# TOWARDS MACHINE LEARNING BASED SOURCE IDENTIFICATION OF ENCRYPTED VIDEO TRAFFIC

By

Yan Shi

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering – Doctor of Philosophy

2019

## ABSTRACT

TOWARDS MACHINE LEARNING BASED SOURCE IDENTIFICATION
OF ENCRYPTED VIDEO TRAFFIC

By

Yan Shi

The rapid growth of the Internet has helped to popularize video streaming services, which has now become the most dominant content on the Internet. The management of video streaming traffic is complicated by its enormous volume, diverse communication protocols and data formats, and the widespread adoption of encryption. In this thesis, the aim is to develop a novel firewall framework, named Soft-margined Firewall, for managing encrypted video streaming traffic while avoiding violation of user privacy. The system distinguishes itself from conventional firewall systems by incorporating machine learning and Traffic Analysis (TA) as a traffic detection and blocking mechanism. The goal is to detect unknown network traffic, including traffic that is encrypted, tunneled through Virtual Private Network, or obfuscated, in realistic application scenarios. Existing TA methods have limitations in that they can deal only with simple traffic patterns – usually, only a single source of traffic is allowed in a tunnel, and a trained classifier is not portable between network locations, requiring redundant training. This work aims to address these limitations with new techniques in machine learning. The three main contributions of this work are: 1) developing new statistical features around traffic surge periods that can better identify websites with dynamic contents; 2) a two-stage classifier architecture to solve the mixed-traffic problem with state-of-the-art TA features; and 3) leveraging a novel natural-language inspired feature to solve the mixed-traffic problem using Deep-Learning methods. A fully working Soft-margin Firewall with the above distinctive features have been designed, implemented, and verified for both conventional classifiers and the proposed deep-learning based classifiers. The efficacy of

the proposed system is confirmed via experiments conducted on actual network setups with a custom-built prototype firewall and OpenVPN servers. The proposed feature-classifier combinations show superior performance compared to previous state-of-the-art results. The solution that combines natural-language inspired traffic feature and Deep-Learning is demonstrated to be able to solve the mixed-traffic problem, and capable of predicting multiple labels associated with one sample. Additionally, the classifier can classify traffic recorded from locations that are different from where the trained traffic was collected. These results are the first of their kind and are expected to lead the way of creating next-generation TA-based firewall systems.

# ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Subir Biswas for his extraordinary support and guidance during my work. He has advised me not only on my research but also on my career and life. I thus feel fortunate to work with him and proud of being a member of his research group. I would also like to thank my committee, Dr. Tongtong Li, Dr. Daniel Morris, and Dr. Eric Torng for their time and support.

Thanks must be given to my lab mates Stephan Lorenz, Bo Dong, Faezeh Hajiaghajani, Saptarshi Das, Benjamin Sanda, Dezhi Feng, Brandon Harrington, Henry Griffith, Rui Wang, and Tianyi Wu, for all of the brainstorming and implementation discussions. Finally, I would like to give thanks to my parents and family for their unconditional love and support and encouragement throughout my life. My achievement would not be possible without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# Chapter 1 : Introduction

## 1.1 Motivation

The rapid growth of the Internet has helped to popularize various data-intensive applications. One prominent example is video streaming, which has become the dominant type of application on today's Internet in terms of traffic volume according to a 2016 research report [1]. For example, the top 2 most heavily accessed video streaming sites, Netflix and YouTube, constitute roughly 50% of the North American Internet traffic (according to the same 2014 report). As the popularity of video streaming sites grows, their usage also starts to spread into private enterprises, where watching certain videos may be undesirable. For example, employers may not want their employees to watch certain videos containing political, sexual, or violence related messages. Often the administrator of an enterprise network may want to block users from watching videos in order to conserve bandwidth and/or to maintain enterprise productivity. At the same time, an administrator may wish to allow users to access some of the video streaming sites due to other business reasons. As such, the rise of video streaming applications poses an interesting problem in network traffic management.

Enforcing the regulations on network traffic first requires the classification of traffic, which is a problem that draws much interest from both the networking community and the business side. It is well investigated in the past. Various solutions have been proposed and applied to the problem[2][3]. Existing solutions fall into two categories: rule-based solutions, and Traffic Analysis (TA) based solutions. They are effective in their respective problem domains, and they have found applications in commercial networking devices.

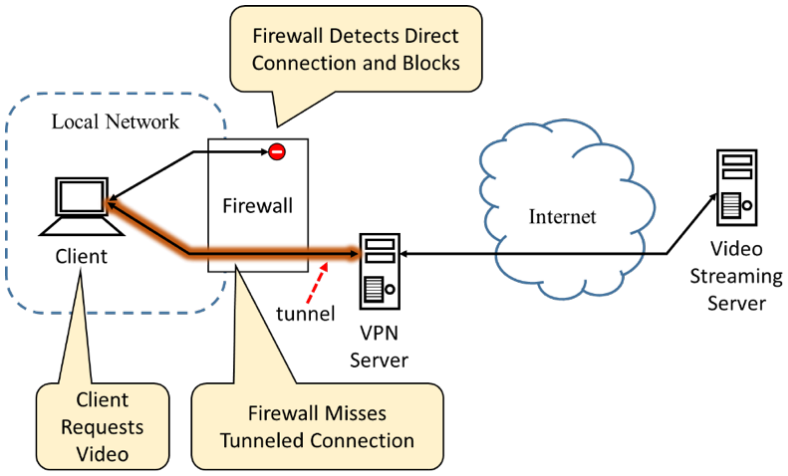



Figure 1-1: Firewall circumvention via virtual private networks

However, existing solutions face several challenges. Firstly, the adversaries, who want to avoid detection, upgrade their methods over time, resulting in an arms race between them and the network administrators (an example scenario is shown in Figure 1-1). Traffic encryption technologies such as VPN [4] and The Onion Router (Tor) [5] are widely accessible nowadays, and researchers are finding new ways to masquerade traffic [6]. Therefore, the current methods need reevaluation and improvement.

Secondly, the amount and diversity of network traffic have increased drastically during the last decade. Existing solutions might not scale well and needs reevaluation under the new traffic conditions. Growth in video streaming traffic is arguably the most significant recent change in network traffic, but existing studies, including recent ones [7], [8], only target web browsing traffic. Analysis of heterogeneous traffic (where multiple types of network traffic occur at the same time) is left out of the existing research as well, but happening quite often in real-world situations. Therefore, the targeted traffic type needs to be extended to cover these changes.

Lastly, traffic classification function is still limited to professional networking devices [9], while a need for traffic monitoring exists outside of the professional environment (for example,

from parents who want to manage the traffic going in/out of the home network). The technology needs miniaturization and cost reduction for it to reach a wider population. At the same time, the increasing amount of data has turned this into a problem that requires the processing of big datasets. The divide between processing power and cost-effectiveness needs addressing.

This research sets out to tackle these challenges by combining the latest machine-learning technology and a distributed data collection/analysis framework.

### 1.2 Proposed Solution

This research proposes a solution to the traffic classification problem, which is a new-concept firewall extension system that we will call "Soft-margin firewall" (SMFW) for the rest of this paper. The system uses machine learning and TA to classify traffic that cannot be easily classified by a rule-based firewall. The word "Soft-margin" refers to the way the firewall classifies traffic. While a rule-based firewall can classify each packet with no ambiguity (within its capacity), an SMFW must base its decisions on statistical features extracted from a number of packets. Therefore, some packets might avoid the classification. In exchange, an SMFW could detect higher-order traffic patterns that a rule-based firewall cannot, allowing more fine-grained knowledge gathering and policy enforcing on the traffic.

Aspects of the system are designed to address the challenges that traffic classification systems are facing. The traffic classifier is based on TA methodology, which works on types of traffic that are difficult for a rule-based firewall system. Targeted traffic types include encrypted traffic, tunneled traffic and aggregated heterogeneous traffic, as well as compound traffic in which any aforementioned conditions can happen at the same time. The classifier uses the Deep Neural Network (DNN) and other machine learning tools to accurately classify traffic.

The proposed system is also specially designed for classifying heterogeneous traffic. Having a mixture of application layer protocols in the traffic stream increases the chance for the traffic to be misclassified. With a novel two-stage classifier design, only the part of the traffic that looks homogeneous enough is classified. The architecture reduces the false positive rate drastically, while still being able to use a considerable amount of traffic for classification.

Lastly, the implementation of the system is an extension to the *iptables* firewall framework, which is an integral part of the Linux operating system. Since Linux is widely used in networking devices, an SMFW probe can be easily plugged into an existing network infrastructure. The extension only takes a minimal amount of data from the traffic, and therefore can scale to high network load situations. The system can off-load computationally intensive data processing such as feature extraction and training the neural network to a computing cloud so that the end devices can remain cost-effective. *Iptables* also provides ways to interact with traffic, such as blocking, throttling or prioritization, once the SMFW makes a decision.

### 1.3 Main Contribution

The main contribution of the research is three-fold. It proposes the concept of Soft-margin Firewall as a solution to the traffic classification problem. On top of proposing the idea, it also demonstrates its feasibility by showcasing a prototype implementation. In this work, we also reevaluate several state-of-the-art Traffic Analysis techniques against network traffic in contemporary settings. The evaluations especially target video streaming traffic for its prevalence and lack of representation in literature. The goal is to find ways to extend the capability of the existing TA method to work with video streaming protocols. The limitation of existing methods when classifying heterogeneous traffic is also explored, to provide a clue for better system design. These evaluations either result in the method being incorporated into the proposed system, or

improvements being made in the system design. Evaluations of two novel traffic feature extraction methods proposed by the author for the system are also present. One method is evaluated against conventional classifiers, while the other is proposed for using a deep neural network (DNN) based classifier that achieves better noise robustness and capable of multi-label classification.

### 1.4 Organization of the Chapters

The rest of the paper is organized as follows. Chapter 2 reviews the existing body of research on Traffic Analysis, Video Streaming protocol, firewall systems, and machine learning technology, which form the basis of the research. Chapter 3 and Chapter 4 discuss the usages of Packet Arrival Interval as well a newly proposed feature named Surge Period in video source identification. Chapter 5 explores the feasibility of the main goal of traffic classification using existing features. Chapter 6 discusses the performance of a single-stage classifier for source identification. Chapter 7 introduces the concept of two-stage classification, which improves the performance of source identification on mixed-traffic streams. Chapter 8 introduces the DNN-based classifier and highlights its unique capabilities when working with mixed traffic and multi-label traffic data.

# Chapter 2 : Related Works

## 2.1 Introduction

This engineering solution to solve the problem introduced in Chapter 1 is built on top of techniques developed in several research fields including TA, machine learning, deep learning and firewall technology. A review of relevant technologies in those fields will be provided in this chapter. In particular, development in TA technique is reviewed detail. To provide pointers for improvement, we focus on the technical challenges that TA has to overcome before it can be used in practical scenarios. Reviews are also provided for the intended targets of the solution, namely VPNs and video streaming protocols.

## 2.2 Firewall Systems

### 2.2.1 Types of Firewall

A firewall is a network security system that monitors and controls the traffic coming in or going out of a certain domain [10]. Firewalls operate on top of pre-defined rules. The firewall matches traffic with these rules, and execute corresponding actions when it detects a match.

Firewalls can be divided into 2 major categories: network layer firewalls and application layer firewalls.

Network layer firewalls work at the network layer of the Open Systems Connection (OSI) model [11] and handle packet-level traffic. A packet is a small chunk of data with the information required for routing it to its destination (which is the network layer header) attached. It is the data unit of the networking layer. They work at a lower layer than the application layer firewalls. As such, their understanding of high-level protocol information is limited, but they make faster per-packet decisions. For a network layer firewall, the concept of “traffic flow” is a series of packet

exchanges between two ports (a port is an identifier for an application running on a host) on two network hosts.

Application layer firewalls work at the application layer of the OSI model. Their data units are higher in level (a file, a stream, etc.). As a result, they could match traffic based on more high-level protocol information than network layer firewalls. For example, Deep Packet Inspection (DPI) is a popular subcategory of application layer firewalls [12] that saw many applications.

### 2.2.2 Netfilter based Firewall

Netfilter [13] offers support for packet filtering in kernel-mode on GNU/Linux operating systems. It forms a firewall solution that is widely used in Linux based networking systems. Due to its popularity among embedded Linux systems in home and small-business routers, Netfilter is available on most of the devices that the proposed firewall targets. This makes Netfilter an ideal platform for the proposed firewall. Netfilter provides an interface for third-party drivers to extend its functionality, a function that allows highly customized processing of packets. This extensibility is the key to making the prototype implementation.

### 2.2.3 Cisco NetFlow

NetFlow [14] protocol supports the transmission of traffic statistics to remote data aggregators, letting network administrators collect data from many routers for analysis at a central location. This architecture of NetFlow, which separates routers and data aggregators, is a good fit for the proposed SMFW system. The author chose to base the prototype implementation of the firewall probe on an open-source implementation of Cisco's NetFlow protocol is provided by the OpenWall Project.

## 2.3 Privacy Enhancing Networking Protocols

### 2.3.1 Virtual Private Network (VPN)

A VPN is a means to connect isolated private networks through the public Internet [15]. VPN keeps sensitive information private by transmitting the network traffic through a secure tunnel. The tunnel ensures that the message body and the real source/destination information are encrypted between the 2 ends of the tunnel. The tunnel can be created on top of public networks, where it is possible for adversaries to tap into the traffic yet unable to decrypt the information or reveal the actual source and destination.

There are many variations of VPN in terms of the tunneling protocol a VPN uses to create a secure tunnel. Two of the popular protocols in use are Internet Protocol Security (IPsec), and the Secure Socket Layer/Transport Layer Security (SSL/TLS) [16]. OpenVPN [17], the VPN solution used in this work, supports an SSL-based tunneling protocol.

Video streaming traffic tunneled through VPN services is difficult to identify because of the added obfuscation, padding, encryption, packet fragmentation, etc. Different VPN solutions may have different ways of obfuscating traffic. Combined with the diverse types of video streaming protocol, a significant variation could be introduced into the traffic stream by these operations. In [6], the authors survey several obfuscation techniques and categorize them into four groups: encryption, randomization (make traffic behave like random data), mimicry (masquerading the traffic to look like another protocol) and tunneling (embedding traffic as the payload of another protocol). VPNs belong to the tunneling group, but they can also employ techniques from other groups to further obfuscate traffic.

### 2.3.2 The Onion Router Network (Tor)

Tor is an anonymous routing protocol/network that hides client identity by routing a packet, encrypted with certificates from multiple routers one after another, through the chain of routers while each router decrypts the data only to reveal the next hop in the routing chain. Only at the exit node can the packet be restored and routed accordingly. Because Tor changes the route randomly it is hard to trace. The packet that Tor creates has an onion-like nested encryption structure. One layer of encryption encloses another, making the payload at each stop. To further confuse TA attackers, Tor also randomly changes routes every 10 minutes, Tor pads data into 512-bit cells and tries to hide the trace of HTTPS handshake by pre-establishing the encryption keys to separate the process from actual traffic. Tor browser is a modified Firefox browser with HTTPOS-like capability. It is used with Tor router to combine several TA counter-measures together in order to provide maximum anonymity. A depiction of a Tor virtual route and how the packet is structured is shown in Figure 2-1.

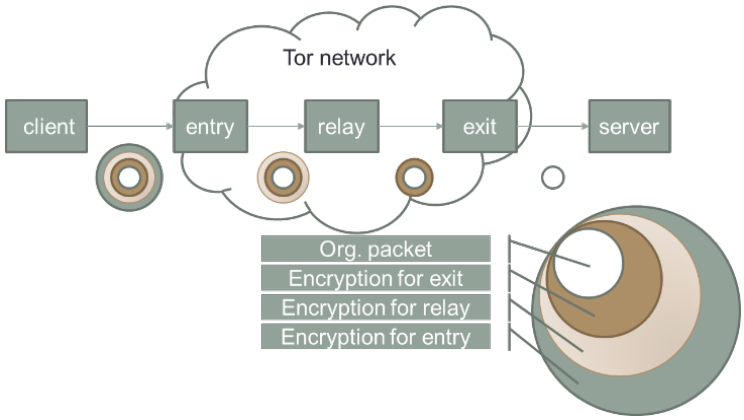



Figure 2-1: Onion Router Network

### 2.3.3 Traffic Obfuscation

An example of obfuscation technology is Obfsproxy [6], which is a pluggable obfuscation layer that is developed by the same team that develops Tor. Obfsproxy extends the traffic mimicry

functionality of Tor, which makes its traffic look like SSL traffic, to be able to masquerade Tor traffic as a wide range of protocols. Obfsproxy makes the identification of traffic more difficult by padding packets with misleading information, changing delays between packets, and inserting dummy packets into the traffic flow. To make the study more focused, we choose OpenVPN, which is widely used by inexpensive personal VPN services as well as corporate VPN services, as the target platform [16]. OpenVPN does not have additional protocol obfuscation besides tunneling. The concepts proposed in this work can scale to other streaming protocols and VPN/Proxy mechanisms.

### 2.4 Traffic Analysis

Traffic Analysis (TA) is a machine-learning based technique to inspect network traffic and determine its nature, such as application layer protocol, traffic source, and destination, among other properties. TA works with metadata such as packet direction, packet size, packet count, and timing. Interesting pieces of information are then extracted from the traffic and combined into a feature vector in the feature space. A classifier is then trained on this feature space to distinguish between various classes. The classifier can then classify unknown traffic trace and assign it to one of the classes.

Since totally masking traffic activities is not practical (it will require the channel to be active at the maximum capacity constantly to hide the activities), TA is effective in situations where extracting information from encrypted and obfuscated traffic is the goal. TA is proven to be effective in identifying security-enhancing protocols such as the Onion Router (Tor) protocol[3], [7], [18] and other commonplace protocols including FTP, IMAP and HTTP[19]. The work of Hermann et. al. [2] is an early piece that established that good recognition can be achieved when

using a conventional VPN system. Panchenko et. al. [3] target the Tor protocol with a combination of features, including traffic amount, packet count, packet size among others, and achieved a significantly higher recognition rate (55%) compared to previous works. This is considered a breakthrough and helped popularize Tor as a target for further TA based researches, including[7], [18], [20]. At the same time, other researchers expressed concern about the practicality of TA attacks, since most proponents use idealized scenarios that are not representative of the real-world traffic[8]. All of the above works assume the role of an attacker when performing TA, and the targets are usually specific web pages, since revealing the browsing history of the target is the attacker's goal.

On the other hand, there are relatively few researches on streaming video source identification. In[21], the authors present a TA based framework for recognizing video content from streaming traffic. Other works from the same researchers[22] have used a wavelet-based feature to identify video content. The application of both methods is limited because the video content needs to be known, which is an unrealistic scenario for an SMFW configuration. In [23], the authors present the detection results on the traffic data for several popular video streaming sites with various combinations of classifier and features. In [24], several different aspects of the feature set are explored. It identifies Packet Size Distribution as one of the effective classification features for identifying protocols used by tunneled video streaming traffic. It also shows that for the same streaming protocol in a secure tunnel, there exists enough difference between various video streaming sites that permit a classifier to separate them. The work in[25] explores the *packet arrival interval (PAI)* as a feature for video streaming source identification. A detailed analysis of the PAI feature in[25] also revealed that the network path for traffic flow could leave a signature in PAI. The results in[25], [26] pave the way for site-based TA applications, as attempted in this

work. The classifier in this paper is designed based on the previous results described in the aforementioned works. It uses the classification method that was proven to work well in this context, and the feature set will include the features that were proven to work well (packet size and PAI).

Another issue with previous works on TA is that they assume homogeneity in network traffic. In other words, the traffic flow in the tunnel is assumed to contain traffic of only one protocol type from one source to one destination. The reality of tunneled traffic (e.g., through encrypted VPN or Proxy server) is that it often contains traffic of different protocol types from different sources, and to different destinations. The mixture of traffic can alter features significantly and is identified as a limiting factor in the real-world applicability of TA[8]. This mixed-flow problem, which to the best of our knowledge has not been addressed by other works, is the target of this work. Mixed (heterogeneous) traffic in a tunnel is defined as traffic flow in which there is a video streaming flow coupled with an unspecified volume of non-video traffic in the tunnel. Pure (homogeneous) traffic flow in a tunnel is defined as traffic flow in which there is just one video streaming traffic flow. Traffic that contains more than one video streams is excluded from the scope of this chapter but will be addressed in future chapters.

We propose 2 solutions to the mixed-flow problem, the first one is based on classic machine-learning techniques, which is based on the classifier documented in [27]. It is a 2-stage classifier that adds a pre-filtering stage to mitigate the effect of mixed-flow on classification. It is demonstrated to be able to classify with high confidence in the presence of BitTorrent Traffic and Video Streaming Traffic, at the expense of having fewer usable samples due to the pre-filtering. In this work, the design of the 2-stage classifier is significantly augmented with new methods for feature selection (i.e., Correlation Feature Selection (CFS)), and various forms of mix-and-match

training of both the classifier stages. Second, a new, much larger dataset that consists of web traffic and BitTorrent traffic is included to evaluate the performance of the classifier architecture. Third, new architecture level performance indices are developed for generalized evaluation of the two-stage classifier architecture. Finally, Packet Size Interval (PAI) is analyzed in detail to reveal its contribution in the feature-set for enhancing the performance of source identification.

The second solution is a DNN based classifier. The rapid development of deep learning has attracted researchers to apply it to TA problems. The features used in deep-learning are much higher in dimensionality, and technologies like Word Embedding helps handle even higher dimensional data with no need for manual feature engineering. The increased resolution also means that it might be possible to identify individual streams within a heterogeneous flow. There is potential in using the deep-learning technique to solve the feature engineering problem and the mixed-flow problem at once.

Researchers have taken on the challenge of applying the deep-learning technique to traffic analysis problems before. Existing researches can be divided into 3 groups regarding how the features are constructed. The first group contains researches that use purely flow-based, statistical features, like the ones used in traditional TA studies. They are a natural progression from using classical Machine-Learning technology but fails to exploit the unique capability of DNNs. Such studies include [28], [29]. The second group is the researches that focus on analyzing the packets. The features used in this group are commonly directly based on the packet payload, and therefore can be regarded as a natural extension of deep-packet investigation with deep learning. Deep-packet is a famous example here [30]. Researches in this group are less applicable to the scenario that this work aims to tackle since the analysis of the packet payload is ruled out by the presence of the encrypted tunnel. The third group is where new flow-based features are developed to take

advantage of the feature selection capability of DNNs. There are few works in this group. In [31] the researchers extracted a time-series of statistical features from the traffic flow. Each point in the series represent 3 seconds of traffic flow, and the resulting time series is used to train a multi-layer LSTM network to classify the traffic streams between the 2 protocols (BitTorrent protocol and plain HTTP protocol) tunneled through an encrypted proxy. The experimental results show that the new features could yield 92% accuracy, which rivals the state-of-the-art non-DNN based methods.

The proposed DNN method in this work is based on several widely-used DNN architecture on top of a novel natural language-inspired feature which the author created with the intention of solving the mixed-flow problem. This feature allows the classifier to exploit the temporal characteristic of the traffic flow. For more detailed information on developments in deep-learning based Natural Language Processing, which gave the inspiration and tools to the proposed method, please refer to Chapter 8.2.

The idea of integrating traffic analysis capability into firewalls has been realized in commercial products before, but existing implementations focus more on the basic task of performing protocol identification on encrypted traffic. Applying fine-grained classification faces numerous obstacles, the most significant being that home/small business routers do not have the computing power to carry out the analysis. To address this, the proposed solutions assume that the devices have access to cloud-based computing power to perform sophisticated analysis. The separation of operation and analysis is the key to keep the deployment cost of the technology low. On the other hand, this separation also demands a novel feature format that is efficient to compute and transmit to avoid adding overhead to the network. Both of the solutions are proposed with the

efficient implementation in restricted environments in mind, as will be detailed in the following chapters.

### 2.5 Video Streaming Protocols

Streaming protocols are networking protocols designed to ensure good streaming video quality under fluctuating network bandwidth limitations. It is important to the proposed solutions since the protocol has a significant effect on the features of the traffic flow. The most important change in the field in the last 10 years is the rising popularity of HTTP adaptive streaming (Figure 2-2). HTTP adaptive streaming transfers video data and control data through the HTTP protocol rather than a special-purpose one. At the beginning of the 2010s, the video streaming protocol that comes with Adobe Flash, the Real-time Messaging Protocol (RTMP) [32], is used by most of the providers due to the dominant position Flash had in web development. Microsoft's Smooth Streaming [33] (which is an HTTP adaptive streaming protocol) is used by only 2 providers (Amazon and Netflix). By 2017 however, Adobe Flash is not widely used anymore, and HTTP adaptive streaming dominates the scene. This change is due to several reasons: 1) the HTML5 standard that HTTP adaptive streaming is based on makes both development and deployment easier, since the application will work on virtually all mainstream browsers; 2) The protocol enables the providers to utilize existing HTTP servers to deliver video content, which saves cost; and 3) The developer tools (for example, Javascript frameworks) are vastly improved during this time period, which also makes development easier.

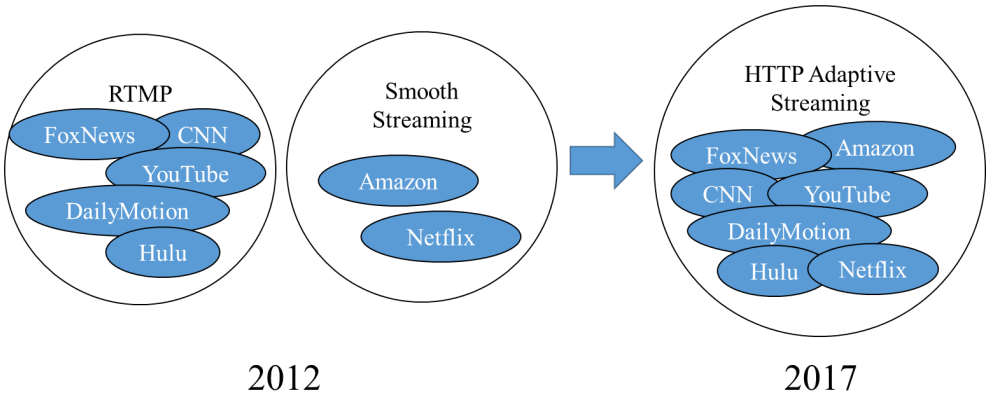



Figure 2-2: Evolution of Video Streaming Protocol Adoption, 2012-2017

As the following chapters will show, the proposed methods work under all protocols. Experiments in Chapter 3-Chapter 5 feature RTMP and Smooth Streaming, much like the "2012" part in Figure 2-2, while experiments in Chapter 7 and Chapter 8 feature datasets that look like the 2017 part. Experiments in Chapter 6 were conducted in a state when YouTube was still using RTMP. The TA technique is therefore proven to be robust against changes in protocol.

### 2.6 Summary

The works reviewed in this chapter are crucial for the development of the proposed systems. We design the systems with the intention of eventual deployment in a firewall that can analyze the diverse and mixed traffic types in today's network. The existing firewall technologies provided tools for such a development. The solution is a TA based traffic probe built on machine-learning and deep learning technologies. However, a new TA technique has to be developed to address the critiques that the existing ones received before it can be used practically. These reviews provide pointers for the following chapters to investigate and build upon. Finally, we focus on widely used VPN systems serving the dominant traffic type on today's Internet, the video streaming traffic, as a good target to test the system.

# Chapter 3 : Source Identification of Dynamic Web Traffic

## 3.1 Introduction

Website Finger Printing (WFP) is a form of Traffic Analysis that is widely researched since it has many administrative applications, including preventing access to forbidden websites and site-specific Quality of Service (QoS) provisioning. Previous works in this area mainly looked at the problem of identifying traffic from relatively static websites, thus limiting the applicability of the technique for websites with dynamically changing contents. In this chapter, we attempt to generalize the mechanism for dynamic sites by the way of introducing a new classification feature traffic surge period and adapting the first n Components of Haar Wavelet Transformation, which is commonly used in traditional signal processing applications. The results in this chapter prove the newly proposed features to be superior when classifying web pages with dynamic content.

## 3.2 Experimental Setup

In the experimental set up in Figure 3-1, a web client is used from home to browse different servers through an SSL/TLS VPN service provided by one of Michigan State University's Juniper VPN servers. All packets are encrypted through the VPN tunnel which terminates at MSU's VPN server. The server interprets the packets sent by the client through an SSL session within the tunnel and forwards them to the intended target web server. When the target server responds, the VPN server uses the same SSL session for sending traffic back to the client. From the client application's perspective, the content of the web page comes from the target server.

The SSL/TLS tunneled traffic is probed at an egress router by the traffic capture software Wireshark. At a probing point, packets do not have any destination-specific identifiers/addresses since they are embedded into the application layer header. To separate the traffic stream, it is assumed that the client IP address as well as the VPN server IP address is known to the probing

entity, which is consistent with the QoS provisioning or site blocking applications as mentioned before. The clients run Internet Explorer *V.11* in a windows 7 machine to access the target web pages. Note that this arrangement also emulates scenarios in which probing is done by an ISP when it intends to perform website blocking or site-specific QoS allocation based on traffic analysis based website fingerprinting.

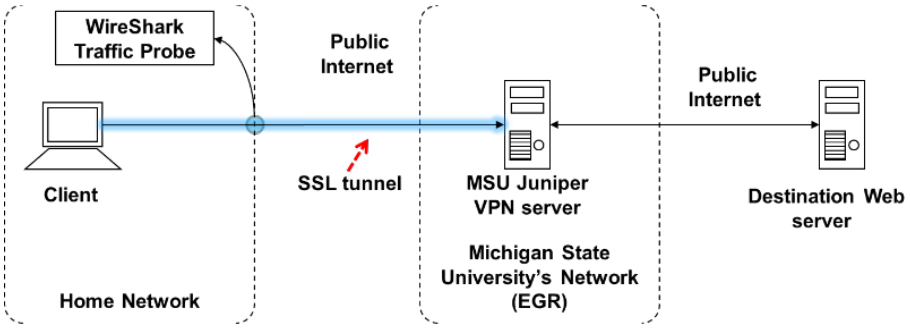



Figure 3-1: Experimental traffic probing arrangement

Traffic is collected for 11 popular website front pages that represent dynamic content. Those are listed in Figure 3-2. This group of websites is referred to as the experimental group. Content dynamism is ensured by accessing personal home pages with enough interval so that the home page changes significantly in terms of text and image as parts of newsfeeds, status updates, etc. Quantification of such intra-site dynamism is presented in Section *3.5*. Data collection is done through a time span of 10 days, with an average of 30 samples of each page being taken a day. Consecutive data collections are separated by at least 20 minutes to ensure that the web pages are sufficiently different. During the data acquisition phase, we collected 3250 valid samples. The 11 sites chosen comprises of major classes of websites with dynamic contents.

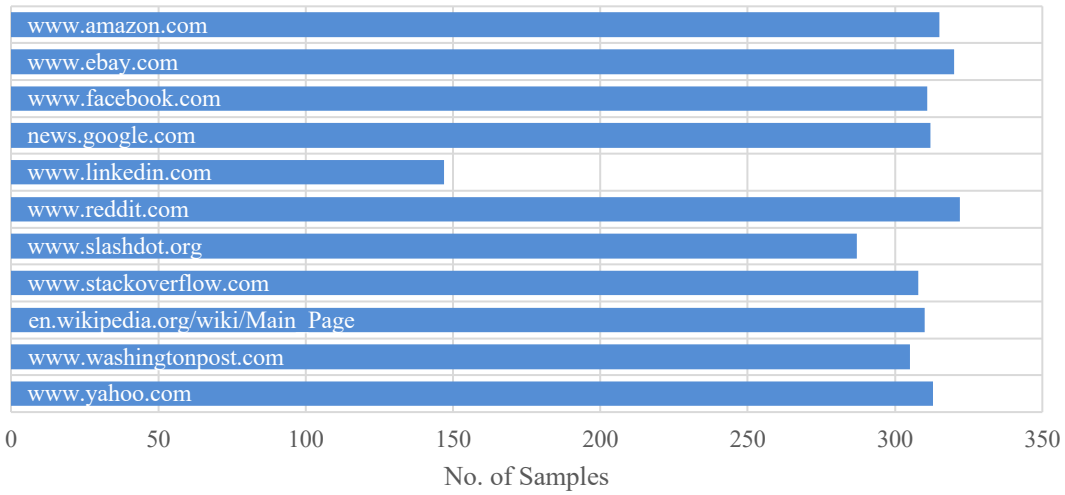



Figure 3-2: Target dynamic websites as the experimental group

As for the control group, we also collect data for several static web pages as shown in Figure 3-3. The web pages in the control group don't change much over time. Traces from the control and the experimental groups are collected under identical conditions. In Section 3.5, we will assess the impacts of dynamic pages on WFP performance with respect to the static control group.

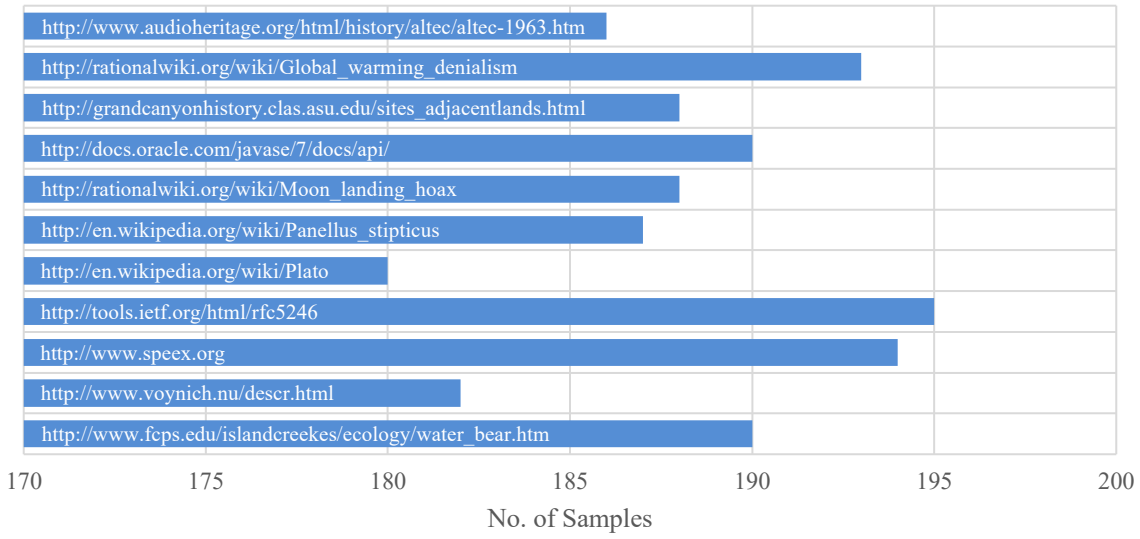



Figure 3-3: Target static websites as the control group

### 3.3 Traffic Pattern for Web Transactions

A typical web transaction is initiated by a request from a web browser to a web server, which replies to the request with objects that are parts of a web page via HTTP protocol. In the beginning, the client sends a request including the URL of the web page and any parameter needed to server. The server responds with the main textual content of the web page in Hypertext Markup Language (HTML) form to the client. The client/browser then parses the main content, discovers a reference to any additional resource needed to show the page, and sends one request for each resource. The server responds in the same order as it receives the requests. This process continues recursively until all references are responded to, and the transaction concludes.

There are several additional complexities that are added due to the following optimizations in the transaction process. First, at the client side, the download and parsing often go in parallel for improved viewing responsiveness. This often means that requests are usually sent out as soon as possible when a reference is revealed by the parsing process. This behavior also has the implication that the timing of the request should indicate its position in the HTML text stream.

Second, for higher performance, a client can choose to establish several HTTP connections to download resources in parallel. Also, if the server supports HTTP pipelining, then the client can send several requests in one TCP datagram. The server can also respond to such requests in one batch. These optimizations can interfere with the order of request/response pairs. The timing of requests can be mingled with response data from other connections. Third, the underlying network conditions can change the shape of traffic, especially in terms of the inter-packet delay jitter. This means that the timing of each burst can change because of different download times.

Fourth, a TCP connection can be reused for multiple requests in order to avoid the delay involved in successive connection setup and termination. This means that it is not safe to assume

that one connection corresponds to one resource. If the protocol is plain HTTPS, then a connection can still tell actual object sizes transferred, while if the protocol is modified by the VPN client, for example distributing one resource among multiple connections, even connection-wise burst sizes can be unreliable. If encryption does not change traffic signature drastically, all the above optimization related traffic pattern changes apply to encrypted web traffic within a VPN tunnel.

Figure 3-4 shows an example of 4 sec. long traffic trace for a transaction from the LinkedIn web server, collected using Wireshark as shown in Figure 3-1. Decomposition of the trace was performed as follows. In the figure, the stripe to the left marks the timing of different HTTP requests sent by the client to the server. The middle plot area is divided into several lanes, and each TCP connection used in this transaction is plotted in one lane. One lane is further divided into 2 halves vertically, with the left half used for upstream traffic and the right half for downstream traffic. Dark stripes mark the point in time where the connection is downloading. It shows how requests are processed by different TCP connections (numbered #1 to #8) and how the traffic of different connections interleaves. To the right, the traffic of all TCP connections combined is shown. This example trace shows the complexity of the underlying process and the challenges of finding traffic signature patterns in the presence of deviations from the baseline transaction process.

### 3.4 Feature Definition and Extraction

Pre-processing of the probed traffic is performed following the approach in [3]. Pre-processing removes the non-TCP and pure ACK packets that might affect the results.

### 3.4.1 Existing Features

The term *trace* is used to describe a time-stamped sequence in which every packet is associated with the time it appears on a traffic probing point. It can be represented as a series of timestamps as. We assume that the content of traffic is generally encrypted and that only the direction and packet size are of importance. Following the convention in [2], we combine the packet size and direction into one scalar. The absolute value of the scalar equals the size of the packet. All upstream packets have negative signs and downstream packets have positive signs. This series of packet sizes is defined as . The following features have been used in the literature.

Burst Size: In [18], a burst is defined as the interval during which there are only packets in one direction, but preceded and followed immediately by packets in the opposite direction. There are upstream and downstream bursts. In Figure 3-4 upstream bursts are shown on the negative half of the graph while downstream packets on the positive half. Each time the stream switches direction from the negative half to positive half or vise-versa, a burst boundary is recorded. Each burst is marked by 2 boundaries and the direction of packets inside that burst becomes the direction of the burst itself. Burst is also used in[3], albeit named differently. It is used as part of the feature set in two different ways. 1) The number of bytes transmitted in each burst is summed up into a histogram and contained in the feature. 2) The number of packets transmitted in each burst is summed up into a histogram and contained in the feature.

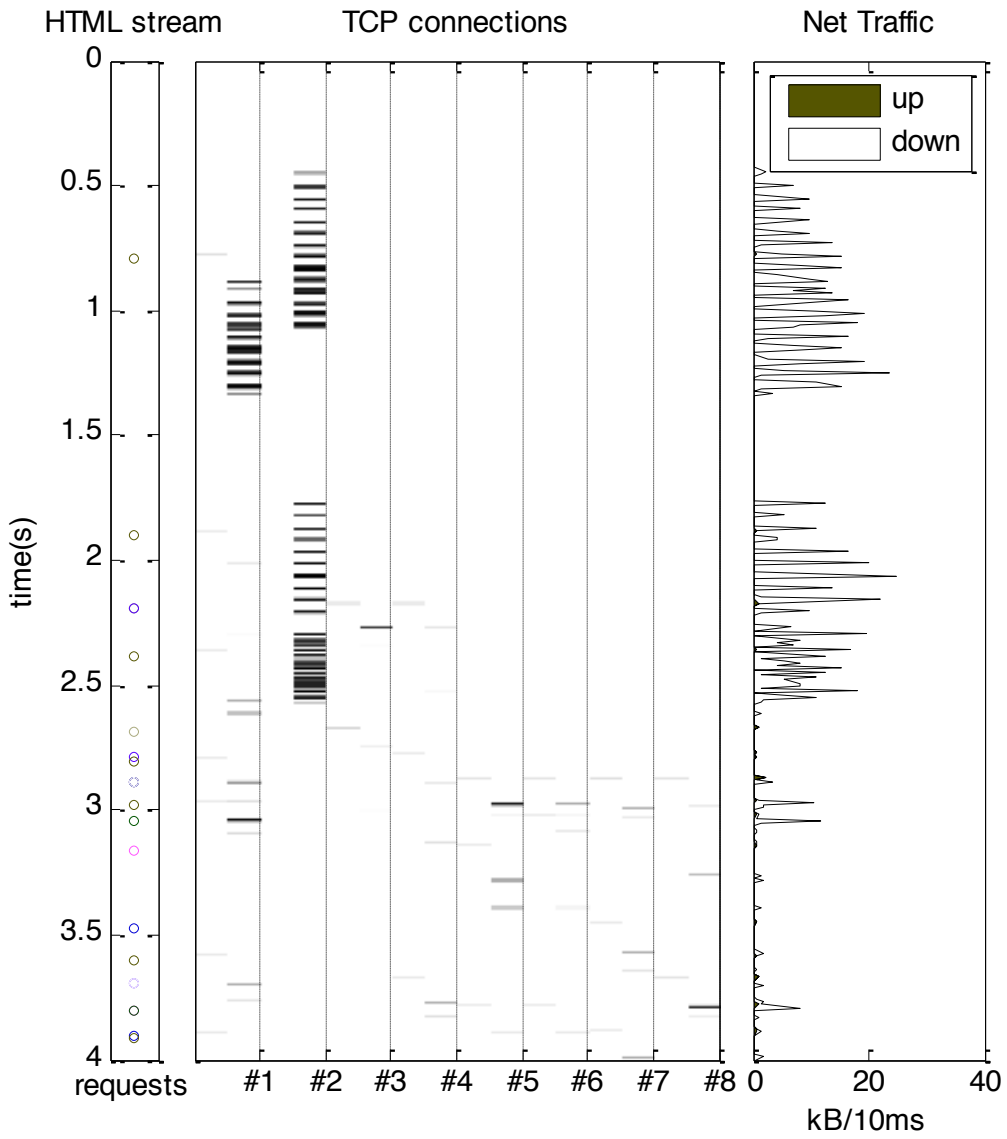



Figure 3-4: Decomposition of the trace from an example transaction

First *n* Components of Haar Wavelet Transformation: Haar wavelet transform is often used in the signal analysis to analyze local information of a signal. We use this as a feature to represent coarse-scale variation information of a trace. Each trace is first converted to a bit-rate series using a 100ms time window, then interpolated to 4096 data points before transforming. We take the first 15 components of the transformed vector as the feature.

### 3.4.2 Surge Period

We propose a new feature, termed as *Surge Period*, which unlike the packet direction-based bursts, is defined based on the timing of surges in traffic density, or high bandwidth utilization. A *Surge Period* marks the parts of traffic trace where the channel is busy transmitting packets upstream or downstream with back to back packets. Any packet except for the first one and the last one within the surge period should be separated from its predecessor and subsequent packets by a time period no larger than a pre-defined time window size. The window size depends on the network condition and typically ranges from 10 milliseconds to several hundred milliseconds.

During web transaction, the timing of a *Surge Period* corresponds to one object or a group of objects being downloaded together. We have speculated that, on a coarse scale, the timing of a *Surge Period* corresponds to the location of those references that resulted in it. This is because modern browsers want to render the contents with the highest speed possible and that they should send requests for resources quickly after the references are discovered by the parser.

As demonstrated in Figure 3-4, while the probed data can be decomposed into individual TCP connections, the *Surge Period* feature is extracted from the combined traffic pattern (the far-right column in Figure 3-4). This is done so that the feature should be prepared for situations, such as Tor [5], in which the traffic cannot always be separated into individual connections.

As done in[2], [3], we assume that all traffic from or to a browser client is from the same transaction. Meaning, we ignore the interleaving of multiple web transactions.

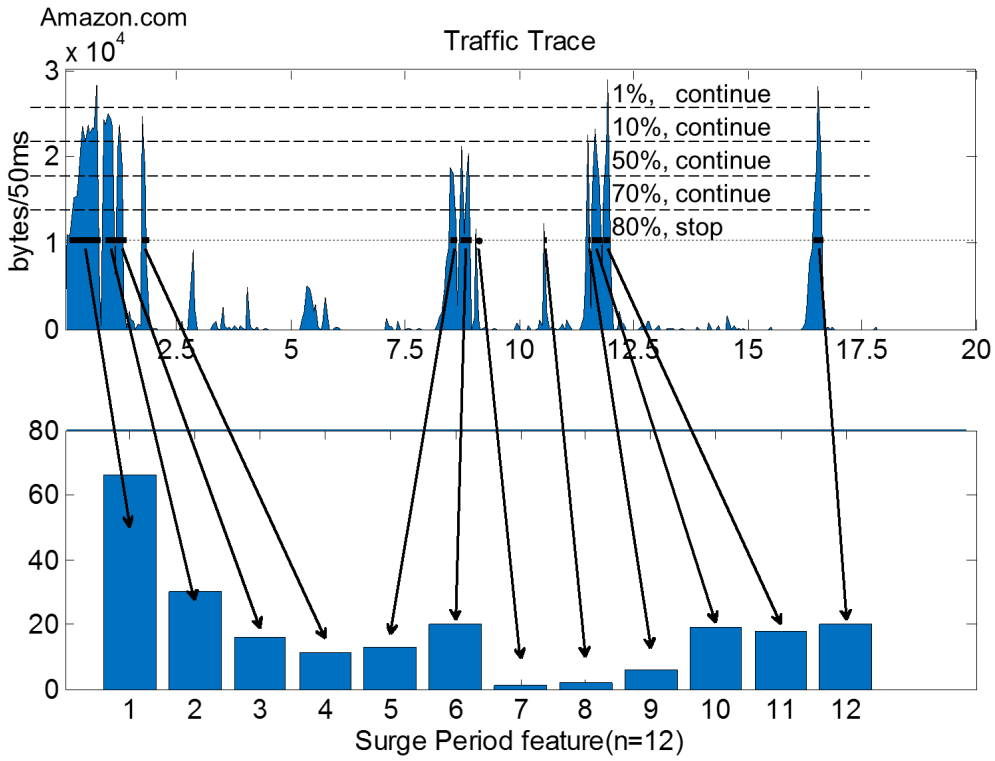



Figure 3-5: Example of *Surge Period* for amazon.com traffic

The *Surge Period* feature is extracted as follows. First, a probed trace is converted into a time-stamped series of (time, packet-size) pairs. Second, the time-stamped series is converted into a bit rate time series computed over 100ms non-overlapping windows. Third, the most significant *Surge Periods* are then extracted from the bitrate time series. To do this, we apply the following adaptive method. A continuous period of bitrate higher than a certain threshold is counted as a *Surge Period*. Multiple thresholds are experiment with, starting from the highest value possible, and gradually descending towards until a threshold where more than 80% of all the bytes transferred are covered by in burst periods. Finally, the number of bytes transmitted in a *Surge Period* is summed up as the 'size' of that period, and all *Surge Periods* are lined up according to their timing order. Top *N* periods in size are then chosen, where *N* is an arbitrary number. The

resulting vector of period sizes are used as the feature that represents the sample. If there is not enough number of *Surge Periods* even after  is tried, the vector is padded with zeroes at the end.

Figure 3-5 shows an example computation of the *Surge Period* feature on an actual trace captured from amazon.com traffic. The bitrate vs. time graph is shown in the upper half of Figure 3-5 with a descending threshold. When the threshold is lowered to a level where 80% of the traffic is included in various surges, the threshold is made fixed. With this threshold level, the surges are extracted and then rounded up to form the vector shown in the lower part of Figure 3-5. The arrows indicate the correspondence between surge periods and feature.

## 3.5 Performance

### 3.5.1 Feature Analysis

In this section, we present performance comparisons of classifying websites using different features. First, we demonstrate the dynamism of the dataset we collected by presenting sample probed traffic trace for web transactions corresponding to different destination sites. Figure 3-6 shows data rate over time during the progression of a transaction. Traffic in both directions is considered in the graphs.

It can be observed that the data rate signatures for different destination sites vary significantly in terms of their burstiness, the number of bursts, and the aggregated average rate. These variations in the traffic patterns translate into distinguishable features across various target websites, thus yielding good classification performance as reported later in this section. To depict dynamism of the pages to the same website, in what follows, we present intra-site feature variation by showing samples of burst sizes, Haar wavelet transformation components, and surge period features from various target sites.

*Surge Periods* of 2 sites are presented in Figure 3-7. One row in the graph corresponds to one site. The site above comes from the experimental dynamic group while the bottom one comes from the static control group. The feature can capture the intra-site variation, as well as the obvious inter-site variation between the 2 sites. Samples of Haar wavelet transformation components are presented in Figure 3-8. The distinction between 2 sites and their two access sessions are obvious for this feature.

Due to space constraint, all intra- and inter-site variations in Figure 3-7 and Figure 3-8 are shown only for a limited number of sites and samples. Similar variations were observed across all experimented sites and across all sessions for the same sites.

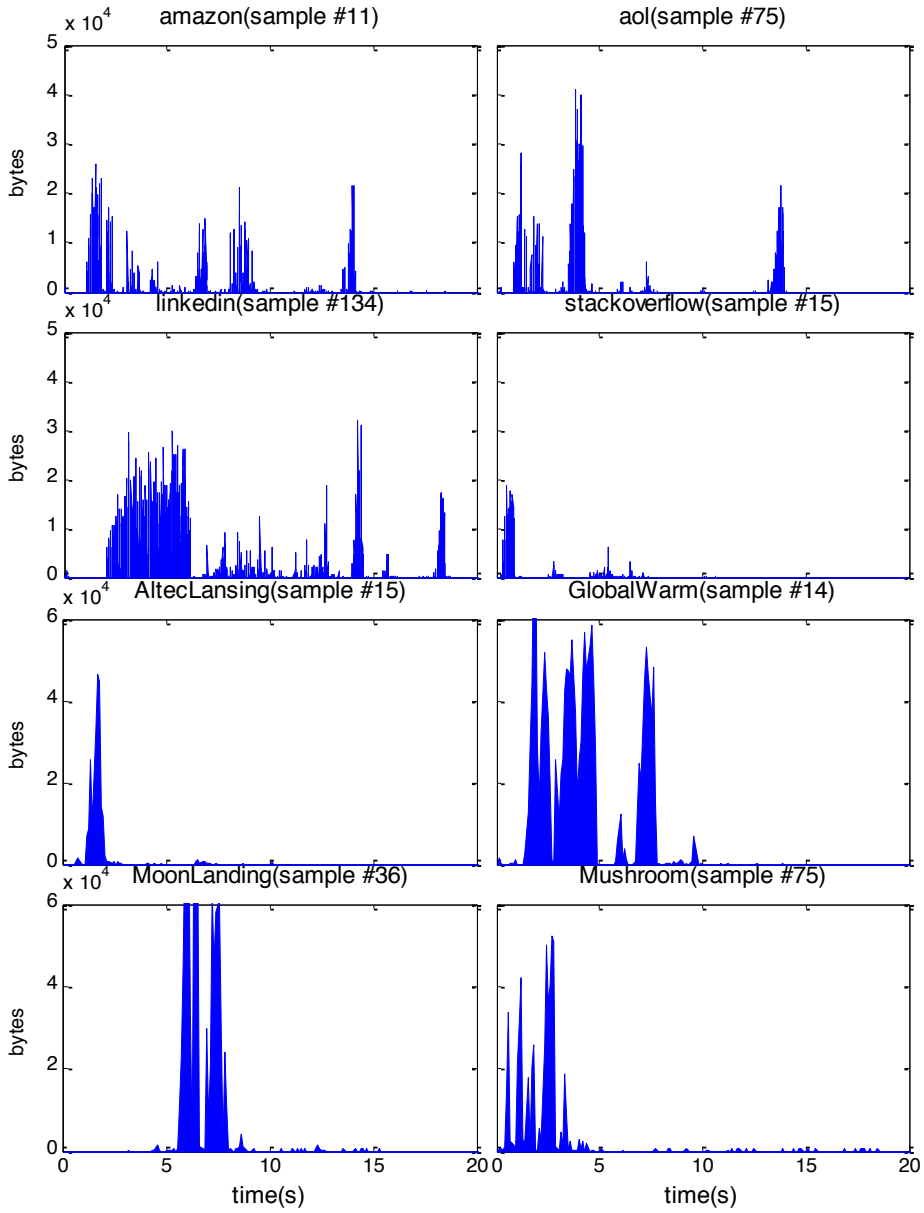



Figure 3-6: Traffic rate traces for web transaction to different sites

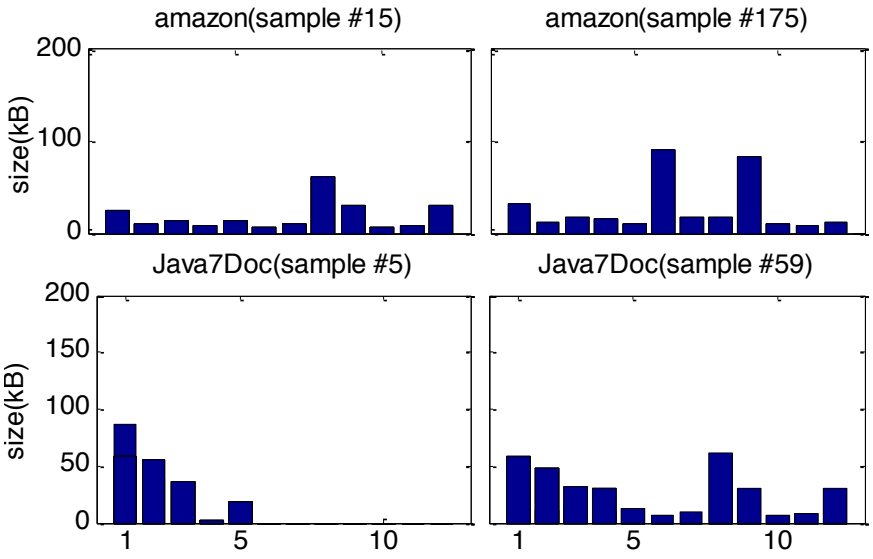



Figure 3-7: Surge Period for dynamic and static sites

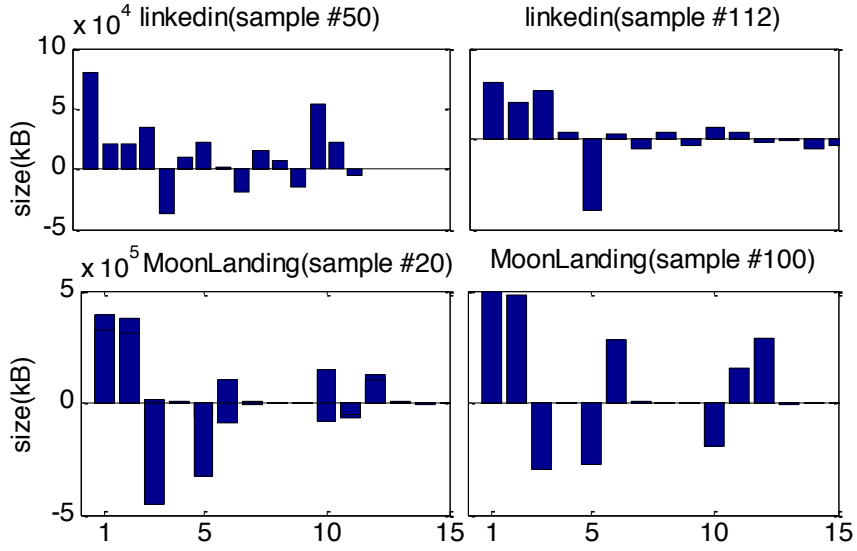



Figure 3-8: First 15 Haar components for dynamic and static site

### 3.5.2 Website Fingerprinting

We use the Bayesian network classifier (BayesNet) from the Weka [34] machine learning library for executing WFP. As shown in Figure 3-9, the classifier is trained offline with individual features from Table 3-1 or combinations of them.

The training is supervised, and features extracted from each trace are marked with a class label that represents the URL where the trace is from. There are 11 classes (i.e., URLs) for the experimental set and 11 classes in the control set. A full list of feature sets experimented with is provided in Table 3-1.

Table 3-1: Feature sets used in experimentation

| | **Feature Set** |
|---|---|
| 1 | Histogram of burst sizes |
| 2 | First *n* components of Haar wavelet transform (n=15) |
| 3 | *Surge Period (N = 10)* |

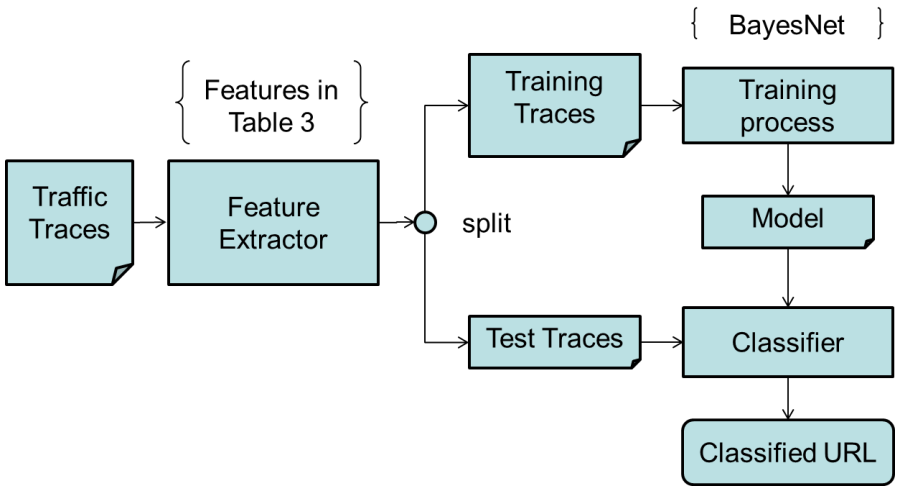



Figure 3-9: Website fingerprinting classification workflow

To test the performance after the training phase, we do a pseudo-random 2:1 training/set partitioning before the classifier is trained, then train the classifier with the training set and test it with the test set. The whole process is shown in Figure 3-9.

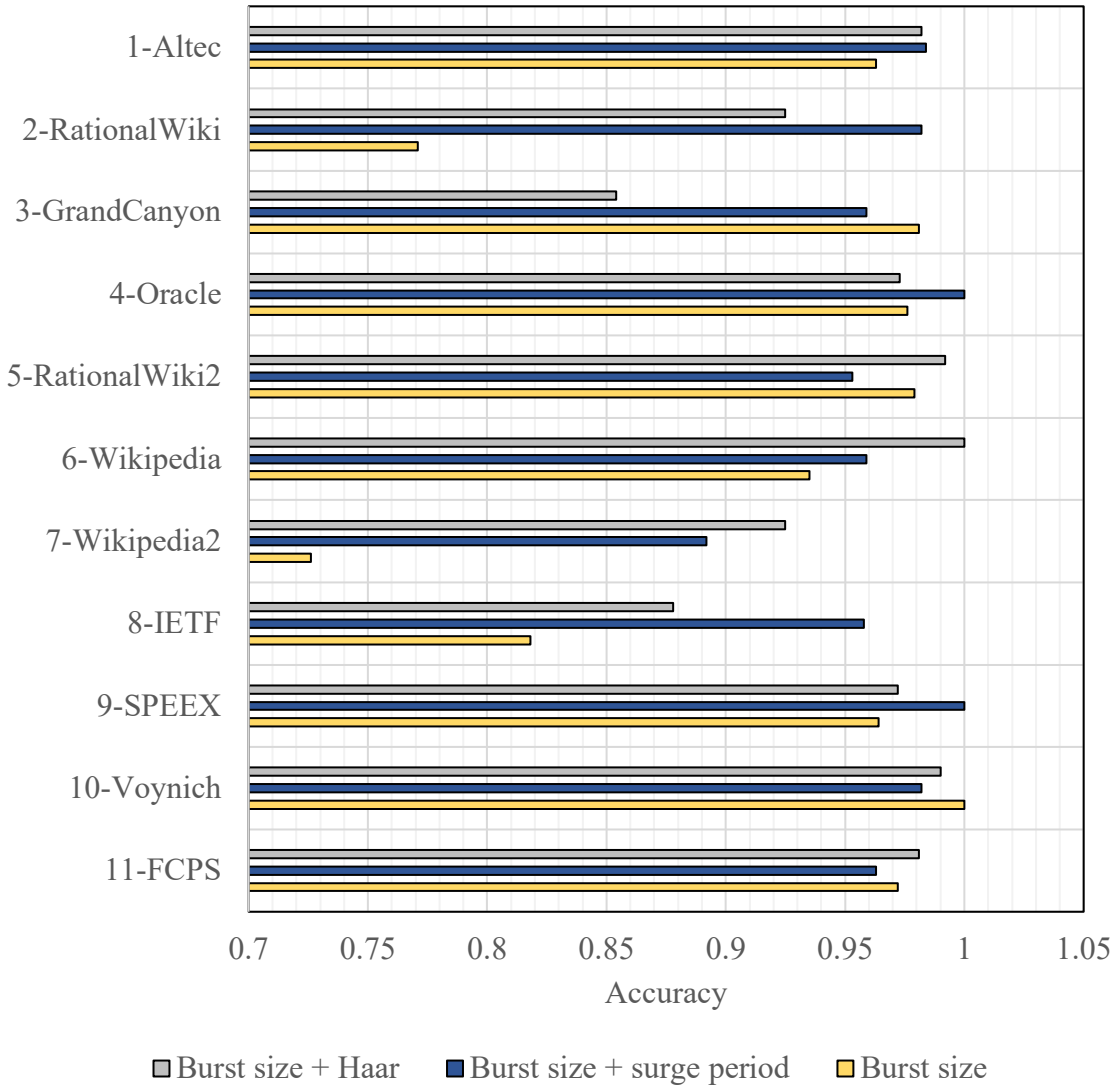



Figure 3-10: Class-wise true positive rates for the control group

Given their lower intra-site variations in the traffic signatures, it is expected that the classification results for the static sites would be better than the dynamic sites. We first experiment with the static sites with three feature scenarios, namely, *burst size, burst size plus surge period, and burst size plus Haar*. Class-wise results for the control group are shown in Figure 3-10.

Observe that all three features scenarios generally work very well for the static website fingerprinting. This is consistent with previous results published in [2], [3], [18]. The next step is

to apply the same feature scenarios for the experimental dynamic sites. Class-wise results for dynamic sites are shown in Figure 3-11.

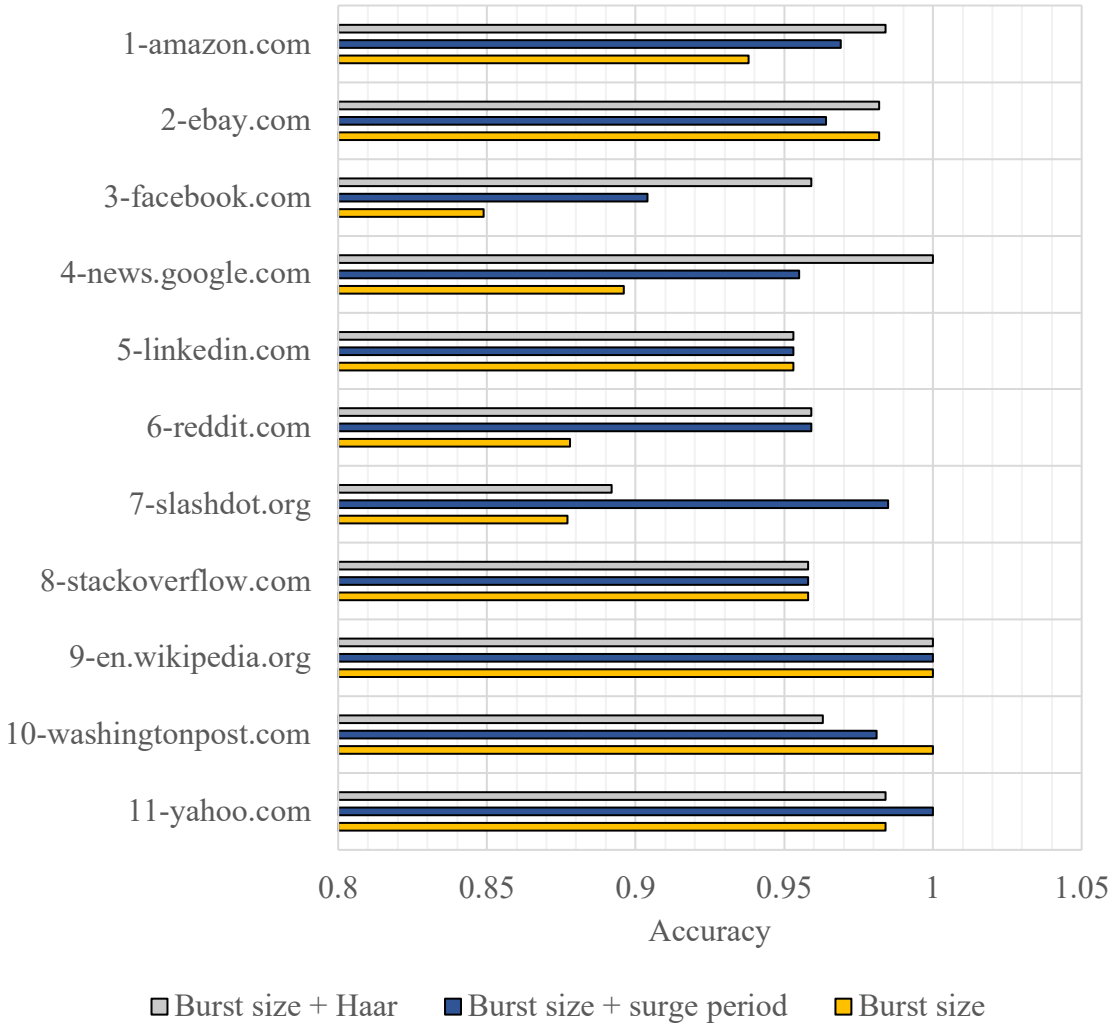



Figure 3-11: Class-wise true positive rates for the experimental group

It should be observed that when the traditional feature *burst size* is used alone, the fingerprinting performance degrades compared to the static sites as shown in Figure 3-10. For certain sites (e.g., sites 2 and 7), the loss of performance was substantial. The reasons for such loss are due to high intra-session diversity for the same target site as depicted in Figure 3-6.

Figure 3-12 shows results when the feature *burst size* is aided with our proposed feature surge size, and the components of the Haar transform. It is evident that these new features improve the overall fingerprinting performance by being able to characterize and identify the inter-site variation in probed traffic data.

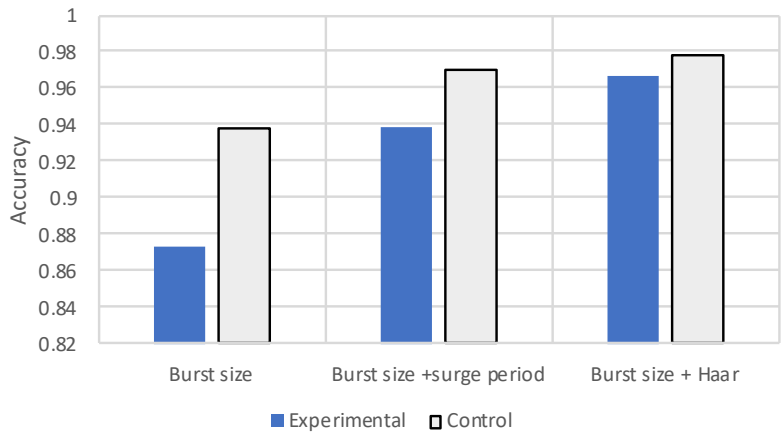



Figure 3-12: Average true positive rates for both the site groups

Figure 3-12 reports the overall true positive rates averaged over all the sites within both the groups. These figures confirm the general observations made for individual site-specific performance as reported in Figure 3-10 and Figure 3-11. The following two summary observations can be made. First, fingerprinting performance on static (i.e., control) sites are generally better than those for the dynamic experimental sites. Second, adding the new features, namely, surge size and the components of Haar transform, improves results for both static and dynamic sites, although more notably for the dynamic group.

### 3.6 Summary and Conclusions

We have presented a characterization and performance enhancement of a website fingerprinting mechanism. Using experiments carried out over an SSL VPN, it was first shown that a widely used classification feature, namely, traffic burst size, is not able to perform well when

the target website contents are sufficiently dynamic. This effect was mitigated by using a new classification feature *traffic surge period* and adapting the *first n Components of Haar Wavelet Transformation*, which is commonly used in traditional signal processing applications. Presented results show that the addition of these features can bridge the fingerprinting performance gap between static and dynamic websites. Overall, the true positive rates for dynamic website fingerprinting were enhanced from about 87% with traditional features, to up to 97% by leveraging those new features. The results in this chapter help to establish the experimental environment that will be used in other works in this thesis and provided benchmark results for reference as the focus is turned to the classification of video streaming protocols.

# Chapter 4 : Protocol Independent Source Identification of Video Streaming Traffic

## 4.1 Introduction

In this chapter, we test the hypothesis that identifying the source of traffic is equivalent to identifying the route that the traffic travels (given that one end of the traffic route is fixed at the firewall). It is hypothesized that the inter-packet arrival interval (PAI) would be useful in identifying the route. A variation of the PAI feature is proposed in this chapter to capture the characteristics of both the route and the traffic source. We will test this feature in a video source detection context. The chapter also provides analysis on the nature of the PAI feature, revealing that it can be used to estimate the distribution of inter-packet delay variation (IPDV), an established end-to-end feature to measure network condition, without measuring end-to-end packet delay.

## 4.2 Problem Definition

In this chapter, we address the problem of identifying encrypted and tunneled video traffic at its source level granularity. Such identification should be performed in a manner that is agnostic to the underlying streaming protocol, coding, and actual content. We are particularly focused on streaming video traffic to achieve our overarching goal: to identify and block traffic from a specific video-streaming source (e.g., Netflix, YouTube) passing through encrypted tunnels. Blocking should be at the firewall through which the encrypted tunnel is passing through.

## 4.3 System Architecture

The proposed system for streaming video traffic source detection is shown in Figure 4-1. The system comprises of three major components: 1) a data collection agent that is inside the domain guarded by a firewall; 2) a traffic probe that is placed at the firewall; and 3) a data

processing engine at the firewall.

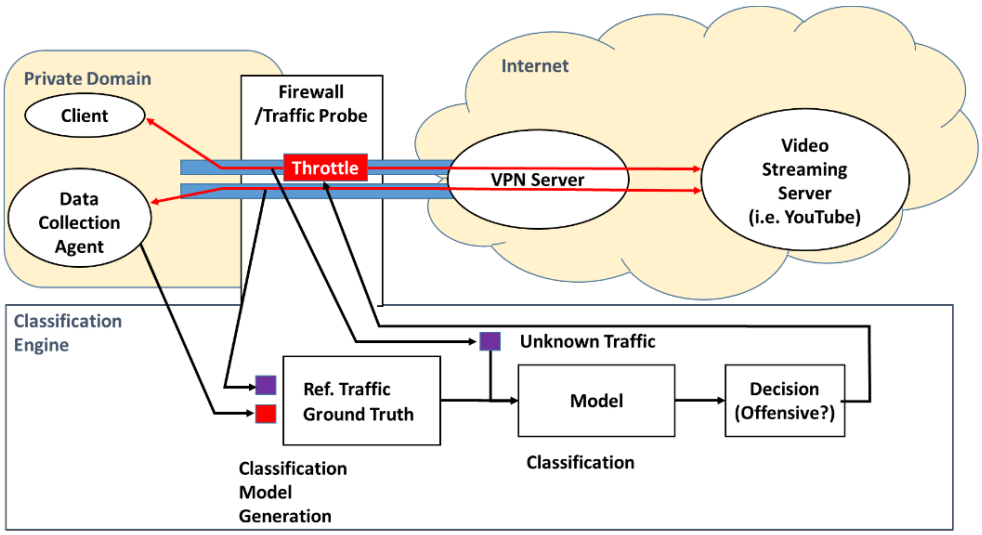



Figure 4-1: Classification architecture for traffic analysis

The data collection agent within the domain is used for collecting traffic for training purposes. For training data collection, sessions from the agent are created to different external video streaming servers, one at a time, through the same VPN server. Collected traffic samples are distinctly labeled with the corresponding server (i.e., YouTube, Netflix, etc.) so that the samples can be used for supervised training of a classifier. Agents in our experiments are computers that run automated scripts so that the training data is periodically collected at different times. Such temporal diversity in data collection is needed to ensure that the classifiers can cope with changing network conditions over time.

The classification engine in Figure 4-1 uses the reference labeled samples collected by the collection agent for training and generating classification models. Such models are then used for classifying unknown traffic probed from the video downloaded by real clients. If a classified video-streaming server appears in the enterprise's blacklist, the firewall attempts to block or throttle the

traffic from that server. In this chapter, we focus only on the source identification part. The mechanisms for throttling/blocking will be handled later.

### 4.4 Packet Arrival Interval (PAI) Feature

The objective of the arrangement in Figure 4-1 is to be able to classify video streaming servers in a protocol- and content-independent manner. Being able to identify a specific client-server route would satisfy that objective. Since an encrypted tunnel prevents any route information to be directly revealed to the firewall, a traffic signature or feature that indirectly indicate the route information is needed.

Most common features are packet and burst size, packet count, single trip delay, round-trip delay, and Packet Arrival Interval (PAI). Since burst and packet sizes depend on the underlying streaming protocol and the video content, those are not applicable for protocol- and content-independent server identification. The single-trip delay is ruled out because it requires coordination between the firewall and the destination server. This is because the probe can be placed only at the client end. In addition, since the server needs to be identified only by observing traffic at the client end, there is no way of measuring round-trip delay. This leaves Packet Arrival Interval (PAI) as the primary feasible feature for the source server classification scenario.

In what follows we show that PAI contains much useful timing information, which can contribute to its usefulness in source server classification. Figure 4-2 depicts two consecutive packet transmissions from the server to the client. The packets are sent from the server with an interval, and they reach the client-side firewall at time instants $T_1$ and $T_2$. The PAI is defined as. The other timing parameter of interest is the Inter Packet Delay Variation (IPDV)[35], represented by. The parameter  is termed as Inter packet Generation Delay (IPGD), which depends on specific video content and coding methods. Path delays, and  represent the properties of the specific route

including the congestion situations in the intermediate routers. The quantity  depends on both  and. From Figure 4-2, when the packets arrive in-order (i.e.,). Considering both in-order and out-of-order cases, PAI can be generalized as:

(4-1)

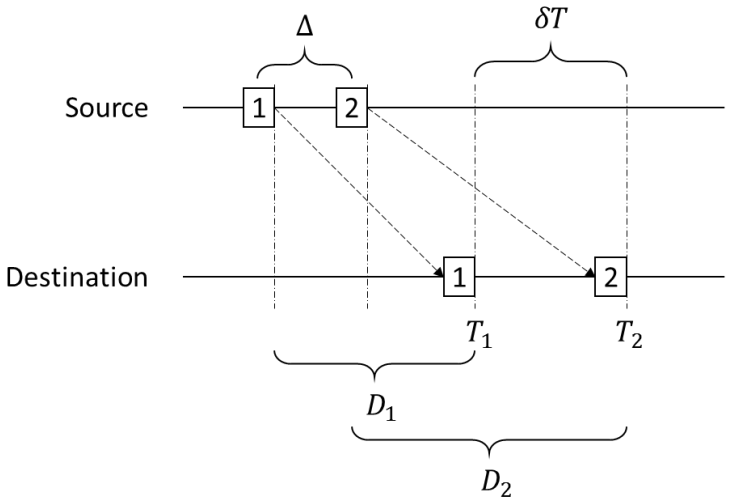



Figure 4-2: Relationship between PAI and other delay metrics

Both  and  can be modeled as stochastic processes. Figure 4-3 depicts example  distributions for a video being downloaded from Amazon, YouTube, and Netflix. The traffic samples presented in the figure are taken from 1-minute slices of actual traffic data from the mentioned sites. Then the distribution of PAI is extracted for all consecutive packet pairs. Note that the distributions are clearly bi-modal, with a stable saddle point at approximately 0.66ms between the 2 modes. The general observation here is that PAI, as observed at the firewall near the client, generally show a bi-modal distribution irrespective of the content and the streaming protocol and video coding combinations used by the server.

To understand the distribution, we start by examining shifted gamma distribution[36], which

is an accepted realistic model for expressing an end-to-end route delay. This model assumes that the path delays (i.e., and) have a constant component, and another component that follows a gamma distribution. It works well when all links in a path have similar delays. The probability density function of this delay model is shown in Eqn. (4-2). Here  is the shift, while $\alpha$ and $\beta$ are the respective shape and rate factors of the gamma distribution.

(4-2)

With the assumption that end-to-end delays are independent, the probability density function of IPDV can be derived from Eqn. (4-2) as its autocorrelation function.

(4-3)

Because of the symmetrical nature of IPDV, its PDF when  is a mirror of the part when. The shape of  is shown in Figure 4-4. It is Gaussian-like because it is the sum of several independent, identically distributed random variables and therefore is subject to the central limit theorem. For comparison, a second dotted line shows a Gaussian density function with identical parameters.

Since  depends on the property of an end-to-end route, it can be assumed as stationary as long as the network conditions change with a time constant that is larger than the classifier-training interval. The author of [37] points out that the stationary assumption can be effective for up to an hour, which is sufficient for the classifier model to be updated with new training done based on data collected within the hour.

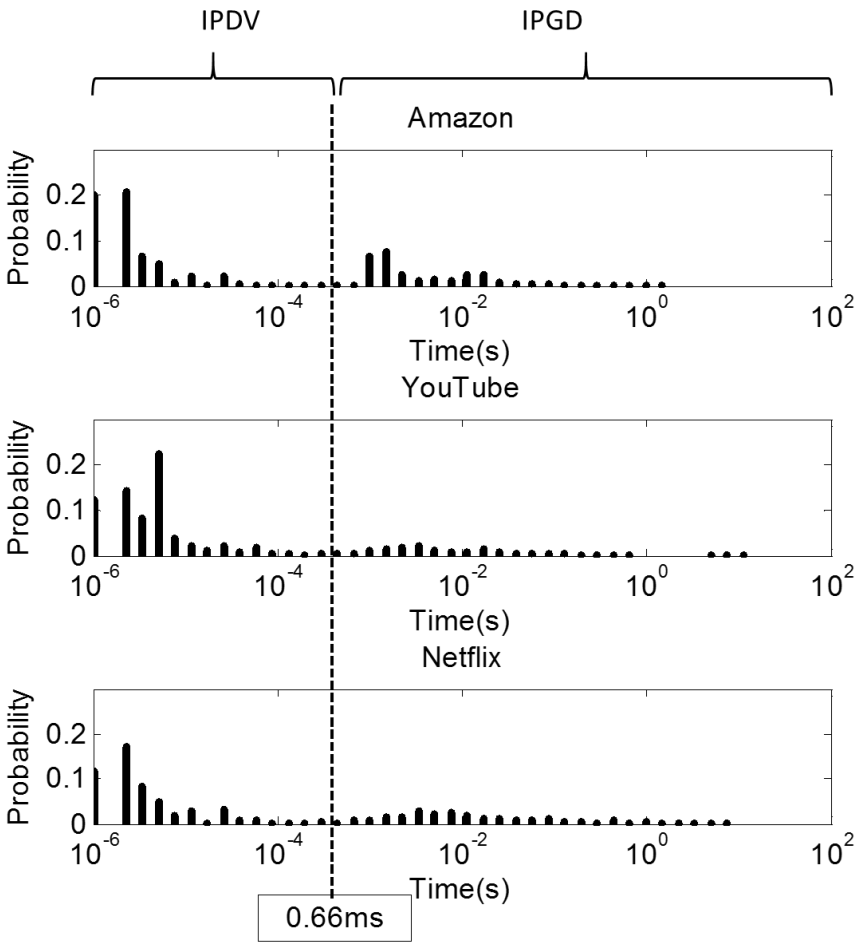



Figure 4-3: Example distributions of PAI for different servers

Observe that the distribution in Figure 4-4 has only one mode, while the distribution in Figure 4-3 for has two modes. The mode from Figure 4-4 actually corresponds to the lower time mode in Figure 4-3. Because the measurement captures only absolute values, the corresponding mode in Figure 4-3 represents only half of the mode in Figure 4-4.

The other mode in Figure 4-3 is contributed by Inter Packet Generation Delay (IPGD), which can also generally be assumed to be stationary when the coded video generation is modeled as an on-off process as shown below.

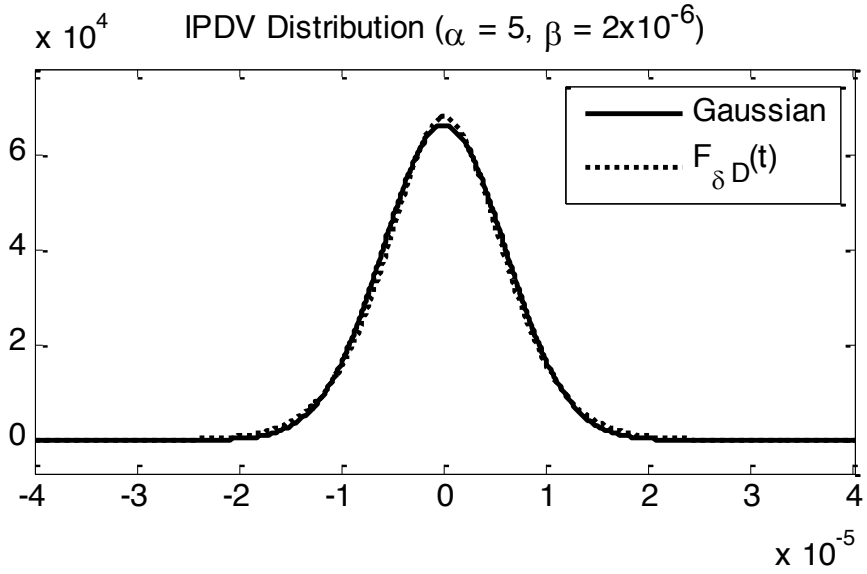



Figure 4-4: Example distribution of IPDV

To investigate this further, we examine probed (i.e., at the firewall in Figure 4-1) traffic samples from various video-streaming services as depicted in Figure 4-5. A common visual pattern across all the samples is that the traffic switches between two distinct states. One with short periods of high intensity, and the other with longer duration lower intensity. This alternating behavior is because data chunks in video streaming protocols are downloaded in bursts. Figure 4-5 generally confirms this.

This general observation can be modeled using an on-off model [38] Under such a model, the packet generation process can be assumed to be stationary when the system is in one of the two states. The process can be modeled as shown in Eqn. (4-4).

(4-4)

Note that the position of the first peak in Figure 4 is close to 10*us*. This is because, for a 1Gbps node, the source delay of a typical IP packet (i.e., 1500 bytes) is about 10*us*. We can use this value as an estimation of. The shape of this peak carries IPDV information with little distortion because of the highly concentrated distribution of , which acts essentially as a Dirac impulse

function. Moreover, since the distribution of *, the shape is very similar to. The second peak in Figure 4-3 exists because the source is switched to off state occasionally, either because the packets are being queued but not sent yet, or the streaming protocol has finished transmitting a data block and waiting for the time to transmit the next block.

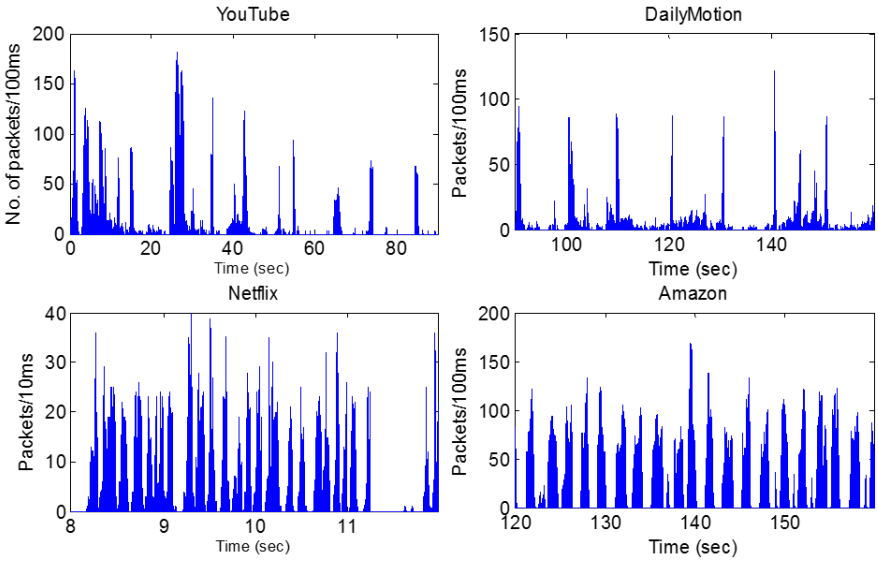



Figure 4-5: On-off patterns in video streaming traffic samples

To summarize, the above analysis confirms the statement in Eqn. (4-1), which claims that the feature Packet Arrival Interval (PAI) does capture information about both Inter packet Delay Variation (IPDV) and the Inter Packer Generation Delay (IPGD). While IPDV reflects network conditions, IPGD indicates application-layer properties. The hypothesis here is that with such information diversity about the underlying streams, PAI would be able to classify video sources with high enough accuracy for our application.

### 4.5 Experimental Setup

Figure 4-6 depicts the experimental setup in which video is streamed from multiple streaming servers to two distinct client locations, both through an OpenVPN server. OpenVPN, a

popular open-source VPN server, uses Secure Socket Layer (SSL) encryption. In OpenVPN version 1.5, a tunnel can be either UDP or TCP based. We use the UDP tunnel.

The server is set up within Michigan State University's Engineering network. Client-1 is placed within MSU's Spartan Village apartment network, which is a part of the broader MSU network. An encrypted OpenVPN tunnel is created from the client to the VPN server, and the videos are streamed from commercial streaming servers such as YouTube, Netflix, etc. (see Table 4-1) to the client via the tunnel. Traffic traces are collected at the client using the Wireshark [39] traffic probe.

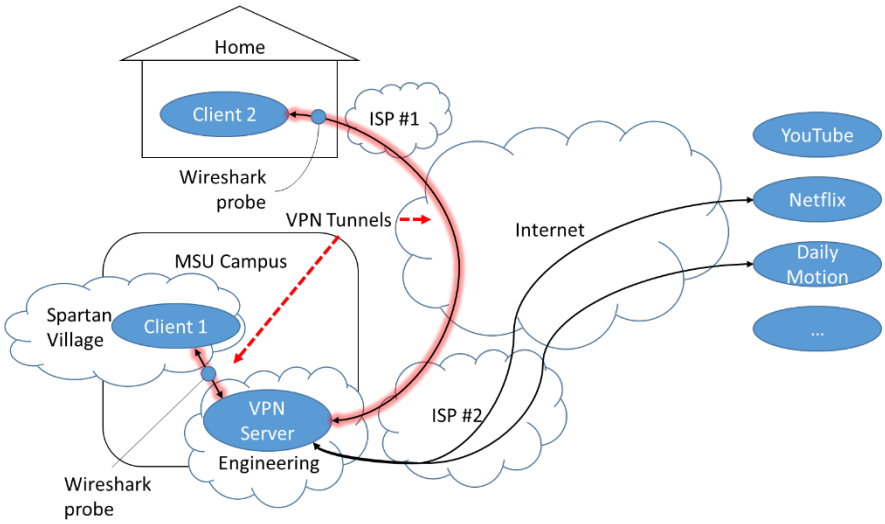



Figure 4-6: Experimental setup with OpenVPN

For Client-2, a similar arrangement is used except that it is located at a home, which is completely out of the MSU's network. The combinations of two client locations and many video-streaming servers provide many possible end-to-end network paths for investigating the performance of source identification in diverse network conditions.

Since most of the modern video streaming servers use their own content distribution network, the actual server machine from which the content is streamed depends on the location of

the client. Table 4-1 depicts the actual geographical location (i.e., obtained using IP to location mapping services) for the services when accessed from the two client locations in Figure 4-4. This client-dependent server machine location adds additional end-to-end path diversity, positively contributing towards the server identification process.

Table 4-1: Video streaming services and server locations

| Service | Client 1 | Client 2 |
|---|---|---|
| **DailyMotion** | New York City | New York City |
| **CNN News** | Ann Arbor, MI | Ann Arbor, MI |
| **Fox News** | Cambridge, MA | Unknown |
| **YouTube** | Mountain View, CA | Mountain View, CA |
| **Netflix** | Southfield, MI | Ann Arbor, MI |
| **Amazon** | Seaford, Delaware | Seaford, Delaware |
| **Vevo** | Ann Arbor, MI | Ann Arbor, MI |

For each server in Table 4-1, a randomly chosen list of video titles was downloaded at the clients at a different time of the day. Collected traffic streams are sliced into 1-minute chunks and a feature vector is extracted from each slice. 10-fold cross validation is used when testing the performance. Three different classification algorithms including J48 tree classifier, SVM, and 1-Nearest Neighbor are used to train models with labeled feature vectors.

Note that Wireshark probe collected the traffic traces in the above setup without any background traffic at the client. In other words, the downloaded video streams are the only traffic present at the client. Video source identification in the presence of background traffic will be a topic of future investigation.

### 4.6 Experimental Results

Feature Extraction: As per the analysis in Section 4.4, we use the measured Packet Arrival Interval (PAI) distribution at the client end as the classification feature. We use 50 bin with the

smallest one covering 0-3μs, and each subsequent bin has an upper limit 1.5 times larger than the previous one. The largest bin covers 56.6 sec to 85 sec. This spread covers most of the interesting activity of the traffic stream. Upstream packets and downstream packets are counted separately, and the resulting features are concatenated together. Therefore the actual feature distribution has 100 bins.

Traffic streams are divided into 1-minute slices. A slice is a record of the time and direction of each packet within that one minute. The set of slices is filtered by network activity before features are extracted. Only those slices that have more than 1500 packets/min are included in the feature set. The distribution data from all 100 bins are used for both classifier training and during the classification process. We collected more than 60 hours of data at each client location. The number of slices collected at each location is listed in Figure 4-7.

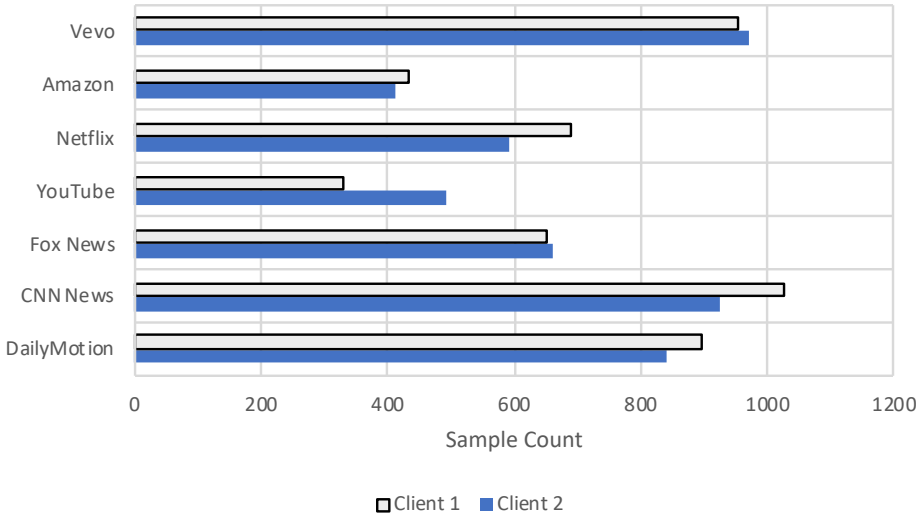



Figure 4-7: Number of collected slices from different servers

Classification Accuracy: Classification results in terms of successfully identifying the video streaming servers in Table 4-1 are reported in Figure 4-8. The high true positive rates for both the clients indicate that PAI as a feature does contain enough information about the end-to-end

network path so that it can uniquely identify the routes between specific client-server pairs. It should be noted that this high true positive rate happens when the classifier is trained and tested with different video contents, thus providing a source-identification mechanism that is content-agnostic.

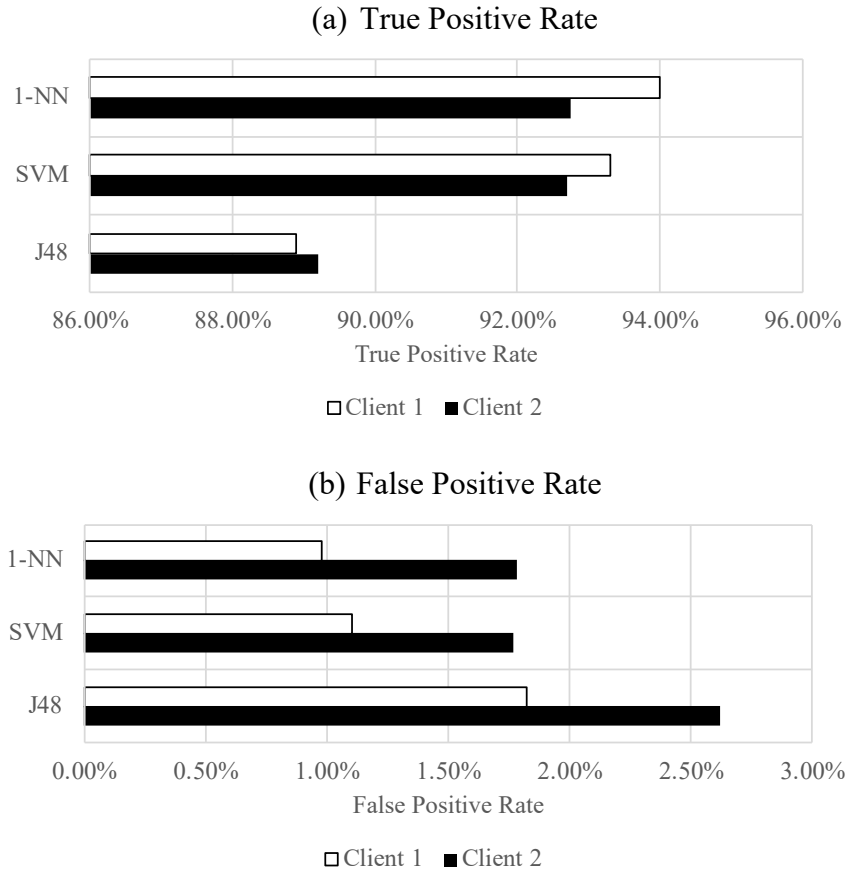



Figure 4-8: Classification accuracy for source identification

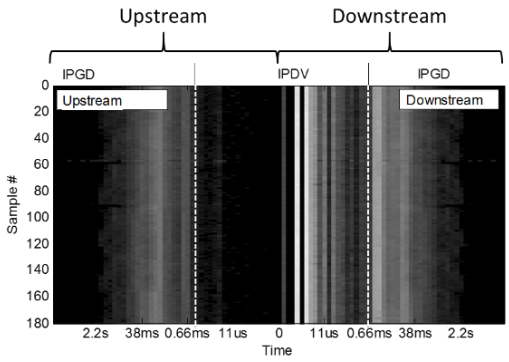
Figure 4-8 also shows false positive rates. Low FP values confirm that when the proposed method will eventually be used for blocking undesirable video servers, very few wrong streams will be blocked mistakenly.

Sub-feature Analysis: As analyzed in Section 4.4, the feature PAI includes information about both IPDV, which reflects network conditions, and IPGD, which reflects application level

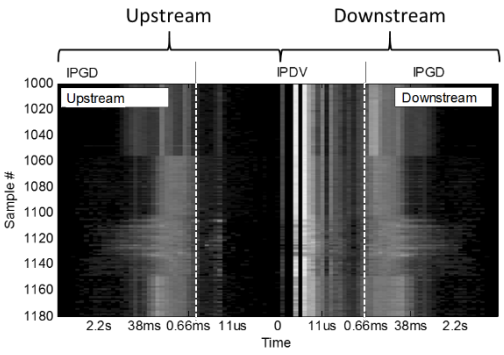
properties. In this subsection, we plan to investigate the individual contributions of those two components or sub-features to the classification process.

After analyzing the PAI distribution for many samples it was found that the time 0.66ms approximately represents the boundary between the two modes as shown in Figure 4-3. This signifies that all data below 0.66ms in PAI distribution represents IPDV and above 0.66ms represents IPGD.

(a) From Amazon.com




(b) From Netflix.com




(c) From YouTube.com

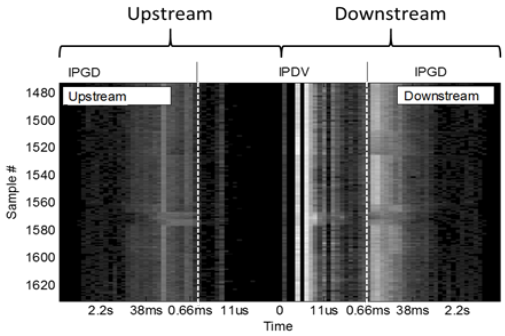



Figure 4-9: PAI distribution for data sample collected at Client 1

Figure 4-9(a) shows the PAI distribution of the samples collected at Client 1 from Amazon.com. Each horizontal line in the plot represents one sample, and the shade of a pixel represents the value of one particular bin in the PAI distribution for that sample. Black represents 0 and while represents 1. Other shades of gray represent values in between. The two vertical lines indicate the 0.66ms

division between IPDV and IPGD.

Figure 4-9(b) and Figure 4-9(c) depict similar results for NetFlix.com and YouTube.com. These, as well as similar plots for other servers, show the very similar boundary between those two sub-features.

In Figure 4-10, the mean vectors of the slices from all three sites in Figure 4-9 are plotted as histograms. For a given site, the bin values for all samples are averaged and the resulting mean distribution is plotted. These graphs confirm the significance of the 0.66ms division points as illustrated in Figure 4-3 and Figure 4-9.

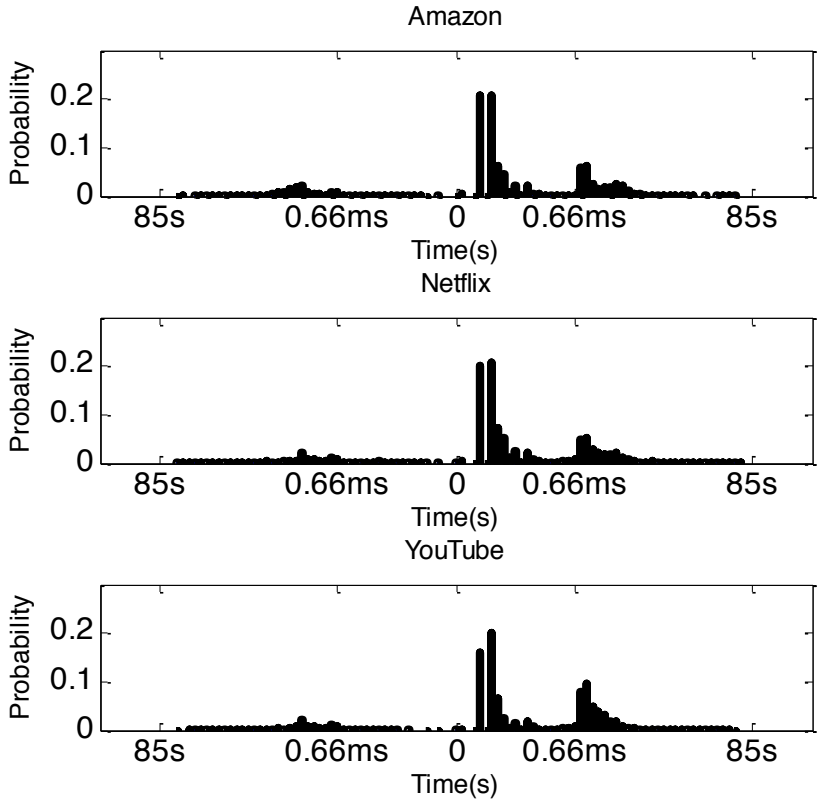



Figure 4-10: Class-average PAI distribution at Client 1

Classification results with individual sub-features are shown in Figure 4-11. The following observations can be made. First, IPGD serves as a better classification sub-feature compared to

IPDV. Second, the true positive and false positive numbers in Figure 4-11:b are close to those in Figure 4-8. This means that IPGD, by itself, would have been an effective classification feature for video source identification.

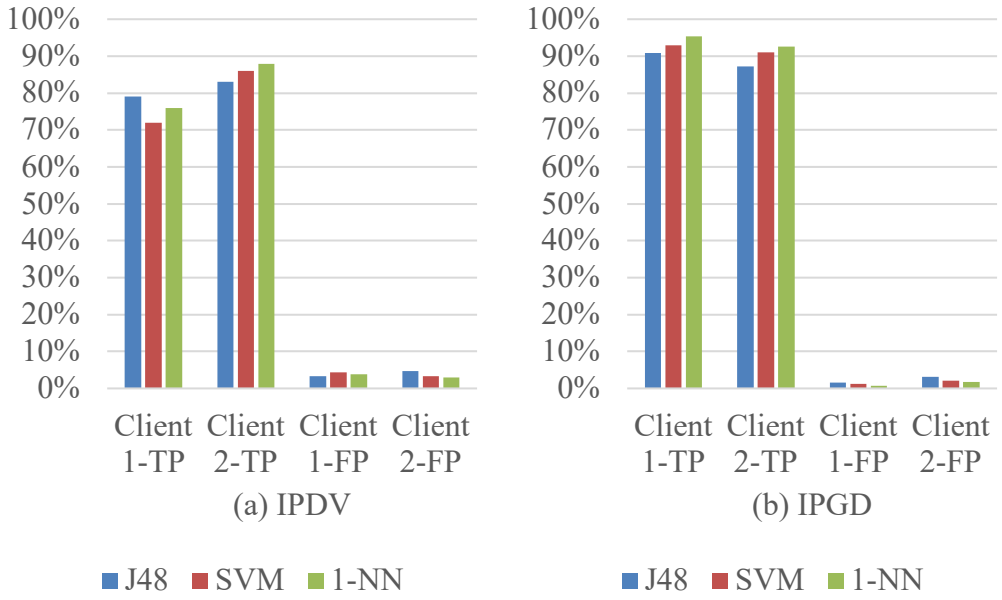



Figure 4-11: Classification Accuracy with Sub-features

However, this, as well as IPDV, are not measurable features at the client. They are available only at the server side. In our setting, since the video source needs to be identified at the client side using the information available at the client, Packet Arrival Interval (PAI) serves as the only feasible feature. The above analysis demonstrates that PAI, in fact, embeds two very important classification sub-features, namely, IPDV and IPGD, both of which contain appreciable classification abilities.

### 4.7 Summary and Conclusions

The chapter investigates Packet Arrival Interval (PAI) as a classification feature for identifying sources of tunneled video streaming traffic. The structure of the PAI feature is analyzed

to reveal its composition and its potential as a server-identifying feature. Extensive experiments were conducted with the OpenVPN server and a number of commercial video streaming services to evaluate the efficiency of PAI as the classification feature. In addition to validating its efficacy, we have also demonstrated that PAI embeds two very important classification sub-features, namely, Inter-Packet Delay Variation (IPDV), and inter-packet generation delay (IPGD), both have good classification abilities.

# Chapter 5 : Characterizing Common Traffic Features

## 5.1 Introduction

In this chapter, we explore the characterization of the factors that influence the performance of TA when identifying sources of tunneled video streaming traffic. Such identification can be used in enterprise firewalls for blocking unauthorized viewing of the tunneled video. We attempt to characterize and evaluate the impacts of the primary influencing factors, namely, streaming protocol, codec, and the actual video content. Analysis of data obtained from the test environment shows that the streaming protocols provide the most dominant source identification distinction. Also, while the codecs provide some weak distinctions, the influence of video content is marginal. In addition to in-laboratory experiments, a real-world verification to corroborate those observations is also made with commercial streaming service providers. Such "long-haul experiments" indicate that the end-to-end network conditions between the streaming server and video client can act as an additional influencing factor for TA towards video stream source identification. Overall, the results suggest the feasibility of TA for unknown video stream source identification with diverse video examples.

## 5.2 Problem Definition

The problem tackled in this chapter is the characterization of Traffic Analysis (TA) for identifying sources of tunneled video streaming traffic. Its purpose is to answer the question of whether network traffic contains information to identify its source. Such identification is useful in enterprise firewalls for blocking unauthorized viewing of the tunneled video. While we attempted to answer the question with mathematical analysis in Chapter 4, we will attempt to characterize and evaluate the impacts of the primary TA-influencing factors experimentally. Three major

factors are evaluated: streaming protocol, codec, and the actual video content, all while using packet size distribution as a feature. A test environment is built so we could control these factors.

As a second step, another factor that is important in this context, the route between server and client computers, is evaluated as well. These are named “long-haul experiments” because of the long, artificially created path between the servers and the clients. This step is designed to answer the question of whether signatures of server location can be detected in traffic while all other parameters being identical.

### 5.3 Experimental In-laboratory setup

Network Configuration: An experimental setup is built with the following components: 1) a privately-owned streaming server running Linux, and 2) a client computer that runs Windows. Both machines are connected to the same router and therefore are just one hop from each other. The server runs both an instance of “C++ RTMP Server” (*crtmpserver)* and a modified instance of “*GStreamer Streaming Server” (GSS)* that supports the newest version of Smooth Streaming protocol and recognizes VC-1 encoded video file. To allow the client to stream the video, an Adobe Flash-based player, JWPlayer, is used to stream from crtmpserver, while a reference implementation of Smooth Streaming player found in Silverlight Media Framework (SMF) is used for streaming from GSS. Client and server communicate through an internal VPN server that runs the IPSec based Juniper VPN service. The entire setup is shown in Figure 5-1.

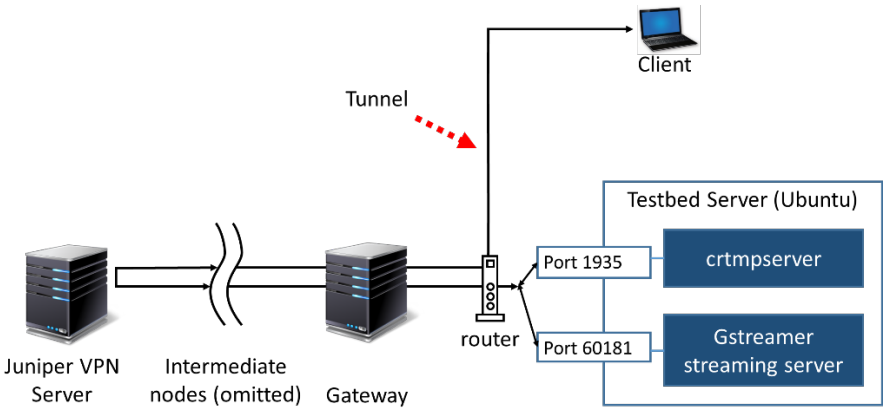



Figure 5-1: Network environment and the experimental setup

Traffic Data Collection: Traffic traces are collected by running Wireshark protocol analyzer on the client machine. Only the traffic between client and Testbed server is collected while other packets are dropped. Since the server is not running other services (no more than a single HTTP browsing session) outside of video streaming, and the client is not running any significant networking application other than video streaming players, we consider the traffic to be free from background noise.

Several public-domain or Creative Common (CC) licensed movies are chosen for this experiment and they are converted to different formats to be served on both crtmpserver and GSS. The complete list of video files and formats is shown in Figure 5-2. Smooth Streaming is shown as "SStream" in the table. Care is taken to choose video files with diverse qualities and picture sizes. We chose "Sita Sings blues" to be encoded into 2 different picture sizes (originally 852x480 pixels, reduced to 500x282 pixels) to study the effect of quality on recognition results. Video files are converted into H.264/AAC format by Handbrake video converter with a fixed keyframe interval of 2 seconds, or into VC-1/WMA format by Microsoft Expression Encoder (MEE). While MEE encodes a video file into multiple quality levels when encoding for Smooth Streaming, only

one randomly chosen file from the encoded files is used for each video class to represent the diversity of real-life video files.

Data Preprocessing: Packet size distribution is used as the classification feature following its usage in [2]. Samples are sliced into 1-minute chunks and packet size distribution is extracted from each chunk. Features are labeled by their respective video file sources. The number of samples extracted is also shown in Figure 5-2. Sample size is in general proportional to the original length of the video, while some variance is because of missing data. Half of the H.264/AAC portion of "Sita Sings Blues" is collected with a video file that has 640x480 frame size. This is done to test the effect of content format on the features. A detailed description of the extraction method can be found in Section 5.4.

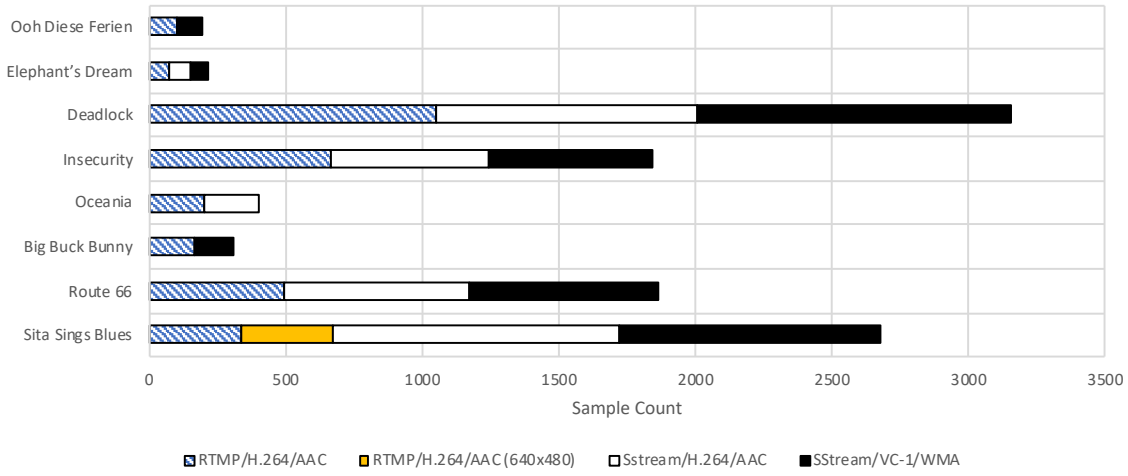



Figure 5-2: Collected video Samples

### 5.4 Feature analysis

Both RTMP and Smooth Streaming are TCP based, and therefore leave the task of splitting data into packets to the underlying IP layer. Although the splitting is transparent from the application layer/transport layer protocol, results have shown that the packet size pattern is

distinctive enough for recognition of not only application layer protocol, but also webpage content and the website accessed.

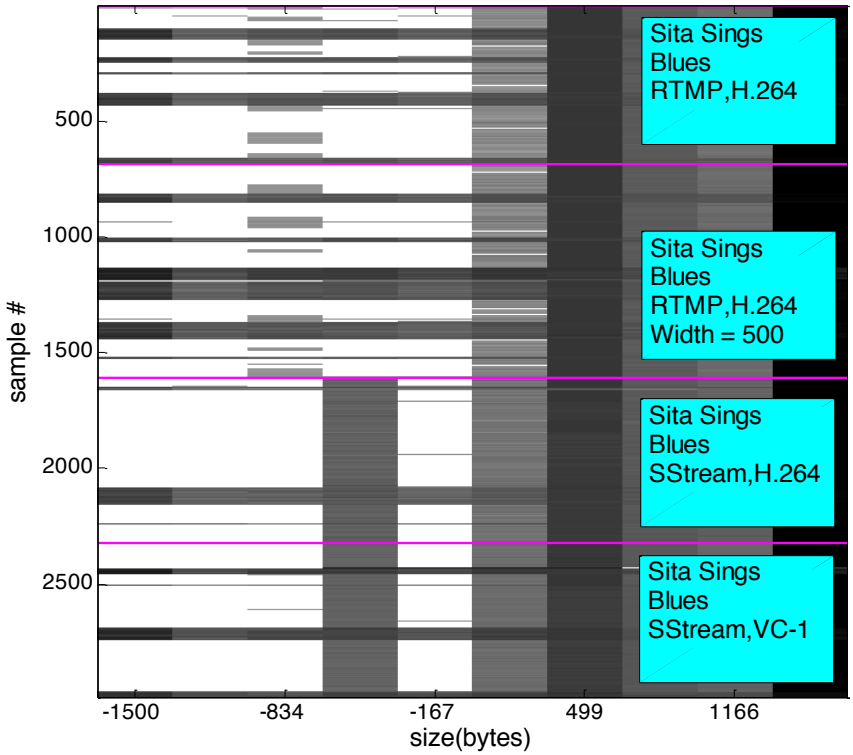



Figure 5-3: Packet size distribution of 4 “Sita Sings Blues” classes

Distribution of packet size is obtained by dividing all the packets in a sample into different bins using their direction and size. The number of packets in each bin is counted and a histogram is built. In this chapter 10 bins that evenly divide the range [-1500, +1500] bytes are used to calculate the feature. The sign before the packet size indicates the direction of this packet (minus is upstream, while plus is downstream). The choice of 10 bins is because of concerns in the robustness of features, as well as computational cost. Before computation of feature, the stream first passes through a filter that removes pure TCP ACK packets from the stream following the practice in [3]. Histogram calculation follows, then each histogram is normalized, i.e. the value of

each bin is replaced by its value divided by the sum of all bins. This mitigates the influence of traffic load on the feature.

A few example features are shown in Figure 5-3 in which all samples of the four classes that belong to "Sita Sings Blues" are plotted as a grayscale map. The *x*-axis corresponds to the 10 bins in the histogram, while the *y*-axis is the count of the samples. Samples are divided into 4 classes by horizontal lines that are most clearly visible from the rightmost of the figure, and on each area, there is a label that names the class it corresponds to. A dark stripe indicates that the histogram has a high value in that bin. Even before classification, we can see that the classes are clearly divided into 2 groups: The RTMP/H.264 group and the Smooth Streaming/VC-1 or H.264 group. The difference between the two can even be spotted on the map without the help of a classifier. This corroborates the established fact that Traffic Analysis can be effective at recognizing application layer protocol.

To get higher resolution and more consistent results, a classifier must be trained. Support Vector Machine (SVM) is used in [3], but [23] also finds that packet size distribution works well with other types of classifiers. A J48 tree classifier is used because it is quick to train and has shown good performance.

To simulate the real-world situation where the classifier has to deal with a large test set with a relatively small training set, the dataset is split between the training set and test set at a 15%-85% ratio. Only 15% of the dataset is used for training while the rest of the dataset is used for testing.

### 5.5 Experimental results

We first train a J48 tree classifier using 15% of the dataset and then test with the rest 85%. We seek answers for the following questions: How good is the classifier in distinguishing between

1) different protocols, 2) different codecs encapsulated in the same protocol, and 3) different video content for the same protocol/codec combination.

Table 5-1: Protocol Identification Rate (%)

| Protocol | As RTMP | As SStream |
|---|---|---|
| **RTMP** | 93.8 | 7.2 |
| **SStream** | 4.8 | 95.1 |

The first result is from the protocol combinations and is shown in Table 5-1. The classifier recognizes the two protocols very well, with more than 90% accuracy. This is to be expected since not only is the protocol identification ability already established by earlier studies, but also the two protocols are obviously different as visible in terms of their features.

In the second set of results shown in Table 5-2, classes are grouped according to their protocol/codec combinations. Each group is assigned a number (1~3 in the 1st column) and the percentage of that class being classified as the $n^{th}$ class is shown in the column "As #*n*" respectively. The majority of RTMP/H.264 samples are classified correctly, which is expected. The interesting point in this table is that while the overall performance is worse than that of RTMP/H.264, the two Smooth Streaming based groups are also separated relatively well. Considering that the codec only affects the traffic load (size of slices) in Smooth Streaming protocol but not the underlying HTTP protocol, a result like this is unexpected. The reason for this difference is still under investigation. However, given higher than 90% accuracy in the protocol identification case, the difference between the two codecs is localized within the Smooth Streaming protocol group. This can also be confirmed from the confusion rates between class #2 and class #3, which are significantly higher than the confusion rates between class #1 and either of the two Smooth Streaming classes. The distinguishing impacts from codec are clearly visible but are weaker than that from the streaming protocol. This is one of the key observations that states

that the streaming protocol is more influential than the codec for distinguishing video sources using traffic analysis when packet size distribution is used as the classification feature.

Table 5-2: Protocol/Codec identification rate (%)

| | **Protocol/Codec** | **As #1** | **As #2** | **As #3** |
|---|---|---|---|---|
| **1** | RTMP/H.264 | 94.8 | 3.8 | 1.4 |
| **2** | SStream/H.264 | 3.1 | 85.7 | 11.2 |
| **3** | SStream/VC-1 | 2.4 | 20.1 | 77.5 |

We also want to know the distinctiveness of the feature among different video files of the same protocol/codec combination. For this, we calculate the conditional probability of a certain class to be classified as another class in the same group. Let  to be the set of all classes within one protocol/codec group. Each class is identified as, then the probability can be expressed as:

(5-1)

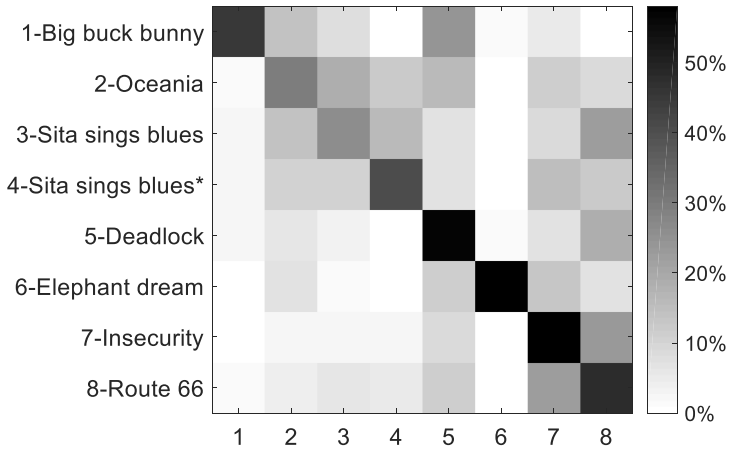
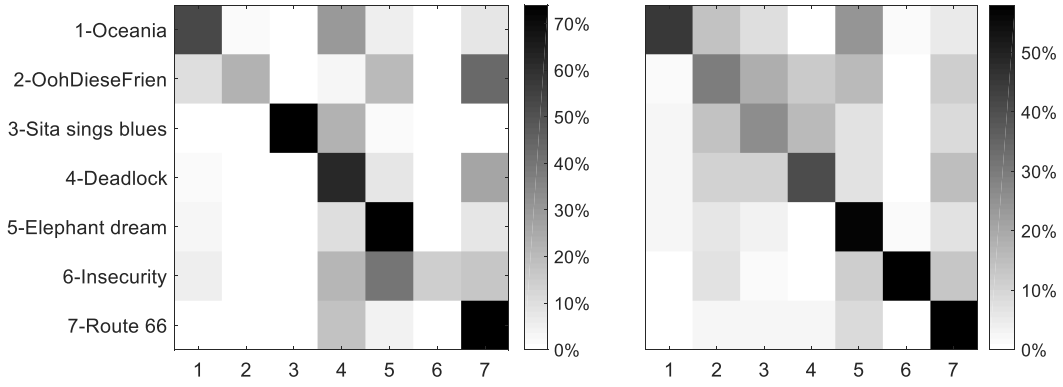



Figure 5-4: RTMP/H.264 Intra-Group Confusion

Following this calculation method, we compute Figure 5-4 and Figure 5-5 that shows the confusion within the RTMP group and Smooth Streaming group respectively. It is clear from the results that the classifier faces difficulty in separating classes with the same protocol/codec combination. This is an indication that the degree by which classes overlap in the feature space is

high. Judging by the performance, the content does have a minimal impact on the feature, but most of the samples do not deviate too much from their protocol/codec combination (hence the good performance with protocol/codec).




(a) (b)

Figure 5-5: Smooth Streaming Intra-Group Confusion (a: H.264, b: VC-1)

Special attention is paid to the two "Sita Sings Blues" classes with RTMP grouping but different sizes. The confusion between the two classes is only marginally higher than the confusion between other classes. This shows that the change in quality makes a video stream almost as different as another video stream within the same protocol/codec group.

### 5.6 Experiments with Commercial Service Providers

To test if the results obtained from the controlled scenarios in Section 5.5 scale to real-world long-haul scenarios, the following experiment, as shown in Figure 5-6, is set up. In this setup, the client is put into a separate network that is at least 10 hops away from the VPN server to allow network variations. Collecting data from internal test server is replaced by collecting from commercial servers that serve YouTube, Netflix, and Hulu, while the VPN server in use is still the same Juniper VPN server as described in section 5.3. This setup allows us to view the difference between the single-server setup in section 5.3 and a multi-provider scenario. Two of the three

chosen sites run the same protocol/codec (RTMP/H.264 is used by both Hulu and YouTube) but run from different locations. This introduces additional variability in terms of network delay and jitter caused by different end-to-end server-client routes. In all, 10 video titles are randomly chosen from the three video streaming sites. For Hulu, we have chosen a playlist that randomizes its content in each playback session in addition to the other two randomly chosen titles. Data has been collected in a similar fashion as in section 5.3. Considering the additional intermediate nodes, as well as the fact that all 3 sites are hosted on multiple servers and potentially in different data centers, this setup is more complex and practical than the isolated test environment used in the previous sections.

We focus on the same results presented in section 5.5. The RTMP group in Table 5-3 is comprised of all video samples from YouTube and Hulu, while the SStream group contains all samples from Netflix. The recognition rates between the two groups shown in Table 5-3 are around 86%, which are lower compared to the isolated test results, but still decent.

The result of identifying samples from different sites is shown in

Table 5-4. Note that this result was not presented in the isolated test scenario in Section 5.5, because there is only one server in that case. Interestingly the confusion between Hulu and YouTube is low, even though they share the same protocol/codec pair. The observation that the recognition rate is higher between Hulu and YouTube than what can be obtained from protocol and codec (shown in section 5.5) can be explained using the following hypothesis.

The network condition (i.e., in terms of end-to-end delay, jitter, etc.) of the route between the streaming server and client has an influence on the traffic analysis feature. If true, this would be good news for a firewall that intends to block certain sites while not affecting others using the same protocol/codec combinations. The hypothesis needs to be verified further in future work in an isolated environment by artificially introducing network condition variations.

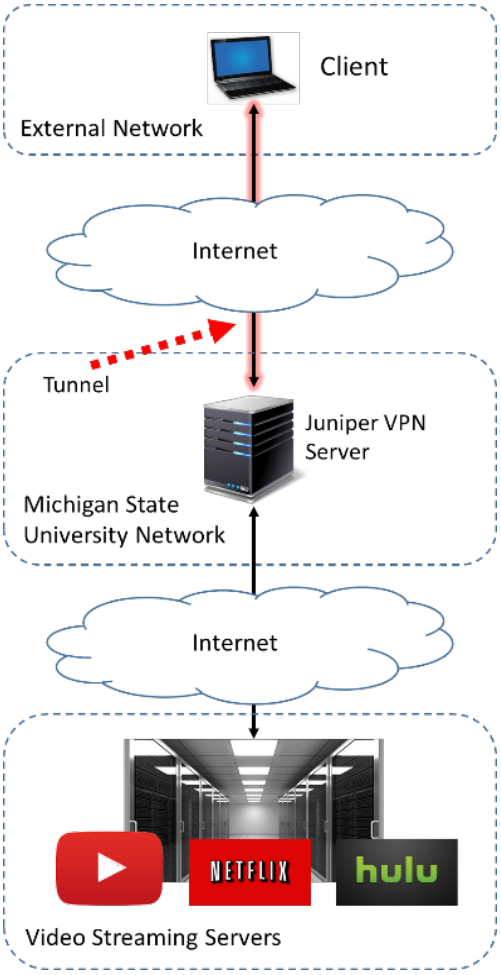



Figure 5-6: Long-haul experiments

Table 5-3: Protocol Identification Rate (%)

| Protocol | As RTMP | As SStream |
|---|---|---|
| **RTMP** | 86.3 | 13.7 |
| **SStream** | 13.2 | 86.8 |

Table 5-4: Site Identification Rate (%)

| | Site | As #1 | As #2 | As #3 |
|---|---|---|---|---|
| **1** | Hulu | 80.4 | 5.6 | 14 |
| **2** | Netflix | 12.1 | 86.8 | 1.1 |
| **3** | YouTube | 9.1 | 10.7 | 80.2 |

Results on multiple videos from the same site are shown in Figure 5-7. They also show a trend that is like that of the isolated test scenario in Section 5.5. The classifier still has difficulty recognizing individual video titles, but the accuracy is statistically better than random guesses, indicating minor influence from the video content. The figures are higher than those in the isolated test, suggesting that the samples from video streaming service providers are less homogenous. A possible reason for higher heterogeneity is that the providers run multiple servers to serve their content.

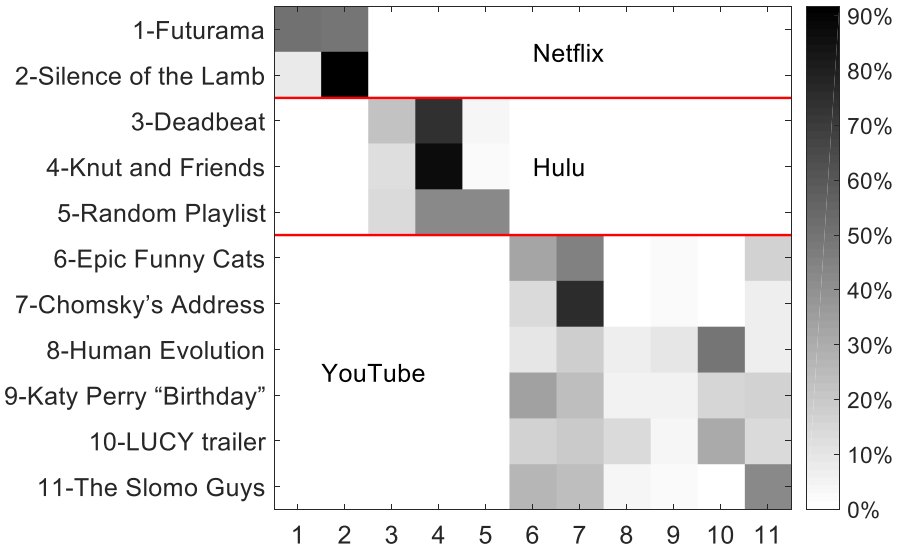



Figure 5-7: Intra-site Confusion in Long-haul Experiments

To summarize, the results from long-haul experiments with commercial video streaming servers corroborate the basic findings from the in-laboratory isolated testing in Section 5.5. Meaning, streaming protocols, codec, and video content have classification influence in decreasing order. Additionally, it also turns out that the underlying network condition can be an important impact factor that influences the classification feature. Further work is needed to characterize and understand it better.

## 5.7 Summary and Conclusions

The goal of this chapter is to answer the question of what factors contribute to the effectiveness of features used in Traffic Analysis based streaming video source identification. The study focuses on the analysis of encrypted tunneled video streaming protocols. Normalized packet size distribution is used as the feature for measurement. The results show that video streaming protocol has the most distinctive impact to feature among the three factors studied. In addition, video codec and video content are also identified as sources of video source distinctions. While the protocol has the greatest impact on feature distinctiveness, the impact of the codec is less significant but clearly visible, and individual content only contributes marginally. These observations provide insights into the effectiveness of Traffic Analysis methods and provoke questions about the nature of the variations. A real-world verification of the observations is made by analyzing data collected from major video streaming service providers over a long-haul network. Not only the results corroborate the observations made in the isolated study, but it also reveals the difference between single-server streaming test environment and the real-world massively distributed streaming environment. The samples become less homogenous in this case, as samples from different service providers show more diversity than from the protocol and codec that we can measure in the test environment.

# Chapter 6 : Effects of Location Variability on Single-Stage Classifier

## 6.1 Introduction

This chapter explores the capability of a single stage classifier to identify video streaming traffic, like a conventional Traffic Analysis setup. This will provide a basis for constructing other improved classification schemes. The classifiers and features used in this chapter are chosen following several state-of-the-art works in this field. The experimental setup is unique in that it collects data between two client locations and one VPN server. The dataset obtained from the experiment is used to verify the generalizability of the classifier between different geological locations. Results show that the features are affected by the client location, and therefore training data for a firewall has to be collected locally.

## 6.2 Problem Definition

This chapter aims to build a TA based blocking framework in which site-specific signatures are identified first by analyzing traffic from different video streaming sites, and then such signatures are used for blocking traffic. Sites that are subject to this study include YouTube, Netflix, and Hulu. We review several popular features used by established works in the TA field, evaluate their effectiveness against video traffic, and develop a framework for recognizing video streaming sites. We also present the performance of applying TA methods to traffic from or to new places that are unknown to a trained classifier.

## 6.3 System Components

In Figure 6-1, the flow of data in a typical TA application involving video streams is presented up to the point of feature extraction. As the client computer accesses a video-streaming

server through a secure tunnel, the firewall is able to obtain the timing, the size, and direction of each packet. The traffic then goes through the preprocessing stage to become a time-stamped series of packet sizes. Packet sizes in the timed series are signed to indicate their direction. The upstream packets are assigned negative sizes and the downstream packets are assigned positive sizes. A variety of feature extraction methods are then applied to the preprocessed data to get the features. Finally, a classifier is trained using such features to identify specific content and/or pattern in the video stream.

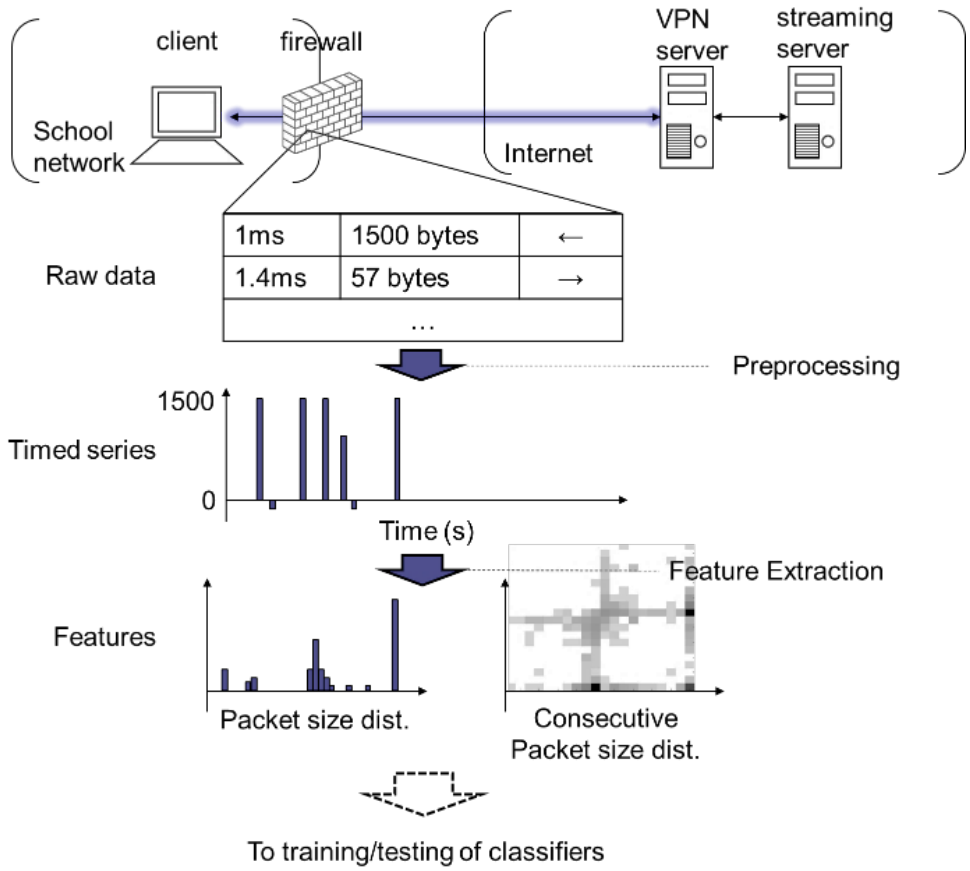



Figure 6-1: Data collection and feature extraction

For example, to recognize traffic from YouTube, the firewall needs to collect enough samples of traffic from YouTube vs. non-YouTube streams, and then extract relevant features as shown in the figure. A classifier is then trained using the extracted features. When unknown stream

traffic is detected, the same feature extraction process as in the training phase is applied to the unknown traffic to extract features for classification. The classifier determines the source of the traffic to be YouTube or non-YouTube. Based on the classification results, the firewall can then take further actions, including throttling or blocking the stream or to provide priority bandwidth allocation to it.

### 6.4 Experimental setup

Network Configuration: We set up an experimental environment to implement the case when an enterprise firewall collects traffic from clients inside its domain. As shown in Figure 6-2, the setup includes clients, one inside the same class B subnet (MSUNet) with the VPN server (A Juniper VPN server) and another outside of the class B network. The setup introduces a variation in client positioning with respect to the network. The intention is to be able to observe the influence of network structure on the features described in Figure 6-1. The Juniper VPN server supports both IPsec based VPN and SSL based VPN. We choose SSL based VPN service for the described experiments.

Traffic traces are collected at different times of the day (morning, afternoon and night) in order to capture realistic network traffic conditions. We collect data across a 10-day period with three sessions each day.

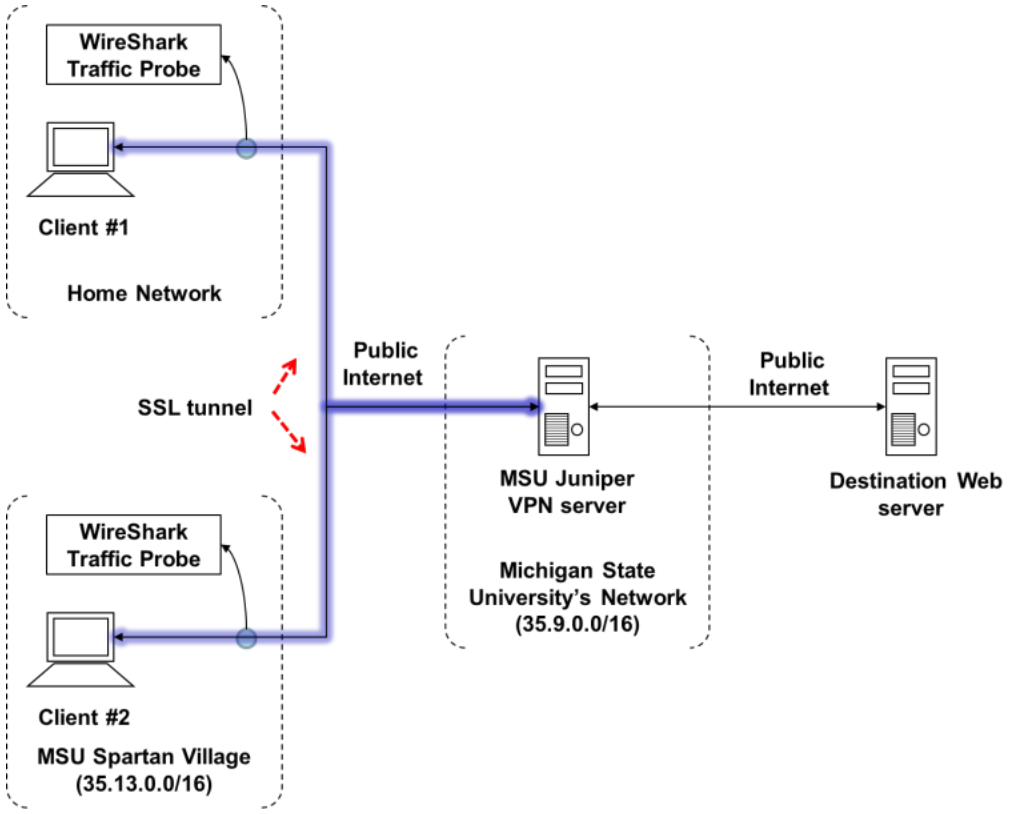



Figure 6-2: Setup of Network Environment

Choice of Sites and Content: In this chapter, we focus the effort on discerning YouTube traffic from the traffic of a few other popular video streaming sites and HTTP based web traffic from other popular websites. The type of content may have an influence on how well the content can be compressed by the codec and therefore, its immediate bitrate. In general, simple images are compressed better than complex images, and stills are compressed better than dynamic scenes. To take content type into consideration, we choose videos from different sites.

The sites and the video clips we choose from each site are listed in Table 6-1. The sites are deliberately chosen to represent the most popular video streaming sites and the type of protocols. Hulu is chosen as an example of a video streaming site using the same streaming protocol as YouTube (RTMP), and Netflix is chosen because it is the most popular video streaming site in

North America. Netflix also uses Smooth Streaming, a different streaming protocol than what YouTube uses.

Table 6-1: Collected video samples

| Site | Video Title | Type |
|---|---|---|
| **YouTube** | Epic funny cats 20 mins | Home Video |
| **YouTube** | 10 Weird Facts about Human Evolution | Slides |
| **YouTube** | Lucy Official Trailer #1 | Movie |
| **YouTube** | Katy Perry – Birthday | Music Video |
| **YouTube** | Underwater Bullets at 27,000fps | |
| **Hulu** | Bones season 9 episode 21 | TV series |
| **Hulu** | Deadbeat season 1 episode 4 | TV series |
| **Netflix** | Futurama season 1 episode 1 | TV series |
| **Netflix** | The Silence of the Lamb | Movie |

Collection of traffic: Video-streaming traffic to a client machine is usually accompanied by other types of traffic caused by activities such as browsing websites, checking E-mails, or downloading data from FTP sites. From the video traffic standpoint, these are background traffic. For the reported experiments in this chapter, we assume that there is no background traffic when the video plays because video streaming usually is so bandwidth intensive that most of the packets going through the channel belong to the video stream being watched. We also assume that web browsing is the only type of other traffic from which the video needs to be distinguished. This is because, on the Internet, web browsing happens to be the most dominating network usage other than watching streaming video [1]. For representative web browsing, we collect traffic traces from popular web sites including Facebook, Yahoo, and Google.

Data preprocessing: The goal for the presented experiments is to be able to distinguish YouTube traffic from other video streaming traffic such as Netflix or Hulu, and web browsing traffic. After

collecting packet traces for different traffic in different sessions, each such trace is sliced into one-minute long chunks that can be used for classifier training and validation purposes. Such slicing is performed for both video and non-video web traffic. For each network layer packet within a slice, its size, direction, and timing are recorded. Classification features are then constructed using those packet-specific parameters.

### 6.5 Feature analysis

Due to the tunneling of traffic through VPN, features of the traffic must be built from the two basic features that are not concealed by tunneling. Those two features are the size and timing of each packet. In this section, we evaluate the traffic analysis features established in [2] and [40].

Packet Size: A video server splits a data stream into network layer data packets before sending the stream to its clients. In certain conditions, the upper layer protocol can be directly discerned by looking at the characteristic data packet lengths. For example, the characteristic 54 bytes ACK packets that are present in large numbers when the transport layer protocol is TCP [3]. Identification of even higher layer protocols can be achieved when one looks at the distribution of different packet sizes present in the traffic. Packet size may also be influenced by the specific implementation of a protocol. As previous works show, packet size is also useful in identifying a website where a certain traffic trace is from, especially when the underlying protocol is known.

In our experiments, we calculate packet size distribution from the data collected for each 1-minute video slice for all the collected sessions. We record the distribution of packet size for each video in a 40-bin histogram. The bins are evenly distributed in the range of -1500 bytes and +1500 bytes, with negative values indicating upstream traffic and positive values indicating downstream traffic. The histograms are then normalized before stored as the final feature.

In Figure 6-3, the, burst characteristics of YouTube, Hulu and Netflix traffic are presented. The first 3 graphs reveal that all three of them download the video in relatively large chunks instead of a continuous stream of packets. YouTube traffic is characterized by two periodically reoccurring bursts that both appear every 30 seconds in the graph. Traffic from Hulu has an average burst interval of about 10 seconds, while Hulu frequently switches into the second mode of operation in which a continuous stream of traffic with lower peak bitrate than what occurs in the burst mode. Traffic from Netflix has shorter burst intervals and both the interval and the size of a burst change frequently. The problem of using burst as a feature is that burst can be influenced by the end-to-end route. In the bottom part of Figure 6-3, we present another trace of the same video on Netflix. The only difference is that the client can only access Netflix on a longer route. The burst characteristic is almost invisible here. Considering the above observations, we exclude the distribution of burst sizes as a feature for video server identification/classification as attempted in this chapter.

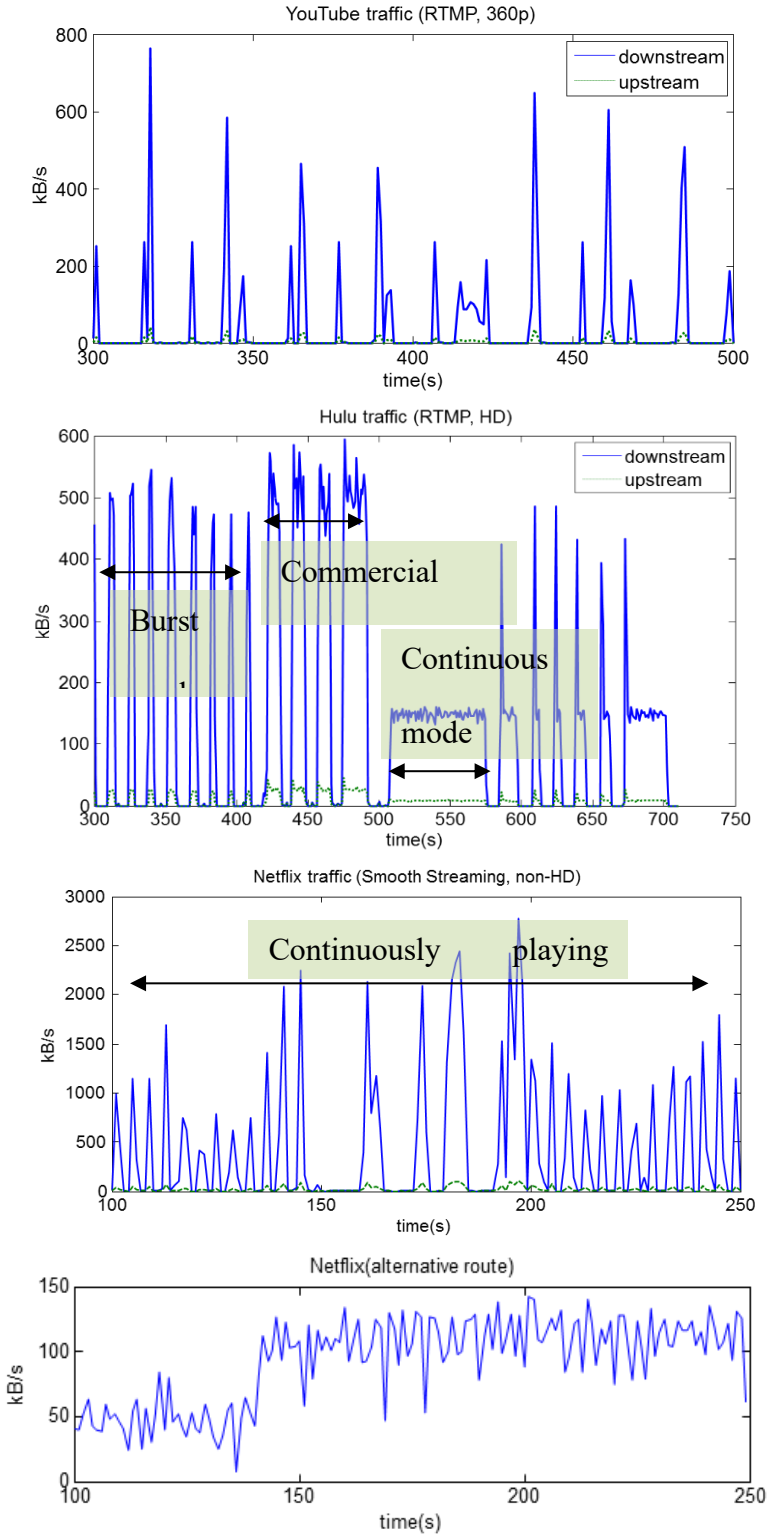



Figure 6-3: Burst characteristics of YouTube, Hulu, and Netflix

Consecutive packet size: Consecutive packet size pair is defined as the 2-tuple of 2 packet sizes appearing one after another in a traffic trace. In [40] the 2-tuples constituted of packets appearing immediately adjacent. The reason behind the effectiveness of this feature in [40] is that the abundance of recurring phrases in natural languages and the variable bit-rate compression that leaves a trace of information about the original voice signal even after encryption. For other protocols that do not use variable bit-rate compression, consecutive packet sizes may still be effective as an indication of the protocol because many protocols have recurring sequences of packets much like recurring phrases in natural languages.

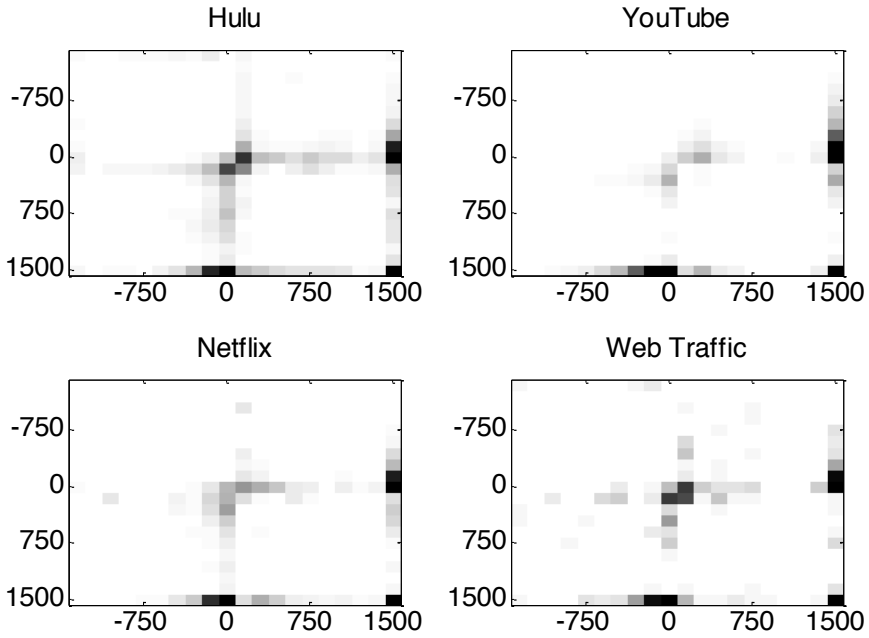



Figure 6-4: Samples of Consecutive Packet Size Pair Distribution

We calculate this feature as follows. The packet size pairs are recorded in a 20x20 bins 2-D histogram. The bins distribute evenly from -1500 bytes to +1500 bytes for each element of the pair. The resulted histogram is normalized before constructing the final feature. Some of the sample features are shown in Figure 6-4. Darker points on the map correspond to the denser distribution of packet size pairs around that bin. The distinctiveness of the feature can be clearly

observed from these sample distributions in Figure 8. While RTMP is used by both Hulu and YouTube, YouTube traffic has fewer small packets followed by another small packet while Hulu traffic has more. Hulu traffic also has more middle-sized packets compared to other sites.

### 6.6 Experimental results

Experiments are done in the following two phases. During the first phase, the focus is put on the performance of distinguishing YouTube traffic from the rest of the sites. This is done using either packet size distribution or consecutive packet size as the classification feature. In this phase, the training dataset and the test dataset contain traces collected from client #1 only (see Figure 6-2). For the non-video traffic class (HTTP accesses), we put traces from the same set of popular websites in both the training set and the test set. This phase is named "closed-world" since the classifier does not deal with unknown locations or unknown websites.

The second phase focuses on the performance of distinguishing YouTube traffic in an "open-world" scenario. In this phase, unlike in the closed-world scenario, a classifier is trained using traces collected from client #1 but tested with traces collected from client #2. Also adding to the difficulty of the problem, the non-video traffic traces in the test dataset are collected from a completely different list of websites than those in the training set. The goal of this two-phase setup is to best mimic the real-world situation that the detection engine will face: classifying traffic from a large number of websites to many internal clients with only a limited set of traces to train the classifier.

We also present the performance of traffic collected across different physical sites/places. The features for training are extracted from the data obtained from client #1 according to the methods introduced in Section 5.4. The trained classifiers are then tested on features extracted

from data obtained from client #2. This is an open-world scenario since not all the clients are known to the classifier apriori.

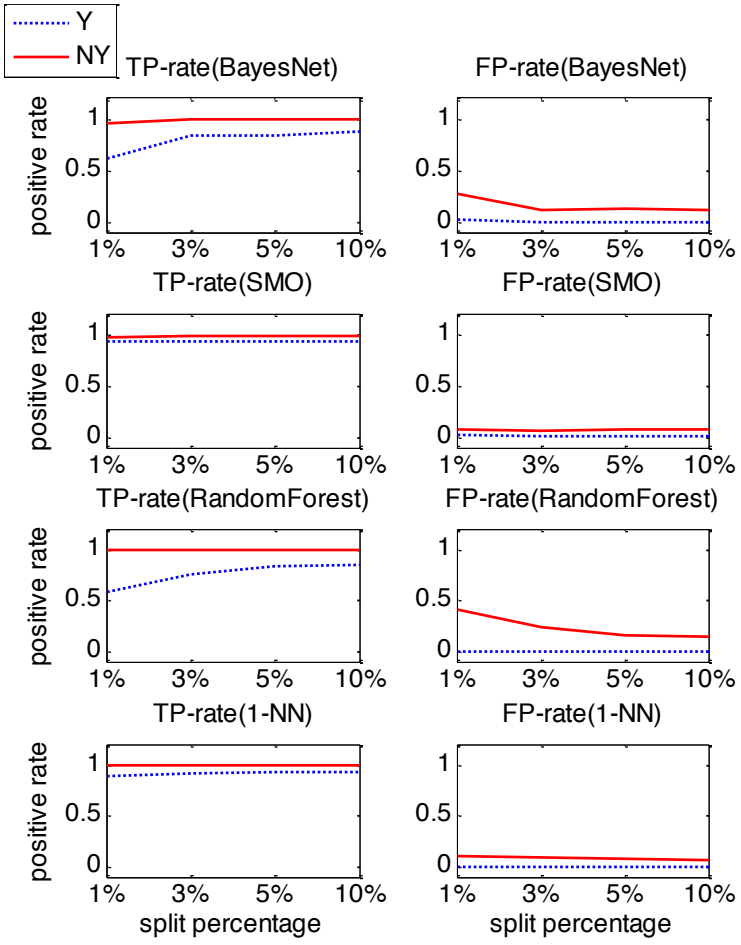



Figure 6-5: Results using packet size distribution feature

Results using packet size feature: Results obtained from dataset #1 are shown in Figure 6-5. The following abbreviations are used: TP-rate = true positive rate; FP-rate = false positive rate; Y stands for YouTube class while NY stands for the Non-YouTube class; All other figures in this section follow the same convention.

The "split" percentage indicates how much of the data set is used to train the classifier. The entire dataset contains 4019 samples, of which 546 samples are collected from YouTube (1

sample = 1 minute). By splitting 1% for training, the training set contains only 40 minutes of data, of which 6 minutes is from YouTube. The results show that packet size distribution has good generalization performance in the closed-world scenario even when the training set is small. Only 3% of data (120 minutes in all 17 minutes from YouTube) is enough for the classifier to yield a good result on the rest of the dataset. The problem with having a small training set is that the false positive rate (FP rate) tends to go up. The figure of the FP rate for YouTube class needs special attention because we want to minimize the innocent victims of the classifier. We can see from the results that BayesNet classifier and 1-Nearest Neighbor classifier yield lower FP rate than SMO classifier and J48 tree classifier. Again, a 3% split is enough for these two classifiers to yield FP rate as low as $10^{-3}$.

Results using consecutive packet size pair feature: Following the analysis done in Section 5.4, consecutive packet size preserves the temporal relationship of packet sizes and is expected to be effective in distinguishing the traffic of different sites. The results of experiments with this feature are shown in Figure 6-6.

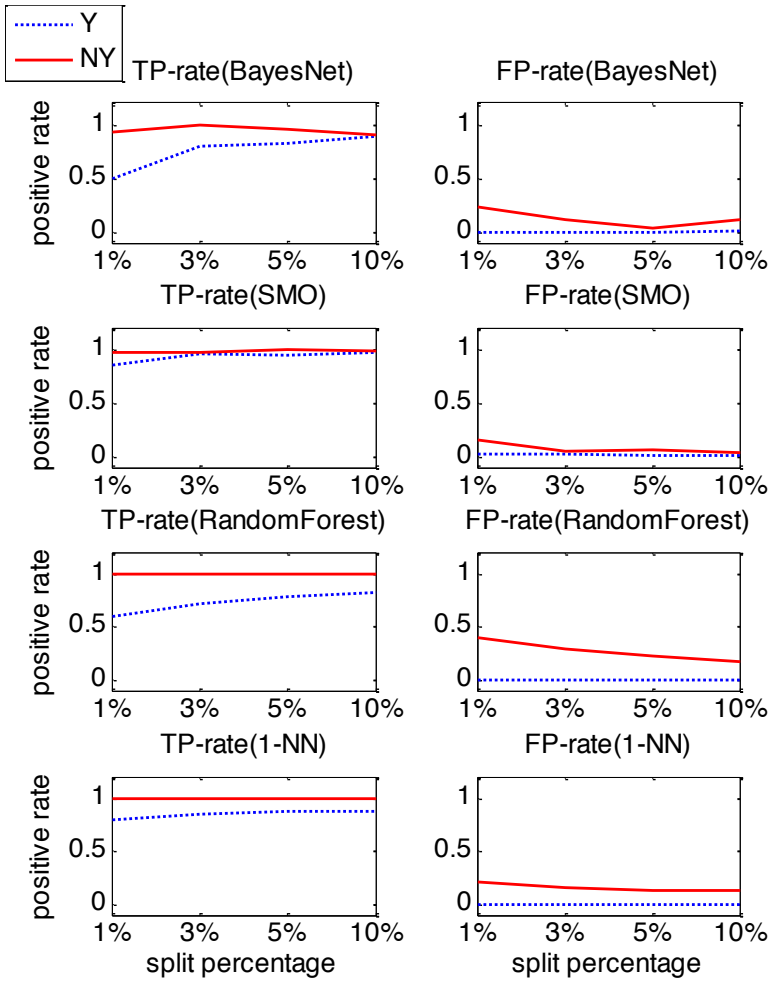



Figure 6-6: Results using consecutive packet size pair feature

The results show that while still effective, the feature performs slightly worse than packet size distribution. The added temporal information does not contribute to the recognition rate.

Results for different locations/sites: In Figure 6-2, we introduced the experimental setup, which includes 2 clients at two different physical sites. The goal of this setup is to evaluate whether the classification/detection can still work when a classifier is trained with data from a specific physical location in the network, and test data is applied from a different location.

We first collect video traffic data on client #2, then collect web traffic data on client #2 on the next day. This formed a data set that consists of 1219 traces, of which 408 are from YouTube,

309 are from Hulu, and the rest is web-browsing traces from various non-video websites. Netflix data is absent here because of a technical issue. None of the non-video websites is included in the site list when we collect data from client #1. Combined with the unknown client, this makes the classification problem more difficult. Packet size is extracted from the dataset for recognition because the previous experiments have shown that it works better than consecutive packet size pair. Additional tests also revealed that the positive rate for consecutive packet size pair, in this case, is usually lower than %1, making it unfavorable. We then use data collected from client #1 to train a classifier to apply to the data set we got from client #2. The results are shown in Figure 6-7. Following the convention in the earlier part of this section, we provide results with classifiers trained with different training set sizes up to 10% of the size of data set from client #1.

The performance in this situation is significantly worse than that obtained from the closed-world tests. The result shows an interesting behavior that larger training sets, in this case, do not necessarily result in better results. For SMO and 1-NN classifiers the TP rate of the YouTube class is lower than 30% even with a 10% split, while for the RandomForest classifier, the TP rate for YouTube stays low until the training data size reached 10% split. The problem this phenomenon signifies is that while the increased training set size makes the classifier perform better with data obtained from the same location, it makes the classifier perform worse on data obtained from other places.

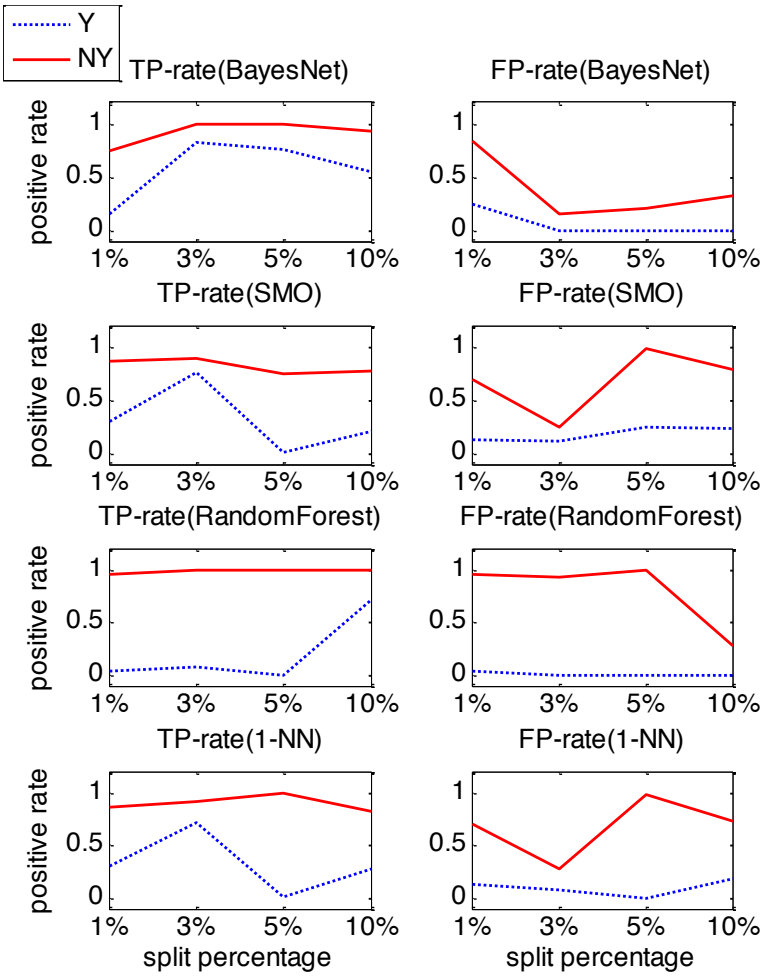



Figure 6-7: Recognition Results for the Open-world scenario

We also investigated the behavior of RandomForest classifier with larger training data sets as large as 30% of the dataset #1, which means more than 1200 minutes of data is used for training. The results show that the recognition rate does not correlate well with the increase in training set size. While the TP rates are not high, the FP rates at higher split percentages are very low for the BayesNet classifier and the RandomForest classifier. This is good news for a detector that wants to detect video traffic. In such applications, the number of samples is so large that misclassifying part of the samples as negative does not affect the performance, while misclassifying legitimate

traffic as positive affects other use of network and is a more severe problem. Another interesting point here is that the most favorable classifiers are different from those of closed-world tests.

### 6.7 Summary

This chapter solves the problem of detecting video streaming traffic tunneled through VPN or proxy servers. It was proven that with Traffic Analysis (TA) based recognition methods, it is possible to recognize video streaming traffic tunneled through a certain VPN service with low false positive rate, even when the classifier deals with traffic from hosts that are unknown in the training process. It was also proven that the packet size distribution feature works better for video streaming protocols when compared to the consecutive packet size pair distribution. The results indicate the possibility of a board range of applications in the field of rule enforcement and surveying. The results also show that TA methods can be successfully applied to protocols other than HTML with decent results when the same-location rule holds.

# Chapter 7 : Two-stage Classifier Design for Mixed-Traffic Problem

## 7.1 Introduction

In this chapter, we present a novel two-stage classifier design that tackles the mixed-traffic problem. The mixed-traffic problem (also called heterogeneous traffic in this work) is defined as the traffic analysis problem when the traffic contains more than one type of protocols. Traditional methods tend to lose performance in the presence of mixed-traffic. However, in this chapter, a two-stage classifier shows that it can classify traffic with higher accuracy at the expense of some discarded samples.

## 7.2 Problem Definition

The mixed-traffic problem in this chapter is defined as the Traffic Analysis of tunneled traffic in which a video streaming traffic flow (the main target for identification) and a web traffic flow (considered as a distraction) are mixed together. The presence of web traffic flow is assumed to be sporadic, so there would be a small pocket of undisturbed video streaming traffic. The two-stage classifier is designed to take advantage of this fact. Experiments are designed to test this hypothesis, as well as verify the design of the two-stage classifier.

## 7.3 Classifier Design

The key part of the architecture is a novel classifier, which is shown in Figure 7-1. This section will explain the components in detail.

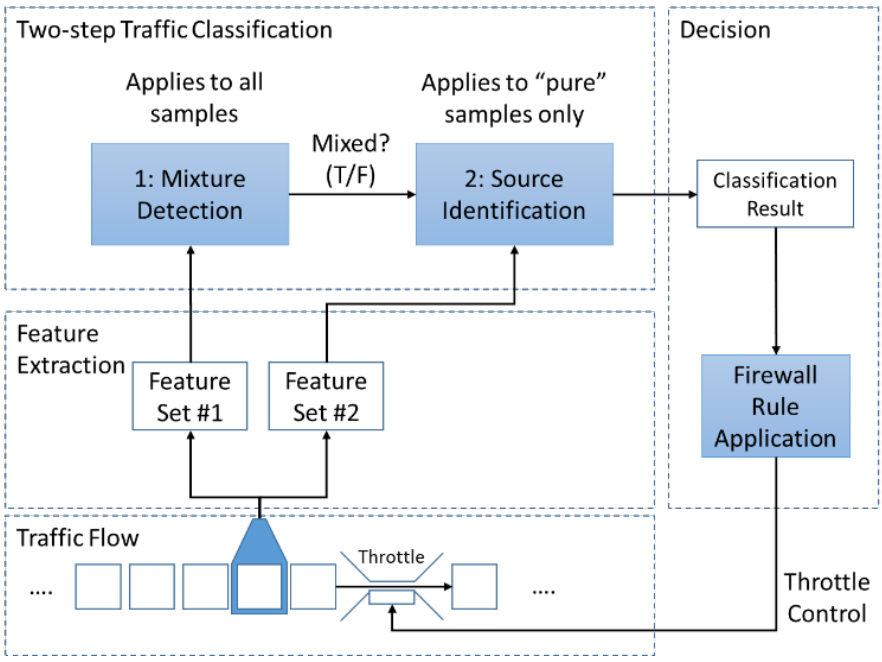



Figure 7-1: Two-stage traffic source classification scheme

### 7.3.1 Two-Stage Classifier

Since the composition of a network traffic flow could change over time, the classifier evaluates traffic in small time slices, which are termed as samples. Each sample is a temporal portion of traffic flow recorded within a certain period. The proposed classifier is split into two stages. In the first stage, a classifier is fed a combination of features extracted from the sampled traffic flow, and it determines if the sample corresponds to mixed traffic flows or a pure flow. For a pure sample, the second stage determines the sample's video source. Only those samples that are classified as pure in stage-1 are passed on to stage-2.

Two conditions are assumed for the classifier to work: 1) the target video streaming traffic forms the majority of the traffic in a tunnel, and 2) non-video streaming traffic only occurs sporadically. These assumptions are based on the observation that video consumption demands a high level of user attention. Therefore, the web traffic will be sporadic. A sample may be mixed

or pure depending on the composition of traffic in the tunnel during that period. We assume 50% of the video watching sessions are accompanied by web browsing activities. We believe these assumptions to be reasonably close to real human behavior.

One goal of the 2-stage classifier is to maximally reduce the chance of false positives. False positives are costlier since they disrupt normal services going through an SMFW. Following the example of [41], it is done on stage-1 classifier by adjusting the cost of false positives to be higher than false negatives, while on the stage-2 classifier, it is done by first converting the n-class classifier to n binary classifiers and then assign a higher cost for the false positive detection (detection of a certain video streaming class when its traffic is not present in data) a higher cost than the false negative detection. Since there are 7 classes in the experiment, the stage-2 classifier is actually an ensemble of 7 binary classifiers. The ensemble is based on a max-confidence rank algorithm, which takes the classifier output that has the highest confidence rank as the final output. For a RandomForest [42] Classifier, that means the output that has the most popular votes wins. The classifier is designed to detect traffic samples from the targeted video streaming providers with a minimal amount of false positive.

### 7.3.2 Feature Set

For each packet within an encrypted tunnel, the SMFW can observe a 3-tuple: {*time*, *direction, size*}. *Time* is the observation timestamp, the *direction* is a binary value indicating the direction of that packet (either from inside the SMFW to the outside, or vice versa), and *size* is the number of bytes in that packet. Converting the raw data to this tuple series is the first step in feature extraction. Although the tuple series is already a lot smaller than the raw traffic data, it can still be quite large, and the raw parameters may lack robustness to compensate for the pseudo-randomness of network traffic. Thus, practical features are usually combinations of one or more statistical

characteristics of the tuple series. Criteria for a good feature set include: 1) distinctiveness: the feature should be able to capture the difference between the various traffic classes, 2) robustness: the feature set should minimize the influence of the inherent randomness in traffic, and 3) ease of computation: it should fit in a resource-limited environment such as a router. From the collected {*time*, *direction, size*} information, the following features are computed.

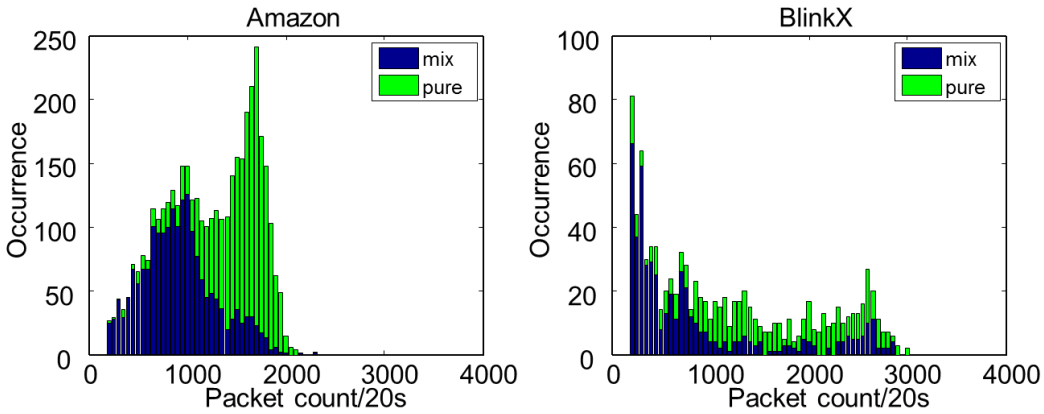



Figure 7-2: Sample distribution of packet counts

Packet Count: The number of packets in each sample is treated as one of the classification features. Data collected from the experiment shows the distribution of packet count to be appreciably correlated to the presence of a mixture in video streaming traffic. A few examples of the difference in packet count distribution between mixed traffic and pure traffic are shown in Figure 7-2. Amazon prime video traffic and BlinkX video traffic collected with or without mixed-in video traffic are shown. While it is more evident in the case of Amazon that pure traffic has a higher packet count, BlinkX traffic also shows the same tendency. This trend has been generally observed for other video sources and different traffic mixes. The difference between pure and mixed traffic is due to rate throttling when a client initiates multiple download sessions. The video streaming client can detect that the network interface is crowded and choose to play a lower bit-

rate version of the video instead, which results in less video streaming traffic and in turn, less overall packet count [43]. The traffic in Figure 7-2 is taken from without the tunnel, and it reflects the difference in traffic clearly.

Packet Size Distribution: Distribution of packet size is obtained by counting the number of packets in a sample as they fall into different bins according to their direction and size. This results in a histogram. In this chapter 30 bins that uniformly divide the range [-1500, +1500] bytes are used to compute the feature. The packet size has a sign associated with it to indicate the direction of this packet (minus is upstream, and plus is downstream). In order to remove bias, before feature computation, all TCP ACK packets are removed from the stream based on the packet size below a threshold of 53 bytes [3]. The feature histogram is then calculated and normalized, i.e. the value of each bin is replaced by its value divided by the sum of all bins. This mitigates the influence of traffic load on the feature.

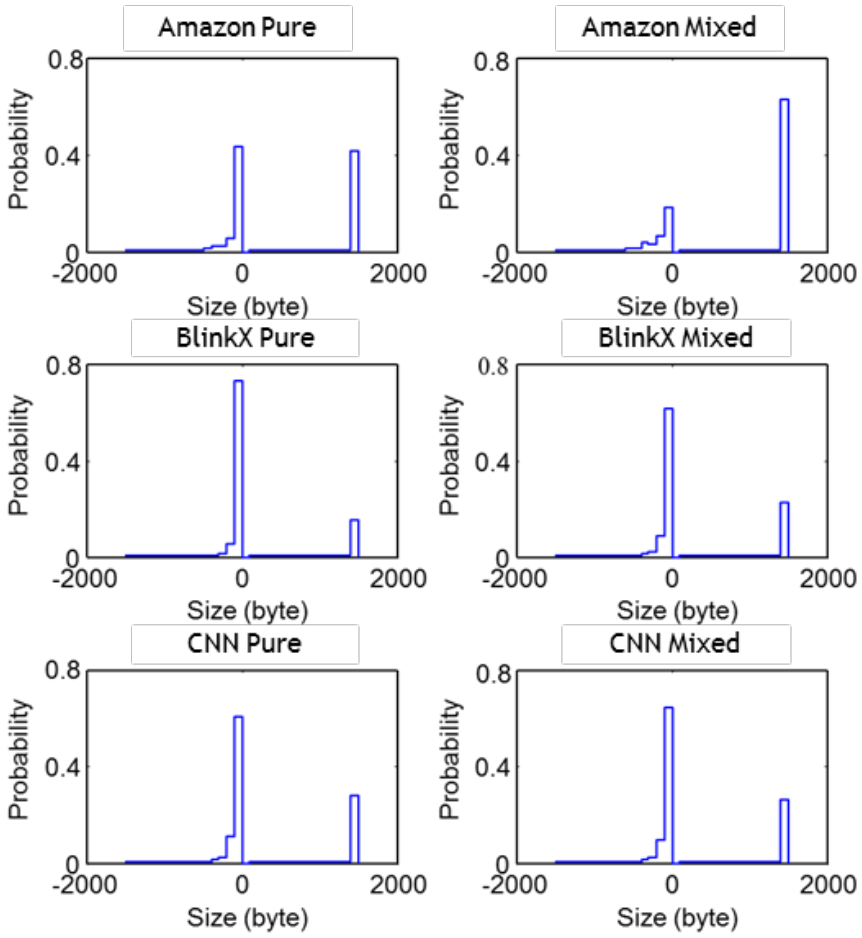



Figure 7-3: Sample packet size distribution

Several sample features that are collected from the experimental environment are shown in Figure 7-3. Samples from 3 different sites both with and without the background traffic (labeled as Mixed and Pure respectively) are shown. The variation of the feature between different sites is clearly visible. Background traffic has an observable influence on the feature.

Packet Arrival Interval (PAI): PAI is defined as the distribution of inter-packet time. In what follows we show that PAI contains useful timing information, which can be helpful in source server classification. Please refer to Chapter 4 for a detailed analysis of the PAI feature.

### 7.3.3 Automated Feature Selection

We use an automated process by which the most relevant features in a feature set can be selected by a feature selector algorithm for a particular classification problem. Due to the high number of features involved in Traffic Analysis, and the low signal-to-noise ratio of some features, feature selection is conducted automatically rather than manually. A naïve approach would be to generate different subsets of the original feature set using all possible combination of attributes, train classifiers based on them and measure the performance of those classifiers against a common test set. The subset that performs best would win the selection process.

However, the cost of such a search would be too high if the original feature set contains many attributes. Therefore, a more practical approach to feature selection involves the use of a suboptimal search of the feature space. Suboptimal search makes the process faster. The search can also be further optimized with a proxy measure chosen as the optimization goal rather than using the classification performance as a goal.

In this chapter, we use the Correlation Feature Selection (CFS) [44] measure to select the optimal features. CFS is only used to on stage-1 classifier because stage-1 needs to run quickly, while the stage-2 classifier uses the full feature since it needs more resolution to classify traffic source. The stages of our classification architecture were summarized in Figure 7-1. CFS constructs subsets based on the assumptions that 1) a good subset should contain mostly uncorrelated features, and that a feature should provide independent information to the subset instead of merely being a combination of other features in the subset, and 2) that features could be weighed individually against the classification problem, and a combination of features that perform well individually indicates a subset that could perform well. We use CFS and the "BestFirst" condition [44] from Weka [34] library for this purpose. Weka is a tool for performing machine-

learning tasks, which is developed by The University of Waikato. It provides common tools for machine learning applications, which includes the CFS algorithm used in this chapter. With the "BestFirst" condition, the CFS selector will greedily search the attribute space for the next attribute that has the best evaluation of the selected subset of attributes. The algorithm also backtracks to try a different path when there is no improvement for several iterations until a stop condition is reached.

### 7.4 Experimental Setup

To test the performance of the proposed classification scheme, a series of experiments are set up as follows. An OpenVPN server [17] is installed on the campus of Michigan State University. The detail of the setup is shown in Figure 7-4. The videos are streamed from multiple streaming providers to a client location that is set off campus (so that the client is in a different domain than the VPN server). In OpenVPN version 1.5, a tunnel can be either UDP or TCP based. We use the UDP tunnel because it is more efficient and therefore used by more service providers. An encrypted OpenVPN tunnel is created from the client to the VPN server, and then the videos are streamed from commercial streaming servers such as YouTube, Netflix, etc. (see Table 4-1) to the client via the OpenVPN tunnel.

A router with Wireshark [39] probe is installed at the client's home. The traffic is collected at this router. The setup emulates a real-world operational scenario of the proposed SMFW. Numerous video streaming servers are chosen for data collection during the experiment.

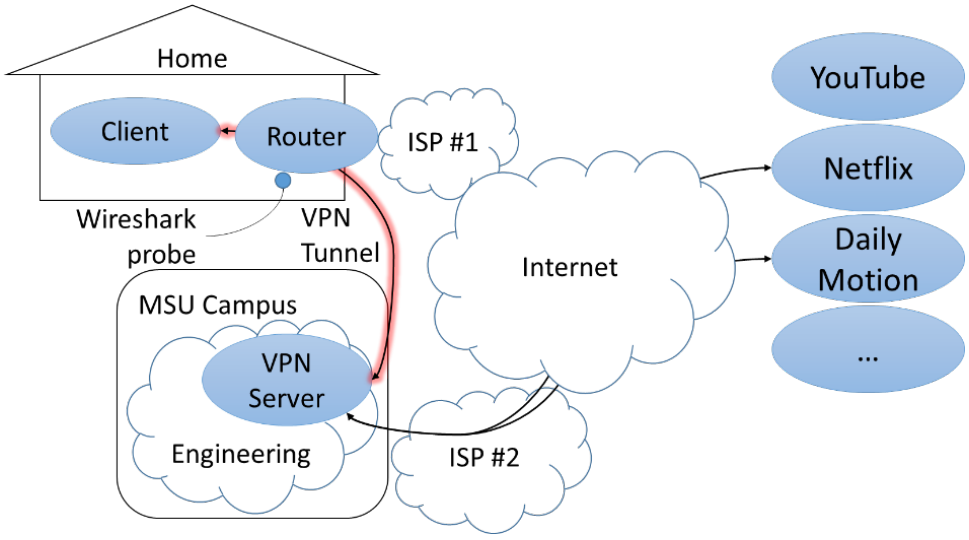



Figure 7-4: Experimental setup with OpenVPN

Most of the video streaming servers use a content distribution network in order to speed up service delivery and save cost. To enable that, the actual server machine that a client downloads from is a function of the client's location, network load, and various other factors. Servers from our experiments are resolved and matched in an IP geolocation dataset to reveal their actual location. The number of samples collected during the test period is shown in Figure 7-5. The difference in sample size from one site to another is a result of the length of the chosen video stream (each sample is 20 seconds long) and the condition by which samples are filtered (samples with less than 1500 packets/sec are discarded).

The dataset is collected over a 6-week period at the client location. The set contains 147563 samples, each 20 second long. Normal Internet usage by human participants within the same period is sampled as "Profile" traffic, which provides a background for the samples to be compared against. The data is collected using an automated script that generates random web page views during video playbacks. The average rate is set to be 1 view/minute, and 50% of the video streams are randomly chosen to have mixed-in web traffic.

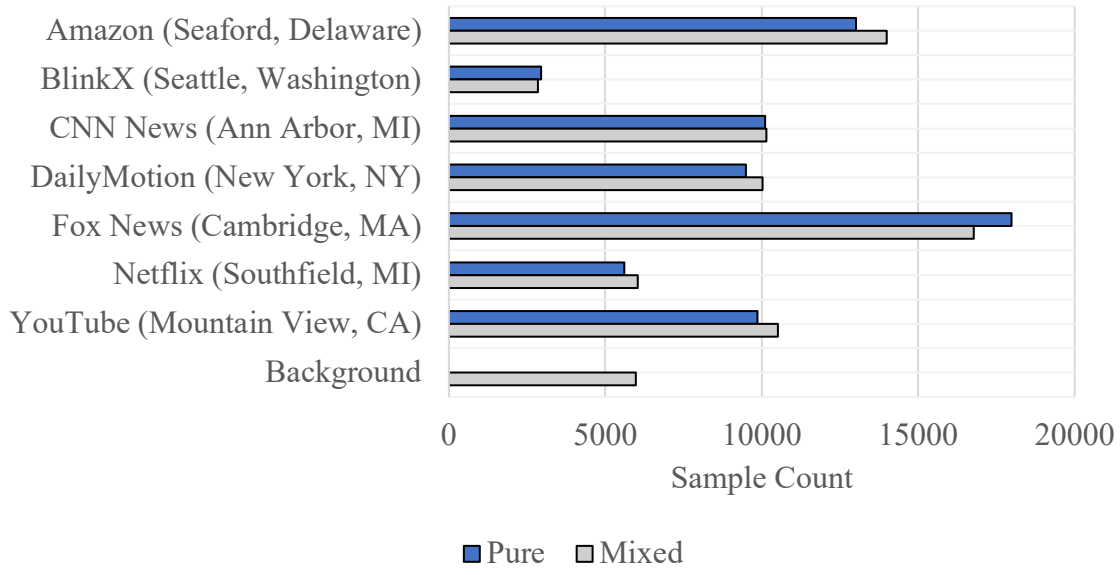



Figure 7-5: Video Streaming Providers in Dataset

Data Processing: All traffic streams collected from the experimental setup are cut into 20-second slices and a feature vector is extracted from each slice. The feature vector contains packet size distribution (30 attributes), the packet count (1 attribute) and PAI (40 attributes) of each sample. The samples are collected either with or without background traffic. The resulting dataset is tagged with 2 labels, one is whether the dataset is collected with or without background traffic present, and another is the video streaming site the traffic originates from. Packet count, packet size distribution, and PAI features are extracted from every sample, then the features are concatenated into a feature vector the format of which is shown in Figure 7-6.

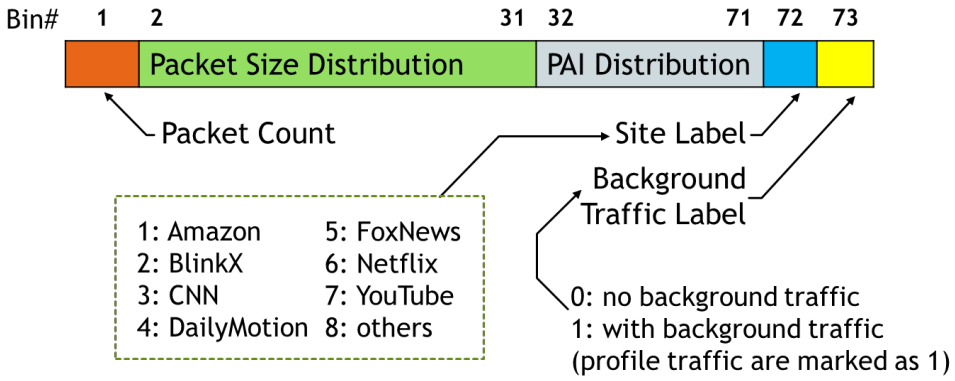



Figure 7-6: Feature formatting with ground truth tags

For a fair comparison, all classification models in this experiment are trained with RandomForest classifier from the Weka machine learning library. A RandomForest [42] classifier is an aggregation of tree classifiers that form a "forest". Each tree in the forest is trained on a random subset of the attributes. The classifier outputs are the result of a popular vote of all the tree classifiers. A tree classifier is a type of classifier which memorizes the training set as a hierarchy of conditions. A sample is classified by starting at the "root" of the tree, which contains the broadest condition, then following the conditional branches that match the sample, until the evaluation reaches one of the leaves of the tree. The label associated with that leaf is the output for the sample. The dataset is separated into 2 different partitions. One partition, which contains only "pure" samples and "profile" samples, is for the training of the stage-1 classifiers, while the other, which contains "pure" samples, "mixed" samples, as well as "profile" samples, is used for training stage-2 classifiers.

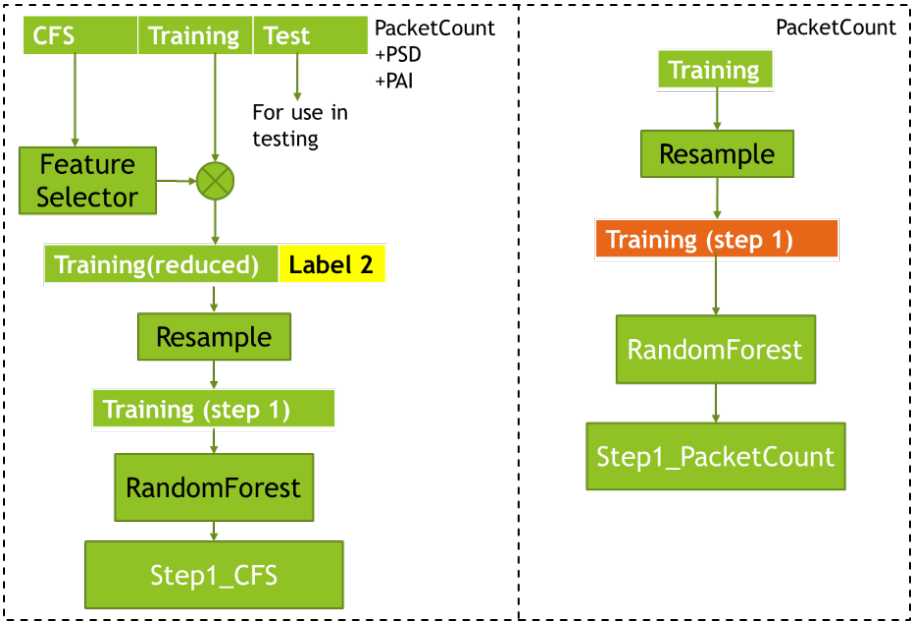



Figure 7-7: Training the stage-1 classifier

Stage-1 classifier (Figure 7-7) solves a binary classification problem of distinguishing traffic samples with background traffic from those samples that do not have background traffic. Before training, the partition for stage-1 is further split into 3 sub-partitions: one for CFS feature selection, one for training, and one for testing. A feature selector is first trained using the CFS partition. The 2 candidates of the stage-1 classifier are trained using different subsets of attributes chosen from the training set. The first candidate is trained using the packet count feature alone, and the 1st label (video streaming site) is erased. The second candidate is trained using an optimal subset of attributes selected by the feature selector, with the 1st label also erased. To decrease the chance of misclassifying mixed samples as pure samples, the cost of misclassifying mixed samples as pure is set to be 1.5 times the cost of misclassifying a pure sample as a mixed one. The data flow for training stage-1 classifier is shown in Figure 7-8. In the following text, the 1st candidate classifier will be referred to as *Stage1_PacketCount,* and the 2nd candidate will be referred to as *Stage1_CFS*. The test set will be used in the testing procedure, described in Figure 7-10.

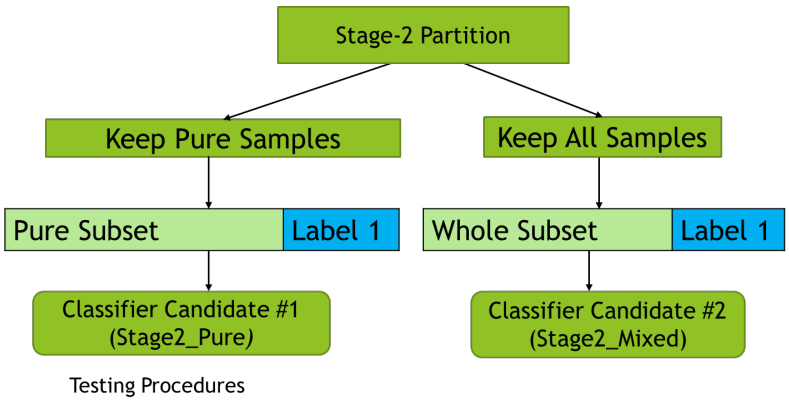



Figure 7-8: Training the stage-2 classifier

After passing the first stage classifier, samples are either labeled as "mixed" or "pure". The stage-2 classifier will only examine the samples labeled as pure by the stage-1 classifier. Two candidates for the stage-2 classifier are trained. The first candidate is a classifier that is only trained with the subset of all pure samples from the stage-2 partition, and the second classifier is trained with the entire stage-2 partition. Both candidates are comprised of 7 RandomForest classifiers as specified in section 7.3. Each of the RandomForest classifiers is cost sensitive. The cost of making a false positive prediction is 10 times that of making a false negative prediction. Finally, the prediction of a sample is determined by the "winning" classifier, which is the classifier that 1) made a positive prediction, and 2) made the prediction with the highest confidence among the classifiers. Also, since there is a default class in the training set (class 8 which are comprised of "profile" samples) every sample that is rejected by all other classifiers is regarded as class 8. Henceforth, the candidates will be called *Stage2_Pure* and *Stage2_Mixed* respectively. The data flow for training the stage-2 classifier is shown in Figure 13.

The experiments are run from a personal computer with an AMD 7700K CPU. The Java heap size for running the Weka classifiers is 1GB. The runtime for training stage-1 classifier is about 165 seconds for the CFS dataset, and 22 seconds for the packet count dataset, while the

stage-2 classifiers range from the shortest 75 seconds (BlinkX) to the longest 367 seconds (YouTube) depending on the size of the dataset. Once trained, the evaluation time of both the stage-1 classifier and the stage-2 classifier on the test set, which contains 29512 samples, are within a few seconds.

Because there are two classification stages, the performance evaluation is more complex compared to a usual single-stage classification scheme. First, the same test set is used in testing both stages. The stage-1 classifier will receive a selected subset of the attributes (Packet Count or CFS chosen) while the stage-2 classifier will receive the full range of attributes. Second, the result from the stage-1 classifier will have an impact on the stage-2 classifier, so the test set has to be separated into subsets based on predictions made by stage-1 classifier. From here on,  is defined as the entire test set,  is defined as the part of the test set that is pure, according to the ground truth, while  is the part of the test set that is pure according to the prediction of the stage-1 classifier. Similarly,  and  are the mixed part of the test set according to the ground truth and prediction of stage-1 classifier respectively.

The performance of the stage-2 classifier has to be gauged in a way that respects the output of the stage-1 classifier. The overall process is shown in Figure 7-10. After the samples are tagged by the stage-1 classifier, both  and the stage-2 classifier will be tested twice, first time with  as input and second time with  in the test separately. Note it that the design goal of the 2-stage classifier is to filter out unqualified samples and as such, only the result of the stage-2 classifier on  is the actually intended input for the stage-2 classifier. However, evaluating the performance of the stage-2 classifier on  allows us to measure its performance gain with respect to the scenario where there is no stage-1 classifier. The output of the stage-2 classifier is the estimated label of the sample

(possible values are listed in Table 4-1). We define the performance indicators in Eqn. (**7-1**)(**7-2**)(**7-3**)(**7-4**).

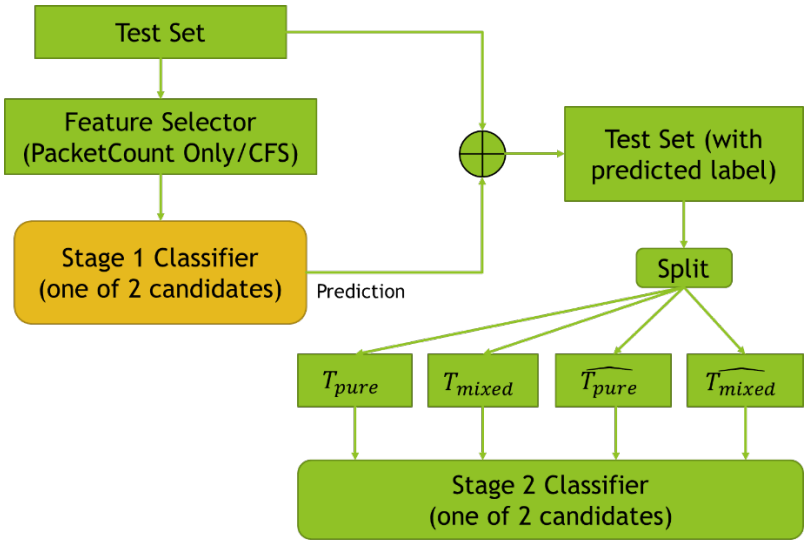



Figure 7-9: Testing the classifiers at both stages

Filtered Accuracy: defined as the accuracy of the stage-2 classifier on .

(7-1)

Unfiltered Accuracy: defined as the accuracy of the stage-2 classifier on .

(7-2)

Filtered False Positive Rate (FPR): defined as the weighted average of the FPR of the 7 classes on. Eqn. **(7-3)**. Here  in the equation stands for the subset of samples that belong to class in.

(7-3)

Unfiltered False Positive Rate: defined as the weighted average of the FPR of the 7 classes on. (Eqn. (**7-4**)). Here  in the equation stands for the subset of samples that belong to class  in T.

(7-4)

The definition of these 4 performance indices is intended to reveal a full picture of the effect of the 2-stage classification process. It is known [23] that mixture leads to lower recognition rate in stage-2 classifier, but it may also result in lower false positive because of removal of potentially noisy (therefore lower in quality) samples early on. It is important to evaluate the performance gain from introducing the stage-1 classifier, which is defined in Eqn. (**7-5**):

(7-5)

This index shows the improvement introduced by the 2-stage classifier, in terms of the reduction of false positive detection rate. The smaller the ratio, the more the stage-1 classifier has contributed to the reduction of false positives.

## 7.5 Experimental Results

### 7.5.1 Feature Selection

Correlation Feature Selector (CFS) algorithm is first applied to the CFS slice in the stage-1 dataset. The results of the CFS selector is then applied to the stage-1 training set and also saved for the testing stage since the same selector has to be applied to the test set as well. Streaming-site label (see Figure 7-6) is erased when invoking CFS. The attributes chosen by CFS for stage-1 is shown in Figure 7-10. Each square block in the figure represents one attribute in the combined feature vector, and those marked as black are the ones chosen by the CFS algorithm. The CFS

algorithm chose the features partly from the *packet size distribution* and partly from the *packet arrival interval* to form the optimal subset. Packet count is not included in the selected subset.

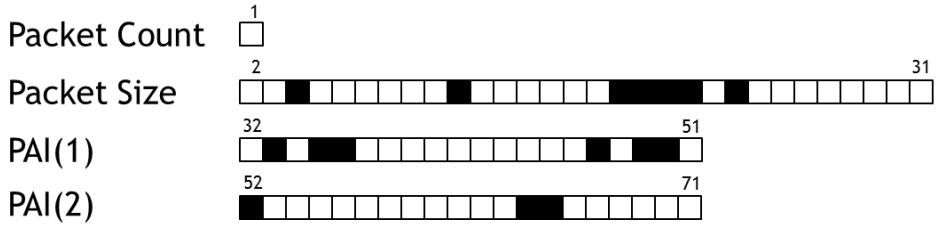



Figure 7-10: Optimal feature subset (stage-1) chosen by CFS

An interesting observation is a tendency for the CFS algorithm to focus on certain regions of the PAI feature. In both cases, the chosen subset of PAI concentrates on certain regions. Those regions correspond to the small end of the time scale (< 50μs), and the larger end of the time scale (>0.17s). According to the observations formulated in [25], the smaller end is where the influence of the network condition is coded, while the larger end is where the source characteristic is coded. The fact that CFS algorithm picks up those regions is a corroboration of the observations in [25].

### 7.5.2 Scenario #1 (Stage1_PacketCount+Stage2_Pure)

Scenario #1 simulates the following operational condition in a real-world traffic classification system. When collecting training data from the SMFW, all traces are collected without any background traffic mixed in, and packet count is used as an indicator of the pure/mixed nature of traffic trace in the stage-1 of the classification process. In this case, the confusion matrix of the stage-1 classifier is shown in Table 7-1. The matrix is normalized by the sum of each row, so the table shows true positive/false positive, true negative and false negative rates. Because the

training set is generated differently between the Stage2_Pure and Stage2_Mixed classifiers, there are 2 different confusion matrices.

The accuracy of *Stage1_PacketCount* is 41.33% and the false positive rate is 6.97%. On top of the results of *Stage1_PacketCount*, the confusion matrix of *Stage2_Pure* is shown in Figure 7-11: (a). The ratio of accepted samples is shown in (b) part the same figure. Samples that are not accepted are defaulted to be of class 8. The acceptance rate is low due to the fact that the 2nd stage classifier is trained on pure samples only and is likely to reject ambiguous samples.

The filtered accuracy of *Stage2_Pure*, with the filtering of *Stage1_PacketCount*, is 91.30%, compared to the unfiltered 80.9%, while only 11.95% of the samples passed the filter. The $\eta$ value, in this case, is 18.37. The Accuracies of different subsets are shown in Figure 7-12.

Table 7-1: Confusion matrix (*Stage1_PacketCount*)

| **Test Passes** | | **Mixed** | **Pure** |
|---|---|---|---|
| | As | | |
| **Stage2_Pure** | Mixed | 93.03% | 6.97% |
| | Pure | 85.47% | 14.53% |
| **Stage2_Mixed** | Mixed | 93.03% | 6.97% |
| | Pure | 88.18% | 11.82% |

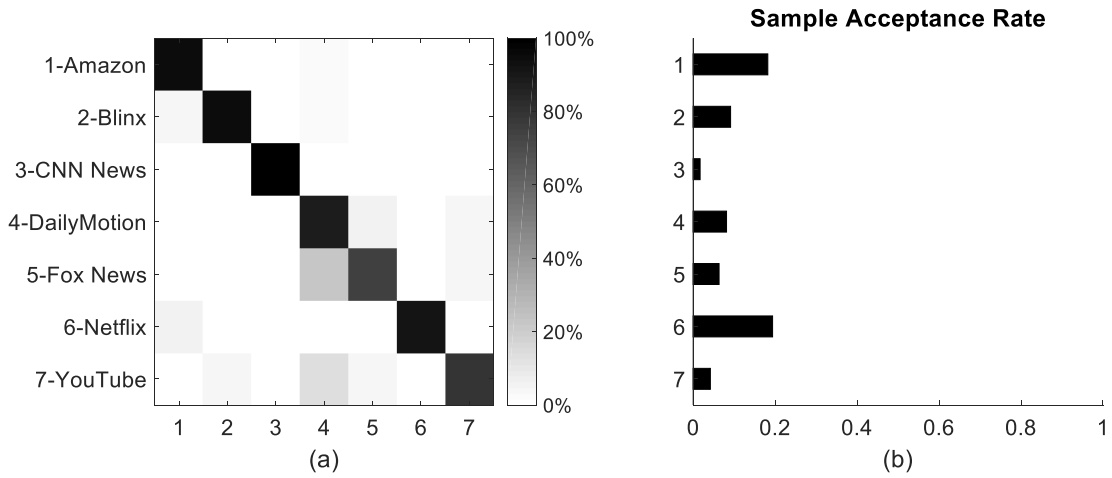



Figure 7-11: Confusion matrix (*Stage2_Pure*)

We can see that while the accuracy of *Stage1_PacketCount* is low, the filtering still boosted the *filtered accuracy* of *Stage2_Pure* from the unfiltered 80.49% to 91.30%, at the expense of passing only 11.95% of the samples to the 2nd stage.

### 7.6 Scenario #2 (Stage1_CFS+Stage2_Pure)

Scenario #2 simulates the following operational condition. When collecting training data from the SMFW, all traces are collected without any background traffic mixed in, and the CFS subset is used as an indicator of the pure/mixed nature of traffic trace in the stage-1 of the classification process. In this case, the confusion matrix of the stage-1 classifier is shown in Table 7-2.

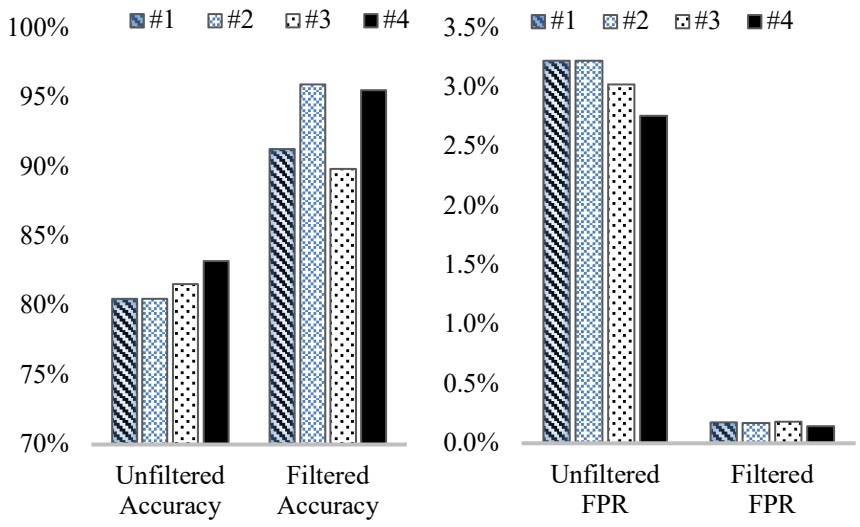



Figure 7-12: Performance indices for 4 Test Scenarios

The accuracy of *Stage1_CFS* is 58.05% and its false positive rate is 2.08%. The confusion matrix of *Stage2_Pure* is shown in Figure 7-11. Accuracies on different subsets are shown in Figure 7-12.

We see that while the filtering of *Stage1_CFS* performs moderately well, the filtering boosted the *filtered accuracy* of *Stage2_Pure* from the unfiltered 80.49% to 95.95%, while only 25.33% of the samples passed the filter. The $\eta$ value, in this case, is 18.96.

Table 7-2: Confusion Matrix (*Stage1_CFS*)

| **Test Passes** | | **Mixed** | **Pure** |
|---|---|---|---|
| | As | | |
| **Stage2_Pure** | Mixed | 97.92% | 2.08% |
| | Pure | 62.62% | 37.38% |
| **Stage2_Mixed** | Mixed | 97.92% | 2.08% |
| | Pure | 74.90% | 25.10% |

### 7.6.1 Scenario #3 (Stage1_PacketCount+Stage2_Mixed)

Scenario #3 simulates the following operational condition. When collecting training data from the SMFW, all traces are collected with background traffic mixed in, and packet count is used as an indicator of the pure/mixed nature of traffic trace in the stage-1 of the classification process. In this case, the confusion matrix of the stage-1 classifier is shown in Table 7-1 since it's the same *Stage1_PacketcCount* classifier. The accuracy of *Stage1_PacketCount* is 32.50% and the false positive rate is 6.97%. The confusion matrix of *Stage2_Mixed* and acceptance rate of samples are shown in Figure 7-13(a) while the acceptance rate is shown in part (b) of the same figure.

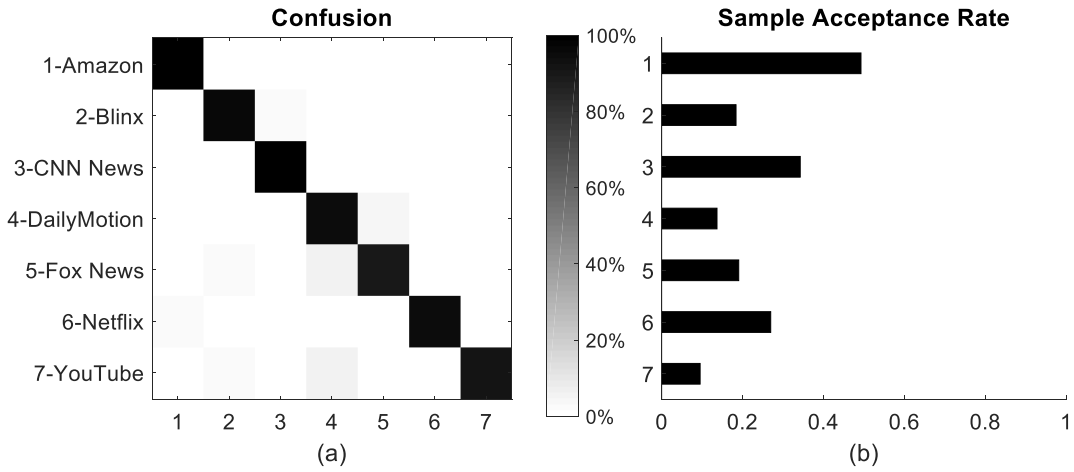



Figure 7-13: Confusion Matrix (*Stage2_Mixed*)

It can be observed that *Stage2_Mixed* accepts more samples than *Stage2_Pure* while simultaneously has less confusion. The performance indices are shown in Figure 7-12. The filtering boosted the *positive accuracy* of *Stage2_Mixed* from the unfiltered 81.57% to a higher value (89.88 %), while only 10.58% of the samples passed the stage-1 classifier. The $\eta$ value, in this case, is 16.79.

### 7.6.2 Scenario #4 (Stage1_CFS+Stage2_Mixed)

Scenario #4 simulates the following operational condition. When collecting training data from the SMFW, the collection agent adds background traffic to the tunnel, and the optimal subset of feature chosen by CFS algorithm is used as an indicator of the pure/mixed nature of traffic trace in the stage-1 of the classification process. In this case, the confusion matrix of the stage-1 classifier is shown in Table 7-2 since the first stage classifier is also *Stage1_CFS* here.

On top of the results of *Stage1_CFS*, the confusion matrix of *Stage2_Mixed,* including both accepted samples and rejected samples, is shown in Figure 7-13. The overall accuracy of *Stage2_Mixed*, if without the filtering of *Stage1_CFS*, is 83.22%. Accuracies on different subsets are shown in Figure 7-12. We see that while the filtering of *Stage1_CFS* only performs moderately well, the filtering boosted the *filtered accuracy* of *Stage2_Mixed* from the unfiltered 83.22% to 95.56%, while only 33.16% of the samples passed the filter.

### 7.6.3 Discussions

Comparing the 4 different scenarios, it could be observed that with appropriate tuning, the 2-stage classifier configuration can boost the performance of the RandomForest classifier to have higher accuracy and significantly lowers false positive rate. As can be seen in Figure 7-12, the combination that got the most boost is the Stage1_PacketCount + Stage2_Mixed pair. But this comes at the expense that Stage1_PacketCount passes fewer samples to the $2^{nd}$ stage than Stage1_CFS does. This low pass rate will affect the response time of the classifier since it takes more time for the classifier to make a decision. In this regard, the CFS classifiers perform much better. The best performance is achieved by scenario #4 (Stage1_CFS+Stage2_Mixed). The reduction rate ($\eta$) in that case is 16.79, and the absolute false positive rate is as low as 0.14%. The performance of the filtered classifier on a VPN tunnel is comparable to that of [2], which assume

homogeneous traffic. The conclusion is that the classifier can choose the subset of mixed-traffic data on which performance like that of pure traffic data can be achieved.

### 7.7 Summary

This chapter proposes and analyzes a TA framework for identifying sources of tunneled video streaming traffic. The main contribution is a two-stage classifier that combines the power of a pre-filter classifier, which filters traffic samples according to their pureness (i.e. whether the traffic is heterogeneous or pure) with a video source classifier. Using OpenVPN servers for creating encryption tunnels, extensive experiments were conducted on many popular video streaming sources with various combinations of feature extraction and data processing techniques to verify the effectiveness of the 2-stage classifier. The results confirm the effectiveness of the design. It was demonstrated that the system works best when an optimal feature subset was chosen using a Correlation Feature Selector (CFS), and the training data contains mixed traffic samples. Accuracy of up to 95% is achieved using this setup. Future work includes: a) studying of the model of background traffic in order to assist training noise addition, b) handling multiple video streams, probably from multiple machines in addition to the background web traffic, and c) verifying the design of the classifier in a custom-built SMFW setting that closely reflects real-world scenarios. An implementation of the proposed SMFW system that can detect a mixture of video streams in the traffic flow is being developed and experimented upon. Future work will present it in more detail.

# Chapter 8 : Deep-Learning Based Traffic Analysis

## 8.1 Introduction

In this chapter, we introduce a Natural Language Processing (NLP)-based feature and its usage with deep-learning based traffic classifiers. The introduction of deep learning methods into Traffic Analysis will profoundly change the way the TA methods are applied. So far, the existing methods rely on hand-engineered features with the hope that they will capture some identifiable trait of the underlying traffic flow. These features are usually low in dimensionality compared to the traffic flow. For example, a traffic flow lasting for 30 seconds could have more than 10,000 packets, each can be described by several properties, yet a packet size distribution like the one introduced in Chapter 7 contains only 10 to 40 values. Most of the information is discarded in the feature extraction process. Defining a feature that is good at capturing traits of the traffic flow is a painful trial-and-error process, and finding a good feature requires a substantial amount of resource as well as luck.

With the introduction of deep learning, the machine learning subsystem will be able to handle inputs that are much higher in dimensionality. On top of that, deep learning is good at finding subsets of the input that is good for identifying the classes. In other words, the classifier can do feature selection by itself when it is trained. To exploit this advantage, we need a new way of encoding traffic into a data format that not only keeps more fine-grained information about the original traffic but also easy for deep-learning to handle. This chapter document one such effort by the author.

## 8.2 Deep Neural Network

In this section, we review relevant literature from Deep Learning. We will focus on the application of deep-learning to NLP problems. The rationale behind the decision is that 1) network

traffic flow and text flow are similar in structure, and 2) it enables the classifier to borrow techniques from NLP.

### 8.2.1 Deep Learning in Natural Language Processing

Deep-learning is a family of biologically inspired machine learning algorithms that feature multiple layers of cascading non-linear representations of data that can be trained. The most popular of deep-learning algorithms is Deep Neural Network (DNN), in which the layers are made of artificial neurons that can be trained through error backpropagation [45]. The learning of multi-layer representations is why DNN is "deep" and is also the reason behind its classification power. Each layer depends on the previous layer(s) and therefore can be considered as a feature extractor. Having multiple layers means there is potential in learning a deep hierarchy of features directly from a high-dimensional representation of the data. In other words, instead of relying on manually engineered features, deep learning algorithms can learn features from raw data. For this reason, deep learning classifiers are commonly trained against densely sampled representations of the data. Images and time series, for example, are often used with minimal preprocessing.

For natural language data, two neural network architectures are used: Convolutional Neural Network (CNN) [46] and Recurrent Neural Network (RNN) [47]. A CNN does a convolution operation on the data with a kernel function that is shared throughout the series, the most significant component is then picked as the input for subsequent layers via max-pooling. An RNN keeps internal states so that it can react to input sequences. The states are updated at each input step by a combination of the past state and the current input. CNN and Variations of RNN, (Especially Long Short-Term Memory, LSTM [48] & Gated Recurrent Unit, GRU [49]), have had great success in NLP[50].

The most notable recent development in DNN based NLP is the Attention-Based Network (ATTN). ATTNs define attention as a neural network (potentially deep) that accepts some form of input and outputs a weight factor that regulates how much weight a part of that input has on the output. ATTN can be chained with other networks to regulate their inputs. Researchers have found that adding attention to a network increases its memory capacity and accuracy. As an example, [51] is regarded as one of the state-of-the-art text classification results to date.

### 8.2.2 Word Embedding

An efficient word representation is important for NLP. A straightforward method to map words to vectors is one-hot encoding. In one-hot encoding, the *i*th word in a vocabulary that contains *N* words is converted to an *N*-element long vector, with only the *i*th element being 1 while all others being 0. One-hot encoding is simple to implement, but scales poorly to larger vocabularies and does not maintain information on the correlation between words. Word embedding [52] is developed to reduce dimensionality while maintaining the correlation. A Word Embedding is a unique projection from a vocabulary of N words onto N dense vectors in an M-dimensional space (usually M≤N). The embedding ensures that related words (i.e. those that appear close to each other in the original sequence) also have vector representations that are like each other. An embedding can be trained from a dataset separately from the classifier, and then reused for different tasks to save time. Widely used methods to train word-embedding include word2vec [53] and Global Vectors for Word Representation (GLOVE) [54]. Word-embedding is an example of the NLP techniques that TA can benefit from.

### 8.3 Language-Like Feature for Traffic Analysis

To bridge the gap between NLP and traffic analysis, a language-like feature is defined to encode network traffic. The goal of this feature is 1) achieve high data reduction ratio before the traffic reaches the classifier. Achieving this goal helps reduce the network load when the classifier is eventually deployed to a real-world networking system. 2) Produce a language-like feature that existing NLP techniques can process. And 3) Achieve high classification accuracy.

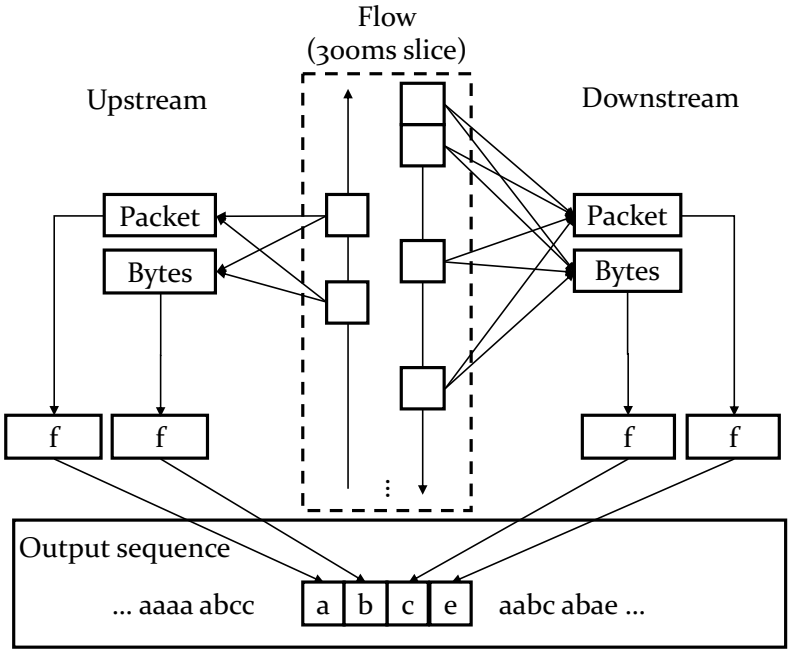



Figure 8-1: Extracting Language-Like Traffic Feature

To achieve the goals, we define the feature as depicted in Figure 8-1. A 1-minutes-long bidirectional traffic stream is split into 200 non-overlapping windows each 300ms long. Within each window, the sum of upstream/downstream packet counts is taken as well as the upstream/downstream byte counts. Each value is mapped onto the English alphabet using the formula defined in Eqn. (8-1).

(8-1)

We use a base-10 log-scale to convert an input value to an index. For a 100Mbps link, which is used in the data collection, the highest possible letter is "g" (the range between $10^7$ and $10^8$). The reason for a log-scale is that it makes the error tolerance in-scale with the absolute values. It also helps to reduce the number of symbols in the feature. The 4 letters are concatenated to form a single symbol ("word") in the output sequence.

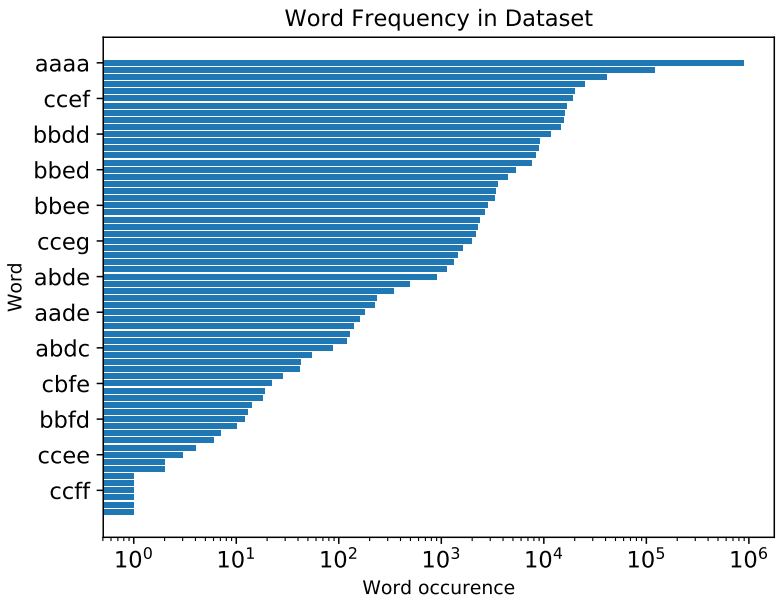



Figure 8-2: Word Frequency of Traffic Feature

After the feature extraction, every sample is turned into a sequence of 200 words. Figure 8-2 shows the word frequency distribution of the dataset. The distribution is sorted by popularity, and every 1 in 5 words is labeled on the vertical axis. We can observe that 1) the resultant feature has a vocabulary of only 68 words, a much smaller number than the number of possible words. 2) The word frequency follows an exponential distribution. Each word is about 83.83% as popular as the previous one. A significant anomaly happens at *aaaa*. The over-representation of *aaaa* is because it is the word that represents an idle channel, and therefore happens more often in the data.

### 8.4 Neural Network Architecture for Traffic Analysis

We verify the effectiveness of the feature with the DNN-based text classification methods listed in Table 8-1. We also choose the traffic classifier by Cruz et. al [31] for our dataset as a comparison.

Table 8-1: NLP Processing Methods Used

| Method | Architecture | Optimizer |
|---|---|---|
| **HAN** | GRU+Attention | Adam |
| **Yoon Kim** | CNN | Adam |
| **Mark Berger** | GRU | RMSProp |

Word2vec embedding is used across all NLP methods. The embedding layer is a part of each network and will be trained alongside it.

The first architecture used is based on Hierarchical Attention Network (HAN) proposed by Zhichao Wang, et. al.[51].

We define 2 types of classification problems in this chapter. The first type is a binary classification problem: given unknown traffic, classify it as either containing traffic from a certain video provider or not. The second type is the multi-label problem: given unknown traffic, identify the probability of it containing traffic from each of the video providers in a list. In both cases, the class labels are video streaming providers, but the outputs are categorical in case of the binary problem, and binary in case of the multi-label problem. We use *binary_crossentropy* loss function to train the models in both cases. The performance is measured by *categorical_accuracy* in the binary case and by Receiver Operating Characteristic (ROC) area in the multilabel case.

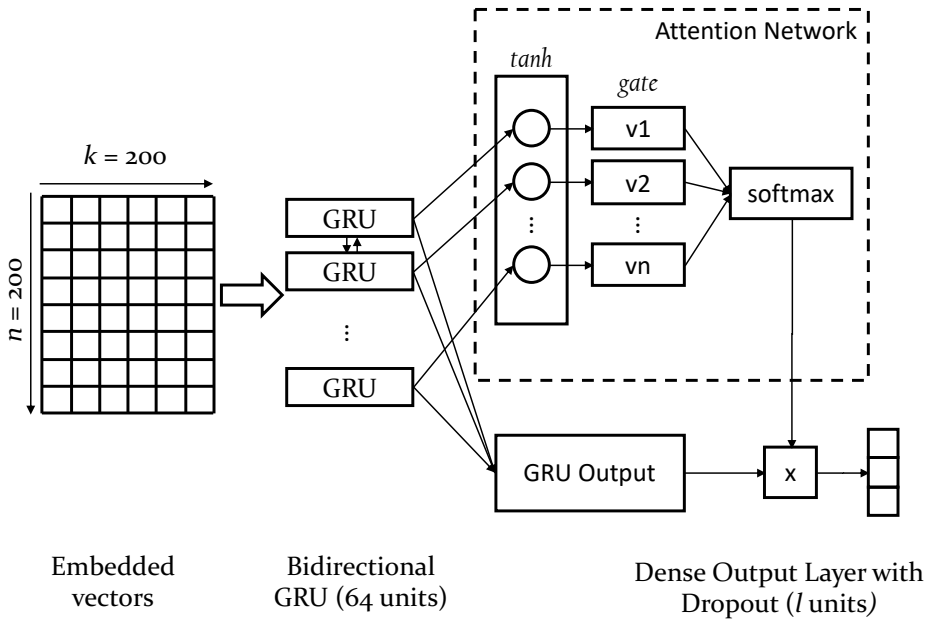



Figure 8-3: Hierarchical Attention Network (Figure 2 in [51])

In this network (Figure 8-3), the output is a weighted sum of the outputs of a bidirectional GRU layer, and the weights are from the output of an ATTN. This ATTN is equivalent to the word-level attention network in the original work. We do not need a document-level ATTN as the original work does since there is no higher-level structure in our dataset. A 200-dimensional Word2Vec embedding is used as inputs to this network and all other network models. The output is a dense sigmoid layer of labels, with the labels either in categorical format or binary format.

The second method used in this chapter is proposed by Yoon Kim [55]. This network (depicted in Figure 8-4) has multiple 1D-CNN layers with ReLU activation, each having a different kernel shape. According to this method, a layer with a kernel shape *n* is defined for *n*-grams in input data. We use kernels with lengths from 1 to 5 to focus on unigrams and up to 5-grams. Each 1D-CNN layer contains 256 units. The input is 200 10-dimensional Word2Vec embedded vectors. The CNN layers are flattened to a dense vector with max-pooling-over-time before connecting to the output layer. The output layer is the same format as HAN.

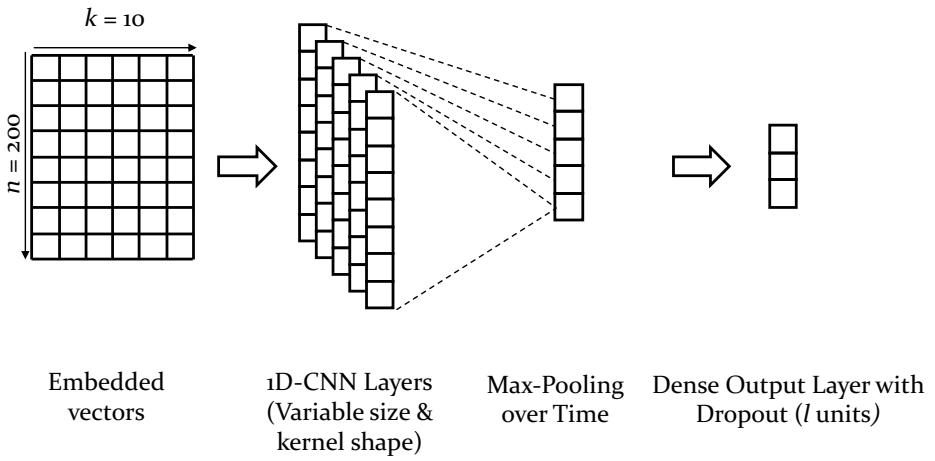



Figure 8-4: Network Architecture by Yoon Kim (Figure 1 in [55])

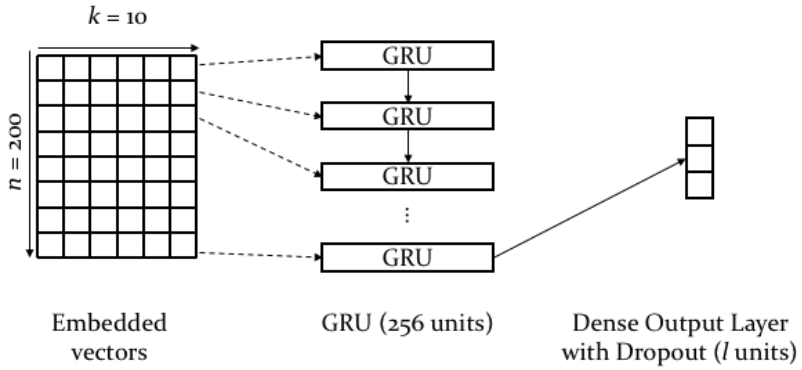



Figure 8-5: Network Architecture by Mark Berger

The third method (Figure 8-5) used is proposed by Mark Berger [56]. The network accepts the same 10-dimensional embedded vectors as in Figure 8-4 and consists of a GRU layer with ReLU activation and a dense output layer that also follows the same format in Figure 8-2.

We feature the traffic classification network by M. Cruz et. al [31] as a recent example of a DNN network designed for TA. The network accepts a vector of traffic features. Each feature vector is calculated from 3 seconds of traffic and contains many components. To adapt it for our dataset the TCP specific features (for example roundtrip time and TCP flags) are discarded because

we do not assume a TCP-based tunneling protocol. The remaining 60 feature values are listed in Table 8-2.

Table 8-2: Available Features by M. Cruz et. al

| Field | Description |
|---|---|
| **Packet count 1** | Bidirectional packet counts of the 1st second of traffic |
| **Packet count 2** | Same as above for the 2nd second |
| **Packet count 3** | Same as above for the 3rd second |
| **Byte count 1** | Bidirectional byte counts of the 1st second of traffic |
| **Byte count 2** | Same as above for the 2nd second |
| **Byte count 3** | Same as above for the 3rd second |
| **Bidirectional packet size stats over a 3-second interval**<br>**&**<br>**Bidirectional inter-arrival interval stats over 3-second interval** | Each stat contains the following values:<br>- maximum value<br>- minimum value<br>- mean value<br>- median value<br>- variation ($\sigma^2$)<br>- extreme outlier (>2$\sigma$) count<br>- mild outlier (>$\sigma$) count<br>- Shannon entropy<br>- 2,3,4 and 5-permutation entropy |

The network (Figure 8-6) contains a dense input layer with ReLU activation, then a chain of 6 LSTM layers with various hidden state sizes that have tanh activation, and another dense layer with ReLU activation before the sigmoid output layer. The LSTM layers (except for the last one) output the entire output sequence to allow them to be chained. The original network uses a binary output format and uses *binary_crossentropy* loss function for the binary classification problem. We change the output layer to use the categorical output (in binary problems only) to match the other networks in this chapter.

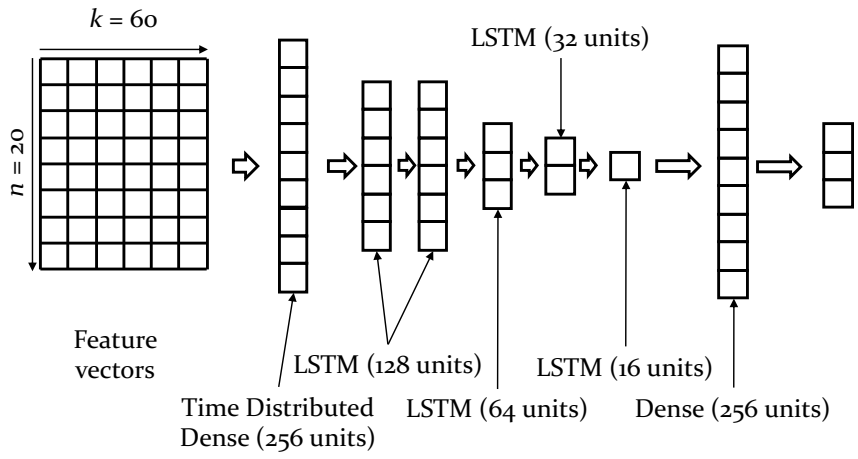



Figure 8-6: Network Architecture by M. Cruz et. al. (Figure 3 in [31])

### 8.5 Experimental Setup

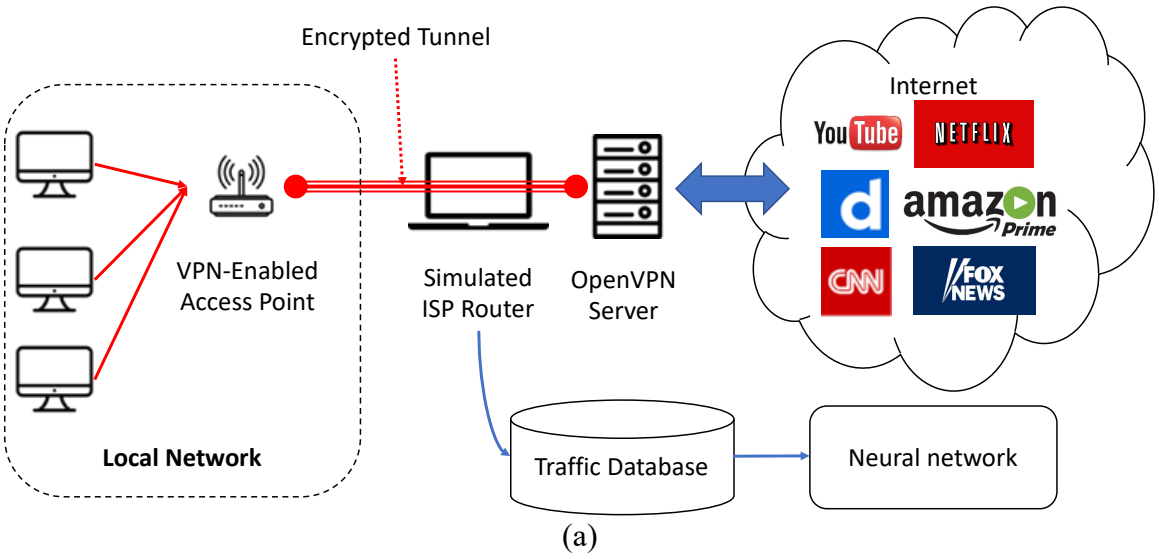
We construct the network shown in Figure 8-7 (a) for the experiments. A Wi-Fi access point runs an OpenVPN[17] client which tunnels to an OpenVPN server. The client and the server talk in the SSL-over-UDP protocol. The tunnel is monitored by a probe router that has the prototype SMFW installed (the architecture is shown in Figure 8-7 (b)) that records metadata for each packet coming through and calculate features for each flow on-the-fly. A brief sample of the recording that the router produces is shown in Table 8-3. Up to 4 clients can connect to the Internet through this tunnel to do different activities. The clients are randomly assigned to watch a video (various length) from one of the six video streaming sites listed in Table 8-4, to access a website (1min/access), or to stay idle for reference. We reserve one client for accessing a web page every minute to create noise in the dataset. Any sample could contain 0 to 3 video streaming traffic flow and some web traffic flow. The tunnel is also used for day-to-day Internet access when data collection is idle, and the traffic from the casual usage is recorded to create a referential "background" data.

Table 8-3: Traffic Data Format

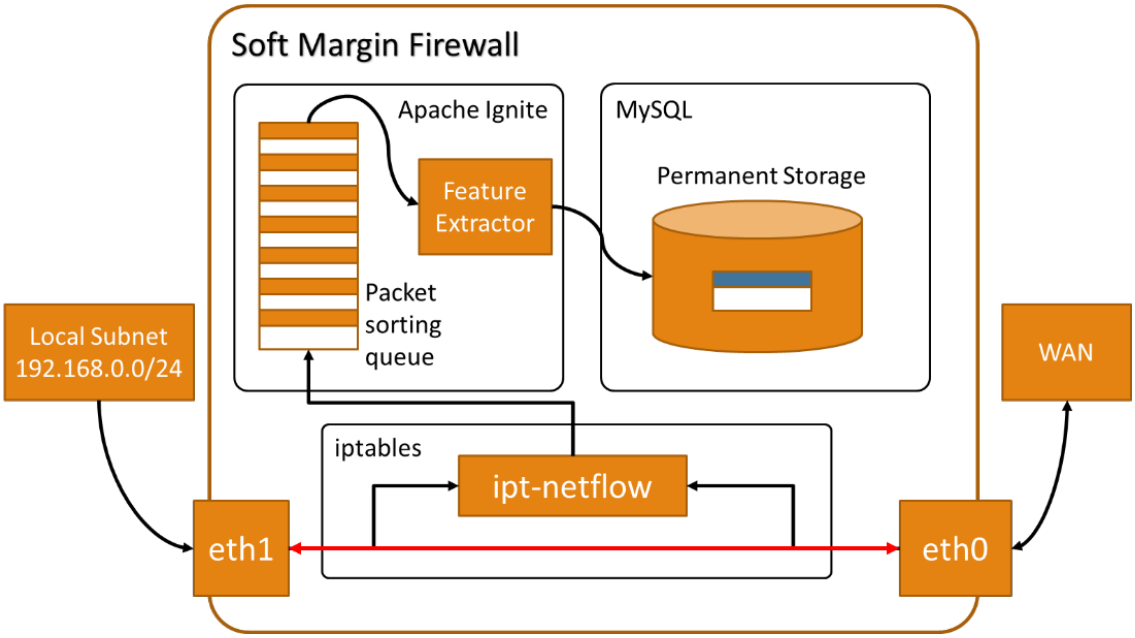
| t(ms) | size(byte) | src ip | dst ip | sport | dport |
|---|---|---|---|---|---|
| **0.000** | 1514 | 54.192.39.46 | 192.168.0.95 | 443 | 59666 |
| **0.060** | 1514 | 54.192.39.46 | 192.168.0.95 | 443 | 59666 |
| **0.180** | 1514 | 54.192.39.46 | 192.168.0.95 | 443 | 59666 |
| **0.590** | 66 | 192.168.0.95 | 54.192.39.46 | 59666 | 443 |
| **…** | ... | … | … | … | … |
| **2.490** | 1514 | 54.192.39.46 | 192.168.0.95 | 443 | 59666 |

The data collection continued for about a month, during which time about 19k 1-minute long samples are collected. A breakdown of the collected data is shown in Table 8-5. The accessed video streaming sites are recorded and later used to assign labels to the samples. Each sample might be assigned 1-4 labels depending on the client activities at the time of recording. 4-label samples are rare in the dataset (28 samples), the reason is that they only happen between the transitions of videos. These samples are then processed with the proposed feature extraction method to be classified by the 3 NLP-based methods. The HAN implementation is by Lei Li from Language Computing and Machine Learning Group (LANCO), Peking University. The Y.Kim/M. Berger implementations are from INSPIRE-HEP project by the European Organization for Nuclear Research (CERN). Features that resemble Table 8-2 are also extracted to be classified by Cruz. et. al.'s classification method.

In all cases, we randomly choose 80% of the dataset as the training set and 20% as the test set. The training set is balanced when evaluating the binary classification problems but is left as is when evaluating the multi-label classification problems.




(a)




(b)

Figure 8-7: Experimental Setup

Table 8-4: Classes based on video streaming sources

| Video Source | Number of videos in the playlist |
|---|---|
| **Amazon** | 2 |
| **CNN News** | 1 (autoplays other videos) |
| **Fox News** | 1 (autoplays other videos) |
| **DailyMotion** | 1 (autoplays other videos) |
| **Netflix** | 2 |
| **YouTube** | 2 (autoplays other videos) |

Table 8-5: Sample counts by No. of Clients

| All | 1 Client | 2 Clients | 3 Clients | 4 Clients |
|---|---|---|---|---|
| **19321** | 8703 | 9128 | 1402 | 28 |

In the following sections, we will study the following aspects of the performance. 1) The accuracy of the method when solving a binary classification problem. 2) Comparison when solving a multi-label classification problem compared with the binary classification case, 3) the performance of the testing samples that bear a combination of labels that are absent in the training set. (This problem is called the "Knockout Problem" from here on) and 4) the performance of the classifier when the training and the testing set are not from the same network location.

The rationale behind aspect 1) and 2) are obvious, while the reason for aspect 3) and 4) is because they allude to the Zero-shot Learning ability of the network. Zero-shot learning a name coined for the ability of a classifier to classify classes not present in the training set). It is a desirable trait because this ability reduces the training overhead of the network greatly by allowing the network to be trained on an incomplete subset of all possible label combinations. Evaluating aspect 4) also let us revisit the same-location rule that is generally established in TA studies and is discussed in Chapter 6 in this work.

## 8.6 Experimental Results

### 8.6.1 Binary Classification

The accuracies of all binary classification experiments are shown in Figure 8-8. The result shows that the newly defined feature is effective. The accuracy in binary problems is defined as follows: if the activation level of the neuron corresponding to the label (0 or 1) is higher than the activation level of the other output neuron, it is considered a hit. The accuracies of the binary classifiers vary from 86.3% to 95.7% depending on the video streaming site being focused on. Cruz et. al performs erratically: its accuracy is 65.3% on Netflix and 64.4% on Amazon, but its highest accuracy reaches 92.3% (on Fox). The newly defined feature performs either comparable or significantly better than the Cruz. et. al. method in all cases.

Table 8-6: Training Performance

| **Method** | **HAN** | **Y.Kim** | **M.Berger** | **Cruz. et.al.** |
|---|---|---|---|---|
| **Epoch Time(s)** | 17 | 35 | 54 | 14 |
| **#Epochs** | 30 | 50 | 50 | 50 |

Table 8-6 shows the time needed for each network to complete one training epoch and the number of epochs needed for convergence. The training/testing is done on a setup that consists of TensorFlow 1.11 running on a GeForce 750Ti GPU with 4GB of video memory. While Cruz. et. al.'s method is the fastest to complete an epoch, HAN is the fastest to converge among the methods, with good classification performance achieved at 30 epochs while other methods take about 50 epochs to reach good performance.

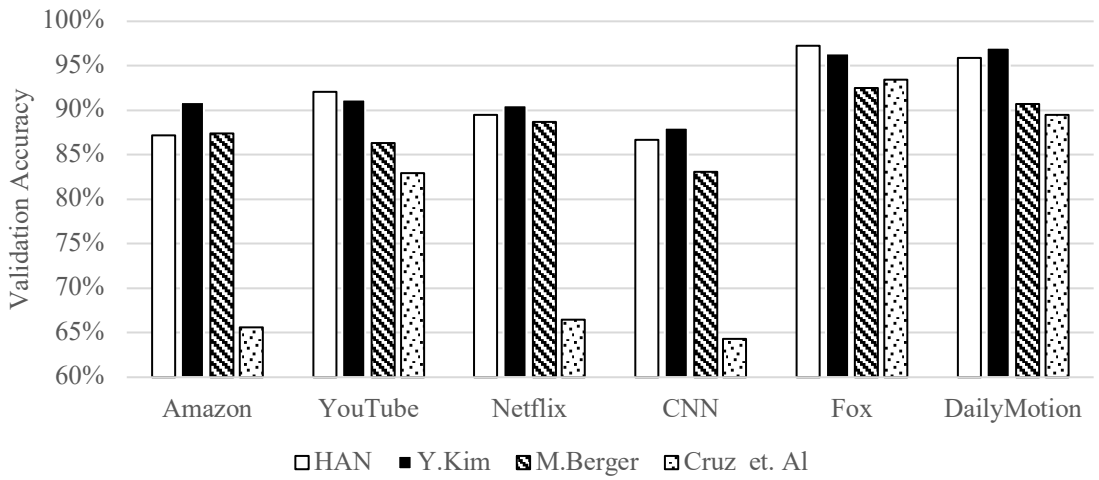



Figure 8-8: Binary Classifier Accuracy

Effect of Word Embedding: We highlight the fact that the embedding size used in HAN method is set to $k$=200, while there are only 68 unique words in the dataset. The choice is based on observations featured in Figure 8-9, which show that the oversized embedding improves the accuracy (from 88.4% when embedding size=10 to 92.1% when $k$=200). The observation is sensible since in machine learning, mapping the features into a higher-dimensional space in order to find better separations between classes is an established practice. Increasing the embedding size further has the opposite effect. Other methods also benefit from a larger embedding size albeit to a lesser degree. For example, Y. Kim's method gains another 1.3% accuracy on YouTube when its embedding size is increased to 200. A large embedding size, however, costs more memory, which is why the embedding sizes for the other methods are set to 10. These observations hint at potentially interesting uses of embedding in TA and other applications.

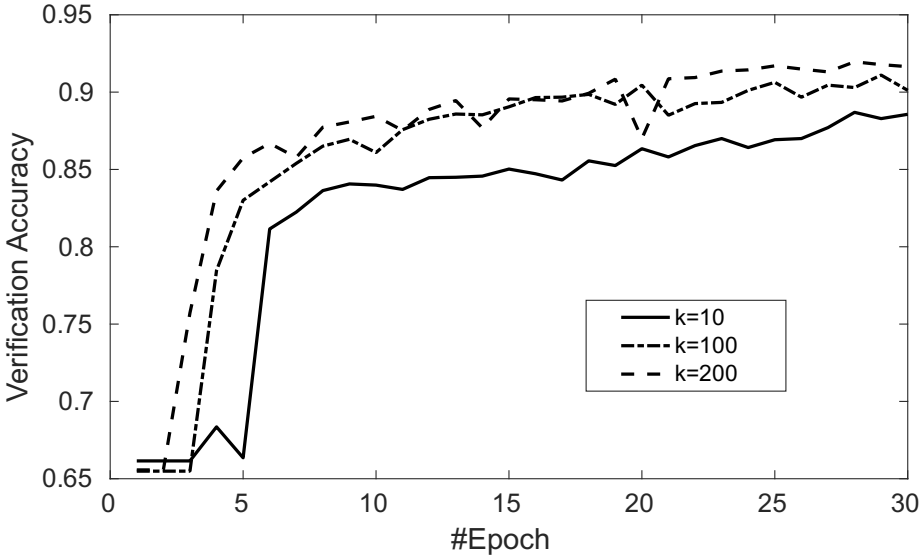



Figure 8-9: Effect of Embedding Size on HAN-based Binary Classifier (YouTube)

### 8.6.2 Multi-label Classification

To evaluate multi-label classification performance we obtain the activation level at the output stage for all of the methods and convert them into ROC curves. The AUCs would indicate the overall performance. We let the classifiers to classify the 6 video streaming sites together.

For comparison, the AUCs of Y. Kim's binary classifiers are provided in Table 8-7. We choose them since Y. Kim's classifier performs the best in the binary experiments. These numbers provide the performance reference for the multi-label classifiers.

Table 8-7: AUC of Binary Classifiers using Y.Kim Method

| Site | Amazon | YouTube | Netflix | CNN | Fox | DailyMotion |
|---|---|---|---|---|---|---|
| **AUC** | 0.984 | 0.941 | 0.883 | 0.887 | 0.984 | 0.937 |

The ROC curves and their respective AUCs of the multi-label classifiers can be seen in Figure 8-10. The horizontal axis in these graphs is the false positive rate, and the vertical axis is the true positive rate. The AUCs can be found in the legends in the figures.

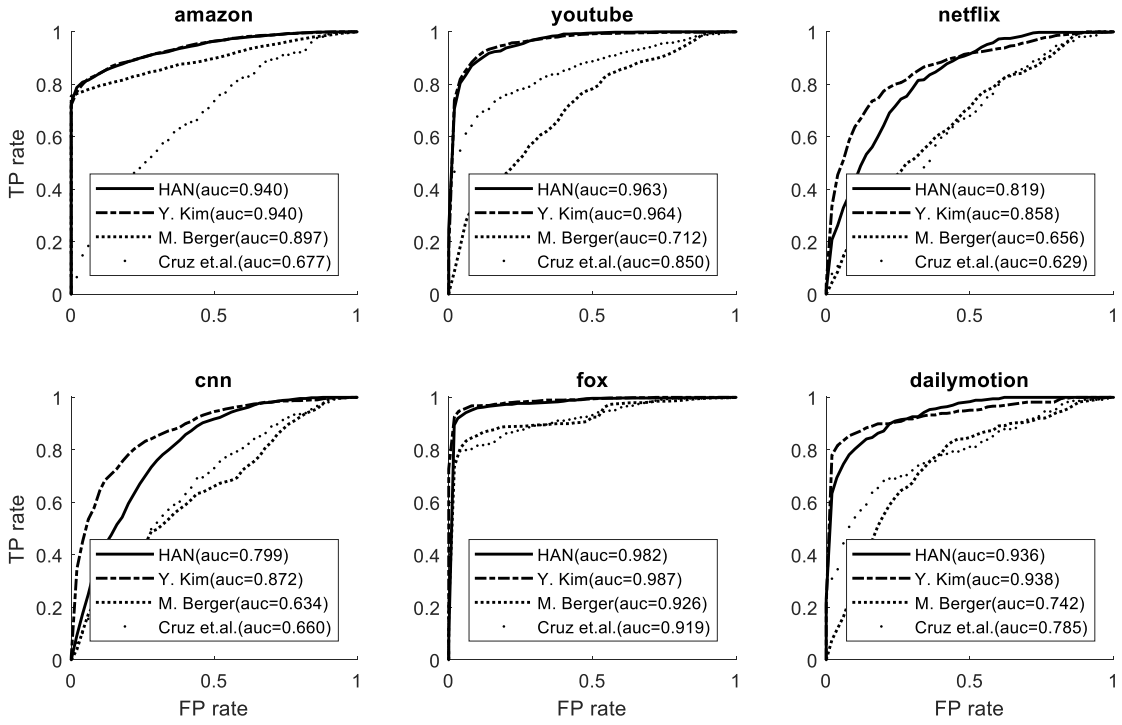



Figure 8-10: Multi-label Classifier ROC

Y. Kim's method again performs well across all classes, with HAN following closely. M.Berger's method performs moderately on Amazon and YouTube while showing declining performance on other classes. Cruz et. al.'s method only performs well enough in one class (Fox). The AUCs of Y. Kim method does not change much from the binary AUC values listed in Table 8-7, proving that the proposed feature is effective in multi-label classification.

Figure 8-12 shows the ROC curves and AUCs of the same multi-label classifiers evaluated on the subset of samples with 3 or more active clients. These samples are considered challenging because of the traffic mixture. We also observe that the AUCs do not change much from those in

Figure 8-10 when we limit the evaluation to the subset of samples with 2 or more active clients. The difference in AUCs of this subset from the whole set is provided in Figure 8-11 for reference. The only significant difference happens on Netflix samples.

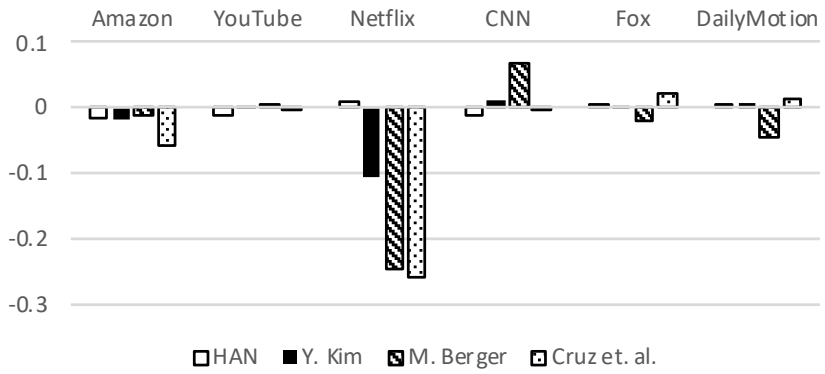



Figure 8-11: Difference in Multi-label Classifier AUC (No. of Client ≥ 2)

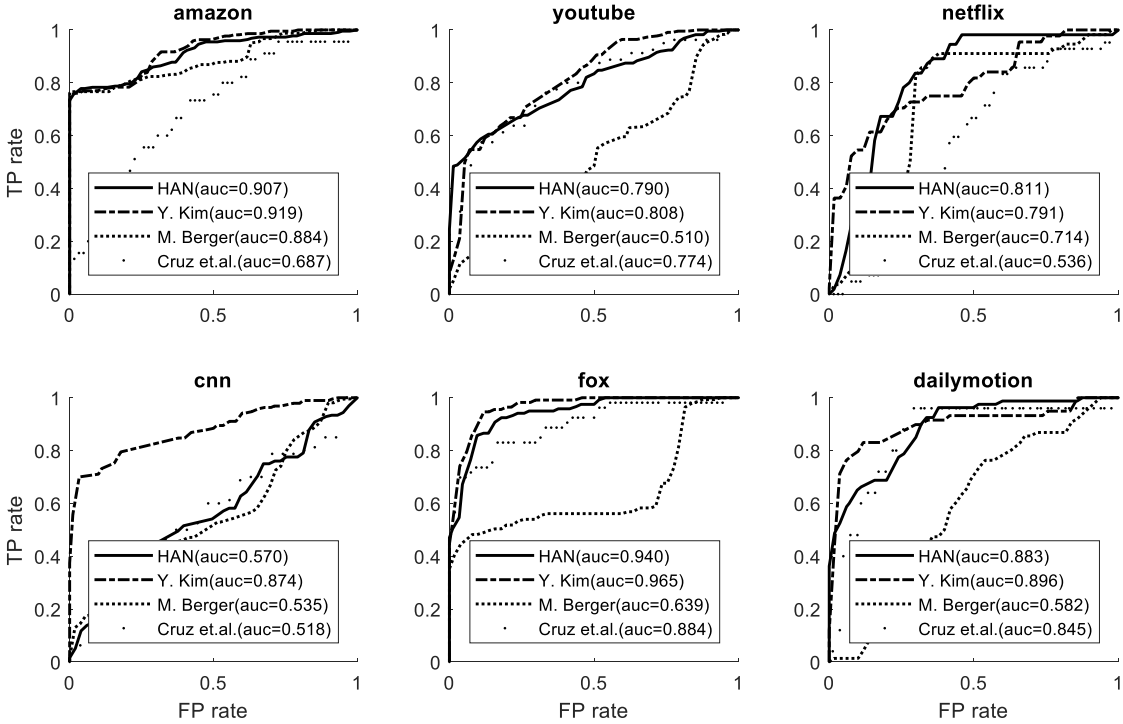



Figure 8-12: Multi-label Classifier ROC (No. of Client ≥ 3)

According to Figure 8-12, the resilience of the classifiers is challenged on the subset where client number is greater than 2. Y. Kim's method again performs well compared to the overall results suffering only minor drops in AUC. YouTube is a major exception, where the method suffered a drop of AUC from 0.964 to 0.808. It still performs better than other methods in all cases. Cruz et al. method achieves better performance on YouTube and DailyMotion but does not perform well on Amazon, Netflix or CNN. The observations here show the effectiveness of the proposed feature applies to the case of multi-label classification.

### 8.6.3 Knockout Test

To evaluate the performance of the classifier, we remove samples bearing certain pairs of labels from the training set, one pair at a time, and train HAN and Y.Kim classifier on the same conditions as the multi-label tests using these modified datasets. The trained classifiers are then evaluated on the augmented test sets that contain those previously removed samples. The ideal output is for both output neurons (corresponding to the pair of knocked-out classes) to have high activation levels. The before-and-after comparisons of two pairs, YouTube-Amazon on the left and Amazon-DailyMotion on the right, are shown in Figure 8-13. The 1$^{st}$ row is the result before the knockout for reference, the 2$^{nd}$ row is the result of Y. Kim's method after the knockout and the 3$^{rd}$ row is the result of HAN after the knockout. The observations here are generalizable to other cases.

Before the knock-out, the network can be trained to drive both neurons to high activation levels. But while the network can still identify the sites after the removal, it becomes weaker at identifying both at once. For many samples, the network is only able to drive one of the two

neurons to high activation levels at a time. This effect is more visible with HAN, for no sample can drive the network output to the top-right corner after the knockout.

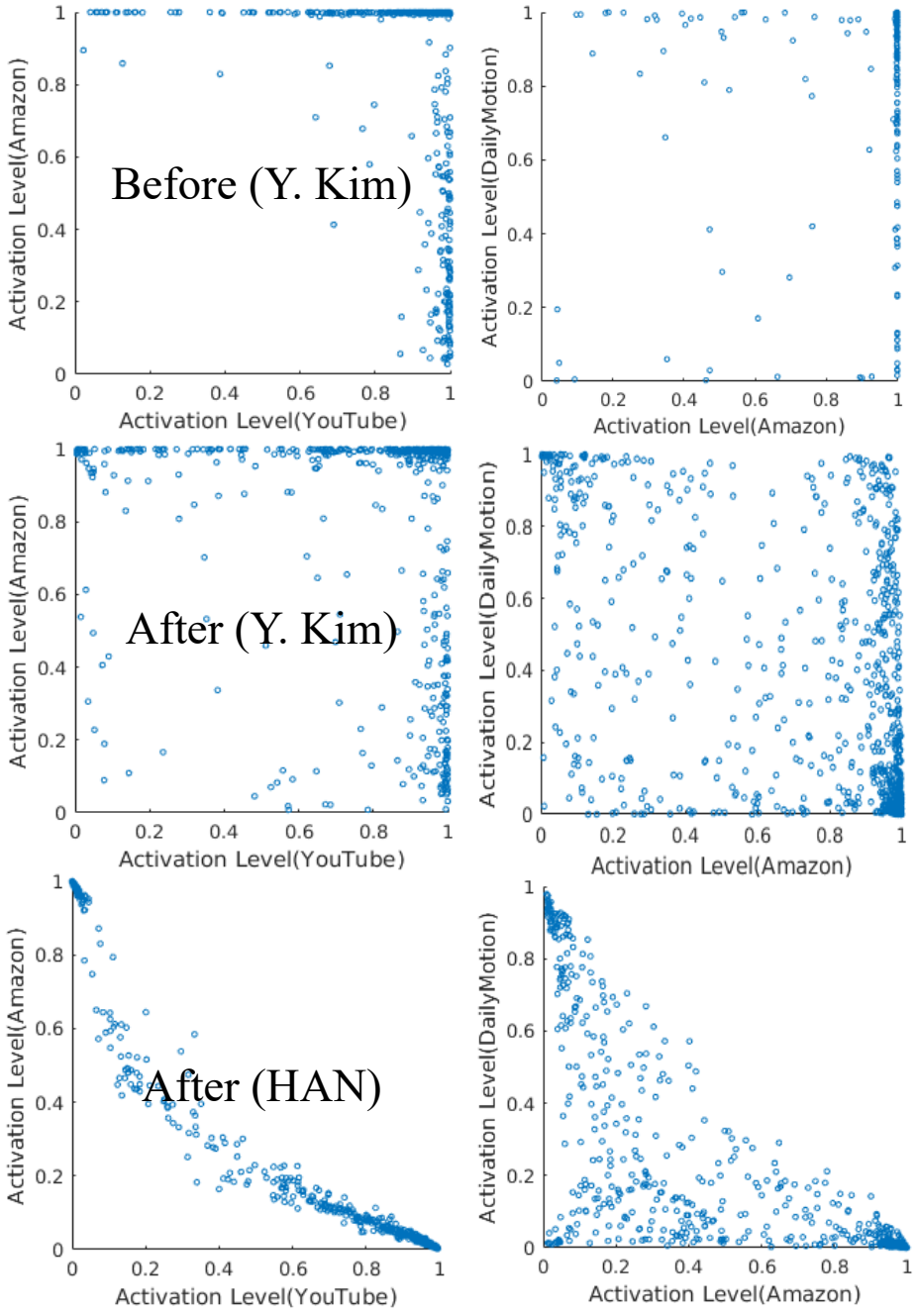



Figure 8-13: Selected Results from Knockout Tests

The observation might be a result of the max-pooling mechanism or the attention mechanism. Max-pooling or attention mechanism helps the network focus on the most significant input components (assumed to be the site generating the most traffic in that sample) while suppressing other components, resulting in the observed results. The performance might be improved if a different network architecture that can focus on both is used.

### 8.6.4 Testing at a Different Location

To test the performance of the network when the test traffic is not from the same location as where the training set is collected, we made changes to the experimental setup as depicted in Figure 8-14 to collect a new dataset. The VPN server is migrated to an Amazon Web Services (AWS) instance in the US East (Ohio) region. We chose a residential location in East Lansing, MI in addition to the lab at Michigan State University as client locations. The access point and probe router are installed at the lab for about one week to collect the first half of the dataset, then transported to the residential location to collect the second half.

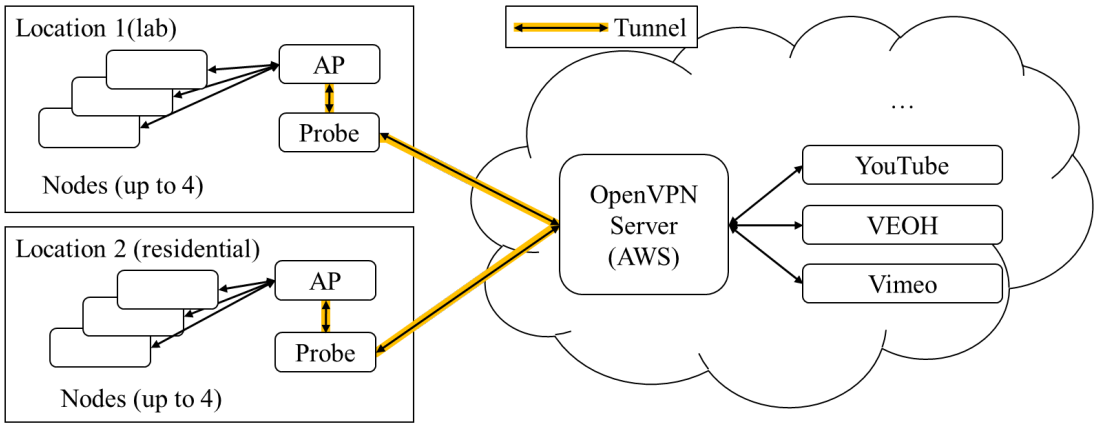



Figure 8-14: Experimental Setup for Testing a Different Location

The video streaming sites in this dataset are listed in Table 8-8. Due to the limitations AWS posts to the VPN server, Amazon Prime and Netflix are removed from the list of sites, while more

sites are added to make up for the loss. Twitch and Vimeo Livestream are added to evaluate the performance of the classifier on live streaming services, while all other services serve video-on-demand (VOD). The dataset serves as a chance to test on a larger number of video streaming providers to evaluate the generalizability of the proposed technique.

Table 8-8: Video Streaming Sites for Testing on a Different Location

| Video Source | Number of videos in the playlist |
|---|---|
| **VEOH** | 2 |
| **Vimeo** | 2 |
| **Fox News** | 1 (autoplays other videos) |
| **DailyMotion** | 1 (autoplays other videos) |
| **NatGeo (ng)** | 2 |
| **Twitch** | 1 |
| **Vimeo Livestream** | 1 |
| **YouTube** | 2 (autoplays other videos) |

Results: The performance figures in the rest of this subsection are derived from training the HAN and Y.Kim classifiers on the "lab" part of the dataset while testing on the "residential" part. We use Y. Kim's method and HAN for comparison since they are the two methods that are verified to perform well in single-location cases. The ROC curves of HAN and Y. Kim methods are shown in Figure 8-15. All 8 classes are trained at once, the same as the multi-label cases in single-location experiments.

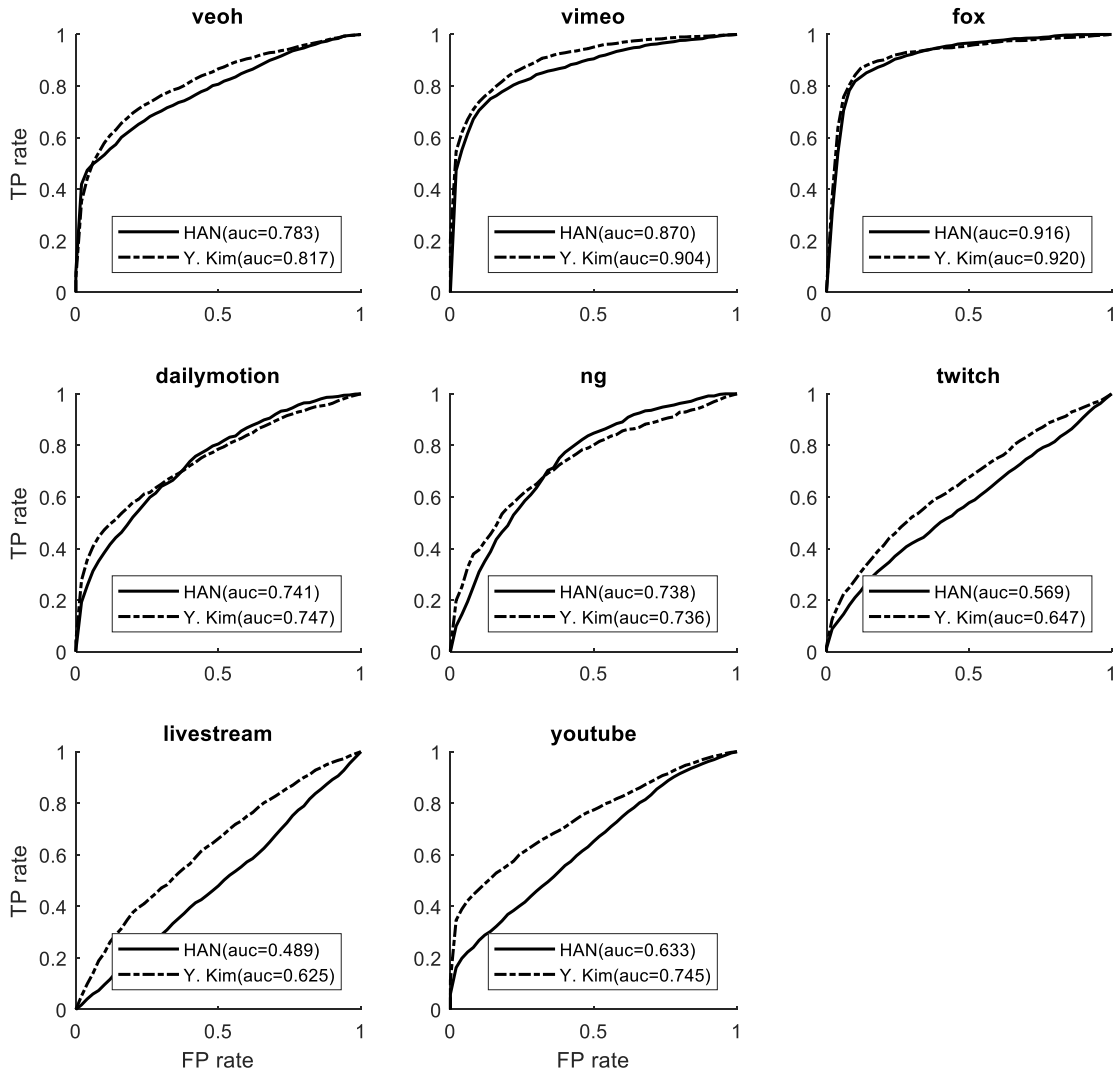



Figure 8-15: Classification Performance (Two Locations)

Compared to single-location results we can see that the change in location impacts performance. Some sites (YouTube and DailyMotion) are impacted more, having their ROC areas reduced to about 0.74 compared to the generally >0.9 areas in single-location experiments. On the other hand, the classifiers perform well on VEOH, Vimeo, and Fox. The tests on live streaming sites (Livestream and Twitch) show very poor performance compared to the VOD sites. Another trend that can be observed is that for VOD sites, about 30% to 40% of the positive samples can be identified with small false positive rates. The rest of the samples can become ambiguous. This

means that classifiers can still be used across locations by adjusting the thresholds, albeit with a performance penalty. For completeness, the ROC areas of two-location test on samples with more than 2 active clients are shown in Figure 8-16. The networks struggle to classify YouTube in this case, while other sites received mild hindrance.

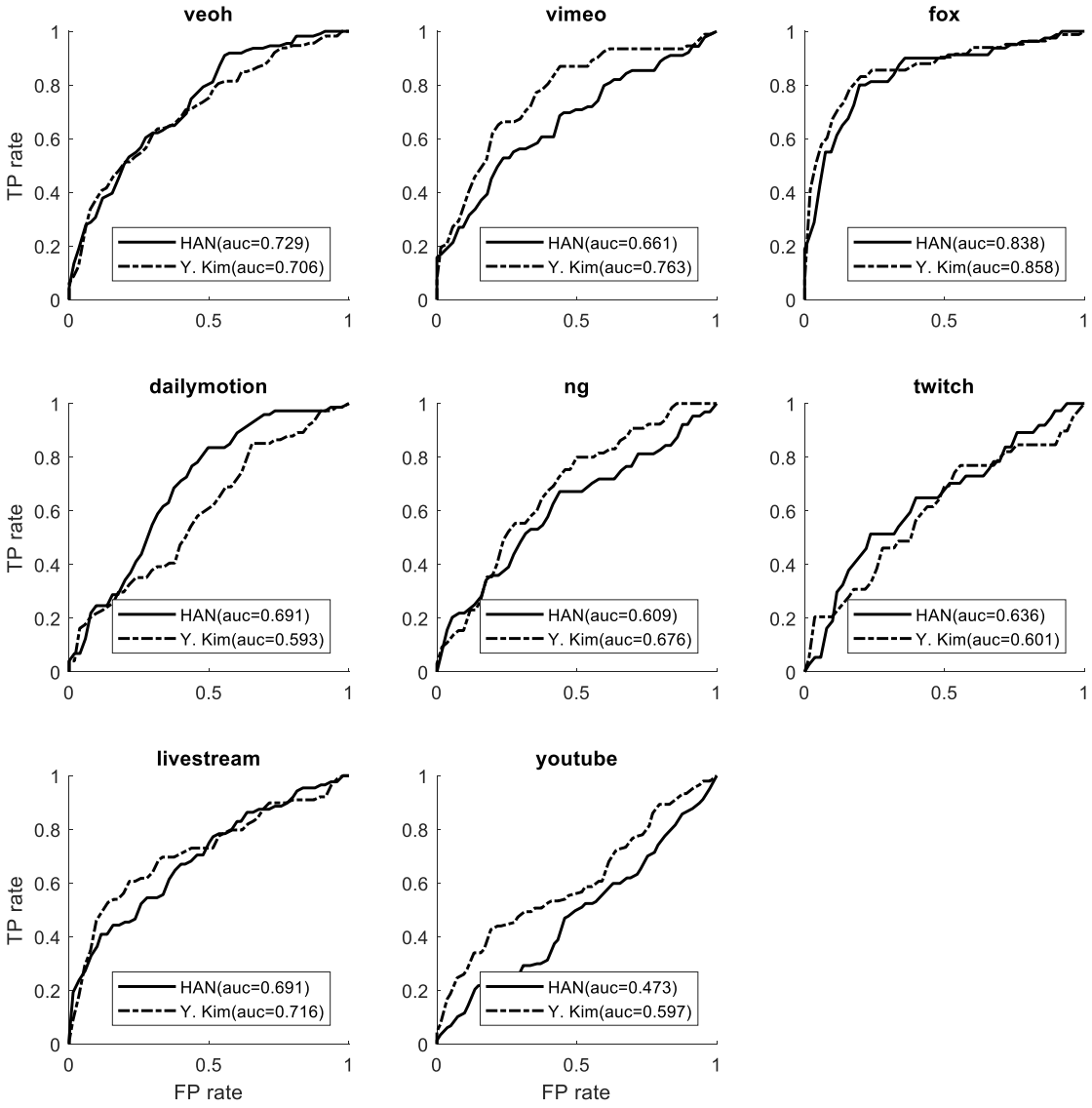



Figure 8-16: Classification Performance (Two locations & Client ≥ 3)

## 8.7 Summary

In this chapter, we propose a novel classification method for identifying the source of heterogeneous network traffic flow using deep-learning based classifiers. The method consists of a novel feature that is inspired by the researches in NLP. The feature is a concise text-like representation of the encrypted network traffic. Once calculated the feature can be processed by several NLP techniques that outperforms traditional methods. We combine the feature with neural networks that take advantage of NLP methods such as word embedding and attention to build several classifiers as our test subjects.

Using a dedicated experimental setup, we test the classifiers we built with the novel feature and 3 NLP methods, as well as a traffic classification method that uses hand-crafted traffic features. The results show that the new feature performs better on binary classification problems. We then test the performance of the methods on the multi-label classification problems. Again our method is shown to work well.

Extra evaluations to test the robustness of the feature are also conducted. First, we explore the possibility of classifying traffic containing a combination of labels that are not present in the training dataset ("knockout" problem). The method is shown to be able to identify the predominant component of the traffic. Second, we train the classifier against a dataset containing traffic from one location but evaluate against another dataset collected from another location that contains traffic from the same video streaming sites. The evaluation shows that while the performance is impacted by the discrepancy in training location and test location, the classifier nonetheless could classify about 30-40% of the VOD data with tolerable false positives. This is a breakthrough compared to the erratic performance figures when evaluating against 2 locations as presented in

Chapter 6. The proposed Deep-Learning based traffic analysis is significant in overcoming the obstacles toward practical TA applications.

Future work should include improvement of the feature to include more information while maintaining a concise format and analysis and improvement of its performance when classifying living streaming services.

# BIBLIOGRAPHY

# BIBLIOGRAPHY


[1] S. Incorporated, "Global Internet Phenomena Report, 1H 2016," Apr. 2016.

[2] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, 2009, p. 31.

[3] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website Fingerprinting in Onion Routing Based Anonymization Networks," in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, 2011, pp. 103–114.

[4] S. Khanvilkar and A. Khokhar, "Virtual Private Networks: An Overview with Performance Evaluation," *IEEE Commun. Mag.*, vol. 42, no. 10, pp. 146–154, Oct. 2004.

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-generation Onion Router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, 2004, p. 21.

[6] L. Dixon, T. Ristenpart, and T. Shrimpton, "Network Traffic Obfuscation and Automated Internet Censorship," *IEEE Secur. Priv.*, vol. 14, no. 6, pp. 43–53, Nov. 2016.

[7] T. Wang and I. Goldberg, "Improved Website Fingerprinting on Tor," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, 2013, pp. 201–212.

[8] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A Critical Evaluation of Website Fingerprinting Attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, 2014, pp. 263–274.

[9] N. Namdev, S. Agrawal, and S. Silkari, "Recent Advancement in Machine Learning Based Internet Traffic Classification," *Procedia Comput. Sci.*, vol. 60, pp. 784–791, 2015.

[10] R. Oppliger, "Internet security: firewalls and beyond," *Commun. ACM*, vol. 40, no. 5, pp. 92–102, May 1997.

[11] ISO/IEC, "Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model," Jun. 1994.

[12] R. Bendrath and M. Mueller, "The end of the net as we know it? Deep packet inspection and internet governance," *New Media Soc.*, vol. 13, no. 7, pp. 1142–1160, Apr. 2011.

[13] M. Rash, *Linux Firewalls*. 2003.

[14] R. Hofstede *et al.*, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[15] R. Stanton, “Securing VPNs: Comparing SSL and IPsec,” *Comput. Fraud Secur.*, vol. 2005, no. 9, pp. 17–19, Sep. 2005.

[16] T. Rowan, “VPN technology: IPSEC vs SSL,” *Netw. Secur.*, vol. 2007, no. 12, pp. 13–17, Dec. 2007.

[17] M. Feilner, *OpenVPN: Building and Integrating Virtual Private Networks*. Packt Publishing, 2006.

[18] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail,” in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, 2012, pp. 332–346.

[19] Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *Commun. Surv. & Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, Oct. 2008.

[20] R. Morla, P. Gonçalves, and J. Barbosa, “High-performance network traffic analysis for continuous batch intrusion detection,” *J. Supercomput.*, vol. 72, no. 11, pp. 4107–4128, 2016.

[21] Y. Liu, A.-R. Sadeghi, D. Ghosal, and B. Mukherjee, “Video Streaming Forensic – Content Identification with Traffic Snooping,” in *Information Security*, vol. 6531, M. Burmester, G. Tsudik, S. Magliveras, and I. Ilić, Eds. Springer Berlin Heidelberg, 2011, pp. 129–135.

[22] Y. Liu, C. Ou, Z. Li, C. Corbett, B. Mukherjee, and D. Ghosal, “Wavelet-Based Traffic Analysis for Identifying Video Streams over Broadband Networks,” in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, 2008, pp. 1–6.

[23] Y. Shi and S. Biswas, “Detecting tunneled video streams using traffic analysis,” in *2015 7th International Conference on Communication Systems and Networks, COMSNETS 2015 - Proceedings*, 2015.

[24] Y. Shi and S. Biswas, “Characterization of Traffic Analysis based video stream source identification,” in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, 2015, pp. 1–6.

[25] Y. Shi and S. Biswas, “Protocol-independent identification of encrypted video traffic sources using traffic analysis,” in *2016 IEEE International Conference on Communications, ICC 2016*, 2016.

[26] Y. Shi and S. Biswas, “Characterization of Traffic Analysis based video stream source identification,” in *International Symposium on Advanced Networks and Telecommunication Systems, ANTS*, 2016, vol. 2016–Febru.

[27] Y. Shi and S. Biswas, “Using traffic analysis for simultaneous detection of BitTorrent and streaming video traffic sources,” in *2017 9th International Conference on Communication Systems and Networks, COMSNETS 2017*, 2017, pp. 79–86.

[28] R. Vinayakumar, K. P. Soman, and P. Poornachandrany, “Secure shell (SSH) traffic analysis with flow based features using shallow and deep networks,” in *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, 2017, vol. 2017–Janua, pp. 2026–2032.

[29] J. Yan and J. Kaur, “Feature Selection for Website Fingerprinting,” *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 4, pp. 200–219, 2018.

[30] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, “Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning,” Sep. 2017.

[31] M. Cruz, R. Ocampo, I. Montes, and R. Atienza, “Fingerprinting BitTorrent Traffic in Encrypted Tunnels Using Recurrent Deep Learning,” in *Proceedings - 2017 5th International Symposium on Computing and Networking, CANDAR 2017*, 2018, vol. 2018–Janua, pp. 434–438.

[32] M. T. H. Parmar, “Adobe’s Real Time Messaging Protocol,” *Adobe Systems Incorporated*, 2012. .

[33] A. Zambelli, “IIS Smooth Streaming Technical Overview,” Mar. 2009.

[34] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update,” *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[35] C. Demichelis and P. Chimento, “IP Packet Delay Variation Metric for IP Performance Metric (IPPM),” 2002. .

[36] D. Chen, X. Fu, W. Ding, H. Li, N. Xi, and Y. Wang, “Shifted gamma distribution and long-range prediction of Round Trip Timedelay for Internet-based teleoperation,” in *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, 2009, pp. 1261–1266.

[37] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, “Network Tomography: Recent Developments,” *Stat. Sci.*, vol. 19, no. 3, pp. 499–517, 2004.

[38] T. M. Chen, “Network Traffic Modeling,” in *Handbook of Computer Networks*, vol. 3, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2012, pp. 326–339.

[39] “Practical Packet Analysis: using Wireshark to solve real-world network problems,” *Netw. Secur.*, vol. 2011, no. 8, p. 4, 2011.

[40] C. V Wright, L. Ballard, F. Monrose, and G. M. Masson, “Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob?,” in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, 2007.

[41] X. Gong, N. Kiyavash, N. Schear, and N. Borisov, “Website Detection Using Remote Traffic Analysis.” Sep-2011.

[42] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[43] B. Vandalore, W. Feng, R. Jain, and S. Fahmy, “A Survey of Application Layer Techniques for Adaptive Streaming of Multimedia,” *Real-Time Imaging*, vol. 7, no. 3, pp. 221–235, Jun. 2001.

[44] M. Hall, “Correlation-based Feature Selection for Machine Learning.” 1998.

[45] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, “A survey on deep learning for big data,” *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018.

[46] A. Krizhevsky and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Neural Inf. Process. Syst.*, 2012.

[47] A. L. Caterini and D. E. Chang, “Recurrent neural networks,” in *SpringerBriefs in Computer Science*, 2018.

[48] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[49] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” Jun. 2014.

[50] D. W. Otter, J. R. Medina, and J. K. Kalita, “A Survey of the Usages of Deep Learning in Natural Language Processing,” Jul. 2018.

[51] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical Attention Networks for Document Classification,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.

[52] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *2016 IEEE Work. Spok. Lang. Technol. SLT 2016 - Proc.*, pp. 414–419, Jan. 2013.

[53] H. Caswell, C. de Vries, N. Hartemink, G. Roth, and S. F. van Daalen, “Age × stage-classified demographic analysis: a comprehensive approach,” *Ecol. Monogr.*, vol. 88, no. 4, pp. 560–584, Nov. 2018.

[54] J. Pennington, R. Socher, and C. Manning, “Glove: Global Vectors for Word Representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[55] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” *Proc. EMNLP*, Aug. 2014.

[56] M. J. Berger, “Large Scale Multi-label Text Classification with Semantic Word Vectors,” *Tech. Rep.*, pp. 1–8, 2014.