Intelligent IP Traffic / Flow Classification System

Isara Anantavrasilp

Technische Universität München – Institut für Informatik I1
anantavr@in.tum.de

Abstract: In QoS-aware networks, such as DiffServ-enabled IP networks, UMTS, or IEEE 802.11e, the QoS-aware applications that run over them can identify service classes to their flows. The flows are then treated by the networks differently with respect to their classes. In contrast, legacy applications are not aware of the concept of QoS and do not specify any classes to their flows. Thus they cannot benefit from the QoS-support provided by the networks.

To this end, this paper brings forth a new intelligent flow classification system (FCS) that can automatically identify the service classes of legacy flows. The proposed FCS employs a new flow-level characteristics or "features" allowing it to identify flows on-the-fly and no packet-level data are required. Equipped with a machine learning technique, it is also adaptive and self-updatable. Moreover, the FCS is evaluated using packet traces from a sizeable network. The results show that our FCS works remarkably well with average accuracy of 99.66%.

1 Introduction

Recent network standards, such as Internet Protocol version 6 (IPv6), 802.11e Wireless LAN, WiMAX, and UMTS, incorporate QoS management schemes into their specifications. The implemented schemes are based on "Differentiated Services (DiffServ)" architecture, under which an application in a network node can specify an appropriate service class to its network connection or "flow". The packets within the flows are then marked with a "service class" indicating the flow's type of service so that the network can treat each of them appropriately.

Regardless of which QoS management scheme is employed, effective QoS management within a network requires certain applications to specify the QoS requirement to their flows. We call such applications "QoS-aware applications". In contrast, the commonly used applications at present are designed based on the best-effort scheme. These so-called "legacy applications" are not aware of the concept of QoS and do not specify any classes to their flows — hence they cannot receive the QoS-support provided by the network. Therefore, a mechanism that can correctly assign the service classes to the flows is essential. The class assignment process is called "flow classification" and the mechanism that carries out the classification is called a "flow classification system (FCS)".

In our previous works [AS07b, AS07a], a novel FCS that aimed to support QoS management is proposed. It uses features that concern only flow-level information that can

be observed in the network layer, no packet-level data including packet payloads are required. It also uses a machine learning algorithm, which provides learning capability to the FCS. As a result, our FCS can "learn" the characteristics of the flows from the known applications and use the learned knowledge to classify flows from the unknown ones.

Nevertheless, the proposed FCS is aimed at the end-user's legacy applications and is designed to be used only in the user's end-device. Therefore, the only a handful of flows can be computed simultaneously and only a small number of network applications are concerned. In this paper, we extend our previous works by investigating feasibility of deploying the system in a large-scale network that contains a sizeable number of hosts and applications. The results of our evaluations show that the system can effectively be used in a large networks with average correctness of 99.66%. We also evaluate our method in term of learning time. The evaluation result shows that different learning algorithms, although providing similar prediction correctness, they vary greatly in term of computational time.

This paper is organised as follows. Section 2 discusses existing flow classification systems and related works. Section 3 examines each component of a FCS as well as our proposed service classes, features and learners. Section 4 describes our evaluation methodology, benchmark test set, as well as reporting the experiment results. Finally, Section 5 concludes the paper.

2 Related Works

Flow classification is a method to categorise a given flow to an appropriate class. It is originated in the area of network security where attacks or unauthorised flows must be detected [ZP00, ML05]. The network administrators also use the FCSs to distinguish different types of flows for network management and provision [SSW04, MZ05, BTS06]. Recently, along with the need of QoS management, the flow classification has also been employed to identify service classes of the legacy flows [RSSD04, AS07b, AS07a].

Traditionally, well-known Internet application protocols have specific transport port numbers registered to themselves by the Internet Assigned Numbers Authority (IANA). The simplest approach to distinguish different kinds of the flows is thus to look at their transport ports. This method, however, is proven to be unreliable as applications might not communicate via registered ports or some application protocols are not registered to IANA [RSSD04].

Another approach, called "signature-based" approach, searches through payloads of packets in a flow for specific contents or "signatures" that can be used to identify the application whose the flow belongs to. This method is sometimes referred to as "deep packet inspection" [ZP00, SSW04, MP05]. Although this approach usually yields very high classification accuracy, the signatures must be identified beforehand. In addition, searching for signatures in packet payloads might raise privacy issues and might not work if the data in the payload are encrypted [BTS06].

To attack those problems, the so-called "flow-behaviour-based" FCSs try to capture common characteristics or behaviours of the flows in the same classes without relying on spe-

cific signatures in the packet payloads. Some of these FCSs are equipped with classifiers that have been pre-programmed by an expert [ZP00, ML05]. Some are equipped with machine learning algorithms allowing them to learn common characteristics of the flows without human-intervention [EBR03, RSSD04, BTA+06, EMA+07]. Other approaches such as [XZB05] take a different route and focus on classifying the hosts instead. After the type of the host (e.g., a server, a client, or an attacker) is determined, all flows from that host will be classified as the same class. While this approach is useful in a network provision, it is rather ineffective for flow-level classification.

Our proposed method is different from the aforementioned approaches in that it does not require huge dataset at classification time [XZB05, RSSD04]. Also, unlike all signature-based approaches, no human-intervention is necessary. More importantly, as our approach does not require flows to be bidirectional, it can classify both TCP and UDP flows, which is not the case in [EBR03, BTA+06, EMA+07].

3 Flow Classification System

In this section, the components of a flow classification system will be discussed. We will also see how our proposed system is unique and more generic than other works.

3.1 Flows and Their Characteristics

In our framework, a flow is defined as a sequence of packets with the same 5-tuple value. After the flow is observed by the FCS, its features are extracted. In our implementation, all features of a flow is stored together as a vector, called "feature vector". After a flow is abstracted into a feature vector, it will be classified by the "classifier", which in general is a set of rules that analyses the values in the feature vector and assigns the class accordingly. Our FCS employs a machine learning algorithm, or "learner", to automatically construct and update existing classifier without human-intervention. This facility, thus, makes the FCS adaptive and self-updatable. Two other important components of a FCS, the service classes and features, will be discussed below. Formal definitions of all components of a FCS are provided in [Ana08, Ana09].

3.2 Service Classes

The choice of service classes to be used in a FCS depends on the purpose of the system as it specifies how the flows would be categorised. Since our FCS is developed to assist QoS support, the set of service classes has to be able to capture the QoS requirements. Currently, there are few standardised service classes available, for example, G.1010, 23.107, and RFC 4594. Still, they cannot be used in our scenario directly as they are designed to identify how the packets should be treated (based primarily on delay-sensitivities) and

not to group similar services together. We, thus, propose a set of service classes, which is meant principally for flow classification systems, with each class being designed to be general enough to cover all services with similar QoS requirements as follows.

Strict Conversational Class Real-time audio/video applications are symmetric applications, which are sensitive to delay and delay variation as well as require relatively high data rates (e.g., VoIP, videoconference, and real-time online games).

Relaxed Conversational Class This class of applications is quite similar to the previous class but requires less bandwidth, less delay variation sensitive and intolerable to error. Example applications include telnet, remote desktop and instant messaging.

Streaming Class The streaming services serve streams of data, which include audio and video streams. These services expect high data rate but are not sensitive to delay or delay variation because the data can be buffered and do not need to be used in real-time.

Interactive Class All server access applications fall into this class. The key characteristic of the applications is request-and-response behaviour. This kind of service is asymmetric and only requirement is error intolerance. The example applications are web browsers and email clients.

Background Class In background traffic, the other side of the transmission does not expect the data within a certain period of time and the application will use network resources as they are available, i.e., in best-effort manner. Examples such applications are SMTP, FTP and Server Message Block (SMB) protocols.

3.3 Real-Time Features

The streaming and interactive classes can in turn be distinguished from each other by data volume and burstiness. Flow burstiness characterises how uniform the packet inter-arrival time (IAT) of the flow is. Packet IATs of streaming flows, which transfer data in streams, would be more stable than those of interactive flows, which have to wait for interaction from the other sides of the transmissions. To capture such characteristic, we proposed a feature called throughput difference, which captures the changes of the throughput along the flow [AS07a]. Our previous experiments showed that the feature is highly discriminative. Nevertheless, it is not suitable for real-time flow-capturing as it requires a certain number of packets per calculation window to ensure the correct estimation of throughput.

Therefore, we propose here a new feature, called "packet-size difference", which tries to capture the differences of packet sizes throughout the flow. Like throughput difference, it is aimed at capturing the changes of the flow characteristics over time. However, it can be computed at any flow length that is greater than two. Precise definition of the feature, denoted by $pktSizeDiff$, is given by:

$$pktSizeDiff(p_1, \ldots, p_n) = \frac{\sum_{i=1}^{n-1} |sizeOf(p_i) - sizeOf(p_{i+1})|}{n}$$

Features	Description
protocol	transport protocol of the flow
srcPort	source port
dstPort	destination port
connTime	flow run time (in seconds)
dataVolume	sum of the sizes of all packets in the flow
pktCount	number of packets in the flow (flow length)
pktSizeAvg	average packet size
pktSizeDiff	sum of differences of packet sizes throughout the flow
pktSizeSD	standard deviation of the sizes of packets in the flow
pktSizeRMS	root mean square of the sizes of packets in the flow
dataTPUTAvg	average data throughput (data rate)
pktTPUTAvg	average packet throughput (packet rate)
iatAvg	average packet inter-arrival time
iatSD	standard deviation packet inter-arrival time
iatRMS	root mean square packet inter-arrival time
iatVar	ratio of the iatSD and iatAvg

Table 1: Descriptions of employed features.

where n is the number of packets in a flow, p_i denotes the i-th packet and $sizeOf(p)$ denotes the size of the packet p.

Table 3.3 summarises all features that are used in our approach. Our features, unlike, e.g., [ZP00] and [ML05], do not require any calibrations or fine-tuning by the experts. Features that are not well-defined such as maximum or minimum throughput are also avoided. More importantly, they are designed from ground-up to be able to operate on both TCP and UDP flows, which is not the case in some FCSs [EBR03, BTA$^+$06, EMA$^+$07].

3.4 Learners

In machine learning literature, there exist a large number of learning algorithms (or "learners") designed for different classification problems. As discussed in our previous work [AS07a], "supervised" learners are suitable for flow classification scenario. This is because the set of classes is distinctively defined. In turn, our research is focused on evaluating several supervised learners including J4.8, which is an implementation variant of the famous C4.5 decision tree algorithm [Qui93], PART [FW98] and RIPPER [Coh95] rule generators, Naive Bayes [JL95], and k-Nearest Neighbour (k-NN) [Aha97]. J4.8 first examines all the features at each level of the tree and then determines which one is the most discriminative in separating the classes at their respective levels. Rules generator algorithms, on the other hand, consider each class individually and try to find rules that cover as many data instances of that class as possible, while simultaneously excluding

Class	Number of flows	Percentage
Strict Conversational	4,532	0.01
Relaxed Conversational	4,264,464	12.17
Streaming	8,863	0.03
Interactive	25,138,596	71.74
Bulk	5,609,562	16.01
Total identified flows	35,042,233	

Table 2: Number of identified flows within each class in WIDE traces

the maximum number of instances from the other classes. Naive Bayes classifier, which employs Bayes theorem, works by statistically classifying the flows based on background knowledge. Lastly, k-NN classifies a data instance based on its similarity (or distance) between the new instance and other instances in the dataset.

4 Evaluation

To see how our new flow classification technique performs in real-world, we evaluate our approach on a large dataset that is collected from a sizeable network. In the following, the benchmark dataset, evaluation methodology and evaluation results will be discussed.

4.1 Dataset

In our evaluations, we use the traces obtained from the Widely Integrated Distributed Environment (WIDE) traffic archive [Cho08], which contains partial packet payloads. The advantage of payload-traces is that we can use a signature-based flow classification system to precisely identify the flows service classes. This can be used as the ground truth to evaluate our flow classification system. The traces from the WIDE project are captured in March 2008 in both directions on a 150 Megabit per second Ethernet external link, which connects WIDE backbone and its upstream. They contain the first 96 bytes of every packet's payload. The whole traces are captured in the course of 72 hours from March 18 - 20. For each day, we selected five two-hour traces from different time periods, namely, 0:00-02:00, 08:00-10:00, 12:00-14:00, 16:00-18:00, and 20:00-22:00. In total, we have 30 hours of packet records of real-world traffic, consisting of 138,898,361 flows.

Before we can evaluate the prediction performance of our FCS, we have to identify the actual class of each flow in the packet traces. In doing so, we have developed a signature-based flow classification system, which identifies the application whose a flow belong to based on the signature in the flow payload. As a result, we are able to identify more than 35 million out of 138 million flows (see Table 2). At any rate, as shown in the table, the class distributions are highly skewed. The number of flows in each class varies greatly,

from only 4,532 instances in the Strict Conversational class to more than 25 million in the Interactive class. Therefore, the flow instances are equally sampled into a smaller dataset before the experiments are conducted.

We have randomly sampled 4,000 instances from each class, constituting a dataset of 20,000 instances. However, to avoid any biases, the sampling process is done 10 times. This results in 10 datasets, each of which has 20,000 flow instances stratified with 4,000 instances per class. Note that the number of instances per class is set to 4,000 in this case because it already covers almost all of the 4,532 instances in the strict conversational class. After the randomisation, the flows are extracted into feature vectors, which will be later used by the learner to analyse the relationships between the flows and their classes.

4.2 Evaluation Strategy

The evaluations are carried out in two main phases: accuracy and computational time. In the accuracy phase, the each learning algorithm is evaluated using 10-fold cross-validation (CV) method. In cross-validation, the data are divided into k equal partitions or *folds* and each fold is held out to be used as the test set while the rest are used as the training set. A classifier is then induced from the training set and evaluated against the test set by a learner. This process is repeated over and over every fold has been used as test set (i.e., k iterations). The 10-fold CV method is repeated 10 times resulting in the total of 100 individual tests. In computational time phrase, we analyse the CPU time required by each learner to learn the relationships. The experiments are conducted on WEKA [HFH+09], an open source data mining platform on a 2.8 GHz Intel Core 2 Duo with 2 GB of RAM using Mac OS X 10.5 as the operating system.

4.3 Classification Accuracy

The evaluation results are shown in Table 4.3. PART, RIPPER and J4.8 perform particularly well with more than 99% average correctness compared to the 92.97% achieved by k-NN. Naive Bayes, on the other hand, has much lower average accuracy of just 42.08%.

Moving to per-class correctness, while other methods can classify flow instances of all classes equally well, Naive Bayes performs poorly on all classes except Bulk as it classifies most of the instances as Bulk. This might be because the employed features are correlated and the feature-independence assumption of Naive Bayes does not hold in our domain.

4.4 Computational Time

We have also evaluated the learners in terms of computational time required to induce classifiers (i.e., learning time). In our observations, the classification times of all learners are

Class	J4.8	RIPPER	PART	Naive Bayes	k-NN
Strict Conversational	99.98	99.82	99.91	38.86	99.84
Relaxed Conversational	99.15	99.47	99.57	20.95	95.74
Streaming	98.76	99.21	99.36	41.29	92.33
Interactive	98.75	99.02	99.39	19.10	84.31
Bulk	99.97	99.94	99.97	99.39	91.73
Average	**99.35**	**99.51**	**99.66**	**42.08**	**92.97**

Table 3: Average and class-wise accuracies of the learners (in percentage).

extremely low and not significantly different. Thus, they are not considered here. Figure 1 shows the average learning time of each learner on each of the datasets. Naive Bayes outperforms other methods with only half a second learning time followed by J4.8, PART, and RIPPER respectively. This is because Naive Bayes performs only simple calculations to establish the likelihood of feature values and classes. Conversely, RIPPER requires more than 40 seconds to learn. This is due to the fact that it employs a slow grow-and-prune technique to generate the rule set and, after the rule set is induced, each rule in the set has to be revised again [WF05].

Considering the computational time alone, Naive Bayes would clearly be the preferred choice, even though its prediction accuracies are still inadequate. In comparison the other learners, J4.8 provides the best trade-off between accuracy and computational time.

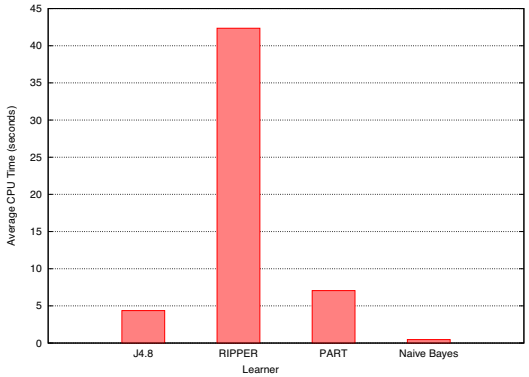

Figure 1: Average CPU time taken to learn from WIDE dataset. k-NN learning time is not included as it is a lazy algorithm.

5 Conclusion

In this paper, we have proposed a set of service classes, features, and machine learning algorithms to construct an adaptive flow-classification system. Particularly, we have proposed a new feature, $pktSizeDiff$, which is intended to capture the burstiness of flows. Advantage of the proposed feature is that it can be computed and any given time. Feasibility of using the feature in real-time classification scenario, where the classification has to be done in a specific period of time, will be invested in our future works.

The excellent evaluation results of the WIDE dataset show that our adaptive classification method is accurate, usable and versatile. Moreover, with its effective learning capability, our FCS is adaptive and self-updatable. In addition, our approach can operate on any transport protocols. The computational time evaluation also shows that learning can be carried out in a very short period of time.

References

[Aha97] David W. Aha. *Lazy learning*. Kluwer Academic Publishers, 1997.

[Ana08] Isara Anantavrasilp. A General Model for Flow Classification Systems. http://www.computational-logic.org/ isara/publications/FCSModel.pdf, 2008.

[Ana09] Isara Anantavrasilp. A Unified Framework for Flow Classification. In *International Conference on Computer Design and Applications*, 2009.

[AS07a] Isara Anantavrasilp and Thorsten Schöler. Automatic Flow Classification Using Machine Learning. In *15th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, 2007.

[AS07b] Isara Anantavrasilp and Thorsten Schöler. Providing Quality-of-Service Support to Legacy Applications Using Machine Learning. In *IADIS International Conference on Telecommunications, Networks and Systems*, pages 75–83, 2007.

[BTA+06] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic Classification on the Fly. *ACM SIGCOMM Computer and Communication Review*, 36(2):23–26, April 2006.

[BTS06] Laurent Bernaille, Renata Teixeira, and Kav Salamatian. Early Application Identification. In *2nd Conference on Future Networking Technologies (CoNext 2006)*, 2006.

[Cho08] Kenjiro Cho. WIDE-TRANSIT 150 Megabit Ethernet Trace 2008-03-18 (collection), 2008. http://imdc.datcat.org/collection/1-05L9-X=WIDE-TRANSIT+150+Megabit+Ethernet+Trace+2008-03-18 (accessed on 2008-07-08).

[Coh95] W. W. Cohen. Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[EBR03] James P. Early, Carla E. Brodley, and Catherine Rosenberg. Behavioral Authentication of Server Flows. In *19th Annual Computer Security Applications Conference*, 2003.

[EMA+07] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, and Carey Williamson. Offline/Realtime Traffic Classification Using Semi-Supervised Learning. *Performance Evaluation*, 64(9-12):1194–1213, October 2007.

[FW98] Eibe Frank and Ian H. Witten. Generating Accurate Rule Sets Without Global Optimization. In J. Shavlik, editor, *Machine Learning: Proceedings of the Fifteenth International Conference*, 1998.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[JL95] George H. John and Pat Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, 1995.

[ML05] Annie De Montigny-Leboeuf. Flow Attributes For Use In Traffic Characterization. Technical Report CRC-TN-2005-03, Communications Research Centre Canada, December 2005.

[MP05] Andrew W. Moore and Konstantina Papagiannaki. Toward the Accurate Identification of Network Applications. *Lecture Notes in Computer Science*, 3431:41–54, 2005.

[MZ05] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 50–60, New York, NY, USA, June 2005. ACM Press.

[Qui93] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[RSSD04] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick Duffield. Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification. In *IMC '04: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pages 135–148, New York, NY, USA, 2004. ACM Press.

[SSW04] S. Sen, O. Spatscheck, and D. Wang. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In *13th International World Wide Web Conference (WWW)*, 2004.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, second edition, 2005.

[XZB05] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 169–180, New York, NY, USA, October 2005. ACM Press.

[ZP00] Yin Zhang and Vern Paxson. Detecting Backdoors. In *Proc. 9th USENIX Security Symposium*, pages 157–170, aug 2000.