管理プロトコルへの適応性を有するネットワーク管理システム用フレームワーク構成法

浜田雅樹[†] 蔭山克禎[††] 藤崎智宏[†††]

多様な管理プロトコルに対して高い再利用性を実現するネットワーク管理システム（NMS）用アプリケーションフレームワークの構成法を提案する．ネットワーク機器は管理目的のアクセス手段として管理プロトコルを提供する．それらは機種ごとに多様であり，NMSの開発コストが高くなる問題がある．多様な管理プロトコルに対して高い再利用性を有するNMS用アプリケーションフレームワークを以下の方法で実現した．ソフトウェアが実行時に呼び出し先を判断する動的起動により，無修整で再利用できるコードを増やすとともに，レイヤ構造と継承により，修正が必要なコードを局所化して再利用性を高める．具体的には，(1) 管理機能のロジックを，ネットワーク機器の制御・情報処理と独立して実装するレイヤ構造を導入した，(2) ネットワーク機器の制御・情報処理にマルチプロトコルハンドリングメカニズムを実現した．これは，管理プロトコルを動的に選択する機構と，管理プロトコルを使った機器アクセス処理に分けて構成する．前者は，動的なオブジェクト呼び出しを，後者は継承を用いて実装することで，管理プロトコルに依存しないコードを増やし，かつ依存するコードを局所化させる効果を持つ．以上は実際の適応事例で高い再利用性を発揮した．

An NMS Application Framework Architecture Adaptable for Network Management Protocols

Masaki Hamada,[†] Katsuyoshi Kageyama[††] and Tomohiro Fujisaki[†††]

This paper proposes a method of implementing an NMS (Network Management System) application framework adaptable for diverse network management protocols. The high reusability of the application framework for diverse network management protocols is successfully realized by the effective combination of a layered architecture, inheritance and dynamic object invocation that separates the classes requiring modification from these that do not, increasing the ratio of the latter classes. To put it concretely, 1) the layered architecture makes management function implementation independent from network device query and control, and 2) a multiprotocol handling mechanism is developed for network device query and control. This mechanism consists of a network management protocol selection part implemented using the dynamic object invocation, and network device access parts implemented using inheritance. The former part increases the ration of program codes independent from management protocols, and the latter part localizes them. The application framework has been evolved through actual use in network management applications, and its effectiveness is discussed.

1. まえがき

コンピュータネットワークを管理するには，多数の分散配置されたネットワーク機器（ルータなど）を遠隔地から監視する必要性が生じ，ネットワーク管理システム（NMS）が必要になる．コンピュータネットワークはオープンな市場であり，複数のベンダによるネットワーク機器を利用するのが一般的である（マルチベンダネットワーク）．これらのネットワーク機器の多くは，管理のためのインタフェース（管理プロトコル）として，業界標準であるSNMP（Simple Network Management Protocol）を実装している．しかし，実際には，各社とも独自の制約や拡張を設けていることが多く，処理によっては別の管理プロトコルを用いる必要がある．これらの差異はNMSで吸収することができるが，事例それぞれに対応した開発となり

† NTTコミュニケーションズ
NTT Communications Corporation
†† NTT西日本
NTT West Corporation
††† NTT情報流通プラットフォーム研究所
NTT Information Sharing Platform Laboratories

コスト高になる問題がある．しかも，技術的進歩が早いことから，新しいネットワーク機器の追加・入れ替えの頻度は高く，NMS の維持コストが高くなる．

NMS の生産コストを下げるにはアプリケーションフレームワークが有望である．特に，管理プロトコルの差異を少ないコストで吸収できる必要があり，そのためには，アプリケーションフレームワークにおいて，異なる管理プロトコルに対して高い再利用性を持つ構造を実現する必要がある．クラスをそのまま（修正せず）組み合わせて NMS を構築できるようにすれば，再利用性は上がるが実行時の判断処理が増え実行効率が落ち，かつフレームワークの設計も困難になる．逆に，継承による再利用は，実行効率は良いがプログラミングを必要とするため再利用性が上がらない．これらの適切な組合せが課題になる．

筆者らは，以下のアプローチにより課題を解決した．

- 管理機能のロジックを，ネットワーク機器の制御・情報処理と独立して実装するレイヤ構造を導入した．
- ネットワーク機器の制御・情報取得処理を，管理プロトコルを動的（実行時）に選択する機構と，管理プロトコを利用した機器のアクセス処理に分けて構成した．これにより，管理プロトコルに依存する処理を後者のみに局所化させ，かつ管理プロトコルの追加・変更を容易にできる．
- 管理プロトコルの処理は，継承を用いて実装する．その際，機器固有の制約や拡張に対する処理部分がサブクラスとして作成される．継承を利用し，実行時の判断を減らすことで，実行効率の低下をおさえ，管理プロトコル依存部を小さくかつ局所化する．

2. 現状の問題点と要求条件

2.1 現状の NMS 技術

NMS は，

(1) 種々のネットワーク機器から情報を収集し，
(2) それらを解析，処理，提示し，
(3) オペレータがネットワーク機器をコントロールする作業を支援する．

コンピュータネットワークにおいて，ネットワーク機器の遠隔制御や情報取得で最もよく用いられている管理プロトコルは，SNMP（Simple Network Management Protocol）[6]である．これで定められている，ネットワーク機器が保持する情報を MIB（Management Information Base）と呼ぶ．

SNMP は，シンプルなトラップ&ポーリングという管理モデルを持つ．ポーリングとは，コンピュータネットワーク管理システムが，必要な情報をネットワーク機器から取得する動作である．逆に，トラップは，ネットワーク機器が，あらかじめ決められた条件（たとえばリンクが切れた）が満たされたとき，その旨を通知するため管理システムに向かって送るものである．通常は，トラップと定期的なポーリングを併用することで，信頼性を上げている．

上記の (2) はネットワークの構成や管理する組織の方針により機能が異なり，標準もない．また，(1) と (3) についても，SNMP は「標準」であるが，SNMP 準拠のネットワーク機器はベンダ独自の拡張や制約を設けている場合が多い．たとえば，あるルータでは，ルーティングテーブルを SNMP で更新できない．Telnet セッションを用いた独自のコマンドを利用する必要がある．また，MIB は，ベンダが独自に定義できるエンタープライズパートを持っている．さらに，SNMP 以外の管理プロトコル（たとえば CMOT：CMIP over TCP/IP）を実装している機器も存在する．

また，技術革新が早く管理技術の標準化が未完了の段階でネットワークを運用しなければならない場合も少なくない．たとえば，現時点では，QoS（Quality of Services）やバーチャル LAN などの管理には，ベンダごとに管理プロトコル（コマンド体系を含む）を使い分ける必要がある．これは，将来的には COPS（Common Open Policy Service protocol）の標準化・普及によりある程度解決されることが期待されている．

NMS の市販パッケージには拡張性を有するものがある．1 つは，プライベート MIB について，定義ファイルをインストールして SNMP により取得・設定できるようにするものである．しかし，一般にプライベート MIB の意味付けや構造はベンダごとに異なるため，統一した管理機能を使うためにはそれらの違いを吸収する処理を開発する必要が生じる．また，SNMP トラップの種別や MIB 値の閾値判断に応じて外部プログラムを起動したり，ベンダ固有の管理プロトコルに変換する外部プログラムと連携させたりする拡張性を有するものがある．これら機種・ベンダごとに必要な外部プログラムなどを効率的に開発できる必要がある．

このように，事例ごとに異なる機能が要求される NMS の導入コストをおさえるには，アプリケーションフレームワークが有望である．アプリケーションフレームワークとは，アプリケーションの骨格を実装したもので，これを利用してアプリケーションを短期間，低コストでしかも安定した品質のものを開発&カスタマイズできるようにするものである[2]．しかし，マル

チベンダネットワーク機器への適応性を考えた場合，既存の NMS 用アプリケーションフレームワークは SNMP などの代表的なプロトコルの標準処理を提供するのみであり，拡張・制限などに効率的に対応できない．

2.2 要求条件

NMS の導入や維持のコストをおさえるには，アプリケーションフレームワークが，種々の管理機能および複数の種類や制約・拡張を持つ管理プロトコルに対して高い再利用性を持つ必要がある．ここでは，再利用性を以下の尺度で評価する．

- 修正規模，すなわち，修正せず再利用したコード量と修正や追加したコード量の比
- 修正個所の局所性，すなわち，無修正で再利用したクラス数に対する修正や追加したクラス数の比率

さらに，NMS には，スケーラビリティ（拡張性）とフォールトトレランス（耐故障性）が要求される．前者は，管理システムが管理対象数の増加に対応できる能力を，後者は，管理システムが，自身の故障にかかわらず機能し続ける能力を指す．また，管理プロトコルを処理内容に応じて使い分けるだけでなく，障害などの場合，そのときどきに利用可能な管理プロトコルを選んで処理を継続できることが望ましい．

処理速度については，コンピュータネットワークで用いる NMS の場合，厳しいリアルタイム性は要求されない場合が多い．実用的なレベルとして，数個のサブネットについて 1 分間隔前後でポーリングができる処理速度を目標とした．

筆者らは，これらの要求条件を満たす NMS 用アプリケーションフレームワークを開発した[1),3)]．本論文では，複数の種類と制約・拡張がある管理プロトコルに対して高い再利用性を持つフレームワーク構成法について報告する☆．

3. NMS 用アプリケーションフレームワークの構築

3.1 技術的課題

アプリケーションフレームワークは，ある特定の問題領域のアプリケーションの骨格を実装したものである．オブジェクト指向技術を用いれば骨格を抽象クラスの集合として実現することが可能である．

アプリケーションフレームワークは大きく 2 つのタイプ：ホワイトボックスフレームワークとブラックボックスフレームワークに分類される．ブラックボックスフレームワークの場合は，クラスをそのまま組み合わせてプログラムを構築するのに対し，ホワイトボックスフレームワークでは，クラスをインヘリタンスにより特殊化して利用する．ブラックボックスフレームワークの方が再利用性が高いといわれている．ホワイトボックスフレームワークは，実装まで理解しなければ利用できない．しかし，一般的にブラックボックスフレームワークは，設計が難しく実行スピードも劣るという欠点がある[2)]．アプリケーション領域ごとに処理パターンや要求条件を注意深く分析し，これら 2 種類のフレームワークを適切に使い分ける必要がある．

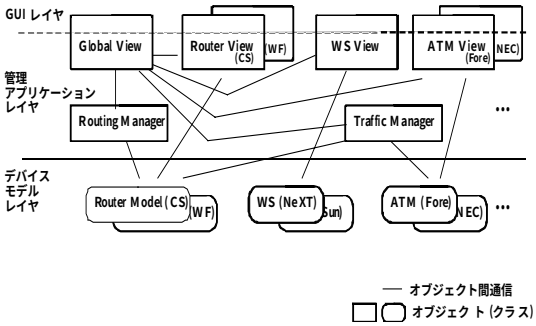

図 1 管理システム用アプリケーションフレームワークのアーキテクチャ
Fig. 1 NMS application framework architecture.

3.2 フレームワークの構成

筆者らが開発したアプリケーションフレームワークの構造を**図 1** に示す．

ネットワーク機器 1 台に対して 1 つのオブジェクト（デバイスモデルオブジェクトまたは DM オブジェクトと呼ぶ）を対応させる．

DM オブジェクトは，上位レイヤのオブジェクトからの指示に従い，担当するネットワーク機器のコントロールと情報取得・維持を適切な管理プロトコルを使い分けて実施する．DM オブジェクトは，ネットワーク機器に共通する部分を実装した抽象クラスを特殊化して，特定の製品/機種ごとに作成する．DM オブジェクトはデバイスモデルレイヤを形成する．

中間のレイヤを管理アプリケーションレイヤと呼ぶ．そこに属するオブジェクトは管理アプリケーションオブジェクト（または MA オブジェクト）と呼ばれ，ネットワーク管理で行われる種々の作業（経路情報管理やトラヒック管理など）のロジックを実装している．これらの作業は，DM オブジェクト経由でネットワーク機器の情報を取得し，分析し，ネットワーク管理者に

☆ スケーラビリティとフォールトトレランスについては，位置透過型の分散オブジェクト技術を拡張し実現した．本技術については別の機会に議論する．

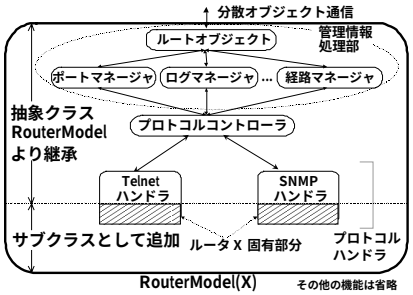

図 2 デバイスモデルオブジェクト
Fig. 2 Device model object.

提示し，指示を得て，DM オブジェクト経由でネットワーク機器をコントロールするという形で進められる．

GUI レイヤは，ユーザへのインタフェースを提供する．

図 1 に示したすべてのオブジェクトは分散オブジェクトである．その他，これらいずれのレイヤにも属さない，分散オブジェクトの監視など共通機能を提供するオブジェクトが存在する（図 1 では省略）．

フレームワークは NeXT Step OS 上の Objective-C 言語で記述されている．管理システムは，この OS 上で開発され，かつ動作する．

DM レイヤはネットワーク機器の制御と情報取得処理の抽象化されたインタフェースを提供する．これにより，管理機能のロジックは，管理プロトコルと独立に実装できることを目指している．したがって，ネットワーク機器ごとに作成する DM オブジェクトの生産性が重要になる．

3.3 デバイスモデルオブジェクト

デバイスモデルオブジェクトは，管理情報処理部，プロトコルコントローラ，複数のプロトコルハンドラ，およびその他（主にネットワーク機器の構成情報を扱う）より構成される（**図 2**）．管理情報処理部の 1 つであるルートオブジェクトは，MA オブジェクトからの要求を受け取り，適切な管理情報処理部，たとえばポート情報を管理するポートマネージャやトラップの履歴を管理するトラップマネージャなどへ転送する．管理情報処理部（ルートオブジェクトを除く）で必要となるネットワーク機器へのアクセスは，今回開発したマルチプロトコルハンドリングメカニズムを用いて，管理プロトコルに依存しない形で実装されている．

3.4 マルチプロトコルハンドリングメカニズム

デバイスモデルオブジェクトは，1 つのプロトコルコントローラと複数のプロトコルハンドラから成るマルチプロトコルハンドリングメカニズムを含む．

プロトコルハンドラは，たとえば，ルータ X 用

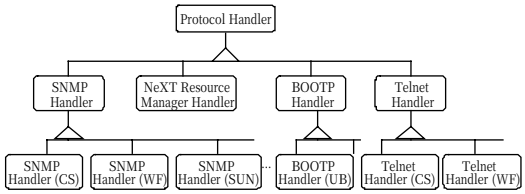

図 3 プロトコルハンドラの継承構造
Fig. 3 Class hierarchy of the protocol handlers.

SNMP ハンドラのように，ある特定の管理プロトコルを用いて特定の種類のネットワーク機器へアクセスする処理を実装したものである．プロトコルコントローラは管理情報処理部より処理要求を受け取り，それを処理するのに適切なプロトコルハンドラを動的に選択，要求を転送する．プロトコルハンドラは処理要求に従いネットワーク機器をコントロール/情報取得し，結果をプロトコルコントローラ経由で処理要求の依頼元に返す．この機構を用いることにより，管理情報処理部のプログラムコードを，対象とするネットワーク機器の管理プロトコルに依存しない形で記述することができる．プロトコルハンドラのクラス階層を**図 3** に示す．最上位のクラスでは，処理要求に対する可否を調べる機構（後述）などを実装している．中間のクラスは，各プロトコルの機種に依存しない部分を，末端のクラスは依存する部分をそれぞれ実装する．新しいネットワーク機器に対応する場合はこの末端のクラスを実装する．

マルチプロトコルハンドリングメカニズムの動作例は以下のとおりである．プロトコルハンドラは，配下のプロトコルハンドラの優先順位リストを持っている．処理要求（ある特定のコントロール/情報取得）を受け取ると，それができるかどうかプロトコルハンドラに優先順位の順番に問い合わせ，返答を得る．最初に処理可能と返答したプロトコルハンドラにその処理を委任する．

管理プロトコルを動的に選択することにより，特定の管理プロトコルの障害に対応できる．たとえば，ある機器に SNMP でアクセスできなくなった場合，代わりに telnet を用いてアクセスを継続する．これは以下のような仕組みにより実現される．各プロトコルハンドラは，それぞれの管理プロトコルによる通信障害の回数を記憶している．閾値を超えた場合には，そのプロトコルハンドラの優先順位を最下位に変更する．

このように，管理情報処理部とプロトコルコントローラはブラックボックスタイプ，プロトコルハンドラはホワイトボックスタイプである．この構成には以下の利点がある．

- 管理情報処理部とプロトコルコントローラは無修整で再利用できる．また，それ以外の部分では修正箇所が局所化され，かつ事前に特定されているため，修正作業が短縮される．
- 新しい管理プロトコルを簡単に追加できる．そのプロトコル用のプロトコルハンドラを作成するだけで，全体の制御構造を変更する必要がない．
- プロトコルハンドラは実際のプロトコル処理を行うところであり，処理スピードが速いほうが望ましい．以下の理由により，プロトコルハンドラは，差分プログラミングに適しており，ホワイトボックスタイプにしても開発が容易であることが期待できる．管理プロトコルに対する製品ごとの制約や拡張は，管理プロトコル処理全体からみると小さく，かつ管理プロトコルの構造が明確なため差分として記述しやすい．

3.5 アプリケーションフレームワークの利用手順

管理システムを構築するイメージを**図 4** に示す．管理するネットワーク機器はルータ X と Y（異なる機種）だと仮定する．ルータ X と Y に対応する DM オブジェクトは，抽象クラス RouterModel のサブクラス RouterModel (X)，RouterModel (Y) としてそれぞれ作成する．その際，RouterModel の管理情報処理部とプロトコルコントローラは何の変更もなしに利用できる．プロトコルハンドラは，継承され，それぞれのルータに固有の部分をサブクラスとして実装する（たとえば，図中のルータ X 用 SNMP ハンドラ）．図 4 の例では，RouterModel (Y) に BOOTP ハンドラを追加している．これは，抽象クラスにまだなかったために必要となった．その後，開発された BOOTP ハンドラは，再利用できるように一部修正後に抽象クラス RouterModel に追加される．

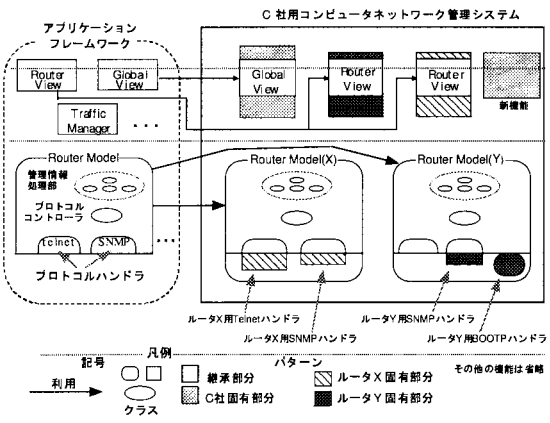

図 4 管理システムの構築手順のイメージ
Fig. 4 Developing NMS using the framework.

管理アプリケーションレイヤでは，主に，過去の事例を再利用して，GUI 部分，管理機能および管理機能間の呼び出し関係などを修正する．GUI 関係は，NeXT Step のインタフェースビルダにより効率的に開発することができる．

4. 評 価

4.1 概 要

開発したアプリケーションフレームワークの主な適用事例を**表 1** に示す．本論文の論点であるネットワーク機器（管理プロトコル）への適応性を評価した．評価では，代表的なネットワーク機器と管理機能を実装した表 1 の事例 1 を解析した．

4.2 ネットワーク機器への適応性

DM オブジェクトのプログラムコードを分析し，ネットワーク機器製品に固有のクラスとそうでない共通するクラスとに分類した（**表 2**）．その結果，プロトコルコントローラおよび管理情報処理部は，製品に固有のプログラムコードを含まない，すなわちどの製品にもそのまま再利用できていることが確認できた．製品に固有のプログラムコードは，プロトコルハンドラおよびネットワーク機器の構成を処理するクラス（表 2 では「その他」に分類）に限定され，規模も小さい，たとえば，Cisco4000 ルータでは 43 クラス中 7（16%）または 22,992 行中 1,093（5%），UB 社ハブでは，32 クラス中 6（19%） または 23,805 行中 1,235（5%），および SunOS のワークステーションでは，39 クラス中 3（8%）または 21,880 行中 459（2%）であった．これは，アプリケーションフレームワークに少ないコードを追加するだけでネットワーク機器に対応できたことを示している．

ベンダ間で MIB の内容が大きく違う例についても適応性を分析する必要がある．表 1 にある適用事例の多くは権利の関係上解析することができないため，2 社のイーサネットスイッチに実装されている MIB の定義ファイルを机上比較した．イーサネットスイッチは，VLAN や QoS など機能が複雑化しており，かつ標準化中のものがあり，比較的ベンダ間で MIB 定義が異なる部分が多い．結果を**表 3** に示す．

両社のプライベート MIB で，意味が同じでそのまま対応づけられるものは少ない（4 個）ことが分かる．NMS はすべての MIB を使うわけではなく一概にはいえないが，NMS を作成する場合，プライベート MIB の処理にはベンダごとに個別のコードを作成する必要がでてくると思われる．その比率は，共通化できる部分に対して，E 社は MIB 数の比で 10%程度ですむが，

表 1 アプリケーションフレームワークの主な適応事例
Table 1 Framework application example.

No	事例	主な特徴	管理対象数	期間，工数
1	LAN 管理	マップ表示，機器状態表示・制御 GUI，アラーム，自動構成認識，トラヒックのグラフィカルレポート	約 10（ルータ，ハブ，WS）	—
2	商用インターネットサービス (1)	マップ表示，機器状態表示・制御 GUI，ポケベルによるアラーム	約 100（ルータ，dial-up ルータ，ハブ）	2 カ月 4 人月
3	仮想 CALS ネットワーク管理（PC のネットワーク）	マップ表示，機器状態表示・制御 GUI，アラーム，パソコンの監視	約 50（PC，ハブ，ルータ）	1 カ月 1 人月
1〜3 全体	主な管理対象のベンダ数：ルータ (2)，WS(4)，ATM SW(1)，HUB (1) MIB アクセス行数：230，MIB オブジェクト種類：109，プロトコルハンドラのメソッド数：406（SNMP 関係は 225）（注）			
4	ビデオオンデマンドシステム管理	マップ表示，機器状態表示・制御 GUI，アラーム，障害経路解析，ATM 性能管理	約 300（STBs，ATMs，SLTs）	3 カ月 8 人月
5	企業ネットワーク管理のアウトソーシング	トラヒック収集スケジューリング，マップ表示，機器状態表示・制御 GUI，アラーム	約 5（ハブ）	1 カ月 2.5 人月
6	学術ネットワーク管理	経路情報チェック，マップ表示，機器状態表示・制御 GUI，アラーム，ループ経路の自動検出，帯域監視・アラーム，トラヒックレポート	約 5（ルータ）	—
7	商用インターネットサービス (2)（設備管理システムの一部）	アラーム，RDB アクセス，ルータ状態表示・制御 GUI	約 50（ルータ，SLTs，ハブ，WSs）	—
8	社内ネットワークのサーバ管理	Proxy server，News server，WWW server management 管理	約 7（WS，ルータ，gateway）	1.5 カ月 3 人月
9	商用インターネット・バックボーン	従量制課金システム管理	2（WS）	0.6 カ月 0.6 人月

（注）権利の関係などで詳細な解析ができないため参考情報として示す.

表 2 デバイスモデルオブジェクトの分析結果
Table 2 DM object analysis.

オブジェクト名	共通		Cisco4000 固有		WF 固有	
	クラス数	行数	クラス数	行数	クラス数	行数
Router Model	36	21899	7	1093	5	848
管理情報処理部	17	1976	0	0	0	0
プロトコルコントローラ	3	403	0	0	0	0
SNMP ハンドラ	4	6458	1	28	1	28
Telnet ハンドラ	1	306	2	291	-	-
その他	11	12756	4	774	4	820

オブジェクト名	共通		UB 固有	
	クラス数	行数	クラス数	行数
Hub Model	26	22570	6	1235
管理情報処理部	4	334	0	0
プロトコルコントローラ	2	403	0	0
SNMP ハンドラ	3	6177	1	222
Simulator ハンドラ	0	0	1	278
その他	17	15656	4	735

オブジェクト名	共通		SunOS 固有		HP-UX 固有		NeXT 固有	
	クラス数	行数	クラス数	行数	クラス数	行数	クラス数	行数
WS Model	36	21421	3	459	3	303	2	516
管理情報処理部	16	1387	0	0	0	0	0	0
プロトコルコントローラ	2	403	0	0	0	0	0	0
SNMP ハンドラ	4	6499	1	172	1	19	-	-
RPC ハンドラ	2	376	0	0	0	0	-	-
その他	12	12756	2	287	2	284	2	516

C 社の場合は約 34%程度になり，SNMP の処理部分については，提案した手法を用いてもあまり高い再利用性は得られない可能性がある．ただし，SNMP 以外の方法で対応しなければならない表 3 の MIB 59 個分については，将来的に新プロトコル（COPS など）が C 社スイッチでサポートされ NMS を対応させる際に手法の効果が期待できる（プロトコルハンドラを追加するだけでよい）.

表 3 イーサネットスイッチの MIB 比較
Table 3 Ethernet switch MIB analysis.

ベンダ	RFC 準拠 MIB 数	プライベート MIB 数
C 社	771	399
E 社	1067	120

E 社のプライベート MIB の内訳	
対応種別	オブジェクト数
C 社プライベート MIB とそのまま対応付け可能	4
C 社プライベート MIB の変換により対応可能	30
C 社スイッチに MIB なし（要 SNMP 以外の手段）	59
C 社スイッチに機能がなく対応不可	27

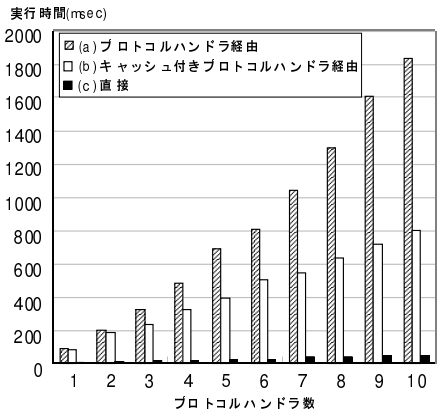

図 5 マルチプロトコルハンドリングメカニズムの処理速度
Fig. 5 Performance of the multiprotocol handling mechanism.

以上は，比較的小さな規模の NMS に対する分析であったが，大規模で複雑な NMS の開発に対する手法の有効性についてはさらなる評価が必要となる．

一方，ブラックボックスフレームワークは処理効率の低下をもたらす．その影響を調べるため，プロトコルハンドラの各メソッドをテストプログラムから呼び出す処理にかかる時間について，直接呼び出す場合（**図 5** (c)）と，プロトコルコントローラを経由して呼び出す場合（図 5 (a)）（マルチプロトコルハンドリングメカニズム方式）について測定した．その結果は，後者は前者に比べて処理時間が 18 倍（プロトコルハンドラが 3 つ，それぞれ 10 メソッドを持つ場合）かかり，処理効率の低下が大きいことが判明した．プロトコルハンドラ数の増加に対する処理時間の増加の傾向から判断し，プロトコルコントローラとプロトコルハンドラのネゴシエーションのオーバヘッドが大きいことが予想できる．

以上の処理効率の低下は，表 1 の各適応事例では許容範囲であった．その理由を以下に示す．

- 1 回のメソッド呼び出しにかかる時間はネットワークの通信時間に比べて小さい．
- 1 つの DM オブジェクトが含むプロトコルハンドラの数は 1〜3 個程度のことが多い．また，低い優先順位のプロトコルハンドラに処理要求が行くことは比較的少ない．

4.3 動的なプロトコル選択処理の洗練化

動的な管理プロトコルの選択機処理には他の実現方法が考えられる．たとえば，表 1 の適用事例により以下の傾向があることが判明した．

- 多くの処理は第 1 優先順位のプロトコルハンドラで処理される，
- 1 つのネットワーク機器用に必要なプロトコルハンドラは 3 つ以内．

したがって，プロトコルハンドラ間で優先順位に沿って処理を委任（delegate）する方法は，処理効率が良いことが期待できる．また，マルチプロトコルハンドリングメカニズムにおいて，プロトコルコントローラとプロトコルハンドラのネゴシエーションのオーバヘッドが大きいことを考慮すると，プロトコルコントローラが，プロトコルハンドラとのネゴシエーション結果をハッシュテーブルなどでキャッシュすることによる処理効率改善が考えられる．図 5 (a) と同じテストで実行速度を測定した（均等な呼び出しではあるが）結果，プロトコルハンドラ数が大きくなるにつれて，(a) に比べて処理速度も向上している（図 5 (b)）．この実験ではキャッシュサイズに上限を設定していないが，NMS はポーリングという比較的決まった処理を周期的に行う傾向があるため，これに近い効果を得られることが期待できる．プロトコルハンドラ数が多く処理速度が求められる場合に有効である．

4.4 事例への適応性

レイヤ構造は，上位レイヤを，管理プロトコルから独立させるには有効であったがネットワーク機器の依存性をなくすことはできなかった．たとえば，表 1 に示した多くの事例で，ネットワーク機器の実際のパネルを模した GUI を表示することにしたため，MA オブジェクトにネットワーク機器への依存部を作ることとなった．その影響を調べるため，MA オブジェクトについて，事例またはネットワーク機器に依存しているクラスと依存しない共通クラスに分類した．たとえば，あるルータ（Cisco4000）の管理をするための MA オブジェクトの場合，事例または機種依存部分は 38 クラス中 10（26%）または 31,042 行中 1,729（6%），UB 社ハブでは，24 クラス中 3（13%）または 22,091 行中 2,053（9%）であり小さく，比較的再利用性が高いことを示した．これは，事例やネットワーク機器へ

の依存部分が 管理機能の起動インタフェース部分と，そこから起動される管理機能の一部に限定されているためと分析できる．

5. 関連する研究

管理プロトコルの多様性はよく知られた問題である．強い標準化により解決するアプローチが存在するが（たとえば CMIP，CMOT[8]），標準化は時間がかかり，かつ普及しなければ効果がない．各ネットワーク機器ベンダは競争力強化のため独自の機能を導入する傾向にあるのも障害になっている．異なる管理プロトコル標準に対応する技術として，プロトコル変換や複数を包含する上位のプロトコルも検討されている[7]．これらは，標準化により固定化されているプロトコルには有効だが，実用の際には機器ごとの制限・拡張を吸収する手段が別途必要になる．

CMIP においてベンダごとの MIB の違いを効率的に吸収するため，MIB 間の対応関係を記述し，その参照プログラムを自動生成する手法が提案されている[12]．これは SNMP への応用も可能と思われる．CMIP は，筆者らの事例には存在しなかったが，比較的通信事業者向けの装置に採用される傾向にあり，管理機能が充実している，すなわち，必要な処理が CMIP でできるようになっていることが多い．したがって，CMIP だけで処理する NMS を作成するような場合，管理プロトコルが固定されるため参照プログラム生成系を作成・維持するコストの負担が小さくなり，このような手法は有効である．本論文で問題としている，複数の管理プロトコルが必要となるケースでは，筆者らの手法をこの方法と組み合わせることでより効果を発揮できる可能性がある．MIB の関係記述の仕様やそのコンパイラを複数の管理プロトコル用に拡張してしまうと，システムのメンテナンス性が悪くなる（特に，ベンダごとに固有のコマンドを使うような場合は問題）．提案したフレームワーク構成法はこの問題を解決してくれると考える．MIB の関係記述とコンパイラは管理プロトコルごとに作成し，それぞれで生成したプログラムを実行時に切り替えながら動作させる（個々のプロトコルハンドラを関係記述より自動生成することに相当）．別の管理プロトコルで処理したい部分が生じた場合は，独立した関係記述の仕様とコンパイラを開発し，独立したプログラム（モジュール）を生成し，プロトコルハンドラとして NMS に追加すればよい．その管理プロトコルの利用が一過性の場合は，言語仕様やコンパイラを作成せず，直接プロトコルハンドラをプログラミングして NMS に組み込む方がより効果的と思われる．

アプリケーションフレームワークは，ツールキットやライブラリなどのような再利用可能なプログラムコードの単純な集合とは異なる．難しい設計判断が少なくてすみ，再利用される規模が大きくなる傾向にある[2]．従来 NMS は，SNMP や GUI のツールキット/ライブラリを利用して開発されることが多く，アプリケーションフレームワークに比べ再利用性が低く，生産性と品質の安定化には多くの労力とスキルが要求された．アプリケーションフレームワークの大きな成功例としてグラフィカルユーザインタフェースがある．ネットワークプロトコルの実装[4]やソフトウェア開発ツール[11]に対する開発例も報告されている．最近では，NMS 用のものが製品化されているが，いずれも管理プロトコルについては SNMP などの標準を想定しており，機器単位で異なるプロトコルの制約・拡張を効率的に吸収する仕組みは考えられていない．

これら従来のアプリケーションフレームワークは，継承を利用するため，アプリケーションフレームワークの実装まで理解したうえでプログラミングする必要がある．より高い再利用性と品質の安定化のためには，できるだけソースコードを変更しないで組み立てたり機能変更したりできる適応的なソフトウェア構成技術が重要である．プログラム変換[5]などの静的なアプローチとともに，ソフトウェアが実行時に呼び出し先などを判断する動的なアプローチが存在する．特に後者はコンパイルなどが不要になり，動的な組立てや機能変更が可能になるため，アプリケーションフレームワークの利便性を向上することが期待できる．利用される技術として，リフレクション[13]，分散オブジェクトの動的起動[9]（コンポーネントウェア）や委任（delegation）[4]などがある．その際，動的なアプローチは実行時のオーバヘッドが大きいため，各アプリケーションの特徴を考慮し，静的な呼び出しとの適切な組合せが必要となる[10]．本論文のマルチプロトコルハンドリングメカニズムは，NMS における 1 つの組合せを提案している．

6. あとがき

本論文では，ネットワーク管理システム用のアプリケーションフレームワークにおいて，多様な管理プロトコルに対して高い再利用性を実現する構成法を提案した．

ネットワーク機器の制御・情報取得処理を，管理プロトコルを動的に選択する機構と，継承を用いた実装を行う管理プロトコルを利用した機器へのアクセス処

理に分けて構成するマルチプロトコルハンドリングメカニズムを実現した．これは，管理プロトコルに依存する処理を局所化させ，かつ管理プロトコルの追加・変更を容易にする．これにより，種々の管理プロトコルや制約・拡張を持つネットワーク機器に対して高い再利用性を持つアプリケーションフレームワークを実現できる．

複数の NMS 開発に適用した結果，多様な管理プロトコルに対して高い再利用性を有することを確認した．ただし，より大規模・複雑な NMS への有効性についてはさらに評価する必要がある．

参 考 文 献

1) Fujisaki, T., Hamada, M., et al.: A Scaleable Fault-tolerant Network Management System Built Using Distributed Object Technology, *Proc. 1st Int'l Enterprise Distributed Object Computing Workshop*, IL, pp.140–148, IEEE Computer Society Press (1997).
2) Gamma, E., et al.: *Design Patterns*, Addison-Wesley (1995).
3) Hamada, M., et al.: Building An Application Framework for Computer Network Management Systems, *Proc. 9th Int'l Conf. on Software Engineering and Knowledge Engineering*, IL, pp.579–587, Knowledge Systems Institute (1997).
4) Huni, H., Johnson, R. and Engel, R.: A Framework for Network Protocol Software, *Proc. OOPSLA'95*, New York, pp.358–368, Press (1995).
5) Lieberherr, K.: *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns*, PWS Publishing Company (1996).
6) Rose, M.: *The Simple Book: An Introduction to Internet Management*, 2nd edition, PTR Prentice Hall (1994).
7) Mazumdar, S., Brady, S. and Levine, D.: *Design of Protocol Independent Management Agent to Support SNMP and CMIP Queries*, Integrated Network Management, III, pp.377–388, Elsevier Science Publishers (1993).
8) McCloghrie, K. and Rose, M.: Common Management Information Services and Protocol Over TCP/IP (CMOT), RFC 1189, (1991).
9) OMG: The Common Object Request Broker: Architecture and Specification, Revision 2.0 (1995). http://www.omg.org/.
10) Posnak, E.R.L. and Vin, H.: An Adaptive Framework for Developing Multimedia Software Components, *Comm. ACM*, Vol.40, No.10, pp.43–47 (1997).
11) Sparks, S., Benner, K. and Faris, C.: Managing Object-oriented Framework Reuse, *IEEE Computer*, Vol.29, No.9, pp.52–61 (1996).
12) 堀内浩規，吉原貴仁，杉山敬三，小花貞夫，鈴木健二：ネットワーク管理のための管理情報ベース（MIB）に対する柔軟なビュー提供方式，情報処理学会論文誌，Vol.39, No.2, pp.367–378 (1998).
13) 渡部卓雄：リフレクション，コンピュータソフトウェア，Vol.11, No.3, pp.5–14 (1994).

(平成 10 年 10 月 23 日受付)
(平成 11 年 11 月 4 日採録)

浜田 雅樹（正会員）

1960 年生．1983 年慶応義塾大学工学部電気工学科卒業．1985 年同大学大学院修士課程修了．同年，NTT 入社．現在，NTT コミュニケーションズメディア技術開発センタ担当課長．1998 年より慶応義塾大学非常勤講師．インターネット・ソフトウェア技術に興味を持つ．1991 年情報処理学会奨励賞受賞．電子情報通信学会，IEEE CS 各会員．

蔭山 克禎（正会員）

1968 年生．1989 年明石工業高等専門学校電気工学科卒業．同年，NTT 入社．現在，NTT 西日本研究開発センタに在籍．主に，インターネット管理システムの研究に従事．

藤崎 智宏（正会員）

1966 年生．1989 年東京工業大学理学部情報科学科卒業．1995 年同大学大学院修士課程修了．同年，日本電信電話（株）ソフトウェア研究所入所．現在，NTT 情報流通プラットフォーム研究所研究主任．オブジェクト指向ソフトウェア開発，分散オブジェクト技術，コンピュータネットワーク管理技術に関係する研究に従事．ACM 会員．