Timely and Continuous Machine-Learning-Based Classification for Interactive IP Traffic

Thuy T. T. Nguyen, Grenville Armitage, *Member, IEEE, ACM*, Philip Branch, and Sebastian Zander

***Abstract*—Machine Learning (ML) for classifying IP traffic has relied on the analysis of statistics of full flows or their first few packets only. However, automated QoS management for interactive traffic flows requires quick and timely classification well before the flows finish. Also, interactive flows are often long-lived and should be continuously monitored during their lifetime. We propose to achieve this by using statistics derived from sub-flows—a small number of most recent packets taken at any point in a flow's lifetime. Then, the ML classifier must be trained on a set of sub-flows, and we investigate different sub-flow selection strategies. We also propose to augment training datasets so that classification accuracy is maintained even when a classifier mixes up client-to-server and server-to-client directions for applications exhibiting asymmetric traffic characteristics. We demonstrate the effectiveness of our approach with the Naive Bayes and C4.5 Decision Tree ML algorithms, for the identification of first-person-shooter online game and VoIP traffic. Our results show that we can classify both applications with up to 99% Precision and 95% Recall within less than 1 s. Stable results are achieved regardless of where within a flow the classifier captures the packets and the traffic direction.**

***Index Terms*—Interactive traffic, Machine Learning (ML), sub-flows, traffic classification.**

I. Introduction

FINDING viable solutions for a QoS-enabled Internet has attracted considerable research effort since the early 1990s. A key component of many QoS schemes has been the provision and management of robust, automated, and real-time IP traffic classification (IPTC). IPTC can serve as a core part of an automated QoS-enabled Internet, assist the QoS signaling process by identifying the traffic of interest quickly and automatically (without requiring human intervention), and trigger an automated QoS control system to allocate network resources for priority applications [1]. It also has the potential to support class-based QoS accounting and billing, assist in automated intrusion detection [2], and play an important role in lawful interception [3].

Commonly deployed IPTC involves inspection of packets' contents or TCP/UDP port numbers. Yet the effectiveness of such techniques is diminishing. Regular updates are required to track minor changes in applications' packet payload formats, applications obfuscate packet contents through encryption, and government privacy regulations may constrain the ability of third parties to lawfully inspect payloads. The research community has responded with IPTC based on statistical patterns in externally observable attributes of traffic, and in particular the application of Machine Learning (ML) techniques. Attributes of flows (such as packet lengths or packet interarrival times) are known as *features*. Classification involves two stages—training the ML algorithm to associate sets of features with known traffic classes (creating "rules"), and applying these rules to classify unknown traffic.

Most published research focused on the efficacy of different ML algorithms when trained and tested over full flows, which in the case of long-lived interactive flows have thousands of packets. Yet in real networks with automated QoS control, IPTC must reach decisions quickly (well before flows stop). Some newer work investigated classification using only the first few packets of flows. However, it is not sufficient as, for example, malicious attacks might disguise themselves with the statistical properties of a trusted application early in their flow's lifetime. Alternatively, the classifier itself might have been started (or restarted) while hundreds or thousands of flows were already active through a network monitoring point (thereby missing the starts of these active flows). Consequently, each flow should ideally be classified correctly and continuously throughout its lifetime.

Another issue not addressed by previous work is that *directionality* has been an implicit attribute of features. Usually, application flows are defined as bidirectional, and features used for training are calculated separately in *forward* and *reverse* directions (the forward direction typically indicated by the first packet of a flow is commonly client-to-server). This approach works well if during subsequent operations the classifier always sees the first packet of every flow and calculates features with the correct sense of direction. However, a real-world classifier may not see the first packet because of packet sampling, high load, or missed flow starts (such as a classifier starting or rebooting while long-lived interactive flows are already in progress). Since, for many applications, traffic characteristics are asymmetric in both directions, this can lead to degraded classification accuracy.

We present and evaluate novel modifications to traditional ML techniques that optimize the timely and continuous classification of long-lived interactive flows and can be deployed in operational IP networks with limited physical resources.

We propose to operate an ML classifier using a sliding window over each flow; the classifier uses no more than N consecutive packets of a flow at any given time, called a *sub-flow*. The ML classifier is trained using sets of features

Manuscript received May 05, 2011; revised November 15, 2011; accepted January 29, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P. Crowley. Date of publication March 05, 2012; date of current version December 13, 2012. The DIFFUSE system was developed in part with support from a grant by the Cisco University Research Program Fund at Community Foundation Silicon Valley.

The authors are with the Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Vic. 3122, Australia (e-mail: tnguyen@swin.edu.au; garmitage@swin.edu.au; pbranch@swin.edu.au; szander@swin.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2012.2187305

calculated from multiple sub-flows at different points within the original application flow's lifetime. Training on multiple sub-flows allows the classifier to quickly identify an application at any point of a flow's lifetime. In general, N is chosen based on an upper bound on the time allowed for classifying a flow or memory limitations in a classifier's implementation.

To overcome the directionality issue, we propose to augment the training dataset by synthesizing a complementary version of every sub-flow with forward and backward features swapped, called "Synthetic Sub-flow Pairs" (SSP). The first packet of a flow captured by the classifier can then represent client-to-server (C–S) or server-to-client (S–C) traffic. SSP trains the classifier to recognize the application either way.

Finally, to train a classifier, we can automate the selection of a limited number of representative sub-flows that best capture the statistical variations of the full flows. Besides manual hand-picked sub-flows and simple random sampling, we make use of an unsupervised clustering ML technique to automate the selection process, called "Assistance of Clustering Techniques" (ACT).

We demonstrate the effectiveness of our proposed approach with two different classes of network delay-intolerant applications that can benefit greatly from a QoS-enabled network: a first-person-shooter (FPS) game (*Wolfenstein: Enemy Territory*, or ET [4]), and VoIP traffic (G.711 and GSM codecs). The traffic characteristics of ET can change over a flow's lifetime and are asymmetric in the C–S and S–C directions. The VoIP traffic is more stable and symmetric. Our results show up to 99% Precision and 95% Recall for both applications, using sub-flows of 25 packets. Stable results are achieved regardless of where within a flow the classifier captures the packets and the traffic direction. Recently published works showed that our proposed approach also works with Skype [5] and BitTorrent traffic [6], confirming that it can benefit the identification of other applications traffic.

The core elements of our proposal (training on multiple sub-flows, SSP, and ACT) were first articulated in [7]–[9], respectively. This paper now integrates and significantly expands on our previous work. Not only are we analyzing another application class besides FPS game; we also provide a more detailed analysis encompassing two different ML algorithms. For both, we show distinct improvements in classification accuracy and timeliness, which suggests our approach is applicable to different ML algorithms. We also conduct a preliminary investigation on the impacts of packet loss on the classification accuracy for both applications. Finally, we analyze the performance of a real-world implementation of our approach that works on PC-based routers, bridges, or passive monitors (called DIFFUSE) [10].

Our paper is organized as follows. Section II highlights significant related work and open issues. Section III introduces our proposed approach. Section IV describes our experimental methods, with our results analyzed in Section V. Section VI analyzes the effects of packet loss. Section VII presents results achieved with DIFFUSE. Section VIII discusses our findings before we conclude and outline future work in Section IX.

II. Related Work and Problem Statement

The application of ML techniques to IPTC has attracted significant attention over the past few years. McGregor *et al.* [11] applied ML in IPTC using the Expectation Maximization (EM) algorithm. The approach clusters traffic with similar observable properties into different application types. The algorithm was found to separate traffic into a small number of basic classes. Roughan *et al.* [12] used the nearest neighbors and Linear/Quadratic Discriminant Analysis ML algorithms to map different network applications to predetermined QoS traffic classes. Zander *et al.* [13] used AutoClass, an unsupervised Bayesian classifier, to determine the best cluster set from the training data. They studied a range of applications and showed that some separation between them can be achieved.

Moore and Zuev [14] proposed applying the supervised Naive Bayes technique to categorize Internet traffic by application. The work was extended with the application of a Bayesian neural network approach in [15]. Bernaille *et al.* [16] used Simple K-Means to classify different types of TCP-based applications using the first few packets of traffic flows. Following the same path of early traffic classification, Sena and Belzarena [17] and Li *et al.* [18] investigated the use of the first packets' statistical properties in both directions of a flow as features. Li and Moore analyzed the theoretical complexity of an ML-based classification system [19]. Park *et al.* [20] proposed the use of a feature selection technique based on a genetic algorithm.

Erman *et al.* [21] addressed the challenge of traffic classification at the core of the network, where the available information about the flows and their contributors might be limited. Their work in [22] and that of Rotsos *et al.* [23] investigated semi-supervised traffic classification approaches, which combine unsupervised and supervised methods. Teja *et al.* [24] looked at the classification of network traffic using a combination of ML methods. Crotti *et al.* [25] proposed an optimization mechanism that allows tuning a number of classification parameters to optimize the classifier's performance according to its environment. A number of other works that compared and evaluated different ML algorithms in IPTC include [26]–[30]. Palmieri and Fiore [31] noted the nonstationarity in traffic's statistical characteristics over time and proposed using recurrent phenomena and hidden nonstationary transition patterns in the time series for traffic classification.

Due to space limitation, we cannot list all related work here. More comprehensive surveys for Internet traffic classification using ML can be found in [32] and [33].

Most published research evaluated the effectiveness of different ML algorithms when trained and classified over full flows consisting of thousands of packets. The efficacy of ML classifiers has not been explored when they have access to only a subset of a flow's packets or they do not see the start of every flow (even the recent work in [16]–[18] assumes the initial packets of flows are available for classification).

Application flows are assumed to be bidirectional, and the application's statistical features are calculated separately in the forward and reverse directions (e.g., [11], [13], [14], [16]–[18], [30], and [31]). This can lead to the directionality issue discussed earlier. In real networks, traffic classifiers may not see the actual start of a flow and have no knowledge of where clients or servers are actually located, thus they cannot be sure in which direction the first packet they see (of any new bidirectional flow) is heading. This can lead to degraded classification performance.

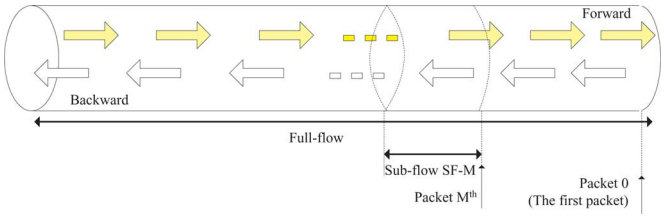

Fig. 1. Illustration of the sub-flow definition.

III. Proposed Approach

To achieve timely and continuous traffic identification, we propose to perform classification over a small sliding window of the most recent N packets of a bidirectional flow (for some small value of N). We define S as the step-width of the sliding window. We define a sub-flow as a window of N packets of a flow starting at packet number M of the flow (see Fig. 1). A sub-flow starting at position M is referred to as SF-M.

The selection of N is a tradeoff between timeliness (the smaller N, the quicker the initial classification) and accuracy (the larger N, the better the accuracy). The selection of S determines the reclassification frequency and resources needed (the smaller S, the more frequently we reclassify, hence the more CPU/memory may be needed).

Our sub-flow approach also can help to reduce memory requirements compared to full-flow analysis, particularly on high-speed networks where a classifier may be observing (tens of) thousands of concurrent flows. Computing features over full flows often may require $O(T)$ memory complexity (where T is the total number of packets of a flow), although some features may be computed with only $O(1)$ memory complexity. Features computed over sub-flows have at most $O(N)$ memory complexity ($N \ll T$) and at best only require $O(1)$ memory complexity as well (although this may require nonoverlapping windows, i.e., $S = N$).

The following steps describe how sub-flows are identified for training/operation and selected for training.

A. Step 1: Sub-Flow Identification

This step identifies all possible sub-flows to train the classifier or during operation. Automated identification of sub-flows may be achieved as follows.

- Pick a window size N and step fraction S (e.g., $\frac{1}{2}N$).
- Starting at the first packet of a flow, slide across the packets in steps of S packets, creating sub-flows of N consecutive packets each.

B. Step 2: Sub-Flow Selection

Training the classifier using all sub-flows identified in Step 1 may require too much processing overhead. Sub-flow selection reduces the load on the classifier, both during training and actual operation (smaller dataset for training and simpler model for classification). It selects only a limited number of representative sub-flows for training. Ideally, these are the sub-flows that best capture the distinctive statistical variation of the full flows during their lifetime.

This step can be done manually based on expert knowledge for well-understood traffic statistical properties. For example, the initial game setup for ET looks quite different to the traffic during a game, hence sub-flows can be taken at the beginning and middle of flows. However, for a new application, this step should be done automatically.

In the earlier part of this paper, we describe a method to automate this step through the use of clustering ML techniques. An unsupervised clustering algorithm identifies "natural" clusters among the initial set of sub-flows from Step 1. A cluster contains members of different sub-flows that have similar statistical characteristics. From each cluster, *representative* members can be used to train the classifier instead of using the whole cluster's population. DIFFUSE, described in Section VII, shows that stratified random sub-flow selection also demonstrates good results.

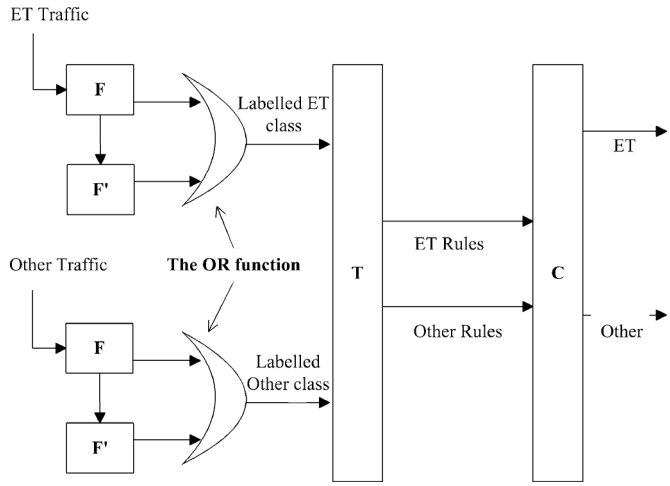

Fig. 2. Steps in training an ML classifier with/without SSP. F are the sub-flow features, and F′ are the mirror-image sub-flow features.

	Mean forward packet length	*Mean backward packet length*	...	*Mean forward packet inter-arrival time*	*Mean backward packet inter-arrival time*
Sub-flow instance X	L_F	L_B	...	I_F	I_B
Mirror-image replica of sub-flow instance X	L_B	L_F	...	I_B	I_F

Fig. 3. SSP creates a mirror-image replica for a sub-flow instance.

C. Step 3: Synthetic Sub-Flow Pairs

Each sub-flow selected in Step 2 results in a set of feature values derived from the sub-flow's N packets (with the forward direction identified as either C–S or S–C). The key steps of feature calculation (F), training (T), and classification (C) are illustrated in Fig. 2, using ET as the application of interest. Each sub-flow's instance then is labeled as either ET or Other class before training the ML classifier. The output of the training is the classification rules to identify ET and Other traffic.

With SSP, a synthesized complimentary version (called *mirror-image replica*) of each sub-flow instance is created in a separate step called F′ by swapping the feature values from forward to backward direction, and vice versa. Fig. 3 illustrates this process for a sub-flow instance X. Consider L_{F} and L_{B} to be the mean forward and backward packet lengths (respectively) of X. The mirror-image replica of X is assigned mean forward and backward packet lengths of L_{B} and L_{F},

respectively. The same mirroring is repeated for other features of X.

There are two options for training and classifying.

1) The sub-flow instances and their mirror-image replicas are labeled as the same class. For example, ET instances and their mirror-image replicas are both labeled as ET class. This trains the classifier such that a new flow that has traffic characteristics similar to either ET *or* its mirror-image replica will be classified as ET traffic.
2) Alternatively, sub-flow instances and their mirror-image replicas are labeled as two separate classes. For example, ET instances are labeled as ET class, and their mirror-image replicas are labeled as ET′ class. This trains the classifier to identify ET, ET′, Other, and Other′ classes separately. Then, a new flow that is classified as ET or ET′ will be classified as ET traffic.

Both approaches significantly improve a classifier's performance compared to training with one explicit definition of flow direction. For our datasets, option 2 provides slightly better overall accuracy (the improvement for Naive Bayes is greater than for C4.5), but lower computational performance (longer model build time and slower classification speed). Considering overall accuracy and computational performance, we chose to use option 1 in this paper.

Note that when a classifier knows the server(s), an alternative to SSP is to group packets into flows using that knowledge (i.e., regardless of a packet's direction, the server IP always becomes the destination IP address in the flow table). However, in practice, often the server(s) may not be known. Furthermore, this approach also requires to update the classifier every time server updates occur.

IV. Experimental Approach

A. Machine Learning Algorithms

There exists a number of supervised ML classification algorithms. We selected two algorithms with different internal training and classification mechanisms: C4.5 Decision Tree (C4.5) [34] and Naive Bayes (NBayes) [35]. Demonstrating our proposal with both reveals benefits in either case, suggesting our approach is applicable to more than just one particular type of ML algorithm. Finding the *optimal* ML algorithm is the subject of future work.

Williams *et al.* [30] compared five ML algorithms for practical IPTC and showed that given the same features and traffic flows, different ML algorithms (including NBayes and C4.5) provided similar classification accuracy. However, there were greater differences in their computational performance (model build time and classification speed). NBayes was the algorithm that was quickest to build models, and C4.5 was one of the most accurate classifiers and the fastest in terms of classification speed. NBayes was also used in a pioneer work in ML-based IPTC [14].

We used the Waikato Environment for Knowledge Analysis (WEKA) implementation of NBayes and C4.5 [36].

B. Flows and Features

In our experiments, for UDP traffic, a full flow is considered to have stopped when no more packets are seen for a *timeout* of 60 s in both directions. For TCP traffic, a full flow is stopped when the connection is explicitly torn down or no packets are seen for 60 s in both directions (whichever comes first).[1]

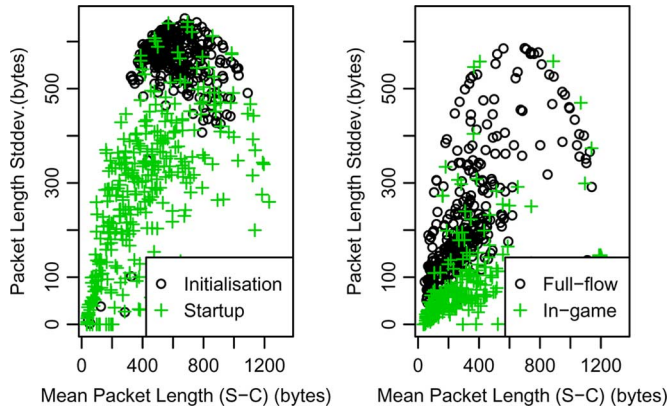

Fig. 4. Packet length from S–C for ET traffic: clear disjoint regions for different phases during game flows' lifetime.

The ML literature often utilizes two additional metrics known as *Recall* and *Precision*. If a classifier is trained to identify members of class X, these metrics are defined as follows. Both metrics range from 0% (poor) to 100% (optimal). Note that high Precision only is useful when the classifier achieves good Recall, and vice versa.

- *Recall*: Percentage of members of class X correctly classified as belonging to class X, among all members of class X. $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$.
- *Precision*: Percentage of those instances that truly belong to class X, among all those classified as class X. $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives+False Positives}}$.

We used the following features, calculated separately in the forward and reverse directions: interpacket arrival time, interpacket length variation, and IP packet length. For all, we computed minimum, maximum, mean, and standard deviation values. These statistics are simple to calculate. We used Netmate [37] to calculate the feature values for our analysis.

C. Applications

1) FPS Game Traffic: We chose *Wolfenstein: Enemy Territory* (ET), a team-based FPS game, as the representative application of the popular fast-paced online FPS game genre. Our ET dataset consists of two separate month-long traces collected during May and September 2005 at a public ET server in Australia. Full-payload traffic was captured with packet timestamps of microsecond resolution and an accuracy of ± 100 μs.

Consistent with other online FPS games, ET traffic seen at a server can exhibit three different phases: clients probing the server (Initialization phase), clients connecting to the server (Startup phase), and clients playing a game on the server (In-game phase) [38].

Fig. 4 shows the variation of an ET flow's characteristics as a scatter plot of two features—standard deviation versus mean of packet length from S–C—calculated with $N = 25$ for 1000 ET flows. In this illustrative example, the Initialization phase's features are calculated for SF-0 (covering the first 25 packets of the full flows), the Startup phase's features are calculated for

[1]A timeout can result in some truncated flows. However, our approach allows a classifier to start at an arbitrary point during a flow's lifetime, which makes it less sensitive to a particular flow timeout setting.

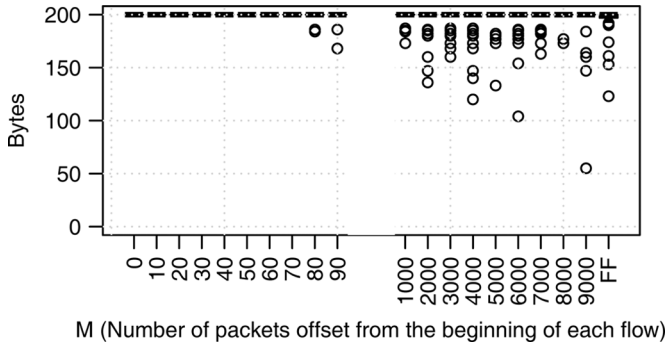

Fig. 5. Mean packet length G.711 in forward direction.

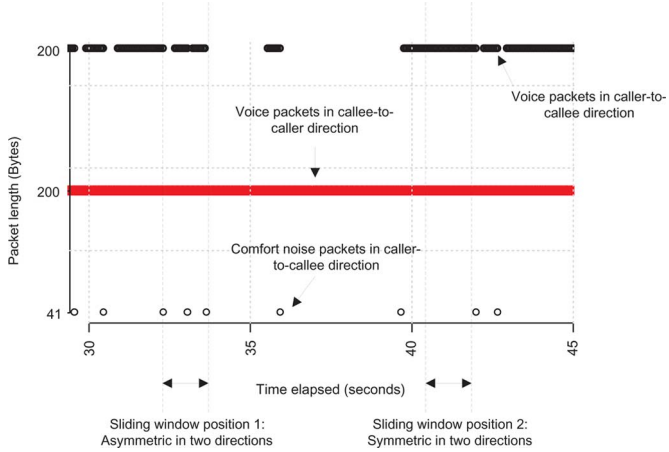

Fig. 6. Comfort noise packets and silence suppression periods during a conversation can create asymmetry and multiple packet sizes.

SF-20, and the In-game playing phase's features are calculated for SF-2000.[2] Full-flow features are calculated over the entire flows.

With only two features, the regions are partially overlapped and partially disjoint. While there is considerable overlap between the different phases, there is also some separation, which needs to be learned by a classifier to identify the different phases. Similar behaviors are seen in the C–S direction (not shown in the figures). A similar mix of overlapping and disjoint regions also occurs with other features (such as interpacket arrival time and interpacket length variation). This suggests that a classifier trained on full-flow feature values may have trouble recognizing the clusters of feature values calculated for sub-flows.

2) VoIP Traffic: Our traffic data comprises 3.4 GB of VoIP traffic in 644 Real-time Transport Protocol (RTP) flows (92% G.711 and 8% GSM flows) collected on a home network between November 2006 and August 2007. Calls' durations range between 19 and 8207 s. Median call duration is 80 s (with a total of 7061 packets in both directions), with the 75th percentile at 355 s (with a total of 30 530 packets in both directions).

Fig. 5 shows the mean packet length (including IP/UDP/RTP header) feature values from the caller to callee for G.711 traffic for sub-flows with $N = 25$ packets. As specified earlier, M is the packet offset from the beginning of each flow. (On the far right of the x-axis, FF represents features calculated on full flows.) G.711 traffic in the reverse direction exhibits similar characteristics.

As shown in Fig. 5, most packets are 200 B long. For $M \geq 80$, there are outliers, which are less than 200 B. These are due to the presence of comfort noise packets (each 41 B long) within the sub-flow. These outliers are only evident when conversations are in progress.

Examining the packet interarrival times (not shown) indicates that most packets arrive in 20-ms intervals, but there are outliers with packet interarrival times of more than 20 ms. These longer packet interarrival times are due to jitter, packet loss,[3] and silent periods.

The analysis of packet length and interarrival time statistics for G.711 traffic reveals that, in most cases, voice packets have the same length with little variation of packet arrival intervals. This makes the voice traffic stationary for the duration of a conversation, and symmetric in both directions. GSM flows have similar traffic characteristics.

Fig. 6 shows the impact of silence suppression by focusing on 15 s of traffic in each direction of a G.711 call (according to RTP sequence numbers no packets were lost). The y-axis shows the IP packet length (200 B in both directions). To differentiate the traffic in each direction, we offset the plot in the y-axis for packets from callee to caller. In this example, the traffic is asymmetric at position 1 and symmetric at position 2.

Although our voice traffic is usually stationary and symmetric in both directions, there are cases where:

- features calculated over full flows differ from those calculated on sub-flows;
- features calculated on sub-flows at different points during a flow's lifetime differ from each other;
- features calculated in one direction differ from those calculated in the other direction.

D. Other Traffic

The interfering (non-ET and non-VoIP) traffic is taken from two daily data traces collected by the University of Twente, Enschede, The Netherlands, on February 6 and 7, 2004 [39]. The interfering traffic datasets were built by extracting flows belonging to a range of common applications. As payloads were missing, we inferred application types from the well-known default port numbers. For each application's *default port(s)*, we sampled a maximum of 10 000 flows per raw trace file[4]. Space limitations preclude a detailed analysis of these applications' traffic characteristics. However, their statistics are all asymmetric in the C–S and S–C directions and vary over a flow's lifetime.

E. Construction of Training and Testing Datasets

We use nonoverlapping datasets for training and testing our classifiers. Flows in each dataset are divided into two classes—ET and Other (non-ET) or VoIP and Other (non-VoIP)—because supervised learning algorithms must be

[2]SF-20 and SF-2000 are points where we can be certain the game protocol has moved from one phase to the next phase (see Fig. 1 for an illustration of the sub-flow definition and sub-flow naming convention).

[3]Based on RTP sequence numbers, 93% of the flows are missing less than 2% of their packets. The largest loss of packets (4.9%) occurred in a single voice conversation.

[4]P2P applications have a range of default port numbers, hence the number of flows collected for one application can be larger than 10 000.

TABLE I
APPLICATION FLOWS DATASET (ET + OTHER OR VOIP + OTHER)

Applications	Training (Dataset1) Flows (x1000)	Bytes (MB)	Testing (Dataset2) Flows (x1000)	Bytes (MB)
ET	8.688	14900	6.888	26600
VoIP	0.341	1800	0.303	1600
Other:				
HTTP, HTTPS	13.8	329.2	13.3	267.2
DNS, NTP	2.4	1.4	2.7	1.4
SMTP, IMAP, POP3, Telnet, SSH	0.6	15.8	0.5	10.1
Kazaa, Bittorrent, Gnutella, eDonkey	48.0	1,354.6	56.4	1,524.5

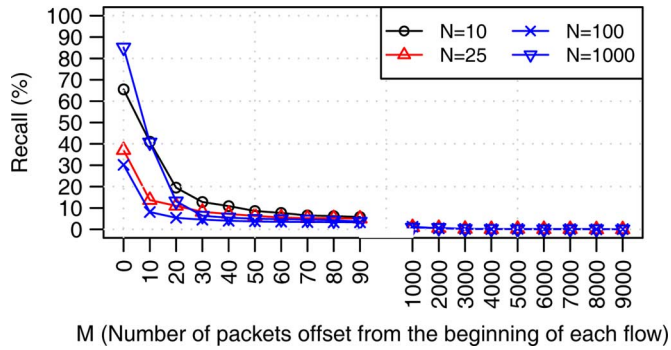

Fig. 7. ET Recall: NBayes classifier trained with full flows, tested with four different sub-flow sizes.

trained with examples of traffic in the class of interest and traffic known to be outside the class of interest. Table I summarizes the overall mix of traffic for training and testing. For training a classifier, Dataset1 consists of the whole May data for ET, or the first five months of data for VoIP, and February 6 data for Other traffic. The remaining data (Dataset2) are used for testing: the whole September data for ET, or the last four months of data for VoIP, and the February 7th data for Other traffic.[5]

Detailed information on how many instances from each class (ET/VoIP and Others) used for training and testing in each experiment in Section V is presented in the Appendix. For training on SSP-ACT, our approach is described as follows.

1) Step 1: Sub-Flow Identification: To manage computational processing overhead, we manually identified a set of sub-flow positions for ET and VoIP based on protocol knowledge. We divided the full flows into two phases, "starting" and "in-progress," and selected a number of sub-flows for each phase. Sub-flows for "starting" started at $M = 0$, increasing by steps of 10 packets, until $M = 90$. Sub-flows for the "in-progress"' phase started at $M = 1000$ and increased by steps of 1000 packets until $M = 9000$. This resulted in sub-flows starting at 19 different positions of the full flows. Instances of these sub-flows were labeled with their position M (for post-clustering analysis only) and then fed to the WEKA's EM implementation.

2) Step 2: Sub-Flow Selection: EM can find the number of clusters automatically using 10-fold cross validation. Beginning with one cluster, it continues to add clusters until the estimated log-likelihood decreases. As EM is initialized with a random parameterization, 10 iterations were used, and the result was clusters with the highest log-likelihood.

To select representative sub-flows for each cluster, we used the classes to clusters evaluation mode. In this mode, WEKA first ignores sub-flow labels and performs the clustering. During the test phase, it assigns a sub-flow label (a sub-flow position) to each of the clusters, based on the majority class within each cluster.

For ET traffic, EM found eight clusters of sub-flows and assigned eight sub-flow positions to these clusters: SF-0, SF-10, SF-20, SF-30, SF-40, SF-50, SF-60, and SF-3000 (with a total of 12 804 sub-flow instances, approximately half of instances from sub-flows identified in Step 1). For VoIP traffic, EM found five clusters of sub-flow positions: SF-0, SF-10, SF-30, SF-1000, and SF-3000. For other traffic, since most flows are short, we use SF-0 and SF-10 in Dataset1 for training, and use SF-20 in Dataset2 for testing. We use the identified sub-flow positions to train and test the NBayes and C4.5 classifiers.

V. RESULTS AND ANALYSIS

First, we demonstrate with ET traffic that training on multiple sub-flows produces better classification accuracy compared to training on full flows or on a single sub-flow, and then demonstrate the benefit of SSP-ACT. The combined approach with SSP-ACT is then evaluated for VoIP traffic.

Our goal is to classify ET and VoIP traffic in less than 1 s. We choose a window size of $N = 25$ for our experiments, which roughly corresponds to a time window of 0.5 s for both ET and VoIP traffic.[6]

A. Classification of ET Traffic

1) Training With Full Flows: First, we analyze the classification accuracy using an ML classifier trained on full flows and test with sub-flows of $N = 10$, 25, 100, and 1000 packets (corresponding to 0.2, 0.5, 2.1, and 20.8 s of actual time). We focus on the accuracy of classifying ET traffic, so we only report Recall and Precision for the ET class. This experiment demonstrates the poor performance of training on full flows when classifying using small sub-flows (even with N as big as 1000 packets), particularly when the sub-flow position moves beyond the start of the flow.

Fig. 7 shows Recall for the NBayes classifier. For all N, Recall degrades rapidly as M moves farther away from the flow start. Recall for $N = 1000$ is good (85%) when the flow is captured from the beginning, but rapidly drops below 10% when $M \geq 30$. Recall for small sub-flows is poor ($\leq$66%) even when the beginning of a flow is captured. Moving beyond the first 20 packets further degrades Recall to less than 20% for all N. Classifying sub-flows in the middle of gameplay ($M > 1000$) provides a Recall close to 0%. Precision was up to 100% for certain values of M, however, given the exceedingly poor Recall, the high Precision values convey no useful insight.

Similarly, Recall for C4.5 classifiers (not shown in the figures) is quite good when M is 0. It even achieves 98% Recall for $N = 1000$ when the flows' beginning is captured. However,

[5]Swapping datasets for training and testing (twofold cross validation) showed similar results, hence only results with this training and testing dataset are presented in Section V. Twofold cross-validation results are presented in Section VII.

[6]During ET gameplay, there are 20 pps from S–C and roughly 28 pps from C–S, so a 25-packet window corresponds to 0.5 s. VoIP has a packet interarrival time of 20 ms, and in the worst case (when there is voice traffic only in one direction), it takes 0.5 s to collect 25 packets in a talk spurt.

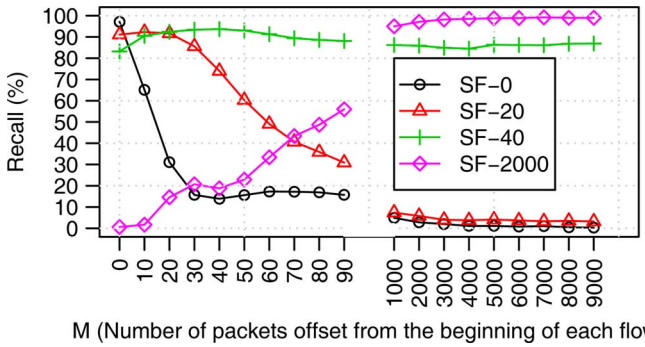

Fig. 8. ET Recall: NBayes classifier trained and tested on 25-packet sub-flows.

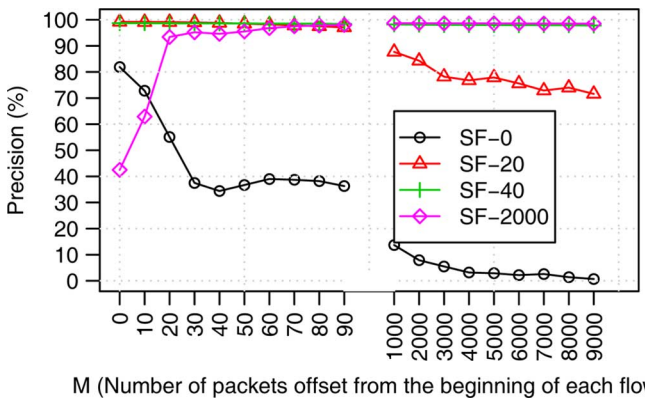

Fig. 9. ET Precision: NBayes classifier trained and tested on 25-packet sub-flows.

there is significant degradation in Recall for all sub-flow sizes (dropping to approximately 25% when $M \geq 20$ packets). With $N = 1000$, Precision stays above 80% for all M; again, given the exceedingly poor Recall figures, the high Precision is less meaningful.

2) Training With One Sub-Flow Position: We now show that it is better, but still not sufficient, to train the classifier using a single sub-flow of N packets when classifying sub-flows. No matter how carefully we might select one sub-flow for training, it is unlikely to find the one that has characteristics similar to all the phases of traffic flows over their lifetime.

Fig. 8 presents Recall for NBayes classifiers trained using sub-flows at specific positions where $M = 0, 20, 40, 2000$ (the SF-0, SF-20, SF-40, and SF-2000 models, respectively). Each sub-flow model represents a particular phase with distinctive statistical properties of an ET flow during its lifetime. Consequently, training on a sub-flow at a particular phase of a flow tends to result in higher Recall at the same phase in the test dataset, and lower Recall otherwise.

For example, using the SF-0 model results in Recall starting very high at the beginning of a flow and dropping off quickly if $M \geq 10$. Conversely, using the SF-2000 model results in Recall staying low until the sliding window has moved beyond the early period of each flow ($M \geq 90$). Recall with the SF-20 model is good even if we miss 30–40 packets, but eventually becomes poor. The SF-40 model exhibits good overall Recall of greater than 80% with all M.[7]

[7]The SF-40 model contains a blend of "Startup" and "In-game" statistics because the game transitions from "Startup" to "In-game" phase during or near $M = 40$.

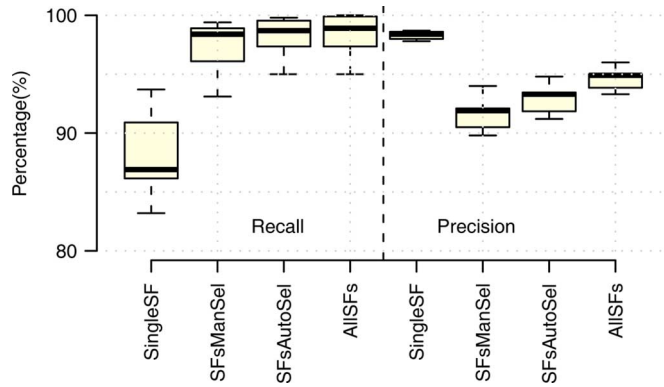

Fig. 10. Comparing NBayes classifiers trained using single versus multiple sub-flow positions.

Fig. 9 shows Precision results following the same trend as Recall. The C4.5 classifier shows similar results. The SF-0, SF-20, and SF-2000 models show good Precision/Recall results around $M = 0, 20, 2000$, respectively, and the SF-40 model maintains good Recall of $\geq$90% and Precision of $\geq$97% for all M.

3) Training With Multiple Sub-Flow Positions and ACT: We propose to improve on the previous two approaches by training the classifier on a combination of multiple sub-flows. The classifier will then recognize new flows if they have statistical properties similar to any of the sub-flows on which the classifier was trained.

To evaluate this approach, we compare Recall and Precision for classifiers trained on a single sub-flow (SingleSF), a combination of the four handpicked sub-flows (SF-0, SF-20, SF-40, and SF-2000) analyzed in Section V-A.2 (SFsManSel), a combination of eight sub-flows selected by ACT (SFsAutoSel), and all sub-flows identified in Step 1 Section IV-E (AllSFs).

The results are presented using boxplots. The thick line in the box indicates the median; the bottom and top of the box indicate the 25th and 75th percentiles. The difference between the 75th and 25th percentile is the interquartile range (IQR), and the vertical lines drawn from the box (whiskers) extend up to 1.5 times the IQR. Any observations beyond the whiskers are deemed outliers and are drawn as individual points.

Fig. 10 shows that training on multiple sub-flow positions achieves higher and more stable Recall than training on a single position. All NBayes classifiers trained on multiple sub-flows show very good median Recall of more than 98% compared to 87% Recall for the best single sub-flow model. Yet, this comes with a slight decrease in Precision. Precision when trained on the multiple sub-flows is 5%–8% lower than that of the best single sub-flow model.

This tradeoff is illustrated in Fig. 11. Sub-flows at different M form different clusters. These clusters are partially overlapped and partially disjoint. Training using a single M, e.g., SF-40 for the best single sub-flow model (dark gray area), leaves out many sub-flows at other positions (outliers to the SF-40 cluster). Training on multiple positions makes sure these are included, and therefore the classifier's Recall is improved. However, the inclusion of more positions also creates a greater area (light gray area). With this increasing area, the opportunity for false positives (sub-flows of other classes covered) increases, and hence the Precision decreases.

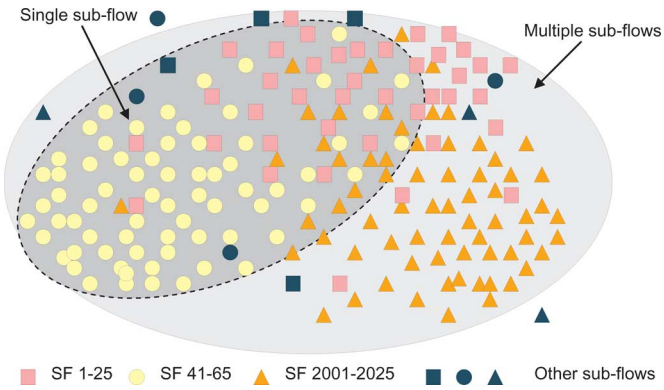

Fig. 11. Illustration of creating a multiple sub-flow positions classifier from a number of individual sub-flow positions (data points are artificially created for illustrative purposes only).

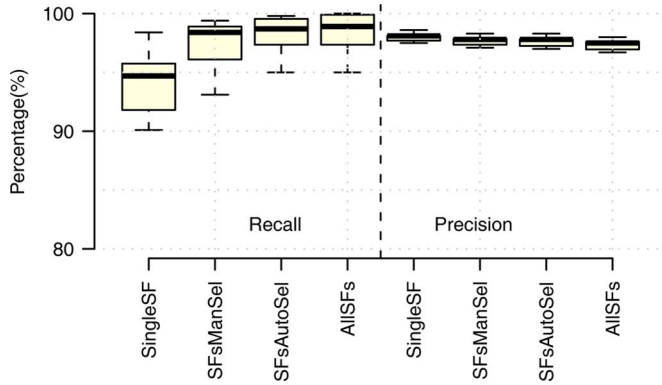

Fig. 12. Comparing C4.5 classifiers trained using single versus multiple sub-flow positions.

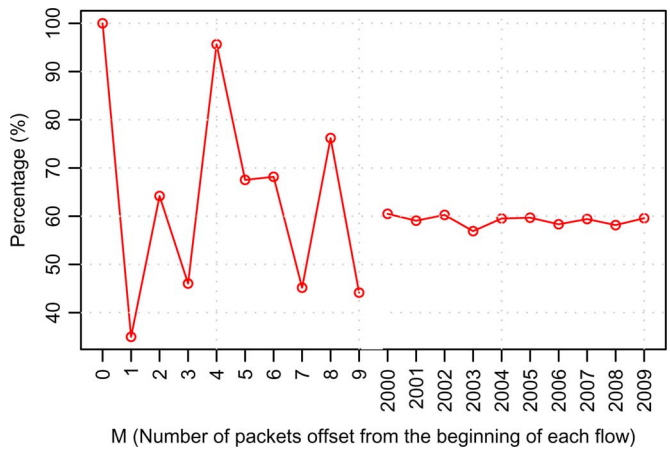

Fig. 13. Percentage of flows that have the first packet captured in the C–S direction if the first M packets are missed.

Among all classifiers trained with multiple sub-flows, the SFsAutoSel classifier has the highest median Recall of 99%, followed by the AllSFs and SFsManSel classifiers with median Recall of 98.5% and 98.3%, respectively. All classifiers using multiple M achieve higher than 91% Precision. The AllSFs classifier achieves the highest median Precision (94.9%), followed by the SFsAutoSel (93.3%) and the SFsManSel (91.9%) classifiers.

The high accuracy of the SFsAutoSel classifier suggests that ACT effectively assists the selection of sub-flows that cover critical phases of the application's flows during their lifetime. In addition, it takes 15% less time to build the SFsAutoSel classifier,[8] and the classification speed increases by 4% compared to the AllSFs classifier.

Fig. 12 shows similar results with the C4.5 model. All classifiers trained on multiple sub-flow positions produce greater than 98% Recall and 97% Precision. The AllSFs classifier has the highest median Recall of 98.9%, followed by the SFsAutoSel (98.7%) and the SFsManSel (98.4%) classifiers. The SFsManSel and SFsAutoSel classifiers achieve nearly identical median Precision of 97.8%. The AllSFs classifier achieves slightly lower Precision, with a median of 97.5%.

Our results confirm that for applications with time-varying traffic characteristics, there are significant benefits of training ML classifiers using features calculated from multiple sub-flow positions. In addition, manual selection of positions for training is not necessary and not optimal. Higher Precision and Recall can be achieved using ACT for automated sub-flow position selection. Using all sub-flows identified may lead to suboptimal accuracy and lower performance in terms of model build time and classification speed.

4) Evaluation of SSP-ACT: When a classifier model is trained with an explicit definition of the direction (the direction of the first packet of a full flow, usually C–S direction), its Recall depends on the proportion of flows with the first packet seen by the classifier traversing in the same direction (i.e., from C–S). If the first packet of a flow seen traverses in the opposite direction, the classifier's performance will be negatively affected by asymmetric statistical properties in both directions.

[8]Excluding the clustering time, which can be optimized by only using a small number of sampled instances or a different clustering algorithm. Using 100 samples per sub-flow position reduced clustering times by almost 95% with less than 0.3% reduction in Precision and Recall.

In Section V-A.3, we demonstrated high and stable Precision and Recall. However, these results were achieved under the assumption that the first packet of a flow can always be captured; hence the classifier always know the "correct" direction of the traffic. Now, we demonstrate that the classification performance significantly degrades when the first packet captured by the classifier is in the "wrong" direction (i.e., S–C direction). For example, in practice this can happen if traffic is sampled, or the classifier is rebooted. Then, we show how training with SSP improves Recall and Precision.

Measured across all ET flows in the test dataset, Fig. 13 shows the percentage of sub-flows whose first packet is in the C–S direction as a function of M (with a step-width $S = 1$[9]). Not surprisingly, this is 100% when $M = 0$, and fluctuates significantly for $1 \leq M \leq 9$ (the value does not reach 0% for $M = 1$ because for $\sim$35% of ET flows, both the first and second packets are in the C–S direction). This fluctuation is expected in the Probing phase where the client is probing the server. In the In-game phase $2000 \leq M \leq 2009$, it is more stable ($\sim 60 : 40$ chance that packets $2001 \ldots 2009$ traverse in the C–S direction). This is consistent with an analysis of the data trace showing that during ET gameplay there are roughly 28 pps from C–S and 20 pps from S–C.

[9]$S = 1$ was chosen to make the alternating direction from C–S and S–C of the first packet in the window clearer.

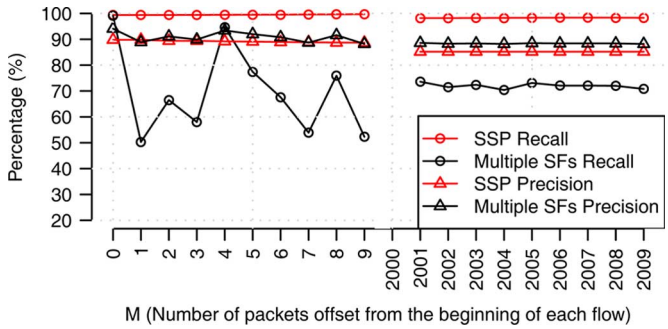

Fig. 14. Recall and Precision for NBayes classifiers trained using SSP and multiple sub-flows.

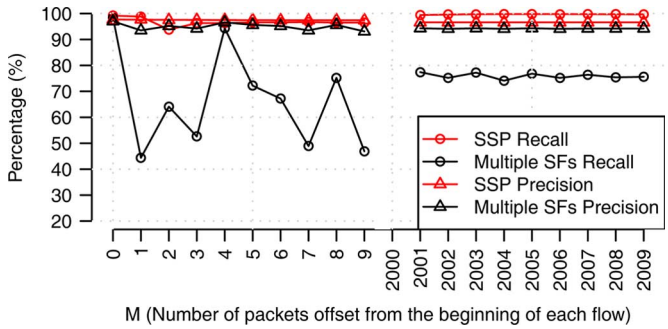

Fig. 15. Recall and Precision for C4.5 classifiers trained using SSP and multiple sub-flows.

Fig. 14 compares Recall and Precision as a function of M for the NBayes classifier trained with the SFsAutoSel model with and without SSP. It shows that without SSP the accuracy suffers when the classifier incorrectly assumes the direction of the first packet. Its median Recall is low, fluctuating around 66% when $0 \leq M \leq 9$ and remaining at ~76% when $2000 \leq M \leq 2009$. Its Precision fluctuates above 90% for all M. An NBayes classifier trained with SSP shows significant improvement in Recall (a median of 98.9%) compared to training without SSP (a median of 72.1%). More importantly, Recall is more stable and not affected by the directions of the packets across all M.

However, there is a tradeoff between Recall and Precision. Compared to the classifier trained on SFsAutoSel without SSP, the classifier with SSP achieved 3% lower Precision on average for all M, and its Precision remains at 85.2%–89.8%. It is also notable that the median Precision is slightly lower when $M \geq 2000$ compared to when $0 \leq M \leq 9$. This is due to a smaller number of ET flows when $M \geq 2000$, which leads to a smaller number of true positives for ET traffic.[10]

Fig. 15 shows that a C4.5 classifier trained using SSP also shows significant improvement in Recall (a median of 99.3%) compared to the one trained without SSP (a median of 75.2%). Recall is not only higher, but also more stable and not affected by the directions of packets. In contrast to the NBayes classifier, there is also a gain in Precision when training with SSP. The Precision increases by 2.7% on average, staying at a 97.3%–98.2% for the SSP classifier.

[10]The number of testing instances reduces when M increases, as there are fewer flows longer than $M + N$ packets. Consequently, the number of true positives is reduced. On the other hand, the number of false positives is constant, as the same number of instances of Other traffic is used for testing with all M. Hence, Precision reduces when M increases (see Table IV).

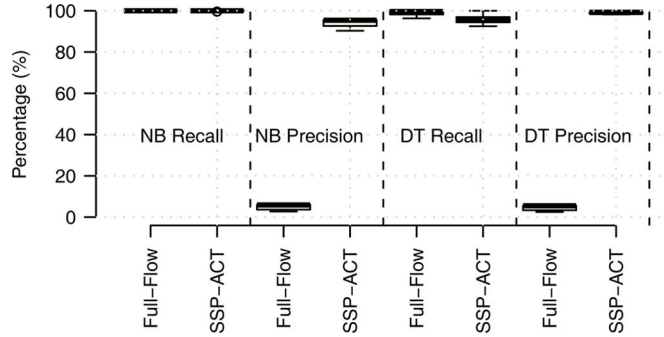

Fig. 16. VoIP Recall and Precision: Naive Bayes (NB) and C4.5 Decision Tree (DT) classifiers trained on full flows and sub-flows with SSP-ACT.

B. Classification of VoIP Traffic

Now we will demonstrate the benefit of SSP-ACT when identifying VoIP traffic (whose statistical characteristics differ significantly from those of ET traffic) among other traffic with time-varying asymmetric statistical characteristics.

Fig. 16 shows Recall for VoIP traffic for the NBayes and C4.5 classifiers trained on full flows and sub-flows with SSP-ACT. Training on full flows resulted in a median Recall of 100% for all M. However, the median Precision is less than 6%. On the other hand, the classifier trained using sub-flows with SSP-ACT achieved a median Recall and Precision of 99.6% and 95.4%, respectively. We obtained similar results with C4.5 classifiers. Training on full flows achieved a median Recall of 99.6% with less than 5.6% Precision. Training using sub-flows with SSP-ACT achieved a median Recall and Precision of 95.7% and 99.2%, respectively.

The high Recall and low Precision when using classifiers trained on full flows can be explained as follows. As most VoIP flows are stable and symmetric in both directions, training on the full flows is sufficient to identify VoIP traffic even when classifying on a small window (high Recall). However, the statistical characteristics of Other traffic vary during a flow's lifetime and are asymmetric in the forward and reverse directions. Both the NBayes and C4.5 classifiers trained on full flows misclassify a large percentage of Other traffic as VoIP traffic (very poor Precision). In contrast, classifiers trained using SSP-ACT can identify both VoIP and Other traffic well.

Results for the C4.5 classifier also reveal that training using SSP-ACT can sometimes produce slightly lower Recall than training on full flows. The distinction between VoIP and Other traffic features calculated on sub-flows may not be as significant as when calculated on full flows. Training on sub-flows hence generates more conservative classification rules than training on full flows. While the former rules produces high Precision for VoIP traffic, they come with a tradeoff of more false negatives, resulting in a lower Recall.

This explanation is illustrated in Fig. 17. The classification model trained on full flows covers a greater range of VoIP instances compared to the model trained using SSP-ACT. With VoIP traffic, the difference between feature values calculated on sub-flows and full flows is less significant than in the case of Other traffic. Hence, the full-flow model can produce high Recall but very low Precision. Training with SSP-ACT provides much greater Precision, with a little reduction in Recall due to a greater number of false negatives.

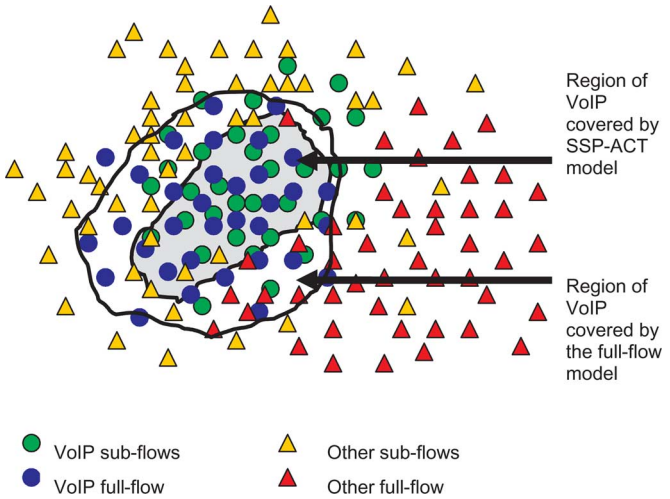

Fig. 17. VoIP classification using classifiers trained on full flows and sub-flows with SSP-ACT: Training on full flows may cover a larger area of VoIP instances than training on sub-flows, hence resulting in higher Recall but lower Precision. (The data points are artificially created for illustration purposes only.)

Our technique of training on multiple sub-flows (without SSP-ACT) also works for other VoIP traffic that has more variable characteristics, such as Skype traffic. Skype's default audio codec varies its rate according to which party is talking and adapts to the available bandwidth [5]. With 1-s sub-flows, a NBayes classifier achieved 91% Precision and 97% Recall when separating Skype from ET and Other traffic (with 5-s sub-flows accuracy increased to 98% Precision and 99% Recall) [5].

VI. Classification Accuracy With Packet Loss

Like other IPTC approaches that rely on statistical properties of traffic at the network layer, the performance of a classifier could be affected by network-layer perturbations. Packet loss, packet reordering, traffic shaping, packet fragmentation, and jitter are likely to result in variations of the feature values. Here, we present preliminary results of the robustness of our proposed method in the event of packet loss.

Packet loss will have an impact on feature values that are based on packet length and packet arrival time statistics. For example, a few lost packets in a sliding window would result in longer gaps in packet arrivals, which consequently affects the packet interarrival time statistics. Similarly, the biased loss of large or small packets would affect the packet length statistics, and so on. Different packet loss patterns (such as random or bursty losses [40]) can also have different impacts on the feature value changes.

We study the changes in Precision and Recall of a classifier trained using SSP-ACT when classifying a test dataset with synthetic packet loss. The dataset was created from the original trace file by randomly skipping packets when calculating features statistics (any given packet might have been lost with a prespecified probability p, and these random losses are independent). Since there may be some packet loss present in the original dataset already, we also refer to the loss we introduced as *additional synthetic* loss.

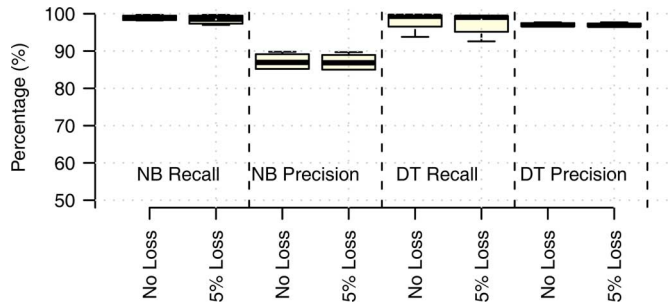

Fig. 18. ET Recall and Precision: training with SSP-ACT and classifying with ET traffic experiencing 0%/5% random packet loss—Naive Bayes (NB) and C4.5 Decision Tree (DT) classifier.

ET and VoIP traffic are not reactive to packet loss (no significant flow-control or packet retransmissions occur at the application or transport layers). On the other hand, many Other applications are TCP-based, and consequently exhibit network-layer retransmissions and adjustment of their sending rate in the presence of packet loss. The loss-reactive nature of TCP flows makes it hard to predict the impact of additional synthetic loss on previously calculated feature values. Therefore, the synthetic loss is only applied for ET and VoIP traffic, but not for the Other traffic. The experimental Precision results may be higher than the actual Precision achieved when packet loss also occurs with Other traffic.

In terms of selecting a realistic loss rate for testing, [41] reveals that both cable and Digital Subscriber Line (DSL) have remarkably low packet loss rates of less than 1% for more than 95% of all paths. In addition, packet loss as low as 1% or 2% is known to trigger dramatic degradation in TCP throughput [42]. In this preliminary study, we chose a 5% packet loss rate (total loss in both directions for bidirectional traffic) to approximate a reasonable upper bound on tolerable packet loss in an Internet access link.

For sub-flows of $N = 25$ packets, a 5% packet loss rate typically results in the loss of only 1–2 packets per sub-flow. This has relatively little impact on ET feature values. For example, the median of the maximum packet interarrival time feature (calculated over all selected sub-flows) is increased by only $\sim$2 ms (0.2%) compared to the original test dataset. Analysis of the other features shows similar small effects.

For VoIP traffic, 5% packet loss greatly increases the median of the maximum packet interarrival time feature by $\sim$18.7 ms (88.3%)[11] compared to the original test dataset. However, the minimum and mean packet interarrival time features change only slightly (by less than 4%) compared to the original test dataset. There is little impact on the maximum, minimum, and mean packet length features because VoIP traffic is quite stable in terms of packet length.

Fig. 18 shows Recall and Precision for the NBayes and C4.5 classifiers both with and without a 5% packet loss for ET (classification model trained with SSP-ACT). For the NBayes classifier, packet loss caused the median Recall and Precision to reduce by 0.45% and 0.4%, respectively. For the C4.5 classifier,

[11]This is not increased by 100%, as a bigger gap caused by packet loss may still be smaller than the gap caused by silent periods.

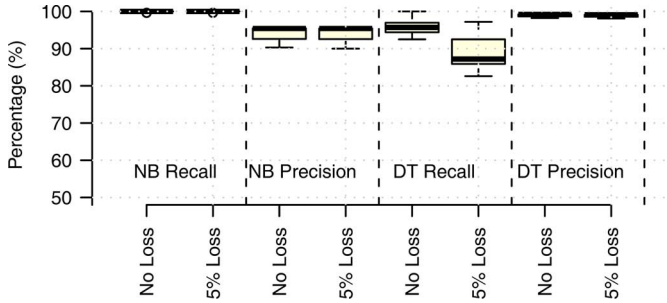

Fig. 19. VoIP Recall and Precision: training with SSP-ACT and classifying with VoIP traffic experiencing 0%/5% random packet loss—Naive Bayes (NB) and C4.5 Decision Tree (DT) classifiers.

median Recall and Precision reduced by only 0.5% and 0.25%, respectively.

Fig. 19 shows Recall and Precision for the NBayes and C4.5 classifiers trained with SSP-ACT both with and without a 5% packet loss rate for VoIP traffic. The packet loss does not have a noticeable impact on Recall for the NBayes classifier, which remains the same at more than 99.6%. Not surprisingly, there was no noticeable impact on Precision. This is because Recall is maintained, and the same test dataset is used for Other traffic in both tests.

In contrast, the C4.5 classifier showed a significant negative impact of packet loss on VoIP Recall, the median Recall was reduced by up to 8.5%. Median Precision was only slightly reduced by 0.1% due to the variation in VoIP Recall (a reduction in the number of true positives). The only slight reduction in Precision versus the significant reduction in Recall is due to the relatively very small number of false positives versus the larger number of false negatives. Despite these degradations, median Recall and Precision for the C4.5 classifier still remained above 87% and 99%, respectively.

The C4.5 classifier is more sensitive to packet loss than the NBayes classifier because the former builds a tree with tests based on precise feature values, while the latter builds a model based on continuous probability distributions. For C4.5, a small change in feature values can lead to different paths within the tree, which can subsequently lead to a different classification result (as noted in [43]). The NBayes classifier is more tolerant to small variations in feature values. Overall, the results suggest our proposed approach is robust against a moderate 5% packet loss.

VII. Evaluation of Real-World Implementation

The techniques described in our paper have led to the development and public release of an open-source, ML-based traffic classification and control extension [10] to FreeBSDs IP Firewall (IPFW) [44]. Called DIFFUSE (for DIstributed Firewall and Flow-shaper Using Statistical Evidence), it implements classification on sub-flows and SSP, runs on generic PCs, and allows new features or ML algorithms to be easily added. DIFFUSE is fast because it runs entirely in the FreeBSD kernel and there is an efficient interface between ML classification and subsequent blocking (firewall) or shaping (with IPFW's Dummynet). A preliminary Linux port also exists [10].

DIFFUSE also allows decoupling of traffic classification and treatment, enabling scenarios previously envisaged in [45] and [46]. *Classifier Nodes* (CNs) classify traffic flows and then instruct *Action Nodes* (ANs) to carry out actions for the classified flows. The CN and AN may exist within a single host (as in a traditional firewall), or a CN may control one or more ANs distributed around the network. For example, a CN inside an Internet Service Provider (ISP) network differentiates each customers' traffic into real-time multimedia (e.g., games, VoIP) and other traffic, and then instructs ANs to prioritize the real-time traffic heading each way over a customer's link.

We used DIFFUSE to evaluate the accuracy of our proposed approach for the datasets described in Section IV-C. Although DIFFUSE supports offline analysis, we here report the results of online experiments with real packets flowing across our testbed network; a traffic source PC (Intel Core i5 2.8 GHz, 4 GB RAM) used tcpreplay [47] to send the traffic contained in the traces, and the traffic was monitored by a classifier PC (Intel Core i7 2.8 GHz, 6 GB RAM) connected to the same switch (using port mirroring).

For the DIFFUSE experiments, we trained classifier models based on a large number of randomly sampled sub-flows from all positions (instead of a small number of handpicked sub-flows).[12] The proportion of training flows sampled was tweaked to be 100 000–150 000 sub-flows in total with roughly equally sized classes (for both ET and VoIP). We used the same datasets for other traffic as before. However, for ET traffic, we randomly selected two 24-h periods with high gaming activity, and for VoIP, we randomly selected two 6-h periods with very high call activity (G.711 and GSM). Since we performed online experiments with interarrival time features, all experiments run in *real time*, making it unfeasible to use the full ET or VoIP traffic covering multiple months.

We performed twofold cross validation, using one dataset for training and another for testing, and vice versa. We used the same features as described in Section IV-B, a sub-flow size of $N = 25$ packets as before and $S = 25$ (nonoverlapping windows).[13] Interarrival times features were truncated to milliseconds (in order to avoid large numbers that could create integer overflows since the kernel-based DIFFUSE is restricted to integer arithmetic). Since accuracy for Naive Bayes and C4.5 are quite comparable in our experiments, we only used C4.5 for the experiments here. We used WEKA's C4.5 default parameters for training, except that we configured a more aggressive tree pruning to reduce overfitting.

Table II contains the results and shows that traffic is classified with 98%–99% accuracy in all cases. The accuracy for ET and VoIP is slightly higher than the results shown in Section V, but that is expected since we only use a small part of the overall ET/VoIP data. Recall for the ET/VoIP class is similar across the tests without loss. However, in the experiments with ET, Recall for the other traffic varies; it is higher when training on the largest trace (Dataset1), and lower when training on the smallest data (Dataset2).

Table II also shows results with emulated uniform random 5% packet loss occurring before the classification (packets not

[12]DIFFUSE does not support ACT yet.

[13]All packets are classified, but features are only updated for every sub-flow.

TABLE II
TOTAL NUMBER OF PACKETS/BYTES, CLASSIFIED PACKETS/BYTES (FLOWS ARE ONLY CLASSIFIED ONCE AT LEAST 25 PACKETS HAVE ARRIVED), AND PRECISION AND RECALL FOR THE ET AND VOIP CLASSES BASED ON CLASSIFIED PACKETS AND BYTES (IP PACKET SIZE INCLUDING IP HEADER). LOSS IS PACKETS NOT SEEN BY THE CLASSIFIER (NOT NETWORK PACKET LOSS)

Experiments	Metric	Train Dataset 1, test Dataset 2		Train Dataset 2, test Dataset 1	
		Loss 0%	Loss 5%	Loss 0%	Loss 5%
ET+Other	Total packets/bytes	41M / 14GB	39M / 13GB	54M / 16GB	51M / 16GB
	Classified packets/bytes	35M / 12GB	33M / 11GB	47M / 14GB	44M / 14GB
	ET Precision packets/bytes	99.2% / 99.4%	99.2% / 99.5%	98.6% / 98.9%	98.7% / 99.0%
	ET Recall packets/bytes	99.9% / 99.8%	99.9% / 99.6%	99.8% / 99.7%	99.8% / 99.7%
VoIP+Other	Total packets/bytes	21M / 11GB	20M / 10GB	21M / 11GB	20M / 11GB
	Classified packets/bytes	15M / 9GB	14M / 9GB	14M / 9GB	13M / 9GB
	VoIP Precision packets/bytes	99.6% / 99.9%	99.7% / 99.9%	99.6% / 99.9%	99.8% / 99.9%
	VoIP Recall packets/bytes	99.1% / 99.3%	97.9% / 98.4%	99.3% / 98.4%	99.2% / 98.2%

seen by the classifier and *not* network packet loss). Overall, the accuracy does not change much since many of our features are not very sensitive to packet loss (given $N = 25$ packets) as explained earlier. For ET, the accuracy does not decrease, as the features that are more sensitive to loss (such as the maximum interarrival times) are not used at important places in the classifier tree, i.e., they are only used in parts of the tree that cover a very small fraction of the (training) traffic. For VoIP, the maximum interarrival times are more important, and consequently Recall decreased slightly by about 1% in one experiment. The maximum interarrival time test values in the tree selected during training mean smaller uniform loss rates do not change classification results much (there already are larger than normal interarrival times in the training data due to loss as explained earlier).[14]

In our experiments, DIFFUSE consumed <10 MB of RAM (for up to 10 000 simultaneous entries in the flow table), and only one of the four CPU cores was used with a maximum CPU load of <10% for maximum traffic rates of 50–60 Mb/s (10 000–15 000 packets per second). On our classifier PC, DIFFUSE can classify 400 000 packets per second using only one core, and in terms of CPU load, it scales well to up to 100 000 simultaneous flows [48]. Thus, plausibly one PC can monitor multiple 1-Gb/s links.

VIII. DISCUSSION

There are several noteworthy points about our proposed technique as well as a few limitations in our experimental analysis.

Sub-flow positions near the start of flows usually result in more training instances than positions toward the end of flows due to variations in flow length. This gives more weight to the characteristics at the start of flows and may create intraclass imbalance effects. Also, in our training sets, the number of sub-flows is different for each class (although the differences are not huge), which may lead to interclass imbalance problems. We selected sub-flows at different phases during a flow's lifetime and obtained the maximum number of instances available to train a classifier; this worked well with our datasets. An evaluation of the classifier's performance with intra- and interclass balancing (such as [49]) is left for future research.

Depending on the particular application of interest and the ML algorithm, there will be a trade-off for the selection of N. A very small N allows timely classification and reduces memory consumption, but may not be good enough to differentiate between different applications (and result in low accuracy). A large N can help eliminate short "junk" traffic, such as scans, probes, and failed connection attempts, hence reducing the load on the classifier. A large N also may improve the classifier's Precision and Recall, yet increase the time before a classification decision can be made. We performed similar experiments to the ones described using $N = 10$. For the NBayes classifier, the median Recall and Precision were 24.3% and 5% lower than for $N = 25$, respectively. For the C4.5 classifier, the median Recall and Precision were 0.5% and 5% lower than for $N = 25$, respectively. Detailed analysis of this tradeoff is future work.

In continuous classification, there is a possibility of flapping (oscillation) in classification results when monitoring traffic flows over their (potentially long) lifetime. This can be overcome by applying a filter scheme to verify the classification result before taking further action. For example, the classifier only sends an update of the flow classification result if it sees n new, consecutive, and identical results.[15] This technique was demonstrated to work well in [51].

Our training dataset had a mixture of traffic from different locations. This approach is practical when examples of traffic collected at a single monitor point are not sufficient for learning. This does not affect our results as it does not change the intraflow characteristics used for training and classification. However, the portability of the classifier (training in one location and classifying in another location) should be evaluated further in future work.

A limited number of common interference applications were used for training the classifier models. Extending the training dataset for the Other class is subject to future research. Sub-flow positions used to train the Other class were not optimized. One reason for this is that most of the Other class's flows are shorter than ET or VoIP flows. Further investigation on which sub-flow positions are best for the Other class may lead to improved results.

In our evaluation of the robustness against packet loss, we only applied 5% packet loss to ET and VoIP and assumed a simple random, independent loss process. This could be extended with studying the performance when packet loss also occurs to the Other traffic, using more complex loss models (for example, the Gilbert model [52]), and different packet loss rates. A deeper investigation into the sensitivity of the NBayes and

[14]However, for very high loss rates, VoIP Recall decreases greatly—e.g., with 25% packet loss byte, Recall decreases to 78%.

[15]The classifier then has *hysteresis* included, so that the result is sustained for a longer duration, with noise suppression during the steady state (e.g., [50]).

C4.5 classifiers and further evaluation of the effects of network perturbations are also left for future work.

How our approach would scale best to classify a large number (e.g., tens or hundreds) of classes and/or applications simultaneously is a question that requires further study.[16] Although a business scenario that requires the identification of many QoS classes simultaneously is not clear, it would be interesting to evaluate the tradeoffs between accuracy and computational performance when scaling to that extent. Even if there are only two classes, the question remains how many applications can be grouped into a single class before a degradation in classification accuracy and computational performance occurs.

In our experiments, sub-flows of 25 packets worked well in terms of classification timeliness, Precision, and Recall for both ET and VoIP traffic. However, another application may require a different size for an *optimal* classification window. An optimal size for a classification window should balance the tradeoff between the classifier's Precision and Recall, classification timeliness, classification speed, and processing overhead. Nevertheless, using one classifier for the classification of multiple applications simultaneously would require the same sub-flow size for all applications. A detailed characterization of this tradeoff is future research.

IX. Conclusion and Future Work

The ML-based approaches used in the literature for classifying IP traffic have relied on the analysis of statistics of full flows or their first few packets only. They do not allow a timely and continuous classification of flows needed for the QoS management of interactive traffic. In this paper, we proposed and analyzed the novel approach of training an ML classifier using a set of short sub-flows extracted from full flows generated by the target application(s) and augmented with their mirror-image replicas. We analyzed the performance of our proposed approach in experiments where first-person-shooter online game (ET) and voice (VoIP) traffic is detected among other IP traffic. Although the experiments are confined to online game and VoIP applications, our results reveal a potential solution to the accurate and timely classification of traffic belonging to other Internet applications.

The results showed that using a sub-flow size of $N = 25$ packets, the Naive Bayes classifier achieved 98.9% median Recall and 87% median Precision when classifying ET traffic, and 99.6% median Recall and 95.4% median Precision when classifying VoIP traffic. The C4.5 Decision Tree classifier achieved 99.3% median Recall and 97% median Precision when classifying ET traffic, and 95.7% median Recall and 99.2% Precision when classifying VoIP traffic. Both classifiers maintained their performance well regardless of how many packets are missed from the beginning of each flow or of the direction of the first packet of the most recent N packets.

Furthermore, we also demonstrated a novel approach of using unsupervised ML clustering techniques to choose appropriate, representative sub-flows, from which a classifier may be trained. This extension eliminates the need for expert knowledge of the application and relieves the complexity of manually choosing the best combination of sub-flows to train a classifier.

We also briefly explored the impact of packet loss on our proposed method. With a 5% random, independent synthetic packet loss, the Naive Bayes and C4.5 Decision Tree classifiers maintained their performance well. For ET traffic, the packet loss only degraded Recall and Precision of both classifiers by less than 0.5%. For VoIP traffic, the packet loss did not produce any noticeable degradation of the Naive Bayes classifier's Recall and Precision. However, it degraded the C4.5 Decision Tree classifier's Recall and Precision by 8.5% and 0.1%, respectively. Despite this degradation, median Recall and Precision of the C4.5 Decision Tree classifier remained above 87% and 99%, respectively.

Finally, we evaluated DIFFUSE—our publicly available, open-source implementation. We showed that DIFFUSE achieved 98%–99% classification accuracy for ET and VoIP traffic replayed across the network. Performance measurements we published in [48] showed that plausibly a high-end PC with DIFFUSE can monitor one or more 1-Gb/s links (assuming a realistic traffic mix). These results confirm the practicality of our proposal.

Overall our results clearly indicate the benefits of our novel approach, which is a significant improvement over the previous published state of the art for IP traffic classification. Our work has opened up a new path for research on the optimization of ML classifiers for timely and continuous real-time IP traffic classification. Our work can be extended in a number of future research directions:

- expanding our approach toward the recognition of different or new applications (e.g., video streaming);
- identifying how different sub-flow sizes affect classification accuracy, classification timeliness, classification speed, and the stability of results for continuous classification, as well as characterizing the optimal sub-flow size for applications;
- evaluating the impact of different traffic mixes and different ML classification and clustering algorithms on classification accuracy;
- evaluating the scalability of our approach to classify a large number (for example, hundreds) of applications simultaneously;
- evaluating the portability of classification models (when being applied in different network environments);
- evaluating the stability of classification accuracy and characterizing the sensitivity of different ML algorithms when using packet sampling, employing different features, and in the presence of network perturbations, such as packet loss, delay, and reordering.

Appendix
Data for Training and Testing in Each Experiment

The datasets used for training/testing for VoIP/ET contain flows of unequal lengths, hence the reduction of flow examples used for training/testing when the sub-flow position is chosen

[16] We evaluated the simultaneous classification of VoIP versus ET versus Other (one classifier with three classes). The classifier achieved more than 95% Recall and Precision for both ET and VoIP traffic. Due to the space limitation, detailed results are omitted.

TABLE III
APPLICATION SUB-FLOW INSTANCES FOR TRAINING

Application	Model							
	SF-0	SF-20	SF-40	SF-2000	SFsManSel	SFsAutoSel	AllSFs	SSP-ACT
ET	2567	1599	1459	1089	6716	12804	23892	47784
VoIP								2632
Other	15258	9275	9275	9275	24935	24935	24935	49870

TABLE IV
APPLICATION SUB-FLOW INSTANCES FOR TESTING

Application	Sub-flow position M																	
	0	10	20	30	40	50	60	70	80	90	1000	2000	3000	4000	5000	6000	7000	8000	9000
ET	2487	2201	2110	2063	2034	2016	2006	1996	1992	1978	1777	1691	1631	1593	1557	1526	1495	1468	1437
VoIP	235	234	233	233	232	232	231	230	230	229	187	156	135	123	111	104	92	84	81
Other			10662																

toward the end of the flows. For Other traffic, since all flows are short, only one sub-flow position is chosen for classifiers trained on a single sub-flow (e.g., SF-0, SF-20), and two sub-flow positions are chosen for classifiers trained on multiple sub-flows (including SFsManSel, SFAutoSel, and AllSFs models), and one position is chosen for testing. Detailed information is presented in Tables III and IV.

ACKNOWLEDGMENT

The authors would like to thank their colleagues L. Stewart and W. Harrop for their permission to use the VoIP dataset. They would also like to thank the anonymous reviewers for their very helpful feedback to improve the paper.

REFERENCES

[1] J. But, G. Armitage, and L. Stewart, "Outsourcing automated QoS control of home routers for a better online game experience," *IEEE Commun. Mag.*, vol. 46, no. 12, pp. 64–70, Dec. 2008.

[2] J. Frank, "Machine learning and intrusion detection: Current and future directions," in *Proc. 17th Nat. Comput. Security Conf.*, Oct. 1994, pp. 22–33.

[3] F. Baker, B. Foster, and C. Sharp, "Cisco architecture for lawful intercept in IP networks," RFC 3924, Oct. 2004.

[4] Planet Wolfenstein Enemy Territory, "Wolfenstein: Enemy Territory," Oct. 2010 [Online]. Available: http://www.planetwolfenstein.com/enemyterritory/

[5] P. A. Branch, A. Heyde, and G. J. Armitage, "Rapid identification of Skype traffic flows," in *Proc. 18th NOSSDAV*, 2009, pp. 91–96.

[6] J. But, P. Branch, and T. Le, "Rapid identification of BitTorrent traffic," in *Proc. 35th IEEE LCN*, Oct. 2010, pp. 536–543.

[7] T. Nguyen and G. Armitage, "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world IP networks," in *Proc. 31st IEEE LCN*, Nov. 2006, pp. 369–376.

[8] T. Nguyen and G. Armitage, "Synthetic sub-flow pairs for timely and stable IP traffic identification," in *Proc. Australian Telecommun. Netw. Appl. Conf.*, Dec. 2006, pp. 293–297.

[9] T. Nguyen and G. Armitage, "Clustering to assist supervised machine learning for real-time IP traffic classification," in *Proc. IEEE ICC*, 2008, pp. 5857–5862.

[10] S. Zander and G. Armitage, "DIstributed Firewall and Flow-Shaper Using Statistical Evidence (DIFFUSE)," 2010 [Online]. Available: http://caia.swin.edu.au/urp/diffuse/

[11] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *Proc. PAM*, Apr. 2004, pp. 205–214.

[12] 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 *Proc. 4th ACM SIGCOMM IMC*, Oct. 2004, pp. 135–148.

[13] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *Proc. IEEE 30th LCN*, Nov. 2005, pp. 250–257.

[14] A. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, Jun. 2005, pp. 50–60.

[15] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 223–239, Jan. 2007.

[16] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *Comput. Commun. Rev.*, vol. 36, no. 2, pp. 23–26, 2006.

[17] G. Gómez Sena and P. Belzarena, "Early traffic classification using support vector machines," in *Proc. 5th LANC*, 2009, pp. 60–66.

[18] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Comput. Netw.*, vol. 53, pp. 790–809, Apr. 2009.

[19] W. Li and A. W. Moore, "A machine learning approach for efficient traffic classification," in *Proc. Symp. Model., Anal., and Simul. Comput. Telecommun. Syst.*, 2007, pp. 310–317.

[20] J. Park, H.-R. Tyan, and C.-C. J. Kuo, "GA-based internet traffic classification technique for QoS provisioning," in *Proc IIHMSP*, Dec. 2006, pp. 251–254.

[21] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between Web and peer-to-peer traffic in the network core," in *Proc 16th WWW*, May 2007, pp. 883–892.

[22] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Semisupervised network traffic classification," *Perform. Eval. Rev.*, vol. 35, no. 1, pp. 369–370, 2007.

[23] C. Rotsos, J. Van Gael, A. W. Moore, and Z. Ghahramani, "Probabilistic graphical models for semi-supervised traffic classification," in *Proc. 6th IWCMC*, 2010, pp. 752–757.

[24] F. Rodríguez-Teja, C. Martinez-Cagnazzo, and E. G. Castro, "Bayesian classification: Methodology for network traffic classification combination," in *Proc. 6th IWCMC*, 2010, pp. 769–773.

[25] M. Crotti, F. Gringoli, and L. Salgarelli, "Optimizing statistical classifiers of network traffic," in *Proc. 6th IWCMC*, 2010, pp. 758–763.

[26] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proc. SIGCOMM MineNet*, 2006, pp. 281–286.

[27] J. Erman, A. Mahanti, and M. Arlitt, "QRP05-4: Internet traffic identification using machine learning," in *Proc. IEEE GLOBECOM*, Dec. 2006, pp. 1–6.

[28] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proc. ACM CoNEXT*, 2008, pp. 1–12.

[29] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: Automated construction of application signatures," in *Proc. ACM SIGCOMM MineNet*, Aug. 2005, pp. 197–202.

[30] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *Comput. Commun. Rev.*, vol. 36, no. 5, pp. 5–16, 2006.

[31] F. Palmieri and U. Fiore, "A nonlinear, recurrence-based approach to traffic classification," *Comput. Netw.*, vol. 53, pp. 761–773, Apr. 2009.

[32] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, 4th Quart., 2008.

[33] A. Callado, C. Kamienski, S. Fernandes, D. Sadok, G. Szabo, and B. P. Ger, "A survey on internet traffic identification and classification," *IEEE Commun. Surveys Tutorials*, vol. 11, no. 3, pp. 37–52, 3rd Quart., 2009.
[34] R. Kohavi, J. R. Quinlan, W. Klosgen, and J. Zytkow, "Decision tree discovery," *Handbook Data Mining Knowl. Discovery*, pp. 267–276, 2002.
[35] G. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. 11th Conf. Uncertainty Artif. Intell.*, Aug. 1995, pp. 338–345.
[36] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations*, 2nd ed. San Mateo, CA: Morgan Kaufmann, 2005.
[37] C. Schmoll and S. Zander, "NetMate," Oct. 2010 [Online]. Available: http://sourceforge.net/projects/netmate-meter/
[38] G. Armitage, M. Claypool, and P. Branch, *Networking and Online Games—Undertanding and Engineering Multiplayer Internet Games*. Hoboken, NJ: Wiley, 2006.
[39] The University of Twente, "Traffic measurement data repository," Mar. 2009 [Online]. Available: http://traces.simpleweb.org/
[40] J.-C. Bolot, "End-to-end packet delay and loss behavior in the Internet," *Comput. Commun. Rev.*, vol. 23, no. 4, pp. 289–298, 1993.
[41] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *Proc. 7th ACM SIGCOMM IMC*, 2007, pp. 43–56.
[42] M. Mathis, J. Semke, and J. Mahdavi, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Comput. Commun. Rev.*, vol. 27, no. 3, pp. 67–82, 1997.
[43] R.-H. Li and G. G. Belford, "Instability of decision tree classification algorithms," in *Proc. 8th ACM KDD*, 2002, pp. 570–575.
[44] J. J. Barbish and B. Davis, "FreeBSD Handbook, Chapter 30 Firewalls," 2012 [Online]. Available: http://www.freebsd.org/doc/en/books/handbook/firewalls.html
[45] L. Stewart, G. Armitage, P. Branch, and S. Zander, "An architecture for automated network control of QoS over consumer broadband links," in *Proc. IEEE Region 10 Conf. (Tencon)*, Nov. 21–24, 2005, pp. 1–6.
[46] J. But, N. Williams, S. Zander, L. Stewart, and G. Armitage, "Automated network games enhancement layer—A proposed architecture," in *Proc. ACM SIGCOMM NetGames*, Oct. 2006, Article no. 9.
[47] A. Turner, Tcpreplay [Online]. Available: http://tcpreplay.synfin.net
[48] S. Zander and G. Armitage, "Practical machine learning based multimedia traffic classification for distributed QoS management," in *Proc. 36th IEEE LCN*, Oct. 2011, pp. 399–406.
[49] A. Nickerson, N. Japkowicz, and E. Milios, "Using unsupervised learning to guide resampling in imbalanced data sets," in *Proc. 8th Int. Workshop Artif. Intell. Statist.*, 2001, pp. 261–265.
[50] T. G. Renna, I. Bar-Kana, and P. Kalata, "A two-level gain stochastic disturbance observer with hysteresis," in *Proc. IEEE Int. Conf. Syst. Eng.*, Aug. 1990, pp. 77–80.
[51] J. But, T. Nguyen, L. Stewart, N. Williams, and G. Armitage, "Performance analysis of the ANGEL system for automated control of game traffic prioritisation," in *Proc. 6th ACM SIGCOMM NetGames*, Sep. 2007, pp. 123–128.
[52] W. Jiang and H. Schulzrinne, "Comparison and optimization of packet loss repair methods on VoIP perceived quality under bursty loss," in *Proc. 12th NOSSDAV*, 2002, pp. 73–81.

Thuy T. T. Nguyen received the Bachelor's degree (First Class Honour) from the University of Technology, Sydney, Australia, in 2003, and the Ph.D. degree from Swinburne University of Technology (SUT), Melbourne, Australia, in 2009, both in telecommunications engineering.

She is currently a Lecturer and Researcher in telecommunications with SUT. Her research interests include Internet quality of service, traffic classification, Internet pricing, and energy efficiency for the Internet.

Grenville Armitage (M'03) received the B.Eng. degree (Hons.) in electrical engineering and the Ph.D. degree in electronic engineering from the University of Melbourne, Melbourne, Australia, in 1988 and 1994, respectively.

He is currently a Professor of telecommunications engineering and Director of the Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia. He authored *Quality of Service In IP Networks: Foundations for a Multi-Service Internet* (Macmillan, 2000) and coauthored *Networking and Online Games—Understanding and Engineering Multiplayer Internet Games* (Wiley, 2006).

Prof. Armitage is a member of the Association for Computing Machinery (ACM) and the ACM Special Interest Group on Data Communication (SIGCOMM).

Philip Branch received the B.Sc. degree in mathematics from the University of Tasmania, Tasmania, Australia, in 1981, and the Ph.D. degree in telecommunications engineering from Monash University, Melbourne, Australia, in 2000.

He is a Senior Lecturer with the Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia. Before joining Swinburne, he worked for a telecommunications equipment manufacturer, an Internet startup, and a joint-venture funded by Monash University, Telstra, and Siemens to investigate the capabilities of the then emerging broadband networks. His research interests are in the application of machine learning to telecommunications network management, characterization of teletraffic generated by online games, network security, and lawful interception.

Sebastian Zander received the Dipl.-Ing. degree in applied computer science from the Technical University Berlin, Berlin, Germany, in 1999, and the Ph.D. degree in telecommunications engineering from Swinburne University of Technology, Melbourne, Australia, in 2010.

From 1999 to 2004, he worked as Researcher and Project Manager with Fraunhofer FOKUS, Berlin, Germany. Since 2010, he has been a Research Fellow with the Centre for Advanced Internet Architectures (CAIA), Swinburne University of Technology. His research interests include IPv6, network measurement, traffic classification, and network security.