E-21

Service Defined Network (SvDN) による L2 ネットワークの自動設定

Automated Configuration of L2 Network using Service Defined Network (SvDN)

寺本 泰大†　岡部 寿男††　新 麗†††
Yasuhiro Teramoto　Yasuo Okabe　Ray S. Atarashi

1．はじめに

近年、ネットワークシステムを構成するネットワーク機器数は爆発的に増加しており、それに伴いネットワークシステムも複雑化してきている。現在はネットワークシステムの設定変更を行う際には、ネットワーク管理者はtelnetやSSH などのコマンドラインインターフェイス (CLI) を利用して、個々の機器に対して手作業で設定変更を行う事が多い。しかしながらネットワークの複雑化により、手作業によるネットワーク管理が困難となってきている。 手作業でネットワーク管理を行う場合、ネットワーク管理者は 管理するネットワークの構成を把握し、細心の注意を払って設 定変更を行う必要がある。さらに、設定変更が複数の機器にまたがるような場合も多く、そのような場合ネットワーク管理者 は複数の機器の設定内容の整合性を確認しつつ、1つずつ設定 変更を行っていかなければならない。特にL2レイヤーではより上位のレイヤーと比較すると、VLANやMACアドレスなど管理すべき内容が多く、またIPレベルなどのL3の設定と比較すると設定する機器の数が非常に多く、また整合性の確認が非常に手間のかかる作業である。

また、CLI は、機器によって設定方法や書式が大きく異なるため、管理者は管理している 全ての機器の設定方法と書式について知っている必要がある。

複数のネットワーク機器を管理するために、プログラムを作成したり、例えばvlan.config[1]といった既存の製品を利用したりする事によって自動化するというような事も可能ではある。しかし、現在のところネットワーク機器への設定の変更や取得を行う方法が統一化されていないため、特定の機器の方式に依存せざるを得ず、プログラムの汎用性に乏しい。また、ネットワーク構成や機器が少し変化するだけで、プログラムが正しい動作をしなくなる可能性がある、という欠点が存在する。

このような問題が起こる原因として、ネットワークシステムへの設定変更を行う際に、個々のネットワーク機器を強く意識する必要があり、ネットワーク全体をソフトウェアで制御するには不向きな構造になっているという事が挙げられる。

ネットワークをソフトウェアから制御するための試みとしては、OpenFlow [2]を代表とするSoftware Defined Network(SDN) [3] が存在する。SDN ではネットワーク機器からOS などの概念を取り除き、いわばネットワークを制御するような OS で集中的にネットワークを管理する。しかしながら、 SDN ではOpenFlowに対応したネットワーク機器の導入が必須である事や、コントローラーによって集中制御する事によるボトルネックやセキュリティ上の問題点など解決すべき問題が多々存在する。また今のところ高レイヤーの制御方法などについては考えられておらず、クラウドなどのアプリケーションレイヤーを含めたネットワークシステム全体を制御するための統一的な手段が存在しない。

本研究では、既存の資源のみによってそのような問題点を解決し、柔軟な制御を可能とするための仕組みとして提唱するService Defined Network(SvDN)[4]、Service Defined Infrastructure(SvDI)[5]を用いる。SvDNではネットワークシステム全体の振舞いを「サービス記述」という形で記述する。サービスを管理するサーバーは、サービス記述を基にネットワークシステムの各機器の適切な設定を導出し、各機器に対して設定変更を行う。

本稿では、SvDN によるシステム管理への第一歩として、まずは自動化が比較的容易で、また手作業での設定が非常に煩雑である VLAN などの L2 レイヤーを中心としてSvDN の実装及び応用についての検討を行う。

2．Service Defined Network

本研究の提案する Service Defined Network(SvDN)において、ネットワークシステムを定義するものは「サービス定義」と呼ばれる。サービス定義とはネットワークシステム全体の構成や振る舞いについて記述したものである。基本的にサービス定義はネットワーク構成に依存しない形で記述を行い、サービス定義を基にサービス管理サーバーがネットワーク構成情報を基にそのネットワークに適したコンフィグを自動的に導出し、設定変更を行う。

従来のネットワークシステムを定義していたものはネットワークシステムを構成する各機器に設定されたコンフィグであるが、このような設定管理方法ではネットワークに関する設定が分散してしまい管理や把握が困難になる上に、ドキュメントなどでコンフィグを管理していたとしても実際の機器に設定されているコンフィグとの差異が生じるといった状況も発生しうるという欠点が存在する。

一方 SvDN ではネットワークシステムの構成を一元管理

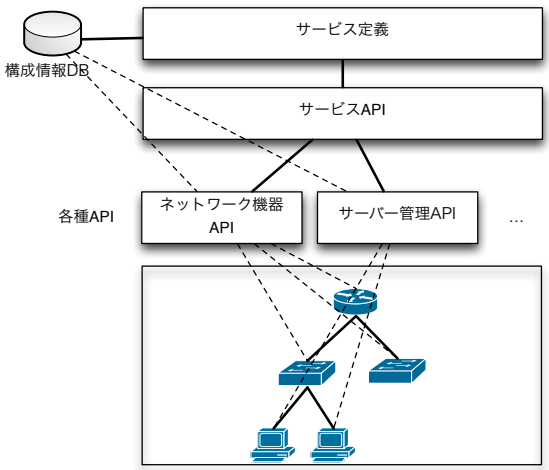

図 1 SvDN の構成

†京都大学大学院情報学研究科，Graduate School of Informatics, Kyoto University
††京都大学学術情報メディアセンター，Academic Center for Computing and Media Studies of Kyoto University
†††株式会社 IIJ イノベーションインスティテュート，IIJ Innovation Institute Inc.

する事によってそのような問題が生じるリスクを軽減している。SvDN の全体の構成の概要を図 1 に示す。

2.1 SvDN サービス管理サーバー

SvDN ではサービス管理サーバーがサービス定義や構成情報データベースの管理、ネットワーク機器への設定やトラフィックのモニタリングなどを行う事によってネットワークシステムの管理を行う。サービス管理サーバーはサービス定義と構成情報、モニタリング情報などを元に必要な設定を生成し、ネットワーク機器やサーバーに対してサービス API を通じて設定を行う。

2.2 サービス定義

ネットワークの構成や振る舞いについてはサービス定義という形で記述を行う。サービス定義は基本的にネットワーク構成に依存しない形で記述される。サービス定義と構成情報を元にコンフィグを生成する事によって、ネットワークを構成する機器の種類や、接続情報などの変化時にもサービス定義には手を加えずに構成情報を更新するのみでコンフィグが再生成される。この事によって、ネットワーク構成の変化等に柔軟に対応するようなシステム構築が可能となり、また構成変化時に設定ミスなどが起こりにくいような頑強なネットワークシステムの構築が可能となる。

2.3 構成情報データベース

SvDN ではサービス定義や、サーバーやネットワーク機器などの接続情報、設定情報などの構成情報についてデータベースを用いて集中的に管理を行う。これによってネットワークを構成する機器に関する情報を静的にかつ厳密に管理する事が可能となる。

2.4 API による抽象化

ネットワーク機器を設定する手段として CLI の他に NETCONF[6]や独自の API を用いた設定変更手段をサポートしている事が多い。しかしながら、現状ではその設定手段やフォーマットが機器ごとに大きく異なっており、プログラムから制御する事は困難である。

そこでネットワーク機器の持つ共通の機能のモデル化を行い、機器ごとにそのモデルに適したライブラリ(ドライバー)を用意する事によってその差異を吸収する。

これによってサービス管理サーバーは構成するネットワーク機器の文法など細かな差異について意識しなくとも統一化された手段で設定変更を行う事が可能となる。

3. SvDN の実装

本章では SvDN の実装方法について検討を行う。

3.1 ネットワーク機器の設定

SvDN ではネットワーク機器への設定にサービス API を用いて設定変更を行う。サービス API はネットワーク機器の機能のモデル化を行う事によって各機器の機能の微妙な差異を吸収すると同時に、機器ごとに異なる設定手段を隠蔽している。そのために機器の機能のモデル化及び、設定手段の抽象化を実装しなければならない。

そのような抽象化の実現のために図 2 に示すようなドライバー方式のフレームワークの設計を行った。これによってサービス管理サーバーからの設定する際に実際の機器への設定方法などの差異を吸収する事が出来る。また新たな機器が増えた場合も新たにドライバーの実装を行うだけで良く、サービス管理サーバー側に手を加えなくても済むという長所がある。

3 データベースの構成

ネットワークシステムを構成している機器の状態を厳密に管理するにはネットワーク構成を動的に取得するだけでは不十分であり、データベースなどを用いて静的に管理する必要がある。SvDN ではネットワークシステムの接続情報、VLAN の情報、各端末に割り当てられている IP アドレスの情報などネットワークの設定変更に際して必要となる情報をデータベースという形で静的に保持している。

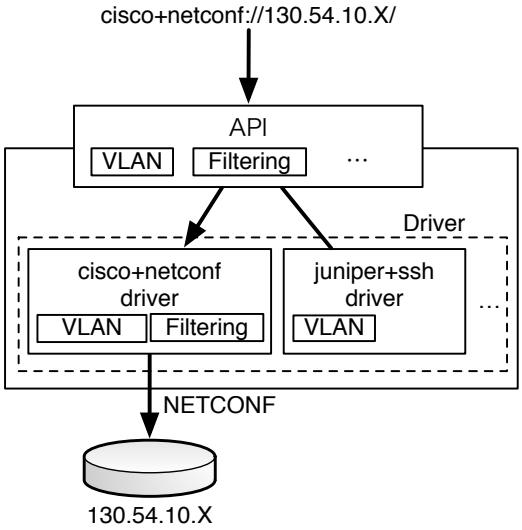

図 2 機器の設定のモデル化

物理接続及び L2 の設定情報に関するデータベースのエンティティ 図を図 3 に示す。このように構成情報をデータベースとして保存する事により、仮に意図しない構成変更が行われたりした時にも検出する事が可能となる。

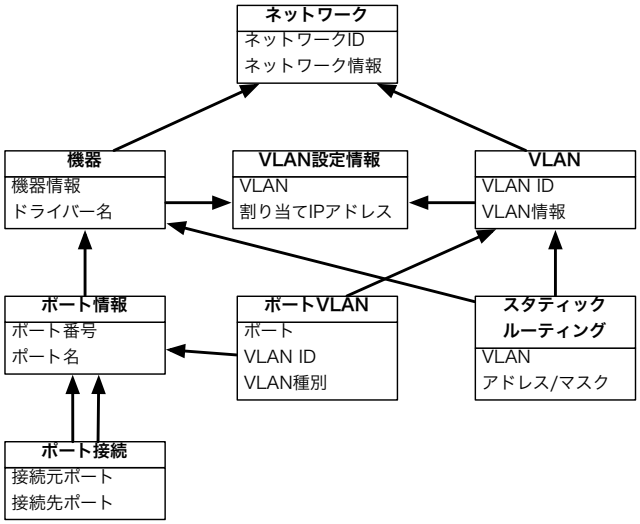

図 3 L2 の構成情報 DB 関係図

3.3 L2 レイヤーの設定

L2 レイヤーの VLAN に関する API モジュール及びデータベースの設計及び実装を行った。L2 レベルで現在実装を行った機能は、

- 機器へのタグ・アンタグ VLAN 設定 / 取得
- 接続情報を利用しての、VLAN を設定する経路の探索
- 2 ポートを指定しての VLAN の設定

等である。

3.4 イベントによる設定変更

SvDN では静的な設定のみでなく、SNMP などのフロー

監視や端末から送られてくるイベントを利用した動的な設定変更なども考えられる。動的な設定変更を行う事によって攻撃を受けた際に自動的に迅速に攻撃元を検出し、対応したりする事が可能となる。

4. SvDN によるシステム実装

4.1 L2 ネットワークの構成

ネットワークの構成情報をデータベースとして保存しているため、ネットワークに関する設定を自動的に行う事が可能となる。今回検討及び実装を行ったシステムでは、VLAN の構成を意識しない仮想ネットワークの構築も可能となる。将来的には VLAN 番号の 4096 個の制約などの解決策として、独立した経路では同じ VLAN 番号を利用するなどの高度な設定も可能となる。動作のイメージを図 4 に示す。

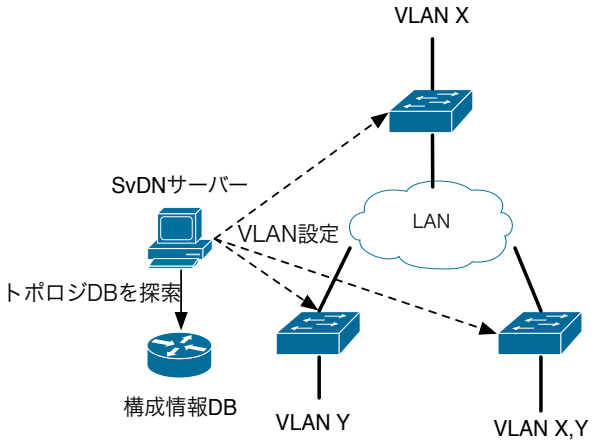

図 4 VLAN 設定

4.2 VLAN 隔離、アクセスフィルタリング

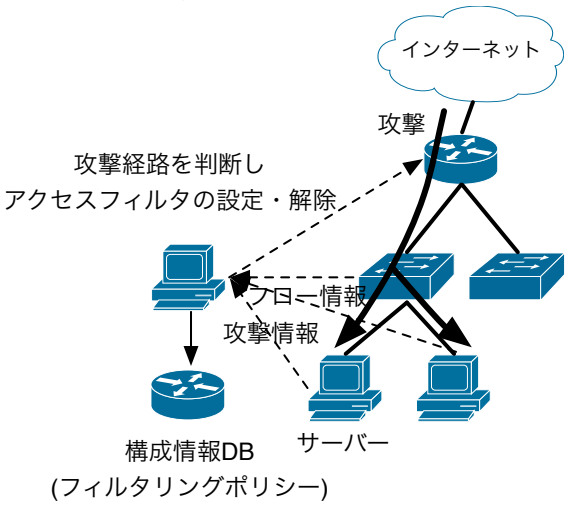

図 5 アクセスフィルタリング

ネットワークを構成するコンピューターがマルウェア等に感染している疑いがあったり外部からの攻撃を検出したりした際に、一時的にネットワーク機器にアクセスフィルタリングを設定したり VLAN を分けて隔離したいというような状況が存在する。

このような時はネットワーク管理者がネットワーク機器に設定を行ったり、IDS などが特定の機器に対してアクセスフィルタリングの設定を行ったりしてきたが、ネットワークが複雑になると複数の機器に対して設定を行う必要があるような事も多い。また、脅威が去った際にはそのフィルタリングの設定を解除する事が望ましいが、手作業で設定した際にはこれも手作業で行う必要がある。

SvDN では構成情報 DB によりネットワーク全体の構成を把握しているため、どのような設定を行う必要があるのかさえ判明すれば設定や解除は自動的に行う事が出来る。図 5 は外部からの攻撃があった際の例である。攻撃を受けたサーバー自身から攻撃に関する情報が届いたり、スイッチやルーターのフロー情報をモニタリングして攻撃を検出すると、SvDN サーバーは構成情報 DB からトポロジーの情報と攻撃検出時のポシリーを取得し、ポリシーに従って適切な位置にフィルタリング等の設定を行う。また一定時間し、攻撃が止んだと判断される際にはポリシーに従ってフィルタリングの解除などの動作を行う事も可能である。

4.3 トラフィックの制限

外部からの DoS 攻撃などがあった際には、ネットワーク内に大量のトラフィックが流れ込む事によって攻撃を受けた端末のみでなく回線を共有している他の端末も帯域が圧迫されて影響を受ける。このような問題点の対策方法としては、端末ごとに利用可能な最大帯域を設定する事によってその他の端末が影響を受けないようにするような方法も考えられるが、回線を最大限に生かす事が出来ない。攻撃検出時に特定の端末に対する通信だけ帯域制限を行うような方法も考えられるが、

例として、サーバーに対して攻撃が行われた際に外部からの特定のパケットのみを遮断する例について述べる。

本大学では IRC サーバーを運用管理しているが、IRC サーバーに対しては定期的に数 Gbps にもなる DoS 攻撃が行われている。数 Gbps の帯域がネットワークに届くと、10G 回線では問題なく通信可能であるが、1G 回線や 100M 回線になる部分で帯域が埋め尽され、それ以下のネットワークで通信が困難になるという問題が発生している。このような攻撃に対しては、IRC サーバー宛ての SYN などのパケットに対して帯域制限をかける事で対処可能であるが、10G 回線と接続されている機器に帯域制限の設定を行わなければ対処する事が出来ない。しかしこのような設定を行うためには以下のような障害が存在する。

- 10Gbps 回線には常に大量のパケットが流れており、攻撃を検出する事が困難である。
- 1Gbps 回線や IRC サーバーで攻撃を検出して、その機器に設定しても上位のネットワーク機器との接続が困難になる問題点は解消出来ない
- IRC サーバー等で攻撃を検出しても、上位のネットワーク機器に対して設定出来る権限を持っているとは限らない。このような時は、上位の管理者に依頼する必要があるが、時に対処が遅れる事がある。

以上のような問題点は SvDN で解決する事が可能である。図 6 に示すように、IRC サーバーが攻撃を検出し SvDN サーバーに攻撃を受けた事を通知する(ただし、ここでは DoS 攻撃で帯域が埋まっているような際にも何らかの手段で SvDN サーバーまでの到達性は保証されているものとする)。SvDN サーバーは、構成情報 DB を参照し、IRC サーバーからの依頼によって設定可能な権限や、対処方法のポリシー、トポロジーDB、帯域情報などの情報を基にアクセスフィルタリングの設定を行う。一定時間フローを監視し、攻撃が止んだと判断された場合はフィルタリングの設定を解除する。

このように SvDN サーバーを介する事によって、設定可

能な権限を細分化して設定したり、適切な設定箇所を判断して安全かつ自動的に設定変更を行う事が可能となる。

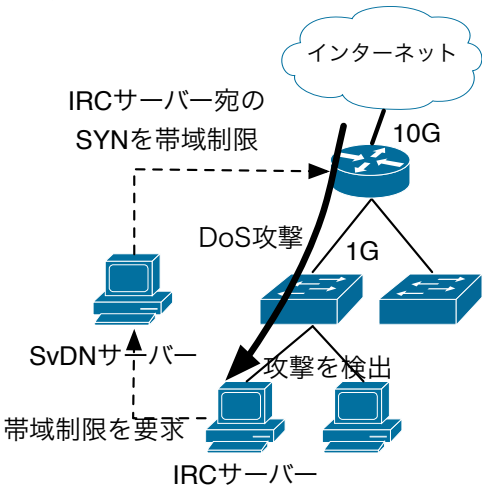

図 6 攻撃の検出と帯域制限の設定

4.4 マイグレーション

近年では仮想マシンなどを用いたクラウドなどが盛んに行われ、libvirt[7]など仮想マシンを操作するためのライブラリなども非常に充実してきている。仮想化の代表的な技術の一つとして仮想マシンのインスタンスを他のサーバーに集約するマイグレーションがある。マイグレーションは仮想マシンのインスタンスを実行したまま他の仮想マシンサーバーに移動する事が出来る技術である。これによって仮想マシンを遠距離の別の拠点へと移動させたり、仮想マシンの集約や隔離を行う事によって最適化を図る事が出来る。しかしながら、通信環境を意地したままマイグレーションを行うために VLAN などのネットワークの設定も同時に変更する必要がある。このように複数のレイヤーにまたがる設定変更を行う統一的な手法は存在せず、そのような事を実現するための実装は非常にコストや時間がかかる。

このような場合にも SvDN によって仮想マシンサーバーのマイグレーションやネットワークの制御を抽象化する事によって、柔軟な制御が可能となる。

4.5 複数拠点の管理

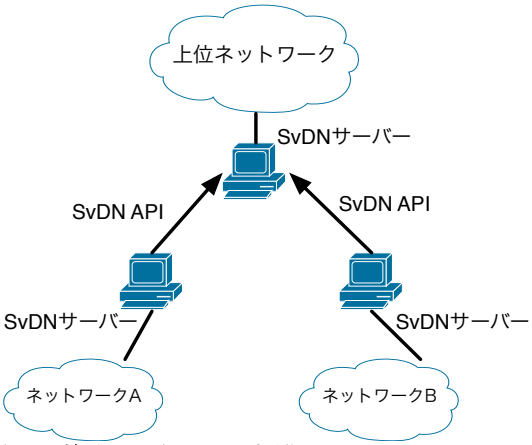

図 7 複数拠点での連携

SvDN ではサービス管理サーバーの機能を図 7 に示すようにサービス API として提供する事によって階層化が可能となり、大学や企業といった階層化された組織においても柔軟な運用が可能となる。例えば大学では複数の部屋を VLAN 接続したいような場合には、これまでは自分の管轄外のネットワークを通過する際には書面などの手続きが必要であった。このような設定についても必要時にサービス管理サーバーが上位のサービス管理サーバーに VLAN の設定などを要求する事によって自動設定を行う、といった事が可能となる。

5. まとめ

本論文では、Service Defined Network(SvDN)の提案を行い、L2 レベルの実装及び SvDN を利用したシステム実装の検討について述べた。

SvDN の実現のためには、様々なベンダーの機器やソフトウェアに対応する必要がある。そのためのモデル化及び実装について今後より検討していく予定である。

6. 参考文献

[1] “ネットワーク設計ツール vlan .config / 株式会社イイガ”. http://www.iiga.jp/solution/config/vlan.htmlt.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G.M. Parulkar, L.L. Peterson, J. Rexford, S. Shenker, and J.S. Turner, “Openflow: enabling innovation in campus net- works,” Computer Communication Review, vol.38, no.2, pp.69–74, 2008.

[3] “Open networking foundation”. https://www.opennetworking.org/.

[4]寺本, 岡部, 新, “トポロジーデータベースを利用したネットワークの自動設定変更,” 信学技報 IA2011-58, vol. 111, no. 375, pp. 21-26, Jan. 2012.

[5] 新, 新,岡部, “Service Defined Infrastructure (SvDI)の実装方法と応用の検討“ 信学技報IA2011-95, vol. 111, no. 485, pp. 167-172, Jan. 2012.

[6] “Netconf configuration protocol”. http://tools.ietf.org/html/rfc4741.

[7] “libvirt: The virtualization api”. http://libvirt.org/.