vConductor: An Enabler for Achieving Virtual Network Integration as a Service

Wenyu Shen, Masahiro Yoshida, Kenji Minato, and Wataru Imajuku

Wenyu Shen is with NTT Communications.

Masahiro Yoshida, Kenji Minato, and Wataru Imajuku are with NTT Network Innovation Laboratories.

Wenyu Shen was with NTT Network Innovation Laboratories when the paper was submitted.

ABSTRACT

Network function virtualization is regarded as a promising candidate for future networks. Although many advantages are expected, there is still a lack of services that directly link to increased revenue for telecom operators. Therefore, we propose a new NFV-based service, virtual network integration as a service (VNIaaS). The service allows enterprise network administrators to construct and monitor virtual enterprise networks as necessary. As a key enabler, we further propose vConductor, an innovative technology due to its automatic network provisioning, multi-objective resource scheduling, and NFV-oriented inventory management characteristics. Finally, we show the feasibility through implementation of a prototype system. We believe that this contribution will be a catalyst to accelerate the industrialization of NFV.

INTRODUCTION

Network function virtualization (NFV) has recently attracted much attention worldwide [1]. Academia and industry have come to regard it as a promising candidate for the so-called future networks, along with software defined networking [1, 2]. Originally, by eliminating dedicated physical devices and fully utilizing the cloud infrastructure, there is the possibility of significant cost reduction. However, recently a more promising prediction has come to light in that software technologies increase network flexibility and enable rapid delivery of new functions. This yields insight into the possibility of creating a new service. After all, telecom operators are more interested in increasing revenues nowadays.

With the focus of creating a new source of income, we propose a new NFV-based service, virtual network integration as a service (VNIaaS), which is positioned to serve enterprise users. VNIaaS provides a portal site in which a network administrator constructs a virtual enterprise network on the fly by utilizing a virtualized network function (VNF) store and a graphical user interface (GUI)-based virtual network editor. In fact, VNIaaS provides one-stop integration of cloud and networking services so that all related provisioning tasks, including provisioning of network functions, application servers, and logical connections, are handled automatically. We also provide monitoring as an optional service.

However, completely software-based network construction causes unexpected complexity. In addition, a variety of business needs have to be handled for individual enterprise users when performing resource scheduling. Fault management becomes extremely difficult when the virtualized layer is mixed with the existing physical layer, and this situation becomes further complex when spanning multiple domains.

To address these problems, we propose vConductor, a key enabler for achieving VNIaaS. First, vConductor achieves full automation of virtual network provisioning by simplifying and normalizing the provisioning procedure. Second, a multi-objective resource scheduling algorithm is adopted so that individual business needs can be precisely met with limited investment. To facilitate fault management, especially fault isolation, we further enhance the existing inventory management. This is achieved in the design of an NFV-oriented data model based on the TM Forum Information Framework (SID) [3]. Third, vConductor is designed to be generic. It is positioned as a common solution to NFV management and orchestration (MANO) according to the end-to-end architecture defined in the NFV group specification. We maximize the system modularity so that vConductor can easily be applied to other use cases such as the virtualized evolved packet core. To achieve this, we employ a plug-in architecture to enhance the interoperability with various VNF managers and virtualized infrastructure managers (VIMs). This article summarizes the business goals, technical challenges, and essential design concepts on the whole toward realizing VNIaaS, while we leave the discussion of individual implementation details to our previous works [4, 5].

VNIAAS: A NEW REVENUE SOURCE FOR TELECOM OPERATORS

Figure 1 shows a service image of VNIaaS. We aim to provide a portal site that allows a network administrator to construct a virtual enterprise network as necessary.

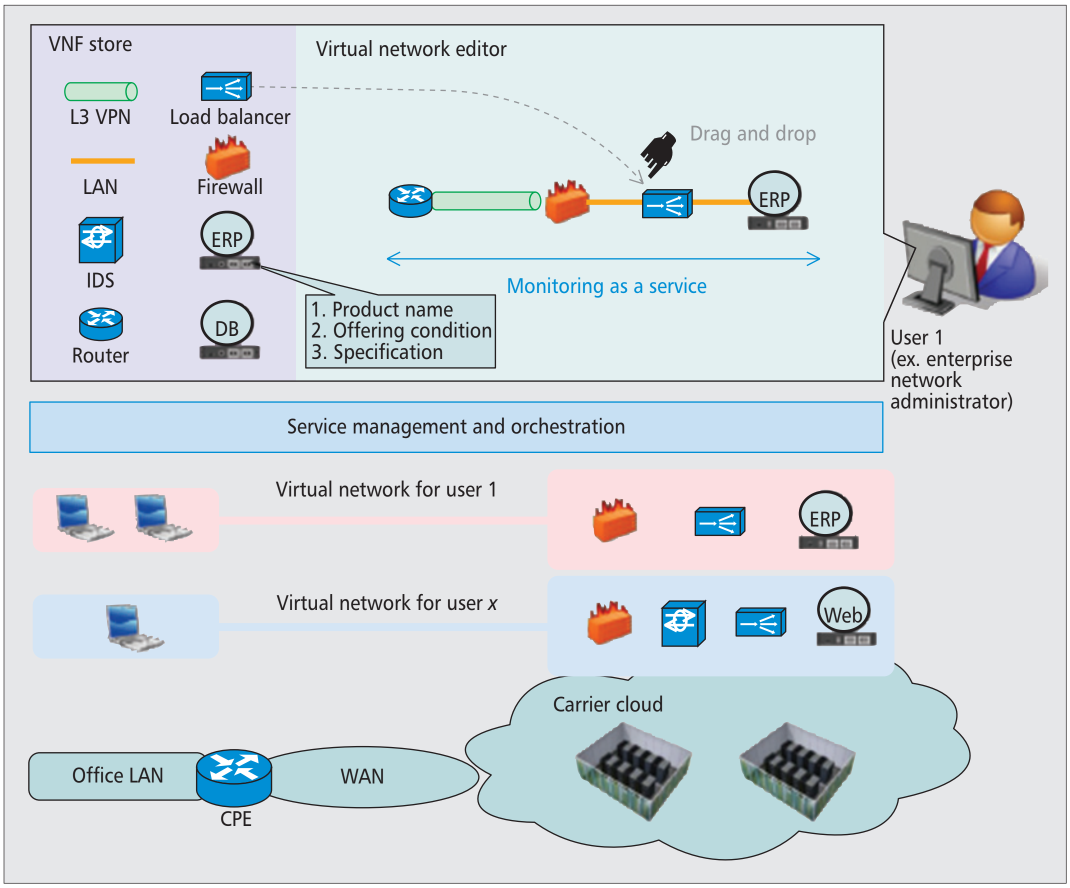

Figure 1. VNIaaS concepts.

The concepts of VNF store, virtual network editor, end-to-end connection management, and monitoring as a service differentiate VNIaaS from many of the existing cloud services.

VNIaaS Concepts

We discuss four main VNIaaS concepts below. These concepts differentiate it from many of the existing cloud services that provide users with only isolated virtualized resources instead of end-to-end solutions.

VNF store: Similar to many popular application stores nowadays, VNIaaS includes a VNF store that provides a variety of choices when users construct a virtual network. Figure 1 shows some VNFs such as an intrusion detection system (IDS), load balancer, and even application templates, such as an enterprise resource planning (ERP) system and a database (DB). To assist further in the procurement of VNFs and network design, detailed catalog information is available, which covers product names, offering conditions, and specifications.

Virtual network editor: The virtual network editor provides a GUI for users to design a virtual network comprising VNFs purchased at the VNF store. The only operations required of users are to drag and drop icons representing the VNFs, and to connect them using virtualized links (VLs). Our MANO solution enables full automation of all actual provisioning tasks.

End-to-end connection management: When an enterprise network is located in a telecom cloud, connections between the customer premises equipment and remote data centers (DCs) must be considered. This falls into the traditional area of expertise of a telecom operator. Here, the combination with a virtual private network (VPN) service makes VNIaaS a true end-to-end solution.

Monitoring as a service: We believe that VNIaaS has the potential to accelerate the transformation of communications, in which enterprise networks will be gradually moved to the telecom cloud. VNIaaS enables telecom operators to participate in the operation of enterprise networks. As an option, VNIaaS further provides a monitoring service from the viewpoint of enterprise users.

Related Work

The concept of a network as a service (NaaS), regarded as a starting point of network virtualization, has been studied for several years. We see that a variety of VPN services have become a main revenue source for many telecom operators. An overlay network is yet another form of NaaS, which is typically implemented in the application layer, although various implementations at lower layers of the network stack do exist [6]. A more challenging concept has been proposed in academia, advocating dynamic instantiation of layers [7]. NFV diversifies network virtualization; in the NFV context, VNFs can be provided on demand. Therefore, the new concept of function as a service (FaaS) appeared [8].

The proposed VNIaaS provides VNFs on demand in addition to the tunnels that are the only focus in the traditional NaaS. Besides, VNIaaS integrates functions of both networks and cloud applications to create an end-to-end ser-

To realize VNIaaS, we propose vConductor. The initial result of our proof of concept has shown the feasibility of achieving full automation of network provisioning, multi-objective resource scheduling, and end-to-end inventory management for virtualized environments.

vice that spans multiple domains, while this falls outside the scope of FaaS, which considers only one function or functions within a single domain. We clarify that VNIaaS is more than a simple combination of the two concepts; as discussed below, VNIaaS has great potential to change the business model in the current ecosystem of communications industry, while new technical challenges occur in the way of actualizing the concept.

Benefits for Enterprise Users and Telecom Operators

We believe that VNIaaS will benefit enterprise users and telecom operators. For enterprises, it is no longer necessary for network administrators to purchase expensive hardware, because software is generally less expensive and even free in many cases. Actually, our VNF store offers a platform where network administrators can select the most suitable software from the perspective of price and functionality. When constructing a network, users benefit from a one-stop service, integrating wide area networks (WANs) and local area networks (LANs). After network construction, the functions, topology and even architecture can be modified easily according to dynamic requirements. In VNIaaS, we achieve these using a graphical editor. Finally, the optional monitoring service potentially releases the network administrator from routine tasks and contributes to reducing operating expenses.

As a telecom operator, we anticipate a revenue increase. Initially, we expect to create new sources of income by selling VNFs. There is also potential for product bundling (WAN and cloud service) by providing end-to-end connections. Hence, we foresee an opportunity for a telecom operator to transform from an infrastructure provider to a solution provider.

Technical Challenges for Realizing VNIaaS

In VNIaaS, hardware devices are substituted with VNFs that are implemented as software appliances running on virtual machines (VMs) while the cables are substituted with VLs that are implemented as logical tunnels. As a result, the network construction process becomes execution operations for a variety of scripts, which unexpectedly result in a complex and error-prone task. Therefore, one challenge is to achieve automatic network provisioning.

Due to the essence of virtualization, extra considerations must be taken on issues such as the amount of resources that must be reserved to guarantee VNF performance, and selecting a physical container, e.g., a DC, for deploying a VNF. In the enterprise business domain, it is critical to offer multiple service options at competitive prices to meet diverse needs. Moreover, those needs often conflict with each other. Therefore, a resource schedule oriented to multiple objectives is desirable.

Next, a network can now incorporate virtualized entities, which raises the issue that managed objects that can be dynamically modified [6]. Furthermore, the services tend to be end-to-end in most cases and are assumed to span multiple domains, e.g., DCs. These changes actually complicate the inventory management. In order to convince potential customers of the reliability of VNIaaS, it is necessary to provide a complete end-to-end network view that includes all the information regarding the physical layer, virtualized layer, and mapping between the layers. A shortfall in any of the information can lead to failed fault management resulting in a breach of the service level agreement (SLA). Therefore, achieving end-to-end inventory management for a virtualized environment becomes a must.

vConductor: Enabler Technology

To realize VNIaaS, we propose vConductor. The initial result of our proof of concept, described later, has shown the feasibility of achieving full automation of network provisioning, multi-objective resource scheduling, and end-to-end inventory management for virtualized environments. These are enabled through our work on the simplification and normalization of provisioning procedures based on a standard system structure, the design of a novel resource scheduling algorithm, and NFV-oriented data modeling.

Standard System Structure and Provisioning Procedures

The system structure, shown in Fig. 2, fully conforms to the NFV end-to-end architecture, where it has an orchestrator as the core including the virtual network life cycle manager (VNLM), resource designer, service level controller, and databases, and also embeds the VNF managers. In addition, the user portal and element managers correspond to the operation support system (OSS) and element management system, respectively. vConductor relies on existing VIMs such as OpenStack (www.openstack.org) for managing DCs and OpenDayLight (www.opendaylight.org) for managing WANs. They are assumed to be external systems.

VNLM: The VNLM controls operations of the internal components and external systems based on the received service orders (SOs) regarding network generation, modification, and deletion; they are implemented as a series of instance descriptors [9]. A SO is first input into an analyzer, where its format and feasibility are validated, after which the analyzer calculates the necessary computing and network resources by referring to the information stored in the catalog DB. It is the resource reservation component that actually performs reservations.

Resource Designer: The resource designer schedules resources and generates parameters. The former calculates how VNFs are allocated to the underlying infrastructure (e.g., DCs), while the latter designs the necessary logical resources such as IP addresses for practical resource reservation. The resource designer interacts with the resource DB to retrieve the necessary information regarding physical and logical resources. A detailed description of resource scheduling is presented in the next part.

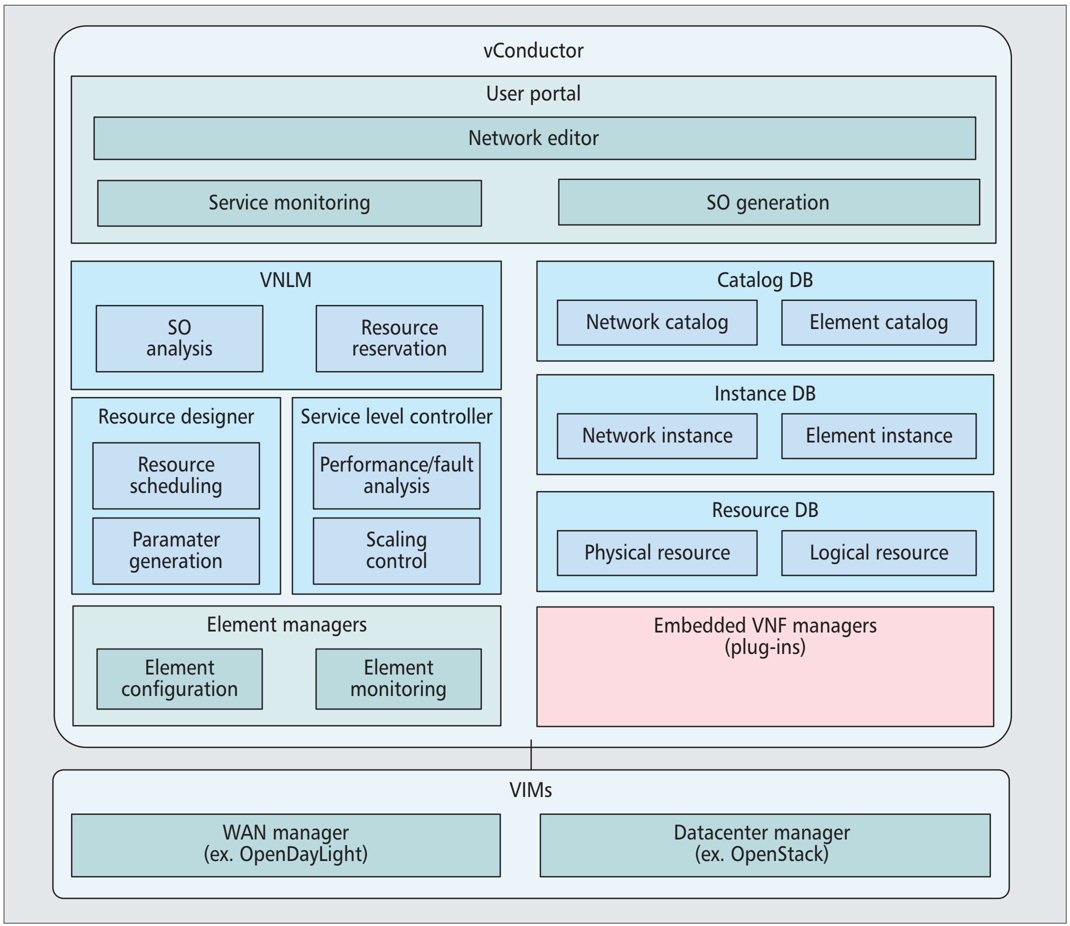

Figure 2. System structure of vConductor.

The user portal provides an editor for end users to customize their virtual networks. We designed a GUI to minimize the user's effort so that the user is only required to drag and drop some icons that represent VNFs and connect them together with VLs.

Service level controller: This component is designed to guarantee the SLA by handling hardware faults and performance degradation. To achieve this, the performance/fault analysis component analyzes the status of the VNFs and their supporting resources retrieved from the VNF managers and VIMs, based on which the root cause is located. For the recovery process, we duplicate the affected VM and reallocate it. Here, the scaling control component determines the most proper performance timing.

Catalog, instance, and resource DBs: These DBs perform information management for the whole system. The catalog DB stores network and element catalogs in the form of descriptors [9]. The element catalog stores resource-related information for implementing, for example, VNFs and VLs, while frequently used connection patterns are summarized as templates, known as network catalogs. Next, the instance DB manages the configuration information of the elements and networks that are set according to the catalogs. Let us consider VNFs as an example. Parameters such as assigned IP addresses, performance indicators, and log data are covered. To fulfill the inventory management role, the instance DB also manages a large number of correlations among the network layers and multiple domains. This is discussed in the data modeling section. Finally, the resource DB stores information regarding the physical resources (e.g., the number of CPU cores) and logical resources (e.g., IP addresses), and interacts with the VIMs to perform updates on a regular basis.

VNF manager: The VNF manager controls the life cycles of its VNFs, covering (un)installation of their hosting VMs, which are defined as virtual deployment units (VDUs) in European Telecommunications Standards Institute (ETSI) terminology, and their initialization and termination [9]. It also calls the element manager for the initial configuration. In order to perform the actions above, a manifest file is needed for each VNF package to record the operational sequence.

User portal: The user portal provides an editor for end users to customize their virtual networks. We designed a GUI to minimize the user's effort so that the user is only required to drag and drop some icons that represent VNFs and connect them together with VLs. The user input is translated into a SO before it can communicate with the VNLM. Furthermore, a user-level monitoring service can be provided, which helps the user understand the real-time condition of the services. The element managers are in charge of the real configuration and monitoring of individual elements. Currently, we are using some vendor-specific tools that are designed for the corresponding VNFs. However, we believe that more work should be performed such as interface specification and information abstraction in order to facilitate management.

We assume that a variety of business needs, also known as objective functions, coexist. Each of these factors influences the VNF allocation decision. Moreover, these factors tend to conflict with each other in many cases. A simple example is an enterprise user with a limited investment expects a minimum end-to-end latency and maximum throughput.

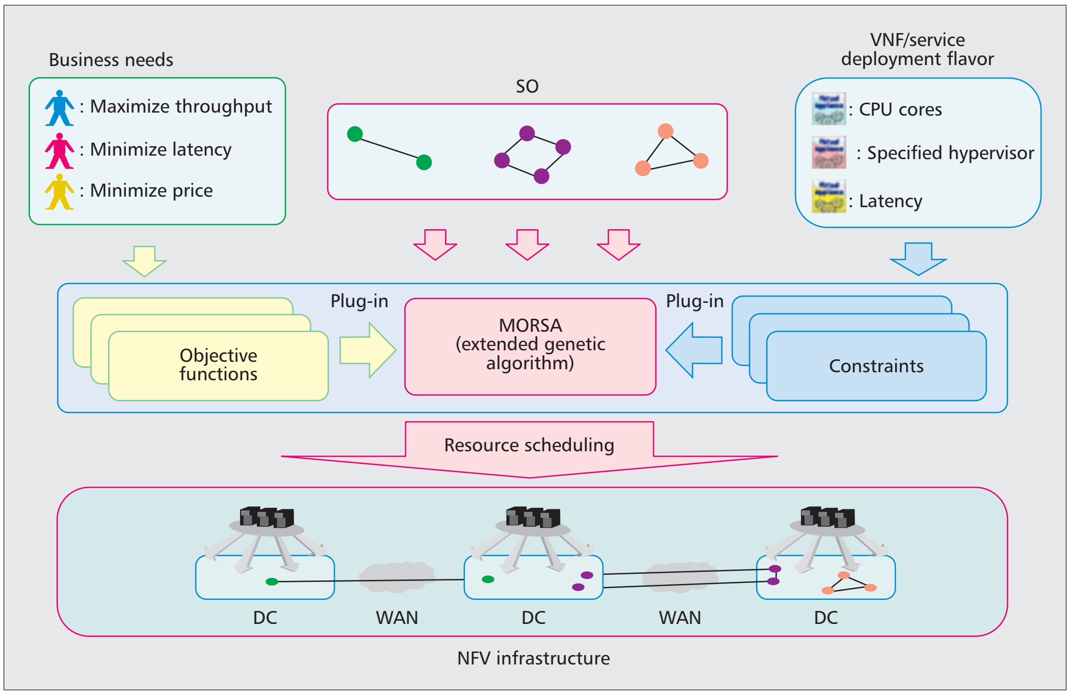

Figure 3. MORSA concept.

We further present a refined virtual network provisioning procedure, which facilitates the understanding of the function of individual components and enables automation. We also introduce a VIM driver (not illustrated in Fig. 2) in order to absorb the diversity of VIMs so that the provisioning procedure can be simplified and normalized, focusing only on the general purpose. In the following, we describe the normalized process for network provisioning that spans multiple DCs.

Step 1: The VNLM receives SOs from the user portal.

Step 2: The VNLM retrieves the necessary catalog-related information from the catalog DB.

Step 3: Based on the SOs and catalog-related information, the VNLM further calculates the resources required for running the requested service and sends the information to the resource designer.

Step 4: The resource designer polls the resource DB for the available physical resources and decides the deployment destination for the VNFs.

Step 5: The resource designer polls the resource DB for the available logical resources and designs configuration parameters for implementing VNFs and VLs.

Step 6: Based on the design results, the VNLM reserves resources via the VIM driver.

Step 7: The VIM driver establishes underlying inter-DC tunnels, interacting with the WAN manager.

Step 8: The VIM driver reserves computer resources and establishes the underlying inner-DC tunnels, interacting with the DC manager.

Step 9: The VNF manager installs and initializes the component VDUs and performs the initial VNF configuration.

Step 10: The instances of the deployed network are registered in the instance DB.

Multi-Objective Resource Scheduling

The allocation of VNFs to the underlying infrastructure, which for VNIaaS is a cloud environment spanning multiple DCs and WAN connections, is regarded as complex. First, when deciding the NFV allocation, we must consider many constraints on the resources that remain in the infrastructure in order to prevent overprovisioning, while great care should also be taken to guarantee the agreed performance of individual VNFs and end-to-end services. This refers to the constraints on the deployment flavor. Furthermore, we assume that a variety of business needs, also known as objective functions, coexist. Each of these factors influences the VNF allocation decision. Moreover, these factors tend to conflict with each other in many cases. A simple example is an enterprise user with a limited investment expecting a minimum end-to-end latency and maximum throughput. Clearly, the optimal solution that meets the needs for all the requirements is not practical. Hence, it is sometimes a must to search all the possible solutions in order to make the most reasonable decision.

There are several existing algorithms for solving the virtual network embedding problem [10, 11], but we think that none of them can meet the specific needs in the context of VNIaaS. Therefore, we propose our multi-objective resource scheduling algorithm (MORSA) for vConductor, which simultaneously considers multiple objective functions. Figure 3 shows the concept; details are given in our previous work [5]. MORSA originates from the traditional genetic algorithm (GA) [12], but we extend it to the specific needs of vConductor. In the MORSA concept, we first convert the constraints into the form of objective functions. A charac-

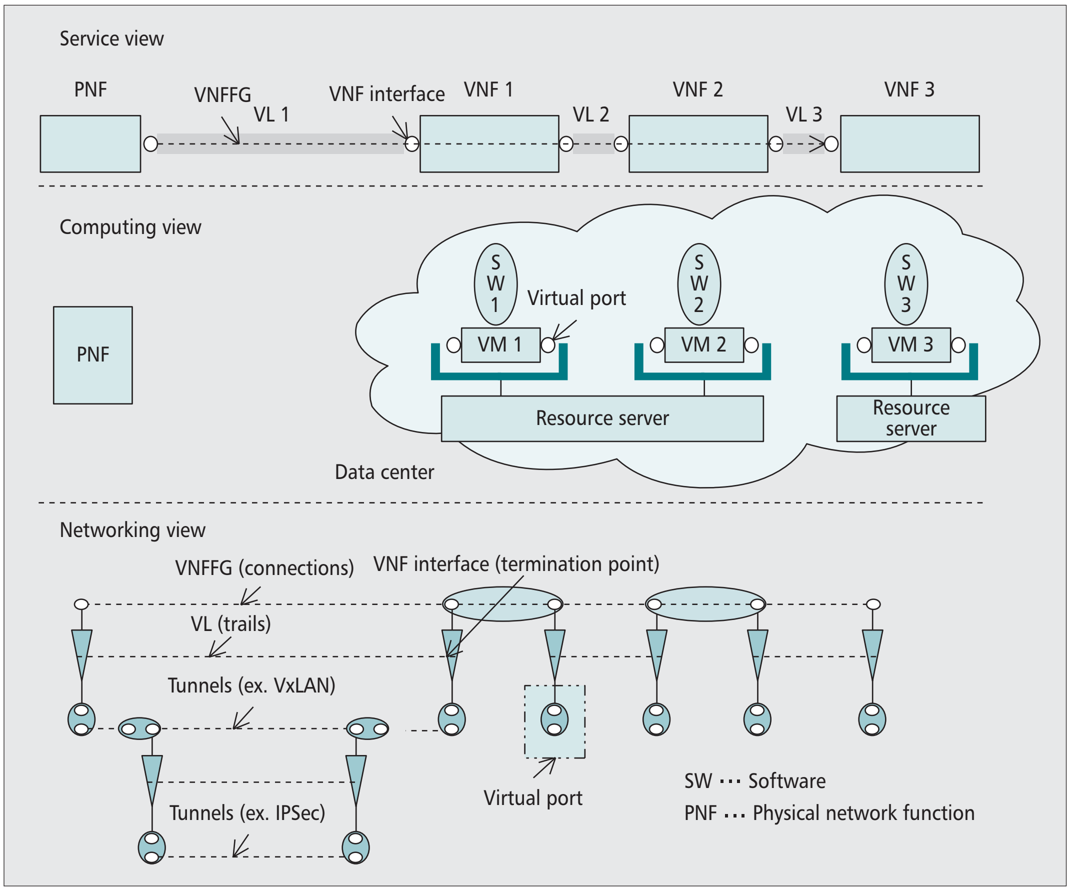

Figure 4. Function modeling for an example virtual network.

To manage the correlations behind the virtualized environment, we further propose a powerful data model. The proposed model completely conforms to the SID so that it supports newly proposed NFV concepts.

teristic of the GA is that the computation time will not drastically increase with the addition of objective functions. By utilizing this characteristic, resource scheduling in vConductor will not take a long time even if there are many constraints. Furthermore, MORSA is totally plug-in-based. Objective functions can be plugged in and out according to dynamic demands. Another advantage is that MORSA further extends the GA to perform optimization while adjusting the weights of each objective function. In this way, more solutions can be obtained instead of limited localized solutions in the case of the original GA.

The final service selection is left to the user based on a service vs. pay scale. That is to say, the user can select the final solution by arbitrarily and freely adjusting the weights among the objective functions to arrive at a desirable price point.

Data Modeling

We discuss the challenge in fulfilling inventory management for virtual networks. We advocate that the essence is to handle a variety of correlations. Thus, our discussion starts with function modeling of virtual networks. Figure 4 models an example virtual network from three different views: service, computing, and networking. The service view takes into consideration the modeling of service function chaining (SFC), as discussed in the Internet Engineering Task Force (IETF) (e.g., the attachment of a VNF interface to a VNF and its connection to a VL) and routing (e.g., a VNF forwarding graph, VNFFG) [9], but our focus is only on the management plane. From the computing view, we need to further handle, for example, how a VNF is implemented on a virtual machine (VM) and how a VM and its virtual port are allocated on a resource server. The networking view is constructed mainly based on the modeling techniques in ITU-T Recommendation G.805 [13]. By utilizing the concept of connections and trails, we envision a layering correlation of logical tunnels for implementing the VLs and VNFFGs.

To manage these correlations, we further propose a powerful data model, the core of information management in vConductor. The proposed model completely conforms to the SID model [3], so it is regarded to be general and have high affinity to many existing OSSs. We further extend SID so that it supports newly proposed NFV concepts such as VNFs and VLs. Therefore, most of the NFV-specific correlations illustrated in Fig. 4 can be handled by our proposal, the details of which are shown in Fig. 5.

First, we discuss service-related modeling. The top left of Fig. 5 shows that a virtual network service (NetworkService) is created as a subclass of ResourceFacingService. The associations such as NSRequiresVNFFGs, NSRequiresVLs, NSRequiresVNFs, and NSRequiresPNFs represent the correlations between a service and its components. We further combine a VL with a VNF by introducing

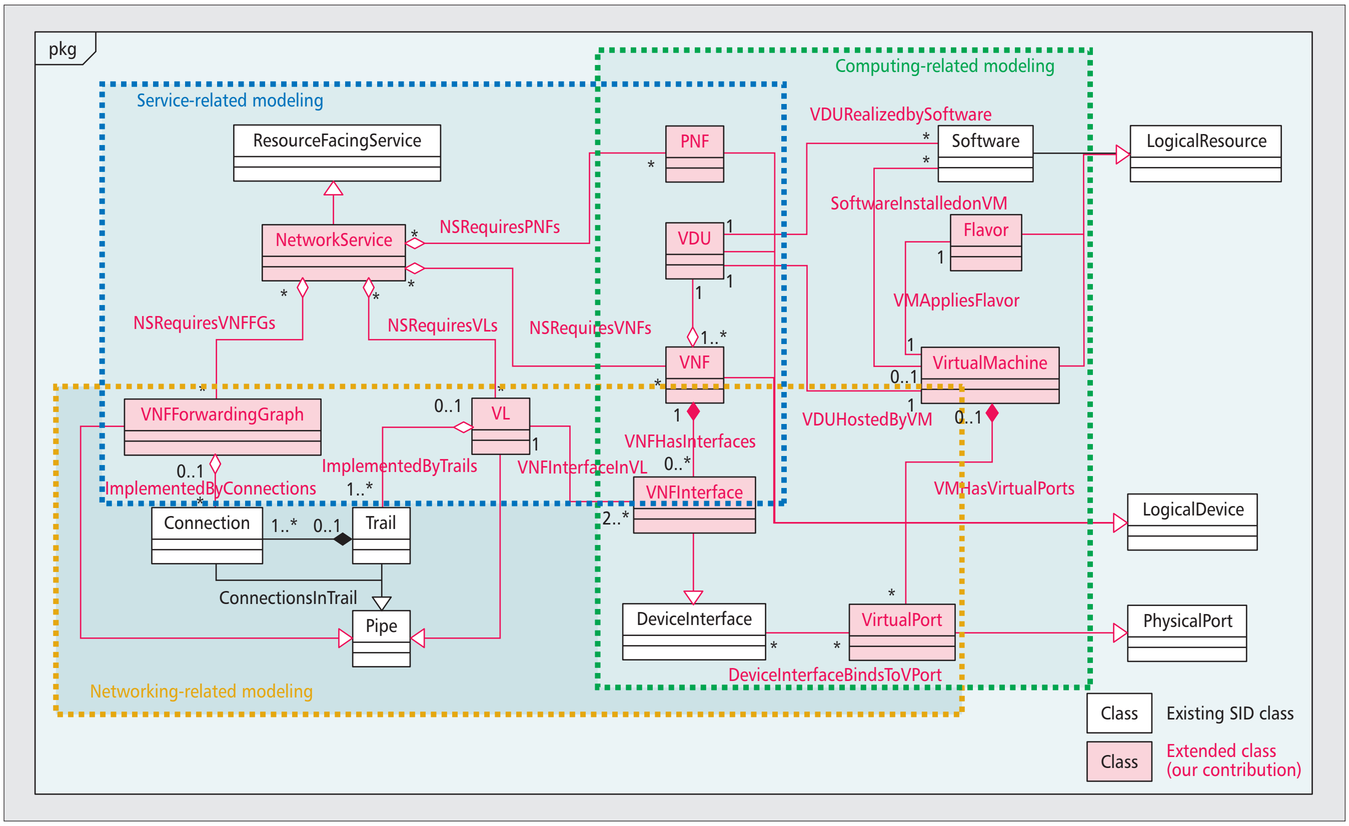

Figure 5. Data modeling in vConductor.

the VNFInterface class and two associations (VNFHasInterfaces and VNFInterfaceInVL). Inherited from DeviceInterface, VNFInterface represents a virtualized interface of a VNF.

Computing-related modeling is presented on the right of the figure, where VNF, PNF, and VDU are designed as subclasses of LogicalDevice. The fact that a VNF comprises several VDUs is represented by the association of HasVDUs. Moreover, we design a class called VirtualMachine to represent VMs and the Flavor class to represent its specification so that the concept of server virtualization can be modeled by a series of associations of SoftwareInstalledOnVM, VMAppliesFlavor, VDURealizedBySoftware, and VDUHostedByVM. These associations enable the management of correlations among a VDU, its implementing software, and the container VM and its flavor.

Finally, we discuss the networking-related modeling. The VL and VNFForwardingGraph classes are introduced and designed as subclasses of Pipe (bottom left of Fig. 5). The ImplementedByTrails association represents the fact that a VL is implemented by one or more trails. The existence of multiple trails actually means a kind of 1:*N* protection mechanism. Next, the ImplementedByConnections association represents the concept of a VNFFG, which is implemented by connecting a series of link connections and subnetwork connections. We use the VirtualPort class to represent virtual network interface cards (vNICs) while using the VPortBindsToDeviceInterface association and VMHasVirtualPorts to link a VNF interface with the vNIC and then to a VM.

Proof of Concept

We expect the emergence of new NFV-based services enabled by the vConductor proposal. vConductor enables the provisioning for constructing a virtual network to be performed automatically. In addition, user-level monitoring becomes available, so it is possible for a telecom operator to provide one-stop management services.

We implemented a prototype to show the feasibility and are planning a demonstration aimed at partially proving the concept in a future event. Figure 6 shows the detailed implementation of the prototype structure, covering the data plane and control and management plane. One physical customer premises equipment (CPE) in premises #C and two scattered DCs (DC #A, DC #B) are assumed. As DC managers, we reuse OpenStack, which comprises one cloud controller, one network node, and multiple computer nodes. Then we use simulated WAN connections to link the CPE to the DCs. Since we assume that the simulated WAN configuration is fixed, we use OpenFlow switches at the edges to control the switching of flows for each service. These switches are controlled by an overlay tunnel manager. For simplicity, we use Ansible (www.ansible.com) to control the VNF life cycle and implement a tool for configuring the VNFs selected for our example. We connect vConductor to the Ansible system, the VNF manager, the overlay tunnel manager, and the OpenStack system to yield a running system. System integration is not an easy job due to the existence of different interfaces; as a workaround, we create some wrappers to absorb the diversity. However,

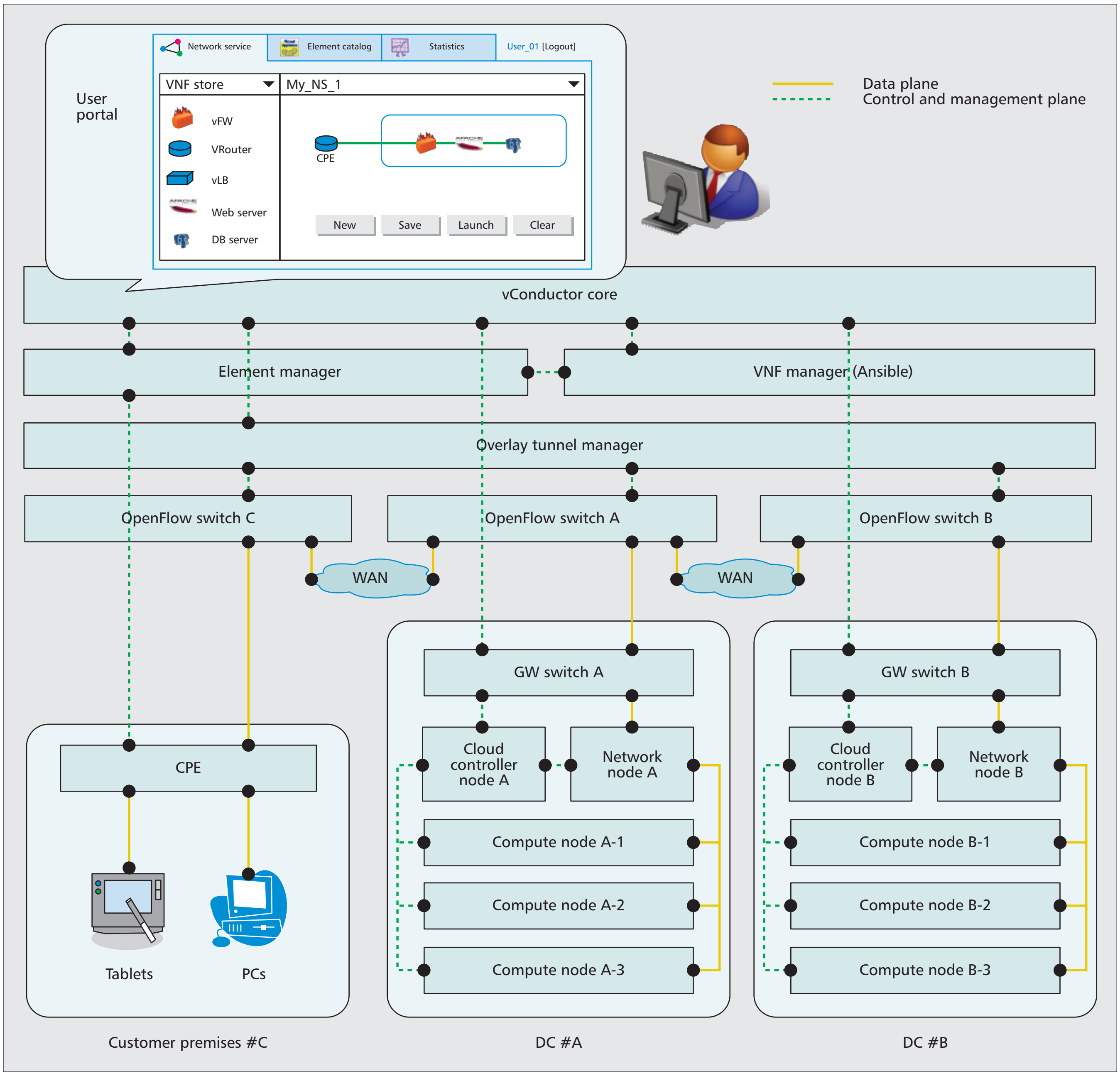

Figure 6. Implementation of prototype structure.

more effort is planned to be put into the interface unification and standardization.

Figure 6 also illustrates a GUI image where all virtualized elements are presented as icons. More details, such as the catalog information, can be viewed by double clicking an icon. The right part presents an editor that allows the user to customize his/her network. The shown service chain is only an example that covers some network and server applications. The topology can be created and changed by simply dragging and dropping the icons. Here, we hide the existence of multiple DCs. Finally, we should note that the prototype further provides extensibility as future extension to an operator interface, where inventory information of a specific service is displayed graphically, although not illustrated. Hence, when any fault occurs, the scope of its influence can be known immediately.

Conclusion and Future Work

This article proposes a new NFV-based service, VNIaaS, by leveraging vConductor. vConductor is a powerful MANO system to support VNIaaS by providing not only automatic network provisioning but also multi-objective resource scheduling and NFV-oriented inventory management capabilities. As a result, VNIaaS allows a network administrator to construct a virtual enterprise network as necessary, while we successfully achieve inter-layer virtual resource mapping to support low-cost operation to actualize rapid fault isolation. VNIaaS is regarded as an attempt to integrate cloud and networking services.

As future work, we plan to investigate the adaptation of vConductor to our commercial services such as the metro Ethernet and cloud

Although the prototype system validates the feasibility, there is still the question of how to evaluate our proposal. We are currently working closely with other telecom operators to define the necessary carrier-oriented service quality indicators in terms of scalability and performance.

services. In addition, we will try to standardize the format of resource catalogs so that new VNFs can be rapidly integrated into the system. Although the prototype system validates the feasibility, there is still the question of how to evaluate our proposal. We are currently working closely with other telecom operators to define the necessary carrier-oriented service quality indicators in terms of scalability and performance. Finally, although CPE is assumed as a physical device in the prototype, we plan to virtualize and move it to our edge cloud. We believe that this contribution will be a catalyst for accelerating the industrialization of NFV.

References

[1] M. Chiosi *et al.*, "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action," *SDN and OpenFlow World Congress*, Darmstadt, Germany, Oct. 2012.

[2] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Commun. Review*, vol. 38, no. 2, Apr. 2008, pp. 69–74.

[3] J. Strassner, *Policy-Based Network Management: Solutions for the Next Generation*, Morgan Kaufmann, 2004.

[4] W. Shen *et al.*, "vConductor: An NFV Management Solution for Realizing End-to-End Virtual Network Services," *Proc. APNOMS '14 (TS2-2)*, Hsinchu, Taiwan, Sept. 2014.

[5] M. Yoshida *et al.*, "MORSA: A Multi-Objective Resource Scheduling Algorithm for NFV Infrastructure," *Proc. APNOMS '14 (TS4-2)*, Hsinchu, Taiwan, Sept. 2014.

[6] N. M. Mosharaf, K. Chowdhury, and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges," *IEEE Commun. Mag.*, vol. 47, no. 7, July 2009, pp. 20–26.

[7] F. Esposito *et al.*, "Dynamic Layer Instantiation as a Service," demo at NSDIv '13, Lombard, IL, Apr. 2013.

[8] G. Monteleone and P. Paglierani, "Session Border Controller Virtualization Towards Service-Defined Networks Based on NFV and SDN," *Proc. IEEE SDN4FNS*, Trento, Italy, Nov. 2013.

[9] J. Quittek *et al.*, "Network Function Virtualization (NFV) Management and Orchestration (v0.8)," GS NFV-MAN 001, ESTI NFV, Nov. 2014.

[10] Y. Zhu and M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.

[11] M. Yu *et al.*, "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration," *ACM SIGGOMM Computer Commun. Rev.*, vol. 38, no. 2, Apr. 2008, pp. 17–29.

[12] M. Mitchel, *An Introduction to Genetic Algorithms*, MIT Press, 1996.

[13] ITU-T Rec. G.805, "Generic Functional Architecture of Transport Networks," Mar. 2000.

Biographies

WENYU SHEN (wenyu.shen@ntt.com) received his M.S. degree in computer science from Tongji University in 2008. In the same year, he joined NTT Network Innovation Laboratories, Yokosuka, Japan, where he worked as a research scientist. He joined NTT Communications in 2014, where he is now in charge of some SDN/NFV-based development projects for commercial services.

MASAHIRO YOSHIDA received his Ph.D. degree from the University of Tokyo in 2013. He received a research fellowship for young scientists from the Japan Society for the Promotion of Science (JSPS) during 2011–2013. He is currently a researcher at NTT Network Innovation Laboratories. His research interests include SDN, NFV, and data center networking.

KENJI MINATO [M] received his B.E. and M.E. in electrical engineering from Ehime University in 1987 and 1989, respectively. He joined NTT Laboratories in 1989. From 2005 to 2012, he conducted research and developed a Home ICT system at NTT Cyber Solution Laboratories. In 2012, he joined NTT Network Innovation Laboratories. He is a member of IEICE.

WATARU IMAJUKU, Ph.D, is a senior research engineer, supervisor in NTT Network Innovation Labs. He contributed to the innovation of NTT's optical network as well as the development of a number of new technologies which include over 65 patents, 85 international publications, and international standardizations such as IETF RFC 6163.