情報処理学会論文誌 Vol.56 No.9 1782–1793 (Sep. 2015)

次世代ネットワークに向けた ネットワーク利用者認証システムの設計と実装

末永 光弘[1,a)] 田中 久治[1] 大谷 誠[2] 堀 良彰[3] 岡崎 泰久[1] 渡辺 健次[4]

受付日 2014年12月8日, 採録日 2015年6月5日

概要：ネットワークの複雑化とそれにともなう通信機器の増加により，近年，ネットワーク管理者の管理コストが大きくなっている．機器に個別のネットワーク設定が不要で管理者の負担軽減が期待できるネットワークは今後重要になると考えられる．これらの条件を満たすネットワークとして SDN という概念が提唱されているが，SDN においては，ネットワークや Web システムなどのリソースの利用において重要な，認証による利用資格を確認を行う機能を SDN アプリケーションとして実装する必要がある．そこで我々は，次世代ネットワークに向け，Web によるシングルサインオンに対応したネットワーク利用者認証システムを設計し，OpenFlow を用いて実装した．本システムでは，Shibboleth 認証の成否を OpenFlow に通知する方法，および利用監視により利用終了を検知する方法の実装により，SDN アプリケーションとして，ネットワーク利用者認証システムを実現した．

キーワード：OpenFlow，Shibboleth，シングルサインオン，ネットワーク利用者認証

Design and Implementation of Network User Authentication System for Next Generation Network

Mitsuhiro Suenaga[1,a)] Hisaharu Tanaka[1] Makoto Otani[2] Yoshiaki Hori[3] Yasuhisa Okazaki[1] Kenzi Watanabe[4]

Received: December 8, 2014, Accepted: June 5, 2015

Abstract: With the increase of communication devices according to complexity of the network, management costs of network administrators are increasing in recent years. It is considered that the network which does not need an individual network setup to devices and can expect an administrator's reduction of incidence will become important from now on. Concept called SDN have been proposed as network which these conditions are satisfied. In SDN, it's necessary to implement the function which confirms the use qualification by authentication in use of "resources of a network and web systems" as SDN application. In this research, we have designed the network user authentication system which is adapted for a Single Sign-On with Web for a next generation network, and have implemented this system using OpenFlow. In this system, we have realized a network user authentication system as a SDN application by implementations of methods of notifying the success of the Shibboleth authentication to OpenFlow and detecting the end of use by the monitoring of the use.

Keywords: OpenFlow, Shibboleth, Single Sign-On, network user authentication

1. はじめに

ネットワークリソースの利用時に，認証により利用資格

[1] 佐賀大学大学院工学系研究科
Graduate School of Science and Engineering, Saga University, Saga 840–8502, Japan
[2] 佐賀大学総合情報基盤センター
Computer and Network Center, Saga University, Saga 840–8502, Japan
[3] 佐賀大学全学教育機構
Organization for General Education, Saga University, Saga 840–8502, Japan
[4] 広島大学大学院教育学研究科
Graduate School of Education, Hiroshima University, Higashihiroshima, Hiroshima 739–8524, Japan
[a)] suenaga@ai.is.saga-u.ac.jp

を確認することは一般的となっている．また企業や大学などにおいては，複数の Web システムの ID・パスワードをユーザが個別に管理する負担を軽減するために，シングルサインオンによる認証が注目されている．次世代のネットワークにおいても，これらを実現することは重要である．

次世代のネットワークとして，SDN（Software Defined Network）と呼ばれる，プログラマブルであり，仮想化が可能なネットワーク制御技術の研究・開発が進められている．SDN の特徴の 1 つとして，プログラマブルであることを前提とした多数のネットワーク機器の集中管理や自動化があげられ，ネットワークの複雑化により増大する管理コストや管理者への負担を軽減できると考えられている．

SDN に基づくネットワーク制御の実装の 1 つとして，Open Network Foundation を中心にグローバルな枠組みで開発が行われている OpenFlow [1] がある．OpenFlow はすでにデータセンターなどでの活用が開始され，各ベンダからの商用スイッチやコントローラの販売が始まっている [3]．今後は OpenFlow やその他の SDN を利用することで管理コストを軽減し，フレキシブルなリソースの活用を行うネットワークが普及していくと考えられる．

OpenFlow/SDN が今後普及していくにあたって，既存のネットワークにおいて利用されていたサービスや機能を OpenFlow/SDN においても利用できる必要がある．しかし，現在の OpenFlow/SDN の利用は，データセンタなどの業務分野における，負荷分散や管理コストの削減を目的としたケースがほとんどであり，OpenFlow/SDN の認証技術やアクセス制御への応用研究は活発に行われているとはいえない．OpenFlow/SDN における認証およびセキュリティの研究としては，アクセス制御に関する研究 [4] や，OpenFlowSec [5] による取り組みなどがあるが，これらはネットワークリソースの利用時における利用者認証機能を実装するものではない．そのため，OpenFlow/SDN 上で動作するネットワーク利用者認証システムをアプリケーションとして設計・実装する必要がある．特に，認証においてはシングルサインオンなどのセキュアな認証技術を提供できることが望ましい．

そこで本研究では，OpenFlow/SDN の認証・アクセス制御分野での応用を目的とし，OpenFlow ネットワークに導入可能な，Shibboleth [2] によるシングルサインオン認証に対応した，ネットワーク利用者認証システムの設計と実装を行った [6], [7]．Shibboleth を用いたネットワーク利用者認証機能を，OpenFlow のアプリケーションとして実装する場合，Shibboleth における認証の成否を OpenFlow コントローラ/スイッチに通知する方法が課題となる．本システムでは，特定の形式のパケットを用いた OpenFlow コントローラ/スイッチに対するネットワーク開放/閉鎖の通知と，ユーザ端末の生存監視によるネットワーク利用終了の検知により，OpenFlow を導入したネットワークにおいて，Web ブラウザを用いたネットワーク利用者認証が可能である．ユーザ端末の生存監視には，WebSocket によって実装した接続監視サーバを用いる．ユーザ端末と接続監視サーバの間の WebSocket コネクションの維持により，ユーザ端末のネットワーク接続監視を行う．WebSocket コネクションによる接続監視以外にも，定期的に ARP リクエストを送信し，ユーザ端末からの応答の有無による生存監視を行う．OpenFlow の利用により，管理者は機器のネットワーク設定を個別に行うことなく，フレキシブルにネットワークを構築し，少ない負担で管理することができる．ユーザ端末には特別なソフトウェアやハードウェアをインストールすることなく，Web ブラウザのみで認証が可能である．また，Shibboleth によるシングルサインオン認証により，セキュアな認証を実現し，複数の Web システムをシームレスに利用でき，学術認証フェデレーション（学認）[8] による，複数の組織にまたがる訪問者の利用認証も可能である．

2. システムの概要

2.1 システムの設計

本システムでは，ネットワーク利用時に Web ブラウザを用いた Shibboleth によるシングルサインオン認証を採用する．そこで，ユーザ端末による認証前の Web アクセスを検知した場合，IdP（Identity Provider）へ誘導し，認証させることが望ましい．

認証成功後，ユーザ端末に対してネットワークを開放する．OpenFlow を用いるネットワークにおいては，OpenFlow コントローラの命令により OpenFlow スイッチにフローエントリを書き込むことでユーザ端末による通信が可能となる．認証を行った Web システムは認証の成功を知ることができるが，OpenFlow コントローラや OpenFlow スイッチは認証の成功を直接知ることができない．OpenFlow スイッチはパケットをフローエントリに従って転送する機能しか持たず，OpenFlow コントローラはパケットの処理方法を判断し，スイッチに指示する機能しか持たないためである．また，OpenFlow の仕様としても，パケットのペイロード部は先頭 20 バイトまでしか参照できず，ペイロード部の内容に基づいた判断を行うことも難しい．そこで，OpenFlow コントローラ/スイッチに対して認証の成功を通知し，その通知に基づいてネットワークを開放する必要がある．

ネットワークの開放後，ネットワークトラブルやユーザ端末のトラブルによりネットワークの利用が突然終了してしまうことがある．また，ユーザ端末のシャットダウンなどにより，ユーザはネットワークの利用を正常に終了する場合もある．このとき，ユーザ端末に対するネットワークの開放が行われたままでは，セキュリティ上のリスクが発生する．そこで，トラブルやユーザの意志によるネットワーク利用の終了を検知するため，ユーザ端末の接続監視

や定期的な生存の確認を行い，利用の終了を検知した場合や生存確認に応答がなかった場合にネットワークを閉鎖する必要がある．

以上の点をふまえると，OpenFlow を用いた Web ブラウザによる Shibboleth 認証を行うためは次に示す機能が必要である．

- 利用開始の検知と認証ページへの誘導
- 認証成功の通知とネットワークの開放
- 利用終了の検知とネットワークの閉鎖

本システムでは，これらの機能を OpenFlow コントローラおよび接続監視サーバや CGI プログラムなどとして実装し，OpenFlow を用いるネットワークにおける Shibboleth による利用者認証を実現した．

2.2 システムの構成

本システムの構成を図 1 に示す．

本システムに必要な機器について以下に記述する．

2.2.1 接続制御サーバ

Shibboleth 認証によりユーザの接続制御を行う SP（Service Provider）や，接続監視サーバを設置し，認証成功時および利用終了時の処理に必要な Web ページや CGI を保持する．認証を行っていないユーザ端末が Web の利用を開始すると，OpenFlow コントローラ/スイッチは，ユーザ端末からの HTTP リクエストを接続制御サーバへリダイレクトする．次に，未認証端末からの通信であるので，認証させる IdP にリダイレクトする．

認証が成功した場合，OpenFlow コントローラの動作する OpenFlow 制御サーバに認証成功を通知する．また，接続監視サーバが Web ブラウザに認証成功ページを返す．この認証成功ページはユーザ端末の Web ブラウザと接続監視サーバとの間に WebSocket コネクションを作成し，コネクションの維持によりユーザ端末の監視を行う．ユーザによるブラウザの終了や，コネクションの切断を検知すると，認証システムは利用を終了したと判断し，OpenFlow 制御サーバに対して当該端末のネットワーク利用終了を通知する．

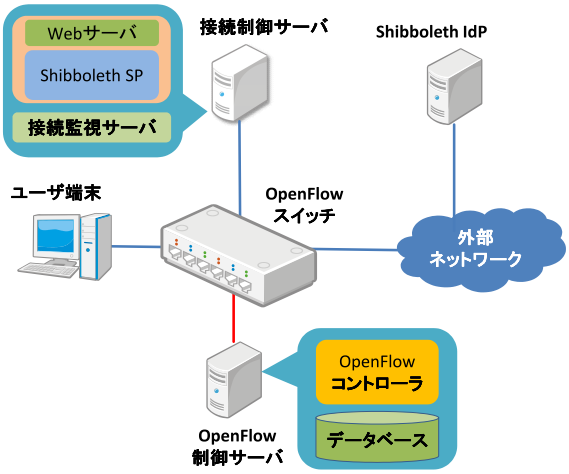

図 1 システム構成
Fig. 1 System Architecture.

2.2.2 OpenFlow 制御サーバ

OpenFlow を使用するネットワークの制御を行う．本システムにおいては，制御機能を実装した OpenFlow コントローラおよび利用者情報を格納するデータベースを設置する．OpenFlow コントローラは，OpenFlow スイッチから未認証のユーザ端末によるアクセスを検知すると，接続制御サーバにリダイレクトすることで Shibboleth によるユーザ認証を促す．

認証成功後に接続制御サーバから認証成功の通知を受け取ると，当該ユーザ端末による通信を許可し，ネットワークを開放する．また，利用終了時には当該端末に対するネットワーク利用許可を取り消し，ネットワークを閉鎖する．

ユーザ端末の生存確認を行うため，OpenFlow スイッチに対し，ユーザ端末への定期的な ARP リクエストの送信を命令する．ユーザ端末からの応答がない場合は当該端末がネットワーク上に存在しないと見なし，ネットワークの閉鎖と利用者情報の削除を行う．

2.2.3 OpenFlow スイッチ

OpenFlow を使用するネットワークにおいてデータ転送を行う．本システムにおいてはデータ転送のほかに，OpenFlow コントローラの命令によりユーザ端末へ送信した ARP リクエストに対する応答の有無を，OpenFlow コントローラに通知する．

2.2.4 Shibboleth IdP

IdP（Identity Provider）は，Shibboleth 認証を行うための認証サーバである．一度 IdP で認証を行うと，IdP と連携する SP ではシングルサインオンが可能なため，再度認証をする必要がない．

未認証のユーザ端末が SP にアクセスすると，通信は IdP へリダイレクトされ，認証ページを表示し，認証を促す．

2.3 システムの動作

利用開始から利用認証成功までのシステムの動作を図 2 に基づき，以下に示す．

(1) ネットワークを利用するユーザ端末から HTTP パケットが OpenFlow スイッチに届く．
(2) OpenFlow スイッチは OpenFlow コントローラに対して処理方法を問い合わせる．
(3) OpenFlow コントローラはユーザ端末からのパケットの宛先を接続制御サーバ宛に書き換えるように命令する．
(4) 書き換えられたパケットは接続制御サーバへ転送される．
(5) 接続制御サーバがパケットを受け取り，SP は IdP へとリダイレクトする．

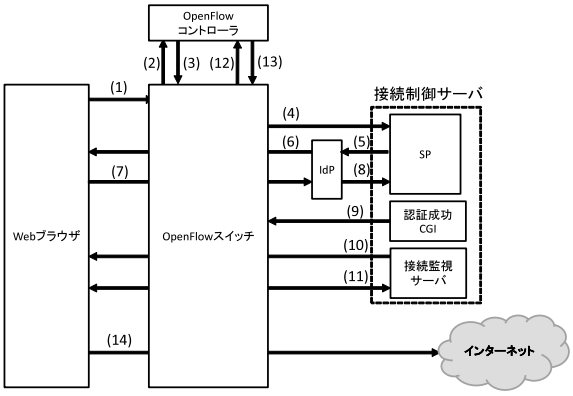

図 2 ネットワーク利用開始時の動作フロー
Fig. 2 Operation flow at the start of network use.

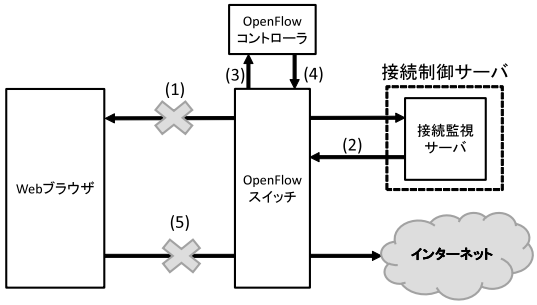

図 3 ネットワーク利用終了時の動作フロー
Fig. 3 Operation flow at the end of network use.

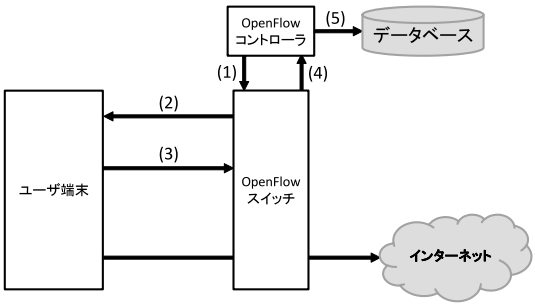

図 4 ARP によるユーザ端末生存監視の動作フロー
Fig. 4 Operation flow to monitor the existence of user terminals with ARP.

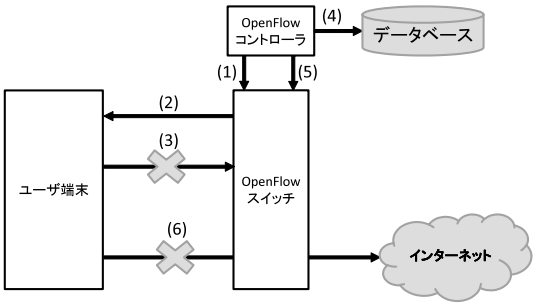

図 5 ARP 未応答のユーザ端末に対するネットワーク閉鎖の動作フロー
Fig. 5 Operation flow to close the network for user terminals which non-response to ARPs.

(6) 認証サーバである IdP からユーザ端末に対して認証ページが返される．
(7) ユーザが ID とパスワードを入力し認証をする．
(8) 認証に成功すると SP へ通知する．
(9) SP が通知を受け取ると，認証成功 CGI が認証成功通知パケットを送信する．
(10) 接続監視サーバはユーザ端末に認証成功ページを返す．
(11) 接続監視サーバはユーザ端末の監視を開始する．
(12) OpenFlow スイッチは OpenFlow コントローラにユーザ端末が認証に成功したことを伝え，(9) で作成され，不要となったパケットを削除する．
(13) OpenFlow コントローラはユーザ端末の通信を許可し，その後の通信ではユーザ端末に対して MAC アドレスを条件としたフローエントリを作成する．
(14) ユーザ端末は自由にネットワークを利用できる．

利用終了のシステムの動作を**図 3** に基づき，以下に示す．
(1) ブラウザが閉じられると接続監視サーバは WebSocket コネクションの切断を検知する．
(2) 接続監視サーバは利用終了通知パケットを送信する．
(3) OpenFlow スイッチは OpenFlow コントローラにユーザ端末がネットワークの利用を終了したことを伝え，(2) で作成され，不要となったパケットを削除する．
(4) OpenFlow コントローラはユーザ端末の利用許可を取り消し，利用許可を失った端末に対するフローエントリを削除する．
(5) ユーザ端末に対するネットワークは閉鎖される．

ARP によるユーザ端末生存監視の動作を**図 4** に基づき，以下に示す．
(1) OpenFlow コントローラはユーザ端末に対して定期的に ARP パケットを送信するように OpenFlow スイッチに指示する．
(2) OpenFlow スイッチは ARP パケットをユーザ端末に対して送信する．
(3) ユーザ端末は ARP リプライを返す．
(4) OpenFlow スイッチは OpenFlow コントローラに対してユーザ端末ごとに ARP リプライがあったことを通知する．
(5) OprnFlow コントローラは ARP に応答したユーザ端末の ARP 未応答回数を 0 にリセットする．

また，ARP に対して，本システムで設定した回数を越えて応答しないユーザ端末に対するネットワーク閉鎖について，**図 5** に基づき，以下に示す．
(1) OpenFlow コントローラはユーザ端末に対して定期的に ARP パケットを送信するように OpenFlow スイッチに指示する．
(2) OpenFlow スイッチは ARP パケットをユーザ端末に対して送信する．
(3) ユーザ端末からの ARP リプライが返らない．

(4) OpenFlow コントローラは，ARP リプライを返さない端末の未応答回数を 1 増やす．
(5) ARP 未応答回数が規定回数に達したユーザ端末に対するフローエントリと通信許可を取り消す．
(6) 該当するユーザ端末に対してネットワークは閉鎖される．

3. システムの実装

本項では作成したシステムの実装について述べる．本システムの OpenFlow コントローラプログラムの実装には，NEC の開発している OpenFlow コントローラ開発フレームワークである Trema [9] を使用した．

3.1 OpenFlow コントローラ設定ファイル

OpenFlow コントローラの動作に必要な情報を設定ファイルに記述する．設定ファイルに記録されている情報を次に示す．

- OpenFlow スイッチの IP アドレス
- OpenFlow スイッチの MAC アドレス
- 接続制御サーバの IP アドレス
- 接続制御サーバの MAC アドレス
- フローエントリ削除までの待機時間
- ARP リクエスト送信間隔
- ARP 未応答許容回数

OpenFlow スイッチの IP アドレスおよび MAC アドレスは，生存確認のためのユーザ端末への ARP リクエスト送信時に送信元として ARP パケットに設定される．APR リクエストを受け取ったユーザ端末は，送信元として設定されているこれらの IP アドレスおよび MAC アドレス宛に ARP リプライパケットを送信する．

接続制御サーバの IP アドレスおよび MAC アドレスは認証成功および利用終了を OpenFlow コントローラに通知するパケットの送信元の判別に用いる．送信元が接続制御サーバの IP アドレスおよび MAC アドレスであり，特定の TCP ポート番号に当てたパケットであれば認証成功通知パケットあるいは利用終了通知パケットであると判別する．

フローエントリ削除までの待機時間は，使われないフローエントリが残留し，フロー数が増えすぎることを防止するために使用する．接続制御サーバとの通信に用いるものや，DNS，DHCP などといったネットワークに必須の通信に使う特殊なフローエントリを除き，ユーザの通信のために登録されたフローエントリは，各エントリの条件に合致する最後の通信から，この項目に設定されている時間が経過すると，自動的に削除される．

ARP リクエスト送信間隔と，ARP 未応答許容回数は，ARP による生存確認に使用する．ユーザ端末への ARP リクエストは ARP リクエスト送信感覚時間ごとに定期的に送信され，ARP 未応答許容回数連続して ARP 応答がない場合，該当するユーザ端末への通信許可を取り消す．

これらの情報は OpenFlow コントローラの起動時に読み込まれる．

3.2 OpenFlow コントローラ起動後の初期化

OpenFlow コントローラの起動後，初期化処理として ARP パケット作成用のインスタンスと ARP パケット送信フラグ，利用者情報データベースのテーブル作成を行う．ARP パケットの作成は OpenFlow コントローラで読み込んだ外部モジュール [10] を使用する．このように，OpenFlow の標準仕様では ARP パケットを作成する機能はなく，別途実装が必要である．

ARP パケット送信フラグはユーザ端末に対して生存確認の ARP リクエストを送信したかどうかをデータベースに記憶する．生存確認の ARP リクエストを送信した場合には true，未送信の場合には false となる．起動直後の初期値は false である．ARP パケット送信後は true となる．

また，利用者情報の保持のために SQLite3 によるデータベースを使用するが，初回の起動であるなどの理由でデータベースが存在しない場合は，OpenFlow コントローラの命令により自動的に作成される．この情報は，ユーザ端末の情報，通信の許可/不許可を保持する．このデータベースの情報を参照し，OpenFlow コントローラおよび OpenFlow スイッチはパケットの転送の可否の判断や転送先の決定を行う．

利用者情報を保持する arp_table テーブルの定義を**表 1** に示す．主キーは mac_addr フィールドである．

in_port フィールドは，ユーザ端末などの，OpenFlow スイッチと接続する端末やサーバが接続している物理ポート番号を格納している．この物理ポート番号は，パケットの入力元となる物理ポート番号の確認や，パケットの転送先の設定などに使用する．

permit フィールドはネットワーク利用の可否を表し，未認証の場合は “0”，認証成功後は “1” が設定される．認証成功後，ユーザ端末からの通信はこのフィールドの値が “1” である場合にのみ該当する接続先に転送される．

表 1 arp_table テーブルの定義
Table 1 The definition of the arp_table table.

フィールド	型	説明
mac_addr	text	ユーザ端末の MAC アドレス
ip_addr_src	text	ユーザ端末の IP アドレス
in_port	integer	ユーザ端末が接続されている物理ポート番号
permit	integer	ユーザ端末に対する利用許可
arp_judge	text	ユーザ端末からの ARP リプライの有無
count	text	ARP に対する未応答回数

arp_judge フィールドはユーザ端末からの ARP リプライの有無を判定するテキストが入る．OpenFlow スイッチが定期的な生存確認の ARP リクエストを送信すると，このフィールドの値は “send” に設定される．ユーザ端末からの ARP リプライを OpenFlow スイッチが受け取ると，値 “reply” に更新される．

count フィールドは ARP への未応答回数が記録される．この数値は ARP へ応答すれば 0 に更新され，未応答のたびに 1 ずつ増加する．未応答回数が 5 となると，OpenFlow コントローラは，該当ユーザ端末は生存していないと判断し，フローエントリを削除し，通信許可を取り消す．

3.3 スイッチの接続/切断検知

Trema に実装されたイベントハンドラにより，OpenFlow コントローラと OpenFlow スイッチの接続/切断を検知できる．これを利用し，OpenFlow コントローラのコンソール上に OpenFlow スイッチの接続/切断を知らせるメッセージを表示する．

スイッチの接続を検知すると，ユーザ端末が Web 認証を行うために必要な DHCP，DNS などのパケットに対して通信を許可するフローエントリを書き込む．

また，ネットワークトラブルなどにより，OpenFlow スイッチと OpenFlow コントローラの接続が切断された場合，利用者情報データベース内の利用許可をすべて不許可に変更し，起動時にスイッチに書き込んでおいた，コントローラと切断された場合のための，あらゆる通信を遮断する緊急エントリが有効となり，全端末の通信を停止する．OpenFlow コントローラと OpenFlow スイッチの再接続が行われた場合，ユーザ端末は再度認証を行う必要があり，ネットワークセキュリティ上の安全も確保される．

3.4 データベース登録と認証ページへの誘導

OpenFlow スイッチに新たなユーザ端末が接続され，Web ブラウザによりネットワーク利用を開始すると，利用者情報データベース（表 1）に登録が行われる．permit フィールドの初期値は “0” であるため，Web ブラウザ起動直後の通信は接続制御サーバへとリダイレクトされる．

ここでは，ユーザ端末が Web ブラウザ起動時にアクセスしようとした宛先の IP アドレスを，OpenFlow のパケット書き換え機能を用いて強制的に接続制御サーバの IP アドレスおよび MAC アドレスに書き換えることにより，リダイレクトさせる．接続制御サーバへ強制的にリダイレクトされたのち，未認証であることから IdP へリダイレクトされ，認証ページが表示される．この認証ページにおいて，ユーザは認証を行う．

3.5 認証成功の通知とネットワークの開放

接続制御サーバが認証成功の通知を受け取ると，CGI が動作し，特別なパケットを送信することで認証成功を通知する．このパケットは宛先を認証に成功したユーザ端末の IP アドレスとしており，デフォルトの設定としてポート番号 10000 に対して送信される．また，この CGI は利用ログとして，利用開始日時，ユーザ名，IP アドレスなどを記録する．

このパケットは OpenFlow スイッチでフローエントリに合致しないパケットとして扱われ，OpenFlow スイッチは OpenFlow コントローラへ処理方法を問い合わせる．OpenFlow コントローラは認証成功を OpenFlow コントローラに通知するパケットと判断し，宛先である IP アドレスを持つユーザ端末の情報を利用者情報を保持するデータベースに追加し，通信の可否を許可に変更する．以後，ユーザ端末は自由にネットワークを利用できる．

3.6 フローエントリの登録

ネットワークの利用が許可されたユーザ端末が通信を行う場合，新たな接続先にアクセスするごとに，ユーザ端末とアクセス先の間の通信を許容するフローエントリが個別に作成される．このエントリは，送信元をユーザ端末の IP アドレス，宛先をアクセス先の IP アドレスとしたものである．また，アクセス先からの返信がある場合に備え，送信元をアクセス先の IP アドレス，宛先をユーザ端末の IP アドレスとするフローエントリが登録される．これらのフローエントリには有効な時間が設定されており，各フローエントリに合致する通信が 60 秒以上行われない場合，フローエントリは削除される．

削除されるのはフローエントリのみであり，認証情報は削除されないため，再度，同一の接続先にアクセスを行うとフローエントリは新たに追加される．フローエントリの最終参照から削除されるまでの時間は設定により変更可能である．

3.7 接続監視サーバによるユーザ端末監視

認証成功後，Web ブラウザは認証成功ページへ誘導される．このページは接続監視サーバにより表示され，ユーザ端末と接続監視サーバの間に WebSocket コネクションを作成する．このコネクションの維持により，ユーザ端末のネットワーク接続を監視する．ネットワークを利用する間は，ユーザはこのページの表示を維持する必要がある．

接続監視サーバは，Node.js [11] および Socket.IO [12] を用い，WebSocket によって実装した．WebSocket は独自のプロトコルを使用し，一度サーバとクライアントの間にコネクションを作成すると，利用が終了されるまで同一のコネクションを使用してデータ通信を行う．HTTP による通信のように通信ごとに新たにコネクションを作成する必要がなく，安定した通信と動作の軽量化が可能であるため，ユーザ端末の監視に採用した．

3.8 ARP による端末生存確認

本システムでは接続監視サーバによるユーザ端末の利用監視のほかに，ユーザ端末に対する定期的な ARP リクエストの送信による生存確認も行っている．

ARP リクエストは 60 秒ごとに送信しており，OpenFlow スイッチからの ARP リクエストに対して応答がない場合，利用者情報データベース（表 1）の arp_judge フィールドの値が “send” のままとなる．この際，応答しなかった化数も記録される．定期的な ARP 送信の前に，このフィールドが “send” のままとなっているユーザ端末の未応答回数が 1 つ増加する．5 回連続で応答がないユーザ端末は，生存していないと判断され，データベースからの削除とネットワークの閉鎖が行われる．

ARP リクエストの送信間隔と未応答許容回数は設定変更可能である．

3.9 利用終了の検知とネットワーク閉鎖

ユーザ端末のブラウザの終了や，認証成功ページからの移動などにより WebSocket によるコネクションが切断されると，接続監視サーバはネットワークの閉鎖処理を行う CGI を実行する．

ネットワーク閉鎖処理を行う CGI は，宛先をユーザ端末の IP アドレスとし，デフォルトの設定としてポート番号 20000 に宛てた利用終了を通知するパケットを，OpenFlow スイッチに送信する．また，利用終了日時を利用ログに記録する．

認証成功を通知するパケットと同様に，OpenFlow スイッチは未知のパケットであるとして OpenFlow コントローラに問い合わせ，コントローラは利用終了を通知するパケットであると判断する．宛先となっている IP アドレスを持つユーザ端末に対するネットワーク利用許可を取り消し，当該ユーザ端末の使用するフローエントリの削除を行う．

ユーザ端末から ARP リクエストに対し，規定回数連続して応答がない場合も，OpenFlow コントローラは当該ユーザ端末のネットワーク利用許可を取り消し，当該ユーザ端末の使用するフローエントリの削除を行う．

3.10 未知パケットの処理

OpenFlow には，OpenFlow スイッチがフローエントリに登録された条件に合致しないパケット（未知パケット）を受け取った場合に実行されるイベントハンドラが実装されている．このイベントハンドラを利用し，OpenFlow スイッチに入力されたパケットの情報を取得し，パケットの種別ごとに処理を行う．パケット種別ごとに取得する情報を**表 2** に示す．

ARP 以外の未知パケットについて，送信元 MAC アドレスあるいは宛先 MAC アドレスが通信許可済みのユーザ端末である場合は利用者情報データベースをもとに転送する．通信許可がないユーザ端末の場合，認可の判断を行うべく，SP へとリダイレクトさせる．同一ネットワーク内部から発信された ARP パケットであれば，未知パケットであっても，通信の許可/不許可にかかわらず転送を行う．

表 2 パケット種別ごとの取得情報
Table 2 Acquisition information for each packet type.

パケット種別	取得情報
ARP リクエスト or ARP リプライ	送信元 IP アドレス 宛先 IP アドレス 送信元 MAC アドレス 宛先 MAC アドレス 入力ポート番号
未知パケット	送信元 IP アドレス 宛先 IP アドレス 送信元 MAC アドレス 宛先 MAC アドレス 入力ポート番号 TCP/UDP 送信元ポート番号 TCP/UDP 宛先ポート番号

表 3 開発環境
Table 3 Development Environment.

分類	詳細
コントローラ OS	Ubuntu 14.04
コントローラ開発フレームワーク	Trema 0.4.7 （OpenFlow1.0 対応）
OpenFlow スイッチ	OpenFlow 1.1 for OpenWRT BUFFALO WHR-G301N
データベース	SQLite3 3.7.9
開発言語	Ruby 2.0.0, JavaScript
Shibboleth IdP	2.1.5
Shibboleth SP	2.5.1
接続監視サーバ	Node.js v0.10.28 Socket.IO 1.0.4

3.11 開発環境

開発環境を**表 3** に示す．OpenFlow スイッチとして市販のブロードバンドルータに OpenFlow 対応のカスタムファームウェア [13], [14] を導入したものを使用した．

4. 動作検証

動作検証にあたって，検証用のネットワークを作成し，検証を行った．

まず，各 OS とブラウザで，認証時のネットワーク開放処理と利用終了時のネットワーク閉鎖処理が動作することを確認した．検証に用いた OS とブラウザを**表 4** に示す．

いずれの OS・ブラウザから Web アクセスを行った場合においても，未認証である場合は，OpenFlow コントロー

表 4 動作検証に用いた OS および Web ブラウザ
Table 4 OSs and browsers used for the operation verification.

OS	ブラウザ
Windows 7 SP1	Internet Explorer 11
	Firefox 29.01
	Google Chrome 35.0
Fedora 20	Firefox 29.01
	Google Chrome 35.08
Ubuntu 13.10	Firefox 29.01
	Google Chrome 35.08

ラにより，未認証端末からの未知パケットであると判断され，Shibboleth による認証ページへとリダイレクトされることを確認した．認証後には認証成功ページが表示され，Web の閲覧をはじめとした外部との通信が可能であることを確認した．また，接続監視サーバ上において，当該ユーザ端末との間に WebSocket コネクションによる監視プロセスが動作，OpenFlow スイッチへのフローエントリの書き込み，利用者情報データベースにおける通信許可が行われていることを確認した．

利用終了においては，Web ブラウザの終了にともない，接続監視サーバにおける当該端末に対する WebSocket による監視プロセスの終了，OpenFlow スイッチからのフローエントリの削除，利用ログの記録，利用者情報データベースにおける通信許可の取り消しがすべて行われていることを確認し，正常に利用終了処理とネットワークの閉鎖が行われていることを確認した．このことから，OpenFlow ネットワークに導入可能なシングルサインオン認証機能を実装できたといえる．

次に，外部サイトに対する ping による応答時間計測を行った結果を**表 5** に示す．OpenFlow による認証ネットワークと Opengate [15] による認証ネットワークの 2 つのネットワークで，3 つの外部サイトに対して 100 回ずつの ping 送信を行った．検証では，どちらのネットワークにおいても認証システムを用いてあらかじめ利用者認証を行い，ネットワークの利用許可を得たユーザ端末を用いているため，ユーザの認証情報の入力および認証システムや IdP などにおける認証処理にかかる時間は含まれていない．

ping の送信は Windows 7 SP1 から行い，送信データは 1 回あたり標準サイズの 32 バイトである．ping を送信した対象サーバのリストを次に示す．

- 対象サーバ A：www.saga-u.ac.jp
- 対象サーバ B：www.google.co.jp
- 対象サーバ C：www.yahoo.co.jp

OpenFlow ネットワークにおいてもパケット送信時の損失は 0 であり，正常に転送されていることが確認できた．本システムにおいては，ユーザによる認証後，ユーザ端末の通信先ごとにフローエントリを作成する．ping 送信時の OpenFlow コントローラと OpenFlow スイッチ間の通信を

表 5 ping による応答時間計測
Table 5 Response time measurement by ping.

	対象サーバ	A	B	C
OpenFlow	最小（ms）	2	24	30
	最大（ms）	132	137	174
	平均（ms）	6	26	34
	損失（%）	0	0	0
OpenFlow（初回を除く）	最小（ms）	2	24	30
	最大（ms）	6	28	34
	平均（ms）	3	25	30
	損失（%）	0	0	0
非 OpenFlow	最小（ms）	0	21	27
	最大（ms）	1	23	39
	平均（ms）	0	21	27
	損失（%）	0	0	0

表 6 フローエントリ書き込みによる遅延
Table 6 The delay caused by writing a flow entry.

対象サーバ	A	B	C
最大（ms）	132	137	174
初回を除く平均（ms）	3	25	30
コントローラにおける遅延（ms）	25	28	27
フローエントリ書き込みによる遅延推定（ms）	104	84	117

分析すると，初回の ping 要求をパケットを OpenFlow コントローラが処理し，ユーザ端末と対象サーバの通信を許可するフローエントリを OpenFlow スイッチに書き込む命令を出すまでに 25〜28ms の時間が必要であった．初回の応答では，この命令を出すまでの時間に加え，OpenFlow スイッチが受け取ったフローエントリを書き込む時間が必要であると考えられる．

OpenFlow スイッチのカスタムファームウェアではパケットを記録する機能を使用できないため，フローエントリの書き込みに必要な時間を計測することはできなかったが，初回の応答時間から初回以外の平均応答時間，および OpenFlow コントローラでの処理時間を差し引くと，100ms 程度の時間が必要であると推察できる．**表 6** にこれらに関する数値をまとめる．

2 回目以降の応答は，非 OpenFlow のネットワークと比べ，遅延は 3〜4ms となっており，全体の平均に大きな差はない．

5. 考察

5.1 OpenFlow を用いた認証システム

OpenFlow を用いた認証システムとしては，特定のコンテンツに対する認証とアクセス制御に OpenFlow を利用する研究 [4] がある．この研究では Web 上のコンテンツ

へのアクセスに対し，認証に成功した端末の通信を許可するフローエントリを書き込むことでアクセス制御を行い，特定のコンテンツへのアクセスのための認証基盤としてOpenFlow を用いている．利用の終了には専用の利用終了ページにおいて利用終了手続を行う．

本システムでは，WebSocket によりユーザ端末の接続を監視し，WebSocket の切断検知による終了が可能である．このため，ユーザは終了手続を別途行うことなく，Web ブラウザの終了やユーザ端末のシャットダウンによりネットワークの利用を終了できる．また，Shibboleth 認証に対応し，シングルサインオンにより他の Web システムをシームレスに利用できる．

5.2 OpenFlow スイッチ

検証においては市販のブロードバンドルータに対して組み込みシステム用に開発されている OpenFlow 対応のファームウェアを導入し，接続検証などを行った．このブロードバンドルータは家庭用であることから，商用スイッチと比べると処理性能そのものが低く，遅延なども大きいと考えられる．市販のブロードバンドルータの有線接続用の物理ポートは，1 つのイーサネットを仮想的に複数の物理ポートに分割しているものが多く，物理ポートごとの管理ができないという問題がある．また，同時接続可能なセッション数は，本研究で用いたものでは 2,048 セッション，同一メーカーの上位機種でも 4,096 セッションであり，多数の端末を接続して使用する場合，接続セッション数が不足する可能性がある．一方，商用スイッチの場合，比較的安価な Pica8 シリーズの下位モデルで，同時接続可能なセッション数は 12,000 である．商用の OpenFlow スイッチを使用すれば，本システムも実用に耐えるパフォーマンスを発揮すると考えられる．

5.3 認証成功通知と利用終了通知

3.5 節に記したとおり，現在，利用開始および終了を接続監視サーバから OpenFlow コントローラへ通知する際に，宛先 IP アドレスをユーザ端末の IP アドレスとし，宛先ポート番号を特定の TCP ポート番号としたパケットを送付することにより通知している．

使用している TCP ポート番号はユーザ端末とその他の端末や外部ネットワークとの通信で使用される可能性がある．そこで，開放や閉鎖を OpenFlow コントローラに通知するパケットと同じ宛先 IP アドレスおよび TCP ポート番号を持つパケットが，偶発的，あるいは悪意をもって意図的に作成され，ネットワークの開放や閉鎖が行わることを防ぐ必要がある．本システムにおいては，パケットの入力元となる OpenFlow スイッチの物理ポート番号と，表 1 の in_port フィールドの値を参照・比較し，接続制御サーバからのパケットであることを確認することで，ユーザやシステムの意図しないネットワークの開放や閉鎖を防止している．パケットの入力元となる OpenFlow スイッチの物理ポート番号は，パケットのヘッダに記述されているものではなく，OpenFlow が判別・取得するものであるため，偽装は困難であり安全性に問題はないと考えられる．

また，OpenFlow コントローラおよび OpenFlow スイッチ間の通信におけるセキュリティについては，OpenFlow Sec などによる取り組みが期待されている．

5.4 シングルサインオン認証への対応

シングルサインオン認証をネットワーク利用認証とその他 Web システムに対して利用しているネットワークとして，佐賀大学 [16] および広島大学 [17] のネットワークがあげられる．どちらも Shibboleth によるシングルサインオンを行っている．

本システムにおいても Shibboleth によるシングルサインオン認証に対応し，他のネットワーク上から，同一の IdP を使用する Web サービスに対して Shibboleth 認証を行った後，本システムを導入したネットワークに接続を切り替えて Web アクセスを行うと，ID とパスワードの入力処理を行うことなく，シームレスにログイン処理と監視ページの表示が行われた．また，同一の IdP により管理された他の Web システムをシームレスに利用できることを確認した．

5.5 柔軟なネットワーク構築

本研究による認証システムでは OpenFlow を用いており，柔軟なネットワーク構成が可能である．そのため，ネットワーク構成の変更を行う場合でも OpenFlow コントローラにより一括してネットワーク機器を制御し，ネットワーク構成の変更をすることができる．

本システムは L2 スイッチと同様に容易にネットワークへの導入が可能である．これにより，IP アドレスベースで認証を行い，L3 レベルでパケットフィルタリングによりネットワークの開放と閉鎖を行う認証システムのように，制御するブロックごとに別々のネットワークを作成し，管理・運用する必要がない．本システムを導入する際も，OpenFlow 対応のスイッチがあれば，容易に利用者認証を行うネットワークを構築し，運用することができる．

本システムを導入したネットワークにおいては，従来と同様に，認証を行うユーザ情報の管理や，スイッチの死活状態，スイッチのメモリ消費量，トラフィック処理量，各物理ポートの状態，その他のハードウェア故障の管理が必要となる．

また，本システムおよび OpenFlow を導入する場合，フローエントリ数の変化を含むフローテーブルの状態，フローエントリ単位あるいはユーザ単位でのトラフィック処理量，OpenFlow コントローラの死活状態および CPU・

メモリなどのリソース消費量，OpenFlow コントローラと OpenFlow スイッチの接続状態の管理が必要となる．

5.6 Opengate との比較

佐賀大学では，ネットワーク利用者認証システムとして，Opengate を開発・運用している．このシステムは，ネットワークの出口となるゲートウェイマシン上のファイアウォールにおいて通信の可否を判断している．そのため，ゲートウェイマシンに通信の可否の判断のために負荷が集中し，ボトルネックとなりやすい．また，L3 レベルでのアクセス制御を行うため，同一ネットワーク上や，同一セグメント上にある他のユーザ端末などに対しては，利用者認証を行わずに通信が可能である．これを防止するためにはスイッチなどの L2 機器に対してセパレート機能などの設定を行わなければならない．

本システムでは，OpenFlow を用いたことにより，通信の可否の判断を各スイッチのフローエントリで行うことができる．そのため，負荷が各スイッチに分散し，ゲートウェイにおけるボトルネックの解消を期待できる．また，同一ネットワーク上，あるいは同一セグメント上にある他のユーザ端末に対しても，利用者認証を行わなければ通信できない仕組みとなっている．

5.7 モバイルデバイスへの対応

本システムでは端末監視に Web ブラウザと Websocket を用いている．そのため，タブレットやスマートフォンの中には本システムによる認証後のセッションの維持が不可能なものがある．たとえば，iOS では他のアプリを使用するためにブラウザをバックグラウンドに回すと，ブラウザによる通信自体をスリープさせるため，Websocket によるセッションが切断される．

このようなモバイルデバイスに対しては，本システムによる認証とは別の認証手段を用意する必要がある．

5.8 複数スイッチへの接続

現状では，実験環境の制約から，単一のスイッチとコントローラによる接続にのみ対応している．実際にシステムを導入する場合には，複数のスイッチを接続して使用する．OpenFlow においては，各スイッチと OpenFlow コントローラを接続するには管理用のネットワークを別途構築する必要がある．

また，複数のスイッチを接続し，ネットワークを構築する場合，OpenFlow においては，トポロジー検出や，同一ネットワーク内のホストへの最短経路を計算し，フローエントリの設定などを行う必要がある．フローエントリの書き込みについても，現在は単一のスイッチのみに書き込んでいるが，複数のスイッチ ID を記録し，それらすべてに同時にフローエントリを書き込む必要がある．

これらについては，今後の課題である．

6. まとめと今後の課題

本研究では OpenFlow ネットワークに導入可能な，Shibboleth によるシングルサインオンに対応したネットワーク利用者認証システムを開発した．Shibboleth 認証を用いるため，学術認証フェデレーション（学認）により複数の組織にまたがる訪問者の利用認証も可能である．また，ユーザ端末には特別なソフトウェアやハードウェアをインストールすることなく，Web ブラウザのみで認証が可能であるため，利用者の視点から見ても手軽である．プログラマブルなネットワークを利用するため，利用者認証を行うネットワークを容易に構築できる．

SP への強制遷移を IP アドレスの書き換えによって行っているが，現在利用している OpenFlow v1.0 では，IPv6 を扱うことができない．今後は IPv6 が急速に普及していくことが考えられるため，OpenFlow v1.2 以降を採用し，IPv6 に対応させる必要がある．また，商用の OpenFlow スイッチを導入し，複数のスイッチ接続への対応や，接続検証および稼働試験を行うことが課題である．

参考文献

[1] Open Networking Foundation, available from ⟨https://www.opennetworking.org/ja/⟩ (accessed 2014-11-25).

[2] Shibboleth, available from ⟨http://shibboleth.internet2.edu/⟩ (accessed 2014-11-25).

[3] TECH.ASCII.jp: OpenFlow/SDN の波が来た，入手先 ⟨http://ascii.jp/elem/000/000/716/716414/⟩（参照 2014-11-25）.

[4] 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，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).

[5] OpenFlowSec.org, available from ⟨http://www.openflowsec.org/⟩ (accessed 2014-11-25).

[6] Yamashita, S., Tanaka, H., Hori, Y., Otani, M. and Watanabe, K.: Development of Network User Authentication System using OpenFlow, *The 5th International Workshop on Network Traffic Control, Analysis and Applications* (*NTCAA-2013*), pp.566–569 (2013).

[7] 山下翔平，田中久治，堀 良彰，大谷 誠，渡辺健次：OpenFlow と Shibboleth 認証を用いた利用者認証システムの開発，インターネットと運用技術シンポジウム 2013 論文集，pp.103–106 (2013).

[8] 学術認証フェデレーション 学認 GakuNin，入手先 ⟨http://www.gakunin.jp/⟩（参照 2014-11-25）.

[9] Trema Full-Stack OpenFlow Framework in Ruby and C (online), available from ⟨http://trema.github.io/trema/⟩ (accessed 2014-11-25).

[10] Trema による OpenFlow コントローラ routing switch Ruby 版，入手先 ⟨http://www1.bbiq.jp/~trema/⟩（参照 2014-11-25）.

[11] Node.js, available from ⟨http://nodejs.org/⟩ (accessed 2014-11-25).

[12] Socket.IO, available from ⟨http://socket.io/⟩ (accessed

2014-11-25).
[13] Pantou: OpenFlow 1.0 for OpenWRT (online), available from ⟨http://archive.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT⟩ (accessed 2014-11-25).
[14] OpenFlow in theBox (online), available from ⟨http://openflow.inthebox.info/Documents⟩ (accessed 2014-11-25).
[15] Opengate - A Network User Authentication System for Public and Mobile Terminals, available from ⟨http://www.cc.saga-u.ac.jp/opengate/⟩ (accessed 2014-11-25).
[16] 大谷 誠，江藤博文，渡辺健次，只木進一，渡辺義明：シングルサインオンに対応したネットワーク利用者認証システムの開発，情報処理学会論文誌，Vol.51, No.3, pp.1031–1039 (2010).
[17] 藤村喬寿，田島浩一，大東俊博，西村浩二，相原玲二：学術認証フェデレーションに基づくキャンパスネットワークの認証機構，研究報告インターネットと運用技術（IOT），Vol.2010-IOT-8, No.37, pp.1–6 (2010).

末永 光弘 （正会員）

2010 年 3 月佐賀大学理工学部知能情報システム学科卒業．2012 年 3 月同大学大学院工学系研究科博士前期課程知能情報システム学専攻修了．2015 年 3 月同大学院工学系研究科博士後期課程システム創成科学専攻修了．同年 4 月国立情報学研究所特任技術専門員．現在に至る．博士（工学）．電子情報通信学会正会員．

田中 久治 （正会員）

1990 年九州大学理学部物理学科卒業．同年文部技官佐賀大学理工学部．2012 年九州大学数理学府単位取得退学．現在に至る．計算機科学，情報ネットワーク，知的教育システムの研究に従事．教育システム情報学会会員．

大谷 誠 （正会員）

1998 年 3 月佐賀大学理工学部情報科学科卒業．2000 年 3 月同大学大学院工学系研究科博士前期課程情報科学専攻修了．2003 年 3 月同大学院工学系研究科博士後期課程システム生産科学専攻修了．同年 4 月佐賀大学海洋エネルギー研究センター COE 研究員．2004 年 12 月佐賀大学学術情報処理センター（現，総合情報基盤センター）講師．2009 年 4 月佐賀大学総合情報基盤センター准教授．2011 年 4 月国立情報学研究所学術ネットワーク研究開発センター外来研究員併任（2012 年 3 月まで）．現在に至る．インターネットの研究に従事．博士（工学）．

堀 良彰 （正会員）

1992 年九州工業大学情報工学部電子情報工学科卒業．1994 年同大学大学院情報工学研究科情報システム専攻修士課程修了．同年九州芸術工科大学芸術工学部助手．2004 年九州大学大学院システム情報科学研究院助教授．2013 年佐賀大学全学教育機構教授．現在に至る．ネットワークセキュリティ，コンピュータシステムセキュリティ，ネットワークアーキテクチャ，情報通信技術活用教育支援の研究に従事．博士（情報工学）．ACM，電子情報通信学会，IEEE 各会員．

岡崎 泰久 （正会員）

1988 年九州大学理学部物理学科卒業．1990 年同大学大学院理学研究科修士課程修了．1992 年佐賀大学理工学部助手．2005 年同助教授，2007 年同准教授，現在に至る．博士（工学）．コンピュータによる学習支援の研究に従事．電子情報通信学会，教育システム情報学会，日本教育工学会，人工知能学会各会員．

渡辺 健次 （正会員）

1987 年佐賀大学理工学部物理学科卒業．1989 年同大学大学院理工学研究科物理学専攻修士課程修了．同年同大学情報処理センター助手．1993 年和歌山大学経済学部産業工学科講師．1996 年同大学システム工学部情報通信システム学科講師．1998 年同助教授．1999 年佐賀大学理工学部知能情報システム学科助教授．2006 年同教授．2010 年同大学院工学系研究科知能情報システム学専攻教授．2012 年広島大学大学院教育学研究科技術・情報教育学講座教授．現在に至る．学習支援システム，インターネット応用，分散システム運用技術の研究に従事．博士（工学）．電子情報通信学会，人工知能学会，教育システム情報学会，日本教育工学会，日本産業技術教育学会，IEEE 各会員．