Using SDN To Facilitate Precisely Timed Actions On Real-Time Data Streams

Thomas G. Edwards
FOX Network Engineering & Operations
Los Angeles, CA, USA
thomas.edwards@fox.com

Warren Belkin
Arista Networks
Santa Clara, CA, USA
warren@aristanetworks.com

ABSTRACT

Both legacy networking technology and SDN APIs generally do not allow for highly precise timing of changes in data plane forwarding behavior. This can complicate the network processing of real-time synchronous data streams. For example, in the case of media flows, flow changes may be required to occur on the boundary of atomic elements such as video frames, requiring actions to be timed on the order of microseconds. Most SDN APIs and legacy networking tools do not allow for the changing of forwarding rules with a temporal accuracy of less than tens of milliseconds. To achieve precise timing of actions, the relatively slow and temporally imprecise process of establishing new data plane forwarding behavior can be decoupled from the actual timing of payload-aware flow changes. A proof-of-concept was developed to demonstrate seamless "clean" switching of uncompressed HD video (transported in UDP multicast RTP streams) using an off-the-shelf Ethernet switch and SDN techniques. In this PoC, specialized controller software uses SDN to prepare the network with new data-plane forwarding behavior, and then causes "source-timed" changes at a precise time in the flow's packet headers to match the new forwarding rules. Old forwarding rules that no longer match the new header values are then removed from the network.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network management; C.2.4 [**Distributed Systems**]: Network operating systems; H.5.1 [**Multimedia Information Systems**]: Video

General Terms

Design; Theory; Performance

Keywords

Source-Timed; real-time; media; SDN

HotSDN'14, August 22, 2014, Chicago, Illinois, USA.

http://dx.doi.org/10.1145/2620728.2620740.

1. INTRODUCTION

Software Defined Networking provides a novel architecture for precise control of network flows based on application requirements. However neither legacy network protocols nor SDN APIs can generally allow for highly precise timing of changes in data plane forwarding behavior. Some types of real-time synchronous data streams require highly precise timing of network processing. Packetized uncompressed high-definition video is one of these kinds of flows. The synchronous "clean" switching between video feeds must occur at particular locations within the video raster, and thus at precise times, or else undesirable video artifacts will be produced. This paper introduces the concept of "source-timed" network flow changes, a mechanism to allow for precisely timed flow changes through the separation of the temporally inaccurate installation of new flow rules using SDN techniques, and the precise timing of the flow change based on packet header changes from the flow sources.

2. PACKETIZED UNCOMPRESSED VIDEO

The particular problem that inspired the source-timed flow change concept was the "synchronous" or "clean" switching of packetized uncompressed high definition (HD) video. Video sources emit multicast RTP packets containing the video data. The video emissions of these sources are synchronized by means such as external video genlock or Precision Time Protocol. Video source synchronization implies that all sources are emitting packets from the same location of the video raster at the same time, beginning at the top left of the frame, and proceeding line by line down to the bottom right of the frame.

Uncompressed video can be packetized using the SMPTE 2022-6 protocol [4]. This standard provides for encapsulation of the payloads of a variety of SMPTE serial digital video standards. The standard carries 1376 octets of video payload in each RTP datagram.

A portion of the raster contains video data that is not made visible to the end user, and is known as the Vertical Blanking Interval (VBI). An area on a specific line of the VBI is designated by SMPTE RP 168 [1] as the recommended switching area such that the effects of any signal discontinuity in the processing chain is minimized. To change flows within this region for a 1280x720 progressive image of 59.94 frames per second [2], the required accuracy would need to be within the length of time represented by one SMPTE 2022-6 datagram, or about 7.4 μs.

Unfortunately, reported OpenFlow rule update rates [5][8] of network devices range from 100 to 1000 per second, im-

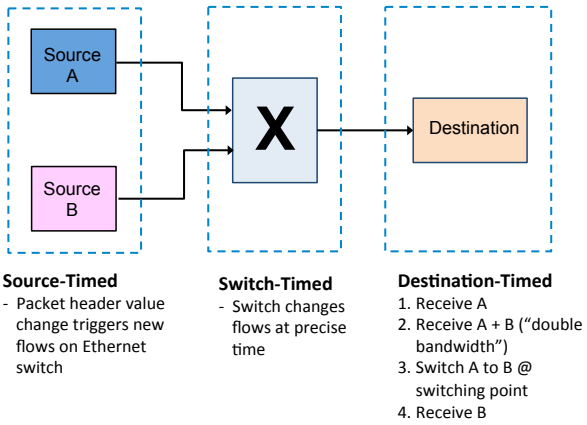


Figure 1: Three Ways to Change Flows

plying an upward limit of temporal precision between 1ms to 10ms. It is possible that data plane rule update rates on common-off-the-shelf (COTS) network devices will become faster over time, but they are not fast enough for reliable synchronous video switching today.

3. APPROACHES FOR SWITCHING REAL-TIME PACKETIZED MEDIA STREAMS

Three potential strategies for switching real-time packetized synchronous video streams are shown in Figure 1.

First, there is the concept of Source-Timed Flow Changes, which is described in detail in section 4. In addition, previous work has attempted to solve the problem of precisely timed stream switching through precise timing of flow changes at the Ethernet switch ("switch-timed"), or by using buffering to allow for flows to be precisely switched at the destination ("destination-timed"). Unfortunately, both of these techniques have significant real world drawbacks.

The switch-timed flow change technique suffers from the fact that today (as mentioned earlier), COTS Ethernet switches generally cannot provide the temporal accuracy of flow changes required for synchronous video switching. However one specialized Ethernet switch that included an integral FPGA subsystem has been used to demonstrate switch-timed synchronous switching of packetized uncompressed HD video [6]. But this solution has a higher cost model compared to non-specialized Ethernet devices.

Destination-timed flow change requires the destination to receive the flow to be switched into before ending the reception of the flow to be switched out of ("make before break"). During that time, the destination buffers data from both flows, and the destination device itself can determine the precise video switch point. This solution can be readily implemented using IGMP [7]. However during the period between joining the new multicast group and leaving the old multicast group, one or more data paths through the network will have to carry twice the bandwidth of the media flow. In particular, the "last mile" Ethernet switch port that the receiving device is directly attached to is likely to be limited to handling half of the media flows that it could otherwise carry in a steady-state environment, representing a significant waste of bandwidth capacity.

4. SOURCE-TIMED FLOW CHANGES

The concept of a source-timed flow change is to separate the temporally inaccurate process of using SDN to update data plane forwarding rules from the actual precise timing of a flow change. To do this, a specific element of the packet header is selected as the "timing signal" match field whose value will be used to trigger a precise flow change by matching previously-configured SDN flow rules in the network. Preferably this header will have little or no impact on other stream processing functions. In our proof-of-concept, the UDP source port value was used.

Figure 2 shows the initial state of an example source-timed flow changing network. There are two flow sources and two flow destinations. A notional "flow routing table" (FRT) shows the relationship between flow sources and destinations. The FRT has a version number (in this case 10001), which will be used to match the timing signal match field of the data plane flow rules. Initially in this network, a flow from Source A is sent to Destination A, and a flow from Source B is sent to Destination B. The flows have the UDP source port 10001, which matches the flow rules on the switch to carry out the routing as directed by the FRT. This example only shows one-to-one flows, but the flows could also be one-to-many.

A request to change video flows is received by the SDN controller in Figure 3. The notional FRT is first updated to reflect flows from Source A to be sent to Destination B, and flows from Source B to be sent to Destination A. This FRT has the new version number 10002. The controller responds by adding new flow table rules to the Ethernet switch to steer packets that have UDP source ports matching the new FRT version number (a process that could take tens of miliseconds). During this time, the sources are still transmitting flows with UDP source port 10001, so the flows are still following the paths as per FRT version 10001.

After enough time has passed for the new SDN rules to have been reliably added to the Ethernet switch forwarding plane, the controller informs the sources of the new FRT version number. This prepares the sources to change the UDP source port of their emitted flows at the precise time of the next SMPTE RP 168 video switching point.

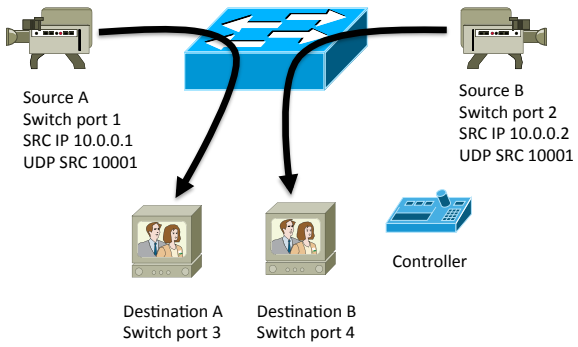
In Figure 4, the sources have changed their UDP source port to 10002 at the precise video switching point, and since the flows now match the new flow rules on the switch, they are steered as directed by FRT version 10002. After this point in time, the old (now unused) flow rules may be removed from the switch, either programmatically or using OpenFlow **idle_timeout**.

5. PROOF-OF-CONCEPT

A proof-of-concept was developed to test and demonstrate the source-timed SDN flow changing technique implementing synchronous video switching.

5.1 Hardware

A schematic diagram of the proof-of-concept hardware is shown in Figure 5. Uncompressed HD video sources were a Blackmagic Design HyperDeck Studio SSD recorder/player and a Panasonic AG-HPX250P HD camera. Both video sources were emitting 720p/59.94 fps video, and they were both synchronized using a signal from a blackburst generator. These output of these video sources was the high defini-



Initial Flow Routing Table (FRT):

FRT #10001	Source A	Source B
Destination A	X	
Destination B		X

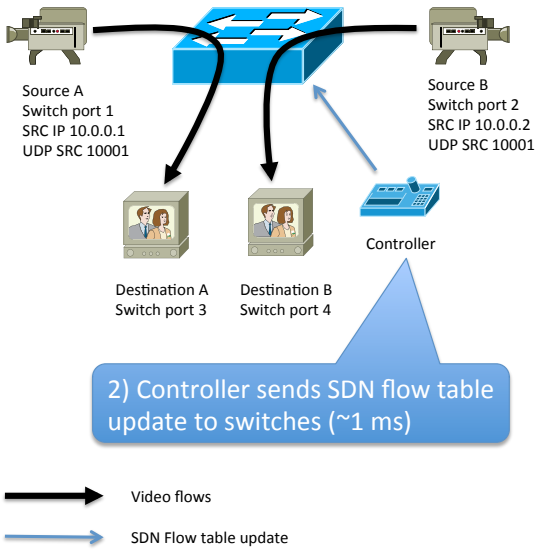
Initial SDN switch flow table:

IP SRC	UDP SRC port	Action
10.0.0.1	10001	Forward Port 3
10.0.0.2	10001	Forward Port 4

UDP SRC port used as "timing header" to signal FRT Version

Video flows

Figure 2: Initial Conditions of Source-Timed Flow Changing



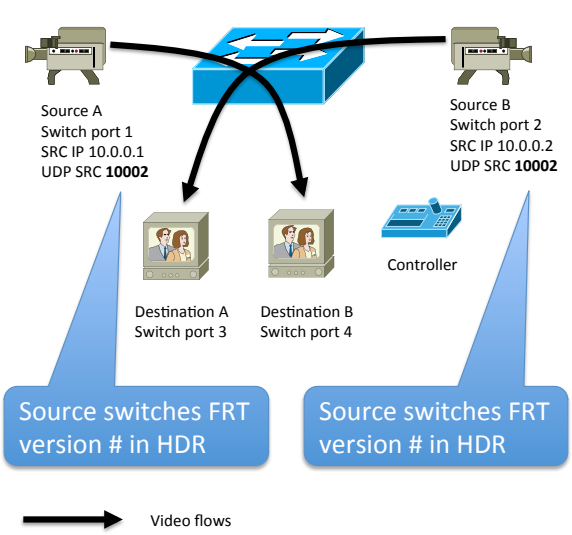
1) FRT Updated & FRT Version incremented:

FRT #**10002**	Source A	Source B
Destination A		**X**
Destination B	**X**	

3) New entries added to SDN switch flow table:

IP SRC	UDP SRC port	Action
10.0.0.1	10001	Forward Port 3
10.0.0.2	10001	Forward Port 4
10.0.0.1	10002	Forward Port 4
10.0.0.2	10002	Forward Port 3

Figure 3: After Receiving a Flow Change Request



Flow Routing Table:

FRT #10002	Source A	Source B
Destination A		X
Destination B	X	

SDN switch flow table:

IP SRC	UDP SRC port	Action
10.0.0.1	10001	Forward Port 3
10.0.0.2	10001	Forward Port 4
10.0.0.1	10002	Forward Port 4
10.0.0.2	10002	Forward Port 3

Figure 4: Sources Change FRT Version

tion serial digital interface (HD-SDI) over 75Ω coaxial cable, as per SMPTE 292-1 [3].

The uncompressed HD-SDI video from the sources was packetized using a Nevion VS902 SMPTE 2022-6 gateway. This device has four HD-SDI inputs, four HD-SDI outputs, and two 10 Gbps SFP+ Ethernet ports. One gateway was used for packetization of both HD-SDI inputs, which allowed RTP and SMPTE 2022-6 HBRM metadata to be consistent between the streams. This gateway also de-packetized one video flow output from a particular port of the Ethernet switch into HD-SDI. A second gateway was used only for a de-packetization of a second video flow output from another port of the Ethernet switch. HD-SDI outputs from the gateways were displayed on professional monitors.

A server was used to host software implementing the source-timed packet header changes as well as SDN orchestration. The server had a SuperMicro X9SPU-F motherboard, 1 Intel Xeon E3-1230 V2 @ 3.30 GHz, and 8GB of RAM. The Ethernet card was a dual Intel 82599EB-based 10 Gbps SFP+ NIC. The NIC was operated using "DNA (Direct NIC Access)" DMA-enabled drivers from Ntop.org. The server ran under Ubuntu 12.04.2 LTS, GNU/Linux 3.2.0-44-generic x86_64. Attached to the server was a USB-connected button box that emulated pressing a space bar on a keyboard to command a switch of the video flows.

The Ethernet switch was an Arista 7050S-52, a 52-port 10 Gbps Ethernet switch with SFP+ interfaces and a 1GbE out-of-band management (control) port. The 7050S-52 is based on the Broadcom Trident+ switching ASIC and runs Arista's EOS (Extensible Operating System).

5.2 Server Software

A controller was written in Python to orchestrate the source-timed flow change operation. When the button box was pushed, the controller contacted a DirectFlow agent on the switch to install new flow rules. After a delay of 100ms, the controller then commanded a source-timed packet header adjustor application (also running on the server) to change the UDP source ports of the video flows at the next SMPTE RP 168 switch point. After another 100ms delay, the controller then contacted the DirectFlow agent on the switch to remove old flow rules.

The source-timed packet header adjustor received incoming SMPTE 2022-6 video flows from the sources on one NIC of the server, and re-broadcasted them out of another NIC with modified UDP source ports. The adjustor was written in C using the Ntop.org PF_RING high-speed packet processing socket library.

5.3 Flow Steering with the DirectFlow Agent

Arista's DirectFlow feature was used to implement custom steering of network flows. DirectFlow can be though of as a "controllerless OpenFlow" running in hybrid mode. The switch performs normal L2 / L3 processing, but custom flow rules can be implemented using OpenFlow semantics. In this case, DirectFlow rules were used to match multicast UDP traffic from a specific source IP address and source UDP port and direct that flow out a specific destination port, while also changing the VLAN.

There are several ways that these DirectFlow rules could be programmatically applied by the orchestration system. In the interests of minimizing the time for flow rules to propagate to the forwarding plane, it was decided to use the Arista EOS Python API to apply DirectFlow rules. A Python script is scheduled through EOS to run when the switch is first powered. The script listens on the 1GbE out-of-band management port for UDP packets. The payload of the UDP packet contains instructions from the orchestration system to the switch. Upon detecting a valid new command packet, the Python script adds or removes the appropriate DirectFlow rules via the EOS Python API.

Figure 6 shows an example of a flow rule that directs UDP datagrams with UDP source port 20001 from IP source 10.10.10.52 to be output on port Ethernet 17 with VLAN ID 2.

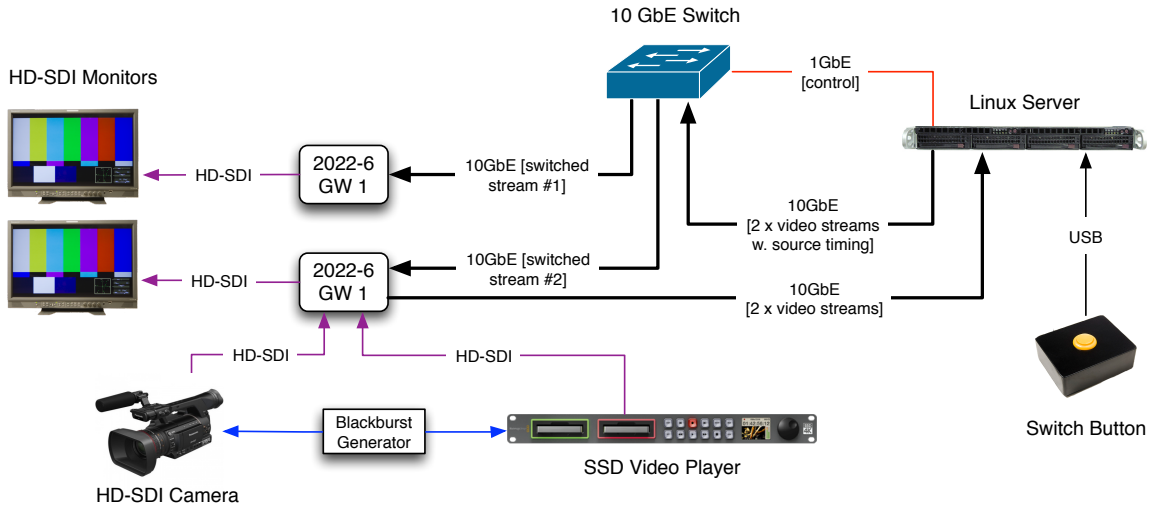


Figure 5: Proof-of-Concept Schematic Diagram

```
Flow DirectFlow_Video_52_to_port_Ethernet17:
   match:
    Ethernet type: IPv4
    source IPv4 address: 10.10.10.52/255.255.255.255
    IPv4 protocol: UDP
    source TCP/UDP port or ICMP type: 20001
  actions:
    set VLAN ID to: 2
    output interfaces: Ethernet17
```

Figure 6: Example DirectFlow Rule used in PoC

5.4 Results

The proof-of-concept delivered a system capable of synchronous switching of real-time uncompressed HD video sources at the SMPTE RP 168 switch point using an off-the-shelf Ethernet switch. PCAP captures of the video switching points were made at the output ports of the Ethernet switch to show a clean switch from one source to another at the proper point in the video raster. In casual testing among users in our lab and at a broadcast industry trade show, no "bad switches" were detected.

In automated "torture testing" of switching as fast as possible for many hours, there were some occasional video glitches in the switching on the order of one out of 10,000 switches. We believe this is due in part to occasional de-synchronization of the SMPTE 2022-6 packetization gateways. The SMPTE 2022-6 standard contains no metadata for de-packetizers to know where in the video raster a particular datagram comes from. The only hint regarding a datagram's location in the raster is that the datagram that ends a frame has its RTP marker bit set. We believe that additional metadata in SMPTE 2022-6 that informs de-packetizers about the position of datagrams in the raster could allow de-packetizers to better handle lost datagrams or those that duplicate data in a particular section of the raster.

6. CONCLUSIONS

Source-timed network flow changes represent a way to achieve high temporal accuracy of flow changes despite the comparatively slow and temporally inaccurate nature of typical SDN APIs. In the particular application of packetized professional video, it shows useful division of labor between general-purpose Ethernet switches that can process billions of packets per second according to slowly changing rules, and video devices from this specialized industry that must be highly temporally accurate. And there may be other types of specialized, synchronous flows that can benefit from the source-timed flow change concept.

7. REFERENCES

[1] Definition of Vertical Interval Switching Point for Synchronous Video Switching. *SMPTE Recommended Practice 168*, 2009.

[2] 1280x720 Progressive Image 4:2:2 and 4:4:4 Sample Structure - Analog and Digital Representation and Analog Interface. *SMPTE Standard 296*, 2012.

[3] 1.5 Gb/s Signal/Data Serial Interface. *SMPTE Standard 292-1*, 2012.

[4] Transport of High Bit Rate Media Signals Over IP Networks (HBRMT). *SMPTE Standard 2022-6*, 2012.

[5] J. Bilberry, M. Palmer, and R. Sullivan. Network Service Security Through Software-Defined Networking. `http://institute.lanl.gov/isti/summer-school/cluster-student-projects/network-service-security-through-software-defined-networking`, 2013.

[6] T. Edwards, A. Bechtolsheim, and W. Belkin. Video Processing in an FPGA-Enabled Ethernet Switch. *SMPTE Motion Imaging Journal*, 123(2):4, March 2014.

[7] M. Laabs. SDI over IP - Seamless Signal Switching in SMPTE 2022-6 and a Novel Multicast Routing Concept. *EBU Technical Review*, Q4 2012.

[8] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee. Devoflow: Cost-Effective Flow Management for High Performance Enterprise Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 1:1–1:6, New York, NY, USA, 2010. ACM.