SECURITY AND COMMUNICATION NETWORKS

Security Comm. Networks **00**: 1–25 (2008)

Published online in Wiley InterScience

(www.interscience.wiley.com) DOI: 10.1002/sec.0000

Protocol Oblivious Classification of Multimedia Traffic

<Jieyan Fan, Dapeng Wu[1]*, Antonio Nucci, Ram Keralapura and Lixin Gao>

[1]*University of Florida, Dept. of Electrical & Computer Engineering, P.O.Box 116130, Gainesville, FL 32611*

Summary

Voice and video over IP are becoming increasingly popular and represent the largest source of profits as consumer interest in online voice and video services increases, and as broadband deployments proliferate. In order to tap the potential profits that VoIP and IPTV offer, carrier networks have to efficiently and accurately manage and track the delivery of IP services. The traditional approach of using port numbers to classify traffic is infeasible due to the usage of dynamic port number. In this paper, we focus on a statistical pattern classification technique to identify multimedia traffic. Based on the intuitions that voice and video data streams show strong regularities in the packet inter-arrival times and the associated packet sizes when combined together in one single stochastic process, we propose a system, called *VOVClassifier* , for voice and video traffic classification. *VOVClassifier* is an automated self-learning system that classifies traffic data by extracting features from frequency domain using Power Spectral Density analysis and grouping features using Subspace Decomposition. We applied *VOVClassifier* to real packet traces collected from different network scenarios. Results demonstrate the effectiveness and robustness of our approach that is capable of achieving a detection rate of up to 100% for voice and 96.5% for video while keeping the false positive rate close to 0%. Copyright © 2008 John Wiley & Sons, Ltd.

KEY WORDS: traffic classification; multimedia; VoIP; IPTV

1. Introduction

Over the past 60 years or so, voice and video services like telephony and television have established

*Correspondence to: University of Florida, Dept. of Electrical & Computer Engineering, P.O.Box 116130, Gainesville, FL 32611. Email: wu@ece.ufl.edu

Prepared using secauth.cls *[Version: 2008/03/18 v1.00]*

themselves as an integral part of everyone's life. Traditionally, voice and video service providers built their own networks to deliver these services to customers. However, tremendous technical advancements in the last decade has revolutionized the mode of delivery of these services. Today, these services are delivered to the users over the Internet and we believe that there are two main reasons for this: (i) Delivering services over the Internet in IP packets is much more economical for voice and video service providers, (ii) The massive penetration of broadband (i.e., higher bandwidth) Internet service has ensured that the quality of voice and video services over the Internet is good enough for everyday use. The feasibility of a more economical alternative for voice and video services attracted many Internet service providers (ISPs) like Comcast, AT&T, and Verizon among several others, to offer these services to end users at a lower cost. However, non-ISPs like Skype, Google, Microsoft, etc. have also started offering these services to customers at extremely competitive prices (and on many occasions for free).

From an ISP's perspective, traffic classification has always been a critical activity for several important management tasks like traffic engineering, network planning and provisioning, security, billing, and quality-of-service (QoS). Given the popularity of voice and video services over the Internet, it has now become all the more important for ISPs to identify voice and video traffic from other service providers for 3 reasons: (i) Voice and video services other than ISP's own service will severely impact its revenues and hence ISPs may wish to shape/block flows from these services, (ii) From a traffic engineering perspective, ISPs may sometimes need to prioritize other more important traffic (e.g. VPN traffic) to ensure the promised QoS is met, than voice and video services from other service providers like Skype, Microsoft, and Google. (iii) From a security perspective, an ISP should always have the capability to accurately identify *all* flows and block all malicious ones.

Service providers like Skype, Microsoft (MSN), and Google (GTalk) have evolved over the last few years from providing isolated services to bundled services. In other words, service providers initially offered only individual services (either voice, video, file transfer, *or* chat) to end-users. However, today, they offer voice, video, file transfer, and chat as one bundled service. This change in service paradigm has resulted in changes to the voice and video traffic/flow characteristics. In general, we categorize all multimedia (i.e., voice and video) flows into two types: (i) *Homogeneous flows:* These are flows where every multimedia flow is either a voice or a video flow. (ii) *Hybrid flows:* These are flows where voice and video streams are bundled with other services such as file transfer and chat. In other words, the same flow at layer-3/layer-4 can now carry multiple streams at the application layer.

Identifying voice and video traffic in both homogeneous and hybrid flows is a very challenging task. Initial approaches for voice detection relied on application payload signatures. For example, in [1] the authors study the problem of identifying voice traffic that use the

DOI: 10.1002/sec

standard H.323 protocol [2]. It identifies voice traffic by first recognizing the TCP setup phase of H.323 using payload signatures, and subsequently analyzing the UDP data to identify the associated RTP stream. Such techniques are simple, efficient, and extremely accurate. However, service providers started using advanced encryption techniques for their services, thus making signature based approach obsolete. Hence, the research community started focusing on developing general classification techniques that rely solely on layer-3/layer-4 information (like packet size and packet inter-arrival time) about traffic flows. These techniques can be broadly classified into two: (i) Techniques that classify multimedia (i.e. voice and video) from the rest of the Internet traffic [3] [4] [5], and (ii) Identifying traffic originating from specific applications (for example, [6] tries to identify Skype voice flows, while [7] tries to identify all Skype flows). These techniques have several drawbacks when dealing with realistic traffic on the Internet: (i) None of these approaches can separate generic homogeneous voice and video flows from each other irrespective of the application that generated them. (ii) When it comes to hybrid flows, there are no known approaches that can effectively identify the existence of hidden voice and/or video streams.

In this paper, we address the above issues and propose a self-learning voice and video traffic classifier called *VOVClassifier* , i.e., Voice Video Stream Identifier, that not only identifies voice and video traffic in both homogeneous and hybrid flows, but also labels these flows with the application that generated these flows. This classifier works in two phases: *offline training* phase and *online detection* phase.

In the offline training phase, a sample set of flows from applications of interest are passed as an input to the *VOVClassifier* . For example, if the application of interest is Skype, then in the training phase we feed the *VOVClassifier* with *homogeneous* Skype voice and video flows. Note that irrespective of whether we are interested in classifying homogeneous or hybrid flows, the input to the *VOVClassifier* during the training phase is always *homogeneous* flows. Similar to other classifiers [3], the *VOVClassifier* also relies on two main characteristics of packets in voice and video flows: *packet size* and *packet inter-arrival time*. However, unlike other approaches that consider such metrics independently from each other, we have developed a novel methodology that extracts the hidden temporal and spatial correlations of these features and studies its regularities in the frequency domain. The approach that we propose in this work first models these features into a two-dimensional stochastic process, and then analyzes the properties of this process using the *Power Spectral Density* (PSD) analysis. This PSD analysis results in application fingerprints that can now be used for accurate classification of flows from the trained application. Note that this fingerprinting mechanism not only identifies voice and video traffic, but further clusters these voice and video flows into specific applications. Such a grouping will help differentiate between voice/video flows from different applications (for example, Skype voice traffic and MSN voice traffic),

DOI: 10.1002/sec

thus enabling ISPs to administer application priorities depending on the service level agreements. In the online detection phase, *VOVClassifier* first computes the PSD fingerprint for the flow, and then compares it to the existing fingerprints to classify and subsequently label the flow as belonging to a particular application.

Our main contributions in this paper are:

- We propose a novel voice and video classifier, called *VOVClassifier* , that is capable of identifying voice and video streams even if these streams are hidden inside bundled application sessions. This classifier works in two phases: offline training phase and online detection phase. In the training phase, we extract fingerprints for voice and video flows, and in the detection phase we use these fingerprints to accurately classify and label flows in real-time.
- We propose a novel methodology for extracting the fingerprint of voice and video flows. We first model the packet size and inter-arrival time of packets in a flow as a two dimensional stochastic process, and subsequently use power spectral density analysis to extract the hidden regularities constituting the fingerprint of the flow. We show that these fingerprints are unique for voice and video flows (and also for each application that generates these flows) and can be easily clustered to create a voice and video subspace. These subspaces can be separated by a linear classifier.
- We use real packet traces containing voice, video, and file transfer sessions in Skype, Google Talk, and MSN to comprehensively evaluate our methodology. Our results show that our approach is very effective and extremely robust to noise. In fact, we found that the detection rate for both voice and video flows in *VOVClassifier* was 99.5% with negligible false positive rate when considering only homogeneous flows. When considering hybrid flows, our voice and video detection rates were 99.5% and 95.5% for voice and video traffic while still keeping the false positive rate close to 0.

The rest of the paper is organized as follows. In Section 2 we introduce related research work and present the background for our current work. Section 3 presents some key intuitions that constitute the essence of our methodology. In Section 4 we present the overall architecture of our system. Section 8 demostrates the effectiveness of our system using real packet traces while Section 9 concludes the paper.

2. Related Work, Background, and Data Description

In this section, we first present related work and then give a brief introduction to voice, video, and file transfer streams. We also describe the voice, video, and file transfer streams that we collect for all our experiments.

2.1. Related Work

Traffic classification has received a lot of attention in the past [3,8–18]. The focus of most of these works has been to identify and classify traffic into categories like web,

DOI: 10.1002/sec

email, bulk transfer, peer-to-peer, and/or multimedia (i.e., voice and video flows) among several others. For example, in [11], the authors propose a novel approach called *BLINC* that looks at the flow attributes at multiple levels (social, functional, and application levels) to classify traffic into different categories. Other works like [3,12,17,18] focus on a identifying a particular type of application class like peer-to-peer, multimedia, chat, etc. Our current work differs from all of above research in several aspects:

- We focus not only on identifying mutimedia flows (similar to other works), but also focus on identifying if the flow contains a voice and/or a video stream. In other words, our aim is to be able to classify any given flow as containing voice or video streams by looking at the layer-3/layer-4 headers in the packets belonging to the flow.
- We emphasize on identifying the presence of voice and video streams in hybrid flows that combine voice and video streams with other applications like file transfer and chat. To the best of our knowledge, we are not aware of any other work that addresses this problem.

2.2. Background: Voice, Video and File Transfer Streams

Voice Stream: A key aspect of most voice streams on the Internet is the interactive behavior of users. For example, a voice-over-IP (VoIP) phone call typically involves two parties communicating with each other in real-time. Hence one of the important parameters that characterizes voice traffic is the inter-packet delay (IPD). Several standard codecs can be used for this voice communication, and each of these codecs specify different IPD values. We list a few of these standard codecs in Table I. VoIP service providers like Skype, Microsoft (or MSN), and Google (or GTalk) give users the ability to configure different codecs depending on the network conditions. However, by default most of these service providers use proprietary codecs whose specifications are not available to analyze. In our experiments, we noticed that Skype and MSN voice traffic use proprietary codecs that could either transmit packets every 20 ms or 30 ms. Although these codecs specify the IPD at the voice transmitter side, the packets that arrive at the receiver do not have this IPD. One of the primary reasons for this is the indeterministic delay (due to router queues, packet paths, etc.), also referred to as jitter, experienced by the packets that traverse different links in the Internet. In order to minimize the impact of jitter on the quality of voice traffic, voice applications typically generate packets that are very small in size. Thus, despite the variations due to jitter, voice streams still exhibit strong regularities in the IAT distribution at the receiver side.

Video Streams: Video applications (i.e., live streaming media) send images from a transmitter to a receiving device by transmitting frames at a constant rate. For example, the H.323 codec tries to dispense frames at constant rate of 30 frames per second. Typically there are two types of frames, *Intra frames (I-frame)* and *Predicted frames (P-frame)*, that are

DOI: 10.1002/sec

transmitted/interleaved in a periodic fashion. Usually, the number of P-frames between two consecutive I-frames is constant. An *I-frame* is a frame coded without reference to any frame except itself and usually contains a wider range of packet sizes depending on the texture of the image to be transmitted. I-frames serve as starting points for a decoder to reconstruct the video stream. A *P-frame* may contain both image data and motion vector displacements and/or combinations of the two. Its decoding requires access to previously decoded I- and P-frames. Typically, each *P-frame* is contained in a small sized packet in the range of 100-200 Bytes. Due to the above properties of video streaming, we expect to still see some regularities in the IAT distribution at the receiver side.

File Transfer Streams: File transfer applications deliver bulk data from a transmitter to a receiver as fast as possible leveraging the network conditions. These applications typically partition a file into equal-sized segments and transfer one segment at a time to the receiver. Therefore, a file transfer flow is likely to be composed by equal-sized packets of large size, except a few packets at the beginning and at the end of the data transfer. From our analysis, we observed that all the applications under investigation show that 90% of packets have size between 1400 and 1500 Bytes (1397 bytes for Skype, 1448 bytes for MSN), while only 10% have size between 50 and 150 Bytes.

2.3. Data Sets

To collect data for our experiments, we setup multiple PCs (1.8 GHz, Pentium 4 with Windows XP) in two different universities to run 3 applications - Skype, MSN, and GTalk. These universities are located in different parts of the North American continent. We generated voice, video, and file transfer streams between the end hosts over the timeframe of a week (May 15-22, 2007). We generated both homogeneous and hybrid flows. Since we manually generated all of the flows in the data set, we can easily label each of these flows as either voice, video, or file transfer along with the application that generated the flow (Skype, MSN, or GTalk). The average time and number of packets in each of the sessions were 8 minutes and 9500 packets respectively. In total, we generated about 690 sessions, and a full breakdown of the sessions generated are given in Table II.

Standard	Codec Method	Inter-Packet Delay (ms)
G.711	PCM	.125
G.726	ADPCM	.125
G.728	LD-CELP	.625
G.729	CS-ACELP	10
G.729A	CS-ACELP	10
G.723.1	MP-MLQ	30
G.723.1	ACELP	30

Table I. Commonly used speech codec and their specifications

As we mentioned earlier in the paper, our classifier works in two phases: training and detection. The input to the training phase are homogeneous flows with labels, while the input to the detection phase are both homogeneous and hybrid flows that need to be labeled. Hence we split our data set into two sets - training set and detection set. The training data set contained 90 homogeneous flows (45 of them were voice and 45 of them were video). We use this set to train the

Prepared using secauth.cls
DOI: 10.1002/sec

VOVClassifier . The detection set contained the other 600 flows that includes both homogeneous and hybrid flows. We use this set to evaluate the performance of our classifier.

3. Challenges, Intuitions and Discussion

3.1. Challenges

We start this section, by considering the simpler case of homogeneous voice and video flows.

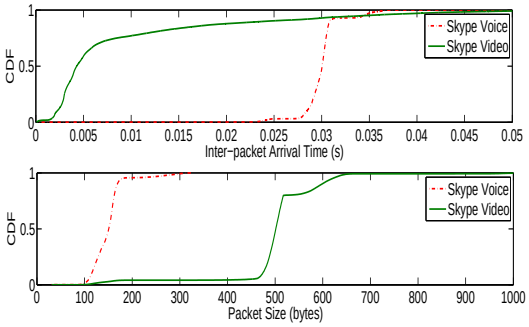

Fig. 1. CDF of IAT (on the top) and PS (on the bottom) for a typical Skype homogeneous flow.

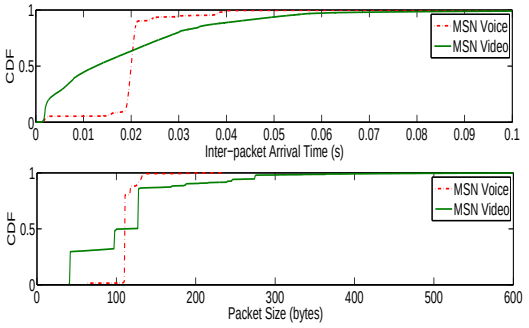

Fig. 2. CDF of IAT (on the top) and PS (on the bottom) for a typical MSN homogeneous flow.

Differentiating homogeneous voice flows from videos flows has been addressed as a research problem with limited success [3]. Two features that have been repeatedly proposed for this are *Packet Inter Arrival Time* (IAT) and *Packet Size* (PS). Figure 1 shows the cumulative distribution function (cdf) of the two metrics computed independently for a Skype voice and video homogeneous flow. The sharp knees in the cdf of IAT shows that both Skype voice and video flows show regularity in the packet inter-arrival time (30 ms for voice and 5 ms for video). Similarly, the knees in the cdf of PS shows that most of the packets in Skype voice flows are 120-160 bytes long, while most of the packets in Skype video flows are 480-520 bytes long. We find similar results for all of the Skype homogeneous flows in our data set. In order to differentiate between Skype voice and video homogeneous flows, one can use a simple filter that first checks regularity in IAT, and subsequently separates voice and video flows based on the PS distribution of the flow. In other words, Figure 1 suggests that IAT and PS can be computed independently from each other and we can use simple cut-off thresholds on the two distributions to distinguish voice from video flows.

Although such a simple technique works reasonably well for differentiating Skype traffic, it cannot be used in the general case for differentiating voice flows from video flows. For example, consider the case of MSN (Figure 2) which shows the cdf of IAT and PS for a MSN voice and video flow. We can see that voice packets tend to reach the destination at very regular time intervals of about 20 ms, with packets that are 105-120 bytes long. For the case of MSN video flows, the detection turns to be not as friendly as it was for Skype

Prepared using secauth.cls
DOI: 10.1002/sec

	Homogeneous Flows		Homogeneous Flows	
	Voice	Video	Voice+File	Video+File
Skype	54 / 20000/ 10	42 / 125000 / 25	88 / 220000 / 60	84 / 478000 / 60
MSN	82 / 58000/ 20	58 / 68000/20	80 / 428000 / 60	93 / 420000 / 60
GTalk	26 / 39000 / 20	38 / 33000 / 20	25 / 333000 / 60	30 / 401000 / 60

Table II. Number of flows, Average number of packets per flow, and the average duration of each flow collected for each application to evaluate the performance of *VOVClassifier*

video flows. First, video packets do not exhibit a strong regularity in IAT as shown by the complete absence of any knees in the cdf of IAT. Video packets reach the destination almost in a complete random fashion, thus can be easily interpreted as packets belonging to any other non-voice or non-video application. Second, even if we were able to find some regularities in IAT for video packets, we cannot distinguish video flows from voice flows due to the significant overlap of the PS distribution. As a consequence, it is hard to draw a clear cut boundary between the two, thus making the simple approach proposed for distinguishing Skype voice and video flows not generalizable for other applications.

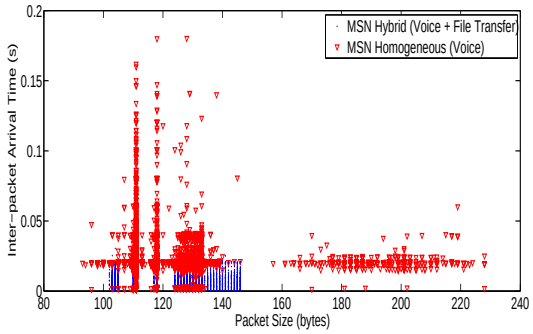

Fig. 3. IAT vs PS for a typical MSN hybrid video flow. We show the restricted range of packets between 100 and 240 Bytes to remark the impact to the IAT of voice when considering file transfer packets.

Next, we show how the strong regularity observed in the IAT distribution for both Skype and MSN homogeneous voice flows is severely affected when we consider hybrid flows. Figure 3 shows the PS and IAT for all packets in a typical MSN homogeneous and hybrid voice flows. We plot this graph using the following technique. We traverse the flow from the first packet to the last packet. For each packet, P_i, encountered in the flow, we record its size, PS_i, and the associated inter-arrival time to the next packet, IAT_i,. Each packet P_i is then represented by the pair $< PS_i, IAT_i >$. In Figure 3 we plot all packets P_i belonging to a MSN homogeneous and hybrid voice flow. As we also noted previously in Figure 2, Figure 3 shows that MSN homogeneous voice flows have a strong regularity with IAT of 20 ms around which the majority of packets lie. However, in case of hybrid flows, we can clearly see that this regularity is lost. Packets of size 100-140 Bytes (that represent the most common packet sizes used by MSN homogeneous voice flows) now span a large range of IAT values. As a consequence, such a flow does not exhibit any significant patterns that can reveal the presence of voice.

In order to quantify this, in Figure 4, we show the CDF of IAT for a MSN homogeneous and hybrid voice flows. As we can see, the sharp knees that exist for homogeneous flows (around 20 ms range) completely disappears for hybrid flows suggesting that these packets reach the destination in a pure random fashion.

Prepared using secauth.cls
DOI: 10.1002/sec

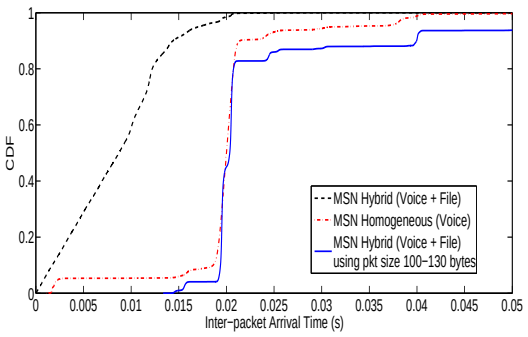

Fig. 4. CDF of IAT for a typical MSN homogeneous (on the top) and hybrid (on the bottom) voice flow.

3.2. Intuition and Approach

As we saw in Figures 2 and 4, homogeneous and hybrid voice and video flows do not always exhibit regularities in IAT. Also, the packet size distribution could vary considerably from one application to another. Hence, considering IAT and PS as independent metrics will not result in effective classification of voice and video.

However, our hypothesis is that *all* voice and video flows should exhibit some kind of a regularity/pattern in IAT and PS. These patterns could be hidden in the packet stream. Next we show how analyzing the spatial and temporal correlation of these two features, can reveal such hidden regularities. We carry out this analysis using MSN and show that such intuition has great potential in revealing the presence of voice and video streams in the context of both homogeneous and hybrid flows.

First, we consider the case of MSN homogeneous video flows for which no clear pattern can be observed when the two features (PS and IAT) were computed and analyzed independently (see Figure 2). For this case, we conduct two experiments using the original video flow[†]:

- We consider 128-byte packets in the flow along with their time stamps (i.e. the time at which we received the packet). Note that we discard all other packets in the flow, and the flow now has only 128 byte packets with their time stamps. We compute the cdf of the IAT of this new flow (Figure 5). We can now see that these packets exhibit regularities in their IAT.
- We consider sequence of packets, i.e., if two consecutive packets in the flow have sizes 128 and 42 Bytes respectively then we extract those packets along with their time stamps and discard the rest. We now plot the cdf of the IAT of this series and find that these sequence of packets exhibit regularity as well.

The main take-away point from the above experiments is that, although we do not find any significant patterns by considering the IAT and PS metrics individually, we find very strong patterns when we combine and analyze these metrics together. In other words, the regularities of a video flow reside in specific combinations of packet sizes and inter-arrival times that are maintained for the entire duration of the session. We observe the same phenomena for video traffic as well.

Next, we consider the case of MSN hybrid voice flow. We apply the same concept as before and focus on packets of sizes 100-130 bytes (the range of most common packets observed for MSN homogeneous voice

[†]Looking at the PS vs. IAT graphs for MSN homogeneous video flows, we found that there was a large number of packets of size 42 and 128. Hence we pick these packets for these experiments.

DOI: 10.1002/sec

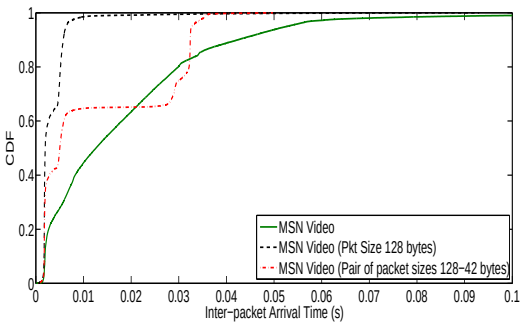

Fig. 5. CDF of IAT with certain PS grouping for MSN homogeneous video flows.

flows). In Figure 4, we report the CDF of IAT for such a flow. As we can clearly see, the shape of the CDF looks very similar to the original MSN homogeneous voice flow, and clearly is much different from the one computed by considering the two metrics independently from each other.

This forms the basis of our approach. In fact, we combine these two metrics using a stochastic process that is further analyzed in the frequency domain using power spectral density (PSD) analysis. Such a methodology searches for combinations of packet sizes and IAT pair-wise that occur more frequently than others and thus carry the majority of the energy of the stochastic process being created.

3.3. Chat traffic vs File Transfer traffic in Hybrid Flows

Hybrid multimedia flows can contain voice and/or video streams along with file transfer and/or chat streams. Both chat and file transfer streams do not show any hidden regularities in IAT distributions. Hence, from the perspective of our problem, both of these streams represent pure noise to the voice and video stream classifier. From our data set, we see that the PS distribution of chat streams typically ranges from 50 to 600 bytes where as the same for file transfer streams range from 60-1500 bytes. While the PS distribution of chat streams can be randomly distributed in its range, the distribution of PS in file transfer streams are mainly concentrated in two ranges: 90% of packets are in 1400-1500 bytes and 10% in 60-110 bytes.

In this work, we only consider hybrid flows that contain *voice or video streams along with file transfer streams*. We do this for two main reasons:

- The data rate of chat streams is much lower than the data rates of file transfer, voice, and video streams. During our experiments, we noticed that the number of chat packets observed in a 10 seconds observation window is negligible compared to the number of file transfer packets encountered. As a consequence, the presence of chat traffic can be interpreted as low level random noise that minimally impacts the IAT regularities of voice and video streams. On the other hand, we can find a lot more file transfer packets in any observation window. File transfer packets are highly interleaved with voice and video packets, and thus have great potential to severely impact the IAT distribution of the overall flow (as we can see in Figure 4).
- Although the PS distribution of chat spans a wide range of packets, from 50 to 600 Bytes, the average number of packets that fall in the same

DOI: 10.1002/sec

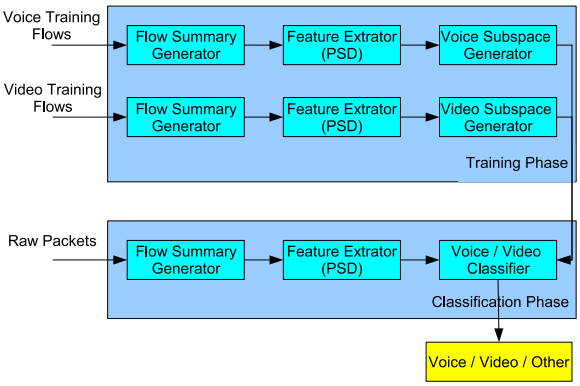

Fig. 6. VOVClassifier System Architecture

range of voice and video packets is minimal. On the other hand, the PS distribution of file transfer is heavily centered around 1400-1500 bytes, and the remaining packets resides in the range of 60-110 bytes, typical range used by voice traffic. As a consequence, the PS distribution of voice traffic might be severely impacted by the presence of file transfer. Note that although 10% is a small fraction of the overall file transfer PS distribution, it translates to a large number of packets (orders of magnitude more that the number of packets in chat streams), since the total number of packets in file transfer streams are very large.

4. System architecture

In this section we present the overall architecture of our system, named *VOVClassifier* and shown in Figure 6, and provide a high-level description of the functionalities of each of its modules. Generally speaking, *VOVClassifier* is an automated learning system that uses packet headers from raw packets collected off the wire, organizes them into transport network flows and processes them in realtime to search for voice and video applications. *VOVClassifier* first uses voice and video datastreams for training before being used in realtime for classification. During the training phase, *VOVClassifier* extracts feature vectors, which is a summary (also known as a statistic) of raw traffic bit streams, and maintains their statistics in memory. During the online classification phase, a classifier makes decision by measuring similarity metrics between the feature vector extracted from on-the-fly network traffic and the feature vectors extracted from training data. Flows with high values of similarity metric with the voice (or video) features are classified as voice (or video); datastreams with low values of similarity with voice/video are classified as other applications.

In general, *VOVClassifier* is composed by four major modules that operate in cascade: (i) *Flow Summary Generator* (FSG), (ii) *Feature Extractor* (FE) via Power Spectral Density analysis, (iii) *Voice/Video- Subspace Generator* (VSG) and (iv) *Voice/Video- CLassifier* (CL).

Flow Summary Generator(FSG): All packets collected off-the-wire are processed by the *Flow Summary Generator* module, that reorders packets by removing any duplicated packet, and organizes them into network transport flows according to their 5-tuple, e.g., source IP address, destination IP address, source port number, destination port number, and protocol type. The processed flow is then characterized in terms of packet sizes and inter-arrival times between packets within any generic flow $\mathcal{F}_S$, e.g.

Prepared using secauth.cls
DOI: 10.1002/sec

$$\mathcal{F}_S = \{\langle \mathcal{P}_i, \mathcal{A}_i \rangle ; i = 1, \ldots, I\}, \quad (1)$$

where $\mathcal{P}_i$ and $\mathcal{A}_i$ denote the packet size and relative packet arrival time of the ith packet in the flow with I packets, respectively. As we only consider relative arrival time, $\mathcal{A}_1$ is always 0.

Feature Extractor (FE) and Voice/Video- Subspace Generator(VSG): The FSG output is forwarded to the *Feature Extractor* module that computes a feature vector for each flow processed by analyzing its power spectral density (PSD) in order to exploit regularities residing in voice and video traffic. The *Voice/Video-Subspace Generator* processes the high dimensional feature vectors received and projects the feature vectors into a low dimensional space that embeds the fine granularity properties of the data stream in process. This is achieved by first partitioning the feature vector space into a few non-overlapping clusters or data sets and then extracting the characteristic of each cluster. As shown in Figure 6, the FSG and FE modules are used during both the training and the detection phases.

Voice/Video CLassifier (CL): During the detection phase, the data are processed by a module, named *Voice/Video- Classifier*, which calculates the distance from the feature vectors extracted from current datastreams entering the system to the voice and video subspaces generated during training in order to classify the stream as voice, video or other. The problem of datastream classification requires the implementation of a similarity metric. In literatures, there are many similarity metrics. For example, Bayesian classifier uses cost function, and nearest-neighbor (1-NN) and K-nearest-neighbor (KNN) use Euclidean distance. In general, no similarity metric is guaranteed to be the best for all applications. For example, Bayesian classifier is applicable only when the likelihood and prior probabilities are well estimated, which requires the number of training samples to be much larger than the number of feature dimensions. As a consequence, it is not suitable for classification based on a high dimensional feature vector, such as the PSD feature vector. Furthermore, neither 1-NN classifier nor K-NN classifier is suitable for mixed data that are in a collection of subspaces. We overcome the above problem by employing a similarity metric based on the normalized distance from feature vector representing the ongoing flow to the two subspaces obtained during training phase. The subspace at minimum distance will be elected as candidate only if the distance is below specific thresholds.

We conclude this section by highlighting one minor limitation of our approach. Our system is unable to distinguish a flow containing video only from a flow containing video packets piggybacked by voice data (when video and voice applications are simultaneously launched in Skype, voice data is piggybacked on video packets). This is because the feature for video packets piggybacked by voice data is very similar to that for video only. Hence, our traffic classifier will declare a flow containing video packets piggybacked by voice data as "video".

Prepared using secauth.cls
DOI: 10.1002/sec

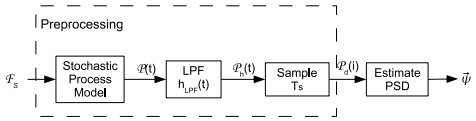

Fig. 7. PSD features extraction module. Cascade of processing steps.

5. Feature Extractor Module

As explained in Section 3, the extraction of traffic features like packet inter-arrival times and packet size, and the study of each of them individually, do not solve the problem of reliably detecting and separating voice and video data streams from other applications and from each other. In this section, we first introduce the preliminary steps that we take to transform each generic flow $\mathcal{F}_S$ obtained from the *Flow Summary Generator* (FSG) into a stochastic process that combines the inter-arrival times and packet sizes. Then we describe how to use power spectral density (PSD) analysis as a powerful methodology to extract such hidden key regularities residing in real-time multimedia network traffic.

5.1. Modeling the network flow as a stochastic process

Each flow $\mathcal{F}_S$ extracted from the *Flow Summary Generator* (FSG) is forwarded to the Feature Extractor (FE) module that takes several steps in cascade, as shown in Figure 7. First, any $\mathcal{F}_S$ extracted (see Equation (1)) is modeled as a continuous-time stochastic process as illustrated in Eq. (2):

$$\mathcal{P}(t) = \sum_{<\mathcal{P},\mathcal{A}>\in\mathcal{F}_S} \mathcal{P}\delta(t-\mathcal{A}), \qquad (2)$$

where $\delta(\cdot)$ denotes the delta function. As the reader can notice, our model combines packet arrival times and packet sizes to form a single stochastic process. Because digital computers are more suitable to deal with discrete-time sequences than continuous-time processes, we transform $\mathcal{P}(t)$ to a discrete-time sequence by applying sampling at frequency $F_s = \frac{1}{T_s}$.

Due to the fact that the signal defined in Equation (2) is represented as a summation of delta functions, its spectrum spans the whole frequency domain. In order to correctly reshape the spectrum of $\mathcal{P}_h(t)$ to avoid aliasing when it is sampled at interval T_s, we apply a low pass filter (LPF) characterized by its impulse response $h_{LPF}(t)$. $\mathcal{P}_h(t)$ can then be mathematically described as follows:

$$\mathcal{P}_h(t) = \mathcal{P}(t) * h_{LPF}(t) = \sum_{<\mathcal{P},\mathcal{A}>\in\mathcal{F}_S} \mathcal{P}h\left(t-\mathcal{A}\right). \qquad (3)$$

By sampling at interval T_s, we obtain the following discrete-time sequence:

$$\mathcal{P}_d(i) = \mathcal{P}_h(iT_s) = \sum_{<\mathcal{P},\mathcal{A}>\in\mathcal{F}_S} \mathcal{P}h\left(iT_s-\mathcal{A}\right), \qquad (4)$$

where $i = 1, \ldots, I_d = \frac{\mathcal{A}_{max}}{T_s} + 1$, $\mathcal{A}_{max}$ is the arrival time of the last packet in the flow.

We note that the sampling interval T_s cannot be arbitrarily chosen. If T_s is too large, then the spectrum of the flow F_s contains information only related to

DOI: 10.1002/sec

low frequencies and thus lacks of information related to the high frequency spectrum. On the other hands, if T_s is too small, then the length I_d of the resulting discrete-time sequence will be very large, resulting in very high complexity in computing the PSD of F_s. After an extensive analysis of widely-used voice and video applications such as Skype, MSN, and GTalk, we observed that choosing $T_s = 0.5$ ms is sufficient to extract all useful information for our purpose.

Next, we provide a methodology to extract the regularities residing in the signal $\mathcal{P}(t)$. We achieve this by studying the power spectral density of the extracted signal $\mathcal{P}_d(i)$.

5.2. Power spectral density computation

The power spectral density of a discrete-time signal represents its power distribution in the frequency domain. Regularities in time domain translate into dominant periodic components in its autocorrelation function and finally to peaks in its power spectral density.

For a general second-order stationary sequence $\{y_i; i \in \mathbb{Z}\}$, the power spectral density (defined in [19]) can be computed as:

$$\psi(\varpi; y) = \sum_{k=-\infty}^{\infty} r(k; y) e^{-j\varpi k},\ \varpi \in [-\pi, \pi), \quad (5)$$

where $\{r(k; y); k \in \mathbb{Z}\}$ represents the autocorrelation function of the signal $\{y_i; i \in \mathbb{Z}\}$, i.e.,

$$r(k; y) = \mathcal{E}\left[y(i)y^*(i-k)\right]. \quad (6)$$

Note that y in $r(k; y)$ is not an argument of function $r(\cdot)$ but an index, indicating that $r(\cdot)$ is the autocorrelation function of $\{y_i; i \in \mathbb{Z}\}$. This holds true for all other functions with two arguments separated by a semi-colon.

Although ϖ in Eq. (5) can take any value, we restrict its domain to be within $[-\pi, \pi)$ because $\psi(\varpi; y) = \psi(\varpi + 2\pi; y)$.

According to Eqs. (5) and (6), the computation of the PSD for a discrete-time signal theoretically requires an infinitely-long sequence. Since in reality, we cannot have an infinitely-long sequence, we need to use a finitely-long sequence to estimate the power spectral density with an admissible accuracy. In literature, two different families of PSD estimation are available: parametric and non-parametric. Parametric methods have shown to perform better under the assumption that the underlying model is correct and accurate. Furthermore, these methods are more interesting from a computational complexity perspective as they require the estimation of fewer variables when compared with non-parametric methods.

There are several parametric methods like autoregressive (AR) model, moving average (MA) model, and autoregressive moving average (ARMA) model. For simplicity, we choose the AR model and solve it using the Levinson-Durbin Algorithm (LDA). In this paper we skip the details of this algorithm for lack of space. For an extensive survey of parametric models for PSD estimation and details on the implementation of the LDA algorithm, we refer the reader to [19].

Prepared using secauth.cls
DOI: 10.1002/sec

Next, we assume $\{\mathcal{P}_d(i)\}_{i=1}^{I_d}$ to be second-order stationary. We can compute the PSD of $\{\mathcal{P}_d(i)\}_{i=1}^{I_d}$, denoted by

$$\psi\left(\varpi;\mathcal{P}_d\right),\varpi\in[-\pi,\pi). \tag{7}$$

Recall that $\{\mathcal{P}_d(i)\}_{i=1}^{I_d}$ are obtained by sampling a continuous-time signal $\mathcal{P}_h(t)$ with time interval T_s (see Fig. 7). Thus, one can further formulate the PSD in terms of real frequency as

$$\psi_f(f;\mathcal{P}_d) =\psi\left(\frac{2\pi f}{F_s};\mathcal{P}_d\right), \tag{8}$$

where $F_s=\frac{1}{T_s}$. Eq. (8) shows the relationship between the periodic components of a stochastic process in the continuous-time domain and the shape of its PSD in the frequency domain.

$\psi_f(f;\mathcal{P}_d)$ is a continuous function in frequency domain. To represent it in a computer, we need to do sampling in frequency domain. In other words, we select a series of frequencies,

$$0\le f_1<f_2<\cdots<f_M\le\frac{F_s}{2}, \tag{9}$$

and define the PSD feature vector as

$$\vec{\psi}=\left[\begin{array}{ccc}\psi_f\left(f_1;\mathcal{P}_d\right), & \psi_f\left(f_2;\mathcal{P}_d\right), & \psi_f\left(f_M;\mathcal{P}_d\right)\end{array}\right]^T. \tag{10}$$

$\vec{\psi}\in\mathbb{R}^M$ is the feature vector we use to perform classification.

In the next section, we introduce a new technique that we use to translate the characteristic of these high-dimensional feature vectors into a more tractable low dimensional space.

6. Subspace Decomposition and Bases Identification on PSD Features

In many scientific and engineering problems, the data of interest can be viewed as drawn from a mixture of geometric or statistical models instead of a single one. Such data are often referred to in different contexts as "mixed," or "multi-modal," or "multi-model," or "heterogeneous," or "hybrid." Subspace decomposition is a general method for modeling and segmenting such mixed data using a collection of subspaces, also known in mathematics as a subspace arrangement. By introducing certain new algebraic models and techniques into data clustering, traditionally a statistical problem, the subspace decomposition methodology offers a new spectrum of algorithms for data modeling and clustering that are in many aspects more efficient and effective than (or complementary to) traditional methods, e.g., principle component analysis (PCA), Expectation Maximization (EM), and K-Means clustering [20].

As illustrated in Fig. 6, we collect voice and video training flows during the training phase. After processing the raw packet data through the feature extraction module via PSD, one obtains two sets of feature vectors, i.e.,

Prepared using secauth.cls
DOI: 10.1002/sec

1.

$$\Psi^{(1)} \triangleq \left\{ \vec{\psi}_1(1), \vec{\psi}_1(2), \ldots, \vec{\psi}_1(N_1) \right\}, \quad (11)$$

which is obtained through voice training data, where N_1 is the number of voice flows;

2.

$$\Psi^{(2)} \triangleq \left\{ \vec{\psi}_2(1), \vec{\psi}_2(2), \ldots, \vec{\psi}_2(N_2) \right\}, \quad (12)$$

which is obtained through video training data, where N_2 is the number of video flows.

To facilitate further discussion, let us also regard $\Psi^{(i)}$ as a $M \times N_i$ matrix, for $i = 1, 2$, where each column is a feature vector. In other word, $\Psi^{(i)} \in \mathbb{R}^{M \times N_i}$.

In this section, we present techniques to identify the low dimensional subspaces embedded in $\mathbb{R}^M$, for both Ψ_1 and Ψ_2.

There are a lot of low dimensional subspace identification schemes, such as Principal Components Analysis (PCA) [21] and Metric Multidimensional Scaling (MDS) [22], which identify linear structure, and ISOMAP [23] and Locally Linear Embedding (LLE) [24], which identify non-linear structure.

Unfortunately, all these methods assume that data are embedded in one single low-dimensional subspace. This assumption is not always true. For example, as different software uses different voice coding, it is more reasonable to assume that the PSD feature vector of voice traffic is a random vector generated from a mixture model than a single model. In such case, it is more likely that there are several subspaces in which the feature vectors are embedded. The same holds for video feature vectors.

As a result, a better scheme is to first, cluster the trained feature vectors into several groups, known as subspace decomposition; and second, to identify the subspace structure of each group, known as subspace bases identification. We describe the two steps in the following sections.

6.1. Subspace Decomposition Based on Minimum Coding Length

The purpose of subspace decomposition is to partition the data set

$$\Psi = \left\{ \vec{\psi}(1), \vec{\psi}(2), \ldots, \vec{\psi}(N) \right\} \quad (13)$$

into non-overlapping K subsets such that

$$\Psi = \Psi_1 \cup \Psi_2 \cup \cdots \cup \Psi_K. \quad (14)$$

Ma et al. [20, 25] proposed a method to decompose subspaces according to the minimum coding length criterion. The idea is to view the data segmentation problem from the perspective of data coding/compression.

Suppose one wants to find a coding scheme, $\mathcal{C}$, which maps data in $\Psi \in \mathbb{R}^{M \times N}$ to a bit sequence. As all elements are real numbers, an infinitely long bit sequence is needed to represent each element without error. To make it practical, one has to specify a tolerable decoding error, ϵ, to obtain a mapping with finite coding length, i.e.,

Prepared using secauth.cls
DOI: 10.1002/sec

$$\left\|\vec{\psi}_n - \mathcal{C}^{-1}\left(\mathcal{C}\left(\vec{\psi}_n\right)\right)\right\|^2 \leq \epsilon^2, \text{ for } \forall n = 1, \ldots, N. \tag{15}$$

Then the coding length of the coding scheme $\mathcal{C}$ is a function

$$\mathcal{L}_{\mathcal{C}} : \mathbb{R}^{M \times N} \rightarrow \mathbb{Z}_+. \tag{16}$$

The optimal partition (see Eq. (14)) should minimize coding length of the segmented data, i.e.,

$$\min_{\Pi} \hat{\mathcal{L}}_{\mathcal{C}}(\Psi; \Pi) = \min_{\Pi} \left\{ \sum_{k=1}^{K} \mathcal{L}_{\mathcal{C}}(\Psi_k) + \sum_{k=1}^{K} |\Psi_k| \left[-\log_2 \left(\frac{|\Psi_k|}{K} \right) \right] \right\}, \tag{17}$$

where Π denotes the partition scheme. The first term in Eq. (17) is the summation of coding length of each group, and the second one is the number of bits needed to encoding membership of each item of Ψ in the K groups.

The optimal partition is achieved in the following way. Let the segmentation scheme be represented by the membership matrix,

$$\Pi_k \triangleq \operatorname{diag}\left([\pi_{1k}, \pi_{2k}, \ldots, \pi_{Nk}]\right) \in \mathbb{R}^{N \times N}, \tag{18}$$

where π_{nk} denotes the probability that vector $\vec{\psi}(n)$ belongs to subset k, such that

$$\sum_{k=1}^{K} \pi_{nk} = 1, \text{ for } \forall n = 1, \ldots, N \tag{19}$$

and $\operatorname{diag}(\cdot)$ denotes converting a vector to a diagonal matrix.

Hong [25, page 34] proved that the coding length is bounded as follows.

$$\begin{aligned} &\hat{\mathcal{L}}_{\mathcal{C}}(\Psi; \Pi) \\ &\leq \sum_{k=1}^{K} \left[\frac{\operatorname{tr}(\Pi_k) + M}{2} \log_2 \det \left(I + \frac{M}{\operatorname{tr}(\Pi_k)\, \epsilon^2} \Psi \Pi_k \Psi^T \right) \right] \\ &\quad + \sum_{k=1}^{K} \left[\operatorname{tr}(\Pi_k) \left(-\log_2 \frac{\operatorname{tr}(\Pi_k)}{N} \right) \right] \\ &\triangleq \hat{\mathcal{L}}(\Psi; \Pi), \end{aligned} \tag{20}$$

where $\operatorname{tr}(\cdot)$ denotes the trace of a matrix, and $\det(\cdot)$ denotes matrix determinant. Combining Eqs. (17) and (20), one achieves a minimax criterion

$$\hat{\Pi} = \arg\min_{\Pi} \left[\max_{\mathcal{C}} \hat{\mathcal{L}}_{\mathcal{C}}(\Psi; \Pi) \right] = \arg\min_{\Pi} \hat{\mathcal{L}}(\Psi; \Pi). \tag{21}$$

There is no closed form solution for Eq. (21). Hong [25, page41] proposed a pairwise steepest descent method to solve it.

Using the above method, we obtain a partition of voice feature vector set $\Psi^{(1)}$,

$$\Psi^{(1)} = \Psi_1^{(1)} \cup \cdots \cup \Psi_{K_1}^{(1)}, \tag{22}$$

Prepared using secauth.cls
DOI: 10.1002/sec

1. function $\left[\vec{\mu}, \hat{\mathcal{U}}, \hat{\Sigma}, \bar{\mathcal{U}}, \bar{\Sigma}\right]$ = *IdentifyBases*$\left(\Psi \in \mathbb{R}^{M\times N}, \delta\right)$
2. $\vec{\mu} = \frac{1}{|\Psi|}\sum_{\vec{\psi}\in\Psi}\vec{\psi}$
3. $\bar{\Psi} = \left[\vec{\psi}_1 - \vec{\mu}, \vec{\psi}_2 - \vec{\mu}, \ldots, \vec{\psi}_{|\Psi|} - \vec{\mu}\right]$
4. Do eigenvalue decomposition on $\bar{\Psi}\bar{\Psi}^T$ such that

$$\bar{\Psi}\bar{\Psi}^T = \mathcal{U}\Sigma\mathcal{U}^T, \tag{25}$$

where $\mathcal{U} \triangleq [\vec{u}_1, \cdots, \vec{u}_M]$, $\Sigma \triangleq \text{diag}\left(\left[\sigma_1^2, \ldots, \sigma_M^2\right]\right)$, and $\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_M^2$.

5. $\mathcal{J} = \arg\min_{\hat{\mathcal{J}}} \sum_{m=1}^{\hat{\mathcal{J}}} \sigma^2(m) \geq \delta \sum_{m=1}^{M} \sigma^2(m)$
6. $\hat{\mathcal{U}} = [\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_{\mathcal{J}-1}]$
7. $\bar{\mathcal{U}} = [\vec{u}_{\mathcal{J}}, \vec{u}_{\mathcal{J}+1}, \ldots, \vec{u}_M]$
8. $\hat{\Sigma} = \text{diag}\left(\left[\sigma_1^2, \ldots, \sigma_{\mathcal{J}-1}^2\right]\right)$
9. $\bar{\Sigma} = \text{diag}\left(\left[\sigma_{\mathcal{J}}^2, \ldots, \sigma_M^2\right]\right)$
10. end function

Fig. 8. Function *IdentifyBases* identifies bases of subspace.

and a partition of video feature vector set $\Psi^{(2)}$,

$$\Psi^{(2)} = \Psi_1^{(2)} \cup \cdots \cup \Psi_{K_2}^{(2)}. \tag{23}$$

Next, we describe the method to identify subspace bases in each of the segmentations.

6.2. Subspace Bases Identification

In this section, we use PCA algorithm to identify subspace bases for each segmentation,

$$\left\{\Psi_k^{(i)}; k = 1, \ldots, K_i, i = 1, 2\right\}, \tag{24}$$

which obtained in the previous section. The basic idea is to identify uncorrelated bases and choose those bases with dominant energy. Fig. 8 shows the algorithm.

In Fig. 8, argument Ψ represents the feature vector set of one segmentation and δ is a user defined parameter which specifies the percentage of energy retained, e.g., 90% or 95%. The algorithm returns five sets of output variables. $\vec{\mu}$ represents the sampled mean of all feature vectors. It is the origin of the identified subspace. The columns of $\hat{\mathcal{U}}$ are the bases with dominant energy (i.e.,variance), whose corresponding variances are denoted with $\hat{\Sigma}$. These bases determine the identified low dimensional subspace spanned by Ψ. The columns of $\bar{\mathcal{U}}$ compose the null space of the previous subspace, whose corresponding variances are $\bar{\Sigma}$. The last two outputs are required to calculate the distance of an ongoing feature vector to the subspace, which will be described in Section 7.

Applying the function *IdentifyBases* on all segmentations, we obtain

$$\left[\vec{\mu}_k^{(i)}, \hat{\mathcal{U}}_k^{(i)}, \hat{\Sigma}_k^{(i)}, \bar{\mathcal{U}}_k^{(i)}, \bar{\Sigma}_k^{(i)}\right] = \text{IdentifyBases}(\Psi_k^{(i)}) \tag{26}$$

for $\forall k = 1, \ldots, K_i$, $\forall i = 1, 2$. These are the outputs of subspace identification module, and hence the results of training phase, in Fig. 6.

During the classification phase, these outputs are used as system parameters, which will be presented in the next section.

7. Voice/Video Classifier

In Section 6, we presented an approach to identifying subspaces spanned by PSD feature vectors of training voice and video flows. Specifically, one obtains the following parameters:

Prepared using secauth.cls
DOI: 10.1002/sec

$$\left[\vec{\mu}_k^{(i)}, \hat{\mathcal{U}}_k^{(i)}, \hat{\Sigma}_k^{(i)}, \bar{\mathcal{U}}_k^{(i)}, \bar{\Sigma}_k^{(i)}\right] \quad (27)$$

for $\forall k = 1, \ldots, K_i$, $\forall i = 1, 2$. In this section, we use these parameters to do classification.

As shown in Fig. 6, during the classification phase, for each ongoing flow $\mathcal{F}$, one creates a sub-flow, $\mathcal{F}_S$, by extracting small packets, i.e., packets smaller than θ_P, and passes it through PSD feature extraction module to generate PSD feature vector $\vec{\psi}$. This is the input to the voice/video classifier.

The voice/video classifier works in the following way. It first calculates the normalized distances between $\vec{\psi}$ and all subspaces of both categories. Then it chooses minimum distance to each category. The decision is made by comparing the two distance values to two thresholds, θ_A and θ_V respectively for voice and video. Fig. 9 shows the procedure of the voice/video classifier.

1. function $\left[\hat{\mathcal{H}}_1, \hat{\mathcal{H}}_2\right]$ = *voicevideoClassify*$\left(\vec{\psi}, \theta_A, \theta_V\right)$
2. For $\forall i = 1, 2, \quad \forall k = 1, \ldots, K_i \quad d_k^{(i)} =$ *NormalizedDistance* $\left(\vec{\psi}, \vec{\mu}_k^{(i)}, \bar{\mathcal{U}}_k^{(i)}, \bar{\Sigma}_k^{(i)}\right)$
3. For $\forall i = 1, 2$, $d_i = \min_k d_k^{(i)}$
4. if $d_1 < \theta_A$ and $d_2 > \theta_V$
5. $\hat{\mathcal{H}}_1 = 1, \hat{\mathcal{H}}_2 = 0$, i.e., voice flow.
6. else if $d_1 > \theta_A$ and $d_2 < \theta_V$
7. $\hat{\mathcal{H}}_1 = 1, \hat{\mathcal{H}}_2 = 1$, i.e., video flow.
8. else
9. $\hat{\mathcal{H}}_1 = 0, \hat{\mathcal{H}}_2 = 0$, i.e., neither voice nor video.
10. end if
11. end function
12. function d = *NormalizedDistance*$\left(\vec{\psi}, \vec{\mu}, \bar{\mathcal{U}}, \bar{\Sigma}\right)$
13. $d = \left(\vec{\psi} - \vec{\mu}\right)^T \bar{\mathcal{U}} \bar{\Sigma}^{-1} \bar{\mathcal{U}}^T \left(\vec{\psi} - \vec{\mu}\right)$
14. end function

Fig. 9. Function *voicevideoClassify* determines whether a flow with PSD feature vector $\vec{\psi}$ is of type voice or video or neither. θ_1 are θ_2 are two user-specified threshold arguments. Function *voicevideoClassify* uses Function *NormalizedDistance* to calculate normalized distance between a feature vector and a subspace.

Note that, in Function *voicevideoClassify*, line 7, when $\hat{\mathcal{H}}_2 = 1$, we always set $\hat{\mathcal{H}}_1 = 1$. The reason is discussed in Section 4.

8. Experimental Results

In this section, we first validate our approach and then demonstrate the effectiveness and feasibility of *VOVClassifier* . In other words, we will first show how the FE module extracts fingerprints that are unique to voice and video flows. We then show how the VSG modules generates voice and video subspaces that are distinct from each other, thus giving us an opportunity to use simple linear classifiers to separate voice and video flows. Finally, we will show the effectiveness of our entire system in clearly identifying voice and video streams in the context of hybrid flows and multiple applications.

8.1. Feature Extractor Module

As we explained in Section 5, the FE module takes homogeneous voice and video flows as input to first compute a stochastic process for a flow (Equation 2) using IAT and packet sizes, and then extract a fingerprint using the PSD distribution (Equation 10). Figure 10 shows the PSD of the stochastic process for voice and video flows. We randomly picked two Skype homogeneous voice and video flows from our data traces. The top and bottom graphs in Figure 10 show the PSD fingerprint generated by the FE module for the two voice flows and two video flows respectively. There are two main observations in this figure: (i) The PSD

DOI: 10.1002/sec

fingerprints for voice and video flows are very distinct from each other. In other words, these fingerprints can be used to clearly distinguish voice flows from video flows. (ii) The PSD fingerprint of the two voice flows (and the two video flows) are very close to each other. This implies that when these flows are clustered together by the VSG, *all* the voice flows can be clustered into a sphere in hyperspace with very small radius. This small and concentrated cluster can be easily differentiated from other clusters using linear classifiers. Notice that in the context of application classification this characterstic can be extremely useful. Being able to extract such similar fingerprints for voice or video flows generated by the different applications, helps to easily make the distinction between one application from another a very straightforward task.

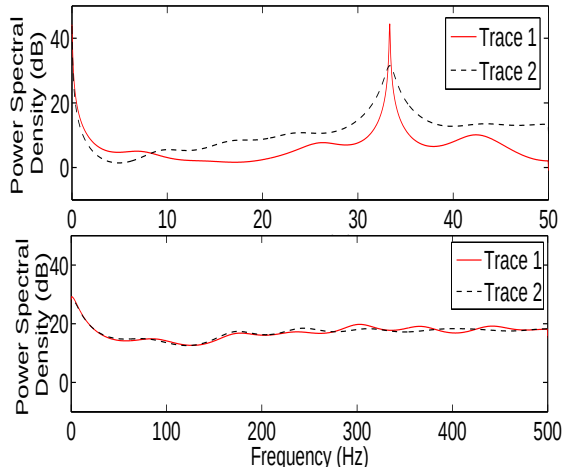

Fig. 10. PSD Fingerprints of Skype homogenenous voice and video streams.

8.2. Voice and Video Subspace Generator

The output from the FE module is used by the VSG module to generate voice and video subspaces (or clusters) that are far away from each other. The larger the distance between the two subspaces, the easier it is to classify any flows containing voice and video streams. Figure 11 shows the distance of a Skype homogeneous flow from the voice subspace on the x-axis and the distance from the video subspace on the y-axis. For this result, we first train the *VOVClassifier* using several homogeneous voice and video flows from our traces. After the training phase, we choose many hybrid flows (containing voice, video, and file transfer) for which we already know the correct classification, and feed it to the classifier to conduct online classification. We computed the distance of all of these flows the two subspaces that we had computed in the offline training phase, and plot them in Figure 11. We can clearly see that all the hybrid flows that contain voice (marked in red in the figure), are very close to the voice subspace as compared to the hybrid flows that contain video streams(which are very far from the voice subspace). This implies that as soon as the *VOVClassifier* computes the PSD fingerprint and calculates the distance from the existing subspaces, it can tell whether a flow contains voice or video streams using a simple threshold based scheme. This figure has two important take-away points: (i) The VSG module actually generates subspaces that are far away from each other, thus making it easy to classify traffic based on simple thresholds. (ii) The *VOVClassifier* can perform efficient and effective online classification.

Prepared using secauth.cls
DOI: 10.1002/sec

Finally, we wish to point out a limitation in our system. In Figure 11, we can see that some of the video flows are clustered with voice flows. Our analysis shows that these video flows are hybrid flows that contain both voice and video. In other words, when a flow contains both voice and video streams, the *VOVClassifier* will classify this as only voice flows, but not video flows. The reason of this is the well-known piggy-back methodology characteristic of voice and video applications that piggyback voice and video packets together to reduce the overall overhead on the data rate.

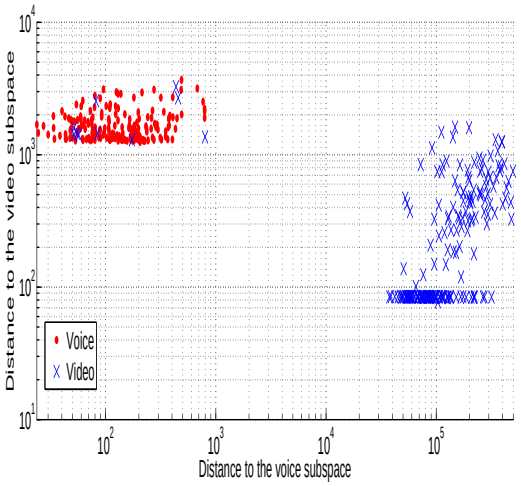

Fig. 11. Output of the training phase when considering multiple Skype homogeneous voice and video flows.

8.3. Overall System

After the offline training phase, the main goal of the *VOVClassifier* is to label all the incoming flows as containing voice, video, or neither. Note that these labeling categories could be more specific than just voice and video. For example, we could train the *VOVClassifier* to label the flows as Skype voice, MSN voice, Skype video, and MSN video. However, for the ease of presentation in this section, we choose to label an incoming flow as one of the following: voice, video, and neither.

Figures 12 and 13 show the *Receiver Operating Characteristics* (ROC) curve for *VOVClassifier* . In other words, the y-axis in the graph shows the probability of *correctly* labeling a flow, P_D, and the x-axis shows the probability of a false alarm or *incorrectly* labeling a flow, P_{FA}. We formally define P_D and P_{FA} as follows: when a flow with actual label X is sent through the *VOVClassifier* for classification, and the classifier labels the flow as $\hat{X}$, then

$$P_{D|X} \stackrel{\triangle}{=} P\left(\hat{X}=1 \middle| X=1\right), \tag{28}$$

$$P_{FA|X} \stackrel{\triangle}{=} P\left(\hat{X}=1 \middle| X=0\right), \tag{29}$$

We first consider the case of flows generated by a single application only, i.e., Skype. In Figure 12, we can see that the detection rate of the Skype homogeneous voice and video flows are very high (over 99%) when the false alarm rate is very small (<1%). Comparable results were obtained for hybrid voice flows, while for hybrid video flows, we can observe a slight drop in performance, with 94% detection rate for a false alarm rate of 1% or below. Similar results were obtained for MSN and Gtalk. This first set of results shows that the VOVClassifier can effectively and accurately classify voice and video flows generated by one single

DOI: 10.1002/sec

application even when these flows are mixed with other streams like file transfer.

Now we consider a more complex scenario in which homogeneous and hybrid flows are generated by a mixture of applications, i.e., Skype, MSN and GTalk. In this case, the *VOVClassifier* is asked to (i) classify voice and video flows as before and (ii) label each flow with the associated application being used to generate such a flow. As a consequence, a homogeneous voice flow being generated by MSN but labeled as Skype homogeneous voice will be considered as a false positive. Figure 13 shows the ROC curve for our classifier while using several hundred homogeneous and hybrid voice and video flows (as described in Section 2) as the input. As the first step, we used 15 voice and 15 video flows (5 each for Skype, MSN, and GTalk) to train our classifier. We then use the rest of the flows to test the system accuracy. Note that we already know the actual labels (i.e. the ground truth) for all of the input flows. We now let our system make a decision on these flows and label them. In Figure 13, we can see that the detection rate of the homogeneous voice and video flows are very high (over 99%) when the false alarm rate is very small ($<$1%). However, when we input hybrid flows, the results are not as good. In other words, if we want to keep the false alarm rate to less than 1%, then the detection rate will also be very low (between 20-30%). However, when the false positive rate is about 4%, the overall detection rate jumps beyond 99%, thus showing that the *VOVClassifier* can effectively and accurately label voice and video flows even when these flows are mixed with other streams like file transfer and chat.

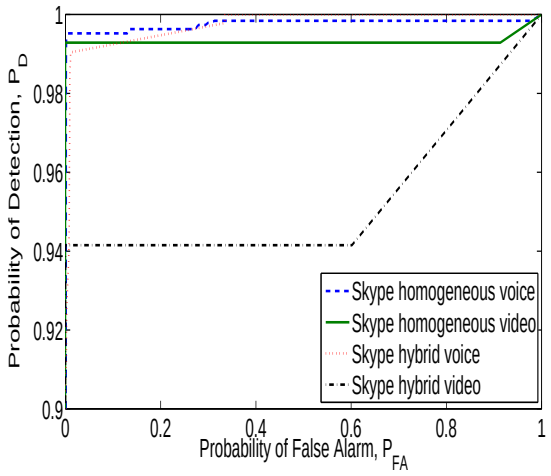

Fig. 12. ROC values for homogeneous and hybrid flows generated by Skype.

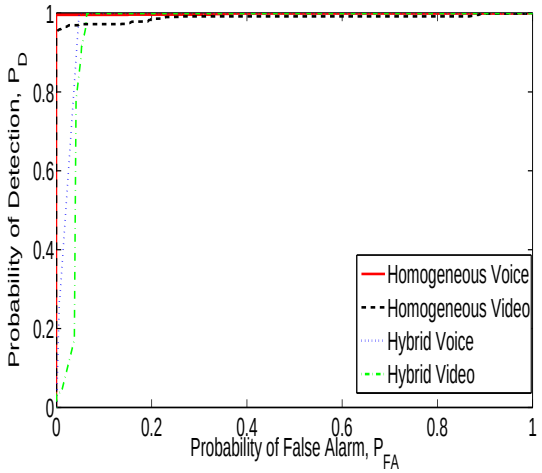

Fig. 13. ROC values for homogeneous and hybrid flows generated by Skype, MSN, and Google Talk.

Based on the aboe figures we can make the following observations about the overall results:

- **Voice vs. Video Flows.** From Figures 12 and 13, we can see that the classification of voice flows is more accurate than video flows. Specifically, we can achieve 100% accurate classification for

Prepared using secauth.cls
DOI: 10.1002/sec

Skype voice flows. This is due to the fact that voice flows have a higher regularity than video flows, as shown in Figure 10. From the voice PSDs, we can see that the dominant periodic component for voice flows is at 33Hz, the frequency corresponding to 30-millisecond IPD of the employed voice coding. On the other hand, video PSDs have peaks at 0, implying that a non-periodic component dominates video flows. Also, the PSDs of two video flows are close to each other. This is one of the main reasons why our approach achieves high classification accuracy by using PSD features.

- **Homogeneous vs. Hybrid Flows.** From Figures 12 and 13, we can see that the classification of homogeneous flows is more accurate than that of hybrid flows. Mixing multiple types of traffic together is like increasing noise. Hence, it is not surprising that classification accuracy is reduced.
- **One vs. Multiple Applications.** We can also see that the classification of Skype flows is more accurate than the classification of flows generated from three applications (Skype, MSN, and GTalk). Empirically, we found that Skype flows are similar to GTalk flows, but quite different from MSN. For example, both Skype and GTalk voice flows have 33-millisecond inter-arrival time, whereas MSN voice flows have a 25-millisecond inter-arrival time. When these flows are mixed together, the classification accuracy is reduced. However it is importat to note that the reduction in accuracy still results in acceptable detection rates. Specifically, for hybrid voice traffic at $P_{FA} \approx 0$, P_D is reduced from 1 to 0.986, and for hybrid video traffic, P_D is reduced from 0.965 to 0.948. This shows that our approach is robust. The robustness results from the fact that the subspace identification module, as presented in Section 6, decomposes multiple subspaces in the original high-dimensional feature space. As a result, PSD feature vectors of Skype and GTalk are likely to be within different subspaces than those of MSN. Therefore, we can still classify traffic accurately.

9. Conclusions

In this paper, we presented a novel system, called *VOVClassifier* , that provides a robust and accurate classification technique for voice and video flows. We have shown that this system is able to detect the presence of voice and video data streams both in the context of homogeneous and hybrid flows. To determine the existence of voice and video traffic, our system (i) models a network flow using a stochastic process that combines the *inter-arrival times* of packets within the flow and the associated *packet size*; (ii) extracts the hidden regularities of voice and video streams, and highlights their major differences by applying power spectral density analysis. Their fundamental properties are captured as feature vectors; (iii) groups feature vectors characterized by some degree of similarity (e.g., associated to the same type of traffic and application)

DOI: 10.1002/sec

into subspaces and reduces the high dimension of this subspace into a more manageable one by applying principal component analysis, and (iv) uses minimum coding length as the similarity metric to perform classification.

The results from this system demonstrates the effectiveness and robustness of our approach. We showed that *VOVClassifier* could achieve 99.5% detection rate with false positive close to 0% for both voice and video in the case of homogeneous flows, and 99.5% and 95.5% (false positive close to 0) respectively for voice and video when dealing with the more complex scenario of hybrid flows.

References

1. J. Curtis, J. Cleary, A. McGregor, M. Pearson, Measurement of Voice over IP Traffic, in: Proccedings of Passive and Active Measurements, Hamilton, New Zealand, 2000.
2. http://www.packetizer.com/ipmc/h323/standards.html.
3. M. Roughan, S. Sen, O. Spatscheck, N. Duffield, Class-of-service mapping for qos: A statistical signature-based apporach to ip traffic classification), in: In Proceedings of the ath ACM Sigcomm conference on Internet Measurement, Taormina, Italy, 2004.
4. A. Moore, D. Zuev, Internet Traffic Classification Using Bayesian Analysis Techniques, in: ACM Sigmetrics, Banff, Canada, 2005.
5. W. Charles, M. Fabian, M. Geral, Hmm profiles for network traffic classification (extended abstract), in: In Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, Washington, DC, USA, 2004.
6. S. Ehlert, S. Petgang, T. Magedanz, D. Sisalem, Analysis and signature of skype voip session traffic, in: In Proceeding (533) Communications, Internet, and Information Technology, St. Thomas, USVI, USA, 2006.
7. D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, P. Tofanelli, Revealing Skype Traffic: When Randomness Plays with You, in: ACM Sigcomm, Kyoto, Japan, 2007.
8. J. Erman, M. Arlitt, A. Mahanti, Traffic Classification using Clustering Algorithms, in: ACM SIGCOMM Workshop on Mining Network Data, 2006.
9. N. Williams, S. Zander, G. Armitrage, A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification, in: CCR, 2006.
10. L. Bernaille, R. Teixeira, K. Salamatian, Early Application Identification, in: CoNEXT'06, 2006.
11. T. Karagiannis, K. Papagiannaki, M. Faloutsos, Blinc: multilevel traffic classification in the dark, in: SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, ACM Press, New York, NY, USA, 2005, pp. 229–240.
12. F. Constantinou, P. Mavrommatis, Identifying Known and Unknown Peer-to-Peer Traffic, in: IEEE International Symposium on Network Computing and Applications, 2006.
13. S. Zander, T. Nguyen, G. Armitage, Automated Traffic Classification and Application Identification using Machine Learning, in: LCN'05, 2005.
14. A. Moore, K. Papagiannaki, Toward the Accurate Identification of Network Applications, in: PAM, 2005.
15. S. Zander, T. Nguyen, G. Armitage, Self-Learning IP Traffic Classification Based on Statistical Flow Characteristics, in: PAM, 2005.
16. A. McGregor, M. Hall, P. Lorier, J. Brunskill, Flow Clustering Using Machine Learning Techniques, in: PAM, 2004.
17. T. Karagiannis, A. Broido, M. Faloutsos, K. Claffy, Transport layer identification of p2p traffic, in: IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM Press, New York, NY, USA, 2004, pp. 121–134.
18. C. Dewes, A. Wichmann, A. Feldmann, An analysis of internet chat systems, in: IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, ACM Press, New York, NY, USA, 2003, pp. 51–64.
19. P. Stoica, R. Moses, Spectral Analysis of Signals, 1st Edition, Prentice Hall, 2005.
20. Y. Ma, H. Derkesen, W. Hong, J. Wright, Segmentation of multivariate mixed data via lossy coding and compression, to appear in IEEE Transactions on Pattern Analysis and Machine

DOI: 10.1002/sec

Intelligence.

21. L. I. Smith, A tutorial on principal components analysis (Feb. 2002).
    URL `http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf`
22. K. V. Deun, L. Delbeke, Multidimensional scaling, University of Leuven.
    URL `http://www.mathpsyc.uni-bonn.de/doc/delbeke/delbeke.htm`
23. J. B. Tenenbaum, V. de Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
24. S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
25. W. Hong, Hybrid models for representation of imagery data, Ph.D. thesis, University of Illinois at Urbana-Champaign (Aug. 2006).

Prepared using secauth.cls
DOI: 10.1002/sec