分散オブジェクトを用いたネットワーク管理における TMN/SNMP 収容方式

堀内浩規† 吉原貴仁†
小花貞夫† 鈴木健二†

ネットワーク管理システムの構築への分散オブジェクト技術の適用が注目されており，今後の普及が予想される．ここでは，分散オブジェクト環境へTMN（電気通信管理網）やSNMP（Simple Network Management Protocol）に準拠した既存装置の収容が必須となる．本論文では，分散オブジェクトの業界標準であるCORBA（Common Object Request Broker Architecture）環境にTMN/SNMP装置を収容する方式を提案する．提案方式は，X/OpenとTeleManagement ForumによるJIDM（Joint Inter Domain Management）が規定した，TMNやSNMPの管理情報定義とCORBAのIDL（インタフェース定義言語）定義との対応付けの規定をベースとして，そのIDL定義を拡張し，また，新たにIDL操作と管理操作の対応付け，オブジェクトの名前の対応付けを規定した．提案方式を実証するため，SNMP装置を収容するCORBA/SNMPゲートウェイ，ならびに，任意の管理情報定義に対応可能とするためのCORBA/SNMPゲートウェイプログラムジェネレータを実装した．さらに，生成したCORBA/SNMPゲートウェイプログラムを用いた処理時間等の評価を通して提案方式の有効性を示した．

Accomodation Method of TMN/SNMP in Network Management System Based on Distributed Object Technology

Hiroki Horiuchi,† Kiyohito Yoshihara,† Sadao Obana† and Kenji Suzuki†

This paper proposes accomodation method of TMN (Telecommunications Management Network)-based and SNMP (Simple Network Management Protocol)-based equipments in network management systems based on distributed object technology. Although JIDM (Joint Inter-Domain Management) group by X/Open and TeleManagement Forum specifies translation algorithms from TMN and SNMP MIB (Management Information Base) definition to CORBA (Common Object Request Broker Architecture) IDL (Interface Definition Language) definition, the IDL definition is not sufficient for the accomodation. Therefore, the proposed method extends the IDL definition for efficient operations and newly defines the mapping rules among management operations and IDL operations, based on JIDM. Furthermore, we implement CORBA/SNMP gateway based on the proposed method and we show the effectiveness of the method through the implementaion and evaluation.

1. はじめに

ネットワーク管理システムの構築への分散オブジェクト技術の適用が注目されており，今後の普及が予想される．ここでは，分散オブジェクト環境へTMN（電気通信管理網）[1]やSNMP（Simple Network Management Protocol）[2]に準拠した既存装置を収容することが必須となる．

このため，X/OpenとTeleManagement Forum（旧Network Management Forum）によるJIDM（Joint Inter Domain Management）は，TMNとSNMPの管理情報定義を分散オブジェクト環境の業界標準であるCORBA（Common Object Request Broker Architecture）[3]のIDL（インタフェース定義言語）定義への対応付けを規定している[4]．しかしながら，TMN/SNMP装置のCORBAへの収容には，この規定だけでは十分でない[5),6]．また，TMNとCORBAの相互接続に関した提案[7),8]もあるが，枠組みのみで具体的な実現方法が示されなかったり[7]，TMNの実装を容易とすることを目的として，TMN環境にCORBAを収容していたりして[8]，CORBAへのTMN/SNMP

† 株式会社KDD研究所
KDD R&D Laboratories

装置の収容には適用できない．

本論文では，JIDM の規定をベースにその IDL 定義の拡張や新たな IDL 操作と管理操作の対応付け等を規定した，CORBA への TMN/SNMP 装置の収容方式を提案する．さらに，提案方式に基づき，SNMP 装置を収容する CORBA/SNMP ゲートウェイを実装し，処理時間等の観点から評価を行う．

以下，2 章で CORBA と JIDM の概要を述べる．3 章で収容方式の提案を行い，4 章で CORBA/SNMP ゲートウェイの実装について述べる．5 章では性能評価等を通して，提案方式の有効性を示す．

2. CORBA と JIDM 仕様の概要

2.1 CORBA の概要

CORBA のアプリケーションはクライアントとサーバから構成され，サーバはオブジェクト（以下，CORBA オブジェクトと呼ぶ）を持ち，クライアントはサーバ上の CORBA オブジェクトが提供するサービスを利用する．CORBA オブジェクトが提供するサービスのインタフェースは，IDL を用いて，CORBA オブジェクトの属性と操作（以下，IDL 属性と IDL 操作と呼ぶ），データ型等が定義される．IDL 属性は，クライアントからアクセス可能な CORBA オブジェクトのデータであり，また，IDL 操作はクライアントが CORBA オブジェクトに依頼する処理である．図 1 (a) に IDL 定義の例，図 1 (b) にインタフェースと CORBA オブジェクトの関係を示す．

クライアントが IDL 属性や IDL 操作を呼び出す際には，対象となる CORBA オブジェクトをオブジェクトリファレンスにより指定する．オブジェクトリファレンスは，サーバで CORBA オブジェクトが生成される際に付与され，CORBA オブジェクトの名前とオブジェクトリファレンスの組を管理する名前管理サーバに登録する．クライアントは，名前管理サーバを通じて，名前からオブジェクトリファレンスを取得する．CORBA オブジェクトの名前は Naming Graph（NG）と呼ばれる木構造で管理され，節を示す Naming Context（NC）と葉を示す Name（NA）からなる．

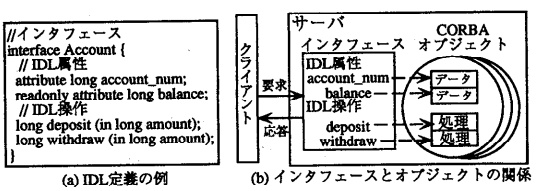


図 1 CORBA の概要
Fig. 1 Overview of CORBA.

2.2 JIDM の IDL 定義への対応付けの概要[4)]

TMN の管理情報定義（GDMO 定義）の IDL 定義への対応付けでは，(1) 管理オブジェクト（MO）クラスをインタフェースに，(2) 属性，アクションおよび通知を，それぞれ，IDL 操作に対応付ける．

SNMP の管理情報定義の IDL 定義への対応付けでは，(1) グループおよびテーブルエントリをインタフェースに，(2) オブジェクトタイプを IDL 属性に対応付ける．

3. CORBA への TMN/SNMP 収容方式の提案

3.1 収容の形態と課題

3.1.1 収容形態

CORBA 環境に TMN/SNMP 装置を収容するには，TMN の MO クラスや SNMP のグループやテーブルエントリのインスタンスを CORBA オブジェクトに対応付けて，図 2 に示すように，CORBA の IDL 操作と TMN/SNMP の管理操作や通知を相互に変換するゲートウェイが必要となる．言い換えれば，ゲートウェイは，CORBA クライアントに対しては CORBA サーバとして，TMN/SNMP 装置に対してはマネージャとして振る舞う．また，ゲートウェイは CORBA オブジェクトの名前とオブジェクトリファレンスの組を名前管理サーバに登録する．

3.1.2 収容の課題

2.2 節で述べた JIDM の管理情報と IDL 定義との対応付け規定（以下，JIDM の規定と呼ぶ）に基づいて，ゲートウェイを実現するためには，以下の課題がある．

(1) IDL 操作と管理操作の対応付けの規定

ゲートウェイの実現に必要となる IDL 操作と管理操作や通知の対応付けは JIDM の規定では定義していないため，新たに規定する必要がある．

(2) IDL 属性の取得/設定時における非効率性の改善

TMN では，単一の管理操作で複数の属性や MO イ

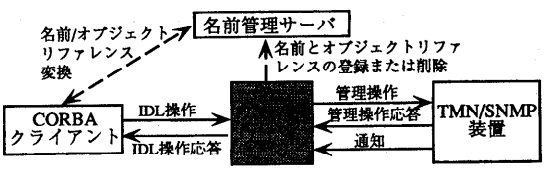


図 2 TMN/SNMP 装置の CORBA 環境への収容形態
Fig. 2 Accomodation of TMN/SNMP equipments in CORBA environment.

```
interface System : SNMPMgmt :: SmiEntry {// SNMPの場合
  readonly  attribute  DisplayStringType sysDescr;
  attribute  DisplayStringType sysName; ..... // 属性定義
  enum SystemAttribute { sysDescr_type, sysName_type, ..... };
  typedef sequence<SystemAttribute> SystemAttributeList;
  struct SystemValue { systemAttribute attributeType;
                       Value attributeValue; };
  typedef sequence<SystemValue> SystemValueList;
  // 複数IDL属性取得のためのIDL操作
  void system_get_attribute_list ( in SytemAttributeList attributeList,
                                   out SytemValueList valueList ) ;
  // 複数IDL属性設定のためのIDL操作
  void system_set_attribute_list ( in SytemAttributeList attributeList,
                                   in SytemValueList valueList ) ;
```

(a) 同一CORBAオブジェクトにおけるIDL操作定義の拡張

```
interface mo_scope_filter { // TMNの複数MOインスタンスへの処理
  // 型定義省略
  // 複数IDL属性取得のためのIDL操作
  void get_scope_filter ( in ObjectReference base, in ScopeType scope
       in FilterType filter, out GetResults selectedMOs ) ;
  // 複数IDL属性設定のためのIDL操作
  void set_scope_filter ( in ObjectReference base, in ScopeType scope
       in FilterType filter, in ValueList valueList ) ;  }
interface get_next { // SNMPのGetNextに対応
  // 型定義省略
  void get_next ( in ObjectTypeList typeList, out ValueList valueList ) ;}
```

(b) 複数CORBAオブジェクトにまたがるインタフェース定義の拡張

図3 IDL 属性の取得/設定のための拡張
Fig. 3 Extension for retrieval and update of IDL attribute.

ンスタンスを操作対象とすることができる．しかしながら，JIDM の規定では，TMN の 1 つの属性を 1 つの IDL 操作に対応付けるため，CORBA クライアントからの複数 IDL 属性の取得/設定時には，属性の個数分だけ IDL 操作と M-GET 等の管理操作が発行され効率的でない問題点がある．また，SNMP の場合にも，オブジェクトタイプを IDL 属性に対応付けるため，複数 IDL 属性の取得/設定時にオブジェクトの個数分だけ IDL 操作と GetRequest 等の管理操作が発行されて効率的でない同様な問題点がある．

(3) CORBA オブジェクト生成/削除のためのインタフェースの規定

TMN では，MO インスタンスの生成/削除のための管理操作が用意されている．しかしながら，JIDM の規定では，対応する CORBA オブジェクトの生成/削除のためのインタフェースが規定されていないため対応付けられない問題点がある．

(4) CORBA オジェクトの名前対応付けの規定

CORBA オブジェクトの名前は 2.1 節で述べた木構造で管理され，TMN の MO インスタンスと SNMP のオブジェクトの識別名の体系とは異なるため，対応付けを規定する必要がある．

上記課題 (2) と (3) で示した問題点を解決するため，本方式では JIDM の規定における IDL 定義の対応付けを拡張することとし，その解決方法を 3.2 節で述べる．課題 (1) に対しては拡張する IDL 定義を含めた管理操作への対応付けを 3.3 節で規定し，課題 (4) の名前の対応付けについては 3.4 節で規定する．

3.2 JIDM の IDL 定義への対応付けの拡張

IDL 属性の取得/設定における管理操作変換の効率化と CORBA オブジェクトの生成/削除を可能とするため，以下のように JIDM の規定を拡張する．

3.2.1 IDL 属性の取得/設定のための拡張

IDL 属性の取得/設定時の非効率性の問題点を解決するために，取得/設定対象の複数 IDL 属性が同一 CORBA オブジェクトにある場合と，複数 CORBA オブジェクトにまたがる場合とに分けて，以下の IDL 定義の拡張を行う．

(1) 同一 CORBA オブジェクトの場合

JIDM に従ったインタフェースに，複数の IDL 属性の取得/設定を可能とする新たな IDL 操作 (get_attribute_list, set_attribute_list) を追加定義する．図 **3** (a) に，SNMP の MIBII[9] におけるグループ system に対応したインタフェース定義の場合の例を示す．該当する IDL 操作 system_get_attribute_list では，取得対象の IDL 属性を示す入力引数はオブジェクト sysDescr や sysName 等を含む列挙型のシーケンス型を持ち，取得結果の IDL 属性値を示す出力引数はとりうる値のシーケンス型を持つ．

(2) 複数 CORBA オブジェクトの場合

TMN の複数の MO インスタンスにまたがる M-GET 等の管理操作に対応付け可能なインタフェース (mo_scope_filter) を新たに定義する．SNMP の場合も，複数のグループにまたがる GetNextRequest

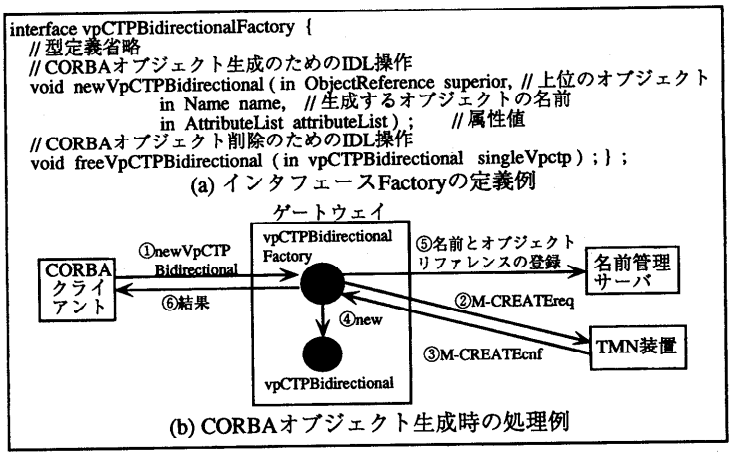


図4 CORBA オブジェクトの生成と削除
Fig. 4 Creation and deletion of CORBA objects.

管理操作のためのインタフェース（get_next）を新たに定義する．図3(b)に，TMNとSNMPのそれぞれの場合のインタフェース定義を示す．TMNにおけるインタフェース mo_scope_filter の IDL 操作（get_scope_filter）では，入力引数として基準のオブジェクトのオブジェクトリファレンス，Naming Graphの対象範囲，選択条件を持ち，出力引数は取得結果のCORBA オブジェクトの属性値を持つ．

3.2.2 CORBA オブジェクト生成/削除の拡張

CORBA オブジェクトの生成/削除と TMN 装置の MO インスタンスの生成/削除の管理操作を対応付けるため，生成/削除可能な MO クラスごとにインタフェースを新たに定義する．このインタフェースは生成と削除の IDL 操作を持ち，このインタフェースに従う CORBA オブジェクトを Factory と呼ぶ．図 **4**(a) に，ATM（非同期転送網）交換機における VP（Virtual Path）の終端点を示す MO クラス（vpCTPBidirectional）[10]のインタフェースの例を示す．生成のための IDL 操作（newVpCTPBidirectional）には，生成オブジェクトの上位のオブジェクトのオブジェクトリファレンス，生成オブジェクトの名前，IDL 属性の値を入力引数として持つ．また，SNMP の場合にも，Factory を用い，SNMP 装置のインスタンスの存在に対応させてCORBA オブジェクトの生成/削除を可能とする．

3.3 IDL 操作と管理操作の対応付け

JIDM の規定と 3.2 節で拡張した IDL 操作に対する管理操作と通知の対応付けを**表 1** に示す．ここでは，対象となる CORBA オブジェクトのインタフェースと IDL 操作に対して，対応付ける TMN または SNMP の管理操作や通知を示す．

（1） 属性の取得/設定とアクション

3.2.1 項で述べた MO クラスのインタフェースごとの IDL 操作 get_attribute_list/set_attribute_list は，TMN の attribute_list パラメータに複数の属性を指定した M-GET/M-SET 管理操作，SNMP の VarBindList パラメータに複数のオブジェクトインスタンスを指定した GetRequest/SetRequest 管理操作に対応付ける（表 1 ①，②，⑦，⑧）．また，インタフェース mo_scope_filter の IDL 操作を，TMN の scope パラメータと filter パラメータを持つ M-GET，M-SET および M-ACTION の管理操作に（表1 ①，②，③），インタフェース get_next の IDL 操作は SNMP の GetNextRequest 管理操作に（表 1 ⑦）対応付ける．

（2） MO インスタンスの生成/削除

3.2.2 項で述べた図 4(a) の Factory のインタフェース定義の拡張に従って，TMN における MO インスタンスの生成と CORBA オブジェクト生成時の対応付け例を図 4(b) に示す．ここでは，生成の IDL 操作に対して，M-CREATE 管理操作により TMN 装置の MO インスタンスの生成，および，MO インスタンスに対応する CORBA オブジェクトの生成を行う（表 1 ④）．一方，CORBA オブジェクトを削除する IDL 操作の場合には，M-DELETE 管理操作により TMN 装置の MO インスタンスの削除，および，対応する CORBA オブジェクトの削除を行う（表 1 ⑤）．また，ゲートウェイは，TMN 装置からの MO インスタンス生成/削除の通知受信時にも Factory に IDL 操作を発行する．

SNMP の場合には，ゲートウェイは障害の検出や動的なインスタンスの変化を検出するため，SNMP 装

表1 IDL 操作と管理操作や通知の対応付け
Table 1 Mapping between IDL operations and management operation.

		管理操作と通知	インタフェース名	IDL操作名
TMN	①属性取得	M-GET - 単一属性 - 複数属性 - scope/filter指定	MOCのIF MOCのIF mo_scope_filter{} *	<属性ラベル>Get() <MOCラベル>_get_attribute_list() * get_scope_filter() *
	②属性設定	M-SET - 単一属性 - 複数属性 - scope/filter指定	MOCのIF MOCのIF mo_scope_filter{} *	<属性ラベル>Set() <MOCラベル>_set_attribute_list() * set_scope_filter() *
	③アクション	M-ACTION - 単一MOI - scope/filter指定	MOCのIF mo_scope_filter{} *	<アクションラベル>() action_scope_filter() *
	④MOI生成	M-CREATE	MOCのFactory *	new<MOCラベル>() *
	⑤MOI削除	M-DELETE	MOCのFactory *	free<MOCラベル>() *
	⑥通知	M-EVENT-REPORT	Notification{}	<通知ラベル>()
SNMP	⑦オブジェクト取得	GetRequest - 単一OBJT - 複数OBJT GetNextRequest	グループ/エントリのIF グループ/エントリのIF get_next{} *	<OBJTラベル>() <グループラベル>_get_attribute_list() * get_next() *
	⑧オブジェクト設定	SetRequest - 単一OBJT - 複数OBJT	グループ/エントリのIF グループ/エントリのIF	<OBJTラベル>() <グループラベル>_set_attribute_list() *
	⑨通知	Trap	SnmpNotifications{}	snmpTrap()

(注) IF：インタフェース．MOC：MOクラス．MOI：MOインスタンス．OBJT：オブジェクトタイプ．
グループ/エントリ：グループまたはテーブルエントリ．*：JIDMのIDL定義を拡張して追加した定義

置へ GetNextRequest 管理操作を順次発行するポーリングを行う．新たなオブジェクトを検出した場合，オブジェクトを生成する IDL 操作を Factory に発行し，CORBA オブジェクトを生成する．一方，オブジェクトの削除を検出した場合，対応する CORBA オブジェクトを削除する IDL 操作を Factory に発行する．

上記 TMN と SNMP のいずれの場合にも，ゲートウェイは，CORBA オブジェクトの生成の場合には名前管理サーバに当該 CORBA オブジェクト（名前とオブジェクトリファレンス）の登録を，削除の場合には名前管理サーバに当該 CORBA オブジェクトの登録抹消をあわせて行う．

（3） 通知

TMN の M-EVENT-REPORT 受信時には，パラメータ eventType に基づいて IDL 操作を決定し，MO インスタンスの識別名から CORBA オブジェクトの名前への変換等のパラメータの変換後，IDL 操作を CORBA クライアントへ発行する（表1 ⑥）．

また，SNMP の Trap 受信時には，IP アドレスや generic-trap 番号等のパラメータの値から IDL 操作 snmpTrap のパラメータを設定し，CORBA クライアントへ発行する（表1 ⑨）．

3.4 CORBA オジェクトの名前対応付け

TMN の MO インスタンスと SNMP のオブジェクトの識別名を，それぞれ，以下の規則により CORBA オブジェクトの名前に対応付ける．

（1） TMN の場合

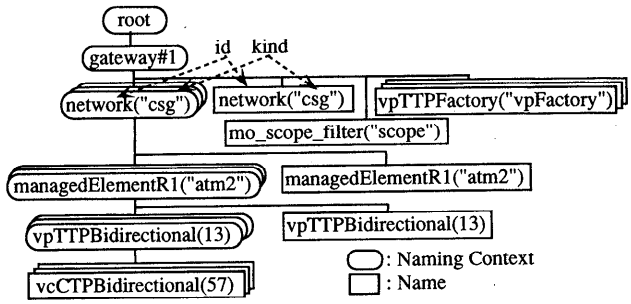
root の下に (1) ゲートウェイ Naming Context（NC），(2) 名前木の葉以外の MO インスタンスの相対識別名ごとに対応する NC，(3) 上記 (2) の各 NC に対して同じレベルに対応する Name（NA），(4) 名前木の葉の MO インスタンスの相対識別名に対応する NA から構成する．この際，NC と NA の id には MO クラス名を，kind には属性値を設定する．また，ゲートウェイの NC の直下に，3.2 節で述べた mo_scope_filter の NA，ならびに，生成/削除が可能な MO クラスの個数分の Factory を付加する．

（2） SNMP の場合

root の下に (1) ゲートウェイ，(2) SNMP エージェント，(3) グループ名に対応する NC，(4) 上記 (3) の直下でグループに対して1つだけ規定されるオブジェクトタイプを示す NA，(5) 上記 (3) の直下にテーブルエントリに対応して複数存在する NA から構成する．また，ゲートウェイの NC の直下に，3.2 節で述べた get_next と Factory の NA を付加する．

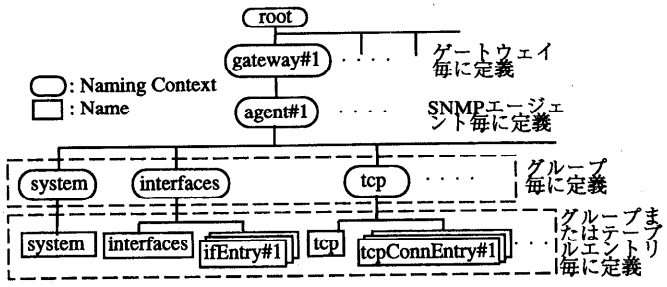
上記の対応付けでは，TMN の場合には1個の MO インスタンスに対して NC と NA の双方を割り当てるが，SNMP の場合にはオブジェクトに対して単一の NA または NC を割り当てる．これは，TMN の場合は識別名の木構造の葉と節の両方で管理情報を保持するが，SNMP では，CORBA と同様に，葉のみに管理情報の値を持つためである．

TMN と SNMP のそれぞれに対する Naming Graph（NG）の構成例の一部を**図5** (a)，(b) に示す．図5 (a)



注：対応させたMOインスタンスvcCTPBidirectionalの識別名は，{network, "csg"}/{managedElementR1, "atm2"}/{vpTTPBidirectional, 13}/{vcCTPBidirectional, 57}とする。

(a) TMNにおけるNamig Graph



(b) SNMPにおけるNamig Graph

図5 Namig Graphの構成例

Fig. 5 Example of Namig Graph.

はTMNにおけるATMのNGを示し，4つのMOクラスnetwork，managedElementR1，vpTTPBidirectional，vcCTPBidirectionalからなる識別名を対応付けた．図5(b)はSNMPにおけるMIBII[9]のNGを示し，たとえば，interfacesグループのテーブルエントリの1つの名前は/gateway#1/agent#1/interfaces/ifEntry#1/となる．

4. CORBA/SNMPゲートウェイの実装

3章で提案した収容方式を実証するため，SNMP装置を収容するCORBA/SNMPゲートウェイ（以下，CORBA/SNMP GWと呼ぶ）を実装した．CORBAはIONA社Orbix，計算機はSUN，プログラム開発言語はC++を使用した．また，SNMPの任意の管理情報定義に対応可能とするため，SNMPの管理情報定義からCORBA/SNMP GWプログラムを自動生成させた．

4.1 システム構成

CORBA/SNMP GWは，クライアントからのIDL操作とSNMP装置からの通知が非同期に発行されるため，(1)管理操作変換サーバと，(2)通知変換サーバの2つから構成し，それぞれ別プロセスとして実現した（**図6**）．両サーバは，SNMPの管理操作や通知とIDL操作との変換を行う管理操作変換部あるいは通知変換部に加えて，IDL操作の符号化/復号を行うスタブ，SNMP PDUの符号化/復号を行うSNMPライブラリから構成される．管理操作変換部では，オブジェクトタイプごとに，SNMPの管理操作とIDL操作との変換を行うC++の関数（メソッド）を用意する．

管理操作変換サーバはCORBA/SNMP GW起動時およびCORBAオブジェクトが生成されるとき，CORBAオブジェクトの名前とオブジェクトリファレンスの組を名前管理サーバに登録する．また，名前管理サーバのあるホスト名や，SNMP装置のIPアドレス，SNMPオブジェクト識別子，コミュニティ等のSNMP装置の情報，ならびにポーリングを行う間隔を初期化ファイルに設定し，起動時に読み込んで変換に使用する．

4.2 CORBA/SNMP GWの処理

CORBA/SNMP GWは，基本的には3.3節のIDL操作と管理操作の対応付けに従った処理を行う．このうち，3.3節(1)の属性値取得の処理の詳細を以下に述べる．

CORBAクライアントはオブジェクトリファレンスにより，SNMPのグループまたはテーブルエントリに対応するCORBAオブジェクトにバインドし，オブ

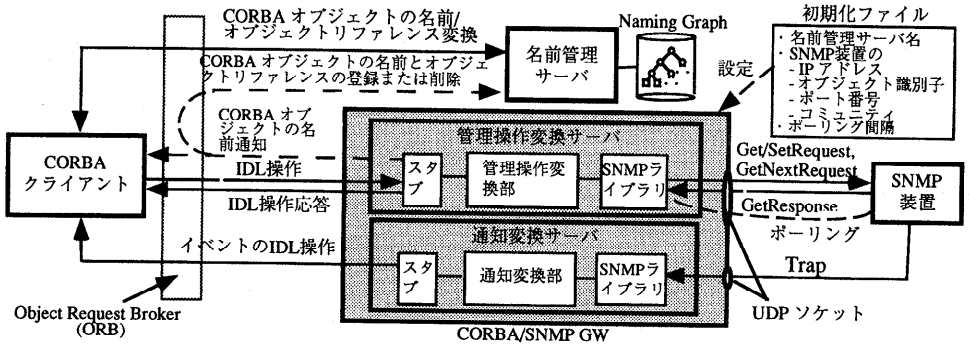


図 6 システム構成
Fig. 6 System configuration.

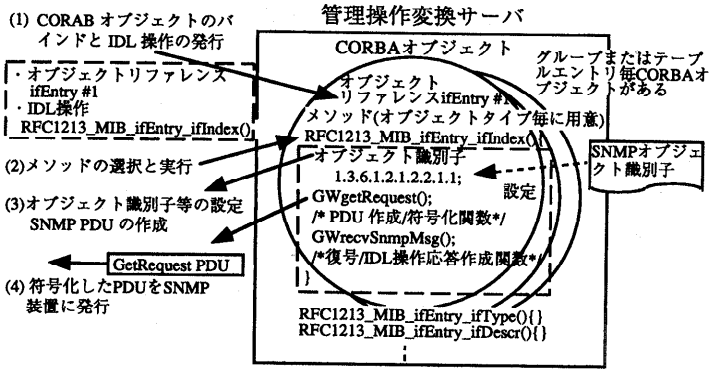


図 7 IDL 操作の処理
Fig. 7 Processing of IDL operations.

ジェクトタイプの値取得または設定のための IDL 操作を発行する（図 7(1)）．CORBA オブジェクトは，IDL 操作の名前をキーとして，オブジェクトタイプごとに用意された関数（メソッド）を選択し，実行する（図 7(2)）．

このメソッドはすべての CORBA オブジェクトに共通の SNMP ライブラリ（たとえば，GWgetRequest()）を用いて PDU を作成し，オブジェクト識別子等のパラメータを設定し（図 7(3)），次いで PDU を符号化して SNMP 装置に発行する（図 7(4)）．

GetRequest 管理操作に対する GetResponse 管理操作応答を受信した場合，取得値を SNMP PDU 作成ライブラリで復号し，IDL 操作応答に詰め替えて CORBA クライアントに返す．

GetNextRequest 管理操作に対する GetResponse 管理操作応答を受信した場合，取得値の SNMP オブジェクトを特定し，対応する CORBA オブジェクトの名前を CORBA クライアントに応答する必要がある．このため，GetResponse 管理操作応答の VarBindList パラメータに含まれる SNMP オブジェクト識別子から，対応するオブジェクトリファレンスを抽出し，名前管理サーバで CORBA オブジェクトの名前に変換させ，取得値とあわせて CORBA クライアントへ応答する．

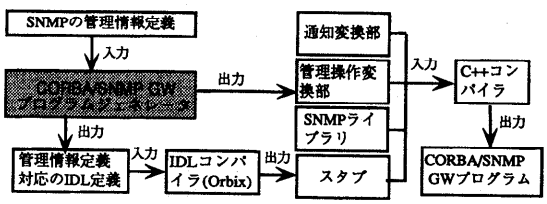


図 8 プログラムの自動生成
Fig. 8 Automatic generation of program.

4.3 プログラムの自動生成

SNMP の任意の管理情報定義に対応可能とするため，CORBA/SNMP GW プログラムジェネレータ（以下，プログラムジェネレータと呼ぶ）を実装した．図 8 に CORBA/SNMP GW プログラムの自動生成手順を示す．プログラムジェネレータは，SNMP の管理情報定義から対応する IDL 定義と管理操作変換部のプログラムを生成する．また，生成した IDL 定義

からIDLコンパイラを用いてスタブを生成する．通知変換部とSNMPライブラリは，SNMPの管理情報定義に依存しないためあらかじめ用意した．これらをコンパイル・リンクし，CORBA/SNMP GWプログラムを生成する．

5. 評価と考察

5.1 性能評価

MIB IIの管理情報定義に対して，自動生成したCORBA/SNMP GWプログラムを使用し，応答時間を計測した．**図9**に試験構成を，**表2**に応答時間を示す．表2の測定項目のうち，JIDM規定を拡張したIDL操作を使用したものは表2(b)，(d)，(e)である．なお，応答時間は図9(1)〜(10)の処理時間の合計とした．

表2のすべての測定項目の応答時間は13〜30 ms程度であり，このうちゲートウェイの処理時間は6〜17 ms程度でそれほど大きなオーバヘッドでなく，実用的な性能を達成していると考えられる．

5.2 IDL定義の拡張の効果

複数のIDL属性の取得（表2(b)）および設定（表2(d)）の応答時間は，それぞれ29.8 ms，24.7 msとなる．これを単一IDL属性の取得および設定を順次行った場合には，それぞれ51.8 ms，34.4 msとなり，IDL定義の拡張によるIDL操作と管理操作の個数削減の効果が大きいことが分かる．SNMPでは，ATM（非同期転送網）におけるVP（Virtual Path）の設定等のようにGetRequestやSetRequest管理操作に複数のオブジェクトを設定する場合は多く，このような観点からも本拡張は有効と考えられる．さらに，表2(e)は複数オブジェクトにわたる属性値の取得となるが，同一オブジェクトの場合の応答時間と同等に実現できた．

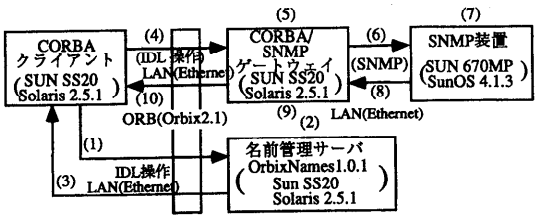


図9 試験構成と観測点

Fig. 9 Test configuration and its measurement points.

表2 プログラムの応答時間

Table 2 Response time in CORBA/SNMP GW.

測定項目	応答時間(ms)
(a) sysUpTime(TimeTicks型)の1個の属性値取得	17.3 (6.2)
(b) system_get_attribute_list()によるsysUpTime, sysObject, sysDescの3個の属性値取得	29.8 (16.8)
(c) sysLocation(DisplayString型) の1個の属性値設定	16.7 (6.2)
(d) system_set_attribute_list()によるsysLocation (DisplayString型)等の2個の属性値設定	24.7 (11.4)
(e) get_next()によるsysNameの次のifNumberの属性値取得	20.8 (6.2)
(f) snmpTrap()による Trap(coldStart)の受信	13.0 (9.5)

(注1) 応答時間の括弧内の数値はゲートウェイの処理時間(図9の(5)と(9)の和)を示す．
(注2) 各数値は100回の試行の平均値．

5.3 プログラム規模

CORBA/SNMP GWプログラムジェネレータが生成する管理操作変換部およびスタブのプログラム規模は管理情報定義に依存し，システムやTCP/IP等の管理を行うためによく使用されるMIB II（17個のグループまたはテーブルエントリを含む）の場合，それぞれ，16.1 Kstep，7.8 Kstepであり，コンパクトなプログラムが生成されている．また，管理情報定義に依存しないTrap変換部とSNMPライブラリは，それぞれ，0.2 Kstep，6.7 Kstepであった．

5.4 TMN装置を収容するゲートウェイとの変換処理の比較

4章で述べたCORBA/SNMP GW（以下，SNMP GWと呼ぶ）と，提案方式に従ったTMN装置を収容するゲートウェイ（以下，TMN GWと呼ぶ）の実装[6]，との違いを，管理操作変換サーバと通知変換サーバにおける処理，ならびに性能とプログラム規模の観点から考察する．

(1) 管理操作変換サーバの処理

TMNの管理操作の種類は，SNMPの管理操作より多いため，TMN GWでは，SNMP GWと同様な属性値取得/設定や通知の変換に加えて，MOインスタンスの生成/削除ならびにアクションの変換を行う必要がある．さらに，複数MOインスタンスにまたがるIDL操作受信時には，SNMP GWの場合には単一の管理操作応答（VarBindListを使用）を単一のIDL操作応答に変換したが，TMN GWの場合には複数の管理操作応答（Linked-Replyを使用）を単一のIDL操作応答にまとめる処理が必要となる．

SNMPの管理情報定義は，使用するASN.1のシンタックスは構造形を使用できない等の多くの制限があるとともに，オブジェクトの継承ができない等，GDMO定義と比べて単純な構造となる．このため，属性値取得時の管理操作応答の復号結果をIDL操作応答に詰め替える際等には，SNMP GWでは単純な1回の代入で済むが，TMN GWではASN.1の構造

形のネスト等に対処するため，C++におけるデータ型をたどって値を詰め替える処理が必要となる．また，TMN GW の場合には，継承の展開や属性のプロパティごとの IDL 操作の定義等により，変換処理を行う IDL 操作の数も増加する．

(2) 通知変換サーバの処理

TMN の通知は，SNMP と異なり，通知の種類によりパラメータや ASN.1 型が異なる．このため，通知変換部のプログラムを SNMP GW ではあらかじめ用意したが，TMN GW の場合には，GDMO 定義の通知の種類に合わせてプログラムジェネレータにより生成する必要がある．また，上記 (1) の管理操作変換サーバの処理と同様に，通知パラメータの IDL 操作への詰め替えは，SNMP GW と比べて複雑な処理となる．

TMN 装置では，自律的に MO インスタンスの生成/削除や属性値の変化が発生した場合に，通知 (objectCreation, attributeValueChange 等) を発行する．TMN GW の通知変換サーバはこれらの通知受信時に，CORBA オブジェクトの生成/削除や属性値設定の IDL 操作を発行する必要がある (ただし，管理操作を用いた生成/削除や属性値変更の場合は不要)．一方，SNMP 装置はこれらの通知を発行しないので，SNMP GW は SNMP 装置をポーリングしてインスタンスの生成/削除を検出し，CORBA オブジェクトの生成/削除を行った．

(3) 性能とプログラム規模

5.1 節と同一の試験構成で，TMN GW における処理時間を測定した．その結果，MO インスタンス system の 3 個の属性値取得 (supportedFeature 等) の場合には 43.9 ms，通知 (attributeValueChange) 受信の場合には 40.9 ms であった．SNMP GW と比較すると，対象とする MO インスタンスや属性の種類等によってばらつきはあるが，全体的に 1.5～4.0 倍程度の処理時間となった．プログラム規模は，ITU-T 勧告 X.721 の 4 個の MO クラス (system, log 等) の場合，管理操作変換部 29.9 Kstep，スタブ 26.1 Kstep，通知変換部 5.8 Kstep となり，SNMP GW より増大する．

以上，TMN GW は，SNMP GW と比較して管理操作変換や通知変換の処理が複雑になり，性能は低下する．しかしながら，処理時間は 40 ms 程度であり，TMN GW においても実用的な性能は達成している．

6. おわりに

本論文では，CORBA への TMN/SNMP 装置の収容方式を提案した．ここでは，JIDM が規定する TMN や SNMP の管理情報定義と IDL 定義との対応付けの規定をベースとして，その IDL 定義を拡張し，また，新たに IDL 操作と管理操作の対応付け，オブジェクトの名前の対応付けを規定した．提案方式を実証するため，SNMP 装置を収容する CORBA/SNMP ゲートウェイ，ならびに，任意の管理情報定義に対応可能とするためのプログラムジェネレータを実装した．さらに，生成した CORBA/SNMP ゲートウェイプログラムを用いた処理時間等の評価を通して提案方式の有効性を示した．

今後，ネットワークの大規模化とサービスの多様化により，ネットワーク管理とサービス管理との緊密な連携がますます重要となり，ネットワーク管理への分散オブジェクトをはじめとする分散処理技術の適用が進むことが予想される．TMN や SNMP に準拠した既存装置をシームレスに分散オブジェクト環境に収容可能とする提案方式は，この実現に有効な方式となる．

謝辞 日頃ご指導いただく株式会社 KDD 研究所村谷拓郎所長に感謝します．

参考文献

1) ITU-T Rec. M.3010: Principles for Telecomunications Management Network (1992).
2) Case, J., et al.: A Simple Network Management Protocol (SNMP), IETF, RFC1157 (1990).
3) Object Management Group: The Common Object Request Broker: Architecture and Specification Rev. 2.0 (1995).
4) X/Open Preliminary Specification: Inter Domain Management: Specification Translation (1996).
5) 堀内，吉原，小花：分散オブジェクトを用いたネットワーク管理における TMN/SNMP 収容方式の提案，情報処理学会マルチメディア通信と分散処理研究会，DPS-85-40 (1997).
6) 堀内，吉原，小花：CORBA/TMN ゲートウェイの実装と評価，第 56 回情報処理学会全国大会論文集，4H-07 (1998).
7) Lee, W.-C. and Mitchell, G.: A Framework for TMN-CORBA Interoperability, *Proc. IEEE 1988 Network Operations and Management Symposium*, pp.90–99 (1998).
8) Chadha, R. and Wuu, S.: Incorporating Manageability into Distributed Software, *Proc. 5th IFIP/IEEE International Symposium on Integrated Network Management*, pp.489–502 (1997).
9) McCloghrie, K., et al.: Management Infor-

mation Base for Network Management of TCP/IP-based internets: MIB-II, IETF, RFC 1213 (1991).
10) ATM Forum af-nm-0027: CMIP Specification for the M4 Interface (1995).

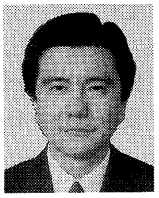
(平成10年5月11日受付)
(平成10年9月 7 日採録)

堀内 浩規（正会員）

昭和35年生．昭和58年名古屋大学工学部電気工学科卒業．昭和60年同大大学院情報工学専攻修士課程修了．同年国際電信電話（株）入社．現在，(株) KDD研究所ネットワークサービスグループ主任研究員．この間，ネットワークアーキテクチャ，OSIプロトコル実装方式，通信プロトコルの形式記述技法，ネットワーク管理の研究に従事．平成4年度電子情報通信学会学術奨励賞，平成8年度情報処理学会全国大会大会優秀賞を各受賞．電子情報通信学会会員．

吉原 貴仁（正会員）

昭和45年生．平成5年東京工業大学情報工学科卒業．平成7年同大大学院理工学研究科情報工学専攻修士課程修了．同年国際電信電話（株）入社．現在，(株) KDD研究所ネットワーク管理グループ主任．この間，ネットワーク管理，ネットワークアルゴリズム，分散処理の研究に従事．平成9年度情報処理学会全国大会大会優秀賞を受賞．電子情報通信学会会員．



小花 貞夫（正会員）

昭和28年生．昭和51年慶応義塾大学工学部電気工学科卒業．昭和53年同大大学院修士課程修了．同年国際電信電話（株）入社．現在，(株) KDD研究所ネットワーク管理グループリーダ．工学博士．この間，パケット交換方式，ネットワークアーキテクチャ，OSIプロトコル実装，データベース，ビデオテックス，分散処理，ネットワーク管理の研究に従事．電子情報通信学会会員．

鈴木 健二（正会員）

昭和20年生．昭和44年早稲田大学理工学部電気通信学科卒業．昭和44年から45年までオランダのフィリップス国際工科大学に招待留学．昭和51年早稲田大学大学院博士課程修了．同年国際電信電話（株）入社．現在，(株) KDD研究所代表取締役副所長．工学博士．この間，磁気記録，パケット交換方式，ネットワークアーキテクチャ，高速・分散処理の研究に従事．昭和62年度前島賞，平成4年度電子情報通信学会業績賞，平成7年度科学技術庁長官賞を各受賞．平成5年度より電気通信大学大学院情報システム学研究科客員教授．電子情報通信学会，IEEE各会員．