Internet Engineering Task Force B. Zhuge
Internet-Draft Zhejiang Gongshang University
Intended status: Standards Track Y. Wang
Expires: June 21, 2018 Simon Fraser University
Y. Qi
Y. Zhu
W. Wang
Zhejiang Gongshang University
December 18, 2017

An Intelligent SDN Framework based on Meta-Model with Software-Defined Pricing (SDP)
draft-zhuge-sdnrg-sdn-sdp-06

Abstract

This document defines a notion called Software-Defined Pricing (SDP) and introduces it into a Software-Defined Networks (SDN) framework. The SDN system with SDP inside is expected to promote the efficiency on SDN resources usage and ease management for service providers.This document also defines a mechanism that can efficiently mannage SDN framework orderly and intelligently.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on June 21, 2018.

Copyright Notice

Copyright (c) 2017 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 (https://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. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Software-Defined Networks(SDN) is in the research process. With the idea of SDN, networking resources like switches, routers and types of Network Elements (NEs)are managed as kinds of virtual resources, forming virtual networks so as to provide rather flexible services to network users. In this research process, we noticed that how to price the services and the use of virtual network resources in an SDN is as critical as how the SDN is defined. We consider that it seems a precious idea to treat a service pricing mechanism as part of the SDN framework and to manage it in a software-defined way.

Network service prices are traditionally determined by service providers in a rather rigid way, which lacks of flexibility and sometimes even fairness to resources users. By means of the idea of SDP, it is able to treat service pricing as a part of SDN, forming a service pricing model flexible to time, traffic and other network factors and status. In this way, it is expected to promote the efficiency of SDN resources usage and ease the management for service providers.

Due to the ever increasing network complexity, the operators of intelligent network are driven toward a virtualization of network functionality that calls for a paradigm shift from a hardware-based approach to a software-based approach. We will correspondingly develop an intelligent management framework based on the concept of SDN, which is featured by the decoupling of control plane from data plane.The intelligent SDN framework aims to provide a viable way to solve the existing challenges in a unified manner.

2. Software-Defined Pricing (SDP)

Software-Defined Pricing (SDP) is an idea specific to network management, whose core is that the service prices of network resources are determined by means of software-defined algorithms and/or mechanisms, which figure the prices according to various factors and status of the network resources. In contrast to SDP, service prices may be pre-determined rigidly by service providers.

An SDP Protocol is an instance of SDP implementation shown in a way of protocol, which specifically defines algorithms and/or mechanisms to price specific services and use of network resources. An SDP protocol may be a private protocol if it is defined by a service provider personally, or a public protocol if defined publicly by standardization organizations.

By use of the software-defined mechanism, SDP essentially supports automatic negotiations of prices in a pricing process. Automatic resource and price negotiation features a Guaranteed Service (GS). As a result, SDN with SDP essentially supports GS services.

Network users must accept and abide by the network SDP protocol when they use the network resources and the services.

An SDP protocol usually includes trading partners, trading content, obligations and other transaction costs. Service providers can make provisions for users in terms of workload and resource use.

As an example, we present a typical process for an SDP protocol. When users expect to use resources from a virtual network by a service provider, users first query prices of various resources and services by means of the SDP protocol. The service provider returns the resource prices to users. Then, users will start up a price negotiation process with the service provider by use of the SDP protocol. Both the user and the service provider will proceed the price negotiation process based on their specific price negotiation algorithms. The negotiation process will be ended only from the user with the SDP protocol. It will end with an agreement is either met or not. The SDP protocol process is shown in Figure 1. Usually, in

a negotiation algorithm, the user or the service provider are able to take into consideration of current network status and other network factors, which make the price negotiation process much more efficient and flexible than traditional pricing methods.

```
+------+ +                             +                    +-----------+
|      | | --------SDP protocol------->| -----------------+ |           |
|      | |                             | search price     | |           |
|      | | <-------SDP protocol--------|<-----------------+ |           |
|      | | --------SDP protocol------->|------------------+ |  service  |
| user | |                             | price negotiation| | sprovider |
|      | | <-------SDP protocol--------|<-----------------+ |           |
|      | | --------SDP protocol------->|------------------+ |           |
|      | |                             | negotiation ends | |           |
|      | | <-------SDP protocol--------|<-----------------+ |           |
+------+ +                             +                    +-----------+
```

Figure 1: Process of an SDP Protocol

To fulfill above process, an SDP protocol header may usually include fields like shown in Figure 2, where:

- ID: the unique identifier of the protocol header.

- Level: service priorities identified.

- Expression: including one or more ITP(ID-Type-Properties) formats, where ID is the unique identifier of a resource, Type is the type of resource, Properties is the attributes of the resource.

- TimeSpec: a structure of service time, mainly refers to the selection of the service period.

- Price: the price of the transaction.

- ContractTime: trading hours.

- State: trading status with success or failure.

```
+-----------------------------------------------------------------------+
| ID | Level | Expression | TimeSpec | Price | ContractTime |  State    |
+------------|------------|----------|----------------------------------+
             |            |
             |            +----------------------+
             V                                   |
+------------------------+                       |
| ID | Type | Properties |                       |
+-----------|------------+                       V
            |         +-------------------------------------+
            |         | Y/M/D | Mon-Fri/Weekend | 8:00-0:00 |
            V         +-------------------------------------+
+-----------------------------------------+
|  rate  |  delay  |  shake   |  etc.     |
+-----------------------------------------+
```

Figure 2: An SDP Protocol Header

(TBD)

3. SDN with SDP

3.1. Adopting SDP in SDN

SDP can be applied to SDN architecture well because of its natural software-defined feature.

In SDN architecture, control plane and data plane are separated to achieve the segregation of the control and forwarding. A typical SDN architecture usually includes: application layer, control layer, and infrastructure (forwarding) layer. To adopt SDP in SDN, an SDP module is applied. An SDP module implements the SDP protocol and corresponding negotiation algorithms/mechanisms. An SDP module can be applied to any layer in the SDN, where resources need to be priced. In this way, theoretically, all kinds of network resources and services can be programmed in terms of use prices as well as resources functions. This makes SDN more complete regarding its software-defining characters.

In SDN application market, resource providers and resource consumers actually hardly know each other fully for the value of resources and services. Hence, the trade between them is an information asymmetry game. To take this into consideration, an SDP module with its protocol and associated negotiation mechanisms applied to an SDN system is usually of the following features:

o 1) An SDP module can be distributed across parts of SDN system, to get the optimal level of service quality under budget constraints

of service consumers. As a result, the SDP module usually further contains a pricing module and a trading module, used for pricing and trading of resources respectively. With an SDP module, users can submit their requirements according to their budgets at the SDN application layer to SDN control layer. Then, the SDN control layer can get results of optimal resource services based on user's budget.

o 2) An SDP module usually includes an auto-negotiation mechanism. During the trading process, resource providers first get the price based on the price algorithm and/or mechanism, and present them to resources consumers. If consumers are not satisfied with the prices, process of negotiation with auto-negotiation algorithm or mechanism will be triggered.

o 3) With SDP, use of resources and their prices are not unique anymore. Different resources providers may provide different prices even for the same resources. SDP module may query different resources providers for optimal prices. This process usually takes place at the SDP protocol stage of searching prices.

o 4) In an SDP transaction, an SDN application usually act as a resource provider to end users. Whereas, at the same time, it can also act as resource consumers to SDN control plane as well as SDN forwarding plane. It sells resources to end users. At the same time, it may buy or hire resources from SDN core systems. All these can be done by use of SDP module.

o 5) With a time attribute, SDP can respectively support SDN applications well for temporary term users or long term users regarding optimal use prices.

3.2. Framework of SDN with SDP

As mentioned, a typical SDN framework usually includes Application Layer, Control Layer, and Infrastructure (forwarding) Layer. In SDN Application Layer, things like virtual servers and other SDN personalized services will be presented as individual SDN Applications. Based on the idea above on adopting SDP to SDN, a typical framework of an SDN system which adopts SDP module is as shown in Figure 3.In this framework, the SDP-App includes an SDP module inside and makes the module support software-defined pricing function.

SDP-App may exist in each layer of the SDN framework. Note that, SDN Application communicates with SDN controllers via the AD-SAL and Service Interface.Either should require that the AD-SAL and Service Interface must support SDP protocol to support the SDN with SDP.

Also note that, SDN Control Layer includes the network service, SDP-App, and control abstraction Layer(CAL), it is defined to communicate with SDN forwarding layer by means of the resource abstraction layer(RAL) and the uniformly defined SDN southern interface protocols like ForCES ,OpenFlow, etc. To support SDN with SDP, SDP protocol must be designed supportable by these protocols for messaging purpose. This may become a key question for the design of an SDP protocol. The SDN Forwarding Layer includes the network element, and SDP-App. It is exposed via the Resource Abstraction Layer (RAL), which may be expressed by one or more abstraction models.

(TBD)

```
                   o--------------------------------o
                   |                                |
                   | +-------------+   +----------+ |
                   | | Application |   |  SDP-App | |
                   | +-------------+   +----------+ |
                   |       Application Layer        |
                   o---------------Y----------------o
                                   |
     *-----------------------------Y---------------------------------*
     |     Application-Driven Services Abstraction Layer (AD-SAL)    |
     *-----------------------------Y---------------------------------*
                                   |
                                   |Service
                                   |Interface
                                   |
     o-----------------------------Y--------------------------------o
     |                   Control   |   Layer                        |
     |                  +----------Y--------+  +---------+          |
     |                  |  Network Service  |  | SDP-App |          |
     |                  +----------Y--------+  +----Y----+          |
     |                             |                |               |
     |                *------------Y----------------Y------*        |
     |                |  Control Abstraction Layer (CAL)   |        |
     |                *------------Y-----------------------*        |
     |                             |                                |
     o-----------------------------|--------------------------------o
                                   |
                                   | Southbound
                                   | Interface
                                   |
     *-----------------------------Y---------------------------------*
     |              Resource Abstraction Layer (RAL)                 |
     *-----------------------------Y---------------------------------*
     |                             |                                 |
     |                    o--------Y-----------o   +----------+      |
     |                    |   Network Element  |   | SDP-App  |      |
     |                    o--------------------o   +----------+      |
     |                           Forwarding Layer                    |
     +---------------------------------------------------------------+
```

Figure 3: An SDN Framework with SDP

As another example, we try to present an SDN application which uses SDP to access network resources so as to get optimal resources use price. We call the SDN application a 'Chat' App, which is to construct a social App platform to connect, communicate and share among people and things by means of Guaranteed-Service (GS) rather than Best-Efforts (BE) services to users. Hence, the App needs to

hire network resources from cloud network service providers to provide virtual server and Guaranteed Service (GS) resources.

Fig 4 shown the process for 'Chat' to access the GS Resources by use of SDP. 'Chat' client and 'Chat' Server makes service agreement via SDP module. 'Chat' server may be implemented as a virtual server, whose pricing is also implemented by SDP module. Further more, resources to support the virtual server and the 'chat' message forwarding are used based on SDP negotiations. As shown inFigure 4 , in this case, SDN controller inside the virtual server of 'chat' may send requests to multiple cloud platforms by SDP module(such as Sina cloud, Baidu cloud and Ali cloud in the figure). All the cloud service providers return with resource prices, and SDN controller inside the 'chat' server select or negotiate with the cloud service providers. SDN controller finally may select or get a successful or failed negotiation results and returns to the 'chat' client via SDP protocols. As a result, a transaction for a GS service pricing ends.

```
                        +---------------------+
                        |   'Chat' client     |
                        |     ( With SDP )    |
                        +---------------------+
                                   A
                                   |
                                   V
                        +---------------------+
                        |    'Chat' server    |
                        |     ( With SDP )    |
                        +---------------------+
                                   A
                                   |
                                   V
                        +---------------------+
                        |    virtual server   |
                        |     ( With SDP )    |
                        +---------------------+
                                   A
                                   |
                                   V
                        +---------------------+
                        |    SDN controller   |
                        |     ( With SDP )    |
                        +---------------------+
                                   A
                                   |
             +-----------------------------------------------+
             |                     |                       |
             V                     V                       V
     +-----------------+      +---------------+       +--------------+
     |   Sina cloud    |      |  Baidu cloud  |       |  Ali cloud   |
     |   (With SDP)    |      |  (With SDP )  |       |  (With SDP   |
     +-----------------+      +---------------+       +--------------+
```

Figure 4: The Process for 'Chat' Accessing Resources by Use of SDP

3.3. Framework of SDN Bases on Meta-Model

A Meta-Model is a model architecture in which each defined layer will supply services and functions that built in a meta-model, to be exactly, APP-likely way. Then all of APPs could be refactoried and combined to satisfied sorts of diversified needs from users in upper layer. To be more precisely to defined the meta-model, the following elements will be invoked:

- o Meta-APP:The minimum logical elements in application layer that be used to combine and register applications with more complexity.

Meta-APPs includes all of function feature and needs of application, and they could abstract the fundamental functions of network service needed by business according to feature and requirements of application.

- o Con-App:Business Abstraction Layer:It is a mechanism that be used to logically mapping the meta-app onto the corresponding meta-service in the application layer. Business Abstraction Layer will recognize and adapte all of meta-service supplied by controller, and then select suitable meta-service to service a meta-app bases on the requirement of relevant business.

- o SDP-App:The modules proposed here which could support software-defined pricing function in the aspects of resource and service from each layer.

- o Meta-Ability: The fine-grained elements of switch function in switch layer, which is the atomic elements in divide assignment. It is the fundamental host components. All of the meta-ability can supply diversified host ability for meta-service within the scope of world-wide network.

- o Resource Abstraction layer: Mapping the physical resources to virtual resource. Resource Abstraction Layer uses virtualization technology to abstract physical resources at the bottom in order to shidding the difference between facilities. In addition, Resource Abstraction Layer can schedule resources to achieve its reasonable alloction, which can avoid the quality reduction of upper layer application due to the resource shortcut and waste in result of its long-term idleness, raising the resource untilization ratio.

Based on the idea above on Meta-Model to SDN, a typical framework of an SDN with SDP bases on Meta-Model system is as shown in Figure 5.In this framework, the application layer includes a meta-app part inside and makes the module support dividing and refactoring the meta-service .

(TBD)

```
  o----------------------------------------------------------------o
  |                                                                |
  | +--------------+   +-----------+  +-----------+   +-----------+   |
  | |   Meta-App  |   |  SDP-App |  | Meta-APP |   | Meta-App |   |
  | +--------------+   +-----------+  +-----------+   +-----------+   |
  |                       Application Layer                        |
  o----------------------------------------------------------------o
                               |
```

```
     *-----------------------------Y---------------------------------*
     |                  Business Abstraction Layer                   |
     *-----------------------------Y---------------------------------*
                                   |
                                   |Service
                                   |Interface
                                   |
     o-----------------------------Y--------------------------------o
     |                   Control   |   Layer                        |
     |                  +----------Y--------+  +---------+          |
     |                  |    Meta-Service   |  | SDP-App |          |
     |                  +----------Y--------+  +----Y----+          |
     |                             |                |               |
     |                             |    +-----------+               |
     |                             |    |                           |
     |                      *------Y----Y-*                         |
     |                      |   Con-App   |                         |
     |                      *------Y------*                         |
     |                             |                                |
     o-----------------------------Y--------------------------------o
                                   |
                                   | Southbound
                                   | Interface
                                   |
     *-----------------------------Y---------------------------------*
     |                  Service  Abstraction Layer                   |
     *----Y-------------Y----------Y-----------Y-------Y---------Y---*
          |             |          |           |       |         |
     *----Y-----*  *----Y--*  *----Y----*  *---Y--* *--Y--*  *---Y----*
     | OpenFlow |  | OVSDB |  | NETCONF |  | LISP | | BGP |  | ForCES |
     *----Y-----*  *----Y--*  *----Y----*  *---Y--* *--Y--*  *---Y----*
          |             |          |           |       |         |
     *----Y-------------Y----------Y-----------Y-------Y---------Y---*
     |              Resource Abstraction Layer (RAL)                 |
     *---------------------------------------------------------------*
     |                                                               |
     |  o--------------o    o----------------o   +----------+        |
     |  | Meta-Ability |    |  Meta-Ability  |   | SDP-App  |        |
     |  o--------------o    o----------------o   +----------+        |
     |                         Forwarding Layer                      |
     +---------------------------------------------------------------+
```

Figure 5: An SDN Framework with SDP Bases on Meta-Model

3.4. The Relationship between the Layers in the SDN Framework with SDP Bases on Meta-Model

As the mentioned definiton of SDN framework with SDP bases on Meta-Model above, the minimum units of application are meta-app, which of control layer are meta-service, and the minimum units in forwarding layer are meta-ability. Meanwhile, Refering to the idea of reconfigurable network architecture, The features of business and the load capacity of network could be abstract as a chained model like "Application -> Meta-App -> Meta-Service -> Meta-Ability" which be showed in Figure 6. A complicated network application can be divided to amounts of meta-app abstractly, and each meta-app contains features and functions of network applications. In this case, The meta-app is also combined by a series of fine-grained meta-service sets that called network fundamental function components. In addition, A set of meta-abilities can combine a series of fundamental load components of network, which can associate meta-service.

Conceptually, the meta-ability is a tiny unit defined by fundamental network, which having the abilities to certain capacity and function information. By using meta-abilities, The core of internet can be abstracted to lots of distinct LFBs (Logical Functional Block) which combined by the set of meta-abilities. Meanwhile, This also allows the core of internet to add new meta-ability extensions for promoting network extensibility.

Meta-Abilities are described as senary struction including type, mark, attribute set, operation, vendor and price. This "type" element specified the type of meta-abilities for classifying the meta-abilities with similar network function into separate type class. This "mark" element should be defined as natural number. Via type and mark element, the unique identity of meta-abilities can be defined within the scope of world-wide network. This "attribute set" element includes parameters about targets and relevant capacity of those targets. This targets in attribute set can specify the targets that meta-abilities should deal with. And this relevant capacity in attribute set can indicates the parameterized capacity goal of meta-abilities. This "operation" element indicates the operation that meta-ability can implemente to targets. This "vendor" element on behalf of the vendors of this meta-ability (e.g., telecom operators). This "price" element state the cost in the process of using this meta-ability.

(TBD)

```
                 Service                 Service             Application
                 Abstract                Map                 Abstract
o---------------o       o---------------o       o-----------o
|               |       |               |       |           |
| +-----------+ |       | +-----------+ |       | +-------+ |
| | M-ability |->------->-| M-service |->-+----->-| M-App |->---+
| +-----------+ | +----->-+-----------+ | |     | +-------+ |   |
|               | |     |             | | |     |           |   |
| +-----------+ | |     | +-----------+ | +----->-+-------+ |   |
| | M-ability |->-+----->-| M-service |->---+--->-| M-App |->-+ |
| +-----------+ | | +--->-+-----------+ |   | +->-+-------+ | | |
|               | | |   |               |   | | |           | | +->o-----o
| +-----------+ | | |   | +-----------+ |   +-|->-+-------+ | +--->| App |
| | M-ability |->-|-+--->-| M-service |->-+---|->-| M-App | |  +-->o-----o
| +-----------+ | |  +-->-+-----------+ | |   | | +-------+ |  |
|        .      | |  |  |         .     | | +-+ |           |  |
|        .      | |  |  |         .     | | |   |           |  |
|        .      | |  |  |         .     | | |   |           |  |
|        .      | |  |  |         .     | | |   |           |  |
| +-----------+ | +--|-->-+-----------+ | +-|--->-+-------+ |  |
| | M-ability |->----+  | | M-service |->---+--->-| M-App |->--+
| +-----------+ |       | +-----------+ |       | +-------+ |
|               |       |               |       |           |
o---------------o       o---------------o       o-----------o
```

where:

M-ability = Meta-ability
M-service = Meta-service
M-App = Meta-App

Figure 6: The Relationship between Meta-Model Layers

4. The Trading between the Layers

As shown inFigure 7A complete SDN environment is made up of application layer, control layer, data forwarding plane, if regard SDN environment as an economy market ,Then corresponding to the three layers structure of SDN environment, the economy market can be divided into: user layer, trading platform and provider layer. But each layer is embedded with the pricing model and consumption pattern which is apply to this layer , the communication between each other is accomplished by special protocol, each of them is independent but closely linked.In application layer, there are many users, the users were independent of each other, and they belonged to different platforms.In control layer there are multiple platforms, on the two ends of platform respectively connected to different users and providers, the existence of multiple platforms can solve the monopoly

of a single platform and the problem that users and providers'choice unicity.In fowarding layer,there are many providers, they can offer different types of resources for each platform.

```
*-------------------------------------------------------------------------*
|Application +--------------+  +---------------+      +----------------+  |
|  Layer     | Application 1|  | Application 2 |  ... | Application n |   |
|            +--------------+  +---------------+      +----------------+  |
*--------------Y-Y-Y---------------Y-Y--Y--------------------Y--Y--Y------*
               | | | +-------------+ |  |    +---------------+  |  |
               | | | |   +-----------|--|----|------------------+  |
               | | +-|---|-----------|--|----|----------------+    |
               | +---|---|-------+   |  +----|-----------+    |    |
               |     |   |       |   |       |           |    |    |
*--------------V-----V---V-------V---V-------V-----------V----V----V------*
|  Control   +--------------+  +---------------+      +----------------+  |
|  Layer     |Control Plane1|  |Control Plane2 |  ... |Control Plane n|   |
|            +--------------+  +---------------+      +----------------+  |
*--------------Y-Y-Y---------------Y-Y--Y--------------------Y--Y--Y------*
               | | | +-------------+ |  |    +---------------+  |  |
               | | | |   +-----------|--|----|------------------+  |
               | | +-|---|-----------|--|----|----------------+    |
               | +---|---|-------+   |  +----|-----------+    |    |
               |     |   |       |   |       |           |    |    |
*--------------V-----V---V-------V---V-------V-----------V----V----V---------*
|Forwarding +-----------------+ +-----------------+     +------------------+ |
|  Layer    |Forwarding Plane1| |Forwarding Plane2| ... |Forwarding Plane n| |
|           +-----------------+ +-----------------+     +------------------+ |
*----------------------------------------------------------------------------*
```

Figure 7: Multi-Ownership Combinatorial Double Auction Model

We summarize the differences of three kinds of trading pattern and the position of SDN architecture which applied them , as shown in Figure 8:

Trading Pattern	Product Pattern	Trading market	Trading Risk	Trading price	Location of SDN
spot trading	Retail Commodity	No	Greater risk	Negotiation non-standard	Application layer
Futures trading	A kind of products as a unit	Future	Has margin, less risk	Settlement based on the price of the exchange	Data forwarding layer
Planned trading	Goods in any combination	Overall market	PlannedSpending almost no risk	Control price, according to supply and demand	Entire architecture

Figure 8: the Differences among Three Trading Patterns

The commodities mode, trading market and other factors of planned trading decides it apply to the entire SDN market; the commodities mode of spot trading determines which is suitable for small number of resources trading, and it has some risks, therefore it works in the application layer of SDN architecture; the commodities mode of futures trading trade in a kind of resource as a unit, and the risk is small, possess the futures market, so it works in data forwarding layer of SDN architecture.

5. Intelligent Mechanism bases on Meta-Model

5.1. Intelligent Component

As shown inFigure 9We defines a advenced mechanism of SDN framework, the primary module sustains the speciality of intelligence and smart, which composed by three main components, including network sense function, SDP module, and policy control.

The network sence function, within this intelligence mechanism, in change of preceiving each details , monitoring network state on a domain scale, to be special, bandwidth, quality of service, line loading, besides, node capacities of calculation, storage and speed. While network sence function learns those information from SDN south interface, they could analyze and encapsulate them with an overlay, following, send encapsulated message to SDP module though inner port.

The SDP module porvides pre-customized algorithm based upon SDP protocol, which would operated in encapsulated message processing, consists the interaction and the relation between meta-abilities and requirements of applicatoin, and manages preliminary SFC policies, overviewed of RFC 7665 [RFC7665], to construct meta-abilities with knowledge reasoning.

Policy, in contrast, interacts with the system in other places.Policies and SDP module may monitor meta-abilities to decide if additional (or fewer) instances of services are needed. When applicable, those decisions may in turn result in interactions that direct the control logic to change the placement of meta-abilities service function chain,in short, MSFC.

The policy control module is part of the overall intelligent mechanism, and is responsible for constructing MSFCs, translating MSFCs to forwarding paths, and propagating path information to participating meta-ability nodes to achieve requisite forwarding behavior to construct the service overlay and qualify the requirements of application. For instance, the physical placement of meta-ability nodes may be static; selecting exactly which MSFCs and which meta-abilities from those MSFCs are to be used, or it may be dynamic, allowing the network to perform some or all of the choices of MSFC or meta-abilities to use to deliver the selected service chain within the constraints represented by the service path. While, within this mechanism, physical resource and logicl meta models state are permitted to be registered on the policy control module.

Architecturally, within the same policy control module domain, some MSFCs may be fully specified, selecting exactly which MSFC and which meta-abilities are to be visited by packets using that MSFC, while other MSFCs may be quite vague, deferring to the traffic the decisions about the exact sequence of steps to be used to realize the MSFC.

```
       o  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
       .                   +------------+
       .                 + | SDP Module |+  + -----------------------
South  .   +---------+  /  +------------+ \ | Policy   +----+   +----+
Intf   .   | Network | /                 +| Control  |SFC1|...|SFCn|
+--------->| Sence   |+  +---------------+  | Module   +----+   +----+
Network.   | Function|   |Kernel Control|   +------------V--------V---
State  .   +---------+   |  Module      |              /        /
       .                 +--------------+<------------+--------+
       o  .  .  .  .  .  .  .  .  .  Y  .  .  .  .  .  .  .  .  .  .  .  .  .
                                   +-------------+
                                   |             |
                                   V             V
                              Meta-Ability1...Meta-Abilityn
```

Figure 9: Composite Intelligent and Software-Defined-Price Mechanism Model

5.2. Mechanism Principles

Intelligent mechanism bases on meta-model is predicated on several key architectural principles:

1. Universality:IPv6 networks should be supported. And diversified user requirements and endpoint also can transport in this mechanism.

2. Sensation:Intelligent mechanism should have the capacity to sence diversified behavior characteristic of network services and users. While, this mechanism should support the intellgent recognition and analysis based on users, context and application, and on this situation to achieve a flexible policy scheduling and routing.

3. Efficient: Requires the network to provide on-demand support for users and application requirements, intelligent traffic control capabilities, and the ability to alleviate the unreasonable consumption of uncontrollable traffic to network resources. At the same time,this mechanism can improve network adaptability reasonably, and achieve content distribution efficiently by enhancing the dynamic regulation of resources on demand quality assurance capabilities.

4. Openness:The application of security and network awareness, traffic scheduling and other capabilities should be combined in the netwok of intellgent open-architecture. This mechanism could lower the threshold to further innovation of the upper

application, and meet the needs of personalized, diverse user, meanwhile, optimize the existing carrier network management model, improve the operational efficiency of the network.

5. Evolution: Network development must support future users and applications through overlaying new-built or existing system upgrades based on active network-level architectures. Seldom change to the running underlay network forwarding facility -- implicit, or explicit -- are needed to deploy and invoke intelligent mechanism. And this mechanism should provides standardized communication protocols and interfaces for collaboration processes between different levels, between external systems, between meta-models, with little influence on existing networks and can be gradually upgraded amongst existing networks architecture.

6. Autonomy:With the development of network, it is necessary to introduce artificial intelligence technology to achieve self-adjustment, self-optimization, self-recovery of the network through collection of huge data of network state and machine learning. The areas of machine learning which are easier to be used in the network field may include: troubleshooting of network problems, network traffic prediction, traffic optimization adjustment, security defense, security auditing, etc., to implement network perception and cognition.

5.3. MSFC

1. Meta-model-based service function linking method in this draft encapsulates the logic function blocks in the metamodel network into multiple MetaService Functions (MSFs) in combination with SFC[37]. MSF is a virtual element or is embedded into an industry-standard universal physical network element. It is mainly responsible for receiving various types of traffic and forwarding the data to a designated MSF or network logical service block. We define a Meta Service Function Chain (MSFC) based on a metamodel network architecture to describe an ordered application of services or data to upper layers for processing data packets, network frames and service flows A collection of abstract MSFs that classify and forward the processing results.

2. SFC Management orchestration domain refers to the network or network area that enables an SFC and serves the upper layer. An SFC can only be constrained in a single management orchestration domain and is subject to management, coordination and scheduling of the management orchestration domain. Figure 24 shows an MSFC architecture in a metamodel-based SDN network. This system establishes service sublayer and service execution sublayer in

each layer of SDN network. The service organization sub-layer includes core control module, SDP module and other functional nodes responsible for MSF scheduling. It receives the upper-layer service request through the northbound interface, and the SDP module is responsible for generating a specific MSF policy and uses the internal control module Interface with the underlying MSF to deliver the policy to the service execution sub-layer. The service execution sublayer includes a series of MSF collections that have business traffic acceptance, processing, and forwarding capabilities. EP represents a terminal node, and the terminal may be a network element device, a user terminal or an MSFC in other management domains.

3. The next generation of new networks requires that services can dynamically adapt to the changing needs of services. Network service providers need to combine basic meta-capabilities based on orthogonal decomposition into different composite network services to achieve service-oriented service customization and agile development. When a service layer user sends a request the SDN control layer, the SDN agent reconfigures from the NE node to the service link by issuing a policy and makes a series of internal adjustments to adapt to this type of service.

4. Meta-capabilities within a single-cell node can be dynamically combined into a sequence of metaclassic capabilities that we call the meta-capability stack. The meta-capability stack is a basic logical structure for providing data transmission and initial processing of information in a network element node. When a meta-capability stack is created, it is temporarily created for service. When the request is completed, it will continue to survive for some time until it dies naturally or a similar strategy arrives.

6. Security

TBD

7. IANA Considerations

This document has no actions for IANA.

8. Informative References

[China-Communications]
Zhuge, B., Deng, L., Dai, G., Wan, L., Wang, W., and J. Lan, "Resource Scheduling Algorithm and Ecnomic Model in ForCES Networks.China Communications", 2014.

[ONF-White-Paper]
Fundation O N., "Software-defined networking: The new norm for networks", 2012.

[RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, DOI 10.17487/RFC5812, March 2010, <https://www.rfc-editor.org/info/rfc5812>.

[RFC6956] Wang, W., Haleplidis, E., Ogawa, K., Li, C., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Logical Function Block (LFB) Library", RFC 6956, DOI 10.17487/RFC6956, June 2013, <https://www.rfc-editor.org/info/rfc6956>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[Telecommunications-Science]
Zhuge, B., Wang, B., and Y. Wang, "Architecture of SDN applications based on Software-Defined price", 2015.

[Telecommunications-Science.2]
Zhuge, B., Zhu, H., and B. Wang, "Research on the Meta Model Construction Mechanism in SDN Architecture", 2016.

Authors' Addresses

Bin Zhuge
Zhejiang Gongshang University
18 Xuezheng Str., Xiasha University Town
Hangzhou 310018
P.R.China

Phone: +86 571 28877723
Email: zhugebin@zjsu.edu.cn

Yining Wang
Simon Fraser University
8888 University Drive
Burnaby
Canada

Phone: +1 (778) 885-0009
Email: ywa165@sfu.ca

Yihang Qi
Zhejiang Gongshang University
18 Xuezheng Str., Xiasha University Town
Hangzhou 310018
P.R.China

Phone: +86 571 28877723
Email: 1107811460@qq.com

Yingjie Zhu
Zhejiang Gongshang University
18 Xuezheng Str., Xiasha University Town
Hangzhou 310018
P.R.China

Phone: +86 571 28877723
Email: zhuyj6055@163.com

Weiming Wang
Zhejiang Gongshang University
18 Xuezheng Str., Xiasha University Town
Hangzhou 310018
P.R.China

Phone: +86 571 28877761
Email: wmwang@zjsu.edu.cn