分散システム運用技術 12－3
（1998. 11. 20）

ネットワーク構成の自動探索に基づく Java ベース IPv6 ネットワーク管理システムの実装

柏木 伸一郎　藤崎 智宏　浜田 雅樹
NTT ソフトウェア研究所 広域コンピューティング研究部

次世代インターネットプロトコル IPv6 の普及が始まろうとしている．IPv6 は現在のインターネットプロトコル IPv4 の後継となるプロトコルであり，既に多くのプラットフォーム上の実装が存在する．我々は IPv6 サービスプロバイダ実験を行っており，大規模な IPv6 ネットワークを効率よく管理運用する必要にせまられているが，現在入手可能な IPv4 用のネットワーク管理システムでは，IPv6 ネットワークを直接管理できない．便利である．本稿では，IPv6 ネットワークの動的な特性をふまえながら，現在試作中の Java ベース IPv6 ネットワーク管理システムの概要について述べる．

Java-based IPv6 Network Management System using Dynamic Topology Discovery

Shin-ichiro Kashiwagi　Tomohiro Fujisaki　Masaki Hamada
NTT Software Laboratories, Global Computing Laboratory

Internet Protocol version 6 (IPv6) which can solve various problems of current Internet Protocol (IPv4), has begun to be deployed on many experiment networks. There are many IPv6 implementations for almost all platforms, even routers. We have started the IPv6 experimental ISP and felt more necessity for an network management system which can manage IPv6 networks efficiently. This paper describes an IPv6 network management system which we have been developed, with dynamic network topology discovery techniques.

1 はじめに

現在のインターネットプロトコルの諸問題を解決する新技術として，次世代インターネットプロトコル IPv6 が注目されてきている．端末側としては，既に多くの OS やプラットフォームにおいて IPv6 が利用可能であるし，ルータ等のネットワーク機器も IPv6 に対応した製品が増えてきている．今後のインターネットも，既存の IPv4 機器と共存しながら将来的には IPv6 へ段階的に移行してゆくと考えられる．

現在，我々は IPv6 ネットワーク運用における基盤技術の確立を目的とした IPv6 インターネットサービスプロバイダ実験 NTTv6Net を運用している [1, 2]．国際 IPv6 実験網である 6bone にも最上位階層メンバ (pTLA) として参加しており，大規模な IPv6 実験ネットワークを効率よく管理運用する必要が生じている．

本稿では，IPv6 ネットワーク管理に関する現状に触れた後，現在我々が試作している IPv6 ネットワーク管理システム (NMS) の概要を述べる．特に，動的に変化するネットワーク構造を把握するた

めの，ネットワーク構造の自動探索手法について説明する．

2 IPv6 ネットワークの管理

大規模なネットワークの管理には，ネットワーク管理システム (NMS) が用いられる．IPv4 用には非常に多くの製品が出ており，ISP だけでなく一般のイントラネット管理に広く用いられている．

しかし，IPv6 ネットワークを監視可能な商用 NMS は，まだほとんど存在しない．今後 NMS ベンダの IPv6 対応も進むと考えられるが，現状の IPv6 ネットワークの管理には，これまでとは異なる以下のような特徴を考慮する必要がある．

2.1 ネットワーク構成の動的変化

従来の IPv4 NMS では，ネットワークトポロジはあまり変化しないという前提により，トポロジ情報は静的な構成情報として固定されていることが多かった．IPv6 では，ホストは原則としてルータからアドレス設定情報を受け取る．また，ルータのアドレスリナンバ機構についても議論されているので，IPv4 に比べてネットワークのリナンバやトポロジ変更が容易になると考えられる．また，現在存在する IPv6 ネットワークは実験目的で構築されていることが多いので，ネットワーク構成が頻繁に変化することは充分に予想される．

このため，IPv6 対応 NMS は，ネットワーク上のノードだけでなく，トポロジ構成も含めて監視し，変化を検出可能であることが望ましい．

2.2 管理手法の多様化

IPv6 ネットワークの特性や目的に応じて，今後様々な管理手法の提案がなされると思われる．そこで，新たな管理手法が容易に追加できたり，複数の手段で得た情報を同じ枠組みで扱うことができると便利である．例えば，管理対象から情報を得る手段だけに限定しても，IPv4 とは以下のように事情が異なる．

- IPv6 における SNMP 対応
  IPv6 に関する MIB 群も提案されている [3] が，現在では SNMP 対応機器がまだ少ない．また，実装に依存した特定のデーモンの動作など，MIB で定義されていない情報も必要である．さらに，障害の原因が対象機器のプロトコルスタックの実装の相性の問題であることも多い現状では，SNMP で得られる情報も必ずしも信頼できるとは限らない．
- 管理基盤としての IPv6
  経路や到達性自身に安定性がない場合には，IPv6 経由で IPv6 機器の監視を行うことは難しい．現状では IPv4/IPv6 デュアルスタック機器が多く，既存の IPv4 ネットワーク上に IPv6 ネットワークを構築していることも少なくないので，IPv4 経由で IPv6 ネットワーク管理を行えることが望ましい．

2.3 IPv4 との依存関係

現在では，DNS を初めとする IPv6 ネットワークを支える多くの機構が IPv4 に依存している．また実際にも，NAT やアプリケーションゲートウェイ等の設置により，何らかの形で IPv4 インターネットのサービスを利用することが多いと思われる．管理対象ネットワークにおける IPv4,IPv6 両方のアドレス体系を管理する仕組みが必要である．

3 Java ベース IPv6 NMS の設計

現在我々は，上記の問題を解決するために IPv6 用の NMS を試作中である．NMS の主な機能を，4 つの階層に分離した．

情報収集エージェント 上位のレイヤが決定したポリシに従い，ネットワークに関する情報収集を行う．

ネットワークモデルレイヤ 得られた情報は，ネットマップと呼ばれるネットワーク図に記入される．ネットマップには，ノードやリンク等，ネットワークを構成する要素がオブジェクトとして存在し，時間の経過により適宜更新・削除される．

ネットワーク監視レイヤ ネットワーク構造の自動探索を行い，ネットマップを構築するとともに，情報収集のポリシを実現する．従来の NMS における主な機能はここに含まれる．検出されたアラームは，より上位のネットワーク管理レイヤに高位イベントとして伝えられる．

管理アプリケーションレイヤ　アラーム等，下位レイヤから伝えられた高位イベントにより起動され，障害発生を管理者に通知したり，監視画面に反映させるなどのユーザインターフェースを担当する．

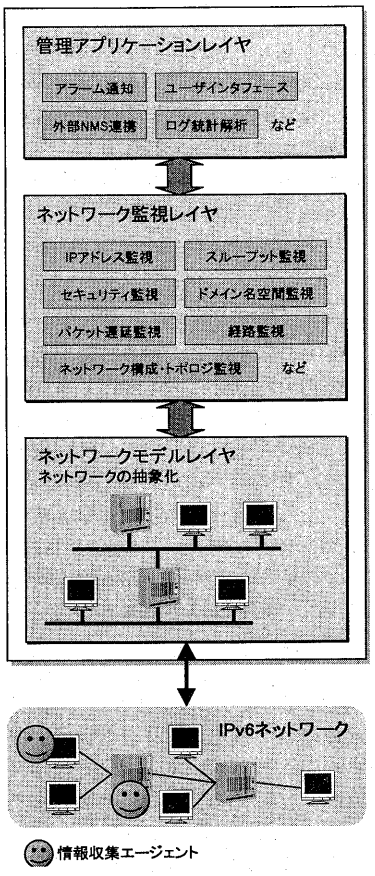

図 1: IPv6 NMS システム全体構成

3.1　情報収集エージェント

管理対象機器の近くに配置され，IPv6 ネットワークに関する情報の収集動作を行う．

従来の NMS では，以下のような情報収集手段が主に用いられていた．

- SNMP などネットワーク機器の管理用インターフェースを用いて，カウンタ情報等を取得する (MIB 値など)
- プローブを配置し，ネットワーク上のパケットを直接監視する

しかし前述したように，現在の IPv6 ネットワークでは SNMP の利用は困難である．UNIX マシン上の IPv6 スタックの動作状態を把握するためデーモンソフトウェアが出力するログの解析が必要になることも考えられるが，従来の NMS では別扱いとなる．

他の例を挙げる．IPv6 においては，近隣探索プロトコルによる近隣キャッシュの頻繁な更新が行われるので，多くのノードについて，ノードがアクティブかどうかを ping せずに検出できる可能性がある．しかし従来の NMS では，パケット観測により得られるノード情報と，ping によるポーリング結果の情報を同じように扱うことが難しい．

IPv6 ネットワークの発展にともない，これらの情報収集手段は今後大きく変化するだろう．従って，ログの解析結果も，パケット監視情報も，SNMP による情報も全て同じ枠組みで扱える機構が必要だと言える．このため，ネットワーク上における状態変化を，イベントという形式ですべて統一して表現する．

イベントは Java オブジェクトであり，少なくとも以下の項目を含む．処理内容はイベントの所属するクラスによって規定される．

- 対象ノード識別子 (MAC アドレス，IPv6 アドレス等)
- 時刻
- イベントクラス (パケット統計，MIB 値通知，ログ追加など．イベントの所属するクラスであり，イベント処理のためのメソッドを示す)．
- イベント内容によるオプション情報

3.2　ネットワークモデルレイヤ

情報収集エージェントで発生したイベントは，ネットワークモデル構築に使われ，時系列情報とともに蓄積される．

3.2.1　ネットマップ

イベントは到着後，以下のように処理される．

1. 自分自身を記録するためのスレッドを割り当てられる．

2. イベントが示している対象物(ホスト，ルータ，リンク等)をあらわすオブジェクト(ネットオブジェクトと呼ぶ)を検索する．見つからなければ，新たに生成する．
3. イベント内容に従い，これらのネットオブジェクトの持つ情報を更新する．必要であれば，他のネットオブジェクトへの参照を操作する．
4. イベントにより影響を受けるネットオブジェクトの更新が全て終了したら，イベントは消滅する．

このようにして，実行時に動的にネットワーク構成をあらわすオブジェクトグラフが構築される．これをネットマップと呼ぶ．ネットマップ自身は，イベントを受けとり，ネットオブジェクトの検索，配置(生成，削除)や，イベント記録等を可能にするインターフェースを持つ．

現在のネットワークの状態は，全てネットマップ上に表現される．ネットオブジェクトは，管理対象機器を抽象化したものであり，実際の機器へのアクセス手段(情報収集エージェントへの参照など)を持つ．上位レイヤの管理アプリケーションは，全てネットオブジェクトを管理することになる．

3.2.2 ネットオブジェクト

ネットオブジェクトして表現される管理対象は，ノード(ホスト，ルータ等)，リンク(共有型ハブ，スイッチ，point-to-point 型リンク等)などがある．これらの構成要素は共通する構造を持つ．例えば，ノードは各インターフェース毎に，インターフェース名，接続しているリンク(ノード)への参照，MACアドレス，IPv6アドレス，ホスト名等の情報が最低限必要である．しかし上位の管理アプリケーションにより他の属性が必要になることも考えられるので，これらの属性はオブジェクトのインスタンス変数とはせず，プロパティリストとして動的な追加削除を可能にする．各プロパティは，以下の構造を持つ．

- キー
- 値(整数，アドレス，文字列，オブジェクト参照など)
- 時刻(生成時刻，最終更新時刻，最終アクセス時刻など)
- 変更履歴記録

プロパティアクセスのため，create()，set()，get()，delete() 等のアクセスメソッドを用意している．

3.2.3 イベントヒストリ

ネットオブジェクトはイベントヒストリ(履歴記録)を持ち，自分に到着したイベントを時刻とともに記録する．管理者が定期的に監視する他に，上位の管理アプリケーションが必要に応じてイベントヒストリを参照しネットワークの状態判断に役立てるためである．

3.2.4 エージング

ネットワークの状況は動的に変化するので，ネットマップもその変化に追従する必要がある．ネットワークの変化の差分を完全に検出できればよいが，大規模なトポロジ変更などが頻繁に生じる実験環境において，差分を正しく求めるのは困難であり，差分によるネットマップ更新には限界がある．

そこで，全ての情報には有効期限があり，古い情報は信頼できないとする「エージング機構」を導入する．全ての情報はその有効期限内に再確認される必要があり，一定期間以上確認されないネットオブジェクトは消滅する．

ネットオブジェクトが存在し続けるには，誰かに存在を証明してもらう必要がある．この方式のメリットは，対象のネットオブジェクトが自分自身で存在証明をすることや，ping によるポーリングのように他人に調べてもらう[1]など，様々な方式をネットオブジェクトごとに選べることである．

定期的な監視の結果，ネットマップ上のオブジェクトやプロパティが変化すると，高位のイベントとして，ネットワーク監視レイヤに通知される．

3.3 ネットワーク監視レイヤ

ネットワーク監視機能を実現するレイヤである．

このレイヤからは直接対象機器をアクセスせず，ネットマップ上のネットオブジェクトを監視することにより，SNMP インターフェースを持たないダムハブなど論理的な実体がないものについても，監視が可能になる．例えば，リンクに流れている特定の種類のパケットを監視する場合は，以下のような手順になる．

[1] ping によるポーリングは，後述するネットワーク構造の自動探索に含まれている．

1. リンク (ネットオブジェクト) に対して指示を出す.
2. リンクは自分のリンク上に存在する情報収集エージェントを探し，指示を転送する.
3. 情報収集エージェントは，該当するリンクのパケット監視を開始する.
4. 監視結果は，イベントとして返される.

3.3.1 ネットワーク構成の自動探索

ネットマップを動的に作成，維持してゆくためには，ネットワーク構成の実行時の探索を行う必要がある．ネットワークに対して直接操作が可能なのは情報収集エージェントのみなので，各エージェントを起点にして，他のエージェントとの中間を埋めてゆくことになる．以下に具体的な手順を示す.

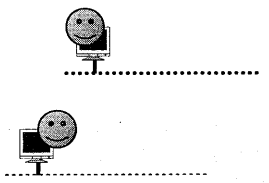

図 2: ネットワーク構成探索開始前

ノード探索 エージェントから，リンクローカルスコープの全ノードマルチキャストアドレス (FF02::1) 向けへ ICMP echo を送る．その返答により，リンク内に存在するノードのリンクローカル IPv6 アドレスの一覧が取得できる．これらの下位 64bit は，EUI-64 形式の第 2 層アドレスが含まれている.

また，通常エージェントは，自らのインターフェース情報からリンクのプレフィクスを知ることができると考えられるので，前述の情報と合わせるとグローバルな IPv6 アドレスの一覧を得ることができる．個々に ICMP echo を打って確認することも可能である．[2]全ルータマルチキャストアドレスを用いることにより，ルータとホストを区別できる.

発見された各ノードについて，ノードオブジェクトを作成し，ネットマップ上に配置する．また，自分のリンクについてもリンクオブジェクトを作成し，各ノードを登録する.

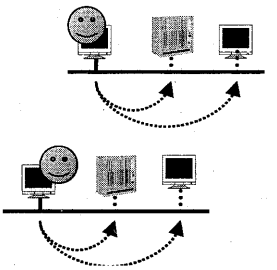

図 3: ノード探索

リンク / プレフィクス探索 基本的に，エージェント単独の情報では他のプレフィクスを知ることはできないが，幾つかの工夫が考えられる.

1. 遠隔エージェント，あるいはそれまでの探索においてアドレスを得られた遠隔ノードに向けて，traceroute を行う．これにより，途中のルータが持つインターフェースの IPv6 アドレス (プレフィクス) だけでなく，リンク間のトポロジも判明する.
2. 経路情報がエージェントから観測できれば，到達可能なプレフィクスの一覧として利用できる．リンク内のルータの位置は，近隣探索プロトコルのルータ要請メッセージを観測してもよいし，全ルータマルチキャストアドレスへ ICMP echo を送ってもよい．求まったルータに対して，(使用している経路制御プロトコルによるが) 経路一覧を問い合わせることや，リンクに実際にアドバタイズされる経路情報を収集できる可能性がある.
3. 実際にリンクに流れている IPv6 パケットのソースアドレスを観測することにより，特定の IPv6 ノード (プレフィクス) がどのルータの先にあるかという情報を得ることができる.

求まった複数のリンクについて，それぞれリンクオブジェクトを作成し，ネットマップ上に配置する.

トポロジ推測 これまでに求まったリンク群を，実際のネットワーク上のトポロジに従って接続することにより，ネットマップが完成する．トポロジ推測の材料としては，前述したように traceroute の実行結果や，経路情報に含まれる hop 数などが考えられる．往復で同じ経路だと仮定すると，traceroute を

[2]対象ノードが ICMP who are you を実装していれば，(正確さの保証はないが)FQDN 名を得ることが可能である.

双方向に行うことのより，複数のインターフェースを持つルータの同一性を推測できる．

traceroute に類似したアイデアとして，ICMP echo reply に含まれる hop limit 値により距離を測定することもできそうだが，この hop limit 初期値はノードの IPv6 スタックの実装によって異なるので困難である．他には，経路制御ヘッダ(拡張ヘッダ)を用いて，遠隔リンク上のノードを経由して自分のところへ戻ってくるようなソースルートを指定してテスト用 IPv6 パケットを送ることより，最小の hop 数を求めることも可能であろう．

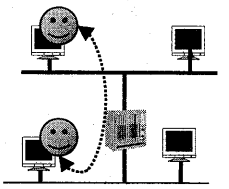

図 4: トポロジ探索終了

3.4 ネットワーク管理レイヤ

ネットワーク監視レイヤで検出された事象(ネットマップ構造の変化など)は，高位のイベントとしてネットワーク管理レイヤに到着する．このレイヤでは，これらの高位イベントをアラームとして管理者に伝えたり，監視画面に表示するなどの対処作業を実現している．

4 Java 分散オブジェクトによる実装

本システムは，Java の標準的な分散オブジェクト環境である RMI[4] を用いて実装を進めている．レイヤの各機能を担当するオブジェクトは他のレイヤとの通信のためのリモートインターフェースを持つ．ユーザインターフェースは WWW ブラウザ上の Java アプレットである．

NMS において最も分散性が要求されるのは，管理対象の近くに配置するエージェントであろう．エージェントも分散オブジェクトとして扱えると，負荷分散や耐故障性確保などの多くの要求について，分散オブジェクト環境側のサービスを期待できる．しかし，現在の JDK 1.1 Core API では IPv4 ソケットしか対応しておらず，RMI も IPv4 上でしか動作しない．また，エージェントの実体側も Java だけでは記述できず，パケット監視等はネイティブメソッド化する必要がある．現在の実装では，エージェントの wrappper を Java オブジェクトとし，そこからエージェント本体と IPv6 ソケットで通信するコマンドを呼び出している．

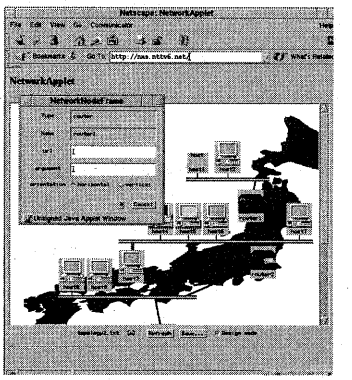

図 5: Java ベース IPv6 NMS の監視画面例

5 まとめ

IPv6 用の NMS を，Java 分散オブジェクトを用いて設計，試作している．IPv6 は，アドレス体系や近隣探索機構など，IPv4 と異なる点がいくつかあり，特にネットワークの動的側面は IPv6 用 NMS を設計する上で考慮すべきことである．特に，ネットワーク構成を自動探索しトポロジの変化を検出する機能は，現時点の IPv6 実験的なネットワークにおいて有効であると考える．今後，実装を進めて，運用上の問題点を検討するとともに，他の NMS の IPv6 対応動向をふまえて，分散協調管理の枠組みを提案していく予定である．

参考文献

[1] 藤崎智宏，柏木伸一郎．IPv6 サービスプロバイダの構築と運用．情処 DSM 研究会報告 10-3, 1998.

[2] NTT IPv6 サービスプロバイダ実験. http://www.nttv6.net/

[3] Management Information Base for IP Version 6: Textual Conventions and General Group, Internet Draft. <draft-ietf-ipngwg-ipv6-mib-04.txt>

[4] Java Remote Method Invocation, http://java.sun.com/products/jdk/rmi/