Characterization, Estimation and Detection of Network Application Traffic

H. J. Trussell, A. A. Nilsson, P. M. Patel and Y. Wang
Electrical and Computer Engineering Department
North Carolina State University
Raleigh, NC

***Abstract*—The classification of Internet traffic is of interest in areas like differentiated services and network security. Such classification is usually done using the packet header field of 'port number'. However, recent developments in networking techniques have rendered the port numbers unreliable for this purpose. Our scheme of classification uses the distribution of packet sizes in a buffer or collected during a short time interval at a switch or router. We demonstrate that applications can be classified by these distributions and, estimations of the amount of each application is possible. We compare three methods for estimation of the traffic in various applications; MMSE estimation, POCS and neural networks. Detection of the presence of individual applications can be done reliably. Methods that use artificial neural networks performed best in our tests.**

I. Introduction

The introduction of voice, video and other real-time applications has changed the way the Internet is used. This has triggered the need for a change in traffic handling on the Internet. In particular, there is increasing demand for service differentiation. The traffic on the Internet can be classified using various parameters such as source and/or destination IP address (or prefix), type-of-service, application, etc. We note that we do not classify the traffic on a per-packet basis, rather the traffic flows are classified as containing packets from various types of applications that have different needs for timed service. By identifying the flows that have significant quantities of time-sensitive data, such as voice-over-IP or real-time video, a switch or router can give preference to these flows. This will allow an increase in quality of service (QoS). In addition, the detection of certain applications, such as peer-to-peer file transfer, such as Napster and eDonkey, can aid the network administrator in limiting unwanted actions by users.

Packets are identified in the network traffic by determining to which application an incoming packet belongs. In this work, we detect and estimate characteristics of Internet traffic based on the application to which the packets belong. An easy approach to classification is by extracting the port number from the layer 4 (TCP/UDP) header [1]. However, there are certain problems with this approach. With the increasing use of the Network Address Port Translation (NAPT), the port numbers may not be a trustworthy source for determining the application type. In a free environment like the Internet, it is not mandatory for applications to use specific port numbers [2]. If QoS were based on port number, it is possible that other applications would spoof port numbers in order to gain better service. For this work, we use port number to establish the application for recorded packet data. However, we recognize that this method may yield some inaccuracies. As we shall see, this does not appear to be a problem for our training data.

We propose using the packet size distribution as an indicator of application type. The distribution is part of the application software characteristics. While it may be changed, any change to disguise its character would likely degrade performance. The distribution is obtainable from the packet size field at OSI layer 3 i.e. the IP layer. We avoid prying into the TCP header, which takes additional time and computation and may be encrypted in the future. We read the TCP header in this work only to establish training sets and performance measures.

The fact that applications can be identified by their packet size distributions was shown in our previous work [3]. Even though that work used data that is now several years old and many characteristics have changed, the fact that applications can be identified by their packet size distributions has not. Our experience shows that, even though the character of the distributions may evolve over a long time period, applications may be distinguished based on this statistic.

II. DATA Description

A. Data Collection

The data for this project is collected from the North Carolina State University backbone network using TCPDUMP software. The data was collected continuously for four hours. Each 5 minute interval was recorded in a separate text file resulting in 48 data sets. The recorded parameters of interest are: Source port number, Destination port number, Packet size (in bytes). Note that packets may be reconfigured by subnets through which they pass. Large packets may be divided into smaller segments to pass through these networks. Having noted this, the quantity in the header that is used to indicate the size of the collection of bytes, will be referred to as packet size. The applications were identified using the source and destination port numbers depending on the port assignments by IANA [1].

The top 20 applications in the data, their associated port numbers, their percentage in the total traffic and their cumulative percentage are tabulated in Table 1 in the order of their percentage in total traffic. All the other applications in the traffic mixture are grouped under 'Other'.

B. Histogram Generation

In order to reduce the dimensionality of the data, the Ethernet packet sizes range from 60-1514 bytes was divided into a manageable number of bins. Because of the sparseness of the data in the histograms and to reduce the dimensionality of the problem, we constructed histograms with variable bin sizes. We first examined histograms with unit bin-width. Then new bin sizes were determined based on the criterion that no two peaks, seen in the unit bin-width histograms, fall in the same histogram bin. The packet size distributions of some of the major applications using 60 bins are shown in Figure 1.

Table 1. Top 20 Applications, its port numbers and %

	Application	Port #	%	Cum %
1	HTTP	80	15.28	15.28
2	Kazaa	1214	4.35	19.63
3	FTP	20	2.73	22.36
4	Gnutella	6346	1.77	24.12
5	Unassigned	3933	1.23	25.35
6	RTP	6970	1.13	26.48
7	Napster	6699	1.08	27.56
8	eDonkey	4662	0.97	28.52
9	AOL	5190	0.92	29.45
10	Multicast	16384	0.92	30.37
11	Half Life Server	27015	0.87	31.24
12	Plethora	3480	0.77	32.01
13	Reserved	0	0.63	32.64
14	IRC	6667	0.62	33.26
15	ms-olap2	2394	0.56	33.83
16	SMTP	25	0.54	34.37
17	Half Life Client	27005	0.54	34.91
18	ICAP	1344	0.52	35.43
19	tragic	2642	0.51	35.95
20	mloadd	1427	0.49	36.44
21	Other		63.56	100.00

C. Clustering Analysis

To verify the conjecture that applications could be reliably characterized by their hisograms, we analyzed the histogram collection using clustering. We used several clustering methods, which all resulted in natural groupings of the histograms of applications. The clustering results using Ward's method for 12 clusters are shown in Table 2, for the top 21 applications using a 60-bin histogram scheme. The percentages in the table denote the percentage of the application in the network traffic present in the cluster. From Table 2, it is clear that for each application, all the 48 samples fall into a single cluster. This shows the similarity of the applications.

III. ESTIMATION AND DETECTION

The total distribution of packet sizes at a particular network node is the mixture of the distribution of the individual applications. We can model the total network traffic as the linear combination of major applications, as

$$E\{\mathbf{h}\} = E\{\mathbf{B}\}\mathbf{a} \qquad (1)$$

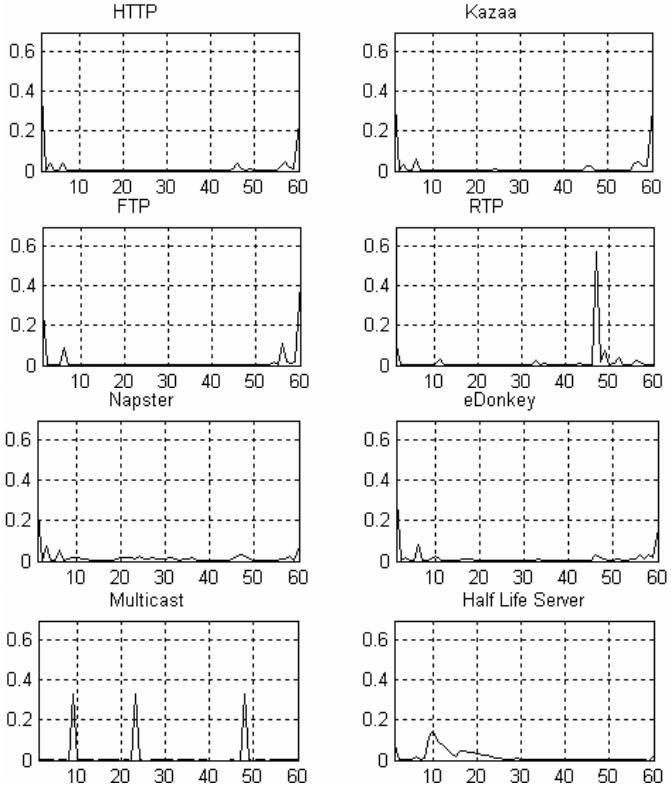

Figure 1. Packet Size Distribution of some applications

where $E\{\cdot\}$ is the expected value operator, $\mathbf{h}_{M\times 1}$ is the total network traffic distribution, given as an M bin histogram, $\mathbf{B}_{M\times N}$ is the distribution of M bins of a histogram of each of the top N-1 applications and one histogram of the "other", and $\mathbf{a}_{N\times 1}$ is the proportion of each of the applications. We know $\mathbf{h}$ and $\mathbf{B}$ in a statistical sense by collecting packet samples over a fixed period of time. Since the components of the mixture are probability distributions, the elements of $\mathbf{a}$ are constrained to the set S_a defined in eq.(2).

$$S_a = \left\{ \mathbf{a} \in \mathbf{R}^N \,\middle|\, \sum_{k=1}^{N} a_k = 1, \quad 0 \le a_k \le 1 \right\} \qquad (2)$$

Table 2. Clustering 20 major applications and others into 12 clusters using Ward's Minimum Variance method

Ward's Method 12 clusters - 60 bins												
	CL28	CL13	CL12	CL14	CL36	CL15	CL39	CL17	CL34	CL31	CL23	CL24
HTTP		100%										
Kazaa						100%						
FTP											100%	
Gnutella			100%									
Unassigned		100%										
RTP					100%							
Napster										100%		
eDonkey		100%										
AOL			100%									
Multicast	100%											
Half Life Server								100%				
Plethora				100%								
Reserved												100%
IRC									100%			
ms-olap2						100%						
SMTP							100%					
Half Life Client								100%				
ICAP						100%						
tragic		100%										
mloadd		100%										
Other				100%								

We can estimate $E\{\mathbf{B}\}$, denoted as $\overline{\mathbf{B}}$, from observations. For any particular time interval, we can estimate $\mathbf{a}$ from

$$\mathbf{h} = \overline{\mathbf{B}}\,\mathbf{a} \tag{3}$$

However, this would be a crude estimate since we know that for any particular sampling interval the histogram associated with any application will not be the mean. We can include this uncertainty in the problem by writing

$$\mathbf{h} = (\overline{\mathbf{B}} + \Delta\mathbf{B})\,\mathbf{a} \tag{4}$$

where $\Delta\mathbf{B}$ represents the variation from the mean.

This paper gives a comparison of three methods of estimating the percentage of traffic in each class, as well as showing that detection of various important classes is possible with very high accuracy. The estimation methods are constrained least squares, projection onto convex sets (POCS) and neural networks. The POCS methods can handle the uncertainty in the basis matrix by using an approach similar to total least squares (TLS). All of these methods are able to take into account the constraints imposed by the fact that the estimated quantities are probabilities. The usual TLS methods cannot use these constraints directly in the solution formulation.

The architecture used for the neural networks was a simple single hidden layer with a single output neuron. In all cases, the hidden neurons used a log-sigmoid function response. For estimation, the output neuron used a linear function; while for the detection case, the output neuron used a log-sigmoid function. In the case of estimation, we found that using six hidden neurons produced good results with no improvement in performance if that number were increased. In the case of detection, it was found that two hidden neurons were sufficient to give good results, with little improvement if the number were increased.

A typical result of estimation performance is given in Table 3. The RMS error obtained by the neural networks is better than the other methods. Note that this result is obtained by training on one set of 24 samples and testing on the other set of 24. If we limit the estimation to the percentage of a single application, all methods improve but the neural net still performs best, as shown in Table 4.

For detection of the presence of a single application, we wished to estimate the probability of a specific application being present in the traffic flow. The detection was done using the neural network. Since the original data contains most applications in each data set, to test detection, we created artificial data sets, based on actual data files. These sets had varying proportions of certain application packets. To create these sets, we randomly remove P% of the target application packets (RTP, Napster or eDonkey) and Q% of other applications to form a new set, with $P = \{0,20,40,60,80,100\}$ and $Q = \{0,10,20,30\}$. This varies both the application under study and its background traffic.

Table 3 Estimation of top 100 applications in set 1

Application	Average a	rms err		
		CLLSQ	POCS	NN
HTTP	0.1532	0.0109	0.0027	0.0047
Kazza	0.0436	0.0108	0.0031	0.0029
FTP	0.0340	0.0339	0.0019	0.0092
Gnutella	0.0176	0.0132	0.0011	0.0008
Unassigned	0.0122	0.0156	0.0008	0.0087
RTP	0.0119	0.0014	0.0016	0.0004
Napster	0.0111	0.0035	0.0028	0.0004
eDonkey	0.0097	0.0127	0.0019	0.0003
AOL	0.0092	0.0030	0.0007	0.0004
Multicast	0.0092	0.0002	0.0036	0.0006
…	…	…	…	…
Other	0.4460	0.0240	0.0171	0.0136
Average rms err		0.0059	0.0027	0.0012

Table 4. Estimation of single application in set 1

Application	Average a	rms error		
		CLLSQ	POCS	NN
RTP	0.0119	0.0010	0.0029	0.0004
Napster	0.0111	0.0016	0.0013	0.0001
eDonkey	0.0097	0.0052	0.0010	0.0002

The method obtains a very high accuracy of detecting the present of specific applications, even at low percentages. Table 5 shows an example of the detection performance. The lower detection rate of eDonkey is caused by the fact that eDonkey has statistical properties that are similar to other applications. This is apparent from the clustering methods that showed separation of eDonkey from other applications late in the process.

An advantage of the neural network approach is that the weights associated with the hidden layer give a good indicator of the importance of various bins in the histogram for detecting the present of the applications. This will allow the reduction of the size of the histograms and a corresponding decrease in computation time.

We noted that we reduced the data histograms from 1514 bins to 60 using simple heuristics. This reduced computation time and the amount of information that needs to be stored. By considering the weight vectors associated with the input to the hidden layer neurons, we can determine if this number of bins can be reduced further. Very small weight on a particular bin of input vectors for all neurons indicates that this bin is not needed for estimation or detection.

An example of the weights for a realization of the six hidden neuron estimation networks is shown in Figure 2. We notice that bins which are associated with smaller size packets play an important role in our estimation experiments. While bins in the range of [10, 12], [21, 23], [35, 45] and [55, 60] always have small weights. We can eliminate them in the future histogram generation.

In the detection problem, the weights for two neurons are shown for the cases of Napster and eDonkey in Figures 3 and 4, respectively. The applications have their own characteristic bins, but they are not the same. In all test cases for training, the significant weights in the two neurons usually appear at the same bins. It should be noted that it was not possible to reduce the number of hidden neurons to one and maintain acceptable detection performance.

Table 5. Detection results of RTP, Napster and eDonkey

		RTP		Napster		eDonkey	
P%	Q%	S12	S21	S12	S21	S12	S21
0	0	100	100	100	100	100	100
0	10	100	100	100	100	100	100
0	20	100	100	100	100	100	100
0	30	100	100	100	100	100	100
20	0	100	100	100	100	100	100
20	10	100	100	100	100	100	100
20	20	100	100	100	100	100	100
20	30	100	100	100	100	100	100
40	0	100	100	100	100	100	100
40	10	100	100	100	100	100	100
40	20	100	100	100	100	100	100
40	30	100	100	100	100	100	100
60	0	100	100	100	100	100	100
60	10	100	100	100	100	100	100
60	20	100	100	100	100	100	100
60	30	100	100	100	100	100	100
80	0	100	100	92	100	88	96
80	10	100	100	100	100	96	96
80	20	100	100	100	100	100	100
80	30	100	100	100	100	100	100
100	0	100	100	100	54	75	33
100	10	100	100	100	54	75	38
100	20	100	100	100	54	79	33
100	30	100	100	100	54	75	33

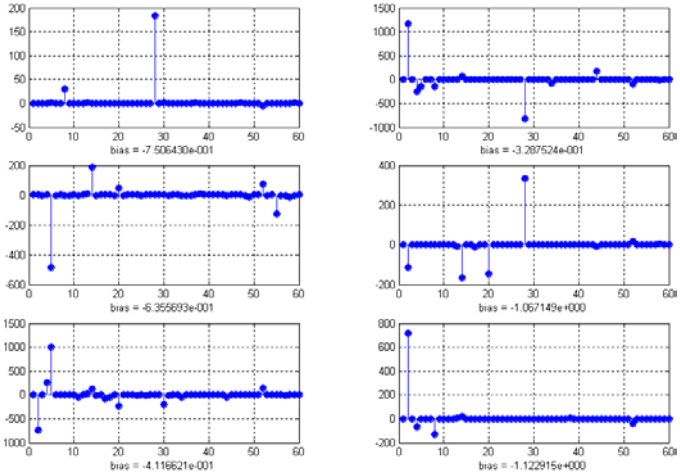

Figure 2. Weights of 6 hidden layer neurons in estimation of the percentage of 21 applications

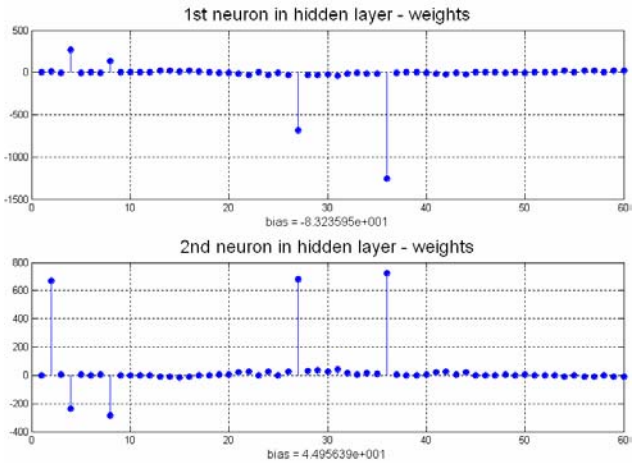

Figure 3. Weights of hidden layer neurons in Napster detection

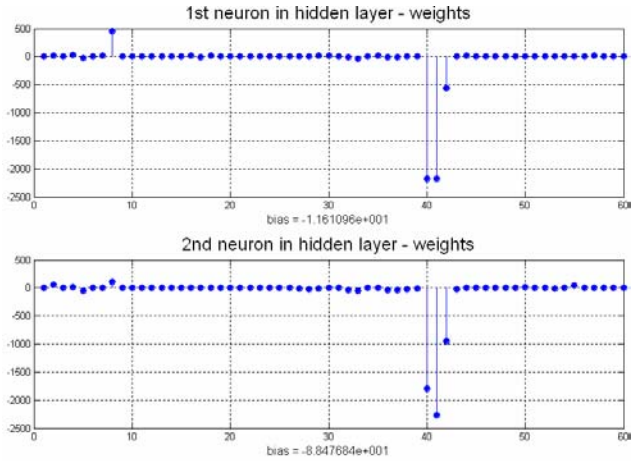

Figure 4. Weights of hidden layer neurons in eDonkey detection

REFERENCES

[1] Assigned Port Numbers Available:

http://www.iana.org/assignments/port-numbers

[2] Fulu Li, Nabil Seddigh, Biswajit Nandy, Diego Malute, "An empirical study of today's Internet traffic for Differentiated Services IP QoS", Proceedings of ISCC 2000.

[3] Chintan Trivedi, H. Joel Trussell, Arne A. Nilsson and Mo-Yuen Chow, "Implicit Traffic Classification for Service Differentiation," *ITC Workshop*, July 24&25, 2002, Wurtzburg, Germany