ARL-TR-8043 JUN 2017

US Army Research Laboratory

# OpenFlow Extensions for Programmable Quantum Networks

by Venkat Dasari, Nikolai Snow, Billy Geerhart, and Sam Snodgrass

ARL-TR-8043 ● JUN 2017

US Army Research Laboratory

# OpenFlow Extensions for Programmable Quantum Networks

by Venkat Dasari, Nikolai Snow, and Billy Geerhart
*Computational and Information Sciences Directorate, ARL*

Sam Snodgrass
*Drexel University, Philadelphia, PA*

| **REPORT DOCUMENTATION PAGE** | | | *Form Approved* *OMB No. 0704-0188* |
|---|---|---|---|

| | | |
|---|---|---|
| **1. REPORT DATE** *(DD-MM-YYYY)* June 2017 | **2. REPORT TYPE** Technical Report | **3. DATES COVERED (From - To)** June 2015–March 2017 |
| **4. TITLE AND SUBTITLE** OpenFlow Extensions for Programmable Quantum Networks | | **5a. CONTRACT NUMBER** |
| | | **5b. GRANT NUMBER** |
| | | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Venkat Dasari, Nikolai Snow, Billy Geerhart, and Sam Snodgrass | | **5d. PROJECT NUMBER** |
| | | **5e. TASK NUMBER** |
| | | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** US Army Research Laboratory ATTN: RDRL-CIH-S Aberdeen Proving Ground, MD 21005-5067 | | **8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-8043 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | | **10. SPONSOR/MONITOR'S ACRONYM(S)** |
| | | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
Networks are essential for both quantum and classical systems for transporting information between network nodes. Networks define abstractions and interfaces required for establishing stable communication frameworks required for network applications to communicate. Advances in network protocols and architectures have led to the development of software-defined programmable networks. Previously, the feasibility of extending programmable network principles to build multinode quantum networks has been described. This project built upon a previously described programmable quantum network model to leverage new optical labels in the latest OpenFlow protocol to manage optical channels and lambda switching. Custom modifications to the OpenFlow protocol base code were developed to create new labels for managing the optical channels associated with quantum networks. Additionally, the OpenFlow match/action tables were modified to recognize and take action on new OpenFlow labels related to the quantum network attributes.

**15. SUBJECT TERMS**
quantum, teleportation, OpenFlow, controller, metadata

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Venkat Dasari |
|---|---|---|---|---|---|
| **a. REPORT** Unclassified | **b. ABSTRACT** Unclassified | **c. THIS PAGE** Unclassified | UU | 16 | **19b. TELEPHONE NUMBER (Include area code)** 410-278-2846 |

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

## Contents

## List of Figures

## 1. Introduction

Quantum networks and quantum computing have been receiving a surge of interest recently.[1–3] However, there has been little development of network abstractions to allow for communication between the quantum and classical components of a quantum network. We are interested in developing these abstractions by extending OpenFlow, which is a well-known software-defined network (SDN) protocol. Defining software-defined quantum networks has been proposed previously,[4] and important quantum metadata attributes that need to be passed between the quantum and classical channels have been defined.[5] We build upon these previous works by extending the OpenFlow protocol to recognize the metadata labels within packets and to match on those labels to route the packets accordingly.

In the remainder of the report we

1) formulate the specific problem we are trying to address,
2) give a brief introduction to the work that has previously been performed on this topic,
3) outline our approach for addressing the stated problem,
4) discuss the results of our approach, and
5) draw conclusions and indicate plans for future extensions to this work.

We are addressing the problem of communication between quantum and classical channels within a quantum network. Quantum applications generally require the synchronized transfer of information through both the quantum and classical channel. However, currently there are no standard communication protocols for communication between these channels. To address this issue, we extend the OpenFlow protocol with additional fields corresponding to quantum metadata that can be passed between the quantum and classical channels to facilitate quantum applications.

## 2. Background

In this section we give a brief introduction to quantum networks before briefly discussing SDNs.

## 2.1 Quantum Networks

Quantum networks are composed of a set of quantum systems connected via quantum communication channels[6] and a set of classical systems connected via classical communication channels. These networks differ from standard classical networks by their use of quantum physical phenomena to achieve network functionalities on the quantum channels, such as encoding, transmitting, and decoding information.[7] Specifically, quantum nodes communicate using entangled particles and perform calculations using quantum logic gates. Additionally, quantum computing uses a quantum bit (qubit) as its basic unit of computation, which exists in a probabilistic complex vector space where each qubit can be in a superposition of values, whereas classical computing uses bits from a discrete Boolean state space as its basic unit of computation. These differences make communication between the quantum and classical channels very difficult. However, communication between these channels is required for the transmission of important quantum metadata needed for quantum applications, and Dasari and Humble[5] have previously defined the metadata attributes that are required for a quantum network. The current work extends theirs by adding the attributes to the OpenFlow protocol for software-defined networks, allowing for the transmission of those attributes throughout our emulated network and, therefore, communication between the classical and quantum channels (Figs. 1 and 2).

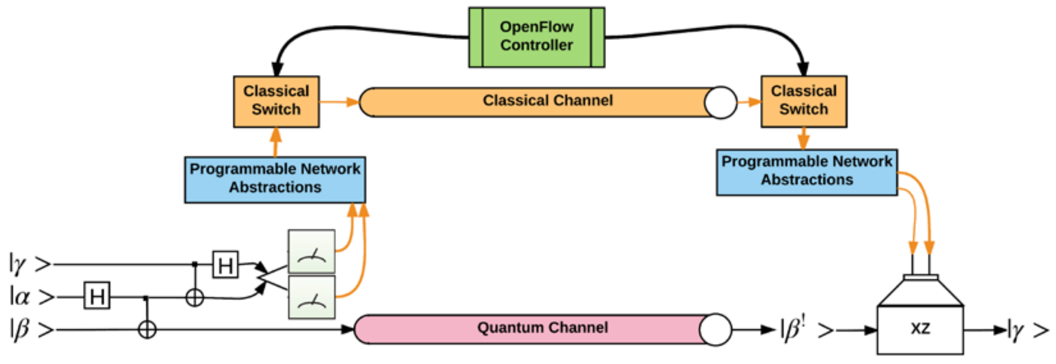


**Fig. 1 Quantum teleportation using quantum circuits. The gamma photon on the left is teleported to the right using both the quantum and classical channel.**

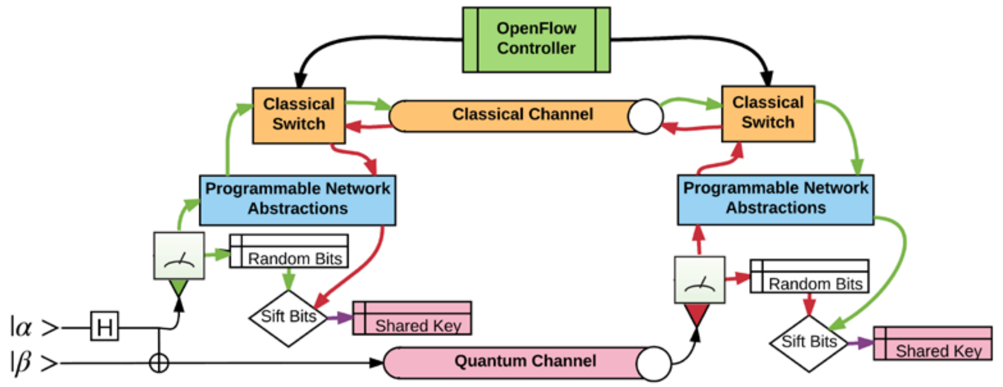


**Fig. 2** **Symmetric key distribution using a quantum and classical channel working together using OpenFlow**

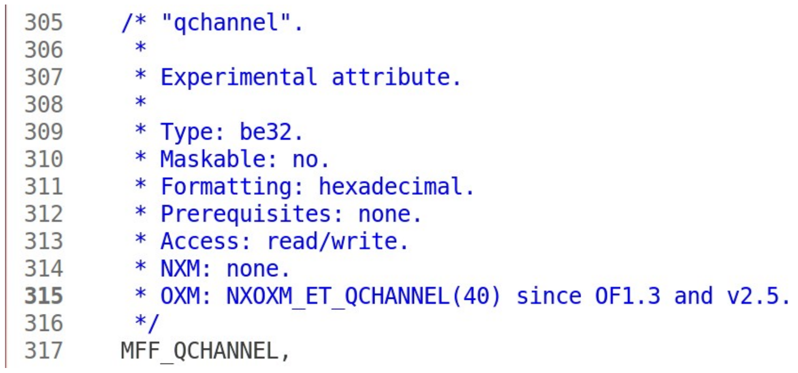
### 2.2 Software-Defined Networks

SDNs differ from standard networks by separating the control and data plane of the network. That is, while in a standard network each switch has its own control intelligence, in an SDN the control intelligence is removed from the switches and placed in a centralized controller. This forces the switches to act as forwarding devices, while the centralized intelligence installs logic into the switches as needed.[8] The controller itself can service many switches, which allows the controller to build a global view of the network; therefore, more-informed and more-intelligent behavior can be implemented.

SDNs have previously been proposed for use with quantum networks.[4] SDN principles applied to a quantum network allow for easier communication between the classical and quantum components; in particular, abstractions can be defined within the networking protocols to allow for such communications, and logic can be easily installed onto the controller to handle those communications appropriately. OpenFlow is a widespread protocol that adheres to SDN principles and is commonly used to develop such networks.[9]

## 3. Approach

SDN principles allow for a programmable control plane from which a network operating system (OS) can be made. The network OS can be used to coordinate the synchronized transfer of information across disparate communication channels. This is especially useful when trying to build a standardized communication protocol to allow the quantum and classical channels to work together. Synchronizing the quantum and classical channels requires the introduction of new quantum metadata attributes at the packet level, which also requires changes in the software at the switch and controller level (Fig. 3).



```
/* "qchannel".
 *
 * Experimental attribute.
 *
 * Type: be32.
 * Maskable: no.
 * Formatting: hexadecimal.
 * Prerequisites: none.
 * Access: read/write.
 * NXM: none.
 * OXM: NXOXM_ET_QCHANNEL(40) since OF1.3 and v2.5.
 */
MFF_QCHANNEL,
```

**Fig. 3 A snippet of code from the OpenVSwitch code base where we added a new field corresponding to the quantum channel**

## 3.1 Metadata

Quantum metadata specifying parameters and protocols for the quantum channel and application is required for quantum applications to run on a quantum network. The metadata attributes that we added to OpenFlow are several of those proposed and defined by Dasari and Humble.[5] Specifically, we added the following labels:

- QCHANNEL: a unique identifier for the quantum channel for which the rest of the metadata applies (i.e., which channel communications and quantum applications will be performed)
- QCOM: a unique identifier for the quantum application to be performed (e.g., quantum key distribution or teleportation)
- QEC: a unique identifier for the quantum error correction protocol to be used for detecting and correcting errors.

## 3.2 Switch

In an SDN, switches function as fast-forwarding devices with no internal intelligence of their own. However, the switches still need to be able to recognize the labels or attributes of the incoming packets in order to route them according to the logic installed by the controller.

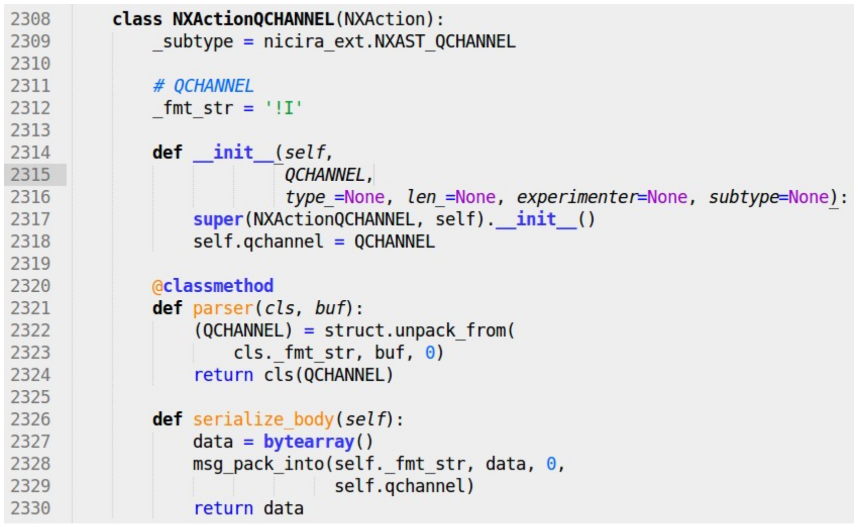
We chose OpenVSwitch[10] as our switch mostly because of its widespread use and support as well as its inclusion in the Mininet network emulator. The main drawback of OpenVSwitch is that it is an enormous code base (comprising hundreds of files and hundreds of thousands of lines of code). As a result, ensuring consistency throughout the code base is a challenge. Additionally, though there is a fairly large community, there has not been much work in adding new fields to the code base, resulting in only a small amount of documentation and guidance for this task.

Despite these setbacks we were able to define our labels in OpenVSwitch and achieve consistency throughout the code base, allowing us to recognize our labels and send packets with our labels on to the controller. Figure 3 shows one place we added our labels.

## 3.3 Controller

In an SDN the controller contains all of the intelligence of the network.[8] When a switch encounters a packet it does not know how to route, it forwards it to the controller, which decides what should be done with the packet. Furthermore, the controller will install that logic onto the switch so that the switch will know what to do the next time a similar packet arrives.

We chose Ryu[11] as our controller for 2 reasons: Mininet easily interfaced with Ryu, and of all the controllers we investigated, Ryu appeared to be the most straightforward to add new label capabilities. To add our labels to Ryu, we needed to define the labels as well as functions for parsing our new labels. Figure 4 shows the functions we created for one of our new labels. After adding our labels and parsing functions, we are able to see our labels and values within the controller as well as being passed between the switch and controller using WireShark.

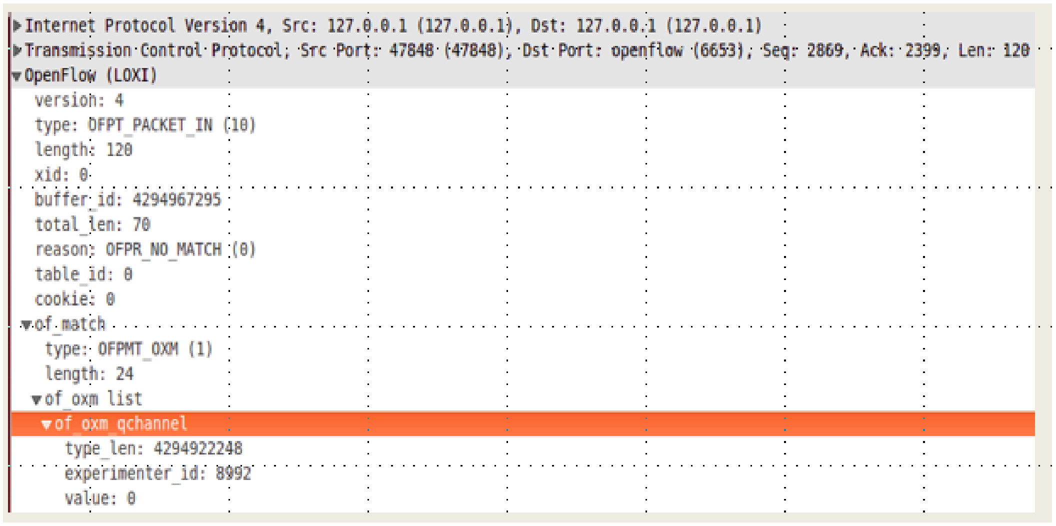


```
class NXActionQCHANNEL(NXAction):
    _subtype = nicira_ext.NXAST_QCHANNEL

    # QCHANNEL
    _fmt_str = '!I'

    def __init__(self,
                 QCHANNEL,
                 type_=None, len_=None, experimenter=None, subtype=None):
        super(NXActionQCHANNEL, self).__init__()
        self.qchannel = QCHANNEL

    @classmethod
    def parser(cls, buf):
        (QCHANNEL) = struct.unpack_from(
            cls._fmt_str, buf, 0)
        return cls(QCHANNEL)

    def serialize_body(self):
        data = bytearray()
        msg_pack_into(self._fmt_str, data, 0,
                      self.qchannel)
        return data
```

**Fig. 4 A code snippet from the Ryu code base showing the parsing function we wrote for one of our new labels**

## 3.4 WireShark

To verify that the packets being sent through the network contain our new labels, we used the packet sniffing tool WireShark.[12] However, our labels are not present in standard packets, so we modified a plugin for WireShark, allowing it to parse and display our labels when it encountered them in a packet. Using Wireshark, we could see a packet with one of our labels traveling from the switch to the controller (Fig. 5).



```
▶Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▶Transmission Control Protocol, Src Port: 47848 (47848), Dst Port: openflow (6653), Seq: 2869, Ack: 2399, Len: 120
▼OpenFlow (LOXI)
   version: 4
   type: OFPT_PACKET_IN (10)
   length: 120
   xid: 0
   buffer_id: 4294967295
   total_len: 70
   reason: OFPR_NO_MATCH (0)
   table_id: 0
   cookie: 0
  ▼of_match
    type: OFPMT_OXM (1)
    length: 24
   ▼of_oxm_list
    ▼of_oxm_qchannel
       type_len: 4294922248
       experimenter_id: 8992
       value: 0
```

**Fig. 5 WireShark output showing our custom qchannel label**

## 4. Results

After extensive modifications to OpenVSwitch and adding our labels and parsing capabilities to Ryu, we were able to send and recognize packets between the switch and controller. Furthermore, we were able to verify this claim by modifying WireShark to be able to recognize our labels. Achieving this transmission of new labels between the switch and controller is a promising result, as it shows that our quantum metadata labels are able to be passed through the network allowing for communication between the quantum and classical channels, which will facilitate quantum applications and make communication on the quantum channel more reliable.

## 5. Conclusions and Future Work

In this report we discussed our approach for enabling communication between the quantum and classical channels within a quantum network. We accomplished this by extending the OpenFlow protocol to allow for additional quantum metadata labels, corresponding to necessary information about the quantum channel and quantum applications, to be transmitted through the network. Standardized communication between the quantum and classical channels eases the process of developing quantum applications by removing the burden of achieving that communication from the developer. Additionally, these communications will allow for more-reliable communications between the quantum channels by setting up the quantum channels and networking environment before the important data are transmitted.

In the future we plan to extend this work by allowing a nonswitch node to send a packet containing the quantum metadata labels to the switch and have it routed appropriately. We are also planning to apply our framework to a physical multinode quantum network after more testing is completed in our emulated environment.

## 6. References

1. Dasari VR, Humble TS. OpenFlow arbitrated programmable network channels for managing quantum metadata; 2016 [accessed 2017 May 10]. https://arxiv.org/pdf/151.08545.

2. Yin Z-Q, Yang WL, Sun L, Duan LM. Quantum network of superconducting qubits through an optomechanical interface. Phys Rev A. 2015;91:012333.

3. Bacsardi L, Imre S. Analyzing the quantum based satellite communications. Procedia Computer Science. 2011;7:256–257.

4. Humble TS, Sadlier RJ; Software-defined quantum communication systems. Opt Eng. 2001;53(8):086103.

5. Dasari VR, Humble TS. OpenFlow arbitrated programmable network channels for managing quantum metadata. Journal of Defense Modeling and Simulations; 2016 Oct. doi: 10.1177/1548512916661781.

6. Mouradian S, Schroder T, Poitras CB. Scalable integration of solid state quantum memories into a photonic network. IEEE Xplore Digital Library; 2015.

7. Dasari VR, Sadlier RJ, Prout R, Williams B, Humble TS. Programmable multi-node quantum network design and simulation. Proc SPIE 9873; 2016. doi:10.1117/12.2234697.

8. Lantz B, Heller B, Mckeown N. A network in a laptop: rapid prototyping for software-defined networks. Stanford (CA): Stanford University HotNets; 2010.

9. Raychev N. Algorithm for switching 4-bit packages in full quantum network with multiple network nodes. International Journal of Scientific and Engineering Research. 2015;6(8):1289–1294.

10. Pfaff B, Pettit J, Koponen T, Jackson EJ, Zhou A, Rajahalme J, Gross J, Wang A, Stringer J, Shelar P, Amidon K, Casado M. The design and implementation of OpenVSwitch. Presented at the 12th USENIX Symposium on Networked Systems Design and Implementation; 2015 May 4–6; Oakland, CA.

11. Hosseini-Suny K, Momeni H, Janabi-Sharifi F. A modified adaptive controller design for teleoperation systems. Robotics and autonomous systems. 2010;58(5):676–683.

12. Cardwell K, Dalziel H. Essential skills for hackers. Network Security. 2016;7:4.

| | |
|---|---|
| 1<br>(PDF) | DEFENSE TECHNICAL<br>INFORMATION CTR<br>DTIC OCA |
| 2<br>(PDF) | DIRECTOR<br>US ARMY RESEARCH LAB<br>RDRL CIO L<br>IMAL HRA MAIL & RECORDS<br>MGMT |
| 1<br>(PDF) | GOVT PRINTG OFC<br>A MALHOTRA |
| 4<br>(PDF) | DIR USARL<br>RDRL CIH S<br>V DASARI<br>N SNOW<br>B GEERHART<br>S SNODGRASS |

INTENTIONALLY LEFT BLANK.