Implementation of a Novel Management Development Platform for Virtual Networks

Wenyu Shen, Kenji Minato, Yukio Tsukishima, Katsuhiro Shimano
NTT Network Innovation Laboratories
1-1 Hikari-no-oka, Yokosuka, Kanagawa 239-0847 Japan
wenyu.shen@lab.ntt.co.jp

***Abstract*—Virtual networks, although regarded as a candidate for future networks, still suffer from a complex management plane. A hierarchical role model was proposed that represents different forms of network virtualization in a simple and unified way. As an application of the role model, this paper implements a novel management development platform for virtual networks. By using the proposed platform, a developer can develop modularized management systems of virtual networks through simply writing definition files. As a result, the platform has the potential to reduce greatly the development cost. Finally, a prototype system is constructed to validate the feasibility. Although this research is still ongoing, we believe that it can be a catalyst for further deployment of network virtualization technology.**

Keywords—network virtualization; network management system; OSGi; modularization

I. INTRODUCTION

Network virtualization technology has been regarded as a candidate solution for future networks that satisfies the carrier demand for rapid deployment of new network services and infrastructure technologies with lower capital expenditure (CAPEX) and operating expense (OPEX) [1-4]. However, due to the dynamism and polymorphism of virtual networks, their management systems become extremely complex. In fact, the data plane is complicated by the dynamism of virtual networks. Objects being managed change on the fly with the generation and deletion of virtual networks. Furthermore, different organizations still utilize different network virtualization definitions resulting in a mass of managed objects in different forms. As a result, it is still expensive to develop such a system using the traditional development approach.

Fortunately, a hierarchical network model called the Role Model (RM) was proposed that is able to represent different forms of network virtualization in a simple and unified way [5]. The theory behind the RM is that the essence of network virtualization is simply the process of defining a network view of a specific abstraction level. Moreover, a new network view is generated from an existing network view or physical networks, and this process is iterated in order to generate more network views [5].

Based on the consistency and hierarchical feature of the RM, we hypothesize that it is possible to generate a new management system for virtual networks by writing simple and structured definition files to some extent, instead of the traditional development approach, which is based entirely on a source code. In this way, the development cost is expected to be reduced greatly. In order to prove this, we implement a novel management development platform for virtual networks based on OSGi [6], the features of which are summarized below.

- The development platform supports dynamic generation and customization of modularized management systems with the changes of the managed virtual networks.
- The framework of a management system such as the module interfaces and interaction behaviors between different modules can be generated by writing a role definition file (RoDF), the structure of which strictly applies to the RM.
- As an essential part of a management system, event processing functions such as the function to prevent mass event cascades across different modules can be generated by writing a Rule Definition File (RuDF). To achieve this, we implement a general event processor by extending the OSGi Event Admin Service (EAS) [6].

The rest of the paper is organized as follows. Section 2 discusses the hierarchical role model, which is the basis of this implementation. Section 3 describes the implementation details of the proposed development platform including the whole structure and the related definition files, i.e., RoDF and RuDF. In order to validate the implementation, we utilize it to assist in the development of a management system for a next generation mobile service based on OpenFlow [7]. The details are given in Section 4. Finally, Section 5 concludes the paper and describes some open issues.

II. HIERARCHICAL ROLE MODEL

A. Review of Concept

This section briefly reviews the RM, which forms the basis of the proposed management development platform. The design details of the RM can be found in [5]. When people describe the concept of "network virtualization," actually a specific network view is defined. Different forms of network virtualization differ only in the abstraction level of the

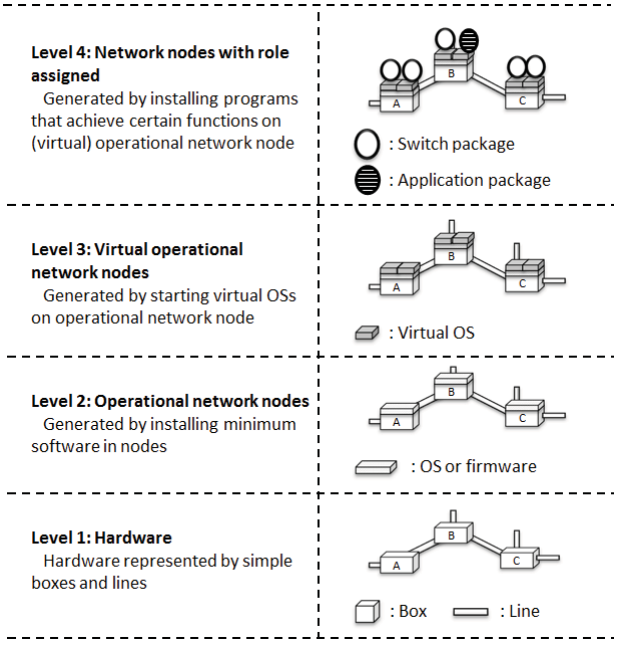


Fig. 1. Primitive roles and their generation method.

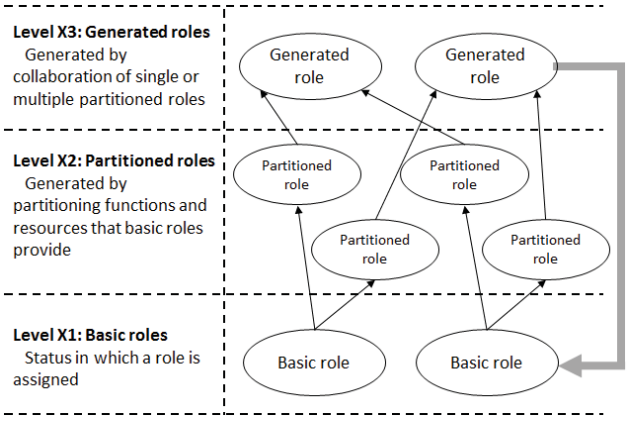
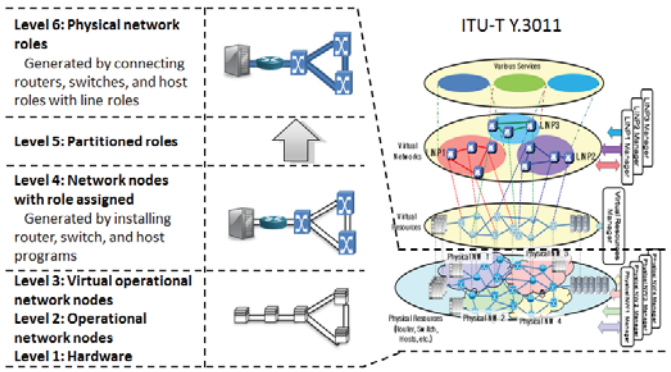


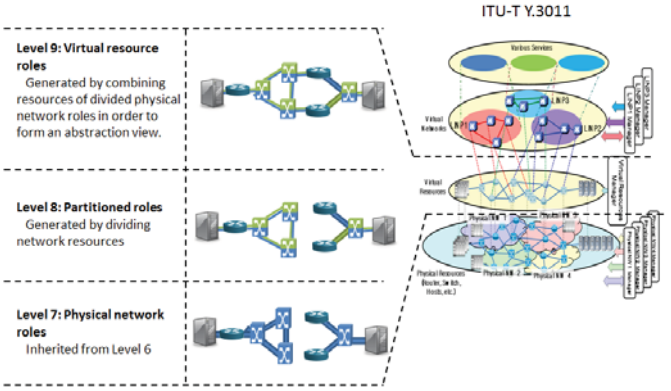
Fig. 2. Iterative generation of new roles.

network views. We note here that different network views have similar generation processes, a new network view can be generated from an existing network view or physical networks, and this process is iterated yielding more network views. As a result, the combination of these generated network views forms a hierarchical structure. Inspired by the hierarchical characteristics of network virtualization, the RM was proposed, which is able to represent different forms of network virtualization in a simple and unified way [5].

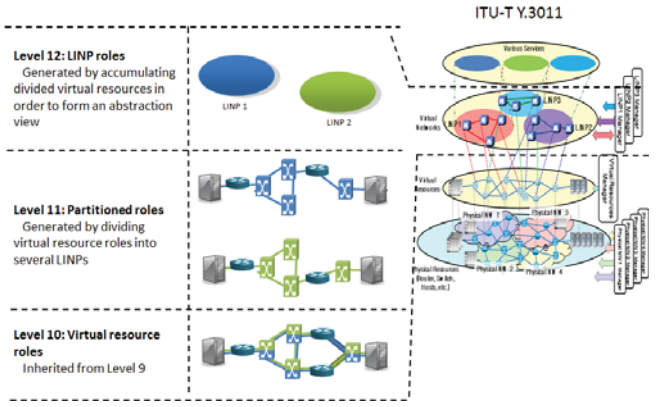
As the name suggests, the core concept behind this model is a role that reflects a specific network view. In this model, as illustrated in Fig. 1, installing an operating system (OS) in Level 2, virtual OSs in Level 3, and programs in Level 4 on common hardware generates a primitive role, the most basic component, i.e., a specific network node performing certain functions, such as an IP router or an Ethernet switch. Starting from the primitive roles, Fig. 2 shows that new roles are generated iteratively through a series of actions: role assignment (X1), role partitioning (X2), and role collaboration



(a) Mapping of physical resources



(b) Mapping of virtual resources



(c) Mapping of virtual networks

Fig. 3. Mapping of the network virtualization model defined in ITU-T Y.3011 to the RM.

(X3). This yields higher roles such as network roles, path roles and session roles, which are necessary to describe virtual networks. In this way, all forms of network virtualization or abstraction can be represented using roles, which are generated from roles in a lower level and this cycle can be traced back to the roles initially assigned to hardware. As a result, as long as the relationships between physical entities and the initially assigned roles can be managed, other roles can be represented independently on the physical entities in the management plane.

B. Mapping of ITU-T Network Virtualization Model to the Role Model

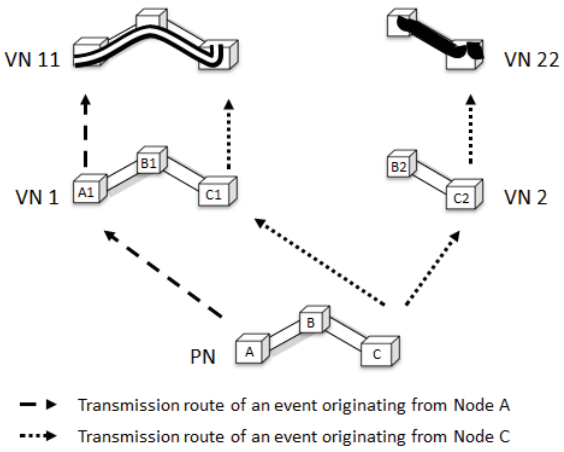


Fig. 4. Example of event distribution among virtual networks.

Fig. 3 illustrates a mapping between the virtualization model defined in ITU-T Y.3011 [2] and the RM. For the mapping of physical resources defined in Y.3011, network nodes with OSs installed are first assigned roles such as router roles, switch roles, and host roles, after which these roles collaborate with each other generating a role in the physical network. A mapping of virtual resources in Y.3011 to the RM is illustrated in Fig. 3(b). In Level 7, multiple physical networks in Level 6 are positioned as basic roles. In Y.3011, it states that "physical resources are partitioned and abstracted as virtual resources," which can be mapped to role partition and role collaboration in the RM [2]. In Level 8, the physical network roles are divided into several groups, and the resources of the divided physical network roles are gathered together to form an abstraction view, which equals a new generated virtual resource role in Level 9. A mapping of virtual networks to the RM is illustrated in Fig. 3(c). Iteratively, the virtual resource roles become the basic roles in Level 10. In Level 11, the virtual resource roles are divided into several groups, similar to the Logical Isolated Network Partition (LINP) defined in Y.3011 [2]. Finally, the divided virtual resources are accumulated to generate the new role of the LINP.

C. Consideration of Event Management in the RM

In a network virtualization environment, fault management becomes extremely difficult due to multiple layers of virtual and physical networks. Therefore, in order to implement fault management such as fault detection and root cause analysis, event distribution among all related virtual networks and physical networks should be controlled carefully, which is a complicated task. Moreover, since virtual networks are not static, the problem becomes even more complicated given that the event distribution among virtual networks must be modified dynamically following their dynamic generation and deletion. Fig. 4 is a simple example. Let us assume that a physical network (PN) consists of Nodes A, B, and C. Virtual network (VN) 1 consists of Virtual Nodes A1, B1, and C1, and VN 2 consists of nodes B2 and C2, which are all generated from the PN. Furthermore, we suppose that from VN 1 and VN 2, VN 11 and VN 22 are generated. In this case, when Node A fails, in order to indicate an error in every related virtual network, an event should be transmitted to VN 1 and further transmitted to VN 11, while when Node B or C fails, the event should be transmitted to all the virtual networks (VN 1, VN 2, VN 11, and VN 22) in the figure, since they all rely on the PN. Compared to Fig. 2, it is clear that while acting as a resource management model for network virtualization, the RM also helps to express event distribution among different virtual networks.

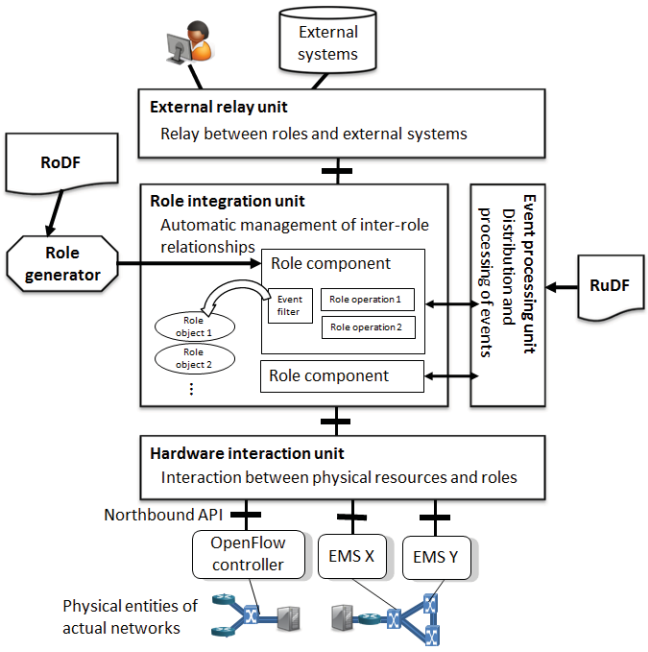


Fig. 5. Structure of proposed development platform.

III. Implementation of Development Platform

Based on the RM described above, we conclude that according to the defined relationship between the roles at different levels, some of the source code of the management system such as the skeleton code of the role managers and inter-role interaction, e.g., resource allocation/event distribution, can be automatically generated. Therefore, we implement our management development platform, which mainly consists of a tool (the role generator) that can automatically generate the referred source code from a RoDF and an executor (the role integration unit) that actually executes the generated source code. In addition, we implement a general event processor called EAS+ by extending the OSGi EAS. This event processing function is particularly designed to prevent mass event cascades across modules within a management system, and similarly it can be customized using a RuDF to meet the needs of a specific system. It is expected that the management system for virtual networks can be developed in a short time with the assistance of the implemented development platform.

Fig. 5 illustrates the structure of the proposed development platform. First, three terminologies are clarified: role component, role object, and role module. Role components are the programs generated by the role generator from the RoDF. One role component exists to manage a set of role objects of one type. In fact, a role component acts as a factory program

that generates and deletes role objects. In the implementation, role components are implemented as OSGi bundles. Role objects represent role instances. In fact, they are structured objects that contain the attribute data of a role. Role modules are programs such as common functions and father classes of existing role components, which are stored in a server and they can be reused. Let us consider path management as a simple example. A management system that performs operations such as path creation and deletion is regarded as a role component. Here, different paths are generated and deleted as different role objects. Algorithms such as path computation are reusable and are regarded as role modules.

A. Role Integration Unit

The role integration unit instantiates the role components generated by the role generator and automatically maintains the relationship among individual instances. By maintaining information consistency among role components, the role integration unit enables dynamic addition and deletion of role components and even the implementation of event distribution among role components. The role integration unit is implemented using the OSGi framework.

B. Hardware Interaction Unit

The hardware interaction unit represents physical entities internally as primitive roles and manages the relationship between the physical entities and the primitive roles. In addition, it provides the role integration unit with interfaces to these primitive roles. The primitive roles are roles that are initially assigned to hardware and are designed in order to absorb the differences among various kinds of network equipment.

C. External Relay Unit

The external relay unit offers network operators a way to mediate external systems or Graphical User Interfaces (GUIs) using the role instances provided by the role integration unit.

D. Role Generator

The role generator creates role components that are defined in the RoDF. Since the role integration unit is based on OSGi, the role generator will generate the source code for role component correspondent bundles and download them into the OSGi platform. When event distribution among roles is considered, the role generator generates an event filter and the skeleton code of an event handler for each type of event. The event filter is set automatically according to the role relationship as described in the RM so that only the events destined for the role component itself will be received. Moreover, an event handler skeleton is generated so that an event and the corresponding event handler that processes the event are combined.

E. RoDF

Since the development platform is designed based on the RM, all the management objects should be roles. Therefore, a RoDF contains only actual role definitions, which cover network resources related to a role, the inter-role relationship, and the event distribution among roles. The basic elements of a role definition file are given in Table I.

TABLE I. BASIC ELEMENTS OF RODF

Basic Element	Explanation	Example
Role name	Identification of individual roles	Path
Attributes	Information to describe a role	♦Path constitution (Ex. node1<>link<>node2…) ♦Bandwidth
Operations	Possible operations on a role	♦CreatePath(); ♦DeletePath();
Inter-role constraints	Basic constraints that guarantee the existence of a role	♦Topology constraints (Ex. necessary lower roles and their combination sequence) ♦Resource constraints (Ex. relationship with the resources in lower roles)
Event list	List of events intended to be received	♦MESG_NODE_FAIL ♦MESG_LINK_FAIL

F. Event Processing Unit

The event processing unit is based on EAS+, which is an extension of the OSGi EAS. EAS+ specifically targets mass event cascades across different modules. As a promising feature, EAS+ can be customized by a simple and structured RuDF in order to meet the needs of different systems. Based on several existing works [8, 9], EAS+ was designed to be as general as possible, so it can be applied easily to almost all network management systems. Fig. 6 shows the structure of EAS+, which further comprises a rule-setting interface, event classifier, aggregated event generator, and event distributor. In the design of EAS+, rules such as filtering rules and accumulation rules must be set properly in order to achieve the desired event processing functions. The rule-setting interface is designed to load the RuDF that consists of these rules. The event classifier classifies all events into categories according to the filtering rules as written in the RuDF. It forwards the events to different destinations according to their categories. For example, normal events are directly forwarded to the event distributor while mass events are forwarded to the appropriate aggregated event generation instances. The aggregated event generator, regarded as a core component of

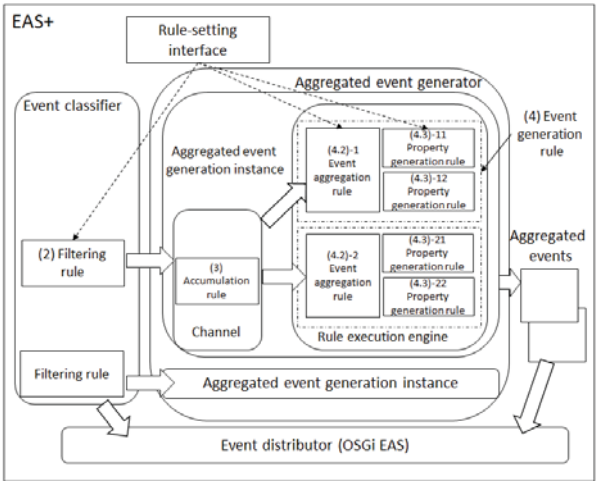


Fig. 6. Overall architecture of EAS+.

EAS+, generates aggregated events and forwards them to the event distributor. It holds multiple aggregated event generation instances that are generated for each filtering rule in the event classifier. The aggregated event generation instances comprise a channel and a rule execution engine so that the events forwarded from the event classifier are first accumulated in the channel, based on which new aggregated events will be generated by applying the event aggregation rules and the property generation rules in the rule execution engine. Finally, the aggregated events are forwarded to the event distributor. The event distributor provides an environment for distributing events among different components in the network management system. Luckily, the OSGi EAS perfectly satisfies this requirement here and can be used without change.

G. RuDF

A RuDF determines the functionality of the desired event processing functions. In a RuDF, each aggregated event generation instance has its own rule. The structure of a RuDF is specified below.

1) Instance name

The instance name is used to identify the individual aggregated event generation instance. A rule definition file consists of rules for multiple aggregated event generation instances, and the following items exist for each instance.

2) Filtering rule

The filtering rule is used to define the functionality of the event classifier. Since an event holds the topic and property attributes in the context of OSGi [6], a filtering rule in EAS+ considers both the topic and property of an event. In the case that an event matches both the topic and property of a filtering rule, it will be forwarded to the designated aggregated event generation instance.

3) Accumulation rule

The accumulation rule defines the attributes (maximum number accumulated and maximum accumulation time) of a channel in the aggregated event generation instance. In fact, it sets a trigger to forward a certain number of events buffered in the channel to the rule execution engine at the same time.

4) Event generation rule

The event generation rule is used to generate aggregated events, each of which consists of an event aggregation rule and one or multiple property generation rules. It should be noted that multiple event generation rules can be defined for a single aggregated event generation instance. Details of event generation rules are given below.

a) Aggregated event name

The aggregated event name represents the topic name of an aggregated event that will be generated by EAS+.

b) Event aggregation rule

The event aggregation rule is applied to the events buffered in the channel and decides how event aggregation is performed. All the events matching the event aggregation rule are expected to be aggregated to a single event.

c) Property generation rule

The property generation rule specifies the property information that will appear in the aggregated events that are newly generated. The following are details regarding the property generation rule.

i) Property name

The property name specifies a list of property keys of an aggregated event.

ii) Property type

The property type defines the type of information that appears in the property value: a list of original events or the number of events aggregated. If the property type represents the number of events aggregated, the object key name and topic existence flag items are unnecessary.

iii) Property aggregation rule

The property aggregation rule is actually a filtering rule that considers both the event topic and property. It considers the events that are buffered in the channel and is matched to the event aggregation rule above. In fact, it decides the content of aggregated events generated by EAS+.

iv) Object key name

The object key name decides the key names that will be written in the property value of the aggregated event.

v) Topic existence flag

The topic existence flag indicates if the original event topic names are to be written to the object key name as the property of an aggregated event.

IV. VALIDATION

In order to validate the feasibility of our implementation, we utilize it to assist the development of a management system for a next generation mobile service.

A. Design of Prototype Infrastructure

The prototype infrastructure was designed by referencing the 3GPP Long Term Evolution (LTE) mobile service, which is based on an all-IP based network [5, 10]. In the prototype infrastructure, voice services are enabled by dynamically allocating both network resources and server resources. The prototype infrastructure adopts both server virtualization and OpenFlow technology to achieve sufficient flexibility.

Fig. 7 shows a rough configuration of the prototype infrastructure. Since the IP multimedia subsystem (IMS) is used for voice services in this case, we prepare two data centers, called “DC-X” and “DC-Y,” to accommodate the functions such as the Interrogating-Call Session Control Function (I-CSCF), Proxy (P)-CSCF, and Serving (S)-CSCF in the IMS. The S-CSCF is actually actualized as two processes (p 1 and p 2), corresponding to two VMs, in order to implement load balancing for the incoming call requests.

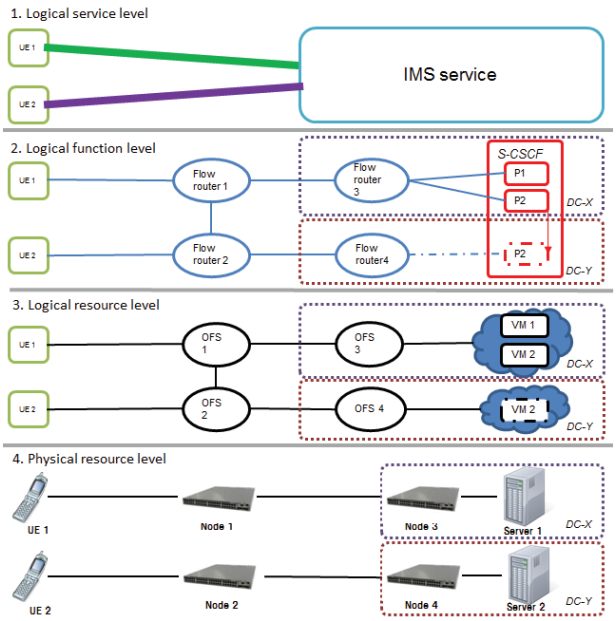


Fig. 7. Configuration of prototype infrastructure.

Initially, they are both located in DC-X. OpenFlow switch (OFS) 1 and OFS 2 provide application-aware virtual paths between the mobile terminals and the data centers, while OFSs 3 and 4 work inside the data centers in order to balance traffic.

B. Validation

We developed a management system for the prototype infrastructure by using the prototype development platform. As illustrated in Fig. 7, 4 levels of views (service, logical function, logical resource, and physical resource) were created and maintained in order to manage the prototype infrastructure. Regarding the 4 levels of views as 4 networks (3 virtual networks and 1 physical network), we designed a role architecture for the prototype infrastructure as illustrated in Fig. 8, which is the basis of the RoDF. In addition, we utilized EAS+ to assist the development of the event processing function, which handles event forwarding and processing among these components such as the CPU alarms and interface alarms.

The validation confirmed the functioning of the management system for the prototype infrastructure. If an S-CSCF process (p 2) failed, the remaining process (p 1) would become more highly loaded since it would receive an unexpected number of call requests. To deal with resource overloading in DC-X, the management system that is generated from the proposed management development platform responds by setting up a new process (p 2) in the other data center (DC-Y) which achieves load balancing. At the same time, the management system also automatically modifies the flow tables in the related OpenFlow switches in order to re-route traffic.

V. CONCLUSION

This paper presented an implementation of a novel management development platform for virtual networks,

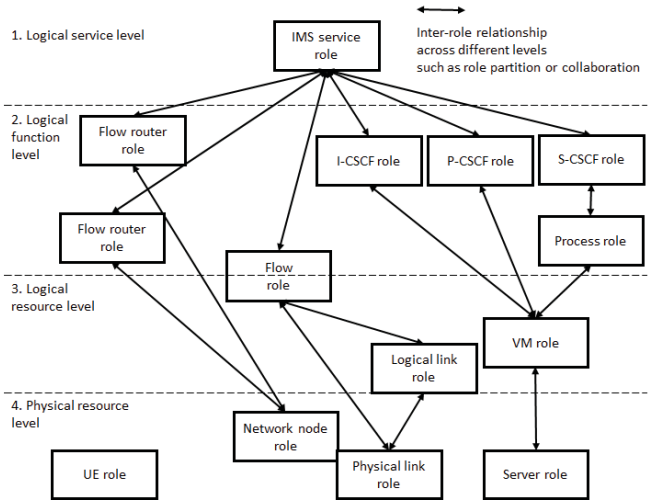


Fig. 8. Role architecture for prototype infrastructure.

through which a developer can develop modularized management systems for virtual networks by writing definition files. Although the prototype system validates the feasibility to some extent, there are still several unsolved problems. For example, we listed some basic elements of role definition files in this paper, however, their construction needs further research. File design (structure and element) will impact the functionality and effectiveness of the platform. In addition, there is the question of even how to evaluate our proposal. The degree to which the volume of source code can be decreased strongly depends on the system used and the number of role modules that can be reused, so more experiments are necessary. We believe that this research can be a catalyst for the further deployment of network virtualization technology.

REFERENCE

[1] N.M. Mosharaf, K. Chowdhury, and R. Boutaba, "Network virtualization: State of the art and research challenges," IEEE Communication Magazine, vol. 47 (7), pp. 20-26, July 2009.

[2] Draft ITU-T Recommendation Y.3011, "Framework of network virtualization for future networks," Oct. 2011.

[3] A. Galis, S. Denazis, C. Brou, and C. Klein, Programmable Networks for IP Service Deployment, Artech House, May 2004.

[4] ONF White Paper, "Software-defined networking: The new norm for networks," April 2012.

[5] W. Shen, K. Minato, Y. Tsukishima, and K. Shimano, "Management engine using hierarchical role model," in Proc. of IFIP/IEEE DANMS2013, Ghent, Belgium, May 2013 (to be published).

[6] OSGi Specification, "OSGi core release 5," March 2012.

[7] N. McKeown, *et al.*, "OpenFlow: Enabling innovation in campus networks," in ACM SIGCOMM Computer Communication Review, vol. 38 (2), pp. 69-74, April 2008.

[8] S. Kandula, D. Katabi, and J.-P. Vasseur, "Shrink: A tool for failure diagnosis in IP networks," In Proc. of ACM SIGCOMM workshop on Mining network data (MineNet), pp. 173-178, New York, 2005.

[9] R.R. Kompella, J. Yates, A. Greenberg, and A.C. Snoeren, "Detection and localization of network blackholes," In Proc. of IEEE INFOCOM, pp. 2180-2188, San Diego, USA, May 2007.

[10] Y. Nakajima, *et al.*, "Design and implementation of virtualized ICT resource and management system for carrier network services toward cloud computing era," in ITU Kaleidoscope 2013, April 2013.