THESIS

MULTIMEDIA TRANSMISSION RULES AND ENCRYPTED AUDIO AND VIDEO TRAFFIC IDENTIFICATION ALGORITHM IMPLEMENTED IN P4

Submitted by

Jiping Lu

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2021

Master's Committee:

Advisor: Craig Partridge

Joseph Gersch
Stephen Hayne

ABSTRACT

MULTIMEDIA TRANSMISSION RULES AND ENCRYPTED AUDIO AND VIDEO TRAFFIC IDENTIFICATION ALGORITHM IMPLEMENTED IN P4

With Internet traffic exponentially growing, it is critical for operators to identify voice call and video conference traffic and ensure their quality. Yet, the widespread deployment of encryption protocols makes it challenging to classify encrypted traffic. This research achieves a significant discovery of audio and video traffic transmission rules. Based on the rules, this paper proposes a general audio and video traffic identification algorithm. In order to evaluate this algorithm's performance, we designed and implemented an audio and video traffic identification algorithm in P4 (Programming Protocol-Independent Packet Processors). This promising research reveals that this algorithm achieves 98.98% accuracy on voice call identification. For video conferences, this algorithm achieves 96.24% accuracy on audio data identification and 88.75% accuracy on video data identification. Compared to current pervasive machine learning-based traffic classification approaches, this innovative algorithm bypasses complicated machine learning processes by directly applying audio and video traffic transmission rules on network functions, consuming less computation and memory resources.

ACKNOWLEDGEMENTS

I acknowledge Raytheon BBN Technologies Corp. for the financial support.

Research performed was funded by the U.S. Army Research Office and Defense Advance Research Project Agency to Raytheon BBN Technologies Corp. Contract W911NF19C00421 and subcontract agreement with Colorado State University.

The content of this publication does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

I appreciate everyone who has helped me on my path toward this dissertation.

I would like to express my sincere gratitude to my advisor, Prof. Craig Partridge, for his fantastic guidance. He has always been encouraging, helpful, respectful, and thoughtful to me. He always patiently gave me insightful, concise, and clear step-by-step directions. Each step is neither too simple nor too difficult but fun and possible to be fulfilled. Finally, he led me to a high leveled research achievement. I am so grateful for him giving me the most pleasant and valuable experience. The strategies, attitudes, and time management skills I learned from him will benefit my future work and life.

I also would like to show my big appreciation to my other committee members, Dr. Joseph Gersch and Prof. Hayne Stephen. They are always ready to help and have given me valuable comments and support.

I am grateful to Dr. Daniel Ellard, who directed and helped me with experimental data collections, which are vital for this research.

I appreciate all CSU professors and staff, who are always very nice, and love to support us to succeed.

I thank A.C. Heaps for the editorial review.

I am grateful to my family for their love and support.

DEDICATION

This dissertation is dedicated to my husband Yanhui who loves and supports me to finish this dissertation, to my lovely son Sam and daughter Sandra who motivate me to work hard, to my great parents who always love and encourage me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Recently Internet traffic has been exponentially growing. Some applications like peer-to-peer file sharing consume a large amount of network bandwidth. Thus, it remains essential that operators identify voice call and video conference traffic and assign them a high priority. However, with the widespread deployment of encryption technologies, identifying encrypted traffic becomes a great challenge. Some encryption protocols encrypt whole IP data, this makes it even harder to identify end-to-end encrypted traffic. For example, IPsec (IP Security) tunnel mode [1] encrypts whole IP data including the IP header, then attaches a new IP header before the encrypted IP data. As the source and destination IP addresses in the new IP header differ from the actual IP addresses inside the IPsec tunnels, and the whole IP packet contents are not visible, it increases the difficulty for network operators to identify the encrypted traffic. Traffic classification approaches generally include port-based, payload-based, and flow statistical-based methods. Port-based and payload-based approaches require inspection of packet port numbers or payload contents, so they do not apply to encrypted traffic. Flow statistical approaches can classify encrypted traffic, yet they usually require pre-labeled datasets and complicated machine learning steps.

In this research, we find voice call and video conference flow transmission rules, which reveal that an audio packet is sent out every 20ms, and the audio packet length range is 100 bytes to 200 bytes. For video conferences, audio and video data are separately transmitted. The data of a video frame is sent out about every 33ms. A frame data consists of a few or several large size packets which are sent out continuously. The experimental results indicate that 99.8% of video packet lengths are larger than 400 bytes. Based on this discovery, we propose a general algorithm identifying voice and video traffic. We also designed and implemented a voice and video traffic identification algorithm in P4 [2]. Our algorithm applies to both encrypted and unencrypted audio and video traffic. Compared to the prevalent machine learning-based traffic classification approaches, this novel algorithm does not require pre-labeled datasets and training phases. Moreover, it requires

few memory and CPU resources. The experimental results reveal that this algorithm achieves high accuracy on audio traffic identification.

We use true positive rate (TP), false positive rate (FP), and true negative rate (TN) to measure traffic identification accuracy. TP refers to the percentage of audio or video traffic which are correctly identified as audio or video traffic. FP refers to the percentage of interfering random packets which are falsely recognized as audio or video traffic. TN refers to the percentage of random interfering packets which are not identified as audio or video packets. The random packets are generated randomly to interfere with audio or video flows. Accuracy is defined as the sum of true positive packet number and true false packet number divided by the total packet number. For voice call traffic mixed with random packets, our algorithm achieves 99% TP, 1% FP, and 99% accuracy. For video conference traffic mixed with random packets, our algorithm achieves 95.6% TP, 2.5% FP, and 96.2% accuracy in audio traffic identification, 93.5% TP, 28.11% FP, and 88.75% accuracy in video traffic identification. Usually, the quality of audio traffic is more important than video traffic for customers because the decrease of audio quality is easier to be perceived than the decrease of video quality. Thus, this new and novel algorithm can improve customer satisfaction since it can identify the prevalence of audio traffic and assign them a high priority.

This paper is presented in the following format. Section 2 discusses related traffic classification and identification approaches. Section 3 describes our discovery of audio and video data transmission rules. In section 4, we propose a general audio and video traffic identification algorithm. The design and implementation of an audio and video traffic identification algorithm in P4 are discussed in section 5. Section 6 presents the experimental results in P4 and section 7 concludes the paper and suggests future work.

Chapter 2

Related Work

Network traffic classification plays a vital role in network management and resource allocation. Network traffic identification and classification methods evolved from port-based and payload-based to flow characteristics based using machine learning technologies [3, 4]. Port-based traffic classification approaches rely upon well-known TCP or UDP port numbers registered in Internet Assigned Numbers Authority (IANA) [5]. Due to the fact that many applications do not employ well-known port numbers to avoid inspection or access control restrictions, or use dynamically allocated port numbers, port-based traffic classification methods have become less reliable. Consequently, payload-based traffic classification approaches emerged, which are also called deep packet inspection (DPI). Payload-based approaches first define application signatures, then inspect and compare packet contents with predefined application signatures to classify traffic. For instance, Sen et al. [6] studied five types of P2P protocols and discovered that each P2P protocol has distinctive fixed values at specific positions, employing these signatures to identify P2P traffic. Moore and Papagiannak [7] devised nine methods that were used separately or in a combination to classify flows. The first method examined port numbers, the second method inspected packet header, and other methods examined payload contents.

Yet, with the popularity of encryption technology adoption, port-based and payload-based approaches prove ineffective on encrypted traffic. As a result, a great number of flow-based traffic classification approaches based on machine learning technologies are proposed. Nguyen et al. [3] reviewed 18 significant machine learning-based IP traffic classification works from 2004 to 2007. The authors described that machine learning methods rely on traffic statistical properties such as flow duration distribution, flow idle time, packet inter-arrival time, packet lengths, and so on. Machine learning technologies are divided into two categories: supervised machine learning and unsupervised machine learning. Supervised machine learning technology uses pre-labeled datasets to find a function that processes input data to produce outputs that are most close to pre-labeled out-

puts. Unsupervised machine learning technology naturally clusters data with similar features into groups without the requirement of pre-labeled datasets. Supervised machine learning technologies can be used to identify specific application traffic.

In addition, some hybrid methods are introduced. For instance, Sun et al. [8] proposed a hybrid approach which combines signature-based methods and machine learning-based methods to identify applications encrypted with Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols. The authors first used SSL and TLS signatures to identify traffic encrypted with SSL or TLS protocols, then used machined learning technologies to further classify the traffic into TOR or HTTP applications. This method works for identifying applications encrypted with TLS or SSL protocols, but it does not apply to end-to-end encryption traffic.

In 2019, Pacheco et al. [4] presented a systematic overview of steps to achieve traffic classification based on machine learning technologies. The authors described that general supervised machine learning technology steps are comprised of data collection, feature extraction, feature reduction, algorithm selection and model construction, and validation of classification models. The authors explained each step. Data collection is the process to gather information and label datasets. Feature extraction step measures and computes different attributes' contribution to the model. Feature reduction is an optional step to find the features that have high influences on classification decisions. Algorithm selection and model construction step chooses a machine learning algorithm and finds the model parameters which minimize the differences between model outputs and pre-labeled outputs. This step is also referred to as the training phase. Validation of classification models utilize the trained model to test labeled data, also referred to as the testing phase. Indeed, machine learning technologies are complex.

Most recently, deep learning technologies have been attracting more attention for traffic classification. Wang et al. [9] reviewed deep learning applications for mobile encrypted traffic classification and presented a deep learning-based mobile encrypted traffic classification framework. Deep learning technologies are divided into supervised, unsupervised, and semi-supervised methods.

The authors mentioned that for supervised deep learning methods, dataset labeling is a difficult and time-consuming task.

Chapter 3

Multimedia Transmission Rules

According to RFC3550 [10] that audio conference application participants send audio data in small chunks of 20ms duration, we estimate that an audio packet is sent out every 20 milliseconds. As [10] noted that audio and video media are transmitted as separate RTP (Real-time Transport Protocol) sessions if both of them are used in a conference, we know that audio and video data are separately transmitted. As described in H.264 [11] and [12], in order to compress video frame data, video frames are transmitted in I, B, and P frame types. An I-frame is a complete picture, a P-frame only holds changes to the previous frame and a B-frame only stores differences to the previous and following frames for further data compression. Some video applications send 30 frames per second (FPS) [13]. From these specifications, we presume that video frame inter-transmission time is about 33 milliseconds, and packets carrying the data of a video frame are continuously transmitted.

In order to verify our estimation about audio and video traffic patterns, we performed two Skype voice call tests test_1 and test_2, and two video conference tests test_3 and test_4, and captured packets at both terminals to measure traffic properties. Terminal_1 was in Colorado and terminal_2 was in Massachusetts (both are located in the United States). For all the tests, after filtering captured packets by conditions such as source and destination IP addresses and protocols, we utilized R statistical program [14] to compute traffic properties in terms of packet lengths and packet interval time statistics. First, we collected and analyzed voice call packet lengths in order to separate the audio traffic from a video conference by packet lengths. Next, we filtered captured voice call packets in test_1 and test_2 by the conditions that the source IP address is terminal_1, destination IP address is terminal_2, and protocol is UDP. Then, we used R program to produce graphical and numerical audio packet statistical representations.

3.1 Audio Data Transmission Rules

Figure 3.1 displays the audio packet length distribution in test_1. Figure 3.2 displays the audio packet length distribution in test_2. The horizontal axis denotes packet length in bytes, and the vertical axis denotes packet number distribution. These two figures display that the most majority of audio packet lengths are between 100 and 200 bytes. Numerical statistical results computed by R explain that on average 98.32% of audio packet lengths are between 100 and 200 bytes.

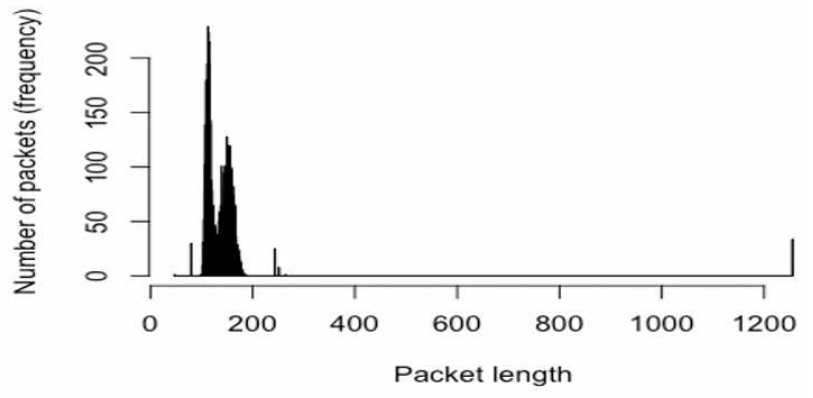

Figure 3.1: Skype audio packet length distribution in test_1.

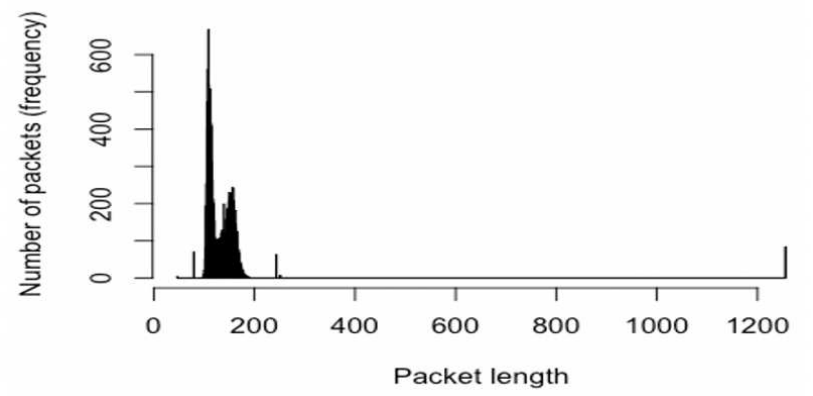

Figure 3.2: Skype audio packet length distribution in test_2.

In order to compute the statistics of Skype audio packet inter-transmission time, we filtered out Skype audio packets by the conditions that source IP address equals to IP address of ter-

minal_1, destination IP address equals to IP address of terminal_2, protocol equals to UDP and packet lengths are between 100 and 200 bytes. Then computed packet inter-transmission time by using the current packet time minuses the previous packet time. Next, we programmed in R to compute packet inter-transmission time statistics. Figure 3.3 represent the Skype audio packet inter-transmission time frequency in test_1. Figure 3.4 represent the Skype audio packet inter-transmission time frequency in test_2. The horizontal axis represents packet interval time in microseconds, and the vertical axis represents packet number. These two figures show that most audio packet inter-transmission time is around 20ms. The numerical statistical results computed by R indicate that an average of 97.68% Skype audio packet interval time is between 10ms and 26.2ms, an average of 76.14% Skype audio packet interval time is between 19.7ms and 21.5ms, and an average of 86.11% Skype audio packet interval time is between 19.7ms and 26.2ms.

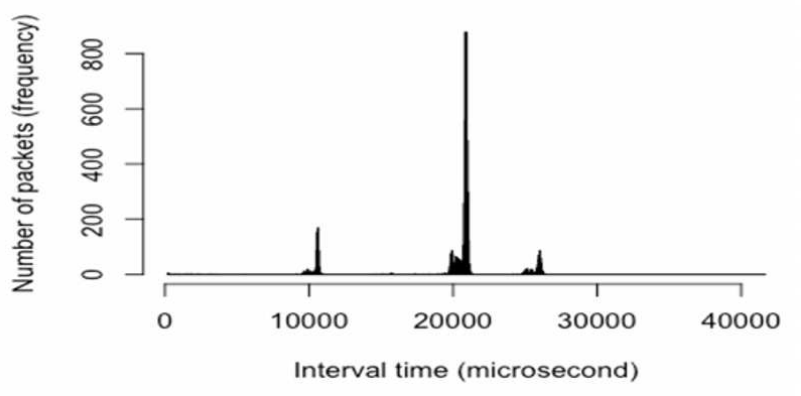

Figure 3.3: Skype audio packet interval time distribution in test_1.

3.2 Video Data Transmission Rules

Next, we performed two Skype video conference tests test_3 and test_4 to examine video traffic patterns. Based on the above statistical results indicating audio packet lengths are less than 200 bytes, we filtered out packets larger than 200 bytes to exclude audio packets in test_3 and test_4. Then we further filtered out packets from terminal_1 destinated to terminal_2 and the protocol is UDP. We discovered that the video data are transmitted in groups of packets. Each group includes

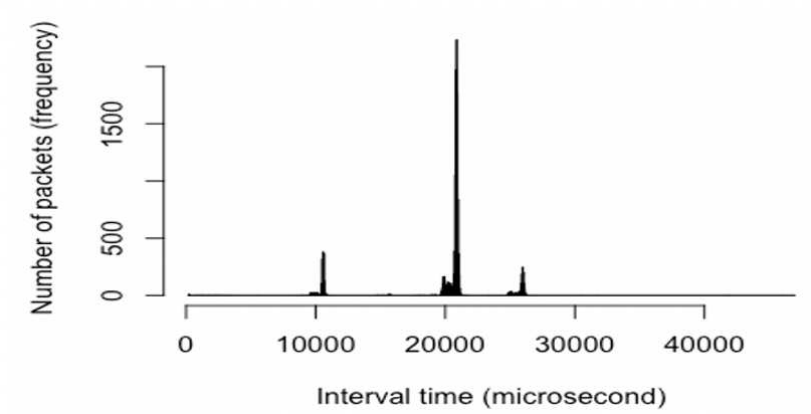

Figure 3.4: Skype audio packet interval time distribution in test_2.

several or a few packets which are sent out continuously. Use the first packet of the current group packets minus the first packet of the previous group packets to get the interval time between frames. Finally, we programmed in R to compute frame interval time statistics. Figure 3.5 demonstrates the video frame interval time frequency distribution in test_3. Figure 3.6demonstrates the frame interval time frequency distribution in test_4. The horizontal axis represents interval time between frames in microseconds, and the vertical axis represents packet number. These two figures indicate that a large part of the frame interval time is between 20ms and 48ms.

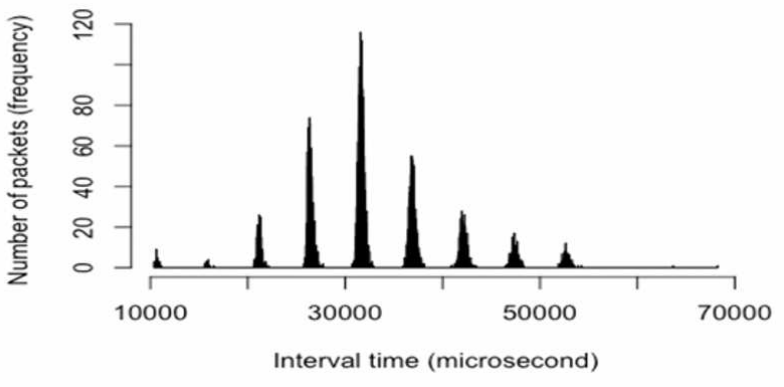

Figure 3.5: Skype video frame interval time frequency distribution in test_3.

Table 3.1 demonstrates the Skype video frame interval time ratio and mean computed by R in test_3 and test_4. The table shows that the average 95.84% frame interval time is between 20ms

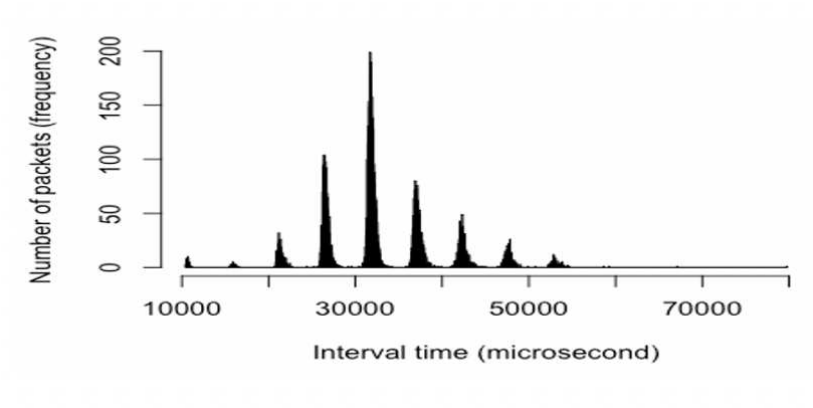

Figure 3.6: Skype video frame interval time frequency distribution in test_4.

and 48ms, and the mean frame interval time is 33.05ms. The results prove our estimation that video frame interval time is around 33ms.

Table 3.1: Skype video data interval time ratio and mean.

Test name	20ms < Interval time < 48ms / total	Mean interval time (ms)
test_3	95.51%	33.03
test_4	96.16%	33.07
average	95.84%	33.05

We also analyzed interval time between packets belonging to one frame. We used the current packet arrival time minus the previous packet time belonging to a frame, then used R to compute interval time statistics. Table 3.2 shows the interval time ratio of Skype video packets belonging to a frame in test_3 and test_4 computed by R. The table displays that an average of 84.99% packet interval time within a frame are less than 5 microseconds, and 99.04% packet interval time within a frame are less than 100 microseconds. The results prove our estimation that packets carrying one frame data are continuously sent out.

We also computed the statistics of duration time transmitting a video frame. Table 3.3 represents the statistics of a video frame duration time computed by R. The statistical results indicate that on average 99.15% of video frames finish transmitting a frame data within 0.2ms.

Table 3.2: Skype video packets belonging to a frame interval time ratio.

Test name	Interval time < 5 microsecond / total	Interval time < 100 microsecond / total
test_3	84.76%	99.04%
test_4	85.22%	99.04%
average	84.99%	99.04%

Table 3.3: Skype video a frame duration time ratio.

Test name	A video frame duration time less than 200 microseconds ratio
test_3	99.16%
test_4	99.13%
average	99.15%

Next, we computed Skype video packet length statistics. Figure 3.7 demonstrates the Skype video packet length distribution in test_3. Figure 3.8 demonstrates the video packet length distribution in test_4. The horizontal axis denotes video packet length in bytes, and the vertical axis denotes packet number. Numerical statistical results computed by R indicate that on average 99.82% of video packet lengths are larger than 400 bytes, and 99.59% of video packet lengths are larger than 600 bytes. The experimental results prove our estimation that video frame data are transmitted by large size packets.

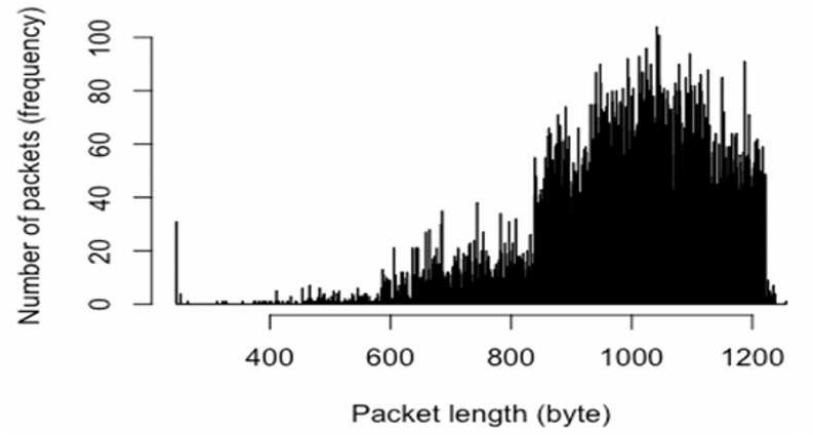

Figure 3.7: Skype video packet length distribution in test_3.

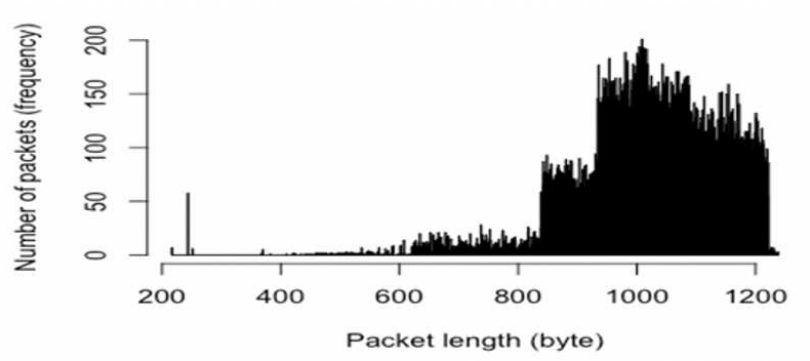

Figure 3.8: Skype video packet length distribution in test_4.

3.3 Summary

In summary, according to audio and video standard specifications, we know that audio and video data in a video conference are separately transmitted, we also presume that an audio packet is sent out every 20ms, video data is sent out by groups of large size packets, and group interval time is around 33ms. Our voice call and video conference tests confirm our estimations. The experimental results also reveal that audio packet sizes are between 100 and 200 bytes; the mean frame interval time is 33ms; 95.8% frame interval time is distributed between 22ms and 48ms; 99% packet interval time within a frame is less than 0.1ms; 99.2% video frames finish transmitting a frame data within 0.2ms; 99.8% video packet lengths are larger than 400 bytes.

Chapter 4

Traffic Identification Algorithm

Based on the multimedia traffic pattern discovered in section 3, we propose a general multimedia traffic identification algorithm described in figures 4.1, 4.2, and 4.3. Figure 4.1 explains the initial multimedia traffic process, figure 4.2 presents the audio packet process, and figure 4.3 describes the video packet process. R(0), R(1), and R(2) are global variables or registers in P4. R(0) stores the last audio packet arrival time, R(1) stores the arrival time of the first packet of a video frame, and R(2) saves the previous video packet arrival time. Constant AUDIO_PRIORITY denotes audio packet priority, VIDEO_PRIORITY denotes video packet priority. In figure 4.1, when the network function implemented our algorithm receives a packet, it examines the packet length. If the packet length is between 100 and 200 bytes, it is forwarded to audio packet process module. Else if the packet length is larger than 400 bytes, it is forwarded to video packet process module.

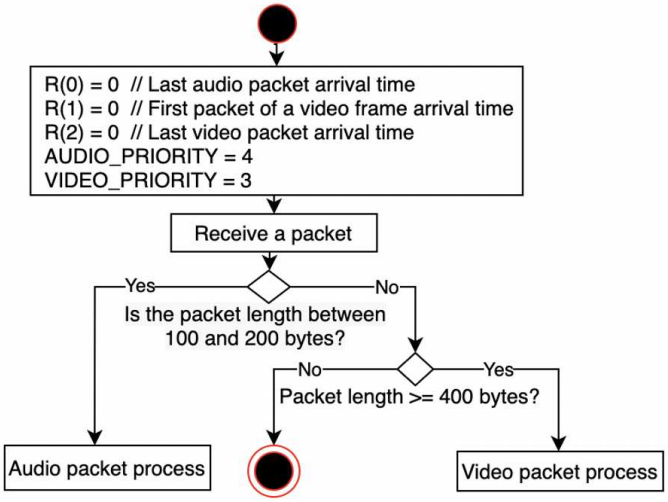

Figure 4.1: Multimedia packet initial process.

Figure 4.2 describes the audio packet process. First, the process examines the R(0) value. If R(0) equals zero, this packet may be the first audio packet, so R(0) is updated with this packet

arrival time, then the packet goes to an end in this algorithm. Otherwise, interval time is calculated by using the current packet arrival time minus the last audio packet arrival time R(0). According to the statistical results in section 3 that 97.68% of audio packet interval time is between 19.7ms and 26.2ms, if the packet inter-arrival time is in this range, the packet is treated as an audio packet by updating R(0) with this packet arrival time and setting the packet Diffserv value to AUDIO_PRIORITY. Else if the packet inter-arrival time is larger than 26.2ms, R(0) is updated with the current packet arrival time, and the packet process goes to an end in this algorithm function.

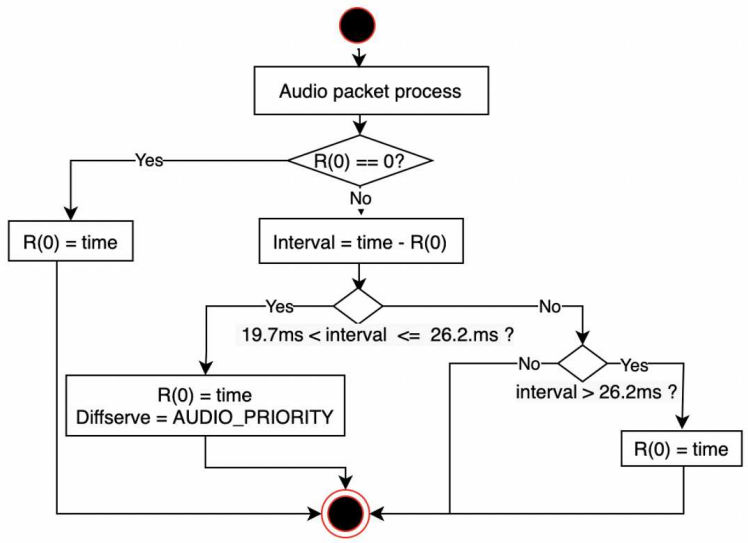

Figure 4.2: Audio packet process.

Figure 4.3 describes video packet process. R(1) records the first packet of a frame arrival time and R(2) records the last video packet arrival time. If R(1) is zero, R(1) and R(2) are updated with the packet arrival time. Otherwise, calculate inter-arrival time between frames using the current packet time minus R(1), also calculate packet inter-arrival time within a frame using the current packet arrival time minus R(2). Section 3 reports that 99% of video packet interval time within a frame is less than 0.1ms and more than 99% of video pictures finish transmitting a frame data within 0.2ms. Hence, when packet interval time within a frame is less than 0.1ms and frame interval time is less than 0.2ms, this packet is treated as a video packet by updating R(2) with the current packet arrival time, and setting this packet Diffserv value to VIDEO_PRIORITY. Else

if frame interval time is larger than 20ms and smaller than 48ms, according to the traffic pattern discovered in section 3, we infer this packet is the first packet of a new frame, so R(1) and R(2) are updated with this packet arrival time, and the packet Diffserv value is set to VIDEO_PRIORITY. Otherwise, if frame interval time is equal to or larger than 48ms, R(1) and R(2) are updated with the current packet arrival time, then the video packet process goes to an end in this function.

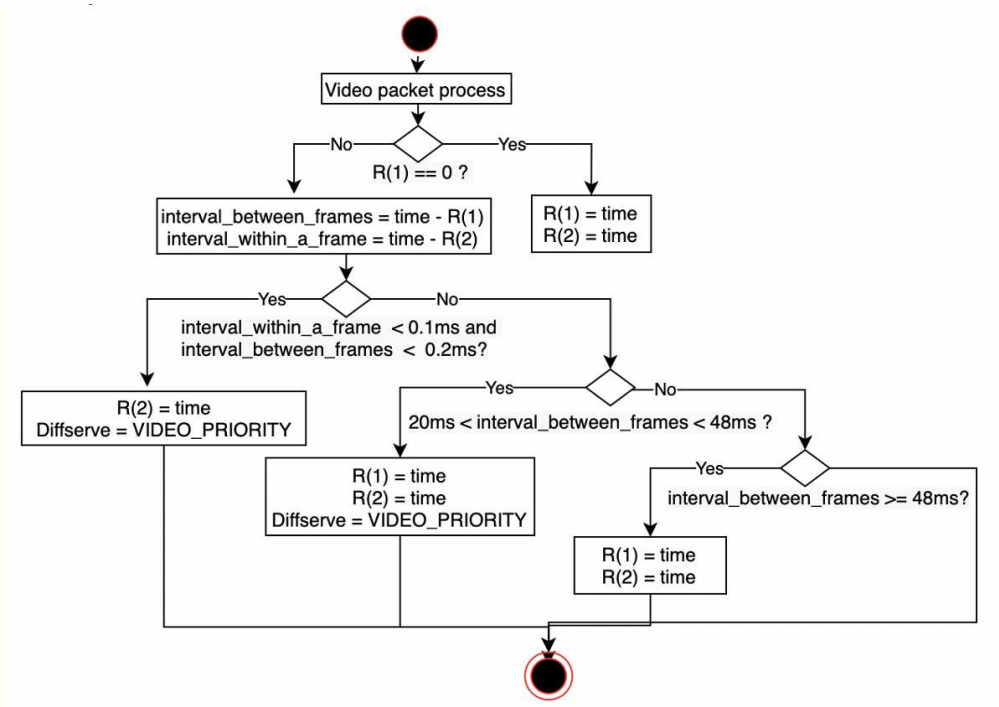

Figure 4.3: Video packet process.

Chapter 5

Algorithm Implementation in P4

In order to evaluate our traffic identification algorithm performance, we designed and implemented a multimedia traffic identification algorithm in P4. We chose P4 because it is a protocol independent data plane language [2, 15]. It is easier to instruct network devices like switches or routers how to process packets in P4.

5.1 Network Topology

First, we configured the P4 environment. Then we downloaded and configured the P4 virtual machine image from P4 tutorials [16, 17]. Based on the basic example in the tutorials, we created a new network topology by modifying the file topology.json, then established IP packet routing rules for each router. We created eight files s1-runtime.json to s8-runtime.json specifying IP packet forward rules for routers s1 to s8 respectively. Figure 5.1 shows the network topology. It consists of eight routers s1 to s8, and eight hosts h1, h4, h5, h6, h61, h7, h71, and h8.

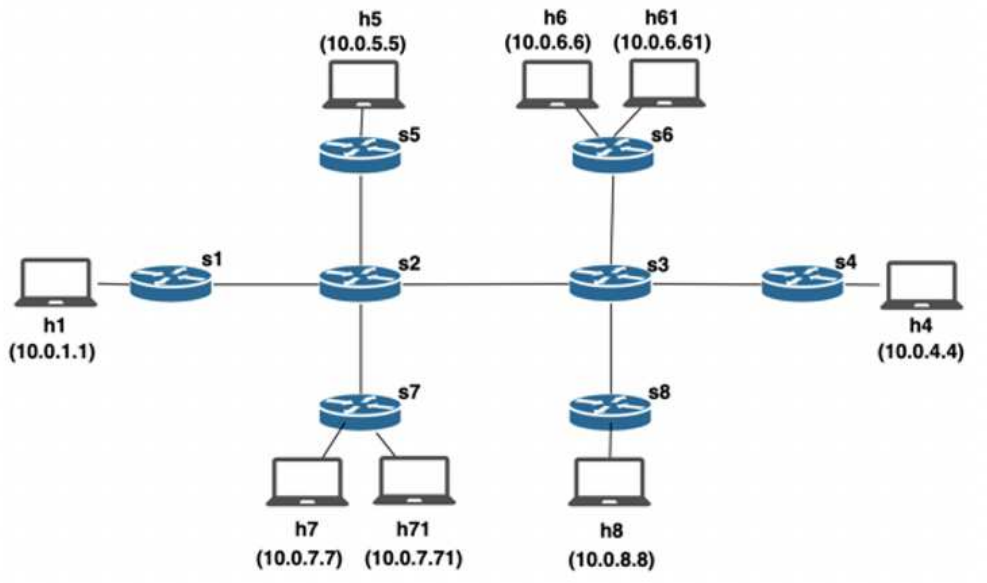

Figure 5.1: P4 network topology.

5.2 Challenging Issues and Solutions

One issue we encountered is that P4 performance is not fast. P4 software switch runs on Behavior Model version 2 (BMv2), which is a simulation environment developed as a tool for P4 developing and testing, not a production-grade software, so its throughout and latency is slow [18]. In our tests, when h1 uses send.py, which uses Scapy [19] to send packets, to continuously send packets to h4, the test results revealed that packet inter-arrival times are all greater than 10ms, and about 97% of packet interval times are between 10ms and 40ms, 99% of packet interval times are between 10ms and 60ms. However, video packet statistics in section 3 indicated that about 85% of packet interval times within a frame are less than 5 microseconds. In addition, about 99% of packet interval times within a frame are less than 100 microseconds. Therefore, P4 performance is substandard in processing actual multimedia traffic. In order to evaluate our multimedia traffic identification algorithm in P4, our solution is to amplify packet inter-transmission time. Accordingly, audio packet inter-transmission time is amplified from 20ms to 200ms, and video packet inter-transmission time between frames is amplified from 33ms to 300ms. A host sends small size packets every 200ms simulating audio traffic and sends a group of large size packets every 300ms simulating video traffic. A group of packets carrying the data of a frame are continuously sent out.

In figure 5.1, h1 sends simulated audio or video packets to h4 through s1, s2, s3, and s4. We specify that an audio packet IP length is 45 bytes, and a video packet length is 57 bytes. We also specify that a group of packets carrying the data of a video frame consists of three packets that are sent out continuously. Some other hosts may send out random packets at random interval time to interfere with the audio or video traffic from h1 to h4. The statistical results in section 3 demonstrate 98.32% of audio packet lengths are between 100 and 200 bytes, and packet lengths vary from 1 byte to 1514 bytes. In our experiments, we assume packet lengths are evenly distributed. Then the portion of packets their lengths are between 100 and 200 bytes is about 1/15. This means among 15 random packets, one packet size if same as an audio packet, while other 14 packets their sizes differ from an audio packet size. Therefore, our random audio packet size strategy is that choosing a random integer number from 1 to 15, when the random number equals to 2, random au-

dio interfering senders send a packet the same size as an audio packet but with different contents. Otherwise, random interfering senders send a different size packet. As the statistical results in section 3 show that 99.59% of video packet lengths are between 600 and 1230 bytes, we estimate that the portion of packets their lengths are between 600 and 1230 bytes is about 6/15. Hence, our random interfering video packet length decision strategy is that choosing a random integer number from 1 to 15, when the random number is between 6 and 12, random video interfering senders send packets the same size as a video packet but with different contents. Otherwise, send different size random packets to interfere with video traffic. Our traffic identification algorithm is deployed on router s3.

Based on the basic example in P4 tutorials, our traffic identification algorithm is implemented by adding functions in file basic.p4. File basic.p4 declares ethernet and IPv4 header, a parser type block MyParser() and five control type blocks: MyVerifyChecksum(), MyIngress(), MyEgress(), MyComputeChecksum() and MyDeparser(). All of our algorithm functions are added in the control block MyIngress(). In P4, a packet received at a router only stores information about this packet. However, time information of previous packets is required. Our solution is to use a stateful object register to store previous packet information. At the start position of control block MyIngress(), declare a stateful object register<bit<48»(3) packet_register, which includes three elements. In the rest of this paper, R represents packet_register. R(0), R(1), and R(2) store previous packet time information. Our multimedia traffic identification algorithm is put in function apply in control block MyIngress(). Another issue we encountered is that all routers in P4 systems apply rules defined in file basic.p4 to process packets by default. However, we endeavored to implement our traffic identification algorithm on one router for performance optimization. Our final strategy is to use a router's MAC (Media Access Control) address. Considering that all packets received at s3 whose destination addresses are router s3 MAC address, our strategy is that when a received packet destination MAC address hdr.ethernet.dstAddr equals to s3 MAC address 0x080000000300, apply our algorithm to process this packet. This method ensures that other routers ignore our algorithm except s3.

5.3 Algorithm Initial Process

Figure 5.2depicts our algorithm's initial process. Register R(0) stores the last audio packet arrival time, R(1) stores the first packet arrival time of a frame, and R(2) stores the last received video packet arrival time. Constant AUDIO_PRIORITY defines audio packet priority, and constant VIDEO_PRIORITY defines video packet priority. When a router receives a packet, it examines the packet destination MAC address. If hdr.ethernet.dstAddr equals to s3 MAC address 0x080000000300, apply our algorithm to process the packet. Next, classify packets based on packet lengths. If the packet length equals to audio packet length of 45 bytes, it is processed by the audio process module. Else if the packet length equals to video packet length 57 bytes, it enters the video process module. Otherwise, the packet bypasses our identification algorithm.

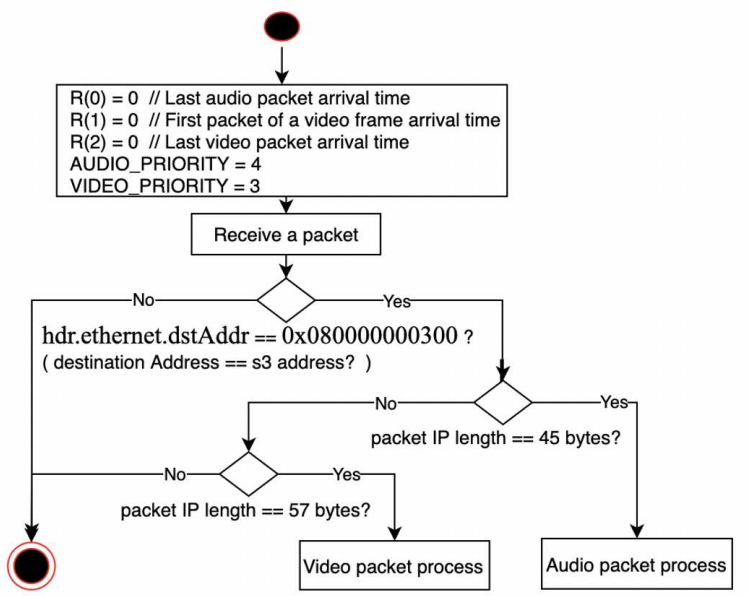

Figure 5.2: Packet initial process.

5.4 Audio Packet Process

Figure 5.3 describes the audio packet process. If R(0) equals zero, R(0) is updated by the packet arrival time. Else calculate interval time using current packet arrival time minus last audio

packet arrival time R(0). We set 200ms as a low threshold because the sender sends an audio packet every 200ms. We set 400ms as a high threshold, one reason is that after 400ms the next packet will arrive; another reason is that once the expected packet arrives, R(0) is updated, then the interval time between R(0) and 400ms will be less than 200ms, so interfering packets arrived between R(0) and 400ms can be successfully excluded. As a result, audio packet identification accuracy is improved. Therefore, if the interval time is between 200ms and 400ms, this packet is treated as an audio packet, assigning the Diffserv value with AUDIO_PRIORITY. If interval time is larger than 400ms, we infer that the network function misses audio packets or network latency is significant, so treat this packet as the first audio packet updating R(0) with the packet arrival time. In this way, the following audio packets will not be missed to be identified when network latency is significant or audio packets are missed.

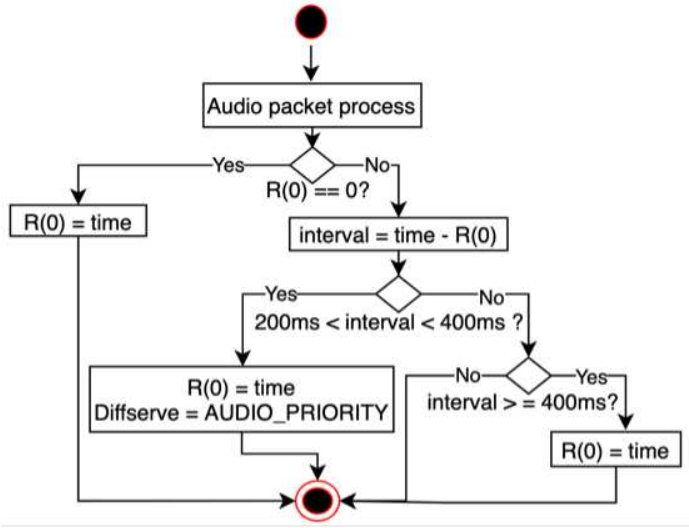

Figure 5.3: Audio packets process.

5.5 Video Packet Process

Figure 5.4 describes the video packet process. The function first examines R(1). If R(1) equals zero, update R(1) and R(2) with the current packet arrival time. Otherwise, calculate frame interval time using the current packet arrival time minus R(1), and calculate packet interval time within a

frame using current packet arrival time minus R(2). In our P4 system, test results demonstrate that the majority of packet inter-arrival times are less than 90ms when packets are continuously transmitted from a host. Consequently, if packet interval time within a frame is less than 90ms, and frame interval time is less than 120ms, treat this packet as a video packet of the current frame, set this packet Diffserv value to VIDEO_PRIORITY, and update R(2) with the current packet arrival time. As a video sender sends a group of three packets every 300ms, we set frame interval time boundaries are 300ms and 600ms. If interval time between frames is larger than 300ms and smaller than 600ms, this packet is treated as the first packet of a video frame updating R(1) and R(2) with this packet arrival time and setting Diffserv value to VIDEO_PRIORITY. If the frame interval time is larger than 600ms, we infer that the system may miss video packets, or the system latency is significant. Thus our solution is treating this packet as the first video packet updating R(1) and R(2) with this packet arrival time, in order to avoid the situation that all following video packets cannot be identified when video packets are lost or network delay is large.

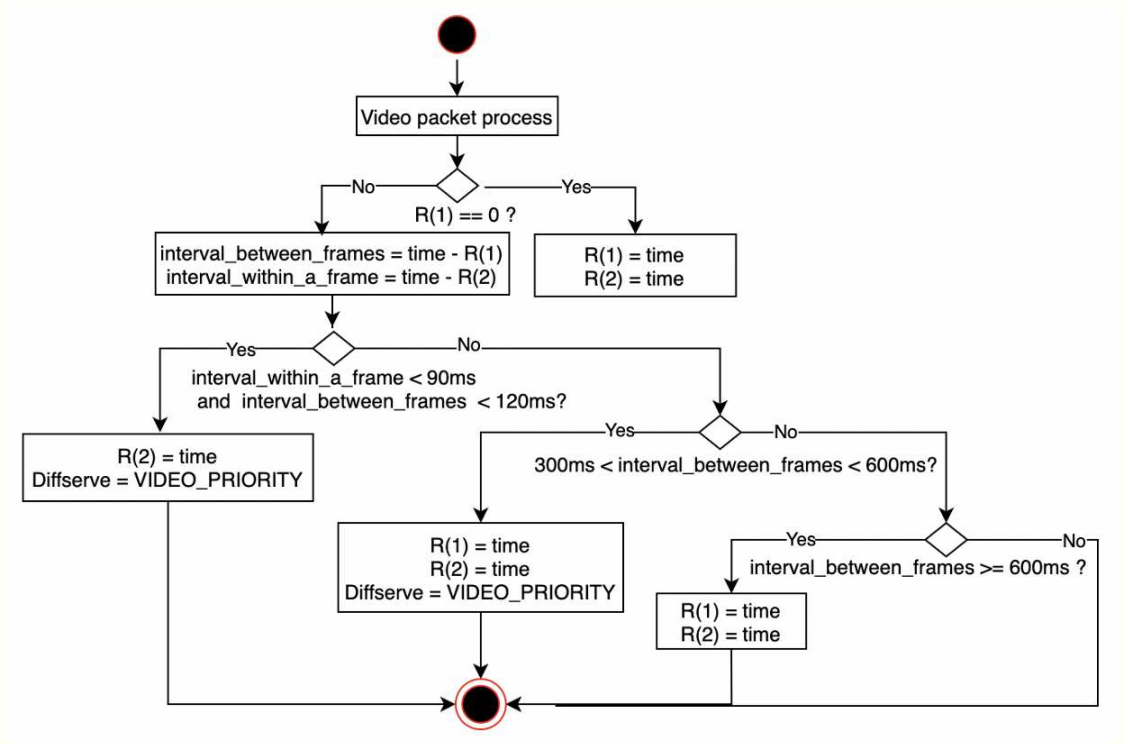

Figure 5.4: Video packet process.

5.6 Summary

This section describes our multimedia traffic identification algorithm implemented in P4. As P4 performance is too slow to process real multimedia traffic, we make a host send small size packets every 200ms simulating audio traffic and send groups of large size packets every 300ms simulating video traffic. Only router s3 employs our algorithm to identify simulated multimedia flows, while other routers skip our algorithm.

Chapter 6

Evaluation

This section presents our testing results in P4 described in section 5. The laboratory environment is shown in figure 5.1. First, we performed four audio traffic identification tests to evaluate our algorithm performance on audio traffic, then performed four audio and video traffic identification tests to evaluate our algorithm performance on multimedia traffic. In the audio traffic identification tests, h1 sends out simulated audio packets at an interval of 200ms to h4. Concurrently, h5 sends random packets to h7, h71 sends random packets to h6, and h61 sends random packets to h8 to interfere with the audio traffic from h1 to h4. The forwarding path from h1 to h4 goes through s1, s2, s3, and s4. The route from h5 to h7 passes through s5, s2, and s7. The traffic path from h71 to h6 goes through s7, s2, s3, and s6, and the traffic path from h61 to h8 passes through s6, s3, and s8. Router s3 employs our algorithm to identify audio packets from mixed flows and set packet Diffserv value to AUDIO_PRIORITY if it is identified as an audio packet. At receivers h4, h7, h6 and h8, the received packets are examined. At h4, if a received packet Diffserv value equals AUDIO_PRIORITY, this packet is identified correctly. At h7, h6, and h8, if a received packet Diffserv value equals AUDIO_PRIORITY, this packet is falsely classified as belonging to audio flows. Host h1 sends to h4 200 audio packets in test1, 400 audio packets in test2, 600 audio packets in test3, and 800 audio packets in test4 at interval 200ms. Table 6.1 shows audio traffic identification test results. The test results reveal that on average 99.05% of audio traffic from h1 to h4 are correctly identified, 1.13% of random audio interfering (RA) packets from h71 to h6 are falsely identified as audio traffic, and 1.07% of random audio interfering packets from h61 to h8 are falsely recognized as audio packets. The FP from h5 to h7 is zero since the traffic does not go through s3. The average accuracy is 98.98%.

Next, we performed four audio and video tests to evaluate our algorithm on multimedia traffic identification. The testing scenario involved h1 sending simulated audio and video traffic to h4, while h5 sends random audio interfering packets to h7, h71 sends random audio interfering packets

Table 6.1: Audio packets mixed with random interference packets identification results.

Test Name	h1 to h4 audio TP	h71 to h6 RA FP	h61 to h8 RA FP	Accuracy
test1	99.0%	1.09%	1.0%	98.99%
test2	99.0%	1.16%	1.75%	98.79%
test3	99.3%	1.10%	0.39%	99.29%
test4	98.88%	1.16%	1.15%	98.86%
average	99.05%	1.13%	1.07%	98.98%

to h6, and h61 sends random video interfering (RV) packets to h8 at random interval time to interfere with multimedia traffics from h1 to h4. In the test1, h1 sends out 300 simulated audio packets and 200 groups of simulated video packets to h4, and each group of video packets includes three packets. Host h1 sends out 900 audio packets and 600 groups of video packets to h4 in test2; h1 sends out 1500 audio packets and 1000 groups of video packets to h4 in test3; h1 sends out 2100 audio packets and 1400 groups of video packets to h4 in test4. Router s3 employs our multimedia identification algorithm to classify packets belonging to audio or video flows. Table 6.2 presents the test results. The table indicates that an average of 95.55% simulated audio packets from h1 to h4 are correctly identified, 93.5% of simulated video packets from h1 to h4 are correctly recognized, 2.51% of random audio interfering packets from h71 to h6 are falsely classified as belonging to audio traffic, and 28.22% of random video interfering packets from h61 to h8 are falsely recognized as video packets. The average audio traffic accuracy is 96.24%, and the average video traffic accuracy is 88.75%.

Table 6.2: Audio and video packets mixed with random packets classification results.

TestName	Audio TP	RA FP	Audio accuracy	Video TP	RV FP	Video accuracy
test1	95.0%	3.45%	95.57%	92.83%	27.84%	88.14%
test2	95.0%	2.82%	95.77%	94.44%	28.6%	89.43%
test3	96.47%	1.79%	97.09%	93.17%	28.4%	88.55%
test4	95.71	1.98%	96.53%	93.55%	27.6%	88.87%
average	95.55%	2.51%	96.24%	93.5%	28.11%	88.75%

In summary, the test results in P4 reveal that our audio traffic identification algorithm achieves an average of 99.05% TP, 1.1% FP, 98.98% accuracy on audio traffic identification. For audio and video traffic mixed with random packets, our algorithm achieves an average of 95.05% TP, 2.51% FP, 96.24% accuracy in audio packet identification, an average of 93.5% TP, 28.11% FP, 88.75% accuracy in video traffic identification.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This novel research leads to the discovery of audio and video traffic patterns. Based on the transport protocol for real-time applications and the video coding standards, we presumed audio and video traffic patterns, then conducted tests to verify our estimations. This study supports the fact that in a video conference, audio and video data are separately transmitted. We were able to reveal several key elements pertaining to the aforementioned audio and video traffic patterns. First, audio packet lengths are between 100 and 200 bytes, and the interval-transmission time is 20ms. Next, a video frame is transmitted by a group of large size packets, and the majority of video packet lengths are larger than 400 bytes. Last, the packets carrying the data of a video frame are sent out continuously with a mean video frame interval time of 33ms.

The findings of these multimedia traffic patterns allow us to design a general audio and video traffic identification algorithm, further implement the algorithm in P4 and perform various experiments. The experimental results reveal that the algorithm achieves high accuracy for audio traffic identification. The audio traffic identification testing results indicate that an average of 99.05% audio packets are correctly identified, only 1.1% of interfering random packets are falsely identified as audio packets, average accuracy is 98.98%. The audio and video traffic identification testing results show that an average of 95.55% audio packets and 93.5% video packets are correctly classified, an average of 2.51% random packets are falsely classified as belonging to audio flows, and 28.11% random packets are falsely recognized as belonging to video flows. Even in a video conference, the quality of audio traffic is crucial as opposed to video traffic when it comes to customer satisfaction.

Our algorithm can identify an overwhelming majority of audio traffic and assign them a high priority to assure audio traffic quality. Compared to the current dominant machine learning-based

traffic classification methods, our traffic identification approach neither requires costly pre-labeled datasets nor does it involve complicated machine learning processes such as model function constructions and training phases. Our algorithm is more efficient as it only requires three global variables and three local variables.

7.2 Future Work

Future work could potentially involve implementing this general traffic identification algorithm in actual networks at different positions such as edge routers and center routers, mixing with various application traffic. According to the experimental results, further optimization of algorithm parameters to improve traffic identification accuracy could be another possible avenue. Another direction could extend this general traffic identification algorithm to identify multiple multimedia sessions simultaneously.

Bibliography

[1] S. Kent and K. Seo. Security architecture for the internet protocol. *RFC 4301*, 2005.

[2] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.

[3] Thuy TT Nguyen and Grenville Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, 10(4):56–76, 2008.

[4] Fannia Pacheco, Ernesto Exposito, Mathieu Gineste, Cedric Baudoin, and Jose Aguilar. Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Communications Surveys & Tutorials*, 21(2):1988–2014, 2018.

[5] Internet assigned numbers authority (iana). https://www.iana.org/, Sep. 2017.

[6] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web*, pages 512–521, 2004.

[7] Andrew W Moore and Konstantina Papagiannaki. Toward the accurate identification of network applications. In *International workshop on passive and active network measurement*, pages 41–54. Springer, 2005.

[8] Guang-Lu Sun, Yibo Xue, Yingfei Dong, Dongsheng Wang, and Chenglong Li. An novel hybrid method for effectively classifying encrypted traffic. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5. IEEE, 2010.

[9] Pan Wang, Xuejiao Chen, Feng Ye, and Zhixin Sun. A survey of techniques for mobile service encrypted traffic classification using deep learning. *IEEE Access*, 7:54024–54033, 2019.

[10] Henning Schulzrinne, Steven Casner, R Frederick, and Van Jacobson. RFC3550: RTP: A transport protocol for real-time applications, 2003.

[11] ITU-T Recommendation H.264. Advanced video coding for generic audiovisual services, 2010.

[12] Video compression picture types. https://en.wikipedia.org/wiki/Video_compression_picture_types, 2020.

[13] Frame rate. https://en.wikipedia.org/wiki/Frame_rate, November 2020.

[14] The r project for statistical computing. https://www.r-project.org/, November 2020.

[15] P416 language specification. version 1.2.1. https://p4.org/p4-spec/docs/P4.16-v1.2.1.html, November 2020.

[16] Sigcomm 2019 p4 tutorial. https://github.com/p4lang/tutorials/tree/sigcomm19, November 2020.

[17] Sigcomm 2019 p4 tutorial. https://github.com/p4lang/tutorials/tree/sigcomm19, November 2020.

[18] P4 language consortium. https://p4.org/, November 2020.

[19] Scapy: Packet crafting for python2 and python3. https://scapy.net/, November 2020.