Management Engine Using Hierarchical Role Model

A new management 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

***Abstract*—Network virtualization facilitates the carriers' introduction of new network services and infrastructure technologies with low CAPEX and OPEX. However, the virtual networks' dynamic and polymorphic features lead to a complex management plane. This paper proposes Role Model to represent network virtualization in a simple and unified way; we use it to design Management Engine, which can dynamically customize modularized management systems through simple and structured definition files. Finally, we develop a prototype and validate our proposals. We believe that Management Engine is a new management platform for virtual networks.**

Keywords—virtual network; OSGi; management system

I. INTRODUCTION

We are now experiencing a boom in new network services and infrastructure technologies, the handling of which will determine the carrier's fate in this competitive world. Another trend is the acceleration in the convergence of networks and their management systems to simplify both the data plane and management plane and satisfy demand for lower CAPEX (capital expenditure) and OPEX (operating expense). One possible solution is network virtualization technology [1, 2, 3].

The management in network virtualization is complicated by the dynamicity of virtual networks; objects being managed change on the fly with the generation and deletion of virtual networks, unlike the static managed objects in existing networks. Therefore systems to manage virtual networks must offer higher flexibility in functions and architectures. Unfortunately, although it is generally regarded that the flexibility of the management plane tightly relies on the simplicity and unification of the data plane, different organizations still hold different definitions of network virtualization [1, 2, 4], which leads to a mass of virtual networks in different forms.

Through extensive survey and comparison [1, 2, 4], we found that describing the concept of "network virtualization" actually defines a specific network view. Different forms of network virtualization differ only in the abstraction level of the network views. We note 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.

In this paper, inspired by the hierarchical characteristic of network virtualization, we propose Role Model (RM) which can be used to represent different forms of network virtualization in a simple and unified way. Second, based on the proposed model, we design a novel management platform for virtual networks, named Management Engine (ME), which can dynamically customize modularized management systems through simple and structured definition files. In order to validate ME, we develop a prototype system for a next generation mobile service based on OpenFlow [5].

II. REQUIREMENTS FOR MANAGEMENT PLANE OF VIRTAUL NETWORKS

Network management systems used to be designed to manage physical entities in actual network equipment, so a management system's internal functional composition was comparatively fixed. In contrast, the introduction of network virtualization technology means that managed objects now include logical entities of virtual networks; this raises an essential difference, each virtual network corresponds to a specific network service. In other words, virtual networks are no longer static and they are dynamically constructed with the introduction of new network services. Accordingly, the management system should customize its functions in order to adapt to changes in the data plane. Therefore, the requirement for flexibility can be summarized as below.

Requirement I: A highly flexible management plane whose functions can be dynamically customized in an easy way with the addition and deletion of virtual networks.

A simple and unified data plane has a huge impact on the fulfillment of Requirement I. However, the definition of network virtualization remains unclear. For example, Open Networking Foundation (ONF) defines network virtualization simply as an abstraction view of OpenFlow networks [4], while in ITU-T's definition network virtualization reflects partition and recombination of physical resources [1]. Furthermore, in a Japan's national project, named "Virtualized Node", virtual networks are generated by adding software functions to a common hardware platform such as Linux servers [2]. Therefore, Requirement II raises the need for unification.

Requirement II: A management model through which different forms of network virtualization can be represented in a simple and unified way.

978-3-901882-50-0

Although different forms of network virtualization coexist, they exhibit some similarity. We believe that the essence of network virtualization is simply the process of defining a specific network view of a specific abstraction level. Moreover, different network views can have similar generation processes, in which a new network view is generated from an existing network view or physical networks, and this process is iterated to generate more network views. As a result, the combination of these generated network views forms a hierarchical structure. Inspired by the hierarchical characteristic of network virtualization, we propose RM, which can be used to represent different forms of network virtualization in a simple and unified way, see Section 3.

III. HIERARCHICAL ROLE MODEL

As the name suggests, the core concept in this model is a role which reflects a specific network view. In this model, installing programs on common hardware generates the 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, new roles are generated iteratively through a series of actions: role assignment, role partitioning, and role collaboration. This yields higher roles such as network roles, path roles and session roles, which are necessary to describe virtual networks.

A. Primitive Roles and their Generation

Fig. 1 shows the iterated process, Level 1 to Level 4, of generating primitive roles from common hardware. Level 1 represents the most basic physical entities in actual networks; the only components are boxes and lines. Boxes are endpoints in networks performing certain functions, and lines are media connecting the boxes. Level 2 indicates the installation of minimum software, such as Operating Systems (OSs), on the boxes, so that the boxes become operational network nodes. Level 3 is only necessary when a virtual OS exists. In fact, it is the lowest level form of network virtualization. Level 4 indicates the installation of programs that realize certain functions on OSs or virtual OSs. A box takes charge of a certain role when the programs on it start running, so we represent the action of starting programs as the assignment of a certain role to a network node. These initially assigned roles are regarded as primitive roles.

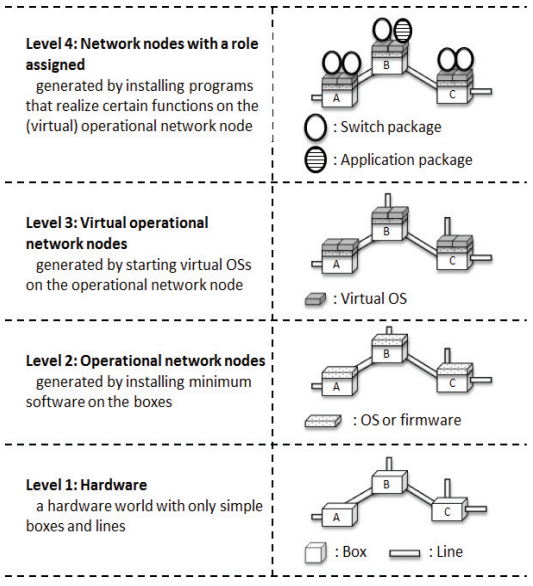


Fig. 1. Primitive roles and their generation

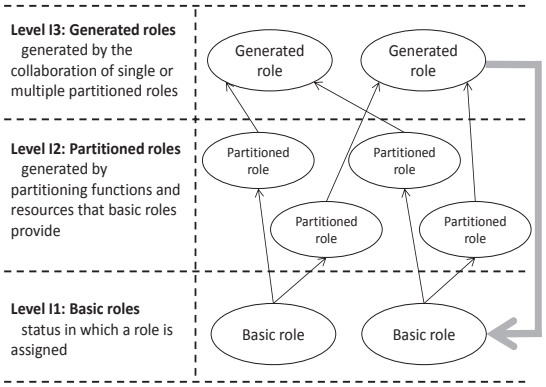


Fig. 2. Iterative generation of new roles

B. Iterative Generation of New Roles

As shown in Fig. 2, starting from the primitive roles in Level 4, more roles can be generated iteratively. Level I1 consists of basic roles, which are defined as abstraction of lower levels. One or more basic roles of different types are possible. In Level I2, the functions and resources that basic roles provide are split into partitioned roles, which are isolated from each other. Some basic roles cannot be split, in which case, the basic roles are directly set as partitioned roles without any change. Finally, a single partitioned role, multiple partitioned roles of a single type, or multiple partitioned roles of different types can collaborate to generate a new role as illustrated in Level I3. It is possible that a partitioned role is directly accepted as a new role. The generated roles are actually an abstract view of the lower roles, so the generation process here can be regarded as a form of network virtualization from Level I1. Furthermore, a new generated role can be positioned as a basic role in an upper level so that the role generation process can be iterated continuously.

C. Application Examples

1) Representation of networks, paths, channels and domains

Fig. 3 illustrates the generation of network roles from the network node roles in Level 4 in Fig. 1. The Virtual OSs on the boxes are divided into two groups in Level 5. Each group shares the resources of boxes, and at the same time keeps separation from each other. In addition, we divide the line roles into two groups in the same way. Two new network roles are generated by connecting the partitioned roles in Level 5 using the lines between the corresponding boxes. In Level 6, each box is represented as different network equipment according to the different roles that are assigned. For instance, box B works as a switch in the left network, while it works as an application server in the right network.

Based on the generated network roles, we can further generate path roles and channel roles by using the iterated generation process in Fig. 2. It should be noted that the path roles and channel roles explained here are just examples and

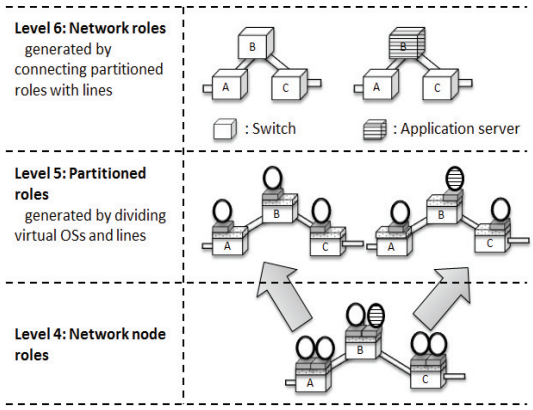


Fig. 3. Network roles and their generation

more detailed role design should be considered for actual scenarios. For unique level names, we use "Level Xx" in the following explanation.

We reuse the network roles generated above as the basic roles in this new cycle. Furthermore, in X2, we divide the switch roles by partitioning the switches' forwarding tables, and divide the line roles by partitioning each line's bandwidth. Finally, Level X3 holds new path roles generated by the collaboration of the partitioned switch roles and line roles.

The generation of channel roles is almost the same as that of path roles, so we omit a detailed explanation, due to the limited space. Fig. 4 illustrates the generation of path roles and channel roles from the network roles in Level 6.

Last, but not least, sometimes there is need to hide all network details so that the network looks like a cloud. This often happens when a network spans several domains which are run autonomously by different organizations. Luckily, we can simply generate a domain role to meet this need. As illustrated in Fig. 4 (from X3-1 to X3-3), omitting the process of role partitioning, we generate a domain role by putting the path roles together as a logical switch.

2) Representation of network virtualization defined in ITU-T Y.3011

We try to map between the virtualization model defined in

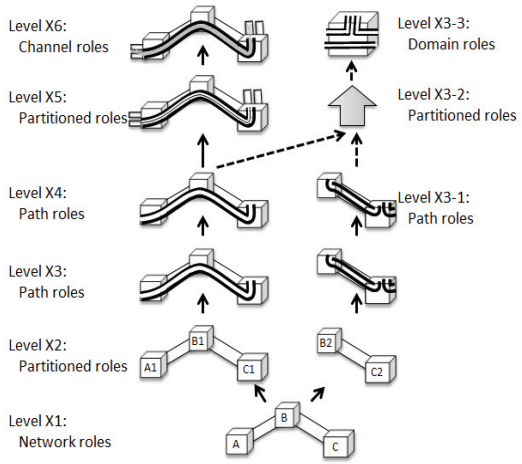


Fig. 4. An overlook of the levels upper to X1 to represent the generation of paths, channels and domains

ITU-T Y.3011 [1] and our RM in this section. The virtualization model in Y.3011 is divided into three layers: physical resources, virtual resources and virtual networks. Physical resources defined in Y.3011 can be mapped to the levels below Level 6 as described above. In this mapping, 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 of the physical network. In the level of virtual resources, Y.3011 states that "physical resources are partitioned and abstracted as virtual resources", which can also be mapped to role partition and role collaboration in RM. In fact, the physical network roles are divided into several groups, and the resource of the divided physical network roles, the basic roles, are put together to form an abstraction view, which equals a new generated virtual resource role. RM also works in the level of virtual networks, in which a new role of LINP (Logical Isolated Network Partition) is generated from the virtual resource roles in the same manner.

D. Consideration of Event Management in Role Model

In the network virtualization environment, fault management becomes extremely difficult due to the multiple layers of the 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 known to be a complicated task. Moreover, since virtual networks are not static, the problem becomes even more complicated, given that we have to dynamically modify the event distribution among virtual networks following with their dynamic generation and deletion. Luckily, while acting as a resource management model for network virtualization, RM also helps to express event distribution among different virtual networks. Section 4 describes how we utilize this feature to generate event management functions.

E. Discussion

The examples above show that paths, switches and even network domains can be represented by the proposed RM in a unified way. Moreover, due to the simplicity of the model, it is also possible to map it to existing virtualization models such as ITU-T Y.3011. In addition, we can also come to the following conclusions about the proposed model.

Conclusion I: All roles are regarded to be generated from roles in the lower level and this cycle can be traced back to the roles initially assigned to hardware.

Conclusion II: All forms of network virtualization or abstraction can be represented using roles.

Conclusion III: As long as the relationship 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.

IV. Design of Management Engine

A. Inspiration from Role Model

As one of the conclusions in the last section, all roles are regarded to be generated from roles in the lower level and this cycle can be traced back to the primitive roles in the lowest level. Therefore, we can accept that the relationship among roles in different levels can be clearly defined, and moreover, using the defined inter-role relationship, *(a) some of the source code for managing roles (e.g. generation/modification/deletion) can be automatically generated*. Moreover, *(b) a platform that realizes actual inter-role interaction (e.g. resource allocation/event handling) is necessary*.

Based on the above assessment, we design our ME which mainly consists of a tool (the role generator) that can automatically generate the referred source code and a platform (the role integration unit) for actually executing the generated source code. It is expected that by using roles to represent different forms of virtual networks, their corresponding management systems can be developed in a short time with the assistance of ME.

B. Design of Management Engine

Fig. 5 shows the structure of ME. The following explains all components one by one.

Hardware Interaction Unit

The hardware interaction unit represents physical entities as primitive roles internally and manages the relationship between the physical entities and the primitive roles. In addition, it also provides the role integration unit with interfaces to these primitive roles. The primitive roles are the roles that were initially assigned to hardware and are designed in order to absorb the differences among various kinds of network equipment. Although the definition of primitive roles can differ system by system, the design of primitive roles is important, as it has a huge impact on system implementation. For instance, expressing primitive roles in extremely fine granularity can lead to great complexity in upper roles, and vice versa.

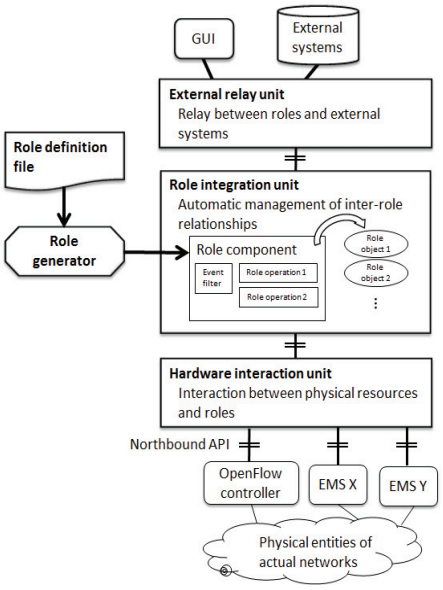


Fig. 5. Structure of Management Engine

TABLE I. BASIC ELEMENT OF A ROLE DEFINITION FILE

Basic Element	Explanation	Example
Role name	Identification of individual roles	Path
Attributes	Information to describe a role	♦Path constitution (e.x. 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 intented to be received	♦MESG_NODE_FAIL, ♦MESG_LINK_FAIL

Role Definition File

In ME, we generate the management system automatically and dynamically by writing simple and structured definition files. Since ME is designed based on RM, all the management objects should be roles, so a definition file contains only actual role definitions, which cover network resources related to a role, the inter-role relationship, the event distribution among roles and so on. The basic elements of a role definition file are illustrated in TABLE I.

Role Integration Unit

First, three terminologies (role component, role object and role module) are clarified. Role components are the programs generated by the role generator from a role definition file. One role component exists so as to manage a set of role objects of one type. In fact, a role component acts as a factory program that realizes the generation and deletion of role objects. The control methods of roles including event handlers other than the generation and deletion operations are also parts of a role component. In the OSGi environment, role components are implemented as bundles [6]. Role objects represent role instances. In fact, they are structured objects that hold the attribute data of a role. Compared with the role components, which provide with the control methods of roles, only the simple setter or getter methods to access the attributes of role objects are considered. 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. Take path management as a simple example. The management system that realizes operations such as path creation and deletion is regarded as a role component; different paths are generated and deleted as different role objects. Finally, algorithms, such as path computation, are reusable, and are regarded as role modules in this case.

The role integration unit instantiates the role components generated by the role generator and it also automatically maintains the relationship among individual instances. By maintaining information consistency among role components, the role integration unit makes possible dynamic addition and deletion of role components and even the implementation of event distribution among role components. As is seen below, the role integration unit can be implemented using OSGi

framework [6].

Role Generator

The role generator creates role components that are defined in a role definition file. When the role integration unit is based on OSGi, the role generator will generate the source code of role component correspondent bundles and download them into the OSGi platform [6].

When event distribution among roles is considered as in Section 3, 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 RM so that only the events destined to the role component itself will be received. Moreover, the event handler skeleton is generated so that an event and the correspondent event handler to process the event are combined.

External Relay Unit

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

C. Features of Management Engine

The features of ME can be summarized briefly as follows:

First, based on RM, ME supports the dynamic customization of modularized management systems for virtual networks to follow the generation and deletion of virtual networks. Next, the above function can be realized easily by writing simple and structured definition files. In addition, ME can support different infrastructure technologies.

V. Prototype and Validation

In order to validate the feasibility of ME, we developed a prototype 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 [7]. In the prototype infrastructure, voice services are enabled by dynamically allocating both network resources and server resources, unlike the current mobile infrastructure which adopts the fixed allocation of maximum resources. The prototype infrastructure adopts both server virtualization and OpenFlow technology to realize sufficient flexibility.

Fig. 6 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 Interrogating-Call Session Control Function (I-CSCF), Proxy (P)-CSCF, and Serving (S)-CSCF in IMS. In the prototype, P-CSCFs and I-CSCFs, although not illustrated in Fig. 6, are setup separately as different virtual machines (VMs) and scattered between the two data centers. Furthermore, S-CSCF is actually realized as two processes (p1 and p2), corresponding to two VMs, in order to implement

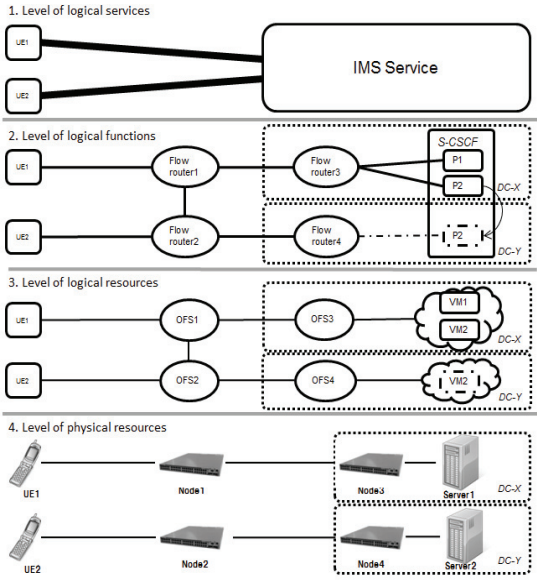


Fig. 6. Configuration of prototype infrastructure

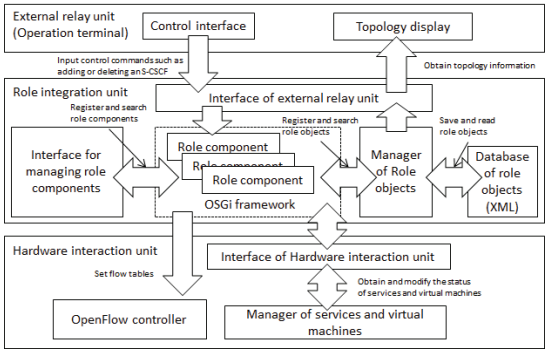
load balancing for the coming call requests; initially they both located in DC-X. Although the wireless access points are omitted from the figure, they actually exist to connect mobile terminals (e.x. UE1 and UE2). OpenFlow switch (OFS) 1 and OFS2 provide application-aware virtual paths between the mobile terminals and the data centers, while OFS3 and OFS4 work inside the data centers in order to balance traffic.

B. Design of prototype Management Engine

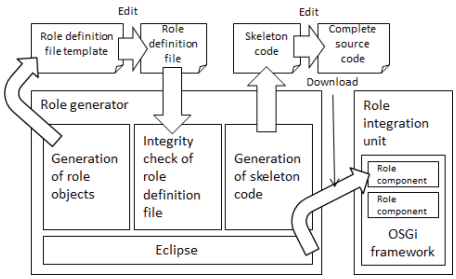
We developed a prototype system to realize ME; design details are shown in Fig. 7. Fig. 7 (a) shows a block diagram of the hardware interaction unit, role integration unit, and external relay unit. In the hardware interaction unit, an OpenFlow controller is used to control the OpenFlow switches, while the functions to obtain and control the statuses of VMs and services were newly developed with REST interfaces facing the role components in the role integration unit. The role integration unit is implemented based on the OSGi framework [6], so all role components, role modules, are implemented as bundles, and we use Event Admin Services (EAS) to implement event distribution [6]. Also we developed several tools to facilitate the management of role components and role objects. The operation terminal was developed as the external relay unit, to provide the GUI for displaying topology and the input of control commands such as those to reinforce and weaken the process ability of S-CSCF. Fig. 7 (b) shows a block diagram of the role generator. The role generator is mainly based on Eclipse; Eclipse standard functions such as XML editor can be reused and the ME's special functions such as generating role projects, checking role definition files, and generating source code are implemented as plugins.

C. Validation

We developed a management system for the prototype infrastructure by using the prototype ME. As illustrated in Fig. 6, actually 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



(a) Block diagram of the hardware interaction unit, the role integration unit and external relay unit



(b) Block diagram of the role generator

Fig. 7. Design of prototype Management Engine

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 role definition files.

The validation confirmed the functioning of the management system for the prototype infrastructure. If an S-CSCF process (p2) failed, the remaining process (p1) 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 ME responds by setting up a new process (p2) in the other data center (DC-Y) which achieves load balancing. At the same time, the management system also modifies the flow tables in the related OpenFlow switches automatically in order to re-route traffic.

D. Discussion

While the validation confirms that ME shows promise, it is certainly too early for us to claim that ME is able to cut development costs in all cases. Some challenges must be overcome. First, there is the question of even how to evaluate ME. ME takes advantage of definition files in order to reduce the volume of source code that must be edited directly when developing a management system. However, the volume of source code that can be cut strongly depends on the system used and the number of role modules that can be reused. Although the prototype demonstrated a very promising level of source code reduction, more experiments are necessary. Second, network nodes that can be programmed by software on the fly are a hot topic [2, 9, 10]. In fact, the customization

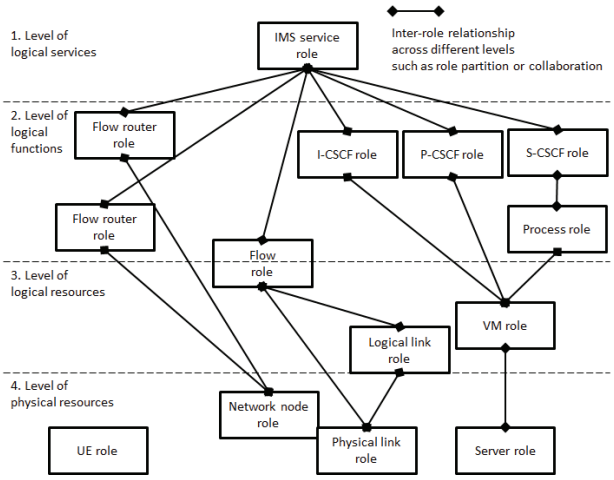


Fig. 8. Role architecture for the prototype infrastructure

process of a network node can be mapped from Level 1 to Level 4 in our RM. However, the prototype ME doesn't cover these levels, which is left as future work. Third, although we listed some basic elements of role definition files in this paper, their construction needs further research. File design (structure and element) will impact the functionality and effectiveness of ME. For example, we are currently investigating if algorithms can be generated automatically from the definition file.

VI. CONCLUSION

This paper introduced RM (to meet Requirement II) which can be used as a simple and unified representation approach that supports various forms of network virtualization. Based on the model, we proposed ME (to meet Requirement I) and validated its feasibility by testing a prototype system. Although some questions still remain as discussed above, we believe that ME is a promising candidate management platform for virtual networks.

REFERENCE

[1] Draft ITU-T Recommendation Y.3011, "Framework of network virtualization for Future Networks," Oct., 2011.

[2] A. Nakao, "Network virtualization as foundation for enabling new network architectures and applications," IEICE Trans. Comun., vol. E93-B, no. 3, pp. 454-457, March 2010.

[3] S. Davy, C. Fahy, Z. Boudjemil, L. Griffin, and J. Strassner, "A model based approach to autonomic management of virtual networks," in Proc. IEEE IM 2009, pp. 761-774, New York, USA, June 2009.

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

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," in ACM SIGCOMM Computer Communication Review, vol. 38 (2), pp. 69-74, April 2008.

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

[7] Y. Nakajima, H. Masutani, W. Shen, 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, to be presented.

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

[9] NFV White Paper, "Network functions virtualisation," Oct. 2012.