Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on June 2, 2009.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
Abstract

While both ForCES and OpenFlow follow the basic idea of separations of forwarding plane and control plane in network elements, they are technically very different. This document analyzes the differences between OpenFlow and ForCES technically from the aspects of goals, architecture, forwarding model and protocol interface. The two techniques can learn much from each other in their standardization process.

Table of Contents

1. Introduction ................................................ 2
2. Definitions used in this document............................ 3
3. Comparisons between ForCES and OpenFlow...................... 4
   3.1. Difference in Goals.................................... 4
   3.2. Difference in Architecture.............................. 5
   3.3. Difference in Forwarding Model........................ 8
   3.4. Difference in Protocol Interface........................ 9
4. Security Considerations...................................... 9
5. IANA Considerations ........................................ 9
6. Conclusions ................................................ 9
7. References .................................................. 10
   7.1. Normative References................................... 10
   7.2. Informative References................................ 10
8. Acknowledgments ............................................ 11

1. Introduction

ForCES (Forwarding and Control Element Separation) [RFC5810] defines a framework and associated protocols to standardize information exchange between the control and forwarding plane in a ForCES network element (NE).

OpenFlow [McKeown2008][OpenFlow1.1] is an implementation of the idea of so-called SDN (Software-Defined Networking). In network elements, i.e., OpenFlow switches, control plane has been separated from forwarding plane and only forwarding plane is retained. The centralized controller is used to control the behavior of OpenFlow switches by adding, updating and deleting flow table entries in switches. ONF (Open Networking Foundation, Website: https://www.opennetworking.org/) has been founded in March of 2011 to promote the SDN, and especially Standardize OpenFlow protocol.

Both ForCES and OpenFlow follow the basic idea of forwarding plane and control plane separation in network elements, and result in the
new architecture of network devices, e.g., routers and switches. However, they are technically different in many aspects. It is necessary to compare the two techniques so that they can learn much from each other. This document analyzes the differences and similarities between ForCES and OpenFlow from design goals, architecture, forwarding model and protocol interface.

2. Definitions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

The following definitions related to ForCES and relevant to this document are taken from [RFC3654][RFC3746][RFC5810].

Forwarding Element (FE) - A logical entity that implements the ForCES Protocol. FEs use the underlying hardware to provide per-packet processing and handling as directed by a CE via the ForCES Protocol.

Control Element (CE) - A logical entity that implements the ForCES Protocol and uses it to instruct one or more FEs on how to process packets. CEs handle functionality such as the execution of control and signaling protocols.

ForCES Network Element (NE) - An entity composed of one or more CEs and one or more FEs. An NE usually hides its internal organization from external entities and represents a single point of management to entities outside the NE.

ForCES Protocol - While there may be multiple protocols used within the overall ForCES architecture, the term "ForCES protocol" refers to the Fp reference points in the ForCES framework in [RFC3746]. This protocol does not apply to CE-to-CE communication, FE-to-FE communication, or communication between FE and CE managers. Basically, the ForCES protocol works in a master-slave mode in which FEs are slaves and CEs are masters.

ForCES Protocol Layer (ForCES PL) - A layer in the ForCES protocol architecture that defines the ForCES protocol messages, the protocol state transfer scheme, and the ForCES protocol architecture itself (including requirements of ForCES TML as shown below). Specifications of ForCES PL are defined by [RFC5810].

ForCES Protocol Transport Mapping Layer (ForCES TML) - A layer in ForCES protocol architecture that uses the capabilities of existing transport protocols to specifically address protocol message
transportation issues, such as how the protocol messages are mapped to different transport media (like TCP, IP, ATM, Ethernet, etc.), and how to achieve and implement reliability, multicast, ordering, etc. The ForCES TML specifications are detailed in separate ForCES documents, one for each TML.

LFB (Logical Function Block) - The basic building block that is operated on by the ForCES protocol. The LFB is a well-defined, logically separable functional block that resides in an FE and is controlled by the CE via the ForCES protocol. The LFB may reside at the FE’s data path and process packets or may be purely an FE control or configuration entity that is operated on by the CE. Note that the LFB is a functionally accurate abstraction of the FE’s processing capabilities, but not a hardware-accurate representation of the FE implementation.

LFB Class and LFB Instance - LFBs are categorized by LFB classes. An LFB instance represents an LFB class (or type) existence. There may be multiple instances of the same LFB class (or type) in an FE. An LFB class is represented by an LFB class ID, and an LFB instance is represented by an LFB instance ID. As a result, an LFB class ID associated with an LFB instance ID uniquely specifies an LFB existence.

3. Comparisons between ForCES and OpenFlow

ForCES and OpenFlow are very similar in the following aspects:

- Both ForCES and OpenFlow are efforts to separate control plane from forwarding plane;

- Both ForCES and OpenFlow protocols standardize information exchange between the control and forwarding plane.

Although both ForCES and OpenFlow can be considered as the solutions for forwarding and control plane separation, they are different in many aspects. This section compares them in their goals, architecture, forwarding model and protocol interface.

3.1. Difference in Goals

The goal of ForCES is to break the closed box of network elements. After separation of forwarding elements and control elements, it is natural to define the standard, open communication interface between the two kinds of elements. By using ForCES, the two kinds of functional elements can be developed independently, provided both of
them implement the standard ForCES protocol. In this way, innovations of network devices can be speeded up.

In ForCES, there are two kinds of physical separations: blade level and box level [RFC3746]. In blade level, current network devices, e.g., routers, use proprietary interfaces for communication between CEs and FEs, so "the goal of ForCES is to replace such proprietary interfaces with a standard protocol" [RFC3746]. In box level, the CEs and FEs in one NE are physically separated boxes, all of which form one network element together.

The basic idea of OpenFlow is also separation of control plane and forwarding plane. But the goal of OpenFlow is beyond the new architecture of network devices. In fact, OpenFlow is a concrete implementation of the SDN idea, and the goal is to separate control plane from network devices and use a centralized controller to act as the control plane of the whole network. The controller can provide open APIs for users to add new features in the form of applications running on the controller. Such a separation simplifies the control functions and speeds up innovations in the network. That is just the idea of software defined networking. OpenFlow provides the standard protocol between OpenFlow controller and OpenFlow switches.

3.2. Difference in Architecture

ForCES proposes a new architecture for network devices (NEs). It separates control plane and forwarding plane in one network element and allows multiple instances of CEs and FEs inside one NE. ForCES protocol defines the standard communication interface between CEs and FEs. But in ForCES, network architecture remains unchanged [RFC3746]:

- The interfaces between two ForCES NEs are identical to the interfaces between two conventional routers;
- ForCES NEs can connect to existing routers transparently;
- ForCES still uses distributed protocols for control functions, e.g., routing protocols.

Figure 1 shows ForCES Architectural Diagram [RFC5810].
Compared to ForCES, OpenFlow changes the architecture of both network devices and even network itself. OpenFlow is very different from the current Internet architecture. In OpenFlow, network elements (OpenFlow Switches) only retain forwarding plane to forward packets by using flow tables, and provide open interfaces to the centralized controller. There is no need to run control functions such as routing protocols, signaling protocols, etc., in OpenFlow switches. The centralized controller serves as the control plane in OpenFlow networking. It will

- Collect the network view and make decisions according to control logics (or applications);
- Interact with OpenFlow switches to install their flow tables;

Figure 1: ForCES Architectural Diagram [RFC5810]
o Provide open APIs to users to add new features.

Using the terms NEs (Network Elements), FEs (Forwarding Elements) and CEs (Control Elements), the OpenFlow architectural diagram can be shown in Figure 2. In this architecture, the OpenFlow controllers can be multiple ones, which form the CEs in OpenFlow networking, although in most of current deployments, only one controller is used.

Fi/f: FE external interface

Figure 2: OpenFlow Architectural Diagram
by Using the terms NEs, FEs, CEs
3.3. Difference in Forwarding Model

In ForCES, [RFC5812] defines the FE (Forwarding Element) model based on an abstraction of Logical Functional Blocks (LFBs). In this model, each FE is composed of multiple LFBs that are interconnected in a directed graph, which is represented by the LFB topology model. Each LFB defines a single action of how to handle the passing packets. For example, typical LFBs include IPv4/IPv6 Longest Prefix Matching, etc. XML is used to describe LFB model formally.

In OpenFlow, the forwarding model is abstract to flow table manipulations. The current OpenFlow specification (version 1.1.0) [OpenFlow1.1] defines multiple flow tables structure in one OpenFlow Switch. Each flow entry contains three parts: match fields, counters and a set of instructions. Match fields are used to match packets. Counters are used for statistics of matching packets. If a packet is matched, it will be processed according to the instructions of the corresponding flow entry.

In OpenFlow networking, the controller controls the behavior of OpenFlow switches by adding, updating and deleting flow table entries in switches. OpenFlow switches process packets in the granularity of "flow": Match fields contained in each flow entry range from Ethernet source/destination addresses, IPv4 source/destination addresses, to TCP/UDP source/destination ports. In the current OpenFlow version, the possible actions can be output (which means forwarding a packet to a specified port), Set-Queue (which is used for Quality-of-Service support), Set-field, Push-Tag/Pop-Tag, Drop, etc. By associating various actions in a given order with each flow, OpenFlow controller can control the behavior of OpenFlow networking flexibly. However, in ForCES, the combination of multiple LFB instances with specified topology forms each FE. [LFB-Lib] has defined various LFB classes in LFBs base library, which can be used to implement functions of the current network devices. Compared to ForCES, the forwarding model of OpenFlow is more flexible. But it is more difficult to implement some current typical network functions in OpenFlow. OpenFlow can learn from ForCES to predefine some functional blocks to simplify the implementation of its applications.
3.4. Difference in Protocol Interface

ForCES defines two layers of protocols: ForCES Protocol Layer (ForCES PL) and ForCES Protocol Transport Mapping Layer (ForCES TML). ForCES PL defines protocol between FEs and CEs (Fp Reference Point). ForCES Protocol Transport Mapping Layer (ForCES TML) is defined to transport the PL messages. It is expected that more than one TML will be standardized and interoperability is guaranteed as long as both endpoints support the same TML. [RFC5811] has defined a SCTP-based TML for ForCES.

OpenFlow defines the protocol between controller and OpenFlow switches, i.e. OpenFlow protocol. Like ForCES, OpenFlow protocols also use TLVs (Type-Length-Value structure) formats for message encapsulation. For massage transportation, OpenFlow controller and switches communicate through a TLS/SSL connection or a TCP connection in the current version.

ForCES has longer history and is more mature than OpenFlow. OpenFlow can learn much from the protocol standardization of ForCES, for example:

- Defining Capabilities negotiation and configuration mechanism, just as ForCES can do by using LFBs;
- Defining Protocol Transport Mapping Layer to allow various standard transportation protocols.

4. Security Considerations

No security considerations.

5. IANA Considerations

No IANA considerations.

6. Conclusions

Both ForCES and OpenFlow follow the basic idea of separations of forwarding plane and control plane in network elements. This document analyzes the differences between OpenFlow and ForCES technically from the aspects of goals, architecture, forwarding model and protocol interface. From goals and architecture, OpenFlow is very different from ForCES. ForCES results in a new architecture of network devices but the current network architecture remains unchanged, while
OpenFlow results in a new NETWORK architecture, which is so-called SDN. In forwarding model and protocol interface, ForCES and OpenFlow are similar, but from the point of view of protocol standardization, ForCES are more mature and well-defined. OpenFlow can learn much from ForCES protocol in its standardization process.

At last, we can point out the potentials of ForCES protocol in SDN. Although ForCES is not designed for SDN, perhaps it can also be used to design a new protocol in SDN, because it is a well-defined communication protocol between CEs and FEs.

7. References

7.1. Normative References


7.2. Informative References


8. Acknowledgments

The authors would like to thank Susan Hares, who inspired us to write a Draft to indicate how ForCES compares with OpenFlow.
Authors’ Addresses

Zhiliang Wang
Network Research Center, Tsinghua University
Beijing 100084
P. R. China

Email: wzl@csnet1.cs.tsinghua.edu.cn

Tina Tsou (Ting Zou)
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: Tina.Tsou.Zouting@huawei.com

Jing Huang
Huawei

Email: james.huang@huawei.com

Xingang Shi
Network Research Center, Tsinghua University
Beijing 100084
P. R. China

Email: shixg@csnet1.cs.tsinghua.edu.cn

Xia Yin
Department of Computer Science and Technology
Tsinghua University
Beijing 100084
P. R. China

Email: yxia@csnet1.cs.tsinghua.edu.cn