Internet Draft S.Hares
Intended status: Informational Huawei
Expires: January 6, 2013
July 6, 2012

Analysis of Comparisons between OpenFlow and ForCES
draft-hares-forces-vs-openflow-00.txt

Status of this Memo

Copyright Notice

Abstract

While both ForCES and OpenFlow follow the basic idea of separations of forwarding plane and control plane in network elements, they are technically different. ForCES specification contains both a modeling language [RFC5812] which allows flexible definition of the Flow tables and flow logic. ForCES flow logic include Logical Functional Blocks (LFBs) connected in flow logic that is described in logic of direct graphs augmented by passage of Metadata and grouping concepts.

OpenFlow's specifications contain a specific instantiation of Flow tables and flow logic which has emerged from the research community theories. OpenFlow's logic varies based on the revision of the specification (OpenFlow-Paper [McKeown2008], OpenFlow Switch Specification 1.0 [OpenFlow1-0], OpenFlow 1.1 [OpenFlow-1.1] Open Configuration 1.0 [OpenFlowConfig-1.0]).

Table of Contents

1. Introduction

This document analyzes the differences between OpenFlow and ForCES technically from the aspects of goals, architecture, forwarding models, forwarding policy, control plane interaction, configuration of nodes, and applications (firewalls, load-balancers, High-availability nodes). This informational document compares OpenFlow and Forces as of March, 2012 seeking to provide clarity for other discussions of controller/forwarding split, Software Defined Networking, Software Driven Networking, Cloud Service Oriented networking (CSO), and a host of orchestrators for virtualized network devices.

A fellow Engineer provided inspiration for this deeper comparison by saying: "OpenFlow-0 is the Diff-Serv Tspec, OpenFlow-1.0 is Forces--, and OpenFlow 1.1 is Forces++." Jamal Salim suggests Open Flow 1.1 does not have the same functionality[Jamal-01].

While this summary brings the expert listener quickly into heart of the issues, this document examines:

- Is OpenFlow Switch 1.1.0 really "ForCES++", and "is the group table safe ++ logic? What direction does Open Flow Switch 1.2 and 1.3 take us?

- Where does Open Flow’s Config fit in the picture?

- How does this help us get to Clouds Service Oriented Networks (CSO) enable by Software defined networks (SDN) or software driven networks (SDN)?

And that, as the saying goes is the "rest of the story" of this draft. Here’s hoping the readers of this document will decide and argue with the author to refine the next-generation of hardware devices.

1.1. ForcES Introduction

ForCES (Forwarding and Control Element Separation) work in IETF has defined a new environment to build network devices that split the network devices into control plane and forwarding plane into units. For example, a router could be considered a network element (NE) with a control plane running router protocol and a data plane forwarding IP traffic.

The drive to have ForCES NE device split arose from the desire to build hardware forwarding blades out of flexible hardware components. These hardware devices included Network Processors and network specific ASICs.

The ForCES environment defines requirements [RFC3654], goals [RFC3565], architecture and protocol requirements [RFC3654], a controller-forwarder communication protocol [RFC 5810]. ForCES also describes a policy on how to building the forwarding engine out of a set of logical functional blocks (LFBs)which are connected as a directed graph [RFC5812]. ForCES allows many different Forwarding Engines (FE) to linked to Controller Engines (CE) via the protocols. ForCES provides a modeling language [RFC-5812] to describe these FE devices so that controllers can load control the devices, load forwarding tables, and keep track of statistics. ForCES RFCs also define how the ForCES protocol runs over SCTP [RFC5811.

1.2. OpenFlow Introduction

OpenFlow[McKeown2001, p. 1]] arose out of the frustration that network research projects felt at not being able to experiment with new protocols on large-scale networks. Experimentation on research networks did not have a large enough scale to provide a reasonable test-bed for new research ideas for the Internet. Pure commercial networks would not allow experimental protocols, and commercial router vendors took 3-5 years to create a new protocol features. The OpenFlow researchers suggested an alternative to allow the research to creating a slice out of commercial network to try out new ideas for network.

OpenFlow’s initial paper grew into OpenFlow Switch Specification versions 1.0 [OFS-1.0], 1.1 [OFS-1.1], 1.2[OFS-1.2], 1.3[OFS-1.3], and (likely) 1.4 [OFS-1.4]. Additionally a Config and Management Protocol version 1.0[OFC-1.0] and version 1.1 [OFC-1.1], and a set of papers and application notes on implementations. A hybrid Specification [OFHy-1.0] suggests how Open Flow may be combined with existing network OpenFlow switches which mix existing network devices (routers/switches) with OpenFlow controlled switches in either a Ships-in-the Night (SIN) or a hybrid model

OpenFlow’s host of specifications and ForCES flexible reprogramming of the network element architecture can be considered the first wave of Software-Defined Networking (SDfN) where software can alter how the forwarding engine’s logic operates. This work arises out of the GENI research [GENI][McKeown2008]. The current industry discussion regarding Software-Defined Networking (SDfN) or Software-Driven Networking (SDrN), Network Virtualized Overlays [NVO3], Service-Oriented Protocols (SoP)[SoP] or network orchestrators of Cloud Service Oriented Network (CSO)forwarding [CSO-Arch] have a great deal confusions as they apply the terms to ForCES and OpenFlow. Rather than carefully define each of these terms explicitly at the outset, we will give brief expansions of the abbreviations and return to the definitions later in the draft after examining the FORCES draft.

This document will examine ForCES and OpenFlow goals, architecture, forwarding conceptual models, Controller-forwarder communication mechanisms and protocols, the policies in the loading of forwarding state, configurations of nodes, and sanity checking of the forwarding.

These basic concepts will be then examined in terms of specific implementations (switch, hybrid router/switch, wireless, load-balancer, firewall) as described by ForCES and OpenFlow reports.

Finally, the document will return to defining S[*]N, SOP, and Cloud-Oriented Services (CSO).

2. Definitions

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 RFC2119 definitions used in this document are:

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].

ForCES definitions relevant to this documents discussion are taken from [RFC3654][RFC3746][RFC5810] as noted below. The quoted italized definitions come from the ForCES RFC, and the non-quoted text applies ForCES RFC text to this document.

2.1. New Common Configurations

Controlling Entity (CE): is defined as an entity which remote controls the forwarding engine. This Entity can be either a ForCES CE or a Open Flow Controller.

Controlling Entity Manager (CE-MGR): This documents loosens the ForCES CE-Manager definition to allow Open Flow and ForCES to be compared. This document defines the CE manager as a logical entity (distributed or located in physical or virtual device) which controls which controllers attach to which logical Forwarding Entities. The Controllers can be in the same physical switch/device in the control plane or other logical software. A CE-Mgr may also be within a VM hypervisor, a VM hypervisor manager, or other virtual software. The CE-Mgr logical function may be distributed across many CE as a defined function. This definitional Allows both ForCES CE-Mgrs and Open Flow Controller collaboration/management via coordinated remote configuration of OF Capable Switches.

Controlled Router-Switch (CRSW): A Controlled Switch is a network entity performing switching capability that is controlled by remotely by either the ForCES protocol (FP) or the Open Flow Protocol (OFP). This switch can perform IP routing, MPLS switching, Trill Switching or Layer 2 Switching.

Forwarding Entity (FE): is defined as an entity which forwarding packets or frames under the control of the CE. This entity can be an ForCES FE (F-FE) or an Open Flow Capable Switch (OF-CS). An Open Flow Capable switch can either be a hybrid switch or a Open-Flow Only switch.

FE Manager (FE-MGR): The FE-Manager controls the FEs assignments. This document defines FE-Manager’s logical entity may be a logical software process residing within local switch/device in the control plane or management plane. The FE-Mgr can also within a VM hypervisor, or a VM hypervisor manager, or other virtual software. The FE-Mgr can be a remote service managing the forwarding engine. The Open Flow Configuration [OFC-1.0] Configuration Service point with its logical configuration function may also have a FE-MGR function. This FE-Mgr capability is an capability outside the [OFC-1.0] specification.

Pre-Association Phase (Pre-A): This document defines a Pre-Association Phase (Pre-A) as the period during which a CE-Management(Forces CE-Mgr or OF controller groups) and FE-Managers (Forces FE-MGR (F-FE-MGR)or OF-CS management) determines which Controlling entity (CE)controls which Forwarding Entity (FE).

2.2. Forces Definitions

Force Forwarding Element (F-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." [RFC3654] ForCES forwarding FE supports forwarding rules insertion.

ForCES Control Element (F-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." [RFC3654] The ForCES CE controller may be located within the same hardware box on a different blade or across an Ethernet connection, or across a L3 Link (if security used).

ForCES Network Element(F-NE)- "An entity composed of one or more CEs and one or more FEs. To entities outside a NE, the NE represents a single point of management. An NE usually hides its internal organization from external entities and represents a single point of management to entities outside the NE." [RFC3654] The NE’s single point of management can be at the IP layer, the Ethernet layer, and at a virtual layer. In this document, the network element is examined as being the set of network functions in the hardware that collaborates to act like a switch. This less strict definition allows ForCES to be compared with the Open Flow work.

ForCES Pre-Association Phase (F-Pre-A): ForCES defines the Pre-Association Phase (F-Pre-A) as "the period of time during which a FE Manager(see below)and a CE Manager (see below) are determining which FE is a part of the network element" [RFC3654].

FE Manager(F-FE-Mgr)- ForCES (F-FE-Mgr)is "A logical entity that operates in the Pre-Association Phase and is responsible for determining to which CE(s) a FE should communicate. This process is called CE Discovery and may involve the FE manager learning capabilities of available CEs." [RFC3654]

CE Manager (CE-Mgr) - Forces CE-MGR[F-CE-Mgr] is "A logical entity that operates in the pre-associaation phase and is responsible for determining to which FE(s) a CE should communicate. This process is called FE discovery and may involve the CE manager learning the capabilities of available FEs. The CE manager may use anything from statics configuration to a pre-association phase protocol." [RFC3654]

ForCES Protocol (ForCES-Proto) - "While there may be multiple protocols used within the overall ForCES architecture, the term "ForCES protocol" refers to only the post-association phase protocol." [RFC3654] The ForCES protocol operates between the "Fp reference points" of the ForCES architecture (as shown in figure 1) [RFC5810]. "Basically, the ForCES protocol works in a master-slave mode in which the FEs are slaves and the CEs are masters." [RFC5810] The location and exact instantiation of the CE logical entities associated with the FE logical entity is flexible. The CE entities could reside on a process on a local switch communicating to other process off the local switch.

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)" [RFC5810] This layer is defined in 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." [RFC5810, p. x]. The ForCES TMLs focused on are STCP [STCP] and SSL[SSL]. TM handles transport of messages [reliable or non-reliable], "congestion control", "multicast", ordering, and other things [RFC5810, p. 14].

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.

Physical Forwarding Element - The physical element that forwards the packets.

2.3. Open Flow Definitions

Open Flow (OF): [McKeown-xx] defines OF as a "way for researchers to run experiments in networks they use every day"[McKeown-2008, p.1].

Open Flow Action [OF-Act]: [OF-1.1.0] defines an OF-Act as an action that may: "forward the packet to a port", or modifies the packet". Actions may be specified in "Open Flow Instruction" [OF-Inst] in Flow Table Entry or "action buckets in Group Table Entry"[OF-1.1.0,p.4].

Open Flow Action Set [OF-ActSet]: [OF-1.1] defines an OF-ActSet as "a set of actions associated with the packet that are accumulated while the packet is processed by each table, and are executed when the packet exits the processing Pipeline."[OF-1.1.0, p. 5].

Open Flow Capable switch [OF-CS]: [One or more physical or virtual switch device which can act as an operational context for an Open Flow Logical Switch [OF-LS]. A OF-CS hosts an Open Flow Data Path [OF-DP].

Open Flow Configuration and Management Protocol [OFCMP]: [OFC-1.0] states the [OFCMP-1.0]enables the remote configuration of Open Flow Data Path (OF-DP). The OFCMP allows a controller to configure the OF-DP on the Open Flow Logical Switch (OF-LS) "so that the controller can communicate and contro" the OF-LS via Open Flow Protocol (OFP). The OFCMP allows dynamic association of resources with OF-LS in an OF-CS. [OFC-1.0] defines the protocol, but not the resource allocation mechanisms.

Open Flow Configuration Point (OFCPT): [OFC-1.0] defines an OFCPT as "a service" which sends OFCMP messages to a OF-CS with an OF-LS inside.

Open Flow Controller [OF-CTLER]: [McKeown-2008] defines the OF-CTLER as a "controller that adds and deletes Flow entries on behalf of experiments" [McKeown-2008, p. 3].

Open Flow Datapath [OF-DP]: [OFC-1.0] defines OF-DP as an abstraction called Open Flow Logical Switch [OF-LS].

Open Flow Dedicated Switches [OF-DS]: [McKeown-2008] defines OF-DS as a "dumb datapath element that forwards packets between ports as defined by a remote process" [McKeown-2008, p3.] The Open Flow process programs the forwarding engine for this dumb datapath switch.

Open Flow Enable Switches [OF-ES]: [McKeown-2008] defines OF-ES as "commercial switches, routers or access points" enhanced by adding the OF feature

Open Flow Feature [OF-Feature]: Open Flow[McKeown2008] and [OF-1.0.0] defines the OF-Feature as adding the features of an Open Flow Logical Switch [OF-LS]. These features are the Open Flow "Flow Tables", "Secure Channel that connects the switch to the controller", and "the Open Flow Protocol" [McKeown-2008, p. 3][OF-1.0.0, p.2].

Open Flow’s Flow Table (OF-FT): [McKeown-2008] defines a Flow table in OF as having "an action with every flow table entry to tell the switch how to process the flow" [McKeown-2008, p. 2]

Open Flow’s Flow Table Entry (OF-FTE): [McKeown-2008], [OF-1.0.0], [OF-1.1.1], [OF-1.1.2], and [OF-1.1.3] define the specific of an single entry in a flow table. See Section x.x for a detailed comparison of this entry.

Open Flow Group (OF-G): [OF-1.2] defines an OF group (OF-G) as a list of Open Flow "action buckets" and "a means to choose one or more buckets to apply on a per-packet basis" [OF-1.2, p. 5].

Open Flow Group Table (OF-GT):

Open Flow Logical Switch[OF-LS]: OFC-1.0 defines the OF-LS as an abstraction of the "open flow datapath".

Open Flow Packet (OF-Pkt): [OF-1.1.0] defines OF-Pkt as "ethernet frame including header and payload" [OF-1.1.0, p. 4].

Open Flow Pipeline [OF-PipeLine]: [OF-1.2] defines OF-Pipeline as "a set of linked tables that matching, forwarding, and

Open Flow Port [OF-Port]: [OF-1.2] defines an Open Flow port as a place where packets enter and exit the Open Flow Pipeline.

Open Flow Protocol (OFP): OF 1.0 defines "an open protocol to program the flow table in switches and routers" in which "a controller communicates with a switch"[McKeown-2008,p. 2-3].

Open Flow Switch (OFS): [McKeown-2008] defines an Open Flow Switch as a ethernet switch with "at least" the following three functions: "(1) a Flow Table","(2) "a secure channel that connects the" controller with the switch over which the open flow protocol runs, and (3)an "open flow protocol" [McKeown-2008,p 2.][OF-1.0.0,p.2].

[OF-1.1.0] defines an OFS as "one or more flow tables and a group table which perform packet look-ups and forwarding", and an open flow channel to an external controller"[OF-1.1.0,p.3]. The external controller controls the switch via the Open Flow protocol (OF-Proto)[OF-1.1.0, p.3]. [OF-1.3.0] adds that the "switch communicates with the controller, and the controller controls the switch"[OF-1.3.0, Section 2, paragraph 1.]

3. Comparisons between ForCES and OpenFlow

ForCES and OpenFlow are very similar in the following aspects:

- o Both ForCES and OpenFlow are efforts to separate control plane from forwarding plane;
- o 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 history, goals, architecture, forwarding model and protocol interface.

3.1. Difference in Historical setting

ForCES work began during the 1995-2000 timeframe with the development of netlink [RFC3549]. The linux netlink began its linux driver history as first a "character device /dev/netlink for Linux kernel 1.3.31" but was superceded by "Alexey Kunzetsov's socket-based af_netlink.c in Linux v 2.1.15" [Englehardt-2010]. The rtnetlink brought configuration and router queries to links. The netlink socket allowed messages between kernel and user space regarding routes, firewalls, and monitoring.

The historical context of the 1995-2002 timeframe saw the initial growth of the US Internet and spread into non-US sites. The historical changes included changes that began the split of tight binding of control plane to data plane. Forwarding plane elements (ASICs, TCAMs, Network processors) and backplanes began the growth of non-stop forwarding with high-availability. ForCES notes that the data processors have "forwarding tables", "per-flow Qos Tables and access control lists" [RFC365]. ForCES had control processors were general-purpose processors that support route calculations.

ForCES began in quasi-commercial realm of linux development where linux developers used routing software with netlink to build early Internet networks. Alexey's early work was deployed in Russian and European networks to turn linux boxes into Routers. By early 2000, this work had migrated to router boxes seeking to harden routers to provide non-stop forwarding. Netlink implementations were provided with many commercial OEM standards for switches and routers.

ForCES work came out of the desire to expand the basic netlink protocol into an architecture that allow quick modeling of new forwarding hardware and an extensible control-plane to forwarding plane communication. Early discussions in ForCES look to allow coordination of multiple control planes as well as control plane to forwarding plane functionality. However, the IETF decision was to restrict the first versions of ForCES to architecture, CE-FE communication and FE modeling.

OpenFlow arises out of the academic's communities realization that the hardening of commercial of network infrastructure [1995-2006] to support businesses, caused a "reluctance to experiment with production traffic"[McKeown-2008, p. 1]. The GENI (Global Environments for Network Research)[2006-2007] suggested that: a)Internet's infrastructure faced "serious problems" in "security, reliability, manageability, and evolvability" and "possible solutions" existed in research, but there were "severe experimental barriers" to test new ideas in wide-spread deployments [GENI-2007, p. vii]. A new US research network would allow slices of routers to be used for researcher's experiments in network protocols. McKeown and colleagues work examined how these experiments could be extended to run [McKeown-2008] on local campuses. McKeown and colleagues examined persuading commercial routers to provide an open interface for experimentation or using existing open source solutions (linux, XORP[XORP-2008]). Their conclusion was that "commercial solutions are too closed", and "research solutions have insufficient performance or fanout, and are too expensive." [McKeown-2008, p. 2].

3.2. Difference in Goals

RFC3654 lists the the architectural goals of ForCES. OpenFlow Switch (OFS) Specification version 1.0.0[OFS-1.0.0] and OFS version 1.1.0 [OF-1.1.0] refer to [McKeown-2008] as the basis of these specifications. This document’s goal comparison compares the four goals [McKeown-2008] sets against [RFC3654].

The goal ForCES is to define the "architecture", "architectural modeling" and protocols to "logically separate control and data forwarding plans of an IP (IPv4, IPv6, etc.) networking device" [RFC3654, p. 1]. ForCES network device (aka. network element (NE)) can be a router, IP switch, firewall, switch. ForCES redefines network elements to be logical relationships placed on physical devices.

McKeown et al. state the goals of the OpenFlow approach was to find "high-performance and lost-cost implementations" of new network algorithms, capable of being used in "broad range of research", adaptable to commercial "close platforms", and able to "isolate experimental traffic from production traffic [McKeown-2008, p. 2].

Difference in goals underscores the original commercial focus of ForCES and the experimental focus of OpenFlow.

3.3. Difference in Architectural Requirements

Architecture sets the building blocks for system, and architectural requirements sets rules for interconnecting the building blocks.

Building blocks for ForCES include the CE(s), FE(s), ForCES protocol (CE to/from FE), FE-Manager, CE-Manager, and logical input flows. Within the FEs there are Logical Forwarding Blocks connected together in a directed graph. The flow processing passes along input port, (modified)frame, metadata (which may include actions). The flow stream may be output to interfaces (logical or physical).

Building blocks for OpenFlow include Controllers (˜CEs) and Forwarding Units (˜FEs) with OpenFlow processing. OpenFlow logic is designed in terms of Flow Processing controlled by Flow Tables (McKeown-2008][OFS-1.0][OFS-1.1], and Group tables [OFS-1.1]which operate on the modified frame, metadata, or group of actions via actions or instructions (a group of actions and forwarding commands).

Both flow streams Flow processing may cause the flow to multiple into several streams or combine multiple streams into one.

3.3.1. ForCES System Building Blocks

The building blocks within the ForCES architecture are the CEs (controller elements), FEs (forwarding elements), and an interconnect protocol between CE(s) and FE(s). ForCES also recognized the logical functions of a FE-Manager and a CE-Manager. Figure 1 shows a diagram from RFC5810 that details interaction between all these components.

The ForCES CE controls switching, signaling, routing, and management protocols. Each CE is a logical unit which may be located within the same box, different boxes, or across the network. ForCES architecture [RFC3746] allows CEs to control forwarding in multiple FEs.

ForCES defines logical Forwarding Elements (FEs) that reside on a variety of physical forwarding elements (PFE) such as a "single blade (PFE)", partition within blade, or multiple PFEs in a single box, or among multiple boxes [RFC3746, p. 2]. The ForCES logical FEs could also be run within Virtual Machines (VMs) within a single box or a set of boxes or a cloud. A single FE may be connected to multiple CEs providing strong redundancy. FE internal processing is described in terms of Logical Forwarding Blocks (LFBs) connected together in a directed graph that "receive, process, modify and transmit packets along with metadata"[RFC5810, p. 6]. The FE model determines the LFBs, the topological description, the operational characteristics, and the configuration parameters of each FE.

The Forces Logical Forwarding Block (LFBs) Library [FORCES-LFB] provides the class descriptions for Ethernet, IP Packet Validation, IP Forwarding LFBs, and Redirection, MetaData, and Scheduling. Forces-LFB document demonstrates how these logical blocks can be placed within a machine to support IP Forwarding (IPv4/IPv6)for unicast & multicast and ARP processing[Forces-LFB, p. 17].

ForCES architecture [RFC3746] allows CEs to control forwarding in multiple FEs. ForCES also recognized the logical functions of a FE-Manager and a CE-Manager. The FE manager determines the CE(s) each FE should communicate with. The CE manager determines which FEs each CE should communicate with. The ForCES defines the FE-Manager and CE-Manager to operate in a "pre-association" phase of the communication to set-up the FORCES communication path. Similarities between the functions of the CE-Mangers and FE managers of ForCES and modern hypervisors may come from the creative interplay of early open source communities [1995-2005]. Applications directly interacting with ForCES components (CEs or CE-Managers or FE-Manager) could be described as interactions with the CEs or CE-Managers.

Figure 1 shows ForCES Architectural Diagram [RFC5810].

```
                            ---------------------------------------
                            | ForCES Network Element              |
     --------------   Fc    | --------------      --------------  |
     | CE Manager |---------+-|     CE 1   |------|    CE 2    |  |
     --------------         | |            |  Fr  |            |  |
           |                | --------------      --------------  |
           | Fl             |         |  |    Fp       /          |
           |                |       Fp|  |----------| /           |
           |                |         |           |/            |
           |                |         |           |             |
           |                |         |    Fp    /|----|        |
           |                |         |  /--------/      |        |
     --------------     Ff  | --------------      --------------  |
     | FE Manager |---------+-|     FE 1   |  Fi  |     FE 2   |  |
     --------------         | |            |------|            |  |
                            | --------------      --------------  |
                            |   |  |  |  |          |  |  |  |    |
                            ----+--+--+--+----------+--+--+--+-----
                                |  |  |  |          |  |  |  |
                                |  |  |  |          |  |  |  |
                                  Fi/f                 Fi/f
```

Fp: CE-FE interface
Fi: FE-FE interface
Fr: CE-CE interface
Fc: Interface between the CE manager and a CE
Ff: Interface between the FE manager and an FE
Fl: Interface between the CE manager and the FE manager
Fi/f: FE external interface

Figure 1: ForCES Architectural Diagram [RFC5810]

3.3.2. OpenFlow Building blocks

OpenFlow architecture consists of set of OpenFlow Switches with a Flow Table, a Secure Channel between controller and switch, and an Open flow protocol. OpenFlow switches can either be only controlled by OpenFlow, enabled by Open Flow [OFS 1.0] (shared), or hybrid Switches (OFS 1.2).

```
               ----------------  ----------------
                | Application1 |  | Application2 |  ......
                ----------------  ----------------
                        |     APIs       |
                 ----------------------------------------
              CE | ---------------      --------------- |
                 | |  OpenFlow   |------|  OpenFlow   | |
                 | | Controller  |      | Controller  | |
                 | ---------------      --------------- |
                 ----------------------------------------
                      |              |             |
                      |      OpenFlow Protocol     |
                      |              |             |
               NE&FE  |              |             |     NE&FE
               --------------        |          --------------
               |  OpenFlow  |        |          |  OpenFlow  |
               |   Switch   |        |          |   Switch   |
               --------------        |          --------------
                 |  |  |  |          |            |  |  |  |
                 |  |  |  |          |            |  |  |  |
                 |  |  |  |          |            |  |  |  |
                   Fi/f   |    NE&FE |            |  Fi/f
                          |    --------------     |
                          |    |  OpenFlow  |     |
                          |    |   Switch   |     |
                          |    --------------     |
                          |      |  |  |  |       |
                          |      |  |  |  |       |
                          --------  |  |  --------
                                    Fi/f

            Fi/f: FE external interface
```

Figure 2: OpenFlow Architectural Diagram
by Using the terms NEs, FEs, CEs

The Flow table provides entries on how to process a flow whose header fields match a pattern in the header field [OFS-1.0.0] or a set of meta data generated from pipeline processing of a header[OFS-1.1.0, figure 4][OFS-1.3].

The matching of a packet in OFS-1.0.0 based on first exact match of header and/or meta data, and secondly wild-card entries. The wild-card entries contain a priority field to order the process of matching. For example, a priority of "1" will be the first wild card processed.

3.3.2.1. Match Fields in OFS

The match field has been expanding as the ONF specifications evolve - from a 10-tuple [McKowen-2008], to 12-tuple [OFS-1.0], and to a OFS-1.1.0 a 15-tuple, to 39-tuple in OFS-1.3[OFS-1.3] (14 required and 25 optional).

The original 10-tuple includes ingress port, VLAN ID, Ethernet source address, destination address and type, the IPv4 source/destination address, and IP protocol, TCP/UDP source & destination port. The 12-tuple adds VLAN priority, and IP Tos bits. The 15-tuple adds: metadata, MPLS label, MPLS traffic class. The TCP/UDP source & destination port have redefined for ICMP packets to have instead he the ICMP Type and ICMP code. [OFS-1.3-pre]required matches include the 10-tuple plus IPv6 source and destination addresses and UDP source and destination ports.

3.3.2.2. Flow Logic - Flow Table and Group tables

The flow table operation and data structures vary based on the version of OpenFlow Switch specification.

OFS in versions [McKeown-2008] and [OFS-1.0.0] operate on logic in flow tables which are executed in ascending order. Each Flow Table ID must be greater than the current Flow table id. [OFS-1.1.0][OFS-1.3] flow logic operates on flow tables and group tables which allow jumps based on "Goto table" logic or combinations of flows.

```
|=============|    |==============|      |=============|

|Flow table 0 |---.|Flow Table 1 | _ -. |Flow Table n |

|=============|    |==============|      |=============|
```

Figure 3: Flow Table [OFS-0.9][OFS-1.0]

All OFS Flow tables match on data and perform actions [OFS-0.8][OFS-0.9][OFS-1.0][OFS-1.1][OFS-1.2][OFS-1.3]. Later versions [OFS-1.1.0][OFS-1.2][OFS-1.3] use instructions to immediately perform actions or to queue specific actions for later processing. These same later versions also allow metadata to be stored to be passed along to additional processing.

```
                |----------------|

                | Group table 1  |

       |------. |action-bucket1 |----. egress-port-1

       |        | action-bucket2 ]

       |        |----------------|

       |

|=============|    |==============|      |=============|

|Flow table 0 |---.|Flow Table 1 | _ -. |Flow Table n |

|=============|    |==============|      |=============|
```

Figure 3: Flow Table [OFS-1.1]

1) Action Specifics

[McKeown-2008] has three actions in the flow tables: forwarding of a matched packet to a specific port or ports, sending the packet to the controller, or dropping the packet. This simply processing is why some engineers suggest that OFS-2008 is similar to the RSVP T-Spec [RFC2870].

[OFS 1.0.0] actions direct switch processing to forward packet, drop packet, enqueue packet (optional), and modify-field in packet (optional). Forwarding packets can be sent to all ports, the controller, local switches forwarding stack, send out input port, and specific ports after performing table actions & send out specified port, send via normal (L2/L3/VLAN) processing, and flood (via minimum spanning tree) ports. This causes some engineers to consider OFS 1.0 equivalent to be Forces--.

[OFS-1.1.0] uses instructions within each flow entry to determine how a packet and associated data is processed. These associated data includes: ingress port packet came in on, the generated meta-data and an action set. The action set is a set of commands to execute prior to sending the packet out. The [OFS-1.1.0] instructions are: "Apply-Actions", "Clear-Actions", "Write-Actions", "Write-metadata", "Goto Table" [OFS-1.1.0, p. 14]. Actions are either applied immediately with apply action command, or stored (via an Write-Action)in action sets for later processing in the action-set via a Write-Action. The existing actions can be cleared with a "Clear-Action". [OFS-1.1.0] actions are output packet, set queue, drop packet, process via group table, push/pop tags, and set-field. [OFS-1.1.0] action sets include single entries for any of the following: copy TTL inward in packet, pop/push actions to packet, copy TTL outwards, decrement TTL, set fields in packet, apply QoS Actions (e.g. set queue), and apply group actions, and output packet.

2) Flow Logic Encoding

[OFS-1.0.0] encodes the ForCES LFB in table sequences. The LFB directed graph of ForCES modeling is encoded in sequences of Flow Table. OFS 1.0 specifies that the Ethernet header (similar to ForCES Ethernet II) is the basic frame for all input. A fixed processing ARP, IPv4, TCP/UDP, and ICMP packets are specified based matches beyond the Ethernet header. The Forces LFB library provides building blocks for matches beyond the Ethernet to ARP, IPv4 and IPv6 packets, and Meta data. The OFS-1.0.0 does not have Metadata.

[OFS-1.1.0] match of a flow goes against specific table 15-tuple header. If the frame/packet matches, the flow table can alter the packet (via immediate actions), add metadata (for later handling), set an actions in action set, pass the processing to a specific table (Flow Table or Group Table), or pass the processing to the next table in the sequence. The information passed on to the next processing is [ingress port, (post-modification) frame/packet, metadata, and action set.

[OFS-1.1.0] allows processing between tables to carry the ingress port, the packet, generated metadata, and an action set. An action set is a set of commands to execute prior to sending the packet out. [OFS-1.1.0] uses Flow table with instruction. The instructions can be which can be "Apply-Actions", "Clear-

Actions", "Write-Actions", "Write-metadata", "Goto Table" [OFS-1.1.0, p. 14]. Actions are either applied immediately with apply action command, or stored (via an Write-Action) for later processing in the action-set via a Write-Action. The existing actions can be cleared with a Clear-Action.

[OFS-1.1.0] supports two types of tables: flow tables, and group tables. The group table structure has a group identifier, group type, counters, and an order list action buckets. Action buckets contain a set of actions with parameters. If the group table has "zero" action buckets, then no processing occurs.

[OFS-1.1.0] group type field specifies how the action buckets operate on the packet. The group can be "all", "select", "indirect", and "fast-failover" [p.7]. The "all" bucket provides multicast and/or broadcast support execute all buckets. The packet is effectively cloned and sent out. The select, indirect, and fast-failover execute one bucket. In select, the switch determines which port via internal algorithm (e.g. round-robin). In "indirect", the group table logic selects the bucket. The "fast-failover", the first live bucket is chosen. The single-bucket choses may restrict flows if the specified buckets and/or output ports are down.

This new sequential logic is the basis of some engineers comment that OpenFlow 1.1 is ForCES++.

ForCES people indicate that that the group table logic plus "Go to" logic is simply a LFB model of a specific type. [Haleplidis-2012] demonstrates on the LFB library concept can be used to capture all of the OFS-1.1.0 specification.

3.3.3. ForCES FE types

ForCES and OpenFlow FEs can operate either new switching entities or integrated with existing processing as a hybrid. In OFS-1.2, the Ships-in-the-Night (SIN) mode divides existing ports into groups controlled by specific ports (see figure x) or VLANs (figure-x)

```
|=============|    |==============|      |=============|

| standard    |    | Open Flow    |      |  Forces     |

|   CE        |    | controller   |      |    CE       |

|=============|    |==============|      |=============|

     ||                 ||                    ||

     ||                 ||                    ||  (forwarding)

|------------------------------------------------------------

|  |==========|    |==============|      |=============|    |

|  | standard |    | Open Flow    |      |  Forces FE  |    |

|  | FE board |    |  FE          |      |             |    |

|  |==|==|====|    |====|==|======|      |====|===|====|    |

|-----|--|-------------|--|-----------------|---|----------

Standard ports         OFS ports         ForCES ports
```

Figure x - Hybrid mode per port

```
   |=============|    |==============|      |=============|
   | standard    |    | Open Flow    |      |  Forces     |
   |   CE        |    | controller   |      |    CE       |
   |=============|    |==============|      |=============|
        ||                  ||                     ||
        ||                  ||    (forwarding)     ||
   |------------------------------------------------------------|
   |  |==========|    |==============|      |=============|    |
   |  | standard |    | Open Flow    |      |  Forces FE  |    |
   |  | FE board |    |  FE          |      |             |    |
   |  |==|==|====|    |====|==|======|      |====|===|====|    |
   |     |  |              |  |                 |       |      |
   |  vlan1 vlan10     vlan2  vlan15         vlan3   vlan7     |
   |   | |   | |         | |   | |            | |   | |        |
   |---|-|---|-|---------|-|---|-|------------|-|---|-|--------|
   S      OFS ports         ForCES ports
```

Figure x - Hybrid mode per port

3.3.4. ForCES Pre-Association

Neither the ForCES protocol nor the OFS protocols [McKeown-2008/OFS-0, OFS-1.0.0, OFS-1.1.1] specify how the CEs/controllers and FEs/forwarding switch meet.

Do CEs go to the FEs meeting place and CEs pick up the FE that delights their forwarding fancy? How do they found out where eligible FEs are meeting? Do FEs have choice on which CEs select them, and if so what are the criteria?

The forming of intimate relationships between CEs and FEs remains to the readers of the specification as mysterious as the pre-association stages of human group dating or clique forming.

ForCES specifications specifically call this phase a pre-association phase. The ForCES architecture names the entity that coordinates the forming of associations (like a Jewish match-maker) for CEs and for FEs. The CE-Manager determines which FEs each CE should talk to. The FE-Manager determines which CEs each FE should talk to. Only when the associations between each CE and its FEs, and each FE and its CEs are complete, does the system complete pass from pre-association phase to association phase.

It is assumed that some protocol interactions within the logical ForCES network entity (or entities) determine how CEs will coordinate their work. However, the IETF work specifically denoted this CE-CE coordination work as a second phase of work.

OpenFlow Switch specifications ([McKeown-200][OFS-0.8][OFS-0.9][OFS- 1.0] OFS-1.1] ignore the concept of this dynamic meeting processing.

Either OFS specification missed this concept. Perhaps the OFS specifications assumed a static configuration as part of a boot process of the hybrid switch will set-up some basics. It is possible that the lack specification may come from the sponsors of the specification wanting proprietary pre-association interactions. If so, this provides an interesting line of demarcation between standards and OFS standard.

In any case, this oddity from the OFS proponents leaves one to ask "What is the rest of the story on the pre-association phase?"

3.3.5. Architectural requirements

RFC3654 specifies 15 architectural requirements. Table 15 provides summary of this requirements and possible OFS (McKeown-2008/OFS 0, OFS 1.0, OFS 1.1).

ForCES Architectural Requirement	OpenFlow Switch Architecture
1. CE/FEs connect via variety of communicate interconnect technologies. [RFC3654, p. 5]	Controllers/switch over a secure connection [McKeown-2008][OFS-1.0][OFS-1.1]
FE can use different technology than CE/FE topologies.	not specified
2. "FEs MUST support a minimal set of capabilities necessary for establishing network connectivity (e.g. interface discovery, port up/down functions.) Beyond this minimal set, the ForCES architecture MUST NOT restrict the types of numbers of capabiliti8es that FEs may contain. [RF3654, p.5]	[McKeown-2008][OFS-1.0][OFS-1.1] specify a set of required actions, instructions, Flow actions, and an implied set of port functions(e.g.interface discovery, port up/down). These OFS specifications also declare some set of features optional.
3. "Packets MUST be able to arrive at the NE by one FE and leave the NE via different FE." [RFC3654, p.5]	[OFS-1.0][OFS-1.1]specifies in ofp_port the OFPP_IN_PORT flag which allows the port to explicitly send it back out the input port [OFS-1.0, p. 18] [OFS-1.1, p. 26]

ForCES Architectural Requirement	OpenFlow Switch Architecture
4. "A NE must support the appearance of a single functional device." (e.g. single IP TTL reduction.) [RFC3654,p. 5]	[OFS-1.0][OFS-1.1] describes the devices as an individual switch, and provides the capability to reduce TTL [OFS-1.1,section 4.7, p. 12] such as push/pop of VLAN IDs and/or MPLS headers [OFS-1.1, p. 12]
4b. However, external entities (e.g. FE managers and CE managers) MAY have direct Access to individual ForCES protocol elements for providing information to transition [RFC3654, p. 5]	4b. No pre-association logic has been defined.
5."The architecture MUST provide a way to prevent unauthorized FORCES protocol elements from joining an NE." [RFC3654, p. 5]	Beyond a secure channel between some "controller entity and switch" no prevention of unauthorized access has been encoded.
6. A FE must be able to asynchronously inform the CE of a failure or increase/decrease in available resources or capabilities on the FE.	[OFS-1.0][OFS-1.1] have "three message types: controller-to-switch, asynchronous, symmetric [OFS-1.0, p. 10] [OFS-1.1, p. 16]. Asynchronous messages

ForCES Architectural Requirement	OpenFlow Switch Architecture
	Switches send a controller (CE) Messages with a received packet (packet-in), notification of a removed flow table entry (flow-removed), changes in port status (port-status), and error conditions (error) [OFS-1.0, pp-10-12)][OFS-1.1, p. 16-17]
"Thus, the FE MUST support error monitoring and reporting" (e.g. "number of physical ports Or "memory changes"). [RFC3654,p. 5]	The change in port status is covered,but memory status is not covered
7. "The Architecture MUST support mechanisms for CE redundancy or CE failover.	[McKeown-2008], [OFS-1.0] [OFS-1.1]do not specifically provide CE Redundancy or CE failover.
1. This includes the ability for CE and FEs to determine when there is a loss of association between them, ability to restore the association and efficient state (re)synchronization mechanisms.	The OFS protocol supports echo request/reply message [OFS-1.0, p. 41] [OFS-1.1, p. 55-56]. The OFS-1.1 provides a barrier message that provides synchronization [OFS-1.1, p. 50].

```
  ForCES Architectural Requirement    OpenFlow Switch Architecture

   --------------------------------    ----------------------------
                                       The OFS-1.1 protocol supports
                                       query by the controller of the
                                       switch features, capabilities,
                                       configuration, flow table
                                       configuration, flow table
                                       entries, group table  entries,
                                       port configuration (pp. 36-42).
                                       FS-1.1 also provides
                                       Statistics on description of
                                       Switch (OFPST_DESC),
                                       Flow table status
                                       (OFPST_FLOW),aggregate flow
                                       statistics (OFPST_AGGREGATE),
                                       port statistics (OFPST_PORT),
                                       queue statistics (OFSPST_QUEUE),
                                       group statistics (OFSPST_GROUP),
                                       and experimenter extension
                                       (OFPST_EXPERIMENTER) (p.43-49).

   7. (CE redundancy - continued).
       2. This also includes the        OFS-1.1 states:
          ability to preset the        "In the case the switch loses
```

actions an FE will take in reaction to loss of association to its CE (e.g., whether the FE will continue to forward packets or whether it will halt operations." [RFC3654, p. 6.]	contact with the current controller, as a result of an echo request timeout, TLS session timeout, or other disconnection, it should attempt to contact one or more backup controllers. The ordering of the backup Controllers is not specified by the protocol." "The switch should immediately enter either "fail secure mode", or "fail standalone mode" if it loses connection to the controller, depending on the switch implementation and configuration." [OFS-1.1, p.18] In "fail secure mode", the FE behavior remains the same except FE drops "packets and messages" destined for CE. In "fail-standalone mode", FE processes all packets acts as a "legacy switch or router." [OFS-1.1, p. 18]
ForCES Architectural Requirement	OpenFlow Switch Architecture

	Flow entries persist over Failure in either fail secure Mode or fail standalone mode.
8. "FEs must be able to redirect control packets addressed to their interfaces to the CE. The (FE) MUST also redirect other relevant packets (E.g., such as those with Router Alert Option set) to their CE.	[McKeown-2008][OFS-1.0][OFS-1.1] allow packets to forward to Controller for processing.
The CEs MUST be able to configure the packet redirections information/filters on the FEs.	[OFS-1.0][OFS-1.1] Flow tables allow packet redirection filters on the FEs with action Forward controller (OFS-1.0, p. 6), (OFS-1.1, p. 13).
The CEs MUST also be able to create packets and have its FEs deliver them.	[OFS-1.0][OFS-1.1] allow a CE to FE message (packet-out) to send frames/packets out a specific port [OFS-1.0, p. 10], [OFS-1.1,p.17]
9."Any proposed ForCES architecture MUST explain how that architecture supports all the router functions as defined in [RFC1812]."	[OFS-1.0][OFS-1.1] do not consider this requirement. Future OFS work may consider this set of features.

1. Includes: IPv4 Forwarding Options

2. Should include: IPv6 forwarding options

ForCES Architectural Requirement	OpenFlow Switch Architecture
10. "In a ForCES NE, the CE(s) MUST be able to learn the topology by which the FEs in the NE are connected."	[OFS-1.0][OFS-1.1] do not consider this requirement.
11. The ForCES NE architecture MUST be capable of supporting (i.e. must scale to) at least communications. hundred of FEs and tens of thousands of ports.	[McKeown-2008], [OFS-1.0] [OFS-1.1] do not consider scale of CE/FE
12. "The ForCES NE architecture MUST allow FEs and CEs to join and leave NEs dynamically."	[McKeown-2008], [OFS-1.0], [OFS-1.1] do not consider issues relating to active join/leaving of CEs and FEs in communication.
13. "The ForCES architecture MUST support multiple CEs and FEs. However, coordination between CEs is out of scope of ForCES. [Historical note: The restriction of CE coordination was a desired phase 2 work of the ForCES group.]	[McKeown-2008],[OFS-1.0], [OFS-1.1] do not directly discuss how multiple CEs will attach to FE. Or FEs attach to a CE.

ForCES Architectural Requirement	OpenFlow Switch Architecture
14. For pre-association phase set-up, monitoring, configuration issues, it MAY be useful to use standard management mechanisms for CEs and FEs. The ForCES architecture and requirements do not preclude this.	[McKeown-2008], [OFS-1.0], [OFS-1.1] do not consider issues relating setting up the links between CEs and FEs. Some "magic" occurs and the CE is talking to a particular FE.
In general, for post-association phase, most management tasks SHOULD be done through interactions with the CE. In certain conditions (e.g., CE/FE disconnection), it may be useful to allow management tools (E.g., SNMP) to diagnose and repair problems. The following guidelines MUST be observed: 1. The ability for a management tool (e.g., SNMP) to be used to read (but not change) the state of FE SHOULD NOT be precluded. 2. IT MUST NOT be possible	[OFS-1.0][OFS-1.1] give no discussion on other management process (SNMP) outside the [OFC-1.0] using netconf and beep

for management tools
(e.g., SNMP, etc) to
change the state of an
FE in a manner that
affects overall NE behavior
without the CE being notified.

3.3.6. ForCES versus OpenFlow - A Central Controller

ForCES and OpenFlow seek to split the control plane and the forwarding engine. Both protocols using a secure connection can be used to interact with a central controller. ForCES has spent more time determining how CEs and FEs might find one or more central controllers. OpenFlow Specifications are just beginning to rediscover the need for this work.

Both Forces an OpenFlow can provide the ability of a logically centralized controller to:

- o Collect the network view and make decisions according to control logics (or applications);

- o Interact with forwarding hardware (FE) to install forwarding policy and state,

- o Provide open APIs to users to add new features.

ForCES has considered security issues (such as Denial of Service (DOS)) and the mechanisms for grouping CEs with an FE, or FEs with a CE, Forwarding Models, and Forwarding Libraries.

OFS specifications have focused on defining one simple functionality that can be implemented in specific networks. For example, many discussions point to code deployed in Google.

3.4. Difference in Forwarding Model

ForCES and OFS pipeline processing of frames/packets include the basic steps of matching framing, processing frame, and outputting/dropping frame. The processing of a frame occurs in a pipeline of processes where initially processing adds the metadata and actions that subsequent processing will use to create the final packet that will be sent via output port or dropped.

Key ingredients of a good pipeline process are:

1) a deterministic logic based that doesn’t loop,

2) handles both unicast and multicast traffic,

3) Flexible matching that can growth with new features,

4) Metadata to allow passing of results to subsequent stages,

5) Logic that allows some stages to be skipped, and

6) Allows for no match (Table Miss).

3.4.1. Looping

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. The directed graph model prevents the recycle of processing in a loop. Each LFB defines a set of processing on handling frames/packets. For example, typical LFBs include IPv4/IPv6 Longest Prefix Matching, etc. XML is used to describe LFB model formally.

In [OFS-0.8][OFS-0.9][OFS-1.0] the forwarding model has been static defining specific functions for early experimentation of the switch. Loops have been prevent

[OFS-1.0] defines the Flow tables identified by a sequential number [0,1,2_n]. Processing loops are prevent by defining that a flow table can only transfer to a higher flow table.

[OFS-1.1] provides a Group table to augment the Flow Table logic described above. If a flow matches, instructions in the flow table may direct the packet toward specific table (group table or flow table). This jumping provides skips in sequential process unlike [OFS-1.0]. In addition, the "goto" action allows skips between tables.

3.4.2. Handling unicast and multicast

[OFS-0.9][OFS-1.0] do not provide an easy to provide cloning for multicast. The group table in [OFS-1.1] provides the necessary cloning for multiple outputs of a single packet.

A ForCES IPv4MultiLPB and IPv6MultiLPB could be defined beyond today’s ForCES standards. These LFBs would use an LPM to match the multicast address, and generate a list of "L3PortID" metadata to identify a set of ports the cloned packet could be sent out. This metadata could be passed to the EthernetEncap LFB.

3.4.3. Flexible matching

All OFS specifications ([OFS-0.8][OFS-0.9][OFS-1.0][OFS1.1][OFS1.3-pre] seeks to match the header data against the flow table’s match field. Ranging from a 10-29 possible matches in the header and metadata, the OFS provides flexible matching within the data packet (Ethernet, MPLS, IP, TCP, UDP).

[Heleplidis-Forces-LFB] shows the matching capability in OFS-1.1 can be implemented in ForCES LFBs.

3.4.4. Metadata to allow passing of results to subsequent stages,

Both ForCES and OFS ([OFS-1.1][OFS-1.3-pre]) allow metadata to be passed to subsequent spaces.

3.4.5. Optionally skipping logic

[OFS-1.1] provides a Group table to augment the Flow Table logic described above. If a flow matches, instructions in the flow table may direct the packet toward specific table (group table or flow table). This jumping provides skips in sequential process unlike [OFS-1.0]. The Group Table concepts provides the ability to group flows for execution, single out a single flow for additional processing, and use port liveness mechanisms for fast-failover. [OFS-1.1,p, 7].

The ForCES directed graph model can also allow the skips in processing by having multiple exits from.

3.4.6. Table Miss

A frame may not match any table in the forwarding pipeline.

[OFS-0.9] states "if no matching entry can be found for a packet, the packet will be sent to the controller over the secure channel"[p. 8].

Experience has taught the OpenFlow community this can be problematic.

[OFS-1.0.0-errta] states the following exceptions: "Sending the packet to the controller on table-miss may overload the switch or the controller, and some of those packets may have to be dropped, and this may be an issue in some deployments" [p., 3].

The OFS-1.0.0-errta suggests that the vendor extension may allow the packet to be dropped or forwarded via pipeline. However, due to many application use of table-miss to do topology discovery or watch traffic - this feature is continued.

ForCES does not define a global Table-Miss, but allows the LFB model to define these issues.

3.5. Difference in Logical Forwarding Block Libraries

The Open-Flow group is beginning to consider flexible description of the next OFS switches using a modeling language. No modeling language has been approved as yet.

The Force LFB Library [ForCES-LFB-Lib] has been defined and implemented. [Heleplidis-Forces-LFB] shows the modeling language can support OFS-1.1 definitions.

3.6. Difference in Protocol Interface

The OFS protocol and the ForCES protocol both use:

- Secure transport protocols over which they operate (3.6.1)
- Messages to establish the Controller/CE - FE connections,
- Messages to Loading of forwarding logic (3.6.3)
- Messages to Configuration the box (3.6.4),
- Error handling messages (3.6.5),
- Liveness protocols (3.6.6), and

. Sending packets for processing to/from controller (3.6.8).

3.6.1 Secure Transport

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. OFS-1.1 states that the data channel is "usually encrypted using TLS, but may be run over TCP"[OFS-1.1.0,p. 16]

3.6.2 Types of Messages

As Table-x shows, ForCES and OFS protocol are remarkably similar. Many OFS authors indicate the influence of the ForCES protocol on the OFS work. Due to the IETF review, ForCES protocol's top level is carefully designed with orthogonal features of association setup, association teardown, config, query, events, packet redirect, and heartbeat.

The OFS protocols [OFS-0.8][OFS-0.9][OFS-1.0][OFS-1.1] provide similar features, but have some overlapping functions. Similarly, the OFS protocol has

- o Hello (initial association),
- o Reading of switch Features (read/response),
- o config of switch and flow pipeline's via Flow tables [OFPT_FLOW_MOD), group tables [OFPT_GROUP_MOD], ports [OFPT_PORT_MOD].
- o Flow Removed [OFPT_FLOW_REMOVED] - Flow entry is removed due to either an idle timer or a hard timeout.
- o Query of statistics (OFPT_STATS_REQUEST, OFPT_STATS_RESPONSE),

- o Packet-OUT- redirect of packet from controller out a port,
- o PACKET-IN - redirect packet inbound port to controller,
- o Echo request/reply - heartbeat from OFS switches.

In addition, the OFS protocol has a Barrier Request/reply message that allows the controller to synchronize message processing. This general (but lose) definitional has allowed experimentation with OFS switches.

Due to IETF review, the similar ForCES protocol has a clear orthogonal set of actions described in terms of execution and transaction models. The CE can set execution flags on sets on transactions (groups of functions). The execution of transactions can be: execute-all-or-none, continue-execute-on-failure, and execute-until failure. A transaction set is must me the ACDity test (atomicity, consistency, isolation, and durability)[RFC5810,section 4.3.1.2]. Transaction sets have an start, middle, and end. The transaction can also signal an abort.

The notification messages for Error and Port status are uniquely specified in the OFS as a notification. In ForCES this is included with the general notification category.

Table-x ForCES vs. OFS messages

Message:	ForCES	OFS-0.9	OFS-1.0	OFS-1.1
Associate		----------Hello----------		
CE/FE (Req/Rsp)	X	X	X	X
Association		-----Feature Request/Response---		
Stop (Req/Rsp)	X	X	X	X
Query/	X	----Feature or STATS(Req/Rsp)---		
Query Response	X	X	X	X
Config [Switch]		---Config (non-flow table)-----		
(Req/Rsp)	X	X	X	X

Config (Req/Rsp)	X	------ FLOW_MOD-------		
	X	X	X	X

Table-x ForCES vs. OFS messages

Message:	ForCES	OFS-0.9	OFS-1.0	OFS-1.1
Heartbeat(Forces)	X	---Echo-Request/Response---		
/Echo-Request(OFS)	X	X	X	X
Redirect		---Packet_in & Packet-Out)		
	X		X	X
Execution flags	X		----Barrier-Set-Queue	
			X	X
Notification	x		X	X

3.6.3 Loading of forwarding logic

ForCES and OFS both use TLVS to add, modify, and delete the flow entry. In addition, ForCES has a concept of "commit" to a set of changes to allow multiple stages of set.

OFS has the concept of modifying or deleting only strictly matching flows (OFPFC_MODIFY_STRICT, OFPFC_DELETE_STRICT). This is different that the OFS default of modifying all flows with that match (with wildcard).

3.6.4 Configuration

ForCES defines changing configuration of the switching Forwarding pipeline within the protocol. OF-Config-1.0 has provided a protocol to use an OpenFlow Configuration Point (logical mode) that can configure one or more OFS via the OpenFlow configuration protocol. The configuration protocol runs on top of TLS or TCP. The configuration protocol sets the following information:

- o Failure standby mode (fail secure or fail standalone)

- o Encryption mode (TLS or not),
- o Queue configurations (min-rate, max-rate, experimenter),
- o Ports (speed, no-receive, no forward, no packet-in, link-down, blocked, life) and optionally (duplex-mode, copper-medium, fiber-medium auto-negotiation, pause, asymmetric-pause),
- o Data path id of switch.

3.6.5 Error handling and sanity checking

The error handling indicates errors that occur within the protocol. The Error handling includes message form and action failure. For OFS the action failure includes all interactions such as: hello failure, bad request, bad flow action, bad flow instruction, bad match, cannot modify entry in flow table, cannot modify entry in group table, cannot modify port.

Due to IETF review, the ForCES errors and notifications are define to contain all cases within the protocol. The error processing contains sanity checking.

3.6.6 Failure of CE/FE connection

Heart beat messages in both ForCES and OFS insure "liveness" of the CE/FE connection. The ForCES heartbeats are traffic sensitive, and are only sent if no traffic has occurred.

OFS predefines that switches should enter the following based on losing connections with controller: "Fail secure mode" or "fail standalone mode" [OFS-1.1.0,section 5.3]. In Fail secure mode, forwarding continues as previous with the only change that no packets can be uploaded to the processor.

In fail standalone mode, the OSF switch drops into the Ethernet legacy mode [OFPP_NORMAL].

If the ForCES protocol is supporting the high-availability function, the begins the engage the high-availability statement machine. OpenFlow specifications have not yet described how High-availability will work in Open-Flow.

4. Use of ForCES and OFS in Applications

ForCES and OFS [OFS-1.0][OFS-1.1] have been encoding in a variety of applications. These application include:

- Firewalls,
- Loads balancers,
- Switches
- High-availability routers,
- Wireless devices.
- Table-x ForCES vs. OFS messages

5. The use of ForCES or OpenFlow in S(D)N or CSO/SOP

This section will contain a summary of the common capabilities of ForCES and OpenFLow in environments of centralized controllers, distributed controllers, and hybrid (centralized/distributed control) suggested by open flow.

5.1 - Centralized controller logic

ForCES and OFS have been designed for centralized controller logic. ForCES has considered the pre-association and association phase of the CE-FE relationship with all the timing issues. The execution and transaction model provide a strongly reviewed model to provide roll-forward and roll-back of transactions. The high-availability drafts for ForCES provide a clear case on how to keep high-availability of forwarding and CE processing while distributing the flows.

ForCES has a clear body of work developed over years of implementation experience.

OFS specifications do not deal with how controllers find FEs. However, numerous companies are developing centralized controllers. The standardization efforts for Hybrid (OFS-1.2) and the next generation OFS switch (OFS-1.3) indicate an effort to capture this growing body of wisdom.

5.2 - Distributed controller logic

ForCES was built to distribute the controller logic to automonomous network elements that operate either as ForCES controlled or as integrated hybrid controller.

OFS has created distributed logic per switch, but considers grouping of these switches outside the OFS specifications. The Hybrid [OFS-1.2] provides use cases for Ships-in-the-Night and integrated. The Ships-in-the-Night provide per port allocation to either OFS or standard processing. The Integrated seeks to run both on a set of ports.

5.3 - Hybrid controllers

ForCES was built for the hybrid environment where routing and switching protocol.

OFS is now entering the processing of standardizing for hybrid controllers [OFS-1.2].

6. Security Considerations

No security considerations.

This is an informational comparison used to inform clarify ForCES work.

7. IANA Considerations

No IANA considerations.

8. Conclusions

Both ForCES and OpenFlow follow the basic idea of separations of forwarding plane and control plane in network elements. Both are capable of operating for centralized control, distributed control, and hybrid control.

[OFS-1.1] Flow Table Logic with the instructions and Group Tables is the major difference between the ForCES RFCs. As this paper has shown, the full ramifications of this difference need to be considered in terms of differences in capability of implementation. The author welcomes any additional implementation experience.

[OFS-1.0][OFS-1.1] lacks a forwarding model, a standardized LFB library and the concepts of FE-CE associations (FE-Manger, CE-Manager, pre/post association phase). It appears the OpenFlow work is starting to invent the equivalent of existing ForCES work as OpenFlow work. The guide of this reinventing seems to be the Google code snippets passed to the OpenFlow Forum as examples of "running code" to provide rough consensus.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5810] Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, March 2010.

[RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, March 2010.

[RFC5811] Hadi Salim, J. and K. Ogawa, "SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol", RFC 5811, March 2010

9.2. Informative References

[McKeown2008]
McKeown, N., Anderson, T., Balakrishnan, H., et al, "Openflow: enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review. 2008, 38(2):69-74.

[OFS-1.0.0]

OpenFlow Switch Specification - Version 1.0.0 (Wire Protocol 0x01), December 31, 2009.

[OFS-1.0.1-rc3] OpenFlow Switch Errata - Version 1.0.1-r3, June 12, 2012.

[OFS-1.1.0]
OpenFlow Switch Specification Version 1.1.0 (Wire Protocol 0x02). February 2011. (http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf)

[OpenFlow-1.2] Open Flow 1.2 - Hybrid Switch (Terminology, Use cases, etc), April-May notes, work-in-progress.

[OFS-1.3-rc4[OpenFlow Switch Specification 1.3 (version 1.3-rc4) (Wire Protocol 0x04) April4, 2012. [OpenFlow members only]

[OFS-1.1.0]
OpenFlow Switch Specification Version 1.1.0 (Wire Protocol 0x02). February 2011. (http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf)

[OFC-1.0] OpenFlow Configuration and Management Protocol OF-CONFIG 1.0 (January 15, 2012).

[RFC3654] Khosravi, H. and T. Anderson, "Requirements for Separation of IP Control and Forwarding", RFC 3654, November 2003.

[RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004.

[LFB-Lib] Wang W., Haleplidis E., Ogawa K., Li C., J. Halpern, "ForCES Logical Function Block (LFB) Library", draft-ietf-forces-lfb-lib-06, Work in Progress.

10. Acknowledgments

The author would like to thank Tina Tsou, Zhiliang Wang,Jing Huang, Xingang Shi, and Xia Yin. Their earlier draft comparing the FORCES and ONF Technology inspired this draft providing an opposing viewpoint. It is said that "iron sharpens iron" so that in our debates we sharpen our understandings.

The author also acknowledges Edward Crabbe's succinct comments which also inspired the in-depth comparison found in this draft. I only hope that this "wordy" draft proves worth of his pithy and concise insights.

Authors' Addresses

Susan Hares
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: Susan Hares Susan.Hares@huawei.com