An Approach to Comprehending and Detecting Networked Applications through Analogy

Maxim Shevertalov, Edward Stehle, Chris Rorres, Spiros Mancoridis, and Moshe Kam
Department of Computer Science
Drexel University
Philadelphia, PA 19104
{ms333,eds23,crorres,spiros}@cs.drexel.edu, and kam@minerva.ece.drexel.edu

Abstract

Distributed applications rely on packet-switched networks to connect their various elements. This paper describes a technique that can help software engineers and network administrators characterize and detect unfamiliar networked applications by matching them to a single, or a combination of several, analogous and familiar networked application(s). This matching is based on the size distribution of the packets sent and received by the application undergoing scrutiny. It can be used by network administrators to gain a greater understanding of the network they are administering.

1. Introduction

Distributed software systems generally rely on packet-switched networks to connect their various elements. For example, web browsing and e-mail systems transfer text, images, audio, and video content as well as control and protocol establishing data by sending packets across the Internet and edge networks [1] [2] [3] [4]. Similarly, network conferencing software transfers varying combinations of video, voice, and text between clients using packet-switched networks. This work presents a technique to assist in the identification and comprehension of unfamiliar networked applications by drawing analogies to previously studied (*i.e.,* familiar) networked applications.

After observing the packets sent and received by an unfamiliar networked application, our technique generates a composition of familiar applications that have analogous network traffic patterns to the application under scrutiny. For instance, our technique may characterize an unfamiliar video conferencing application as having a 40% resemblance to a specific familiar streaming video application and 60% of a resemblance to a familiar web browser. This knowledge will allow system administrators to develop a better understanding of exactly the type of services their networks provide. It will allow them to further configure the network to help these services and improve the over all user experience.

Our technique analyzes networked software systems by observing the size distribution of the packets sent and received. The aim is to discover which networked applications, or which combination of networked applications, are most analogous to the software undergoing scrutiny. No a priori knowledge of the application is required to use our technique. To date, we have analyzed five networked applications whose features cover a broad spectrum of the capabilities that are commonly associated with networked applications. We will refer to this set of analyzed applications as the canonical set of applications. To date, this set includes the following:

1. Streaming audio: GnomeMeeting (G711 codec)
2. Streaming video: GnomeMeeting
3. Text messaging: Jabber
4. Web browsing/File transfer: Firefox and gFTP
5. E-mail: Unix Mail

We suspect that a significant fraction of existing networked applications can be represented as some combination of the five applications in the canonical set. For example, a software system such as Internet Explorer may be used to stream video, browse the Internet, and check e-mail simultaneously and so can be considered analogous to a combination of the second, fourth, and fifth applications in the canonical set.

The rest of this paper is organized as follows: Section 2 discusses previous work in the areas of packet-size distributions and the comprehension of distributed systems; Section 3 presents our software characterization technique and the results of simulated composed network flows; Section 4 describes a case study where applications are analyzed and our discoveries and conclusions about the applicability of our approach are presented; Section 5 draws conclusions and presents plans for future work.

2. Previous Work

2.1. Packet-size Distributions

Previous research on packet-size distributions focused on application detection [5] [6] and suggests that a networked application can be identified by its packet-size distribution pattern. It further suggests that User Data Protocol (UDP) applications are more identifiable in this way than their Transmission Control Protocol (TCP) counterparts. Our results confirm these conclusions.

Past approaches to application detection by packet-size distribution begin by placing each packet, in an observed flow of packets, into an appropriate interval, or bucket, based on the packets size measured in bits. For example, Trivedi *et al.* used a distribution that splits the networked application's traffic flow into fifty buckets of varying widths [6]. From the packet-size distributions defined by these fifty buckets they were able to achieve fairly successful application detection. We applied the technique of Trivedi *et al.* to a variety of applications, and found that we could improve their results by attaching meta-data to the distributions. For example, we observed noticeable improvement in application detection when we added a meta bucket that recorded the number of buckets which contain no packets.

In this study we used five different applications. Fractional histograms of the packet-size distributions of these five applications are shown in Figure 1. The streaming audio and streaming video systems rely heavily on UDP, while the remaining applications use TCP. As can be seen in Figure 1, the histograms for UDP applications vary significantly from the histograms of TCP applications. Note that a single histogram is used to represent both web browsing and file transfer. In our experiments we found a large variance between the different uses of file transfer, especially in cases of a single large file, for example a DVD image, and many smaller files, for example files containing small text strings. The histogram distributions, or centroids, for each of those cases were very similar to the centroid for web browsing, therefore we chose to group them together.

We wish to characterize a new application by providing analogies to known applications. We are not concerned with how the application connects to the network, but with what network traffic patterns the application enacts on that network.

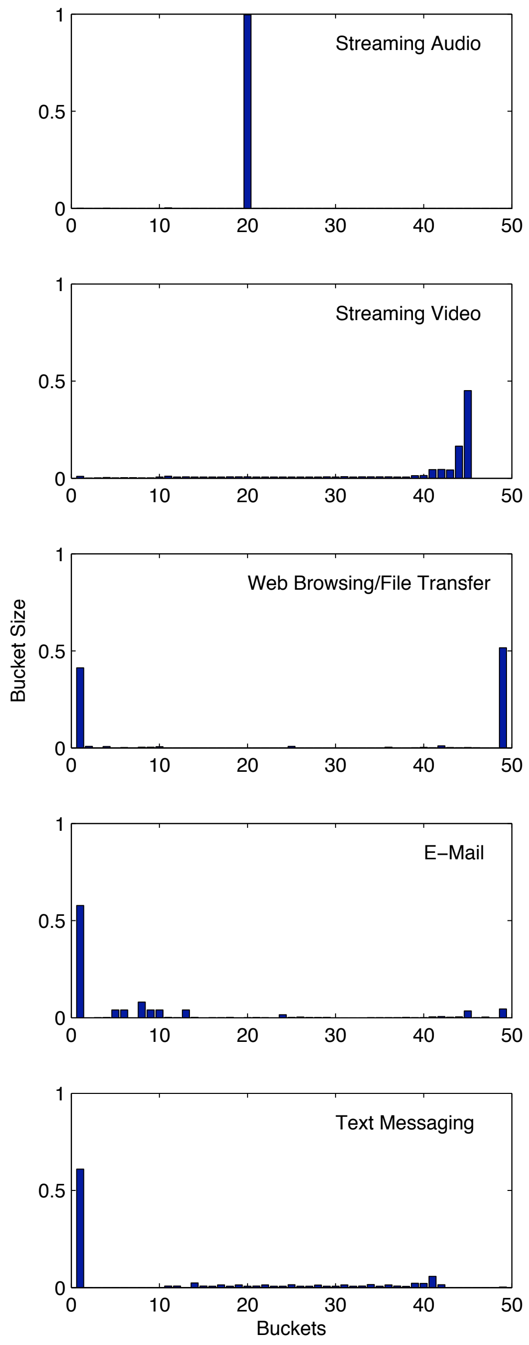

Figure 1: Centroid Application Histograms

2.2. Comprehension of Distributed Systems

Previous work by DiLucca *et al.* [7] [8] concentrates on the comprehension of Web Applications (WAs). DiLucca *et al.* describe a number of challenges that face anyone trying to comprehend a WA, or more generally, a distributed application. They use the idea of categorizing WAs by features, and employ a cluster-based approach to discover them.

There have also been a number of other dynamic techniques for analyzing distributed systems. For example Moe and Carr [9] discuss a way to analyze these systems using trace data. They, like DiLucca *et al.*, are also concerned with the actual, rather than the intended, behavior of the system.

Another interesting form of analysis was performed by Souder *et al*, who developed a method to correlate dynamic method invocations between distributed objects [10]. This approach allowed them to create call sequence diagrams for distributed systems. This work, and previous dynamic analysis techniques, focus more on providing developers with a detailed understanding of what their system does.

Brownlee and Claffy [11] discuss Internet traffic patterns. They classify Internet flows as either "mice" or "elephants", and either "fireflies" or "tortoises". Mice are described as streams of small packets, while elephants contain larger chunks of data; fireflies are streams that stay connected for less than fifteen minutes, while tortoises can be connected for much longer periods. Brownlee and Claffy then go on to describe a case study involving two universities; one that imposes limits and charges departments for bandwidth, and another that does not impose limits and charges. They find that the usage patterns are very different for the two universities. Additionally, they are able to correlate these patterns to the types of applications used by the students and faculty of the studied universities.

3. Technique

In order to characterize an unfamiliar networked application in terms of canonical applications, we generated representations of the packet-size distributions of our canonical applications (Figure 1). These representations were generated from applications in the following way: for each canonical application we recorded 40 samples consisting of the bit lengths of 1000 consecutive packets sent by the application. We then formed a fractional histogram of each application in which the contents of each bin was the fraction of the 1000 packets falling into 50 bins using bin widths as provided by Trivedi *et al.* [6]. Finally, we formed a mean histogram for each application by taking the means of the contents of the fifty bins over the forty samples.

We will refer to this final mean histogram as the *centroid histogram* of the canonical application and represent the centroid histogram of the i-th canonical application mathematically by a 50-dimensional column vector $\mathbf{v}^{(i)}$ whose j-th entry is the entry in the j-th bin of the centroid. Note that the entries of $\mathbf{v}^{(i)}$ are nonnegative numbers whose sum is equal to one.

Next, let $\mathbf{v}$ be the fractional histogram of a sample of 1000 consecutive packets of an unfamiliar application. Specifically, we wish to express $\mathbf{v}$ as a convex combination of the five centroid histograms of the canonical applications. That is, we wish to find nonnegative numbers $c_1, c_2, ..., c_5$, which sum up to one, such that:

$$\mathbf{v} = c_1\mathbf{v}^{(1)} + c_2\mathbf{v}^{(2)} + \cdots + c_5\mathbf{v}^{(5)} \qquad (1)$$

Each c_i represents the fraction of the i-th canonical application that is, in some way, present in the unfamiliar application. We shall assume that none of the centroid histograms is a convex combination of the others, otherwise we cannot generally express a given application as a unique combination of the canonical applications. This assumption follows from our belief that networked applications having similar histograms should be grouped together into a single category because they have similar impact on the rest of the system (e.g., the grouping of the web browsing and file-transfer canonical applications). Creating these categories will become more relevant as our repertoire of canonical applications grows.

Geometrically, as $c_1, c_2, ..., c_5$ run through all possible non-negative values that sum up to one, the right-hand side of Eq. (1) sweeps out the *convex hull* of $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(5)}$, considered as points in 50-dimensional Euclidean space. The *convex hull* of a set of points is the intersection of all convex sets that contain the points. If the points are finite in number, then their convex hull is called a *convex polytope*. For example, the convex polytope of two points is the line segment connecting the two points; for three points it is the triangle that has the three points as its vertices; and for four points it is the tetrahedron that has the four points as its vertices.

Let us denote by S the convex polytope in a 50-dimensional Euclidean space generated by our five canonical histogram vectors $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(5)}$. Our assumption that none of them is a convex combination of the others means that S is also a *simplex*, specifically a *4-simplex*. In general, a *p-simplex* is the convex hull of $p + 1$ points, none of which is a convex combination of the others. The p points themselves are called the *vertices* of the simplex. For example, a 1-simplex is a line segment whose endpoints are its vertices; a 2-simplex is a triangle, and a 3-simplex is a tetrahedron.

The coefficients $c_1, c_2, ..., c_5$ in Eq. (1) are referred to as the *barycentric coordinates* of points in S. Each point in S has unique barycentric coordinates and, conversely, each set of barycentric coordinates defines a unique point in S. In our case, the barycentric coordinates of the histogram of an unfamiliar application $\mathbf{v}$ define the fractions of each of the five canonical applications that we regard as present in the application.

If $\mathbf{v}$ does not lie in S, then Eq. (1) has no solution. This

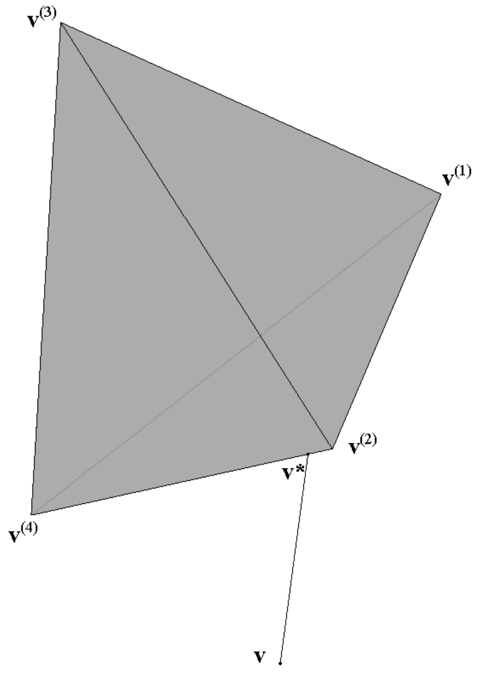

Figure 2: The 3-simplex (tetrahedron) generated by the four vectors $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}, \mathbf{v}^{(4)}$ and the vector $\mathbf{v}^*$ within it that is closest to the vector $\mathbf{v}$ that is outside of it.

is the usual case since Eq. (1) represents a linear system of fifty equations in five unknowns and is consequently very overdetermined. We shall instead seek a vector $\mathbf{v}^*$ in S that is closest to $\mathbf{v}$ in some sense and take its barycentric coordinates within S as the approximate fractions of the canonical applications in $\mathbf{v}$. We interpreted $\mathbf{v}^*$ to be the closest vector to $\mathbf{v}$ in S if its Euclidean distance to $\mathbf{v}$ was as small as possible. To be precise, if $||\mathbf{w}||$ denotes the Euclidean length of the vector $\mathbf{w}$ (the square root of the sum of the squares of its entries), then the problem we posed was the following:

Problem 1: Given the six 50-dimensional vectors $\mathbf{v}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(5)}$, find numbers $c_1, c_2, ..., c_5$ that minimize $||\mathbf{v} - c_1\mathbf{v}^{(1)} - c_2\mathbf{v}^{(2)} - \cdots - c_5\mathbf{v}^{(5)}||$ subject to the conditions that $c_1 \geq 0$, $c_2 \geq 0$, ..., $c_5 \geq 0$ and $c_1 + c_2 + \cdots + c_5 = 1$.

The solution of this problem provides the barycentric coordinates of the point $\mathbf{v}^*$ in S that is closest to $\mathbf{v}$. If $\mathbf{v}$ lies in S, then the quantity $\| \mathbf{v} - c_1\mathbf{v}^{(1)} - c_2\mathbf{v}^{(2)} - \cdots - c_5\mathbf{v}^{(5)} \|$, called the *least-squares error*, can be reduced to zero and the corresponding vector $\mathbf{v}^*$ is $\mathbf{v}$ and the barycentric coordinates we obtain are the exact solution to Eq. (1).

Figure 2 illustrates the idea behind this problem, using four canonical applications rather than five. The figure shows the 3-simplex generated by the four vectors $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$, $\mathbf{v}^{(3)}$, $\mathbf{v}^{(4)}$ (more precisely, it shows the projection of the 3-simplex in 50-dimensional space onto the 2-dimensional plane of the paper) and the vector $\mathbf{v}^*$ within it that is closest to the given vector $\mathbf{v}$. In this particular example, $\mathbf{v}^*$ lies on one of the six edges of the simplex (that is, one of the six line segments connecting the four vertices), but it could also have been situated at one the four vertices of the simplex, one of its four facets, or even within what we perceive as its "interior". The simplexes generated by all nonempty subsets of the vertices of a given simplex are called the *faces* of the simplex and a p-simplex has precisely $2^{(p+1)} - 1$ faces (e.g., the 3-simplex in Figure 2 has 15 faces).

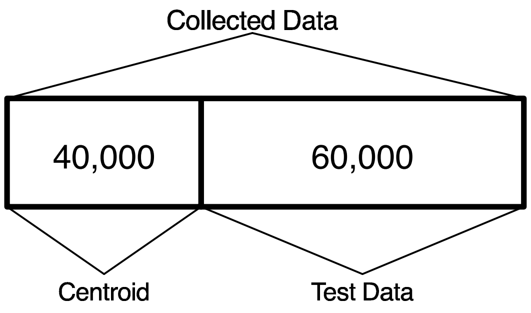

Figure 3: Data Composition

	Vo	Vi	WF	E	T
Vo	**0.02**	1.98	2.00	1.99	1.98
Vi		**0.24**	1.86	1.77	1.44
WF			**1.17**	1.00	1.09
E				**0.97**	0.76
T					**1.02**

Table 1: Cluster Distances: (Vo)IP, (Vi)deo, (W)eb browsing/(F)ile transfer, (E)mail, (T)ext Messaging

Standard theorems in optimization theory guarantee that Problem 1 has a unique solution. In addition, standard algorithms exist to solve it, usually by searching recursively for candidate solutions that lie on the faces of the simplex. Minimizing the Euclidean distance to find an approximate solution is also the appropriate criterion in a regression analysis if we assume that the contents of the bins in our histograms are normal random variables and we seek to find the maximum likelihood estimates for $c_1, c_2, ..., c_5$; that is, the estimates we would most often find by taking many samples of the application histogram $\mathbf{v}$.

4. Canonical Applications Distances

In order to verify our approach we first generated the centroid histograms of five canonical applications from data obtained by running representative canonical applications in a laboratory. We gathered a total of 100,000 consecutive packets (Figure 3) for each canonical application and used the first 40,000 packets to create that application's centroid histogram by dividing the packets into forty 1000-packet samples and taking the mean of the forty resulting fractional histograms.

Table 1 illustrates the distances between sample data from each of the canonical applications. For this table we divided our 100,000-packet data for each of the five canonical applications into 100 samples of 1,000 packets each. The bold-face values down the main diagonal are the diameters of the five canonical applications, where the diameter of an application is defined as the largest distance between any two sample histograms of that application. Each entry in the top section of the grid represents the distance between the centroids of the two canonical applications corresponding to the row and column of the entry. From Table 1 we can see why audio and video conferencing are easily detected. They are both very far away from any other canonical applications, and have small diameters. The other three canonical applications are more difficult to detect. All of these applications communicate over TCP. Their diameters are bigger than those of the first two applications. Also the TCP applications are much closer to each other.

5. Error Analysis

After using the first 40,000 packets to generate each canonical application's centroid, we used the remaining 60,000 packets to form sixty sample histograms of a synthetic application consisting of a combination of the canonical applications. We did this by taking a specific convex combination of the 60,000 packets of each of the five canonical applications, thereby creating a single stream of 60,000 packets for our synthetic application. We then divided the packet stream into sixty 1000-packet streams and constructed their sixty sample histograms for the synthetic application.

Our objective in this test was to see if we can recover the coefficients of the specific convex combination of the canonical applications used to form the synthetic application. We also wanted to estimate the variation in the recovered coefficients over the sixty samples we generated for each specific convex combination.

Figure 4 summarizes the errors in our recovered coefficients for various convex combinations of the five canonical applications. Each of the five graphs is associated with one of the canonical applications, where the horizontal axis specifies the percentage of that application in the original convex combination. This percentage varied from 0% to 100% in 10% increments. The vertical scale indicates the error, in percent, from the original composition as determined by our algorithm. The percentage error plotted is the average error in which, for each fixed percentage value on the horizontal axis, we varied the percentage of the compo-

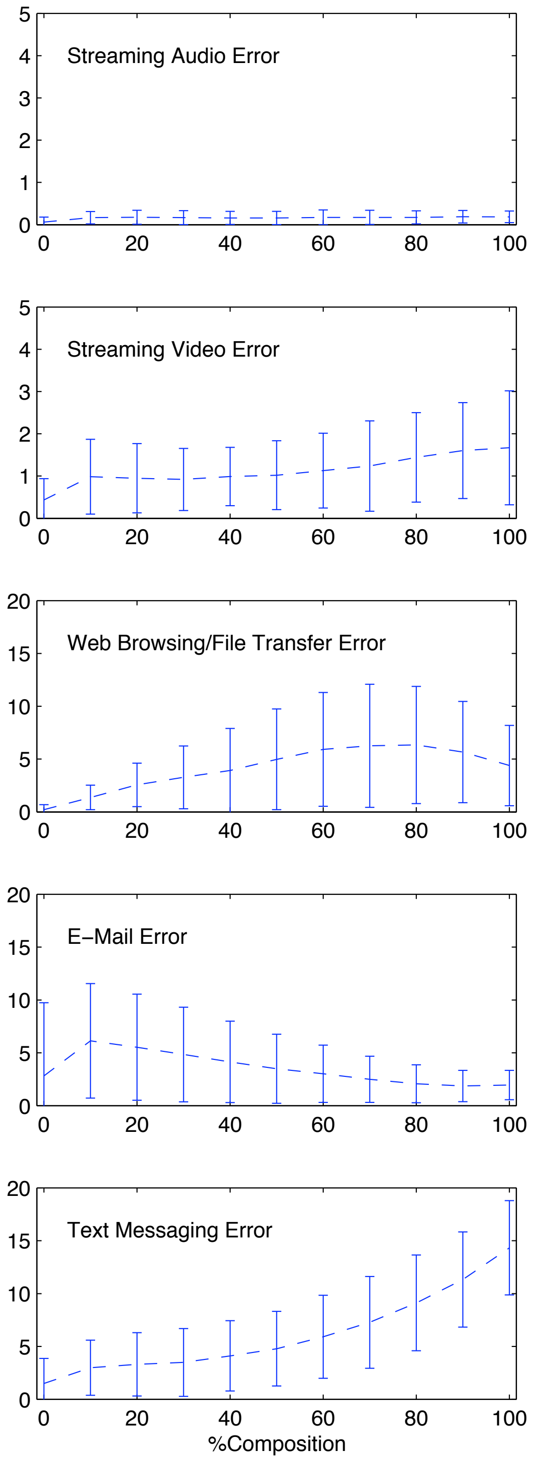

Figure 4: Error in Results

Application	JWChat
Audio Streaming	2.0%
Video Streaming	13.5%
Web Browsing/File Transfer	5.8%
Email	0.0%
Text Messaging	78.7%

Table 2: JWChat Results

nents of the five applications over all possible values in 10% increments, and for each of the computed sixty samples.

For example, consider a hypothetical application whose histogram is formed by taking 20% of the VoIP centroid histogram and all percentage combinations of the remaining four centroid histograms that add up to 80% (taken from 0%, 10%, ... , 80%), a total of 165 possibilities. For each of these 165 combinations we computed the coefficient of the VoIP class using all 60 samples, giving a total of 9900 values of this coefficient. Next we computed the fractional deviation of each of these 9900 values from the theoretical value of 20%. Finally we plotted the average of these 9900 fractional deviations on the graph at the 20% horizontal coordinate. The vertical line about that value is the one-standard deviation interval of the 9900 errors, representing the interval in which about 68% of the errors were situated.

As can be seen from Figure 4, our results for the canonical VoIP application and the canonical Video application were very good, consistent with the findings of Parish *et al.* [5] and Trivedi *et al.* [6], who describe UDP detection as being more accurate than TCP detection. The results for the remaining three applications were not quite as good.

6. Case Studies

In addition to synthetic applications, we applied our approach to a number of real-world applications,

6.1. JWChat

One of the applications we used in this study was JWChat [12]. This is a Jabber-based, web-chat client which uses standard HTML and AJAX[13] to allow users to chat inside their web browsers.

In our tests we had two people communicate using JWChat and captured 1000 successive packets from which we constructed the histogram of the JWChat application (Figure 5). Table 2 gives the fractions of the five canonical applications that our technique provided and Figure 6 is the histogram that contains these precise fractions. Geometrically, the histogram in Figure 5 corresponds to the vector **v** outside of the simplex in Figure 2 and the histogram in Figure 6 corresponds to the vector $\mathbf{v}^*$ in the simplex that is closest to **v**.

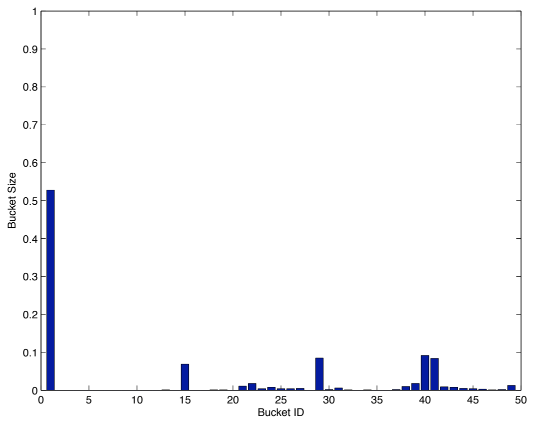

Figure 5: JWChat Histogram

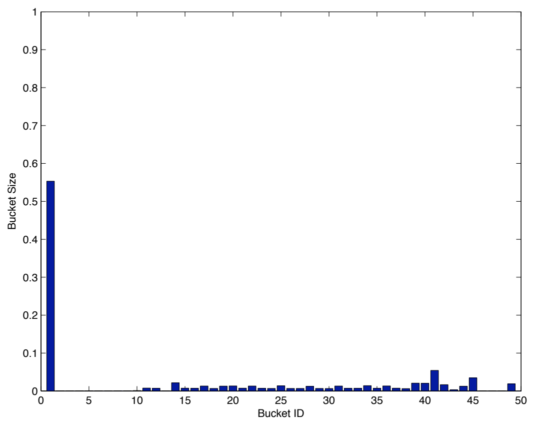

Figure 6: Combination of the five canonical centroid histograms closest to the JWChat histogram

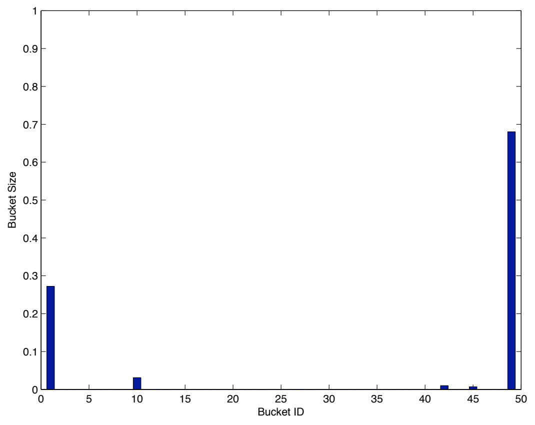

Figure 7: WF Histogram: Single File

Our method detected the overwhelming presence of the text messaging centroid (78.7%) which indicates the JWChat generates network traffic analogously to the text messaging canonical application. Our method also detected a little of the web browsing and file transfer centroid (5.8%), which was expected. However it also detected some of the streaming video centroid (13.5%), even though no video conferencing appears in JWChat.

There are a number of conclusions that can be drawn from this analysis. Even though the application is really a proxy for Jabber messages, we can still detect a significant amount of chat in the system. This would not be detectable with an approach like deep-packet analysis or port matching, since all the data is hidden by the HTTP protocol.

Another conclusion stems from the detected presence of streaming video in this system. It is possible that a Jabber HTTP proxy has similarities to streaming video. When looking at the centroids for both streaming video and text messaging they appear very similar. Both have a number of packets in buckets 10-40. The only other canonical application with a similar signature is e-mail. However e-mail exhibits major differences by having a large number of very small packets, buckets 0-10, as well as a number of very large packets, buckets 40-50. These disparities help us understand why our algorithm chose nothing from the e-mail application, while it chose a significant portion from the streaming video application.

6.2. Windows Filesharing

We applied our technique to Windows Filesharing. We expected Windows Filesharing to behave differently under various circumstances, therefore, we applied our techniques to two different use cases. For both cases we collected 1000 packets to analyze. In the first case we shared a single large file, while in the second case we shared a number of small files.

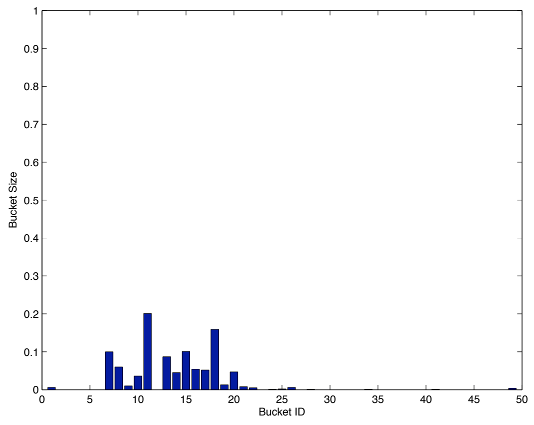

Figure 8: WF Histogram: Multiple Files

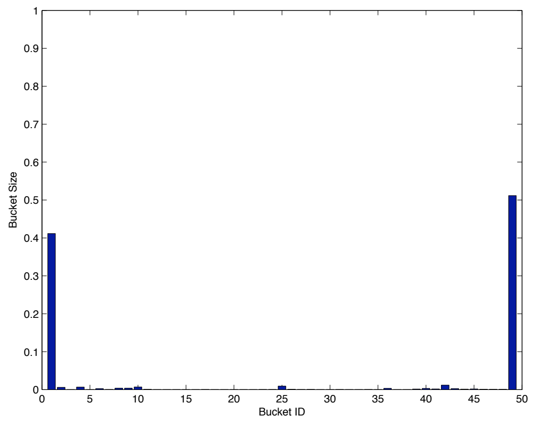

Figure 9: Combination of the five canonical centroid histograms closest to the WF:Single File histogram

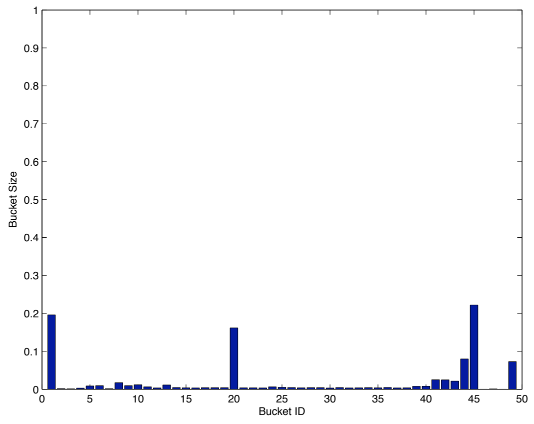

Figure 10: Combination of the five canonical centroid histograms closest to the WF:Multiple Files histogram

Application	WF: Single	WF: Many
Audio Streaming	0.0%	15.9%
Video Streaming	0.0%	46.7%
Web Browsing/File Transfer	100.0%	12.5%
Email	0.0%	19.2%
Text Messaging	0.0%	4.7%

Table 3: Windows Filesharing Results

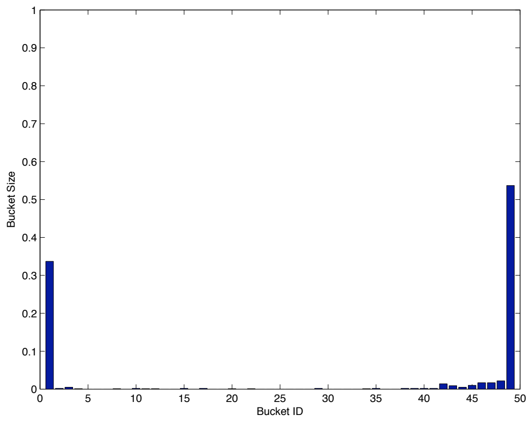

Figure 11: iSpQ Histogram: Video

The histograms generated for each of the use cases can be seen in Figures 7 and 8. Table 3 contains the fractions of the five canonical applications that our technique provided, and Figures 9 and 10 are the histograms containing these precise fractions.

Before performing these tests we expected both studied cases to overwhelmingly match the web browsing and file transfer centroid. While this expectation held in the first case, of a single large file, it did not hold in the second scenario. In the first scenario the actual and generated histograms are relatively close together. However in the second case the distance between the actual and generated histograms is larger. This suggests that additional canonical applications may be needed to classify unfamiliar applications better.

6.3. iSpQ VideoChat

We also applied our technique to a video-chatting application iSpQ[14]. This application allows multiple users to chat using either text messaging or video conferencing. Unlike other video chat clients that use UDP, iSpQ uses TCP to send video information. Because of the dual mode of iSpQ we decided to apply our technique to video and text chat features separately. We collected 1,000 packets for each feature and the generated histograms for iSpQ video and chat can be seen in Figures 11 and 12 respectively. The results provided by our technique are summarized in Table 4

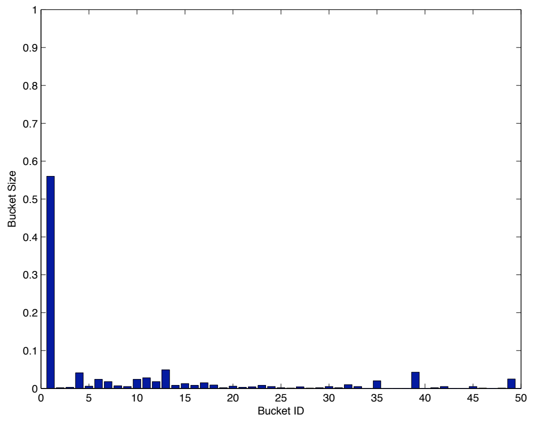

Figure 12: iSpQ Histogram: Chat

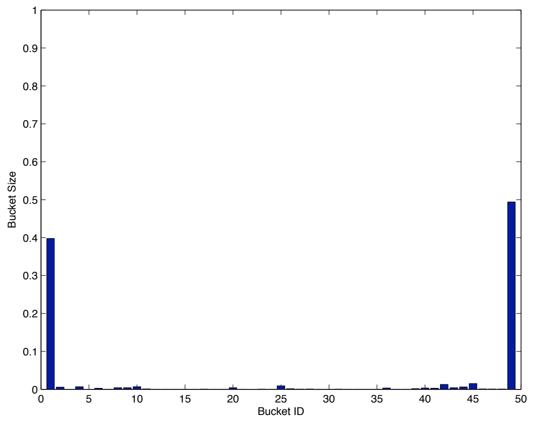

Figure 13: Combination of the five canonical centroid histograms closest to the iSpQ:Video histogram

and Figures 13 and 14 are histograms containing these precise fractions.

These results appear to support the conclusion that applications with similar features can have different behaviors and impacts on the network. In this case Video conferencing is more analogous to web browsing and file transfer. iSpQ's video transfer technique is optimized toward image quality, rather than frame rate. Using TCP, iSpQ guaranties that every frame sent will be delivered. However if more loss is introduced into the system, fewer frames will actually be sent to the users.

By identifying that iSpQ's video conferencing feature is analogous to the web browsing and file transfer category we observe that iSpQ attempts to push large, Maximum Transmission Unit (MTU) sized, packets across the network. In addition, iSpQ also sends an almost equally large number of small acknowledgment packets. While it may seem surprising at first that iSpQ is analogous to the web browsing and file transfer category, the fact that the actual and gener-

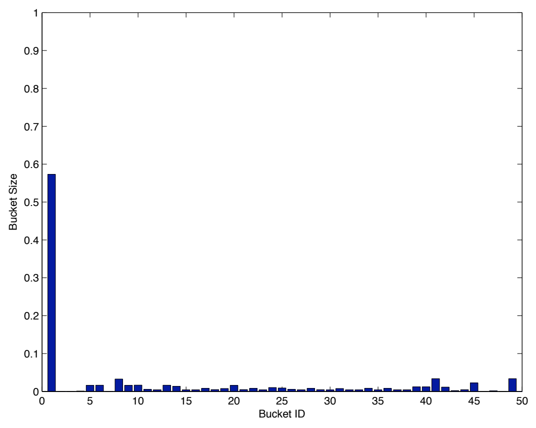

Figure 14: Combination of the five canonical centroid histograms closest to the iSpQ:Chat histogram

Application	Video	Chat
Audio Streaming	0.4%	1.2%
Video Streaming	3.0%	1.8%
Web Browsing/File Transfer	96.6%	2.7%
Email	0.0%	40.3%
Text Messaging	0.0%	54.0%

Table 4: iSpQ Results

ated histograms are close to each other, signifies that these results are fairly accurate.

The text chat mode of iSpQ also produced acceptable results. Our technique detected a significant presence of chat in the system. However it also found a significant presence of e-mail. This can be explained in a number of ways. First, the fact that the distance between the actual and generated histograms is small, signifies that our technique is moderately accurate. However, one should also notice the presence of packets in buckets 10-30. These are also present in other applications with non-intuitive classification. This suggests additional canonical applications may be needed to classify unfamiliar applications.

6.4. gMail

Finally we applied this technique to gMail[15], a web based e-mail client. This application allows users to send and receive e-mail messages. We collected 1,000 packets worth of 'send' messages and the generated histogram can be seen in Figure 15. The results provided by our technique are summarized in Table 5 and Figure 16, which is a histogram containing these precise fractions.

Due to the closeness of the actual and generated histograms, these results appear to be fairly accurate. We did expect to find a large quantity of HTTP traffic considering that gMail loads a number of pages in order to send a mes-

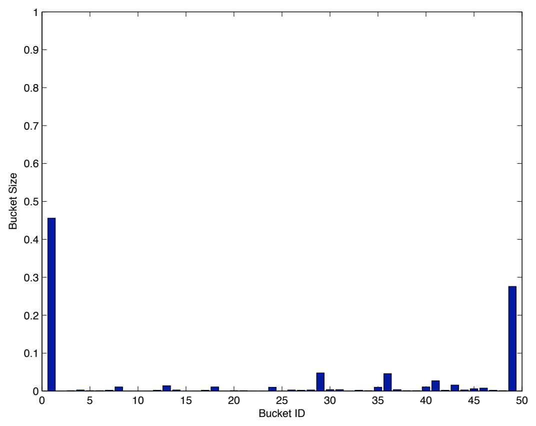

Figure 15: gMail Centroid

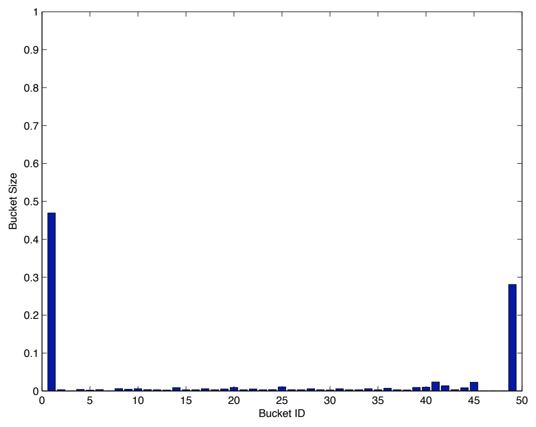

Figure 16: Combination of the five canonical centroid histograms closest to the gMail histogram

Application	Video
Audio Streaming	0.6%
Video Streaming	4.5%
Web Browsing/File Transfer	54.3%
Email	5.0%
Text Messaging	35.6%

Table 5: gMail Results

sage. What we did not expect was the large quantity of chat type packets. The text messaging canonical application captures the small quantity of packets appearing in bucket 30-40.

This analysis allows to draw a number of conclusions about gMail. First we observe that most of the traffic produced by gMail is web browsing and file transfer type traffic. Second, about a third of the traffic is attributed to text messaging. To a developer this observation signifies that roughly half of the packets will be small acknowledgment-type packets. However, actual data packets will be larger, with most being of MTU size.

7. Conclusions and Future Work

Distributed systems are not only difficult to create, but also difficult to analyze. It is possible to create two systems that perform identical functions, yet generate very different network packet distributions. For example, one can stream video using TCP or UDP packets and accomplish identical results from the end-users' perspective, but the effects of the applications on the network in which they operate can be drastically different.

Our approach to characterizing network applications is through analogy to previously studied canonical applications. We are able to decouple a single distributed system into a composition of canonical systems. Even though our repertoire of canonical applications is fairly limited, we were able to get good results: within all our tests of different combinations we were able to detect the correct makeup with relatively small error, especially for UDP applications.

In the next phase of this project we would like to compose a larger repertoire of canonical applications and group them into a number of canonical application categories. This grouping would likely be achieved using a clustering approach.

Also, our analysis did not consider the temporal pattern of the packet sizes; that is, we did not consider the order of the sizes of the packets within our 1000-packet sample. It may be that this pattern contains information that will allow us to classify unknown applications in terms of canonical applications more accurately.

References

[1] David Bargeron, Anoop Gupta, Jonathan Grudin, and Elizabeth Sanocki. Annotations for streaming video on the web: System design and usage studies. In *International World Wide Web Conference*, 1999.

[2] Zhigang Chen, See-Mong Tan, Roy H. Campbell, and Yongcheng Li. Real time video and audio in the world wide web. In *International World Wide Web Conference*, 1995.

[3] H. Lieberman, N. van Dyke, and A. Vivacqua. Let's browse: a collaborative browsing agent. *Knowledge-Based Systems*, 12(8):427–431, December 1999.

[4] Jim Conallen. Modeling web application architectures with uml. *Commun. ACM*, 42(10):63–70, 1999.

[5] D.J. Parish, K. Bharadia, A. Larkum, I.W. Phillips, and M.A. Oliver. Using packet size distributions to identify real-time networked applications. In *Communications, IEE Proceedings*, volume 150, pages 221–227, August 2003.

[6] C Trivedi, HJ Trussell, A Nilsson, and M Chow. Implicit traffic classification for service differentiation. *ITC Specialist seminar*, 2002.

[7] G.A. Di Lucca, A.R. Fasolino, F. Pace, P. Tramontana, and U. De Carlini. Comprehending web applications by a clustering based approach. In *International Workshop on Program Comprehension*, pages 261–270, 2002.

[8] G.A. Di Lucca, A.R. Fasolino, and P. Tramontana. Towards a better comprehensibility of web applications: lessons learned from reverse engineering experiments. In *Web Site Evolution*, pages 33–42, 2002.

[9] J. Moc and D.A. Carr. Understanding distributed systems via execution trace data. In *International Workshop on Program Comprehension*, pages 60–67, 2001.

[10] T.S. Souder, S. Mancoridis, and M. Salah. Form: A framework for creating views of program executions. In *Proceedings of the 2001 International Conference of Software Maintenance*, 2001.

[11] N. Brownlee and K.C. Claffy. Understanding internet traffic streams- dragonflies and tortoises. *Communications Magazine, IEEE*, 40(10):110–117, October 2002.

[12] Jwchat - jabber web chat. `http://jwchat.sourceforge.net/`.

[13] Adaptive path ajax: A new approach to web applications. `http://www.adaptivepath.com/publications/essays/archives/000385.php`.

[14] Video chat and webcam community from ispq videochat. `http://www.ispq.com/`.

[15] Welcome to gmail. `http://mail.google.com/`.