ON-API（Open Networking-Application Programming Interface）とON-APIを用いたネットワーク管理アプリケーションの開発

飯島智之[†1] 新善文[†1]
木村浩康[†1] 木谷誠[†1]

システム管理者は，複数ベンダから提供される様々なネットワーク機器に対応しなければならない．各ネットワーク機器の設定方法をマスタするためには膨大な時間を要し，システム管理者にかかる運用負荷は大きい．システム管理者の運用負荷を軽減するため，様々なネットワーク管理アプリケーションが開発されたが，現在ネットワーク機器が提供している設定インタフェースは主にCLI（Command Line Interface）である．CLIは元々人間が操作するために規定されたユーザインタフェースであり，アプリケーションの開発には不向きである．ネットワーク管理アプリケーションを容易に開発可能とするため，著者らはネットワーク機器制御用のAPIを開発した．開発したAPIは，Javaの開発環境で利用可能であり，管理アプリケーションとネットワーク機器間の通信にIETF（Internet Engineering Task Force）で標準化中のNETCONFを採用した．標準技術を採用したため，このAPIを用いて複数ベンダ対応のネットワーク管理アプリケーションを開発することも可能である．著者らは実際に，このAPIを用いてネットワーク管理アプリケーションを開発した．アプリケーションの設定性能を測定したところ，CLIと比べて遜色のない性能を示した．

Development of ON-API and Network Management Application Using ON-API

Tomoyuki Iijima,[†1] Yoshifumi Atarashi,[†1] Hiroyasu Kimura[†1] and Makoto Kitani[†1]

Network operators have to cope with various types of network devices provided from multi-vendors. It takes huge amount of time to master each network device's configuration method and it results in increased workload. In order to reduce operator's workload, several kinds of network management applications were developed. But configuration interfaces toward network devices are mainly provided in a form of CLI (Command Line Interface). CLI was originally developed for human-use, hence it is not suitable to be used for a development of network management application. In order to develop network management application more easily, we developed APIs to configure network devices. The APIs can be used in the Java development environment and is using NETCONF standardized at IETF (Internet Engineering Task Force) as a transportation protocol between a network management application and network devices. By adopting standardized technology, it is possible to easily develop a unified network management application which can deal with multi-vendors. We actually developed network management applications using the APIs. The applications showed as good performance as CLI.

1. はじめに

ブロードバンドネットワークや無線LANの普及により，インターネットの規模が拡大している．インターネットを介したビジネス形態も拡大の一途をたどっており，それにともないIPネットワークやシステムの運用管理も複雑になってきている[1)]．ネットワーク管理者は増加するIT機器にそれぞれ複雑な設定をすることになるため，その負担は指数関数的に大きくなっている[2)]．それにもかかわらず，ネットワーク運用に対する注目度は低いままである．ネットワークを統合的に運用管理する技術に，これまで革新的な技術はなくいまだに管理者の経験に基づいた旧態依然の手法が用いられ

†1 アラクサラネットワークス株式会社
ALAXALA Networks Corporation

ている[3]．IT システムの変更や開発にあわせて，ネットワークも柔軟で敏速に変更可能であるべきである．

著者らは，IT システムの変更や開発にあわせて日々更新が可能なネットワークを実現するため，ネットワークの「運用の自動化」と「IT システムとの高い親和性」に重点をおき，研究に着手した．

(1) 運用の自動化の実現

従来ネットワーク機器は，主にコマンドベースの CLI で制御していた．CLI はネットワーク管理者が利用することを前提として開発されているため，仕様は機器や機種，提供ベンダに依存する．それでも，操作する対象が人間という柔軟で知的な存在であったため，機器や機種，提供ベンダごとに異なる仕様を吸収して運用することが可能であった．

ところが，「運用の自動化」の実現を目標としてこれをコンピュータに代行させようとすると，複雑な処理を盛り込む必要がある．たとえば CLI には，戻り値の概念がなく，応答メッセージを解析するコードを用意する必要がある．また，応答メッセージのフォーマットやメッセージの出現順序なども機器ごとに異なるため，対応するラッパプログラムを書くにも，多大な労力を必要とする．

SNMP（Simple Network Management Protocol）は，この状況を改善するために登場し，それをベースとして様々な試みがなされた[3]．しかしながら，そもそもオブジェクト指向でなく変数指向であるため，ネットワーク装置に設定を施す場合，MIB（Management Information Base）ツリーの変数を 1 つずつ設定していかなければならなかった．さらに，複数のマネージャが同時に MIB へ変更を加えようとした場合，衝突を回避する仕組みがなかった[4]．このような状況のため，結局は設定に不向きで，主に情報収集のためにしか利用されてこなかった[1),5)]．その証拠に，インターネットが日々進化しているにもかかわらず，SNMPv4 へ向けた作業部会は作られていない[6]．

「運用の自動化」を実現するための新たな運用方式が模索されている．

(2) IT システムとの高い親和性の実現

IT システムで最もコストがかかるのがプログラム開発である．プログラムの生産性を向上させ，開発期間を短縮するため，Cobol，PL/I，C，Java などのプログラミング言語が登場した．その一方，ネットワーク機器に対する制御は CLI の域を出ていない．アプリケーションからネットワーク機器の機能やネットワーク構成を制御可能な制御用インタフェースが求められている．

そのような制御用インタフェースがあれば，IT システムとも親和性の高いネットワーク管理アプリケーションが開発可能となる．たとえば，「会議室予約システムで会議室が予約されると，それと合わせて会議室に設置されたネットワーク機器のポートを開ける」などの，業務と連動してネットワークを制御するネットワーク管理アプリケーションが開発可能となる．

2. 従来技術の課題と解決する技術

これまでに述べた，CLI や SNMP のネットワーク運用方式としての課題とその対策，およびその対策技術を**表 1** にまとめた．

従来のネットワーク運用方式の課題を解決し，「運用の自動化」「IT システムとの高い親和性」を実現する技術として，「NETCONF」，「データモデル」，「Java API」をあげた．ネットワーク機能をオブジェクト化して制御可能なインタフェースとしては，Java API のほかにも C++ や Ruby などもあるが，開発者の人口，技術成熟度，そしてそれにともなう NETCONF 実装のしやすさの観点から Java API を採用した．

それぞれの対策技術の関係を**図 1** に示す．それぞれの技術の詳細と，それらがどのように課題を解決するのかを以下に述べる．

2.1 NETCONF

NETCONF は，IETF（Internet Engineering Task Force）において標準化が進められている，ネットワー

表 1 「運用の自動化」「IT システムとの高い親和性」の実現に必要な対策
Table 1 Necessary measures for automation and high compatibility with IT systems.

課題	対策	対策技術
結果判定の枠組みに標準的な仕様がなく，機器や機種，提供ベンダ毎に異なる	標準プロトコルを策定する	NETCONF
MIB ツリー化した設定項目は，コンピュータから扱いにくい	設定項目を論理的に構造化する	データモデル
アプリケーションからネットワーク機器やネットワーク構成を制御する開発技術がない	ネットワークの機能をオブジェクト化した制御用インタフェースを提供する	Java API

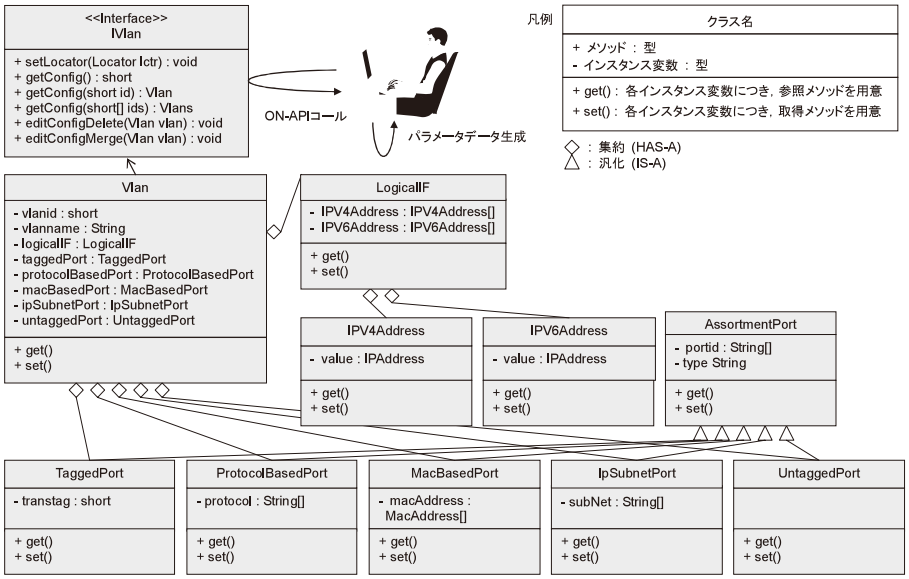


図 2 VLAN のデータモデル
Fig. 2 VLAN's data model.

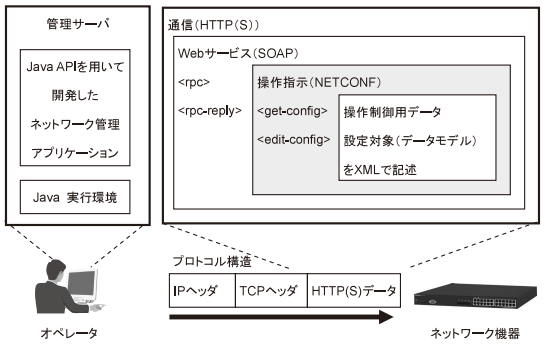


図 1 NETCONF，データモデル，Java API の関係
Fig. 1 The relation among NETCONF, data model and Java API.

ク機器を制御するための次世代プロトコルである[7),8)]．図 1 に示すように，NETCONF は設定項目の操作手順を標準化しており，ネットワーク機器とネットワーク管理アプリケーションの両方が NETCONF を実装することにより，互いの会話が成立することになる．また，複数のマネージャが同時に設定を試みた際に衝突を回避する仕組みとして，コンフィグレーションのロック/アンロック機能を規定している．ただし，設定対象に関しては，今後議論がなされる見込みである[9)]．

ネットワーク機器に NETCONF を実装する際は，トランスポートプロトコルの選択を検討する必要がある．NETCONF では，NETCONF プロトコルを運ぶ下位トランスポートプロトコルとして，BEEP，SOAP，SSH を規定している．ただし，ネットワーク機器の自動運用や複数の管理アプリケーションからの制御を実現するならば，下位トランスポートプロトコルには SOAP を用いるのが最適である．なぜならば，SOAP はネットワーク経由で複数のアプリケーションを連携可能にする Web サービスのメッセージ仕様だからである[10)]．これによって，NETCONF を用いたネットワーク管理アプリケーションを他システムと連携させることも容易となる．

2.2 データモデル

NETCONF は，ネットワーク機器とネットワーク管理アプリケーションとの間の通信については規定しているが，どの対象をどう管理し制御するかは規定していない．たとえば，LAN スイッチが備える VLAN 機能や QoS 機能の設定をどのように変更するのかは決めていない．これらの機能の設定手順や方法を機器に依存しない形で共通化する必要があり，それが「モデル化」である．モデル化された設定項目が，「データモデル」である．

モデル化の具体例として VLAN のモデルを**図 2** に示す．図 2 のように UML（Unified Modeling Language）を用いてモデル化することによって，それぞれの設定項目の関係が視覚的に明確となり，管理アプリケーションの開発もしやすくなる．モデル化されたフォーマットに則ってネットワーク機器とネットワーク管理アプリケーションが会話することで初めて，ネットワーク機器の機種やベンダに依存することなく，1 つの言語でマルチベンダのネットワークを制

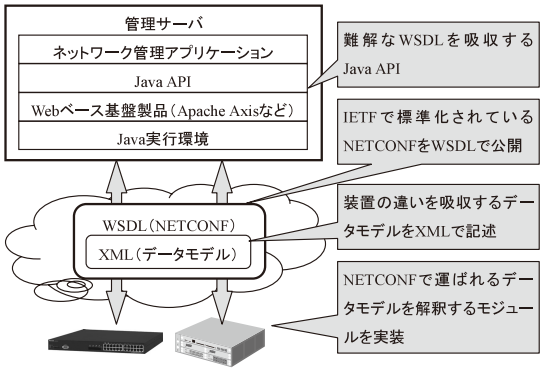


図 3 Java API と WSDL の関係
Fig. 3 Relation between Java API and WSDL.

御できるようになる．モデル化されたフォーマットは，コンピュータからの可読性が高い XML で記述する．XML で記述することによってデータが階層化され，コンピュータに理解しやすい設定項目構造となる．

2.3 Java API

NETCONF の下位トランスポートプロトコルに SOAP を利用すると，**図 3** に示すようにネットワーク制御用インタフェースは WSDL（Web Service Description Language）で提供することになる．WSDL は，Web サービスに対するインタフェースを記述するための標準仕様で，XML で記述されているため，コンピュータに直接取り込んでネットワーク機器にアクセスさせることができる．

しかし現在，アプリケーション開発者にとって WSDL を直接編集してネットワーク機器を制御するのは簡単なことではない．理由は，WSDL を扱える開発者の数がまだ少なく，WSDL によるアプリケーション開発の土壌が整備されていないことである．そこで現在，WSDL から Java API を自動生成するツールが提供されており，このツールで生成された Java API を使うことによって，ネットワーク機器の制御ができるようになっている．Java API により，ネットワークの機能がオブジェクト化され，ネットワークの機能や複雑さが隠蔽され，ネットワーク技術者以外でもネットワーク管理アプリケーションを開発しやすくなっている[5]．

3. ON-API の開発

これまでに述べた「NETCONF」，「データモデル」，「Java API」を用いたネットワーク機器制御用インタフェースを実装し，ON-API（Open Networking — Application Programming Interface）と名付けた[11]．

ON-API はサーバにインストールすることによって利用が可能となり，そのサーバ上で開発したネットワーク管理アプリケーションは ON-API でネットワーク機器を制御する．**図 4** に ON-API を実装したネットワーク管理アプリケーションとネットワーク機器の構成を示す．ネットワーク機器においては，HTTP メッセージを受信するため HTTP デーモンを稼動させておく．また，HTTP メッセージ内に納められた NETCONF メッセージを HTTP デーモンから受信するため NETCONF デーモンを稼動させておく．NETCONF デーモン内には XML パーサを用意し，NETCONF メッセージを解析し，管理アプリケーションからの指示を認識する．それぞれのデーモンを実装するために要したメモリ量を**表 2** に示す．両デーモンを稼動させると，さらに 0.9 M バイトの動的メモリを消費した．

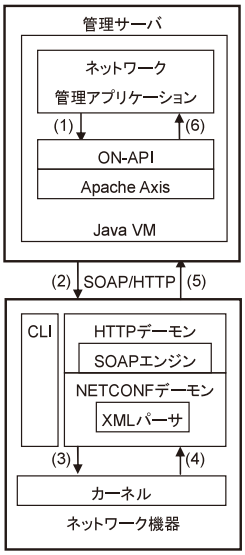


図 4 ON-API 実装サーバとネットワーク機器の構成
Fig. 4 Architecture of a server implementing ON-API and a network device.

表 2 各デーモン実装に要したメモリ消費量
Table 2 Memory consumption of each daemon.

デーモン名	メモリ量
HTTP デーモン	10.75M バイト
NETCONF デーモン	1.6M バイト

ON-API を用いて開発されたネットワーク管理アプリケーションが，ネットワーク機器を設定するシーケンスを，図 4 を用いて解説する．

(1) 管理アプリケーションからネットワーク機器の設定要求を発行する．
(2) 設定要求は NETCONF メッセージのフォーマッ

表 3 ON-API 機能一覧
Table 3 ON-API's functions.

機能	インスタンス変数
VLAN	VlanId, VlanName, etc.
Line	PortId, Speed, Name, etc.
Link Aggregation	LaId, LaType, etc.
Node	Name, Location, etc.
Filter	SourceIP, DestinationIP, etc.
Route	Destination, NexthopAddress, etc.

表 4 ON-API のメソッド例
Table 4 ON-API's method examples.

機能	変数・オブジェクト	設定・取得メソッド
VLAN	VLAN ID	setVlanid(short vlanid)
		getVlainid()
	VLAN Name	setVlanname(String vlanname)
		getVlanname()

トに整形され，SOAP エンベロープ内に納められ，サーバからネットワーク機器へ送られる．

(3) SOAP エンジン部で受信メッセージから SOAP エンベロープを外し，NETCONF メッセージを取り出す．NETCONF メッセージは NETCONF デーモンへ送られ，XML パーサで解析される．解析された結果に基づいてネットワーク機器の設定をする．

(4) 設定結果が NETCONF デーモンへ返され，NETCONF デーモンは戻り値を持った NETCONF メッセージを生成する．

(5) SOAP エンジン部は NETCONF メッセージを SOAP エンベロープに納めサーバへ返信する．

(6) アプリケーションは，戻り値に応じてあらかじめ設定しておいた処理を実行する．

3.1 ON-API の機能

現在，ON-API がサポートする機能と，各機能で設定可能なインスタンス変数例は，**表 3** に示すとおりである．

サポートする機能は，ネットワーク管理アプリケーションによって自動化できると望ましい機能からサポートした．エンタープライズネットワーク内で頻繁に変更がなされる VLAN 機能や，容易に多数の設定をする傾向にある Filter 機能などを優先的にサポートした．今後サポートしていく機能についても，ネットワーク管理アプリケーションによって得られる効果を見極めつつサポートするか否かの判断をしていくことになる．

設定項目については，管理アプリケーションで主に利用されると考える項目を検討し，各機能の動作に必要となるベーシックな項目からサポートすることにした．

アプリケーション開発のためには，表 3 に示した機能ごとの設定項目をモデル化（データモデルの規定）する必要がある．

3.2 モデル化

モデル化とは，図 2 に示す体裁を設計することである．図 2 は，表 3 に記載の VLAN のデータモデルである．アプリケーション開発者がアプリケーションを開発する際は，毎回このモデルに従ってプログラミングをすることになる．

3.3 ON-API のメソッド

開発ツールを使うことによって，図 2 のデータモデルから自動で Java API を生成できる．**表 4** に，図 2 のデータモデルから自動生成されたメソッドの例を示す．これらのメソッドを利用することによって，VLAN の各インスタンス変数を設定，参照することができる．インスタンス変数の値を設定する場合は set 系メソッドを，値を参照する場合は get 系メソッドを使用する．たとえば，VLAN ID が 100 の VLAN を作成したい場合，setVlanid() メソッドに 100 を引数として設定し，setVlanid(100) とコールすればよい．ネットワーク機器にすでに設定されている VLAN の VLAN ID 値を参照したい場合，getVlanid() メソッドを利用すればよい．

これらのメソッドを利用することによって，従来よりも容易に「運用の自動化」が可能で，「IT システムとの親和性の高い」ネットワーク管理アプリケーションが開発できるようになる．

4. ON-API を用いたアプリケーション開発

実際に，ON-API を利用したネットワーク管理ツールとして AXCM（AX Config Master）を開発した．AXCM は GUI（Graphical User Interface）でネットワーク機器を管理できるネットワーク管理ツールである．AXCM の GUI 画面を**図 5** に示す．提供している機能は以下である．

(1) 構成管理

ネットワーク上の機器，PC をグラフィカルに一元管理することができ，ネットワークの全体構成を容易に把握できる．

図 5 AXCM の GUI 画面
Fig. 5 GUI of AXCM.

(2) コンフィグレーション管理
各装置の持つコンフィグ情報を管理することができ，障害時のコンフィグレーションダウンロードや，機器変更時のコンフィグレーションのアップロードが容易にできる．

(3) 障害管理
装置や回線の障害を検知することができ，その状態をネットワークマップに表示できるので，障害箇所を容易に特定できる．

(4) 性能管理
装置の性能情報を表示することができ，装置の稼動状況をリアルタイムで監視することができる．

ON-API を用いて開発されたネットワーク管理アプリケーションからネットワーク機器へ送られる設定要求メッセージは，**図 6** のような形式となる．

設定要求メッセージは，初めに XML 宣言部があり，SOAP エンベロープ，SOAP ボディと続く．SOAP ボディ内に <rpc>，</rpc> タグがあり，NETCONF メッセージは，この <rpc> と </rpc> に挟まれた部分である．図 6 のメッセージには，NETCONF でコンフィグレーション変更用のタグとして定義された <edit-config> タグが使われており，このメッセージがコンフィグレーションの変更を要求していることが分かる．図 6 の場合，アプリケーションは，VLAN ID が 100 の VLAN ID を作成し，ポート 0/7 とポート 0/11 をポート VLAN 用のポートとして設定しようとしている．

以上のようにして，Java API で作成したネットワーク管理アプリケーションが，XML で記述された NETCONF メッセージを生成，送信し，所望の設定を施す．

5. 評　　価

ON-API を用いて開発した管理アプリケーションと CLI とで，それぞれ同じネットワーク機器を設定し，それぞれが要した時間を比較した．ここで，CLI を用いた管理アプリケーションは，いかようにも複雑化することが可能であるため，CLI を用いて開発した管理アプリケーションと ON-API を用いて開発した管理アプリケーションとの比較はしない．設定時間を比較するにあたっては，CLI を用いて開発した管理アプリケーションよりも負荷の少ない CLI のみによる設定と比較することによって，ON-API を用いて開発した管理アプリケーションが CLI に比べて遜色ない数値を示せば，実運用に十分耐えうると証明されるからである．比較に用いた処理は，1) コンフィグレーションファイルの一括送信，2) Filter リストの 1,000 件登録である．1) のコンフィグレーションファイルの一括送信は，コンフィグレーションを全変更する際に使用する機能である．2) の Filter とは，ネットワーク装置を通過するパケットを，IP アドレスやポート番号によっ

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc message-id="395">
   <edit-config xsi:type="ns1:editConfigType"
    xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
     <running xmlns=""></running>
    </target>
    <config>
     <ns2:Vlans xmlns:ns2="urn:net:alaxala:oan:onapi:commons:netmod:1.0">
      <ns2:Vlan operation="delete">
       <VlanId xmlns="">100</VlanId>
       <VlanName xmlns="">VLAN0100</VlanName>
       <UntaggedPort xmlns="">
        <PortId>port 0/7</PortId>
        <PortId>port 0/11</PortId>
        <Type>UNTAGGED_PORT</Type>
       </UntaggedPort>
      </ns2:Vlan>
     </ns2:Vlans>
    </config>
   </edit-config>
  </rpc>
  </rpc>
 </soapenv:Body>
</soapenv:Envelope>
```

図 6 アプリケーションからの設定要求メッセージ
Fig. 6 Request message from application.

て中継または廃棄する機能であり，不正アクセスや情報漏洩を防ぐなどの目的のために利用される．Filter リストは，容易に多数の設定をする傾向にあり，また複数装置間で同じ Filter リストを共有することも多い．セキュリティアプライアンスなどから Filter リストを多数設定するケースも想定されうるため，実験を試みた．

CLI の場合の設定方法は，1) については TFTP（Trivial File Transfer Protocol）サーバからコンフィグレーションファイルを送る方法，2) については，TeraTerm のマクロ機能を利用して telnet 上でコマンドを 1 つ 1 つ自動投入する方法である．ON-API の場合，1)，2) を実行するための管理アプリケーションを作成した．1) 用に作成した管理アプリケーションは，コンフィグレーションファイルを Running.cnf というファイル名で保存しておき，それを copyConfig() という ON-API の引数にし 1 回コールする．2) 用に作成した管理アプリケーションは，ON-API の利点が複数分のコマンドを配列に持たせて一括登録できることであるため，CLI のようにコマンドを 1 件 1 件登録するのではなく，Filter リストを 100 件ずつ配列にし，editConfigMerge(FlowDescInfo[], FlowId) という ON-API を 10 回コールする．結果を**表 5** に示す．

表 5 CLI と ON-API との評価
Table 5 Evaluation of CLI and ON-API.

評価項目	CLI	ON-API
1)コンフィグレーションファイルの送信	2 秒	22 秒
1)のコード量	—	50 行
2)Filter リストの 1,000 件登録	8 分 23 秒	1 分 25 秒
2)のコード量	—	150 行

1) については，CLI はアプリケーション処理がないため，ON-API よりも早く設定が完了した．それでも，両設定が短時間で終了しているのは，両設定とも 1 回のトランザクションしか発生していないからである．一方 2) については，CLI はアプリケーション処理がなかったにもかかわらず，ON-API の方が，処理が速かった．これは，CLI の場合 1 コマンドの応答が約 0.5 秒でシーケンスが 1,000 回発生するのに対し，ON-API の場合シーケンスが 10 回しか発生しなかったからである．以上により，ON-API を用いて開発したネットワーク管理アプリケーションが CLI と比較しても遜色なく，十分に実運用に耐えうることが分かる．

なお，SNMP は主に情報収集のために利用されており，設定用の MIB が充実していない．今回実験に用いたネットワーク装置も，Filter 関連の MIB としては，Filter に合致したパケット数をプライベート MIB としてサポートしているのみである．今回 SNMP と ON-API を比較する項目がないため，SNMP と ON-API との評価は見合わせた．

6. 考　　察

ON-API で開発したネットワーク管理アプリケーションを利用することによって，ネットワーク管理者がネットワークを運用するために要する時間は短縮可

能である．IT システムの変更にともなうネットワークの更新も迅速となる．

また，図 5 の AXCM のようなネットワーク管理アプリケーションを使用すれば，ネットワークの構成のみでなく障害発生箇所の特定も視覚化されるため，敏速な障害対応も可能となる．この効果は，ネットワークの規模が大規模になればなるほど顕著である．これによって，ネットワークの運用コストも低減可能である．

ON-API は，今後次のような応用が可能である．ON-API は SOAP を用いているため，Web サービスのフレームワークの中で，サーバ，ストレージと連携させてネットワークを管理することも可能である．

7. おわりに

従来のネットワーク機器制御方式として CLI や SNMP が主に使われてきたが，今後の大規模ネットワーク運用には力不足である．そこで，新たなネットワーク機器制御方式として ON-API を提案した．CLI や SNMP を用いてネットワーク管理アプリケーションを作成するのではなく，「NETCONF」，「データモデル」，「Java API」の技術を採用した ON-API を利用することによって，「運用の自動化」，「IT システムとの高い親和性」を実現するネットワーク管理アプリケーションが容易に開発できるようになる．ON-API と CLI を用いてそれぞれネットワーク機器を設定させ，それに要する時間を計ったところ，ON-API の方が CLI よりも短時間で設定が済むケースがある．これにより，ON-API で開発した管理アプリケーションが十分に実運用に耐えうるものであることを明らかにした．このようにしてネットワーク管理アプリケーションを開発し利用することによって，これまで個人の「ノウハウ」や「経験」によって運用してきた部分をアプリケーションへ置き換えていくことが可能となる．

今後の ON-API の課題として，NETCONF 上で転送するデータモデルの標準化がある．現在著者らは，VLAN，Filter のデータモデルを IETF に提案中である[9),12)]．

参考文献

1) Mi-Jung, C., et al.: XML-based configuration management for IP network devices, *IEEE Communications Magazine*, Vol.42, No.7, pp.84–91 (2004).
2) 宮本貴朗，田村武志，鈴木亮司：大規模ネットワークにおける VLAN 管理システム，情報処理，Vol.41, No.12, pp.3234–3243 (2000).
3) 長田智和，谷口祐治，玉城史朗：大規模分散ネットワーク運用管理システムの提案，情報処理学会研究報告，Vol.2000, No.113, pp.31–36 (2000).
4) Gerhard, M., et al.: Using Netconf for Configuring Monitoring Probes, *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pp.1–4 (2006).
5) George, P., et al.: On management technologies and the potential of web services, *IEEE Communications Magazine*, Vol.42, No.7, pp.58–66 (2004).
6) Aiko, P., et al.: XML-Based Management of Networks and Services, *IEEE Communications Magazine*, Vol.42, No.7, pp.56–57 (2004).
7) IETF, Network Configuration (Netconf). http://www.ietf.org/html.charters/netconf-charter.html
8) Rob, E.: NETCONF Configuration Protocol, RFC4741 (2006).
9) Tomoyuki, I., et al.: VLAN data model for NETCONF, draft-iijima-ngo-vlandatamodel-00, work in progress (2007).
10) Ted, G.: Using NETCONF over the Simple Object Access Protocol (SOAP), RFC4743 (2006).
11) Tomoyuki, I., et al.: Development of Management Interface to Configure Network Equipment, *Proc. IEEE-CS/IPSJ SAINT 2007*, Hiroshima, Japan (2007).
12) Tomoyuki, I., et al.: ACL data model for NETCONF, draft-iijima-ngo-acldatamodel-00, work in progress (2007).

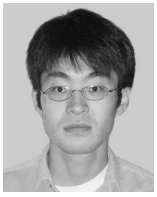
(平成 19 年 5 月 31 日受付)
(平成 19 年 12 月 4 日採録)

飯島 智之

平成 11 年早稲田大学理工学部電気電子情報工学科卒業．同年（株）日立製作所入所．ルータの研究開発に従事．平成 19 年（株）アラクサラネットワークスに出向．以来，ネットワーク管理システムの研究開発に従事．現在，同社技師．電子情報通信学会会員．

新　善文（正会員）

平成 1 年福井大学工学部電気工学科卒業．平成 3 年同大学院工学研究科電気工学専攻修了．同年（株）日立製作所入所．平成 16 年（株）アラクサラネットワークスに出向．現在，同社主任技師．



木村　浩康

平成 10 年横浜国立大学工学部電子情報工学科卒業．平成 12 年同大学院工学研究科電子情報工学専攻修了．同年（株）日立製作所入所．平成 16 年（株）アラクサラネットワークスに出向．以来，ネットワーク管理システムの研究開発に従事．現在，同社技師．

木谷　誠

平成 11 年大阪大学工学部情報システム工学科卒業．平成 13 年同大学院工学研究科情報システム工学専攻修了．同年（株）日立製作所入所．平成 16 年（株）アラクサラネットワークスに出向．以来，ネットワーク管理システムの研究開発に従事．現在，同社技師．