33

Implementation and evaluation of MIB tester for OSI management

Keizo SUGIYAMA, Hiroki HORIUCHI, Sadao OBANA and Kenji SUZUKI
KDD R&D Laboratories
2-1-15 OHARA KAMIFUKUOKA-SHI, SAITAMA 356, JAPAN
Tel: +81 492 78 7328
Fax: +81 492 78 7510
e-mail: sugiyama@csg.lab.kdd.co.jp

Abstract

This paper proposes a conformance test method for Management Information Base (MIB) and discusses an implementation and an evaluation of a MIB tester. The proposed test method is based on and an extension of the existing conformance test method for protocol entity. We show the practical solution to the scope of testing by conducting the capability test to all test cases and the behaviour test limited to the test cases actually used. In the implementation, the MIB tester generates test scenarios for capability tests automatically. We evaluate the proposed test method and demonstrate the effectiveness of the MIB tester through its application to the actual agents.

Keywords

OSI management, conformance test, MIB, GDMO

1 INTRODUCTION

In line with the progress of standardisation on Telecommunications Management Network (TMN) [M3010, 1992], it is indispensable for the development of network management systems to conduct a test of OSI management.

A. A. Lazar et al. (eds.), *Integrated Network Management V*

In OSI management, a manager accesses to Management Information Base (MIB) in an agent, which is a collection of managed objects, by using Common Management Information Protocol (CMIP). The management information and the behaviour are defined by Guidelines for the Definition of Managed Objects (GDMO) templates. It is necessary for the test of OSI management to verify the behaviour of managed objects, as well as the behaviour of CMIP protocol entities. Although a conformance test method for protocol entity has been established [X290, 1992] [Serre, 1993], that for MIB has not been established so far [Ödling, 1994]. The existing conformance test method for protocol entity, however, can not be applied to that for MIB just as it is since there are differences in modelling between protocol entities and managed objects. Development of the conformance test method for MIB is strongly expected.

This paper proposes the conformance test method for MIB and describes an implementation and an evaluation of a MIB tester. First, we discuss the applicability of the existing conformance test method for protocol entity to that for MIB. We propose the conformance test method for MIB by taking account of the differences in modelling between protocol entities and managed objects. We describe the MIB tester based on the proposed test method. Finally we evaluate the proposed test method and the effectiveness of the MIB tester through the application to tests of actual agents.

2 OVERVIEW OF OSI MANAGEMENT AND EXISTING CONFORMANCE TEST METHOD FOR PROTOCOL ENTITY

2.1 OSI management

There are two roles in OSI management systems. One is a manager which has a responsibility for managing and the other is an agent which is managed by a manager. Management operations to an agent are conveyed by CMIP. It defines Protocol Data Units (PDUs) corresponding to services of Common Management Information Service (CMIS). The management information and the behaviour of a managed object are defined by nine kinds of GDMO templates: MANAGED OBJECT CLASS, PACKAGE, ATTRIBUTE, ATTRIBUTE GROUP, BEHAVIOUR, ACTION, NOTIFICATION, NAME BINDING and PARAMETER templates. These templates may refer to Abstract Syntax Notation 1 (ASN.1) modules which specify syntax of management information over CMIP.

2.2 Existing conformance test method for protocol entity

The existing conformance test method for protocol entity [X290, 1992] examines whether or not an implementation by vendors, so called an Implementation Under Test (IUT), conforms to the relevant protocol specification. The test is conducted

by connecting with a test system. The test suite, a set of test cases, is developed from the protocol specification and the Protocol Implementation Conformance Statement (PICS) which gives information about implemented capabilities.

The test suite consists of basic interconnection tests, capability tests and behaviour tests, with the different test objective. Basic interconnection tests verify the basic capability of interconnection. Capability tests and behaviour tests verify the static and the dynamic conformance respectively.

There are two kinds of test components: the Lower Tester (LT) and the Upper Tester (UT), which control and observe the lower and upper service boundaries of the IUT respectively. The Test Coordination Procedure (TCP) is the rules for cooperation between the LT and the UT during testing. The Point of Control and Observation (PCO) is a point where the occurrence of test events is to be controlled and observed. The abstract service primitives (ASPs) are exchanged via PCOs. Four test configuration patterns exist depending on the location of the IUT and the LT, the number of PCOs and the existence of PDUs for controlling tests (i.e. TM-PDUs), as shown in Figure 1.

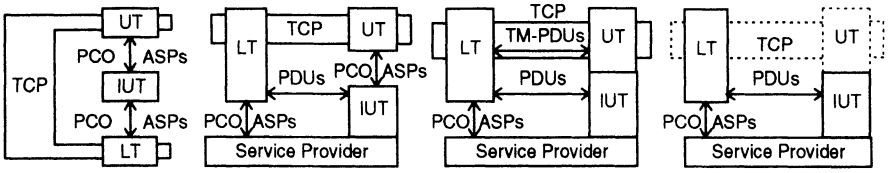


Figure 1 Four test configuration patterns of conformance test for protocol entity.

3 PROPOSAL ON CONFORMANCE TEST METHOD FOR MIB IN OSI MANAGEMENT

3.1 Differences in modelling between protocol entities and managed objects

Table 1 shows the differences in modelling between protocol entities and managed objects. The behaviour of a protocol entity, modelled as a finite state machine, is determined by a current state and an input event from a peer entity. On the other hand, the behaviour of a managed object is determined not only by a set of current attribute values and a CMIP PDU from a manager but also input events from actual resources and other managed objects in MIB. The last two types of input events may result in the modification of attribute values and the emission of voluntary notifications without interacting with a manager.

Table 1 Differences in modelling between protocol entities and managed objects

	Protocol entity	*Managed object*
Model	● Finite state machine	● Abstraction of actual resource by object-oriented approach
Spec. description	● State transition table/diagram ● Formal Description Technique (FDT)	● GDMO definition (The behaviour is described by natural language)
Character-ised by	● Input events ● Set of states ● State transition ● Output events	● Attributes ● Permitted operations ● Emittable notifications
Interaction with outside world	● Primitives with upper and lower layers ● PDUs with a peer entity included in primitives	● CMIP PDUs with a manager ● Actual resources ● Other managed objects in MIB

3.2 Applicability of existing conformance test method for protocol entity to that for MIB

3.2.1 Test configuration

Protocol entities provide service boundaries to exchange primitives with upper and lower layers, while managed objects do not provide upper service boundary from a modelling point of view. Therefore, the local test methods and the distributed test methods, in which a PCO exists between a UT and an IUT, are not suited as a test configuration of conformance test method for MIB. Although the coordinated test methods need a UT, there are few cases that it is possible to implement an UT in MIB additionally. The remaining remote test method, which needs only a LT and an IUT, is suited as the pattern of test configuration.

3.2.2 Test purpose

Basic interconnection test

The purpose of basic interconnection tests is to establish the sufficient conformance for interconnection to be possible, without performing thorough testing. The basic interconnection test for protocol entity verifies the main features in a protocol and/or transfer syntax specification. On the other hand, as a

manager interacts with managed objects of an agent by operations and notifications, the basic interconnection test for MIB should verify the possibility of receiving and responding operations and issuing notifications for managed objects often used, by setting the CMIS parameters to typical values.

Capability test

The purpose of capability test is to verify the existence of claimed capabilities of an IUT. The capability test for protocol entity is conducted according to a PICS. In case of managed objects, their capabilities are claimed by a Managed Object Conformance Statement (MOCS) and a Managed Relationship Conformance Statement (MRCS) for name bindings [X724, 1994]. The capability test for MIB should verify the capability of performing all permitted operations and issuing all emittable notifications for each managed object instance as follows.

- Obtain all readable attribute values by issuing M-GET.
- Modify all writable attribute values by issuing M-SET.

A 'ModifyOperator' parameter value is set according to the implementation status of each attribute.

- Invoke all actions by issuing M-ACTION.
- Create managed object instances by issuing M-CREATE.

A 'managed object instance', a 'superior object instance' and a 'reference object instance' parameter values are set according to the implementation status for 'create with automatic instance naming' and 'create with reference object'.

- Delete managed object instances by issuing M-DELETE.
- Examine all emittable notifications by receiving M-EVENT-REPORT.

Behaviour test

The purpose of behaviour test is to verify an IUT as thoroughly as possible, over the full range of dynamic conformance requirements. The behaviour test further consists of a valid behaviour test and an invalid behaviour test.

- valid behaviour test;

While the valid behaviour test for protocol entity verifies the dynamic behaviour such as the state transition, the variation and the combination of parameter values in PDUs etc., the valid behaviour test for MIB should verify the dynamic behaviour of managed objects and that of MIB as a whole according to the BEHAVIOUR template's definition.

- invalid behaviour test;

The invalid behaviour test for protocol entity uses 1)syntactically invalid test event, 2)semantically invalid test event and 3)inopportune test event. The invalid behaviour test for MIB should use test events 1) and 2) in order to verify the invalid behaviour for a syntax/semantics error of a parameter value since management information is also conveyed as a transfer syntax of ASN.1. As to 3),

this test should be conducted only if the attributes which indicate some states such as an administrative state are included in the managed object.

3.3 Scope of testing

As we can define all test cases of the capability test for protocol entity corresponding to items in a PICS, so we can define all test cases of the capability test for MIB corresponding to items in a MOCS/MRCS.

It is possible to define almost all test cases of the behaviour test for protocol entity since protocol entity is modelled as finite state machines, though the number of test cases becomes huge. The test can be conducted by sending possible input events for all states and observing the reactions. On the other hand, it is practically impossible to define all test cases of the behaviour test for MIB because of the following two reasons. One is that the interaction with actual resources and other managed objects should be taken into consideration. The other is that the same test case may produce different test outcomes on different occasion since the behaviour of actual resources are affected by environmental and time-dependent conditions. Consequently, we should create the test suite for the capability test to all test cases and the one for the behaviour test limited to the test cases actually used.

Hereafter we call the test suite, a *scenario*, and the test system corresponding to LT, a *MIB tester*. The MIB tester assumes that CMIP communication can be correctly performed between a manager and an agent by using the underlying protocols.

4 IMPLEMENTATION OF MIB TESTER

4.1 Fundamental design principles

- The scenario for capability test is automatically generated, and the one for behaviour test is manually created. The capability test is substituted for the basic interconnection test.
- Applicable to various agents with different GDMO definitions and naming trees, without modification of its programs.
- The program is executed under Windows NT®. The CMIP board, commercially available [Idoue, 1990], is used for protocol processing of CMIP. The C language is used for programming.

4.2 Realisation of MIB tester

This section describes the realisation method of each function in the MIB tester. Figure 2 shows the software module configuration required for these functions.

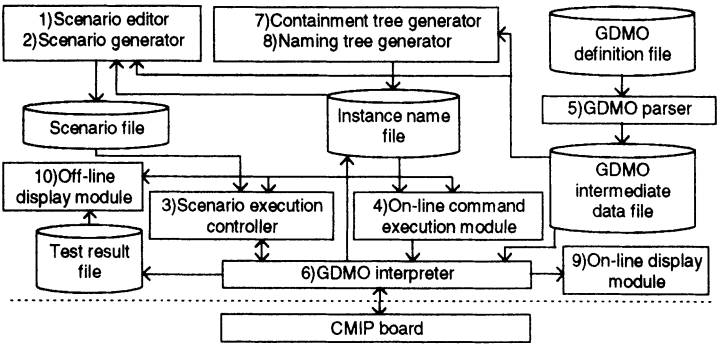


Figure 2 Software configuration of MIB tester.

4.2.1 Scenario creation and execution function

This function supports the creation of a scenario, which is a sequence of CMIS operations and control commands, and the execution of the created scenario by the following modules.

1)**A scenario editor**, which inputs the parameter values for a sequence of CMIS operations and control commands manually and stores them in a scenario file.

2)**A scenario generator**, which generates operations for each managed object instance from the GDMO definitions and the naming tree, and stores them in the scenario file. The details on automatic scenario generation mechanism in a scenario generator are described in 4.3.

3)**A scenario execution controller**, which reads the scenario file and controls the execution of the scenario according to the control commands.

4)**An on-line command execution module**, which immediately issues a CMIS operation input by a test operator manually.

A test operator can specify the control of scenario execution into the scenario file. For this purpose, the scenario editor provides him with control commands of WHILE/WEND (to iterate the portion specified by a label), WAIT (to wait for the specified period), GOTO (to go to the line specified by a label), TIMER(to start at the specified time), and BELL (to sound a beep).

4.2.2 GDMO encoding and decoding function

This function supports the encoding and decoding of parameter values of any GDMO definitions without modification of its programs by the following modules.

5)**A GDMO parser**, which inputs the GDMO definition file and outputs the result of analysis to the GDMO intermediate data file.

6)**A GDMO interpreter**, which encodes and decodes parameter values of managed objects in CMIS primitives by referring to the GDMO intermediate data file.

4.2.3 Instance name management function

This function supports the management of instance names registered manually or extracted from received primitives by the following modules.

7)**A containment tree generator**, which creates a containment tree based on the NAME BINDING definition.

8)**A naming tree generator**, which creates a naming tree from received primitives or names input manually.

4.2.4 Test result analysing function

This function supports the analysis of the responses of operations and the notifications on a CMIS primitive level, including syntactic and semantic check of parameter values by the following modules.

9)**An on-line display module**, which displays the result of analysis for primitives and some CMIS parameter values immediately after the primitive is received.

10)**An off-line display module**, which displays the result of analysis for all parameter values of each primitive stored in a test result file.

The on-line display module displays the following items for each primitive: time for sending or receiving a primitive, InvokeID, LinkedID, Mode (i.e. confirmed/non-confirmed), ManagedObjectClass and EventType. The off-line display module displays all parameter values of a primitive sent or received as shown in Figure 3.

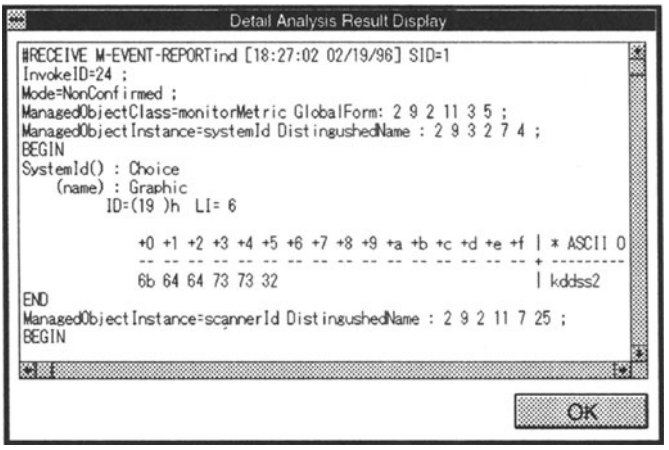


Figure 3 Off-line display window for M-EVENT-REPORT indication.

4.3 Automatic scenario generation mechanism

The following rules are applied to each managed object instance acquired by an automatic acquisition procedure, which traverses the naming tree by issuing M-GET with the scope parameter value set to 'firstLevelOnly' repeatedly. The values of CMIS primitive parameters not mentioned here such as synchronisation and filter are omitted or set to default.

Generation of M-GET operations

The automatic acquisition procedure serves both to acquire the naming tree and to obtain attribute values in each instance, namely, to verify whether or not all attribute values are correctly read. In order to verify the effect of performed operations, an M-GET operation is also generated after an operation other than M-GET described below is generated.

Generation of M-SET operation

An M-SET operation is generated depending on 'propertylist' in an ATTRIBUTE clause within the PACKAGE template as follows.

- 'REPLACE-WITH-DEFAULT' is included in 'propertylist'.

  An M-SET with the 'modifyOperator' parameter value set to 'setToDefault' is generated.
- 'REPLACE/ADD/REMOVE' is included in 'propertylist'.

  Default values for simple types of ASN.1 such as INTEGER and BOOLEAN are given by users in advance. Default values for constructed type are given as the combination of the default values for simple types. For 'REPLACE/ADD' of 'propertylist', an M-SET with the 'modifyOperator' parameter set to 'replace/addValues' and the attribute value set to the above default value is generated. For 'REMOVE' of 'propertylist', an M-SET with the 'modifyOperator' parameter set to 'removeValues' is generated for the attribute added by the previously generated M-SET or retrieved at the acquisition of the naming tree.

Generation of M-ACTION operation

An M-ACTION is generated with the 'ActionType' parameter set to the object identifier in the ACTION template and the 'ActionInfo' parameter set to the default value as described in the above M-SET operation if the 'ActionInfo' parameter needs to be set.

Generation of M-CREATE operation

When a 'CREATE' clause exists within the NAME BINDING template, regarding the superior managed object instance, an M-CREATE is generated according to the following rules.

- 'WITH-AUTOMATIC-INSTANCE-NAMING' exists in 'CREATE' clause.

  An M-CREATE with the 'superior object instance' parameter set to the instance name of the superior managed object is generated.
- 'WITH-AUTOMATIC-INSTANCE-NAMING' does not exist in 'CREATE' clause.

  An M-CREATE with the 'managed object instance' parameter set according to the ASN.1 definition of naming attribute is generated.
- 'WITH-REFERENCE-OBJECT' exist in 'CREATE' clause.

  An M-CREATE with the 'reference object instance' parameter set to any managed object instance of the same class as the created one is generated.

Generation of M-DELETE operation

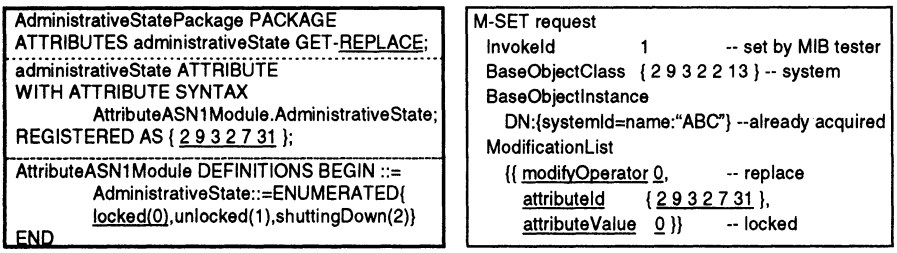
When a 'DELETE' clause exists within the NAME BINDING template for a superior managed object, an M-DELETE operation is generated according to the following rules.

- An M-CREATE has been automatically generated.

  An M-DELETE is generated for the managed object instance generated in the above M-CREATE procedure.
- An M-CREATE has not been automatically generated.

  An M-DELETE is generated for any managed object instance which belongs to the superior object, such as the first one.

Figure 4 shows an example of automatic generation of CMIS operation for 'administrativeStatePackage' in the 'system' managed object class [X721, 1992]. In Figure 4, underlined parts in (a) are reflected to the underlined parameters of an M-SET operation in (b).



(a) GDMO definitions (b) Generated CMIS operation

Figure 4 Example of automatic generation of CMIS operation.

5 EVALUATION AND DISCUSSION

5.1 Proposed conformance test method for MIB

Test configuration

In case of protocol entities, the remote test methods apply when it is possible to make use of some functions of the System Under Test (SUT) to control the IUT during testing, instead of using a specific UT. For example, the Logical Link Control protocol provides the UT functionality for Media Access Control protocol testing. As to MIB, such functions are always supported in the agent since managed objects have to be controlled by management applications of the agent. Furthermore, although some implementation may provide an explicit interface between an UT and a managed object, ASPs can not be standardised since such interface is dependent on the implementation. The remote test methods are best suited to the conformance test for MIB from these points.

Test purpose

We can apply the framework of the conformance test for protocol entity to that for MIB since both protocol entities and managed objects have a static aspect and a dynamic aspect. The test purpose, however, can not be applied as it is.

For example, a preamble is often included in the scenario for protocol entity in order to start a test from a specified state. The preamble for MIB may be meaningless since the model of MIB is not a finite state machine and the attribute value which indicates some states may be changed without any interactions from a manager. Another example is that the human operation to actual resources should be included in test cases of the behaviour test for MIB if its behaviour is controlled by man-machine interaction. The behaviour of protocol entity is generally controllable from the LT by communicating with the protocol entity to be tested and , if necessary, using TM-PDUs.

Scope of testing

It is difficult to conduct the thorough conformance test even for protocol entity whose models are finite state machines since test cases become huge. In case of MIB, the combination of attribute values would be almost infinite and the environmental and time-dependent conditions to actual resources would make the matter worse. As the practical solution, the conformance test for MIB should be conducted by creating the test scenario for the capability test to all test cases and the one for the behaviour test limited to the test cases actually used. For example, we can create a syntactically invalid test event with an attribute set to syntactically invalid value such as wrong tagging for ASN.1 encoding. For the semantically invalid test event, we can set the attribute to the value not permitted in 'property-list' or ASN.1 subtype definition.

5.2 Operability

We applied the MIB tester to the conformance test of an agent which support MIB for managed objects defined in accordance with the ISO/CCITT and Internet Management Coexistence (IIMC) [NMF26, 1993]. The MIB includes 102 managed object classes and 205 managed object instances. The MIB tester is implemented on a PC, DECpc XL560 (CPU: Pentium 60MHz) and is connected with the agents via Ethernet. The executable software size of the MIB tester is approximately 1.3 MB.

All the functions described in 4.2 was correctly performed. The scenario for the capability test was automatically generated for each instance of the MIB. For example, when an 'interface' managed object instance was specified for scenario generation and eight 'ifEntry' managed object instances were contained in it, the MIB tester generated eighteen M-GET operations, (ten for acquisition of the naming tree and eight for confirmation of M-SET operations), and eight M-SET operations to these instances, since an 'ifEntry' managed object class has one writable attribute called 'ifAdminStatus'. As a result, we were able to verify the dynamic conformance of this managed object instance.

It would be convenient to analyse the causality relationship among a sequence of CMIS primitives since an attribute value reflects the status change of an actual resource and an operation for a managed object sometimes causes future notifications. For example, a 'perceivedSeverity' attribute value varies according to the severity of an alarm and an attribute value change notification is caused by a previously issued M-SET operation. We will support this function in the future extension.

5.3 Automatic scenario generation

The number of instances in MIB of actual agents is huge in many cases. One example is managed objects which hold performance attributes for each termination point of the transmission equipment. Some Synchronous Digital Hierarchy (SDH) transmission equipment supports approximately sixty thousand managed object instances including performance related managed object instances. It spends so much time to create the scenario for all managed object instances manually that the automatic scenario generation function must be necessary.

The proposed scenario generation rules does not address the behaviour test. It is possible, however, to generate syntactically invalid test events to some extent by using different GDMO definitions from these for the agent. The method also does not cover the relationship among managed objects. For example, there are some cases that an action for a managed object affect the behaviour of other managed objects. We could generate such test cases from the formal description of BEHAVIOUR templates [X722/PDAM3, 1995] being now studied in ITU-T.

6 CONCLUSIONS

This paper proposed a conformance test method for MIB in an agent of OSI management and described an implementation of the MIB tester according to the test method. The proposed test method was based on and an extension of the existing conformance testing method for protocol entity by taking the differences in modelling between protocol entities and managed objects into account. The remote test methods were adopted as the test configuration of the conformance test for MIB. We defined the test purposes for the basic interconnection test, the capability test and the behaviour test. As to the scope of testing, we showed the practical solution by conducting the capability test to all test cases and the behaviour test limited to the test cases actually used. We have implemented the MIB tester so that test scenarios for capability tests are automatically generated. The MIB tester is flexible to GDMO definitions so as to be applicable to various kinds of agents. We evaluated the proposed test method and demonstrated the effectiveness of the implemented MIB tester through the application to the conformance test for the actual agents.

ACKNOWLEDGEMENT

The authors wish to express sincere thanks to Prof. Yoshiyori URANO of WASEDA University for his guidance and kind suggestions.

REFERENCES

A.Idoue, T.kato, K.Suzuki and Y.Urano (1990), Design and Implementation of OSI Communication Board for Personal Computers and Workstations, Proc. of Eleventh International Conference on Computer Communication

ITU-T Rec. M.3010 (1992), Principles for a Telecommunications Management Network

ITU-T Rec. X.290 (1992), OSI Conformance Testing Methodology and Framework for Protocol Recommendations for CCITT Applications - General Concepts

ITU-T Rec. X.721 (1992), IT - OSI - SMI: Definition of Management Information

ITU-T Rec. X.722/PDAM3 (1995), IT - OSI - SM: Guidelines for the Definition of Managed Objects - Amendment 3 (Use of Z for Managed Object Behaviour)

ITU-T Rec. X.724 (1994), IT - OSI - SMI: Requirements and guidelines for implementation conformance statement proformas associated with OSI management

J.M. Serre. , P. Lewis. and K. Rosenfeld. (1993), Implementing OSI-Based Interfaces for Network Management, IEEE Communications Magazine, Vol.31 No.5

Network Management Forum, OMNIpoint, Forum 26 (1993), Translation of Internet MIBs to ISO/CCITT GDMO MIBs

O. Ödling and S. Wallin (1994), Building MIB Applications, Proc. Of IEEE 1994 Network Operations and Management Symposium

BIOGRAPHY

Keizo Sugiyama is a research engineer of Network Management System Laboratory, KDD R&D Laboratories. Since joining the Labs. in 1987, he has been engaged in the researches on implementation of OSI Protocol, EDI(Electronic Data Interchange) translator and network management systems. He received the B.E. and M.E. from Kyoto University in 1985 and 1987 respectively.

Hiroki Horiuchi is a senior research engineer of Network Management System Laboratory, KDD R&D Laboratories. Since joining the Labs. in 1985, he has been engaged in the researches on formal description techniques for communication protocol, implementation of OSI Protocol and network management systems. He received the B.E. and M.E. from Nagoya University in 1983 and 1985 respectively.

Dr. Sadao Obana is a senior manager of Network Management System Laboratory, KDD R&D Laboratories. Since joining the Labs. in 1978, he has been engaged in the researches in the field of computer communication, database and network management systems. He received the B.E., M.E. and Dr. of Eng. Degree of electrical engineering from Keio University in 1976, 1978 and 1993 respectively.

Dr. Kenji Suzuki is a senior project manager of R&D planning group in KDD R&D Laboratories. Since joining the Labs. in 1976, he worked in the field of computer communication. He received the B.S., M.E. and Dr. of Eng. Degree of electrical engineering from Waseda University, Tokyo, Japan in 1969, 1972 and 1976 respectively. From 1969 to 1970, he was with Philips International Inst. of Technological Studies, Eindhoven, the Netherlands as an invited student. Since 1993, he has been a Guest Professor of Graduate School of Information Systems, in the University of Electro-Communications. He is a member of IEEE.