IPv4 アドレスが重複しうる複数 LAN に対して クラウド連携を可能にする L2 マルチテナント方式の提案

鈴木 和宏　今井 祐二　下國 治　福井 恵右

株式会社　富士通研究所

筆者らは，ユーザの LAN とパブリッククラウド内の VM を L2VPN で結ぶコンシューマ向けアプライアンスである Cloud Junction Box(CJB)を開発している．CJB で接続した VM は，Layer-2(L2) フレームを使用したプロトコルである UPnP や Bonjour 等を使用して LAN 上の PC や周辺機器, ネットワーク家電と直接接続し連携を行うことができる．一般的にユーザの LAN には，プライベート IPv4 アドレスが割り振られることが多いため，L2 で接続する場合には複数のユーザ LAN の間で頻繁にアドレスの重複が発生する．本稿では，IPv4 アドレスの重複があっても, クラウド上のひとつの VM で複数のユーザ LAN に L2 サービスを提供できる L2 マルチテナント方式について提案する．本方式を実装し，KVM の VM 上に実験環境を構築して実験を行った結果，IPv4 アドレスが重複している場合においても正しくユーザを区分して L2 サービスを提供できることを確認した．本方式によって，L2 接続によるクラウド連携において，ICT 資源の使用効率を向上させることができる．

A Proposal of the L2 Multi-tenant System for Providing Cloud Interaction to LANs with IPv4 Address Conflicts

Kazuhiro SUZUKI　Yuji IMAI　Osamu SHIMOKUNI　Keisuke FUKUI

FUJITSU LABORATORIES LTD.

We are developing the Cloud Junction Box which is an appliance for the consumers who connect a user’s LAN to a VM on the Cloud with L2VPN. A VM connected by CJB can cooperate by connecting with PC and peripherals on LAN, home appliances. Generally, since the private IPv4 address is assigned in many cases, the IP address conflict occurs frequently among several users' LANs. In this paper, we propose the L2 multi-tenant system which can provide separated L2 services for every user by one VM, even if there are IPv4 address conflicts.

1. はじめに

VPC(Virtual Private Cloud)[1][2] とは，クラウドの仮想マシン(VM)とユーザのイントラネットを VPN (Virtual Private Network)で接続する IaaS の利用形態で，異なる企業システムを単一のクラウドに安全に区分収容できる．筆者らは，ユーザの LAN と VM を L2VPN で結ぶコンシューマ向けアプライアンスである Cloud Junction Box(CJB)を試作した[3]．CJB で接続した VM は, Layer-2(L2) フレームを使用したプロトコルである UPnP や Bonjour 等を使用して, LAN 上の PC や周辺機器，ネットワーク家電と直接接続し連携を行うことができる．ただし，一般的にユーザの LAN には，プライベート IPv4 アドレスが割り振られることが多いため，複数ユーザの LAN と単一の VM とを L2 で接続する場合にはアドレスの重複が発生する可能性がある．このためひとつのユーザの LAN に対してひとつの VM を用意する必要があった．

本稿では，IPv4 アドレスの重複がある複数のユーザ LAN に，ひとつの VM で L2 サービスを提供できる L2 マルチテナント方式について述べる．プライベート IPv4 アドレスを持つパケットを, v4v6 変換機構によって IPv6 リンクローカルアドレス空間に変換して VM に接続する. VM は RFC 3493[4]準拠の socket API の Scope ID を使って，ユーザ LAN を区別した処理が可能になる．これにより L2VPN を通じた LAN サービスを提供するクラウド上の 1 台の VM で，複数のユーザ LAN にサービスを提供することが可能となり，ICT 資源の使用効率を向上させることができる．

以下，次節で筆者らが試作した CJB の基本的な構成を述べる．続く 3 節で提案する L2 マルチテナント方

式について説明し，4 節で本方式の実装について詳述する．最後に本方式の評価を行う．

2. Cloud Junction Box

図 1 に CJB の概要を示す．CJB は Layer-2(L2) over Layer-3(L3)トンネリング技術に基づく仮想ネットワーク制御技術で，ユーザとクラウド間を安全・安心・簡単に L2 接続するためのネットワークアプライアンスである．

L2 over L3 トンネリング技術は，上位層に対して仮想的に L2 ネットワークを提供するもので，実装はいくつか知られている．例えば，OpenVPN, GRE, L2TPv3, VXLAN，NVGRE 等が提案・実装されている．CJB ではユーザの LAN とクラウド上の VM を接続する L2 トンネリング方式は問わない．

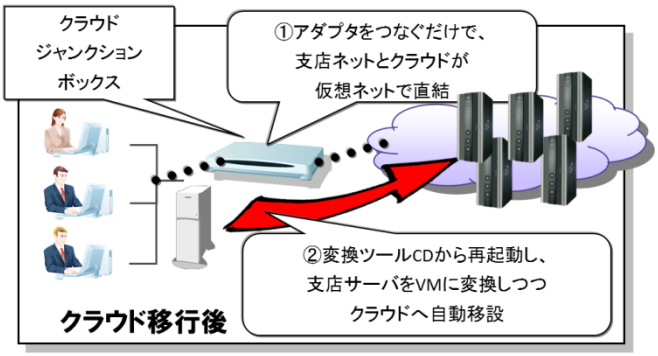

図 1 CJB の概要

ユーザが CJB と呼ぶネットワークアプライアンスを LAN に接続すると，これを契機としてクラウドデータセンタ内に L2 トンネルの端点とユーザ専用の L2 セグメントが生成される．生成された L2 トンネル端点は CJB ゲートウェイと呼び，CJB との間で L2 トンネルが開設され，LAN の L2 セグメントがクラウドに延伸される．L2 セグメントと L2VPN はユーザ専用となるように，パブリッククラウドの他のユーザ VM やネットワークとは隔離され，仮想的なプライベートクラウド(VPC)を構成する．

これと同時に，あらかじめ決められたスペックのサービス VM が起動されてユーザの LAN のセグメントに接続される．サービス VM ではファイルサーバやウェブサーバ等のアプリケーションを動作させる．CJB が構成する L2 接続によって，ファイルサーバやアプリケーションサーバは，設置場所が LAN 内にあるのか，クラウド上で動作しているのかが隠蔽されるため，ユーザは物理所在を意識すること無くサービスを受けることができる．

また，クラウドと LAN が同一の L2 ネットワークにあることを利用することで，オフィスや家庭に設置されたサーバをクラウド上に P2V(Physical to Virtual)移行することが容易になる．移行した後でもユーザ端末から，Windows 共有フォルダやプリンタの共有，AirPlay 機器への出力などのサービスを移行前と同じ操作で利用し続けることができる．移行に際して，NetBIOS，Bonjour などの L2 マルチキャストによる自動ノード発見機構で必要なアドレス解決を行うため，ユーザが持っているクライアント PC で，ファイルサーバやプリンタのホスト名の設定を変更する必要がない．

3. L2 マルチテナント方式

3.1. 概要

VPC においては，パブリッククラウド上の他のユーザとの処理とデータの区分は，アプリケーションが動作する VM およびストレージを区分することで実現する．しかし，SOHO 向けのファイルサーバや DHCP サーバ等のような CPU はネットワークの使用率が比較的低いシステムを動作させる場合には，物理計算機上にユーザ毎の VM を配備すると，VM のオーヘッドが発生するために ICT 資源の使用効率が低下する．一方，PaaS や SaaS では，VM とストレージを他のユーザと共有しながら，アプリケーションロジックでユーザ毎に処理およびデータを区分する．CJB においても，クラウドサービスに必要な ICT 資源の利用形態によって，ユーザ毎に VM を起動することによる処理区分と，アプリケーションレベルでの処理区分を使い分けることによって，クラウドの利用効率を向上させることが期待できる．

ユーザを区分するために，Jail や lxc 等のコンテナを利用する方式も考えられる．コンテナ方式はユーザ毎に VM を起動するよりも区分のためのコストが低いが，コンテナごとにアプリケーションプロセスを起動する必要がある．今回は，単一 OS 上の単一アプリケーションプロセスで複数のユーザにサービスを提供することで，さらに低コストでサービスの重畳が可能なアプリケーションレベルでの区分方式について述べる．

複数ユーザの LAN をひとつの VM に収容するには，複数のユーザ LAN と VM を L2 で接続する必要がある．この際，異なるユーザが自身の LAN で利用している IPv4 アドレスが高い頻度で重複することが考えられる．ユーザの LAN セグメントは NAT ルータによってプライベート IPv4 アドレスが割り当てられていることが多い．NAT ルータをデフォルト設定のまま利用した場合，“192.168.0.0/24” などのプレフィックスを持つアドレスを複数のユーザが利用することになる．

このような状況においても，従来のインターネット

接続経由のクラウド利用であれば NAT 変換後のグローバル IPv4 ソースアドレスによってユーザを区分することが可能であるが，L2 接続した場合にはアドレスの重複によって通信が行えない上に，ソースアドレスも同一となる可能性があるため区分も不可能となる．さらに，ネットワークプレフィックスが同一の複数の LAN を一つの VM に L2 で収容した場合には，ゲスト OS のルーティングテーブルに同じ IPv4 プレフィックスに対して重複する経路情報が作成されることになり，正しくネットワークへのルーティングを行うことができなくなる．

本稿では重複する IPv4 アドレスが割り当てられた複数のユーザ LAN をクラウド上の単一のサービス VM に L2 接続した場合でも，個々のユーザを区分しつつサービスを提供することができる L2 マルチテナント方式を提案する．

図 2 に，提案する L2 マルチテナント方式の構成図を示す．IPv4 端末からのパケットは CJB を通してクラウド内のトンネル端点である CJB ゲートウェイに到達する．CJB ゲートウェイはトンネルを解除し，ユーザ端末と同一セグメントの L2 パケットとして v4v6 変換モジュールに渡す．v4v6 変換モジュールは RFC6145[5] のステートレス変換と同様に，入力されたパケット内の IPv4 アドレスを IPv6 リンクローカルアドレスに置換えながら NAT 変換する．この変換は RFC 6145 に準拠することで，セッション管理や IPv4 アドレスと IPv6 リンクローカルアドレスのマッピング管理を静的にするため，変換の実装をステートレスにすることが可能となる．

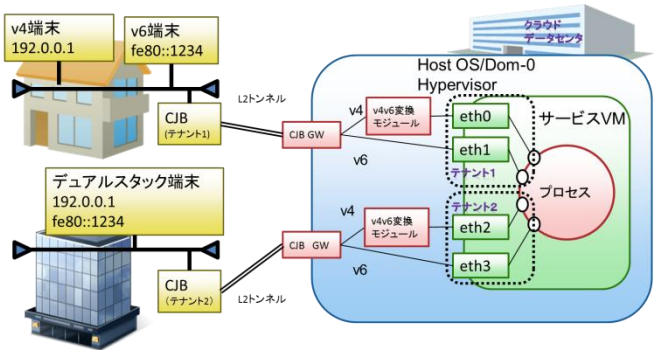

図 2 L2 マルチテナント方式の構成図

IPv6 リンクローカルアドレスは一つのセグメント内での一意性を保証しているが，複数のセグメント間での一意性は保証していない．RFC 3493 は，IPv6 リンクローカルアドレスの末尾にインターフェース番号やインターフェース名を指定する．Scope ID と呼ばれる識別子で，どのインターフェースに接続されたセグメントのリンクローカルアドレスかを特定することを可能にする方法を提供している．

ユーザ LAN の IPv4 アドレスから IPv6 リンクローカルアドレスへのマッピングは，IPv4 アドレスに“fe80::”を付加することで行う．具体的な変換例を以下に示す．

変換前	変換後
192.168.0.10	fe80::192.168.0.10 (fe80::c0a8:a)
192.168.0.100	fe80::192.168.0.100(fe80::c0a8:64)

IPv6 リンクローカルアドレスに変換したパケットは VM のインターフェースに伝えられ，VM 内で動作するアプリケーションプロセスで処理される．

異なるユーザ LAN 上の重複した IPv4 アドレスを持つクライアントからのパケットは，変換後の IPv6 リンクローカルアドレスも重複する．しかし，IPv6 リンクローカルアドレスは同一アドレスであっても Scope ID によって，どのインターフェースから到達したパケットであるかを識別することができる．

また図 2 では IPv6 端末も示されているが，本方式では LAN 内の IPv6 端末にはリンクローカルアドレスが割り当てられていることを想定している．これらのアドレスがユーザ間で重複した場合でも，IPv4 の場合と同様にサービス VM 側で別のインターフェースに接続されるため，ユーザを区分することができる．IPv6 端末にグローバルアドレスが割り当てられる環境でも，サービス VM に IPv6 リンクローカルアドレスのみが割り当てられているため，ユーザを区分することが可能である．

これにより，VM 上のプロセスは入出力インターフェースによってクライアントが属するユーザ LAN を区別しつつ，重複アドレスを含むすべての IPv4 アドレスを持つクライアントと通信が可能となる．

3.2. v4v6 変換モジュール

ユーザ LAN の IPv4 クライアントとクラウド上の IPv6 VM の通信を可能にするため，IPv4 データグラムと IPv6 データグラムを NAT64 で双方向に変換するモジュールを作成した．RFC 6145 のステートレス変換の定義を満たすように，ユーザ IPv4 アドレス空間から IPv6 リンクアドレス空間の一部に対して，1 対 1 の静的な変換規則を定義した．IPv4 から IPv6 リンクローカルアドレスへの変換は，“fe80::0”の上位 12 オクテットと変換前の IPv4 アドレスの 4 オクテットを連結する．逆方向は IPv6 リンクローカルアドレスの下位 4 オクテットを IPv4 アドレスとする．一般的な NAT とは異なり，変換規則のみでアドレスを計算するため，セッション情報の保持と管理を行う必要がないステートレスなモジュールとなる．

各ユーザがセグメント内で使用している IPv4 アドレスの数やユーザ数が増加しても，メモリ量や変換テーブルの検索時間の変動がなく，スケーラビリティの低下も発生しない．

3.3. アプリケーションの拡張

L2 マルチテナント方式では，処理とデータの区分のため，ネットワークの分離に加えて，単一 VM 上のアプリケーションプロセス内でユーザの識別と区分処理を行う必要がある．重複の可能性がある IPv4 アドレスから変換された IPv6 リンクローカルアドレスに加え，データグラムの入出力インターフェースの Scope ID を認識してクライアントを識別する．

具体的には，パケットの送信元ソケットアドレスを RFC 3493 に準拠した socket API である getnameinfo() システムコールによって，Scope ID を含んだ IPv6 リンクローカルアドレスに変換する．得られたアドレスから Scope ID を切り出すことによって，どのインターフェースから到達したパケットかを判定することができる．アプリケーションはクライアントの情報として IPv6 リンクローカルアドレスと Scope ID の組を用いる．

これによって，アプリケーションプロセスでユーザを区分してサービスを提供できるように，アプリケーションロジックを拡張することが可能となる．

4. 実装

本節では実装方式について述べる．今回は v4v6 変換モジュールとサンプルアプリケーションの実装を行った．v4v6 変換モジュールは map646[6][7]をベースとしたユーザランド実装と，Open vSwitch[8]をベースとしたカーネル実装を行った．本方式でマルチテナントが実施できる事を確認するために，Apache httpd をサンプルアプリケーションとした拡張を行った．

4.1. ユーザランド実装

map646 はユーザランドプロセスとして実装された IPv4/IPv6 変換ソフトウェアで，IPv4 アドレスと IPv6 グローバルアドレスを静的なアドレスマッピング規則を用いて NAT64 を行う．map646 は tun 仮想ネットワークデバイスを用いて， IPv6 サーバに対応付けされた IPv4 パケットを読み込んで，IPv6 パケットに変換する．変換された IPv6 パケットは再び tun デバイスに出力され，IPv6 サーバに到達する．逆に IPv6 サーバからのパケットも tun デバイスが読み込んで IPv4 パケットに変換し，再び tun デバイスに出力される．アドレス変換はあらかじめ IPv4 と IPv6 の対応表を静的に与えておくことで決定される．

L2 マルチテナント方式では，IPv4/IPv6 アドレスを変換するものの，L2 ネットワークの延長として利用できることを目指している．そこで map646 に対して以下の拡張を行った．

- 仮想スイッチとしての動作
  IPv4 側と IPv6 側にそれぞれ tap デバイスを作成することで，ユーザ LAN と VM の間でパケットをブリッジできるようにした
- Ethernet フレーム単位の処理
  変換処理の単位を IP データグラムではなく，Ethernet フレームとした．これによって Ethernet ヘッダ内の情報を含む変換処理の実施を可能とした
- 静的なアドレス変換
  IPv4 アドレスにリンクローカルプレフィックス“fe80::”を付加した形式の IPv6 リンクローカルアドレスへ変換する処理を追加した
- MAC アドレス解決プロトコルの変換
  ARP(Address Resolution Protocol)と NDP(Neighbor Discovery Protocol)を相互変換することで，動的に MAC アドレス解決ができるようにした

図 3 にユーザランド実装による構成図を示す．図中，map646 は今回拡張した map646 プロセスであり，IPv4 側と IPv6 側に接続される tap デバイスとして tap4 と tap6 を生成する．gre0 は CJB との間に設定されている GRE トンネルの端点である．gre0 はブリッジデバイス br4 で tap4 と vnet1 にブリッジ接続される．tap6 は br6 によって vnet0 とブリッジ接続される．vnet0 および vnet1 は VM への仮想ネットワークインターフェースであり，VM 内ではそれぞれ eth0, eth1 に接続される．

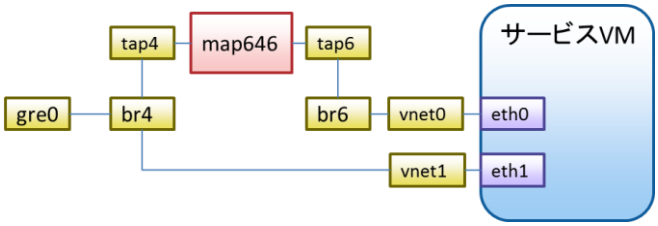

図 3 ユーザランド実装による構成図

ここで br4 では gre0 から入ってきたパケットを IPv4 と IPv6 に分離する必要がある．そこで“ebtables”によってフィルタリングを行った．具体的にはブリッジの出力ポートが tap4 の場合には IPv6 パケットを DROP し，出力ポートが vnet1 の場合には IPv4 パケットを DROP するように設定した． ebtables のデフォルト設定は全てのパケットをフォワードする．この設定から DROP を加えるだけで意図する動作を行うことができる．

4.2. カーネル実装

Open vSwitch はオープンソースの仮想ソフトウェアスイッチで，ユーザランドで動作するプログラム群と Linux カーネルモジュールで構成される．

Open vSwitch のカーネルモジュールでは GRE トンネリングプロトコルをサポートしている．さらにユーザランドのプログラムからフローエントリを指定することによってパケットごとの制御を行うことができる．

今回筆者らは Open vSwitch に対しても v4v6 変換モジュールの実装を行った．具体的には OpenFlow を拡張する形で新たなアクションとして “nat64” を追加した．nat64 アクションには以下の 3 種類のオプションを指定することができる．

- 4toll6
  IPv4 パケットを IPv6 リンクローカルアドレスパケットに変換する
- ll6to4
  IPv6 リンクローカルアドレスパケットを IPv4 パケットに変換する
- normal
  パケットを変換しない

これらをフローエントリ管理ツール“ovs-ofctl”から設定できるようにするための修正を行った．

さらに map646 への拡張と同様に MAC アドレス解決プロトコルの変換も実装した．

カーネル実装による構成を図 4 に示す．図中の OVS は今回実装した Open vSwitch の仮想スイッチデバイスである．OVS のポートとして gre0，vnet0，vnet1 が接続されている．gre0 はクライアント側の CJB との間で設定されている GRE トンネルの端点である．また，図 3 と同様に vnet0 および vnet1 は VM 内でそれぞれ eth0，eth1 に接続される．

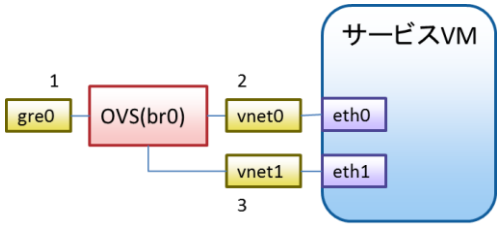

図 4 カーネル実装による構成図

Open vSwitch の具体的な設定例を図 5 に示す．仮想スイッチの設定で仮想スイッチ “br0” を作成し(①)，ポート “gre0”，“vnet0”，“vnet1” を接続している（②）．それぞれのポートは図 4 の各ポートに対応している．“gre0” ポートは CJB との間の GRE トンネルの設定を行っている（③）．

```
#!/bin/sh

# 仮想スイッチの設定
ovs-vsctl add-br br0 ・・・・・・・・・・・・・・・・・・・・・・①
ovs-vsctl add-port br0 gre0 ・・・・・・・・・・・・・・・・・・②
ovs-vsctl set interface gre0 type=gre \
   options:remote_ip=10.0.0.12 options:key=10 ・・・・・・・・③
ovs-vsctl add-port br0 vnet0 ・・・・・・・・・・・・・・・・・・②
ovs-vsctl add-port br0 vnet1 ・・・・・・・・・・・・・・・・・・②

# フローエントリの定義
ovs-ofctl del-flows br0 ・・・・・・・・・・・・・・・・・・・・④
ovs-ofctl add-flow br0 \
   priority=10,in_port=1,ip,actions=nat64:4toll6,output:2・・⑤
ovs-ofctl add-flow br0 \
   priority=10,in_port=1,arp,actions=nat64:4toll6,output:2 ・⑤
ovs-ofctl add-flow br0 \
   priority=10,in_port=2,ipv6,actions=nat64:ll6to4,output:1・⑥
ovs-ofctl add-flow br0 \
   priority=10,in_port=1,ipv6,actions=output:3 ・・・・・・・・⑦
ovs-ofctl add-flow br0 \
   priority=10,in_port=3,ipv6,actions=output:1 ・・・・・・・・⑦
```

図 5 Open vSwitch の設定

フローエントリの定義で，すべての定義を消去（④）した後，actions に “nat64” アクションを指定したフローエントリを定義しており，それぞれ “4toll6” および “ll6to4” オプションを指定している（⑤，⑥）．これらのオプションは IPv4 から IPv6 リンクローカル，または IPv6 リンクローカルから IPv4 に変換すること指示するためのものである．これらによって “gre0” ポートから入ってきたパケットのうち IPv4 パケットは IPv6 リンクローカルに変換して “vnet0” ポートに出力し，反対に “vnet0” ポートからのパケットは IPv4 に変換して “gre0” ポート出力するような設定を行うことができる．また続くフローエントリによって “gre0” ポートから入ってくる IPv6 パケットは変換せずに “vnet1” ポートに出力するように指定している（⑦）．

このように，本実装の OVS デバイスでは v4v6 変換モジュールの設定と同時に IPv4 パケットと IPv6 パケットを分離する設定を行うことで，ユーザランド実装に比べて構成を単純化することができる．

これらの設定によって IPv4 パケットと IPv6 パケットを分離することができ，デュアルスタック環境下のユーザ LAN にも対応が可能となる．

4.3. アプリケーションの拡張の実装

今回の実装ではサンプルアプリケーションとして Apache httpd 2.4.1 を採用して拡張を行った．

Apache httpd はバーチャルホスト(vhost)機能によっ

て異なる IP アドレス宛のリクエストを区分して，各 IP アドレスに対応したコンテンツを返す事ができる．そこでサーバの IP アドレスだけではなく，IPv6 リンクローカルアドレスの Scope ID でもサーバを区分できるように拡張を行った．具体的には，クライアントから httpd に対して IPv6 リンクローカルアドレスでアクセスを行う時に，リクエストの送信元ソケットアドレスを getnameinfo()システムコールによって Scope ID を含んだ IPv6 リンクローカルアドレスに変換する．変換されたアドレスから Scope ID を切り出すことによって，どのインターフェースからのパケットかを判定するようにした．あらかじめユーザとインターフェースとを対応付けることで，インターフェース名からユーザを特定することが可能となる．ユーザの区分は Scope ID によって DocumentRoot を切り替えることで実現した．vhost 機能を設定する“httpd-vhosts.conf”ファイルの例を以下に示す．

```
<VirtualHost [fe80::c0a8:64]:80 >
  ServerName servicevm
  DocumentRoot /var/www/rewrite
  RewriteEngine On
  RewriteRule ^/(.*)$ /%{REMOTE_IF}/$1 [L]
</VirtualHost>
```

これは fe80::c0a8:64 (fe80::192.168.0.100)へのアクセスに対して RewirteRule の指定によって DocumentRoot の末尾に%{REMOTE_IF}を追加するための記述である．ここで%{REMOTE_IF}は Scope ID が格納される変数で，本拡張で実装したものである．実際の拡張規模の概要は以下の通りである．

```
% diffstat apr-1.4.6.patch
 include/apr_network_io.h      |   12 ++++++++++++
 network_io/unix/sockaddr.c    |   16 ++++++++++++++++
 2 files changed, 28 insertions(+)
% diffstat httpd-2.4.1.patch
 modules/mappers/mod_rewrite.c |    6 ++++++
 modules/ssl/ssl_engine_vars.c |    3 +++
 server/util_expr_eval.c       |    1 +
 server/util_script.c          |    2 ++
 4 files changed, 12 insertions(+)
```

今回は Apache httpd を対象として拡張を行ったが，複数ユーザを区分するための機能を持つ他のアプリケーション(例えば smbd 等)への応用も可能であると考えられる．

5. 実験

実験では本方式の有効性を確認するために，疑似 L2 マルチテナントシステムによる複数ユーザ区分の動作確認と性能測定を行った．

5.1. 実験環境

実験にはラックマウント型の PC サーバである PRIMERGY RX200 S5 上にクラウドデータセンタを模した環境を構築した．ホスト OS に Fedora16 を採用し，KVM の仮想環境によってサービス VM を構成した．サービス VM の OS は FreeBSD 9.0 を採用した．また PRIMERGY RX300 S5 にクライアント側の構成を模した環境を構築した．ホスト OS に Fedora16 を用いて KVM の仮想環境によって複数のユーザ VM を作成した．各ユーザ VM の OS に Fedora16 を採用し，それぞれを異なるネットワークセグメントとして構成した．それぞれのハードウェアスペックを表 1 にまとめる．

表 1 ハードウェアスペック

	CPU	コア	メモリ	NIC
RX200 S5	Xeon 2.12GHz	4	24GB	GbE
RX300 S5	Xeon 2.27GHz	16	72GB	GbE

図 6 に実験環境の概要を示す．クライアント側のユーザ VM はすべて同一の IPv4 アドレス“192.168.0.10/24”を静的に割り当て，アドレスの重複を再現している．クラウド側のサービス VM のネットワークインターフェースには同一の IPv6 アドレス“fe80::192.168.0.100(fe80::c0a8:64)”を静的に割り当てた．

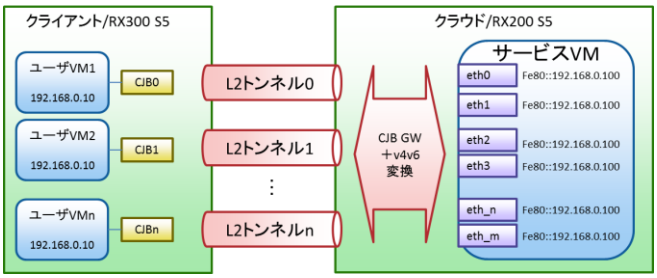

図 6 実験環境

本来は，ユーザ LAN の CJB がサービス VM に対して L2 接続した時に初めてサービス VM に新しい NIC が割り当てられ，ユーザと NIC の対応関係が更新されるべきであるが，今回の実験ではサービス VM に割り当てる NIC とユーザとの間の対応関係をあらかじめ静的に保持することで環境を構築した．

CJB では L2 over L3 トンネリング方式を限定していないため，種々のプロトコルを利用することが可能である．今回の実験では Linux カーネルモジュールや Open vSwitch で採用されている GRE トンネリングプ

ロトコルを採用した．GRE トンネルプロトコルのキー属性に各々のユーザセグメントで異なる値を指定することで，ユーザを区分するための複数の L2 トンネルを開設した．

5.2. 複数ユーザ区分の動作確認

クラウド側のサービス VM 上に拡張した Apache httpd を配置し，各ユーザ VM から HTTP クライアント“curl”によって HTTP リクエストを発行した．サービス VM 上では httpd のコンフィギュレーションによって指定された DocumentRoot の末尾に Scope ID を付加したディレクトを作成し，各クライアント向けのコンテンツとして以下に示す index.html ファイルを格納した．下線で示したテナント番号とインターフェース名は Scope ID 毎に変更している．

```
<html><body>This Tenant 1 of em0.</body></html>
```

複数ユーザ区分の動作結果を図 7 に示す．

```
Vm1% curl http://192.168.0.100:80/
<html><body>This Tenant 1 of em0.</body></html>
Vm1% curl http://fe80::192.168.0.100%eth0:80/
<html><body>This Tenant 1 of em1.</body></html>

Vm2% curl http://192.168.0.100:80/
<html><body>This Tenant 2 of em2.</body></html>
Vm2% curl http://fe80::192.168.0.100%eth0:80/
<html><body>This Tenant 2 of em3.</body></html>

Vm3% curl http://192.168.0.100:80/
<html><body>This Tenant 3 of em4.</body></html>
Vm3% curl http://fe80::192.168.0.100%eth0:80/
<html><body>This Tenant 3 of em5.</body></html>
```

図 7 複数ユーザ区分確認の結果

“Vm1%”はユーザ VM1 のコマンドプロンプトを表しており，各ユーザ VM 上で curl を実行した結果を示している．Vm1 はテナント番号 1 のユーザであり，IPv4 アドレスでのアクセスは em0 に到達し，また IPv6 リンクローカルアドレスでのアクセスは em1 に到達していることがわかる．同様に Vm2 および Vm3 はそれぞれテナント番号 2 および 3 で，em2 と em3 および em4 と em5 に到達していることがわかる．

これによって，IP アドレスの重複があっても各ユーザを区分して，あらかじめ指定されたコンテンツを返すことができていることが確認できる．

5.3. 性能測定

ネットワークスループット測定ツール“iperf”によってクライアントとクラウド間のスループットを測定した．性能測定の結果を図 8 に示す．実験はサービス VM で iperf をサーバモードで起動し，ユーザ VM から iperf をサービス VM の IPv6 リンクローカルアドレスに対応する IPv4 アドレスを指定した場合と IPv6 リンクローカルアドレスを直接指定した場合のそれぞれのスループットを測定した．

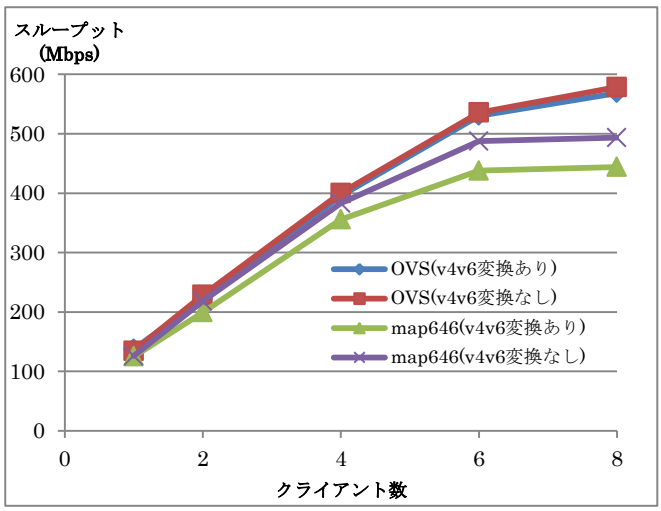

図 8 性能測定結果

横軸はユーザ VM 数で縦軸がスループット(Mbps)を表している．map646 実装と Open vSwitch(OVS)実装のそれぞれについて，IPv4 アドレスを指定して v4v6 変換モジュールによって IPv6 リンクローカルアドレスに変換した場合と，サーバの IPv6 リンクローカルアドレスを直接指定して v4v6 変換を行わない場合の結果である．グラフは 5 回測定した平均値をプロットした．

グラフから Open vSwitch 実装において，v4v6 変換モジュールを使った場合と，使わない場合との差は小さいことがわかる．これは v4v6 変換モジュールがスループットに与える影響が小さいことを意味している．一方 map646 実装において両者の差が比較的大きく，v4v6 変換モジュールはカーネル実装に比べてユーザランド実装のオーバヘッドが大きいことがわかる．

各クライアント数において Open vSwitch 実装は map646 実装よりも高い性能を示していることがわかる．このことからも，ユーザランド実装とカーネル実装の性能差が表れているものと思われる．

またクライアント数が 6 以上の場合に線形から外れ

て性能が下がっているが, これは iperf クライアント数に対応する iperf サーバのスレッド数が物理コア数を超えたためであると考えられる. しかしながら map646 実装よりも Open vSwitch 実装の性能低下は小さく，カーネル実装の方がスケーラビリティを確保するためにも有効であると言うことができる.

6. 関連研究

文献[1]は L2 接続したクラウドに対して，LAN 内の VM のライブマイグレーション手法を提案している. 本稿で提案するような IPv4 アドレスの重複に対する対応は行っていない.

CloudBridge[9] はパブリッククラウドとオンプレミスのプライベートクラウドを L2 トンネルによってシームレスに接続するための製品である. 単一のユーザをパブリッククラウドに接続するものであって，複数ユーザをクラウドで L2 収容することはできない.

文献[10]はプライベートアドレスを用いたネットワーク同士を VPN で接続する際に, アドレスの重複が起きないような制御方式を提案している. 本稿ではアドレスの重複を許したうえで，複数ユーザを区分しつつ L2 サービスを提供することができる.

文献[11]は VPN 環境においてアドレス重複を解決するために，TwiceNAT によってアドレスを変換する方式を提案している. 本稿では重複する IPv4 アドレスを IPv6 リンクローカルアドレスに変換することによってアドレスだけでなく，Scope ID によってユーザを区分する方式を提案している.

7. まとめ

本稿では重複する IPv4 アドレスを持つ, 複数のユーザ LAN をクラウドに収容し，L2 サービスを提供するための L2 マルチテナント方式を提案した. 本方式では IPv4 アドレスを IPv6 リンクローカルアドレスに変換し，ユーザ毎に異なるネットワークインターフェースとしてサービス VM に接続することで，Scope ID によって IPv4 アドレスが重複する場合にも正しくユーザを区分して L2 サービスを提供することが可能である.

本方式の v4v6 変換モジュールを map646 および Open vSwitch に実装した. さらに，IPv6 リンクローカルアドレスの Scope ID によって接続元のユーザを区分できるように，Apache httpd に対して拡張を行った.

実装した v4v6 変換モジュールと httpd を KVM の VM 環境上に構築して実験を行った. 実験の結果，本方式によって IPv4 アドレスが重複している場合にも, ユーザを区分してサービスを提供できることが確認された. さらに性能測定によって，Open vSwitch への実装がオーバヘッドも小さく，スケーラビリティの確保に適していることが分かった.

今回は L2 サービスを提供するサービス VM に接続するインターフェースとユーザの対応関係を静的に決定したが，今後は自動的に対応関係を構成し，ユーザ数の動的な増減に対応する必要があると考える.

さらに Apache httpd 以外のアプリケーションに対応するための L2 マルチテナント向けのアプリケーションフレームワークを開発し，他のアプリケーションに適用し有効性を確認する予定である.

文　献

[1] T. Wood et al. “CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines.” In Proceedings of the 7th International Conference on Virtual Execution Environments, 2011.

[2] Amazon Virtual Private Cloud, http://aws.amazon.com/vpc/

[3] 鈴木 和宏，今井 祐二，福井 恵右，“低コスト FTTH を用いたクラウドアクセス方式の提案”, 電子情報通信学会第 10 回ネットワークソフトウェア研究会, Jun. 2010.

[4] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[5] Baker, F., Li, X. and Bao, C., “IP/ICMP Translation Algorithm”, RFC 6145, April 2011.

[6] 石田渉，島慶一, “IPv6 のみで構成された IaaS システム用の IPv4-IPv6 変換ソフトウェアおよびその運用システムの実装と評価”, wide-tr-cloud-map646-eval-00.pdf, WIDE project, December, 2011

[7] Shima, K. and Ishida, W. “map646: Mapping between IPv6 and IPv4 and vice versa.” https://github.com/keiichishima/map646/.

[8] Nicira Networks, Open vSwitch, http://openvswitch.org

[9] Citrix CloudBridge. www.citrix.co.jp/products/cloudbridge/

[10] 岡田浩一，中山隆二，伊集院正, “VPN を利用するサーバホスティングにおけるアドレス重複の回避”, 電子情報通信学会技術研究報告. IN, 情報ネットワーク 98(589), 43-50, 1999-02-15

[11] 小山高明，山田英樹，岸寿春，深見公彦，北爪秀雄, “VPN 間接続環境における TwiceNAT 方式の一提案”, 電子情報通信学会技術研究報告. IN, 情報ネットワーク 111(9), 19-22, 2011-04-14