INTERNET TRAFFIC CLASSIFICATION FOR SCALABLE QOS PROVISION

Junghun Park[a], Hsiao-Rong Tyan[b], and C.-C. Jay Kuo[a]

[a]Integrated Media Systems Center and Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089-2564
[b]Department of Information and Computer Engineering
Chung-Yung Christian University, Chung-Li 32023, Taiwan
E-mails: junghunp@usc.edu, tyan@ice.cycu.edu.tw, cckuo@sipi.usc.edu

ABSTRACT

A new scheme that classifies the Internet traffic according to their application types for scalable QoS provision is proposed in this work. The traditional port-based classification method does not yield satisfactory performance, since the same port can be shared by multiple applications. Furthermore, asymmetric routing and errors of modern measurement tools such as PCF and NetFlow degrades the classification performance. To address these issues, the proposed classification process consists of two steps: feature selection and classification. Candidate features that can be obtained easily by ISP are examined. Then, we perform feature reduction so as to balance the performance and complexity. As to classification, the REPTree and the bagging schemes are adopted and compared. It is demonstrated by simulations with real data that the proposed classification scheme outperforms existing techniques.

1. INTRODUCTION

Scalable QoS provision over the Internet can be addressed by considering the packet forwarding mechanism and network resource management. They should be designed to support different rate/delay requirements of various flows. The flow-based service depends on the application types. Even though being extensively studied in the literature, QoS-based services are still not yet deployed by Internet service providers (ISPs). One of the main obstacles is that there lacks a simple yet accurate mechanism to classify the application type of a flow.

The classification of flow applications has several major challenges. First, the features to be used in classification have to be simple since a large number of packets have to be processed by the router per time unit. The port-based classification is used traditionally. Its classification performance is however poor since ports are often shared by multiple applications [1, 2]. Furthermore, the observed or extracted features can be corrupted by errors of measuring tools such as PCF [3] and NetFlow [4], which suffer the collision error and the sampling error, respectively. The classification task has to take asymmetric routing into account, which is caused by multiple routers in an ISP. Multiple routers are used for load sharing. However, there is no guarantee that in-coming and out-going packets of the same two-way session go through the same router. Thus, asymmetric routing forces the classification to be performed based on only the information obtained from the one-way direction.

Similar classification work was done previously in [1, 2]. However, they did not consider the scalability problem. To support scalability, we demand that all candidate features used in classification have to be obtained from PCF or NetFlow, and good features are selected by considering the nature of asymmetric routing. Based on selected features, we develop a good Internet traffic classification scheme. Our major contribution is to propose a more practical framework for Internet traffic classification which can be applied to real world data.

The rest of the paper is organized as follows. The feature selection process is presented in Sec. 2. Several classification algorithms used in previous work and our work are described in Sec. 3. Simulation results are given and discussed in Sec. 4. Finally, concluding remarks are provided in Sec. 5.

2. FEATURE SELECTION

The classified object is an IP flow, which consists of 4 tuples, *i.e.* addresses and ports of the source and the destination. The classification is performed based on a feature vector. The components of the feature vector can be decided according to the characteristics of desired class types. The classes and their corresponding characteristics considered in this work are shown in Table 1.

Table 1. Characteristics of classes under classification.

Characteristics	Classes (Applications)
Interactive	www, Telnet, Chat(Messenger)
Bulk data	www, FTP, P2P(Kazza, Grunella)
Real Time	Multimedia
Email	Outlook(smtp,POP, IMAP)
Transactions	DNS, Services(Oracle, X11)

We demand that the extracted features have to be easily obtained or derived from measurement tools of ISP such as PCF or NetFlow. They are primarily the packet-level information, including fields of IP/TCP/UDP packet headers and the arrival time. Their statistics such as average and variance can be updated per observation sample by using the recursive formula that were introduced in [1]. The formula can be computed easily without logging much information in the past so as to reduce the memory requirement, too.

Our selected features include the size and the number of "burst" packets. The burst characteristics can be determined using the arrival time information of packets. That is, consecutive packets are said to be burst, if their inter-arrival time is less than a predefined threshold. Some applications, such as WWW and P2P, have a heavy burst behavior, while streaming and inter-active multimedia applications do not have such a behavior on the average. Thus, this burst feature is expected to provide good discriminant power among these distinctive classes.

Candidate features considered in our work are shown in Table 2. All of them can be obtained with relatively low costs. It is worthwhile to mention that there are features that were discussed in the literature but not listed in Table 2. For example, the FFT (Fast Fourier Transform) of the inter-arrival time can be a good feature as mentioned in [2]. However, it is expensive to compute this feature at ISP due to the limited amount of memory.

Table 2. The list of candidate features.

Method	Features
Directive information	duration of the flow, # of packets, initial AdvertisedWindow bytes, # of actual data packets, # of packets with the option of "PUSH"
Recursive statistics	size of the packets, AdvertisedWindow bytes,inter-arrival time, # and size of the total burst packets
Using a variable updated every sample	# and size of the total burst packets, inter-arrival time

The feature selection process should find a good balance between the complexity (in terms of computation and storage) and performance (in terms of the correct classification rate). More features included, more computation and more memory needed. The complexity issue becomes even more critical when we desire a solution that is scalable to the growth of the network. Furthermore, a large number of features may confuse some classification tools.

There are two commonly used methods in feature reduction; namely, the filter method and the wrapper method. The wrapper method uses classification results as a feedback for feature adjustment. Even though it can provide better performance, the filter method is adopted in our scheme for simplicity. For the filter method, there are several feature selection and evaluation algorithms proposed. For example, the rank search scheme is used in [2] for feature selection. Since low ranked features can be seen as redundancy features, the rank search algorithm can be used to reduce the dimension of the feature space conveniently. The symmetrical uncertain attributed measure is used in [2] for the evaluation. This measure is derived from information theory, and it can represent the correlation between features and classes well.

3. CLASSIFICATION METHODS

A good classification algorithm should meet three criteria: low complexity, high accuracy and robustness. For an ISP to classify a great number of flows, the complexity of the classifier should be low while meeting the desired classification accuracy. To reduce the complexity, the classifier training process should be done in an off-line mode. There exists a gap between the training data set and the test data set since the feasible sample space of Internet flows is too huge and it continues to change due to the emergence of new applications. Thus, a good classifier should be robust with respect to the variation of flows and applications.

The Bayesian analysis technique was used in previous work due to its simplicity, where the conditional probability given a class was assumed to be Gaussian distributed. This is however not true in reality. To overcome this problem, a kernel estimation was adopted in [2] to estimate the distribution. Thus, the density estimation can be expressed by

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

where h is the kernel bandwidth, $j \in J$ is the class index, n_{c_j} is the number of samples in class j in the training data sets and $K(x)$ is the normalized Gaussian distribution of the form $\frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$.

The kernel estimation is however not suitable for this application due to the heavy computation involved. When computing $\hat{f}(x|c_j)$ in (1) to classify an unknown flow, one has to perform n_{c_j} computations. As the number of n_{c_j} increases, the estimation becomes more reliable but the complexity increases as well.

Here, we propose the use of a decision tree to solving the problem. The decision tree classifier demands higher complexity in the training process. However, this is acceptable since the training is done in an off-line mode. A typical decision tree contains internal nodes, which represent the tests to be performed, and leaf nodes, which represent all classification outcomes. The tree construction process (or the training process) consists of two steps. First, we should identify features that provide the best discriminant power among classes and determine a test (*i.e.* a branch in the decision tree) by selecting threshold values of those features. Then, we need to determine a sequence of tests, which corresponds to the decision tree generation.

In the test mode, the decision process can be done by a series of tests, which corresponds to branches of the tree. The final decision goes to a certain leaf node, which gives the application class. Thus, the complexity of the classification process is the depth of the tree, which can be represented by $O(\log n)$, where n is the number of training samples. The computational and memory requirements for Naive Bayesian (NB), Naive Bayesian using Kernel Estimator(NBKE) and the decision tree are shown in Table 3, where d is the number of features and m is the number of testing samples. The space (or memory) complexity in training a tree is the number of nodes, which is 1+2+4+.....+n/2 $\approx n$, that is $O(n)$. By considering the time and space complexities in the classification process, the decision tree classification approach is better than NBKE.

Table 3. Time complexity for Naive Bayesian (NB), NB Kernel Estimator (NBKE) and decision tree.

Operation	NB	NBKE	decision tree
Space for training	$O(d)$	$O(dn)$	$O(n)$
Train on n samples	$O(dn)$	$O(dn)$	$O(dn^2\ logn)$
Test on m samples	$O(dm)$	$O(dmn)$	$O(m \log n)$

The decision tree suffers the overfitting problem caused by noise in the training data, which degrades the classification performance. Thus, we need a pruning process to cut down sub-trees resulted from the noise. However, it is difficult to prune sub-trees optimally since there is no specific relation between the tree size and the classification error. It means that the pruning process is performed repeatedly until the error is reduced as small as possible. After a normal tree is generated, the pruning process is applied to the tree to increase the classification accuracy. Furthermore, the pruning process reduces the size of the decision tree so that the complexity of the decision tree in Table 3 is reduced. The reduction is dependent on the training data sets. Thus, the actual complexity will be predicted after the pruning process with real Internet traffic data sets.

According to the particular method used in calculating errors in the pruning process, there are some variants in the decision tree.

Among several decision tree classifiers, the REPTree (Reduced Error Pruning Tree) classifier is adopted here, since it uses a fast pruning algorithm to increase the accurate detection rate with respect to noisy training data. Furthermore, the pruned tree reduces the complexity in the classification process. Generally speaking, pruning is used to find the best sub-tree of the initially grown tree with the minimum error for the test set. However, the number of sub-trees grows exponentially with the size of the initial tree. Thus, it is computationally impractical to search all sub-trees. REPTree yields a suboptimal tree under the restriction that a sub-tree can only be pruned if it dose not contain a sub-tree with a lower classification error than itself.

More accurate performance can be obtained by paying a higher computation cost. For example, the bagging classfier generates multiple versions of a predictor (which is REPTree in our work) and the final classification result is decided by votes from multiple predictors. Each predictor is trained using a randomly divided sub-set of the entire training set. Due to the complicated nature of the Internet traffic, the bagging classifier is expected to provide better performance than the single REPTree at the cost of higher complexity.

4. SIMULATION RESULTS AND DISCUSSION

4.1. Simulation setup

We used the data sets provided by [5] as training and test data sets in our simulation. The training data were collected at the Pittsburgh Supercomputing Center (PSC) in April, 2005, while the test data were collected at the same place in June and October, 2005. To check the robustness of the proposed algorithm against time, we considered two cases, that had 2- and 6-month gaps between test and training data, respectively. It is difficult to check the payload content for such large supervised data sets. Instead, we used some well known ports to select supervised data sets as performed in [1] first. Then, from selected data sets, features were extracted by following the procedure as described in Sec. 2. The threshold to decide burst packets was chosen empirically as 0.007 second. Then, features were converted to the input format used in WEKA [6], a verified classification tool which has been popularly used in the data mining community. Furthermore, we applied the method mentioned in 2 to reduce the dimension of features. Then, with this reduced set of features, four classifiers were tested and compared. They were: (1) the naive Bayesian classifier, (2) the naive Bayesian classifier using kernel estimation, (3) the REPTree classifier and (4) the bagging classifier using REPTrees. All parameters in each classifier were chosen to be default values provided by the tool. The simulation results remain the same under repeated tests under the above conditions.

4.2. Results and discussion

Table 4 shows seven rank-ordered features in the P2A (passive peer to active peer) and the A2P (active peer to passive peer) directions, respectively, where the rank was decided by the correlation between features and classes mentioned in Sec. 2. The direction is distinguished by TCP SYN flag. It is clear that the P2A and the A2P two directions have different ordered features. The first seven features in each direction stay the same when the number of training data samples is more than 500K. Thus, these seven features are chosen.

We know from Table 4 that the AdvertisedWindow in the TCP header is the key parameter to distinguish applications in the P2A direction. The AdvertisedWindow parameter is the allowed size in the receive buffer. The peer's transmission behaviors are controlled by the parameter. Different applications are distinguished by different transmission behaviors at the peer's side. The statistics of the packet size were shown to be good features in Sec. 2, too.

Furthermore, according to different application types, the client may have a different number of packets flagged with "PUSH" (called push packets for simplicity) due to the following reasons. The TCP sender sets the PUSH flag of a packet to tell its operating system and the receiving end of its TCP connection that all buffered data are delivered to the receiving application. In an interactive application, when a client sends a command to the server, the client would set the PUSH flag and wait for server's response. Without the PUSH flag, this process may hang up because the operating system in the sender or the receiver may continue to wait for additional data. Thus, interactive applications tend to have a larger number of push packets. As another example, when encrypted packets are sent in applications such as sftp, the push option is used for its correct decryption for each packet. Thus, the number of push packets can be used as a good feature as shown in Table 4.

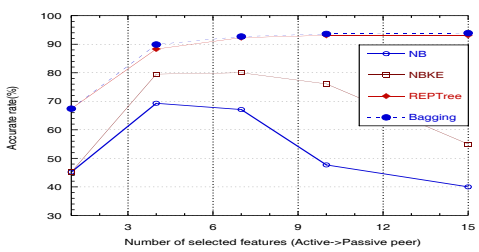
Table 4. The rank-ordered features from symmetrical uncertain attributes. (Avg, Var, and AdvWin mean average, variance and AdvertisedWindow in TCP header, respectively.

Order	P2A direction	A2P direction
1st	initial AdvWin	PUSH packets
2nd	Avg of AdvWin	Avg of AdvWin
3rd	Var of AdvWin	Var of AdvWin
4th	Avg packet size	Var of packet size
5th	size of burst packets	minimum segment size
-	Var of packet size	initial AdvWin
7th	PUSH packets	size of burst packets

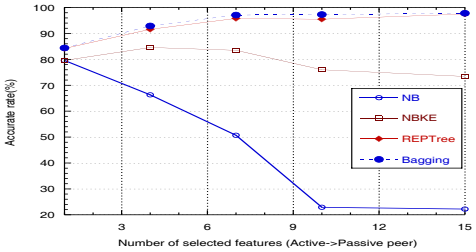
In Fig. 1, (a) and (b) show the the accurate classification rates versus the number of features used in the A2P direction and the P2A direction, respectively. We see that the accurate detection rates of the REPTree and the bagging classifiers start to be saturated if seven or more features are used in both directions. It proves that the seven selected features are enough to classify the application. Also, the two decision tree-based classifiers are better than the two Bayesian methods. The classification performance of the two naive Bayesian methods degrades as the number of feature increases. However, the performance of the bagging classifier is not significantly better than that of the REPTree classifier. It implies that the statistics of training data sets and test data sets are similar. The classifiers are robust under a time gap of 6 months in training and test data sets.

In Sec. 2, the actual complexity of the decision tree can be known after training process with real data sets. Fig. 3 shows the actual complexity of each classifier in P2A direction. As our prediction, Naive Bayesian Kernel estimator(NBKE) suffers the highest computations, while REPTree has the smallest operations. The trend is kept in A2P direction, either. Considering the stationary feature set at the number of 500K training samples, REPTree has the lowest computation complexity satisfying very good accuracy. Thus, REPTree is expected to work as the most powerful classification tool at ISP.

To investigate the sampling effect when NetFlow is enabled, we dropped packets in each data set according to the sampling rate p. In implementation, if the generated random number in [0,1] of a packet was higher than p, it is dropped. The performance of all classifiers degrades for a fixed size of training data. However, if the classifier is trained by training data obtained using the same sampling rate as applied to test data, the classifier has little performance degradation as compared to that with $p = 1.0$ (*i.e.* no sampling is applied). The

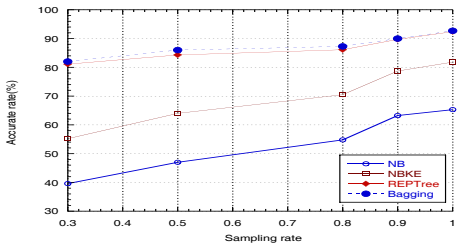


(a) A2P direction

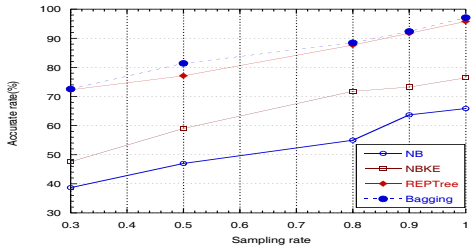


(b) P2A direction

Fig. 1. The accurate detection rates versus the number of features used in (a) the A2P direction and (b) the P2A direction.



(a) A2P direction



(b) P2A direction

Fig. 2. The accurate detection rates versus the sampling rates in (a)the A2P direction and (b) the P2A direction.

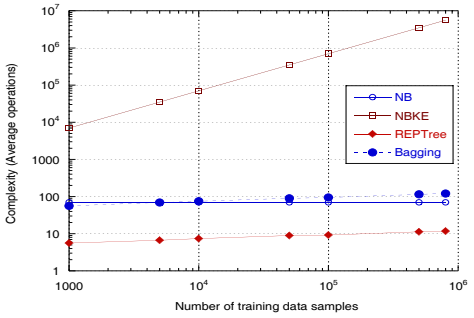


Fig. 3. The complexities according to classifiers.

correct detection performance versus the sampling rate for traffics in the A2P direction and the P2A direction are shown in Fig. 2 (a) and (b), respectively. Given the sampling rate of 0.3, the correct detection rates of the REPTree and the Bagging classifiers was lowered about 10% in the A2P direction and 20% in the P2A direction as compared to those with $p = 1$.

5. CONCLUSION

The problem of classifying application types of Internet traffic flow was examined. This is a challenging problem due to the asymmetric routing and the large number of flows. Furthermore, the accounting tools for the scalability in ISP makes the classification much worse. We proposed effective feature extraction and reduction schemes, which can be applied to one-way traffic, and studied the performance degradation due to the sampling error. Furthermore, the REPTree and the bagging classifiers were presented for the classification task. Both of them were shown to have good detection performance with low complexity. We would like to consider the effect caused by collision errors generated from PCF to design the better classification tool in the future.

6. REFERENCES

[1] 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*, Sicily, Italy, October 2004, pp. 135–148.

[2] A. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proc. ACM Sigmetrics*, Alberta, Canada, June 2005, pp. 50–59.

[3] C. Estan and G. Varghese, "New directions in traffic measurement and accouting," in *Proc. ACM SIGCOMM*, Pittsburgh, Pensylvania, August 2002, pp. 187–200.

[4] Cisco, "Netflow." [Online]. Available: http://www.cisco.com/warp/public/732/Tech/netflow

[5] UCSD PMA(Passive Measurement Analysis), "Nlanr pma." [Online]. Available: http://pma.nlanr.net

[6] Machine Learning Lab in The university of Waikato, "Weka." [Online]. Available: http://www.cs.waikato.ac.nz/ ml