Internet Traffic Classification Demystified: On the Sources of the Discriminative Power

Yeon-sup Lim
University of Massachusetts
Amherst
MA, USA
ylim@cs.umass.edu

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

Jiwoong Jeong
Seoul National University
Seoul, Korea
jwjeong@popeye.snu.ac.kr

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

Ted “Taekyoung" Kwon
Seoul National University
Seoul, Korea
tkkwon@snu.ac.kr

Yanghee Choi
Seoul National University
Seoul, Korea
yhchoi@snu.ac.kr

ABSTRACT

Recent research on Internet traffic classification has yield a number of data mining techniques for distinguishing types of traffic, but no systematic analysis on “Why" some algorithms achieve high accuracies. In pursuit of empirically grounded answers to the “Why" question, which is critical in understanding and establishing a scientific ground for traffic classification research, this paper reveals the three sources of the discriminative power in classifying the Internet application traffic: (i) ports, (ii) the sizes of the first one-two (for UDP flows) or four-five (for TCP flows) packets, and (iii) discretization of those features. We find that C4.5 performs the best under any circumstances, as well as the reason why; because the algorithm discretizes input features during classification operations. We also find that the entropy-based Minimum Description Length discretization on ports and packet size features substantially improve the classification accuracy of every machine learning algorithm tested (by as much as 59.8%!) and make all of them achieve >93% accuracy on average without any algorithm-specific tuning processes. Our results indicate that dealing with the ports and packet size features as discrete nominal intervals, not as continuous numbers, is the essential basis for accurate traffic classification (i.e., the features should be discretized first), regardless of classification algorithms to use.

1. INTRODUCTION

Traffic classification has gained substantial attention within the Internet research and operation community given recent events and struggles over the appropriate use and pricing of the Internet. Accurate and complete traffic classification is an essential task for understanding, operating, optimizing, and financing the Internet as well as planning improvements in future network architectures.

In the early Internet, traffic classification practices largely relied on the use of transport layer port numbers, typically registered with IANA to represent a well-known application. As some increasingly popular applications that support peer-to-peer file sharing hide their identity by assigning ports dynamically and/or masquerading into the well-known ports of other applications, the port-based classification has become less reliable [27, 35, 44]. A more recent and reliable approach adopted by commercial tools [1, 2] inspects packet payloads to find specific string patterns of known applications [22, 28, 35, 44]. While this approach is more accurate, it fails to work on encrypted traffic and entails tremendous privacy and legal concerns which often preclude access to payload data.

More recently, 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 (1) host-level communication behavior-based approach which takes advantage of information regarding “social” interaction of hosts [9, 10, 12, 25, 26], and (2) traffic flow features-based approach, which classifies based on flow duration, the number and size of packets per flow, and inter-packet arrival time [34, 36, 42, 6, 13, 16, 4, 10, 52, 15, 47, 31]. In particular, substantial attention has been invested in data mining techniques and machine learning algorithms using flow features for traffic classification, demonstrating that several different machine learning

ACM CoNEXT 2010, November 30 - December 3, 2010, Philadelphia, USA.

algorithms successfully classify IP traffic flows with > 90∼95% accuracy, when tuned and/or trained properly. Several researchers have published papers where they apply and tune a single machine learning algorithm to classify IP traffic.

Despite many published papers for traffic classification, there are still no clear answers to the following *"Why?"* questions that are critical in understanding and establishing a scientific ground for traffic classification research: *"Why do some machine learning algorithms - even with diverse theoretical or statistical backgrounds and different tuning parameters - achieve such a moderately high level of accuracy (> 90∼95%), while others do not? What is the primary cause of the similar and/or different classification performance of those algorithms? Do the higher accuracies obtained from different algorithms imply a possibility of the existence of "common" sources or techniques of the discriminative power in traffic classification? If so, what are they?"*

In pursuit of the answers related to deeper insight into the nature of the Internet traffic classification problem, we performed a detailed drill-down data analysis to reveal the sources of the discriminative power in traffic classification. We highlight the main contributions from our study:

(a) This paper publicly raises awareness and interest on the above "Why?" questions still under-appreciated by the research community, and then performs an in-depth analysis to derive empirically grounded answers. To this end, we strategically test several commonly used supervised machine learning algorithms (Naive Bayes, Support Vector Machines, k-Nearest Neighbors, and C4.5 Decision Tree, etc.) on a broad range of data sets: five payload traces collected at two backbone and one edge links located in Japan, Korea, and the US. Diverse geographic locations, link characteristics, and application traffic mix in these data allowed us to test the algorithms under a wide variety of conditions.

(b) This is the first work that clearly shows the feasibility of accurate traffic classification at the beginning of *single-directional* TCP and UDP application flows. We found that (i) ports and the sizes of the first one-two packets in a single-directional UDP application (e.g., NTP, DNS, and SMTP) flow provide a sufficient amount of information to accurately identify the causing application, and (ii) ports and the sizes of the first four-five consecutive packets in a single-directional TCP application flow are expressive enough to distinguish among different applications. These findings can be applied to build an early traffic classifier that handles both TCP and UDP as well as edge and backbone traffic, where one direction of a TCP flow may be unobserved due to routing asymmetries.

(c) We also found that classification accuracy decreases when the first few packets are missed; thus, for accurate classification, traffic flows should be captured from the beginning, not in the midst.

(d) We found that C4.5 Decision Tree algorithm consistently achieved the highest accuracy on every trace and application, with 96.7% accuracy on average when used with ports and the sizes of the first five consecutive packets. We also found the reason *Why* the algorithm worked that well; due to its own entropy-based discretization capability, which we will explain in section 5.

(e) We found that the entropy-based discretization technique can be applied to improve the classification accuracies of other machine learning algorithms as well. The entropy-based Minimum Description Length algorithm significantly improved every tested algorithm by as much as 59.8%, making all of them achieve >93% accuracy on average without any algorithm-specific tuning processes. We also found that even the simplest discretization technique called the Equal-Interval-Width significantly improves the classification accuracies of the machine learning algorithms.

(f) Our results indicate that the classification algorithms should deal with ports and packet size features as discrete non-numerical nominal (or categorical) intervals, not as continuous numbers, for accurate traffic classification. Thus, those flow features should be discretized first as a preprocessing step, before classification algorithms run. This means that dealing with those flow features as discrete non-numerical intervals, not as continuous numbers, is the essential basis (i.e., the source of the discriminative power) for accurate early traffic identification, regardless of classification algorithms to use.

The rest of this paper proceeds as follows: After reviewing related work in section 2, we describe our data and methodology in section 3. Section 4 evaluates the discriminative power of traffic flow features, in particular ports and the first few to several consecutive packets of single directional TCP and UDP application flows. Section 5 investigates the impact of discretization on traffic classification performance. We briefly discuss why discretization works well and what it means for traffic classification, concluding this paper in section 6.

2. RELATED WORK

A number of researchers have closely looked at the flow-features based approach typically by applying machine learning techniques to classify traffic, as the three constituent subproblems of traffic classification are essentially identical to the following three main challenges of machine learning-based pattern recognition and classification: (i) key feature selection, (ii) finding the best algorithm(s) for traffic classification, and (iii) obtaining representative data sets for training and testing machine learning algorithms. In this section, we briefly

Table 1: Characteristics of analyzed traces

Set	Date	Day	Start	Duration	Link type	Src.IP	Dst.IP	Packets	Bytes	Avg. Util	Avg. Flows (/5 min.)	Payload
PAIX	2004-02-25	Wed	11:00	2h	Backbone	410K	7465K	250M	91G	104Mbps	1055K	16Bytes
WIDE-I	2006-03-03	Fri	22:45	55m	Backbone	263K	794K	32M	14G	35Mbps	312K	40Bytes
WIDE-II	2007-01-09	Tue	07:45	27h 15m	Backbone	1406K	2154K	1544M	1064G	87Mbps	346K	40Bytes
KAIST-I	2006-09-10	Sun	02:52	48h 12m	edge	148K	227K	711M	506G	24Mbps	19K	40Bytes
KAIST-II	2007-01-09	Tue	19:00	54h 45m	edge	244K	348K	1029M	827G	34Mbps	14K	40Bytes

review previous work along the first two categories of efforts, on which the primary focus of this paper lies.

Key feature selection: finding a set of key traffic features that capture inherently fundamental and distinguishing characteristics of different types of applications. Several papers [18, 34, 36, 42, 13, 4, 6, 10, 52, 15, 31, 46, 47] have worked to determine which set of features work best for classification purposes, sharing the following limitations:

First, they used bidirectional TCP connection statistics, which do not work for UDP application traffic, and for backbone links, which only see both directions of traffic under (atypical) symmetric routing conditions [29]. Erman *et al.* addressed this limitation by proposing an algorithm that uses the packets of an unidirectional TCP flow to estimate the flow statistics of the unobserved opposite direction [16], leaving UDP traffic classification as future work. In this paper, we use a set of single directional flow features only, which will allow an algorithm to classify both TCP and UDP traffic on backbone as well as edge links, as Kim *et al.* did in [29].

Second, they often considered and performed the task of key feature selection only as a pre-processing phase using a single specific technique like Correlation-Based Filter and Consistency-based subset search [46, 47], which outputs only a single bunch of the same features for all applications, not on a per-application basis. In contrast, this paper drills down further to investigate exactly which subset of key features (e.g., which sort of features, which packet size, etc.) are more or less relevant [7] in identifying a specific type of application, providing deeper understanding and insight into the distinguishing characteristics of different applications to both the research and operational communities.

Finding the best algorithm(s) and the reasons "Why": finding the most accurate algorithm(s) that successfully grasp and exploit the discriminative power contained in (a selected set of) key features with acceptable computational cost. Several researchers have published papers based on a single machine learning technique such as K-Means, Support Vector Machines, C4.5 Decision Trees, Bayesian algorithms, Neural Networks, etc. They have often used an off-the-shelf algorithm in a "black-box manner" focusing more heavily on introducing, applying and tuning a specific data mining technique for accurate traffic classification, rather than drilling down to look inside the black-boxed classification processes and interpreting the results to reveal the specific nature of the interrelations between the input variables (e.g., selected set of features and tuned parameters), the used technique/algorithm itself, and the classification results.

Moreover, the results from different research groups are often neither comparable nor reproducible, as every approach has been evaluated using different benchmark traces, typically locally collected but not publicly available, sometimes even without payload (ground truth) [29, 14, 43]. Consequently, there are no definitive answers to our initial "Why" questions raised in the previous section, which we address in this paper by performing various experiments against a broad range of data sets containing a wide variety of link and traffic conditions.

3. METHODOLOGY

This section describes a methodology for identifying sources of the discriminative power in traffic classification, including performance metrics, dataset, establishing reference benchmark (ground truth), and experimental setup for machine learning algorithms.

3.1 Performance metrics

To evaluate the traffic classification performance of machine learning algorithms, we use five metrics: overall accuracy, precision, recall, F-measure, and classification speed:

- Overall accuracy: the ratio of the number of correctly classified traffic flows to the total number of all flows in a given trace. We apply this metric to measure the accuracy of a classifier on the whole trace set. The following three metrics are to evaluate the quality of classification results for each application class.

- Precision: the ratio of True Positives over the sum of True Positives and False Positives or the percentage of flows that are properly attributed to a given application.[1]

[1]True Positives is the number of correctly classified flows, False Positives is the number of flows falsely ascribed to a given application, and False Negatives is the number of flows

- Recall: the ratio of True Positives over the sum of True Positives and False Negatives or the percentage of flows in an application class that are correctly identified.

- F-measure: as a widely-used metric in information retrieval and classification [48], it considers both precision and recall in a single metric by taking their harmonic mean ($\frac{2 \times precision \times recall}{precision + recall}$). We use this metric to measure the per-application classification performance of machine learning algorithms.

- Classification speed: the number of classification decisions performed per second.

3.2 Data set and reference benchmark

Our datasets consisted of five anonymized payload traces collected at two backbone links and one edge link located in Japan, Korea, and the U.S. (Table 1). The PAIX backbone trace was taken on a single directional OC48 link of a US Commercial Tier 1 backbone link connecting San Jose and Seattle. The WIDE traces were captured at a 100 Mbps [2] Ethernet US-Japan Trans-Pacific backbone link that carries commodity traffic for WIDE member organizations. The KAIST traces were captured at one of four external links connecting KAIST campus network and a national research network in Korea at 1 Gb/s [3]

To establish a reference point in evaluating the algorithms, we used the payload-based classifier developed and used in [29, 28, 44, 13, 49]. The resulting classifier includes payload signatures of various popular applications, summarized in Table 2. The payload classification procedure examines the payload contents of each packet against our array of signature strings, and in case of a match, classifies the corresponding flow with an application-specific tag. Previously classified flows are not re-examined again unless they have been classified as HTTP, in which case re-examination may allow identification of non-Web traffic relayed over HTTP (e.g., Streaming, P2P, etc.) [28]. Our experience [29] and Karagiannis *et al.*'s study [27] has suggested that the first 16 bytes of payload suffice for signature-based classification for most legacy and P2P applications except Gnutella [4].

from a given application that are falsely labeled as another application.

[2]The link was upgraded from 100 Mbps to 150 Mbps on June 1, 2007, according to the WIDE MAWI Working Group homepage (http://mawi.wide.ad.jp).

[3]The WIDE-II and KAIST-II traces were collected during the *"A Day in the Life of the Internet (DITL)"* simultaneous Internet data collection event on January 9-10, 2007 [8].

[4]Gnutella (and its variants) uses variable length payload padding; 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 [15].

Table 2: 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	-

3.3 Machine learning experiments

We use the WEKA machine learning software suite [3], often used in traffic classification efforts [29, 13, 17, 34, 36, 37, 46, 47], to perform various experiments with machine learning algorithms . From each trace set in Table 1, we randomly sample 1,000 flows per application (to avoid the class imbalance problem [47, 14]) and then aggregate the sampled flows into a training set. From the remaining flows of each trace set, another 1,000 flows are randomly sampled per application and then merged into a testing set.

3.3.1 Flow features

We use unidirectional flow features of TCP and UDP traffic to build a classifier that handles both TCP and UDP as well as backbone and edge traffic. Throughout this paper, we adopt the definition of a traffic flow based on its 5-tuple (source IP, destination IP, protocol, source port, destination port) with a timeout of 64 seconds [9].

We start with 36 features most of which were inspired from the 248 bidirectional features used in [36, 4], the 22 bidirectional features in [46, 47], and the 37 unidirectional features in [29]. The 36 features include: protocol, source and destination ports, the number of packets, transferred bytes, start time, end time, duration, average packet throughput and byte throughput, the size of the first ten packets, 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).

3.3.2 Machine learning algorithms

In this paper, we choose four machine learning algorithms to evaluate the discriminative power of features and the effect of discretization: Naive Bayes, k-Nearest Neighbors, Support Vector Machines, and C4.5 Decision Tree.

Naive Bayes (NB) [36, 47] is a probabilistic classifier based on Bayes theorem, which analyzes the relationship between each feature and the application class for each instance so as to derive a conditional probability that links feature values and the class. The naive aspect is the assumption that all attributes $(X_1, ..., X_n)$ are conditionally independent of one another, given the class Y. The value of this assumption is that it dramatically simplifies the representation of $P(X|Y)$, and the problem of estimating it from the training data. Despite the naive assumption, Naive Bayes works well in practice, particularly when combined with good feature selection methods. Another strength of the Naive Bayes method is that it produces a probability for each classification which could be used to give a confidence level to each prediction. The Naive Bayes algorithm affords fast, highly scalable model building and classification.

***k*-Nearest Neighbors** (k-NN) [42] computes Euclidean distances from each test instance to the k nearest neighbors in the n-dimensional feature space. The classifier assigns the majority class label among the k nearest neighbors to the tested tuple. This technique scales poorly with the number of training and testing instances, since each new test tuple is compared to every tuple in the training set. We use $k = 1$, by which we obtained the highest overall accuracy after testing with k = 1, 3, 5, 7, 9, 11, 13, 15, 17, and 19.[5]

C4.5 Decision Tree [41] constructs a model based on a tree structure, in which each internal node represents a test on features, each branch an outcome of the test, and each leaf node a class label. In order to use a decision tree for classification, a given tuple (whose class we want to predict) corresponding to flow features, walks through the decision tree from the root to a leaf. The label of the leaf node is the classification result. Decision trees provide us easily interpretable classification rules, which makes the technique favorable.

Support Vector Machines (SVM) [46, 31] refers to a learning system based on recent advances in statistical learning theory. The SVM follows the principle of Structural Risk Minimization that minimizes an upper bound on the expected risk, as opposed to traditional Empirical Risk Minimization that minimizes the error on the training data, so the SVM is known to have a greater ability to generalize which is the goal in statistical leaning. The basic principle of SVM is to construct the optimal separating hyperplane, which maximizes the distance between the closest sample data points in the (reduced) convex hulls for each class, in an n-dimensional feature space [8]. Intuitively, we would expect this boundary between classes to be more generally applicable than others.

We use the Sequential Minimal Optimization (SMO) [38], a faster algorithm for training SVM that uses pairwise classification to break a multi-class problem into a set of 2-dimensional subproblems, eliminating the need for numerical optimization. The two most important parameters in SVM are the complexity parameter C and the polynomial exponent p [46, 31]. Li *et al.* [31] showed that varying the complexity parameter C influenced the overall accuracy of their SVM traffic classifier by only a little (around 1% at most). We use 1 for both parameters as in [46, 48].

[5] k=1 means that the algorithm checks the class of the only nearest (in the n-dimensional feature space) training instance when classifying a testing instance.

4. ON THE DISCRIMINATIVE POWER OF TRAFFIC FLOW FEATURES

In this section, we first measure the discriminative power contained in single directional traffic flow features. Unlike the previous work where they used a single specific technique like Correlation-Based Filter (CFS) [23] and Consistency-based subset search [46, 47], which outputs only a single bunch of the same features for all applications as a pre-processing step, we leverage four machine learning algorithms to further investigate (i) which sort of features possess more discriminative power in classifying traffic and how relevant/powerful they are, (ii) whether the results (i.e., selected subset of features) are consistent across different algorithms and traces (or not), and (iii) which features are most important in identifying a specific type of application. For the experiments, we first group flow features that carry closely related pieces of traffic information, as in Table 3.

Table 3: Specification of Feature Group

Group	Features
PORTS	protocol, srcport, dstport
PKTS	number of packets
BYTES	transferred bytes
DURATION	first time, last time, duration
PKT THRPT	avg. packet throughtput
BYTE THRPT	avg. byte throughtput
PKT SIZE	(max/min/avg/std) packet size, size of the first 10 packets
IPAT	(max/min/avg/std) inter-packet arrival time
TCP FLAG	number of tcp packets with TCP flags set (FIN, SYN, ...)

4.1 Key feature group selection

Figure 1 shows the overall accuracy (averaged over

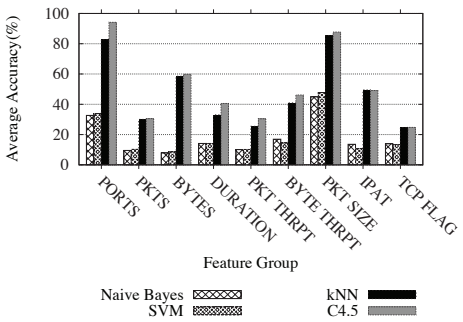


Figure 1: Average overall accuracy by features

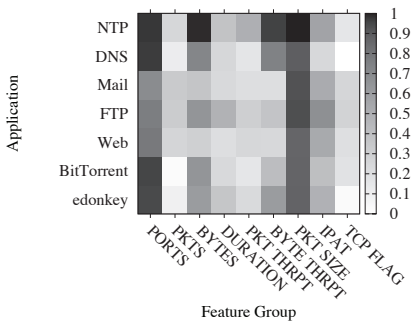


Figure 3: Avg. F-measure by features (k-NN)

all the traces) of the four machine learning algorithms when used with a selected group of flow features. In this figure, two significant observations are obtained: (i) packet size and ports information always yield the highest accuracies across all the algorithms and traces tested. (ii) In particular, they yield an overall accuracy as much as 80∼90% when used with k-NN and C4.5. The two algorithms always significantly outperform other algorithms with every sort of flow features used.

Figure 2 and 3 show the average per-application accuracy (i.e., F-measure) of the two best performed algorithms (C4.5 and k-NN) with each category of features, per selected applications (edonkey, BitTorrent, Web, etc.). In the figures, we observe that almost all applications are identified with >80∼90% of F-measure by ports and packet size information. Interestingly, we find that the "transferred bytes" is also a good feature in identifying UDP flows of NTP, DNS, edonkey and BitTorrent, although it may not be as good as ports and packet size are. Comparing Figure 2 and 3, we find that C4.5 performs notably better than k-NN in accurately identifying Mail, FTP, and Web flows with ports information. This performance gap between the two algorithms is due to the C4.5's entropy-based discretization, which we will explain in section 5.

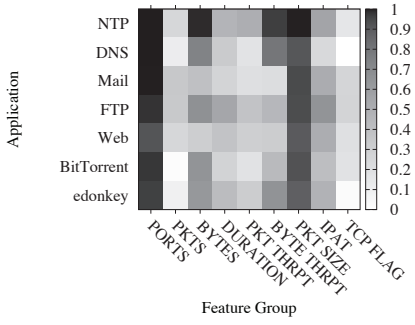


Figure 2: Avg. F-measure by features (C4.5)

4.2 Discriminative power of each packet size feature

In this subsection, we investigate exactly which packet size feature (among those in the PKT SIZE group) has the most or more discriminative power in classifying traffic.

As shown in Figure 4, we find that statistical packet size information (e.g., the max and average packet size) often yields higher accuracy than other individual packet sizes, among which the first (for UDP application flows) and third (for TCP application flows) ones provide the best performance being followed by the fourth, second, and fifth ones. As the max and average packet sizes are obtained only after a flow has finished, these features are not suitable for early traffic classification. Therefore, we focus on the discriminative power contained in the size of the first ten individual packets, targeting early traffic classification with uni-directional flow features only.

Figure 5 and 6 show the average F-measure of C4.5 and k-NN with each individual packet size feature per selected applications. We find that the size of the first packet contributes the most in identifying UDP application flows; C4.5 and k-NN identify NTP and DNS flows with >75∼80% F-measure with the first packet size information only. On the contrary, for (uni-directional) TCP application flows with the three-way handshake, the sizes of the second-sixth (to be more precise, the second packet from a TCP connection receiver - not an initiator -, and the third-sixth packets from both sides) packets are key features to identify the causing applications. This is because applications with TCP finish their connection negotiation phase at the first or second packet in an uni-directional flow thus the following second or third packet typically has an application specific packet size. In the case of Mail and Web flows, the first packet size does not contain any useful information for accurate classification as they all are generated in the pre-defined TCP connection setup phase. For multi-protocol applications like edonkey and BitTorrent, the size of the first packet and the following

second-sixth packets capture distinguishing patterns of their own UDP and TCP flows, respectively. As some FTP flows do not always start with a new TCP connection setup phase (e.g., when using a TCP connection already established more than 64 seconds before), the F-measures on FTP flows with the first packet size is not marked around 0% unlike the cases with Mail and Web.

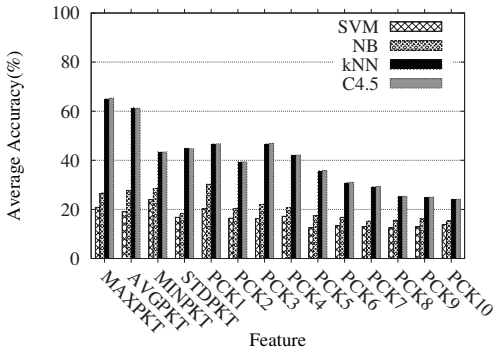


Figure 4: Average overall accuracy by packet size features

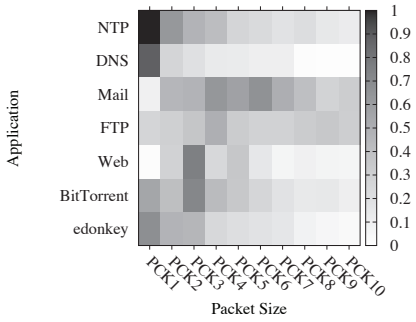


Figure 5: Average F-measure by packet size features (C4.5)

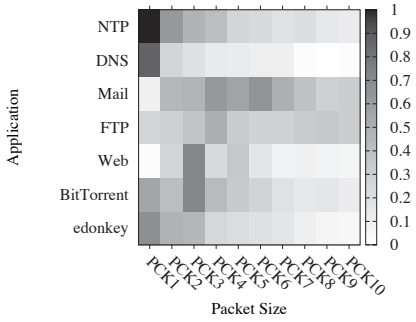


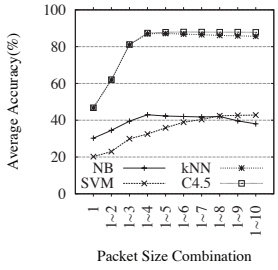
Figure 6: Average F-measure by packet size features (kNN)

4.3 Adaptability to early classification

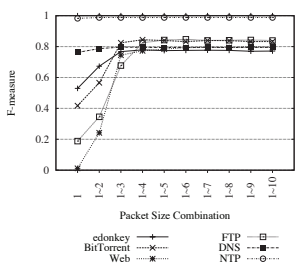
In the previous subsection, we found that (i) the sizes of individual packets in an uni-directional traffic flow carry useful information for distinguishing types of traffic, and (ii) the sizes of the first two packets contribute the most in identifying UDP application flows, while those of the following second-sixth packets do the same for TCP application flows.

In this subsection, we explore the feasibility of early traffic classification using uni-directional flow features only; the sizes of the first n consecutive packets. Figure 7(a) compares the average overall accuracy of the four machine learning algorithms when using the sizes of the first n consecutive packets, varying n from one to ten. Similar to the results in [6] (where/though they used bi-directional TCP connection information), C4.5, k-NN, and Naive Bayes achieve the highest accuracies when used with the sizes of the first four or five packets. The addition of the sixth-tenth packet sizes does not increase classification performance of the algorithms; rather slightly decreases.

Figure 7(b) shows the average F-measure of the best-performed algorithm, C4.5, in the same experiment. Just as we found in the previous subsection 4.2, here we confirmed again that the sizes of the first one-two packets are the most important features for accurate identification of UTP application traffic, while the algorithm needs the sizes of the first four to five packets to accurately classify TCP application traffic.



(a) Avg. overall accuracy



(b) Avg. F-measure (C4.5)

Figure 7: Influence of the number of packets on classification accuracy

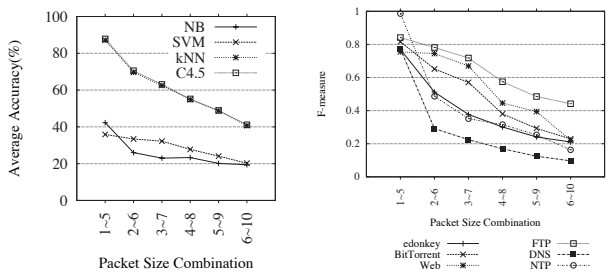
4.4 What if the first few packets are missed?

In the last subsection, we found that the sizes of *the first five consecutive* packets play an important role in classifying uni-directional application traffic. In this subsection, we raise another question; what is the key finding here for accurate traffic classification? *the first five*? or *five consecutive*? What happens if we miss the first few packets? Would the sizes of *any five consecutive packets* provide us good classification performance? To answer the questions, we performed experiments us-

ing the six groups of packet size combination; composed of the sizes of the 1st∼5th, 2nd∼6th, 3rd∼7th, 4th∼8th, 5th∼9th, and 6th∼10th packets.

Figure 8(a) presents the average overall accuracy obtained using each of the six groups. As a group shifts from the 1st∼5th to the 6th∼10th packet sizes, the average accuracy significantly decreases. This indicates that *the first five* packet sizes are the most relevant for accurate classification of uni-directional traffic. In particular, the elimination of the first packet size causes a precipitous decrease in overall accuracy (as much as 20%; due to UDP application traffic).

In order to find the effect of excluding the first few packet sizes on each application, we investigate the average change of F-measure corresponding to selected applications in figure 8(b), which presents the results of the best-performed algorithm, C4.5. As we expected, excluding the first packet size leads to no (for Web) or only a marginal (for FTP) reduction in F-measure for TCP applications. On the other hand, the F-measure for UDP applications such as DNS and NTP drops steeply since their first packet size is the most important feature for accurate identification. For applications with both protocols such as BitTorrent and edonkey, the extent to which the F-measure drops lies between those of the TCP-only and UDP-only applications.



(a) Avg. overall accuracy (b) Avg. F-measure (C4.5)

Figure 8: Classification accuracy when the first n consecutive packets are missed

5. THE IMPACT OF DISCRETIZATION ON CLASSIFICATION PERFORMANCE

In the previous section, we found that C4.5 Decision Tree and k-NN always significantly outperform Naive Bayes and SVM, regardless of the selected flow features and traces. These observations had motivated us to investigate commonalities between the two algorithms. What we found during the investigation was that the algorithms themselves can be used to perform a data preprocessing operation called "Discretization", which will be detailed and tested in this section.

5.1 Discretization

In statistics and machine learning, discretization refers to the process of converting numeric feature values into a number of intervals, and then mapping each interval to a discrete nominal symbol. These discrete nominal symbols are used as new values, replacing the original numeric values of the features. Discretization is defined as follows [45]:

Definition Discretization is a function

$$Q : D \rightarrow C$$

assigning a class $c \in C$ to each value $v \in D$ in the domain of the attribute being discretized. A desirable feature for a practical discretization is:

$$|D| > |C|$$

i.e., a discretized feature has fewer possible values than the original not-discretized one.

Tho choice of the intervals, which is the key process in discretization, can be determined by a domain expert or with the help of an automatic procedure (i.e., discretization algorithm) that makes the task easy. There are two basic approaches to the problem of discretization: One is to quantize each feature in the absence of any knowledge of the classes of the instances in the training set - so-called *unsupervised* discretization. The other is to take the classes into account when discretizing - *supervised* discretization. The former approach, e.g., the simplest *Equal Interval Width* or *Equal Frequency Intervals* discretization, is the only possibility when dealing with clustering problems in which the classes are unknown or nonexistent [48].

As a supervised one, k-NN can be used for the purpose of discretization as follows [50]: Given a feature value, the algorithm tries to estimate which class the value most likely belongs to. The algorithm places a border between two values x_i and x_{i+1} if the estimate is different for them. The estimate is based on the assumption that the most probable class is the most common class among the k nearest examples.

C4.5 can also be used as a supervised discretization method, as the algorithm itself deals with numeric and continuous features using an entropy-based discretization technique [30], by which the algorithm can avoid creating too many branches for a node. The C4.5 uses information gain-ratio [48], an entropy-based metric, to find a most informative border to split the value domain of the continuous feature and then determine the partitions for discrete intervals in the feature space. The algorithm constructs an inductive decision tree by adopting and revising the information gain heuristic used in ID3 [39] that selects a feature if its branching results in the overall minimum entropy at the next layer of the decision tree; the feature with the highest normalized information gain is the one used to make the split de-

cision [19] [6]. The algorithm then recurs on the smaller sublists and binarizes a range at every split, until a stopping criterion is satisfied; e.g., when every leaf node is pure (all instances in the node belong to one class), it stops. The condition can be relaxed when needed [32]. Given space limitations, interested readers are referred to [40, 48, 32] for more details.

Based on (i) the results in the previous section 4 and (ii) the fact that both of the two consistently best-performed algorithms, C4.5 and k-NN, are capable of discretizing input flow features, we had come to hypothesize that the discretization, a general-purpose data preprocessing technique (thus also can be applied to other machine learning algorithms like Naive Bayes and SVM as well), may be the common source of the high classification accuracies of the algorithms. The hypothesis turned out to be true, as we will show in this section.

5.2 The entropy-based discretization with the Minimum Description Length Principle

To test the hypothesis, we conduct the following experiments: We (i) first preprocess our dataset with the Entropy-based discretization method with the Minimum Description Length criterion (Ent-MDL) [20], (ii) apply machine learning algorithms on the discretized dataset, and (iii) then compare their performance with those of the same algorithms with the original non-discretized data.

For the experiments, we choose the Ent-MDL, the default discretization algorithm implemented in WEKA, as it is one of the best general techniques for supervised discretization [48, 30, 32]. Though both the C4.5 discretization and Ent-MDL are entropy-based methods, the latter employs a top-down stopping criterion based on the Minimum Description Length Principle[7], one of the best ways to stop the entropy-based splitting discretization procedure [48], while applying C4.5 to a single feature builds a complete tree for that feature and then applies pruning to find an appropriate number of nodes in the tree (i.e., the number of discretization intervals) in a bottom-up approach [30]. Given space limitations, readers are referred to [20, 21] for more details on the Ent-MDL algorithm and the MDL principle.

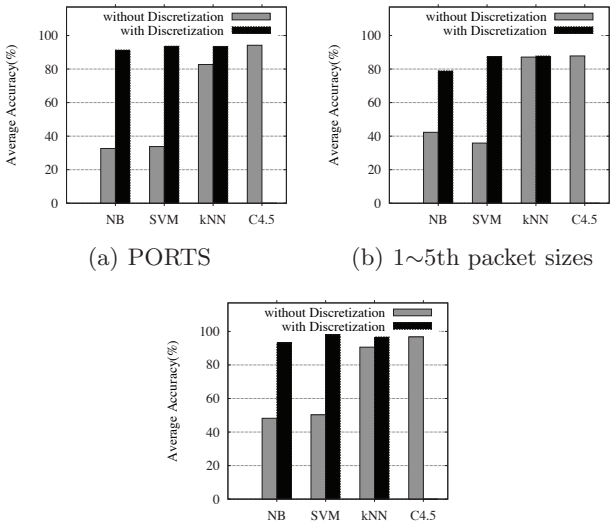
[6]Fayyad *et al.* showed that the maximum information gain by the heuristic is always achieved at a cut point (e.g., the mid-point) between the values taken by two examples of different classes [19].

[7]The Minimum Description Length or MDL principle takes the stance that the best theory for a body of data is one that minimizes the size of the theory plus the amount of information necessary to specify the exceptions relative to the theory. According to the MDL principle, the best generalization is the one that minimizes the number of bits required to communicate the generalization, along with the examples from which it was made [48]. In this discretization case, if we do split, the "theory" is the splitting point, and we are comparing the situation in which we split with that in which we do not.

5.3 Accuracy results

In this subsection, we evaluate the performance of Naive Bayes, SVM, and k-NN when used (i) with and (ii) without the Ent-MDL discretization preprocessing. As the C4.5 performs the entropy-based discretization (similar to the Ent-MDL except the stopping criterion for the number of intervals) itself, the Ent-MDL discretization does not have to be applied for the algorithm[8]. As we found that ports and the sizes of the first five packets are the most important features in the previous section 4, we conduct the experiments using only those features in this subsection, for brevity.

Figure 9(a) shows the average overall accuracy of the Naive Bayes, SVM, and k-NN when used with and without the Ent-MDL discretization on (a) ports, (b) the sizes of the first five packets, and (c) the both features, along with that of C4.5.



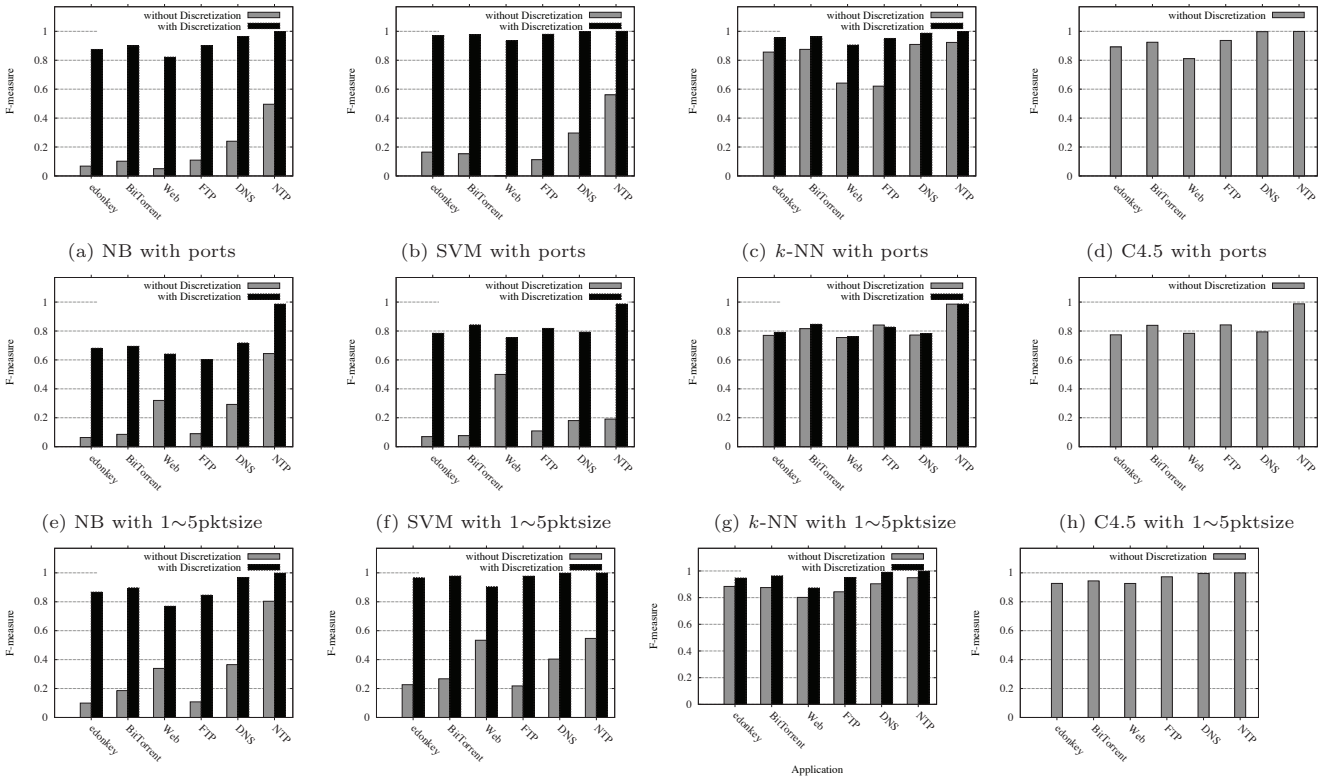
(a) PORTS (b) 1∼5th packet sizes

(c) PORTS and 1∼5th packet sizes

Figure 9: Average overall accuracy with and without the Ent-MDL discretization

To our great surprise, the Ent-MDL discretization preprocess significantly improved the overall accuracy of the Naive Bayes and SVM by (a) 58.7% and 59.8% with ports, (b) 36.7% and 51.5% with the packet size features, and (c) 45.0% and 47.8% with the both features, respectively. In particular, the SVM performed the best when both ports and packet size features are discretized, achieving >98.0% average overall accuracy, which was 1.3% higher than that of the C4.5 with the

[8]According to our results, the Ent-MDL discretization preprocess rather slightly worsens the accuracy of the C4.5; by 1∼2% across all the tested traces. The reasoning is beyond the scope of this paper.



(a) NB with ports (b) SVM with ports (c) k-NN with ports (d) C4.5 with ports

(e) NB with 1∼5pktsize (f) SVM with 1∼5pktsize (g) k-NN with 1∼5pktsize (h) C4.5 with 1∼5pktsize

(i) NB with the both features (j) SVM with the both features (k) k-NN with the both features (l) C4.5 with the both features

Figure 10: Average F-measure with and without the Ent-MDL discretization

same features. The entropy-based discretization also improved the overall accuracy of k-NN by 12% with ports, while only marginally with packet size features (0.7%).

Consequently, all the tested algorithms achieved >93.2% average overall accuracy when used with the entropy-based discretization technique (either Ent-MDL or C4.5), *without any additional algorithm-specific tuning processes.* These results indicate that the entropy-based discretization is (i) a key essential technique for accurate traffic classification, which makes all the tested algorithms achieve high and similar accuracy (93.2∼98.0%), and (ii) more effective on ports than packet size information, as shown in both Figure 9 and 10[9].

5.4 Classification speed

In this subsection, we measure the classification speeds (i.e., the average number of classifications performed per second, which is particularly important when considering real-time classification) of the tested algorithms when used with and without the Ent-MDL discretization, on ports and the sizes of the first five packets. Tests were performed on a dedicated PC server with a 2.83 GHz Quad-core CPU and 8 GB RAM. Note that we have evaluated the performance of concrete implementations in the Java-based (slow) WEKA [3] software suite on our test platform, not the theoretical complexity of the algorithms because (i) traffic classification efforts [29, 34, 17, 13, 37, 36, 47] have often used WEKA, and (ii) this approach yields tangible performance numbers for comparisons [47, 29]. Optimized implementations would likely yield faster classification speeds for all algorithms.

Figure 11 shows the results averaged over all the traces. C4.5 is the fastest (60,178 classifications/sec.),

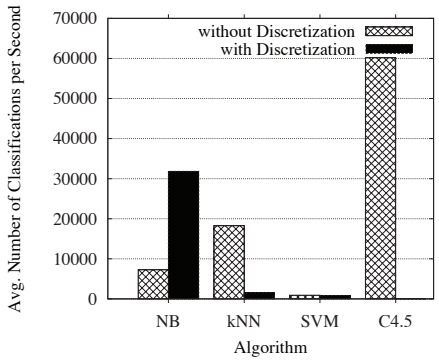


Figure 11: Average classification speed with and without discretization

followed by Naive Bayes with the Ent-MDL discretiza-

[9]We omit more detailed explanation on the F-measure Figure 10 due to lack of space.

tion (31,826 classifications/sec.), and SVM without the discretization (around 18,259 classifications/sec.). SVM with the discretization, k-NN with and without the discretization were the slowest three, with around 1,537, 914, and 849 classifications/sec, respectively.

Whereas the Ent-MDL discretization improved the accuracies of all the tested algorithms, it affects the classification speeds of the algorithms differently; it significantly improves the classification speed of the Naive Bayes algorithm by 438%, while the SVM and k-NN show a precipitous (91.4%) and moderate (7.1%) drop in their classification speeds with it, respectively.

6. CONCLUSIONS: WHY DISCRETIZATION?

We conducted a detailed analysis to identify the three sources (i.e., ports, the first few packet sizes, and discretization of those features) of discriminative power in classifying single-directional application traffic. To the best of our knowledge, this is the first work that clearly shows the feasibility of accurate traffic classification at the beginning of both (i) *single-directional* TCP (with ports and the sizes of the first five packets) and (ii) UDP application flows (with ports and the sizes of the first one or two packets), which can be applied to classify online backbone traffic where one direction of a TCP flow may be unobserved due to routing asymmetries. We also showed that classification accuracy decreases when the first few packets are missed; thus, for accurate classification, application traffic flows should be captured from the beginning, not in the midst.

In section 5, we found that the entropy-based discretization (both Ent-MDL or C4.5) on ports and the sizes of the first five packets substantially improves the classification accuracies of all the tested machine learning algorithms without any algorithm-specific tuning processes. We also obtained the same results with other machine learning algorithms like Bayesian Networks and Neural Networks, though we omitted the results due to lack of space. These consistent results indicate that the classification algorithms should deal with ports and packet size features as discrete non-numerical nominal (or categorical) intervals, not as continuous numbers, for accurate traffic classification; which practically makes sense, since the port "80" for WWW traffic is rather a label, not a number. The same applies to the packet size features as well; as many applications are programmed to have often-used specific (range of) sizes for their first few packets, the (range of) sizes should be dealt with as a label for identification, not as a number.

To our surprise, according to our additional experiments which this margin is too small to contain, even the simplest discretization method like the Equal-Interval-Width on the packet sizes feature improved the performance of Naive Bayes and SVM (achieving around 70-80% average overall accuracy), though not as much as the entropy-based methods do; which also implies that *dealing with those traffic flow features as discrete non-numerical intervals is the essential basis or technique for accurate traffic identification, regardless of classification algorithms to use.*

Discretization has already been shown to work well for Naive Bayes and SVM in multiple areas of various pattern classification problems, since it does not make assumptions about the form of the probability distribution from which the quantitative feature values were drawn, particularly for Naive Bayes [24, 11, 51, 33, 5]. This paper empirically confirms that the entropy-based discretization (and even the simplest Equal-Interval-Width discretization, though the results are omitted due to lack of space) does the same for traffic classification as well, with ports and the sizes of the first few consecutive packets in a single-directional flow.

7. ACKNOWLEDGMENTS

We are grateful to kc claffy, Marina Fomenkov, and KiYoung Lee for having motivated us to pursue this study. Our special thanks to Kenjiro Cho, Kensuke Fukuda and Sue Moon for their excellent feedback and useful discussions. 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.

8. REFERENCES

[1] Ellacoya, http://www.ellacoya.com.
[2] Packeteer, http://www.packeteer.com.
[3] Weka 3: Data Mining Software in Java `http://www.cs.waikato.ac.nz/ml/weka/`.
[4] T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on Neural Networks*, 18(1):223–239, 2007.
[5] I. Babaoğlu, O. Findik, and E. Ülker. Effects of discretization on determination of coronary artery disease using support vector machine. In *Proceedings of ICIS*. ACM, 2009.
[6] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *Proceedings of ACM CoNEXT*, 2006.
[7] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, 1997.
[8] CAIDA. A day in the life of the internet `http://www.caida.org/projects/ditl/`.
[9] K. Claffy, H.-W. Braun, and G. Polyzos. A parameterizable methodology for internet traffic flow profiling. *Selected Areas in Communications, IEEE Journal on*, 13(8):1481–1494, Oct. 1995.
[10] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli. Traffic classification through simple statistical fingerprinting. *SIGCOMM Comput. Commun. Rev.*, 37(1):5–16, 2007.
[11] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of ICML*. Morgan Kaufmann Publishers Inc., 1995.
[12] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. Dynamic application-layer protocol analysis for

network intrusion detection. In *Proceedings of USENIX-SS*, 2006.

[13] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of ACM MineNet*, 2006.

[14] J. Erman, A. Mahanti, and M. Arlitt. Byte me: a case for byte accuracy in traffic classification. In *Proceedings of ACM MineNet*, 2007.

[15] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Perform. Eval.*, 64(9-12):1194–1213, 2007.

[16] 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 WWW*. ACM, 2007.

[17] J. Erman, A. Mahanti, and M. F. Arlitt. Internet traffic identification using machine learning. In *Proceedings of IEEE GLOBECOM*, Nov. 2006.

[18] A. Este, F. Gringoli, and L. Salgarelli. On the stability of the information carried by traffic flow features at the packet level. *SIGCOMM Comput. Commun. Rev.*, 39(3):13–18, 2009.

[19] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.

[20] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Uncertainty in Articial Intelligence*, 1993.

[21] P. Grunwald. A tutorial introduction to the minimum description length principle. In *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

[22] P. Haffner, S. Sen, O. Spatscheck, and D. Wang. Acas: automated construction of application signatures. In *Proceedings of ACM MineNet*, 2005.

[23] M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of ICML*. Morgan Kaufmann Publishers Inc., 2000.

[24] C.-N. Hsu, H.-J. Huang, and T.-T. Wong. Why discretization works for naive bayesian classifiers. In *Proceedings of ICML*. Morgan Kaufmann Publishers Inc., 2000.

[25] M. Iliofotou, H.-c. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, and G. Varghese. Graph-based p2p traffic classification at the internet backbone. In *Proceedings of IEEE INFOCOM*, 2009.

[26] Y. Jin, E. Sharafuddin, and Z.-L. Zhang. Unveiling core network-wide communication patterns through application traffic activity graph decomposition. In *Proceedings of ACM SIGMETRICS*, 2009.

[27] T. Karagiannis, A. Broido, M. Faloutsos, and K. claffy. Transport layer identification of p2p traffic. In *Proceedings of IMC*. ACM, 2004.

[28] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. *SIGCOMM Comput. Commun. Rev.*, 35(4):229–240, 2005.

[29] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of ACM CoNEXT*, 2008.

[30] R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 114–119. AAAI Press, 1996.

[31] Z. Li, R. Yuan, and X. Guan. Accurate classification of the internet traffic based on the svm method. In *Proceedings of IEEE ICC*, June 2007.

[32] H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, Oct. 2002.

[33] Y. Liu, Z. Li, S. Guo, and T. Feng. Efficient, accurate internet traffic classification using discretization in naive bayes. In *Proceedings of IEEE ICNSC*, 2008.

[34] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In *Proceedings of PAM*, 2004.

[35] A. W. Moore and K. Papagiannaki. Toward the accurate identification of network applications. In *Proceedings of PAM*, Mar. 2005.

[36] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of ACM SIGMETRICS*, 2005.

[37] T. Nguyen and G. Armitage. Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks. In *Proceedings of IEEE LCN*, 2006.

[38] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.

[39] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[40] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[41] J. R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[42] 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 IMC*. ACM, 2004.

[43] L. Salgarelli, F. Gringoli, and T. Karagiannis. Comparing traffic classifiers. *SIGCOMM Comput. Commun. Rev.*, 37(3):65–68, 2007.

[44] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of WWW*. ACM, 2004.

[45] S. Techn and K. M. Risvik. Discretization of numerical attributes - preprocessing for machine learning.

[46] N. Williams, S. Zander, and G. Armitage. Evaluating Machine Learning Algorithms for Automated Network Application Identification. Technical Report 060410B, Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Apr. 2006.

[47] N. Williams, S. Zander, and G. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, 36(5):5–16, 2006.

[48] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, June 2005.

[49] Y. J. Won, B.-C. Park, H.-T. Ju, M.-S. Kim, and J. W. Hong. A hybrid approach for accurate application traffic identification. In *IEEE/IFIP E2EMON*, Apr. 2006.

[50] X. Wu. A bayesian discretizer for real-valued attributes. *The Computer Journal*, 39:688–691, 1996.

[51] Y. Yang and G. I. Webb. On why discretization works for naive-bayes classifiers. In *Australian Joint Conference on AI*, 2003.

[52] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *Proceedings of IEEE LCN*, 2005.