NETCONFによるセキュリティアプリケーションの実装

—次世代エンタープライズネットワーク統合管理実現への実践的アプローチ—

An Implementaion of Security Application by NETCONF

—A Practical Approach of Integrated Management on Next Generation Enterprises Network—

千装 俊幸 † 知念 賢一† 宇多 仁‡ 篠田 陽一‡

概 要

近年、ネットワーク機器の新しい基本管理手法としてNETCONFプロトコルが注目されている。NETCONFはルータやスイッチといったネットワーク機器の情報取得と設定インタフェースを標準化するプロトコルであり、ベンダや機種が異なるネットワーク機器を統一的に管理することを可能とするプロトコルとして、活用が期待されている。近年、操作のためのAPIを備えたNETCONF対応製品が市場投入され、ユーザによるアプリケーションの作成も可能となってきている。本論文は実際にNETCONF対応APIを使用したアプリケーションの製作を行い、NETCONFプロトコルの有用性や課題についての考察を行ったものである。

1 はじめに

本研究は、近年注目されているNETCONFプロトコルについて、実際にNETCONF対応APIを用いたアプリケーションを試作し考察を行ったものである。NETCONFについてはその期待が大きく活用が想定されている範囲も広い半面、現状どの程度まで標準化が進んでいて、対応製品を用いればどのようなことが実現可能なのかという実像が見えにくい部分がある。

本論文では、まずNETCONFプロトコルについての概略やNETCONFを構成するレイヤ構造、NETCONFプロトコルが期待される分野について紹介する。すでにユーザが使用可能なNETCONF対応製品としてAX-ON-APIを取り上げる。

続いて先に取り上げたAX-ON-APIを使用したセキュリテリアプリケーションの試作について紹介する。本APIを用いることで従来の管理インタフェースに比してどのようなことが実現可能になるのか、従来のネットワーク管理手法に対してどのようなアプローチが可能になるのかを述べる。本論文ではその一例として、単機能アプライアンス機器の機能代替をNTCONF対応APIを用いて試作した「不正接続PC排除システム」を試作した。試作システムの紹介では、プログラムのコードについて抜粋し、実際にAPIをどのように使用したのかを例示も行う。

最後に試作を通しての考察やその過程で得たNETCONF APIについての知見を述べる。またNETCONFの有用性や可能性、そして見えてきた課題について考察を行う。

†北陸先端科学技術大学院大学 情報科学研究科
‡北陸先端科学技術大学院大学 情報科学センター

2 NETCONF

2.1 NETCONFとは

The Network Configuration Protocol (NETCONF)はネットワーク機器の情報取得と設定インターフェイスを標準化するプロトコルであり、RFC4741で定義されている[1]。NETCONFはネットワーク機器の運用管理の統一だけでなく、将来的にはPCやサーバなどのコンピュータを含むネットワークシステム全体の運用管理も視野に入れているプロトコルである。設定の

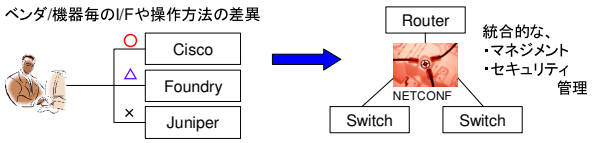

図 1: NETCONF による設定インタフェースの統合

送受信には XML ファイルを利用する。

従来からネットワーク機器が有する管理ツールには、CLI と SNMP がある。ネットワーク機器の設定にはコマンドラインベースの CLI の利用が現在主流であるが、CLI はベンダ独自のコマンド体系で作成されており、ベンダ毎、中には機器毎にコマンド体系が異なる場合もある。ユーザは扱う機器毎にコマンドを学習する必要がある。また監視手法に採用されている SNMP も統一化された標準 MIB の他にベンダ独自機能についてのベンダ独自 MIB が随時拡張されており、NMS 等による SNMP を使用した管理の追従を困難にしている。

これら従来の管理ツールの問題点を回避し、ベンダや機種を問わない管理手法の登場が期待されるのは当然と言える。統一化されたインタフェースが登場すれば、ベンダや機種を意識せずにネットワーク機器を設定・管理することが可能になる。統一化されたインタフェースの操作 API の提供により、ネットワーク機器をプログラムから操作することが可能になり、プログラムによる運用の自動化や障害の自動復旧などへの活用が期待されている。NETCONF によるインタフェース統合のイメージを図 1 に示す。

2.2 IETF での標準化作業

2007 年末時点で策定がほぼ終了しているのは、NETCONF が想定するレイヤ「トランスポート・プロトコル層」、「RPC 層」、「Operations 層」、「Content 層」のうち下位 3 層である。NETCONF プロトコルが想定する階層構造を図 2 に示す。トランスポート・プロトコルは XML を送受信するためのプロトコルレイヤであり、HTTP、SOAP、BEEP などを使用する。その上位に NETCONF インタフェースの呼出を規定する RPC 層とネットワーク機器に対するコマンドを操作である Operations 層がある。

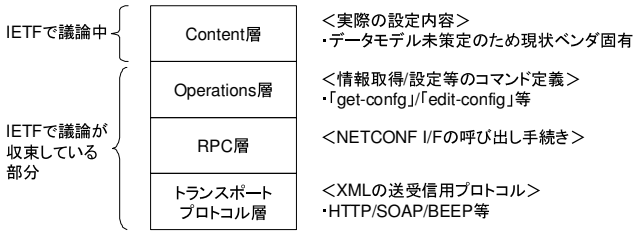

図 2: NETCONF プロトコルの階層構造 [2]

最上位の Content 層は実際にネットワーク機器が実行する設定内容を記述する層である。しかしながらネットワーク機器が持つ機能を規定するデータモデル（記述形式）がまだ未策定のため、ベンダ固有の設定コマンドがそのまま使用されていることが多い。

3 AX-ON-API

2006 年末ごろから、NETCONF に対応しソフトウェア開発のための操作 API を提供するベンダも登場してきている。今回は実際にアプリケーションを試作するために、アラクサラネットワークス株式会社が提供する NETCONF 対応 API である AX-ON-API を取り上げる。

AX-ON-API(AX-Open Networking-Application Programming Interface) は、アラクサラネットワークス社製品の運用・管理の自動化を支援する OAN(Open Autonomic Networking) の基盤技術である [3]。NETCONF を採用しており、通信手段には SOAP を採用している [4]。SOAP メッセージの交換保護に HTTPS を使用することも可能になっており、SOAP メッセージの交換によるネットワーク機器操作をセキュアに行うことも可能である。OAN-API1.3 で操作することのできる機能を次に挙げる。

- Node 操作。ノードの IP アドレスについての参照・変更 API。
- Line 操作。ノードのポート、インタフェースについての参照・変更 API。
- AbstractionPort 操作。リンクアグリゲーションについての参照・変更 API。
- VLAN 操作。VLAN についての参照・変更 API。

- AccessList IFAccessList 操作。アクセスリストについての参照・変更 API。
- IpRouteStatic 操作。スタティックルーティングについての参照・変更 API。
- MacAddressTbl 操作。MAC アドレステーブルについての参照 API。
- NodeStatus 操作。装置情報 (装置のモデル名、ソフトウェアバージョン) の参照 API。

AX-ON-API はアラクサラネットワークス社製品を制御可能とする Java クラスライブラリ集であり、Java によるプログラミングインタフェースを提供する。

4 API を用いたアプリケーションの試作

本研究では NETCONF を採用した AX-ON-API を使用しセキュリティアプリケーションを試作した。今回本研究ではネットワークに不正接続する PC を NETCONF の枠組を用いて排除するシステムを作成した。作成するアプリケーションに不正接続 PC 排除システムを選んだ背景、および本システムの構成について以下に述べる。

4.1 不正接続 PC 排除アプライアンス

今日のエンタープライズネットワークでは、多様化し続ける管理上の要求に応えたり、高度化するセキュリティ問題に対する解法として種々の「ピンポイント解決」を行う単機能アプライアンス機器が導入されることが多くなっている。ネットワーク機器を統合的に運用・管理することの困難さにより、原理的にはルータやスイッチ等の基本的なネットワーク機器内に実現することが最も効果的かつ効率的であるにも関わらず、単機能アプライアンス機器による実現が行われている。

先にあげた単機能アプライアンス機器の例として、不正接続 PC 排除アプライアンスを取り上げる。

端末・ユーザを認証し LAN への不正な接続禁止する認証技術として代表的なものに IEEE802.1X 認証技術がある。IEEE802.1X 認証が RADIUS サーバ、IEEE802.1X 認証対応スイッチ・AP、およびクライアント端末にインストールするサプリカントから構成されるのに対し、不正接続 PC 排除アプライアンスはアプライアンス機器一台で不正接続 PC を排除する機能を有するアプライアンス製品である。IEEE802.1X の導入コストに比べ、不正接続 PC 排除アプライアンスの導入は安価であり、設置も当該アプライアンス機器を LAN に接続するだけでよく、広く普及してきている。

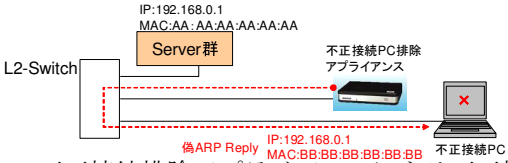

図 3: 不正接続排除アプライアンスによる不正接続排除の仕組

いくつかの代表的な不正接続 PC 排除アプライアンス製品を調査したところ、不正接続 PC 排除アプライアンスは不正接続 PC の排除に ARP Poisoning という手法を用いていた。ARP Poisoning はスニフィングの手法として知られている。不正 PC 排除システムはネットワークを流れる ARP Request パケットを監視し、あらかじめ自身に登録されている正規の PC が持つ MAC アドレス以外からの ARP Request パケットを観測するとネットワーク上に存在しない MAC アドレスを埋め込んだ偽の ARP Reply を Request 元に送信する [5]。これにより、不正接続 PC の ARP テーブルを汚染し、通信不能状態にすることで不正接続 PC を排除する。不正接続 PC 排除アプライアンスによる ARP Poisonig を使用した不正接続 PC 排除の流れを図 3 に示す。

4.2 不正接続 PC 排除アプライアンスの問題点

不正接続 PC 排除アプライアンスが用いている ARP Poisoning による不正接続 PC の排除手法にはいくつかの問題点がある。

- ARP による解法を図っているため、セグメントを越えて端末を管理することができない。
- ネットワーク内に複数のセグメントが存在する場合は、セグメントごとにアプライアンスを設置する必要がある。

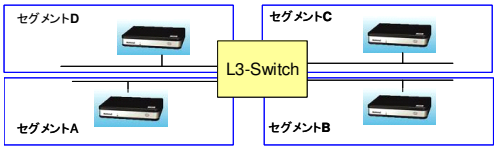

図 4: セグメント毎の不正接続排除アプライアンスの設置

- 不正接続端末が「ping 255.255.255.255」などのネットワーク内の MAC アドレスを一網打尽に取得するような攻撃をとった場合、偽 ARP Reply の生成ができない。これは偽 ARP 生成の際にどの IP アドレスの偽 ARP を生成してよいか不正接続 PC 排除アプライアンスが判断できないためである。

これらのうち、複数セグメントをカバーするために設置台数を増加させることはネットワークの複雑化・ノンマネンジメント化を招く。また排除方法にスニフィングで用いられる手法を用いる点も、セキュリティ製品としての側面から好ましくない。不正接続 PC 排除アプライアンスの設置ケースを図 4 に示す。

4.3 不正接続 PC 排除システム

ある端末を通信不可状態にするという目的だけであれば、Layer2 スイッチによる MAC アドレスフィルタリング機能において簡単に代替可能である。先の不正接続 PC 排除アプライアンスの手法で述べたようなジャミング攻撃を使うのではなく、より簡潔でかつ効率的・効果的な手段であると言える。

しかしながら Layer2 スイッチのようなネットワーク機器に、ユーザがフィルタリング等設定を施すためには専用 CLI 等の操作が必要であり、オペレーションの困難さ・繁雑さの障壁が存在する。これはネットワーク機器に機能はありながらも活用されていないという現状の要因となっている。専用 CLI を用いて手動で複数機器に設定を施す作業は、作業量も多くかつミスも起こりやすい。特に今回機能として使用する MAC アドレスフィルタリングを設定するためにはは 48bit の MAC アドレスを個々間違えることなく、指定されたフォーマットで設定する必要がある。このような手間や繁雑さが単機能アプライアンス機器が好まれる理由でもある。逆に言えば、この専用 CLI 操作の手間や、繁雑さという幾多の障壁を何らかの方法によって回避できれば、ネットワーク機器の機能を活用した手法の採用が促進されると考える。

本研究ではこれらの障壁を回避する方法として、NETCONF API を使用したアプリケーションを製作し不正接続 PC の排除を行うシステムを試作した。

- layer2 スイッチによるフィルタリングを行うため、不正接続 PC 排除のための新たな機器を設置する必要はなく、各スイッチで端末を管理することができる。
- NETCONF による集中統合管理を行うためマネージャの設置は分散配置する必要がなく、スイッチへの reachability がある限り 1 箇所でよい。
- 不正接続端末の MAC アドレスを送信元 MAC アドレスとしてフィルタリングの条件にすることで通信先を意識した排除方法を懸念する必要がない。
- アプリケーションにて GUI 等を用いてユーザの入力作業を助け、作業負荷を低減し、業界ルールを回避・補助する仕組を提供する。

また、ある端末が不正接続 PC であるか否かの判定条件として、不正接続 PC 排除アプライアンスは「MAC アドレスが登録されていない端末 = 不正接続 PC」として定義していた。スイッチによるフィルタリングを行う場合でも、正規の PC の MAC アドレスをあらかじめスイッチに登録しておくことで同様の判定条件とすることは容易に可能である (ホワイトリストによる管理)。本研究で提案する NETCONF による統合管理でももちろん可能であるが、今回は新しい試みとして「DHCP サーバにより割り当てられた IP アドレスを使用していない端末 = 不正接続 PC」という判定条件手法を採用した。この判定条件を使用することで、DHCP サーバの運用と連携した管理や認証 DHCP 方式との連携等も可能になる。図 5 に不正接続排除システムの概要を示す。スイッチから取得した MAC アドレステーブルをリスト化して表示。取得したリストを対象リスト (今回は DHCP サーバから取得した dhcpd.leases と比較した場合を示す) と照合し、リストにある端末であるかどうかを判断する。不正接続 PC と判断された端末の MAC アドレスはスイッチにアクセスリストとして加えられる。今回開発したアプリケーションでは、

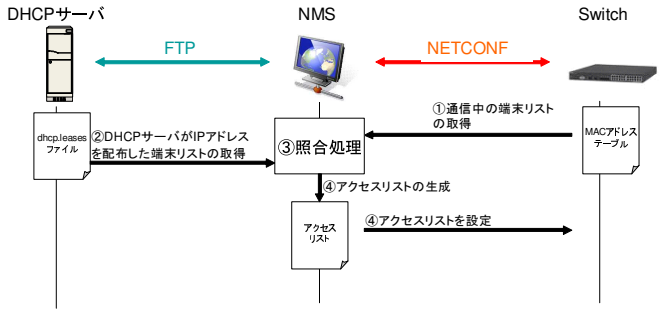

図 5: 不正接続 PC 排除システムの動作概要

通信中の端末が不正接続 PC であるか否かを判断を、DHCP サーバの持つ dhcpd.leases ファイルと照合を行うことで実現した。今回実験で用いた DHCP サーバは isc-dhcpd-v3.04 である。dhcpd.leases ファイルの例を以下に示す。

```
lease 192.168.0.132 {
starts 5 2008/03/15 12:56:40;
ends 6 2008/03/1600:56:40;
hardware ethernet 00:90:cc:a2:f9:36;
client-hostname"dhcp-1";
}
lease 192.168.0.135{
starts 5 2008/03/15 16:22:56;
        ends 5 2008/03/15 18:02:56;
hardware ethernet 00:90:27:18:ef:89;
        uid 01:00:90:27:18:ef:89;
client-hostname "dhcp-2";
}
lease 192.168.0.133 {
starts 5 2008/03/15 15:21:50;
ends 5 2008/03/15 17:01:50;
hardware ethernet 00:90:cc:a2:f9:41;
uid 01:00:90:cc:a2:f9:41;
client-hostname "dhcp-3";
}
lease 192.168.0.130 {
starts 5 2008/03/15 15:45:43;
ends 5 2008/03/15 16:26:52;
hardware ethernet 00:90:cc:a2:f4:e8;
uid 01:00:90:cc:a2:f4:e8;
client-hostname "dhcp-4";
}
```

今回は dhcpd.leases ファイルを用いたが認証 DHCP を運用している環境では dhcpd.conf を用いることも考えられる。AX-ON-API を用いてプログラムからレイヤ 2 フィルタを設定する際のサンプルコードを以下に示す。

```
private static FlowLayer2Standard[] make_FlowLayer2() {
// フロー条件オブジェクトの生成
   FlowLayer2Standard[] flowListObj = new
   FlowLayer2Standard[1];

   // MAC Address Filter 1(MAC Address = AA.AA.AA.AA.AA.AA)
   FlowLayer2Standard flowObj = new FlowLayer2Standard();
   flowObj.setSequence(4001);
   Action action = new Action();
   Filter filter = new Filter();
   filter.setType( Constants.FILTER_DENY );
   action.setFilter(filter);
   flowObj.setAction(action);
   flowObj.setSourceMac(‘‘AAAAAAAAAAAA/000000000000’’);
   flowObj.setDestinationMac(Constants.MAC_ANY);
   flowListObj[0] = flowObj;

   // MAC Address Filter 2(permit any any)
   flowObj = new FlowLayer2Standard();
   flowObj.setSequence(7000);
   action = new  Action();
   filter = new Filter();
   filter.setType(Constants.FILTER_PERMIT );
   action.setFilter(filter);
   flowObj.setAction(action);
   flowObj.setSourceMac(Constants.MAC_ANY);
   flowObj.setDestinationMac(Constants.MAC_ANY);
   flowObj.setEthernetType(null);
   flowListObj[1] = flowObj;

return flowListObj; }
```

プログラムによるフィルタ設定指示から実際にフィルタが有効になり通信が遮断されるまでの時間を複数回測定したところ、平均値は 24 秒であった。

5 アプリケーション試作における知見・考察

今回実際に使用可能な NETCONF API を用いてアプリケーションを試作することで、現状 NETCONF がどの程度使用可能な状況になっているのか、使用感とともに実感することができた。以下、いくつかの事項を挙げる。

5.1 CLI を使用するプログラムとの比較

CLI のような従来の管理ツールによる管理は、機種が少なく、機器台数もあまり多くない環境下においてはオペレータによる作業もある程度可能であるが、現在のエンタープライズネットワークにおけるマルチベンダ構成・大規模ネットワーク環境下においては非常に困難である。CLI はオペレータが機器を 1 対 1 で設定することを想定したインタフェースであり、設定する機器台数が大量である場合や、障害発生時の復旧作業など、定型化された作業をプログラムから設定したいといった運用には不向きなインタフェースである。

CLI は人が確認しながらコマンドを投入することを想定しており、設定内容によってモードの遷移が必要であり、設定後の確認コマンドによる目視等を前提に作成されている。ある程度の単純作業であれば、シェ

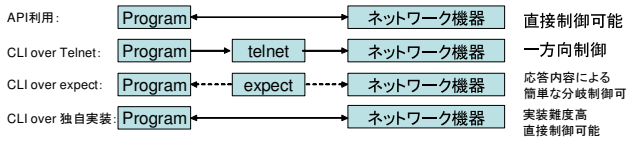

図 6: プログラムからネットワーク機器を操作する手法による双方向性の差異

ルスクリプト等を用いた定型作業のプログラム化も可能であるが、機器からの応答内容によって分岐や判断を必要とするような作業はプログラミングすることが難しい。telnet セッションの開始からモード遷移を伴う MAC アドレスフィルタ設定変更を CLI で行った場合のオペレーションイメージを以下に示す。本来のオペレーションではさらに設定変更後の show コマンドを用いた確認作業等が必要になる。

```
telnet 192.168.0.1
login: ***********
>enable
password:*********
#config terminal
(config)#flow detection layer2-1
(config)#mac access-listextended acl1
(config-ext-acl1)#deny host AAAA.AAAA.AAAA. any
(config-ext-acl1)#permit any any
(config)#int vlan 1
(config-if)#mac access-group acl1 in
```

今回は NETCONF 採用 API によるアプリケーションの試作を行ったが、その最大の利点はソフトウェアの作成が容易になるという点にある。API はプログラムから呼び出されることを想定していることはもちろん、ピンポイントに欲しい情報を取得したり、取得した情報を引数としてメソッドを起動することも記述しやすい。またエラー処理等の記述も行いやすい。図 6 に CLI を telnet や expect、独自のプログラムで利用した場合のプログラムと API を用いた場合の比較を示す。API はもともと情報の取得・設定変更がペアで用意されており、ネットワーク機器との双方向性が意識されて作成されている。telnet を使用して CLI を操作する場合は一方的にコマンドを送信することは可能であるが、ネットワーク機器側からの情報を取得して分岐を行ったり、動作を変更する記述をするにはかなり工夫が必要になる。expect を利用すれば簡単な条件分岐等は可能になるが、複雑な条件分岐を記述することは困難である。telnet に頼らず独自の通信プロトコルを作成して対応する場合も同様である。

5.2 インタフェース共存による問題

ネットワーク機器の機能は従来からある CLI によって操作されることを想定している。ここに新たに NETCONF API から操作されるインタフェースが加わることで両インタフェース共存による整合性問題が生じるケースがある。例えば本アプリケーションの試作で遭遇したのは、ネットワーク機器の業界標準ルールである「暗黙の deny all」によるアクセスリスト末尾への設定である。「暗黙の deny all」とはアクセスリストにより許可・廃棄を明示的に設定したもの以外は廃棄するという多くのネットワーク機器に共通する業界ルールである。これはアクセスリストの設定を CLI から行うオペレータの利便性を考えた措置であるが、NETCONF からアクセスリストの設定をおこなった際にもこれが適用される。プログラム作成者の意図しない動作を行ってしまう事態が発生する。今回は「permint any any」というフィルタルールをアクセスリストに追記するようプログラミングに加えることでで回避したが、他にも業界標準ルールは存在し、CLI での設定を想定している「暗黙の deny all」のような他の補助機能が NETCONF API による設定において意図と異なる動作を発生させる原因となる可能性がある。

5.3 NETCONF API の意義

CLI でのオペレーションに慣れているユーザにとっては、NETCONF API を利用した設定は、コーディング/コンパイル/アプリケーションの起動といったプロセスを必要とするため、あまり利便性を感じられないかもしれない。しかしプログラムから用いることができるインタフェースが使用可能になる意義は大きい。

プログラムからネットワーク機器を操作できるということは、ある機能やアプリケーションをネットワークに導入するにあたり、ネットワーク機器にすべてのコンポーネントを実装する単独で完結するようにすべての機能を実装する必要がないことを意味する。本試作システムにおいては、接続している端末が不正接続なのか正規の接続なのかは外部の機器 (NMS や DHCP サーバの管理ファイル) に委ね、フィルタリング機能だけをレイヤ 2Switch は備えていれば実現可能になる。システムやアプリケーションの判断処理機構をネットワークから分離して実装するケースを図 7 に示す。

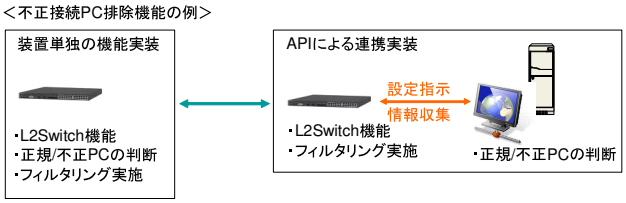

図 7: 実装における機能の分離

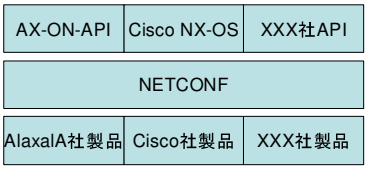

図 8: API がベンダ独自である現状

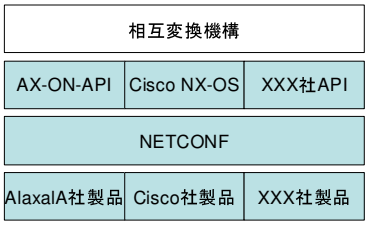

図 9: 相互変換層を設けた統合

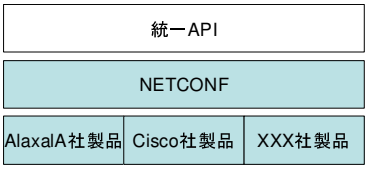

図 10: 統一 API による統合

5.4 NETCONF API によるネットワーク機器既存機能の有効活用

昨今のネットワーク機器は各種フィルタリング機能や情報取得機能等さまざまな機能を有している。しかしながらそれらの機能はあまり有効に活用されいない現状がある。逆にいえば有効活用できないともいえる。CLI による対人制御では使いにくい機能でも NETCONF API による制御では有効に活用できるケースがある。今回アプリケーションの試作に用いた MAC アドレスフィルタリング機能などはその典型といえる。

5.5 NETCONF API の相互互換性

NETCONF の最大の利点はベンダや機種を越えたインタフェースの統合にある。しかしながら現状の NETCONF API はベンダ間に相互互換性がない。先に述べた Content 層標準化の遅れに問題があり、ベンダが先行して独自に Content 層を実装しているためにやむを得ないという現状がある。現状の NETCONF API のイメージを図 8 に示す。Content 層の独自実装を許容しつつ、ベンダ間の API を統合して扱えるよう相互変換する層を用いた統合モデルを図 9 に示す。NETCONF は設定データの記述に XML を用いているために、標準となるデータモデルが策定されれば相互変換が可能となるとされている。しかしながらそれはユーザ側で相互変換モジュールを結局自作する作業が必要になり、その分だけ利便性が損なわれる。著者としては、統一 API の出現を望む。ベンダが独自機能により自社製品の特徴を打ち出していく以上、機能差異は存在し統一 API の実現が困難であることは理解できる。しかしながらネットワーク機器として最低限備える機能や共通の理念を採用している機能等、論理積を取れる部分があると思われる。統一 API による統合イメージを図 10 に示す。ユーザにとってはこれこそが NETCONF の恩恵を最大限に受容できる姿だと思われる。

6 まとめ

最後に本研究における知見・考察をまとめる。

NETCONF API は CLI や SNMP を駆逐するものではなく、今後も CLI や SNMP などのインタフェースとともに適材適所の使用方法が検討されていくインターフェイスである。また現状の製品化されている NETCONF は発展段階であり、ベンダ固有の API としても当該機器が有する機能をすべて API から操作できるわけではなく、設定変更頻度の高いものから実装が着手されているようである。先に述べたベンダ間の相互互換性についてもデータモデルの標準化が見えてきた後に検証される事項として残っている。

しかしながらプログラムからネットワーク機器を操作するインタフェースとして API が提供される意義は大きく、CLI をプログラムから操作するよりも開発が容易になる。

CLI からの操作に不向きであった既存機能も API から操作することにより価値を発揮するケースも出てくる。本論文で試作したシステムのように、API を通し

てネットワーク機器が他のシステムと連携した動作を行うことも可能になる。また、同時複数機器操作のように一対多のオペレーション操作を可能とするアプリケーション開発への活用も期待される。

NETCONF API はネットワーク機器への設定操作だけでなく、ネットワーク機器からの情報取得を行う API も用意されている。例えばネットワーク上に設置されているネットワーク機器から情報を吸い上げトポロジーを把握するような使いかたも考えられる。

各ベンダの実装努力による API の高機能化・利便性向上とともに、やはり各ベンダ NETCONF API の相互互換性やインタフェースの統一化に期待したい。

謝辞

最後に、本共同研究に関して多大なご協力を賜わりました皆様に感謝します。本研究を進めるにあたり、アラクサラネットワークス株式会社の新喜文氏、木村浩康氏、木谷誠氏には AX-ON-API についての質問のやり取りを通じて大変御世話になりました。NICT 技術員の佐野 正行氏には環境の構築、テスト機器の設定等大変御世話になりました。ありがとうございました。

参考文献

[1] R.Enns and Ed. **NETCONF Configuraion Protocol,RFC4741**, 2006

[2] 日経コミュニケーション 2006.6.15 **P80-85「企業を熱くする最新テクノロジ NETCONF」**, 2006.

[3] ALAXALA Networks Corporation **OAN ユーザーズガイド AX-ON-API 編**, 2007.

[4] T.Goddard. **Using NETCONF Protocol over the Simple Object Access Protocol(SOAP),RFC4743)**, 2006.

[5] 松谷健史 **情報処理学会 第 12 回マルチメディア通信と分散処理ワークショップ「ARP を利用したローカルエリアネットワークにおける不正接続の排除」**, 2004.