Lightweight Application Classification for Network Management

Hongbo Jiang
EECS Department
Case Western Reserve Univ.
Cleveland, OH 44106
hongbo.jiang@case.edu

Andrew W. Moore*
Computer Laboratory
University of Cambridge
andrew.moore@cl.cam.ac.uk

Zihui Ge
Adverplex Inc.
100 Quannapowitt Parkway
Wakefield, MA 01880
gezihui@adverplex.com

Shudong Jin
EECS Department
Case Western Reserve Univ.
Cleveland, OH 44106
shudong.jin@case.edu

Jia Wang
AT&T Labs Research
180 Park Avenue
Florham Park, NJ 07932
jiawang@research.att.com

ABSTRACT

Traffic application classification is an essential step in the network management process to provide high availability of network services. However, network management has seen limited use of traffic classification because of the significant overheads of existing techniques. In this context we explore the feasibility and performance of lightweight traffic classification based on NetFlow records. In our experiments, the NetFlow records are created from packet-trace data and pre-tagged based upon packet content. This provides us with NetFlow records that are tagged with a high accuracy for ground-truth. Our experiments show that NetFlow records can be usefully employed for application classification. We demonstrate that our machine learning technique is able to provide an identification accuracy ($\approx 91\%$) that, while a little lower than that based upon previous packet-based machine learning work ($> 95\%$), is significantly higher than the commonly used port-based approach ($50 - 70\%$). Trade-offs such as the complexity of feature selection and packet sampling are also studied. We conclude that a lightweight mechanism of classification can provide application information with a considerably high accuracy, and can be a useful practice towards more effective network management.

Categories and Subject Descriptors

C.2.3 [**Computer Communications Networks**]: Network Operations Network Monitoring; C.2.m [**Computer Communication Networks**]: Miscellaneous

General Terms

Algorithms, Measurement

Keywords

Flow classification, Internet traffic, Traffic identification

1. INTRODUCTION

The continual growth and evolution of applications, hosts, and networks have been the hallmark of the Internet — we assert that this hallmark will be typical of the next generation Internet. Such a high level of continuous development has been a challenge for network management. On the other hand, network monitoring and measurement have been widely used in understanding how well the network performs and ensuring certain level of QoS specified in the Service Level Agreements (SLAs) with customers. In this paper, we focus on one of the essential network measurement primitives – network traffic classification, which provides inputs to a wide range of important network management tasks including network planning, traffic engineering, accounting and billing, and anomaly detection and mitigation. Despite a wide range of work on this in the past, for example [4, 3, 13, 5, 8, 10], the classification of network traffic continues to be a challenge. In particular, network traffic classification is a complicated multi-factor system involving the mutual interaction of a range of networks, hosts, applications, and protocols. Furthermore, there are harsh requirements on system performance and robustness before this class of methodology can be implemented and deployed for practical applications.

The contribution of this paper is a lightweight application classification scheme. In particular, we apply supervised machine learning techniques to the problem of network traffic classification in terms of identifying causal applications of traffic flows. The ready availability and wide use of NetFlow[1] data motivate our work. The NetFlow [2] system, embodied in many commercial routing products, is a source of data in common use by operators. Such data

*This work was supported by EPSRC Grants GR/S93714/02 and GR/T10510/02.

INM'07, August 27–31, 2007, Kyoto, Japan.

[1]We acknowledge that NetFlow may refer specifically to the Cisco implementation of packet-trace statistics; equivalent approaches include the NetStream approach of Huawei and cflowd of Juniper. Throughout this paper we will refer to all such flow-level information, generally, as NetFlow.

have been used for accounting and billing, and more recently has seen increased use for anomaly detection and capacity planning. NetFlow systems can be adjusted to be low in overhead compared to packet level monitoring. Furthermore, there is negligible extra operational burden on using NetFlow data for new applications. A number of the larger ISPs regularly collect many terabytes of NetFlow data each year and archive them for long-term planning purposes allowing the prospect of network *archaeology* to further understand the changing application mix.

In this paper, we adopt the taxonomy that originated with the *Class of Service* (CoS) mapping-work of [11], and was further extended in [7]. The nature of each Internet application allows the wide variety of services to be divided into several discrete categories, for example, web-browsing, multimedia, telephony (VoIP), email activities, peer-to-peer and FTP file transfers, and malicious traffic (e.g., worms). Our approach provides best operation using the full features of a NetFlow record, including ports and IP addresses. While acknowledging the site-specific nature of this approach, we see such a system as fulfilling an important position among identification methodologies. One application has been the identification and tracking the behavior of hosts to better recognize anomalies . Another example is in the monitoring of SLAs between two parties in an outsourcing arrangement. In such an arrangement it is in both parties' interests to ensure that the behavior of plant stays within the agreed operating terms. Our techniques are a low-overhead mechanism to allow for system roles to be identified prior to such an agreement, and then allow issues such as a changing mix of applications to be identified at an early stage.

The remainder of this paper is organized as follows. Alongside the theoretical underpinnings of our classification methodology, the mechanisms of NetFlow and background details on our evaluation datasets are given in Section 2. Section 3 details both our baseline evaluation and improvement gains. Alongside a summary of pertinent conclusions, Section 4 provides directions for future work.

2. NETFLOW DATA AND FLOW-LEVEL TRAFFIC CLASSIFICATION

The classification of network traffic has been performed previously using features derived from streams of packets. Such feature-collections are often huge (200+ features) and range in complexity from Fourier-Transformations and quartile statistics to mean and variance of packet inter-arrival time and the number of TCP SACK packets. In such cases all the packets must be observed for the statistic to be usefully computed, however the classification accuracy has been shown to be very good (e.g., >95%) [8, 14]. The disadvantage of this approach has been the complexity and costs associated with the collection, aggregation, and generation of the desired packet-derived features, a significant factor impeding the wide deployment of the proposed techniques. In contrast, extensive vendor support has meant flow-level statistics, such as Cisco NetFlow, are widely available. Our motivation is to test if NetFlow information is sufficient to accurately classify the traffic. Hence, we first need to examine the capability of the flow-level statistics.

Examining Cisco NetFlow as an example, NetFlow [2] may be one of the most widely used traffic monitoring tools in the Internet today. NetFlow generates flow statistics of traffic flowing through a router. In NetFlow v5, the most common version, a *flow* is defined as a unidirectional sequence of packets between a particular pair of source and destination IP addresses. For each flow, NetFlow maintains a record in the router memory containing a number of fields including the source and destination IP addresses, source and destination port numbers, inbound and outbound network interfaces, transport protocol, IP type of service, flow starting and finishing timestamps, cumulative TCP flags, and number of bytes and number of packets transmitted. Later NetFlow versions contain additional fields such as MPLS labels and flexible template for flow definition in NetFlow v9. Upon timeout of a flow record cache, a UDP/IP packet is generated and forwarded to a NetFlow collector, which processes and stores the NetFlow records. As obtaining and maintaining NetFlow data may also be computationally expensive for routers, especially in high-speed networks, packet sampling, either deterministic or random, can be applied. In the rest of the paper, we will rely on the flow statistical information provided by NetFlow v5 for traffic classification.

2.1 Naïve Bayes Classification and Kernel Estimation

For the purpose of classification, machine learning technique is applied to the NetFlow flow record. The flow record may thus be considered as a series of data samples $\mathbf{x} = \{x_1, \ldots, x_n\}$, which is a realization of $\mathbf{X} = \{X_1, \ldots, X_n\}$. Each random variable X_i is described by m features $(A_1, \ldots, A_m)$ that can take numeric or discrete values. Chosen from the features are a set of discriminators (also referred to as attributes) that are utilized for classification. Assume that there are k known classes of interest, represented by $\mathcal{C} = \{c_1, \ldots, c_k\}$. Each class is characterized by a distribution function, $f(\cdot|c_j)$, which describes the membership characteristics of class c_j over the defined features. Given an unobserved flow y, the posterior probability that y belongs to class c_j can be calculated, according to Bayes rule, as follows:

$$p(c_j|y) = \frac{p(c_j)f(y|c_j)}{\sum\limits_{c_j} p(c_j)f(y|c_j)},$$

where $p(c_j)$ denotes the prior distribution of being in class c_j.

The goal of supervised traffic classification is to estimate $f(\cdot|c_j)$, given some training set $\mathbf{x}$. By assuming the form of multidimensional Gaussian mixtures (with inherent independence assumption across features), naïve Bayes kernel approach estimates $f(\cdot|c_j)$ by fitting over the training data. The estimation of the real density $f(\cdot|c_j)$ is hence given by

$$\hat{f}(t|c_j) = \frac{1}{n_{c_j} h} \sum_{x_i \in c_j} K(\frac{t - x_i}{h}),$$

where h is called the kernel bandwidth and $K(t)$ is a kernel distribution, e.g., standard Gaussian density $K(t) = \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)$, $-\infty < t > \infty$ for numeric values, and uniform density $K(t) = \frac{1}{2}\mathbf{1}$, $-1 \leq t \leq 1$ for discrete values. Similar to [8], we set the kernel bandwidth h as in the default value of the WEKA software suite[12], which we use to perform calculations in this work.

Despite a greatly reduced number of features in the NetFlow records (in contrast to packet-derived features), it remains important to eliminate irrelevant, as well as redundant, features for the set of features that individually correlate well with the class but have little inter-correlation. We use the symmetric uncertainty measure to rank the relevance of the features:

$$U(A_i, C) = 2\frac{H(A_i) + H(C) - H(A_i, C)}{H(A_i) + H(C)},$$

where $H(\cdot)$ is the entropy function, and C is the flow class (considered as an feature). Note that symmetric uncertainty always lies between 0 and 1. Furthermore, we use the correlation-based measure

to determine the goodness of any given subset, S, of the features:

$$\sum_{A_j \in S} U(A_j, C) \Big/ \sqrt{\sum_{A_i \in S} \sum_{A_j \in S} U(A_i, A_j)}$$

By adding features sequentially in the decreasing order of symmetric uncertainty and monitoring the amount of increase in the goodness measure, we select the desired set of features.

2.2 Evaluation Dataset and NetFlow Generation

Throughout this study we have used data collected by the high-performance network monitor described in [6]. We use its loss-limited, full-payload capture to disk providing time-stamps with resolution of better than 35 nanoseconds. The site we examined hosts several Biology-related facilities, and collectively employs about 1,000 researchers, administrators and technical staff. This site is connected to the Internet via a full-duplex Gigabit Ethernet link. It was on this connection to the Internet that our monitor was placed. Traffic was monitored for each traffic-set consists of a full 24-hour, weekday period and for both link directions. The data-details are further provided in [7].

The manner of sampling consists of taking several fixed length periods from randomly placed, non-overlapping periods throughout a 24-hour day. Section 5.1 of [8] provides a comprehensive description of the sampling mechanism.

This collection of traffic data was hand-classified into a number of application classes including web browsing (WWW), Email, Bulk, Attack, P2P, Database, Multimedia, Service, Interaction, and Games. Overall, our experimental dataset contains over 48.8 million IP packets that constitute about 0.8 million TCP flows and 3.5 million UDP/ICMP flows in total. While we focus on the accuracy of classifying TCP flows, all IP data was used as input to our work as we wish for the NetFlow flow-cache behavior to be as realistic as possible.

To obtain NetFlow records from the packet traces collected, we utilize nProbe, a software implementation of Cisco NetFlow. The nProbe, providing a flexible PC-based implementation of NetFlow, allows the use of trace data (e.g., pcap packet-trace files), as input, thereby permitting repeatability of experiments. In this way we are able to feed our packet-traces to nProbe and collect flow records with different settings of packet sampling rate. We set the number of cache entries to 65,536, which corresponds to the default value in Cisco 7200 series and 7500 series routers, and set the cache timeout interval (for both active and inactive flows) to be 60 seconds, which corresponds to the configuration in a Tier-1 ISP network that we surveyed. We presume no loss between nProbe and the NetFlow collector.

Note that not all fields in the nProbe output are meaningful, for example, the inbound and outbound interfaces. We also exclude the transport protocol field as we restrict our study to TCP flows (97%, by volume of the total dataset). Table 1 summarizes the substantial features in the raw flow records. In addition, we also explore a set of derivative features and describe them in the same table.

Baseline features	Explanations
srcIP/dstIP	source/destination IP address
srcPort/dstPort	source/destination port address
tos	IP type of service
sTime/eTime	flow start/end timestamp
tcpFlag	cumulative OR of TCP flags
bytes	total number of bytes observed
pkts	number of packets observed
Derivative features	**Explanations**
length	duration of flow (eTime−sTime)
pktSize	average packet size (bytes/pkts)
byteRate	average flow rate (bytes/length)
pktRate	average packet rate (pkts/length)
tcpFxxx	xxx=syn/ack/fin/rst/psh/urg flag

Table 1: Baseline features and derivative features in the flow records.

3. EVALUATION AND IMPROVEMENT

We apply the described Bayes and kernel estimation technique to the NetFlow records for application identification, and demonstrate the feasibility of this approach. We attempt to answer a number of questions in this section: (1) does this technique accurately identify the applications? (2) is there room to further improve its accuracy? (3) can we bound the computational complexity without affecting its accuracy? (4) is it robust if the flow records are obtained at low sampling rate? The purposes of this evaluation are to demonstrate that the proposed classification techniques are effective, and to investigate the tradeoffs between the accuracy of classification and the complexity/overhead of NetFlow system.

3.1 Baseline Experiments

In our baseline experiment, we take the straightforward approach of exploring only the baseline features in the flow records. Unless otherwise noted, the dataset is split randomly into two subsets of equal size, with one of them being used as the training data and the other to test the accuracy of the classification. As the objective is to identify the type of applications for each flow, the accuracy is measured by the percentage of flows correctly classified. The result is very encouraging: we find the accuracy of classification is about 88.3%. This accuracy is already significantly higher than that of industry standard techniques which use the TCP port number to achieve only an accuracy of 50-70% [8, 7], although it is lower than the accuracy of classification techniques based upon packet-derived features (>95% as reported in [3, 8]). We expect the reason for this lower accuracy is due to that there is not enough information in a tuple alone to differentiate some application types; for example, Voice-over-IP traffic using port 80 may be confused with web traffic.

To understand which features are more relevant, we calculate the symmetric uncertainty values of all features, and show them in Figure 1 in descending order; the *derivative* features in Table 1 are also shown in this figure but will be explained in the next section. As expected, among others, the port numbers, the type-of-service, and the cumulative TCP flag all appear highly relevant. Interestingly, the IP addresses also show high relevance. This is due to that the IP addresses often reflect role-specific information on the placement of the services, e.g., one machine is a mail server, and another is a web server.

3.2 Generic Derivative Features

Alongside the baseline features, Figure 1 also shows high symmetric uncertainty values for several derivative features: tcpFack, pktSize, and length as described in Table 1. As we expect it is possible to improve the performance by using derivative features in classification, the result of our experiment is disappointing: we obtain only an improved accuracy value of 89.06%. We should emphasize that the improvement is insignificant and possibly is an artifact from this particular dataset. We conjecture that the deriva-

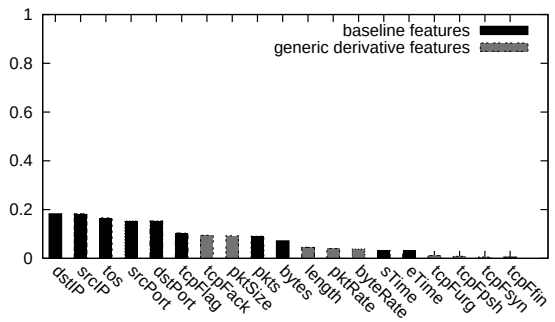

Figure 1: Symmetric uncertainty of the features.

tive features are derived using only simple (albeit non-linear) math operators, and so they may not carry significantly more information than the baseline features. For instance, the derivative pktSize is derived from the number of bytes and the number of packets.

A specific example of the improvement in accuracy provided by derivative features is the addition of tcpFack. This is a single bit representing the presence/absence of the TCP ACK flag on data packets within a flow during a cache timeout interval. This flag allows a differentiation of the Email and Bulk traffic classes from the web browser traffic class, as the former two classes tend to have little full-duplex activity, hence no ACK, in the client-to-server direction.

3.3 Application-aware Derivative Features

While our baseline experiments reveal the effectiveness of the classification technique using the described features, it is possible to derive more meaningful features by understanding how different applications lead to different characteristics in NetFlow records. This subsection gives an example: the TCP port number.[2] Prior work on application identification, for example [9, 1], has frequently used the TCP port number to obtain a reasonable accuracy at 50-70%. In the IANA port allocation it is a common practice for the ports numbered 1–1024 to refer to specific privileged services and protocols. For instance, web servers usually use port 80 as the listening port to accept client connection requests, while a client program may take any random port number which must be large (e.g., larger than 1024). Hence, the smaller port number of a flow, more likely used by a server, is considered more informative in traffic classification. We call this feature the "low port" and the larger port number of the flow the "high port".

The relevance of these two features is best illustrated in Figure 2, which highlights their symmetrical uncertainty values among those from Figure 1. The low port becomes a predominant feature. Its symmetrical uncertainty, and thus its entropy, is at least four times larger than those of the other features. This result is consistent with prior traffic classification efforts that have relied upon this feature. Meanwhile, the symmetrical uncertainty of the high port is only 0.04, and thus may contribute much less to improving the accuracy of classification. Nevertheless, the specific inclusion of these two features in the feature set leads to only a marginally improved accuracy of 91.4%, or intepreted in another way, approximately a 20% reduction in the number of misclassifications. We conjecture it is because the accuracy is already relatively high with the kernel estimation technique, leaving little room for improvement (recall that classification using the more comprehensive feature-set derived from streams of packets yields only a 95% accuracy).

[2]We believe this general approach (of exploring knowledge of the applications) can also be applied to derive other meaningful features to further improve accuracy.

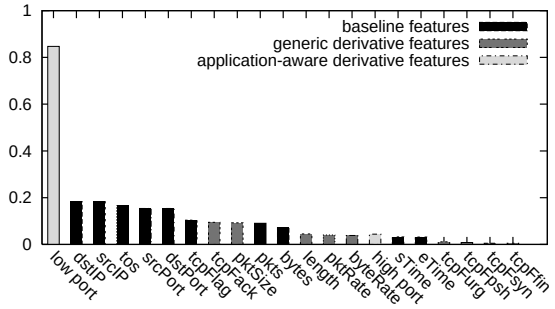

Figure 2: Symmetric uncertainty of all features, including "low port" and "high port".

3.4 Reducing Feature Complexity

The computational complexity of many traffic classification techniques, in terms of both sampling overhead and classification algorithm complexity, can be high due to the large number of features and the large size of the dataset. In many applications, reducing this complexity is desirable if the impact on the accuracy of classification is negligible, although for offline classification tasks, it is not a terrible tradeoff to tolerate a longer processing time for a marginally improved accuracy. In the next two subsections, we consider approaching this problem from both directions: reducing the dimensionality (i.e., number of features), and reducing the dataset size (i.e., number of flows).

The inclusion of the derivative features may suggest that many features become redundant. For example, the starting time and the ending time of a flow may no longer be important once we have the derivative "flow length". The cumulative TCP flags may not be useful either since we have variables to represent each individual flag. In addition, some features, such as the source and destination IP addresses, may be particularly site-dependent, which one may opt to omit in the feature set. While we may exclude these features to reduce time complexity, we hope the accuracy of classification is minimally affected. We thus compare the following feature sets: (1) baseline features, (2) all features, (3) all features excluding srcIP/dstIP, (4) all features excluding srcIP/dstIP, sTime, eTime and tcpFlag, and (5) a *compact* set of features determined by the feature selection algorithm described in Section 2.1. Figure 3 shows the accuracy of the classification result. We observe that the accuracy remains reasonably high by removing irrelevant and redundant features, while the running time of the kernel estimation algorithm is greatly reduced—from about 30 hours using the all features feature set (2) to less than 30 minutes using the compact feature set (5) with three features ("low port", "tos", and "tcpFlag") under the same hardware/software settings.

3.5 Reducing the Training-Set Size

The use of a small training set can be critical in an environment where the (manual) classification of a large quorum of data is impractical. Figure 4 illustrates the relationship between the size of the training set and the accuracy of the classification approach. The accuracy value is the average from 10 experiments with different seeds. Most notably, despite varying the training set through three orders of magnitude, the accuracy of the classification process varies by less than 8%. The results illustrated here allow us to be confident in the promotion of this method as a mechanism for

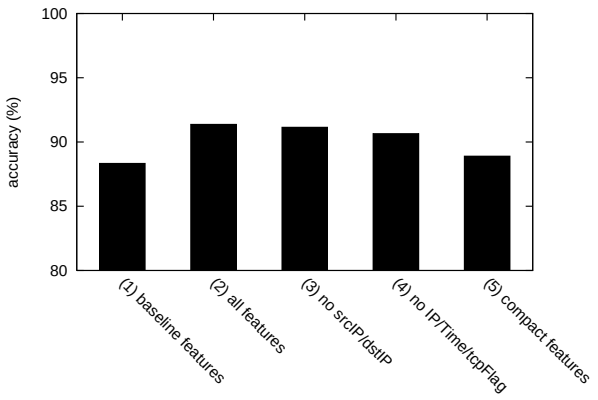

Figure 3: Accuracy of classification with reduced features.

the tracking of host behavior using a relatively small set of training data.

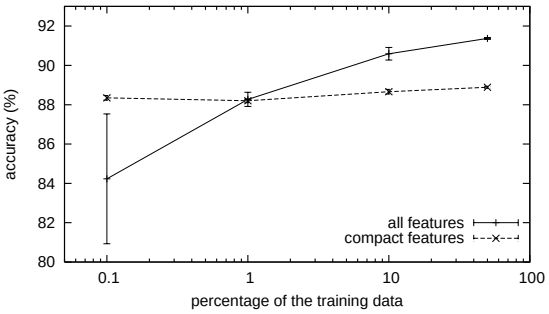

Figure 4: Training set size versus accuracy.

At first glance Figure 4 may appear to be counter-intuitive, however the increasing accuracy of the results is proportional to a combination of the size of the training-set and the number of features. This may be identified as caused by the (relatively) small number of flows that are not (clearly/uniquely) identified by the features of the compact-feature set. The inclusion of a number of additional features along with the inclusion of a number of samples to allow both differentiation of these hitherto unclearly defined flows and a minimization of the mis-classification of other flows leads to an increasing accuracy once the training set contains a sufficient number of samples. The low accuracy of the wide-range of features when the training data does not provide sufficient coverage is caused by the inverse effect: the large error bar results from the sample of data simply not providing a sufficiently representative sample when the classifier is exposed to all features - this in-turn leads to the mis-classification and the level of (in)accuracy noted.

3.6 Impact of Packet Sampling

Packet sampling is a common practice in operational use of NetFlow. Sampling mechanisms are often implemented in hardware to reduce the impact upon both router and NetFlow infrastructure. In contrast to previous sections where no packet sampling occurs, we configure nProbe to extract flow records based upon packet sampling.

While packet sampling, e.g., even at a light sampling rate of 1% or 10%, may dramatically reduce the overhead at routers, especially in high-speed networks, it is not clear whether such sampling will have a negative impact on the accuracy of our flow classification techniques. To that end, we conduct the following experiment. First, we show in Figure 5 the number of collected flows varying with the packet sampling rate. Overall, packet sampling does not result in a significantly smaller number of obtained flows. For instance, at the 0.1% sampling rate, nProbe still outputs 20% of the maximum number of flows obtained when no sampling is used.

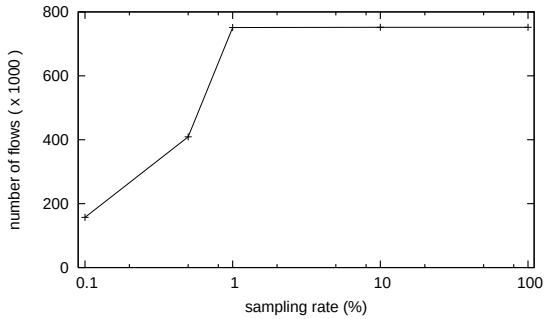

Figure 5: Number of flows observed by varying packet sampling rate.

We then test our classification technique using the flow records collected at different sampling rates. Figure 6 shows the accuracy varying with the sampling rate for two feature sets: all-features set and the compact feature set. The results show that the use of packet sampling does not affect the kernel estimation technique much. An interesting observation from Figure 6 is that when sampling rate is very low, e.g., 0.1%, the accuracy of classification is slightly higher. This, however, is because of an artifact of the packet sampling — when sample rate is very low, a more homogeneous set of large flows can survive the sampling, which perceivably leads to more accurate classification. This also suggests that one should be cautious in using flow characteristics derived by NetFlow data with packet-sampling enabled, as the result may be biased.

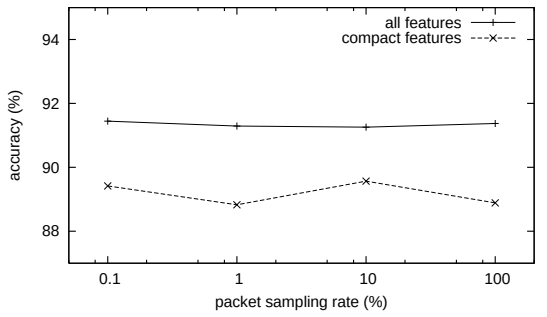

Figure 6: Accuracy of two classification methods when the packet sampling rate varies.

4. CONCLUSION & FUTURE WORK

Internet network management has developed in parallel with an Internet that has seen continual growth in the variety and number of applications, networks, and hosts. In response to such continual development, network management must provide a wide range of services, such as performance guarantees, network planning, and long-term traffic modeling, alongside day-to-day activities of anomaly detection and mitigation. Motivated by this wide range of needs we introduce a technique for lightweight application classification. Our approach to application classification is based upon

flow-level statistics based directly upon standard NetFlow records. Alongside allowing a low-overhead, widely applicable identification mechanism — NetFlow records are commonly available to network operators — we show that significant improvement may be made through the introduction of derivative features computed in a straightforward way from standard NetFlow values. Further, we have illustrated that our approach functions across a range of NetFlow sampling rates.

While acknowledging the site-specific nature of the classifier, we see such a use as fulfilling a unique position among identification methodologies: for example, site-specific bias is ideal if operators wish to identify anomalies in the behavior of server systems. The precise identification (and low-overhead monitoring) of system roles could see a NetFlow-based method applied to the monitoring of SLAs for outsourcing agreements.

We present our work not as a complete solution in this field but as a low-overhead and effective methodology that has a number of valuable uses. We believe we will require adaptive learning and classification schemes like this to enable and maintain solutions in fields such as traffic engineering, anomaly detection, and dynamic application-centric resource control in the next generation Internet.

Future Work

Noting that the use of additional features has proved valuable, a future item would be to explore the use of templates and other enhanced features and extensions to the commonly used NetFlow v5 implementation (see Section 2). Moreover, we acknowledge the use of naïve machine learning methods and foresee that methods such as a combination of C4.5 decision-trees and boosting would provide a contrasting approach: one based only upon the existence of samples and not weighted by evidence-based probabilities. Additionally, further work could usefully investigate more sophisticated application models. Models that encapsulate site-independence would clearly be advantageous but even models that encapsulate varying time-dynamics would be useful. An example application of this might be an application-specific classifier as anomaly detector. Such a model would permit the identification of behavior, normal for one time of the day, but peculiar at another period.

Another important piece of further work is the application and testing of our methods to clarify properties of spatial and temporal stability. With information about both the decay properties of any classifier and of the effectiveness of a classification scheme across multiple locations, it would become clear precisely what was the trade-off between training complexity and wide-ranging application.

Acknowledgments

The authors thank Wei Li and Awais Awan for providing technical input and Ralphe Neill for reading over an early draft of this work.

5. REFERENCES

[1] C. Logg and L. Cottrell. Characterization of the traffic between SLAC and the Internet, July 2003. http://www.slac.stanford.edu/comp/net/slac-netflow/html/SLAC-netflow.html.

[2] Cisco IOS. Netflow white papers, 2006. http://www.cisco.com/en/US/products/ps6601/prod_white_papers_list.html.

[3] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: multilevel traffic classification in the dark. In *Proceedings of ACM SIGCOMM 2005*, pages 229–240, 2005.

[4] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of ACM SIGCOMM 2004*, pages 219–230, Sept. 2004.

[5] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Proceedings of ACM SIGCOMM 2005*, pages 217–228, 2005.

[6] A. Moore, J. Hall, C. Kreibich, E. Harris, and I. Pratt. Architecture of a network monitor. In *Proceedings of Passive & Active Measurement Workshop 2003 (PAM2003)*, La Jolla, CA, Apr. 2003.

[7] A. W. Moore and D. Papagiannaki. Toward the accurate identification of network applications. In *Proceedings of the Sixth Passive and Active Measurement Workshop (PAM 2005)*, volume 3431. Springer-Verlag LNCS, Mar. 2005.

[8] A. W. Moore and D. Zuev. Internet traffic classification using Bayesian analysis techniques. In *Proceedings of ACM Sigmetrics*, pages 50–60, 2005.

[9] D. Moore, K. Keys, R. Koga, E. Lagache, and K. C. Claffy. CoralReef software suite as a tool for system and network administrators. In *Proceedings of the LISA 2001 15th Systems Administration Conference*, Dec. 2001.

[10] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23–24):2435–2463, 1999.

[11] 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 *ACM SIGCOMM Internet Measurement Conference*, Taormina, Sicily, Italy, 2004.

[12] I. H. Witten and E. Frank. *Data Mining*. Morgan Kaufmann Publishers, 2000.

[13] K. Xu, Z.-L. Zhang, and S. Bhattacharyya. Profiling Internet backbone traffic: behavior models and applications. In *Proceedings of ACM SIGCOMM 2005*, pages 169–180, 2005.

[14] S. Zander, T. Nguyen, and G. Armitage. Self-learning IP traffic classification based on statistical flow characteristics. In *Proceedings of the Sixth Passive and Active Measurement Workshop (PAM 2005)*, Apr. 2005.