3 ダイナミックネットワーク制御技術
—NETCONF-API の活用によるネットワーキングの適用拡大—

本稿では，現状の NETCONF/NETCONF-API の「実物」を読者に見ていただきたい．ここでは NETCONF 対応製品を利用したアプリケーションの試作を通して，NETCONF/NETCONF-API の利用の仕方を，またこれらを活用したネットワーキングの応用／適用範囲の拡大可能性について解説する．具体的には，まず最初に NETCONF および NETCONF-API についての簡単な紹介を行う．次に実際に対応製品を利用し，NETCONF-API による制御を行うプログラムのサンプルソースを挙げ，読者に利用方法のイメージを提供する．そしてさらに応用として NETCONF/NETCONF-API の利点を活かしたシステム例を紹介し，NETCONF/NETCONF-API の活用によるネットワーキングの応用／適用範囲の拡大可能性について述べる．

千装俊幸
(株)クーピー 技術部

NETCONF/NETCONF-API

NETCONF/NETCONF-API に対する我々の期待は，
- 「1. ベンダ，機種の差異を問わないネットワーク機器操作方法の統一化」
- 「2. CLI，GUI に代わるネットワーク機器操作 API の提供」

である．これまでネットワーク管理に携わるオペレータは，同一の目的を達するためにベンダや機種ごとに異なるオペレーション方法を習得する必要があった．たとえば IP アドレスを設定するのに，ある機器はコマンドにより CLI（Command Line Interface）で設定し，他の機器は Web ブラウザから GUI（Graphical User Interface）で設定する必要があり，さらに CLI を利用する機器であってもそのコマンド体系が違っていたりと，オペレータは多カ国語を操る必要があった．また，多数の機器に大量の設定を施すためには，多くの人手を要し，ターミナルを複数立ち上げて作業を実施するなど，運用技術者は人海戦術やルーチンワークの繰り返し作業に追われていた（図 -1）．

しかしここに NETCONF および NETCONF-API が登場することで，先に挙げたような「多カ国語習得の苦役」からエンジニアが開放され，プログラムによるネットワーク機器の「一対多」，「自動制御」が可能になる未来が現

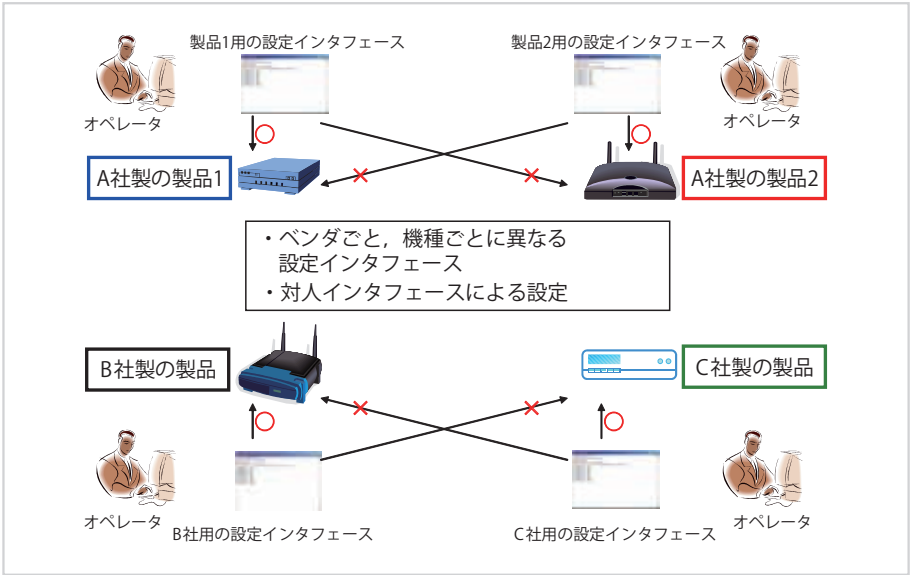


図 -1 ベンダ，機種ごとに異なる設定インタフェースによる人的運用

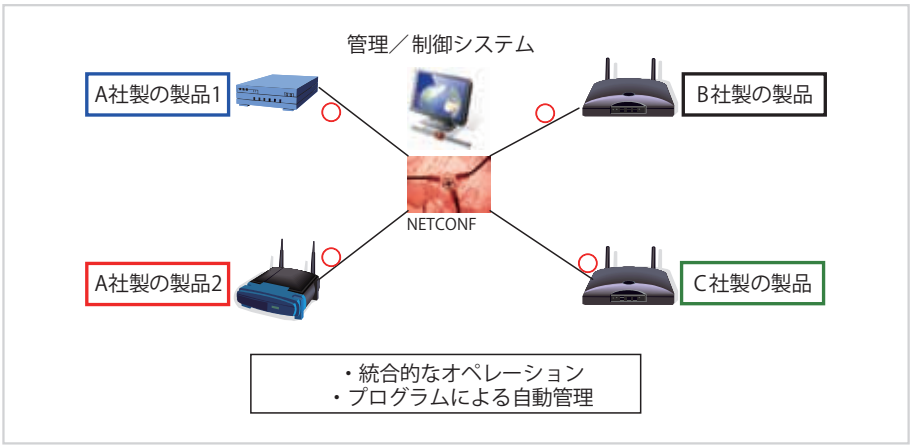


図 -2　NETCONF/NETCONF-API による統合的な自動運用管理

出しようとしている（図 -2）．

これらを実現すべく，市場にもユーザが利用可能な NETCONF 対応製品や NETCONF-API が登場してきた．しかしながら，

- 「NETCONF の現状」
- 「実際に使用可能な NETCONF-API」

についての情報は少なく，

- 「1. ベンダ，機種の差異を問わないネットワーク機器操作方法の統一化」
- 「2.CLI，GUI に代わるネットワーク機器操作 API の提供」

という前述した我々の期待を現在利用可能な NETCONF/NETCONF-API が満たすものなのか，その実態が掴みにくいものとなっている．そこで本稿では実際に NETCONF-API を使用したアプリケーションの試作を通じて NETCONF/NETCONF-API の実情に迫ってみたい．またそれらを利用することで，どのようなことが可能になるのか，システム例を挙げて解説する．次章ではまず NETCONF/NETCONF-API へ至る歴史の簡単な流れ，および NETCONF/NETCONF-API の現状について俯瞰する．

NETCONF 登場以前の試み

前章で述べたような，「ベンダや機種の差異を問わないオペレーション」，「人手によらないプログラムからのネットワーク機器制御」への希求は従来から存在した．これまでは，SNMP や CLI をプログラム上から援用する方法を用いていた．しかしいずれの方法も実装する側からは使い勝手の良いものではなかった．

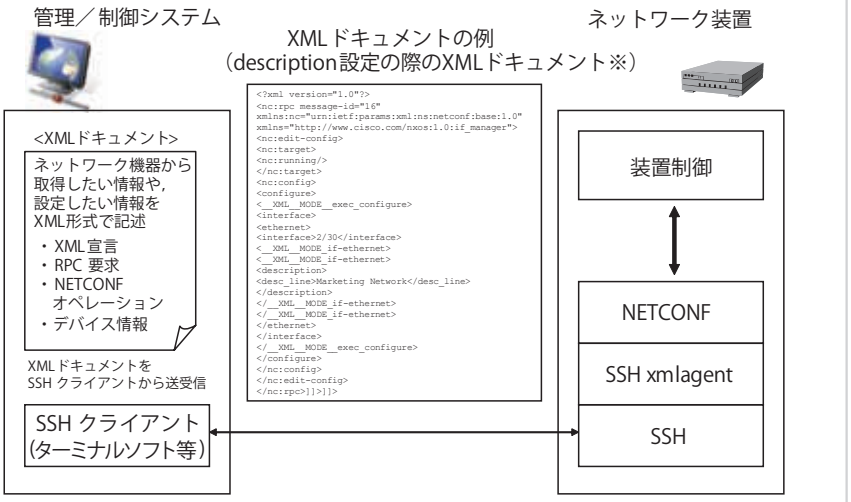
まず SNMP（Simple Network Management Protocol）であるが，SNMP は IETF により標準化され広く普及しているプロトコルである．SNMP による設定変更には，SNMP の SetRequest PDU による MIB（Management Information Base）の設定変更を用いる．しかしながらこれには深く細分化された MIB ツリーの値を，1 つ 1 つ設定していかなければならない．ただでさえ複雑な仕様であるところに，ベンダ独自 MIB といったベンダ固有の依存部分も存在する．複数のマネージャによる設定衝突を回避する仕組みもない．このような理由から SNMP は監視等の情報収集ツールとしては有効であるが，ネットワーク機器の制御を行うインタフェースとして使用するには不向きである．

次に CLI で使用するコマンドライン命令をプログラムから実行する方法についてであるが，当然 CLI によるコマンドライン命令はベンダが独自に開発しているものであり，統一されたものではない．また CLI はもともと人による対話的処理をもとに考案されていることから，プログラムで実行するには不向きなインタフェースである．プログラムから対話的オペレーションを行うためには，ネットワーク機器からの応答メッセージを解釈する構文解析機構が必要となる．エラー処理等を記述するにあたっては特に障壁となる．

では，NETCONF/NETCONF-API の登場によりこれらは解決されたのであろうか．次章では，前述した我々の期待「1. ベンダ，機種の差異を問わないネットワーク機器操作方法の統一化」，「2.CLI，GUI に代わるネットワーク機器操作 API の提供」が満たされているのか，またはどの程度のところまで迫っているのかを検証していく．そして NETCONF/NETCONF-API に対する我々の期待の先にあるもの，NETCONF/NETCONF-API の活用によるネットワーキングの応用／適用範囲の拡大可能性について解説する．

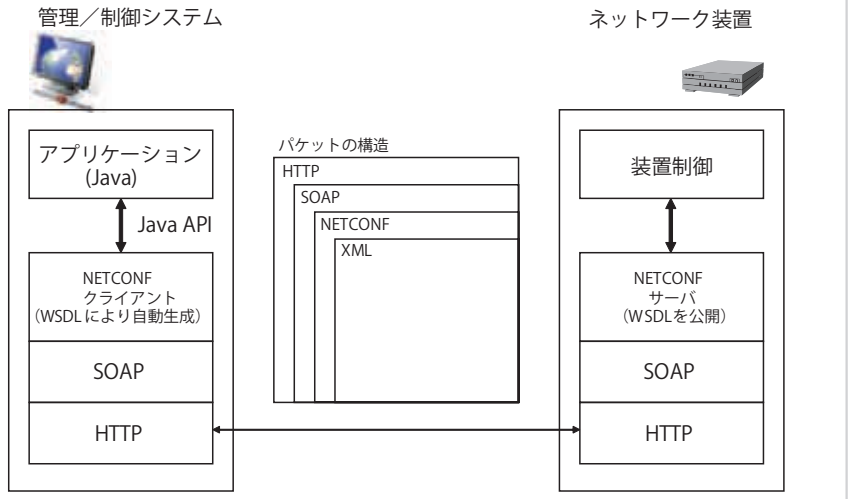
NETCONF/NETONF-API の利用

まず「1. ベンダ，機種の差異を問わないネットワーク機器操作方法の統一化」について見てみる．現状，ネッ



※出典：「Cisco NX-OS XML Interface User Guide, Release 4.2 」より

図 -3　NETCONF over SSH（NX-OS NETCONF）



※出典：「アラクサラネットワークス社プレスリリース（2008年11月10日）」より
筆者により図を一部変更

図 -4　NETCONF over SOAP（AX-ON-API）

トワーク機器ベンダはそれぞれの NETCONF 実装を発表している．主なものは，シスコシステムズ社の「NX-OS NETCONF API」，ジュニパーネットワークス社の「JUNOS NETCONF API」，アラクサラネットワークス社の「AX-ON-API」等がある．現状では残念ながらこれらに相互運用性はない．「1. ベンダ，機種の差異を問わないネットワーク機器操作方法の統一化」という点では，まだ実現されていないのが現状である．今後の標準化の議論やベンダの実装動向に期待するところである．

次に「2.CLI，GUI に代わるネットワーク機器操作 API の提供」についてはどうか．従来のようにネットワーク機器を人がオペレーションする分には，CLI/GUI で十分であるが，コンピュータからネットワーク機器を操作したいという動機の源泉は，

- ネットワーク機器を自動制御したい
- 多数のネットワーク機器を制御したい
- ネットワーク機器に大量の設定項目を設定したい

という要求である．

まず各社の NETCONF/NETCONF-API の実装方法について簡単に触れる．「NX-OS NETCONF API」,「JUNOS NETCONF API」では，端末からネットワーク機器へ SSH によるコネクションを確立した後，コマンドや設定情報を記述した XML ドキュメントを送受信することにより設定変更や情報取得を行う（図 -3）．一方，「AX-ON-API」は NETCONF-API が Java のライブラリ集として提供されており，コネクションの確立から各種設定変更や情報取得を AX-ON-API が提供する Java ライブラリを使用したプログラムを作成し実行することにより行う（図 -4）．簡単に両方式を比較すると，XML を直接編集する方式は，やりとりできる情報の自由度が高い反

面，定義すべき情報量も多くなる．それに比して，他方の Java API による操作方式は，ネットワーク機器の操作範囲や取得できる情報が Java API により提供されている範囲に限定されてしまうが，操作に対応した API が専用に用意されている分，プログラムは作成しやすいといえる．

各社の NETCONF/NETCONF-API 実装は現在も開発が続いており，操作範囲の拡充や利便性の向上が進展していくと思われる．

それでは今紹介した NETCONF-API の実際の利用方法例を挙げてみよう．今回は，アラクサラネットワークス社の AX-ON-API を例に取り，実際に AX-ON-API を用いて，ネットワーク機器を制御するアプリケーション例を示してみた．「2.CLI，GUI に代わるネットワーク機器操作 API の提供」という要求を満たし得るものなのかを見てみよう．

AX-ON-API は Java のライブラリ集という形で提供されている．AX-ON-API を使用し，スイッチに「VLAN を設定するプログラム」のサンプルソースコードの抜粋を図 -5 に示す．このプログラムは装置の IP アドレス，ポート番号，VLAN ID 等を変数として受け取り VLAN を設定するものである．

ソースコードは主に，「装置とのセッション開始に関する部分」，「実際に設定変更，情報の入手をやりとりする部分」，および「装置とのセッションを終了する部分」からなる．このように NETCONF-API（AX-ON-API）はセッション管理を行うことで，設定変更命令の衝突を防いでおり，また設定変更の実施時やエラーの発生時には応答コードが返されるつくりになっているため，エラー処理の記述にも長けている．従来手段の項で述べた種々の問題がクリアされていることが分かる．

このように AX-ON-API は Java のライブラリ集という形で提供されているため，利用には Java のプログラミングスキルが必要になる．プログラムであるため，当然コンパイルして実行する必要があり CLI での操作に比べ手間は多い．よって単純な操作であれば，装置の設定変更に要する時間等のコストは従来の CLI 等に分配があがる．しかしながら我々が「2.CLI，GUI に代わるネットワーク機器操作 API の提供」に期待していることは，

- ネットワーク機器を自動で，
- 同時に多数の設定対象に対して，
- 大量の設定項目を設定したい

という要求を満たすプログラムを作成するためのインタフェース（API）である．そのような観点から AX-ON-API を評価するならば，先に挙げた SNMP や CLI の援用に比べ，設定項目はデータモデルにより論理化されており，応答コードによりエラー処理記述に長けたプログラムを作成できる API であるといえる．

またさらに言うならば，行いたい作業のすべてを Java でコーディングする必要もない．先ほどのプログラムは，装置の IP アドレス，ポート番号，VLAN ID 等を変数と

```
//VlanController.java

import net.alaxala.oan.onapi.basic.Constants;
import net.alaxala.oan.onapi.basic.ISession;
import net.alaxala.oan.onapi.basic.IVlan;
import net.alaxala.oan.onapi.basic.LocatorObj;
import net.alaxala.oan.onapi.basic.SessionImpl;
import net.alaxala.oan.onapi.basic.UntaggedPort;
import net.alaxala.oan.onapi.basic.Vlan;
import net.alaxala.oan.onapi.basic.VlanImpl;
import net.alaxala.oan.onapi.basic.exception.NetconfException;

public class Sample {

    public static String sw_addr;//操作するスイッチの IP アドレス
    public static String phys_port;//操作するスイッチの物理ポート番号
    public static short vlan_no;//設定する VLAN_ID

    public static void main(String args[]) throws NetconfException {

        sw_addr = args[0];
        phys_port = args[1];
        vlan_no = (short) Integer.parseInt(args[2]);

        operate();
    }

    static void operate() throws NetconfException {
        LocatorObj lctr = new LocatorObj();
        // 対象装置のアドレスをセット
        lctr.setAddress(sw_addr);
        // サービス名をセット
        lctr.setService(Constants.SERVICE);
        // プロトコルをセット
        lctr.setTransport(Constants.TRANSPORT_HTTP);
        // ISession クラス
        ISession session = new SessionImpl();

        try {
            // ロケータ情報をセット
            session.setLocator(lctr);
            // セッションの開始
            session.open();
            // 装置のロック
            session.lock();
            // オペレーションの開始-->>
            IVlan vlanImpl = new VlanImpl();
            vlanImpl.setLocator(lctr);
            // Operate-2 Vlan の更新
            vlanImpl.editConfigMerge(add_Vlan());
            // <<--オペレーションの終了
            // 装置のアンロック
            session.unlock();
            // セッションの終了
            session.close();
        // AX-ON-API からの Exception をキャッチ
        } catch (NetconfException e) {
            // アプリケーションのエラー処理
            session.unlock();
            session.close();
        } finally {
            session.unlock();
            session.close();
        }
    }
    //Vlan の更新用データ生成
    private static Vlan add_Vlan() {
        // Vlan オブジェクトの生成
        Vlan vlanObj = new Vlan();
        //UntaggedPort の作成
        UntaggedPort unTaggedPort = new UntaggedPort();
        String[] untagPortIdList = new String[1];
        untagPortIdList[0] = phys_port;
        unTaggedPort.setPortid(untagPortIdList);
        // VlanData 格納
        vlanObj.setVlanid(vlan_no);
        vlanObj.setUntaggedPort(unTaggedPort);
        // Vlan オブジェクトを返却
        return vlanObj;
    }
}
```

図 -5　AX-ON-API を使用したサンプルソースコード

```
#!/usr/bin/ruby
puts `java -jar VlanController.jar 10.0.0.1 "port 0/1" 10`
puts `java -jar VlanController.jar 10.0.0.1 "port 0/2" 10`
puts `java -jar VlanController.jar 10.0.0.1 "port 0/3" 20`
puts `java -jar VlanController.jar 10.0.0.1 "port 0/4" 20`
puts `java -jar VlanController.jar 10.0.0.2 "port 0/11" 10`
puts `java -jar VlanController.jar 10.0.0.2 "port 0/12" 10`
puts `java -jar VlanController.jar 10.0.0.2 "port 0/13" 20`
puts `java -jar VlanController.jar 10.0.0.2 "port 0/14" 20`
```

図 -6　スクリプトによるプログラム実行

して受け取り，VLAN を設定するような作りになっているため，テキストファイルや CSV ファイルからリストを読み込み，実行させるスクリプト（Perl, Ruby 等）を作成することにより，Java によるコーディングの手間を軽減することもできる（図 -6）．

NETCONF-API によるネットワーキング

前章では実際に AX-ON-API を利用し，装置に VLAN を設定するプログラムのソースコードを挙げて，実際の NETCONF-API の使い方を見た．プログラムからネットワーク機器を操作し，

- ネットワーク機器を自動で，
- 同時に多数の設定対象に対して，
- 大量の設定項目を設定する

アプリケーションを作成するための API が NETCONF-API である．しかし，NETCONF-API の登場がもたらすイノベーションは，これらネットワーク機器の自動設定といった単なるオペレーションの代替手段の提供にとどまらない．NETCONF-API 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できるということは，

- ネットワーク機器を手足のように自由に使えるようになること
- ネットワーク機器が持つさまざまな情報を収集し，加工して使えるようになること

を意味する．

我々はこれまで CLI 等によりルーティングの変更や VLAN の設定変更を行ってきた．ネットワークの構築，運用，保守，管理に用いられてきたネットワーク機器の機能（ルーティング，VLAN 等）が，NETCONF-API の登場によりアプリケーションを構成，実現する機能手段として活用されるようになっていくだろう．たとえば，「何らかのイベントを検知してインタフェースを up/down する」といったネットワークの自動運用や，「VLAN の設定を変更することで，瞬時にネットワークのトポロジを変更する」といったダイナミックなネットワーク運用などが実現可能になる．さらには，今までルーティングプロトコルや STP といったネットワーク独自のプロトコルに依存してきた分野にアプリケーションからアプローチをかけることも可能だ．

たとえば，我々は今までの SNMP によりネットワークを監視し，アラームを検知した後，人が CLI を用いて復旧作業にあたっていた．ここで NETCONF-API を用いれば，この間の人が介在する部分を自動化できる．

次章では NETCONF-API による単純な機器操作ではなく，「ネットワーク機器の機能を手足のように自由に使えるようになること」と，「ネットワーク機器が持つさまざまな情報を収集し，加工して使えるようになること」を利用したシステムの試作例を紹介する．部品としては前章で試作した「VLAN 設定変更プログラム」を流用し，NETCONF-API の活用によるネットワーキングの適用拡大の実例を挙げてみる．

NETCONF-API を利用したシステムの例

本章では，NETCONF-API を利用して構築したシステムの一例を紹介する．NETCONF-API によりネットワーク機器をプログラムから操作できることによる，ネットワーキングの応用範囲，適用可能範囲拡大の一端を見ていただければと思う．

今回は NETCONF-API を用いてスイッチを操作し，「VLAN」というネットワーク構築の「一技術」を，システムを構成する「部品」として活用する仕組みを紹介する．VLAN をダイナミックに設定変更し運用することで，ネットワークに柔軟性を持たせ，物理装置や物理トポロジに拘束されないネットワーキングを実現する．

以上を踏まえ，今回は，ネットワーク機器（今回例ではレイヤ 2 スイッチ）を買い換えることなく機能を追加し続け，継続して使用することを可能とするシステムを試作してみた．

■ ネットワークの高機能化と高コスト化

従来，ネットワークは広帯域化やスループットの向上といった通信速度の性能面に焦点があてられてきた．しかし近年では単なる速度といった「性能」面から，ネットワークがどのような機能を提供できるかという「機能」面へ焦点が移ってきている．認証機能やファイアウォール等の機能がそれである．これら機能追加は，既設ネットワーク機器のリプレースによって導入されることが多い．たとえば，既設のスイッチを IEEE 802.1X 機能搭載の認証スイッチに置き換えて認証機能をネットワークに導入するといった例である．

■ ネットワークへの機能追加

このようなリプレースによる機能追加は，機能追加の

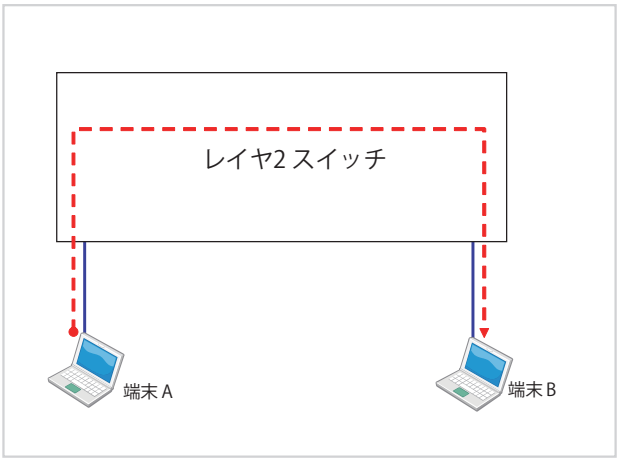


図 -7　スイッチの内部バスを経由する通信

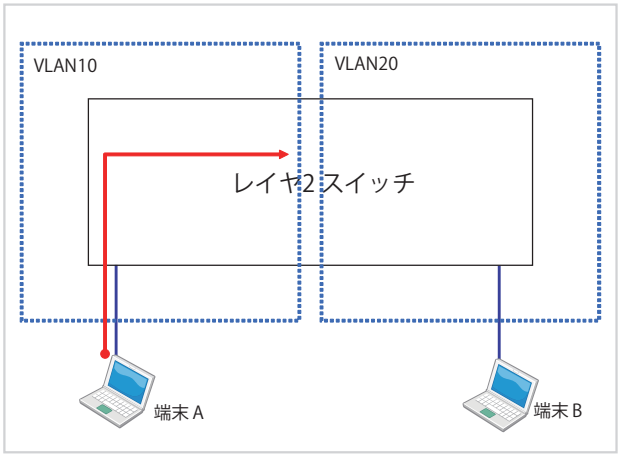


図 -8　VLAN により通信範囲が論理的に分割された状態

たびに「既設機器の廃棄」と「新規導入機器の購入」というコスト増を生じさせる．機能を追加したいがために，データ転送速度性能は十分であるにもかかわらず，同じ転送速度の高機能機器にリプレースするというのは無駄が多い．費用面のコストだけでなく，リプレース作業はネットワークシステムのダウンや機器の再設定作業という機会費用損失や人的コストも生じさせることになる．

■ NETCONF-API を用いた機能提供システム

そこで今回 NETCONF-API を用いて試作するシステムは，ネットワークへの機能追加を以下の要旨で行う．

- 既存(あるいは既設)のスイッチを有効に活用する
- スイッチ自体への機能追加は行わない
- 継続的な機能追加を実現する
- 機能の使い分けの自由度を提供する

今回は NETCONF-API と，NETCONF を実装し，VLAN 機能を搭載したアラクサラネットワークス社のスイッチ AX-2430 を用いて構築した．

基本アイディアは，

- VLAN によるバイパス経路の構築
- ミドルボックスによるフレーム処理
- NETCONF-API を活用した VLAN の切り替えによる機能の ON/OFF

である．それでは上記の基本アイディアについて順次説明する．

VLAN によるバイパス経路の構築

機能を追加するために，スイッチ自体に機能を実装し，高機能化する必要があるのは，スイッチ自身がフレーム処理を行う装置であるからである．そこで本アイディアでは，スイッチ自身はフレームの転送処理に徹し，フレームの加工や処理判断は後述するミドルボックスに委ねることを可能とする機構を構築する．

図 -7 はスイッチに端末が 2 台接続され，互いに通信

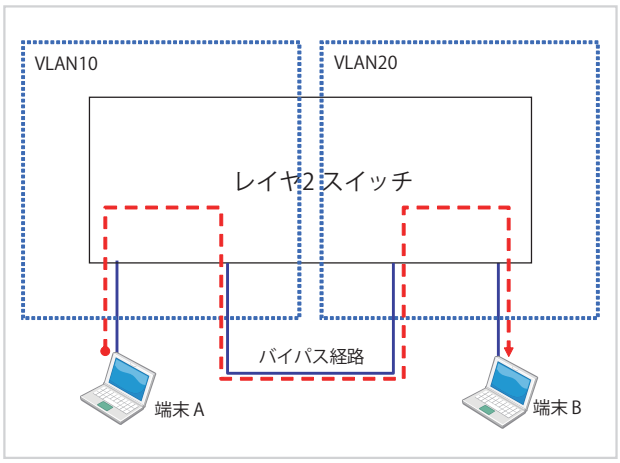


図 -9　VLAN 間を接続するバイパス経路を経由する通信

を行う場合の通信経路の様子を示している．端末 A が送信するフレームはスイッチの内部バスを経由して端末 B に送信される．よって，端末 A の送信するフレームをフィルタリングしたり，端末 A の接続自体を認証したりするような機能を導入しようとすると，スイッチ自体にそれらの機能を実装する必要が発生し，結果，高機能スイッチへのリプレースが生じてしまう．

ここからがアイディアである．

ではここで図 -8 のようにスイッチを 2 つの VLAN に分割し，端末 A と端末 B を別々の VLAN に所属させてみよう．するとそれぞれの所属する VLAN が異なるため，当然スイッチの内部バスを経由した通信ができなくなる．

そこで図 -9 のように端末 A が所属する VLAN と端末 B が所属する VLAN を接続するバイパス経路となるよう LAN ケーブルで接続すると，端末 A と端末 B はこのバイパス経路を経由して通信を行う．このように VLAN によりスイッチを論理的に分割することでもともとスイッチの内部バスを流れていた通信をスイッチの外に出す

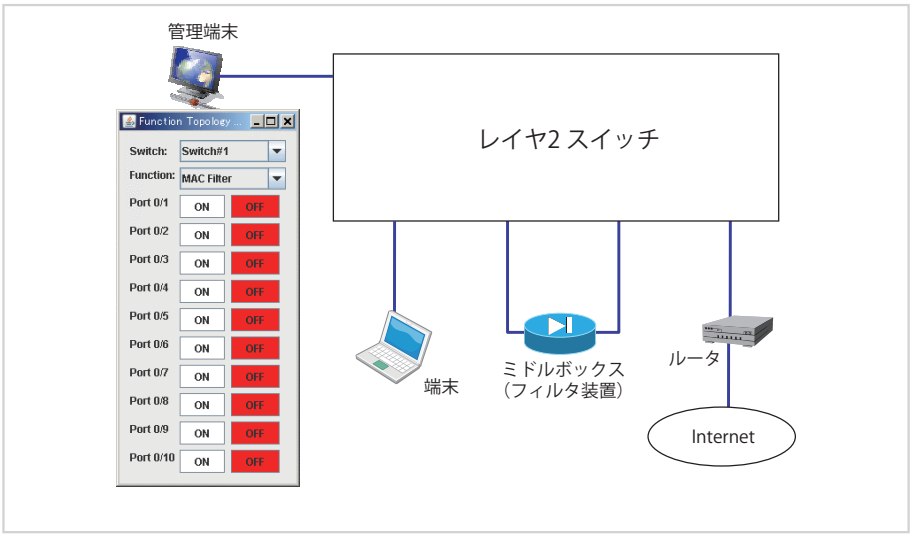


図 -10　本システムの機器接続構成

ことができるようになる．このバイパス経路上に次項で述べるミドルボックスを配置することで，スイッチの内部処理機構に機能追加を施さずにネットワークへの機能追加を実現できる．

ミドルボックスによるフレーム処理

ミドルボックスとは，伝送路上に配置し機能を強制的に適用するための装置である．ミドルボックスの例として，ファイアウォール，ロードバランサー，NAT BOX，IDS/IPS などがある．前項で述べたバイパス経路上にこれらミドルボックスを配置することにより，既設のスイッチをリプレースすることなく機能を追加することが可能になる．

しかし単に VLAN により LAN を分割し，バイパス経路上にミドルボックスを設置するという機能追加方法では，機能を継続的に追加，拡張したり，機能の ON/OFF を使い分けたりといった柔軟な運用ができない．そこで用いるのが NETCONF-API を用いた運用である．

VLAN の切り替えによる機能の ON/OFF

本システムの機器接続構成を図 -10 に示す．管理端末はスイッチに対し NETCONF-API を使用したプログラムにより制御を行う．NETCONF-API によりスイッチの VLAN の設定を変更することで，ミドルボックスを経由しない通信路（機能の OFF）とミドルボックスを経由する通信路(機能の ON）を作り出す．

■ システム動作概要

図 -10 中の管理端末を使用するユーザ（ネットワーク管理者)は，スイッチの「どのポートに対して，どの機能を提供するか」を図 -10 の管理端末で動作するプログラム（図 -10 のウィンドウはプログラムの GUI）にて制御する．今回の例では，各ポートにフィルタリング機能を適用する例を示している．図 -10 中のミドルボックスにはフィルタ機能を搭載したミドルボックスを設置する．図 -10 中の管理端末の GUI にて，クライアント端末が接続されたポートの ON/OFF ボタンをクリックすることで，NETCONF-API により VLAN 構成が変更される．図 -11 は機能 OFF 時の論理ネットワークを，図 -12 は機能 ON 時の論理ネットワーク構成を示す．図 -11 において，クライアント端末はミドルボックスを経由せず，スイッチの内部バスを経由してルータからインターネットへ通信を行う．機能を ON に設定した際に構築される図 -12 においては，クライアント端末はミドルボックスを経由してフレームのフィルタリング処理を施された後にルータを経由し，インターネットへ通信する．

本稿では紙面の都合上から割愛したが，スイッチが多段構成となった場合，追加機能が複数になった場合，またその機能の組合せを実現したい場合についても，柔軟に対応できるよう試作システムでは実現している．

■ システムの例のまとめ

本章では，「VLAN」というネットワーク機器の機能を，プログラムの部品として使用することで，ネットワークへの機能の追加，切り替えを実現するシステムを試作した．本システムが読者にとって，NETCONF/NETCONF-API をさまざまなことに活用するヒントとなったならば幸いである．

NETCONF/NETCONF-API への期待

本稿についてまとめる．冒頭で提示した NETCONF/NETCONF-API への期待のうち，「1. ベンダ，機種の差異を問わないネットワーク機器操作方法の統一化」については現状，ベンダ独自の実装となっているため，まだ遠いと言わざるを得ない．次に「2.CLI，GUI に代わる

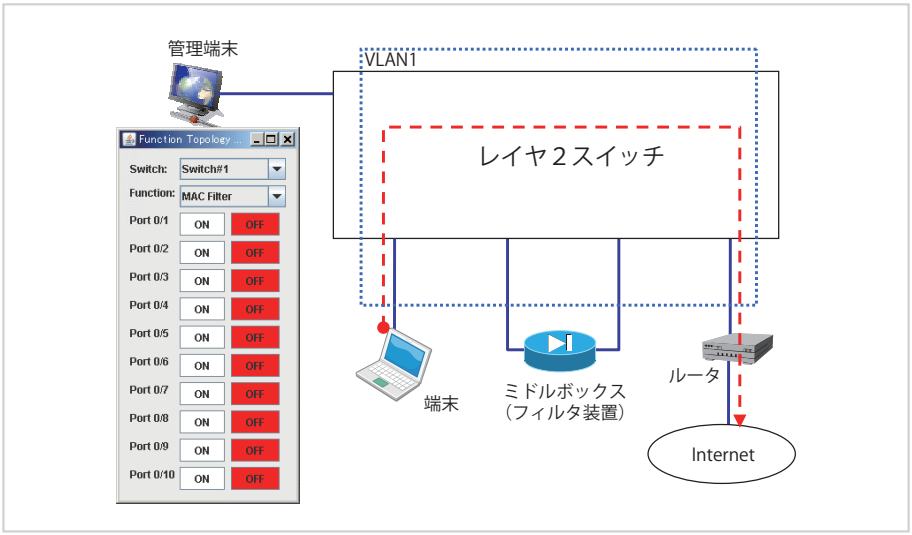


図 -11　フィルタ機能「OFF」時の論理ネットワーク

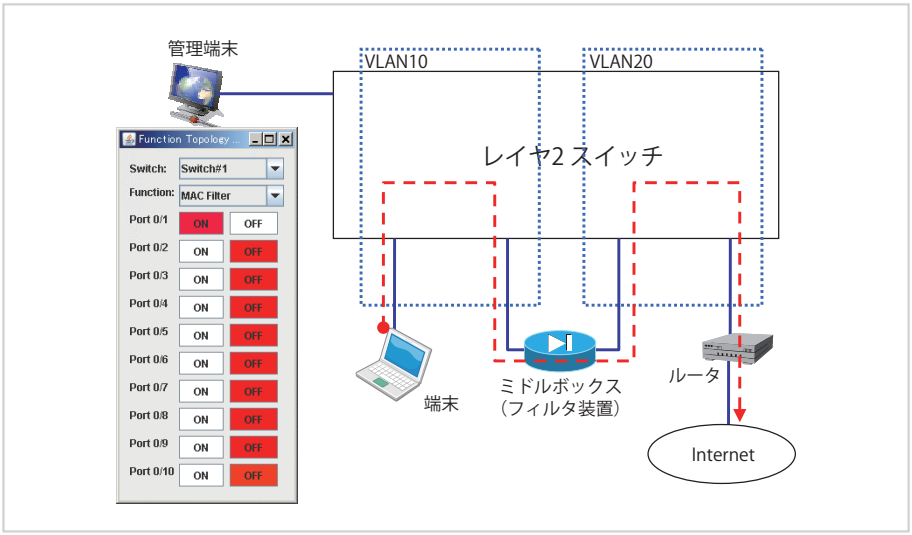


図 -12　フィルタ機能「ON」時の論理ネットワーク

ネットワーク機器操作 API の提供」については，利用できる製品が登場してきたことによりプログラムからネットワーク機器を制御できるアプリケーションの実装が容易になってきたといえる．ネットワーク機器をプログラムから手足のように自由自在に制御できるということは，自動で，同時に多数の機器に，大量の設定変更を施すことを可能とする．さらに前章で示したシステム例のように他の情報資源と連動した動作や，ダイナミックな構成変更等，ネットワーク機器の機能をアプリケーションに部品として組み込んだシステムの作成が可能になる．これらは，ネットワーキングの可能性の拡大，つまり，ICT アーキテクトにとっては，管理者の負担を減らし，自身のイマジネーションを具現化する手段として，ユーザにとってはダウンタイムの少ないネットワークを利用でき，まだ見ぬサービスの恩恵を受けられる未来として語ることができる．NETCONF-API からネットワーク機器の機能を制御できるということは，スイッチングやルーティングといったネットワーク機器の機能が，これまでの単なるネットワーク制御としての機能から，これから登場する，まだ見ぬアプリケーションを実現するための重要なキーテクノロジーとして見直されるようになることを意味する．ネットワーク NETCONF/NETCONF-API の今後の発展に期待したい．

取材協力：
アラクサラネットワークス(株)
http://www.alaxala.com/jp/
シスコシステムズ合同会社
http://www.cisco.com/jp/index.shtml
(株)ディーエスピーリサーチ
http://www.dspr.co.jp/Jpn/index_Jpn.html

（平成 21 年 9 月 29 日受付）

千装俊幸
chigira@coopii.com

（株）クーピー（http://www.coopii.com/）勤務．北陸先端科学技術大学院大学情報科学研究科修士課程修了．（株）クーピーにて耐高負荷サーバシステムの構築，視覚障害者サポート ASP サービスの開発に従事．