Multimedia Transmission Rules and Encrypted Audio and Video Traffic Identification implementation in P4

Jiping Lu, Craig Partridge

Computer Science Department, Colorado State University, USA

{jiping.lu, craig.partridge}@colostate.edu

ABSTRACT

With Internet traffic exponentially growing, it becomes 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 becomes challenging to classify encrypted traffic. In this paper, we present our discovery of audio and video traffic transmission rules, and propose an audio and video traffic identification algorithm. We also design and implement an audio and video traffic identification algorithm in P4 (Programming Protocol-Independent Packet Processors) system based on our discovery of multimedia traffic pattern. The experimental results indicate that our algorithm achieve 99% true positive rate and 1.1% false positive rate for identifying audio traffic mixed with random traffic. 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, and consumes less computation and memory resources.

Keywords

Traffic identification, traffic classification, audio and video traffic transmission rules, voice and video traffic identification, encrypted audio and video traffic identification.

1. INTRODUCTION

Recently Internet traffic has been exponentially growing. Some applications like peer-to-peer file sharing consume a large amount of network bandwidth, consequently real-time applications such as voice calls and video conferences endure poor performance due to increased network congestions and latency. To improve customer satisfaction, network operators try to identify voice calls 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], which is widely employed, 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 including the actual IP addresses are not visible, it becomes extremely challenging for network operators to identify this kind of encrypted traffic.

Traditional traffic classification approaches include port-based, payload-based and flow statistical based approaches using machine learning technologies. Port-based and payload-based approaches require to inspect packet port numbers and payload contents, so they are not applicable for encrypted traffic. Then flow statistical approaches using machining learning technologies are proposed. These methods are able to classify encrypted traffic, but they usually require large number of pre-labeled datasets to construct a trained classification model before they use the model to classify traffic. Furthermore, they require to store a certain number of packet information to compare with the traffic models. Therefore, machine learning based approaches consume a certain quantity of computation and memory resources.

In this work, we discover voice call and video conference flow transmission rules, and our experimental results prove the rules. We discover 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 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 applying the rules in terms of packet lengths and packet interval time on network functions like routers to identify audio and video traffic. Finally, we design and implement an audio and video traffic identification algorithm in P4 (Programming Protocol-Independent Packet Processors) [2]. Our algorithm is applicable for 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 has high accuracy for audio traffic identification. We use true positive and false positive rate to measure traffic identification accuracy. True positive rate refers to the percentage that audio or video traffic are correctly identified as audio or video traffic, and false positive rate refers to the percentage that interference random packets are falsely recognized as audio or video traffic. For voice call traffic mixed with random packets, our algorithm achieves 99% true positive rate and 1% false positive rate. In video conferences, our algorithm achieves 95.6% true positive rate and 2.5% false positive rate for audio traffic identification, and it reaches 93.5% true positive rate and 28.11% false positive rate for video traffic identification. Usually, customers have more demand for audio traffic quality than video traffic quality. Our algorithm can improve customer satisfaction because 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 implementation 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.

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-base traffic classification methods become less reliable. Consequently, payload-based traffic classification approaches emerged, they 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, usually right after TCP/IP header. The authors use these fixed values at specific fields as application signatures to identify P2P traffic. Moore and Papagiannak [7] devised nine methods which are used separately or in a combination to classify flows, and the first method examines port number, the second method inspects packet header, other methods examine payload contents.

However, with the popularity of encryption technology adoption, port-based and payload-based approaches are not able to identify encrypted traffic. As a result, a large number of flow-based traffic classification approaches based on machine learning technologies are proposed, which rely on traffic statistical properties such as flow duration distribution, flow idle time, packet inter-arrival time, packet lengths and so on. The main work of machine learning technology is to find and construct application patterns with provided datasets. 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. To specify a specific application traffic, supervised machine learning technologies are used. 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, while it is not applicable to end-to-end encryption traffic where whole IP data are encapsulated.

The pervasive methods for encrypted traffic classification are based on machine learning technologies. 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. The paper also described general supervised machine learning technology steps: 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 also divided into supervised, unsupervised and semi-supervised methods. For supervised deep learning methods, datasets label is also a difficult and time-consuming task. Deep learning approaches also require complicated learning processes.

In comparison, our traffic identification algorithm bypasses complicated machine learning steps and costly dataset label task because we discover audio and video traffic transmission rules and directly apply these rules on network functions. As a result, our traffic identification algorithm is more efficient.

3. MULTIMEDIA TRANSMISSION RULES

According to RFC3550 [10] specifications that audio conference application participants send audio data in small chunks of 20ms duration, we assume that audio packet is sent out every 20 milliseconds. As noted in [10] 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 is separately transmitted. As described in H.264 [11] and [12], in order to compress video frame data, video frames are transmitted in I, B, 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. Most media applications send 30 frames per second [13]. From these standards, we assume that video frame inter-transmit time is about 33 milliseconds, and a video frame data is continuously transmitted in a few packets if it is a compressed images or several large size packets if it is a full image.

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. R is a programing language and free software environment for statistical computing and graphics. First of all, we collect and analyze voice call packet lengths so as to separate audio traffic from 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. Figure 3.1.1 shows test\_1 packet length statistic, and figure 3.2.1 represents test\_2 packet length statistic. The horizontal axis denotes skype voice calls duration time in second, and the vertical axis denotes packet length in bytes. The two figures indicate that the majority of audio packet lengths are less than 200 bytes.

A picture containing chart

Description automatically generated

Figure 3.1.1: Skype audio packet length statistic of test\_1

A picture containing diagram

Description automatically generated

Figure 3.2.1 Skype audio packet length statistic of test\_2

Figure 3.1.2 displays test\_1 audio packet length distribution, and figure 3.2.2 displays test\_2 audio packet length distribution in R. The horizontal axis denotes packet length in bytes, and the vertical axis denotes packet number. From these two figures, we further draw the conclusion that the vast majority of audio packet lengths are between 100 and 200 bytes.

Chart, histogram

Description automatically generated

Figure 3.1.2: Skype audio packet length distribution in test\_1

Chart, histogram

Description automatically generated

Figure 3.2.2: Skype audio packet length distribution in test\_2

Table 3.1 presents audio packet length distribution ratios computed by R. In test\_1 and test\_2, average 98.91% audio packet lengths are less than 200 bytes, and average 98.32% audio packet lengths are between 100 and 200 bytes. Therefore, we come to a conclusion that major part of audio packet lengths are between 100 and 200 bytes.

Table 3.1: Skype audio packet length ratio

|  |  |  |
| --- | --- | --- |
|  | (packet length < 200) / total | (100 < packet length < 200) / total |
| test\_1 | 98.90% | 98.38% |
| test\_2 | 98.92% | 98.26% |
| average | 98.91% | 98.32% |

In order to compute the statistic of skype audio packet interval time, our next step is to exclude background traffic, filter out only skype audio packets from captured packets in test\_1 and test\_2. For simplicity, we only filter out packets from terminal\_1 destinated to terminal\_2. Based on the above audio packet length statistical results, we filter out skype audio packets by setting the conditions that source IP address is the IP address of terminal\_1, destination IP address is the IP address of terminal\_2, protocol is UDP and packet lengths are between 100 and 200 bytes. With the filtered skype audio packets, we compute packet interval time by using the current packet time minus the previous packet time. Then we program in R to compute interval time statistic. Figure 3.1.3 demonstrates test\_1 packet interval time statistic, and figure 3.1.4 demonstrates test\_2 packet interval time statistic. The horizontal axis indicates audio conference time in second, and the vertical axis indicates packet interval time in microsecond. From these two figures, we can see that most part of audio packet interval time is around 20ms.

A picture containing diagram

Description automatically generated

Figure 3.1.3: Skype audio packet interval time statistic in test\_1

A picture containing chart

Description automatically generated

Figure 3.2.3: Skype audio packet interval time statistic in test\_2

Figure 3.1.4 shows skype audio packet interval time frequency distribution in test\_1, and figure 3.2.4 shows skype audio packet interval time frequency distribution in test\_2. The horizontal axis represents packet interval time in microsecond, and the vertical axis represents packet number or frequency. From these two figures, we draw a conclusion that main part of audio packet interval time is distributed around 20ms.

Chart, diagram, box and whisker chart

Description automatically generated

Figure 3.1.4: Skype audio packet interval time distribution in test\_1

Chart, diagram, box and whisker chart

Description automatically generated

Figure 3.2.4: Skype audio packet interval time distribution in test\_2

Table 3.2 lists test\_1 and test\_2 audio packet interval time ratios computed by R. The table shows that average 97.68% skype audio packet interval time is between 10ms and 26.2ms, and average 76.14% skype audio packet interval time falls between 19.7ms and 21.5ms. The statistical results prove our assumption that audio packet interval time is around 20ms.

Table 3.2: Skype audio interval time ratio

|  |  |  |
| --- | --- | --- |
|  | (10ms < interval time < 26.2ms) / total | (19.7ms < interval time < 21.5ms) / total |
| Test\_1 | 97.49% | 76.12% |
| Test\_2 | 97.86% | 76.16% |
| Average | 97.68% | 76.14% |

Next, we set out to verify our assumption that video conference terminals send out full image or compressed image data at interval 33ms, and each image or compressed image is sent out by several or a few continuous large size packets. We perform two Skype video conference tests test\_3 and test\_4. Based on the above statistical results that audio packet size is less than 200 bytes, we filter test\_3 and test\_4 packets which are larger than 200 bytes so that audio packets are excluded. For simplicity, we only filter packets from terminal\_1destinated to terminal\_2 and protocol is UDP. We discover that video data is transmitted in groups of packets. Each group includes several or a few continuous packets which are sent out continuously. We use the first packet of the current group of audio packets minus the first packet of the previous group to get the interval time between frames. Then we program in R to compute frame interval time statistic. Figure 3.3.1 illustrates Skype video data interval time between frames in test\_3, and figure 3.4.1 illustrates frame interval time in test\_4. The horizontal axis denotes conference time in seconds, and the vertical axis denotes interval time between frames in microsecond. These two graphs indicate that overwhelming majority of frame interval time is between 20ms and 50ms.

A picture containing diagram

Description automatically generated

Figure 3.3.1: Skype video frame interval time statistic in test\_3

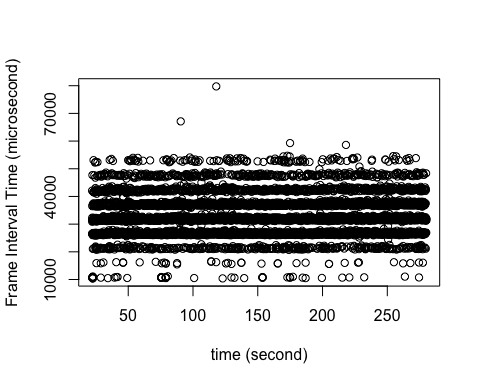


Figure 3.4.1: Skype video frame interval time statistic in test\_4

Figure 3.3.2 demonstrates Skype video frame interval time frequency distribution in test\_3, and figure 3.4.2 demonstrates 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 or frequency. These two figures indicate that large part of frame interval time is distributed between 20ms and 48ms.

Diagram

Description automatically generated

Figure 3.3.2: Skype video frame interval time frequency distribution in test\_3

Diagram

Description automatically generated

Figure 3.4.2: Skype video frame interval time frequency distribution in test\_4

Table 3.3 demonstrates test\_3 and test\_4 Skype video frame interval time ratio and mean interval time computed by R. 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.

Table 3.3: Skype video data interval time ratio and mean

|  |  |  |
| --- | --- | --- |
|  | 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 analyze interval time between packets belonging to one frame. We use the current packet arrival time minus the previous packet time within a frame, then use R to compute interval time statistic. Figure 3.3.3 displays skype video packet interval time statistic within a frame in test\_3, and figure 3.4.3 displays interval time statistic within a frame in test\_4. The horizontal axis represents conference time in second, and the vertical axis represents packet interval time within a frame in microsecond. These two figures indicate that major part of packet interval time within a frame is less than 100 microseconds.

Chart, histogram

Description automatically generated

Figure 3.3.3: Skype video packet interval time within a frame in test\_3

Chart, histogram

Description automatically generated

Figure 3.4.3: Skype video packet interval time within a frame in test\_4

Table 3.4 shows Skype video packet interval time within a frame ratio in test\_3 and test\_4 computed by R. The table displays that average 84.99% packet interval time within a frame is less than 5 microseconds, and 99.04% packet interval time within a frame is less than 100 microseconds. The results prove our assumption that a frame data is sent out by continuous packets.

Table 3.4: Skype video packet interval time within a frame ratio

|  |  |  |
| --- | --- | --- |
|  | 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 |

We also compute statistic of duration time to finish transmitting a video frame by R. Figure 3.3.4 displays transmitting a video frame time duration statistic in test\_3, and figure 3.4.4 shows transmitting a video frame time duration statistic in test\_4. The horizontal axis denotes video conference time in second, and the vertical axis denotes a frame transmission duration time in microsecond. The two figures indicate that majority part of a video frame duration time are less 200 microseconds. Table 3.5 represents a video frame duration time statistical ratios computed by R. The statistical results show that average 99.15% video frames finish transmitting a frame data within 0.2ms.

Chart, scatter chart

Description automatically generated

Figure 3.3.4: Skype video a frame time span statistic in test\_3

Chart

Description automatically generated

Figure 3.4.4: Skype video a frame time span in test\_4

Table 3.5: Skype video a frame duration time ratio

|  |  |
| --- | --- |
|  | a video frame duration time less than 200 microsecond ratio |
| test\_3 | 99.16% |
| test\_4 | 99.13% |
| average | 99.15% |

In order to verify that video data are transmitted in large size packets, we compute Skype video packet length statistic in R. Figure 3.3.5 demonstrates Skype video packet length distribution in test\_3, and figure 3.4.5 demonstrates video packet length distribution in test\_4. The horizontal axis denotes video packet length in bytes, and the vertical axis denotes number of packets or frequency distribution. From these two graphs we come to the conclusion that overwhelming majority of video packet lengths are larger than 400 bytes.

Chart, histogram

Description automatically generated

Figure 3.3.5: Skype video packet length distribution in test\_3

Chart, histogram

Description automatically generated

Figure 3.4.5: Skype video packet length distribution in test\_4

Table 3.6 illustrates Skype video packet length ratio statistic in test\_3 and test\_4 computed by R. The table demonstrates that average 99.82% video packet lengths are larger than 400 bytes, and average 99.59% video packets lengths are larger than 600 bytes. The experimental results approve our assumption that video frame data is transmitted by large size packets.

Table 3.6: Skype video packet length ratio

|  |  |  |
| --- | --- | --- |
|  | packet length > 400 bytes / total | packet length > 600 bytes / total |
| Test\_3 | 99.80% | 98.71% |
| Test\_4 | 99.83% | 99.46% |
| average | 99.82% | 99.59% |

In summary, according to audio and video standard specifications, we know that audio and video data in a video conference are separately transmitted, and we 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 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% group interval time is distributed between 22ms and 48ms, 99% packet interval time within a group 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.

4. TRAFFIC IDENTIFICATION ALGORITHM

Based on the multimedia traffic pattern discovered in section 3, we propose a general multimedia traffic identification algorithm, and the algorithm can be implemented on network functions such as routers and switches. Figure 4.1, 4.2 and 4.3 describe our multimedia traffic identification algorithm. Figure 4.1 explains multimedia traffic initial process. Figure 4.2 expresses 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. For the reason that assurance of audio traffic quality is more important than video traffic quality for customer satisfaction, we assign audio traffic a higher priority than video traffic. 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. Otherwise, the packet goes to an end in this algorithm process.

Diagram

Description automatically generated

Figure 4.1: Multimedia packet initial process

Figure 4.2 explains 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 packet inter-arrival time is in this range, this 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. Then the packet process goes to an end in this algorithm function. 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.

Diagram

Description automatically generated

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 interval time within a frame is less than 0.1ms and interval time between frames 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 interval time between frames 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 interval time between frames is equal to or larger than 48ms, R(1) and R(2) are updated with the current packet arrival time, after that the video packet process goes to an end in this function.

Diagram

Description automatically generated

Figure 4.3: Video packet process

In this section we design a general multimedia traffic identification algorithm based on multimedia traffic transmission rules described in section 3. This algorithm can be implemented on network functions such as routers or switches to identify audio and video traffic.

5. ALGORITHM IMPLEMENTATIOIN 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. First, we configure P4 environment. We download and configure P4 virtual machine image from P4 tutorials [16] [17]. Based on the basic example in the tutorials, we create a new network topology by modifying the file topology.json, then we specify IP packet routing rules for each router. The file s1-runtime.json defines IP packet forwarding rules for s1, file s2-runtime.json defines IP packet forwarding rules for s2, and so on. 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. The basic example in tutorials has implemented basic IP packet forwarding function. A packet from a host can be successfully forwarded by routers to its destination host.

Diagram

Description automatically generated

Figure 5.1: P4 network topology

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 testing show that all packet inter-arrival time at routers is larger than 10ms, and about 97% packet interval time is between 10ms and 40ms, 99% packet interval time is between 10ms and 60ms. However, video packet statistic in section 3 indicates that about 85% packet interval time within a frame is less than 5 microseconds, about 99% packet interval time within a frame is less than 100 microseconds. We can see that 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 multimedia sender packet inter-transmit time. Audio packet inter-transmit time is amplified from 20ms to 200ms, and video packet inter-transmit time between frames is amplified from 33ms to 300ms. In our experimental environments, a host simulates a multimedia sender which sends small size packets every 200ms simulating audio traffic, and it sends a group of large size packets every 300ms, a group of packets carrying a frame data are continuously sent out without a break simulating video traffic.

In our experimental scenarios, h1 sends simulated audio or video packets to h4 through s1, s2, s3 and s4. Host h1 continuously sends small size packets at interval 200ms simulated audio traffic, and we specify that the small size packet IP length is 45 bytes. Host h1 continuously sends groups of large size packets at interval 300ms simulating video traffic, and we specify that video packet length is 57 bytes. We also specify that a group of large size packets carrying a video frame data consists of three packets which are sent out continuously. 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 same size packets as audio packet size but with different contents. Otherwise, random packet senders send packets in different sizes. 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, senders send different size random packets to interfere video traffic. We implement our traffic identification algorithm on router s3.

Based on the basic example in P4 tutorials, we implement our traffic identification algorithm 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: control block MyVerifyChecksum(), MyIngress(), MyEgress(), MyComputeChecksum() and MyDeparser(). We add all our algorithm functions 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. In order to store previous packet information, our solution is to use a stateful object register. At the start position of control block MyINgress(), we declare a stateful object register<bit<48>>(3) packet\_register which includes three elements to keep packet time information. Packet\_register(0) saves last audio packet arrival time, packet\_register(1) stores arrival time of the first packet of a frame, and packet\_register(2) saves the last video packet arrival time. Then we add our multimedia traffic identification algorithm in function apply{} in control block MyIngress(). Another issue we encountered is that in P4 systems, all routers process packets using rules defined in file basic.p4 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. For example, we only specify router s3 to implement our traffic identification algorithm. Considering that all packets received at s3 destination addresses are router s3’s MAC address, we propose the solution that when a received packet destination MAC address hdr.ethernet.dstAddr equals to s3 MAC address 0x080000000300, apply our algorithm to process this packet. Otherwise, the packet bypasses our algorithm. By this way, only router s3 processes received packets using our traffic identification algorithm, while other routers ignore our algorithm.

Figure 5.2 depicts packets initial process. Register R(0) stores last audio packet arrival time, R(1) stores arrival time of first packet 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 examines the packet destination MAC address hdr.ethernet.dstAddr. If hdr.ethernet.dstAddr equals to s3 MAC address 0x080000000300, apply our algorithm to process the packet. By this way, other routers omit our algorithm except s3. Then we continue to classify packets according to 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 process bypasses our algorithm because it is neither an audio nor a video packet.

Diagram

Description automatically generated

Figure 5.2: Packet initial process

Figure 5.3 describes audio packet process. If R(0) equals to zero, we infer that this may be the first audio packet, so 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). In our laboratory system, we choose 200ms as low threshold based on the fact that audio packet interval time is larger than 200ms since the sender sends an audio packet every 200ms. We choose 400ms as high threshold, one reason is that after 400ms the next packet will arrive, the other reason is that once the expected packet arrive, last audio packet arrival time R(0) is updated, then the interval time from R(0) to 400ms will be less than 200ms, so other packets arrived between R(0) and 400ms can be successfully excluded. As a result, audio packet identification accuracy is improved. If interval time is between 200ms and 400ms, this packet is treated as an audio packet, assigning AUDIO\_PRIORITY to its Diffserve value. If interval time is larger than 400ms, we infer that the network function misses an audio packet or the packet latency is larger than 200ms, then we 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. Otherwise, all the following packets will be missed to be identified if network latency is significant or an audio packet is missed.

Diagram

Description automatically generated

Figure 5.3 Audio packets process

Figure 5.4 describes video packet process. The function first examines R(1), if it equals to zero, this may be the first video packet, so update R(1) and R(2) with the current packet arrival time. Otherwise calculate packet interval time between frames by using the current packet arrival time minus R(1), and calculate packet interval time within a frame by using current packet arrival time minus last video packet arrival time R(2). In the P4 system, test results show that majority packet inter-arrival time are less than 90ms when packets are continuously sent out from a host. A host continuously sends three packets simulating a group of packets belonging to a picture data. Consequently, if interval time within a frame is less than 90ms, and interval time between frames is less than 120ms, we 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. In our laboratory system, sender sends a group of three packets every 300ms, so we set interval time between frames 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, thus R(1) and R(2) are updated with this packet arrival time, and Diffserve value is set to VIDEO\_PRIORITY. Else if interval time between frames is larger than 600ms, we infer that the system may miss video packets, or the system latency is significant, then we treat this packet as the first video packet by 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 are large.

Diagram

Description automatically generated

Figure 5.4 Video packet process

This section describes our multimedia traffic identification algorithm implemented in P4. As P4 performance is slow, not capable to process real multimedia traffic, we make a host to send small size packets every 200ms simulting audio traffic and sending groups of large size packets every 300ms, each group includes three large size packets sending out continuously simulating video traffic. Host h1 sends simulated audio or video packets to h4, while hosts h5 sends random packets to h7, h71 sends random packets to h6, and h61 sends random packets to h8 to interfere simulated multimedia flows from h1 to h4. Only router s3 employs our algorithm to identify simulated multimedia flows, while other routers bypass our algorithm for better performance.

6. EVALUATION

This section presents our testing results in the system described in seciton 5. The labotory environment is shown in figure 5.1. First we perform four audio traffic identificaiton tests to evaluate our algorithm performance on audio traffic, then we 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 audio interference 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 routing path 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 identifying audio packets from mixed packets, and sets packet Diffserve valude 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 packt Diffserve value equals to AUDIO \_PRIORITY, this packet is flasely 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 the four audio traffic identification test results. In the columns, true positive rate represents the percentage that audio packets are correctly identified belonging to audio flows, and false positive rate represents the percentage that random packets are falsely classified belonging audio flows. In Test1, packets received at h4 indicate that 99.0% audio packets are correctly identified; packets received at h6 indicate that 1.09% random packets are falsely classified as audio packets; received packets at h8 show that 1.0% random packets are falsely classified as audio packets. In Test2, received packets at h4 indicate that 99.0% audio packets are correctly classified; received packets at h6 show that 1.16% random packets are falsely classified belonging to audio flows; packets received at h8 show that 1.75% random packets are falsely identified as audio packets. In Test3, received packets at h4 indicate that 99.3% audio packets are correctly classified; received packets at h6 show that 1.10% random packets are falsely classified as audio packets; received packets at h8 show that 0.39% random packets are falsely classified as audio packets. In Test4, packets received at h4 show that 98.88% audio packets are correctly classified; received packets at h6 indicate that 1.16% random packets are falsely classified to audio flows; received packets at h8 indicate that 1.15% random packets are falsely identified as audio packets. Average 99.05% audio fraffic from h1 to h4 are correctly identified, average 1.13% random packets from h71 to h6 are falsedly identified as audio traffic, and average 1.07% random packets from h61 to h8 are falsedly recognized as audio packets. The false positive rate from h5 to h7 is zero because the traffic does not go through s3. From the testing results, you can see that our algorithm achieves very high accuracy for audio traffic identification.

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | h1 to h4 audio traffic true positive rate | h71 to h6 random audio traffic false positive rate | h61 to h8 random audio traffic false positive rate |
| test1 | 99.0% | 1.09% | 1.0% |
| test2 | 99.0% | 1.16% | 1.75% |
| test3 | 99.3% | 1.10% | 0.39% |
| test4 | 98.88% | 1.16% | 1.15% |
| average | 99.05% | 1.13% | 1.07% |

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 interference packets to h7, h71 sends random audio interference packets to h6, and h61 sends random video interference packets to h8 at random interval time to interfere multimedia traffics from h1 to h4. In test1, h1 sends out 300 simulated audio packdets and 200 groups of simulated video packets to h4, with each group of video packets including 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; and 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 four multimedia traffic identification test results. The testing reults show 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 belonging to video flows, average 2.51% random audio interference packets from h71 to h6 are falsed classified as audio traffic, average 28.22% random video packets from h61 to h8 are falsedly recognized as video traffic.

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | audio traffic true positive rate | video traffic true positive rate | random audio false positive rate | random video false positive rate |
| test1 | 95.0% | 92.83% | 3.45% | 27.84% |
| test2 | 95.0% | 94.44% | 2.82% | 28.6% |
| test3 | 96.47%% | 93.17% | 1.79% | 28.4% |
| test4 | 95.71 | 93.55% | 1.98% | 27.6% |
| average | 95.55% | 93.5% | 2.51% | 28.11% |

In summary, the test results in P4 show that our audio traffic identification algorithm achieves average 99.05% accuracy for audio traffic indentification, and the average false positive rate is 1.1%. For audio and video traffic mixed with random packets, our algorithm achieves average 95.05% acuuracy for audio packets identification, average 93.5% accuracy for video traffic recognization, and the average audio false positive rate is 2.51%, and the average video false positive rate is 28.11%.

7. CONCLUSION AND FUTURE WORK

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 we perform voice calls and video conferences to verify our assumptions. Our tests prove that that in a video conference, audio and video data are seperatedly transmitted. We discover that audio packet lengths are between 100 and 200 bytes, and 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, and a group of packets carrying a video frame data are sent out continuously, and mean vedio frame interval time is 33ms. Then based on the multimedia traffic patern, we design a general audio and video traffic identification algorithm. In order to evaluate our algorithm perforance, we design and implement a multimedia traffic identification algorithm in P4 and perform some tests. The testing results show that our algorithm achieve very high accuracy for audio traffic identification. Our audio traffic identification testing results indicate that average 99.05% audio packets are correctly identified, only 1.1% interference random packets are falsely identified as audio packets. Our 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. As users have high demand for audio traffic quality, 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 so that the audio traffic quality is assured. Compared to current dominant machine learning based traffic classificaiton methods, our traffic identification approach neither require costly pre-labeled datasets nor require complicated machine learning processes such as model function construction and training phases, our algorithm is more efficient as it only requires three extra global variables and three local variables.

The future work may involve implementing our general traffic identification algorithm in a variaty of real network systems, and implementing our algorithm 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 algorithm accuracy.

ACKNOWLEDGEMENTS

The authors are grateful to Raytheon BBN Technologies for the financial support and to Daniel Ellard for his help in traffic collections.

8. REFERENCES

[1] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](https://tools.ietf.org/html/rfc4301), December 2005.

[2] P. Bosshart et al., "P4: Programming protocol-independent packet processors", *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87-95, 2014.

[3] T. T. T. Nguyen and G. Armitage, ''A survey of techniques for Internet traffic classification using machine learning,'' IEEE Communications Surveys & Tutorials, vol. 10, no. 4, pp. 56\_76, 4th Quart., 2008.

[4] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey", *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1988-2014, 2nd Quart. 2018.

[5] Internet Assigned Numbers Authority (IANA). Accessed: Sep. 27, 2017. [Online]. Available: <https://www.iana.org/>

[6] S. Sen, O. Spatscheck, and D. Wang, “Accurate, scalable in network identification of P2P traffic using application signatures,” in WWW2004, New York, NY, USA, May 2004.

[7] A. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” in Proc. Passive and Active Measurement Workshop (PAM2005), Boston, MA, USA, March/April 2005.

[8] G.-L. Sun, Y. Xue, Y. Dong, D. Wang, and C. Li, “A novel hybrid method for effectively classifying encrypted traffic,” in Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM), Dec. 2010, pp. 1–5.

[9] P. Wang, X. Chen, F. Ye and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning", *IEEE Access*, vol. 7, pp. 54024-54033, 2019.

[10]Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson,"RTP: A Transport Protocol for Real-Time Applications", STD 64,[RFC 3550](https://tools.ietf.org/html/rfc3550), July 2003.

[11][H.264] ITU-T Recommendation H.264, "Advanced video coding for

generic audiovisual services", March 2010.

[12] <https://en.wikipedia.org/wiki/Video_compression_picture_types>

[13]<https://help.apple.com/compressor/mac/4.0/en/compressor/usermanual/#chapter=18%26section=5>

[14] The R Project for Statistical Computing.[online] <https://www.r-project.org/>

[15] “P416 Language Specification”, version 1.2.1, 2020. [https://p4.org/p4-spec/docs/P4.16-v1.2.1.html](https://p4.org/p4-spec/docs/P4-16-v1.2.1.html)

[16]”SIGCOMM 2019 P4 Tutorial”. <https://github.com/p4lang/tutorials/tree/sigcomm19>

[17] <https://github.com/p4lang/tutorials>

[18] <https://p4.org/>

[19] <https://scapy.net/>