Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification

Matthew Roughan[§] Subhabrata Sen[⋆] Oliver Spatscheck[⋆] Nick Duffield[⋆]

School of Mathematical Sciences, University of Adelaide,SA 5005, Australia[§]
AT&T Labs – Research, Florham Park, NJ 07932-0971, USA[⋆]

matthew.roughan@adelaide.edu.au {sen,spatscheck,duffield}@research.att.com

ABSTRACT

The ability to provide different Quality of Service (QoS) guarantees to traffic from different applications is a highly desired feature for many IP network operators, particularly for enterprise networks. Although various mechanisms exist for providing QoS in the network, QoS is yet to be widely deployed. We believe that a key factor holding back widespread QoS adoption is the absence of suitable methodologies/processes for appropriately mapping the traffic from different applications to different QoS classes. This is a challenging task, because many enterprise network operators who are interested in QoS do not know all the applications running on their network, and furthermore, over recent years port-based application classification has become problematic. We argue that measurement based automated Class of Service (CoS) mapping is an important practical problem that needs to be studied.

In this paper we describe the requirements and associated challenges, and outline a solution framework for measurement based classification of traffic for QoS based on statistical application signatures. In our approach the signatures are chosen in such as way as to make them insensitive to the particular application layer protocol, but rather to determine the way in which an application is used – for instance is it used interactively, or for bulk-data transport. The resulting application signature can then be used to derive the network layer signatures required to determine the CoS class for individual IP datagrams. Our evaluations using traffic traces from a variety of network locations, demonstrate the feasibility and potential of the approach.

Categories and Subject Descriptors

C.2.3 [**Computer-Communications Networks**]: Network Operations—*Network Management, Network Monitoring*

General Terms

Algorithms, Management, Measurement

Keywords

Statistical Signatures, Traffic Classification, Class of Service (CoS), Quality of Service (QoS)

1. INTRODUCTION

The past few years have witnessed a dramatic increase in the number and variety of applications running over the Internet and over enterprise IP networks. The spectrum includes interactive (e.g., telnet, instant messaging, games etc.), bulk data transfer (e.g., `ftp`, P2P file downloads), corporate (e.g., Lotus Notes, database transactions), and real-time applications (voice, video streaming, etc.), to name just a few.

Network operators (particularly in enterprise networks) are actively seeking the ability to support different levels of Quality of Service (QoS) for different types of applications. The need is driven by (i) the inherently different QoS requirements of different types of applications (e.g., low end-end delay for interactive applications, high throughput for file transfer applications etc.); (ii) the different relative importance of different applications to the enterprise: e.g., Oracle database transactions may be considered critical and therefore high priority, while traffic associated with browsing external web sites is generally less important; and (iii) the desire to optimize the usage of their existing network infrastructures under finite capacity and cost constraints, while ensuring good performance for important applications. In spite of a clear perceived need, and the fact that various mechanisms (diffserv, traffic prioritization, etc. [8, 17, 7]) have been developed for providing different service quality guarantees in the network, their adoption has not been widespread. A pertinent question then is: *what ails QoS?*

Realization of service differentiation capabilities requires association of the traffic with the different applications, determination of the QoS to be provided to each, and finally, mechanisms in the underlying network for providing the QoS. Based on interactions with large enterprise network operators, we believe that a key issue behind the slow spread of QoS-use is not the lack of interest or need, but rather, the absence of suitable mapping techniques that can aid operators in classifying the network traffic mix among the different QoS classes. We refer to this as the Class of Service (CoS) mapping problem, and hypothesize that solving this would go a long way in making the use of QoS more accessible to operators.

In principle the division into CoS could be done by end-points — for instance by end-user applications. However, for reasons of trust, and scalability of administration and management it is typically more practical to do this within the network, for instance at the router that connects the Local Area Network (LAN) to the Wide Area Network (WAN). Alternatively there might be appliances which sit near the LAN to WAN transition point performing packet marking for QoS.

CoS mapping inside the network is a non-trivial task. Ideally, a network system administrator would possess precise information on the applications running inside their network, along with simple, unambiguous mappings from easily obtained traffic measurements to applications (e.g. by port numbers, or source and destination IP addresses). This information is vital not just for the implementation of CoS (e.g. via diffserv), but also in planning the capacity required for each class, and balancing tradeoffs between cost and performance that might occur in choosing class allocations. For instance, one might have an application whose inclusion in a higher priority class is desirable, but not cost effective (based on traffic volumes and pricing), and so some difficult choices must be made. Good data is required for these to be informed choices.

However, in general, the required information is rarely up-to-date or complete, if it is available at all. The traditional ad-hoc growth of IP networks, the continuing rapid proliferation of new applications, the merger of companies with different networks, and the relative ease with which almost any user can add a new application to the traffic mix with no centralized registration are some factors contributing to this "knowledge gap". Furthermore, over recent years it has become harder to identify network applications within IP traffic. Traditional techniques such as port-based classification of applications have become much less accurate (details in Section 2).

This paper presents a signature-based traffic classification framework as a candidate solution for the CoS mapping problem, and demonstrates the feasibility and potential of the approach using large traffic traces collected from multiple Internet locations. Our classification method is based on utilizing the statistics of particular applications in order to form signatures. By choosing signatures that are determined by the way in which an application is used (e.g. is it used interactively, or for bulk data transport), we should obtain signatures that are insensitive to the particular application layer protocol. We can therefore perform CoS categorization without detailed knowledge of the specific application protocol, or usage case (some applications may be used for multiple tasks with different QoS requirements). The method would be used off-line to form a set of port, or IP address based rules for CoS assignment that would then be applied on-line in the QoS implementation. Our evaluations indicate that the technique has relatively low error rates. For example, our cross-validation tests using 3-Nearest Neighbor (details in Section 5) yield classification error of 2.5% and 5.1% respectively when the traffic is categorized into three and four classes, using two features. We found that even better results (for instance 0.0% errors were possible) with three features. The evaluations even showed relatively low error rates even for fine grain traffic classes. The last suggests that such statistical classification techniques may be good candidates for identifying even individual applications. Such identification of the different applications and their associated network traffic has a number of important usages for network operations and management (outside of QoS implementation), including application-specific traffic engineering, capacity planning, provisioning, and security.

The remainder of this paper is organized as follows. Section 2 overviews existing techniques for identifying IP traffic and their limitations. Section 3 presents a three-phase framework for realizing CoS mapping. Section 4 presents our techniques for CoS classification from network traffic. Section 5 presents evaluations of our techniques using real traffic traces. Finally, Section 6 concludes the paper.

2. IP TRAFFIC CLASSIFICATION

One approach commonly used for identifying applications on an IP network is to associate the observed traffic (using flow level data, or a packet sniffer) with an application based on TCP or UDP port numbers. We argue here that this method (described below) is inadequate.

The TCP/UDP port numbers are divided into three ranges: the Well Known Ports (0-1023), the Registered Ports (1024-49,151), and the Dynamic and/or Private ports (49,152-65,535). A typical TCP connection starts with a SYN/SYN–ACK/ACK handshake from a client to a server. The client addresses its initial SYN packet to the well known server port of a particular application. The source port number of the packet is typically chosen dynamically by the client. UDP uses ports similarly to TCP, though without connection semantics. All future packets in either a TCP or UDP session use the same pair of ports to identify the client and server side of the session. Therefore, in principle the TCP or UDP server port number can be used to identify the higher layer application, by simply identifying which port is the server port and mapping this port to an application using the IANA (Internet Assigned Numbers Authority) list of registered ports [20]. However, port-based application classification has limitations. First, the mapping from ports to applications is not always well defined. For instance,

- Many implementations of TCP use client ports in the registered port range. This might mistakenly classify the connection as belonging to the application associated with this port. Similarly, some applications (e.g. old `bind` versions), use port numbers from the well-known ports to identify the client site of a session.
- Ports are not defined with IANA for all applications, e.g. P2P applications such as Napster and Kazaa.
- An application may use ports other than its well-known ports to circumvent operating system access control restrictions, e.g., non-privileged users often run WWW servers on ports other than port 80, which is restricted to privileged users on most operating systems.
- There are some ambiguities in the port registrations, e.g. port 888 is used for CDDBP (CD Database Protocol) and access-builder.
- In some cases server ports are dynamically allocated as needed. For example, FTP allows the dynamic negotiation of the server port used for the data transfer. This server port is negotiated on an initial TCP connection which is established using the well-known FTP control port.
- The use of traffic control techniques like firewalls to block unauthorized, and/or unknown applications from using a network has spawned many work-arounds which make port based application authentication harder. For example port 80 is being used

by a variety of non-web applications to circumvent firewalls which do not filter port-80 traffic. In fact available implementations of IP over HTTP allow the tunneling of all applications through TCP port 80.

- Trojans and other security (e.g. DoS) attacks generate a large volume of bogus traffic which should not be associated with the applications of the port numbers those attacks use.

A second limitation of port-number based classification is that a port can be used by a single application to transmit traffic with different QoS requirements. For example, (i) Lotus Notes transmits both email and database transaction traffic over the same ports, and (ii) `scp` (secure copy), a file transfer protocol, runs over `ssh` (secure shell), which is also used interactively on the same port (TCP port 22). This use of the same port for traffic requiring different QoS requirements is quite legitimate, and yet a good classification must separate different use cases for the same application. In practice, one use case may dominate on a particular VPN (Virtual Private Network), or use cases will have other discriminating factors such as the servers' IP addresses. Thus a clean QoS implementation is still possible through augmenting the classification rules to include IP address-based disambiguation. Server lists exist in some networks, but again, in practice these are often incomplete, or a single server could be used to support a variety of different types of traffic, so we must combine port and IP address rules.

A possible alternative to port based classification is to use a painstaking process involving installation of packet sniffers and parsing packets for application-level information to identify the application class of each individual TCP connection or UDP session. However, this approach cannot be used with more easily collected flow level data, and its collection is computationally expensive, limiting its application to lower bandwidth links. Also this approach requires precise prior knowledge of applications and their packet formats – something that may not always be possible. We use it here to identify Kazaa and Gnutella traffic from their application layer protocols, but this requires substantial effort for each application, and in some cases, each application version. Furthermore, the introduction of payload encryption is increasingly limiting our ability to see inside packets for this type of information.

In this paper we explore the potential of signatures derived from measured traffic for CoS categorization. In practice, this approach, in conjunction with the above techniques and the partial knowledge available for most corporate networks can be used to bear on the problem of application identification and traffic classification.

2.1 Related work

Previous related work has examined the variation of flow characteristics according to application. Claffy [10] investigated the joint distribution of flow duration and number of packets, and its variation with flow parameters such as inter-packet timeout. Differences were observed between the support of the distributions of some application protocols, although overlap was clearly present between some applications. Most notably, the support of the distribution of DNS transactions had almost no overlap with that of other applications considered. The use of such distributions as a discriminator between different application *types* was not considered. There exists a wealth of other research on characterizing and modeling workloads for particular applications, e.g., [22, 31, 4, 2, 9, 34]. An early work in this space, [29], examines the distributions of flow bytes and packets for a number of different applications. Interflow and intraflow statistics are another possible dimension along which application types may be distinguished. [30] found that user initiated events—such as `telnet` packets within flows or `ftp-data` connection arrivals—can be described well by a Poisson process, whereas other connection arrivals deviate considerably from Poisson.

All these studies assume that one can identify the application traffic unambiguously and then obtain statistics for that application. In contrast, we are considering the dual problem of inferring the application from the traffic statistics. This type of approach has been suggested in very limited contexts such as identifying chat traffic [12]. Our work extends this idea while providing a rigorous mathematical framework for performing classification based on signatures. Signature-based detection techniques have also been explored in the context of network security, attack and anomaly detection (e.g. [6, 5, 36, 26]) where one typically seeks to find a signature for an attack. However, we apply our classification techniques to identify everyday traffic.

3. REALIZING COS MAPPING

The constraints under which CoS mapping must operate are principally related to computational complexity. At high speeds, the rules that can be used for this task are rather limited. Typically, one can use simple criteria such as port numbers or IP source/destination addresses, but not details from the higher layer protocols. Our task is to use statistical signature based classification (trained on prior networks' traffic, and applied to traffic measurements from the current VPN) to form a set of local rules based on port, or IP addresses, which would then be applied on-line for CoS assignment.

We propose to realize this type of CoS mapping using a three stage process.

1. statistics collection,
2. classification,
3. rule creation.

The first stage — statistics collection — involves placing monitors in the network, and collecting appropriate statistics of the traffic from certain aggregates. In this case, the aggregates we consider are the server port P_i, and server IP address[1] I_i of a connection i. One could choose these aggregates in a more flexible manner, with the proviso that the aggregates used must be easily implementable as rules (see below). There are a number of techniques available for efficiently identifying the interesting aggregates, for instance, see [13]. We then form a vector of statistics $\mathbf{S}^C(i)$ for each connection i, and use this to update the statistics of each aggregate that connection is involved in, for instance statistics $\mathbf{S}^P(P_i)$ for port aggregates, and $\mathbf{S}^I(I_i)$ for server aggregates.

[1]Notice that server address are usually statically allocated, rather than allocated via DHCP, and so we do not need to worry about constantly shifting IP addresses, at least on the time scales of measurements considered here.

For statistics collected on TCP connections, the procedure would be:

```
 foreach packet
if new TCP connection (give it index $i$++)
  determine the aggregates for $i$
    server port $P_i$ = dst port of SYN
    server IP address $I_i$ = dst IP of SYN
    ...
  initialize a set of statistics $\mathbf{S}^C(i)$
else if part of existing TCP connection $i$
  update connection statistics $\mathbf{S}^C(i)$
else if end TCP connection $i$
  update connection statistics $\mathbf{S}^C(i)$
  update statistics foreach aggregate
    by server port: $\mathbf{S}^P(P_i)$
    by server IP address: $\mathbf{S}^I(I_i)$
```

The update procedure for connections depends on the statistic in question. Ideally, we should choose statistics that can be updated on-line in a streaming fashion, i.e. recursively. This means that we do not need to store data per packet, but rather per connection, for instance, assuming an update algorithm like

$$S_k^C(i) \leftarrow f(X_j^i, S_k^C(i), \phi(i)),$$

where X_j^i is the measurement for the jth packet in connection i, and $S_k^C(i)$ is the kth statistic for connection i, and $\phi(i)$ is some (small) set of state information (e.g. the packet number j) for connection i, then the memory required to store the state depends on the number of connections, not number of packets. For example, for a series of real-valued data X_j, the following statistics may be easily computed recursively:

1. **average:**

$$\bar{X}_{j+1} = \frac{1}{j+1} X_{j+1} + \frac{j}{j+1} \bar{X}_j,$$

2. **variance:**

$$\mathrm{var}\,(\mathbf{X}_{j+1}) = \frac{1}{j} X_{j+1} + \frac{j-1}{j} \mathrm{var}\,(\mathbf{X}_j) + \frac{j}{j-1} \bar{X}_j^2 - \frac{j+1}{j} \bar{X}_{j+1}^2.$$

where $\bar{X}_j$ and $\mathrm{var}\,(\mathbf{X}_j)$ are the mean and variance, respectively, of the first j samples of data. However, even for more difficult statistics, such as quantiles, there are a number of approximation algorithms [18] that can be used to approximate the statistic on-line. The variables X_j could represent packet size, or inter-arrival time, or other features, and so we can generate a moderately large number of statistics even with the limitation of on-line computation. Some statistics need only be computed at the start and end of the TCP connection — for instance, the duration, which we may compute by including the start time of connection i in the state variables $\phi(i)$.

Likewise, it is appealing to be able to update the statistics of each aggregate recursively, but this is not necessary, as it is much easier to store one set of statistics per connection than per packet. If the statistics for each connection are stored, then we could alternatively compute the statistics per aggregate off-line, after the data collection.

We may also finalize any extant TCP connections at the end of data collection in one of two ways: by including them in our statistics, or excluding them. Either approach biases the results — for instance, if we exclude the connections we naturally exclude any connections longer than our measurement interval, but if we include them we underestimate the duration of the connections. These edge effects will be minimized by having a longer data collection interval, so in this work we propose using one day worth of data, though it may be practical to use longer data sets. Ideally, the collection intervals for training data should be the same as those for test data, so that both data sets are subject to the same biases.

After statistics collection, unsurprisingly, one has a collection of statistics indexed by aggregate (in our case server port, and server IP address). The next step is to classify the traffic on each aggregate. We do so using the classification algorithms described in Section 4 (or alternative algorithms if these are shown to be more accurate), in conjunction with a pre-existing set of training data, carefully collected on well understood networks of the type under study.

Once we have completed classification we will have associated each aggregate with a class. Assuming the classes map directly to CoS we can immediately construct the rules required in the QoS implementation. That is, assume that aggregate A_i has been determined to belong to class C_j, which requires QoS Q_k. We now have a mapping

$$A_i \rightarrow C_j \rightarrow Q_k,$$

which can be instantiated as a rule. For instance, if $A_i \equiv P_i$ corresponds to a particular server port P_i, our rule is

```
place traffic to/from port $P_i$ in class $Q_k$.
```

One can obviously create a large set of such rules, and in general it might be non-trivial to reduce the size of this rule set to something manageable. However, in practice, differential pricing between classes means that only traffic specifically requiring high priority should go into high priority classes, and so the majority of application will most likely be placed in a lower priority class. Hence there are typically only a few classes, and the majority of aggregates will go into a default class, so few rules will be needed.

Once a set of rules has been created, these would then be implemented on (for example) the access router, which would use them to mark the packets appropriately, and place them in appropriate queues for forwarding. Thus the classification process described so far is an off-line process, to be applied before the fact to create a set of simple rules that would be used in the actual on-line QoS implementation. Note that the error rate of the classification algorithm is in forming these rules, not in classification of packets in the actual QoS implementation. This is an important distinction: the error rate does result in packets being placed in suboptimal classes, however, these packets only pay for the class in which they are placed. This is not the same, for example, as paying a business class airfare and being bumped to economy. It is simply a case of an administrator occasionally booking the wrong class of ticket because he has incorrectly categorized the importance of some of his administratees.

Furthermore, the above process (as described) is automated. However, it is desirable for a human to "double-check" the assignment rules. The method described here cannot be made 100% foolproof simply because a particular network operator may have specific requirements which deviate from the norm. A human would be able to map the classes/rules to known traffic and servers, and ensure that the known policy based rules were enforced in priority to those derived above. Think of this as our disgruntled business class passenger updating his administrator's list of important people, so that he receives the air transport he deserves in the future. In essence, the technique above is intended to fill in the unknown gaps in the best possible way.

A side benefit of the above approach is that the collected statistics can be used in other ways. For instance, the traffic volume per aggregate could be useful in planning the required capacity for each QoS class, or the network in general. In fact, given a set of prices per class, the measured volumes, and a set of utilities per application class, one could create the mapping from the application class C_j to QoS class Q_k through an optimization process using traffic data, rather than as a set of fixed rules.

Once monitors are installed, there is no reason one could not use them in an on-going manner. One could continue to make measurements (as above) of the statistics per aggregate, and if something changes significantly, then one could change the rules used. An example might be the introduction of a new application with different QoS characteristics, requiring a rule update. It is not, however, envisioned that these updates would occur often.

4. COS CLASSIFICATION

We next present the different components in our classification approach: identifying different application classes, selecting candidate classification features, and finally specific classification methods.

4.1 Class Definitions

In practice, service differentiation mechanisms like diffserv today only allow a relatively small number of application classes. For our initial study, we focus on the following 4 broad application classes, commonly found in corporate networks.

- *Interactive:* The interactive class contains traffic which is required by a user to perform multiple real-time interactions with a remote system. This class includes such applications as remote login sessions or an interactive Web interface.
- *Bulk data transfer:* The bulk data transfer class contains traffic which is required to transfer large data volumes over the network without any realtime constraints. This class includes applications such as FTP, software updates, and music or video downloads (say through an application such as Kazaa).
- *Streaming:* The streaming class contains multimedia traffic with realtime constraints. This class includes such applications as streaming and video conferencing.
- *Transactional:* The transactional class contains traffic which is used in a small number of request response pairs which can be combined to represent a transaction. DNS, and Oracle transactions belong to this class.

The choices were motivated by the need to select a small number of classes that would be simple, intuitive and still adequately represent the different QoS requirements of commonly used applications. We view these 4 classes as a starting point, with the actual choice of application classes being a topic for research.

To characterize each application class we need a reference data set for each class from which we can extract a set of *representative* features. Acquiring such a reference data set is made difficult by precisely the problems described in Section 2. Selecting the network traffic based on port numbers may not yield reliable statistics that are representative of any particular class, however, to classify them otherwise requires a reference data set. To break this circular dependency we selected some applications per class, which based on their typical use, have a low likelihood of being contaminated by traffic belonging to another application class. In particular we focus on applications which:

- are clearly within one class (to avoid mixing the statistics from two classes);
- are widely used, so as to assure we get a good data-set;
- have server ports in the well known port range to reduce the chance of mis-use of these ports.

These reference applications will then be used to estimate a number of statistics, from which we will select features for use in CoS categorization. Based on the criterion established in the previous section, the reference applications selected for each application class are:

- *Interactive:* Telnet,
- *Bulk data:* FTP-data, Kazaa,
- *Streaming:* RealMedia streaming,
- *Transactional:* DNS, HTTPS.

We include HTTPS in the transactional class, because a large proportion of HTTPS interactions involve users filling out a form, for instance to conduct secure credit card purchases over the WWW. In our current study, we do not use web traffic to train the classification. However we do include WWW traffic statistics (as captured using port 80) in some plots as an example application, as it provides some interesting intuition into the results. Also, although Kazaa does not have registered ports, in particular data sets we have reliable ways of identifying Kazaa traffic (as describe below).

4.2 Candidate Features

The list of possible features one could consider is very large. We can broadly classify these into five categories:

1. Simple packet-level features such as mean packet size and various moments such as variance, RMS (root mean square) size, etc., are simple to compute, and can be gleaned directly from packet-level information. They offer a characterization of the application that is independent of the notion of flows, connections or other higher level aggregations. Another advantage is that packet-level sampling, which is widely used in network data collection, has little impact on these statistics. Other statistics that can be derived from simple packet data are time series, from which we could derive a number of features, for instance relating to correlations over time (e.g. parameters of long-range dependence such as the Hurst parameter). An example of this type of classification can be seen in [23], where the authors use time-of-day traffic profiles to categorize web sites.

2. Flow-level statistics are summary statistics at the grain of network flows. A *flow* is defined to be an unidirectional sequence

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.