1 NETCONFの標準化動向

本稿では IETF における NETCONF の標準化状況と日本からの貢献について紹介する.

東村邦彦
(株)日立製作所 中央研究所

背景と目的

■ ネットワークシステムへの要求

企業の業務や消費者への各種サービスが IT 化されるにつれ，利用者の利便性が高まる一方で，IT システムの複雑度は増大の一途をたどっている．IT システムの運用管理においては，ネットワーク・ストレージ・サーバなどで構成される大量の IT 資源を統合して管理する要求が高まってきている．

これらの IT 資源のうち，ルータやスイッチ等のネットワーク装置の運用管理では，個々の機器が持つ独自のコマンドライン I/F（CLI）や，IETF で標準化されているネットワーク管理プロトコルである SNMP（Simple Network Management Protocol）を用いて管理することが主流であった．

CLI は，適切に設定された端末をシリアルインタフェース経由で装置に接続するのみで利用することができる．このため，装置の初期設定時やハードウェアトラブル時など，装置へのアクセス手段が限られている状態での人手による管理作業において利用されることが多い．CLI を利用するのは人間であるため，コマンドの実行結果や機器状態の表示などは人間が読み取りやすい形（たとえば英語の文章や整形された表形式など）になっている．CLI 経由の操作によって装置の設定が完了し，装置がネットワークに接続されると，以降の装置管理は遠隔地からネットワーク経由で行うことが中心となる．この場合も，人手による管理は Telnet や SSH 上の CLI を用いることとなるが，装置の状態監視を中心に SNMP が用いられることも多い．SNMP は，1988 年に最初の RFC（Request For Comment）が公開されているとおり古くから利用されているプロトコルである．このため，高機能なネットワーク装置はほぼすべて SNMP をサポートしており，装置ベンダは RFC 等で定義された標準の MIB（Management Information Base）のほかにベンダ独自のプライベート MIB を提供することで，標準外の情報へのアクセスを可能としている．

■ 現状の運用管理の問題点

CLI と SNMP を用いてネットワーク管理を自動化することには以下のような問題点がある．

CLI は前述したように人手による管理を対象としたものであるため，ソフトウェアから操作するためには人間向けに表現された文字列を送受信するプログラムを記述する必要がある．人間にとっては扱いやすいプロンプトの表示も，解析するプログラムにとっては処理を複雑にするだけのものとなる．また，多様な装置に対応するためには，ベンダごと・機種ごとに異なる CLI 仕様を解釈する必要がある．そして，設定が成功しているかどうかに関しても，改めて設定情報を取得しそれを字句解析して判定する必要がある．

SNMP はソフトウェアで通信することを前提に作られているため上記のような問題は発生しない．しかし，現状の SNMP は状態取得に使われることが多く，装置の設定に関してはサポートされている機能が少ないという問題がある．また，バイナリベースのプロトコルである上，多くのプログラマが慣れ親しんでいない SNMP 独特のプロトコルを理解する必要があるため，独自の MIB に対して操作を行うことに対する敷居がやや高いという問題がある．これは，開発工数の増大につながることになる．

■ XML によるモデル化・操作手順の確立

この状況に対処するため，XML をベースとしたネットワーク機器設定インタフェースへの要求が高まっている．XML を用いて機器との間で交換されるメッセージを記述することにより，人間およびソフトウェアへの可読性を両立させ，また，メッセージによる操作対象であるネットワーク機器をモデル化する際の表現力を高めることが可能となる．そして，XML で構築されたメッセージを用いた遠隔手続き呼び出し方式を装置とソフトウ

ェア間で確立することにより，前節で述べたプログラミング上の問題点を解消することができる．

この動きに呼応するように，インターネット関連の各種標準化を扱う IETF（Internet Engineering Task Force）では，2002 年の横浜での IETF にて NETCONF（Network Configuration）ワーキンググループの設立が起案され，2003 年から活動を開始した．我々は，アラクサラネットワークスと共同で XML ベースのネットワーク管理プロトコルの研究を進めており，2005 年前後から試作を開始し，2006 年度から IETF へ共同提案を続けている．2009 年 8 月現在，NETCONF 関連の RFC は 7 件[1)～7)]存在するが，そのうちの 1 件である“RFC 5381: Experience of Implementing NETCONF over SOAP”[6)]は我々の提案によるものである．

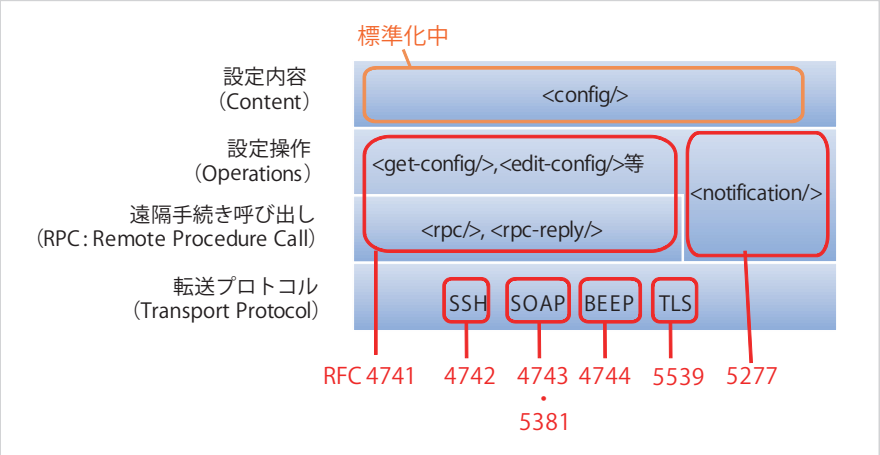

図 -1　NETCONF の階層構造

NETCONF の詳細

本章では，現在標準化が進められている NETCONF の詳細について述べる．

■ NETCONF の全体像

NETCONF ワーキンググループの憲章では，以下の特性を持つプロトコルを作成するとしている．

- 構成定義データおよび非構成定義データ（装置の状態等）を区別して取得するためのメカニズム
- 単一のプロトコルで装置上のすべての構成定義データにアクセスできる手段をベンダが提供できるような十分な拡張性
- プログラムが処理しやすいインタフェース（非構造化テキストからの文字列切り出しや書式変更による影響を排除）
- 特別なツールを使うことなく扱うことができるように（バイナリでなく）テキストでデータを表現
- 既存のユーザ認証方式との統合をサポート
- 既存の構成定義データベースとの統合をサポート
- ネットワークを介して構成定義を扱う際のトランザクションのサポート（ロックやロールバック等）
- トランスポートに可能な限り非依存
- 非同期通知のサポート

これらはすべて，CLI と SNMP による管理の問題点を解決することを目指すものとなっている．

図 -1 は NETCONF の階層構造を図式化したものである．レイヤ構成は高位のレイヤから

- 設定内容（Content）
- 設定操作（Operations）
- 遠隔手続き呼び出し（Remote Procedure Call）
- 転送プロトコル（Transport Protocol）

という形になっている．このように階層化することによって拡張性を高め，コンポーネント間の依存性を最低限に抑えている．標準化に際しても，それぞれの部分を個別に検討していくことが可能となり，プロトコル部，トランスポート部，データモデル部などで並行して標準化作業が行われている．

■ NETCONF プロトコル

NETCONF 標準の中核となるのが，基本的な設定操作と遠隔手続き呼び出しを XML で記述した NETCONF プロトコルであり，RFC 4741[1)]で規定されている．

遠隔手続き呼び出しは，要求に <rpc/> 要素，返答に <rpc-reply/> 要素を用いて実行する．この要素の内部に，基本操作として以下の 9 つの操作が定義されている．

- <get/>：実行中の構成定義およびデバイスの状態の取得
- <get-config/>：指定した構成定義の全部あるいは一部の取得
- <edit-config/>：指定した構成定義の全部あるいは一部の変更
- <copy-config/>：構成定義全体のコピー
- <delete-config/>：構成定義全体の削除
- <lock/>：構成定義に対する排他ロック
- <unlock/>：上記ロックの解除
- <close-session/>：NETCONF セッションの終了
- <kill-session/>：NETCONF セッションの強制終了

また，プロトコルの拡張部分が装置ごとに異なる場合にそれらを互いに調整するための <capabilities/> 要素を定義している．文献 1）では，上記操作による操作

対象となるデータのモデリングに関してはスコープ外としており，データを特定するためにXPathやXML名前空間が利用可能であることを述べるにとどめている．

NETCONFプロトコルによる構成定義変更の例を図-2に示す．なお，NETCONFでは，ネットワーク装置など遠隔手続き要求を受け付ける側を「サーバ」，遠隔手続き呼び出しを行う側を「クライアント」と呼ぶ．この例では，現在装置で使用されている構成定義中のポート0/2の管理ステータスを1に変更する<edit-config/>操作を送信し装置のポートを利用可能にしている．装置側からは，これに対して設定が完了したことを示す<ok/>が返答される．

クライアント→サーバ：

```
<rpc message-id="105">
 <edit-config xsi:type="(略)" xmlns:ns1="(略)">
  <target>
   <running xmlns="">
  </target>
  <config>
   <ns2:Lines xmlns:ns2="(略)">
    <ns2:Line operation="merge">
     <ns3:PortId xmlns:ns3="(略)">port 0/2</ns3:PortId>
     <AdminStatus xmlns="">1</AdminStatus>
    </ns2:Line>
   </ns2:Lines>
  </config>
 </edit-config>
</rpc>
```

サーバ→クライアント：

```
<rpc-reply message-id="105">
 <ok/>
</rpc-reply>
```

(注: 名前空間名などは省略している)

図-2　NETCONFによる構成定義例

このように，NETCONFではモデリングされた構成定義を読み書きする一連の操作によってネットワーク装置の動作をプログラムなどから動的に変更することが容易である．

■NETCONFのトランスポート層

前述したようにNETCONFのトランスポート層は，NETCONFプロトコルと分離されて標準化されている．NETCONFプロトコルのRFCが発行されると同時にトランスポート層としてSecure Shell（SSH）[2]，Simple Object Access Protocol（SOAP）[3]，Blocks Extensible Exchange Protocol（BEEP）[4]が規定され，2009年に新たにTransport Layer Security（TLS）[7]が追加された．現在，これらのトランスポートプロトコルのうち，SSHの実装のみが必須(mandatory)となっている．

NETCONFプロトコルがメッセージ指向であるのに対し，SSHおよびTLSはストリーム指向のプロトコルであるため，何らかの形でメッセージを区切る（フレーミングする）必要がある．このため，SSHマッピング[2]，TLSマッピング[7]では各<rpc/>メッセージを1つのXML文書として扱い，各文書の終わりを“]]>]]>”というシーケンスで示すという方法をとっている．BEEPとSOAPは，XMLをメッセージとして送受信する機能があるため，特別な規定を行わなくてもトランスポート層として自然な対応がとれている．

■通知機能

ここまでに述べたNETCONFの機能は，すべてクライアントからサーバ，すなわちネットワーク管理ソフトからネットワーク装置に対する要求に関するものであった．これに加え，NETCONFにはSNMPのトラップに相当するイベント通知機構が存在する．文献5）では，装置内でのイベントを購読するために追加された<create-subscription/>操作と，通知されるイベントを表す<notification/>要素の定義がなされている．イベント通知はNETCONF仕様としてはoptionalなため，実装は必須ではない．

日本からの標準化提案

現在我々は，ネットワーク装置と同様にサーバ・ストレージなどを統合管理するためのフレームワークの検討を行っている．フレームワークの実現のためには，サーバ・ストレージ系の管理フレームワークとの親和性がある形でネットワーク装置の管理を融合させる必要がある．NETCONFは，XMLによる記述で装置の構成定義を扱うため，サーバ系のシステムで用いられるSOAPとの親和性が高い．そこで我々は，NETCONFのSOAPマッピングを利用したネットワーク管理手法確立を目指して標準化活動を開始した．SOAPマッピング自体は文献3）によってすでに標準化されているため，我々は主にSOAPマッピングを用いて実際に管理システムを構築した際に考慮すべき点および標準から外れた機能が必要となる点を抽出することに注力した．

IETFには“Rough Consensus and Running Code”という標語が存在する．これは，一旦ラフな合意が取れた後，それを実装することであらためて合意した内容について考え直す，ということでもあり，また，実装が伴わない提案は会議内で重要視されない，ということを意味する．

我々は，NETCONF over SOAP を実際の製品上に実装した際の経験をベースとした SOAP 実装の必要性を提案することに努めた．そして，2006 年 11 月の最初の提案から約 2 年後，2008 年 10 月付けで RFC 5381 として採択された．

■ SOAP マッピングの必要性

文献 1）では，NETCONF のトランスポート層として必須のプロトコルは SSH に限定されている．これは，現行のネットワーク装置にはすでに実装されている SSH 上に実装することで装置に対して最小限の拡張で NETCONF を導入可能とするための配慮であること，および現在のネットワーク管理に携わるものにとって，コマンドラインインタフェースが最も多用する装置管理手段であることによる．しかし，SSH は安全なバイトストリームを提供するだけのトランスポートであり，XML などで構造化されたデータをメッセージとして交換することには向いていない．そのため，SSH 上で XML メッセージを交換するための特別な手順が必要となり，ネットワーク管理ソフトウェア作成者には余分な工数が必要となる．

これに対して，多くの Web サービスで利用されている SOAP（over HTTP）のサーバ機能をネットワーク装置側に実装することは，ネットワーク装置側のソフトウェア開発者には負担になるものである．しかし，SOAP を実装することによってサーバシステムの中でネットワーク装置をより積極的に活用可能となる．すなわち，いままでは導入時にネットワークを構成した後は，ネットワーク構成はほぼ固定であり変更することは困難であったが，ネットワーク装置が使いやすいインタフェースを提供することによりネットワークの構成変更は容易となりサーバやストレージの状況にあわせて柔軟にネットワーク構成を変更可能となる．

SOAP とその周辺技術はそれぞれ適切なインタフェースによって独立に検討が進められており，実装も多数存在する．NETCONF のトランスポート層として SOAP を用いることで，SOAP およびその周辺技術および豊富なミドルウェア群を活用でき，ソフトウェア開発者はその中から目的にあったものを選んでシステムを構築することが可能である．我々は，SOAP スタックの 1 つである Apache Axis を用いたクライアント実装経験をベースにした NETCONF over SOAP の利用方法について RFC 5381 内で記述している．図 -3 に Web サービス技術を利用した NETCONF over SOAP の実装の構成図を示す．

SOAP の周辺技術として，最も有用なものの 1 つが WSDL（Web Service Description Language）によるインタフェースの記述である．WSDL では，Web サービスのクライアントとサーバ間でどのようなメッセージが交換されるかを定義することが可能である．このため，WSDL を用いれば，NETCONF クライアントがネットワーク装置から WSDL 記述を取得し，その装置に適合した要求を送信できるようになる．また，Axis などの SOAP スタックは，WSDL の記述をインポートすることによって上位プログラム向けのスタブを生成することが可能であり，ソフトウェア開発者は NETCONF over SOAP の詳細を理解する必要がなくなる．RFC 5381 では，周辺ツールを用いた開発による省力化の例として Apache Axis を用いた具体的な開発の進め方についても記述している．

しかし，これらの技術・ツールを利用することにより，プログラマ側は SOAP メッセージの具体的な送受信方法に関して介入することが困難になる．このため，SOAP スタックの実装によっては文献 3）で定められた方式を完全に満たしたかたちで NETCONF over SOAP を実装することが困難となる場合がある．たとえば，文献 3）では NETCONF 上のセッションと HTTP コネクションを 1 対 1 対応させており，NETCONF セッションが続く限り同じ HTTP コネクションを用いて SOAP メッセージが交換されることを規定している．しかし，SOAP/HTTP の観点から見ると，HTTP 上の SOAP メッセージングはステートレスであり，同一のコネクション上でメッセージが交換されるとは限らない．SOAP の実装によっては，メッセージごとに異なるコネクションを

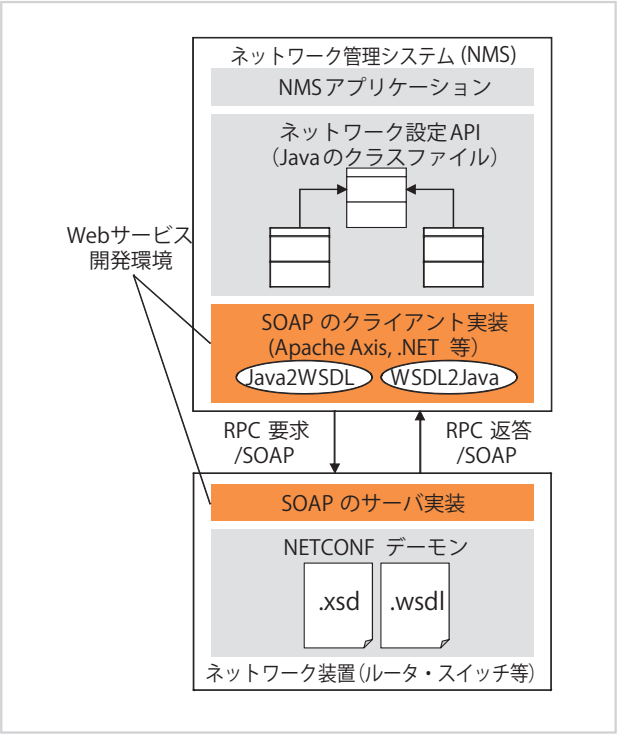

図 -3　Web サービス技術を応用した NETCONF over SOAP の実装

利用することも考えられる．このため，何らかのセッション管理をHTTPより上位のレイヤで実現する必要がある．RFC 5381では，解決策としてHTTPクッキー（RFC 2965）を利用する方法を提案している．最初にクライアントがサーバにアクセスし<hello/>によりセッションを開始した段階で，サーバはクライアントに対してセッションIDを発行する．このとき，文献1）では，<session/>要素の中にそのIDを格納していたが，我々が提案する方式ではクライアントにセッションIDを示すクッキーを格納するように指示することによって，コネクションに独立なセッションを維持することを可能としている．

■ NETCONF/SOAPを使ったプログラミング例

ネットワーク装置に対応したWSDL記述と適切なSOAPスタックを利用することにより，ネットワーク管理システムの実装言語でのスタブを生成することが可能となる．このスタブを元に装置操作のためのAPIを作成することにより装置制御のプログラミングが容易になる．

図-4に生成したAPIを利用して記述したプログラム例を示す．CLIを操作するコードでは文字列の構文解析が大半を占めることになるが，本コードではネットワーク装置が遠隔手続き呼び出しの対象として抽象化されていることが分かる．

```
public class OperationVlan1{
    private static String address = “192.168.0.1”;
    private static String portId_1 = “port 0/1”;
    private static String portId_10 = “port 0/10”;
    private static short vlanid = 10;                          定数値を設定
    public static void main(String[] args) throws NetconfException{
        operate();
    }
    static void operate() throws NetconfException{
        LocatorObj lctr = new LocatorObj();
        lctr.setAddress(address);                              対象装置のIPアドレスを指定
        ISession session = new SessionImpl();
        session.setLocator(lctr);
        session.open();
        session.lock();                                        セッションの開始，ロック
        IVlan vlanImpl = new VlanImpl();
        vlanImpl.setLocator(lctr);
        vlanImpl.editConfigMerge(addVlan());
        session.unlock();
        session.close();                                       セッションのアンロック，終了
    }
    private static Vlan addVlan(){                             VLAN編集メソッド
        UntaggedPort untaggedPort= new UntaggedPort();
        String[] untaggedPortIdList = new String[1];
        untaggedPortIdList[0] =portId_1;                       ポート0/1を設定
        untaggedPort.setPortId(untaggedPortIdList);
        TaggedPort taggedPort = new TaggedPort();
        String[] taggedPortIdList = new String[1];
        taggedPortIdList[0] =portId_10;                        ポート0/10を設定
        taggedPort.setPortId(taggedPortIdList);
        Vlan vlanObj = new Vlan();
        vlanObj.setVlanid(vlanid);
        vlanObj.setUntaggedPort(untaggedPort) ;                VLAN 10を設定
        vlanObj.setTaggedPort(taggedPort) ;
        return vlanObj;
    }
}
```

図-4　APIを利用したプログラム例

■ データモデリング

文献1）～4）の基本仕様がほぼ固まりつつあった2006年当時，NETCONFワーキンググループでは追加仕様の提案が頻発し，各社独自の小さな仕様の議論に時間を浪費していた．このため，標準化作業が実装方式に進む前の段階で停滞しており，何らかのかたちで実装に基づいたNETCONFプロトコルの再検討が必要となっていた．また，NETCONFプロトコル上で交換される構成定義データ自体は規定されておらず，各ネットワーク機器ベンダは従来のテキスト形式の構成定義ファイルを単純にXMLに変換したものをNETCONFの操作対象とするなど，相互運用性の点で問題となることが予想される．そこで我々は，NETCONFによるネットワーク管理技術を，既存ネットワーク管理システムを含んだ全体アーキテクチャの中に位置づけて整理するとともに，具体的な設定項目（VLAN, QoS, フィルタ）のモデル化を提案した．これらの操作は，企業ネットワークのネットワーク仮想化などのアプリケーションで多用されるものであり，サーバ・ストレージとの相互運用をスムーズに行うには早急なモデル標準化が必要である．

現在，NETCONFのデータモデル標準化作業は，その前段階であるモデリング言語の標準化がnetmodワーキンググループで活発に行われている段階であり，ワーキンググループでの議論が収束に向かった後，改めてデータモデルの標準化に向かう予定である．

関連する他の標準化動向

管理対象となるIT資源をモデル化することにより，装置メーカにかかわらず一貫した方法で管理可能とする動きは，ネットワークだけでなくサーバ，ストレージの分野にも存在する．

IT環境のシステム管理のための標準を制定するための業界団体のDMTF（Distributed Management Task Force）は，管理対象を共通のオブジェクトとオブジェクト間の関係によって表現する方法を，CIM（Common Information Model）として，また，CIMをベースとした分散環境の管理技術としてWBEM（Web-Based Enterprise Management）を策定している．

そして，これらの DMTF の技術の上に構築された管理イニシアティブとして，SMASH（Systems Management Architecture for Server Hardware）イニシアティブ，SMI（Storage Management Initiative）などがある．

SMASH イニシアティブは，サーバハードウェアの管理技術の標準化を行っている．初期の標準では CLI を志向していたが，2007 年に公開された SMASH 2.0 において WS-Management（SOAP/HTTP）に対応することにより，より Web サービスへの親和性を高めている．

SMI では，ストレージネットワークの業界団体である SNIA（Storage Networking Industry Association）が DTMF と連携し，SMI-S（Storage Management Initiative – Specification）と呼ばれるストレージ向けの管理プロファイルを策定している．現在，SMI-S では管理プロトコルとして CIM-XML が標準化されているが，Web サービス技術を活用したいベンダなどの働きかけによって WS-Management ベースの管理プロトコルを取り入れようとする動きがある．

このように，IT 環境を取り巻く運用管理システムにおいては，Web サービスとの親和性を高める方向で標準化が進んでいる．Web サービスを共通基盤とすることで，大量の IT 資源が複雑に連携してシステムを構成するデータセンタなどの統合運用管理がより容易に実現可能となることが期待できる．

直近の NETCONF 標準化動向

2009 年 9 月現在，以下の標準化が進行中である．

- 細粒度のロック機能：文献 1）で定義されているロックの機構は，構成定義全体をロックするものであるため，これをより細かい粒度でロックするメカニズムに関する検討
- NETCONF のモニタリング：NETCONF の動作状況（たとえば，確立中の NETCONF セッションの情報など）を取得するための機構と，装置がサポートしている XML スキーマの広告の機構に関する検討
- デフォルト処理：装置の構成定義で明示的に指定されてない「デフォルト値」の扱いに関する検討
- 文献 1）の更新（rfc4741bis）：文献 1）で記載した内容の誤記・明確化に関する検討（次期バージョンの NETCONF を指向するものではない）

また，NETCONF ワーキンググループ外においても関連する標準化活動が進行している．

- NETMOD（NETCONF Data Modeling Language）：NETCONF 上のデータをモデリングする際に，そのスキーマを容易に記述するために使用する言語の定義
- IPFIX（IP Flow Information Export）：フロー情報を監視するための機能設定のためのデータモデル策定
- OPSAWG（Operations and Management Area）：SNMP だけでなく Web 系の技術を含めたネットワーク管理フレームワークの整理

NETCONF の今後

ネットワーク装置の運用管理を自動化するための XML ベースのプロトコルである NETCONF の概要および標準化動向を説明するとともに，筆者らが提案活動を行った NETCONF/SOAP の実装に関する RFC である RFC 5381 について解説した．これまでは構成変更に人手を介すために動的な変更が困難であったネットワークシステムが，NETCONF という統一的なインタフェースによって上位システムの運用管理に容易に組み込むことができるようになると期待される．

参考文献

1）Enns, R. Ed.: NETCONF Configuration Protocol, IETF Request for Comments : 4741（2006）.
2）Wasserman, M. and Goddard, T. : Using the NETCONF Configuration Protocol over Secure Shell (SSH), IETF Request for Comments : 4742（2006）.
3）Goddard, T. : Using NETCONF over the Simple Object Access Protocol (SOAP), IETF Request for Comments : 4743 (2006).
4）Lear, E. and Crozier, K. : Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP), IETF Request for Comments: 4744.
5）Chisholm, S. and Trevino, H. : NETCONF Event Notifications, IETF Request for Comments : 5277（2008）.
6）Iijima, T., Atarashi, Y., Kimura, H., Kitani, M. and Okita, H. : Experience of Implementing NETCONF over SOAP, IETF Request for Comments : 5381 (2008).
7）Badra, M. : NETCONF over Transport Layer Security (TLS), IETF Request for Comments : 5539 (2009).

（平成 21 年 9 月 30 日受付）

東村邦彦
kunihiko.toumura.yv@hitachi.com

2001 年日立製作所入社．中央研究所ネットワークシステム研究部所属．ゲートウェイ装置のソフトウェアアーキテクチャ，ネットワーク管理方式に関する研究に従事．博士（工学）．電子情報通信学会会員．