Streaming Scalable Video Sequences with Media-Aware Network Elements Implemented in P4 Programming Language

Chao-Wen Chen[1], Cheng-Hsin Hsu[1]
[1]Department of Computer Science, National Tsing Hua University, Taiwan

***Abstract*—We aim to reduce the negative impact of dropping packets inside the Internet. There are three drop logic are considered. (i) tail, (ii) enhancement-layer, and (iii) rate-distortion.**

***Index Terms*—Scalable video streaming, media-aware network element, software-defined network, rate-distortion optimization**

I. Motivation

In recent years, people are getting used to rely on Over-The-Top (OTT) services such as Skype, Facebook, video streaming, etc. Among these services, video streaming is one of the services which consumes the most network resources. Video streaming needs more and more bandwidth because receivers prefer higher video quality than before, and thus incur high traffic amount on the best-effort Internet. Turns out streaming high quality video with less network resources becomes much more important.

Scalable Video Coding (SVC) is one of solutions for network congestion. Each of the SVC sequences contains a base and multiple enhancement layers. Furthermore, the encoder will encode the discardability into the packetize header. Therefore, We can drop the discardable packets without affecting its decodability in the middle-box of the Internet.The dynamic decisions on which video packets to drop can be sub-optimally done by streaming servers or clients without the global knowledge of the Internet. The better way to approach is through *Media-Aware Network Elements (MANEs)*, which are switches with knowledge of packets header. However, changing the normal switches into MANEs is quite difficult and thus not likely to happen. Fortunately, with recent advances in *Software-Defined Networking (SDN)* and *Network Function Virtualization (NVF)*, network switches are much more programmable, and make collaborative MANEs into reality.

II. Remark Problem

SVC can help us to reduce network congestions but selecting some proper layer to stream isn't a optimal solution. The network condition would change violently in runtime. Select the packets in middle of Internet would be much better than decide them in the beginning. Nevertheless, regular switches in the Internet can't drop any particular packets. The working logic of them is such easy as store and forward all the packets. The packets will be drop without any remedy in UDP protocol which is the most used protocol in video streaming. In SVC sequence, the higher layer can be decode depends on the lower layer. In other words, forwarding the higher enhancement layer without the lower enhancement layer waste the network resource a lot.

To solve this problem, we are going to use *Programming Protocol-Independent Packet Processors (P4) [2]* programming language to design a switch to drop some useless packets in the middle of the Internet. P4 is a new describing language with following features. (i) Protocol Independent, (ii) Target Independent (iii) Field Reconfigurable. Protocol independent allows us to design any protocol we want to forward the packet, and thus we can add some user-specific field in the header for us to make our decision. Target independent allows us to describe everything from high-performance ASICs to software switches. Field reconfigurable allows us to change the way our switches process packets after we deploy our p4 program.

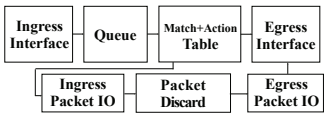

Fig. 1. Packet processing in a P4-based MANE.

Fig. 1 shows how the P4-based MANE process packets. We define some header format and parsers which allow us to understand the structure of the packet. Incoming packets will first be parsed and divide into header and payload. We can design some match+action table in both ingress control program and egress control program to determine how to process the packet. In the ingress program, we can specify some match+action table to apply and may drop the packet if it's not valid or useless. After the packet pass the ingress program, it will be pushed into queues and than processed by the egress program. In P4, we can also calculate the checksum, configure forwarding table, construct some metadata adding to the header. With these tools and the knowledge of the whole Internet, we can drop packets optimally in the middle of the Internet.

On the other hand, P4-based MANE can not detect the condition of the whole Internet. Therefore, we are combining P4-based MANE and SDN controller. Here we select *Open Network Operating System (ONOS)* to be our controller

because it can from a controller plane without too many settings and it supports P4-runtime. P4-runtime is a program for controller to reconfig the P4-based MANEs. In other words, we can change the behavior of the P4-based MANE in runtime through P4-runtime. It works like a RESTful API server, we can simply give some commands to control our P4-based switch.

III. Tentative Solution

To drop scalable video packets to retain streamed video quality when bandwidth is in sufficient and dynamic. We plan to implement the following three drop logics. (i) Tail, (ii) Enhancement Layer (EL), and (iii) Rate-Distortion Optimize (RDO). Tail always drop the last packet while EL drops the enhancement layer packets. The advantage of tail is the simplicity. EL ensure the decodability since we always forward the base layer packet. RDO takes the nature of rates between packet length and distortion into consideration, and than drop the packet with largest rate. Furthermore, we will record the frame number and layer id of the dropped packet. Therefore, we can also drop the other packets with the same frame number and higher layer id. RDO logic aims to minimize the negative impact of dropping packets.

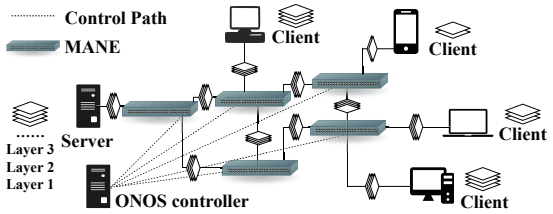

Fig. 2. High-level system architecture with a network of MANEs.

Fig. 2 shows our architecture of the whole system. It is composed of a sender, some clients, some P4-based MANEs, some regular switches, and a ONOS controller connects those MANEs. The ONOS controller forms a control plane disassociate from the data plane. The ONOS controller has the knowledge of the whole Internet. Therefore, it can optimally change the flow among those links. Before the streaming starts, ONOS controller will deploy some flows between each node in the topology. Once the streaming starts, those MANEs inside the topology starts to drop some packets if necessary. During the transmission, ONOS controller will keep monitoring the Internet condition and deploy flows in runtime. It may configure a new flow if one is broken and also notice the sender to let it adjust the sending bitrate if we encounter a network congestion.

IV. Expect Outcome

To evaluate our work, we design the following three scenarios and use real H.264/SVC sequences for streaming. The first two scenarios are simulated in Mininet while the third scenario use real network elements.

- **Intelligent packet drops of a single video stream.** This is the most simple scenario with one sender, one receiver, one MANE, and one ONOS controller simulate in Mininet. We will limit the bandwidth between the nodes and make a network congestion. The MANE should detect the congestion by it's queue size. If the queue size is larger than a administrate-give number. MANE should start to drop some packets using our three logics, tail, EL, and RDO.
- **Optimal packet drops across multiple video streams.** This scenario is shown in Fig. 3. It is composed of multiple sender and receivers streaming multiple video sequences. This is much more difficult because we have to recognize the video sequence of the dropped packets. We evaluate the outcome among different drop logics.
- **Error resilience with real P4 switches.** Fig. 4 shows the last scenario which is composed of real network elements, some Raspberry Pi (running the reference software switch which is called bmv2 [1], a software switch), a Barefoot switch, and a ONOS controller. All the links are fast Ethernet cable. Link d and link e are faster than link f and link g. Therefore, the ONOS controller should add a flow through link d and link e. Then we manually unplug link d and link e, ONOS controller should reroute the streaming sequence through link f and link g so the receiver can keep receiving high-quality videos. When we plug the cable back, the ONOS controller reroutes the video stream over link d and link e.

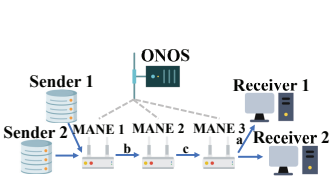

Fig. 3. Mininet testbed topology (scenarios 1 and 2).

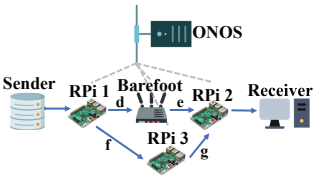

Fig. 4. Real testbed topology (scenario 3).

V. Plan

To approach our goal, we have learned how to write P4 programming language and setup our testbed inside Mininet. The next step is to finish our three drop logics, tail, EL, and RDO in our P4-based MANEs. Then we will run some experiments to make sure we optimally drop the packets. To leverage the advantage of SDN, we connect the P4-based MANEs with the ONOS controller. Next we will set up an server or write a program for developers to command our P4-based MANE in runtime, and we also need to make the controller able to reroute the flows if there's a physical disconnection.

References

[1] Github repository of the second version of the P4 software switch (a.k.a. behavioral model), nicknamed bmv2. https://github.com/p4lang/behavioral-model, January 2018.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, July 2014.