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

Spring 2021

Master's Committee:

Advisor: Craig Partridge

Joseph Gersch
Hayne Stephen

ABSTRACT

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

With Internet traffic exponentially growing, it is essential for network operators to identify real-time applications such as voice calls and video conferences and assign them high priority. However, with the widespread deployment of encryption protocols, it is challenging to classify encrypted traffic. We discover that in a video conference, audio and video data are separately transmitted, audio data is sent out in small size packets at interval 20ms, a video frame data is sent out in large size packets at interval 33ms, etc. Based on the rules, we propose a general audio and video traffic identification algorithm. In order to evaluate our algorithm performance, we design and implement an audio and video traffic identification algorithm in P4 (Programming Protocol-Independent Packet Processors). The experimental results indicate that our algorithm achieve 98.98% accuracy for voice call identification. For video conferences, it 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, our algorithm bypasses complicated machine learning processes by directly applying audio and video traffic transmission rules on network functions, it also consumes less computation and memory resources.

ACKNOWLEDGEMENTS

I would like to thank Dr. Craig Partridge for his step by step lead and directions so that I can accomplish the project. I also thank Dr. Daniel Ellard from Raytheon BBN Technologies who helped me collect separated voice and video data. I also thank Raytheon BBN Technologies for the financial support.

DEDICATION

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. It is essential for network operators to identify voice call and video conference traffic and give them high priority. However, with the widespread deployment of encryption technologies, identifying encrypted traffic becomes a great challenge. Especially 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 IP header, then it attaches a new IP header before the encrypted IP data. As the source and destination IP addresses in the new IP header are different from the actual IP addresses inside the IPsec tunnels, and the whole IP packet contents are not visible, it is extremely challenging for network operators to identify the encrypted traffic.

Traffic classification approaches generally include port-based, payload-based and flow statistical based approaches using machine learning technologies. As port-based and payload-based approaches require to inspect packet port numbers or payload contents, they are not applicable to encrypted traffic. Flow statistical approaches using machining learning technologies are able to classify encrypted traffic, but they usually require pre-labeled datasets and complicated machine learning steps.

In this work, we discover voice call and video conference flow transmission rules. We find that an audio packet is sent out every 20ms, and audio packet lengths are between 100 bytes and 200 bytes. For video conferences, audio and video data are transmitted separately. A full or compressed video frame data is sent out about every 33ms, and a frame data consists of a few or several large size packets which are sent out continuously. Our tests indicate that 99.8% video packet lengths are larger than 400 bytes. Based on the discovery, we propose a general algorithm identifying voice and video traffic. We also design and implement a voice and video traffic identification algorithm in P4 [2]. Our algorithm is applicable to both encrypted and unencrypted audio and video traffic.

Compared to the prevalent machine learning based traffic classification approaches, our algorithm does not require pre-labeled datasets and training phases. Moreover, our algorithm requires few memory and CPU resources. The experimental results indicate that our algorithm achieves high accuracy on audio traffic identification. We use true positive rate (TP) and false positive rate (FP) to measure traffic identification accuracy. TP refers to the percentage that audio or video traffic are correctly identified as audio or video traffic, FP refers to the percentage that interfering random packets are falsely recognized as audio or video traffic, and true negative rate (TN) refers to the percentage that random interfering packets are not identified as audio or video packets. Accuracy refers to that the sum of TP and TN is divided by total packet number. For voice call traffic mixed with random packets, our algorithm achieves 99% TP, 1% FP and 99% accuracy. In video conferences, our algorithm achieves 95.6% TP, 2.5% FP and 96.2% accuracy on audio traffic identification, 93.5% TP, 28.11% FP and 88.75% accuracy on video traffic identification. Usually, the quality of audio traffic is more important than video traffic for customers. Our algorithm can improve customer satisfaction since it is able to identify major part of audio traffic and assign them high priority.

The rest of the paper is constructed as follows. Section 2 discusses related traffic classification and identification approaches. Section 3 describes our discover of audio and video data transmission rules. In section 4, we propose a general audio and video traffic identification algorithm. In section 5, we design and implement an audio and video traffic identification algorithm in P4, and section 6 presents experimental results in P4. Section 7 concludes the paper and suggests future work.

Chapter 2

Related Work

Network traffic classification plays a vital role for network management and resource allocation. Network traffic identification and classification methods evolved from port-based, payload-based to flow characteristics based using machine learning technologies [3, 4]. Port-based traffic classification approaches rely on well-known TCP or UDP port numbers registered in Internet Assigned Numbers Authority (IANA) [5]. Due to the fact that more 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 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, then use these signatures to identify P2P traffic. Moore and Papagiannak [7] devised nine methods which are used separately or in a combination to classify flows. The first method examines port number, the second method inspects packet header, and other methods examine payload contents.

However, with the popularity of encryption technology adoption, port-based and payload-based approaches are ineffective on encrypted traffic. As a result, a great number of flow-based traffic classification approaches based on machine learning technologies are proposed. These methods rely on traffic statistical properties such as flow duration distribution, flow idle time, packet inter-arrival time, packet lengths and so on [3]. Machine learning technologies are mainly 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 which are most close to pre-labeled outputs. 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 a specific application traffic. In addition, some hybrid methods are proposed. 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 use SSL and TLS signatures to identify traffic encrypted with SSL or TLS protocols, then use 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 is not applicable to end-to-end encryption traffic. Nguyen et al. [3] reviewed 18 significant machine learning based IP traffic classification works from 2004 to 2007. In 2019, Pacheco et al. [4] presented a systematic overview of steps to achieve traffic classification based on machine learning technologies. [4] describes that general supervised machine learning technology steps include data collection, feature extraction, feature reduction, algorithm selection and model construction, validation of classification models. Data collection is the process to gather information and label datasets. Feature extraction step measures and computes different attributes' contribution on the model. Feature reduction is an optional step to find the features that have high influences on classification decision. Algorithm selection and model construction step chooses a machine learning algorithm and finds the model parameters that minimize the differences between model outputs and pre-labeled outputs. This step is also referred as training phase. Validation of classification models use the trained model to test labeled data, which is also called testing phase. As you can see that machine learning technologies require complexed steps and processes. In recent years, deep learning technologies obtain more attention for traffic classification. Wang et al. [9] survey deep leaning applications for mobile encrypted traffic classification and present a deep learning based mobile encrypted traffic classification framework. Deep learning technologies are divided into supervised, unsupervised and semi-supervised methods. For supervised deep learning methods, dataset label is also a difficult and time-consuming task. In comparison, our traffic identification algorithm bypasses complicated machine learning steps and costly dataset label task because we directly apply audio and video traffic transmission rules on network functions.

Chapter 3

Multimedia Transmission Rules

According to RFC3550 [10] that audio conference application participants send audio data in small chunks of 20ms duration, we assume 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. I frame is a complete picture, P frame only holds changes to the previous frame, and 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 a video frame data are continuously transmitted.

In order to verify our assumption about audio and video traffic pattern, we perform two Skype voice call tests test_1 and test_2, and two video conference tests test_3 and test_4 and capture packets at both terminals to measure traffic properties in terms of packet interval time and packet lengths. Terminal_1 is in Colorado State and terminal_2 is in Massachusetts State, USA. For all the tests, first we filter captured packets by conditions such as source and destination IP addresses and protocols. Then we use R [14] to compute traffic properties in terms of packet lengths and packet interval time statistic. First, we collect and analyze voice call packet lengths so as to separate audio traffic from a video conference by packet lengths. We filter captured voice call packets in test_1 and test_2 by the conditions that source IP address is terminal_1, destination IP address is terminal_2, and protocol is UDP. Then we use R to get graphical and numerical audio packet lengths statistic.

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 show that the most majority of audio packet lengths are between 100 and 200 bytes. Numerical statistical results computed by R show that average 98.32% audio packet lengths are between 100 and 200 bytes.

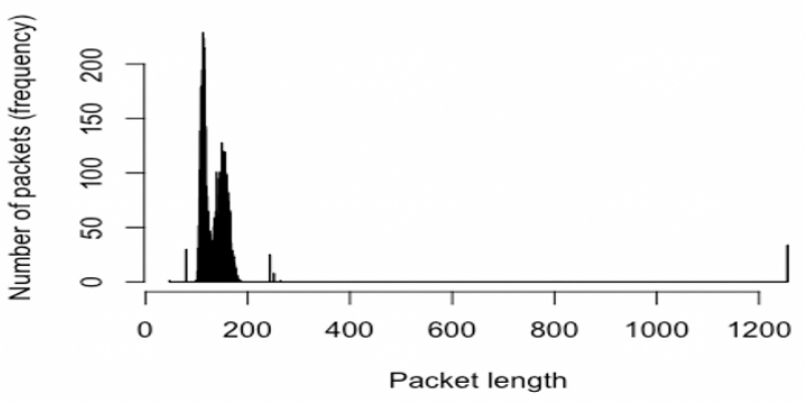

Figure 3.1: Skype audio packet length distribution in test_1.

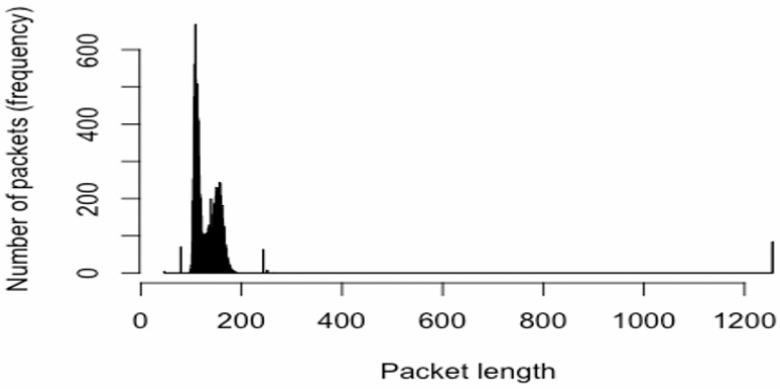

Figure 3.2: Skype audio packet length distribution in test_2.

In order to compute the statistic of skype audio packet inter-transmission time, we filter out skype audio packets by the conditions that source IP address equals to IP address of terminal_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 compute packet inter-transmission time by using the current packet time minuses the previous packet time in C++. Then program in R to compute packet inter-transmission time statistic. Figure 3.3 and 3.4 show the skype audio packet inter-transmission time frequency in test_1and test_2. The horizontal axis represents packet interval time in microsecond, 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 average 97.68% skype audio packet interval time is between 10ms and 26.2ms, and average 76.14% skype audio packet interval time is between 19.7ms and 21.5ms.

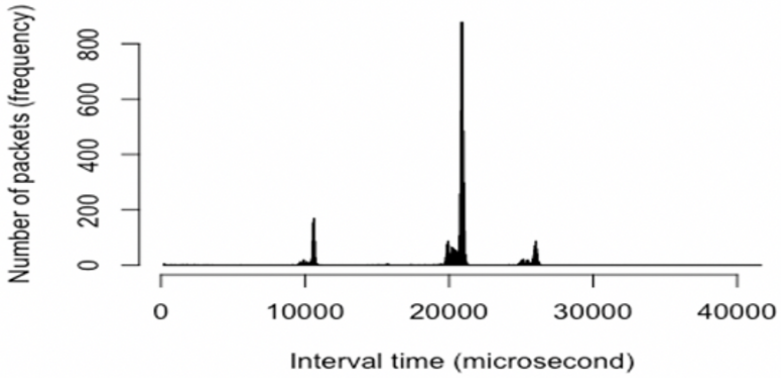

Figure 3.3: Skype audio packet interval time distribution in test_1.

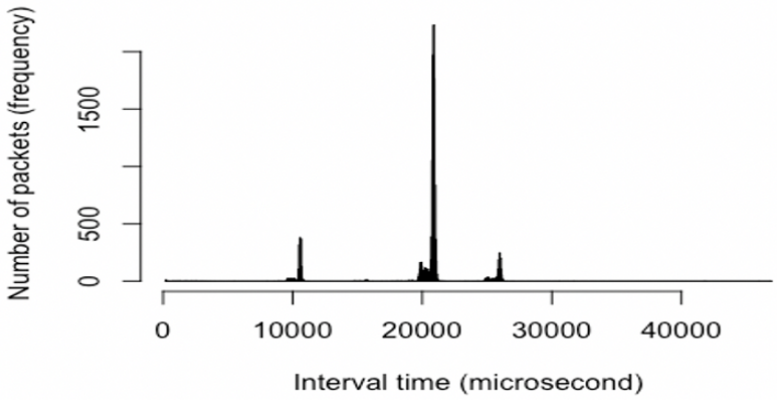

Figure 3.4: Skype audio packet interval time distribution in test_2.

3.2 Video Data Transmission Rules

Next, we perform two Skype video conference tests test_3 and test_4 to examine video traffic pattern. Based on the above statistical results that audio packet lengths are less than 200 bytes, we filter out packets larger than 200 bytes to exclude audio packets in test_3 and test_4. Then further filter out packets from terminal_1destinated to terminal_2 and protocol is UDP. We discover that the video data are transmitted in groups of packets. Each group includes 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. Then program in R to compute frame interval time statistic. Figure 3.5 demonstrates the video frame interval time frequency distribution in test_3, and figure 3.6 demonstrates the frame interval time frequency distribution in test_4. The horizontal axis represents interval time between frames in microsecond, and the vertical axis represents packet number. These two figures indicate that a large part of frame interval time is between 20ms and 48ms.

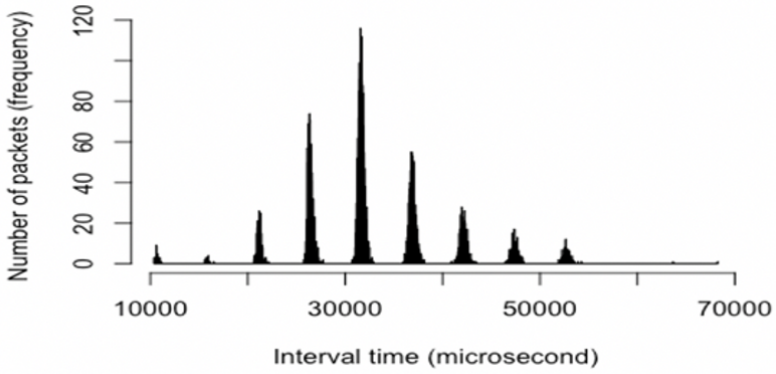

Figure 3.5: Skype video frame interval time frequency distribution in test_3.

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 average 95.84% frame interval time is between 20ms and 48ms, and mean frame interval time is 33.05ms. The results prove our assumption that video frame interval time is around 33ms.

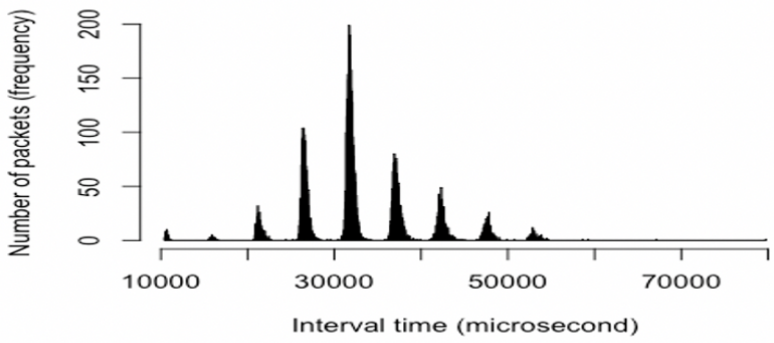

Figure 3.6: Skype video frame interval time frequency distribution in test_4.

We also analyze interval time between packets belonging to one frame. Use the current packet arrival time minus the previous packet time belonging to a frame, then use R to compute interval time statistic. Table 3.2 shows the interval time ratio of Skype video packet belonging to a frame in test_3 and test_4 computed by R. The table displays that average 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 assumption that packets carrying one frame data are continuously sent out.

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

Next, we compute Skype video packet length statistic. Figure 3.7 demonstrates the Skype video packet length distribution in test_3, and 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 average 99.82% video packet lengths are larger than 400 bytes, and average 99.59% video packet lengths are larger than 600 bytes. The experimental results approve our assumption that video frame data are transmitted by large size packets.

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

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 microsecond ratio
test_3	99.16%
test_4	99.13%
average	99.15%

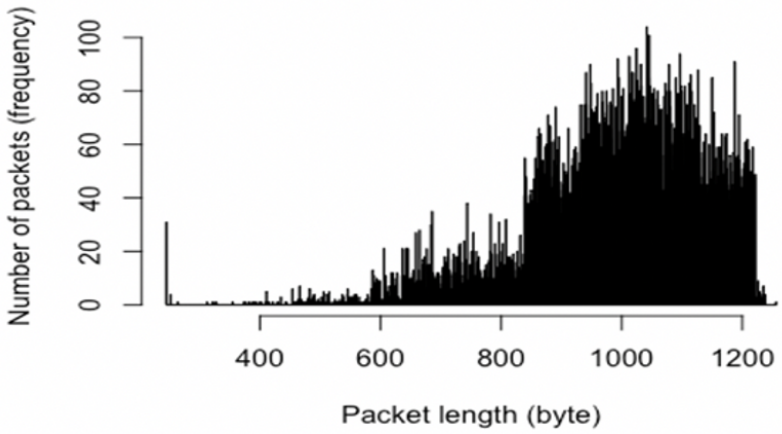

Figure 3.7: Skype video packet length distribution in test_3.

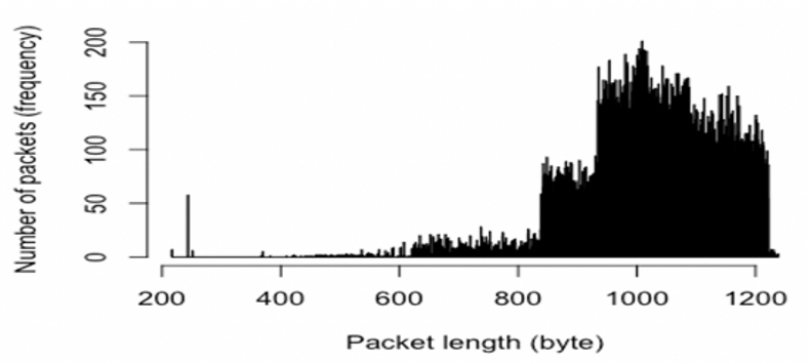

Figure 3.8: Skype video packet length distribution in test_4.

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 assume 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 indicate that an audio packet is sent out every 20ms, audio packet sizes are between 100 and 200 bytes. The test results also show that video data is sent out in groups of large size packets, mean interval time between groups 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, and 99.8% video packet lengths are larger than 400 bytes. The test results prove our assumptions about audio and video traffic pattern.

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 figure 4.1, 4.2 and 4.3. Figure 4.1 explains multimedia traffic initial process. Figure 4.2 presents audio packet process, and figure 4.3 describes video packet process. R(0), R(1) and R(2) are global variables or registers in P4. R(0) stores last audio packet arrival time, R(1) stores the arrival time of the first packet of a video frame, and R(2) saves the last 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.

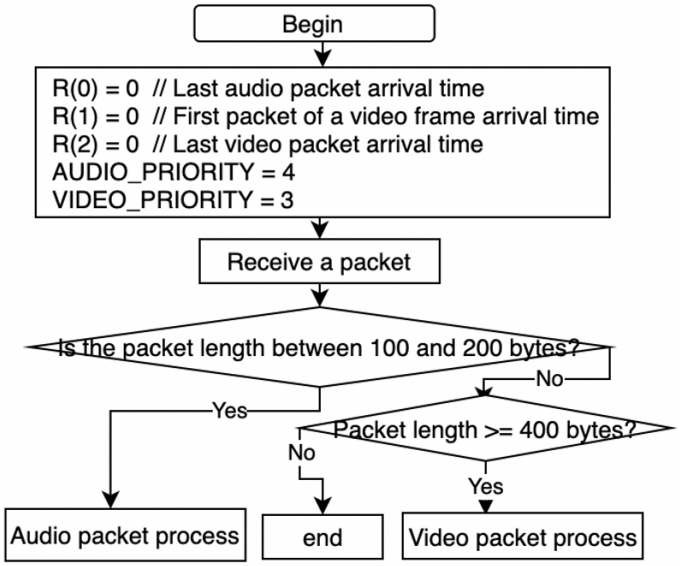

Figure 4.1: Multimedia packet initial process.

Figure 4.2 describes audio packet process. First, the process examines R(0) value. If R(0) equals to 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 last audio packet arrival time R(0). According to the statistical results in section 3 that 97.68% 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 Diffserve 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.

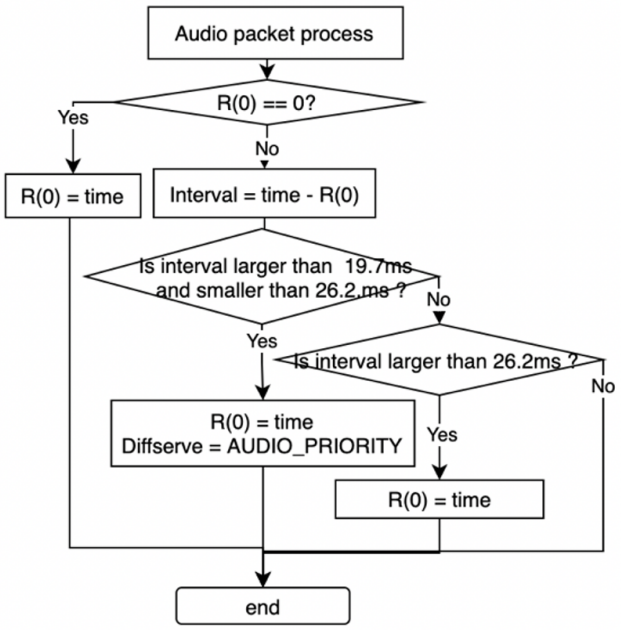

Figure 4.2: Audio packet process.

Figure 4.3 describes video packet process. R(1) records the first packet of a frame arrival time and R(2) records 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). Based on the discovery of skype video traffic pattern in section 3 that 99% video packet interval time within a frame is less than 0.1ms and more than 99% video pictures finishes 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 current packet arrival time, and setting this packet Diffserve 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 Diffserve 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.

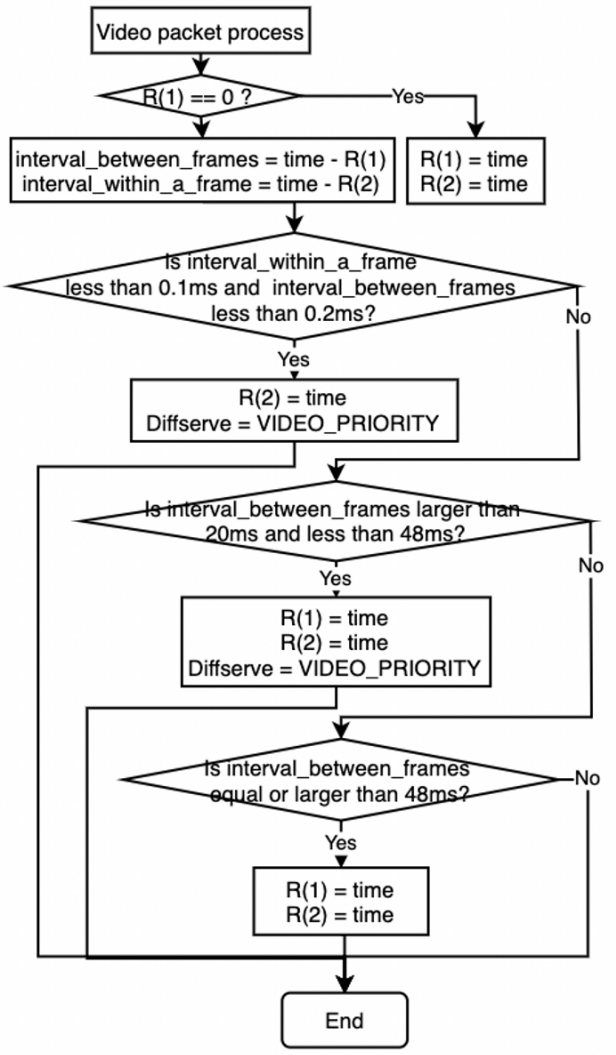

Figure 4.3: Video packet process.

Chapter 5

Algorithm Implementation in P4

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

5.1 Network Topology

First, download and configure P4 virtual machine image from P4 tutorials [16,17]. Based on the basic example in the tutorials, create a new network topology by modifying the file topology.json, then specify IP packet routing rules for each router. We create 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.

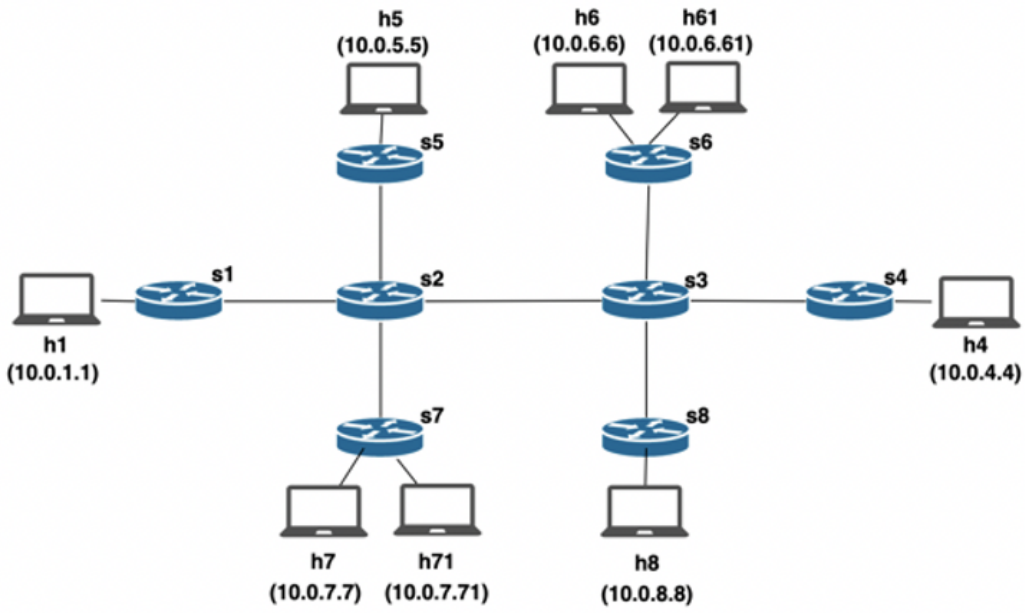

Figure 5.1: P4 network topology.

5.2 Challenging Issues and Solutions

One issue we encountered is that P4 performance is slow. 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 show that packet inter-arrival time are all larger than 10ms, and about 97% packet interval time are between 10ms and 40ms, 99% packet interval time are between 10ms and 60ms. However, video packet statistic in section 3 indicates that about 85% packet interval time within a frame are less than 5 microseconds, about 99% packet interval time within a frame are less than 100 microseconds. Therefore, P4 performance is not good enough to process actual multimedia traffic. In order to evaluate our multimedia traffic identification algorithm in P4, our solution is to amplify packet inter-transmission time. 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 simulates a multimedia sender sending small size packets every 200ms simulating audio traffic and sending a group of large size packets every 300ms, a group of packets carrying a frame data are continuously sent out simulating video traffic.

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 a video frame data consists of three packets which are sent out without a break. Some other hosts send out random packets at random interval time to interfere the audio or video traffic from h1 to h4. As the statistical results in section 3 show that 98.32% audio packet lengths are between 100 and 200 bytes, and packet lengths vary from 1 byte to 1514 bytes, 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 audio interfere senders send the same size packet as an audio packet size but with different contents. Otherwise, random interfering senders send a different size packet. As the statistical results in section 3 show that 99.59% video packet lengths are between 600 bytes and 1230 bytes, our random interference 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 interference senders send the same size packets as a video packet size but with different contents. Otherwise, send different size random packets to interfere 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 our algorithm functions are added in the control block MyIngress(). In P4, a packet received at a router only stores information of this packet. However, we need time information of previous packets. 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 the 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, while we just want 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's MAC address, our solution 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. By this way, other routers ignore our algorithm except s3.

5.3 Algorithm Initial Process

Figure 5.2 depicts our algorithm initial process. Register R(0) stores last audio packet arrival time, R(1) stores the first packet arrival time of a frame, and R(2) stores 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 exam-

ines 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 45 bytes, it is processed by audio process module. Else if the packet length equals to video packet length 57 bytes, it enters video process module. Otherwise, the packet bypasses our identification algorithm.

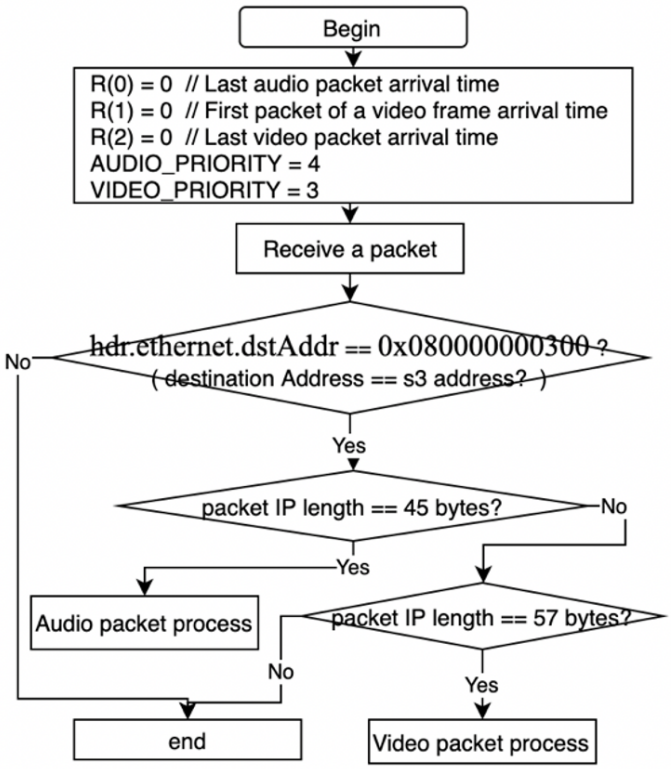

Figure 5.2: Packet initial process.

5.4 Audio Packet Process

Figure 5.3 describes audio packet process. If R(0) equals to 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 low threshold because audio packet interval time is larger than 200ms since the sender sends an audio packet every 200ms. We set 400ms as high threshold, one reason is that after 400ms the next packet will arrive; another reason is that once

the expected packet arrive, 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 interval time is between 200ms and 400ms, this packet is treated as an audio packet, assigning Diffserve 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. By this way, the following audio packets will not be missed to be identified when network latency is significant or audio packets are missed.

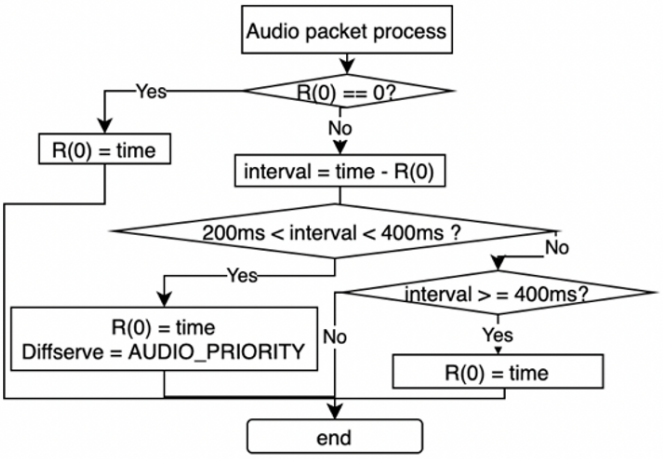

Figure 5.3: Audio packets process.

5.5 Video Packet Process

Figure 5.4 describes video packet process. The function first examines R(1). If R(1) equals to zero, update R(1) and R(2) with the current packet arrival time. Otherwise calculate packet 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 show that majority packet inter-arrival time are less than 90ms when packets are continuously sent out 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, and set

this packet Diffserve 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, so 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 Diffserve value to VIDEO_PRIORITY. Else if frame interval time is larger than 600ms, we infer that the system may miss video packets, or the system latency is significant, our solution is treating this packet as the first video packet updating R(1) and R(2) with this packet arrival time to avoid the situation that all following video packets are missed to be identified when video packets are lost or network delay is large.

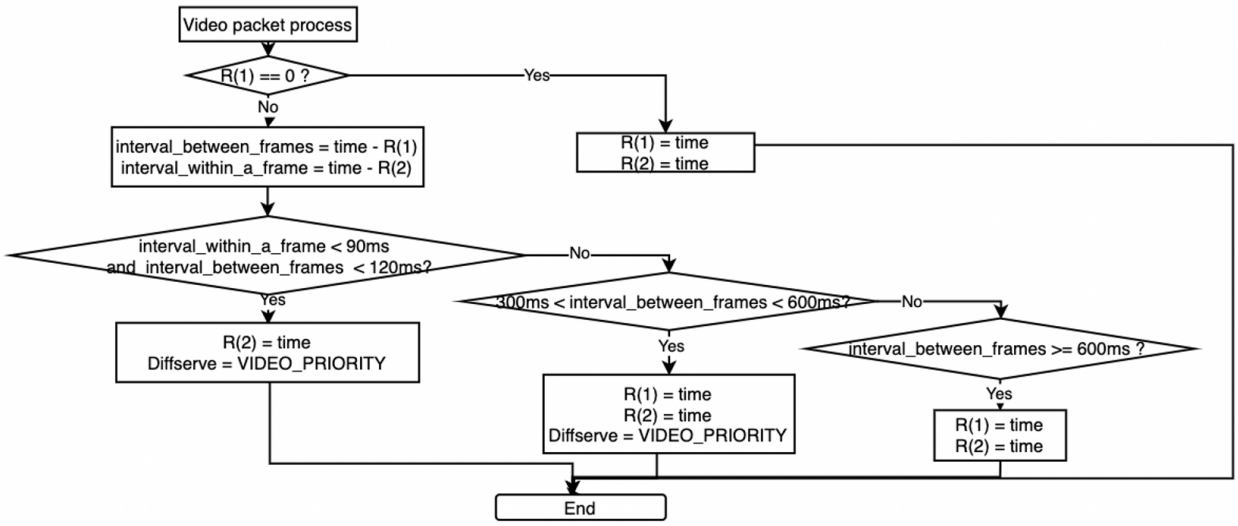

Figure 5.4: Video packet process.

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 to send small size packets every 200ms simulting 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 perform four audio traffic identification tests to evaluate our algorithm performance on audio traffic, then perform four audio and video traffic identification tests to evaluate our algorithm performance on multimedia traffic. In the audio traffic identification tests, h1 sends simulated audio packets at interval 200ms to h4. At the same time, h5 sends random packets to h7, h71 sends random packets to h6, and h61 sends random packets to h8 to interfere 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 Diffserve 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 Diffserve value equals to AUDIO_PRIORITY, this packet is identified correctly. At h7, h6 and h8, if a received packet Diffserve value equals to AUDIO _PRIORITY, this packet is falsely classified 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 show that average 99.05% audio traffic from h1 to h4 are correctly identified, average 1.13% random packets from h71 to h6 are falsely identified as audio traffic, and average 1.07% random 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 perform four audio and video tests to evaluate our algorithm on multimedia traffic identification. The testing scenario is that h1 sends simulated audio and video traffic to h4, while h5 sends random audio interfering packets to h7, h71 sends random audio interfering packets to h6, and h61 sends random video interfering packets to h8 at random interval time to interfere mul-

timedia traffics from h1 to h4. In 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 shows the test results. The table shows that average 95.55% simulated audio packets from h1 to h4 are correctly identified, average 93.5% simulated video packets from h1 to h4 are correctly recognized, average 2.51% random audio interfering packets from h71 to h6 are falsely classified belonging to audio traffic, average 28.22% random video 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%.

In summary, the test results in P4 show that our audio traffic identification algorithm achieves average 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 average 95.05% TP, 2.51% FP, 96.24% accuracy on audio packets identification, average 93.5% TP, 28.11% FP, 88.75% accuracy on video traffic identification.

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

Test Name	h1 to h4 audio TP	h71 to h6 interfering audio FP	h61 to h8 interfering audio 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%

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

TestName	Audio TP	RandomAudio FP	Audio accuracy	Video TP	RandomVideo 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%

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This paper first presents our discovery of audio and video traffic pattern. Based on real-time application transport protocol and video coding standards, we assume audio and video traffic pattern, then perform voice calls and video conferences to verify our assumptions. Our tests prove that in a video conference, audio and video data are separately transmitted. We discover that audio packet lengths are between 100 and 200 bytes, an audio packet is sent out every 20ms, a video frame is transmitted by a group of large size packets, majority of video packet lengths are larger than 400 bytes, a group of packets carrying a video frame data are sent out continuously, mean video frame interval time is 33ms. Then based on the multimedia traffic pattern, we design a general audio and video traffic identification algorithm. In order to evaluate our algorithm performance, we design and implement a multimedia traffic identification algorithm in P4 and perform some tests. The testing results show that our algorithm achieve high accuracy for audio traffic identification. The audio traffic identification testing results indicate that average 99.05% audio packets are correctly identified, only 1.1% interfering random packets are falsely identified as audio packets, average accuracy is 98.98%. The audio and video traffic identification testing results show that average 95.55% audio packets and average 93.5% video packets are correctly classified, and average 2.51% random packets are falsely classified belonging to audio flows, and average 28.11% random packets are falsely recognized belonging to video flows. Even in a video conference, the quality of audio traffic is more important than video traffic for customer satisfaction. Our algorithm is able to identify overwhelming majority of audio traffic and assign them high priority to assure audio traffic quality. Compared to current dominant machine learning based traffic classification methods, our traffic identification approach neither requires costly pre-labeled datasets nor requires compli-

cated 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

The future work may involve implementing our general traffic identification algorithm in variety of real network systems at different positions such as edge routers and center routers, then compare their performances. According to the experimental results, further optimize algorithm parameters to improve traffic identification accuracy.

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.