Service Activation System: IP Service Activator

Saburo Seto†, *Kenichi Yamane, and Souhei Majima*

Abstract

In this article, we introduce our service activation system called IP Service Activator. It is divided into a service module part, which consists of three kinds of scripts, and a platform part. It can handle a new service flexibly simply through the addition of a new service module. This minimizes the range of code that must be modified when service specifications change.

1. Service activation system

With the Internet being so popular not only in business environments but also with home users, telecommunication carriers and service providers are preparing various ranges of services such as VPN (virtual private network), VLAN/VMAN (virtual local area network, virtual metropolitan area network), Internet-access service, and VoIP (voice over Internet protocol). Telecommunication carriers must set up appropriate configurations of network elements (routers, switches, servers etc.) to provide these services. To provide services on network elements (NEs), several kinds of operations support systems (OSSs) are needed, such as a customer relations management system (CRM) and network resource management system (RMS). Among the OSSs, a system that accesses NEs and sets up appropriate configurations of services is called a service activation system. In this article, we introduce the service activation system called IP Service Activator, which NTT Network Service Systems Laboratories has developed as a general service activation system.

2. Requirements for service activation systems

A service activation system accepts service orders (SOs), which contain service request information, from upper-layer OSSs such as CRM and RMS and provides services on the network (**Fig. 1**). The service activation system must process numerous SOs and must provide services to numerous NEs.

With the number of services provided by telecommunication carriers increasing day by day, service activation systems must keep up with the expanding service menus and provide service provisioning methods. It needs to be easy to add a new provisioning method for a new service to the service activation system. After a new service has started, the service specifications and the commands of NEs will change when the carrier starts a value-added service or when the firmware of NEs is upgraded. To keep up with these changes, the activation methods of service activation systems must be easy to change.

3. System architecture of IP Service Activator

The provisioning methods depend on the network topology, service specifications, and know-how possessed by carriers. Once the code of the provisioning methods has been embedded in the system, it cannot be modified easily. Thus, the time required to develop new provisioning methods and modify current methods tends to be long when a new service starts or a current service changes. To overcome these problems, the architecture of IP Service Activator consists of two parts (**Fig. 2**): a service module part that depends on the service specifications and a platform module part that is independent of service specifications and provides common functions. The service module part is a module-oriented architecture: service modules can be added and removed freely

† NTT Network Service Systems Laboratories
Musashino-shi, 180-8585 Japan
E-mail: seto.saburo@lab.ntt.co.jp

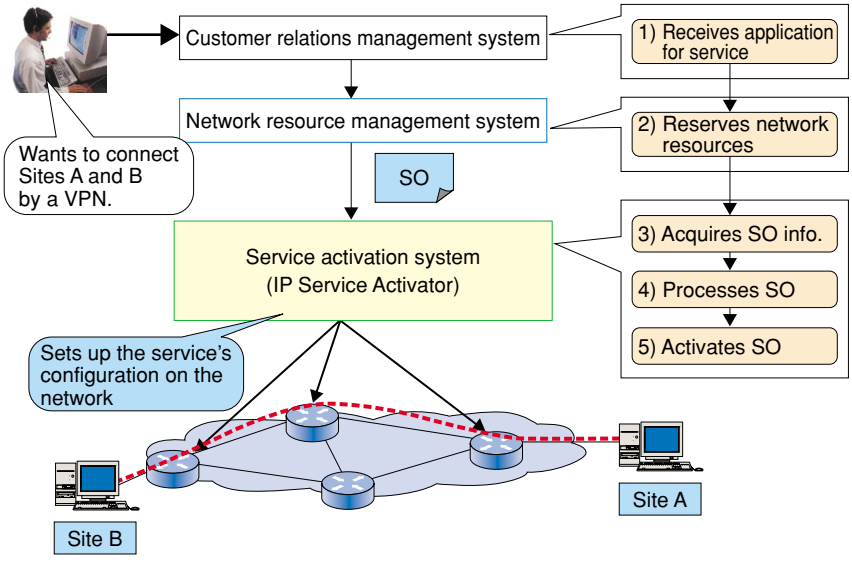

Fig. 1. Service activation system.

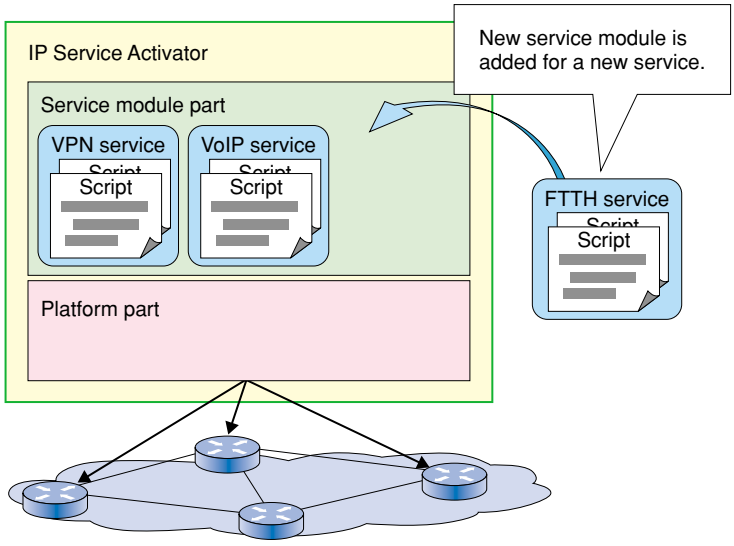

Fig. 2. System architecture of IP Service Activator.

according to the service being provided. When a new service starts, a new service module is added to the IP Service Activator. As the specifications of the interface between the service module and platform parts are defined, and these two parts are completely independent of each other, a provider can start a new service just by preparing a new service module. We chose to use lightweight languages such as Python and Perl for coding the service modules, so programmers can develop service modules in any environment in which scripts can run, such as on a personal computer. Our system reduces the development period and can handle changes to service specifications in a highly flexible manner because of the good characteristics of script languages, such as being easy to code and easy to debug. In our typical cases, a service module can be prepared within about two months.

The platform part supports SO operations and improves their efficiency. It supports the flow of SO operations from SO information acquisition from upper-layer OSSs to NE provisioning. It provides functions for parallel processing of SOs and exclusive access control to enhance performance and prevent interference to SOs. The operators can monitor the status of SO processing via a Web client.

4. Service module part

The service module part is divided into three parts and consists of three kinds of scripts to enable various kinds of provisioning methods (**Fig. 3**). This division minimizes code changes when the service specifications, network environment, or NE command specifications change. The role of each script is described below.

4.1 Type-Judge script

This script gets SO files containing application information, analyzes the file formats, and judges the types of services and types of SO. That is, it judges whether each SO is for a VPN or VoIP service and whether it is a new contract, a change to current settings, an application for additional services, or contract cancellation, and so on. It registers SOs in the IP Service Activator database.

As IP Service Activator can handle multiple Type-Judge scripts, these scripts absorb the differences in SO file formats. Even if there are multiple upper-layer OSSs and each of them outputs files with different formats, IP Service Activator can prepare corresponding Type-Judge scripts.

4.2 SO script

An SO script accepts SOs that have been registered by Type-judge scripts, divides them into parameters for each setting target (e.g., routers, switches, and servers), and defines their setting execution order.

4.3 NE script

An NE script accesses an NE, issues configuration commands, and activates a certain service. Generally, multiple NEs must be provided to activate one service, so multiple NE scripts must be executed. The execution order among the NE scripts is defined by the SO script, and the platform part calls NE scripts in execution order. When the service-providing network consists of multiple kinds of NEs, their setting commands are different, so IP Service Activator can select an NE script appropriate for the kind of NE.

5. Platform part

The platform part provides the functions that are common to all types of SOs. As described above, the service module part consists of three types of scripts. The platform part calls each script of the service modules sequentially and automatically. And to increase the efficiency of SO operations and avoid interfer-

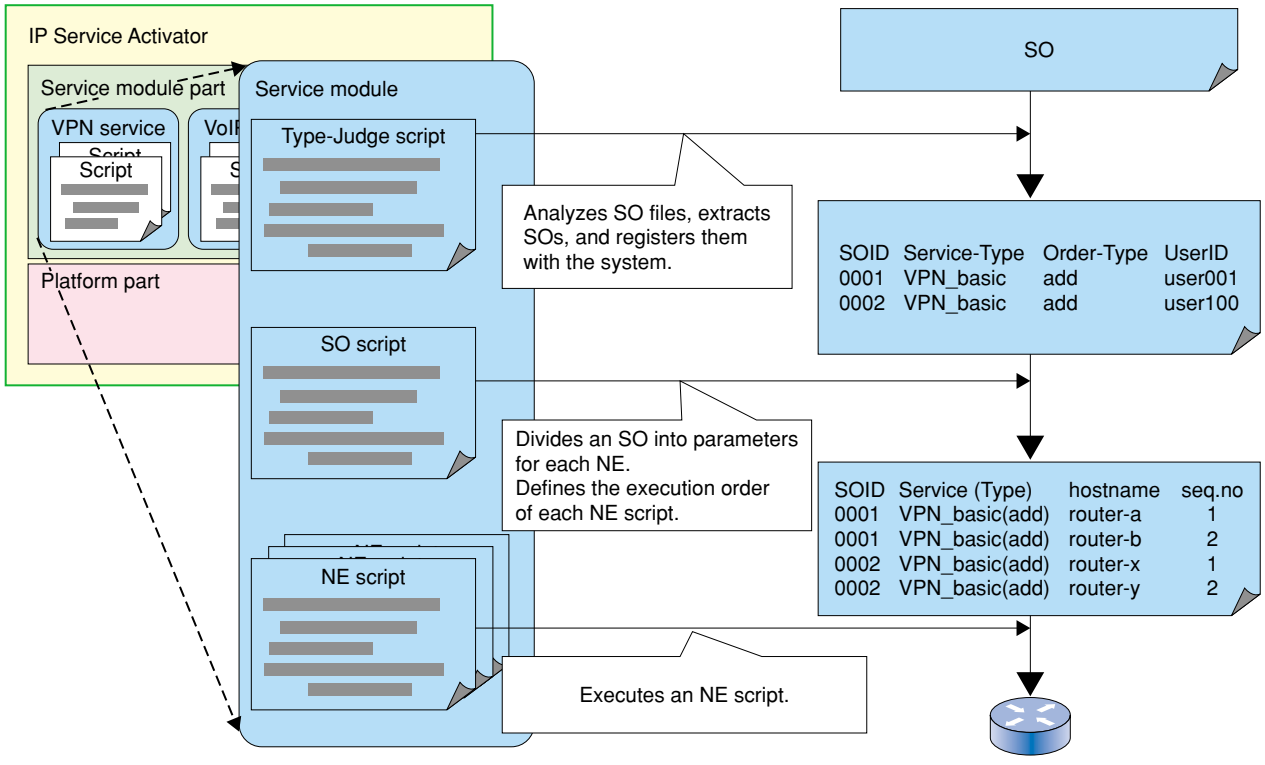

Fig. 3. Three scripts of a service module.

ence among SOs, it has parallel execution and exclusive access control functions.

5.1 Exclusive access control of NE scripts

If NE scripts of two SOs try to access to the same NE simultaneously, one or both of the SOs might be set up incorrectly, depending on whether the NE supports multiple access and the issuing of simultaneous setting commands. The platform part controls each NE script so that it does not access an NE while another NE script is accessing it. The platform part has a queue for each NE, and NE scripts that provide the same NE use the same queue and are executed in first-in-first-out (FIFO) style (**Fig. 4**).

On the other hand, NE scripts that access different NEs are executed simultaneously to enhance the throughput of the entire SO processing. As in the exclusive access control mentioned above, the platform part has queues for each NE, so the platform part places NE scripts in the queues and each queue executes NE scripts in FIFO style independently.

This exclusive access control function can prevent interference among SOs as well as NE scripts, enabling IP Service Activator to process multiple SOs in parallel. Compared with sequential processing, there is a big improvement in the number of SOs that can be processed.

5.2 Execution order control of NE scripts

When multiple NEs are needed to activate one service, we might have to keep a certain order in their provisioning. In IP Service Activator, this execution order is created by SO scripts. The platform part calls NE scripts according to the order in which SO scripts were created. If an NE script fails to set up the configuration, the execution of the following NE scripts must be canceled, so the platform part has a function for canceling the following NE scripts.

6. Future work

IP Service Activator provides the functions that control service modules, in other words, scripts. We think it is also applicable to fields other than SO operations, for example, to an NE testing system. However, for a testing system, realtime performance will be required. Therefore, we are adding realtime features to IP Service Activator.

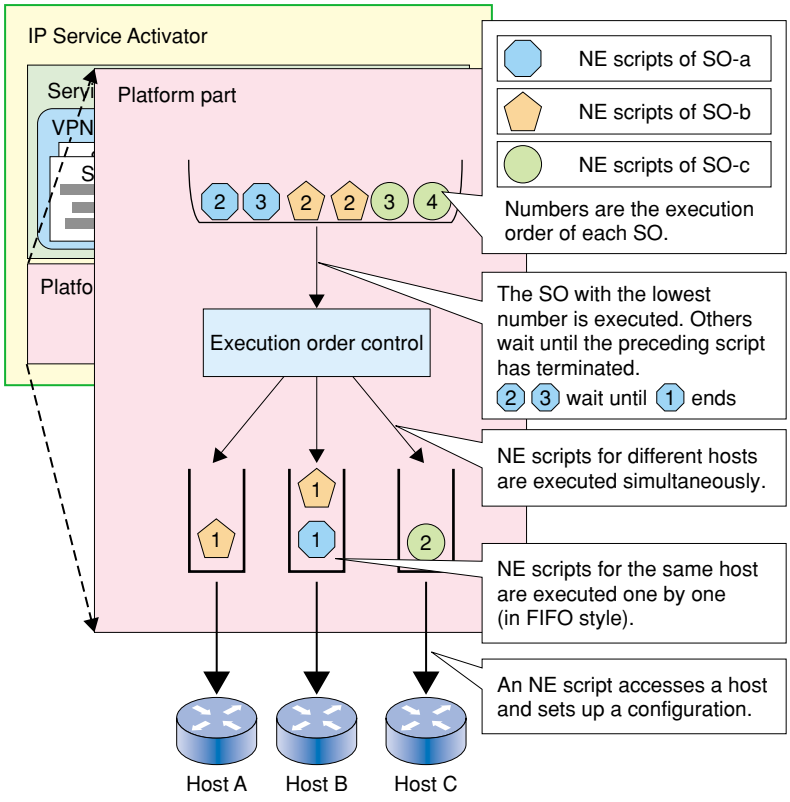

Fig. 4. Parallel execution and exclusive access control.

References

[1] A. Ohizumi, T. Furukawa, M. Ogawa, and K. Yamane, “A Method of Processing Service Ordering Data for Various IP-Network Services,” APNOMS2002, pp. 148-159, Sep. 2002, Jeju, Korea.

[2] A. Yamada, S. Seto, M. Fujiwara, K. Yamane, H. Taniguchi, T. Sugai, and S. Imai, “The Execution Method and Issue of Service Activation System,” IEICE Telecommunication Management 2003.

Saburo Seto

Engineer, Network Management Development Project, Second Promotion Project, NTT Network Service Systems Laboratories.

He received the B.S. and M.S. degrees in information science from the University of Tokyo, in 1996 and 1998, respectively. He joined NTT Network Service Systems Laboratories, Tokyo in 1998. Since then, he has been engaged in developing operations support systems.

Kenichi Yamane

Senior Research Engineer, Network Management Development Project, Second Promotion Project, NTT Network Service Systems Laboratories.

He received the B.E. and M.E. degrees in mathematics from Nagoya University, Nagoya in 1986 and 1988, respectively. Since joining NTT in 1988, he has engaged in R&D of a CTRON-based realtime operating system and operations support systems for the legacy PSTN. In 2001, he joined the next-generation network management systems development project. Currently, he is actively involved with IP communication management systems including VoIP. He is a member of the Institute of Electronics, Information, and Communication Engineers of Japan.

Souhei Majima

Senior Research Engineer, Supervisor, Network Management Development Project, Second Promotion Project, NTT Network Service Systems Laboratories.

He received the B.E. and M.E. degrees from Fukui University, Fukui Japan, in 1982 and 1984, respectively. He joined Nippon Telegraph and Telephone Public Corporation (now NTT) Laboratories in 1984. He has been involved in the development of a computer-aided software development environment (CASE) system and a switching equipment management system for large-scale networks. He is currently engaged in research on the operations support architecture for the next-generation network.