Vol.2014-IOT-24 No.24
2014/2/28

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

橋本 直樹[1]　園生 遥[2]　牛込 翔平[2]　菊田 宏[3]　永園 弘[3]　廣津 登志夫[4]　新村 正明[2]

概要：Software Defined Networking (SDN) はプログラムによって動的に，かつ抽象化されたネットワーク制御を可能とする技術である．制御を転送機能から分離することでアプリケーションとの融和を可能にするアーキテクチャを持つ．本稿ではアプリケーションと連携したネットワーク制御の一つとして，認証基盤と連携したネットワークレベルでのアクセス制御の手法の実現について述べる．ここではキャンパスネットワークを対象とし，大規模ネットワークで一般的なディレクトリサービスである LDAP を用いてネットワーク自体へのアクセス権限を管理する．アクセス権限に付随する属性情報によって内部ネットワークで提供される Web コンテンツへのアクセスを制限することで，より柔軟かつ簡便な制御を行う仕組みを提供する．

キーワード：OpenFlow，認証基盤，SDN

Implementation of network access control in conjunction with authentication infrastructure by OpenFlow

Hashimoto Naoki[1]　Sono Haru[2]　Ushigome Shohei[2]　Kikuta Kou[3]　Nagasono Hiroshi[3]
Hirotsu Toshio[4]　Niimura Masaaki[2]

1. はじめに

近年，通信端末選択やネットワーク接続形態の多様化によりインターネットへの接続性の管理における管理者の負担は増大している．その一方でユーザには通信に使用する端末やネットワークの管理形態に依らず簡単かつ安全に通信が提供されることが望ましい．昨今，ネットワークを介したコンピュータウィルスの蔓延や DoS 攻撃，不正アクセスなど，ネットワーク利用における課題は数多く存在する．ネットワークの管理者はこれらから資源を保護すると同時に外部ネットワークに対する加害者となることがないように，正当なユーザのトラフィックだけを通過させるなどユーザの通信資源利用を適切に管理しなければならない．例えばネットワークの主要なセキュリティであるファイアウォールでは固定的なルールに基づくフィルタリングが行われているため，特定のポートに対する DoS 攻撃を防ぐことが不可能である．逆にセキュリティを重視したフィルタリングルールではユーザが利用するアプリケーションの挙動に影響を及ぼす．一般に，ネットワークの管理ポリシーはネットワーク毎に異なるため，多様な管理ポリシーに柔軟に対応し，管理対象を自在に管理するアーキテクチャが求められる．

その要求を実現する手法として 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を設計・実装した．OpenFlow[1] では，ネットワークを流れる全てのフローについて OSI 参照モデルの第 1 層から第 4 層相当，つまりアプリケーションに依存しない通信識別情報を取得することが可能である．これによって管理者は全通信のフロー情報

1 Hosei University Graduate school of Computer and Information Sciences
2 Shinshu University Graduate School of Science and Technology
3 NTT DATA
4 Hosei University Faculty of Computer and Information Sciences

を把握し，管理情報に合致しないフローを不正なものとして排除することもできる．また合致するフローについても特定通信を遮断することによってアクセス可能な通信資源を限定することが可能である．言い換えればフロー毎のミニネットワークを作成し，管理されたネットワークの上で様々なネットワーク形態の共存を行うことを可能とするということである．本稿では，認証基盤に登録されたユーザ情報によってフローの意味付けを行い，それによってネットワークのアクセス制御を行う．

2. OpenFlow

Software-Defined Networking (SDN) はネットワーク制御機能とデータ転送機能が分離し，プログラムによりネットワークの制御が実現できる新しいアプローチを持つネットワークである．プログラムによって動的に，かつ抽象化されたネットワーク制御が可能となった．また長くベンダー依存であった制御機構が標準化されたことで，ネットワーク管理者自身が個々の環境に適したコンフィグレーションを行えるので柔軟性や管理における利便性が高まった．

OpenFlow は SDN ソリューションを構築するための基本要素の一つである．従来のネットワーク機器におけるパケット転送処理を行うスイッチ，アドレス学習やルーティングなど経路制御機能を担うコントローラの 2 種の装置によって構成され，中央集権型のアーキテクチャを実現可能である．

2.1 フローエントリー

MAC アドレスや IP アドレス，ポート番号などのフィールド情報によって決定される一連の通信単位であるフローに対してアクションを指定し経路制御を行う．これをフローエントリーと呼び，OpenFlow コントローラはフローエントリーを纏めたフローテーブルを各スイッチに設定することで経路情報を制御する．フローエントリーはマッチフィールド，インストラクション，カウンタ，優先度，タイムアウト，クッキーの 6 つの要素で構成される．

マッチフィールドはスイッチがフローを受信した際にトラフィックを識別し特定するためのルールで，OpenFlow 1.3[2] では 40 種，レイヤー 1-4 層相当のフィールドとその依存関係が定義されている．さらに書き込むテーブルも複数存在しており，テーブルと優先度を合わせた組み合わせによって今までの枠組みに縛られずに通信を制御することを可能とする．マッチングできなかったフローについては破棄，コントローラへの通知どちらかを予め選択しておくことが可能である．本論ではこのコントローラへの通知である Packet-In メッセージを，未ログインのユーザに対する検出手段として用いる．Packet-In メッセージはフローエントリーにマッチングしないすべてのフローだけでなく，明示的に出力先を CONTROLLER としたフローについて，スイッチが受信した情報をコントローラへ通知するために送信される．Packet-In の際，スイッチに十分な記憶領域があるときは対象フローをバッファリングし，ヘッダ情報の一部を転送する．

インストラクションは，フローに対する処理定義であるアクションの集合である．マッチフィールドに一致した通信を受け取った OpenFlow スイッチは，アクションの記述に従って通信を制御するが，テーブルを跨ぐような処理はアクションセットとしてプールされ，行われるべき全アクションの追加が終わった時点で実行される．主なアクションには出力，フィールドの書き換え，キュー，ドロップがある．

カウンタはテーブル，フロー，ポート毎に管理され，パケット数やバイト数，当該フローが OpenFlow スイッチ上に生成されてからの時間などの情報を記憶する．カウンタの値によって

- ネットワーク情報の可視化
- トラフィックの流量に応じたフローごとの帯域制御
- トラフィックの傾向に応じた転送パスの切り替え

などを行うことができる．これらの情報はコントローラがスイッチに要求することで出力される．

2.2 トポロジ検出

OpenFlow にはスイッチ間接続構成を検出するための共通的な枠組みは存在しない．そこで本稿では，OpenFlow コントローラに接続済みのスイッチ同士によって構成されるネットワークトポロジの検出手段として Link Layer Discovery Protocol (LLDP) を用いた．この検出方法は OpenFlow 仕様の範囲内で動作するため，多くの OpenFlow コントローラで採用されている．

LLDP は近隣ノードを検知するため，ネットワーク機器や端末の種類，設定情報などを近隣のノードに通知する，レイヤ 2 レベルのプロトコルである．手順としてはコントローラの管理下にある全スイッチのすべてのポートから，送信側のスイッチ ID，物理ポート番号を保持した LLDP パケットを出力する．同一コントローラによる LLDP を別のポートで受信したとき，コントローラは受信ポートと LLDP パケットに記録された送信元ポートが接続されていることを把握できる．

3. 通信資源の管理における問題

ここで通信資源とは，通信を行う際にユーザが直接的ないし間接的に使用する物理的なデバイスとソフトウェア資源，および各リンクが占有する帯域を意味する．また，アプリケーションサービスに必要なデータはすべて実通信端末間でエンドツーエンドで送受信され，通信制御に関するメッセージは制御専用回線を利用するためユーザが使用する端末へは公開されない．管理者はこの前提の元，ユー

ザの快適で安全な通信サービス利用環境を維持するため，ネットワークを管理，保護する．ここでは通信資源の管理における 3 つの問題を提示する．

3.1 通信可能範囲の制限

特定ユーザによる通信資源の占有を防ぐために通信可能範囲の制限は必須である．この制限はポート番号のようなフローの情報を用いた論理的な制限と通信可能なハードウェアを識別し管理する物理的な制限に二分される．まずフローの制限では通常，固定的なフィルタリングによって TCP/IP パケットに含まれる情報を用いた取捨選択が行われる．予め把握しいる端末やポート情報以外への対応が難しく，制御対象のステータスが動的に変化する場合に適していない．またハードウェアの制限では，研究室や講義室に設置された機器のような空間的特性を持つものや，制御専用端末や共有端末のように機器自体に性質を持つものについて特性毎に共通な制御を必要とするものに対して細やかな設定がし難い．

3.2 管理内ネットワークの不正

管理者は自らが管理するネットワークにおけるウィルスの蔓延や外部ホストに対する攻撃など，他のネットワークへの加害行為を検知した場合は速やかに排除する必要があるが，従来のフィルタリングでは問題のあるパケットの情報から MAC アドレスを利用して特定デバイスの通信を妨げることはできても不正なユーザを即座に特定することは難しい．また，多数の端末を用いるなど，パケット単体では問題が無くとも特定のユーザが不当に多くの通信帯域を占有してしまう場合も，ネットワーク全体の利便性を低下させる．

3.3 管理外ネットワークとの通信

インターネットでは誰もが自由にパケットを送り出すことができるため，悪意ある者が特定の対象に対して DoS 攻撃や総当たり攻撃などを行うこと自体を効果的に阻止することは困難である．そのため，管理者はファイアウォールによるフィルタリングなどで不正なパケットの侵入を防ぐことになるが，先に述べたフローの制限のように IP アドレスやポート番号を用いて不特定多数のホストに開放する，あるいは使用用途の定まらないポートを閉じて利用目的に制限をかける，などの画一的なフィルタリングになってしまうことが多い．

4. 認証基盤

本稿では，OpenFlow コントローラとの連携を行うものとしてネットワーク上の認証基盤を提案している．ネットワーク上の認証基盤は，メールサーバへのアクセス，外部ネットワークに接続，ファイルサーバからのデータの取得など，ネットワーク内のサービスにアクセスするユーザのさまざまな行為の正当性を示すために使用される．これら各サービスに対する認証を行うため，それぞれの管理ポリシーに従って認証基盤が存在している．しかしながら，サービスの量の増加に伴って認証基盤の量はますます増加しており，サービスへのアクセス制御を行うためのネットワーク制御機器の管理運用コストも非常に大きくなってきている．そこで，これらの問題点を OpenFlow によるネットワーク制御と認証基盤の連携を行うことにより解決する．認証後のネットワーク制御を OpenFlow によって統合することにより，ネットワーク制御機器の管理運用コストを低減可能である．

前述の通り，認証基盤はサービスごとに様々なものが存在するが，本稿ではディレクトリサーバのひとつである Lightweight Directory Access Protocol(LDAP)[3] を用いることとした．LDAP は高速な検索や読み込みに特化したディレクトリサービスであり，一般にユーザの認証情報の管理を行うためのデータベースサービスとして知られている．LDAP のメリットとして，

- 読み込み，検索が高速であるため，認証用データベースに適している
- 対応しているアプリケーションが非常に多く，他のアプリケーションとの連携が容易
- リレーショナルデータベース (RDB) とは異なり，ユーザに付随する様々な情報の管理，検索が高速に実行可能

である事があげられる．特に，メールや認証情報など様々な情報の管理，検索が高速に実行可能であるため，認証を行うユーザに付随する情報を検索し，ネットワーク制御を行うという一連の動作を高速で実行する事が可能である．また，対応しているアプリケーションが多く，現在認証基盤としての LDAP サーバが非常に一般的である事から，本稿の提案システムをより汎用的なものとするために LDAP は適切な認証基盤であると言える．

5. 設計

5.1 システム概要

本研究で目指す実現目標は，認証基盤と OpenFlow コントローラの連携によってネットワークのアクセス制御を行うことである．認証情報とその管理には，大規模ネットワークで一般的なディレクトリサービスである LDAP を用いる．各ユーザがアクセス可能なネットワーク資源を決定するための情報を属性情報として付与させ，ネットワーク上を流れるパケットの情報と組み合わせることで，似通ったパケットを動的に区別する，講義や部屋が持つユーザ情報以外の属性情報を利用した大まかな制御を行うなど，ネットワークの利用範囲を柔軟に制御させる．

対象とするキャンパスネットワークには，管理者の管理

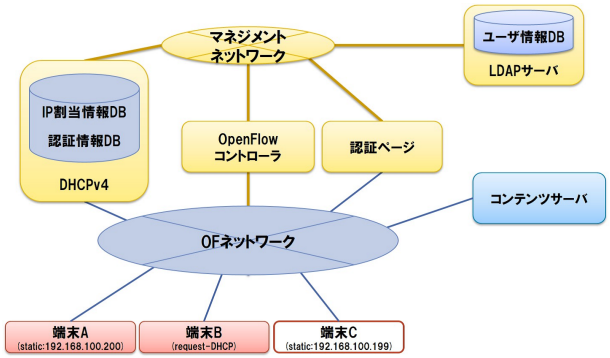

図 1 論理構成

下にある OpenFlow コントローラ，OpenFlow スイッチ，LDAP サーバ, DHCP サーバが存在し, それぞれ OpenFlow コントローラを中心に連携をとる．またユーザ側としてユーザ毎の端末やユーザがアクセスする学内コンテンツを走らせるための Web サーバが存在することとする．

論理構成は図 1 に示した通りである．図の上半分はマネジメントプレーンでありネットワークの管理者が責任を負い，下半分のデータプレーンでは主にユーザや学内コンテンツ間の通信が起こる．また，ネットワーク管理者に申請することで Web(コンテンツ) サーバを設置することも想定する．システムは IP アドレスの割り当て，ユーザ認証，ネットワークへのアクセス許可の 3 フェーズから成り，それぞれのフェーズでは DHCP サーバ，LDAP サーバ，OpenFlow コントローラが主として動作する．

管理ネットワーク内部に存在するすべての装置にファイアウォール機能を設定できるため，不正ユーザの攻撃によって通信を直接受ける装置に不具合が発生したとしても，コントローラに許可されないフローがネットワーク上を流れることはなく，影響は極めて限定的になるという特徴を持つ．さらにネットワーク管理者が利用者の情報を管理することで利用者の不明な通信を拒否できるほか，利用者の特定によって責任の所在を明確にすることが容易になる．

5.2 IP アドレスの割り当て

本研究で対象として想定しているキャンパスネットワークでは，以降のユーザ認証時に認証情報と結び付けるための MAC アドレスと IP アドレスのペアを OpenFlow コントローラが管理している．そこで，学内ネットワークへ正規の手段で接続していることを確認する一つの手段として DHCP サーバによる IP アドレスの払い出しを用いることとした．IPv4 を対象とした DHCPv4 では五つのメッセージ {DISCOVER, OFFER, REQUEST, DHCPACK, DHCPNACK} が存在するが，このうち DISCOVER と DHCPACK メッセージを注目することとする．また，DHCP に用いられるメッセージは UDP の 67，68 ポートが用いられるため，以後の識別ではこれを前提としてマッチングに用いる．

まず，DISCOVER メッセージはユーザがブロードキャスト転送によって DHCP サーバを探すために送信される．そこでブロードキャストされた通信パケットはすべてコントローラへと Packet-In させる．Packet-In によって受け取ったデータは即座に DHCP サーバへと転送すると同時に，以降のユーザ，DHCP サーバ間の通信を許可するためのフローを OpenFlow スイッチへ追加する．これによって OFFER, REQUEST, DHCPACK, DHCPNACK のやり取りがコントローラを介さずに行われる．DHCPACK メッセージが送信され，IP アドレスの割り当てが完了すると DHCP サーバはその旨をデータベースへ保存する．以降ではこの端末情報と払い出した IP アドレスが一致する，もしくは静的に設定済みの通信を正規のものとして認証サーバへのアクセスを許可する．

5.3 ユーザ認証

ユーザは，初めにネットワークの使用権限を得るために認証を行わなければならない．予め設定されたユーザ ID, パスワードによってログインを試行し，確認が取れた場合は処理が続行され，ログインを行ったユーザーの ID とそれに含まれる属性を OpenFlow コントローラから直接参照可能なデータベースへ保存すると同時に OpenFlow コントローラへ通知することで同期を図る．このとき開かれる通信路は認証要求を行う端末とディレクトリサーバ間に限定され，ユーザの認証要求を含むパケットは管理ネットワーク外部を流れることはなく，不必要に拡散されないのでパスワードなどの認証情報の流出を防ぐ．

5.4 ネットワークへのアクセス許可

5.3 節によってログイン情報を受けたコントローラは，5.2 節に示したように，正規に割り当てられたものであることを確認し，MAC アドレスと IP アドレスをキーとしたフローエントリーを各スイッチへ設定する．新たにスイッチが加わったり，ネットワークトラブルによって一時的に切断されていたスイッチが復帰するなどでスイッチのコンフィグレーションが必要な場合，OpenFlow コントローラは IP アドレスの割り当て状況と認証を受けているユーザ情報の二つをデータベースへ確認し，再設定を行う．

ただし，フローエントリーを OpenFlow コントローラが能動的に削除する場合が二つある．一つは DHCP サーバによって割り当てられた IP アドレスに設定されたリース期間が切れた時である．期限が切れたとき，ユーザは再度 DHCP サーバへアクセスし再割り当てを受ける必要があるが，必ずしも以前と同じ IP アドレスが割り当てられるとは限らないため，この端末からのフロー情報を更新しなければならない．スイッチに設定したフローエントリーが設定された秒数後に自動削除する hard timeout によって

表 1　実行環境

ホスト	プロセッサ	Intel® Xeon® Processor X3430 CPU 2.40GHz
	メモリ	8.0GB
	OS	Ubuntu12.04 LTS server(64bit)
仮想	コントローラ，スイッチ，ホスト	
	OS	Ubuntu 12.04 LTS(64bit)
	メモリ	1.0GB

これを実現する．このとき，Flow-Removed メッセージによって OpenFlow コントローラへ通知することで更新手続きを行う．もう一つはユーザがログアウトを行った場合である．不要なフローエントリーを削除するため，ログインと同様の手段でデータベースと OpenFlow コントローラの同期を行ってから関連するフローエントリーを削除する．

6. 実装

6.1 環境

本研究の実験環境は表 1 の通りである．OpenFlow コントローラはバージョン 1.3 相当の Trema-Edge を用い，これを処理するためスイッチには Open vSwitch-1.11.0 を利用した．これらは OpenFlow1.0 に比べ IPv6 対応やマッチフィールドの拡張が行われており，将来の拡張として採用した．連携するシステムは，DHCP サーバに isc-dhcp-server を，LDAP に OpenLDAP を利用し，これらのログ情報は MySQL サーバに保存している．OpenFlow コントローラは必要な時にこのログ情報を参照する．

また，OpenFlow と認証基盤間における認証情報のやり取りには Remote Procedure Call(RPC) の実装である MessagePack-RPC[4] を用いる．ログイン/アウトを行うと同時に OpenFlow コントローラに用意されたログイン/アウトメソッドを呼び出し，引数にユーザ ID を与える．ログイン済みの認証情報を記録したデータベースに接続し，認証基盤から渡されたユーザ ID に紐付けられた属性情報を取得することでアクセス制御を行う．

6.2 導通実験

今回の実験では，ユーザがアクセスする学内コンテンツとして apache2 サーバを用いた RubyCGI ページを作成しており，正規に IP アドレスの割り当てを受け，ログイン処理を行ったユーザのみがその Web ページへアクセスを行える状態を正常とする．ユーザが使用する端末は，静的に IP アドレスを設定済みで OpenFlow コントローラにも登録済み (端末 A)，DHCP サーバから IP アドレスの割り当てを受けた (端末 B)，静的に IP アドレスが設定されているにも関わらず OpenFlow コントローラに未登録である (端末 C)，といった三種類の端末を用意する．予め各端末が未認証の状態で学内コンテンツにアクセスが出来ないことを確認する．初めにユーザはそれぞれの端末を用いて，

```
mysql> select * from Authinglog;
+----------------+---------------------+----------+---------+
| IPAddr         | AuthingTime         | UserName | addInfo |
+----------------+---------------------+----------+---------+
| 192.168.100.200 | 2013-09-26 14:41:11 | hashi    | 2000    |
+----------------+---------------------+----------+---------+
1 row in set (0.00 sec)

mysql>
```

図 3　ログイン後の認証ログ

認証を受けるために指定された Web ページ (認証ページ) にアクセスし，そこで表示された WebUI にユーザ ID とパスワードを入力してログインを行う．ログインを正常に完了した端末は学内コンテンツへアクセスして正常な応答があることを確認する．再度認証ページでログアウトを行うとすべてのネットワークリソースにアクセス出来なくなっていることを確認する．以下に結果を示す．

図 2，図 3 は，ユーザが 192.168.100.200 の正規なアドレスを持つ端末 A から認証要求を行い，正常にログインが完了した後にあるスイッチのフローエントリーと認証ログを記録したデータベースを表したものである．Web サーバには 192.168.100.120 のアドレスを固定しているので，スイッチに対して正しく双方向のフローエントリーが追加されていることと，認証ログにログインを行った端末のアドレスとログインしたユーザ名・ID が記録されていることが確認できた．また，ログアウトを行うと図 2，図 3 で追加されていた項目が削除されており，端末から再度 Web サーバへアクセスを試みても接続ができないことを確認した．端末 B，C も同様の手順で正常な動作を確認し，端末 B では端末 A と同じ動作をすること，端末 C では認証ページのみならず全てのネットワークリソースにアクセスできないことを確認した．

7. 考察

7.1 管理の自由度

OpenFlow 本来の機能として，プログラミングによる動的な設定変更とアプリケーションとしての柔軟性がある．制御の対象として端末，接続経路，通信フローなど物理層からトランスポート層までの組み合わせによって幅広いシチュエーションに対応する．逆にネットワークのエラーに対してもこれらの情報を使うことで早期の要因特定に繋がることが期待できる．また本研究でディレクトリサービスに登録したデータは認証情報に限られていたが，講義情報など様々なデータを適用することで応用が可能となる．

7.2 フィルタリング

本研究におけるアクセス制御では，管理ネットワークを流れるフローのフィルタリングはその通信を最初に受け付ける装置で行われる．過剰な輻輳など，ネットワークの特定の場所で起きる事象ではそれを受け付ける装置に異常を

```
nttdata@ub64-of13:~$ sudo ./dump-flows_ofsw.sh | grep 100.200
[sudo] password for nttdata:
 cookie=0x10, duration=26.802s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=102,ip,dl_src=52
:54:00:b5:d7:94,dl_dst=52:54:00:01:ce:74,nw_src=192.168.100.200,nw_dst=192.168.100.120 actions=output:1
 cookie=0xf, duration=26.802s, table=0, n_packets=0, n_bytes=0, send_flow_rem priority=101,ip,dl_src=52:
54:00:01:ce:74,dl_dst=52:54:00:b5:d7:94,nw_src=192.168.100.120,nw_dst=192.168.100.200 actions=output:2
 cookie=0xe, duration=65.766s, table=0, n_packets=27, n_bytes=5507, hard_timeout=90, send_flow_rem prior
ity=32,ip,dl_src=52:54:00:96:49:c4,dl_dst=52:54:00:b5:d7:94,nw_src=192.168.100.105,nw_dst=192.168.100.20
0 actions=output:2
 cookie=0xd, duration=65.766s, table=0, n_packets=32, n_bytes=7157, hard_timeout=90, send_flow_rem prior
ity=30,ip,dl_src=52:54:00:b5:d7:94,dl_dst=52:54:00:96:49:c4,nw_src=192.168.100.200,nw_dst=192.168.100.10
5 actions=output:3
```

図 2 ログイン後のフローエントリー

きたすことはあっても，その先にフローが流れず，フィルタリングを行う装置自体も末端に属するため，ネットワーク全体へ波及することは少ないと考えられる．また不正なフローが排除されることでネットワーク全体を流れるフローも削減することができる．同時にユーザの利用状況をシステムや管理者が把握することも容易になるため，ユーザのレベルに応じて帯域制御を行うことも可能である．

7.3 フローエントリ欠如によるスイッチでの転送遅延

ログイン，ログアウトを前に示した正規の手順で行った場合，ユーザ認証とネットワークアクセス制御はほぼ同時に行われるため，コントローラと実際のネットワークにおける差異はあまりない．しかし，ネットワーク障害や強制シャットダウンなど予期せぬ切断があったとき，スイッチから必要なアクセス情報が失われたり，不必要な情報が残ったりすることが考えられる．失われている場合についてはコントローラへ認証の有無を確認することで書き戻すことができるが，不必要な情報が残っている場合はそれを検知しだい削除しなければならない．本稿では hard timeout のみ用いたが，最後のマッチングしてから一定時間経過するとフローエントリを削除する idle timeout も設定すれば早い段階での不必要なエントリの削除が期待できる．しかし，例えば共有端末において異常終了が起き，すぐさま別のユーザがログインした場合，端末の IP アドレスが変更されないままユーザ情報が残ってしまう可能性が残されている．また，無線デバイスを携帯してネットワークを利用しながら学内を移動するなど，学内ネットワークへ接続する際に利用するスイッチが変化するときにフローの書き換えが必要になるが，これが頻繁に発生するとスイッチへの適用が遅れた際にパケットロスが発生する．

7.4 フローの偽装

7.3 節でも述べたように，意図するかどうかに限らず，MAC，IP アドレスを偽装されたパケットはフィルタリングを通過してしまう．しかし次に挙げるような事象が起きているときは異常である可能性が高いため，再度の認証を要求する方が良いと考えられる．

- 端末が直接接続している装置が同時に複数存在する
- 該当端末へ ARP など返答が必要なパケットの送出

8. まとめ

本稿では，アプリケーションと連携したネットワーク制御の一つとして，認証基盤と OpenFlow コントローラが連携してネットワークレベルでのアクセス制御を行う実例を示した．実現した機能としては，正規のアドレスを持たない端末のアクセスを著しく制限すること，ログイン，ログアウトによってネットワークアクセスの不可避を制御することの二点であり，これによって単体のネットワークについてアクセス制御は実現した．今後は，実ネットワークで運用することを想定したシステムに対する負荷の検証を行うことが必要となる．

今後の展望として，認証を行ったユーザが所属するネットワーク別のアクセス制御，Kerberos，RADIUS などの別認証基盤との連携が考えられる．ネットワークは特定の所属によって意味のあるサブネットワークに分けることができるが，ハードウェア資源自体は複数のサブネットワークが共有している．ここで，ユーザに与えられた属性情報やその接続場所も利用することで柔軟にアクセス範囲を変える．例えばある研究室に所属する学生が室内に設置したサーバに外部からアクセスする場合，認証されたユーザは制限を受けないが同じ場所から別の学生がアクセスすると遮断されてしまうといったことができるほか，講義単位に通常ブロックされるポート番号を属性情報として保持しておくことで，必要な時にだけ動的に開放されるといったことも可能となる．また別認証基盤を利用する場合，その特徴を活かした制御方法を提案していく必要がある．

参考文献

[1] 「Open Networking Foundation」 https://www.opennetworking.org/
[2] 「OpenFlow Switch Specification Version 1.3.2 Implemented (Wire Protocol 0x04)」 Apr 25, 2013
[3] 「Open LDAP」 http://www.openldap.org/
[4] 「MessagePack」 http://msgpack.org/