5G VIRTUOSA

D1.4: Final Ethernet/IP switch system prototype

Version 1.0 Final

30 January 2019

Only for members of the consortium

Grant Agreement No.:	866656
Project Acronym	VIRTUOSA
Project Title	Scalable Software Defined Network Architectures for Cooperative LIVE Media Production exploiting Virtualised Production Resources and 5G Wireless Acquisition
Project Start Date (and Duration)	1 September 2019 (24 months)

Work Package	D1.4: Final Ethernet/IP switch system prototype
Due Delivery Date	31 January 2019
Actual Delivery Date	30 January 2019
Lead Participant for this Deliverable	Nevion AS (NEVION) Mellanox Technologies LTD (MLNX) LOGIC media solutions GmbH (LOGIC) Institut für Rundfunktechnik GmbH (IRT)
Lead Responsible	Yonatan Piasetzky
Dissemination level	Only for members of the consortium
Status	Version 1.0 Final

History of changes		
Version	**Date**	**Change**
0.1	2 January 2020	Initial Draft
1.0	30 January 2020	Final Version

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

1. Executive summary

This document gives a brief explanation and introduction to the important components of the Mellanox SN3700 switch, required to successfully operate the VIRTUOSA IP network.

This document also summarizes the work done by Mellanox in the scope of the VIRTUOSA project, according to the requirements posed by the consortium.

It briefly goes over SN3700 specifications, and then dives into Openflow interface, PTP and the extended features added to them, as well the clean switching idea and implementation.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

2. SN3700 Ethernet Switch

2.1. General Specifications

Mellanox SN3700 200GbE spine/super-spine offers 32 ports of 200 GbE in a 1U form factor.
It is built upon Mellanox Spectrum-2 switching ASIC and carries a total throughput of 12.8Tb/s.
Full specifications:

Specifications	
Switch Model	**SN3700**
Connectors	32 QSFP56 200GbE
Max. 400GbE Ports	---
Max. 200GbE Ports	32
Max. 100GbE Ports	64
Max. 50GbE Ports	128
Max. 40GbE Ports	32
Max. 25GbE Ports	128
Max. 10GbE Ports	128
Max. 1GbE Ports	128
Throughput	12.8Tb/s
Packet Per Second	8.33Bpps
Latency	425ns
CPU	Quad-core x86
System Memory	8GB
SSD Memory	32GB
Packet Buffer	42MB
100/1000Mb/s Mgmt Ports	1
Serial Ports	1 RJ45
USB Ports	1
Hot-Swap Power Supplies	2 (1+1 redundant)
Hot-Swappable Fans	6 (N+1 redundant)
Reversible Airflow Option	Yes
Power Supplies	Frequency: 50-60Hz Input range: 100-264 AC Input current 2.9-4.5A
Size (H x W x D)	1.72" x 16.84" x 22" (44mm x 428mm x 559mm)
Weight	11.1kg (24.5lb)

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

2.2. Mellanox Onyx Operating System

Mellanox Onyx is a high performance, flexible and cloud-scale switch operating system, designed to meet the demands of next-generation data centers.

Onyx leverages capabilities of the SN3000 series to provide greater magnitudes of scale, with up to 512K ACLs, state-of-the-art telemetry, enhanced QoS, exceptional programmability that enables a flexible pipeline supporting both new and legacy protocols, a larger fully-shared buffer, and more.

2.2.1. Mellanox Onyx Feature highlight

Layer 2	Layer 3	Management and Automation
Multi chassis LAG (MLAG), MLAG with STP support	User and management VRFs	ZTP
IGMPv2/v3, Snooping, Querier	IPv4 & IPv6 routing	Ansible, Puppet, SaltStack
VLAN 802.1Q (4K)	BGP, MP-BGP, OSPFv2, route maps	FTP / TFTP / SCP
Q-In-Q	PIM-SSM, PIM-SM	AAA , RADIUS / TACACS+ / LDAP
802.1W Rapid Spanning Tree	BFD	JSON & CLI, Web UI
• BPDU Filter, Root Guard	VRRP, Multi Active Gateway Protocol (MAGP)	SNMP v1,2,3
• Loop Guard, BPDU Guard	DHCPv4/v6 Relay	In-band and OOB management
802.1s Multiple STP	ECMP, 64-way	DHCP, SSHv2, Telnet
Rapid per VLAN STP and PVRST	IGMPv2/v3 Snooping Querier	SYSLOG
802.3ad Link Aggregation (LAG) & LACP	Consistent/Resilient Hashing*	10/100/1000 Mb/s Ethernet RJ45 mng ports
802.1AB Link Layer Discovery Protocol (LLDP)		USB
Store & forward / cut-through mode of work		Console port for management
Head of Queue LifeTime Limit (HLL)		Dual SW image
Jumbo Frames (9216 Bytes)		Events history
Storm Control		Open Network Install Environment (ONIE)

Quality of Service (QoS)	Monitoring & Telemetry	Security
802.3X Flow Control	High Resolution Streaming Telemetry	Storm Control
WRED with Fast ECN	What Just Happened (WJH) Root Cause Analysis	Access Control Lists (ACLs L2-L4 & user defined)
802.1Qbb Priority Flow Control	sFlow	802.1X - Port Based Network Access Control
802.1Qaz ETS	Real time queue depth histograms & thresholds	Strict Security mode for DoD Apps & NIST 800 181A compliance
DCBX – App TLV support	Port mirroring (SPAN & ERSPAN)	Port Isolation
Advanced QoS – Qualification, Rewrite, Policers – 802.1AB	Enhanced Link & Phy Monitoring	
Simplified RoCE Configuration	BER degradation monitor	
	Single command shared buffer management 3rd party integration (Splunk,etc.)	

Synchronization	Network Virtualization	Software Defined Network (SDN)
NTP	VXLAN Hardware VTEP – L2 GW	OpenFlow 1.3:
	Integration with VMware NSX* & OpenStack, etc.	• True hybrid mode with programmable pipeline
		• Supported controllers: ODL, ONOS, FloodLight, RYU, etc.
		• NAT

Docker Container
Full SDK access through the container
Persistent container & shared storage
Container-secured mode of work: limit CPU, memory and SSD usage

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Standards	SNMP MIBs	SNMP MIBs
802.1D Bridging and Spanning Tree	RFC 4001 INET-ADDRESS-MIB	RFC 4292 IP-FORWARD-MIB
802.1p QOS	IANAifType-MIB	RFC 2790 HOST-RESOURCES-MIB
802.1Q VLAN Tagging	RFC 2863 IF-MIB	RFC 1213
802.1w Rapid Spanning Tree	RFC 4318 RSTP-MIB	SNMPV2-CONF
802.1s Multiple Spanning Tree Protocol	LLDP-MIB 802.1AB-2005	RFC 2579 SNMPV2-TC MIB
802.1AB Link Layer Discovery Protocol	RFC 4363 Q-BRIDGE-MIB	RFC 3417 SNMPV2-TM MIB
802.1Qaz ETS	RFC 4188 BRIDGE-MIB	RFC 3826 SNMP-USM-AES-MIB
802.1Qbb PFC	RFC 4133 ENTITY-MIB	Mellanox SMI MIB
802.3ad Link Aggregation with LACP	RFC 3433 ENTITY-SENSOR-MIB	Mellanox IF-VPI-MIB
802.3ba	RFC 4268 ENTITY-STATE-MIB	Mellanox enhanced ENTITY-MIB
802.3x Flow Control	RFC 2572 SNMP-MPD-MIB	Mellanox Power-Cycle-MIB
1000BASE-KX	RFC 4293 IP-MIB	Mellanox SW-Update-MIB
802.3ae 10 Gigabit Ethernet	RFC 4022 TCP-MIB	Mellanox Config-MIB
	RFC 4113 UDP-MIB	

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

3. OpenFlow Interface

3.1. OpenFlow basics

Mellanox Onyx supports OpenFlow 1.3. OpenFlow is a network protocol that facilitates direct communication between network systems via Ethernet. Software Defined Networks (SDN) allows a centralist management of network equipment. OpenFlow allows the SDN controller to manage SDN equipment. The OpenFlow protocol allows communication between the OpenFlow controller and OpenFlow agent.

OpenFlow is useful to manage switches and allow applications running on the OpenFlow controller to have access to the switch's data path and provide functionality such as flow steering, security enhancement, traffic monitoring and more.

The OpenFlow controller communicates with the OpenFlow switch over secured channel using OpenFlow protocol.

An OpenFlow switch contains a flow table which contains flows inserted by the OpenFlow controller. And the OpenFlow switch performs packet lookup and forwarding according to those rules.

OpenFlow switch implementation is based on the hybrid model, allowing the coexistence of an OpenFlow pipeline and a normal pipeline. In this model, a packet is forwarded according to OpenFlow configuration, if such configuration is matched with the packet parameters. Otherwise, the packet is handled by the normal (regular forwarding/routing) pipeline.

Utilizing the built-in capabilities of the hybrid switch/router is the main benefit of the hybrid mode. It increases network performance and efficiency – faster processing of new flows as well as lower load on the controllers. The hybrid switch processes non-OpenFlow data through its local management plane and achieve better efficiency and use of resources, compared to the pure OpenFlow switch.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

3.2. OpenFlow workflow

The OpenFlow (OF) pipeline is deployed in parallel to the usual Mellanox Onyx pipeline.

The ingress port must be deployed in hybrid mode as to serve both the OF and normal Mellanox Onyx pipeline.

The ingress packet which passes the VLAN filter and is a match to the user ACL tables either progresses to the regular Mellanox Onyx flow, or the OpenFlow pipeline depending on the port coupling.

The OpenFlow pipeline is displayed in figure 1.

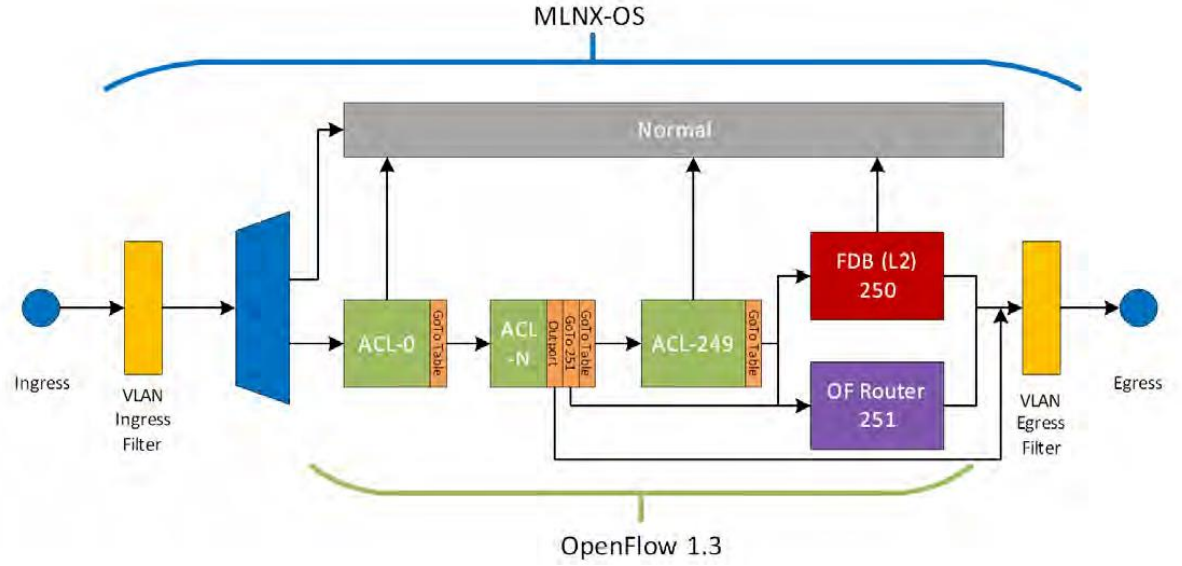


Figure 1 - OpenFlow pipeline in Mellanox Onyx

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Table 1 presents a general summary of the capabilities of the OpenFlow 1.3 pipeline in Mellanox Onyx.

Table	Match	Action	Group	Meters
ACLs [0-249]	• in_port • dl_src • dl_dst • dl_type • vlan_vid • vlan_pcp • ip_src • ip_dst • ipv6_dst • ipv6_src • ip_proto • ip_dscp • ip_ecn • ip_ttl • 14_src_poert • 14_dst_port • tunnel_id • metadata 0xFFF • mpls_label • Table must be configured using "openflow table match-keys" to support the following fields: • ip_src_inner • ip_dst_inner • ignr_eth_type (Dynamic key) (Arbitrary mask)	• Push/pop VLAN • SET_TTL • DEC_TTL • goto_table • Set queue • Eth SRC/DST MAC • VLAN ID • PCP • DSCP • ECN • Output • Group • Meters • Normal	• ALL – Output ports, **Set field** • Select – {weights} Output ports (without LAG) • FF – Output ports	• KBps/PKTs – {Burst} • Drop
FDB [250	• vlan_vid • dl_dst (Exact match)	• OUTPUT • DROP • Normal	Select – {Weights} Output ports (without LAG)	N/A
Router [251]	• ipv4_dst • ipv6_dst (LPM)	• DEC_TTL • SET_DMAC • OUTPUT • DROP (Must have DEC_TTL and SET_DMAC when output action is implemented)	Select – {Weights} output ports + set_dmac + dec_tll	N/A

Table 1 - Summary of OF1.3 Capabilities in Onyx. In red – capabilities added as part of this work

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

3.3. Multicast NAT

As part of the VIRTUOSA project, as stated in the SOW, Mellanox switch will perform Network Address Translation (NAT) of outgoing multicast media flows to route RTP(Real-time Transport Protocol)/ UDP/IP streams to receivers.
This is required, so endpoint configuration of the device will not be required, which will reduce complexity of network infrastructure substantially, while allowing standardized NMOS discovery, registration and control protocols.

As part of this work, in order to support such requirement made by the project, support was added to Set field action in group all ACL tables, as indicated in red in table 1. This allows for multiple multicast streams to egress the switch with each separate destination IP in the packet.

In OpenFlow, in accordance to OpenFlow 1.3 specifications, this is done by creating a group of type all, and adding buckets to this group. Each bucket will have set field actions as well as output port actions.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

4. PTP

Synchronizing network applications require their wall clock time to be aligned precisely with a reference time source (to the order of micro seconds or less). To achieve such accuracy, the application needs the support of networking HW (switch and adapter card), to provide the means to stamp time-sensitive packets. It also requires a time synchronization protocol which would make use of the HW time stamping to adjust its wall clock time to an accurate clock in the network.

4.1. PTP Principles

The basic principle of PTP is as follows:

Slave time = master time + propagation delay + offset.

The purpose of the protocol is to align the slave and the master time so that the gap between them is the propagation delay of the packet. Or in other words, the purpose of the protocol is to use the offset to correct the slave time so the offset between the master sending the packet and the slave receiving the packet is the propagation delay.

Master time is sent periodically by a reliable clock source named Master Clock (MC). In a PTP network, one single reference source is elected called Grand Master Clock (GMC). Propagation delay is calculated between each node and the MC by one of the two methods provided by the standard and further explained below.

To reach sub-microsecond resolutions, all the time stamps which record when a packet is sent and received should be done in the HW. This may impose interaction between SW and HW to query the HW time and send follow-up messages. This issue is further explained below in 2 step section.

Assuming the propagation delay in the network is symmetric, the propagation time is the average time that took the sync and delay req messages to be switched.

Propagation delay = (T4-T1-(T3-T2))/2=(T4-T1+T2-T3)/2

T1 represents the time that the Sync packet left the master which is the master time.
T4 represents the time the Sync packet arrived at the slave.
T2 represents the time the Delay_Req message left the slave.
T3 represents the time the Delay_Req message arrive at the master.

Figure 2 provides an example of the stages required by a slave clock to align its time to the master clock:

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

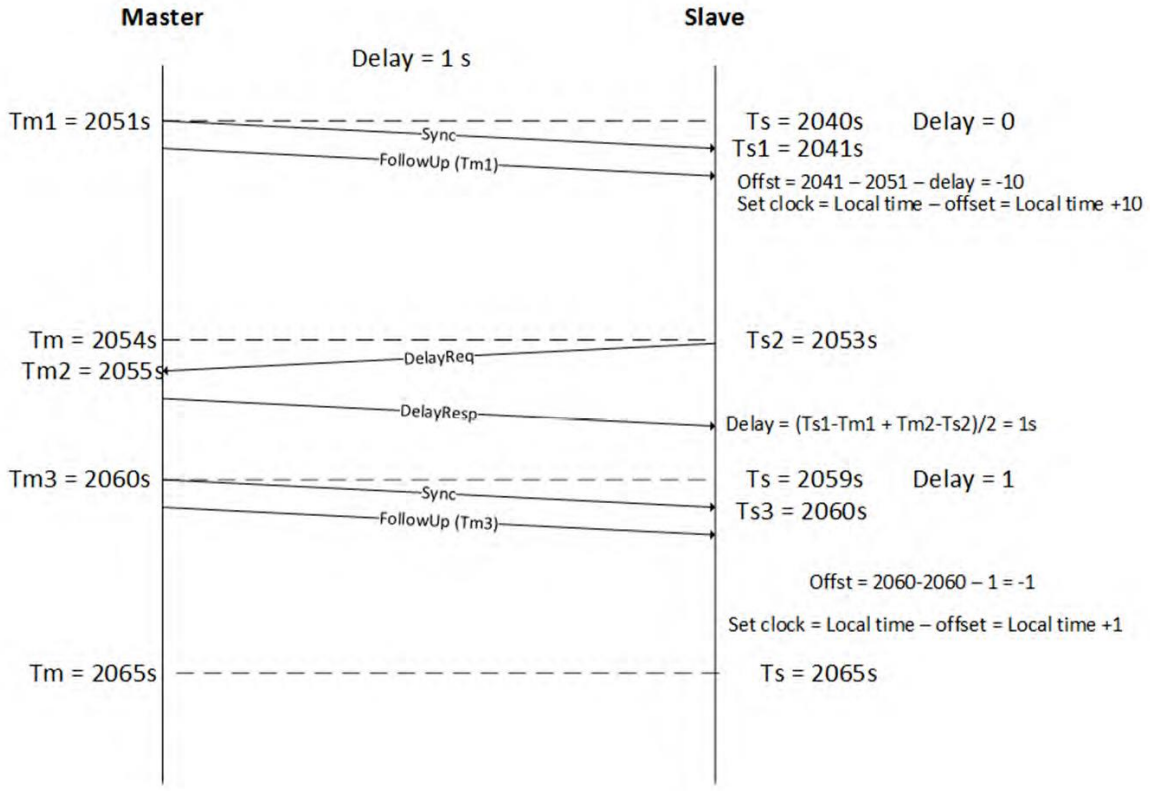


Figure 2 - PTP synchronization process

Table 2 presents the PTP message formats:

Message Type	Hex Value	Class
Sync	0	Event
Follow-up	8	General
Delay_Req	1	Event
Delay_Resp	9	General
Pdelay_Req	2	Event
Pdelay_Resp	3	Event
Pdelay_Resp follow-up	A	General
Announce	B	General
Signaling	C	General
Management	D	General

Table 2 - PTP message formats

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

4.2. Clock Types and Operation Modes

The types of clocks available are as follows:

• Grand Master Clock (GMC) – the reference time source derived from an accurate clock such as a GNSS driven clock (i.e. GPS, GLONASS, GALILEO)

• Boundary Clock (BC) – a network device that acts as slave to its master and as master to its slaves. (Mellanox Onyx implements only this).

• Ordinary Clock (OC) – a clock that operates either as a Master or a Slave. In the case of a slave, the endpoint whose clock is been synced (normally a host/server).

• Master Clock (MC) – a clock which operates as a Master and derives its timing capabilities from the clock chain up to the GMC. It typically serves as a port on a BC connected to a host running as a slave.

• Transparent Clock (TC) – a PTP aware switch capable of measuring the PTP packet switching delay (transient time) and updating the data in the packet. In peer-to-peer (P2P) delay calculation mechanism, a TC device is also required to calculate its delay from the next hop toward the MC and add the value to the switching delay.

Two modes of delay calculations are defined:

• End-to-End (E2E) – each slave calculates its delay from the MC by running Delay request/ delay response sequence (Mellanox Onyx implements only this).

• Peer-to-Peer – propagation delay (Pdelay) is calculated periodically on each link between the slave and the MC independently. The time synchronization packet sent from the MC to all the slaves in the network is updated by each of the downstream nodes with both switching delay (the time that the packet traversed the switch) and upstream hop Pdelay.

4.3. PTP Domains

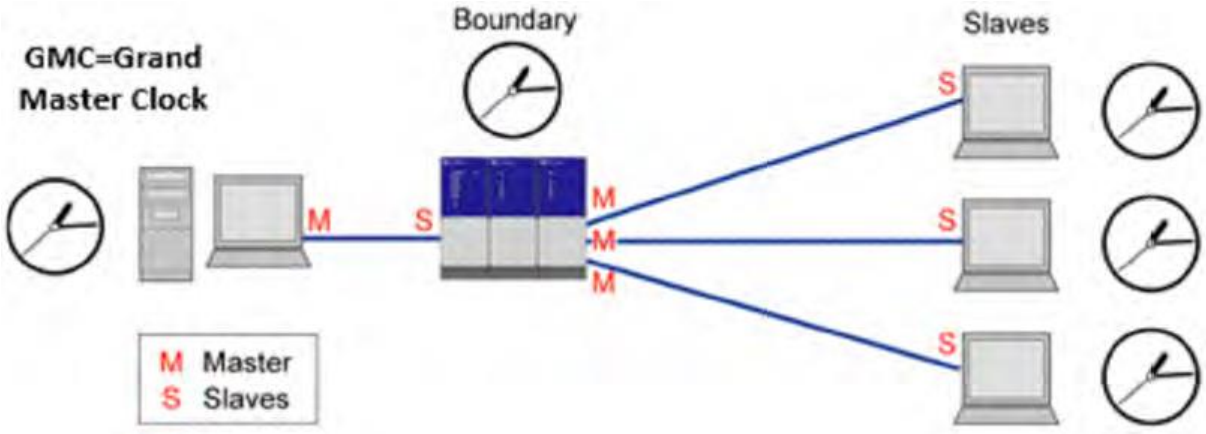
A domain consists of one or more PTP devices communicating with each other. PTP domain defines the scope of PTP message communication, state, operations, data sets, and timescale.

4.3.1. Boundary Clock

In a full E2E PTP deployment, the GMC needs to respond to each slave's delay request message. A normal profile of PTP may require a few delay calculations per second. An average GMC can address few thousands of messages per second. This imposes that direct slave/GMC communication limits the number of overall OCs to ~8K. To scale beyond that, there is a need for a hierarchy between the GMC and the slave. This is achieved by implementing BC, either in the TOR switches or on all the switches in the DC.

The following figure shows the master/slave role that a boundary clock implements between the MC and the Slave (OC).

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.



Each BC acts as a slave towards the GMC and as GMC to its local slaves. Although adding a BC device introduces accuracy degradation as explained above, it becomes mandatory when the number of slaves on a single MC exceeds few thousand devices.

Another use of BC is to bridge between networks. When running PTP over native Ethernet packets, to create larger PTP domains, there is a need to bridge between the broadcast domains. This is done by BC switches.

Default PTP Profile Attributes (SMPTE 2059-2) is shown in Table 3.

Name	Range	Default
Announce interval	-3 (0.125s), 1 (2s)	-2 (0.25s)
Announce timeout interval	2, 10	3
Sync interval (logSyncInt)	-7, -1	-3
Delay request interval	logSyncInt, logSyncInt +5	logSyncInt
PTP domain	0, 127	127
Priority 1	0, 255	128
Priority 2	0, 255	128

Table 3 - Default PTP profile attributes

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

4.4. PTP in VIRTUOSA

According to VIRTUOSA SoW, timing and sync distribution will be according to PTPv2 using SMPTE ST 2059-2 PTP profile. The common timing reference for the PTP grandmaster in the system is a GNSS endpoint, which will be connected to one of the Mellanox leaf switches and distribute PTP time.

To support this requirement, under the scope of this work, support was added for the Boundary Clock mode of operation in SN3700 switches, with required support of up to 48 PTP clients per port.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

5. Clean switching and salvo

5.1. Problem statement

When switching endpoints during broadcast production, the streams need to switch in a "clean" and synchronized manner (i.e., different streams should switch together, on the boundary of a frame).

5.2. Previous solution

Up until this work, the way to implement such requirement, is by having the endpoint subscribe and buffer both flows, switch between them and then unsubscribing from the older one. This creates redundant network and memory usage, as displayed in figure 3. Another problem with this solution is that it requires separate API calls to join/leave each stream, i.e., the switching of different flows does not occur on the same time, thus creating timing constraints on the API calls, according to the buffer size and amount of streams switched simultaneously.

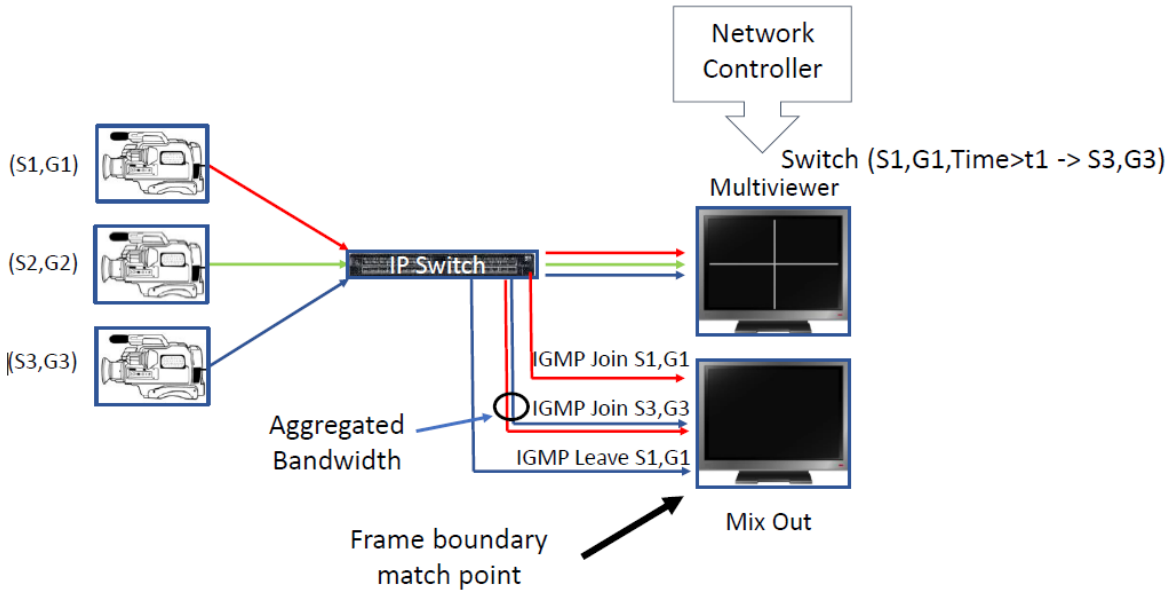


Figure 3 - IGMP based previous solution for switching media streams

5.3. Current solution

Current solution as provided in the scope of this work, comes to solve the issues defined in the above section, and it is divided into 2 – clean switching and salvo.

5.3.1. Clean switching

Clean switching refers to the capability of the switch to perform the switch of the stream routing on the edge of the frame, thus removing the need of the endpoint to accumulate both streams while performing the switch. In VIRTUOSA this capability is implemented by adding an ability to match on an RTP timestamp range and forward accordingly (RTP packets from the same frame all share the same timestamp).

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

5.3.2. Salvo

Salvo is a term used to describe the ability to switch several flows simultaneously in an atomic fashion. i.e., all flows switch in the same time, removing the need to compensate for the lag between all flows subscribe/unsubscribe and thus removing major timing/memory constraints that were induced before.

5.3.3. Implementation

Current solution as explained above, is implemented in Mellanox SN3700 switch, using a P4 program written for this end, and compiled using internal Mellanox compiler. The program adds the ability to match on an RTP timestamp range, to mark a switch event. Then, it is possible to match on the switch event, in order to simultaneously switch all flows registered to this event.

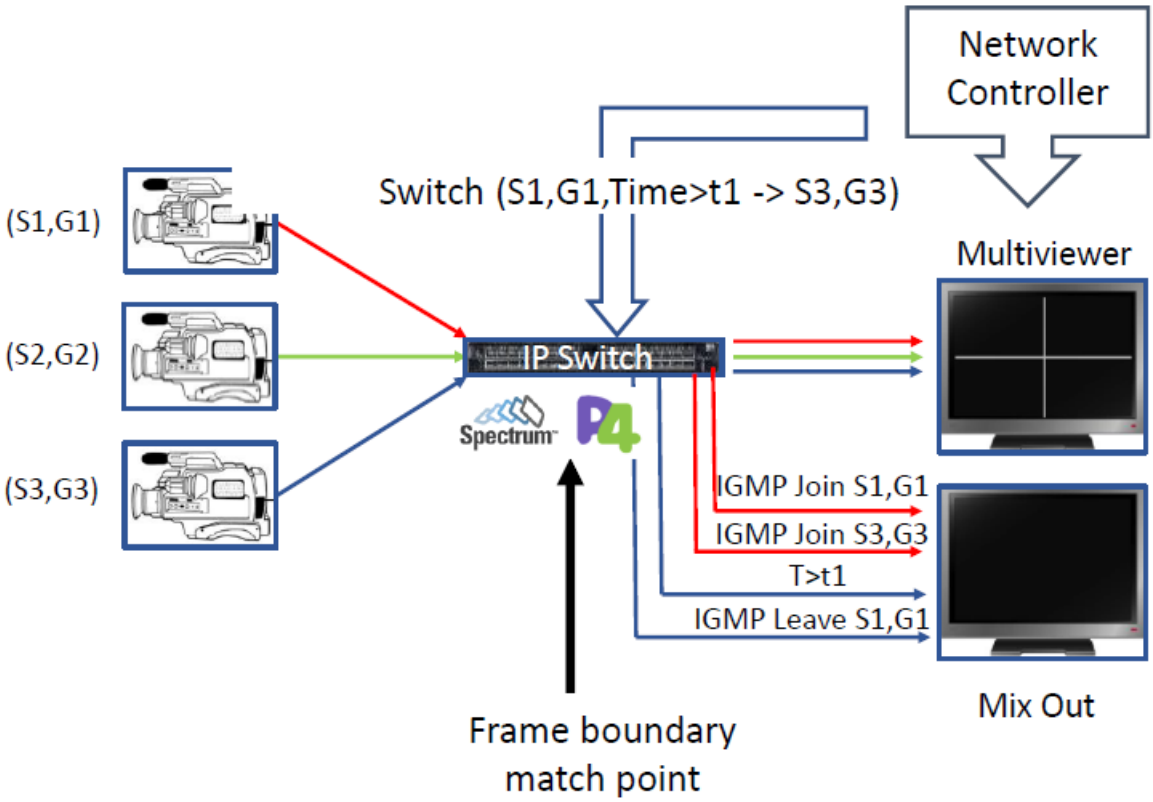


Figure 4 - Flow switch after time t1, using new P4 based clean switching and salvo solution. This type of solution removes the aggregated bandwidth and time constraints that were induced by the old IGMP based solution

5.4. Limitations

Current implementation of clean switching in Spectrum-2 switches, support up to 12 different switching events parallelly (i.e., 12 different time ranges can be defined).

Parsing of the RTP header will occur based on the UDP destination port, and up to 16 different UDP ports are supported simultaneously.

5.5. OpenFlow Interface

Clean switching is exposed to the OpenFlow agent, enabling VideoIPath controller to be able to program the switch. The Interface exposed extends the OpenFlow Extensible Match

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

format (OXM) as defined in the specifications. For the purpose of this work a new OXM TLV was defined, of class Experimenter. The exact experimenter ID and oxm_field are yet TBD.

The RTP timestamp TLV is special in the sense that its mask is a value used to express higher boundary of an integer range, thus an unmasked TLV will mean matching from timestamp given in the OXM value indefinitely, while a masked TLV will mean matching the RTP timestamp in the values between the value and the mask.
Figure 5 shows an example of a Flow Match Field structure containing RTP timestamp OXM, matching from timestamp 5 until timestamp 10.

31 16	15 9	8	7 0
oxm_class = Experimenter (0xffff)	oxm_field = TBD	HM=1	oxm_length
Experimenter ID = TBD			
RTP timestamp value = 0x5			
RTP timestamp value = 0xa			

Figure 5 - RTP timestamp OXM packet header

Currently, RTP timestamp match is only supported as a single match field, and the only supported instruction is Write-Metadata.

Also, a new, dedicated table_id was added that supports RTP timestamp range match as explained above, which is table_id=252. The new datapath flow is shown in figure 6.

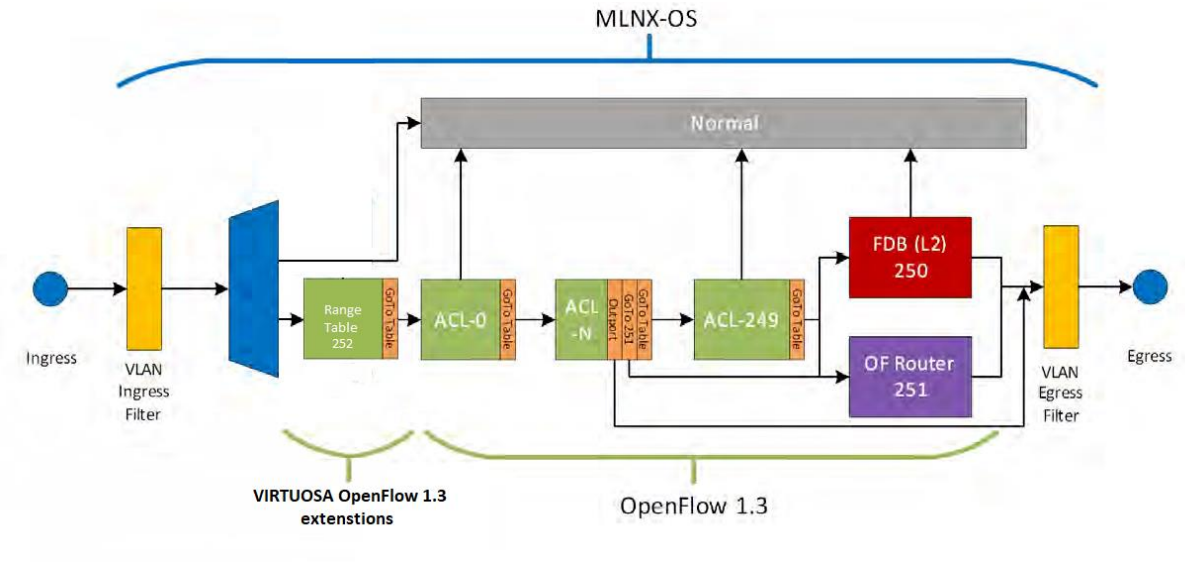


Figure 6 - New OpenFlow datapath which supports clean switching

All other parts of the solution use standard OpenFlow1.3 implementation and will not need any changes.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

5.6. Usage example

Let's say for example we have two flows RTP flows configured in our OpenFlow switch:
priority=10,ip,nw_dst=233.1.1.1->Output port 1
priority=10,ip,nw_dst=233.1.1.2->Output port 2

In order to switch between those flows in a clean fashion (on the edge of the frame), one needs to first configure two additional flows, matching on an unused metadata bit:

Priority=100,ip,nw_dst=233.1.1.1,metadata=0x1/0x1->Output port 2
priority=100,ip,nw_dst=233.1.1.2,metadata=0x1/0x1->Output port 1

Lastly, one needs to configure an RTP timestamp match that will write this metadata and trigger the event:

Table_id=252,Rtp_ts_rng=1234,actions=write_metadata=0x1/0x1

Thus, when packets with timestamp larger than 1234 will arrive on the switch they will switch routing together to their new destination

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 866656. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.