Grant Agreement No. 619572

COSIGN

Combining Optics and SDN In next Generation data centre Networks

Programme: Information and Communication Technologies

Funding scheme: Collaborative Project – Large-Scale Integrating Project

Deliverable D2.1

Integration of OpenFlow SW Interface with POLATIS System and Venture 4x4 OXS

Due date of deliverable: January 15, 2015
Actual submission date: January 16, 2015
Resubmission date: August 28, 2015

Start date of project: January 1, 2014 Duration: 36 months

Lead contractor for this deliverable: POLATIS

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Executive Summary

This document surveys the progress made in WP2 relating to the integration of an OpenFlow software interface to the two optical switching technologies included in the project. The optical switches constitute part of the novel data plane architecture included in the COSIGN project. Both switching technologies represent complementary solutions to the problem of optical interconnects in large scale data centre networks.

The Polatis switch represents a low-loss (< 2 dB) and high-port-count (> 200 ports) optical interconnect with switching time in the millisecond range.

The Venture switch represents an ultrafast (~5 ns) optical switch with a low port count (initially 4x4) and slightly higher losses (< 7 dB for first samples, but improving to 0 dB by end of project). Together they make up the building blocks for the COSIGN photonic data plane switch fabric.

OpenFlow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network. As such it builds upon a centralized controller to manage and control the entire network insuring optimal use of resources and improved throughput and latency under certain traffic patterns. In the COSIGN project OpenFlow has been designated as the software agent to be used to enable an SDN layer in the data plane. The different switches (including also the Top of Rack (TOR) switches) are all required to support the OpenFlow protocol.

In this report we summarize the progress made in the process towards the integration of OpenFlow in the optical switches.

The report is divided into two sections, describing successful integration of OpenFlow with the Polatis switches and Venture switches respectively, and demonstrating the first use of optical circuit switching under the OpenDaylight SDN controller.

There is a Confidential Appendix to this document which provides additional technical information not for public disclosure at this time.

Document Information

Status and Version:	2.0	
Date of Issue:	28 August 2015	
Dissemination level:	Public	
Author(s):	**Name**	**Partner**
	Harm J.S Dorren	TUe
	Oded Raz	TUe
	Tim Durrant	Venture Photonics
	Nick Parsons	Polatis
	Valerija Kamchevska	DTU
	Michael Galili	DTU
	Bingli Guo	UNIVBRIS
	Xi Chen	UNIVBRIS
M12 Version:		
Edited by:	Oded Raz	TUe
Checked by:	Sarah Ruepp	DTU
M20 Version:		
Edited by:	Tim Durrant	VENTURE
Reviewed by:	Oded Raz	TUe
	Michael Galili	DTU
Checked by:	Helene Udsen	DTU

Table of Contents

Executive Summary **2**

Table of Contents **4**

1 Introduction **5**

1.1 Reference Material 5
1.1.1 Reference Documents 5
1.1.2 Acronyms and Abbreviations 5

1.2 Document History 6

2 OpenFlow Integration **7**

2.1 Integration of OpenFlow and Polatis Switch 7
2.1.1 Background 7
2.1.2 Progress 7
2.1.3 Plans 9

2.2 Integration of OpenFlow and Venture Switch 9
2.2.1 Background 9
2.2.2 Progress 11
2.2.3 Future Plans 15

3 Annex 1: OpenFlow 1.0+ Optical Circuit Switch Extensions **16**

3.1 Connect Input to Output Port 16

3.2 Read Power at Given Port 16

3.3 Trigger Alarm for Power below Threshold 16

4 Annex 2: OpenFlow 1.0+ Optical TDM Connection Extension **17**

1 Introduction

1.1 Reference Material

1.1.1 Reference Documents

[1]	COSIGN FP7 Collaborative Project Grant Agreement Annex I - "Description of Work"
[2]	S. Das, “Extensions to the OF Protocol in support of Circuit Switching”, addendum v0.3, June 2010
[3]	S. Das, G. Parulkar, N. McKeown, P. Singh, D. Getachew, and L. Ong, “Packet and circuit network convergence with OpenFlow,” in Proc. OFC/NFOEC 2010, paper OTuG1
[4]	M. Channegowda, P. Kostecki, N. Efstathiou, S. Azodolmolky, R. Nejabati, P. Kaczmarek, A. Autenrieth, J.P. Elbers and D. Simeonidou, “Experimental Evaluation of Extended OpenFlow Deployment for High-Performance Optical Networks,” in Proc.ECOC 2012, Tu.1.D.2
[5]	M. Channegowda, http://www.youtube.com/watch?v=wZTMGRfIKks
[6]	M. Channegowda, http://youtu.be/hhHMJ1i6XiQ?list=UUFt-mPr81Z6oVAX_ppIqAmw
[7]	Y. Yoshida et al,” First International SDN-based Network Orchestration of Variable Capacity OPS Over Programmable Flexi-Grid EON”, in Proc. OFC2014, PDP ThA.2
[8]	S Yan et al., “First Demonstration of All-Optical Programmable SDM/TDM Intra Data Centre and WDM Inter-DCN Communication”, in Proc ECOC2014, PD 1.2
[9]	M Channegowda, B Guo, https://www.youtube.com/watch?v=rwvezPUwSK0&feature=youtu.be
[10]	Xi Chen, James Regan, Tim Durrant, Yi Shu, George Saridis, Georgios Zervas, Dimitra Simeonidou, Valerija Kamchevska, Anna Manolova Fagertun and Siyuan Yu, “Monolithic InP-based fast optical switch module for optical networks of the future”, in 2015 International Conference on Photonics in Switching (PS).
[11]	COSIGN Deliverable D1.4 “Architecture Design”
[12]	http://www.ict-lightness.eu/

1.1.2 Acronyms and Abbreviations

Most frequently used acronyms in the Deliverable are listed below. Additional acronyms can be specified and used throughout the text.

CFP2	http://www.cfp-msa.org/Documents/CFP-MSA_CFP2_HW-Spec-rev03.pdf
DC	Data Centre
DCN	Data Centre Network
ECOC	European Conference on Optical Communication
FPGA	Field Programmable Gate Array
LUT	Look Up Table
NETCONF	Network Configuration
ODL	OpenDaylight
OF	OpenFlow
OXS	Optical Crosspoint Switch
POX	Open-source development platform for Python-based SDN Controllers
REST	Representational State Transfer
SDN	Software Defined Networking
TDM	Time Division Multiplexing
ToR	Top of the Rack

1.2 Document History

Version	Date	Authors	Comment
00	08/12/2014	See the list of authors	TOC first draft
1.0	16/01/2015		Final version covering Polatis Switch
1.7	27/08/ 2015		Updated version additionally covering Venture Switch
2.0	28/08/2015		Final version for resubmission to EC

2 OpenFlow Integration

Subsections 2.1and 2.2 describe the OpenFlow integration with the two different switches from Polatis and VENTURE. Annex 1 in Section 3 relates to the Polatis switch, while Annex 2 in Section 4 adds further documentation for the VENTURE switch.

2.1 Integration of OpenFlow and Polatis Switch

2.1.1 Background

Polatis and the High Performance Networks Group at UNIVBRIS have developed a strong collaboration on SDN applications of optical circuit switching since early 2012. Although the OpenFlow 1.0 protocol was originally designed to support packet switching, extensions for circuit switching have been proposed [2],[3] and further enhanced by UNIVBRIS and others [4].

At the ECOC 2013 exhibition in London, Polatis and UNIVBRIS created a live demonstration of an OpenFlow-controlled hybrid packet-optical circuit switched data centre network with automatic detection and routing of persistent (elephant) flows between servers from a packet-switched connection to a direct optical circuit overlay, as shown in Figure 1. An embedded OpenFlow agent (POLOA) was installed on the network interface cards of all 3 Polatis optical switches used in the demonstration. The network included two OpenFlow NEC top of rack switches and was managed via a POX OpenFlow controller. Video clips of the ECOC demonstration and a subsequent more detailed sequence can be found in [5],[6].

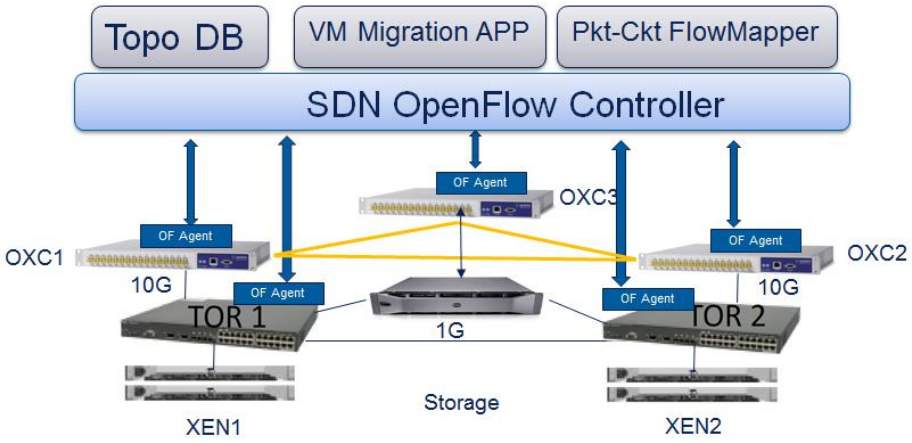

Figure 1 Hybrid packet-optical circuit switched data centre network demonstration at ECOC 2013

2.1.2 Progress

In February 2014, Polatis delivered a fully-featured 192x192 DirectLight optical switch with input and output optical power monitors to the High Performance Networks lab at UNIVBRIS (Figure 2). This unit, along with 4 other OpenFlow-enabled Polatis optical circuit switches installed in the networks lab, was used in a wide-scale UK-Japan SDN orchestration experiment under the STRAUSS project and presented as a post deadline paper at OFC 2014 [5]. The equipment was also used for the programmable all-optical intra-datacentre network experiment reported in [8].

Figure 2 A standard Polatis 192x192 optical circuit switch with LC/APC connectors: front panel (above); rear panel (below) showing redundant hot-swappable network interface cards, power supply units and fans

During the year, the team has created south-bound plugins for the OpenDaylight controller to interface with the UNIVBRIS/Polatis OpenFlow agent and was able to demonstrate OpenDaylight control of optical circuit switches for the first time during the COSIGN consortium meeting in Bristol in September 2014 [9].

2.1.3 Plans

In a separate project, the UNIVBRIS team are developing a faster and more efficient version of the OpenFlow 1.0+ agent which integrates directly with the Polatis user services API (Figure 3), rather than connecting indirectly via the TL1 interface as was done in the first release. The agent size has also been reduced by a factor of ~3 by eliminating redundant code. The agent is currently being ported to the network interface on a 192x192 optical switch and will go through integration testing during January 2015.

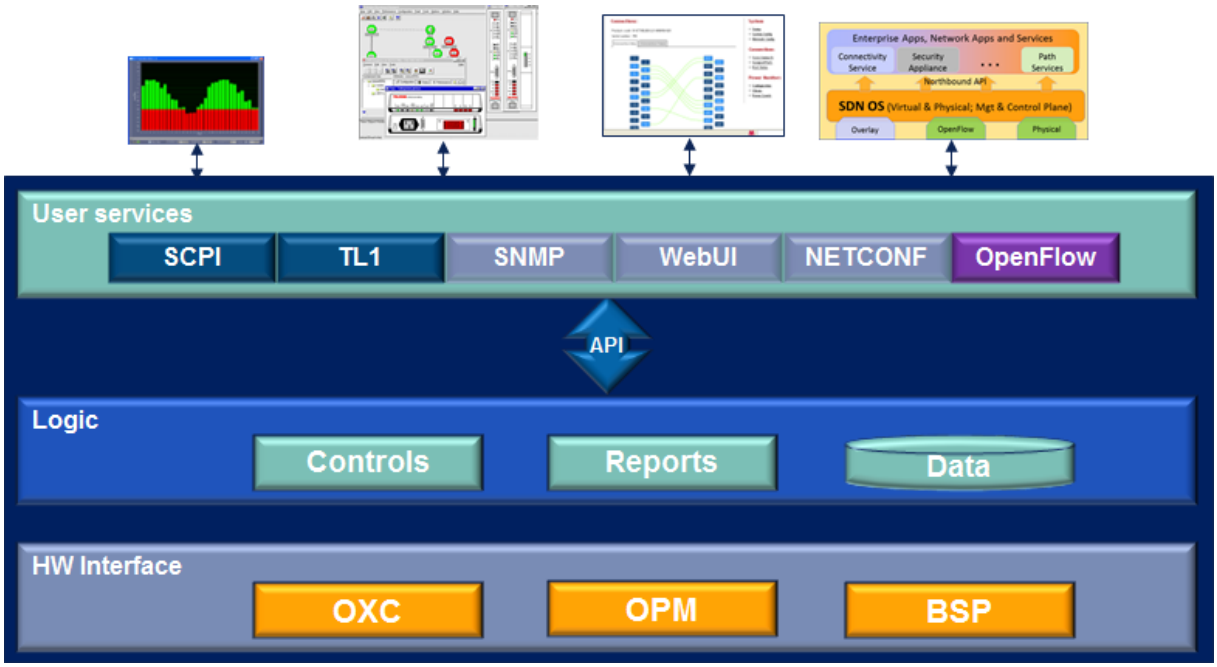

Figure 3 Polatis Optical Switch User Interface Structure

It is intended to develop the embedded OpenFlow agent further over the next 6 months to be compliant with the next stable release of OF 1.4/1.5, which will provide direct support for optical circuit switching, as opposed to the vendor/experimenter-specified extensions used with OF1.0+. Additional user interfaces will also be created to add REST and NETCONF support for configuration management.

2.2 Integration of OpenFlow and Venture Switch

2.2.1 Background

The Venture switch will work as an optical TDM switch and will be employed in the medium-term and long-term scenarios proposed in D1.4 [11]. In these scenarios, the Venture OXS switch will enable the short-lived data communication between servers and racks. More specifically, TDM switching allows for subdividing the available bandwidth between several endpoints which combined with dynamic resource allocation leads to better bandwidth utilization. In order to achieve this, the switching time has to be relatively short (a few ns) enabling fast switching operation with a small portion of the bandwidth being wasted.

The control of a TDM switch requires that an SDN controller issues commands for manipulating flows defined by an input and output port pair as well as a time slot for which the connection is established (more detail is presented in Section 4). The controller is capable of individually addressing each specific time slot for each port. In order to convey these commands, a Field Programmable Gate Array (FPGA) switch controller is needed. The controller stores the switching information for a fixed number of slots and consequently reads from the schedule and generates the corresponding configuration signals with appropriate time duration. The amplitude level of the configuration signals is adjusted by using an additional driver board which generates control signals with output amplitude as required by the switch.

The first demonstration of SDN controlled Venture OXS aims at integration of all the software and hardware components as shown in Figure 4, with simplified functionalities at each layer. The main goal is to demonstrate that the switch can be reconfigured based on the commands issued from the SDN controller. Description of the different components and details about the communication between them are given below.

Venture has provided the switch used for the demonstration, DTU has developed the driver board and the code for the FPGA to Agent communication, while UNIVBRIS has implemented the OF agent with OF message extensions and OpenDaylight[1]-based SDN controller. To enable the communication between the OF agent and the control FPGA board, we defined a raw Ethernet frame to carry all the required information, i.e., for each optical TDM flow, input port and output port pair, as well as the allocated time slots.

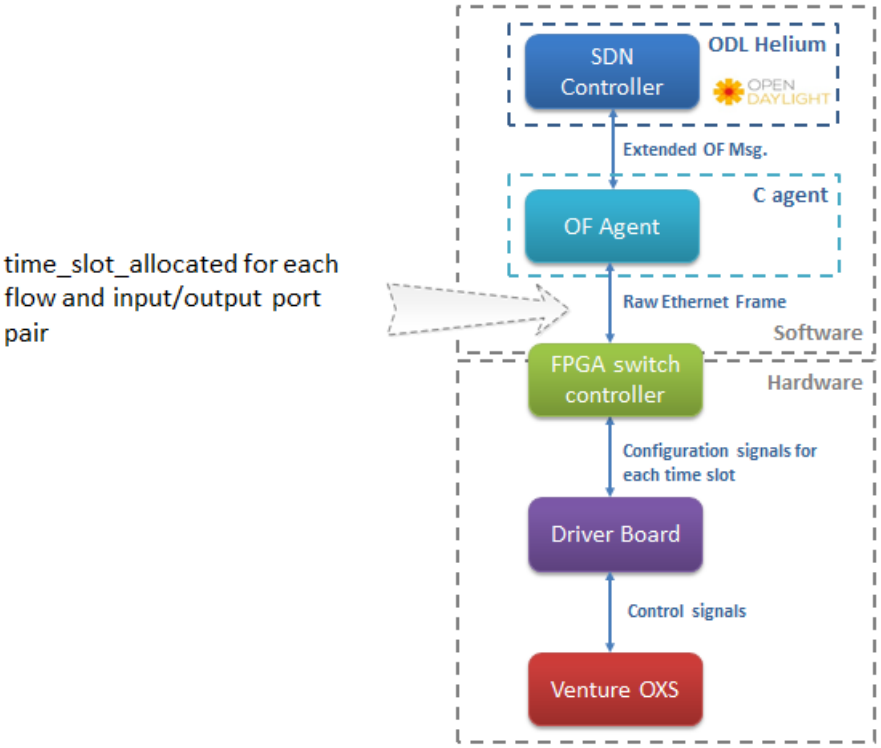

Figure 4 Schematic of the software and hardware components

In the OF agent development process, we get some benefit by sharing a common part of an OF agent (i.e., the OF communication channel and part of the OF specification) developed for another EU project-LIGHTNESS [12]. However, the south-bound interface of the OF agent has been extended to enable the communication with the FPGA switch controller (as shown in Figure 4). More details are provided in Section 2.2.2.

[1] Hydrogen version

2.2.2 Progress

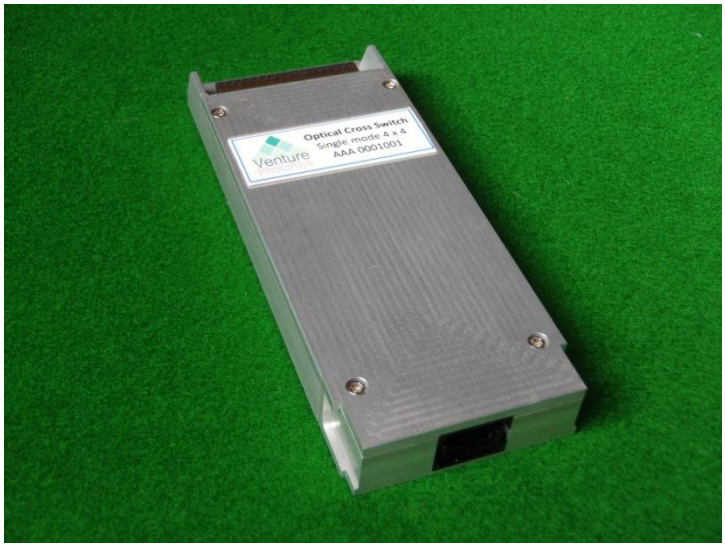

Figure 5 A standard Venture 4x4 fast Optical CrossPoint Switch in CFP2 module format with MTP single mode ribbon optical fibre interface

As shown in Figure 5, a standard Venture switch in CFP2 module is presented, and some details of the product development and its benefits are discussed and published in [9]

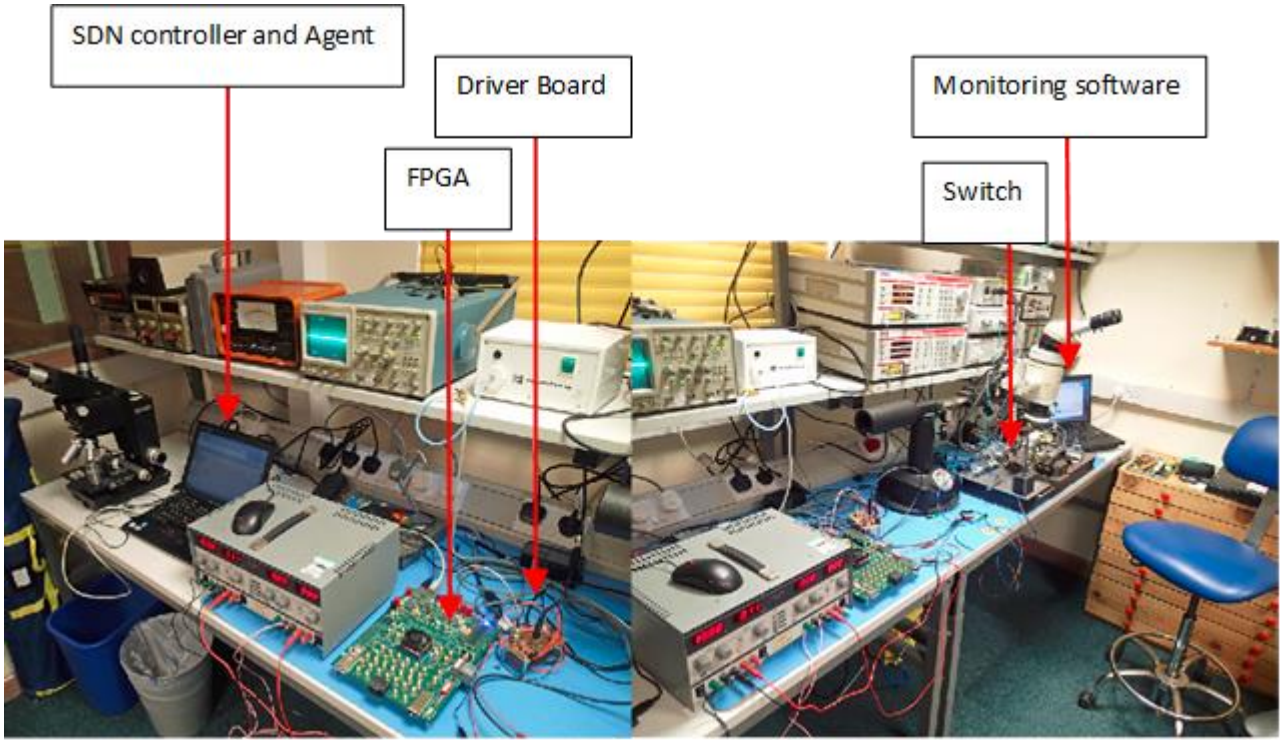

Figure 6 Integration setup

The integration setup is shown in Figure 6. An OpenDaylight (ODL) SDN controller is used, version Helium. The SDN controller and the OF Agent run on a PC and communicate through an extended OpenFlow (OF) protocol. The OF Agent is responsible for translating the commands received from the controller to commands carried in Ethernet frames that are sent to the FPGA. The FPGA communicates with the OF Agent through a 1 Gb/s Ethernet port and receives Ethernet frames with commands for the switch reconfiguration. After receiving a specific switch schedule, the FPGA updates the switch LUT and generates the corresponding driving signals. The driver board regenerates the slope of the control signals generated from the FPGA and amplifies them to a specific amplitude level as required by the Venture OXS. Successful on/off switching according to the controller commands has been demonstrated.

Ethernet Frame for Veture Switch Configuration (byte for each column)																																Ethernet Frame lines (Captured by Wireshark)
ADDR\BYTE	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
																			sequesnce number		Source MAC Address									Destination MAC Address		0
00	RESERVED																																10,20
01	Keep alive message length								Time-slice duration												frame size	Time-slice numbers								Mode			30,40
02	1 to 32 time slot: fast switch control commands for port 1																																50,60
03	33 to 64 time slot: fast switch control commands for port 1																																70,80
04	64 to 96 time slot: fast switch control commands for port 1																																90,a0
05	1 to 32 time slot: fast switch control commands for port 2																																b0,c0
06	33 to 64 time slot: fast switch control commands for port 2																																d0,e0
07	64 to 96 time slot: fast switch control commands for port 2																																f0,100
08																																	110,120

Figure 7 Ethernet frame used for communication between the Agent and the FPGA controller

Figure 7 shows the content of the Ethernet frame that has been used for communication between the Agent and the FPGA controller. The Wireshark snapshot of the frame exchange between the Agent and the FPGA is shown in Figure 8, while the Signal Tap snapshots showing the frame exchange from the FPGA side when the flow is installed and uninstalled are shown in Figure 9. Figure 10 gives a detailed overview of the first fields of the frame received. It can be seen that the source and destination address correspond to the ones in the Wireshark snapshot. Figure 11 shows the slot configuration details for part of the slots when both install and uninstall flow frames have been received. It can be seen that the configuration of part of the slots displayed is either 0x00 or 0x05, i.e., a precondition for generating the configuration signals of zero and one, respectively. Figure 12 shows the oscilloscope traces of the configuration signals generated by the FPGA and the configuration signals at the output of the driver board. The difference in voltage output level is due to the fact that the driver board has been adjusted to generate a signal output with specific voltage level in order to match it with the requirements of the switch.

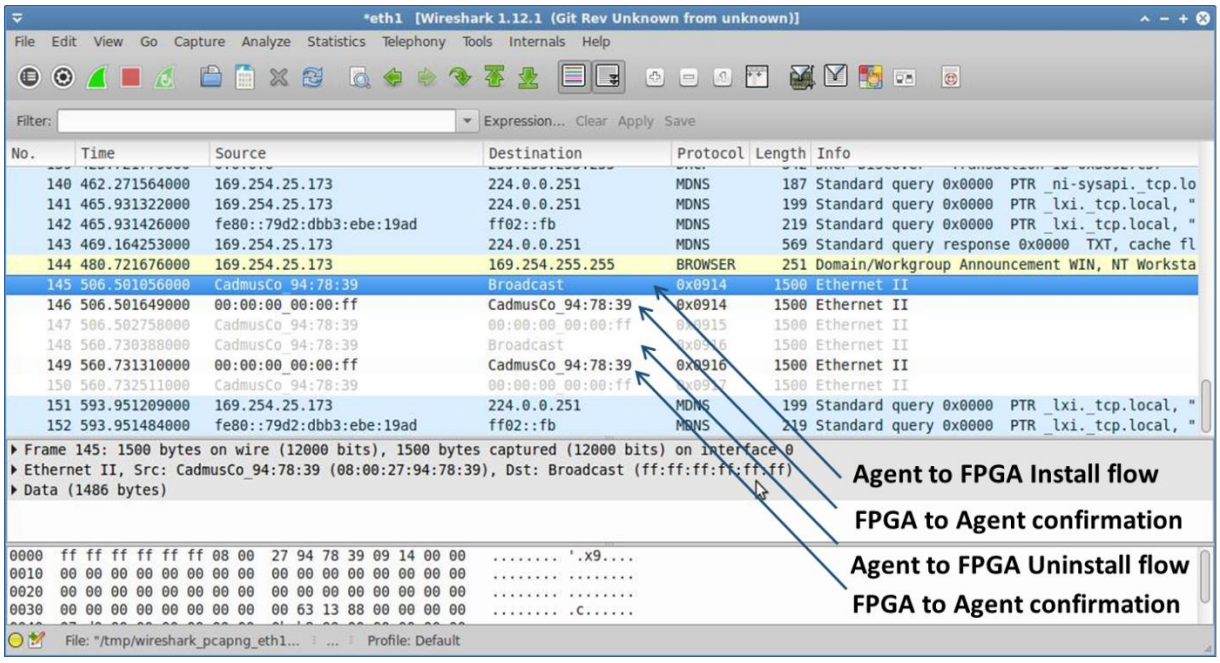

Figure 8 Wireshark snapshots of the Ethernet frames exchanged between the Agent and the FPGA controller

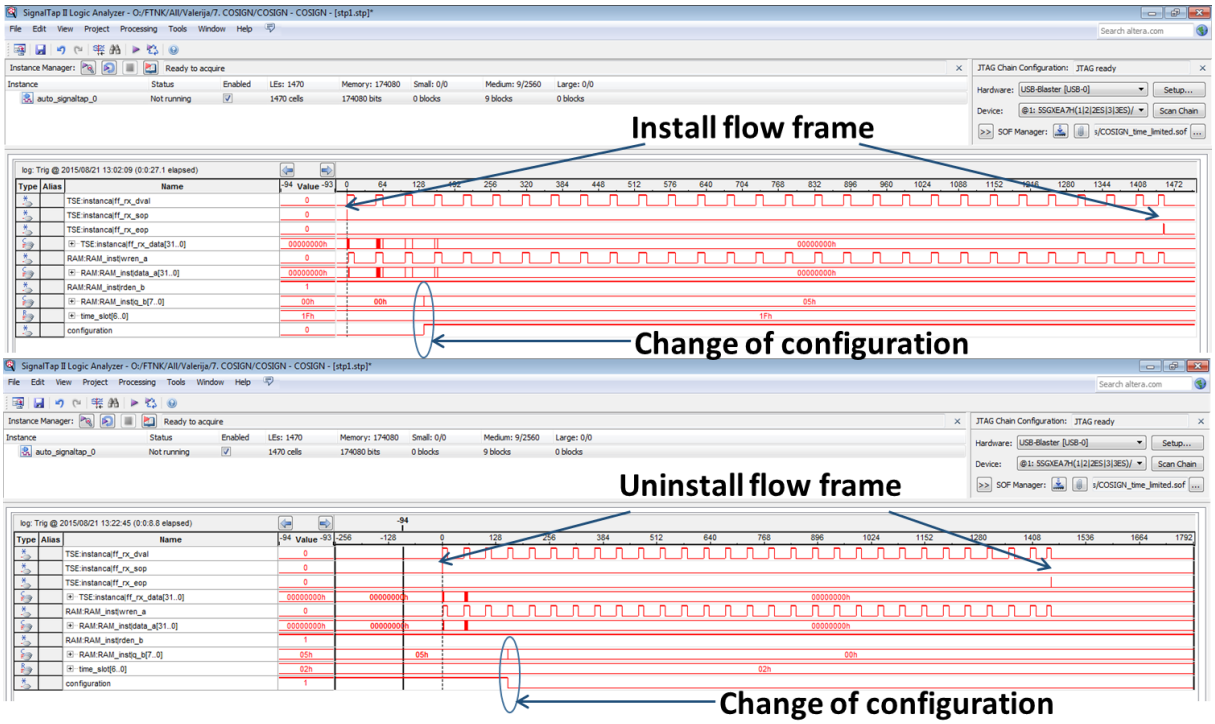

Figure 9 Signal Tap snapshots of the Ethernet frames received at the FPGA and the configuration signal generation

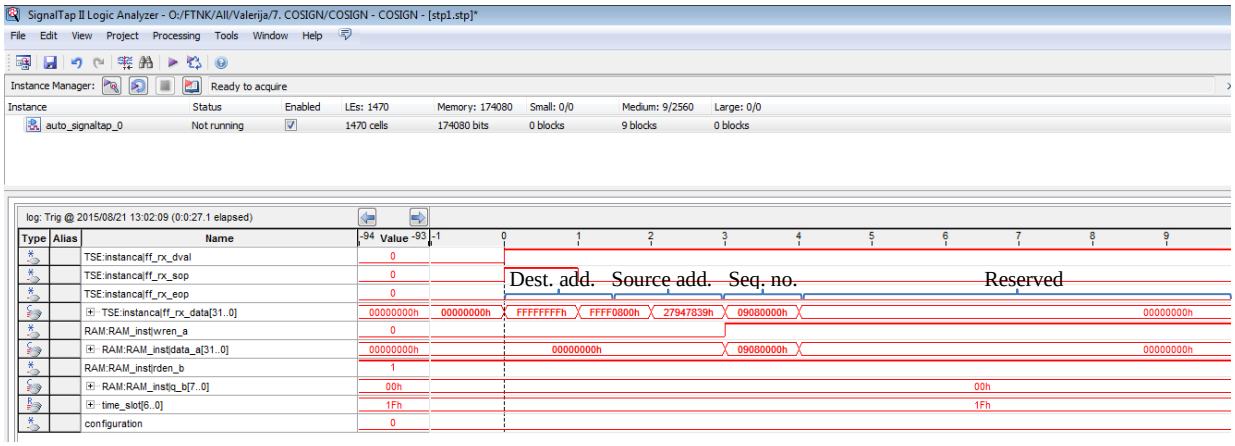

Figure 10 Signal Tap snapshots of the content of the first fields of the Ethernet frames received at the FPGA

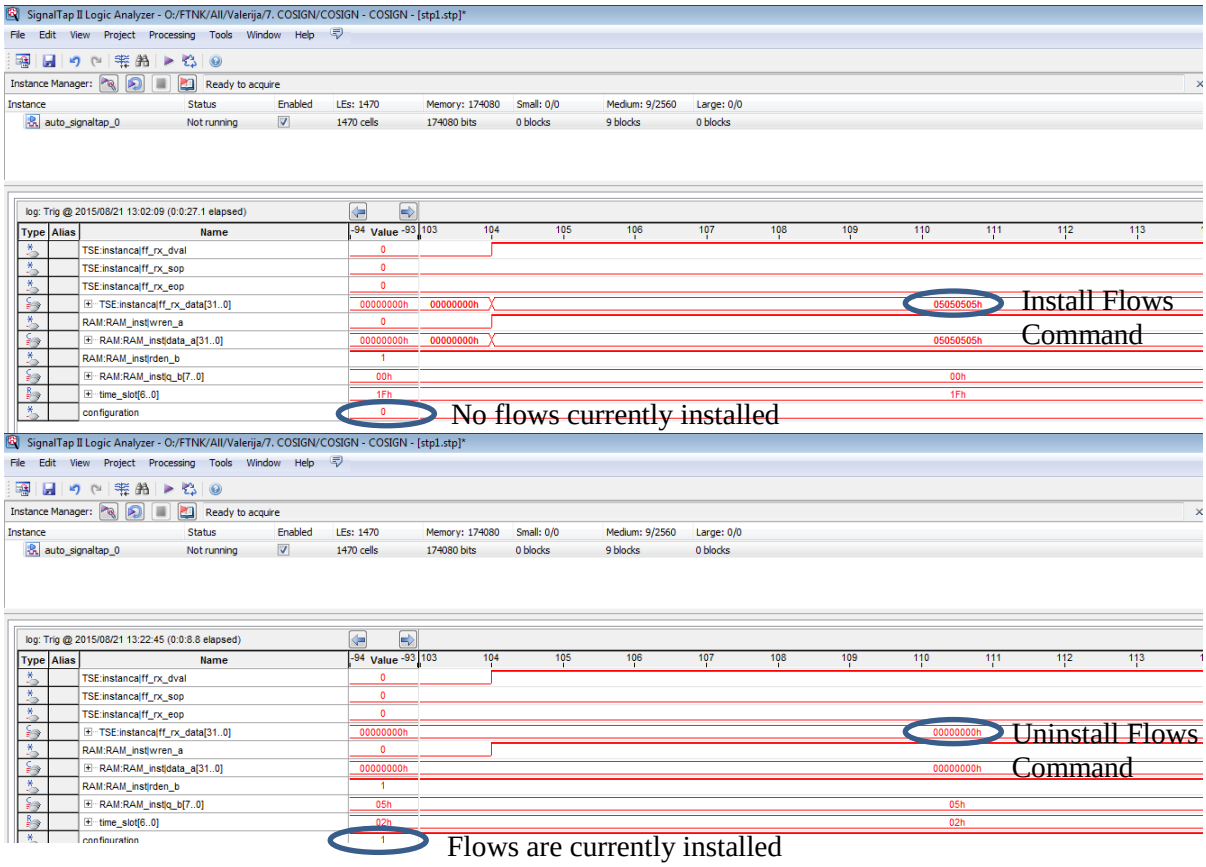

Figure 11 Signal Tap snapshots of the slot configuration field's content of the install flow and uninstall flow Ethernet frames received at the FPGA

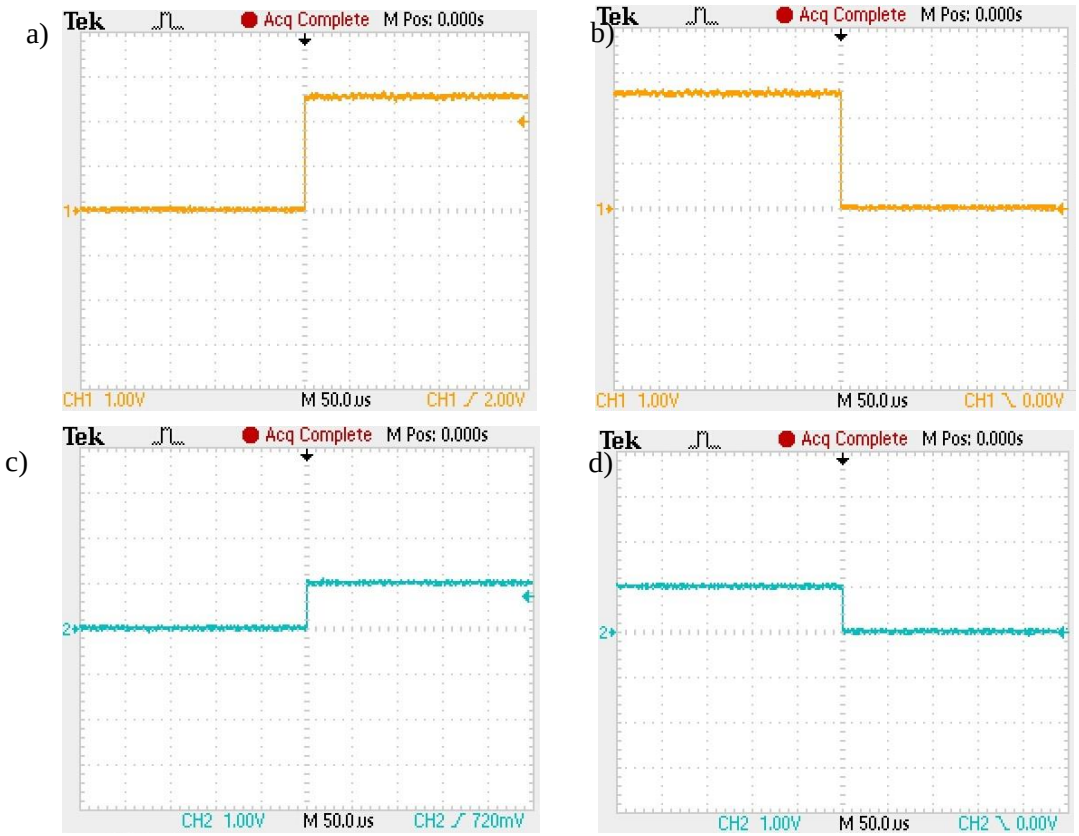

Figure 12 Oscilloscope traces of the configuration signals at the output of the FPGA when a) flows are installed and b) when flows are uninstalled and at the output of the driver when c) flows are installed and when d) flows are uninstalled

2.2.3 Future Plans

In future, DTU and UNIBRIS will continue working on extending the functionalities for the Agent/FPGA communication and adding additional commands for diagnostic purposes as well as support for new services, e.g., monitoring capability, time slot modification and flow modification. VENTURE will continue working on refining the switch product and providing a fully packaged device.

3 Annex 1: OpenFlow 1.0+ Optical Circuit Switch Extensions

This annex describes the current command extensions used by the embedded OpenFlow agent (POLOA) developed by UNIVBRIS to control the Polatis optical circuit switch. The command set is defined by OpenFlow 1.0 and the Circuit Switch Addendum v0.3 [2] together with additional proprietary extensions as detailed below.

Within the lifetime of the project, it is planned to enhance the embedded OpenFlow agent to support OpenFlow 1.4 (which natively supports circuit switching) and so the following information will be revised.

3.1 Connect Input to Output Port

The circuit switch Flow Table is modified using an OFPT_CFLOW_MOD command, and as noted in the addendum it is required that OFPFC_MODIFY_STRICT and OFPFC_DELETE_STRICT are used to modify and terminate existing connections. The Polatis switch only supports real ports.

3.2 Read Power at Given Port

Uses a proprietary extension to the OFPT_STATS_REQUEST command with type set to OFPST_CPORT = 0x18 and a following data block laid out as:

```
struct ofp_cport_stats_request {
     uint16_t port_no;
     uint8_t direction;
     uint8_t pad[5];
 };
  OFP_ASSERT(sizeof(struct ofp_cport_stats_request) == 8);
```

The OFPST_PORT message must request statistics either for a single port (specified in port_no) or for all ports (if port_no == OFPP_NONE).
The response appears in the corresponding OFPT_STATS_REPLY, with a proprietary body either once or multiple times:

```
struct ofp_cport_stats {
    uint16_t port_no;
    uint16_t pmonset_lambda;
    uint16_t pmonset_offset;
    uint16_t pmonset_atime;

    uint32_t mes_pmon;
    uint8_t pad[4];          /* Align to 64-bits. */

    uint8_t voaset_mode;
    uint8_t voaset_level;
    uint8_t voaset_ref;
    uint8_t pad1[5];          /* Align to 64-bits. */

    uint64_t port_alarms;
};
OFP_ASSERT(sizeof(struct ofp_cport_stats) == 32);
```

If a value is unsupported, the field is set to all ones.

3.3 Trigger Alarm for Power below Threshold

The port_alarm field of the OFPT_STATS_REPLY structure can be checked to discover if an alarm has been raised.

4 Annex 2: OpenFlow 1.0+ Optical TDM Connection Extension

This annex describes the OF protocol extensions used by the OpenFlow agent developed by UNIVBRIS to control the OXS switch. This development is based on OpenFlow 1.0 together with additional proprietary extensions as detailed below in Figure 13. Within the lifetime of the project, more function is planned to be added to enhance the OpenFlow agent, e.g., setting the time slot size through OpenDaylight controller and get some physical layer information (power level) from OXS.

This new action is used to set the exact TDM slot number which will be allocated for a specific flow filtered with ofp_match field. In addition, Figure 14 through Figure 17 are Wireshark snapshots showing the detailed content of the Ethernet frames exchanged between the Agent and the FPGA controllers.

```
public class OFActionSetTimeslot extends OFAction {
    private static final long serialVersionUID = 1L;
    public static int MINIMUM_LENGTH = 8;

    protected short timeSlot;
```

Figure 13 New Set Time Slots action in OF protocol

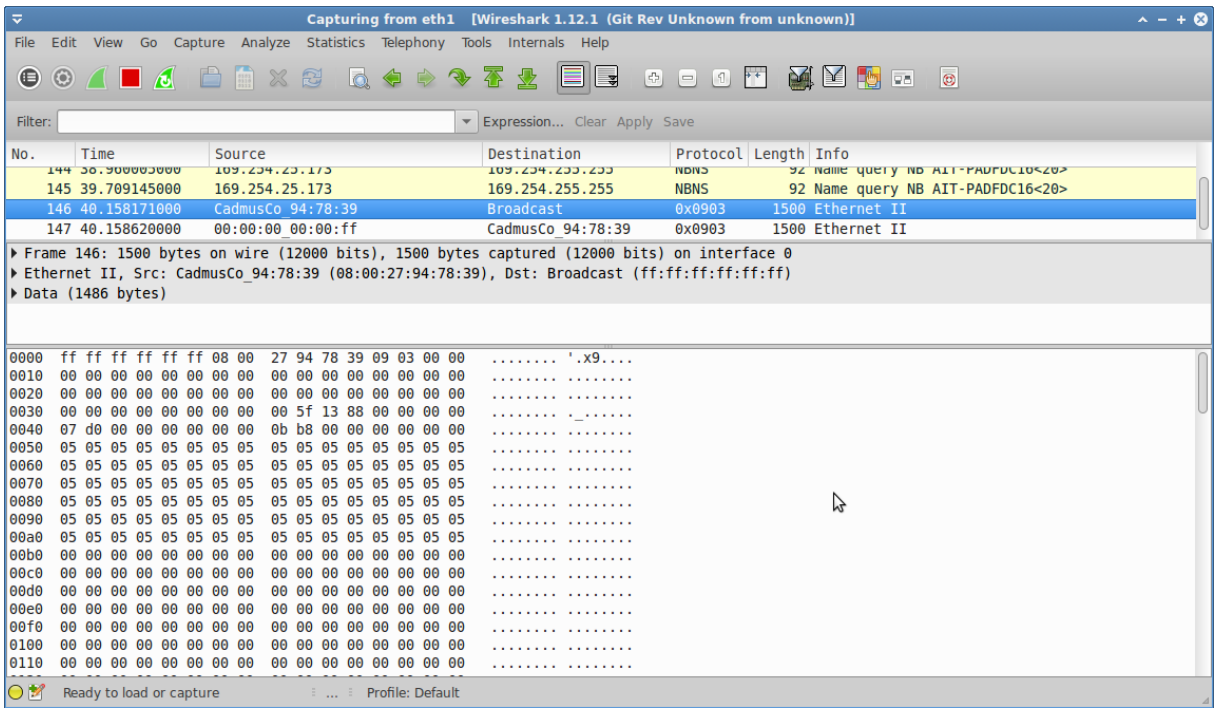

Figure 14 Install flows frame from Agent to FPGA

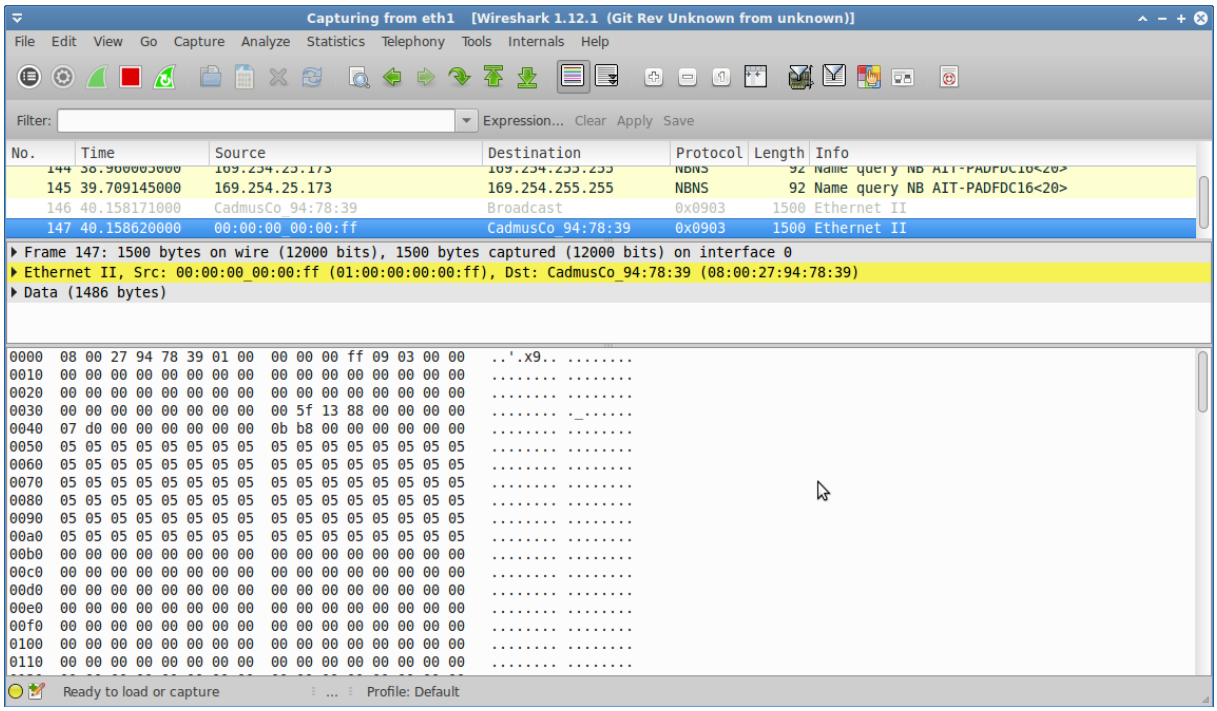

Figure 15 Confirmation of install flows frame from FPGA to Agent

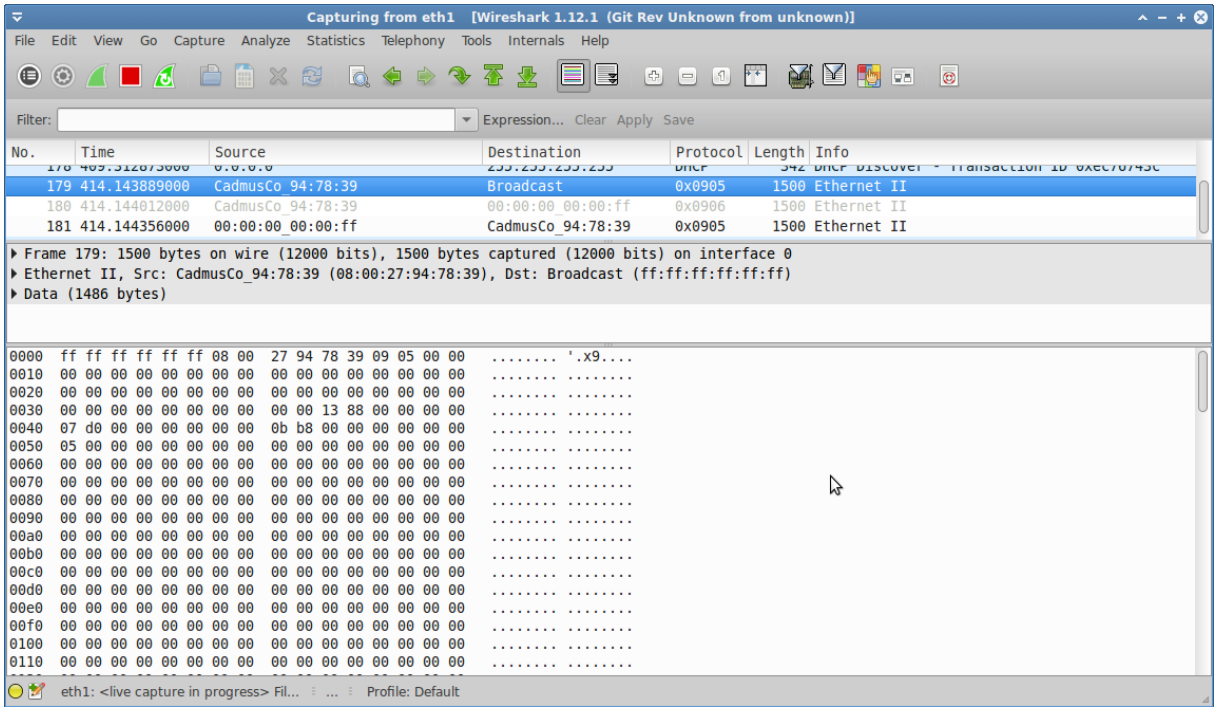

Figure 16 Uninstall flows frame from Agent to FPGA

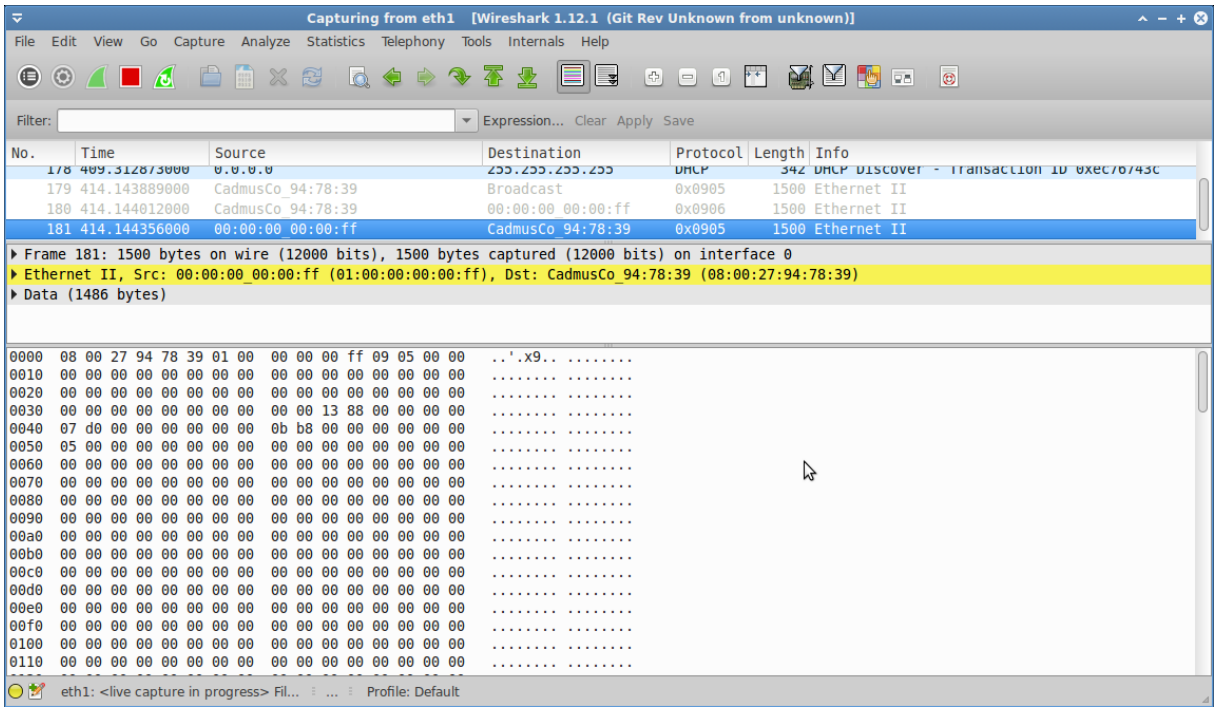

Figure 17 Confirmation of uninstall flows frame from FPGA to Agent