SDNを導入したホームネットワーク環境

柴田　尚紀†　山崎　憲一†

† 芝浦工業大学大学院　理工学研究科　〒135–0061 東京都江東区豊洲 3–7–5
E-mail: †{ma13044,yamaken}@shibaura-it.ac.jp

あらまし　近年登場したSDN技術は，多くのサーバ群を接続するデータセンタなどの大規模ネットワークで利用されているが、一般家庭のネットワークでも有効であると考える．これは今後のホームネットワークでは，家電やマルチメディアサーバーといった多様なデバイスが接続されるためである．しかし，これらのデバイスを安全に分離して管理するには様々な問題が想定される．本稿ではそれらの問題を解決し，同時にそれらを利用するアプリケーションと連係可能とすることが求められることを想定し，それらデバイスとサービスが正常に動作を行えるアーキテクチャを提案する．

キーワード　SDN，OpenFlow，ホームネットワーク

A Home Networking Environment with SDN technology

Naoki SHIBATA† and Kenichi YAMAZAKI†

† Faculty of Science and Engineering,Shibaura Institute of Technology　Toyosu
3–7–5,Koutouku,Tokyo,135–0061, Japan
E-mail: †{ma13044,yamaken}@shibaura-it.ac.jp

Abstract　Software-Defined Network (SDN) technologies are now being used mainly in large scale networks in data centers to connect a huge number of servers. We however think that it is quite useful in home networking which connects many types of devices such as home appliances and multimedia devices. We can separate devices into several groups using a flow control mechanism of SDN to make the home network secure. SDN also provides a mechanism to cooperate such devices with their own service servers. In this paper, we will propose an architecture to provide such functions, and describe a detail system implementation.

Key words　SDN,OpenFlow,Home Network,

1. はじめに

近年の情報社会の発達は著しく，特にデータセンタなどの大規模な企業ネットワークでは，SDN(Software-Defined Network)と呼ばれる技術が利用されるようになってきている．このような大規模ネットワークを構成する場合，従来ではL2スイッチを多数用いていたが，SDNの代表であるOpenFlowでは，複数のスイッチに対し制御の役割を持つコントローラ1つという構成を取っている．このコントローラを用いることで，複数のスイッチを一括制御可能となり，ネットワークの管理が容易となった．SDNはこのような大規模ネットワークの特殊なニーズに応じる形で利用されている．

一方で，家庭の情報デバイスは様々な製品が登場しており，それらデバイスにはネットワーク機能が付与され，新しい形のサービスを提供することで人々の暮らしが更に豊かになりつつある．その例として，マルチメディアサーバ，ネットワーク機能付きのエアコン等の家電，ネットワーク電球がある．例えばマルチメディアサーバは，映像のような大量のトラフィックを発生させる．家電に対しては，ユーザが出先から操作するために，ゲートウェイの外側から家電に対して通信を行うことがありうる．また，ユーザの客が家に訪問したとき，客は一時的に無線LANを利用したいかもしれない．以上のように我々はホームネットワークの本質はその多目的性，多用途性にあると考える．

そのための様々なデバイスやサービスを適切に管理することは，知識が十分にあれば可能だが，家という特殊性からできるだけ簡単かつ安全に運用できることが求められる．このためには既存のホームネットワークでは解決できない問題が2つ考えられる．1つは多種多様なデバイス群のパケットを管理するのは現在のルータにとって困難である点であり，もう1つは各種デバイスに対応するサービスを提供する基盤が必要であるという点である．このようなホームネットワークにおいては，SDNの提供するネットワーク仮想化の機能が有用であると我々は考

える．

以降本稿では，関連研究について第二章で述べる．将来予想されるホームネットワークの利用シナリオを提示し，アーキテクチャに対する要求条件の定義を第三章で行う．第四章でその要求条件を元にアーキテクチャの提案を行う．第五章では，実装の詳細を述べ，六章でまとめる．

2. 関連研究

Yiakoumis らはホームネットワーク上に仮想ネットワークを配置することを提案している [1]．この研究の目的はサービス毎に帯域を制限することと，ネットワークに接続できるデバイスを制限し，セキュリティを向上させることである．この研究ではビデオストリーミング機器などのリッチな機器がホームネットワークに接続されることを考慮しているが，我々は，もっと様々なデバイスが同じネットワークに接続される事を考える．また，この研究ではビデオストリーミングの帯域制御といった点に注目してアーキテクチャの提案を行っていたが，我々は対象とするサービスとデバイスのネットワークの経路の制御に注目してアーキテクチャを提案する．

FlowVisor は，1 つの OpenFlow vSwitch を複数の OpenFlow Controller で利用可能とするためのミドルウェアである [2]．FlowVisor は対象としているネットワークレイヤが異なる．本研究は FlowVisor の上位層に位置するものである．

OpenFlow の機能を他のアプリケーションに提供するためのコントローラがヒューレット・パッカード社から発表されている [3]．このコントローラは，アプリケーションにコントローラの機能を提供するインタフェース (API) を有している．しかし，この API は，ネットワーク管理者のみが利用する前提で設計されており，複数のデバイスアプリケーションのような，製作者が異なるアプリケーションを複数運用することは考えられていない．

我々が提案する Nvisor は，OpenFlow コントローラのアプリケーションに相当する．これを記述するためのフレームワークの 1 つに trema がある [4]．trema は，OpenFlow コントローラアプリケーションをスクリプト言語である ruby で記述でき，他のフレームワークと比べて少ない行数で記述することができる．この利点から，我々は提案アーキテクチャの開発に trema を利用することにした．

3. システムの要求条件

本章では，将来予想されるホームネットワークの利用シナリオを提案し，アーキテクチャに対する要求条件の定義を行う．

3.1 利用シナリオ

ここではネットワーク機能のついた電球（例えばフィリップス社の hue) を家庭で利用するシナリオを考えることで要求条件を検討する．

まず，ユーザは，店頭で電球を購入したネットワーク電球を家庭のソケットに設置し，ネットワーク接続をする．すると，家庭の PC の画面に接続された電球の製造番号が表示される．買った電球のパッケージに記載されている番号と一致していれば，OK ボタンを押す．これでネットワーク電球はユーザが設定したアプリケーションによって制御される．

また，上のアプリケーションから電球を操作することも可能である．例えば，電球の発光する色を変更できる．更に，別のセンサと連携させて入室・退室に合わせて明滅させるなども可能である．

ホームネットワークに接続されるデバイスには様々な種類があり，中には悪意を持って設計されたデバイスも存在すると考えられる．そのようなデバイスが家庭ネットワークに接続されれば，悪意のあるデバイスによって，別のデバイスの情報が盗聴される危険性がある．家庭のネットワークを安全に利用するためには，このような悪意のあるデバイスをネットワークから遮断できる必要がある．

3.2 要求条件

以上をまとめると，将来のホームネットワークには次のようなことが求められる．

- ある群に属しているデバイスは互いに通信可能とし，別の群に属している他のデバイスとは通信不可にできること．
- ネットワーク上のスマートフォン等のアプリケーションに対してデバイスを用いた機能を提供するために，特定のデバイス群と連携した動作を行え，そのような複数のサービスが混在できること．

本稿では，家庭ネットワークとして，現時点では有線 LAN のみを利用した環境を想定する．無線 LAN は [1] と同様にマルチ SSID を用いることで対応できると考えられるが，現時点では考慮していない．

4. システムのアーキテクチャ

まず，1 つ目の要求条件に対して，ある群に属しているデバイス同士のみを通信可能とし，異なる群のデバイス同士の通信を完全に遮断する必要がある．一般的な家庭用ルータの機能では完全にデバイス同士の通信を遮断することは困難であり，ネットワークのフローを分けることが求められる．2 つ目の要求条件については，サービスと特定のデバイス群を連携させるため，サービスが利用するデバイス群を判別する機能と，判別したデバイス群を対応づけ，さらにデバイス群とアプリが連携するための API を提供する必要がある．

これらの必要性から，我々が提案するアーキテクチャでは OpenFlow を利用する．OpenFlow はパケットの送信元，送信先の MAC アドレス，IP アドレス，ポート番号をキーとしてネットワークの経路を制御することができる．たとえデバイスがあるポートに接続されている物理ポートから抜線され，別の物理ポートに接続しなおしたとしても，以前の状態と変わらずに通信を行うことができる．

提案アーキテクチャを図 1 に示す．各デバイスは OpenFlow vSwitch に物理的に接続される．DHCP サーバはそれらのデバイスとは隔離したスライスに配置する．OpenFlow vSwitch の上に OpenFlow Controller，さらにその上に我々が提案する NVisor ミドルウェアを配置する．

OpenFlow vSwitch は，制御部である OpenFlow コントロー

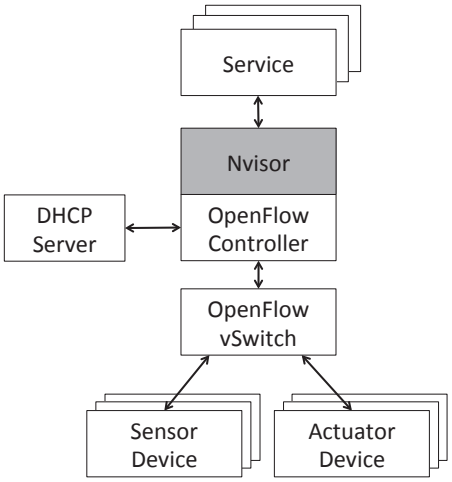

図 1　提案アーキテクチャ

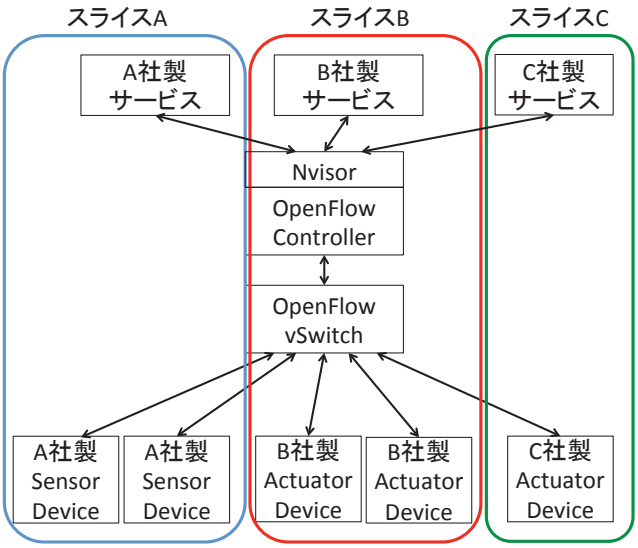

図 2　スライス構成

ラからの命令でパケットの経路を管理する．また，OpenFlow コントローラは，フローテーブルと呼ばれる経路のマップを持っており，特定の送信先，送信元の IP アドレス，MAC アドレス，ポート番号が記載されている．このフローテーブルを用いることで，スライスを作成する．例えば，デバイス A とデバイス B で 1 つのスライスを作るとすると，双方のデバイスを送信元，送信先として，送信元の IP アドレス，MAC アドレスを指定したフローテーブルを作成する．

フローテーブルにないパケットは OpenFlow Controller へと通知され，OpenFlow Controller アプリケーションである Nvisor によって処理される．

Nvisor は，スライスを管理することでサービスとデバイスへ機能を提供する．複数のサービスへそれぞれに対応するデバイス群からのパケットを転送する．サービスはデバイスの種類ごとに存在し，デバイス群を制御することでユーザに利便性を提供する．

各デバイスが各サービスに結び付けられ，デバイス群としてスライスを構成された状態を図 2 に示す．この状態においては，デバイス群同士の直接的な通信は全て OpenFlow vSwitch によって破棄され，互いに影響を与えることはない．

以降は，本アーキテクチャの詳細な動作を順に説明する．

4.1　新規サービス追加時の動作

新規サービス追加時の動作を図 3 に示す．サービスの追加は家庭のネットワーク管理者のコマンドによってサービスから Nvisor へ通知される．通知の内容は，サービスの IP，ポート番号，スライス名，当該サービスが管理対象とするデバイスの MAC アドレスのベンダー番号 (OUI) である．Nvisor は更にサービスへレスポンスを返し，デバイスからの通信用のコネクションを確立する．デバイスからサービスへ通信があった場合，Nvisor はこのコネクションを通して，サービスに転送する．

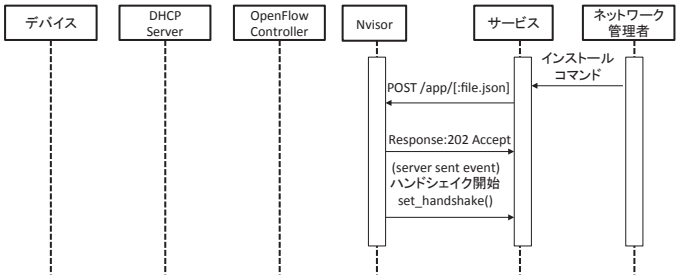

図 3　新規サービス追加時のシーケンス図

4.2　デバイス接続時の動作

本アーキテクチャにおいては，DHCP サーバはデバイスから DHCP リクエストを直接は受け取らない．IP アドレス割り当てはデバイス群管理において重要であり，Nvisor と連携する必要があるからである．デバイス接続時は，図 4 に示す順序で処理する．まず，接続されたデバイスは DHCP リクエストを発信する．それを検出した OpenFlow Controller は，DHCP サーバにそのリクエストをそのままリレーする．リレーされた DHCP サーバは，デバイスに対して IP アドレスを発行し，OpenFlow Controller へ返信する．返信を受けた OpenFlow Controller

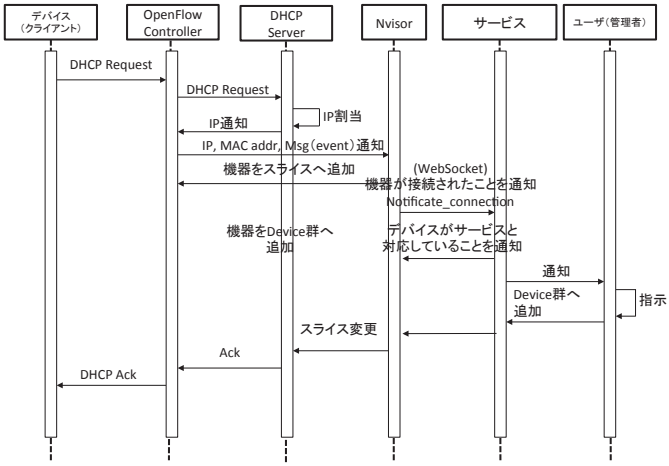

図 4　デバイス接続時のシーケンス図

は，NVisor に対して接続されたデバイスの IP，MAC アドレスを通知する．OpenFlow Controller は，割り当てられた IP

をただちにはデバイスに返信しない．もし，そのデバイスが悪意のあるものであれば，ネットワークに対する攻撃や盗聴を行うことが可能となるためである．Nvisor は，通知された IP，Mac アドレスから，接続されたデバイス 1 台のみのスライスを作成する．これは，この時点ではデバイスはサービスとの関連付けがされていないことから，信用できないためである．その後 Nvisor は，新規サービス追加時に登録されたサービスと MAC アドレスの OUI とパケット発信元の MAC を照合し，対応するサービスへ確認の通信を行う．サービスからの確認がとれた後，Nvisor は，OpenFlow Controller のコマンドを利用してデバイス群専用のスライスへ追加する．このスライスへの追加は，OpenFlow vSwitch 上のフローテーブルを変更することで行われる．

デバイスに割り当てられる IP は DHCP によって管理される．ただし，スライスと IP アドレスの体系は関連していない．OpenFlow vSwitch 内では，スライスだけでフローが管理され，IP アドレスの値とは関係がない．様々なセグメントの IP がデバイスに割り当てられるが，フローさえ保たれていれば通信を行うことができるため，問題はない．

この操作の結果，接続されたデバイスは，サービスとの通信が可能となる．各デバイスはアプリケーションと通信を行うが，その通信はスライスによって制御される．

4.3 デバイスのパケット発信時の動作

デバイスがパケットを発信したときの動作を図 5 に示す．同じデバイス群のデバイス A からデバイス B へパケットが送信されると，OpenFlow vSwitch はデバイス接続時にデバイスが追加されたフローテーブルに従って送信する．もしデバイス B が異なるデバイス群に属しているならば，OpenFlow vSwitch はパケットを破棄する．また，パケット A の宛先 IP がサービスであった場合，OpenFlow vSwitch のフローテーブル上には記述されていないため，OpenFlow Controller へ通知される．OpenFlow Controller は Nvisor へそのパケットを通知し，Nvisor は，TCP を終端する．そして，アプリケーションプロキシとしてデータをサービスへ渡す．データを受けとったサービスは，サービス依存の動作を行う．スマートフォン等のアプリケーションは，利用するサービスと対応しているデバイス群とは異なるスライスに存在する．アプリケーションは，サービスに対して通信を行い，サービスは要求に応じてデバイスと通信を行う．

4.4 サービスからデバイスに対する通信

サービスは，デバイスに対して通信を行いたい場合，送りたいメッセージと宛先 MAC アドレスを Nvisor へ通知する．Nvisor は，先ほどとは逆方向のアプリケーションプロキシとして動作する．すなわち，メッセージからパケットを生成し，それを OpenFlow vSwitch からデバイスへ送信する．

4.5 デバイス切断時の動作

デバイスが家庭のネットーワークから切り離された時，デバイスは何もパケットを発さないかもしれない．そこで Nvisor では，サービスから問い合わせがあった時のみ，ネットワーク上のデバイスの有無を確認している．接続確認には現在は ping を用いている．切断されたデバイスがあった場合，Nvisor 上のデバイス登録を削除するかどうかはサービスが決定する．

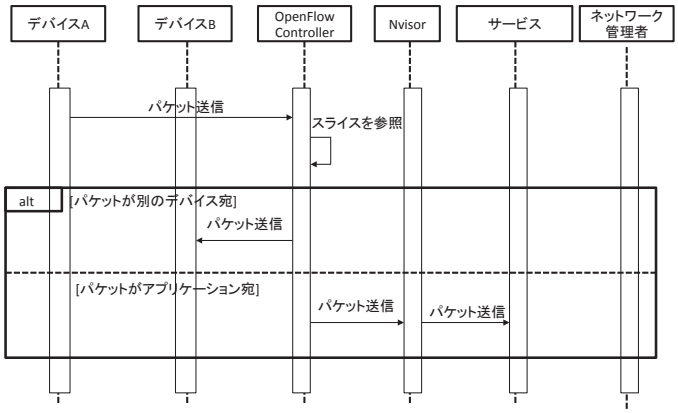

図 5　デバイスのパケット発信時のシーケンス図

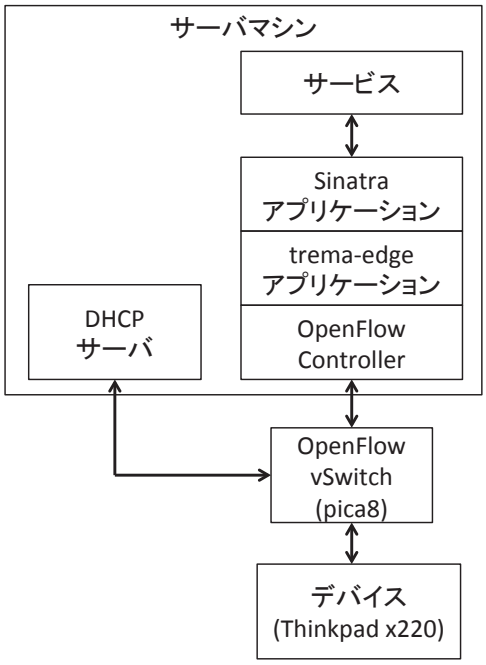

図 6　実験環境での物理構成

4.6 スライスのライフサイクル

以上のシーケンスをスライスの観点から状態遷移をまとめると，次のようになる．まず，サービスが Nvisor に登録された時に，デバイス群専用のスライスが作成される．その群の種類のデバイスが新しく接続された時，スライスにデバイスが追加され，サービスとの通信を行うことが可能となる．デバイス切断時，Nvisor はその切断を検出しない．デバイスの不在が確認された場合のみ解除される．また，スライス自体が削除されるタイミングは，サービスが Nvisor との連携を解除した時である．

5. システムの実装

本提案での現在の物理構成を含むシステム構成を図 6 に示す．Nvisor は，サービスとの連携を行うインタフェースを提供する Sinatra アプリケーションと，OpenFlow Controller の制御を行う trema アプリケーションで構成されている．trema

は openflow バージョン 1.3 である trema-edge を用いている．また，Sinatra は，バージョン 1.4 である．Sinatra アプリケーションは Web ページを扱うこともできるが，本アーキテクチャでは RestAPI サーバとして利用している．

各インタフェースの実装であるが，サービスは RestAPI と WebSocket を通して Nvisor とやりとりを行う．RestAPI は，アプリケーションのインストール，削除，デバイスのスライスへの追加，削除において利用している．WebSocket 通信は，デバイスからサービス宛に送られたパケットを通知する際に利用される．

Sinatra アプリケーションと trema アプリケーションとの通信には，RestAPI と WebSocket を用いる．OpenFlow vSwitch と OpenFlow Controller は，OpenFlow バージョン 1.3 標準の通信を用いる．OpenFlow vSwitch は Pica8 社の P-3290 である．

5.1 サービス

サービスは，RestAPI と WebSocket を用いて Nvisor の sinatra アプリケーションと通信する．本研究の想定では，サービスは第三者が記述し，ホームネットワーク利用者が自らの意思で選択し，ダウンロードして，提案システムにインストールする．多くの場合，サービスはスマートフォンやタブレット上のアプリケーションと連携動作する．これについては，本稿では触れない．また，サービスを置くサーバーは，Nvisor が動作するサーバーと同じである必要はないが，常時動作していなければならない．

5.2 sinatra アプリケーション

sinatra アプリケーションには，複数のアプリケーションを管理するために MySQL データベース上にアプリケーションの情報を持っている．そのアプリケーションの情報は，次のようなテーブルで定義している．また，sinatra アプリケーションでは，OpenFlow コントローラとは別に，スライスの情報を MySQL データベース上で扱う．テーブルの定義は次のとおりである．

表 1　アプリケーションのテーブル

カラム	意味
id	アプリケーションの id．重複なし
app-name	アプリケーション名
ip	アプリケーションの所在 IP
port	アプリケーションの所在ポート

Sinatra アプリケーションには，サービスに機能を提供するために RestAPI を実装している．ここでは一部の RestAPI を説明する．

サービスの追加を行う API は次のとおりである．

```
POST /app
```

これはサービス内容を記述した JSON を POST する．POST された JSON ファイルは，Sinatra アプリケーション上で解析され，MySQL 上に保存されることでサービス登録が完了する．https で通信することを考慮し，JSON 形式の POST とした．

表 2　スライスのテーブル

カラム	意味
id	アプリケーション id. アプリケーションテーブルとの連結用
event_name	スライス名
event_src_ip	デバイスの通信元の ip
event_src_port	デバイスの通信元のポート
event_src_mac	デバイスの通信元の MAC アドレス
event_dst_ip	デバイスの通信先の ip
event_dst_port	デバイスの通信先のポート
event_dst_mac	デバイスの通信先の MAC アドレス

レスポンスとしてアプリケーション ID と認証コードを含む文字列を返す．この操作は，Nvisor にサービスが登録されるだけで，すぐに利用可能とはならない．POST 内容とレスポンス内容の例を図 7 に示す．

```
POST 内容
    {
     "ip":"172.28.32.11",
     "slice-name":"hue-bulb",
     "port":"3000"
    }
レスポンス内容
    {
     "app_id":"123456",
     "auth_key":"keykeykey"
    }
```

図 7　POST /app とそのレスポンス

サービスを開始する API は次の通りである．

```
POST /app/start
```

上記の操作が完了すると，サービスが利用可能となる．この時，認証コードを POST する．アプリケーション ID を元に利用されているサービスを特定する．認証後，サービスは WebSocket を利用した通信が可能となる．

サービスから Nvisor へデバイスを追加する API は次の通りである．

```
POST /app/device
```

これにより新しいデバイスが追加される．JSON 形式の文字列を POST することで，複数のデバイスを一度に登録することができる．この後，登録したデバイスからサービスへ向けた通信を行うことが可能となる．

5.3 trema アプリケーション

trema アプリケーションでは，trema/pio を用いて DHCP パケットの解析を行っている．全てのパケットを PacketIn し，DHCP パケットである場合は Sinatra アプリケーションへ通知する．パケットの解析方法は次のようなプログラムを書くこと

で実現できる．

```
Pio::Dhcp.read(message.data)
```

サービスからデバイスへ通信を行いたい場合，WebSocket を通じて Sinatra アプリケーションへ送信するメッセージの内容と送信先 MAC アドレスを通知する．その後，Sinatra アプリケーションは，パケット生成のためのパッケージである BinData を用いて受信内容からパケットを生成し，該当デバイスへ Pakcet out を行う．この時，送信元 IP はサービス登録時に Nvisor で記録した IP を用いる．

6. 実験と結果

現在の実装において，次の項目の実行速度を測定した．表 3 にその結果を示す．

表 3 実験項目の動作とその結果

動作名	実行時間 (ミリ秒)
IP 割り当て	10088
機器を単独のスライスへ追加	5.3018

7. 今後の課題

本章では，現アーキテクチャの課題，ならびに今後の研究方向について述べる．

7.1 議　　論

現在のシステムのアーキテクチャでは，サービスは Nvisor, OpenFlow Controller を経由してデバイスと通信するが，この方法ではデバイスからの通信とサービスからの通信で非対称となってしまう．通信を対称にするために，別のシステムのアーキテクチャとしてサービスと OpenFlow vSwitch を別の NIC を用いて接続する案が考えられる．しかし，欠点としてアーキテクチャとして複雑になる問題があるため，さらなる議論を要する．

また，Nvisor ではサービスが競合した場合の処理を考えていない．例えば，Nvisor がサービスに対してデバイス群との通信を提供しているが，複数のサービスが同じ OUI を利用することを Nvisor へ通知した場合，どちらにアクセスを許可するか決めていない．この場合どちらのサービスを優先するか，議論を要する．

7.2 その他のシナリオ

本稿では，電球などのデバイスが，外部のアプリケーションと連携するシナリオについて検討した．我々は，家庭での将来的なネットワークの利用について，幾つかのシナリオを検討中である．今後は，これらについても検討を進める．

- ゲストへの接続提供: 家庭への来客に一時的に外部接続を許可し，同時に家庭内のリソースの一部 (例えばプリンター) だけを利用可能にする．
- スマートメーター: 家庭内のエアコン，冷蔵庫などを外部事業者から制御することがありうるが，当然，家人も利用できなければならない．このためにはホームゲートウェイを置くという手段も考えられるが，それらのプロトコルやアプリケーションの標準化などを考えると，ネットワーク側で安全性を担保すべきと考えられる．
- 特定サービス専用ネットワーク: オンラインゲームなどでは，高いリアルタイム性と，外部サーバへの容易な接続性が求められる．一方で，近年の家庭用ゲーム機はメディアサーバーなどとしても利用されるため，それらが混在した形でも安全な運用ができることが求められる．

8. おわりに

関連研究について述べ，将来予想されるホームネットワークの利用シナリオを提示し，アーキテクチャに対する要求条件の定義を行った．その要求条件を元に trema フレームワークを用いた Nvisor のアーキテクチャの提案を行い，実装の詳細を述べた．

文　　献

[1] Yiakoumis, Y., Yap, K. K., Katti, S., Parulkar, G.,& McKeown, N. (2011, August). Slicing home networks. In Proceedings of the 2nd ACM SIGCOMM workshop on Home networks (pp. 1-6). ACM.

[2] Sherwood, R., Gibb, G., Yap, K. K., Appenzeller, G., Casado, M., McKeown, N., & Parulkar, G. (2009). Flowvisor: A network virtualization layer. OpenFlow Switch Consortium, Tech. Rep.

[3] SDN Dev Center — HP®, http://h17007.www1.hp.com/us/en/networking/solutions/technology/sdn/devcenter/index.aspx

[4] trema/apps, https://github.com/trema/apps/wiki

[5] Meethue.com, http://meethue.com/