VLAN の代替となる新しい仮想ネットワーク構成方式 vNet の OpenFlow による実装

髙藏 駿之 † 石橋 勇人 †
Toshiyuki Takakura Hayato Ishibashi

1 はじめに

企業や大学などの組織については，一定の区切り（部署や権限）で 1 つのネットワークを形成し，組織を分割してネットワークを割り当てることが一般的である．

組織内において，権限のない人がネットワーク内の機密性の高い情報にアクセスできないようにするためには，VLAN 間の通信を一部あるいは完全に遮断することが行われる．

また，組織内ネットワークであっても外部との通信は必要不可欠である．そのため，組織内の端末が攻撃を受ける可能性があり，この攻撃によってその端末がアクセスできる範囲にまで攻撃の影響を受ける可能性がある．そこで，外部からの攻撃の影響を最小限に抑え流ためには，組織内をできるだけ小さいネットワークに分割しておくことが望ましい．

こういった目的のために，現在は IEEE802.1Q[1] に基づく VLAN が使用されているが，この方式には一定の制約もある．本研究ではこの制約を克服する OpenFlow を用いた新たなネットワーク方式を提案する．

2 研究背景

2.1 現在のネットワーク方式とその課題

1 章で述べたように，組織は VLAN によって部署や権限などの単位でネットワークを分割することが一般的である．さらに，セキュリティの面を考慮して分割ネットワーク間の通信が遮断されているときの VLAN に対する問題点を挙げる．なお，前提としてネットワークに接続する際にユーザ認証が求められ，成功すれば接続することができるネットワークであるとする．

1 つ目は，分割ネットワーク数の上限である．規格 IEEE802.1Q に基づいている VLAN において，ヘッダ内にある VLAN ID というフィールド値を変えることでネットワークの分割を実現している．しかし，このフィールドには 12bit 分しか用意されておらず，4094 より多くネットワークを分割することはできない．この分割数の上限によって大規模な組織については分割をする際に数が不足する可能性があり，想定していたネットワーク構成の実現が不可能になる．

2 つ目は，利便性を損ねているということである．異なる VLAN 間のネットワーク的なアクセスが許可されていない状態 (セキュリティ上の要請が想定されるが，それに限定しない) において，VLAN1 に接続された端末 A から，異なる VLAN である VLAN2 に接続されたリソース (サーバやプリンタなど) を利用しようとする場合，(1) 端末 A に複数の NIC を用意して複数の VLAN に同時接続する，(2) 端末 A を VLAN2 に接続し直す，(3) 端末 A から VLAN2 に対して VPN 接続を行う，などの方法が考えられる．しかし，これらはいずれも利用者に負担を強いることとなり，利用者の利便性を損ねていると言える．

2.2 関連研究

古賀ら [2] は，大規模組織に対して OpenFlow を用いた利便性の高い仮想ネットワーク構成方式を提案した．この研究では，VLAN の規格で定められた分割数の上限以上にネットワークを生成すること，複数ネットワークへの同時接続，そしてネットワーク接続にかかる時間の短縮を目的として提案された．この提案方式では，パケットの IP アドレスを OpenFlow スイッチで書き換えることで分割ネットワークを実現させている．端末がアクセスできるホストをまとめ，1 つのネットワークを仮想的に作り，その仮想ネットワークに論理的な IP アドレスを割り当てる．すると，端末からネットワークを見ると自分がアクセスできるホストが同じネットワークに属しているように見え，ネットワークの分割が可能になる．これに対して本研究では，OpenFlow スイッチのフローエントリを使用したパケット制御によって仮想的なネットワークを構成することを考える．

橋本ら [3] は，OpenFlow を用いて一般的なディレクトリサービスの 1 つである LDAP と認証基盤を連携させることによってネットワーク自体へのアクセス権限をより柔軟に制御できる仕組みを提案した．この提案ではキャンパスネットワークを対象としたアクセス制限を考えているが，本研究ではキャンパスネットワークより大きさが小さいネットワークを対象としたアクセス制限について考える．

津田ら [4] は SDN の一種である，複数のコントローラでスイッチを制御する HyperFlow と MPLS ラベル，そして階層型 SDN アーキテクチャである HARP（Hierarchical

† 大阪公立大学 大学院 情報学研究科

ARea Partitioned）モデル [5] を用いるルーティング手法を提案した．OpenFlow では原則 1 つのコントローラで複数のスイッチを制御するが，コントローラの負荷を分散するために複数のコントローラを用いることがある．その手法の 1 つに HyperFlow がある．HARP モデルはコントローラをエリアと呼ばれる一定の範囲ごとに 1 つずつ設置することで負荷を分散させているが，隣接エリアとの通信ができなくなると負荷がかかってしまう．また，モデル特有の問題であるパケットのループが生じる可能性があるため，これらの問題を解決するためにパケットに対して VLAN タグを付加し，エリア外が宛先であれば MPLS ヘッダを付加することでルーティングを行なった．この提案手法では MPLS のヘッダラベルにエリアを示す値を格納していたが，本研究ではユーザを示す値をヘッダラベルに格納する．

3　要素技術

3.1　OpenFlow

SDN を実現するために様々な方法があるが，その 1 つとして OpenFlow がある．OpenFlow はネットワーク機器の機能を 2 つに分割し，それぞれ独立して動作させる技術である．具体的にはパケットの経路制御をするコントロールプレーン，データを保存するデータプレーンの 2 つの機能で，それぞれの機能を担う機器を OpenFlow コントローラ，OpenFlow スイッチと呼ぶ．そして，これら 2 つの機器間での通信で使われるプロトコルを OpenFlow プロトコルという．

3.2　MPLS

ヘッダに固定長のフィールドを付与することによって経路制御する経路方式をラベルルーティングといい，この方式によってデータ転送を高速化できる技術に MPLS がある．MPLS ではラベルフィールドに 4 バイトの容量が確保されている．

4　提案方式

4.1　新しいネットワークに求められる要件

2.1 で説明した VLAN の問題点を解決するために，新しいネットワークには以下の要件が求められる．

きめ細やかなネットワーク　外部ネットワークから内部ネットワークへの攻撃に対する影響を最小限に抑えるためには，分割ネットワークの大きさを限りなく小さくすることが求められる．したがって，IEEE802.1Q に基づく VLAN においては分割ネットワークの上限数以上にネットワークを分割できる必要がある．

異なる仮想ネットワーク上のリソースの利用　2.1 で述べた 2 つ目の課題を解決するため，本研究では，利用者の端末が接続されている VLAN(VLAN1 とする) とは異なる VLAN に接続された個別のリソースのうち，その利用者からアクセス可能なものが仮想的に VLAN1 に接続されているような仮想ネットワークを構成し，利用者からはシームレスなアクセスを実現する．

4.2　提案ネットワーク方式

前節の要件を満たし，かつ 2.1 の問題点を解消するようなネットワークの実現のために [2] で提案された方式 vNet を基本とし，いくつかの改良を加えたものが本研究の提案ネットワーク方式である．

4.2.1　ネットワークアーキテクチャの概要

図 1 は提案ネットワークの機器構成である．提案ネットワークにおいて，スイッチの層を考える．この時，最下層にあるスイッチを OpenFlow スイッチとし，そのスイッチに各ホストやサーバを接続する．これにより，OpenFlow を使ってスイッチを制御することが可能となり，各ホストが送受信するパケットの制御も可能となる．また，仮想的に複数のネットワークを形成させることも可能となる．さらに，ネットワーク分割の単位は VLAN では部署や権限などの大きめのまとまりごととしていたが，本研究においてはより細かいまとまりであるチームやユーザなどに対して分割ネットワークを割り当てることを考える．すると，1 つのネットワークにおいて当該ユーザに利用や通信が許可されている全てもしくは一部のホストが接続するため，各ユーザがアクセスの許可されたすべてのホストと 1 つのネットワーク内で通信できるようになる．

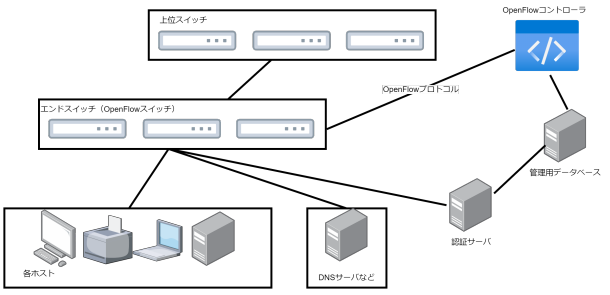

図 1: 提案ネットワークの機器構成

4.2.2　ネットワークの動作概要

ここでは，ホストがスイッチに接続された際，ならびに，ユニキャスト，マルチキャスト，ブロードキャスト

の各通信のパケット処理におけるスイッチの挙動について説明する．

接続時　ホストがスイッチに接続された時，まず IP アドレスを割り当てる．この時点においてホストは認証前の状態であるため，通信できる端末を認証サーバのみに制限する．また，その他の可能な通信として宛先 MAC アドレスがサーバのアドレスである ARP パケットがある．これらの通信のみに制限するために，OpenFlow スイッチが接続した際に認証前の端末が可能な通信が条件となるフローエントリを初期ルールとして保存する．

ユニキャスト　通信において，考えなければならないのは通信の送信元ホスト，宛先ホストが同じ分割ネットワークに接続しているかである．提案ネットワークを実現させるには両ホストが同じ分割ネットワークに属していれば通信を許可し，属していなければ通信できないようにしなければならない．つまり，両ホストが同じ分割ネットワークに属していればパケットを宛先へ転送し，属していなければパケットを破棄する必要がある．また，通信するためには両ホストが認証済である必要もある．よって，両ホストがともに認証済であればパケットを宛先へ転送し，そうでなければパケットを破棄することが求められる．

マルチキャスト　同じネットワークに接続しており, ユーザがアクセス許可されている認証済みのホスト A〜D を考える．マルチキャストアドレスが 239.1.1.1 のグループにホスト B，C が属しているとする．この時，ホスト A がマルチキャストアドレス 239.1.1.1 へ通信を行う際，ホスト A からグループに含まれているホスト B，C への通信は許可されるため，2 つのホストにパケットが転送され，属していないホスト D には転送されない．また, 別ユーザが同じマルチキャストアドレス 239.1.1.1 に通信を行うと，この場合ではアクセスが許可されていないホストのみが所属しているのでパケットは破棄されるが，そのユーザのアクセス許可があるホストが属していればそのホストのみにパケットが転送される．

ブロードキャスト　ブロードキャストパケットがホストから最下層スイッチに入ってきた際には，そのパケットを上位スイッチに転送する．最下層スイッチでは，サブネット外のホストにパケットが届くことがないように通信の可否をパケット情報から判断し，ポートごとに転送もしくは破棄を行う．

5　実装

実装において，プログラムとソフトウェアのバージョンは表 1 の通りである．

表 1: バージョン情報

OpenFlow	1.3.4 [6]
os-ken	2.9.0 [7]
Python	3.12.3 [8]

5.1　システム実装

4 章で説明した方式に則って，実装を行う．なお，今回は DHCP や認証サーバに関しては専用のサーバを用意せず，コントローラ内に簡易な機能を実装している．

5.2　コントローラプログラムの動作

5.2.1　コントローラが保持する情報

本システムにおいて，コントローラが保持する情報は以下の通りである．

- 接続端末の情報
  - MAC アドレス
  - IP アドレス
  - 接続スイッチの dpid
  - 接続ポート番号
- 接続スイッチの情報
  - MAC アドレス
  - フローテーブル
  - グループテーブル
  - 接続ネットワーク ID
  - 接続スイッチの dpid
- ネットワークトポロジに関する情報
  - 各スイッチの dpid
  - 認証サーバの位置

5.2.2　新規スイッチ接続時の動作

ネットワークに OpenFlow スイッチが接続された場合，コントローラは初期ルールをスイッチに送信し，保存させる．表 2 に初期ルールを示す．

表 2: 初期ルールのフローエントリ

優先度	マッチ	アクション
4	IP プロトコル = ARP 宛先 MAC アドレス =サーバのアドレス	上位スイッチへ転送
4	宛先 MAC アドレス = サーバのアドレス	上位スイッチへ転送
4	受信ポート = 上位スイッチの接続ポート 送信元 MAC アドレス = サーバのアドレス	宛先へ転送
0	すべてのパケット （無条件）	破棄

5.2.3 端末の接続

新規接続された端末から認証サーバにパケットが送られると，スイッチがそのパケットをコントローラへ転送し，含まれている情報から端末の情報を保存し，端末と接続されているスイッチに対してパケットが出入りできるようにフローエントリを送信する．表 3 に端末に関するフローエントリを示す．

表 3: 端末に関するフローエントリ

優先度	マッチ	アクション
2	受信ポート = 上位スイッチの接続ポート 宛先 MAC アドレス = 端末のアドレス	宛先 MAC の端末の接続ポートへ転送
1	受信ポート = 端末の接続ポート 宛先 MAC アドレス = 端末のアドレス	コントローラへ転送

5.2.4 ユニキャスト

ユニキャスト通信に対しては，送信元の端末が接続されているスイッチでその端末が通信可能かを判断し，宛先の端末が接続されているスイッチで 2 つのホスト間での通信が許可されているかを判断する (表 4).

表 4: 認証済端末間の通信に関するフローエントリ

優先度	マッチ	アクション
2	受信ポート = 送信元端末の接続ポート 送信元 MAC アドレス = 送信元端末のアドレス 宛先 MAC アドレス = 宛先端末のアドレス	宛先端末へ転送 （上位スイッチ or 宛先端末の接続ポート）

5.2.5 マルチキャスト通信

マルチキャスト通信を行う際，受信するホストについてはグループに参加をしなければならない．その参加表明として IPv4 では IGMP パケットが使われる．このパケットをスイッチによってコントローラへ転送し，参加するグループのマルチキャストアドレスとユーザ情報が保存される．またそのマルチキャストアドレスを宛先としたパケットが送信された時に宛先のグループへパケットを転送されるようにフローエントリをスイッチへ送信する．IGMP パケットに対するフローエントリ (表 5) は端末が認証された際に追加される．

表 5: 認証の際に追加されるフローエントリ

優先度	マッチ	アクション
3	IGMP パケット 送信元 MAC アドレス =端末のアドレス	コントローラへ転送

ここでの課題として，同じマルチキャストアドレスを異なるユーザで使用する可能性がある．提案ネットワークでは異なるユーザのホストに対してはパケットを転送しないこととしているので，ユーザごとでグループを区別させてパケットを転送する必要がある．ここで使用するのが MPLS ヘッダである．MPLS ヘッダはパケットにラベルをつけることができるため，今回の実装ではラベルとしてユーザ ID を持った MPLS ヘッダをマルチキャストパケットに付加することでパケットのみでユー

ザを特定し，かつマルチキャストグループを特定することができる．表 6，7 にマルチキャスト通信に関するフローエントリおよびグループエントリを示す．

表 6: マルチキャスト通信に関するフローエントリ

優先度	マッチ	アクション
テーブル 0		
4	MPLS パケット	テーブル 3 へ
2	宛先 IP アドレス = マルチキャストアドレス	テーブル 1 へ
テーブル 1		
0	受信ポート = 上位スイッチの接続ポート	テーブル 3 へ
0	受信ポート = 下位スイッチ・ホストの接続ポート	テーブル 2 へ
テーブル 2		
1	受信ポート = 送信元端末の接続ポート	ラベルがユーザ ID の MPLS ヘッダを付加 グループ ID：1 を実行
テーブル 3		
0	MPLS ヘッダのラベル値 = ユーザ ID	MPLS ヘッダの取り外し メタデータをラベル値であったユーザ ID に設定 テーブル 4 へ
テーブル 4		
1	メタデータ = ユーザ ID 宛先 IP アドレスがある 1 つのマルチキャストアドレス	ユーザとマルチキャストアドレスによって一意に決まるグループ ID:F(userID, multi-a) のアクションを実行

表 7: マルチキャスト通信に関するグループエントリ

グループ ID	アクション
1	上位ポートへ出力 テーブル 3 へ
F(userID, multi-a)	ID が userID のユーザの端末のうち，マルチキャストアドレスが multi-a のグループに入っている端末に対してパケットを転送する

5.2.6 ブロードキャスト

提案ネットワーク方式におけるブロードキャスト通信とは，同じネットワークに接続しており，かつ認証済であるホストすべてにパケットを送信する方法である．そのためブロードキャスト通信を行う際には，パケットをネットワーク内の認証済ホストにのみ転送させる必要がある．そのため，スイッチは複数のホストにパケットを転送することが可能になるグループテーブルを使用してブロードキャストパケットを処理する．表 8 にブロードキャストに関するフローエントリを示す．

表 8: ブロードキャストに関するフローエントリ

優先度	マッチ	アクション
2	受信ポート = 上位スイッチの接続ポート 送信元 MAC アドレス = 端末 X のアドレス 宛先 MAC アドレス = FF:FF:FF:FF:FF:FF	グループ ID がユーザ ID であるグループエントリ

6 評価

6.1 実験

6.1.1 実験環境

ハードウェアとして M2 チップ搭載の MacBookAir（macOS バージョン 14.2.1，RAM:16GB）を用いて，仮想環境を作成し実験を行なった．仮想マシン制作ソフトウェアである Multipass（バージョン 1.13）[17] を用いて，表 9 の構成の仮想マシンを生成した．

表 9: 実験環境

項目	設定
Architecture	AArch64
CPU	64 bit
Machine	QEMU Virtual Machine virt-8.2
RAM	1GB
Guest OS	Ubuntu 22.04.4 LTS

6.1.2 ネットワークトポロジ

図 2 に使用したネットワークトポロジを示す．◯はホスト，□はサーバ，▱はスイッチを示す．

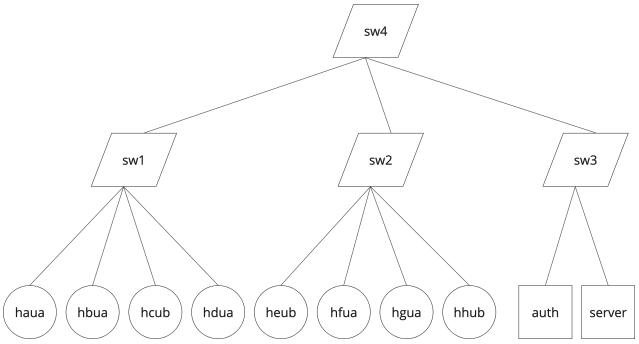

図 2: 実験ネットワークトポロジ

6.1.3 実験内容

実験では，ping コマンドを使って通信することでネットワークが動作しているかを確認した．また，IGMP パケットなどの特殊なパケットについてはプログラムを用いてパケットを生成し，送信することで動作を確認した．

この実験において，確認すべき点は以下の通りである．

- 認証前のホストが未認証端末との通信が可能なホストと通信ができるか
- ホストから DHCP Discover パケットを送信すると，コントローラによってパケット情報が表示されるか
- 通信が許可されているホスト間で通信が可能であるか
- 通信が許可されていないホスト間で通信が失敗するか
- 各ホストが，ネットワーク外のホストの存在を確認することができないか
- マルチキャストグループに参加できるかどうか
- マルチキャストパケットが届くかどうか
- 異なるユーザで同じマルチキャストアドレスを使用した際に，一方のユーザのみがアクセス許可されているホストにのみパケットが転送され，それ以外は転送されないか

6.1.4 実験結果

前節で述べた点について実験を行なったところ，すべての動作が正常であったことを確認することができた．

6.2 スイッチのフローテーブルのサイズに対する評価

提案ネットワークにおいて，スイッチのフローテーブルのサイズについて評価を行う．評価するにあたって，フローエントリを追加するタイミングに分けて考える．

5 章において示したフローエントリをもとに考えると表 10 の通りになる．

表 10: フローエントリと追加するタイミング

フローエントリ		追加するタイミング
表 2		スイッチ接続時
表 3		認証済端末が初めて通信を行う時
表 4		認証済端末が初めて他ホストとの通信を行う時
表 5		端末が認証された時
表 6	テーブル 0	ネットワーク内のホストがマルチキャストグループに初めて参加した時
	テーブル 1	
	テーブル 2	
	テーブル 3	スイッチ内のホストがマルチキャストグループに初めて参加した時
	テーブル 4	

表 10 において，フローエントリが追加されるタイミングが大きく分けてスイッチ接続時，認証済端末の通信時，マルチキャストに関する通信時の 3 つに分けることができる．以下はそれぞれのタイミングでどのようにフローエントリが追加されるかを考える．

ここで，サーバの個数については 10 台程度とする．

6.2.1 スイッチ接続時

スイッチが接続された時に追加されるフローエントリの数はサーバの数によって変化するが，サーバの数が増

えることは頻繁に起こることではないので，考慮に入れないとする．すると接続時に追加されるフローエントリの数は 4 となる．

6.2.2　認証済端末の通信時

認証済端末に関するフローエントリのうち，1 つの端末のみについての条件があるエントリは 3 つである．これらのフローエントリについてはスイッチに接続している認証済端末の個数ごとに追加される．

また，認証端末間の通信に関するフローエントリは送信元の端末と宛先の端末との組み合わせが条件となっているため，スイッチに接続されている端末ごとで，かつ他スイッチにある宛先の端末ごとにフローエントリが追加される．そして端末間の通信に関しては双方向の通信を考える必要があるので追加するフローエントリの数は 2 倍となる．

6.2.3　マルチキャストに関する通信時

認証済端末がマルチキャスト通信を行うために送信した IGMP パケットによって，まずスイッチには 4 つのフローエントリと 1 つのグループエントリが追加される．次に，ユーザが初めて IGMP パケットを送信したときにフローエントリが 1 つ追加され，さらに送信元ユーザと参加するグループのマルチキャストアドレスの組み合わせが初めて出てきた時にフローエントリを 1 つ，グループエントリを 1 つ追加する．

これらを考えると，ユーザや通信に関係なく 4 つのフローエントリと 1 つのグループテーブルが追加され，ユーザごとにフローエントリが 1 つ追加され，そしてユーザとマルチキャストアドレスの組み合わせごとにフローエントリとグループエントリの両方 1 つずつが追加される．

以上の 3 つを考えると 1 つのスイッチに関するフローエントリは数式で以下のように表すことができる．

ユーザ i に対して
スイッチの個数：M，　ユーザ数：N
スイッチ j にユーザ i の認証済端末の数：$a_{i,\ j}$
スイッチ j にユーザ i のマルチキャストアドレス数：$b_{i,\ j}$
スイッチ j に接続しているホストのユーザ数：c_j
とすると

スイッチ j のフローエントリ・グループエントリの数

$$= 4+3\sum_{i=1}^{N} a_{i,j}+\sum_{i=1}^{N}\left\{\sum_{l=1}^{M}(a_{i,j}\cdot a_{i,l})-a_{i,j}^2\right\}+5+c_j+\sum_{i=1}^{N} b_{i,j}$$

$$= 9 + c_j + \sum_{i=1}^{N}\left[3a_{i,j} + b_{i,j} - a_{i,j}^2 + \sum_{l=1}^{M} a_{i,j}\cdot a_{i,l}\right]$$

これらより，スイッチのフローテーブルのサイズはスイッチに接続する端末や異なるスイッチに接続しておりアクセス許可がされている端末が多いほどフローテーブルのサイズは大きくなり，マルチキャスト通信によって変化するフローテーブルの大きさは端末と比べると比較的小さいと思われる．

7　おわりに

本論文では，IEEE802.1Q に基づく VLAN における課題を克服するために，OpenFlow を使って新しい仮想ネットワーク構成方式 vNet の実装を行った．vNet において実験を行ったところ，提案方式が実現可能と見込まれる方式であることが示された．

今後の課題として，IPv6 の対応が考えられる．IPv6 においてはマルチキャスト通信が基本機能として備わっており，本研究は IPv4 の環境下でのマルチキャスト通信を機能として実装した．したがってこの提案は IPv6 対応における足がかりとして考えられる．

最後に，本システムの設計・実装にあたって多大な貢献をいただいた大阪市立大学大学院創造都市研究科修了生の古賀歩氏ならびに大阪市立大学工学部卒業生の寺西惇氏に感謝します．

参考文献

[1] IEEE Local and Metropolitan Area Networks–Bridges and Bridged Networks, 802.1Q Standard 802.1Q-2018 (2022). 10.1109/IEEESTD.2022.10004498.

[2] 古賀　歩，石橋勇人：OpenFlow を用いた利用者にとって利便性の高い大規模組織向け仮想ネットワーク構成方式の提案，情報処理学会研究報告インターネットと運用技術（IOT），Vol. 2019-IOT-44, No. 48, pp. 1–6 (2019).

[3] 橋本直樹, 園生　遥, 牛込翔平, 菊田　宏, 永園　弘, 廣津登志夫, 新村正明：OpenFlow による認証基盤と連携したネットワークアクセス制御の実現，情報処理学会研究報告インターネットと運用技術（IOT），Vol. 2014-IOT-24, No. 24, pp. 1–6 (2014).

[4] 津田英明，今泉貴史：エリアラベルを用いた分散型SDN・HARP モデルでのルーティング手法の提案，情報処理学会研究報告インターネットと運用技術（IOT），Vol. 2021-IOT-55, No. 2, pp. 1–7 (2021).

[5] 宮本翔平，今泉貴史：ネットワークの分割を用いたコントロールプレーン分離型 OpenFlow モデルの提案，情報科学技術フォーラム講演論文集，Vol. 12, No. 4, pp. 303–308 (2013).

[6] ONF OpenFlow 1.3.4 Test Specification-Basic, Open Networking Foundation（ONF） (online), available from ⟨https://opennetworking.org/wp-content/uploads/2013/07/OpenFlow1.3.4TestSpecification-Basic.pdf⟩ (accessed 2024-07-13).

[7] OpenStack Developers os-ken Documentation, OpenStack Foundation (online), available from ⟨https://github.com/openstack/os-ken/tree/2.9.0⟩ (accessed 2024-07-13).

[8] Python Software Foundation Python 3.12.3 Documentation, Python Software Foundation (online), available from ⟨https://docs.python.org/release/3.12.3/⟩ (accessed 2024-07-13).