# An Enhanced Explicit Port Forwarding Solution for Improving Video Delivery over Software-Defined Networks

*Chih-Heng Ke[1], Yu-Wen Lo[2], Yeong-Sheng Chen[3], Hung-Pai Chen[4*]*

*[1] Department of Computer Science and Information Engineering, National Quemoy University, Taiwan*
*[2] Master of Information Technology and Application, National Quemoy University, Taiwan*
*[3] Department of Computer Science, National Taipei University of Education, Taiwan*
*[4] The Graduate School of Arts and Humanities Instruction, National Taiwan University of Arts, Taiwan*
*smallko@gmail.com, s110840703@student.nqu.edu.tw, yschen@tea.ntue.edu.tw, c660610@gmail.com*

## Abstract

In order to improve video delivery, an Enhanced Explicit Port Forwarding (EEPF) solution is proposed. Data are divided into ordinary packets and video packets using a packet identification mechanism. Traditional mechanisms transmit packets through routing table lookup. As the size of the routing table often becomes excessively large, this lookup can be time-consuming. While the EEPF solution directly includes the transmission path in the video packet, which can reduce the transmission time. And by setting header flags/fields in video packets, the transmission can be further optimized. For example, to reduce the amount of duplicate data to be transmitted for broadcast packets are not replicated until they arrive at the destination switch; to avoid packet loss caused by disconnection, the transmission path can be monitored and rerouting can be initiated immediately in the event of disconnection; to increase the chance of successful delivery, packets can be replicated before they are transmitted via a path with high packet loss risk; and to prevent video packets from being discarded, the switch queue status can be monitored and managed. The testing results have shown that the EEPF solution can effectively shorten the time of video packet transmission and improve the quality of transmitted video.

**Keywords:** SDN, P4, Video transmission quality

## 1 Introduction

In recent years, with the advancement of technology and the rise of self-media, many users have adopted live broadcasting/streaming. As a result, the video or image streaming traffic is growing rapidly, often causing network congestion. Due to the limited bandwidth or load capacity of remote servers, users may find those received images distorted or resolution reduced when watching the videos. What's worse, sometimes the entire video cannot be played.

To address this problem, scholars have proposed various solutions. For instance, one solution [1] makes use of software-defined network (SDN) to set the transmission priority of video packets and utilizes the OpenFlow (OF) protocol to set the queue in switch to reserve bandwidth. This method allows Real-time Transport Protocol (RTP) data to be transmitted using reserved bandwidth, thereby avoiding video packet drop due to insufficient bandwidth.

Another solution [2] improves the quality of transmitted videos by predicting bandwidth and measuring buffer occupancy. CERIO [3] arranges priority or adjusts traffic by setting Differentiated Services Code Point (DSCP) values. It places packets waiting to be forwarded by a router into queues of different priorities according to their DSCP values. In [4], in order to guarantee quality of video delivery over SDN, the integration of the learning agent and an adaptive framework is adopted. During periods of congestion, the SDN controller re-adjusts the rules in the forwarding nodes according to the codec-aware network adaption agent. In [5], the authors try to translate application-layer header information into link-layer headers at the edge. Then different types of traffic can be routed differently in the core network to improve the video application quality of experience. But how to map the application layer information into Q-in-Q tag is not well-described. In [6], the authors use the Big Packet Protocol (BPP) to carry layered video. When there is a limited bandwidth, the SDN controller can reduce the packet size eliminating specific chunks without dropping the whole packet to provide minimum guaranteed quality. But the authors don't handle the situation when the transmission link is down.

While effective in addressing the problems to some extent, most methods mentioned above have certain room for improvement. For example, CERIO's method might be used improperly. A packet may disguise itself as a high-priority one to obtain bandwidth by embedding a certain DSCP value. The method [1] reserving bandwidth for RTP data may work ineffectively when encountering a huge amount of data streams (ordinary vs. RTP). What's more, the OF protocol is not flexible in terms of adding or modifying header fields, which is one of the constraints limiting the wide application of this protocol. A new header field may require a new switch processing flow, making the switch design even more difficult. To improve the OF protocol, Nick McKeown, et al. [7] proposed P4 (Programming Protocol-Independent Packet Processors) to provide a programmable language to specify

*Corresponding Author: Hung-Pai Chen; E-mail: c660610@gmail.com
DOI: 10.53106/160792642023032402010

the forwarding behavior and processing logic of a switch, ending the awkward situation that the switches have to be replaced when new protocols emerge.

Based on existing efforts, an Enhanced Explicit Port Forwarding (EEPF) solution for software-defined networks (SDNs) is proposed in this paper to improve the quality of transmitted video by programming switch using the P4 language. A packet identification mechanism is used to divide data into ordinary and video packets. Then, the transmission performance of video packets is further optimized using different features such as broadcasting, rerouting and packet replication to provide better delivery while ordinary packets are forwarded directly. The solution proposed in this paper tries to improve video transmission innovatively in the following three aspects. 1) As RTP video packets are streamed based on the RTSP (Real Time Streaming Protocol) instead of the fixed port number, the proposed P4 programmable switch will replicate the RTSP packet into the controller, which can parse the RTP port number and install new forwarding rules into the P4 switch to differentiate video packets from ordinary ones. 2) EEPF can write the packet's routing information into the packet itself to avoid delays causing by routing table lookup. In addition, multiple header fields are used to further optimize video transmission. For example, for broadcast packet, only one copy needs to be forwarded. Only the egress switch will copy the packet to all nodes according to the label added by the ingress switch, thus reducing repeated transmission of the same data. For data with low safety requirements, the ingress switch can enable a rerouting feature. When a path fails, the controller can tell the middle switch to change route. Or, for paths with interference or often resulting in data loss, the ingress switch can add a label of "replication" to increase the success rate of packet transmission by repeated transmission. 3) Switch can drop packets going into processing queues already under full load. As the P4 programmable switch used in our solution does not support division and floats, a pre-discarding feature is designed to reduce the chance of video packet loss causing by full-load queues.

In the following section, research work related to software-defined network, RTP/RTCP/RTSP protocols (real-time transport protocol/real-time control protocol/real-time streaming protocol) and RTP video packet identification is introduced. Then, the third section describes how to identify the packet type, the design of EEPF solution, and how to use the EEPF solution to implement features, and further explains how the switch can pre-discarding packets based on the queue load. In the fourth section, our solution is tested and the results are analyzed. Eventually, in the fifth section, conclusions are provided.

# 2 Background and Related Work

This section introduces software-defined network, P4, and RTP/RTCP/RTSP protocol.

### 2.1 Software-defined Network

Software-defined network (SDN) [8] is an emerging architecture that uses software to configure and manage networks. An SDN can be divided into three layers from top to bottom: the application layer consisting of services and applications (such as firewall or load balancer, etc.); the controller layer; and the infrastructure layer, which consists of SDN switches controlled by the SDN controller in the second layer. The control layer is like the human brain, and the infrastructure layer is like limbs. Various rules, such as packet forwarding or blacklisting rules, are set in the control layer. The infrastructure layer only needs to operate following these rules. Therefore, the SDN controller can process path querying and routing operations in the traditional network centrally. This new network architecture separates the control plane and forwarding plane to realize centralized control of network devices. In an SDN, the application/service layer and the control layer communicate mainly through a Northbound Interface (NBI), which mainly uses REST APIs currently. And the control layer and the infrastructure layer communicate mainly through a Southbound Interface (SBI), the commonly used protocols of which include OpenFlow [9-10], NetConf and OVSDB. The OpenFlow protocol is first designed by the Stanford's Clean Slate Program [11], which aims to build a new architecture for Internet network.

Compared with traditional networks, the OF protocol allows network administrators to re-set rules for network device operation and maintenance. However, the packet fields and actions supported by the OF protocol failed to keep up with the rapidly changing needs. To improve the OF protocol, Nick McKeown, et al. proposed programming protocol-independent packet processors (P4). P4 [12-13] is a language for programmable data plane, which is designed to support all communication protocols, all platforms and changeable switch rules. The P4 language aims to achieve reconfigurability, protocol independence and target device independence. Reconfigurability refers to the ability to define flexibly a device's packet processing flow and reconfigure a device in a way not disrupting the device's operation. Protocol independence refers to the independence of a switch from protocol syntax and content. This can be achieved because P4 can customize the packet processing logic and controller can program the switch to realize the corresponding protocol processing logic. Target device independence refers to the "irrelevance" of underlying devices' specifications. P4 compiler can compile the general P4 language processing logic into the relevant instructions of the device and write the instructions into the device to complete the configuration of the device.

### 2.2 RTP/RTCP/RTSP

The RTP protocol [14-15] is currently widely used in communication fields such as VoIP, web conferencing, and Internet TV. RTP can be used with UDP or TCP [16] for transmission. At present, UDP is used more commonly. RTP can't avoid packet loss and thereby ensure the quality of service (QoS) when streaming media data. That's why Real-time Transport Control Protocol (RTCP) [17] emerges. The RTCP protocol itself does not transmit data, but provides a means to periodically exchange statistical information (bytes transmitted, how many packets are lost, one-way or two-way network delays, etc.) between the source and the destination devices and improve the transmission rate of RTP based on

such information. Real Time Streaming Protocol (RTSP) [15] is another protocol designed for the communication between the client and server regarding streaming media. It can control the streaming media server by specifying when to start playing a certain video, changing the program broadcasting timepoint, or providing VCR-like commands to control multimedia content, such as stop, pause or resume, fast-forward and rewind.

### 2.3 RTP Video Packet Identification

Due to the rapid growth of multimedia traffic, how to monitor and control it has become one of the most important research topics. Two researchers [18] have proposed a method for identifying RTP packets. As mentioned above, RTP is more often used with UDP. This method focuses on identifying RTP packets transmitted over UDP by detecting five fields. The RTP packet format is shown in Figure 1.

| V | P | X | CC | M | PT | Sequence Number |
|---|---|---|---|---|---|---|
| Timestamp | | | | | | |
| Synchronization Source (SSRC) Identifier | | | | | | |
| Contributing Source (SSRC) Identifier | | | | | | |
| ... | | | | | | |

**Figure 1.** RTP package format

The five criteria are as follows: (1) version 2 (The value in V field should be equal to 2); (2) Direction(n) = Direction(n-1); (3) PT(n) = PT(n-1); (4) SSRC(n) = SSRC(n-1); and (5) TimeStamp(n) ≥ TimeStamp(n-1) && SequenceNum(n) > SequenceNum(n-1). Criterion (1) relates to the first field of an RTP packet, the version number, which is usually defined as 2. Criterion (2) relates to the forwarding direction, which can be identified by comparing the source and destination IP addresses. Criterion (3) relates to the unique RTP session established for each video stream. The same video stream carries the same value of payload type (PT) field, which indicates the encoding method. When the receiver receives a video, it can find the appropriate decoder according to this field. Criterion (4) relates to the synchronization source (SSRC) field which is used to identify the source of an RTP session. The last criterion relates to the time stamp and sequence number fields. The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence. The time stamp field is used to place the incoming audio and video packets in the correct timing order. For example, it can assist in playing out the received packets at the appropriate time when the server re-sends video data in the event of data loss. This method can identify RTP packets but not all of them. As the identification is based on comparison, the first RTP packet without comparable ones may be classified as an ordinary packet.

## 3 Implementation

A system architecture (see Figure 2) consisting of ingress switch, middle switch and egress switch has been proposed. The ingress switch is responsible for checking packet type, adding the EEPF header and optimizing the transmission of video data. The middle switch is responsible for determining whether a packet should be replicated or re-routed during the transmission to improve success rate of transmission. And the egress switch is responsible for removing the EEPF header and determining whether to broadcast the packet or not.

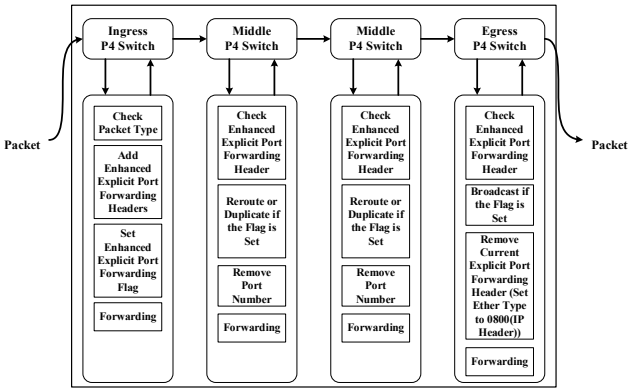


**Figure 2.** P4 switch abstract forwarding model

### 3.1 Packet Type Identification

To improve the transmission of video packet, an identification mechanism must be provided first to screen out these packets. The method [18] described in section 2.3 can't identify the first RTP video packet and may classify it as an ordinary one. The loss of this first video packet during transmission may lead to video playing failure or delay. So, we propose to identify RTP packet based on RTSP. As the port number in RTSP is not fixed, the identification is conducted by analyzing RTSP setup packet. As shown in Figure 3 below, when data is transmitted using RTSP, the client will first communicate with the server, for example, to describe, set up or play. Then, the server will send the video data to the user using RTP. At the same time, the user can control the video using commands such as pause, fast forward, and rewind, etc. In other words, our method sends RTSP packets transmitted over the P4 switch to the controller, which will analyze the RTSP setup packets to obtain the port number used by the RTP protocol, and then write the rules into the P4 switch.

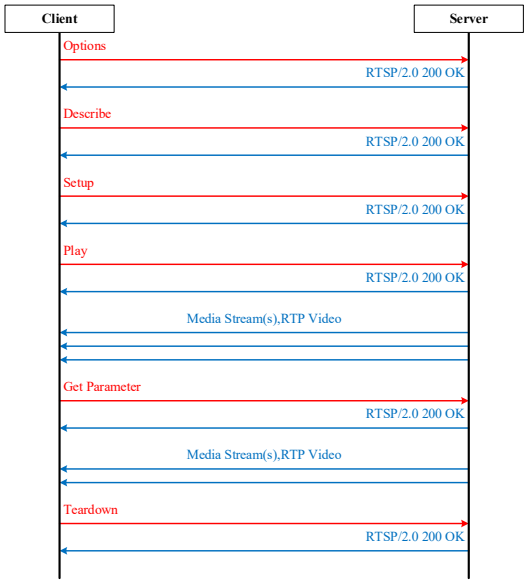


**Figure 3.** RTSP/RTP transmission flow [15]

### 3.2 Enhanced Explicit Port Forwarding (EEPF)

In traditional network architecture, routing tables are built into each switch, and the switch uses the destination IP address in the packet to look up the table. However, the routing table in the switch can become larger and larger with the increase of users and devices, affecting the packet forwarding time. For example, if there is a routing table with more than 8,000 rules in the switch, when the destination IP address is in the middle or at the bottom of the routing table, the packet can be forwarded only after most of the routing table is scanned. And this is the case for a single node. There may be dozens to hundreds of nodes from the server to the client. The time for each node to query the routing table will add up to a large sum. Referencing the concept of source routing [19-20], an Enhanced Explicit Port Forwarding solution is proposed. The transmission path of a packet can be written into the packet, and the intermediate switch only needs to read the transmission path in the packet to forward it. When the packet enters the switch, the switch does not need to look up the information in the routing table piece by piece, but only needs to forward the packet according to the export port number. The format of Enhanced Explicit Port Forwarding header is shown in Figure 4 below. The Bottom of Port Forwarding (BOPF) field is used to identify whether it is the final EEPF header; the Reserved field is reserved for any new features; the DuPliCate (DPC) field is used to confirm whether the packet needs to be copied; the ReRoute (RR) field is used to set whether the packet can be rerouted; the BroadCast (BC) field is used to confirm whether the packet is a broadcast packet; the Video (V) field is used to confirm whether the packet is a video packet, with the value set to 1 for a video packet or set to 0 otherwise; and the last field is about the path storage location.


**Figure 4.** EEPF header format

The full packet format after adding EEPF headers is shown in Figure 5 below. The controller in the ingress switch determines a packet's route. A packet transmitted over n nodes will be added with n EEPF headers, with EEPF header 1 to be processed by the first node and EEPF header n to be processed by the egress switch. Each EEPF header is 2 bytes long. Assume the original packet length is 1400 bytes and 10 hops to traverse from ingress to egress switch. The overhead is only 2*10/1400 = 1.42%. In this paper, we assume that all forwarding nodes are P4-compatible. Therefore, the EEPF headers are inserted between Ethernet header and IP header for speedy packet processing. If some forwarding nodes are not P4-compatible, we can move the EEPF headers between the IP header and Payload. But this will beyond the scope of this paper, we will leave this job in the future work.

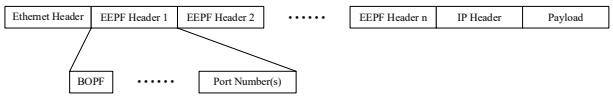


**Figure 5.** Full packet format

An example of designating a packet's transmission path using EEPF is shown in Figure 6. H1 is sending data to user H3. When a packet goes into the P4 ingress switch, an EEPF header will be added into the packet using the P4 language. The number of nodes to go through from this switch to H3 will be calculated, and the transmission path will be written into the EEPF header. When H1 is sending a packet to H3, the controller will judge and determine the shortest path as H1-S1-S2- S4-S5-H3. When the packet enters S1, the output port number of the transmission path, 2, 2, 3, 3 (ports), will be added to the EEPF header. Then the forwarding module of S1 will read the first output port number, forward the packet from port 2 to S2 and remove the first EEPF header. When the packet enters S2, S2 will read the EEPF header and forward the packet from its port 2 to the next node. At the last node, S5, the egress switch, will read the EEPF header and forward it from port 3 to H3. Before forwarding, S5 will remove the EEPF header and change the type value in the Ethernet header to 0800, and then the packet transmission is completed.

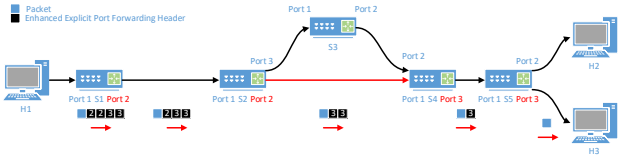


**Figure 6.** An EEPF transmission path example

The EEPF solution makes use of four features to improve the transmission quality of video packets, namely Broadcast feature, Reroute feature, Duplicate feature and Queue Management.

#### 3.2.1 Broadcast Feature

In traditional network architecture, if a server needs to send the data to n clients, the server needs to send n copies of data to each client. Take distance teaching and learning as an example, students in the same classroom need to watch the same stored video in a remote server for learning. This will occupy more transmission bandwidth and place a burden on the server. Using the EEPF solution, the sender only needs to send a copy of data. The ingress switch will identify whether a packet is a broadcast one by matching the destination IP address(es) and set the BC Flag according to the matching result. If the packet is identified as a broadcast one, the ingress switch will set the BC Flag in the EEPF header at the last transmission node to 1. Then the egress switch will copy the packet to all nodes according to the BC Flag value.

#### 3.2.2 Reroute Feature

To improve transmission reliability and stability, the links between nodes are detected to prevent problems such as packet loss caused by link breakage. Using the example in Figure 6, if the link between S2 and S4 breaks, the ingress switch will set RR flag value according to the packet safety requirement. If re-routing is allowed, the RR flag value will be set to 1; or, the value will be set to 0. The RR Flag is mainly used to determine whether a packet can be rerouted. Some data must be delivered in a fixed path due to safety considerations. However, it is recommended to turn on the RR flag for data without this concern, so that the controller

can reroute and change the original S2-S4 transmission path to S2-S3-S4 when the transmission path is found to be interrupted.

### 3.2.3 Duplicate Feature

Data may be transmitted in a wired or wireless way. If a wireless network is used, the packet may be affected and lost due to network interference. Therefore, our solution includes a packet replication feature [21] to increase the probability of successful packet transmission. Using the example in Figure 6, if the path from S2 to S4 carries high risk of packet loss, S1 can set the DPC flag in the second EEPF header to 1. Then, when a packet is forwarded by S2 to S4, S2 will duplicate the packet first and sent the copy one with the original packet to S4, so as to increase the possibility of successful delivery. If S4 received the first forwarded packet successfully, it will discard the second packet received later.

### 3.2.4 Queue Management

If a switch's queue is under full load, packets arriving later will be discarded due to insufficient space. To prevent more important video packets from being discarded, heavier weights are given to video packets. And a fixed chance of discarding (0.1, less than 10%) is set for the video packets, while the drop probability of ordinary packets varies with the queue length. The longer the queue length, the higher the drop probability. The drop probability can be calculated using Formula (1), where $Q_{len}$ is the current queue length, $Q_{max}$ and $Q_{min}$ are two critical values for queue lengths ($Q_{max} > Q_{min}$), and P is the proportion of packets exceeding $Q_{min}$.

$$P = \frac{Q_{len} - Q_{min}}{Q_{max} - Q_{min}}. \quad (1)$$

As P4 does not support division and floats, P in Formula (1) will be multiplied by 100 so that the value of P' is between 0 and 100 as shown in Formula (2).

$$P' = \frac{Q_{len} - Q_{min}}{Q_{max} - Q_{min}} \times 100. \quad (2)$$

Next, we will choose a random number N that the value is between 0 and 100, multiply with ($Q_{max} - Q_{min}$), and compare the result with ($Q_{len} - Q_{min}$)*100. If the conditional expression (3) is true, the packet will be pre-discarding.

$$N*(Q_{max} - Q_{min}) < (Q_{len} - Q_{min}) * 100. \quad (3)$$

The algorithm based on these formulas is shown in Figure 7. The queue status of the switch is detected first to obtain the current queue length ($Q_{len}$), which is then compared with the set minimum threshold of the queue ($Q_{min}$). When $Q_{len}$ is greater than $Q_{min}$, the switch begins to discard packets. When deciding whether to discard or not, the switch operates differently for video and ordinary packets. If a packet is a video packet, a value is obtained randomly from (1, 100), and the packet will be discarded if the obtained value is less than 10. That is, the probability of discarding a video packet is 0.1. And if a packet is an ordinary packet, a value is also obtained randomly from (0, 100), and the packet will be discarded if N*($Q_{max}$ - $Q_{min}$) is less than ($Q_{len}$ - $Q_{min}$)*100.

Algorithm. Procedure for P4 Switches to check switch status and drop packet

```
//set Qmin = 25
//check switch status
1 qdepth.read(Qlen,(bit<32>)1);
2 if (Qlen > Qmin) {
3   random(N,0,100);
4   if (hdr.eepf.v == 1) {  //video packet
5     if ( N < 10) {
6       mark_to_drop(standard_metadata);
7     }
8   } else {  //ordinary packets
9     if ( N * (Qmax-Qmin) < (Qlen – Qmin) * 100) {
10      mark_to_drop(standard_metadata);
11    }
12  }
13 }
```

**Figure 7.** Queue management mechanism

## 4 Performance Evaluation

Our solution has been tested in both Mininet virtual testbed [22] and NetFPGA-SUME card-based actual network environment [23]. As only one NetFPGA-SUME card was available, most features were tested in a Mininet testbed. The NetFPGA-SUME card-based actual network environment was only used to compare the transmission performance of traditional routing table lookup and Enhanced Explicit Port Forwarding solution. The network topology shown in Figure 8 was used as our tests' main network environment framework. H1 is a video sender. When it sends a packet to S1, it will parse the packet and analyze the status of the transmission path first, then select the path using the Enhanced Explicit Port Forwarding solution proposed in this paper, and optimize the video transmission performance based on factors such as path status, packet requirements or switch status.

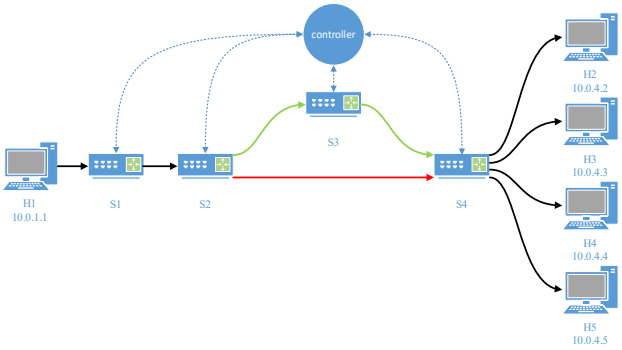


**Figure 8.** Network topology

### 4.1 Packet Type Identification

Figure 9 shows the results of packet type identification. The controller captured and analyzed the RTSP setup packet using tcpdump, tshark and bash shell script, and installed the results into S1, which classified the packet according to

destination IP address and destination port number. We can see the captured information was port 5046. So, a packet with destination IP address as 10.0.4.3 and the destination port number as 5046 was regarded as a video packet, and the V field in the EEPF Header was set to 1. When the packet entered the switch, the packet type could be confirmed and the next action could be taken according to this field value. To verify the packet identification result, we also used Wireshark to capture packets (see Figure 10 below). The IP used by the H3 node was 10.0.4.3. The port number was 5046. The two different methods obtained the same results.

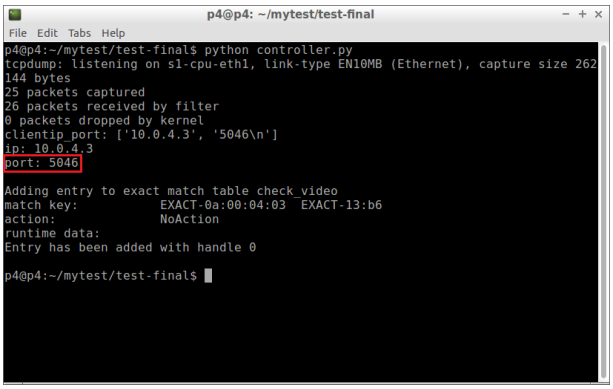


**Figure 9.** Captured information from RTSP setup packet

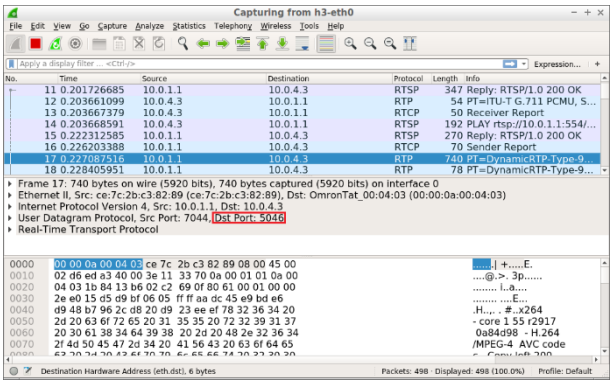


**Figure 10.** Captured packet information using Wireshark

## 4.2 Testing Enhanced Explicit Port Forwarding Solution

### 4.2.1 Broadcast Scenario

In S1 of Figure 8, the destination IP address was analyzed to determine whether the data stream was a broadcast packet. If the data stream was a broadcast packet, the BC Flag was set to 1 and the packet was forwarded to the next node. When the data stream entered the egress switch, the egress switch would copy the data stream and sent it to every user under the node. Therefore, there is no need for H1 to send 4 copies of video data to H2, H3, H4, and H5. Figure 11 shows the results for H1 that only sends one copy after using the broadcast feature. This feature can reduce replicate data over network and the pressure of severs.

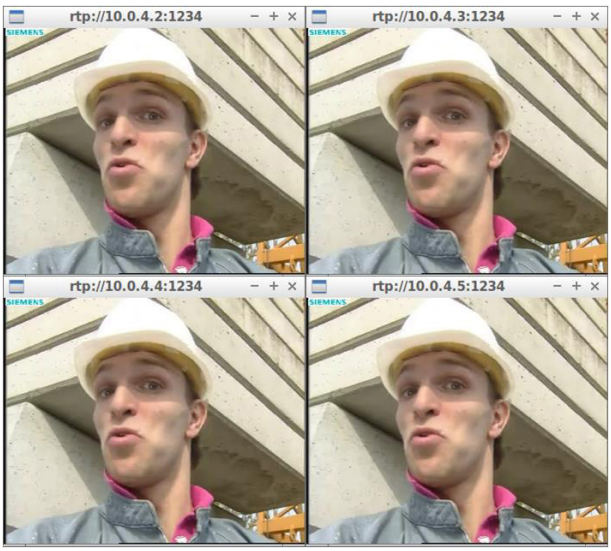


**Figure 11.** Broadcast feature test

### 4.2.2 Re-route Scenario

In this scenario, we want to show that re-route feature is needed for video transmission when the path between source and destination fails. H1 in the Figure 8 transmitted the video data stream to H2. The path between S2 and S4 was disconnected at the fifth second. And the traffic of the H2 network card at this point was analyzed. As shown in Figure 12 below, in the scenario where the rerouting feature was not used (solid line), H2's network card did not receive any packets anymore when the path was disconnected; while in the scenario where the rerouting feature (dashed line) was used, H2 was still receiving packets without interruption. The video packets can be re-routed via S2-S3-S4 to H2. The PSNR (Peak Signal-to-Noise Ratio) [24] value of the original video file was compared with that of the video file received by H2. The PSNR values when using and not using the rerouting feature were 35.9058 and 24.6453, respectively. The latter PSNR value is lower, as the last frame successfully decoded at the time of disconnection remained when the rerouting feature was not used. Tests have shown that this feature can effectively reduce the packet loss caused by the transmission path disconnection.

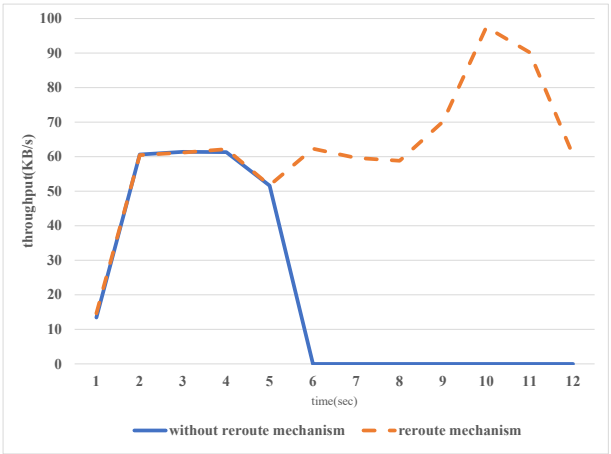


**Figure 12.** Throughput when using and not using the re-routing feature

### 4.2.3 Duplicate Scenario

When there is interference during packet transmission, such as wireless links, the video quality will be degraded. Duplicating the packets before entering the lossy link can help reduce the packet loss rate and then enhance video transmission quality. So, in this scenario, path loss rate between S2-S4 in the figure 8 was set to 1%, 3%, 5%, 7%, and 9%. Then a video data stream was sent from H1 to H2. The packets will be duplicated in S2. Figure 13 shows the measured video PSNR values at different loss rates for a given path. The PSNR value of the original video file was compared with that of the video file received by H2. Test results have proved that packet replication feature (indicated by solid line frame) can effectively reduce packet loss probability in the path with high loss risk.

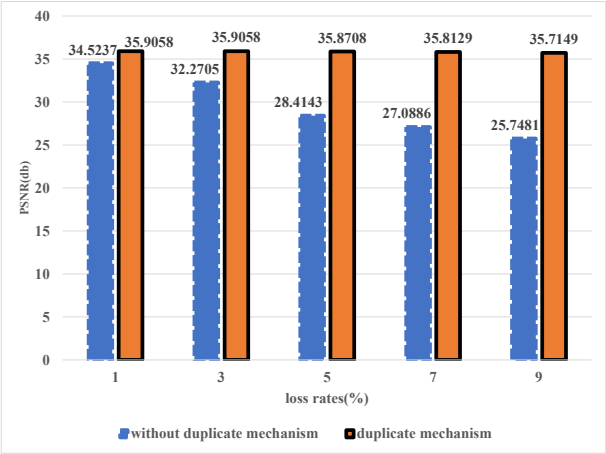


**Figure 13.** PSNR values at different loss rates

### 4.2.4 Queue Management Scenario

When the buffer is full, the forwarding node will discarding all incoming packets including important video packets. Therefore, with active queue management, the forwarding node can pre-discarding ordinary packets and then reserve the resource for the video packets. The delivered video quality can be improved. In this scenario, we implemented the queue management in the P4 switches, sends the video packets and background traffic from H1 to H2, and measure the delivered video quality in H2. Figure 14 shows the PSNR values of video delivered under different traffic conditions. To test whether the feature of monitoring the load level of switch queue can improve the video transmission, Iperf was used to generate background traffic of 11.1Mb/s, 11.3Mb/s, 11.5Mb/s, 11.7Mb/s and 12Mb/s in the transmission path. And the values of queue length and queue rate were set to 100 packets and 1000 packets/second respectively on the P4 switch. Testing results have shown that the PSNR value for video could drop significantly when the queue management is not applied. The switch queue management feature (indicated by solid line frame) can effectively increase the probability of successful packet transmission and reduce the problem of video distortion when network is heavy loaded.

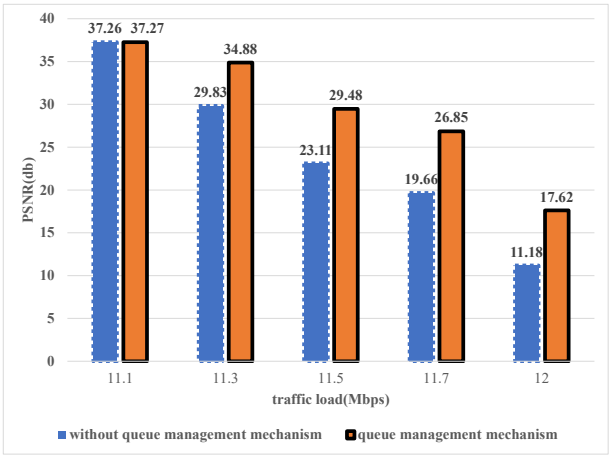


**Figure 14.** PSNR values at different background traffic loads

### 4.2.5 EEPF vs. Traditional Routing

Using NetFPGA-SUME card to simulate a real network environment, the transmission duration of traditional routing and the Enhanced Explicit Port Forwarding solution were compared. Two computers were used. One computer was installed with a NetFPGA-SUME network card and served as a P4 switch, and the second computer served both as the packet sender and receiver. The sender captured the current date & time first and included this information into the packet before sending the packet to the P4 switch, which sent the received packet to the receiver (also the sender). When receiving the packet, the receiver read the recorded sending time and recorded the receiving time. The duration between two recorded times is the time cost to transmit the packet from the sender to the receiver. Five tests were conducted, using 1,000, 2,000, 3,000, 4,000, 8,000 rules and the EEPF solution, respectively. The transmission time when using 1,000, 2,000, 3,000, 4,000, 8,000 rules and the EEPF solution were 4,118,309, 4,121,713, 4,127,314, 4,131,114, 27,913 μs, respectively (see Figure 14). The results have shown that the EEPF mechanism can effectively reduce the packet transmission time.

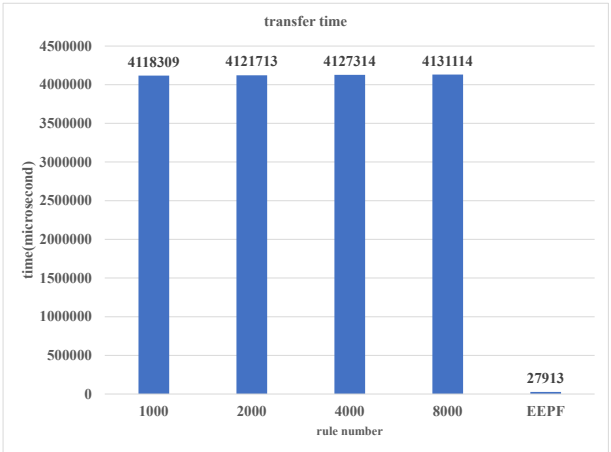


**Figure 15.** Transmission time using traditional routing and the EEPF solution

# 5 Conclusion

The Enhanced Explicit Port Forwarding (EEPF) solution can reduce the time for packet routing processing. By writing the transmission path into the packet header, the P4 switch can analyze and forward the received packet via the set path directly instead of forwarding it to the next node after item-by-item routing table look-up. And the transmission of the video packet can be further optimized by analyzing the packet requirements and detecting the status of the transmission path. For example, a BC field in the EEPF Header can be used to make the egress switch replicate and forward a broadcast packet to all users connected to the node, reducing excess traffic caused by repeated replication. In addition to the above optimization for the transmission path, to prevent the overloaded P4 switch from discarding video packets, switch queue management can be applied int the P4 switch. When the queue length exceeds the set minimum threshold and begins to drop packets, a lower discarding probability can be set for video packets. The tests have shown that these features can effectively improve the performance of network transmission and reduce the occurrence of problems such as video distortion after transmission. Due to insufficient resources, not all mechanisms have been tested in the real network environment simulated using NetFPGA-SUME card. Further research is needed to transfer this solution into the real network environment for performance testing and developing new features or mechanisms.

# References


[1] J. Ohms, O. Gebauer, N. Kotelnikova, D. Wermser, E. Siemens, Providing of QoS-Enabled Flows in SDN (Exemplified by VoIP Traffic), *Proc. of the 5th International Conference on Applied Innovations in IT*, Koethen, Germany, 2017, pp. 25-32.

[2] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, R. K. Sinha, Can accurate predictions improve video streaming in cellular networks? *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, Santa Fe, New Mexico, USA, 2015, pp. 57-62.

[3] CERIO products support Quality of Service (QoS) DSCP PHB/EF/AF advanced setting function, https://reurl.cc/j83MXZ, March, 2010.

[4] O. Izima, R. d. Fréin, A. Malik, Codec-Aware Video Delivery Over SDNs, *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Bordeaux, France, 2021, pp. 732-733.

[5] D. Bhat, J. Anderson, P. Ruth, M. Zink, K. Keahey, *Application-based qos support with p4 and openflow: A demonstration using chameleon*, *Semantic Scholar*, 2018.

[6] S. Clayman, M. Tuker, H. Arasan, M. Sayıt, Managing Video Processing and Delivery using Big Packet Protocol with SDN Controllers, *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, Tokyo, Japan, 2021, pp. 196-200.

[7] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, P4: Programming protocol-independent packet processors, *ACM SIGCOMM Computer Communication Review*, Vol. 44, No. 3 pp. 87-95, July, 2014.

[8] K. Bakshi, Considerations for software defined networking (sdn): Approaches and use cases, *2013 IEEE Aerospace Conference*, Big Sky, MT, USA, 2013, pp. 1-9.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: Enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, pp. 69-74, April, 2008.

[10] OpenFlow Processing Flow, https://ithelp.ithome.com.tw/articles/10220108, October, 2019.

[11] Clean Slate, https://en.wikipedia.org/wiki/Clean_Slate_Program.

[12] P4 Github, https://github.com/p4lang/tutorials, 2016.

[13] P4, https://pc.nanog.org/static/published/meetings/NANOG75/1855/20190219_Bas_P4_Tutorial_v1.pdf, 2019.

[14] RTP, https://ithelp.ithome.com.tw/articles/10205715, October, 2018.

[15] A. Durresi, R. Jain, RTP, RTCP, and RTSP - Internet Protocols for Real-Time Multimedia Communication, in: *The Industrial Information Technology Handbook*, CRC Press, 2005.

[16] C. Perkins, *RTP: Audio and Video for the Internet*, Addison-Wesley Professional, 2003.

[17] Z. Sarker, C. Perkins, V. Singh, M. A. Ramalho, RTP *Control Protocol (RTCP) Feedback for Congestion Control*, RFC 8888, DOI 10.17487/RFC8888, January, 2021.

[18] J. Liang, S. Chen, The design and implementation of RTSP/RTP multimedia traffic identification algorithm, *Journal of Physics: Conference Series*, Vol. 1168, No. 5, Article No. 052033, February, 2019.

[19] S. A. Jyothi, M. Dong, P. B. Godfrey, Towards a flexible data center fabric with source routing, *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, Santa Clara, CA, USA, 2015, pp. 1-8.

[20] S. Li, K. Han, N. Ansari, Q. Bao, D. Hu, J. Liu, S. Yu, Z. Zhu, Improving sdn scalability with protocol-oblivious source routing: A system-level study, *IEEE Transactions on Network and Service Management*, Vol. 15, No. 1, pp. 275-288, March, 2018.

[21] S. Lindner, D. Merling, M. Häberle, M. Menth, P4-protect: 1+1 path protection for p4, *Proceedings of the 3rd P4 Workshop in Europe, EuroP4'20*, Barcelona, Spain, 2020, pp. 21-27.

[22] Mininet, http://mininet.org/.

[23] NetFPGA-SUME, https://github.com/NetFPGA/NetFPGA-SUME-public/wiki.

[24] A. Horé, D. Ziou, Image Quality Metrics: PSNR vs. SSIM, *2010 20th International Conference on Pattern Recognition*, Istanbul, Turkey, 2010, pp. 2366-2369.

## Biographies

**Chih-Heng Ke** received his Ph.D. degrees in electrical engineering from National Cheng-Kung University in 2007. He is a professor at the Department of Computer Science and Information Engineering in National Quemoy University, Kinmen, Taiwan. His current research interests include multimedia communications, wireless networks, software defined networks, and reinforcement learning.

**Yu-Wen Lo** received his MS degree from Master of Information Technology and Application, National Quemoy University, Kinmen, Taiwan, in 2021. His research interests include computer networks and software defined networks.

**Yeong-Sheng Chen** received the Ph.D. degree in electrical engineering from National Taiwan University in 1996. Currently, he is a professor in Department of Computer Science at National Taipei University of Education. His current research interests include wireless sensor networks, wireless communication, software defined network, network security, and context aware computing.

**Hung-Pai Chen** is an Assistant Professor at the Graduate School of Arts and Humanities Instruction, National Taiwan University of Arts. Her research interests include technology-integrated education and arts education. She was a Co-Chair and commissioner of the International Society for Music Education (ISME) Commission on Policy: Culture, Education and Media.