CROW: OpenFlowを用いた動的Webアクセス模倣システム

湯村 翼[1] 高野 祐輝[1] 安田 真悟[1] 宮地 利幸[1]

概要：インターネットへの接続が前提となるソフトウェアの検証を目的として，テストベッド環境においてインターネットへの到達性を維持しつつ，様々なIPアドレスからのWebアクセスを模倣するシステムCROWの設計と実装を行った．CROWは，クローラがWebサーバへアクセスする際に，中間にあるスイッチがパケットヘッダの送信元IPアドレスを書き換えることにより，模倣アクセスのトラフィックを生成する．パケットヘッダの書き換えは，環境内に設置されたOpenFlowスイッチにて実施する．これらの設計を元に，仮想マシンを用いてCROWの実装および環境構築を行った．

CROW: Dynamic Web Access Imitation System with OpenFlow

TSUBASA YUMURA[1] YUUKI TAKANO[1] SHINGO YASUDA[1] TOSHIYUKI MIYACHI[1]

1. はじめに

ソフトウェアの設計や実装において検証は不可欠である．テスト駆動開発 (Test-Driven Development:TDD) や継続的インテグレーション (Continuous Integration:CI) などのテストを中心としたソフトウェア開発手法も広まりを見せる．特にインターネットへの接続が前提となるデバイスやサービスでは，ネットワークを介したソフトウェアの検証が重要となる．

Internet of Things(IoT) 向けのデバイス開発では，ハードウェア製造時における手戻りがコストの増大を引き起こすため，ハードウェア製造前のソフトウェア検証が重要となる．IoTサービスでは大量のデバイスやセンサノード群がWebサーバとの通信を行うため，膨大な数のホストがWebサーバへアクセスすることとなる．

Webサーバで動作するソフトウェアの検証のために，様々なツールが存在する．しかし，IPアドレスを利用したサービスを検証する場合など，実際のインターネットを再現してテストを行う場合には不十分である．テストベッド環境内に模擬インターネットを構築する手法も研究されてきたが，模擬インターネットと実際のインターネットの両方との通信を同時に行う場合の取り扱いが難しい．

本研究では，テストベッド環境において，Webサーバからインターネットへの到達性を維持しつつ様々なIPアドレスからのWebアクセスを模倣するシステムCRawler for emulation Of dynamic Web accesses(CROW) の設計と実装を行う．

2. 関連研究

2.1 Webサービスにおけるテストツール

Webサーバで動作するソフトウェアの検証のために，目的に応じた様々なツールが開発されてきた．Webサービスの振る舞いを検証するには，RSpec[1] などのWebアプリケーション向けのテストフレームワークが用いられる．ブラウザ上のユーザインタフェース (UI) の検証にはSelenium[2] などのUIテストツールが用いられる．大量のアクセスがある場合の耐性を検証するためには，ApacheBench[3] などのベンチマークツールが用いられる．また，ネットワークの接続性などのサーバ自体の振る舞いを検証するServerspec[4] やInfrataster[5] といったツールもある．

しかしこれらのツールを使った検証は，単一ホストからアクセスを行うため，検証のために生成されるパケットの送信元IPアドレスは同一となる．そのため，IP Geolocation(IPアドレスによる位置推定) などの技術を使用した，

[1] 国立研究開発法人 情報通信研究機構
NICT

IP アドレスを利用したサービスを検証したい場合には不十分である．

2.2 テストベッドにおけるインターネット模倣

テストベッドを用いた模擬インターネットの構築は，過去の研究でも試みられた．Suzuki らは様々なテストベッド環境で汎用的なトポロジ設定を可能とする AnyBed[6] を開発した．Miwa らは，AnyBed を活用した XENebula[7] を開発し，10000 個の AS を仮想的につくりインターネットの模倣を行った．また，Miyachi らは，XENebula を利用した模倣インターネット上のトラフィックジェネレータ XBurner[8] を開発した．XBurner では，大規模なネットワークにおけるトラフィック生成に主眼が置かれていたが，より詳細なソフトウェア検証を行うためには，実際のユーザの挙動を模擬したより模倣度の高いトラフィック生成を行う必要がある．

2.3 IP アドレス変換

IP アドレスを変換する技術として，Network Address Transration(NAT) や Network Address and Port Translation(NAPT) が広く利用される．これらの機能は，Linux ではカーネルモジュールの netfilter[9] を利用し iptables[10] や firewalld[11] として実装される．これらを用いてインターネット上の様々なホストからのアクセスを模倣することが可能である．

また，OpenFlow[12] を使用してパケットヘッダを書き換えることもできる．OpenFlow は Software Defined Network(SDN) の代表的な実装例であり，実際にパケットの送受信を行う OpenFlow スイッチと，ルールに応じた制御を行う OpenFlow コントローラの組み合わせで構成される．

3. 提案手法

3.1 概要

本研究の提案手法の概要を図 1 に示す．本研究では，クローラが Web サーバへアクセスする際に，中間にあるスイッチがパケットヘッダの送信元 IP アドレスを書き換えることにより，模倣 Web アクセスのトラフィックを生成する．同様に，Web サーバからクローラへの応答を返す際のパケットヘッダもスイッチが書き換え，レスポンスをクローラへ返す．それ以外のパケットにはヘッダの書き換えは行わず，Web サーバやその他のノードからテストベッド環境外への通信を阻害しない．パケットヘッダの書き換えには，OpenFlow の技術を用いる．Web サーバとクローラの間に OpenFlow スイッチが配置されるトポロジを構成する．スイッチは単純な IP アドレス書き換え機能のみを有し，書き換えの規則はクローラから指示する．この構成により，クローラへの機能追加が容易となり，クローラの台数が増えた場合にもスイッチの処理がボトルネックになりにくい．

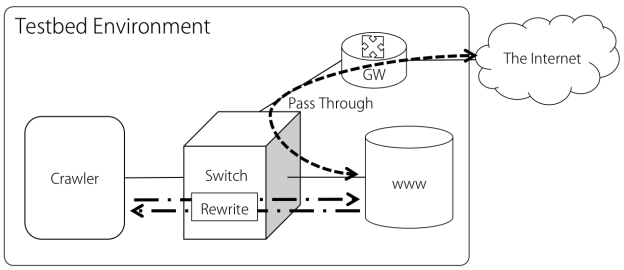

図 1 提案手法の概要

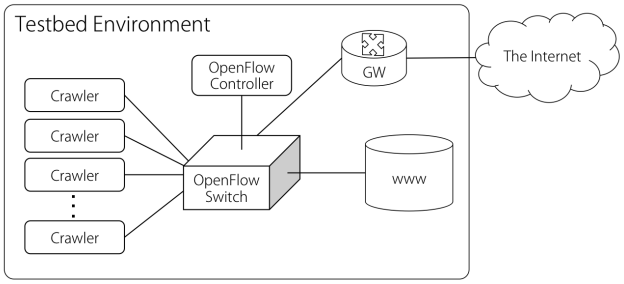

図 2 CROW のシステム構成

3.2 設計

テストベッド環境内に，Web サーバ，クローラ，ゲートウェイ，OpenFlow コントローラ，OpenFlow スイッチが存在し，各ノード間がスイッチ経由で接続しているスター型トポロジを構築する (図 2)．

各ノードは，ゲートウェイを通じて環境外へ通信を行うことができる．スイッチは，パケットを受信した際にヘッダの内容を確認し，条件に合致した場合にヘッダの書き換えを行う．パケットの書き換えルールのアルゴリズムを Algorithm 1 に示す．スイッチは，パケットの IP アドレスをどのように変換するかを示すテーブル (ipmaps) とソケット (IP アドレスとポート番号の組み合わせ) のマッピングを示すテーブル (socketmaps) を持つ．ipmaps には，変換対象となる IP アドレスと，変換する IP アドレスが対となって保存される．ipmaps のレコードの追加，更新や削除は，OpenFlow コントローラが公開する REST API 経由で行うことができる．

クローラから Web サーバへのパケットの場合には，ヘッダの送信元情報を書き換えてアクセス元の書き換えを行う．Web サーバからクローラへのレスポンスのパケットの場合には，ヘッダの送信先情報を書き換えてパケットをクローラへ届ける．

Algorithm 1 パケットの書き換えルール

```
SRCIP = source IP address of packet
DSTIP = destination IP address of packet
SRCSCT = source socket(IP address and port) of packet
DSTSCT = destination socket of packet
WSVSCT = Web server's socket
IPMAPS = table of mappings of IP address
SCTMAPS = table of mappings of IP address
if (DSTSCT==WSVSCT) and (SRCIP∈IPMAPS) then
    if SRCSCT∈SOCKETMAPS then
        rewrite source socket as the SOCKETMAPS
    else
        add new socket mapping to the SCTMAPS
        rewrite source socket as the SOCKETMAPS
    end if
else if (SRCSCT==WSVSCT) and (SRCSCT∈SCTMAPS) then
    rewrite destination socket as the SOCKETMAPS
end if
```

4. 実装

4.1 環境構築

CROW は，テストベッドでの使用を想定したシステムであり，OpenFlow スイッチなどの機材として実機を用意するのが最善の方法ではあるが，必ずしも実機を用意できるとは限らない．そこで本論文では，仮想マシンを用いて環境構築を実施する方法について述べる．

MacBook Pro の OS X 上で VirtualBox を用いて仮想マシンとして図 3 のとおり環境を構築した．構築には仮想環境構築ツール Vagrant[13] を用い，各ノードのネットワーク設定を Vagrantfile に記述した (図 4)．crow-switch に OpenFlow スイッチと OpenFlow コントローラを構築した．crow-switch に割り当てられた IP アドレスは，プライベートネットワークインタフェースを作成するために一時的に割り当てたものである．crow-switch とその他のノード間をそれぞれ異なるプライベートネットワークで接続することによりスター型トポロジを構築した．

環境構築に使用した主なソフトウェアの情報を表 1 にまとめた．

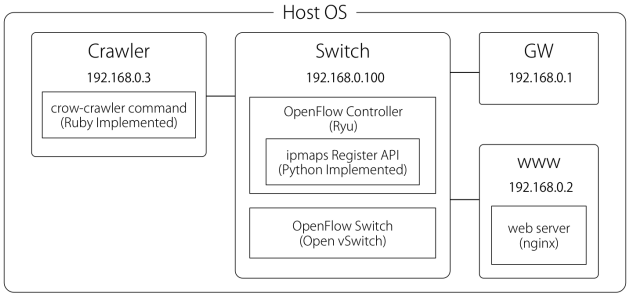

図 3　仮想環境として構築した CROW の構成

4.2 OpenFlow スイッチ

OpenFlow スイッチは，Ubuntu14.04 の Linux 仮想マシン上に Open vSwitch をインストールして構築した．Vagrantfile の設定により，スイッチノードは 3 つのインタフェースを有する．その 3 つのインタフェースをまとめたブリッジインタフェースを作成する．ブリッジインタフェースには，スイッチの IP アドレス（本稿では 192.168.0.100）を割り当てる．この IP アドレスは，後述する REST API へのアクセスのために使用する．

4.3 OpenFlow コントローラ

OpenFlow コントローラは，前節の OpenFlow スイッチと同一の Linux ノードに構築し，Python 製の SDN フレームワーク Ryu[14] を用いて実装した．受信したパケットに対して Algorithm 1 の通りにパケットヘッダを書き換えるよう実装した．書き換え手順を以下に説明する．

(1) ヘッダの送信先が Web サーバで，かつ送信元 IP アドレスが ipmaps に含まれている場合は，次の通り送信元 IP アドレスの書き換えを行う．(クローラから Web サーバへの模倣リクエストパケット)

- 送信元ソケットを socketmaps の origin と照合する
  - 存在する場合は，セッション確立済の TCP ポート番号を使用する．(既存コネクション)
  - 存在しない場合は，新たに送信元 TCP ポートを設ける．この際，使用中 TCP ポート番号をインクリメントする．(新規コネクション)

(2) ヘッダの送信元が Web サーバで，かつ送信先ソケットが socketmaps の rewrite に含まれている場合は，次の通り送信先 IP アドレスの書き換えを行う．(Web サーバからクローラへのレスポンスパケット)

- ヘッダの送信先情報を socketmaps の origin の情報に書き換える．

ipmaps(表 2) および socketmaps(表 3) は，Ryu で実行する Python スクリプト内で保持する．各テーブルは，origin

```
config.vm.define "crow-switch" do |node|
  node.vm.box = "ubuntu/trusty64"
  node.vm.hostname = "crow-switch"
  node.vm.network :private_network, ip: "192.168.0.11",  virtualbox__intnet: "gw"
  node.vm.network :private_network, ip: "192.168.0.12",  virtualbox__intnet: "www"
  node.vm.network :private_network, ip: "192.168.0.13",  virtualbox__intnet: "crawler"
end

config.vm.define "crow-gw" do |node|
  node.vm.box = "ubuntu/trusty64"
  node.vm.hostname = "crow-gw"
  node.vm.network :private_network, ip: "192.168.0.1",  virtualbox__intnet: "gw"
  node.vm.network :public_network, bridge: 'en0: Wi-Fi (AirPort)'
end

config.vm.define "crow-www" do |node|
  node.vm.box = "ubuntu/trusty64"
  node.vm.hostname = "crow-www"
  node.vm.network :private_network, ip: "192.168.0.2",  virtualbox__intnet: "www"
end

config.vm.define "crow-crawler" do |node|
  node.vm.box = "ubuntu/trusty64"
  node.vm.hostname = "crow-crawler"
  node.vm.network :private_network, ip: "192.168.0.3",  virtualbox__intnet: "crawler"
end
```

図 4　CROW の構成を記述した Vagrantfile

表 1　環境構築に使用した主なソフトウェアの情報

役割	ソフトウェア
ホスト OS	OS X 10.10.5
ゲスト OS	Ubuntu 14.04.2
ハイパーバイザ	VirtualBox 5.0.14
仮想マシン管理	Vagrant 1.8.1
OpenFlow スイッチ	Open vSwitch 2.0.2
SDN コントローラ	Ryu 3.30

表 2　ipmaps

origin	rewrite
192.168.0.3	11.22.33.44
192.168.0.4	11.22.33.45
192.168.0.5	11.22.33.46

表 3　socketmaps

origin	rewrite
192.168.0.3:49152	11.22.33.44:50000
192.168.0.3:49153	11.22.33.44:50001
192.168.0.3:49154	11.22.33.44:50002
192.168.0.3:49155	44.33.22.11:50003
192.168.0.3:49156	44.33.22.11:50004

を key，rewrite を value としたディクショナリ型変数として定義した．

ipmaps 書き換え用の REST API は，Python の wsgi モジュールを用いて Ryu の実行スクリプト上で実装した．新規登録およびレコードの書き換えを行う場合は，自身の IP アドレスを key，書き換え先 IP アドレスを value とした連想配列を JSON 形式 (図 5) で HTTP の PUT メソッドにて送信する．

なお，上記の他に，宛先 MAC アドレスと MAC アドレステーブルとの照合やパケットの送出などの通常の L2 スイッチとしての処理も実装している．

```
{'192.168.0.3' : '11.22.33.44'}
```

図 5　ipmaps 書き換え API に送信する JSON データ形式

4.4　クローラ

クローラは，以下の手順

- OpenFlow コントローラの REST API へアクセスし ipmaps を書き換える
- Web サーバの特定の URL への HTTP リクエストを行う

で OpenFlow コントローラと Web サーバへのアクセスを行う (図 6)．これらの一連のアクセスをまとめて実施するコマンド crow-crawler を Ruby で実装した．REST API へ通知する IP アドレスは，コマンド引数にて直接指定する他に，IP アドレスが記述されたファイルを読み込んで指定することもできる．複数行のファイルを読み込む場合は，そのうちひとつをランダムに選択する．空白が含まれる行は最初の空白までの文字列を IP アドレスとして読み込む仕様とし，Apache や nginx 等の Web サーバのアクセスログ (図 7) の読み込みを可能とした．Web へのアクセスは Ruby のヘッドレスブラウザライブラリ capybara-webkit を使用し，取得した html 内に記述された css や img などの情報も取得する実装とした．コマンドの説明を表 4 に示す．また，図 7 のアクセスログを読み込んでコマンドを実行した際に www に記録された模倣アクセスのログを図 8 に示す．

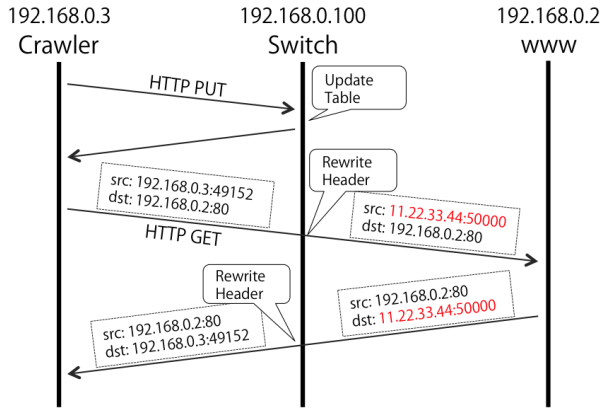

図 6　クローラのアクセスシーケンス図

表 4　crow-crawler コマンドの説明

Usage	crow-crawler [options] <Web server's url>
実行例	crow-crawler -i 11.22.33.44 http://192.168.0.2/ crow-crawler -f access.log http://192.168.0.2/

オプション	説明
-i [IPAddress]	ipmaps に書き込む IP アドレスを指定
-f [File]	ipmaps に書き込む IP アドレスをファイルから読み込む
なし	ipmaps に書き込む IP アドレスをランダムに指定

```
xxx.192.84.76 - - [02/Aug/2015:08:38:30 +0900] "GET /mfttimer.html HTTP/1.1" 200 823...
xxx.192.84.76 - - [02/Aug/2015:08:38:31 +0900] "GET /favicon.ico HTTP/1.1" 404 502 "...
xxx.176.17.88 - - [02/Aug/2015:09:23:15 +0900] "HEAD /mfttimer.html HTTP/1.1" 200 30...
xxx.236.87.125 - - [02/Aug/2015:10:07:11 +0900] "GET /mfttimer.html HTTP/1.1" 200 82...
xxx.176.128.153 - - [02/Aug/2015:10:51:41 +0900] "HEAD /mfttimer.html HTTP/1.1" 200 ...
xxx.249.245.155 - - [02/Aug/2015:10:56:30 +0900] "GET /mfttimer.html HTTP/1.1" 200 8...
xxx.212.144.171 - - [02/Aug/2015:11:16:51 +0900] "GET /mfttimer2.html HTTP/1.1" 200 ...
xxx.212.144.171 - - [02/Aug/2015:11:16:52 +0900] "GET /favicon.ico HTTP/1.1" 404 503...
xxx.236.26.103 - - [02/Aug/2015:11:16:52 +0900] "GET /mfttimer2.html HTTP/1.1" 206 1...
xxx.3.85.155 - - [02/Aug/2015:11:39:57 +0900] "GET /mfttimer2.html HTTP/1.1" 200 830...
```

図 7　Apache の access.log の例．ログはインターネットで公開している Web サーバで実際に取得した．IP アドレスの第 1 オクテットは伏字とした．また，スペースの都合から行の後半は省略した．

```
xxx.176.17.88 - - [10/May/2016:11:20:48 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozilla...
xxx.212.144.171 - - [10/May/2016:11:20:50 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.236.87.125 - - [10/May/2016:11:20:51 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozill...
xxx.212.144.171 - - [10/May/2016:11:20:52 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.3.85.155 - - [10/May/2016:11:20:53 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozilla/...
xxx.176.128.153 - - [10/May/2016:11:20:54 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.176.128.153 - - [10/May/2016:11:20:55 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.176.128.153 - - [10/May/2016:11:20:56 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.249.245.155 - - [10/May/2016:11:20:57 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.3.85.155 - - [10/May/2016:11:20:58 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozilla/...
xxx.3.85.155 - - [10/May/2016:11:20:59 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozilla/...
xxx.176.128.153 - - [10/May/2016:11:21:00 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.212.144.171 - - [10/May/2016:11:21:01 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.176.128.153 - - [10/May/2016:11:21:02 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.212.144.171 - - [10/May/2016:11:21:02 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozil...
xxx.3.85.155 - - [10/May/2016:11:21:03 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozilla/...
```

図 8　www の nginx の access.log に記録された模倣アクセスのログ．IP アドレスの第 1 オクテットは伏字とした．また，スペースの都合から行の後半は省略した．

5. 議論

5.1 netfilter との比較

CROW の IP アドレス変換機能には OpenFlow を採用し，Open vSwitch と Ryu を用いて構築した．この機能は，netfilter を利用したツールである iptables や firewalld を用いて実装することも可能である．しかし，これらのツールは高頻度での変更は想定されておらず，HTTP のリクエストとレスポンスの間に変換ルールを書き換えた場合などに，不整合が起こりうる．CROW では，IP アドレスのマッピングテーブル (ipmaps) とソケットのマッピングテーブル (socketmaps) を別にし，TCP コネクション確立後に IP アドレスの変換ルールを変えてもソケットのマッピングテーブルをそのまま保持することでこの問題を回避した．また，IP アドレス変換機能に加えて L2 スイッチの機能の実装が容易なこと，OpenFlow スイッチを実機に置き換え可能なことからも，OpenFlow の優位性があると考える．

5.2 トラフィック生成の設計

CROW の設計は，スイッチはパケットヘッダの IP アドレス変換の役割のみを担い，トラフィックの生成や制御はクローラが担うこととした．これにより，トラフィックに関する機能追加を行う場合，クローラの修正だけで対応することができる．

現段階では，クローラは指定された URL への HTTP リクエストを送信するだけの非常に単純な実装となっているが，シナリオ実行などの機能を追加してより高機能なクローラへと発展することも考えられる．また，実際に運用されている Web サーバの膨大なアクセスログを解析し，Web 閲覧者の属性や嗜好をもとに擬似的な Web アクセスを作り出すことも考えられる．このようにクローラの機能を発展させることにより，ユーザが Web にアクセスする際の状況に近い環境を提供することができる．

5.3 システムの構成

本論文では VirtualBox と Vagrant を用いたシステム構築手法を示した．仮想マシンを用いて OpenFlow スイッチを構築する手法は，実機の設定の手間を鑑みると実際のテストベッドにおいても有効性があると考えられる．OpenFlow を使用する仮想ネットワーク環境を構築するツール mininet[15] の使用も検討したが，ネットワークの詳細な設定を行うには不向きであるため不採用とした．

今後は，テストベッドを想定した環境構築システム Alfons[16] の活用も考え得る．また，本システムをツールとして公開する際には，テストベッド環境以外にも，Amazon Web Services といったクラウド環境や，Docker などのコンテナ型仮想環境で活用することも考慮し，利用者にとって有用となる配布形式を模索したい．

5.4 適用先の検討

大量のデバイスやセンサノード群を用いた IoT サービスでは，Web サーバへアクセスするホスト数が膨大となるため，CROW を用いたソフトウェア検証は有用になるのではないかと考える．また，Web で用いられる HTTP 以外に，メールで使用される SMTP や IMAP といったプロトコルや，今後 IoT サービスでの活用が増えるであろう MQTT など，様々な通信プロトコルに対応することにより適用先が増えると考えられる．

CROW は，サービスの検証以外にも適用できると考えられる．例えば，サイバー攻撃演習や CTF にて，Web サーバへの攻撃ツールやバックグラウンドトラフィックとしての活用が期待できる．また，Miwa ら [17] は，模擬インターネット上に特定のサーバを配置した仮想環境をつくり，独立したサンドボックス環境を構築した．マルウェアの動的解析においても，Command and Control(C&C) サーバとの通信を遮断せずに解析を行うために模擬インターネット構築が必要である．

5.5 評価計画

本研究では，同一ハイパーバイザ上に仮想マシンを立ち上げて CROW を構築し，その動作を確認した．今後は，実機のスイッチやノードを用いて大規模テストベッドにおけるシステム構築を行い，構築の簡便さや性能の評価を実施する予定である．本システムの有用性を評価するには，活用事例を通じて利点と欠点の洗い出しを行う必要があると考える．ネットワークを利用した実サービスの検証において，現状の課題と，本システムを利用することで解決できる課題を明確にしたい．

6. まとめ

本研究では，インターネットへの接続が前提となるソフトウェアの検証を目的として，テストベッド環境においてインターネットへの到達性を維持しつつ，様々な IP アドレスからの Web アクセスを模倣するシステム CROW の設計と実装を行った．CROW は，クローラが Web サーバへ

アクセスする際に，中間にあるスイッチがパケットヘッダの送信元 IP アドレスを書き換えることにより，模倣アクセスのトラフィックを生成する．パケットヘッダの書き換えは，環境内に設置された OpenFlow スイッチにて実施する．模倣する IP アドレスは，REST API によりクローラから指示することができる．これらの設計を元に，仮想マシンを用いて CROW の実装および環境構築を行った．今後は，実機のスイッチやノードを用いて大規模テストベッドにおけるシステム構築を行い，構築の簡便さや性能の評価を実施する予定である．

参考文献

[1] RSpec: Behaviour Driven Development for Ruby: http://rspec.info/.

[2] Selenium - Web Browser Automation: http://www.seleniumhq.org/.

[3] ab - Apache HTTP server benchmarking tool - Apache HTTP Server Version 2.4: https://httpd.apache.org/docs/2.4/programs/ab.html.

[4] 宮下剛輔，栗林健太郎，松本亮介：serverspec: 宣言的記述でサーバの状態をテスト可能な汎用性の高いテストフレームワーク，研究報告インターネットと運用技術（IOT），Vol. 2014, No. 15, pp. 1–6 (2014).

[5] Infrataster: http://infrataster.net/.

[6] Suzuki, M., Hazeyama, H., Miyamoto, D., Miwa, S. and Kadobayashi, Y.: Expediting experiments across testbeds with anybed: A testbed-independent topology configuration system and its tool set, *IEICE Transactions on Information and Systems*, Vol. E92-D, No. 10, pp. 1877–1887 (online), DOI: 10.1587/transinf.E92.D.1877 (2009).

[7] Miwa, S., Suzuki, M., Hazeyama, H., Uda, S., Miyachi, T., Kadobayashi, Y. and Shinoda, Y.: Experiences in emulating 10K AS topology with massive VM multiplexing, *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures - VISA '09*, p. 97 (online), DOI: 10.1145/1592648.1592664 (2009).

[8] Miyachi, T. and Miwa, S.: Design and Implementation of XBurner, *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SimuTools '13, pp. 227–234 (2013).

[9] netfilter/iptables project homepage - The netfilter.org project: http://www.netfilter.org/.

[10] netfilter/iptables project homepage - The netfilter.org "iptables" project: http://www.netfilter.org/projects/iptables/.

[11] firewalld official website: http://www.firewalld.org/.

[12] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J.: OpenFlow, *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, p. 69 (online), DOI: 10.1145/1355734.1355746 (2008).

[13] Vagrant by HashiCorp: https://www.vagrantup.com/.

[14] Ryu SDN Framework: https://osrg.github.io/ryu/.

[15] Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet: http://mininet.org/.

[16] 安田 真悟，三浦 良介，太田 悟史，高野 祐輝，宮地利幸：ビルディングブロック型模擬環境構築システム，インターネットコンファレンス 2015 論文集，pp. 69–78 (2015).

[17] Miwa, S., Miyachi, T., Eto, M., Yoshizumi, M. and Shinoda, Y.: Design and Implementation of an Isolated Sandbox with Mimetic Internet Used to Analyze Malwares., *DETER Community Workshop on Cyber Security Experimentation and Test* (2007).