A Survey of Techniques for Internet Traffic Classification using Machine Learning

Thuy T.T. Nguyen, Grenville Armitage
Centre for Advanced Internet Architectures.
Swinburne University of Technology
Melbourne, Australia
Email: tnguyen@swin.edu.au, garmitage@swin.edu.au

***Abstract*—The research community has begun looking for IP traffic classification techniques that do not rely on 'well known' TCP or UDP port numbers, or interpreting the contents of packet payloads. New work is emerging on the use of statistical traffic characteristics to assist in the identification and classification process. This survey paper looks at emerging research into the application of Machine Learning (ML) techniques to IP traffic classification - an inter-disciplinary blend of IP networking and data mining techniques. We provide context and motivation for the application of ML techniques to IP traffic classification, and review 18 significant works that cover the dominant period from 2004 to early 2007. These works are categorized and reviewed according to their choice of ML strategies and primary contributions to the literature. We also discuss a number of key requirements for the employment of ML-based traffic classifiers in operational IP networks, and qualitatively critique the extent to which the reviewed works meet these requirements. Open issues and challenges in the field are also discussed.**

I. Introduction

Real-time traffic classification has the potential to solve difficult network management problems for Internet service providers (ISPs) and their equipment vendors. Network operators need to know what is flowing over their networks promptly so they can react quickly in support of their various business goals. Traffic classification may be a core part of automated intrusion detection systems [1] [2] [3], used to detect patterns indicative of denial of service attacks, trigger automated re-allocation of network resources for priority customers [4], or identify customer use of network resources that in some way contravenes the operator's terms of service. More recently, governments are also clarifying ISP obligations with respect to 'lawful interception' (LI) of IP data traffic [5]. Just as telephone companies must support interception of telephone usage, ISPs are increasingly subject to government requests for information on network use by particular individuals at particular points in time. IP traffic classification is an integral part of ISP-based LI solutions.

Commonly deployed IP traffic classification techniques have been based around direct inspection of each packet's contents at some point on the network. Successive IP packets having the same 5-tuple of protocol type, source address:port and destination address:port are considered to belong to a *flow* whose controlling application we wish to determine. Simple classification infers the controlling application's identity by assuming that most applications consistently use 'well known' TCP or UDP port numbers (visible in the TCP or UDP headers). However, many applications are increasingly using unpredictable (or at least obscure) port numbers [6]. Consequently, more sophisticated classification techniques infer application type by looking for application-specific data (or well-known protocol behavior) within the TCP or UDP payloads [7].

Unfortunately, the effectiveness of such 'deep packet inspection' techniques is diminishing. Such packet inspection relies on two related assumptions:

- Third parties unaffiliated with either source or recipient are able to inspect each IP packet's payload (i.e. is the payload visible)
- The classifier knows the syntax of each application's packet payloads (i.e. can the payload be interpreted)

Two emerging challenges undermine the first assumption - customers may use encryption to obfuscate packet contents (including TCP or UDP port numbers), and governments may impose privacy regulations constraining the ability of third parties to lawfully inspect payloads at all. The second assumption imposes a heavy operational load - commercial devices will need repeated updates to stay ahead of regular (or simply gratuitous) changes in every application's packet payload formats.

The research community has responded by investigating classification schemes capable of inferring application-level usage patterns without deep inspection of packet payloads. Newer approaches classify traffic by recognising statistical patterns in externally observable attributes of the traffic (such as typical packet lengths and inter-packet arrival times). Their ultimate goal is either clustering IP traffic flows into groups that have similar traffic patterns, or classifying one or more applications of interest.

A number of researchers are looking particularly closely at the application of Machine Learning (ML) techniques (a subset of the wider Artificial Intelligence discipline) to IP traffic classification. The application of ML techniques involves a number of steps. First, *features* are defined by which future unknown IP traffic may be identified and differentiated. Features are attributes of flows calculated over multiple packets (such as maximum or minimum packet lengths in each direction, flow durations or inter-packet arrival times). Then the ML classifier is trained to associate sets of features with known traffic classes (creating rules), and apply the ML algorithm

Accepted 16 Nov 2007 for 4th edition 2008 of IEEE Communications Surveys and Tutorials (final version copyright IEEE)
Distribution of pre-publication copies approved by Surveys & Tutorials editor-in-chief Nelson Fonseca, nfonseca@ic.unicamp.br, on 12 Mar 2008

to classify unknown traffic using previously learned rules. Every ML algorithm has a different approach to sorting and prioritising sets of features, which leads to different dynamic behaviors during training and classification. In this paper we provide the rationale for IP traffic classification in IP networks, review the state-of-the-art approaches to traffic classification, and then review and critique emerging ML-based techniques for IP traffic classification.

The rest of this paper is organised as follows. Section II outlines the importance of IP traffic classification in operational networks, introduces a number of metrics for assessing classification accuracy, and discusses the limitations of traditional port- and payload-based classification. Section III provides background information about ML and how it can be applied in IP traffic classification. The section also discusses a number of key requirements for the employment of ML-based classifiers in operational IP networks. Section IV reviews the significant works in this field (predominantly from 2004 to early 2007). The section is wrapped up with a qualitative discussion of the extent to which the reviewed works meet the requirements specified in section III. Section V concludes the paper with some final remarks and suggestions of possible future work.

II. Application Context for Machine Learning Based IP Traffic Classification

A. The importance of IP traffic classification

The importance of IP traffic classification may be illustrated by briefly reviewing to important areas - IP quality of service (QoS) schemes, and lawful interception (LI).

In responding to the network congestion problem, a common strategy for network providers is under-utilising (over-provisioning) the link capacity. However, this is not necessarily an economic solution for most ISPs. On the other hand, the development of other QoS solutions such as IntServ [8] or DiffServ [9] has been stymied in part due to the lack of QoS signaling and of an effective service pricing mechanism (as suggested in [10] and [11]). Signaling allows the communication of specific QoS requirements between Internet applications and the network. A pricing mechanism is needed to differentiate customers with different needs and charge for the QoS that they receive. It also acts as a cost recovery mechanism and provides revenue generation for the ISPs to compensate for their efforts in providing QoS and managing resource allocation.

All QoS schemes have some degree of IP traffic classification implicit in their design. DiffServ assumes that edge routers can recognise and differentiate between aggregate classes of traffic in order to set the DiffServ code point (DSCP) on packets entering the network core. IntServ presumes that routers along a path are able to differentiate between finely grained traffic classes (and historically has presumed the use of packet header inspection to achieve this goal). Traffic classification also has the potential to support class-based Internet QoS charging. Furthermore, real-time traffic classification is the core component of emerging QoS-enabled products [12] and automated QoS architectures [4] [13].

Traffic classification is also an important solution for the emerging requirement that ISP networks have to provide LI capabilities. Governments typically implement LI at various levels of abstraction. In the telephony world a law enforcement agency may nominate a 'person of interest' and issue a warrant for the collection of intercept information. The intercept may be high-level call records (who called who and when) or low-level 'tapping' of the audio from actual phone calls in progress. In the ISP space, traffic classification techniques offer the possibility of identifying traffic patterns (which endpoints are exchanging packets and when), and identifying what classes of applications are being used by a 'person of interest' at any given point in time. Depending on the particular traffic classification scheme, this information may potentially be obtained without violating any privacy laws covering the TCP or UDP payloads of the ISP customer's traffic.

B. Traffic classification metrics

A key criterion on which to differentiate between classification techniques is predictive accuracy (i.e., how accurately the technique or model makes decisions when presented with previously unseen data). A number of metrics exist with which to express predictive accuracy.

1) Positives, negatives, accuracy, precision and recall: Assume there is a traffic class X in which we are interested, mixed in with a broader set of IP traffic. A traffic classifier is being used to identify (classify) packets (or flows of packets) belonging to class X when presented with a mixture of previously unseen traffic. The classifier is presumed to give one of two outputs - a flow (or packet) is believed to be a member of class X, or it is not.

A common way to characterize a classifier's accuracy is through metrics known as *False Positives*, *False Negatives*, *True Positives* and *True Negatives*. These metrics are defined as follows:

- *False Negatives* (FN): Percentage of members of class X incorrectly classified as not belonging to class X.
- *False Positives* (FP): Percentage of members of other classes incorrectly classified as belonging to class X.
- *True Positives* (TP): Percentage of members of class X correctly classified as belonging to class X (equivalent to 100% - FN).
- *True Negatives* (TN): Percentage of members of other classes correctly classified as not belonging to class X (equivalent to 100% - FP).

Figure 1 illustrates the relationships between FN, FP, TP and TN. A good traffic classifier aims to minimise the False Negatives and False Positives.

Some works make use of *Accuracy* as an evaluation metric. It is generally defined as the percentage of correctly classified instances among the total number of instances. This definition is used throughout the paper unless otherwise stated.

ML literature often utilises two additional metrics known as *Recall* and *Precision*. These metrics are defined as follows:

Classified as →	X	$\overline{X}$
X	TP	FN
$\overline{X}$	FP	TN

Fig. 1. Evaluation Metrics

- *Recall*: Percentage of members of class X correctly classified as belonging to class X.
- *Precision*: Percentage of those instances that truly have class X, among all those classified as class X.

If all metrics are considered to range from 0 (bad) to 100% (optimal) it can be seen that Recall is equivalent to TP.

2) Byte and Flow accuracy: When comparing literature on different classification techniques it is also important to note the unit of the author's chosen metric. Recall, Precision, FN and FP may all be reported as percentages of bytes or flows relative to the traffic being classified. An author's choice here can significantly alter the meaning of their reported accuracy results.

Most recently published traffic classification studies have focused on *flow accuracy* - measuring the accuracy with which flows are correctly classified, relative to the number of other flows in the author's test and/or training dataset(s). However, some recent work has also chosen to express their accuracy calculations in terms of *byte accuracy* - focusing more on how many bytes are carried by the packets of correctly classified flows, relative to the total number of bytes in the author's test and/or training dataset(s) (e.g. [14] [15]).

Erman et al. in [16] argue that byte accuracy is crucial when evaluating the accuracy of traffic classification algorithms. They note that the majority of flows on the Internet are small and account for only a small portion of total bytes and packets in the network (*mice* flows). On the other hand, the majority of the traffic bytes are generated by a small number of large flows (*elephant* flows). They give an example from a 6-month data trace where the top (largest) 1% of flows account for over 73% of the traffic in terms of bytes. With a threshold to differentiate elephant and mice flows of 3.7MB, the top 0.1% of flows would account for 46% of the traffic (in bytes). Presented with such a dataset, a classifier optimised to identify all but the top 0.1% of the flows could attain a 99.9% flow accuracy but still result in 46% of the bytes in the dataset to be misclassified.

Whether flow accuracy or byte accuracy is more important will generally depend on the classifier's intended use. For example, when classifying traffic for IP QoS purposes it is plausible that identifying every instance of a short lived flow needing QoS (such as a 5 minute, 32Kbit/sec phone calls) is as important as identifying long lived flows needing QoS (such as a 30 minute, 256Kbit/sec video conference) with both being far more important to correctly identify than the few flows that represent multi-hour (and/or hundreds of megabytes) peer to peer file sharing sessions. Conversely, an ISP doing analysis of load patterns on their network may well be significantly interested in correctly classifying the applications driving the elephant flows that contribute a disproportionate number of packets across their network.

C. Limitations of packet inspection for traffic classification

Traditional IP traffic classification relies on the inspection of a packet's TCP or UDP port numbers (*port based classification*), or the reconstruction of protocol signatures in its payload (*payload based classification*). Each approach suffers from a number of limitations.

1) Port based IP traffic classification: TCP and UDP provide for the multiplexing of multiple flows between common IP endpoints through the use of port numbers. Historically many applications utilise a 'well known' port on their local host as a rendezvous point to which other hosts may initiate communication. A classifier sitting in the middle of a network need only look for TCP SYN packets (the first step in TCP's three-way handshake during session establishment) to know the server side of a new client-server TCP connection. The application is then inferred by looking up the TCP SYN packet's target port number in the Internet Assigned Numbers Authority (IANA)'s list of registered ports [17]. UDP uses ports similarly (though without connection establishment nor the maintenance of connection state).

However, this approach has limitations. Firstly, some applications may not have their ports registered with IANA (for example, peer to peer applications such as Napster and Kazaa) [18]. An application may use ports other than its well-known ports to avoid operating system access control restrictions (for example, non-privileged users on unix-like systems may be forced to run HTTP servers on ports other than port 80.) Also, in some cases server ports are dynamically allocated as needed. For example, the RealVideo streamer allows the dynamic negotiation of the server port used for the data transfer. This server port is negotiated on an initial TCP connection, which is established using the well-known RealVideo control port [19].

Moore and Papagiannaki [20] observed no better than 70% byte accuracy for port-based classification using the official IANA list. Madhukar and Williamson [21] showed that port-based analysis is unable to identify 30-70% of Internet traffic flows they investigated. Sen et al. [7] reported that the default port accounted for only 30% of the total traffic (in bytes) for the Kazaa P2P protocol.

In some circumstances IP layer encryption may also obfuscate the TCP or UDP header, making it impossible to know the actual port numbers.

2) Payload based IP traffic classification: To avoid total reliance on the semantics of port numbers, many current industry products utilise stateful reconstruction of session and application information from each packet's content.

Sen et al. [7] showed that payload based classification of P2P traffic (by examining the signatures of the traffic at the application level) could reduce false positives and false negatives to 5% of total bytes for most P2P protocols studied.

Moore and Papagiannaki [20] use a combination of port and payload based techniques to identify network applications. The classification procedure starts with the examination of a flow's port number. If no well-known port is used, the flow is passed through to the next stage. In the second stage, the first packet is examined to see whether it contains a known signature. If one is not found, then the packet is examined to see whether it contains a well-known protocol. If these tests fail, the protocol signatures in the first KByte of the flow are studied. Flows remaining unclassified after that stage will require inspection of the entire flow payload. Their results show that port information by itself is capable of correctly classifying 69% of the total bytes. Including the information observed in the first KByte of each flow increases the accuracy to almost 79%. Higher accuracy (upto nearly 100%) can only be achieved by investigating the remaining unclassified flows' entire payload.

Although payload based inspection avoids reliance on fixed port numbers, it imposes significant complexity and processing load on the traffic identification device. It must be kept up-to-date with extensive knowledge of application protocol semantics, and must be powerful enough to perform concurrent analysis of a potentially large number of flows. This approach can be difficult or impossible when dealing with proprietary protocols or encrypted traffic. Furthermore direct analysis of session and application layer content may represent a breach of organisational privacy policies or violation of relevant privacy legislation.

D. Classification based on statistical traffic properties

The preceding techniques are limited by their dependence on the inferred semantics of the information gathered through deep inspection of packet content (payload and port numbers). Newer approaches rely on traffic's statistical characteristics to identify the application. An assumption underlying such methods is that traffic at the network layer has statistical properties (such as the distribution of flow duration, flow idle time, packet inter-arrival time and packet lengths) that are unique for certain classes of applications and enable different source applications to be distinguished from each other.

The relationship between the class of traffic and its observed statistical properties has been noted in [22] (where the authors analysed and constructed empirical models of connection characteristics - such as bytes, duration, arrival periodicity - for a number of specific TCP applications), and in [23] (where the authors analysed Internet chat systems by focusing on the characteristics of the traffic in terms of flow duration, packet inter-arrival time and packet size and byte profile). Later work (for example [24] [25] and [26]) also observed distinctive traffic characteristics, such as the distributions of packet lengths and packet inter-arrival times, for a number of Internet applications. The results of these works have stimulated new classification techniques based on traffic flow statistical properties. The need to deal with traffic patterns, large datasets and multi-dimensional spaces of flow and packet attributes is one of the reasons for the introduction of ML techniques in this field.

III. Background on Machine Learning and the Application of Machine Learning in IP Traffic Classification

This section summaries the basic concepts of Machine Learning (ML) and outlines how ML can be applied to IP traffic classification.

A. A review of classification with Machine Learning

ML has historically been known as a collection of powerful techniques for data mining and knowledge discovery, which search for and describe useful structural patterns in data. In 1992 Shi [27] noted *'One of the defining features of intelligence is the ability to learn.* [...]. *Machine learning is the study of making machines acquire new knowledge, new skills, and reorganise existing knowledge.'* A learning machine has the ability to learn automatically from experience and refine and improve its knowledge base. In 1983 Simon noted *'Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time'* [28] and in 2000 Witten and Frank observed *'Things learn when they change their behavior in a way that makes them perform better in the future'* [29].

ML has a wide range of applications, including search engines, medical diagnosis, text and handwriting recognition, image screening, load forecasting, marketing and sales diagnosis, and so on. A network traffic controller using ML techniques was proposed in 1990, aiming to maximise call completion in a circuit-switched telecommunications network [30]; this was one of the works that marked the point at which ML techniques expanded their application space into the telecommunications networking field. In 1994 ML was first utilised for Internet flow classification in the context of intrusion detection [31]. It is the starting point for much of the work using ML techniques in Internet traffic classification that follows.

1) Input and output of a ML process: Broadly speaking, ML is the process of finding and describing structural patterns in a supplied dataset.

ML takes input in the form of a *dataset* of *instances* (also known as *examples*). An instance refers to an individual, independent example of the dataset. Each instance is characterised by the values of its *features* (also known as *attributes* or *discriminators*) that measure different aspects of the instance. (In the networking field consecutive packets from the same flow might form an instance, while the set of features might include median inter-packet arrival times or standard deviation of packet lengths over a number of consecutive packets in a flow.) The dataset is ultimately presented as a matrix of instances versus features [29].

The output is the description of the knowledge that has been learnt. How the specific outcome of the learning process is represented (the syntax and semantics) depends largely on the particular ML approach being used.

2) Different types of learning: Witten and Frank [29] define four basic types of learning:

- Classification (or *supervised learning*)
- Clustering (or *unsupervised learning*)
- Association
- Numeric prediction

Classification learning involves a machine learning from a set of pre-classified (also called pre-labeled) examples, from which it builds a set of classification rules (a *model*) to classify unseen examples. Clustering is the grouping of instances that have similar characteristics into clusters, without any prior guidance. In association learning, any association between features is sought. In numeric prediction, the outcome to be predicted is not a discrete class but a numeric quantity.

Most ML techniques used for IP traffic classification focus on the use of supervised and unsupervised learning.

3) Supervised Learning: Supervised learning creates knowledge structures that support the task of classifying new instances into pre-defined classes [32]. The learning machine is provided with a collection of sample instances, pre-classified into classes. Output of the learning process is a classification model that is constructed by examining and generalising from the provided instances.

In effect, supervised learning focuses on modeling the input/output relationships. Its goal is to identify a mapping from input features to an output class. The knowledge learnt (e.g. commonalities among members of the same class and differences between competing ones) can be presented as a flowchart, a decision tree, classification rules, etc., that can be used later to classify a new unseen instance.

There are two major phases (steps) in supervised learning:

- *Training*: The learning phase that examines the provided data (called the training dataset) and constructs (builds) a classification model.
- *Testing* (also known as classifying): The model that has been built in the training phase is used to classify new unseen instances.

For example, let TS be a training dataset, that is a set of input/output pairs,

TS = $\{< x_1, y_1 >, < x_2, y_1 >, ..., < x_N, y_M >\}$

where x_i is the vector of values of the input features corresponding to the i^{th} instance, and y_i is its output class value. (The name *supervised learning* comes from the fact that the output classes are pre-defined in the training dataset.) The goal of classification can be formulated as follows: From a training dataset TS, find a function f(x) of the input features that best predicts the outcome of the output class y for any new unseen values of x. The output takes its value in a discrete set $\{y_1, y_2, ..., y_M\}$ that consists of all the pre-defined class values. The function f(x) is the core of the classification model.

The model created during training is improved if we simultaneously provide examples of instances that belong to class(es) of interest and instances known to *not* be members of the class(es) of interest. This will enhance the model's later ability to identify instances belonging to class(es) of interest.

There exist a number of supervised learning classification algorithms, each differing mainly in the way the classification model is constructed and what optimization algorithm is used to search for a good model. (Examples include the supervised Decision Tree and Naive Bayes classification algorithms [33] [29].)

4) Clustering: Classification techniques use pre-defined classes of training instances. In contrast, clustering methods are not provided with this guidance; instead, they discover natural clusters (groups) in the data using internalised heuristics [33].

Clustering focuses on finding patterns in the input data. It clusters instances with similar properties (defined by a specific distance measuring approach, such as Euclidean space) into groups. The groups that are so identified may be exclusive, so that any instance belongs in only one group; or they may be overlapping, when one instance may fall into several groups; they may also be probabilistic, that is an instance belongs to a group with a certain probability. They may be hierarchical, where there is a division of instances into groups at the top level, and then each of these groups is refined further - even down to the level of individual instances [29].

There are three basic clustering methods: the classic k-means algorithm, incremental clustering, and the probability-based clustering method. The classic k-means algorithm forms clusters in numeric domains, partitioning instances into disjoint clusters, while incremental clustering generates a hierarchical grouping of instances. The probability-based methods assign instances to classes probabilistically, not deterministically [29].

5) Evaluating supervised learning algorithms: A good ML classifier would optimise Recall and Precision. However, there may be trade-offs between these metrics. To decide which one is more important or should be given higher priority one needs to take into account the cost of making wrong decisions or wrong classifications. The decision depends on a specific application context and ones commercial and/or operational priorities.

Various tools exist to support this trade-off process. The *receiver operating characteristic* (ROC) curve provides a way to visualize the trade-offs between TP and FP by plotting the number of TP as a function of the number of FP (both expressed as percentage of the total TP and FP respectively). It has been found useful in analysing how classifiers perform over a range of threshold settings [29]. Another is the Neyman-Pearson criterion [34], which attempts to maximize TP given a fixed threshold on FP [35].

Most of IP classification work reviewed later in this survey do not address the costs of trading between Recall and Precision.

A challenge when using supervised learning algorithms is that both the training and testing phases must be performed using datasets that have been previously classified (labeled)

[1]. Ideally one would have a large training set (for optimal learning and creation of models) and a large, yet independent, testing dataset to properly assess the algorithm's performance. (Testing on the training dataset is broadly misleading. Such testing will usually only show that the constructed model is good at recognising the instances from which it was constructed.)

In the real world we are often faced with a limited quantity of pre-labeled datasets. A simple procedure (sometimes known as *holdout* [29]) involves setting aside some part (e.g. two thirds) of the pre-labeled dataset for training and the rest (e.g. one third) for testing.

In practice when only small or limited datasets are available a variant of holdout, called *N-fold cross-validation*, is most commonly used. The dataset is first split into N approximately equal partitions (or folds). Each partition $(1/N)$ in turn is then used for testing, while the remainder $((N-1)/N)$ are used for training. The procedure is repeated N times so that in the end, every instance has been used exactly once for testing. The overall Recall and Precision are calculated from the average of the Recalls and Precisions measured from all N tests. It has been claimed that N = 10 (tenfold cross-validation) provides a good estimate of classification performance [29].

Simply partitioning the full dataset N ways does not guarantee that the same proportion is used for any given class within the dataset. A further step, known as *stratification*, is usually applied - randomly sampling the dataset in such a way that each class is properly represented in both training and testing datasets. When stratification is used in combination with cross-validation, it is called *stratified cross-validation*. It is common to use stratified ten-fold cross-validation when only limited pre-labeled datasets are available.

6) Evaluating unsupervised learning algorithms: While Recall and Precision are common metrics to evaluate classification algorithms, evaluating clustering algorithms is more complicated. There are intermediate steps in evaluating the resulting clusters before labeling them or generating rules for future classification. Given a dataset, a clustering algorithm can always generate a division, with its own finding of structure within the data. Different approaches can lead to different clusters, and even for the same algorithm, different parameters or different order of input patterns might alter the final results [36] [37].

Therefore, it is important to have effective evaluation standards and criteria to provide the users with a certain level of confidence in results generated by a particular algorithm, or comparisons of different algorithms [38]. Criteria should help to answer useful questions such as how many clusters are hidden in the data, what are the optimal number of clusters [37], whether the resulted clusters are meaningful or just an artifact of the algorithms [38], how one algorithm performs compared to another: how easy they are to use, how fast it is to be employed [36], what is the intra-cluster quality, how good is inter-cluster separation, what is the cost of labeling the clusters and what are the requirements in terms of computer computation and storage.

Halkidi et al. [37] identify three approaches to investigate cluster validity: external criteria, internal criteria and relative criteria. The first two approaches are based on statistical hypothesis testing. External criteria are based on some pre-specified structure, which is known as prior information on the data, and used as a standard to compare and validate the clustering results [38]. Internal criteria approach evaluates clustering result of an algorithm based on examining the internal structure inherited from the dataset. Relative criteria emphasises finding the best clustering scheme that a clustering algorithm can define under certain assumptions and parameters. The basic idea is to evaluate a clustering structure by comparing it to the ones using the same algorithm but with different parameter values [39]. (More details can be found in [37] [38] [36] [29].)

7) Feature selection algorithms: Key to building a ML classifier is identification of the smallest necessary set of features required to achieve one's accuracy goals - a process known as *feature selection.*

The quality of the feature set is crucial to the performance of a ML algorithm. Using irrelevant or redundant features often leads to negative impacts on the accuracy of most ML algorithms. It can also make the system more computationally expensive, as the amount of information stored and processed rises with the dimensionality of a feature set used to describing the data instances. Consequently it is desirable to select a subset of features that is small in size yet retains essential and useful information about the classes of interest.

Feature selection algorithms can be broadly classified into filter method or wrapper method. Filter method algorithms make independent assessment based on general characteristics of the data. They rely on a certain metric to rate and select the best subset before the learning commences. The results provided therefore should not be biased toward a particular ML algorithm. Wrapper method algorithms, on the other hand, evaluate the performance of different subsets using the ML algorithm that will ultimately be employed for learning. Its results are therefore biased toward the ML algorithm used. A number of subset search techniques can be used, e.g. Correlation-based Feature Selection (CFS) filter techniques with Greedy, Best-First or Genetic search. (Additional details can be found in [29] [40] [41] [42] [43].)

B. The application of ML in IP traffic classification

A number of general ML concepts take a specific meaning when applied to IP traffic classification. For the purpose of subsequent discussion we define the following three terms relating to flows:

- *Flow* or *Uni-directional flow*: A series of packets sharing the same five-tuple: source and destination IP addresses, source and destination IP ports and protocol number.

[1]In this context 'labeling' is the process of classifying the members of a dataset using manual (human) inspection or an irrefutable automated process. In contrast to a controlled training and testing environment, operational classifiers do not have access to previously labeled example flows.

- *Bi-directional flow*: A bi-directional flow is a pair of uni-directional flows going in the opposite directions between the same source and destination IP addresses and ports.
- *Full-flow*: A bi-directional flow captured over its entire lifetime, from the establishment to the end of the communication connection.

A class usually indicates IP traffic caused by (or belonging to) an application or group of applications. Instances are usually multiple packets belonging to the same flow. Features are typically numerical attributes calculated over multiple packets belonging to individual flows. Examples include mean packet lengths, standard deviation of inter-packet arrival times, total flow lengths (in bytes and/or packets), Fourier transform of packet inter-arrival time, and so on. As previously noted not all features are equally useful, so practical ML classifiers choose the smallest set of features that lead to efficient differentiation between members of a class and other traffic outside the class.

1) Training and testing a supervised ML traffic classifier: Figures 2, 3 and 4 illustrate the steps involved in building a traffic classifier using a supervised learning (or *supervised ML*) algorithm. In this example, the traffic classifier is intended to recognise a particular class of applications (real-time online game traffic) in amongst the usual mix of traffic seen on an IP network.

Figure 2 captures the overall training and testing process that results in a classification model. As noted earlier, the optimal approach to training a supervised ML algorithm is to provide previously classified examples of two types of IP traffic: traffic matching the class of traffic that one wishes later to identify in the network (in this case online game traffic), and representative traffic of entirely different applications one would expect to see in future (often referred to as *interfering traffic*).

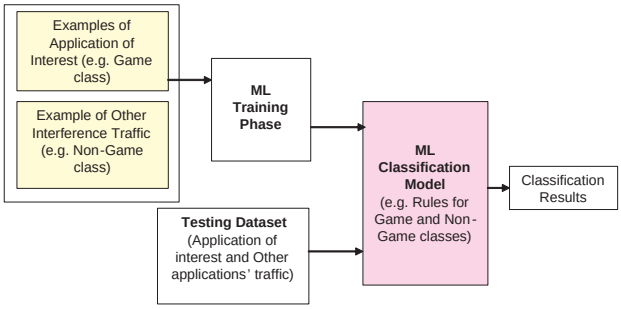

Fig. 2. Training and testing for a two-class supervised ML traffic classifier

Figure 3 expands on the sequence of events involved in training a supervised ML traffic classifier. First a mix of 'traffic traces' are collected that contain both instances of the application of interest (e.g. online game traffic) and instances of other interfering applications (such as HTTP, DNS, SSH and/or peer2peer file sharing). The 'flow statistics processing' step involves calculating the statistical properties of these flows (such as mean packet inter-arrival time, median packet length and/or flow duration) as a prelude to generating features.

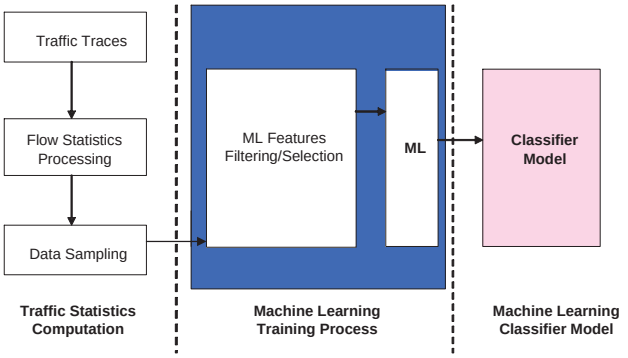

Fig. 3. Training the supervised ML traffic classifier

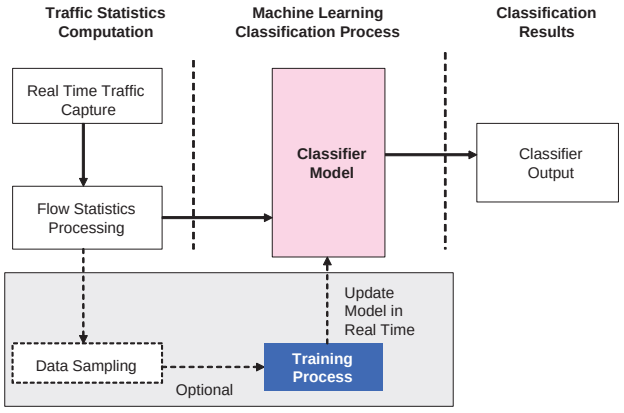

Fig. 4. Data flow within an operational supervised ML traffic classifier

An optional next step is 'data sampling', designed to narrow down the search space for the ML algorithm when faced with extremely large training datasets (traffic traces). The sampling step extracts statistics from a subset of instances of various application classes, and passes these along to the classifier to be used in the training process.

As noted in section III-A7, a feature filtering/selection step is desirable to limit the number of features actually used to train the supervised ML classifier and thus create the classification model. The output of Figure 3 is a classification model.

Cross-validation (or stratified cross-validation) may be used to generate accuracy evaluation results during the training phase. However, if the source dataset consists of IP packets collected at the same time and the same network measurement point, the cross-validation results are likely to over-estimate the classifier's accuracy. (Ideally the source dataset would contain a mix of traffic collected at different times and measurement points, or use entirely independently collected training and testing datasets.)

Figure 4 illustrates data flow within an operational traffic classifier using the model built in Figure 3. Traffic captured in real-time is used to construct flow statistics from which features are determined and then fed into the classification model. (Here we presume that the set of features calculated from captured traffic is limited to the optimal feature set

determined during training.) The classifier's output indicates which flows are deemed to be members of the class of interest (as defined by the model). Certain implementations may optionally allow the model to be updated in real-time (performing a similar data sampling and training process to that shown in Figure 3). (For controlled testing and evaluation purposes offline traffic traces can be used instead of live traffic capture.)

2) Supervised versus unsupervised learning: As previously noted, IP traffic classification is usually about identifying traffic belonging to known applications (classes of interest) within previously unseen streams of IP packets. The key challenge is to determine the relationship(s) between classes of IP traffic (as differentiated by ML features) and the applications causing the IP traffic.

Supervised ML schemes require a training phase to cement the link between classes and applications. Training requires a-priori classification (or *labeling*) of the flows within the training datasets. For this reason, supervised ML may be attractive for the identification of a particular (or groups of) application(s) of interest. However, as noted in section III-A3, the supervised ML classifier works best when trained on examples of all the classes it expects to see in practice. Consequently, its performance may be degraded or skewed if not trained on a representative mix of traffic or the network link(s) being monitored start seeing traffic of previously unknown applications. (For example, Park et al. [44] showed that accuracy is sensitive to site-dependent training datasets, while Erman et al. [45] showed different accuracy results between the two data traces studied for the same ML algorithms.)

When evaluating supervised ML schemes in an operational context it is worthwhile considering how the classifier will be supplied with adequate supervised training examples, when it will be necessary to re-train, and how the user will detect a new type of applications.

It might appear that one advantage of unsupervised ML schemes is the automatic discovery of classes through the recognition of 'natural' patterns (clusters) in the dataset. However, resulting clusters still need to be labeled (for example, through direct inspection by a human expert) in order that new instances may be properly mapped to applications. (A related benefit is that traffic from previously unknown applications may be detected by noting when new clusters emerge - sometimes the emergence of new application flows is noteworthy even before the actual identity of the application has been determined.)

Another issue for unsupervised ML schemes is that clusters do not necessarily map 1:1 to applications. It would be ideal if the number of clusters formed is equal to the number of application classes to be identified, and each application dominated one cluster group. However, in practice, the number of clusters is often greater than the number of application classes [46] [47]. One application might spread over and dominate a number of clusters, or conversely an application might also spread over but not dominate any of the clusters. Mapping back from a cluster to a source application can become a great challenge.

When evaluating unsupervised ML schemes in an operational context it is worthwhile considering how clusters will be labeled (mapped to specific applications), how labels will be updated as new applications are detected, and the optimal number of clusters (balancing accuracy, cost of labeling and label lookup, and computational complexity).

C. Challenges for operational deployment

Section II-A noted some important IP traffic classification scenarios where classification must normally occur *as the traffic is flowing* (or within some fairly short period of time after the traffic occurred). This creates some particular requirements for timely classification as traffic is in transit across a network.

1) Timely and continuous classification: A *timely* classifier should reach its decision using as few packets as possible from each flow (rather than waiting until each flow completes before reaching a decision). Reducing the number of packets required for classification also reduces the memory required to buffer packets during feature calculations. This is an important consideration for situations where the classifier is calculating features for (tens of) thousands of concurrent flows. Depending on the business reason for performing classification, it may be unacceptable to sample the available flows in order to reduce the memory consumption. Instead, one aims to use less packets from each flow.

However, it is not sufficient to simply classify basing on the first few packets of a flow. For example, malicious attacks might disguise themselves with the statistical properties of a trusted application early in their flow's lifetime. Or the classifier itself might have been started (or restarted) whilst hundreds or thousands of flows were already active through a network monitoring point (thereby missing the starts of these active flows). Consequently the classifier should ideally perform *continuous* classification - recomputing its classification decision throughout the lifetime of every flow.

Timely and continuous ML classification must also address the fact that many applications change their statistical properties over time, yet a flow should ideally be correctly classified as being the same application throughout the flow's lifetime.

2) Directional neutrality: Application flows are often assumed to be bi-directional, and the application's statistical features are calculated separately in the forward and reverse directions. Many applications (such as multiplayer online games or streaming media) exhibit different (asymmetric) statistical properties in the client-to-server and server-to-client directions. Consequently, the classifier must either 'know' the direction of a previously unseen flow (for example, which end is the server and which is the client) or be trained to recognise an application of interest without relying on external indications of directionality.

Inferring the server and client ends of a flow is fraught with practical difficulties. As a real-world classifier should not presume that it has seen the first packet of every flow currently being evaluated, it cannot be sure whether the first packet it sees (of any new bi-directional flow of packets) is

heading in the 'forward' or 'reverse' direction. Furthermore, the semantics of the TCP or UDP port fields should be considered unreliable (either due to encryption obscuring the real value, or the application using unpredictable ports), so it becomes difficult to justify using 'well known' server-side port numbers to infer a flow's direction.

3) Efficient use of memory and processors: Another important criteria for operational deployment is the classification system's use of computational resources (such as CPU time and memory consumption). The classifier's efficiency impacts on the financial cost to build, purchase and operate large scale traffic classification systems. An inefficient classifier may be inappropriate for operational use regardless of how quickly it can be trained and how accurately it identifies flows.

Minimising CPU cycles and memory consumption is advantageous whether the classifier is expected to sit in the middle of an ISP network (where a small number of large, powerful devices may see hundreds of thousands of concurrent flows at multi-gigabit rates) or out toward the edges (where the traffic load is substantially smaller, but the CPU power and memory resources of individual devices are also diminished).

4) Portability and Robustness: A model may be considered *portable* if it can be used in a variety of network locations, and *robust* if it provides consistent accuracy in the face of network layer perturbations such as packet loss, traffic shaping, packet fragmentation, and jitter. A classifier also is robust if it can efficiently identify the emergence of new traffic applications.

IV. A Review of Machine Learning based IP Traffic Classification Techniques

In this section we create four broad categories to review significant works published on ML-based IP traffic classification to date in:

- *Clustering Approaches*: Works whose main approach centers around unsupervised learning techniques.
- *Supervised Learning Approaches*: Works whose main approach centers around supervised learning techniques.
- *Hybrid Approaches*: Works whose approach combine supervised and unsupervised learning techniques.
- *Comparisons and Related Work*: Works that compare and contrast different ML algorithms, or consider non-ML approaches that could be considered in conjunction with ML approaches.

The key points of each work are discussed in the following subsections and summarised in Table I, II and III.

A. Clustering Approaches

1) Flow clustering using Expectation Maximization: In 2004 McGregor et al. [48] published one of the earliest work that applied ML in IP traffic classification using the Expectation Maximization algorithm [49]. The approach clusters traffic with similar observable properties into different application types.

The work studies HTTP, FTP, SMTP, IMAP, NTP and DNS traffic. Packets in a 6-hour Auckland-VI trace are divided into bi-directional flows. Flow features (listed in Table I) are calculated on a full-flow basis. Flows are not timed out, except when they exceed the length of the traffic trace.

Based on these features, the EM algorithm is used to group the traffic flows into a small number of clusters and then create classification rules from the clusters. From these rules, features that do not have a large impact on the classification are identified and removed from the input to the learning machine and the process is repeated. The work's implementation of EM has an option to allow the number of clusters to be found automatically via cross-validation. The resulting estimation of performance was used to select the best competing model (hence the number of clusters).

The algorithm was found to separate traffic into a number of classes based on traffic type (bulk transfer, small transactions, multiple transactions etc.). However, current results are limited in identifying individual applications of interest. Nonetheless, it may be suitable to apply this approach as the first step of classification where the traffic is completely unknown, and possibly gives a hint on the group of applications that have similar traffic characteristics.

2) Automated application identification using AutoClass: The work of Zander et al. [46], proposed in 2005, uses AutoClass [50], which is an unsupervised Bayesian classifier, using the EM algorithm to determine the best clusters set from the training data. EM is guaranteed to converge to a local maximum. To find the global maximum, autoclass repeats EM searches starting from pseudo-random points in the parameter space. The model with the parameter set having the highest probability is considered the best.

Autoclass can be preconfigured with the number of classes (if known) or it can try to estimate the number of classes itself. Firstly packets are classified into bi-directional flows and flow characteristics are computed using NetMate [51]. A number of features are calculated for each flow, in each direction (listed in Table I). Feature values are calculated on a full-flow basis. A flow timeout of 60 seconds is used.

Sampling is used to select a subset of the flow data for the learning process. Once the classes (clusters) have been learnt, new flows are classified. The results of the learning and classification are exported for evaluation. The approach is evaluated based on random samples of flows obtained from three 24-hour traffic traces (Auckland-VI, NZIX-II and Leipzig-II traces from NLANR [52]).

Taking a further step from [48], the work proposed a method for cluster evaluation. A metric called intra-class homogeneity, H, for assessing the quality of the resulting classes and classification is introduced. H of a class is defined as the largest fraction of flows on one application in the class. The overall homogeneity H of a set of classes is the mean of the class homogeneities. The goal is to maximise H to achieve a good separation between different applications.

The results have shown that some separation between the different applications can be achieved, especially for certain particular applications (such as Half-Life online game traffic) in comparison with the others. With different sets of features used, the authors show that H increases with the increase

in number of features used. H reaches a maximum value of between 85% and 89%, depending on the trace. However, the work has not addressed the trade-offs between number of features used and their consequences of computational overhead.

To compute the accuracy for each application the authors map each class to the application that is dominating the class (by having the largest fraction of flows in that class). The authors used accuracy (Recall) as an evaluation metric. Median accuracy is $\geq$ 80% for all applications across all traces. However, there are some exceptional cases, for example, for the Napster application there is one trace where it is not dominating any of the classes (hence the accuracy is 0%). The results also show that FTP, Web and Telnet seem to have the most diverse traffic characteristics and are spread across many classes.

In general, although the mapping of class to application shows promising results in separating the different applications, the number of classes resulted from the clustering algorithm is high (approximately 50 classes for 8 selected applications). For class and application mapping, it is a challenge to identify applications that do not dominate any of the classes.

3) TCP-based application identification using Simple K-Means: In 2006 Bernaille et al. [53] proposed a technique using an unsupervised ML (Simple K-Means) algorithm that classified different types of TCP-based applications using the first few packets of the traffic flow.

In contrast to the previously published work, the method proposed in this paper allowed early detection of traffic flow by looking at only the first few packets of a TCP flow. The intuition behind the method is that the first few packets capture the application's negotiation phase, which is usually a pre-defined sequence of messages and is distinct among applications.

The training phase is performed offline. The input is a one-hour packet trace of TCP flows from a mix of applications. Flows are grouped into clusters based on the values of their first P packets. Flows are represented by points in a P-dimensional space, where each packet is associated with a dimension; the coordinate on dimension p is the size of packet p in the flow. Bi-directional flows are used. Packets sent by the TCP-server are distinguished from packets sent by the TCP-client by having a negative coordinate.

Similarity between flows is measured by the Euclidean distance between their associated spatial representations. After natural clusters are formed, the modeling step defines a rule to assign a new flow to a cluster. (The number of clusters was chosen by trial with different number of clusters for the K-means algorithm). The classification rule is simple: the Euclidean distance between the new flow and the centre of each pre-defined cluster is computed, and the new flow belongs to the cluster for which the distance is a minimum. The training set also consists of payload, so that flows in each cluster can be labeled with its source application. The learning output consists of two sets: one with the description of each cluster (the centre of the cluster) and the other with the composition of its applications. Both sets are used to classify flows online.

In the classification phase, packets are formed into a bi-directional flow. The sizes of the first P packets of the connection are captured and used to map the new flow to a spatial representation. After the cluster is defined, the flow is associated with the application that is the most prevalent in the cluster.

The results show that more than 80% of total flows are correctly identified for a number of applications by using the first five packets of each TCP flow. One exceptional case is the POP3 application. The classifier labels 86% of POP3 flows as NNTP and 12.6% as SMTP, because POP3 flows always belong to clusters where POP3 is not the dominant application.

The results of this work are inspiring for early detection of the traffic flow. However, it assumes that the classifier can always capture the start of each flow. The effectiveness of the approach when the classifier misses the first few packets of the traffic flow has not been discussed or addressed. Also, with the use of unsupervised algorithm and its classification technique, the proposal faces the challenge of classifying an application when it does not dominate any of the clusters found.

4) Identifying Web and P2P traffic in the network core: The work of Erman et al. [47] in early 2007 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. The work proposed to classify a flow using only uni-directional flow information. While showing that for a TCP connection, server-to-client direction might provide more useful statistics and better accuracy than the reverse direction, it may not always be feasible to obtain traffic in this direction. They also developed and evaluated an algorithm that could estimate missing statistics from a uni-directional packet trace.

The approach proposed makes use of clustering machine learning techniques with a demonstration of using the K-Means algorithm. Similar to other clustering approaches, Euclidean distance is used to measure the similarity between two flow vectors.

Uni-directional traffic flows are described by a full-flow based features set (listed in Table II). Possible traffic classes include Web, P2P, FTP... For the training phase, it is assumed that labels for all training flows are available (manually classified based on payload content and protocol signatures), and a cluster is mapped back to a traffic class that makes up the majority of flows in that cluster. An unseen flow will be mapped to the nearest cluster based on its distance to the clusters' centroids.

The approach is evaluated with flow accuracy and byte accuracy as performance metrics. Three datasets are considered: data sets containing only client-to-server packets, data sets containing only server-to-client packet, and data sets containing a random mixture of each direction. K-Means algorithm requires the number of clusters as an input, it has been shown that both flow and byte accuracies improved as k increased from 25 to 400. Overall, the server-to-client data sets

consistently give the best accuracy (95% and 79% in terms of flows and bytes respectively). With the random data sets, the average flow and byte accuracy is 91% and 67% respectively. For the client-to-server data sets, 94% of the flows and 57% of the bytes are correctly classified.

The algorithm to estimate the missing flow statistics is based on the syntax and semantics of the TCP protocol. So it only works with TCP, not other transport protocol traffic. The flow statistics are divided into three general categories: duration, number of bytes, and number of packets. The flow duration in the missing direction is estimated as the duration calculated with the first and the last packet seen in the observed direction. The number of bytes transmitted is estimated according to information contained in ACKs packets. The number of packets sent is estimated with the tracking of the last sequence number and acknowledgement number seen in the flow, with regards to the MSS. A number of assumptions have been made. For example, MSS is used as a common value of 1460 bytes, simple acknowledgment strategy of an ACK (40-byte data header with no payload) for every data packet, and assuming that no packet loss and retransmission occurred. An evaluation of the estimation algorithm is reported, the results were promising for flow duration and bytes estimation, with relatively larger error range for number of packets estimation.

The work addressed an interesting issue of the possibility of using uni-directional flow statistics for traffic classification and proposed a method to estimate the missing statistics. A related issue of directionality in the use of bi-directional traffic flows was addressed in the work of [54].

B. Supervised Learning Approaches

1) Statistical signature-based approach using NN, LDA and QDA algorithms: In 2004 Roughan et al. [18] proposed to use the nearest neighbours (NN), linear discriminate analysis (LDA) and Quadratic Discriminant Analysis (QDA) ML algorithms to map different network applications to predetermined QoS traffic classes.

The authors list a number of possible features, and classify them into five categories:

- Packet Level: e.g. packet length (mean and variance, root mean square)
- Flow Level: flow duration, data volume per flow, number of packets per flow (all with mean and variance values) etc. Uni-directional flow is used.
- Connection Level: e.g. advertised TCP window sizes, throughput distribution and the symmetry of the connection.
- Intra-flow/connection features: e.g. packet inter-arrival times between packets in flows.
- Multi-flow: e.g. multiple concurrent connections between the same set of end-systems.

Of the features considered, the pair of most value were the average packet length and flow duration. These features are computed per full-flow, then per aggregate of flows within 24-hour periods (an aggregate is a collection of statistics indexed by server port and server IP address).

Three cases of classification are considered. The three-class classification looks at three types of application: Bulk data (FTP-data), Interactive (Telnet), and Streaming (RealMedia); the four-class classification looks at four types of applications: Interactive (Telnet), Bulk data (FTP-data), Streaming (RealMedia) and Transactional (DNS); and the seven-class classification looks at seven applications: DNS, FTP-data, HTTPS, Kazaa, RealMedia, Telnet and WWW.

The classification process is evaluated using 10-times cross validation. The classification error rates are shown to vary based on the number of classes it tried to identify. The three-class classification has the lowest error rate, varying from 2.5% to 3.4% for different algorithms, while the four-class classification had the error rate in the range of 5.1% to 7.9%, and the seven-class one had the highest error rate of 9.4% to 12.6%.

2) Classification using Bayesian analysis techniques: In 2005 Moore and Zuev [14] proposed to apply the supervised ML Naive Bayes technique to categorise Internet traffic by application. Traffic flows in the dataset used are manually classified (based upon flow content) allowing accurate evaluation.

248 full-flow based features were used to train the classifier (a summary is listed in Table I). Selected traffic for Internet applications was grouped into different categories for classification, e.g. bulk data transfer, database, interactive, mail, services, www, p2p, attack, games and multimedia.

To evaluate the classifier's performance, the authors used Accuracy and Trust (equivalent to Recall) as evaluation metrics. The results showed that with the simple Naive Bayes technique, using the whole population of flow features, approximately 65% flow accuracy could be achieved in classification. Two refinements for the classifier were performed, with the use of Naive Bayes Kernel Estimation (NBKE) and Fast Correlation-Based Filter (FCBF) methods [2]. These refinements helped to reduce the feature space and improved the classifier performance to a flow accuracy better than 95% overall. With the best combination technique, the Trust value for individual class of application ranged, for instance, from 98% for www, to 90% for bulk data transfer, to approximately 44% for services traffic and 55% for P2P.

The work is extended with the application of Bayesian neural network approach in [55]. It has been demonstrated that accuracy is further improved compare to Naive Bayes technique. Bayesian trained neural network approach is able to classify flows with up to 99% accuracy for data trained and tested on the same day, and 95% accuracy for data trained and tested eight months apart. The paper also presents a list of features with their descriptions and ranking in their importance.

[2]The NBKE method is a generalisation of Naive Bayes. It addresses the problem of approximating every feature by a normal distribution. Instead of using a normal distribution with parameters estimated from the data, it uses kernel estimation methods. FCBF is a feature selection and redundancy reduction technique. In FCBF, goodness of a feature is measured by its correlation with the class and other good features. That feature becomes good if it is highly correlated with the class, yet is not correlated with any other good features [14]

3) Real-time traffic classification using Multiple Sub-Flows features: As noted in section III-C timely and continuous classification is an important constraint for the practical employment of a traffic classifier. In 2006 Nguyen and Armitage [56] proposed a method to address the issue by proposing classification based on only the most recent N packets of a flow - called a classification sliding window. The use of a small number of packets for classification ensures the timeliness of classification and reduces the buffer space required to store packets' information for the classification process. The approach does not require the classifier to capture the start of each traffic flow (as required in [53] and [57]). This approach allows classification to be initiated at any point in time when traffic flows are already in progress. It offers a potential of monitoring traffic flow during its lifetime in a timely manner with the constraints of physical resources.

The work proposes training ML classifiers on multiple sub-flows features. First, extract two or more sub-flows (of N packets) from every flow that represents the class of traffic one wishes to identify in the future. Each sub-flow should be taken from places in the original flow having noticeably different statistical properties (for example, the start and middle of the flow). Each sub-flow would result in a set of instances with feature values derived from its N packets. Then train the ML classifier with the combination of these sub-flows rather than the original full flows.

This optimisation is demonstrated using the Naive Bayes algorithm. Bi-directional flows were used. Different training and testing datasets were constructed from the two separate month-long traces collected during May and September 2005 at a public online game server in Australia and two 24-hour periods collected by the University of Twente, Netherland [58]. With the feature set used (listed in Table I), classifier built based on full-flow features is demonstrated to perform poorly when the classifier missed the start of a traffic flow. However, with the application of the proposed method, results show the classifier maintains more than 95% Recall and 98% Precision (flow accuracy) even when classification is initiated mid-way through a flow using only a total of 25 packets in both directions.

However, the work has only been demonstrated with an example of identifying an online game application (UDP-based First Person Shooter game - Enemy Territory [59]). Interference traffic included a range of other Internet applications (Web, DNS, NTP, SMTP, SSH, Telnet, P2P ...). The authors also suggested the potential benefits of using clustering algorithms in automating the sub-flows selection process.

4) Real-time traffic classification using Multiple Synthetic Sub-Flows Pairs: As noted in section III-C directional neutrality an important constraint for the practical employment of a traffic classifier. In 2006 Nguyen and Armitage [54] further extended their work in [56] to overcome this problem. The authors propose training the ML classifier using statistical features calculated over multiple short sub-flows extracted from full-flow generated by the target application *and* their mirror-imaged replicas as if the flow is in the reverse direction.

The optimisation is demonstrated when applied to the Naive Bayes and Decision Tree algorithms with an example of identifying a UDP-based First Person Shooter game - Enemy Territory [59] traffic. Using the same datasets as specified in [56] they demonstrate that both classifiers perform poorly when the classifiers are trained with bi-directional flow features that make assumptions about the forward and backward direction. However, training on Synthetic Sub-Flows Pairs results in significant improvement to classification performance (with up to 99% Recall and 98% Precision (flow accuracy) for the example application) even when classification is initiated mid-way through a flow, without prior knowledge of the flow's direction and using windows as small as 25 packets long.

5) GA-based classification techniques: In 2006 Park et al. [60] made use of feature selection technique based on Genetic Algorithm (GA). Using the same feature set specified in [44] (listed in II), three classifiers were tested and compared: the Naive Bayesian classifier with Kernel Estimation (NBKE), Decision Tree J48 and the Reduced Error Pruning Tree (REPTree) classifier. Their results suggest two decision tree classifiers provide more accurate classification results than the NBKE classifier. The work also suggests the impact of using training and testing data from different measurement points.

Early flow classification is also briefly mentioned. Accuracy as a function of the number of packets used for classification is presented for J48 and REPTree classifiers. The first 10 packets used for classification seems to provide the most accurate result. However, the accuracy result is provided as overall result. It is not clear how it would be different for different types of Internet applications.

6) Simple statistical protocol fingerprint method: Crotti et al. [61] in early 2007 proposed a flow classification mechanism based on three properties of the captured IP packets: packet length, inter-arrival time and packet arrival order. They defined a structure called *protocol fingerprints* which express the three traffic properties in a compact way and used an algorithm based on *normalised thresholds* for flow classification.

There are two phases in the classification process: training and classifying. In the training phase, pre-labeled flows from the application to be classified (the training dataset) are analysed to build the protocol fingerprints. A protocol fingerprint is a PDF vector, estimated from a set of flows of the same protocol from the training dataset. The PDF_i is built on all the i^{th} pairs of P_i ($P_i = \{s_i, \Delta t_i\}$) where s_i represents the size of packet i and Δt_i represents the inter-arrival time between packet i and packet (i-1). In order to classify an unknown traffic flow given a set of different PDFs, the authors check whether the behaviour of the flow is statistically compatible with the description given by at least one of the PDFs, and choose which PDF describes it better. An *anomaly score* that gives a value between 0 and 1 is used to indicate how 'statistically distant' an unknown flow is from a given protocol PDF. It shows the correlation between the unknown flow's i^{th} packet and the application layer protocol described by the specific PDF used; the higher the value, the higher the probability that the flow was generated by that protocol.

Their results show flow accuracy of more than 91% for classifying three applications: HTTP, SMTP and POP3, using the first few packets of each application's traffic flow.

In a similar way to the work of Bernaille et al. [53] reviewed above, this approach demonstrates advanced results for timeliness of the classification. However, it has the same limitation in assuming that the classifier can always capture the start of each flow, and is aware of the locations of client and server (for constructing the PDF of client-server and server-client directions). The effectiveness of the approach when the classifier misses the first few packets of the traffic flow (assumed to carry the protocol fingerprint), or suffers from packet loss and packet re-ordering has not been addressed.

C. Hybrid Approaches

Erman et al. in [62] in early 2007 proposed a *semi-supervised* traffic classification approach which combines unsupervised and supervised methods. Motivations to the proposal are due to two main reasons: Firstly labeled examples are scarce and difficult to obtain, while supervised learning methods do not generalise well when being trained with few examples in the dataset. Secondly, new applications may appear over time, and not all of them are known as a priori, traditional supervised methods map unseen flow instances into one of the known classes, without the ability to detect new types of flows [62].

To overcome the challenges, the proposed classification method consists of two steps. First, a training dataset consisting of labeled flows combined with unlabeled flows are fed into a clustering algorithm. Second, the available labeled flows are used to obtain a mapping from the clusters to the different known classes. This steps allows some clusters to be remained. To map a cluster with labeled flows back to an application type, a probabilistic assignment is used. The probability is estimated by the maximum likelihood estimate, $\frac{n_{jk}}{n_k}$ where n_{jk} is the number of flows that were assigned to cluster k with label j, and n_k is the total number of labeled flows that were assigned to cluster k. Clusters without any labeled flows assigned to them are labeled 'Unknown' as application type. Finally a new unseen flow will be assigned to the nearest cluster with the distance metric chosen in the clustering step.

This new proposed approach has promising results. Preliminary results have been shown in [62] with the employment of K-Means clustering algorithm. The classifier is provided with 64,000 unlabeled flows. Once the flows are clustered, a fixed number of random flows in each cluster are labeled. Results show that with two labeled flows per cluster and K = 400, the approach results in 94% flow accuracy. The increase in classification accuracy is marginal when five or more flows are labeled per cluster. More details results can be found in [63].

As claimed by the authors [63] the proposal has advantages in terms of faster training time with small number of labeled flows mixed with a large number of unlabeled flows, being able to handle previously unseen applications and the variation of existing application's characteristics, and the possibility of enhancing the classifier's performance by adding unlabeled flows for iterative classifier training. However, an evaluation of these advantages has not been demonstrated in the current paper.

D. Comparisons and Related Work

1) Comparison of different clustering algorithms: In 2006 Erman et al. [45] compared three unsupervised clustering algorithms: K-Means, DBSCAN and AutoClass. The comparison is performed on two empirical data traces: one public trace from the University of Auckland and one self-collected trace from the University of Calgary.

The effectiveness of each algorithm is evaluated using *overall accuracy* and the number of clusters it produces. *Overall accuracy* measurement determines how well the clustering algorithm is able to create clusters that contain only a single traffic category. A cluster is labeled by the traffic class that makes up the majority of its total connections (bi-directional traffic flows). Any connection that has not been assigned to a cluster is labeled as noise. Then overall accuracy is determined by the portion of the total TP for all clusters out of total number of connections to be classified. Like any other clustering algorithms, the number of clusters produced by a clustering algorithm is an important evaluation factor as it affects the performance of the algorithm in classification stage.

Their results show that the AutoClass algorithm produces the best overall accuracy. On average, AutoClass is 92.4% and 88.7% accurate in the Auckland and Calgary datasets respectively. It produces on average of 167 clusters for the Auckland dataset (for less than 10 groups of applications) and 247 clusters for the Calgary dataset (for 4 groups of applications). For K-Means, the number of clusters can be set, the overall accuracy steadily improves as the number of clusters (K) increases. When K is around 100, overall accuracy is 79% and 84% on average for the Auckland and Calgary datasets respectively. Accuracy is improved only slightly with greater value of K. DBSCAN algorithm produces lower overall accuracy (upto 75.6% for the Auckland and 72% for the Calgary data sets); however, it places the majority of the connections in a small subset of the clusters. Looking at the accuracy for particular traffic class categories, the DBSCAN algorithm has the highest precision value for P2P, POP3 and SMTP (lower than Autoclass for HTTP traffic).

The work mentions briefly about the comparison of model build time, and has not looked at other performance evaluation measurements, such as processing speed, CPU and memory usage, or the timeliness of classification.

2) Comparison of clustering vs. supervised techniques: Erman et al. [64] evaluate the effectiveness of supervised Naive Bayes and clustering AutoClass algorithm. Three accuracy metrics were used for evaluation: recall, precision and overall accuracy (overall accuracy is defined the same as [45] reviewed in the previous sections).

Classification method using the supervised Naive Bayes algorithm is straight forward. For classification using AutoClass, once AutoClass comes up with the most probable set

of clusters from the training data, the clustering is transformed into a classifier. A cluster is labeled with the most common traffic category of the flows in it. If two or more categories are tied, then a label is chosen randomly amongst the tied category labels. A new flow is then classified with the traffic class label of the cluster it is most similar to [64].

The evaluation was performed on two 72-hour data traces provided by the University of Auckland (NLANR). A connection is defined as a bi-directional flow. The feature set is shown in Table III.

The paper shows that with the dataset used and nine application classes (HTTP, SMTP, DNS, SOCKS, IRC, FTP control, FTP data, POP3 and LIMEWIRE), AutoClass has an average overall accuracy of 91.2% whereas the Naive Bayes classifier has an overal accuracy of 82.5%. AutoClass also performs better in terms of precision and recall for individual traffic classes. On average, for Naive Bayes, the precision and recall for six out of nine classes were above 80%; whereas for AutoClass, all classes have precision and recall values above 80%, six out of the nine classes have average precision values above 90%, and seven have average recall values above 90%. However, in terms of time taken to build classification model, AutoClass takes much longer time than Naive Bayes algorithm (2070 seconds vs. 0.06 seconds for the algorithm implementation, data and equipment used).

The conclusion that the unsupervised AutoClass outperforms the supervised Naive Bayes in terms of overall accuracy might be counterintuitive to some readers on the surface. While the testing methodology of the paper is sound, the results might be impacted by the size of the training data sets (the current work uses 1000 samples per application), the specific dataset used, how the Naive Bayes classifier is being trained (single application classification at a time, or multiple applications classification at a time), and the specific feature set used.

An issue of clustering approaches is the real-time classification speed, as the number of clusters resulted from the training phase is typically larger than the number of application classes. However, this has not been evaluated in the paper.

3) Comparison of different supervised ML algorithms: Williams et al. [65] provides insights into the performance aspect of ML traffic classification. The works look at a number of supervised ML algorithms: Naive Bayes with Discretisation (NBD), Naive Bayes with Kernel Density Estimation (NBK) , C4.5 Decision Tree, Bayesian Network, and Naive Bayes Tree. These algorithms' computational performance is evaluated in terms of classification speed (number of classifications per second) and the time taken to build the associated classification model.

Results are collected by experiments on three public NLANR traces. Features used for analysis include the full set of 22 features, and two best reduced feature sets selected by correlation-based feature selection (CFS) and consistency-based feature selection (CON) algorithms. The features set is shown in Table III.

The results show that most algorithms achieve high flow accuracy with the full set of 22 features (only NBK algorithm achieves $>$ 80% accuracy and the rest of the algorithms achieve greater than 95% accuracy). With the reduced sets of 8 (CFS) and 9 (CON) features, the results achieved by cross-validation show only slight changes in the overall accuracy compared to the use of full feature set. The largest reduction in accuracy were 2-2.5% for NBD and NBK with the use of CON reduced feature set.

Despite the similarity in classification accuracy, the paper shows significant differences in classification computational performance. C4.5 algorithm was seen as the fastest algorithm when using any of the feature set (with maximum of 54,700 classifications per second on a 3.4GHz Pentium 4 workstation running SUSE Linux 9.3 with WEKA implementation). Algorithms ranked in descending order in terms of classification speeds are: C4.5, NBD, Bayesian Network, Naive Bayes Tree, NBK.

In terms of the model build time, Naive Bayes Tree takes significant longer time than the remaining algorithms. Algorithms ranked in descending order in terms of model build time are: Naive Bayes Tree, C4.5, Bayesian Network, NBD, NBK.

Results of the paper also show feature reduction greatly improves performance of the algorithms in terms of model build time and classification speeds for most algorithms.

4) ACAS: Classification using machine learning techniques on application signatures: Haffner et al. [57] in 2005 proposed an approach for automated construction of application signatures using machine learning techniques. Different from the other works, this work makes use of the first n-Bytes of a data stream as features. Though it has the same limitation with those works that require accessing to packet payload, we include it in the survey as it is also machine learning based, and its interesting results may be useful in a composite machine learning based approach that combines different information such as statistical characteristics, contents, and communication patterns.

Three learning algorithms: Naive Bayes, AdaBoost and Maximum Entropy have been investigated in constructing application signatures for a various range of network applications: ftp control, smtp, pop3, imap, https, http and ssh. A flow instance is characterised with n-Bytes represented in binary value, and ordered by the position of the Byte in the flow stream. Collection of flow instances with binary feature is used as input by the machine learning algorithms.

Using of the first 64 bytes of each TCP unidirectional flow the overall error rate is below 0.51% for all applications considered. Adaboost and Maximum Entropy provide best results with more than 99% of all flows classified correctly. Precision is above 99% for all applications and Recall rate is above 94% for all application except ssh (86.6%) (The poor performance on ssh application was suspected due to the small amount of sample instances in the training dataset).

5) Unsupervised approach for protocol inference using flow content: Closely to Haffner et al. [57]'s work, in 2006, Ma et al. [66] introduced and analysed alternative mechanisms

for automatic identification of traffic, based solely on flow content. Unsupervised learning was applied in three different modeling techniques for capturing statistical and structural aspects of messages exchanged in a protocol, namely *product distribution*, *Markov processes*, and *common substring graphs (CSG)*.

Different from other work that made use of flow classification, the work focused on protocol inference, in which a *protocol* was defined as 'a pair of distributions on flows' - one was a byte sequence from the initiator to the responder and one was a byte sequence from the responder to the initiator (which does not include packet-level information such as inter-arrival time, frame size or header fields).

Product distribution model treats each n-byte flow distribution as a product of n independent byte distributions. Each byte offset in a flow is represented by its own byte distribution that describes the distribution of bytes at that offset in the flow. The Markov process is described as a random walk on a weighted directed graph. The nodes of the graphs are labeled with unique byte values. Each edge is weighted with a transition probability such that, for any node, the sum of all its out-edge weights is 1. The next node is chosen according to weight of the edge from the current node to its neighbors. And common substring graphs capture the common structural information about the flows from which it is built.

Detailed model descriptions, how to construct each model, how to merge, compare and classify new instances are described in [66]. Overall, the product distribution resulted in the lowest total misclassification error (1.68%-4.15%), while Markov processes had the highest (3.33-9.97%) and CSGs in the middle (2.08-6.19%).

6) BLINC: Multilevel traffic classification in the dark: Karagiannis et al. [15] developed an application classification method based on the behaviours of the source host at the transport layer, divided into three different levels. The social level captures and analyses the interactions of the examined host with other hosts, in terms of the numbers of them it communicates with. The host's popularity and that of other hosts in its community's circle are considered. The role of the host, in acting as a provider or the consumer of a service, is classified at the functional level. Finally, transport layer information is used, such as the 4-tuple of the traffic (source and destination IP addresses, and source and destination ports), flow characteristics such as the transport protocol, and the average packet size.

A range of application types was studied in this work, including web, p2p, data transfer, network management traffic, mail, chat, media streaming, and gaming. By analysing the social activities of the host, the authors conclude that among the host's communities, neighbouring IPs may offer the same service (a server farm) if they use the same service port, exact communities might indicate attacks, while partial communities may signify p2p or gaming applications. In addition, most IPs acting as clients have a minimum number of destination IPs. Thus, focusing on the identification of that small number of servers can help client identification, leading to the classification of a large amount of traffic. Classification at the functional level shows that a host is likely to be providing a service if during a duration of time it uses a small number of source ports, normally less than or equal to two for all of their flows. Typical client behaviour is normally represented when the number of source ports is equal to the number of distinct flows. The consistency of average packet size per flow across all flows at the application level is suggested to be a good property for identifying certain applications, such as gaming and malware.

Completeness and accuracy are the two metrics used for the classification approach. Completeness is defined as the ratio of the number of flows (bytes) classified by BLINC over the total number of flows (bytes), indicated by payload analysis. The results show that BLINC can classify 80% to 90% traffic flows with more than 95% flow accuracy (70% to 90% for byte accuracy).

This method has to gather information from several flows for each host before it can decide on the role of one host. Such requirements might prevent the employment of this method in real-time operational networks.

7) Pearson's Chi-Square test and Naive Bayes classifier: Bonfiglio et al. [67] recently proposed a framework based on two techniques to identify Skype traffic in realtime. The first, based on Pearsons Chi-Square test, detects Skypes fingerprint through analysis of the message content randomness introduced by the encryption process. The second, based on the Naive Bayes theorem, detects Skypes traffic from message size and arrival rate characteristics.

Using two specific test datasets, the authors' compared the performance of each technique relative to classification using deep-packet inspection. They showed their Naive Bayes technique to be effective in identifying voice traffic over IP regardless of source application. Their Pearsons Chi-Square test effectively identified Skype traffic (including Skype voice/video/chat/data traffic) over UDP and all encrypted or compressed traffic for TCP flows. When used in combination the two techniques detected Skype voice traffic (UDP flows) with 0% false positives and 9.82% false negatives for one test dataset, and 0.11% false positives and 2.40% false negatives for the other. These false positives rates are an improvement compared to each technique being used individually. However, the false negatives rates are slightly worse. Also it is important to note the great imbalance between the amount of Skype traffic compared to other traffic in the test datasets. The results should also be evaluated in terms of precision and recall, to reflect the classifiers' performance per traffic class, instead of only the overall false positives and false negatives.

Both techniques offer real-time traffic classification. The Chi-Square technique looks at the first few bytes of the message. The Naive Bayes technique looks at the statistical characteristics for each window of 30 consecutive packets.

E. Challenges for operational deployment

We wrap up our survey with a qualitative look at the extent to which the reviewed works overlap Section III-C's additional

constraints and requirements for using ML techniques inside real-time IP traffic classifiers.

1) Timely and continuous classification: Most of the reviewed work has evaluated the efficacy of different ML algorithms when applied to entire datasets of IP traffic, trained and tested over full-flows consisting of thousands of packets (such as [14] [18] [46] [48] [64] and [65]).

Some ([53] and [61]) have explored the performance of ML classifiers that utilise only the first few packets of a flow, but they cannot cope with missing the flow's initial packets. Others ([56]) have explored techniques for continuous classification of flows using a small sliding window across time, without needing to see the initial packets of a flow.

2) Directional neutrality: The assumption that application flows are bi-directional, and the application's direction may be inferred prior to classification, permeates many of the works published to date ([14] [48] [53] [46] [68]). Most work has assumed that they will see the first packet of each bi-directional flow, that this initial packet is from a client to a server. The classification model is trained using this assumption, and subsequent evaluations have presumed the ML classifier can calculate features with the correct sense of forward and reverse direction.

As getting the direction wrong will degrade classification accuracy, [54] explores the creation of classifier models that do not rely on external indications of directionality.

3) Efficient use of memory and processors: There are definite trade-offs to be made between the classification performance of a classifier and the resource consumption of the actual implementation. For example, [14] and [55] reveal excellent potential for classification accuracy. However, they use a large number of features, many of which are computationally challenging. The overhead of computing complex features (such as effective bandwidth based upon entropy, or Fourier Transform of the packet inter-arrival time) must be considered against the potential loss of accuracy if one simply did without those features.

Williams et al. [65] provide some pertinent warnings about the trade-off between training time and classification speed. (For example, among five ML algorithms studied, Naive Bayes with Kernel Estimation took the shortest time to build classification models, yet performed slowest in terms of classification speed.)

Techniques for timely and continuous classification have tended to suggest a sliding window over which features are calculated. Increasing the length of this window ([56] [54] and [57]) might increase classification accuracy. However, depending on the particular implementation (opportunities for pipelining, step size with which the window slides across the incoming packet streams, etc.) this may decrease the timeliness with which classification decisions are made (and increase the memory required to buffer packets during feature calculations). Most of the reviewed work has not, to date, closely investigated this aspect.

4) Portability and Robustness: None of the reviewed works seriously considered or addressed the issue of classification model portability mentioned in section III-C.

None of the reviewed works has addressed and evaluate their model's robustness in terms of classification performance with the introduction of packet loss, packet fragmentation, delay and jitter. Unsupervised approaches have the potential to detect the emergence of new types of traffic. However, this issue has not been evaluated in most of the works. It was briefly mentioned in [62].

5) Qualitative summary: Table IV provides a qualitative summary of the reviewed works against the following criteria:

- Real-time Classification
  - *No:* The work makes use of features that require flow completion to compute (e.g. Flow duration, total flow bytes count)
  - *Yes:* The work requires the capture of a small number of packets/bytes of a flow to do classification
- Feature Computation Overhead

  Low: The work makes use of a small number of features (e.g sizes of the first few packets, binary encoded of the first few bytes of a unidirectional flow)
  - *Average:* The work makes use of an average set of features (such as packet length and inter-arrival times statistics, flow duration, bytes count)
  - *High:* The work makes use of a large (comparatively with other work in the area) including computational complex features (such as Fourier transform of packet inter-arrival time)
- Continuous Classification
  - *Not addressed:* The issue was not considered in the work
  - *Yes:* The issue was considered and solved in the work
- Directional Neutrality
  - *No:* The work makes use of bi-directinal flow and features calculations, but did not consider the issue
  - *Yes:* The work makes use of bi-directional flow and feature calculations, addressed the issues and proposed solution
  - *N/A:* The work makes use of uni-directional flow and the issue is not applicable
  - *Not clear*: Not clearly stated in the paper

V. Conclusion

This paper surveys significant works in the field of machine learning based IP traffic classification during the peak period of 2004 to early 2007. Motivated by a desire to move away from port-based or payload-based traffic classification, it is clear that ML can be applied well in the task. The use of a number of different ML algorithms for offline analysis, such as AutoClass, Expectation Maximisation, Decision Tree, NaiveBayes etc. has demonstrated high accuracy (up to 99%) for a various range of Internet applications traffic. Early ML techniques relied on static, offline analysis of previously captured traffic. More recent work is beginning to address the requirements for practical, ML-based real-time IP traffic

TABLE I
A SUMMARY OF RESEARCH REVIEWED IN SECTION IV

Work	ML Algorithms	Features	Data Traces	Traffic Considered	Classification Level
McGregor et al. [48]	Expectation Maximization	• Packet length statistics (min, max, quartiles, ...) • Inter-arrival statistics • Byte counts • Connection duration • Number of transitions between transaction mode and bulk transfer mode • Idle time Calculated on full flows	NLANR and Waikato trace	A mixture of HTTP, SMTP, FTP (control), NTP, IMAP, DNS ...	Coarse grained (bulk transfer, small transactions, multiple transactions ...)
Zander et al. [46]	AutoClass	• Packet length statistics (mean and variance in forward and backward directions) • Inter-arrival time statistics (mean and variance in forward and backward directions) • Flow size (bytes) • Flow duration Calculated on full-flows	Auckland-VI, NZIX-II and Leipzig-II from NLANR	Half-Life, Napster, AOL, HTTP, DNS, SMTP, Telnet, FTP (data)	Fine grained (8 applications studied)
Roughan et al. [18]	Nearest Neighbour, Linear Discriminate Analysis and Quadratic Discriminant Analysis	• Packet Level • Flow Level • Connection Level • Intra-flow/Connection features • Muli-flow features Calculated on full flows	Waikato trace and section logs from a commercial streaming services	Telnet, FTP (data), Kazaa, Real Media Streaming, DNS, HTTPS	Fine grained (three, four and seven classes of individual applications)
Moore and Zuev [14]	Baysian Techniques (Naive Bayes and Naive Bayes with Kernel Estimation and Fast Correlation-Based Filter method)	Total of 248 features, among them are • Flow duration • TCP port • Packet inter-arrival time statistics • Payload size statistics • Effective bandwidth based upon entropy • Fourier transform of packet inter-arrival time Calculated on full flows	Proprietary Hand Classified Traces	A large range of Database, P2P, Buck, Mail, Services, ... traffic	Coarse grained
Barnaille et al. [53]	Simple K-Means	Packet lengths of the first few packets of bi-directional traffic flows	Proprietary traces	eDonkey, FTP, HTTP, Kazaa, NTP, POP3, SMTP, SSH, HTTPS, POP3S	Fine grained (10 applications studied)
Park et al. [44] [44]	Naive Bayes with Kernel Estimation, Decision Tree J48 and Reduced Error Prunning Tree	• Flow duration • Initial Advertised Window bytes • Number of actual data packets • Number of packets with the option of PUSH • Packet lengths • Advertised window bytes • Packet inter-arrival time • Size of total burst packets	NLANR, USC/ISI, CAIDA	WWW, Telnet, Chat (Messenger), FTP, P2P (Kazaa, Gnutella), Multimedia, SMTP, POP, IMAP, NDS, Oracle, X11	N/A (comparison work)
Nguyen and Armitage [56]	Supervised Naive Bayes	• Packet lengths (min, max, mean, standard deviation) • Inter-Packet lengths statistics (min, max, mean, standard deviation) • Packet Inter-arrival times statistics (min, max, mean, std dev.) • Calculated over a small number (e.g. 25 packets) of consecutive packets (classification windows) taken at various points of the flow lifetime - where the changes in flow's characteristics are significant	Traces collected at an online game server in Australia and provided by University of Twente, Netherland	Online Game (Enemy Territory) traffic, Others (HTTP, HTTPS, DNS, NTP, SMTP, Telnet, SSH, P2P ...)	Application specific (Online Game, UDP based, First Person Shooter, Enemy Territory traffic)

TABLE II
A SUMMARY OF RESEARCH REVIEWED IN SECTION IV(CONTINUED)

Work	ML Algorithms	Features	Data Traces	Traffic Considered	Classification Level
Nguyen and Armitage [54]	Naive Bayes and Decision Tree in combination with Clustering algorithms for automated sub-flows selection	• Packet lengths statistics (min, max, mean, std dev.) • Inter-Packet lengths statistics (min, max, mean, std dev.) • Packet Inter-arrival times statistics (min, max, mean, std dev.) • Calculated over a small number (e.g. 25 packets) of consecutive packets (classification windows) taken at various points of the flow lifetime - where the changes in flow's characteristics are significant. • Further extension with synthetic mirroring features.	Traces collected at an online game server in Australia and provided by University of Twente, Netherland	Online Game (Enemy Territory) traffic, Others (HTTP, HTTPS, DNS, NTP, SMTP, Telnet, SSH, P2P ...)	Application specific (Online Game, UDP based, First Person Shooter, Enemy Territory traffic)
Erman et al. [47]	K-Means	• Total number of packets • Mean packet length • mean payload length excluding headers • Number of bytes transferred • Flow duration • Mean inter-arrival time	Self-collected 8 1-hour campus traces between April 6-9, 2006	Web, P2P, FTP, Others	Coarse grained (29 different protocols grouped into a number of application categories for studies)
Crotti et al. [61]	Protocol fingerprints (Probability Density Function vectors) and Anomaly score (from protocol PDFs to protocol fingerprints)	• Packet lengths • Inter-arrival time • Packet arrival order	6-month self-collected traces at the edge gateway of the University of Brescia data centre network	TCP applications (HTTP, SMTP, POP3, SSH)	Fine grained (four TCP protocols)
Haffner et al. [57]	Naive Bayes, AdaBoost, Regularized Maximum Entropy	Discrete byte encoding of the first n-bytes payload of a TCP unidirectional flow	Proprietary	FTP (control), SMTP, POP3, IMAP, HTTPS, HTTP, SSH	Fine grained
Ma et al. [66]	Unsupervised learning (*product distribution*, *Markov processes*, and *common substring graphs*)	Discrete byte encoding of the first n-bytes payload of a TCP unidirectional flow	Proprietary	FTP (control), SMTP, POP3, IMAP, HTTPS, HTTP, SSH	Fine grained
Auld et al. [55]	Bayesian Neural Network	246 features in total, including: • Flow metrics (duration, packet-count, total bytes) • Packet inter-arrival time statistics • Size of TCP/IP control fields • Total packets in each direction and total for bi-directional flow • Payload size • Effective bandwidth based upon entropy • Top-ten Fourier transform components of packet inter-arrival times for each direction • Numerous TCP-specific values derived from tcptrace (e.g. total payload bytes transmitted, total number of PUSHED packets, total number of ACK packets carrying SACK information etc.)	Proprietary hand classified traces	A large range of Database, P2P, Buck, Mail, Services, Multimedia, Web ... traffic	Coarse grained

TABLE III
A SUMMARY OF RESEARCH REVIEWED IN SECTION IV(CONTINUED)

Work	ML Algorithms	Features	Data Traces	Traffic Considered	Classification Level
Williams et al. [65]	Naive Bayes with Discretisation, Naive Bayes with Kernel Estimation, C4.5 Decision Tree, Bayesian Network and Naive Bayes Tree	• Protocol • Flow duration • Flow volume in bytes and packets • Packet length (minimum, mean, maximum and standard deviation) • Inter-arrival time between packets (minimum, mean, maximum and standard deviation)	NLANR	FTP(data), Telnet, SMTP, DNS, HTTP	N/A (Comparison work)
Erman et al. [45]	K-Means, DBSCAN and AutoClass	• Total number of packets • Mean packet length • Mean payload length excluding headers • Number of bytes transfered (in each direction and combined) • Mean packet inter-arrival time	NLANR and a self-collected 1-hour trace from the University of Calgary	HTTP, P2P, SMTP, IMAP, POP3, MSSQL, Other	N/A (Comparison work)
Erman et al. [64]	Naive Bayes and AutoClass	• Total number of packets • Mean packet length (in each direction and combined) • Flow duration • Mean data packet length • Mean packet inter-arrival time	NLANR	HTTP, SMTP, DNS, SOCKS, FTP(control), FTP (data), POP3, Limewire	N/A (Comparison work)
Bonfiglio et al. [67]	Naive Bayes and Pearson's Chi-Squate test	• Message size (the length of the message encapsulated into the transport layer protocol segment) • Average inter packet gap	Two self collected datasets	Skype traffic	Application specific

TABLE IV
REVIEWED WORK IN LIGHT OF CONSIDERATIONS FOR OPERATIONAL TRAFFIC CLASSIFICATION

Work	Real-time Classification	Feature Computation Overhead	Classify Flows In Progress	Directional neutrality
McGregor et al. [48]	No	Average	Not addressed	No
Zander et al. [46]	No	Average	Not addressed	No
Roughan et al. [18]	No	Average	Not addressed	N/A
Moore and Zuev [14]	No	High	Not addressed	No
Barnaille et al. [53]	Yes	Low	Not addressed	No
Park et al. [44]	No	Average	Not addressed	Not clear
Nguyen and Armitage [56]	Yes	Average	Yes	Yes
Nguyen and Armitage [54]	Yes	Average	Yes	Yes
Erman et al. [47]	No	Average	Not addressed	No
Crotti et al. [61]	Yes	Average	Not addressed	No
Haffner et al. [57]	Yes	Average	Not addressed	N/A
Ma et al. [66]	No	Average	Not addressed	No
Auld et al. [55]	No	High	Not addressed	No
Williams et al. [65]	N/A	Average	N/A	N/A
Erman et al. [45]	N/A	Average	N/A	N/A
Erman et al. [64]	N/A	Average	N/A	N/A
Bonfiglio et al. [67]	Yes	Average	Not addressed	Not clear

classification in operational networks. In this survey paper, we have outlined a number of critical operational requirements for real-time classifiers and qualitatively critiqued the reviewed works against these requirements.

There is still a lot of room for further research in the field. While most of the approaches build their classification models based on sample data collected at certain points of the Internet, those models' usability needs to be carefully evaluated. The accuracy evaluated on the test dataset collected at the same point of measurement might not be true when being applied in different point of measurement. There are still open questions as to how well they can maintain their performance in the presence of packet loss, latency jitter, and packet fragmentation. Each ML algorithm may perform differently toward different Internet applications, and may require different parameter configurations. The use of a combination of classification models is worth investigating. Parallel processing for real-time classification at traffic aggregation points in the network may be useful when the classifiers need to cope with millions of concurrent flows simultaneously. And the application of ML algorithms for newer applications (such as Skype, video streaming, voice over IP and peer to peer file-sharing) is still an interesting open field.

Nevertheless, the promising results of ML-based IP traffic classification may open many new avenues for related research areas, such as the application of ML in intrusion detections, anomaly detection in user data and control, routing traffic, and building network profiles for proactive network real-time monitoring and management. The classification of traffic in greynet or darknet networks is also an interesting possible extension of the research field.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their very helpful comments and feedbacks to improve the manuscript.

REFERENCES

[1] Snort - The de facto standard for intrusion detection/prevention, http://www.snort.org, as of August 14, 2007.

[2] Bro intrusion detection system - Bro overview, http://bro-ids.org, as of August 14, 2007.

[3] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, no. 31(23-24), pp. 2435–2463, 1999.

[4] L. Stewart, G. Armitage, P. Branch, and S. Zander, "An architecture for automated network control of QoS over consumer broadband links," in *IEEE International Region 10 Conference (TENCON 05)*, Melbourne, Australia, November 2005.

[5] F. Baker, B. Foster, and C. Sharp, "Cisco architecture for lawful intercept in IP networks," Internet Engineering Task Force, RFC 3924, 2004.

[6] T. Karagiannis, A. Broido, N. Brownlee, and K. Claffy, "Is P2P dying or just hiding?" in *Proceedings of the 47th annual IEEE Global Telecommunications Conference (Globecom 2004)*, Dallas, Texas, USA, November/December 2004.

[7] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in network identification of P2P traffic using application signatures," in *WWW2004*, New York, NY, USA, May 2004.

[8] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," RFC 1633, IETF, 1994.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, IETF, 1998.

[10] G. Armitage, *Quality of Service In IP Networks: Foundations for a Multi-Service Internet*. Macmillan Technical Publishing, April 2000.

[11] L. Burgstahler, K. Dolzer, C. Hauser, J. Jahnert, S. Junghans, C. Macian, and W. Payer, "Beyond technology: the missing pieces for QoS success," in *RIPQoS '03: Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM) workshop on Revisiting IP QoS*. New York, NY, USA: ACM Press, August, 2003, pp. 121–130.

[12] Ubicom Inc., *Solving Performance Problems with Interactive Applications in a Broadband Environment using StreamEngine Technology*, http://www.ubicom.com/pdfs/whitepapers/StreamEngine-WP-20041031.pdf, as of August 14, 2007.

[13] J. But, N. Williams, S. Zander, L. Stewart, and G. Armitage, "ANGEL - Automated Network Games Enhancement Layer," in *Proceedings of 5th Annual Workshop on Network and Systems Support for Games (Netgames) 2006*, Singapore, October 2006.

[14] A. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) 2005*, Banff, Alberta, Canada, June 2005.

[15] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: Multilevel traffic classification in the dark," in *Proc. of the Special Interest Group on Data Communication conference (SIGCOMM) 2005*, Philadelphia, PA, USA, August 2005.

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

[17] *Internet Assigned Numbers Authority (IANA)*, http://www.iana.org/assignments/port-numbers, as of August 14, 2007.

[18] 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 ACM/SIGCOMM Internet Measurement Conference (IMC) 2004*, Taormina, Sicily, Italy, October 2004.

[19] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, IETF, 1996.

[20] A. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proc. Passive and Active Measurement Workshop (PAM2005)*, Boston, MA, USA, March/April 2005.

[21] A. Madhukar and C. Williamson, "A longitudinal study of P2P traffic classification," in *14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, September 2006.

[22] V. Paxson, "Empirically derived analytic models of wide-area TCP connections," *IEEE/ACM Transactions on Networking*, vol. 2, no. 4, pp. 316–336, 1994.

[23] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of Internet chat systems," in *ACM/SIGCOMM Internet Measurement Conference 2003*, Miami, Florida, USA, October 2003.

[24] K. Claffy, "Internet traffic characterisation," PhD Thesis, University of California, San Diego, 1994.

[25] T. Lang, G. Armitage, P. Branch, and H.-Y. Choo, "A synthetic traffic model for Half-life," in *Proceedings of Australian Telecommunications Networks and Applications Conference 2003 ATNAC2003*, Melbourne, Australia, December 2003.

[26] T. Lang, P. Branch, and G. Armitage, "A synthetic traffic model for Quake 3," in *Proceedings of ACM SIGCHI International Conference on Advances in computer entertainment technology (ACE2004)*, Singapore, June 2004.

[27] Z. Shi, *Principles of Machine Learning*. International Academic Publishers, 1992.

[28] H. Simon, "Why should machines learn?" in *R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (editors) Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann, 1983.

[29] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations (Second Edition)*. Morgan Kaufmann Publishers, 2005.

[30] B. Silver, "Netman: A learning network traffic controller," in *Proceedings of the Third International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Association for Computing Machinery, 1990.

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

[32] Y. Reich and J. S. Fenves, "The formation and use of abstract concepts in design," in *Fisher, D. H. and Pazzani, M. J. (editors), Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, 1991.

[33] H. D. Fisher, J. M. Pazzani, and P. Langley, *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, 1991.

[34] R. Duda, P. Hart, and D. Stork, *Pattern Classification (2nd edition)*. JWiley-Interscience, 2001.

[35] O. Carmichael and M. Hebert, "Shape-based recognition of wiry objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 12, pp. 1537– 1552, 2004.

[36] W. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[37] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Cluster validity methods: part I," *SIGMOD Rec.*, vol. 31, no. 2, pp. 40–45, 2002.

[38] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, no. Vol.16, Issue 3, pp. 645– 678, May 2005.

[39] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering validity checking methods: part II," *SIGMOD Rec.*, vol. 31, no. 3, pp. 19–27, 2002.

[40] M. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437– 1447, 2003.

[41] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[42] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligent*, vol. 97, no. 1-2, pp. 273–324, 1997.

[43] P. H. Winston, *Artificial intelligence (2nd ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1984.

[44] J. Park, H.-R. Tyan, and K. C.-C.J., "Internet traffic classification for scalable QoS provision," in *IEEE International Conference on Multimedia and Expo*, Toronto, Ontario, Canada, July 2006.

[45] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*. New York, NY, USA: ACM Press, 2006, pp. 281–286.

[46] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *IEEE 30th Conference on Local Computer Networks (LCN 2005)*, Sydney, Australia, November 2005.

[47] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*. Banff, Alberta, Canada: ACM Press, May 2007, pp. 883–892.

[48] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *Proc. Passive and Active Measurement Workshop (PAM2004)*, Antibes Juan-les-Pins, France, April 2004.

[49] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of Royal Statistical Society*, vol. 30, no. 1, 1997.

[50] P. Cheeseman and J. Stutz, "Bayesian classification (AutoClass): Theory and results," in *Advances in Knowledge Discovery and Data Mining*, 1996.

[51] NetMate, http://sourceforge.net/projects/netmate-meter/, as of August 14, 2007.

[52] The National Laboratory for Applied Network Research (NLANR), *Traffic Measurement Data Repository*, http://pma.nlanr.net/Special/, as of August 14, 2007.

[53] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM Special Interest Group on Data Communication (SIGCOMM) Computer Communication Review*, vol. 36, no. 2, 2006.

[54] T. Nguyen and G. Armitage, "Synthetic sub-flow pairs for timely and stable IP traffic identification," in *Proc. Australian Telecommunication Networks and Application Conference*, Melbourne, Australia, December 2006.

[55] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for Internet traffic classification," *IEEE Transactions on Neural Networks*, no. 1, pp. 223–239, January 2007.

[56] 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. IEEE 31st Conference on Local Computer Networks*, Tampa, Florida, USA, November 2006.

[57] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: automated construction of application signatures," in *MineNet '05: Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data*. Philadelphia, Pennsylvania, USA: ACM Press, August 2005, pp. 197–202.

[58] The University of Twente, *Traffic Measurement Data Repository*, http://arch.cs.utwente.nl/projects/m2c/m2c-D15.pdf, as of 17th August 2007.

[59] *Wolfenstein Enemy Territory*, http://www.enemyterritory.com/, as of 17th August 2007.

[60] J. Park, H.-R. Tyan, and K. C.-C.J., "GA-Based Internet Traffic Classification Technique for QoS Provisioning," in *Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Pasadena, California, December 2006.

[61] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 5–16, 2007.

[62] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Semi-supervised network traffic classification," *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) Performance Evaluation Review*, vol. 35, no. 1, pp. 369–370, 2007.

[63] ——, "Offline/realtime network traffic classificatioin using semi-supervised learning," Department of Computer Science, University of Calgary, Tech. Rep., February 2007.

[64] J. Erman, A. Mahanti, and M. Arlitt, "Internet traffic identification using machine learning techniques," in *Proc. of 49th IEEE Global Telecommunications Conference (GLOBECOM 2006)*, San Francisco, USA, December 2006.

[65] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *Special Interest Group on Data Communication (SIGCOMM) Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.

[66] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. Voelker, "Unexpected means of protocol inference," in *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*. Rio de Janeriro, Brazil: ACM Press, October 2006, pp. 313–326.

[67] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing Skype traffic: when randomness plays with you," in *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, August 2007, pp. 37–48.

[68] N. Williams, S. Zander, and G. Armitage, "Evaluating machine learning methods for online game traffic identification," Centre for Advanced Internet Architectures, http://caia.swin.edu.au/reports/060410C/CAIA-TR-060410C.pdf, Tech. Rep. 060410C, as of August 14, 2007.