paper. This work was supported by NAP of Korea Research Council of Fundamental Science and Technology and the ITRC support program [NIPA-2010-C1090-1011-0004] of MKE/NIPA. The ICT at Seoul National University provided research facilities for this study.

7. ADDITIONAL AUTHORS

Yanghee Choi, Seoul National University (`yhchoi@mmlab.snu.ac.kr`).

8. REFERENCES

[1] *CoralReef.* http://www.caida.org/tools/measurement/coralreef/

[2] *WEKA: Data Mining Software in Java.* http://www.cs.waikato.ac.nz/ml/weka/

[3] *Postgresql.* http://www.postgresql.org

[4] *L7-filter, Application Layer Packet Classifier for Linux.* http://l7-filter.sourceforge.net

[5] K. Claffy, H.-W. Braun, and G. C. Polyzos. A parameterizable methodology for internet traffic flow profiling. In IEEE JSAC Special Issue on the Global Internet, Vol. 13, Num. 8, pp. 1481-1494, Oct. 1995.

[9]For access to the shareable benchmark data sets, please refer to http://mmlab.snu.ac.kr/∼hkim/code-data-access.html and/or http://mawi.wide.ad.jp.

[6] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet Traffic Classification Demystified: Myths, Caveats and the Best Practices. In ACM CoNEXT, December 2008.

[7] Y. Lim, H. Kim, J. Jeong, C. Kim, T. Kwon, and Y. Choi. Internet Traffic Classification Demystified: On the Sources of the Discriminative Power. In ACM CoNEXT, December 2010.

[8] G. Szabo, I. Szabo, and D. Orincsay. Accurate Traffic Classification. In IEEE WoWMoM, June 2007.

[9] L. Salgarelli, F. Gringoli, and T. Karagiannis. Comparing Traffic Classifiers. In ACM SIGCOMM CCR, Vol. 37, Num. 3, pp. 65-68, July 2007.

[10] A. Dainotti, W. de Donato, A. Pescape, and G. Ventre. TIE: A Community-Oriented Traffic Classification Platform, In TMA, May 2009.

[11] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese. Network monitoring using traffic dispersion graphs. In ACM SIGCOMM IMC, October 2007.

[12] M. Iliofotou, H. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese. Graph-based P2P Traffic Classification at the Internet Backbone. In IEEE Global Internet Symposium, April 2009.

[13] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel traffic classification in the dark. In ACM SIGCOMM, August 2005.

[14] A. Moore and K. Papagiannaki. Toward the accurate identification of network applications. In PAM, April 2005.

[15] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In PAM, April 2004.

[16] A. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In ACM SIGMETRICS, June 2005.

[17] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classificaton Using Clustering Algorithms. In ACM SIGCOMM MineNet Workshop, September 2006.

[18] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/Realtime Traffic Classification Using Semi-Supervised Learning. In IFIP Performance, October 2007.

[19] J. Erman, A. Mahanti, and M. Arlitt. Byte Me: A Case for Byte Accuracy in Traffic Classification. In ACM SIGMETRICS MineNet Workshop, June 2007.

[20] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. In ACM SIGCOMM CCR, Vol. 36, Num. 5, pp.7-15, October 2006.

[21] I. H. Witten and E. Frank. Data Mining:Practical Machine Leaning Tools and Techniques, 2nd ed. Morgan Kaufmann, 2005.

[22] J. Crowcroft. Net neutrality: the technical side of the debate: a white paper. In ACM SIGCOMM CCR, Vol. 37, Num. 1, pp. 49-55, January 2007.

[23] B. Park, Y. Won, M. Kim, and J. Hong. Towards Automated Application Signature Generation for Traffic Identification. In IEEE/IFIP NOMS, October 2008.