

ZESPÓŁ REDAKCYJNY

prof. ndz. dr hab. inż. Zbigniew Piotrowski – redaktor naczelny

Bożena Lachowicz – sekretarz redakcji

prof. dr hab. inż. Jerzy Klamka – Senior Redaktor

Stał współpracownik: mgr inż. Cezary Rudnicki,

Redaktorzy tematyczni: mgr inż. Wiesław Jabłoński, mgr inż. Krzysztof Kowalski, dr inż. Marcin Wesołowski

Adres redakcji:

ul. Czackiego 3/5, lok. 108 00-043 Warszawa,
tel./fax (22) 827 38 79; tel.: (22) 826 65 64,
e-mail: elektronika@red.pl.pl, www.elektronika.orf.pl

Kolportaż: ul. Ku Wiśle 7, 00-716 Warszawa,
tel. (22) 840 35 89; tel./fax: (22) 840 59 49, (22) 891 13 74,
e-mail: prenumerata@sigma-not.pl

Zamówienia na reklamę przyjmuje redakcja lub Dział Reklamy i Marketingu,
ul. Ratuszowa 11, 00-950 Warszawa, skr. 1004,
tel./fax (22) 827 43 65, e-mail: reklama@sigma-not.pl

Ministerstwo Nauki i Szkolnictwa Wyższego za opublikowane artykuły przyznaje 8 punktów.

ELEKTRONIKA jest wydawana przy współpracy Komitetu Elektroniki i Telekomunikacji Polskiej Akademii Nauk



 Redakcja współpracuje z Polską Sekcją IEEE
IEEE

Artykuły publikowane w Elektronice są rejestrowane w bazie danych CrossRef. Każdemu artykule przypisany jest numer identyfikacyjny DOI (Digital Object Identifier). Autorzy artykułów publikowanych w Elektronice są proszeni do stosowania numerów DOI w wyciągach literatury, wszędzie tam gdzie jest to możliwe

Publikowane artykuły naukowe są recenzowane przez samodzielnych pracowników nauki

Redakcja nie ponosi odpowiedzialności za treść ogłoszeń.
Zastrzega sobie prawo do skracania i adiustacji nadesłanych materiałów.

Indeks 35722 Nakład do 2000 egz

Skład: Studio DTP SIGMA-NOT

Druk: Drukarnia Wydawnictwa SIGMA-NOT

Wydawnictwo SIGMA-NOT Sp. z o.o.



00-950 Warszawa, skr. poczt. 1004
ul. Ratuszowa 11, tel. 22 818 09 18
e-mail: sekretariat@sigma-not.pl
Internet: <http://www.sigma-not.pl>

Wersja papierowa ELEKTRONIKI jest wersją pierwotną.

W ocenie Index Copernicus ELEKTRONIKA uzyskała – 64,95 pkt.

Ponadto ELEKTRONIKA jest indeksowana w następujących bazach danych: BazTech, INSPEC oraz OCLC – WorldCat: <http://www.worldcat.org/title/elektronika/oclc/10475423>

- ❖ MATERIAŁY ❖ KONSTRUKCJE ❖ UKŁADY
- ❖ SYSTEMY ❖ MIKROELEKTRONIKA ❖ OPTOELEKTRONIKA
- ❖ FOTOELEKTRONIKA ❖ ELEKTRONIKA MIKROFALOWA
- ❖ MECHATRONIKA ❖ ENERGOELEKTRONIKA ❖ INFORMATYKA

SPIS TREŚCI ❖ CONTENTS

DETEKCJA FAZ ODDECHU W REJESTRACJACH STETOSKOPOWYCH (Detection of breath phases in stethoscope recordings)	4
D. MAŁY, A. DOBROWOLSKI	
ZASTOSOWANIE SYMULATORA ZDARZEŃ DYSKRETNYCH DO STEROWANIA SIECIĄ SDN (The use of a discrete event simulator to control the Software Defined Network)	8
J. DOŁOWSKI, K. ŚWIDRAK	
TECHNOLOGIE ELEKTRONICZNE PODWÓJNEGO ZASTOSOWANIA WYKORZYSTANIE TECHNIKI UCZENIA ZE WZMOCNIENIEM W SIECIACH SDN (The use of reinforced learning techniques in SDN networks)	12
P. SPICZYŃSKI, J. KRYGIER	
SPÓJNE ŚRODOWISKO PRZETWARZANIA W CHMURZE OPARTE NA STRUKTURZE FIZYCZNEJ I APLIKACYJNEJ (A consistent environment of the processing in the cloud based on physical and application structure)	17
M. STRÓŻ, D. LASKOWSKI	
GitLab – NARZĘDZIE ZESPOŁOWEJ PRACY PROGRAMISTÓW (GitLab – a tool for team work of programmers)	25
P. KACZMAREK, M. WOJTCZAK	
PRZEGŁĄD ODBIORNIKÓW SZEROKOPASMOWYCH I REJESTRATORÓW SYGNAŁU RADIOSŁU (Overview of wideband receivers and radio signal recorders)	29
M. WOJTCZAK	
UKŁAD DO ZDALNEGO STEROWANIA TEMPERATURĄ PRZY UŻYCIU APLIKACJI ANDROID OS (System for remote control of temperature with use of Android OS application)	32
M. MULTAN, A. PONIECKI	
UKŁAD STEROWANIA PRZYDOMOWĄ ELEKTROWNIAJĄ SŁONEczną (The control system for backyard of solar power plant)	36
A. KARAŚ, A. TYPIAK	
TECHNOLOGIE ELEKTRONICZNE PODWÓJNEGO ZASTOSOWANIA ANALIZA SKUTECZNOŚCI ZAKŁÓCEŃ EMISJI FH (Analysis of the effectiveness of FH emission jamming)	40
D. JENDRUSZCZAK, J. BUGAJ	
PROJEKT SYSTEMU ENERGY HARVESTING PRACUJĄCEGO W PAŚMIE 2,4 GHz (Energy Harvesting system project operating in the 2.4 GHz band)	46
D. MICHALCZYK, J. BUGAJ	
MINIATUROWY UKŁAD ZASILANIA BEZPRZEWODOWEGO QI NA PRZYKŁADZIE ROBOTU EDUKACYJNEGO „EDEK” (A miniature Qi wireless power supply system on the example of Edek's educational robot)	52
G. JĘDRZEJEWSKI	
WŁAŚCIWOŚCI ZASOBNIKÓW SAMOLOTOWYCH W ASPEKcie WALKI RADIOELEKTRONICZNEJ (Properties of aircraft pod's in the aspect of the electronic warfare)	56
P. JĘDRUSIK, J. PIETRASINKI	
VARIOUS APPROACHES TO MODELLING OF THE MASS USING THE SIZE OF THE CLASS IN THE CENTROID BASED CLASSIFICATION (Różne podejście do modelowania masy przy użyciu wielkości klas w klasyfikacji opartej na centroidach)	62
Ł. RYBAK, J. DUDCZYK	
RZECZYWISTOŚĆ ROZSZERZONA NARZĘDZIEM PRZEMYSŁU 4.0 – ROZWIĄZANIA DLA BRANŻY MOTORYZACYJNEJ (Augmented reality as a tool of Industry 4.0 – solutions for automotive industry)	66
H. SKÓRSKA	

ZASTOSOWANIE SYMULATORA ZDARZEŃ DYSKRETYCH DO STEROWANIA SIECIĄ SDN

The use of a discrete event simulator to control the Software Defined Network

Programowalne Sieci Komputerowe (ang. *Software Defined Networks* – SDN) są efektem rosnącego obecnie trendu na przenoszenie logiki działania i zarządzania wielu dziedzin życia na formę programową – na przykład radio programowalne, internet rzeczy itp. Takie podejście otwiera bardzo duże możliwości.

Architektura i zamysł SDN polega na oddzieleniu warstwy sterowania i zarządzania od warstwy infrastruktury (danych). Model warstwowy (rys. 1) opiera się na trzech zasadniczych warstwach [3]:

- Infrastruktury (ang. *Infrastructure*) odpowiadającej najniższej warstwie – sieci, w której pomiędzy przełącznikami SDN realizowany jest przepływ ramek sieciowych;
- Sterowania (ang. *Control*) – sterownika, który zarządza przełącznikami sterując pracą całej sieci zgodnie z usługą zarządzania siecią określona przez aplikację (sterownik jest swoistym tłumaczem pomiędzy siecią a logiką aplikacji);
- Aplikacji (ang. *Application*) – programu, który odpowiednio zarządza pracą sterownika, a przez to logiką całej sieci, określając odpowiednie przepływy i operacje wykonywane przez warstwę najwyższą.



Rys. 1. Model warstwowy SDN
Fig. 1. Layer model of the SDN

Przełączniki SDN operują na tak zwanych tablicach przepływów. Każda ramka, którą otrzyma przełącznik przechodzi przez ścieżkę przetwarzania (ang. *pipeline*), podczas której poszczególne pola nagłówków są porównywane z wpisami w tablicach przepływu. Jeżeli żaden wpis nie pasuje do ramki – zostaje ona przesłana do sterownika za pomocą bezpiecznego kanału (ang. *secure channel*) [3].

mjr dr inż. Jerzy Dołowski,
sierż. pchor. mgr inż. Krzysztof Świdrak

Wojskowa Akademia Techniczna, Wydział Elektroniki,
Instytut Telekomunikacji

STRESZCZENIE

Artykuł opisuje nietypowe zastosowanie symulatora zdarzeń dyskretnych OMNeT++ do sterowania programowalnymi sieciami komputerowymi (SDN). W środowisku OMNeT++ przygotowano sterownik sieci programowalnych obsługujący protokół Open Flow w wersji 1.0.0. Sterownik może wykorzystywać rzeczywisty interfejs sieciowy. Ponadto sterownik wyposażono w odpowiednie liczniki wiadomości i ich rozmiarów. Napisano także aplikację wspierającą zarządzanie ARP w sieci SDN. Przygotowano środowisko do przeprowadzenia badań, oparte o emulator Mininet i bazę danych MySQL oraz omówiono zebrane wyniki.

SŁOWA KLUCZOWE: OMNeT++, Mininet, Programowalne Sieci Komputerowe, SDN, Sterownik, OpenFlow

ABSTRACT

The article describes unusual usage of discret event simulator OMNeT++ to control software defined networks (SDN). An SDN Controller that supports OpenFlow version 1.0.0 has been prepared in OMNeT++. The Controller may use a real network's interface. Moreover, the Controller has been equipped in counters of messages and their sizes. An application that supports maintaining of ARP has been prepared. The test environment based on Mininet and MySQL instances has been created. The results have been presented and discussed.

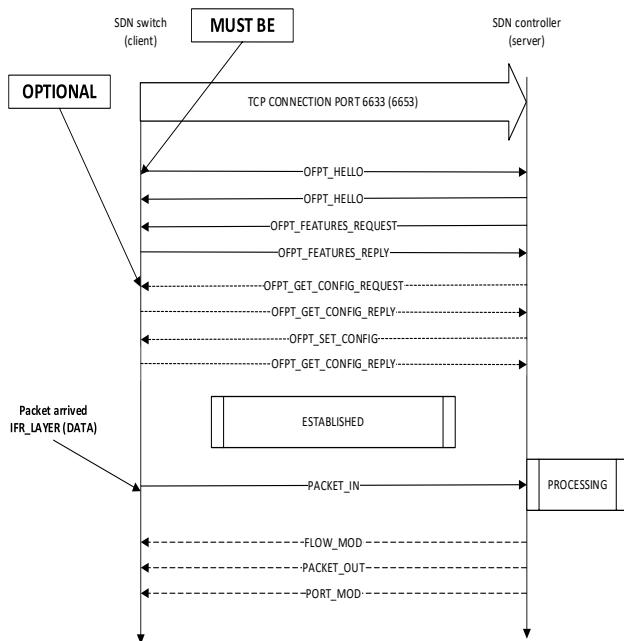
KEYWORDS: OMNeT++, Mininet, Software Defined Network, SDN, Controller, OpenFlow

Sterownik SDN łączy się z przełącznikami za pomocą interfejsu południowego (ang. *south interface*). Po nawiązaniu sesji OpenFlow otrzymuje od przełączników wiadomości o stanie sieci, w tym także ramki, które nie pasowały do żadnego wpisu w tablicy przepływów [3].

Zarządzaniem przez sterownik zdarzeniami z sieci (np. przetwarzaniem ramek) logicznie kieruje aplikacja, która wprowadza reguły zarządzania siecią, wykorzystując przy tym API sterownika wystawione jako interfejs północny (ang. *north interface*). API może być wystawione jako dziedziczone funkcje oraz gotowe rozwiązania, które udostępniane są jako API REST.

PROTOKÓŁ OpenFlow

Protokołem, który pośredniczy wymianie informacji zarządzających siecią pomiędzy warstwą infrastruktury oraz warstwy sterowania (sterownikiem a przełącznikami) jest określony przez ONF binarny protokół wolnego przepływu (ang. *OpenFlow Protocol OFP*). Protokół ten opiera się na komunikacji typu klient-serwer (rys. 2).



Rys. 2. Podstawowa sesja OpenFlow

Fig. 2. Basic OpenFlow session

- W OpenFlow wyróżnić można trzy rodzaje wiadomości [3]:
- Sterownika do przełącznika (ang. *controller to switch*) – zawierające polecenia zarządzania przełącznikiem (ustawienie przepływu, stanu portu itp.);
 - Asynchroniczne (ang. *asynchronous*) – zawierające informację o stanie sieci lub wykonanym zadaniu (przesyłanie statystyk, informacji o stanie portów itp.);
 - Symetryczne (ang. *symetric*) – wysypane obustronnie w celu podtrzymywania lub weryfikacji stanu połączenia pomiędzy sterownikiem i przełącznikiem (inicjacja połączenia, podtrzymanie połączenia itp.).

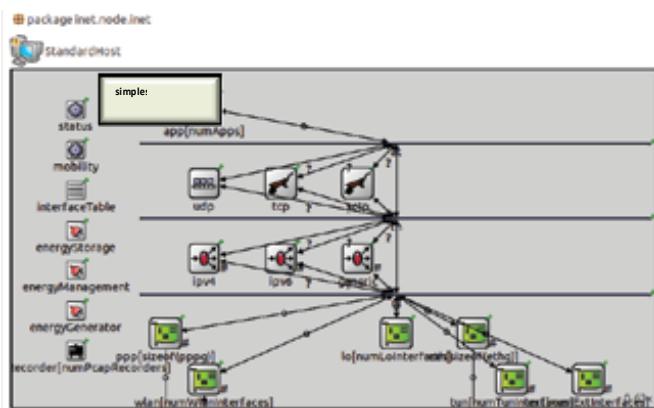
KONCEPCJA STEROWNIKA W OMNeT++

OMNeT++ jest symulatorem zdarzeń dyskretnych. Działanie modułów stworzonych w tym środowisku opiera się na reakcjach wywoływanych przez przychodzące zdarzenia. Zdarzenia reprezentowane są jako wiadomości *cMessage* i ich pochodne (między innymi *cPacket*) [4].

Po przeanalizowaniu istniejących sterowników sieci programowalnych stwierdzono, iż OMNeT++ jest odpowiednim środowiskiem symulacyjnym umożliwiającym prowadzenie różnorakich badań sieci SDN. OMNeT++ umożliwia wygodne zbieranie różnego rodzaju miar statystycznych (np. ilości pakietów, rozmiary pakietów, czasy opóźnień), daje duże możliwości implementacji różnych niekonwencjonalnych rozwiązań wykorzystując programowanie obiektowe w języku C++, NED, XML. Dodatkowo biblioteka INET Framework w wersji 4.0.0 umożliwia korzystanie z rzeczywistych inter-

fejsów sieciowych, przez co istnieje możliwość połączenia środowiska symulatora zdarzeń dyskretnych z rzeczywistymi sieciami komputerowymi [6].

Sterownik SDN został zaimplementowany jako aplikacja TCP dziedzicząca po *cSimpleModule* [5] oraz *ILifecycle*. Jako obiekt strukturalny NED jest ona określona jako aplikacja TCP, korzystająca z gniazda aplikacji standardowego hosta (ang. *Standard Host*, rys. 3) [1]. Bazując na znajomości działania sterowników wspomnianych wcześniej i obiektów OMNeT++ moduł sterownika udostępnia szereg funkcji, które stanowią pewnego rodzaju obsługę przerwań wywołanych wiadomościami. Opiera się to na jednej z podstawowych metod obiektów OMNeT++ – *handleMessage(cMessage* msg)* [4].



Rys. 3. Sterownik SDN jako aplikacja TCP standardowego hosta

Fig. 3. SDN Controller as TCP application of StandardHost

W związku z koniecznością komunikacji z rzeczywistymi urządzeniami sieciowymi (przełączniki SDN w Mininet), działanie środowiska OMNeT++ należało wesprzeć modułem *inet::RealTimeScheduler*, który odpowiada za odpowiednie kolejkowanie zdarzeń w minimalnych interwałach symulujących upływ czasu rzeczywistego i umożliwiający reakcję na wiadomości pochodzące z sieci [6]. Komunikację z rzeczywistym interfejsem osiągnięto poprzez połączenie się środowiska OMNeT++ przez moduł *externalInterface (extInt)* do rzeczywistej karty sieciowej, na której utworzony został RAW_SOCKET [1]. Nadmiarowość przechwytywania pakietów ograniczono dzięki zastosowaniu filtracji za pomocą popularnej biblioteki PCAP [1]. Stworzono klasy odpowiadające nagłówkom wiadomości OFP oraz zapewniono ich serializację i deserializację w celu przekształceń z wymiaru obiektowego na wymiar strumienia bitowego. Moduł dodatkowo wyposażono w automatyzację uruchamiania sieci dedykowanych dla konkretnej symulacji (napisanych w Pythonie sieci Mininet), uruchamianie graficznego interfejsu Wireshark oraz logowanie odpowiednich informacji o zdarzeniach do pliku. Moduł sterownika wyposażono także w odpowiednie liczniki wiadomości OFP oraz metryki rejestrujące wiel-

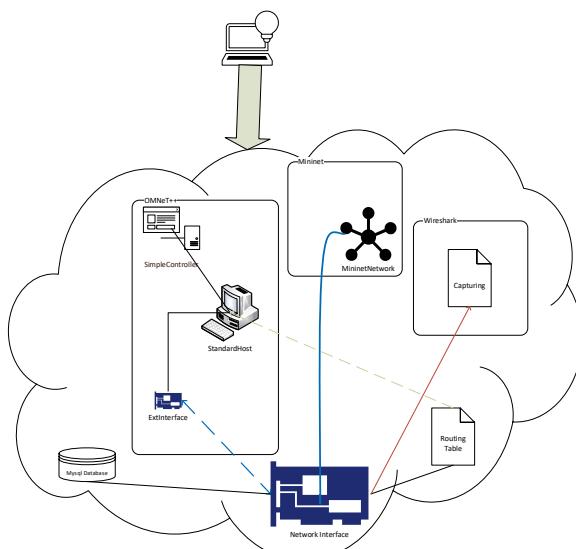
**PROGRAMOWALNE SIECI
KOMPUTEROWE SĄ
EFEKTEM ROSNĄCEGO
OBECNIE TRENDU
NA PRZENOSZENIE
LOGIKI DZIAŁANIA
I ZARZĄDZANIA WIELU
DZIEDZIN ŻYCIA NA
FORMĘ PROGRAMOWĄ.**

kości przetwarzanych w danym czasie przez sterownik wiadomości OFP.

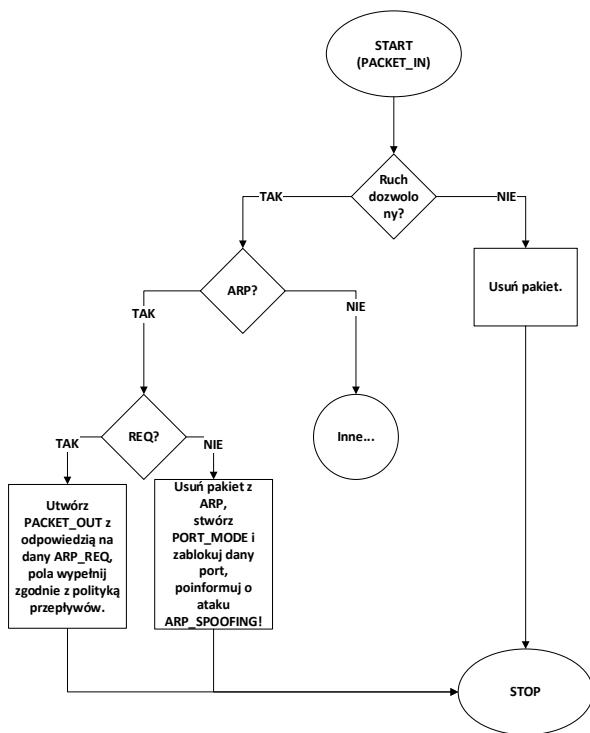
APLIKACJA WSPIERAJĄCA ZARZĄDZANIE ARP

W celu zaprezentowania funkcjonalności działania sterownika przygotowano odpowiednie środowisko weryfikacyjne (rys. 4). Uwzględniając trend w SZ RP na rozwój w dziedzinie Bezpieczeństwa Cyberprzestrzeni zaimplementowano aplikację, która steruje ruchem sieci w oparciu o następujące założenia:

- Zdefiniowana jest polityka ruchu przez osobę zarządzającą siecią.



Rys. 4. Środowisko do weryfikacji działania sterownika
Fig. 4. Testing environment of the Controller



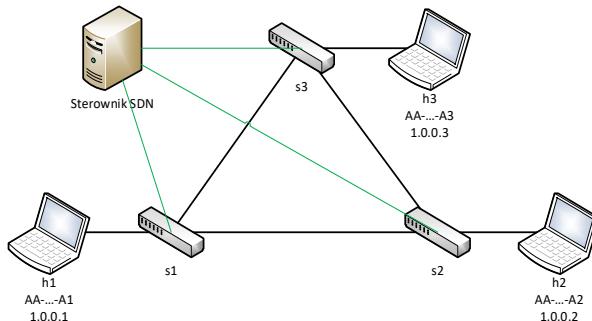
Rys. 5. Algorytm działania aplikacji
Fig. 5. An algorithm of the application

- Zapewniona jest interakcja osoby zarządzającej siecią ze sterownikiem poprzez dokonywanie zmian w polityce ruchu.
- Polityka ruchu opiera się na bazie danych typu MySQL (aplikacja łączy się z bazą za pomocą bibliotek mysql_connection i cppconn).
- Aplikacja pełni rolę niskopoziomowej zapory ogólnosieciowej (tj. pracuje w warstwie MAC urządzeń sieciowych z wykluczeniem hostów) zezwalając na ruch tylko i wyłącznie uwzględniony w polityce ruchu, samodzielnie odpowiada na zapytania ARP zgodnie z polityką ruchu oraz zabezpiecza sieć przed atakami arp spoofing.
- Opiera swoje działanie na algorytmie przedstawionym na grafie na rys. 5.

BADANIA WERYFIKACYJNE

Badania, których celem było sprawdzenie poprawności działania sterownika i opartej na nim aplikacji zarządzania ARP, przeprowadzone na sieci złożonej z trzech przełączników o topologii pierścienia. Do każdego z przełączników podłączono po jednym hoście (rys. 6).

Procedura badania była następująca. W pierwszym etapie dozwolone było połączenie tylko pomiędzy dwoma



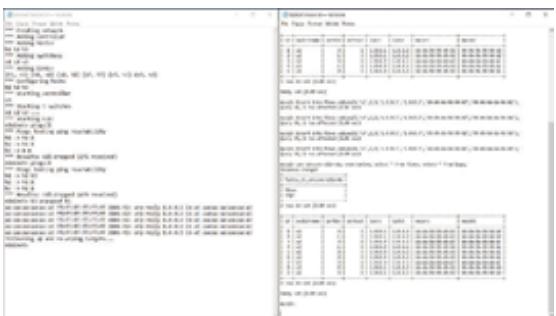
Rys. 6. Sieć weryfikacyjna
Fig. 6. Verification network

hostami: h1 i h2. Przeprowadzono sprawdzenie połączenia za pomocą standardowego polecenia Mininet – pingall [2]. Następnie dokonano wpisu w bazie danych w celu umożliwienia łączności pomiędzy hostami h1 i h3. Ponownie wykonano pingall. Po czym dokonano próby ataku arp spoofing przez hosta h3 (polecenie h3 arp spoof h1) [2]. Założono przy tym, że numery hostów odpowiadają ostatniej cyfrze adresu IP oraz MAC.

ANALIZA WYNIKÓW

Wyniki pokazały, że za pomocą sterownika można sterować ruchem w sieci Mininet poprzez edytowanie wpisów w bazie danych (rys. 7).

Model umożliwia zbieranie liczników wymienianych wiadomości (rys. 8), które informują o ilości otrzymanych i wysłanych przez sterownik wiadomości OFP, przetworzonych po otrzymaniu z OFPT_PACKET_IN IP obu wersji i ARP. OMNeT++ umożliwia również zobrazowanie wielkości rozmiarów pakietów przetwarzanych czasie przez sterownik (rys. 9) [4].



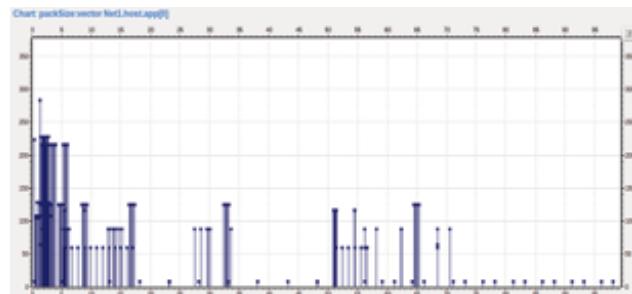
Rys. 7. Logi z konsoli Mininet i mysql

Fig. 7. The logs of the Mininet's and mysql's console

Experiment	Module	Name	Value
ELEKTRONIK	Net1.host.app[0]	packSizeSum	28390.0
ELEKTRONIK	Net1.host.app[0]	Received PACKET_IN	215.0
ELEKTRONIK	Net1.host.app[0]	Received IPv4 via PACKET_IN payload	162.0
ELEKTRONIK	Net1.host.app[0]	Received ECHO_REQ	52.0
ELEKTRONIK	Net1.host.app[0]	Sent ECHO REP	52.0
ELEKTRONIK	Net1.host.app[0]	Received ARP Requests via PACKET_IN payload	24.0
ELEKTRONIK	Net1.host.app[0]	Received PORT_STATUS REP	10.0
ELEKTRONIK	Net1.host.app[0]	Sent ARP Replies via PACKET_OUT payload	6.0
ELEKTRONIK	Net1.host.app[0]	Sent HELLO	3.0
ELEKTRONIK	Net1.host.app[0]	Received HELLO	3.0
ELEKTRONIK	Net1.host.app[0]	Received FEATURES REP	3.0
ELEKTRONIK	Net1.host.app[0]	Sent FEATURES REP	3.0
ELEKTRONIK	Net1.host.app[0]	Received ARP Replies via PACKET_IN payload	1.0
ELEKTRONIK	Net1.host.app[0]	Sent STATISTICS REP	0.0

Rys. 8. Liczniki sterownika

Fig. 8. The controller's counters



Rys. 9. Graficzna reprezentacja wektora rozmiarów pakietów

Fig. 9. Graphical visualization of packet's size vector

Dodatkowo sterownik umożliwił stworzenie aplikacji, która, zabezpiecza sieć działając jak niskopoziomowy firewall, zezwalając tylko na dozwolony ruch oraz blokując próby ataków *arp spoofing* poprzez odłączanie portu, na którym wykryto atak i informowaniu o tym fakcie w logach (rys. 10). Zabezpieczeniem jest objęta cała sieć przełączników.

WNIOSKI

Zaproponowana koncepcja umożliwiła stworzenie z symulatora zdarzeń dyskretnych narzędzia do prowadzenia badań nad sieciami programowalnymi SDN oraz implementowania aplikacji, które mogą wprowadzać nowe, nieszablonowe zachowania sieci.

Sterownik, dzięki temu, że został zaimplementowany w języku C++ wystawia interfejs API w postaci funkcji publicznych dla aplikacji podobnie jak inne stworzone już sterowniki. Umożliwia on także odnoszenie się do zewnętrznych instancji takich jak bazy danych czy chmury

```
host_ErrorLog.txt — Notatnik
Plik Edycja Format Widok Pomoc
Project created by @swidru95@ at Military University Of Technology.
Log created at Thu Jun 20 15:33:54 2019
SDN Controller is listening: 10.1.1.1:6633
ER> Thu Jun 20 15:34:03 2019
Switch: s1 src: 1.0.0.1 (AA-AA-AA-AA-AA-A1) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:04 2019
Switch: s1 src: 1.0.0.1 (AA-AA-AA-AA-AA-A1) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:05 2019
Switch: s1 src: 1.0.0.1 (AA-AA-AA-AA-AA-A1) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:06 2019
Switch: s2 src: 1.0.0.2 (AA-AA-AA-AA-AA-A2) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:07 2019
Switch: s2 src: 1.0.0.2 (AA-AA-AA-AA-AA-A2) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:08 2019
Switch: s2 src: 1.0.0.2 (AA-AA-AA-AA-AA-A2) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:10 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.1
ER> Thu Jun 20 15:34:11 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.1
ER> Thu Jun 20 15:34:12 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.1
ER> Thu Jun 20 15:34:13 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.2
ER> Thu Jun 20 15:34:14 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.2
ER> Thu Jun 20 15:34:15 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.2
ER> Thu Jun 20 15:34:16 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.2
ER> Thu Jun 20 15:34:17 2019
Switch: s2 src: 1.0.0.2 (AA-AA-AA-AA-AA-A2) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:18 2019
Switch: s2 src: 1.0.0.2 (AA-AA-AA-AA-AA-A2) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:19 2019
Switch: s2 src: 1.0.0.2 (AA-AA-AA-AA-AA-A2) trying connect with: 1.0.0.3
ER> Thu Jun 20 15:34:20 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.2
ER> Thu Jun 20 15:34:21 2019
Switch: s3 src: 1.0.0.3 (AA-AA-AA-AA-AA-A3) trying connect with: 1.0.0.2
WARNING!> Thu Jun 20 15:35:06 2019
Switch: s3 src: 1.0.0.1 (AA-AA-AA-AA-AA-A3) Possible ARP_SPOOFING threat
(arp_reply).
Target: <unspec>
```

Rys. 10. Logi sterownika

Fig. 10. The logs of Controller

obliczeniowe, które zapewniają efektywne przetwarzanie danych i odciążają obliczeniowo zasoby sterownika.

Wsparcie środowiska OMNeT++ umożliwia zbieranie wielu przydatnych w badaniach statystyk. Wspiera również posługiwanie się modelami stochastycznymi takimi jak rozkłady statystyczne.

Wykorzystanie biblioteki INET Framework ułatwia proces implementacji, dzięki bogatemu zbiorowi przeróżnych technologii i protokołów sieciowych używanych współcześnie. ♦

LITERATURA

- [1] INET Framework for OMNeT++, “Developer’s Guide”, <https://inet.omnetpp.org/docs/developers-guide/> (dostęp online 20.06.2019).
- [2] Jeyakumar V., Heller B., Handigol N., Lantz B., “Introduction to Mininet”, <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet> (dostęp online 20.06.2019).
- [3] OpenNetworking Foundation, „OpenFlow Switch Specification”, <https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf> (dostęp online 20.06.2019).
- [4] Varga A., “Simulation Manual”, <https://doc.omnetpp.org/omnetpp/manual/> (dostęp online 20.06.2019).
- [5] Varga A., “OMNeT++ IDE Developers Guide”, <https://doc.omnetpp.org/omnetpp/IDE-DevelopersGuide.pdf> (dostęp online 20.06.2019).
- [6] Varga A., “OMNeT++, IDE User Guide”, <https://doc.omnetpp.org/omnetpp/UserGuide.pdf> (dostęp online 20.06.2019).