Service Management System for IP Services

Ken'ichi Yamane[†]*, Tsuyoshi Furukawa, and Toshihiro Nishizono*

Abstract

This paper proposes interface techniques that enable service management systems (SMSs) to make flexible connections with customer relationship management systems (CRMs) and network elements (NEs). In the service ordering process for new IP services, the proposed methods achieve high flexibility in the SMS implementation. The XML-based service order information description at the CRM-SMS interface allows quick IP service addition because the XML parsing mechanism means that interface programs hardly have to be changed. The virtual NE structure, which divides NE functions into atomic primitives, can introduce NEs provided by new vendors without modifying service order applications. We discuss some development results and evaluate the flexibility of the CRM and NE interfaces and the SMS performance.

1. Introduction

These days, many services are available on carrier IP networks and more will appear in the future. For example, VoIP (Voice over IP) services for consumers are now increasing and VoIP services for business users, including dial-in and call transfer, are expected. Another example is IP-VPN service, which provides a virtual private network. It is a popular service on carrier IP networks and remote VPN access services or VPN exchanges services will increase in the near future. Network elements (NEs), such as routers and servers, will change to provide such dynamically changing services.

When we provide services to customers, we start with a service order (SO) from the customer and put the configuration data in NEs so as to provide the services that the customer ordered. To automate this flow, we use two management systems. A customer relationship management (CRM) manages customer information and SOs from customers. A service management system (SMS) gets the SOs from the CRM and sets and stores NE configuration data for the SOs. The SMS decides which NEs should accommodate the customers for the services, and translates the SOs into command strings that make the specified NEs change their configuration data and sets the SO parameters for the specified NEs. Finally, the SMS stores the resulting configuration data and tells the CRM that the SO procedures have finished.

Because SMS must carry out such complicated processes, the SMS implementation must be very flexible when there are frequent modifications of network or services. Currently, however, no commercial systems that meet these requirements are applicable to a large carrier network.

We have developed an SMS that supports auto-configuration over many NEs to provide services in response to SOs from a CRM. Its architecture allows a flexible response to changes in:

1) service specifications
2) network structure, and
3) NE specifications

This flexibility is achieved using two methods

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

based on the adapter technology employed in the TMForum (Tele Management Forum) standardization. One uses the XML (eXtensible Markup Language) for describing the interfaces between CRMs and SMSs; the other uses our virtual NE concept, which abstracts NE service functions for commonly describing the interfaces between the SMS and NEs whose functions differ between vendors. These two methods have been applied to an SMS providing commercial VoIP services.

2. Overview of SMS

2.1 Target network

We considered a network based on the network model [1] put forward by the MSF (Multi-service Switching Forum). In MSF, network functions are logically divided into four planes: the adaptation, switching, control, and application planes. There is also a management plane, which includes several management functions, for example fault management, configuration management, and accounting management functions. An essential characteristic of the architecture is its flexibility in terms of the technology chosen to implement the functionality of each plane.

The target network (Fig. 1) consists of edge nodes in the adaptation plane (each is a media gateway (MG)), core nodes in the switching plane, servers such as call agents (CAs) in the control plane, and CRM and SMS in the management plane, which manage all the nodes and servers. A customer is accommodated by an MG via an access network or public switched telephone network (PSTN). MG-MG communications for transmitting VoIP packets run through the core nodes. The CA supports protocols such as SIP (Session Initiation Protocol) [2], H.323 [3], and Megaco (RFC3015) [4] to control VoIP services. The service provider can provide customers with a VoIP service [5] as well as other IP services such as IP-VPN [6]. This paper explains the SO flow between the CRM and NEs via our SMS using VoIP service as an example.

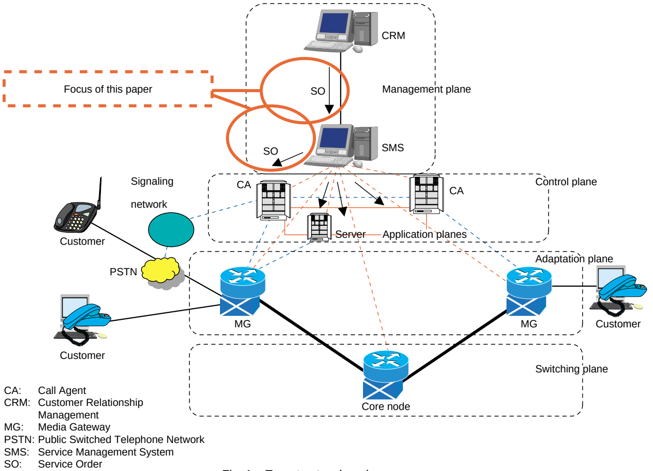

Fig. 1. Target network and our scope.

2.2 System architecture

The SMS is based on the NGOSS (New Generation Operation Support System) architecture [7], defined in TMForum. It consists of a common data bus, common database, and several operation functions. Other systems and NEs can be connected to this common data bus through the adapters, which transform vendor-specific interfaces into a common interface used in the system. Figure 2 outlines the NGOSS architecture. A cost effective and flexible system construction is possible by connecting COTS (Commercial Off The Shelf) products, which perform some of the operations functions, to a common data bus through the adapters. The custom-developed functions perform the rest of the operations for setting the configuration data in the NEs in cooperation with the COTS products. The CRM is also connected through the adapters to interact with the SMS. The methods proposed in this paper are mainly concerned with the adapters described in the NGOSS architecture.

3. Interface between CRM and SMS

To decide the interface between the CRM and SMS, we examined application layer data formats and lower layer protocols.

3.1 Application layer data formats

There are three main data formats that can be used for communication between the SMS and CRM for handling SO flows.

(1) Binary format over socket, CORBA, etc.
(2) Character strings using comma separated values (CSV)
(3) Tag structure using XML

The binary format is mainly used by legacy systems. It can be processed at high speed and allows rich data structures, but it is hard to understand, and data transmission formats cannot be easily changed, so software maintenance is very difficult. Many problems such as the difficulty of correcting interface inconsistency will occur if we extend the system to interconnect with other systems. CORBA (Common Object Request Broker Architecture) alleviates this problem by using IDL (Interface Description Language) and IIOP (Internet Inter-ORB Protocol), but there are still many problems when connecting different commercial CORBA products employed in different OSSs. When we connect two objects through CORBA, we define the interfaces using IDL. The descriptions are interpreted into two interface programs (the stub and skeleton) for each object. Generally, the programming languages of the two objects are different. This mechanism works well when the interpretation is done by the same commercial CORBA products or when the interface definitions are simple. However, when we define complicated interfaces for different commercial CORBA products, problems occur because the interpretation of such a complicated interface frequently differs between commercial CORBA products. So when we connect two or more objects with CORBA, we can only use simple IDL descriptions, such as “wchar” or “wstring”, which denote the simple data type of a character or a string of characters. Complicated structures, such as “any” for any data type or “struct” for

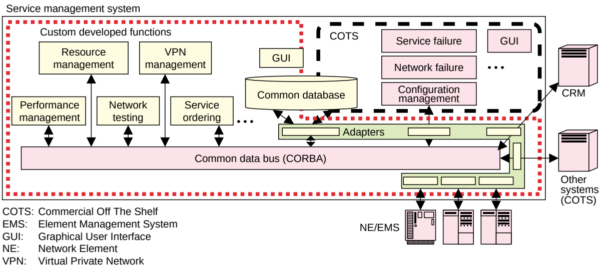

Fig. 2. System architecture.

structured data, cannot be used.

The CSV data format is often used when connecting systems for simply exchanging data, because it can be easily created with commercial software such as a spreadsheet and is easy to understand because it has a simple text format. However, its descriptive ability is too weak to easily describe the structured data types required for defining SO parameters that are frequently changed when adding new network services. So CSV is not suitable for the interface between the CRM and SMS.

The XML data format combines the advantages of the binary and CSV formats. Tags can define a structured data type, and since it is a text format, it is easy for us to understand. Furthermore, if we extend it with XSL (eXtensible Stylesheet Language) [8], then it becomes easy to display on a Web browser. XML is used for combining systems in various fields, such as electronic commerce. In the network management field, trials have already begun, such as expressing the management information base (MIB) on the common management information protocol (CMIP) by XML. For these reasons, we chose XML for describing the data formats for communication between the SMS and CRM (Fig. 3). Table 1 summarizes the comparison of these application layer data formats.

3.2 Lower layer protocol

To transmit the application layer data described in XML, we consider the session layer protocol between the application objects. Two kinds of protocols are currently used for this purpose: synchronous communication like IIOP and asynchronous communication like MOM (Message Oriented Middleware). Since synchronous communication allows response signals which indicate whether the communication peer has received data normally or not, it can provide assured communication. On the other hand, asynchronous communication need not wait for the response from a communication peer, so it allows faster dialogues. For service ordering, assured communication is more important than speed. For example, when SO commands to NEs fail, the SO flows should be rolled back to the last successful synchronous point. Synchro-

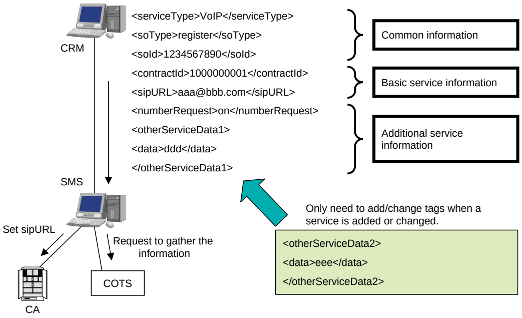
CRM
<serviceType>VoIP</serviceType>
<soType>register</soType>
<soId>1234567890</soId>
<contractId>1000000001</contractId>
<sipURL>aaa@bbb.com</sipURL>
<numberRequest>on</numberRequest>
<otherServiceData1>
<data>ddd</data>
</otherServiceData1>
Common information
Basic service information
Additional service information
SMS
Set sipURL
Request to gather the information
COTS
CA
Only need to add/change tags when a service is added or changed.
<otherServiceData2>
<data>eee</data>
</otherServiceData2>

Fig. 3. Flexible structure of XML-based data of SO.

Table 1. Comparison of application layer data formats.

	Binary format	CSV format	XML format
Performance	Very good	Good	Poor
Ability to describe structured data	Very good	Poor	Very good
Ease of software maintenance	Poor	Poor	Very good
Inter-connectivity	Poor	Good	Very good

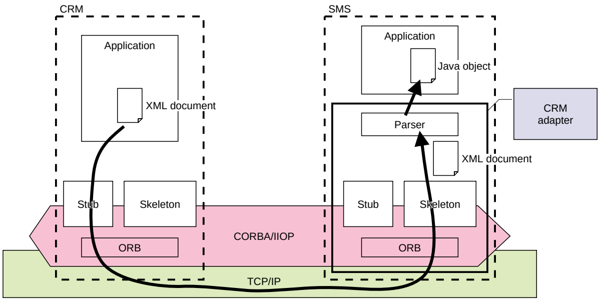

Fig. 4. Overview of XML over CORBA communication.

nous communication like IIOP allows such rollback mechanisms. In addition to the better compatibility with the communication inside the SMS, which is based on CORBA, we chose synchronous communication on IIOP. An overview of communication between the CRM and SMS using XML documents over CORBA is shown in Fig. 4. The outline of actual SO processing flow treated in the adapter is as follows.

(1) The adapter in the SMS contains the stub and skeleton programs translated from the IDL description for CORBA communication with the CRM. An XML parser for analyzing the XML documents of SO information is created.
(2) Using the skeleton program, the adapter gets the XML document from the CRM over the CORBA interface, which encapsulates the document in the "wstring" data type.
(3) It passes the document to the XML parser to create a Java object of the SO information which can be handled in the SO applications.
(4) Finally, the adapter calls an appropriate SO application with the Java object.

4. Interface between SMS and NEs

4.1 Layered interface modules

IP network services are provided through the cooperation of many types of functions that are widely distributed on different types of NEs. We use the generic term network element (NE) for routers, servers, and other elements located in an IP network. For example, a VoIP service based on SIP is provided by SIP-server functions (such as registration, proxy, and redirect functions) cooperating with location-server and DNS-server functions. The SIP-server functions can be integrated into a single server (i.e., an NE) or distributed to three servers. Furthermore, before the SO parameters are set in the servers for VoIP service functions, the VoIP terminal must be connected to the IP network. For this purpose, SO parameters for IP services are set up in edge-routers, RADIUS (Remote Authentication Dial In User Service)-servers, etc. SMS must manage all of the above types of NEs to provide end users with the VoIP service, so the SO processing for IP network services is complicated.

Furthermore, to offer new service quickly and cheaply, it will be necessary to use NEs from multiple vendors. Thus, SMS must:

- Minimize the impact of changes in service specifications and the addition of new services.
- Minimize the impact of changes in network topology like the addition of new types of NEs or the distribution and integration of functions in NEs.

Next, we describe the NE-adapter that connects NEs with the SMS to set SO parameters. We divided SO processes into: i) an SO-application that checks the consistency of all parameters, decides which NE should be set with the SO parameters, manages the result of all SO processes, and so on and ii) an NE-adapter, which translates the SOs into the appropriate NE command strings, communicates with the NEs, and sets the command strings in the NEs. We separated the NE-adapter functions from SO-application functions to localize the impact of changes in NE implementations (which may differ among vendors) or service specifications in the NE adapter. Furthermore, we divided the NE-adapter into a common-NE-adapter and vendor-interface modules.

- Common-NE-adapter

This adapter provides common NE-independent interfaces such as "set" and "get" to SO-applications. It also distributes SO parameters to the vendor NE-adapter described below. It ensures that an SO-application can always handle SO commands and parameters in common forms, so there is no need to worry about what kind of vendor NEs exist in the network. It can be developed independently of the NE specifications (Fig. 5).

- Vendor-NE-adapter

An adapter is prepared for each vendor's NE. This adapter directly communicates with the actual NE using its specific protocol, and translates the common SO parameters into the specific NE command strings. It conceals vendor-dependent specifications.

This two-layer structure localizes the impact of NE implementation within the vendor-NE-adapter. The common-NE-adapter localizes the impact of the network structure within itself and also simplifies the application programming interfaces offered to an SO-application.

4.2 Virtual NE

The structure of interface modules between the SMS and NEs described above enables flexible composition. However, in the following situations, it is still necessary to change the SO application according to changes in NE implementation.

- When two or more functions integrated into NEs of one type are distributed to two or more types of NEs.
- When an NE with new functions is added to the network by service addition.

For example, when a CA supports three SIP-server functions, the SO-application puts SO parameters only in the NE-adapter for the CA. But when the three functions are divided among three kinds of servers, the SO-application must put the SO parameters in three adapters.

To avoid the problem, we employ a virtual network element (VNE). This represents a primitive set of functions atomically divided from the original NE's functions. That is, an NE is divided into several VNEs, each of which can be separately located on a

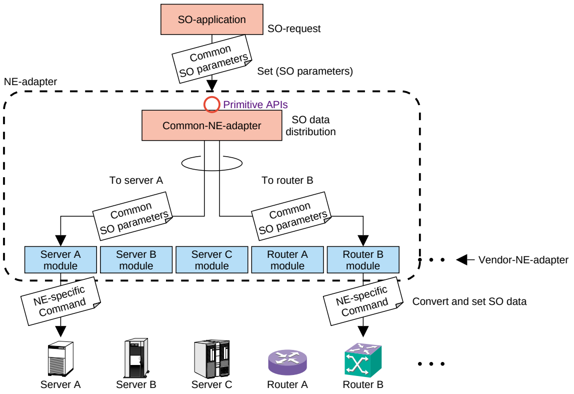

Fig. 5. Overview of NE adapter processing.

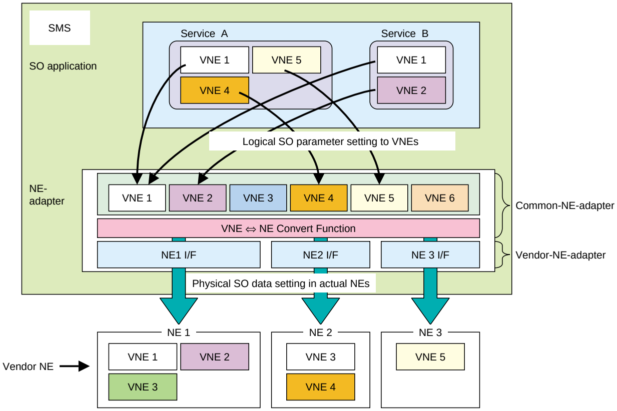

Fig. 6. Outline of SO data processing using vendor network elements (VNEs).

different element acting as a type of primitive NE. For example, when a CA has a SIP-server function and gatekeeper function, the CA is divided into "registration-server", "proxy-server", "redirect-server", and "gatekeeper" VNEs. If the SO application is developed on this VNE model, it is not influenced by a change in the distribution of NE functions as long as the VNEs were appropriately configured, because even if functions in one NE are distributed to two or more NEs, an VNE is not divided and its function does not change.

Figure 6 outlines the processing of SO parameters. Since SO application is aware only of VNEs and sets SO parameters in VNEs, the NE-adapter conceals differences among vendor implementations. The system is flexible with respect to allowing NEs to be changed to ones provided by other vendors.

4.3 SO processing flow in SMS using VNE

Figure 7 shows the flow of processing within an SO-application using the VNE concept. We roughly divide the functions of the SO-application into three functional blocks: scenario control, SO setup, and SO log read/write.

1. When the CRM adapter receives SO data, it notifies the scenario control function in the SO application. This function executes the operation scenario indicated in the header part of the SO parameters. All the scenarios are stored in the scenario database.
2. The SO setup function, which is called by the scenario control function, decides which VNEs should be set up with the SO parameters using the configuration database. This function also calls the common-NE-adapter with the list of selected VNEs.
3. The common-NE-adapter refers to the VNE list to obtain the actual NEs using the configuration database. It also sets the SO data in the vendor-NE-adapter prepared for each vendor's NE.
4. The vendor-NE-adapter converts the SO parameters into the vendor-NE-specific command sequences and sets them in actual NEs.
5. When all processes have finished, the scenario control function reports the results to CRM.

These processes are logged by the log R/W function in the log database. This lets the scenario control function perform an SO rollback if process errors are detected.

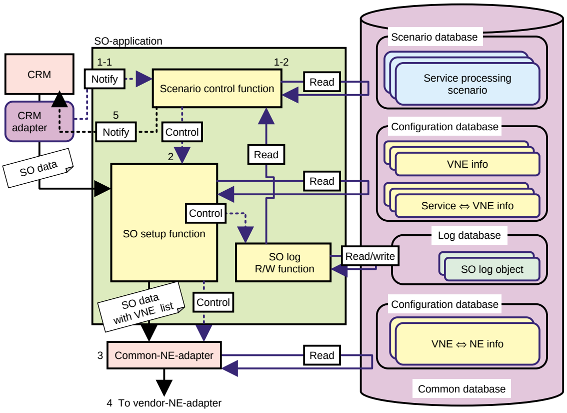

Fig. 7. Service ordering process using VNEs.

5. Evaluation

5.1 Flexibility of CRM interface

We built a flexible CRM interface by introducing adapters with the XML-based interface. As a result, we could add new services for business users (such as call-transfer and dial-in), within three months after determining service specifications, compared with the usual six months. Most of the development was done in the SO-application programs. The interface program development effort was about 1/10 of that for application program development. In the interface program, we only needed to develop definitions of new XML tags and a few additions of SO-application call sub-routines. The session layer communication drivers needed no modification. This shows that our XML-based interface has enough flexibility.

5.2 Flexibility of NE interface

We were able to add a new CA supplied by a different vendor from the old ones. In the interface for setting SOs, the command name, parameter formats, and communication protocols are different from those of the old CA. However, the development was completed within two months because we achieved it simply by added a new VNE module without changing the application programs or the common NE-adapter. This example also shows the validity of the NE adapter architecture based on the VNE.

5.3 Performance

In general, flexibility and performance have a trade-off relationship. In particular, higher performance is needed for the parsing XML. The performance evaluation of our SMS showed that it is possible to process one SO in less than 1 s, and to process thousands of SOs in one hour in the condition that the SOs flow from the CRM to NEs (the SO currently uses 42 XML tags). This satisfies the user requirements and will have sufficient performance for general service order processing systems. However, in the future, the performance should be improved using appropriate parser algorithms because we expect the number of XML tags to increase to handle more complicated services.

6. Conclusion

To cope with increasingly complex IP services, we have proposed interface technologies using two types of adapters: XML technology for the interfaces between CRM and SMS and a virtual network element concept for the interface between SMS and NEs.

In the service ordering process for new IP services, these techniques make the SMS implementation flexible. The XML-based service order information description at the CRM and SMS interface allows quick IP service addition. When adding new services for business users, the changes in the interface programs amount to only 1/10 of those of the application programs thanks to the parsing mechanism. The virtual NE structure which divides NE functions into atomic primitives lets us introduce NEs provided by new vendors without modifying service order applications. These results were confirmed by implementing new VoIP services and introducing a new SIP server. The methods described in this paper are applicable to other IP service provisioning systems, such as IP-VPN service or content delivery service.

References

[1] MSF, "Multiservice Switching Forum Implementation Agreement, MSF-ARCH-001.00-FINAL IA,", 1. May 2000.
[2] IETF, "SIP: Session Initiation Protocol,", 1999.
[3] ITU-T, "ITU-T Recommendation H.323,", 1996.
[4] IETF RFC3015, "Megaco Protocol Version 1.0,", 2000.
[5] Takehisa Ichijo, "The OSS architecture for VoIP service ordering based on the MSF network," APSITT2001, Kathmandu, Nepal/Atami Japan, pp. 259-263, Nov. 2001.
[6] Tamotsu Ohyama, "A Study of Operation Support System Architecture," TECHNICAL REPORT OF IEICE TMWS2001-19, p. 115, 2001.
[7] TMF, "NGOSS Architecture Technology Neutral Specification - TMF053 v3.0,", 2003.
[8] W3C Recommendation, "Extensible Markup Language (XML) 1.0,", 1998.

Ken'ichi Yamane
Senior Research Engineer, First Promotion Project, NTT Network Service Systems Laboratories.
He received the B.S. and M.S. degrees in mathematics from Nagoya University, Nagoya, in 1986 and 1988, respectively. In 1988, he joined the Switching Systems Laboratories, Nippon Telegraph and Telephone Public Corporation (now NTT), Tokyo, Japan. He worked on the development of the public switched telephone network (PSTN) and its operations support systems, ATM network maintenance, design system development for ATM virtual paths, and so on. He is currently working on IP control and management service research.

Tsuyoshi Furukawa
Engineer, First Promotion Project, NTT Network Service Systems Laboratories.
He received the B.S. and M.S. degrees in electronics from Waseda University, Tokyo, in 1993 and 1995, respectively. In 1995, he joined the Optical Network Laboratories, Nippon Telegraph and Telephone Public Corporation (now NTT), Tokyo, Japan. He worked on the development of operations support systems for the optical transport network and IP network. He is currently working on IP control and management service research.

Toshihiro Nishizono
Senior Research Engineer, Supervisor, Network Software Service Project, NTT Network Service Systems Laboratories.
He received the B.S., M.S., and Ph.D. degrees in information engineering from Kyushu University, Fukuoka, in 1978, 1980, and 1990, respectively. In 1980, he joined the Electrical Communication Laboratories, Nippon Telegraph and Telephone Public Corporation (now NTT), Tokyo, Japan. He worked on the development of DDX packet switching systems and ATM switching systems. From 1990 to 1993, he worked in Advanced Telecommunication Research Laboratories, Kyoto, Japan, on software specifications and distributed operating systems. From 1998 to 2001, he was engaged in international communication service deployment and IP service development in NTT Communications. He is currently working on IP control and management service research.