

# 管理効率化のための OSS を用いたネットワーク機器管理システムの構築

吉永 千有希, 中島 順美, 一瀬 光, 吉崎 弘一

大分大学

cyoshinaga@oita-u.ac.jp

j-nakashima@oita-u.ac.jp

ichise-hikaru@oita-u.ac.jp

kyoshi@oita-u.ac.jp

## Construction of a network device management system using open source software for effective management

Chiaki Yoshinaga, Junmi Nakashima, Hikaru Ichise, Koichi Yoshizaki

Oita Univ.

### 概要

大分大学では、2023 年秋に全学で利用するルータ、スイッチを含めたネットワーク機器を更改した。この更改を契機とし、ネットワーク機器管理の見直しを行った。従来は、ネットワーク管理者が Excel ファイルの台帳で機器の構成情報を管理しており、手動操作による台帳の変更漏れや変更履歴の未管理等が課題となっていた。この課題を解決するため、ネットワーク機器の構成情報を一元的に管理するシステムを、オープンソースソフトウェア (OSS) を利用して構築した。本システムには、これまで台帳で管理していたネットワーク機器の設置場所等の情報と実機から取得した機器のインターフェースや隣接する機器の情報等を登録した。また、登録した情報から生成した設定情報と実機の設定情報の差分を検証し、設定情報を実機に投入する検証を行った。その結果、ネットワーク機器管理システムの有益性と問題を確認することができた。

## 1 はじめに

### 1.1 基盤情報システムの更改

大分大学では、2011 年からサーバ・主要 LAN スイッチを主とする基盤情報システムを 6 年、教育用 PC を主とする教育情報システムを 4 年または 5 年ごとに更改している。この更改によって、システムの利便性やセキュリティの向上を図ると共に、経費と管理コストの削減にも重点を置いている。直近では 2023 年度に基盤情報・教育情報の両方のシステムを更改した。

この 2023 年度の更改前後のネットワークスイッチを、旦野原キャンパスのみに焦点を当て、表 1 に示す。これまで、旦野原キャンパスでは基幹となる少数のスイッチのみを基盤情報システムとして導入し、それ以外の多数のスイッチは必要に応じて個別に購入していた。そのため、10 年以上利用している古いスイッチが多数存在していた。また、表 1 で示すように、更改前は多数のベンダのスイッチが混在していたが、更改後は AlaxalA

製と Panasonic 製のスイッチ約 100 台に統一された。尚、挾間キャンパスでは約 200 台のスイッチ (AlaxalA 製/Panasonic 製/FXC 製) をそれぞれ導入している。

表 1 更改前後の導入スイッチ数 (旦野原)

スイッチのベンダ	更改前	更改後
Allied Telesis	36	-
APRESIA Systems	29	-
ALAXALA Networks	21	40
D-Link	5	-
BUFFALO	1	-
NEC	1	-
NETGEAR	2	-
Panasonic	1	62
合計	96	102

また、無線 LAN 環境として旦野原キャンパスでは 161 台、挾間キャンパスでは 139 台の無線 AP

を新規で導入した。キャンパスごとの無線 AP 設置台数を表 2 に示す。

表 2 無線 AP の設置台数

キャンパス	更改で追加	更改後の合計台数
旦野原	161	304
挾間	139	220

さらに、Firewall の更改では、これまで「学内一学外」の通信のみを監視していたものを、更改後は学内のサブネット間の通信も監視するように設計し、セキュリティの強化を図った。

## 1.2 従来のネットワーク機器管理方法と問題

2023 年度に実施した更改までは、本学のネットワーク機器管理は Excel ファイルの台帳で行っており、以下の情報を記録していた。

- 機器の名称、設置場所、ベンダ、モデル名
- アップリンクスイッチ、ケーブル種別
- IP アドレス

これらの情報の記録及び修正は、ネットワーク管理者が手作業で行っていた。この管理方法は以前から問題視されていたが、今回のネットワーク更改に伴い、具体的な問題として表面化した。その一例として、以下のような問題がある。

ネットワーク機器の更新作業中、台帳を参照しながら設定を行ったにもかかわらず、通信が確立せず、その問題を解決するために多大な時間を要した。調査の結果、この問題は台帳で管理していたネットワーク機器や VLAN 情報に誤りがあったことに起因していたことが判明した。この問題の背景には以下の要因が関係していた。

- 台帳の手作業入力による入力ミスや更新漏れ
- スwitchの設定情報の不十分な管理
- 修正履歴の未管理
- ネットワークトポロジーの整合性チェック機能の欠如

これらの要因により、誤った情報や不要であった情報が長期にわたって引き継がれ、今回の問題を引き起こしたと言える。

さらに、本学のネットワーク管理体制にも問題があった。ネットワーク機器の設定変更は、学内ネットワーク管理者に加え、学外の保守業者も行う場合がある。学外の保守業者が直接ネットワーク機器の設定を変更し、設定情報はメールにて報告を受け、学内のネットワーク管理者が台帳の修正を行うというプロセスがあった。このプロセス

において情報漏れが生じ、台帳と実機の設定に差異が生じた。

これらの問題を踏まえ、本学では従来に比べてより効率的かつ正確なネットワーク機器管理方法を検討した。次章では、この検討の一環として、ネットワーク機器管理システムの構築に関連する研究の調査結果について述べる。

## 2 関連研究

ネットワーク機器管理システムの構築に関する研究は、これまで多数なされている。本章では、特に本研究と関連する研究について述べる。

文献[1]では、OSS を用いた構成管理システムを構築しており、NetBox[2]をネットワーク構成情報のデータベースとして、Ansible[3]を構成管理ツールとして活用している。また、複数のベンダで構成されるスイッチのインターフェース情報に対応するため、Batfish[4]を用いてフォーマットを統一化し、REST API を介して NetBox に登録している。ここで、文献[1]と本研究の使用ツールを比較した結果を表 3 に示す。

表 3 構成管理システムの比較

役割	文献[1]	本研究
構成情報 DB	NetBox	NetBox
IF 情報の統一化	Batfish	Ruby 言語のプログラム
構成管理ツール	Ansible	Oxidized NetBox

表 3 で示すように、本研究も文献[1]と同じようにネットワーク機器の一元管理は NetBox を用いている。文献[1]と異なる点は、以下の通りである。

- 複数のベンダで構成されるスイッチのインターフェース情報を NetBox に登録するため、フォーマット統一化と登録に Ruby 言語のプログラムを用いた点。
- Oxidized[5]と NetBox の機能それぞれを活用して、構成管理ツールとしての役割を持たせた点。
- NetBox 上で実機の設定情報とシステムが生成する設定情報を比較検証する点。

Oxidized は、ネットワーク機器の構成バックアッ

ツールである。あらかじめコマンドを設定しておくことで、ネットワーク機器に SSH で接続し、CLI を操作して構成情報を自動取得することが可能である。本研究では、この機能に着目し、Oxidized を用いて実機から構成情報を取得する。取得した情報は、Ruby 言語のプログラムを用いてフォーマットを統一化し、REST API を介して NetBox に登録を行う。また、Oxidized は既存のインフラとの互換性が高く、導入がスムーズに進めることができる利点があった。しかしながら、本研究における主な目的は、登録する機器情報の正確性を向上する点にあるため、他のツールを用いていたとしても、実機から構成情報を効率的かつ正確に取得できるのであれば、使用ツールの選択は二次的な問題と捉えた。

また、文献[1]では、構成管理ツール機能として、実機への VLAN 設定の反映や機器設定の差分確認を、Ansible を利用して実装しているが、本研究では NetBox の Configuration Rendering 機能を用いて確認した。この機能は NetBox 上にあるネットワーク機器のインターフェース情報とテンプレートを組み合わせることで、本来あるべき設定情報としての「生成設定情報」を NetBox 上で生成する（図

1）。

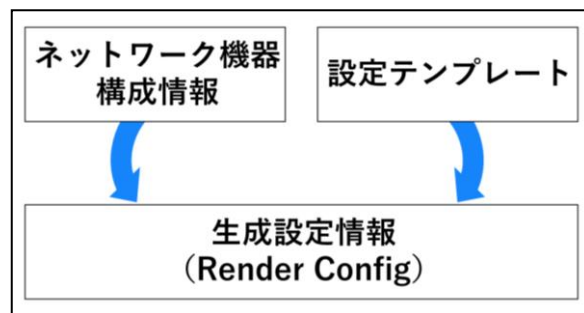


図 1 Configuration Rendering 機能のイメージ図

### 3 ネットワーク機器管理システムの構築

本章では、構築したネットワーク機器管理システムの概要とその設計・実装について述べる。

#### 3.1 システムの概要と提案手法

本システムは、ネットワーク管理者がネットワーク機器を効率的に管理することを目的としたネットワーク機器管理システムであり、NetBox を用いてネットワーク機器の構成情報を一元管理し、Oxidized により実機からの構成情報取得と実機への設定反映を行う。また、構成情報を世代管理するため、GitLab を導入し、Oxidized との連携を実装した。（図 2）

- ・ **NetBox** : ネットワーク機器の設置場所やモデル、インターフェース情報を一元管理している。ネットワークの本来あるべき姿をモデル

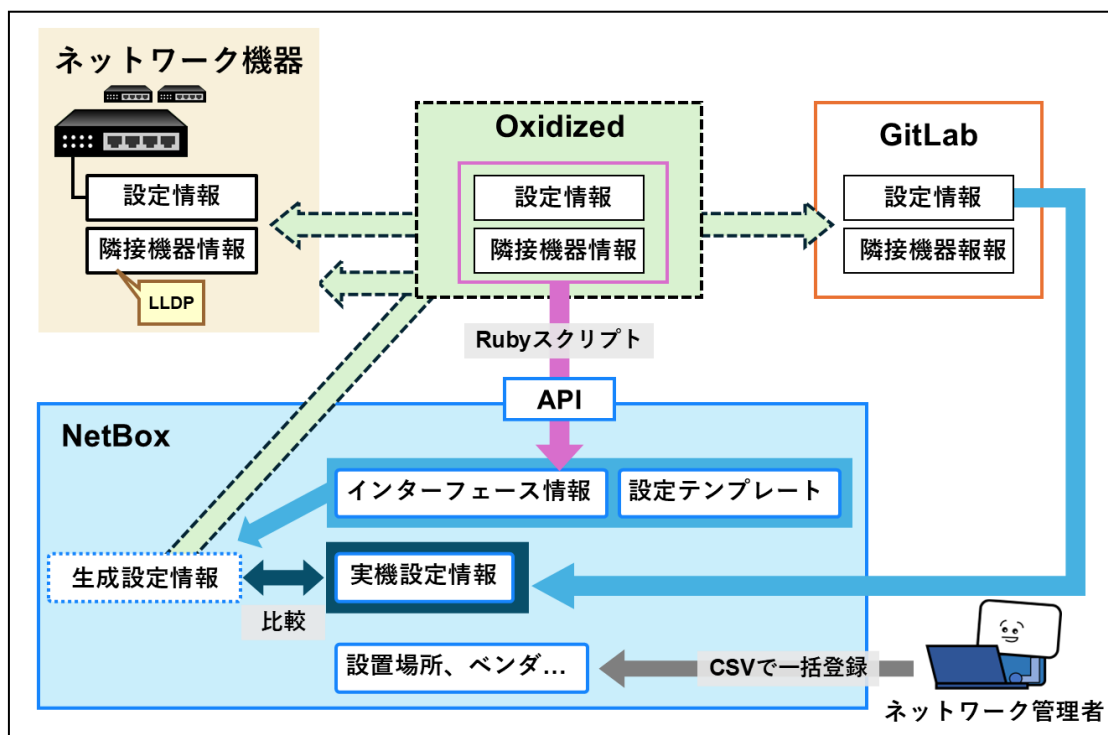


図 2 ネットワーク機器管理システムの構成図

化して管理する。現在管理しているネットワーク機器は、L2 スイッチ、L3 スイッチ、無線 AP、Firewall である。また、本研究での NetBox のバージョンは 4.0.9 である。(2024 年 10 月 18 日時点)

- ・ **Oxidized** : ネットワーク機器の設定情報や LLDP によって取得した隣接機器情報を実機から取得している。また、NetBox の Configuration Rendering 機能を用いて生成した生成設定情報をネットワーク機器に反映させる役割も担っている。
- ・ **GitLab** : Oxidized で取得したネットワーク機器の設定情報や隣接機器情報の世代管理を行っている。
- ・ **LLDP (Link Layer Discovery Protocol)** : 隣接する機器の情報を取得するためのプロトコルである。本研究では、L2 スイッチを対象に、スイッチや無線 AP が接続しているインターフェースに LLDP を設定した。

本システムを構築する上で、管理効率化と正確性を向上させるため「実機からの構成情報取得と登録の効率化」、「ネットワーク機器構成情報の世代管理」及び「設定情報の自動生成」に着目したこれら 3 点について以下に詳細を記載する。

### 3.2 実機からの構成情報取得と登録の効率化

NetBox に必要なデータを登録するため、事前にネットワーク機器に関する情報を整理する。この際、未使用な VLAN や使用用途が不明な VLAN も多く確認できたため、それらも含めて整理する。その後、必要な情報を CSV ファイルにまとめて、一括登録する。しかし、ネットワーク機器のインターフェース情報は、実機から取得した情報を基に登録するため、各ネットワーク機器から構成情報を取得する仕組みを構築する。

本研究では、構成情報の取得対象を L2 スイッチに限定した。また、スイッチや無線 AP が接続しているインターフェースのみを対象に LLDP を設定した。

次に、スイッチから設定情報と隣接機器情報を取得するために、Oxidized を設定した。このプロセスについては、スイッチの構成情報の世代管理にも関係するので 3.3 節で述べる。NetBox への登録は、Ruby 言語のプログラムにより実行する。Oxidized で取得した設定情報と隣接機器情報から VLAN やポート種類等の必要な情報を抽出し、NetBox の REST API を介して登録した。これによ

り、手動でのデータ入力の際の誤りの発生を抑制し、管理効率化と登録情報の正確性の向上を図る。スイッチの構成情報はスイッチのベンダやモデルによって形式が異なるため、本システムでは、AlaxalA 製スイッチと Panasonic 製スイッチ用の 2 種類のプログラムを作成する。

### 3.3 ネットワーク機器構成情報の世代管理

3.2 節では、Oxidized によってスイッチの構成情報を取得していると述べたが、Oxidized の設定によって、取得した設定情報と隣接機器情報はそれぞれ GitLab のリポジトリに保存し、情報の世代管理を行う。これにより、ネットワーク管理者が設定の変更履歴を追跡することが可能となり、設定変更があった際のトラブルシューティングや復旧作業を効率化する。現在、この処理は、日時のバッチ処理として実行している。

また、本研究では Ruby 言語のプログラムからも設定情報や隣接機器情報にアクセスする必要があるため、GitLab に保存した後でローカルディレクトリにも構成情報を保存する仕組みとしている。

これにより、スイッチの設定情報や隣接機器情報の履歴を GitLab で体系的に管理することが可能となる。

### 3.4 設定情報の自動生成

本研究では、構成管理ツール機能を、NetBox の Configuration Rendering 機能と Oxidized を用いて実装するため、機器への投入が可能なレベルまで設定テンプレートの調整を行う。今回、Panasonic 製の L2 スイッチのみを対象に生成設定情報を検証した。生成設定情報は、NetBox 上の機器情報と Jinja2 で書かれた設定テンプレートを用いて生成されるので、Panasonic 製の L2 スイッチに合わせた設定テンプレートを作成する必要がある。

次に、生成設定情報と GitLab で管理している実機の設定情報との差分を比較する機能を実装するため、netbox-config-diff プラグイン[6]を追加した。これにより、本来の理想的な設定情報である生成設定情報と実機の設定情報との比較が可能となる。

スイッチへの生成設定情報の反映については、現在検証中である。具体的には、スイッチに送った生成設定情報を running-config に反映する手段を検証している。また、障害が発生した際のロールバック方法も検討課題となっている。

## 4 機能評価

本節では、提案手法により構築したシステムが設計通りに動作することを確認する。具体的には、管理対象のネットワーク機器を NetBox に登録し、システム全体の機能を評価した。尚、Excel ファイルの台帳で管理していた情報は、本システムでも確認することができた。次に、以下の 3 点を確認する。

- (1) 実機からの構成情報取得と登録の効率化
- (2) ネットワーク機器構成情報の世代管理
- (3) 設定情報の自動生成

確認した結果は以下の通りである。

- (1) 実機からの構成情報取得と登録の効率化に関しては、Oxidized にスイッチのモデルを作成して情報を取得し、Ruby 言語のプログラムを用いて NetBox に登録を行った。その結果、実機の情報 NetBox に登録され、NetBox 上の情報が実機に合わせて適切に更新されていることが確認できた。図 3 は NetBox の Web 画面でスイッチのインターフェース情報の登録を確認した画面である。この画面から、登録前の情報と実機の情報で差分があり、その差分が正しく修正されたことが確認できた。現在、NetBox を理想のネットワーク構成を管理するシステムとして位置づけており、実機の設定を直接変更した場合は、変更内容を確認し、必要に応じて NetBox の情報を Ruby 言語のプログラムもしくは NetBox 上で更新する方針を検討している。

- (2) GitLab によるネットワーク機器構成情報の世代管理に関しては、日常の業務においても有用性が確認された。GitLab を用いることで、機器の設定情報の変更履歴を追跡できるだけでなく、設定変更があった際にはネットワーク管理者へメールが送信される。これにより、実機の設定変更を適宜確認し、必要に応じて適切な対応を取ることが可能となる。図 4 は、実際に送信される

通知メールの画面である。



オブジェクト変更

変更

時間 2024-10-11 08:18:29

ユーザ [REDACTED]

アクション 更新済

オブジェクト DCIM | インタフェース

オブジェクト GigabitEthernet1/0/1

リクエスト ID [REDACTED]

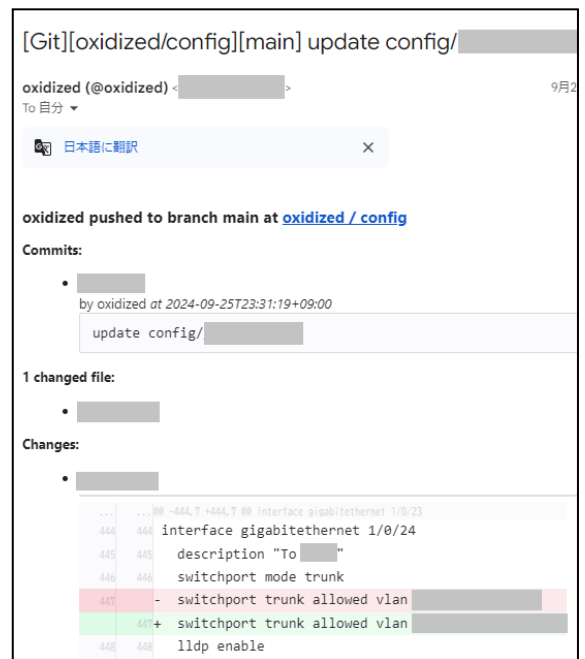
変更前のデータ

```
lag: null
mtu: null
vrf: null
vwan: null
mode: null
name: "GigabitEthernet1/0/1"
tags: []
```

変更後のデータ

```
lag: null
mtu: null
vrf: null
vwan: null
mode: "access"
name: "GigabitEthernet1/0/1"
tags: []
```

図 3 NetBox 上のインターフェース登録情報確認画面



[Git][oxidized/config][main] update config/ [REDACTED]

oxidized (@oxidized) <[REDACTED]> 9月2

To 自分

日本語に翻訳

oxidized pushed to branch main at oxidized / config

Commits:

- [REDACTED] by oxidized at 2024-09-25T23:31:19+09:00  
update config/ [REDACTED]

1 changed file:

- [REDACTED]

Changes:

- [REDACTED]

```
... 444 444 interface gigabitethernet 1/0/23
444 444 interface gigabitethernet 1/0/24
445 445 description "To [REDACTED]"
446 446 switchport mode trunk
447 447 - switchport trunk allowed vlan [REDACTED]
448 448 + switchport trunk allowed vlan [REDACTED]
449 449 lldp enable
```

図 4 GitLab からの通知メール

- (3) 設定情報の自動生成に関しては、NetBox 上の生成設定情報と GitLab で管理しているネットワーク機器の設定情報の差分を NetBox で確認することができた。この結果を踏まえ、NetBox の Configuration Rendering 機能の利点は、以下の点であると考える。

- NetBox の Web 画面から理想の設定

情報を確認できる。

- 実機の設定情報との差分を NetBox 上で確認できる。
- 差分比較によって、不要な VLAN 情報等を整理できる。

一方で、生成設定情報には実機の設定情報の差分としてスイッチへの反映に影響がない項目まで抽出されることがある。具体的には、コメントや記述順序等が挙げられる。また、本研究では、Panasonic 製の L2 スイッチでしか生成設定情報を確認できていないため、AlaxalA 製のスイッチで同様の運用が可能かは未検証である。

上記を踏まえても、ネットワーク機器に設定情報を反映させる前に設定内容と比較できる点は、不必要な設定変更を避ける上で有益であると考えている。尚、生成設定情報の実機への反映については、現在も検証段階にあり、本研究では評価を行うことができなかった。ただし、Panasonic 製の L2 スイッチを 1 台用いて、以下の 2 点を検証できた。

- Ruby 言語プログラムによって NetBox 上の生成設定情報を取得できること。
- 生成設定情報を Oxidized によってネットワークスイッチに送ること。

これらの問題については、現在も検証中である。

## 5 おわりに

本研究では、システム更改の紹介と、ネットワーク管理者のための管理効率化を目的とした、ネットワーク機器管理のシステム構築に取り組んだ。その結果明らかになった課題については、引き続き検証を進める予定である。さらに、通常業務での利用実績を積み重ねること、他にもルータや次世代ファイアウォールを管理することを目標としている。また、長期運用を見据えた際の課題として、システムのバージョンアップを含むメンテナンスと、新たな機器導入に伴う変更作業の検討が

必要であると考えている。

## 参考文献

- [1] 北口義明, 金勇, 友石正彦, OSS を活用したキャンパスネットワークの構成管理システム, 情報処理学会研究報告, Vol.2022-IOT-58, No.16, pp. 1-6, July.2022.
- [2] NetBox-community – netbox, <https://github.com/netbox-community/netbox> (2024 年 8 月 5 日 参照)
- [3] Ansible Documentation, <https://docs.ansible.com/> (2024 年 10 月 15 日 参照)
- [4] Batfish – An open source network configuration analysis tool, <https://www.batfish.org/> (2024 年 10 月 15 日 参照)
- [5] ytti - oxidized, <https://github.com/ytti/oxidized> (2024 年 8 月 5 日 参照)
- [6] NetBox Config Diff Plugin, <https://miaow2.github.io/netbox-config-diff/> (2024 年 10 月 15 日 参照)