

静的 IP アドレス設定の有無を簡易判定する Web アプリケーションの作成

○金城 行真 ^{A)}

^{A)}自然科学研究機構 分子科学研究所 技術推進部

1. 背景

分子科学研究所(以下、分子研)内の端末情報の管理は、DHCP サーバと連携した専用の管理システムによって運用されている。端末の IP アドレスを変更する場合、当該端末が DHCP 設定であれば、管理システムの WebUI 上で編集することで端末側の設定値も更新される。

一方、端末側で静的 IP アドレス設定が有効化されている場合はその限りではない。この場合、管理システム上の編集に加え、利用者に対して端末の設定変更を依頼する必要がある。また、SSID を所内無線ネットワークから別のネットワーク(eduroam 等)に切り替えた場合、端末が切り替え前の IP アドレスを保持し続けることで通信不能に陥る、といった課題も存在する。

管理者側としては、静的 IP アドレス設定の有無を遠隔から判定できることが好ましい。しかし、実際は管理者がユーザの元へ出向き、端末側の設定を直接確認していた。従って静的 IP アドレス設定に起因する通信不能などのトラブルが発生した場合、原因の特定までに時間を要し、管理者・利用者双方の負担となるケースが散見されていた。

そこで、端末の IP アドレス及び MAC アドレスを予め収集したデータと照合し、静的 IP アドレス設定の有無を簡易的に判定する機能を開発した。更に本機能を Web アプリケーションとしてデプロイした。考案した判定方法や技術的構成、実際の使用ケースについて報告する。

2. 静的 IP アドレス設定の判定方法

分子研の所内ネットワークにおいて、端末が IP アドレスを取得する手法は、DHCP による自動割り当て、もしくは静的 IP アドレス設定のどちらかに限定される。DHCP を利用する場合、端末(DHCP クライアント)は LAN 上に DHCPREQUEST メッセージを送出する。これは、一度 DHCP サーバから IP アドレスのリースを受けた端末が、リースの更新(延長)を要求する場合も同様である。つまり、DHCPREQUEST の送出状況を、当該端末における静的 IP アドレス設定の有無を判定する指標として活

用できる。

また、現行のネットワークの設計上、LAN に接続した端末が他端末と通信する際、各棟に配置されたフロアスイッチを経由する。そのため、あるタイミングでどの端末がネットワークに参加しているかは、フロアスイッチが保持する MIB 情報(Management Information Base) から把握することが可能である。

従って、ある端末から DHCPREQUEST が送出されておらず、かつフロアスイッチ上で当該端末の接続を示す MIB 情報が取得できた場合、その端末は DHCP を介さず IP アドレスを保有している。すなわち、静的 IP アドレス設定が施されていると判定できる。

3. バックエンドの作成

以上の方針を定めた後、判定システムの構築に着手した。まず、判定材料となる DHCP ログ(DHCPREQUEST)及びフロアスイッチへの接続情報を収集・管理するバックエンド環境を設計した。DHCP ログについては、既存のログ収集サーバに集積されているデータから DHCPREQUEST メッセージのレコードを抽出し、データベース(MariaDB)に格納することにした。一方、フロアスイッチへの接続情報については、各スイッチに対して snmpwalk を実行し、現在接続されている端末の IP アドレス及び MAC アドレスを取得し、同じデータベースに格納する手法を採った。

システム基盤として、Rocky Linux 9.7 で稼働する仮想マシンを構築し、MariaDB 10.5.29 をインストールした。データベース内には、DHCP ログ用および SNMP データ用のテーブルをそれぞれ作成した。各データの収集・登録処理には、OS 標準の Python 3.9 を用いてスクリプトを実装した。加えて、保持期間が 1 年を経過した古いレコードの削除や、テーブルの最適化などを行うデータベース保守用スクリプトも併せて Python3.9 で実装した。これらのスクリプトは、systemd.timer を用いて定期的に自動実行されるよう構成している。

以上の実装により、静的 IP アドレス設定の判定に必要な情報を、一元的に収集・管理するバックエン

ドの仕組みが完成した。

4. フロントエンドの作成

続いて、収集したデータベース内のレコードへ Web ブラウザ経由でアクセスし、静的 IP アドレス設定の判定を行うためのフロントエンド環境を構築した。

Web フレームワークには Django を採用した。Rocky Linux 9 の標準パッケージ(dnf)で提供される Django 3.2 は旧バージョンであったため、長期サポート(LTS)対象である Django 5.2 系の導入を決定した。しかし、Django 5.2 系は Python 3.10 以降を要求するため、OS デフォルトの Python 3.9 環境とは分離し、venv を用いて Python 3.11 の仮想環境を新たに構築した。その上で pip により Django 5.2.7 をインストールしている。その後、Django の関連ファイル (views.py 等) を編集し、ローカルの MariaDB から DHCP ログおよびフロアスイッチの接続情報を検索する機能を開発した。

WebUI の構築においては、視認性と操作性を高めるため CSS フレームワーク Bootstrap の拡張ライブラリである django-bootstrap5 を導入した。また、本番環境の Web サーバとしてのデプロイは Apache 2.4.62 上で実施した。

完成した WebUI を図 1 に示す。

図 1 Web アプリケーションのフロントページ

利用者がフロントページのフォームに IP アドレスまたは MAC アドレスを入力して検索を実行すると、バックエンドで各テーブルへの照会が行われる。その検索結果(DHCP ログと接続情報の有無)に基づき、以下の 4 パターンのいずれかの判定結果が画面上に表示される仕組みとなっている。

- 両方のログがヒットした場合、DHCP 設定で IP アドレスを取得していると判定する。
- フロアスイッチへの接続情報のみがヒットした場合、静的 IP アドレス設定の可能性が高いと判定する。

- DHCPREQUEST のログのみがヒットした場合、ネットワーク上に端末が存在しないにもかかわらず DHCP 要求の記録があるため、データベースの収集に不整合が生じていると判定する。
- 両方のログがヒットしない場合、ネットワーク未接続、または端末の電源がオフである等の理由により、判定不能とする。

ある IP アドレスで検索し、静的 IP アドレス設定と判定された際の様子を図 2 に示す。

図 2 静的 IP アドレス設定と判定される例

5. 実際の使用ケース

分子研のネットワーク環境では、1 つの部門(研究グループ等)に VLAN を 1 つ割り当て、/24 で切る構成が標準的である。しかし、特定の部門において IoT 機器の接続台数が増加しており、割り当てられた IP アドレス空間が枯渇する懸念が生じていた。

この問題を回避するため、当該部門へ追加の VLAN を割り当て、利用可能な IP アドレスを拡張することとした。しかし、単純に新規端末を追加 VLAN に収容した場合、既存 VLAN に属するプリンタ等のブロードキャストが新 VLAN の端末へ到達しない、といった運用上の課題が予見された。そこで、日常的に使用する PC やプリンタ等を一斉に新 VLAN へ移行し、IoT 機器等を既存の VLAN へ残留させ、新規端末はその性質によって所属させる VLAN を割り振るセグメント分離運用を実施することとした。このネットワーク環境の更新に伴い、新 VLAN へ移行する端末において IP アドレスの変更作業が必要になった。

移行作業に先立ち、対象となる端末群に対して開発した Web アプリケーションを適用し、静的 IP 設定の有無を事前に判定した。その結果、数台の PC、プリンタ、および NAS において静的 IP アドレスが設定されていることを検知できたため、該当機器の利用者へ事前に情報共有と注意喚起を行った。運用

上の事情により、これらの機器を事前の段階で DHCP 設定へ変更することは見送られた。しかし、「どの端末で静的 IP アドレス設定に起因するトラブルが発生しうるか」を事前に把握できていたため、移行時のトラブル発生に対して、原因の切り分けや復旧対応をスムーズに行うことができた。

結果として、冒頭で課題として挙げた「原因特定に長時間を要し、管理者・利用者双方の業務負担となる」事態を回避することができ、実運用における本システムの有効性を確認できた。

6. 謝辞

この Web アプリケーションを作成するにあたり、岩橋建輔氏、神谷基司氏、内藤茂樹氏、長屋貴量氏（以上、分子研）に技術面・運用面での多数の助言という形でご協力いただいた。ここに深く感謝の意を表する。