Flexible Network Resource-Allocation Architecture Using Specification Injection

Masataka SATO, Shingo HORIUCHI
Access Network Service Systems Laboratories
NTT
Tokyo, Japan
E-mail address: {masataka.sato.bm, shingo.horiuchi.kg}@hco.ntt.co.jp

***Abstract*— To manage hybrid networks, telecommunications carriers should transform their network management architecture. We previously proposed a management architecture that has common management functions and data that are not specialized for a specific network. Using this architecture, carriers will be able to easily manage new networks to accelerate deployment. In this paper, we propose a resource-allocation architecture, which is an extension of our previous architecture. This architecture can manage the network-capacity amount and flexibly allocate it according to user demand. With this architecture, operation can be flexibly changed based on external specifications. This allows carriers to quickly release a new network service.**

Keywords—Network Management, Multi-Layer Network, TM Forum, SID

I. INTRODUCTION

A. Background

The business of telecommunications carriers has been changing worldwide. Virtualization technology has made much progress and enables network functions to be implemented using software and low-cost, general-purpose servers. However, there are still conventional networks consisting of dedicated network equipment. Therefore, carries need to manage both virtualized and non-virtualized networks [1]. It is also necessary to combine network functions and provide new services for a rapidly changing business environment.

To manage such a hybrid network and provide new services quickly, an operation architecture is necessary. Telecommunications carriers usually use operation support systems (OSSs) to accept service requests, allocate network resources satisfying the requests, and control network equipment. However, some legacy OSSs are individually optimized for specific network services and have management functions and data depending on the specific service. Therefore, the legacy OSS architecture cannot keep up with rapid service changes, and the management functions and data models of OSSs are modified according to new services and network equipment. This development and construction approach, called “silo”, incurs considerable development costs and time.

We propose a resource-allocation architecture that has common management functions and data not specialized for a specific service. The OSS based on the proposed architecture does not need to modify the functions and data to change a service and use new network equipment. Since the characteristics of each network are described in the external operation specifications and the function executes based on these specifications, carriers can easily change the function depending on the network by changing the external specifications and injecting them into the architecture. As a result, carriers can quickly release a new network service and respond to various requirements of service providers. Previously, we proposed a management architecture [2] for describing network information by using a unified model and mapping method [3] for converting service orders into resource orders.

Our resource-allocation architecture is an extension of the above architecture and can manage the network-capacity amount and flexibly allocate it according to user demand.

B. Approach

Our previous architecture defines the characteristics of each network externally and has common functions independent of a specific network. We apply this architecture to resource-allocation logic to solve the above problem.

The resource allocation process is as follows. Based on a service request from a user, a carrier creates a logical path on constructed physical networks and provides a communication service to the user. In this process, the carrier cannot generate unlimited paths because there is an upper limit of the capacity of the network equipment. Therefore, it is necessary to generate a path after judging whether it can fit in the capacity.

For example, consider the case of setting a 10-Mbps bandwidth logical path from switch (SW) A to SW F in Fig. 1. The link between SW A and C cannot meet the 10-Mbps requirement because it has only a 3-Mbps capacity.

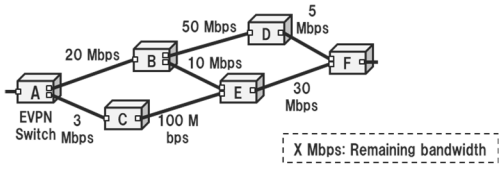


Fig. 1 Example of Ethernet virtual private network

For such judgment, legacy network management OSSs hold the capacity information to be managed for each network protocol and add or count each time a logical path is generated. This determines whether the capacity requirements are met.

However, because the capacity information management and determination function legacy network management OSSs specializes in a specific network protocol, it is necessary to modify the function every time the managed network changes, as mentioned in Section 1.

Therefore, the proposed architecture carries out this function independent of a specific network as a common function and can inject the characteristics of each network as external specifications.

II. PROPOSED ARCHITECTURE

A. Overview of Proposed Architecture

Figure 2 illustrates our proposed architecture.

The proposed architecture has two databases (DBs). The first is the Entity DB that holds physical and logical information of the network. The types of entities include physical resources that represent physical information of networks and devices and logical resources that represent logical connections and endpoints. The second is the Spec DB that holds specifications externally defining the characteristics of entity information held in the entity DB. Among the characteristics defined by the specifications, what capacity the entity consumes are described. The relationship between Spec and Entity DBs is described later in the paper.

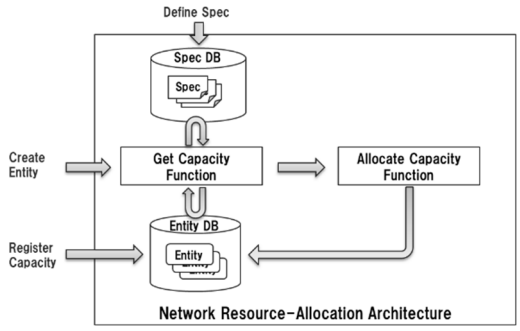


Fig. 2. Proposed architecture

A network service designer first defines managed network characteristics as specifications in the Spec DB. The operator then creates physical and logical resources in the Entity DB as he/she installs network devices and configures them. When creating these resources, the capacity is also registered in the resource to determine whether the capacity limit of the device is exceeded.

Based on the relationship of the communication protocol stack, each resource has an upper-lower relationship. For example, we consider a relationship in which an electrical signal flowing on a physical cable is encoded into an Ethernet frame, and an IP packet flows in the Ethernet frame. In this case, the logical resources of the Ethernet layer are defined above the physical resources, and the logical resources of the IP layer are defined above the Ethernet layer.

Within such upper and lower relationships, creating upper-layer resources consumes the capacity of lower-layer resources. Therefore, our architecture determines whether the upper-layer resource-capacity request is satisfied by the capacity of the created lower-layer resource when the upper-layer resource-generation request is received. If the request is satisfied, the architecture creates the upper-layer resource and updates the lower-layer resource to record the capacity consumed.

The flow of our architecture is as follows. Capacity allocation consists of the two functions, i.e., Get Capacity and Allocate Capacity, as shown in Fig. 2.

1) Get Capacity: The Get Capacity function obtains the capacity-consumption specifications of the requested upper-layer resource, and the lower-layer resource becomes a candidate to be consumed by the upper-level capacity-consumption specifications.

2) Judgement: The information is passed to the Allocate Capacity function, which obtains the capacity-consumption specifications and determines whether the lower-layer resource capacity satisfies the upper-layer request acccording to the specifications.

3) Create and Update Resources: If the capacity satisfies the requirements, the Allocate Capacity function creates the upper-layer resource and updates the lower-layer resource to record the capacity consumed.

Therefore, capacity-allocation specifications can be defined externally, and it becomes possible for an allocation function to be independent of a specific network. Next, we will explain the relationship between definition of specifications, entities, and capacity allocation specifications.

B. Definition of Specifications & Entities

To develop the proposed architecture, we used an information framework standardized by the TM Forum [4]. The information framework is the Shared Information and Data model (SID) [5], which defines management models common to the telecommunications industry.

The SID defines many classes, their attributes, and relationships among them. The classes include Resource Entity, which means a managed object. A resource entity is characterized by its resource specifications. As shown in Fig. 3, a resource-entity class and resource-specification class are defined separately in the SID.

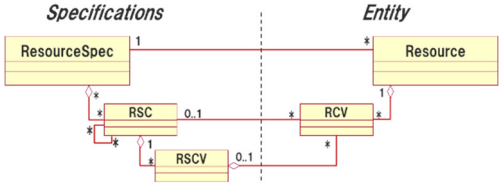


Fig. 3. Specifications and entity of SID models

Our proposed architecture has a Spec DB and Entity DB. The specifications for each resource can be modeled using three classes.

- ResourceSpecification (ResourceSpec)
- ResourceSpecCharacteristic (RSC)
- ResourceSpecCharacteristicValue (RSCV)

ResourceSpec defines the characteristics of a resource entity, which holds a reference to indicate the underlying ResourceSpec. The attributes of resource entities are characterized by the RSC and RSCV. In each case, the RSC means the attribute name, and RSCV means allowable values. An attribute of a resource entity is defined as a ResourceCharacteristicValue (RCV) and holds a reference to the base RSC.

An example of a resource specification and resource entities is shown in Fig. 4. To provide an Ethernet virtual private network (EVPN) service, a simple network with two layer-2 (L2) SWs is deployed and a virtual local area network (VLAN) is configured.

Resources include physical and logical resources.

A physical resource is a resource of a physical layer such as a communication device, physical port, or cable. Physical resources are modeled by classes that inherit a PhysicalResource class [6] such as

- PhysicalDevice (PD)
- PhysicalPort (PP)
- PhysicalLink (PL) .

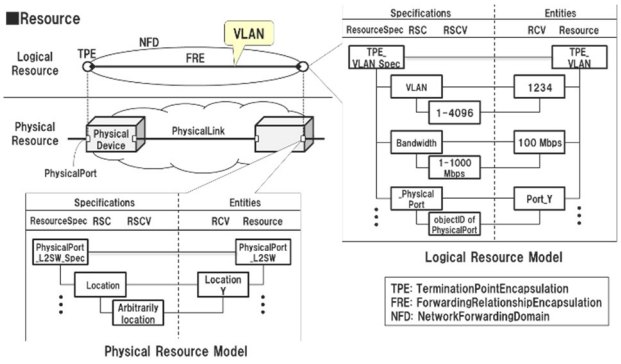


Fig. 4. Example of resource specifications and entity for EVPN service

A logical resource is a resource of a logical layer, such as an endpoint of a network protocol, communication path, and domain that can set paths. Logical resources are modeled by classes that inherit a LogicalResource class [7] such as below.

- TerminationPointEncapsulation (TPE)
- ForwardingRelationshipEncapsulation (FRE)
- NetworkForwardingDomain (NFD)

A VLAN connection is modeled as an FRE, endpoints of the VLAN are TPEs, and the area where the VLAN can be created is modeled as an NFD.

In Fig. 4, "PhysicalPort_L2SW_Spec and RSC/RSCV" means a characteristic of a L2SW's port. Its RSC means an attribute name (Location) and its RSCV means allowable values (available location).

Similarly, in the case of logical resources, "TPE_VLAN_Spec and RSC/RSCV" means a characteristic of a VLAN endpoint. Its RSC means an attribute name (VLAN, Bandwidth, _PhysicalPort) and its RSCV means allowable values (1–4096, 1–1000 Mbps, ObjectId of physical port).

The _PhysicalPort attribute is for storing ObjectID of physical port for indicating a reference from the logical TPE entity to the belonging PhysicalPort entity. In our architecture, the upper-lower relationships between entities are expressed by holding ObjectIDs to refer to related entities. This makes it possible to express the multiple relationships of logical and physical layers.

C. *Capacity expression and specification definition*

We can assume that lower-level entities provide capabilities such as transfer with higher-level entities. For example, lower physical cable entities have capabilities to transfer upper-level Ethernet entities because the upper Ethernet frame is transferred by the electric signal of the lower physical cable. In our architecture, the lower entity is called the capacity-providing entity.

The capacity-providing entity holds the amount of capacity for the higher level entity. For example, transfer bandwidth (e.g. 100 Mbps) and memory capacity (e.g. 8 GB).

Our architecture can hold such capacity-consuming/-providing relationships. The capacity attribute was defined with reference to SID Capacity [8]. As shown at the bottom of Fig. 5, such capacity is expressed in the attributes of entities in our architecture. This example expresses that SW1-SW2_PL has a capacity of 100-Mbps bandwidth by using Capacity name, unit, and amount.

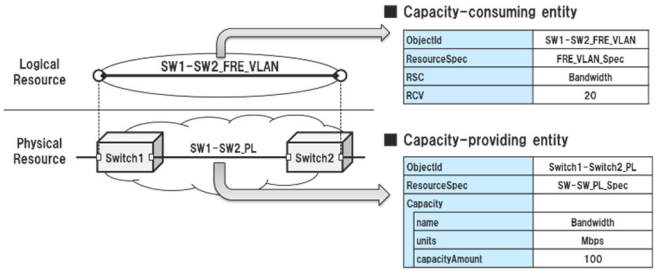


Fig. 5 Capacity-consuming/-providing entities

On the other hand, the upper entity is provided by consuming the capacity of the lower entity and is called a capacity-consuming entity. At the top of Fig. 5, the SW1-SW2_FRE_VLAN entity is a capacity-consuming entity and requests to consume 20 Mbps of 100-Mbps bandwidth of the lower PL.

Our proposed architecture includes the ResourceSpec, which describes the specifications that determine which entity consumes capacity. In the example in Fig. 5, the capacity-consuming entity SW1-SW2_FRE_VLAN is generated based on FRE_VLAN_Spec, and the specifications describe capacity consumption.

Next, we describe how to define the capacity-consumption specifications for identifying lower-level entities from upper-layer capacity-consuming entities. We add the definition of ConsumeCapacityInfo to the ResourceSpec (FRE_VLAN_Spec in Fig. 6), which is the basis of the capacity-consuming entity. ConsumeCapacityInfo has the following elements.

- Attribute: It shows an attribute that consumes capacity. In the example in Fig. 6, it refers to the bandwidth attribute defined in the RSC.
- Provider: It shows how traces between entities for identifying the lower-level capacity-providing entity and an array element. Each array element has ObjectType, ReferDirection, ReferKey, ConstrainedKey, and ConstrainedValue to be traced.

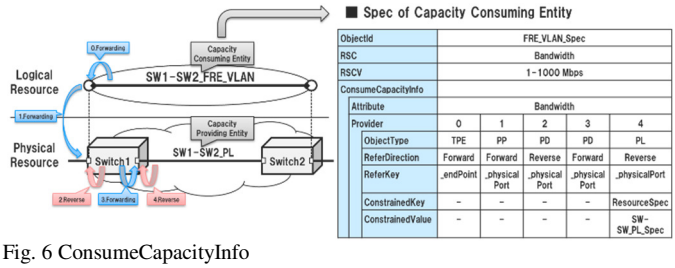


Fig. 6 ConsumeCapacityInfo

D. *Procedure of Allocate Capacity Function*

The Allocate Capacity function in Fig. 2 reads the capacity consumption specifications and determines whether allocation is possible. This flow is described based on Fig. 6.

Figure 6 is a situation in which the Allocate Capacity function attempts to create SW1-SW2_FRE_VLAN as a capacity-consuming entity based on FRE_VLAN_Spec, and the entity requires 20-Mbps bandwidth. This function searches for a capacity-providing entity that meets the 20-Mbps demand based on the capacity-consumption specifications.

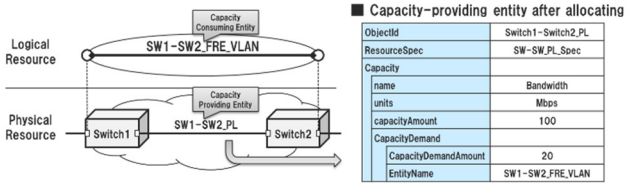
At index 0 of the Provider array, ObjectType is TPE, ReferDirection is Forward, and ReferKey is _endPoint. By using this information, this function searches a TPE entity referred from SW1-SW2_FRE_VLAN. The reference

relationship from the FRE to TPE is indicated by the ObjectId of the TPE referenced in the _endPoint" attribute of the FRE. For the next index, Index 1, this function similarly executes a process of searching for a PP referenced from the searched TPE. Index 2 is a process of searching for the PD from PP, but ReferDirection is Reverse. In this case, the reference relationship is indicated not by the PP to PD direction but by the object ID of the PP referenced by the _physicalPort attribute of the PD. In the last index, Index 4, when searching for the PL from PP in the reverse direction, this function uses ConstrainedKey/Value to select one that satisfies the condition (here, ResourceSpec of the candidate PL must be SW-SW_PL_Spec and SW1-SW2_PL is finally selected).

After finding the capacity-providing entity, it is determined whether the capacity of the entity satisfies the requirements of the capacity-consuming entity of the upper layer. As described in Fig. 5, capacity-consuming entity SW1-SW2_FRE_VLAN requests 20 Mbps and capacity-providing entity SW1-SW2_PL has 100-Mbps bandwidth. As a result, the request "20 Mbps" is satisfied by "100 Mbps".

Thus, this function can identify the capacity-providing entity and determine whether the request is satisfied. Because the process of this function is controlled by the capacity-consumption specifications, the functional process and specifications are separated. This enables the easy customization of the process in accordance with the operating network by modifying the specifications.

When determining the generation of the entity by the above function, it is necessary to record in the capacity-providing entity that the capacity has been consumed by that entity. We describe how to record this in Fig. 7. We add attribute CapacityDemand in the entity definition. CapacityDemand consists of the ObjectId of the entity that has requested the capacity consumption and the required amount (CapacityDemandAmount).



■ Capacity-providing entity after allocating

ObjectId	Switch1-Switch2_PL
ResourceSpec	SW-SW_PL_Spec
Capacity	
name	Bandwidth
units	Mbps
capacityAmount	100
CapacityDemand	
CapacityDemandAmount	20
EntityName	SW1-SW2_FRE_VLAN

Fig. 7 Capacity-providing entity after allocating

Finally, we show the class diagram of the entity added with capacity related definitions.

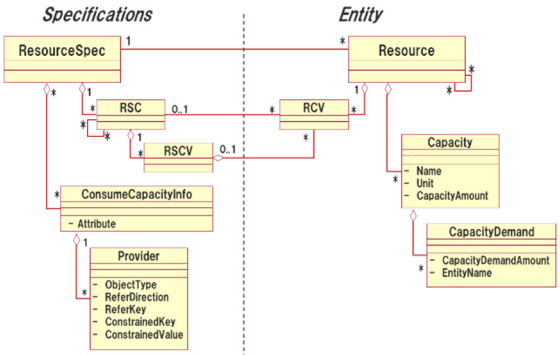


Fig. 8 Class diagram related to capacity definition

III. CONSIDERATION

In this section, we give two examples of making capacity decisions and consider the behavior of allocating resources.

A. *Application Example of EVPN*

The first example is the EVPN SW network in Fig.1. The remaining bandwidths of the network are expressed at the top of Fig. 9 (e.g. the remaining bandwidth of SwitchA-SwitchB_PL is 20 Mbps because 80 of 100 Mbps have already been consumed by FRE_VLAN_100). There is a service request for User A to this network, and a network operator tries to create a logical network that allocates 10-Mbps bandwidth. The capacity consumption specification for this network is the same as ConsumeCapacityInfo (from the VLAN layer to the EVPN SW network of the physical-resource layer) shown in Fig. 6. When the Allocate Capacity function executes based on this specification, the function does not select physical links having less than 10 Mbps (such as SwitchA-SwitchC_PL) as a result of judging whether it satisfies the request, and the function generates logical resources that refer to physical links exceeding 10 Mbps. The generated result is illustrated in Fig. 10, and the record of CapacityDemand is shown at the bottom of Fig. 9.

■ Capacity-providing entity before allocating

ObjectId	SwitchA-SwitchB_PL	SwitchA-SwitchC_PL	SwitchB-SwitchE_PL	SwitchC-SwitchE_PL	.....
ResourceSpec	SW-SW_PL_Spec	SW-SW_PL_Spec	SW-SW_PL_Spec	SW-SW_PL_Spec	
Capacity					
name	Bandwidth	Bandwidth	Bandwidth	Bandwidth	
units	Mbps	Mbps	Mbps	Mbps	
capacityAmount	100	100	100	100	
CapacityDemand					
CapacityDemandAmount	80	97	90	-	
EntityName	FRE_VLAN_100	FRE_VLAN_101	FRE_VLAN_102	-	

■ Capacity-providing entity after allocating

ObjectId	SwitchA-SwitchB_PL	SwitchA-SwitchC_PL	SwitchB-SwitchE_PL	SwitchC-SwitchE_PL	.....
ResourceSpec	SW-SW_PL_Spec	SW-SW_PL_Spec	SW-SW_PL_Spec	SW-SW_PL_Spec	
Capacity					
name	Bandwidth	Bandwidth	Bandwidth	Bandwidth	
units	Mbps	Mbps	Mbps	Mbps	
capacityAmount	100	100	100	100	
CapacityDemand					
CapacityDemandAmount	80 / 10	97	90 / 10	-	
EntityName	FRE_VLAN_100 / SWA-SWB_FRE_VLAN_UserA	FRE_VLAN_101	FRE_VLAN_102 / SWA-SWB_FRE_VLAN_UserA	-	

Fig. 9 Capacity-providing entities about EVPN

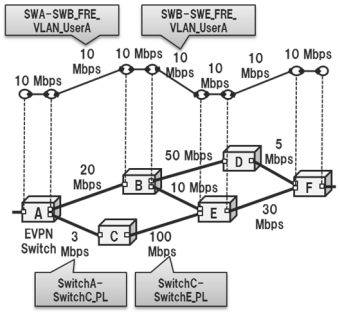


Fig. 10 EVPN after creating logical resources

B. *Application Example of EVPN and OTN*

The second example is a network combining EVPN and optical transport network (OTN). As shown at the bottom of Fig. 11, EVPN SWs (A and E) are at both ends, and OTN SWs (B, C, and D) are relaying between them. In this network, the network operator needs to manage the number of OTN wavelengths in addition to the Ethernet bandwidth capacity. This figure also shows the remaining amount of bandwidth in

the link between an EVPN SW and OTN SW and the number of remaining wavelengths in the link among the OTN SWs.

Similar to the first example, we assume that there is a 10-Mbps user request. The connectivity of the Ethernet layer transferred by the link from EVPN SW A to OTN SW B is expressed by the logical resources of the Ethernet layer. For these logical resources, the consumption specification is defined by "Capacity consumption spec1" shown in Fig. 12(this specification is the same as that of Fig. 6). As a result, the resource entities consume the bandwidth capacity held by the physical link.

The link among OTN SWs B, C, and D plays the role of relaying Ethernet transfer by OTN, and logical resources are expressed by two layers of the Ethernet and OTN layers. We determined whether the remaining number of wavelengths of the OTN layer is one or more, and one wavelength is consumed to generate logical resources. For these logical resources, the consumption specification is defined by "Capacity consumption spec2" shown in Fig. 12. This specification indicates the order of searching the OTN layer from the Ethernet layer and searching the physical layer from the OTN layer (from Indices 0 to 5). According to this order, the Allocate Capacity function searches physical links and checks the remaining wavelength number of each selected link. As a result, there is no remaining number in the link between SWs B and D, and there are remaining numbers in the link between SW B to SW C and SW C to SW D. Therefore, the Allocate Capacity function consumes the capacity of the links from SW B to C and SW C to D and generates logical resources of the Ethernet and OTN layers.

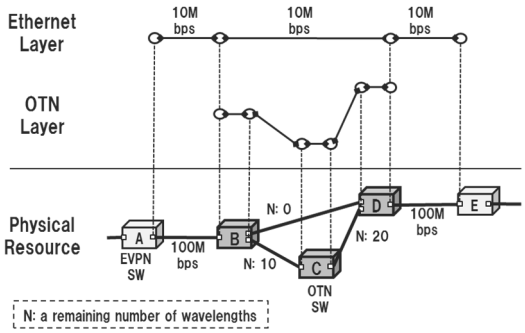


Fig. 11 Combined network of EVPN and OTN

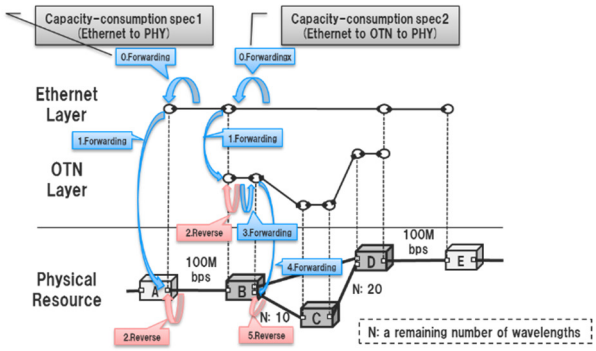


Fig. 12 Capacity-consumption specifications for EVPN/ OTN network

Thus, we can apply the proposed architecture to the network combining EVPN and OTN.

IV. CONCLUSION

We proposed a resource-allocation function extended from our previous architecture that has common management functions and data that are not specialized for a specific network device and protocol. This architecture can manage the network capacity amount and flexibly allocate it according to user demand. The allocation function of the proposed architecture has an execution procedure, and the capacity-consumption specifications are separated, enabling easy customization of the process in accordance with each network by modifying the specifications.

We also showed that the proposed architecture can be applied to two different networks (EVPN and combined EVPN/OTN networks).

REFERENCES

[1] TM Forum, "Open Digtal Architecture Whitepapers," August 2018.

[2] S. Horiuchi, K. Akashi, M. Sato, and T. Kotani, "Network Resource Management Architecture with Unified Information Models," APNOMS 2017, pp. 489–499, Sep. 2017.

[3] K. Akashi, M. Sato, S. Horiuchi, and T. Kotani, "OSS Architecture and Order Mapping Function for Providing Various Services," APNOMS 2017, pp. 500–513, Sep. 2017.

[4] TM Forum, https://www.tmforum.org.

[5] TM Forum GB921, "Core Frameworks Concepts and Principles: Business Process, Information and Application Frameworks," Version 16.0.1, April 2016.

[6] TM Forum GB922, "Information Framework (SID): Physical Resource Business Entities," Version 16.0.1, May 2016.

[7] TM Forum GB922, "Information Framework (SID): LogicalResource and ConmpoundResource Bisiness Entities," Version 16.0.2, May 2016.

[8] TM Forum GB922, "Information Framework (SID): Common Business Entities - Capacity," Version 15.0.1, Nov 2015.