sierż. pchor. inż. Krzysztof ŚWIDRAK

* Wojskowa Akademia Techniczna,
Wydział Elektroniki, Eksploatacja Systemów Łączności
ul. W. Urbanowicza 17, 01-476 Warszawa
e-mail: krzysztof.swidrak@student.wat.edu.pl

SECON 2019

KONCEPCJA WYKORZYSTANIA SDN DO WSPARCIA ARP

1. Wstęp

W bieżącej erze Internetu i IoT przyszedł czas na technologię, w której sterowanie zachodzi całkowicie w warstwie programowej. Coraz częściej usłyszeć można o Software Defined Radio (SDR) czy Software Defined Networks (SDN). Niniejszy artykuł poświęcony jest wykorzystaniu SDN do wsparcia w lokalnych sieciach protokołu Address Resolution Protocol (ARP).

Pierwszy rozdział stanowi wstęp do sieci lokalnych i protokołu ARP. Przedstawiona jest w nim również wspomniana już technologia SDN. Sieci programowalne pokazane są od strony ogólnego zamysłu, architektury, a zwieńczeniem pierwszej części jest zwięzły opis protokołu OpenFlow.

Druga część stanowi opis koncepcji, łączącej wymienione wcześniej technologie w celu zapewnienia efektywnego ruchu pakietów protokołu ARP w sieciach lokalnych (LAN). Opisane i porównane są trzy propozycje autora odnośnie zaprogramowania sterownika SDN. Należy podkreślić, iż założono, że administrator sieci definiuje jak w sieci ma przebiegać ruch (flow). Poruszona jest też w ograniczonym stopniu kwestia bezpieczeństwa sieci w każdej konfiguracji.

Trzecia część zawiera opis praktycznej implementacji wykorzystywanych mechanizmów. W niej też opisana jest weryfikacja proponowanych rozwiązań jak również przedstawione pokrótce zostaje środowisko testowe.

Artykuł w dużym stopniu odnosi się do pracy magisterskiej autora na temat: „Opracowanie kontrolera SDN w środowisku OMNeT++”, która została zrealizowana na Wojskowej Akademii Technicznej w latach 2018/2019 pod opieką i nadzorem mjr. dr. inż. Jerzego Dołowskiego.

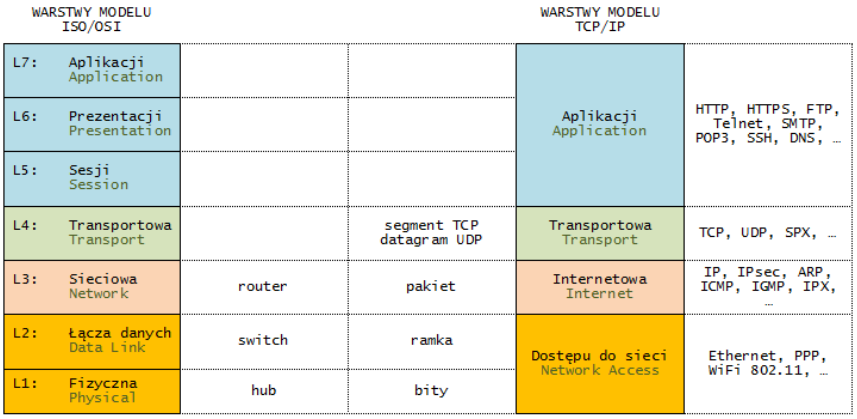
2. Rozwinięcie

2.1. Wprowadzenie w tematykę

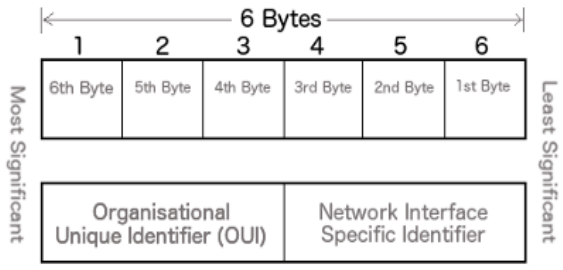
Raczej ciężko wyobrazić sobie dzisiaj życie bez urządzeń wykorzystujących protokół IP i łączących się z Internetem, który jest tak naprawdę zbiorem wielu sieci złożonych z połączonych między sobą urządzeń. W celu sprawnego działania sieci komputerowej opracowano wiele protokołów, które określają w niej sposoby komunikacji.

2.1.1. Address Resolution Protocol

W sieci lokalnej urządzenia swoje połączenia opierają głównie na bazie adresów warstwy drugiej [Rysunek 1]. Adresy te powinny być unikalne w skali globalnej. Najpopularniejszym jest Media Access Controll address (MAC). Składa się on z 48 bitów [Rysunek 2]. Pierwsze 24 bity określają Organizationally Unique Identifier. Każdy z producentów kupuje swój orginalny OUI od Institute of Electrical and Electronics Engineers (IEEE), którym sygnuje pierwsze 24 bity adresu MAC swojego urządzenia. Pozostałe 24 bity to numer seryjny urządzenia.



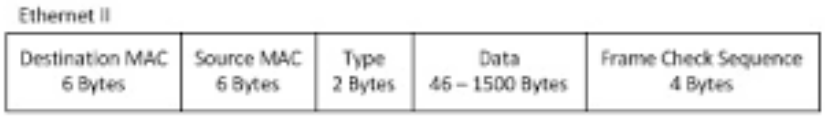
Rysunek 1 Model ISO/OSI i TCP/IP wraz z protokołami i rodzajami wiadomości.



Rysunek 2 MAC adres.

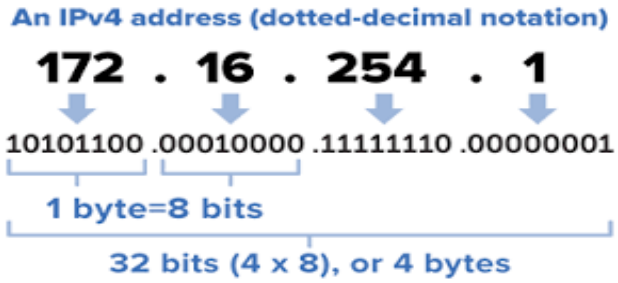
Oczywiście w dzisiejszych czasach istnieje możliwość podmiany adresu MAC np. dzięki użyciu w Kali Linux *macchanger*.

Urządzenie sieciowe chcąc połączyć się z drugim urządzeniem z sieci wewnętrznej musi posłużyć się jego adresem MAC, który umieszcza w polu docelowego adresu w ramce Ethernet. Ramka Ethernet posiada strukturę przedstawioną na rysunku [Rysunek 3]. W niniejszym artykule brana pod uwagę jest wyłącznie wersja podstawowa Ethernet, bez wsparcia VLAN (IEEE 802.1Q) oraz IEEE 802.3.



Rysunek 3 Ramka Ethernet.

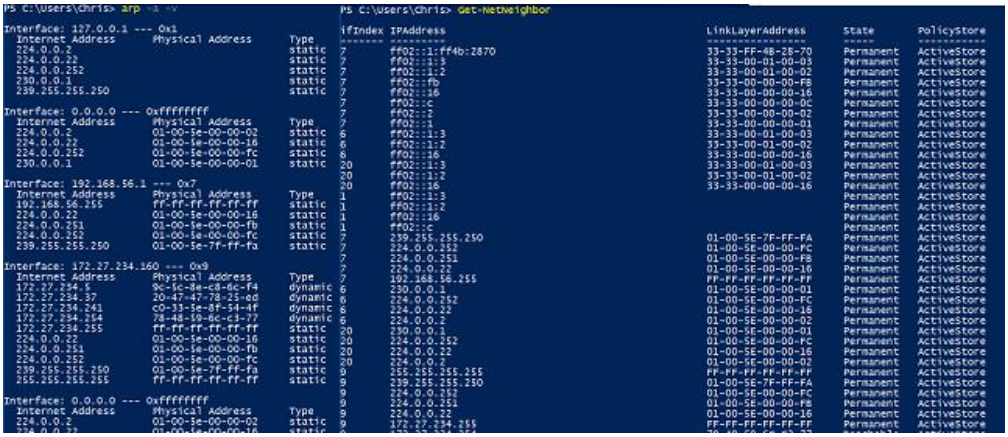
Każde urządzenie sieciowe powinno również posiadać swój adres IP [Rysunek 4] (rozważaniom poddany jest tylko IPv4). W sieciach lokalnych jest to rodzina adresów należących do jednej puli sieci. Adres IP ma 32 bity, z czego część oznaczona maską sieci określa adres sieci a pozostałe bity tworzą numer hosta oraz adres rozgłoszeniowy (wyzerowana część hosta to adres sieci, maksymalny numer to broadcast).



Rysunek 4 Adres IP.

Istnieje potrzeba powiązania adresu IP z adresem MAC, ze względu chociażby na chęć korzystania z usług protokołów warstw aplikacji [Rysunek 1]. W tym celu komputer posługuje się tablicą Address Resolution Protocol Table, czyli w skrócie tablicą ARP [Rysunek 5].

Zauważyć można, że niektóre wpisy są statyczne. Są to przede wszystkim adresy wirtualnych interfejsów np. usług VPN, maszyn wirtualnych. Jednakże w sieci liczącej chociażby 4 urządzenia trudno byłoby wpisywać ręcznie zwykłemu użytkownikowi wszystkie wpisy ARP do każdego urządzenia (może być to również trudne technicznie w niektórych urządzeniach). Z tego powodu pan David C. Plummer opracował i opublikował w ramach normę dotyczącą działania protokołu mapowania adresów sieciowych z ethernetowymi (IETF RFC 826)[1]. Wprowadza ona w użycie Address Resolution Protocol, czyli protokół, dzięki któremu urządzenia same dowiadują się o swoim istnieniu w sieci i dynamicznie uzupełniają tablice ARP.



Rysunek 5 Przykładowe wpisy ARP w Windows10 (Powershell).

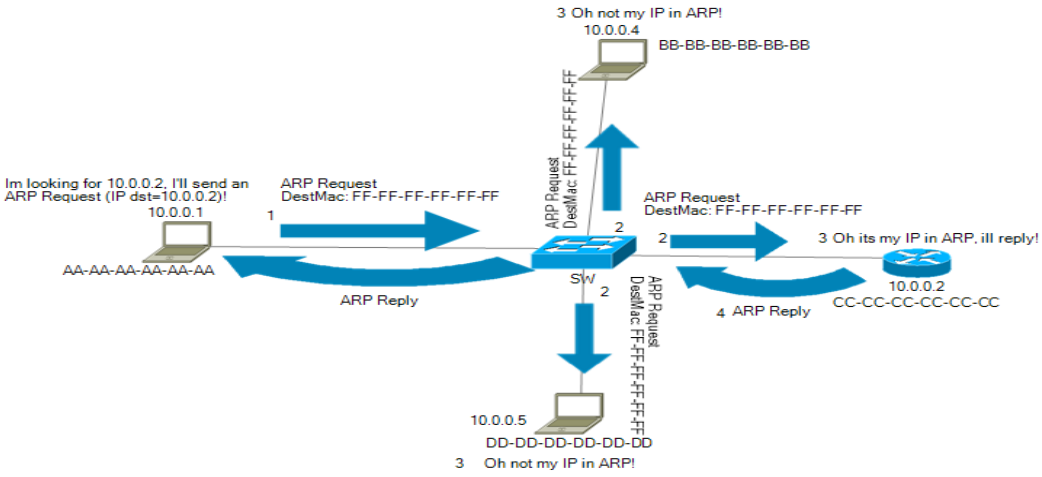
Pola ramki protokołu ARP przedstawione są na rysunku [Rysunek 6]. Określone są rodzaje adresów warstwy drugiej oraz protokołu warstwy trzeciej. Określone są odpowiednie długości adresów oraz ustawiona jest operacja. Operacja może być żądaniem (REQ=1) lub odpowiedzią (REP=2).

	Bits 0–7	Bits 8–15	Bits 16–23	Bits 24–31
0	Hardware address type (HTYPE)		Network protocol type (PTYPE)	
32	Length of hardware address (HLEN)	Length of protocol address (PLEN)	Operation	
64 96	Senders' MAC address			
112	Senders' IP address			
144 176	Recipients' MAC address			
192	Recipients' IP address			

Rysunek 6 Pola ramki protokołu ARP.

Działanie protokołu opisane jest na podstawie schematu przedstawionego na rysunku [Rysunek 7]. Komputer o adresie 10.0.0.1 chce wysłać ICMP echo request do routera 10.0.0.2 w sieci lokalnej 10.0.0.0/24. W tym celu sprawdza swoją tablicę routingu. W niej znajduje wpis, że jego interfejs znajduje się w sieci lokalnej i adres routera pasuje do tej sieci. Patrzy więc w tablicę ARP w celu zaadresowania ramki Ethernet. Nie widzi wpisu. Tworzy więc wiadomość protokołu ARP, gdzie jako adres odbiorcy IP ustawia 10.0.0.2 a w polu MAC odbiorcy ustawia zera. Jako nadawca podstawia swoje dane i uzupełnia pozostałe pola w tym operację REQ. W ramce ustawia adres docelowy jako broadcast (FF-FF-FF-FF-FF-FF) i wysyła wiadomość. Switch przesyła ją zgodnie z zasadą przełączania – przekazuje broadcast na każdy port. Hosty po odebraniu jej analizują zawartość. Router, wiedząc, że adres docelowy IP jest jego, uzupełnia adresy zgodnie z adresami interfejsu sieciowego i odsyła ramkę z powrotem zmieniając operację na REP. Pozostali

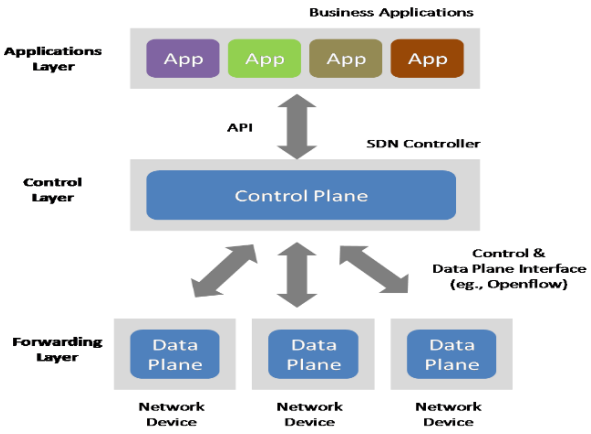
odrzucają ramkę. Często w literaturze, na stronach internetowych można znaleźć mylnie opisywaną sytuację, gdzie pola ARP nie są zamieniane (nadawcą nie staje się odbiorca). Jest to niezgodne z RFC jak również z praktyką (wystarczy przechwycić ARP w pcap albo Wireshark).



Rysunek 7 Schemat sieci.

2.1.2. Software Defined Networks

Sieci programowalne opierają się na rozdzieleniu płaszczyzny sterowania od płaszczyzny danych. Open Network Foundation [3] wyróżnia trzy warstwy na których opiera się SDN. Przedstawiono je na rysunku [Rysunek 8].



Rysunek 8 Warstwy SDN.

W praktyce oznacza to, iż sieć, w której przepływają tradycyjne dane oparte np. na IP zbudowana jest z węzłów (*switch*) nowej generacji sterowanych przez sterownik *(controller*). Sterownik jest fizycznie i logicznie oddzielony od urządzeń przełączających. Logiką sterownika steruje aplikacja, napisana przez administratora/użytkownika systemu SDN. Przełączanie danych przez switch opiera się na tablicach przepływów (*flow tables*), które definiowane są przez sterownik. Najczęściej jeżeli switch odbierze pakiet i żadna z reguł w tablicach przepływu nie pasuje do pakietu,

przekazuje on wiadomość do sterownika wraz z zamieszczonym pakietem, który może na tej podstawie podjąć jakąś decyzją (aplikacje przetwarzania przez sterownik mogą być oparte na algorytmach decyzyjnych i *machine learning*). Controller tworzy regułę, którą przekazuje do switcha oraz odsyła z powrotem pakiet, który jest traktowany zgodnie z przypisaną jemu akcją przez przełącznik. Tablice przepływów posiadają wejścia złożone z pól nagłówkowych (tabela poniżej), liczników i akcji. Pola z pakietów służą do porównania z analizowanym pakietem, liczniki służą zbieraniu statystyk (ile pakietów pasowało, ile nie itp.), akcje to operacje jakie ma wykonać switch na pakietach. Może on np. skierować pakiet na dane wyjście, zmienić jakieś pole w ramce itp.

Pola z pakietów używane do porównywania z tablicą przepływów											
Port wejściowy	MAC źródłowy	MAC docelowy	Typ w warstwie 2	VLAN id	VLAN priorytet	IP źródłowy	IP docelowy	Protokół w IP	IP ToS	Port źródłowy	Port docelowy

2.1.3. Protokół OpenFlow

Controller do komunikacji z data plane posługuje się south interface. Poprzez south interface rozumie się interfejs używany do komunikacji ze switchami. Najpopularniejszym i promowanym przez Open Networking Foundation protokołem jest OpenFlow. Istnieje kilka wersji tego protokołu, jednakże każda nowa wersja jest rozwinięciem poprzednich, najczęściej o dodatkowe funkcje. W niniejszym artykule używana jest wersja OpenFlow v.1.0.0 [2]. Protokół ten wykorzystuje transport poprzez TCP (dla wersji pierwszej port 6633, późniejsze 6653). Działanie protokołu opiera się na relacji typu klient-serwer. Klientem są switche, natomiast serwerem sterownik. Jednakże można wyróżnić kilka rodzajów wiadomości:

1) wspólne (HELLO, ERROR, ECHO, VENDOR),
2) konfigurujące switch (FEATURES, GET_CONFIG, SET_CONFIG),
3) asynchroniczne (PACKET_IN, FLOW_REMOVED, PORT_STATUS),
4) sterownika- zarządzające (PACKET_OUT, FLOW_MOD, PORT_MOD),
5) przerwania (BARRIER),
6) konfiguracji kolejek (QUEUE_GET_CONFIG).

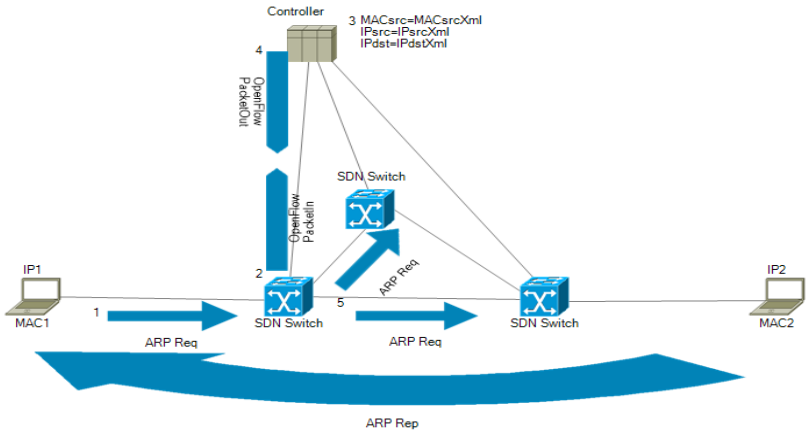
Wspólne wysyłane są przez obie strony niezależnie, natomiast konfigurujące są wiadomościami od sterownika do ogólnej konfiguracji switcha. Asynchroniczne, to wiadomości, które inicjowane są przez przełącznik. Zarządzające, to wiadomości, które służą sterownikowi do określania przepływów i operacji na przepływach switcha. Zaś przerwania, jak sama nazwa wskazuje, nakazują zakończenie pewnych procesów switcha, natomiast konfiguracji kolejek służą do zarządzania przez sterownik kolejkami switcha.

2.2. Koncepcja

Celem zastosowania opracowanych aplikacji jest zapewnienie zwiększenia bezpieczeństwa i efektywności przez zastosowanie SDN w sterowaniu ruchem pakietów ARP.

2.2.1. Pierwsza wersja

Opierając się na tym, iż administrator sieci zna swoją sieć lokalną, może on określić, jakie adresy MAC i IP mogą przepływać konkretnymi ścieżkami w sieci. Wiedząc, że protokół ARP zawiera w sobie informacje o hoście generującym ruch oraz hoście poszukiwanym, można stworzyć aplikację, która będzie działać w następujący sposób. Sterownik otrzymując w PacketIn dołączoną ramkę ARP, sprawdza w zdefiniowanym przez administratora zbiorze reguł, czy dany ruch jest dozwolony. Polega to na porównaniu adresów źródłowych MAC i IP oraz docelowego IP, jak również zgodności portów, z których przyszła ramka. Po podjęciu decyzji, jeżeli ruch jest dozwolony sterownik wysyła wiadomość PacketOut kierując otrzymaną wcześniej ramkę z ARP na port FLOOD (jak w zwykłym switchu, ramka wysyłana zostaje na każdy port, oprócz portu, z którego dotarła). Sytuacja jest przedstawiona na rysunku [Rysunek 9]. Należy podkreślić, że każdy switch po otrzymaniu ARP Request będzie wykonywał te same czynności. Odpowiedź ARP będzie przekazywana zgodnie z określoną trasą na podstawie adresów MAC i IP.



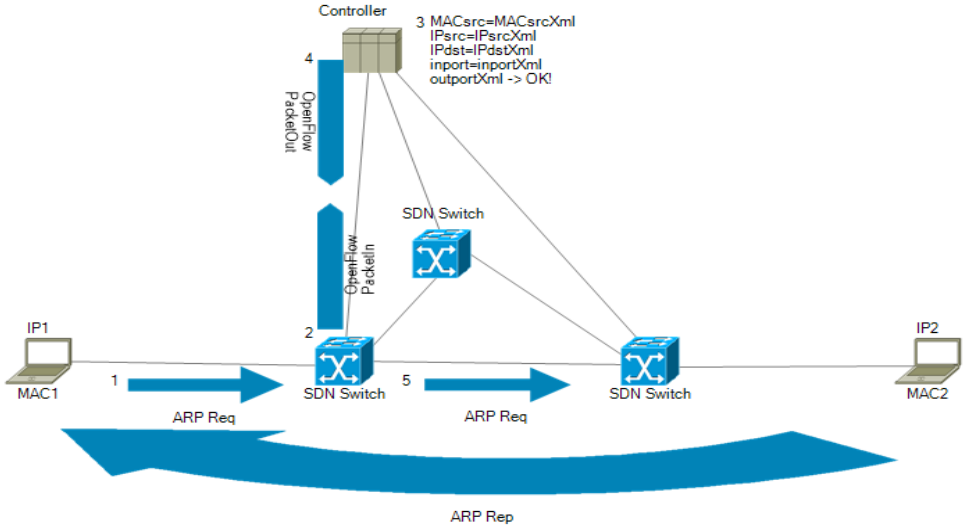
Rysunek 9 Schemat działania pierwszej wersji aplikacji.

Powyższe rozwiązanie pozwala na zabezpieczenie sieci przed nieautoryzowanym ruchem, zalewaniem tablic przełączania switcha oraz atakiem ARP Spoofing (pod warunkiem, że atakującym nie jest host, którego ruch jest dozwolony). Zasadniczą wadą zaś jest to, że zalewana pakietami ARP jest cała sieć (cała domena broadcastowa), co ogranicza w pewnym stopniu zasoby sieci.

2.2.2. Druga wersja

Druga wersja również, podobnie jak pierwsza opiera się na tym, że administrator przygotowuje wcześniej reguły określające dozwolone ruchy pakietów w sieci. Różni się ona jednak kierowaniem

pakietu na zdefiniowany w zbiorze reguł odpowiedni port wyjściowy do danego hosta (a nie tak jak w pierwszej wersji na FLOOD - controller wczytując zbiór reguł zna określoną przez administratora logiczną topologię sieci i wie, którędy dany ruch powinien przebiegać). Po sprawdzeniu parametrów podobnie jak w wersji pierwszej, controller odsyła PacketOut z akcją wysłania ramki ARP na dany port do switcha. Sytuacja zobrazowana na rysunku [Rysunek 10].

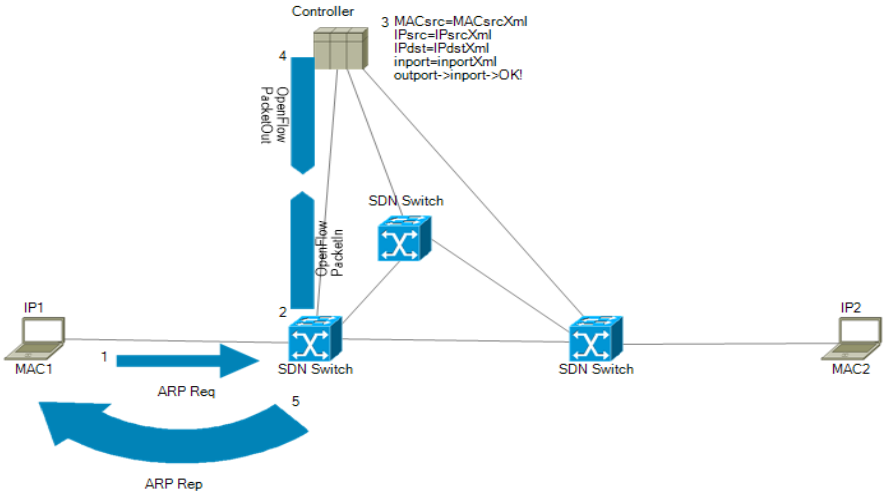


Rysunek 10 Schemat działania drugiej wersji aplikacji.

Wersja druga ma podobne zalety jak pierwsza. Dodatkowo ogranicza zalewanie sieci ramkami rozgłaszanymi na broadcast, przez co osiągnięta jest oszczędność zasobów sieci (urządzenia w domenie znajdujące się poza zdefiniowaną drogą do docelowego hosta, nie tracą czasu na przetwarzanie zbędnego ARP).

2.2.3. Trzecia wersja

W tej wersji aplikacji, sterownik podobnie jak w poprzednich wersjach sprawdza zgodność adresów w ramce żądania ARP oraz numerów portów. Jeżeli odpowiednie parametry zgadzają się z zadeklarowanymi w zbiorze reguł, sterownik tworzy odpowiedz ARP, w której umieszcza docelowy adres MAC zgodnie z wpisem w polityce przełączania. Następnie stworzony pakiet kapsułkuje w stworzoną nowo ramkę, w której ustawia odpowiednio nadawcę (poszukiwany ARP Requestem adres MAC) i odbiorcę (żądający ARP Request adres MAC). Sytuacja zobrazowana na rysunku [Rysunek 11].



Rysunek 11 Schemat działania trzeciej wersji aplikacji.

Zalety rozwiązania podobne jak w poprzednich wariantach. Należy zauważyć, że w tym wariancie znacząco odciążone zostają zasoby sieci. W zasadzie host korzysta z usług tylko switcha dostępowego sieci (bezpośrednio do niego podłączony). Dodatkowo w tym wariancie pakiety ARP Request, które są podstawą działania ataku ARP Spoofing nie mają prawa pojawić się na wejściu switcha- generować je może tylko i wyłącznie switch. Pojawienie się takiego pakietu będzie bezpośrednim dowodem na atak typu ARP Spoofing.

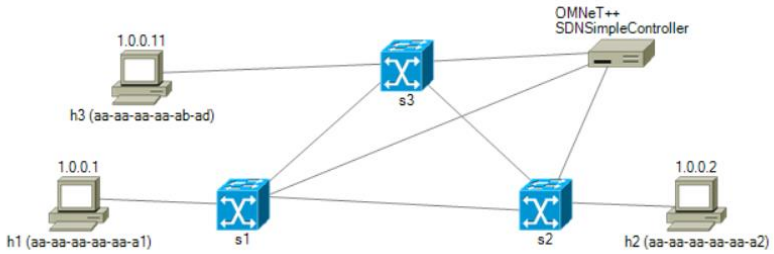
Wszystkie aplikacje swoją wiedzę na temat topologii sieci i dozwolonych tras w sieci opierają na pliku XML, który jest definiowany przez administratora sieci i zawiera zbiór reguł przełączania.

2.3. Implementacja i weryfikacja koncepcji

Do implementacji i weryfikacji aplikacji sterownika użyto zaimplementowanego przez autora artykułu Sterownika SDN w środowisku OMNeT++ [4]. Środowisko to jest symulatorem zdarzeń dyskretnych, jednakże umożliwia także emulację kart sieciowych (poprzez bezpośrednie wpięcie się na interfejs rzeczywistej karty sieciowej) oraz komunikację z rzeczywistymi urządzeniami w czasie rzeczywistym. Jako emulację sieci wykorzystano Mininet [5], w którym stworzono odpowiednią topologię (skrypt Python) do badań. Topologia sieci jest zgodna z topologiami z rysunków 9-11. Dodatkowo do północnego switcha podłączono hosta, którego nie uwzględniono w XML. Miało to na celu sprawdzenie czy sieć odporna jest na niedozwolony ruch. Dozwolona komunikacja w sieci to host IP1/MAC1 (h1) z IP2/MAC2 (h2). Tabela poniżej przedstawia odpowiednie adresy ww. hostów.

Zestawienie adresów hostów		
host	IP	MAC
IP1/MAC1 (h1)	1.0.0.1	aa-aa-aa-aa-aa-a1
IP2/MAC2 (h2)	1.0.0.2	aa-aa-aa-aa-aa-a2
Bad Host (h3)	1.0.0.11	aa-aa-aa-aa-ab-ad

Rysunek poniżej obrazuje topologię sieci oraz określone ścieżki ruchu [Rysunek 12, Rysunek 13].

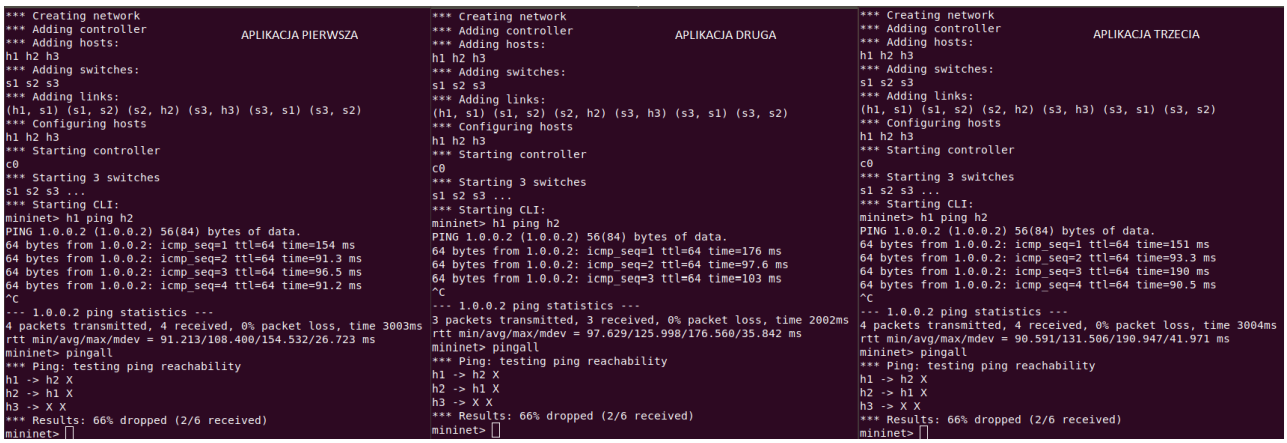


Rysunek 12 Topologia sieci do weryfikacji aplikacji.

```
<?xml version="1.0" encoding="UTF-8"?>
- <flows>
  - <switch name="s1">
    - <entry num="1">
        <mac-dst>AA-AA-AA-AA-AA-A1</mac-dst>
        <mac-src>AA-AA-AA-AA-AA-A2</mac-src>
        <ip-dst>1.0.0.1</ip-dst>
        <ip-src>1.0.0.2</ip-src>
        <in-port>2</in-port>
        <out-port>3</out-port>
      </entry>
    - <entry num="2">
        <mac-dst>AA-AA-AA-AA-AA-A2</mac-dst>
        <mac-src>AA-AA-AA-AA-AA-A1</mac-src>
        <ip-dst>1.0.0.2</ip-dst>
        <ip-src>1.0.0.1</ip-src>
        <in-port>3</in-port>
        <out-port>1</out-port>
      </entry>
    </switch>
  - <switch name="s2">
    - <entry num="1">
        <mac-dst>AA-AA-AA-AA-AA-A1</mac-dst>
        <mac-src>AA-AA-AA-AA-AA-A2</mac-src>
        <ip-dst>1.0.0.1</ip-dst>
        <ip-src>1.0.0.2</ip-src>
        <in-port>3</in-port>
        <out-port>2</out-port>
      </entry>
    - <entry num="2">
        <mac-dst>AA-AA-AA-AA-AA-A2</mac-dst>
        <mac-src>AA-AA-AA-AA-AA-A1</mac-src>
        <ip-dst>1.0.0.2</ip-dst>
        <ip-src>1.0.0.1</ip-src>
        <in-port>1</in-port>
        <out-port>3</out-port>
      </entry>
    </switch>
  - <switch name="s3">
    - <entry num="1">
        <mac-dst>AA-AA-AA-AA-AA-A1</mac-dst>
        <mac-src>AA-AA-AA-AA-AA-A2</mac-src>
        <ip-dst>1.0.0.1</ip-dst>
        <ip-src>1.0.0.2</ip-src>
        <in-port>1</in-port>
        <out-port>2</out-port>
      </entry>
    </switch>
</flows>
```

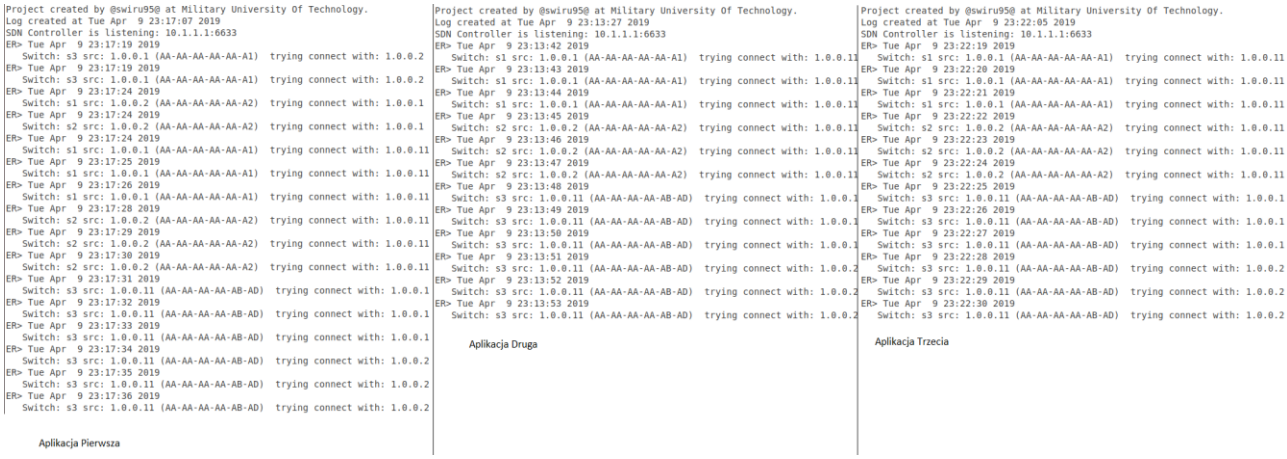
Rysunek 13 Przepływy XML dla sieci do weryfikacji aplikacji.

Pomiary polegały na sprawdzeniu połączenia pomiędzy hostami (*h1 ping h2*) oraz sprawdzeniu połączenia pomiędzy wszystkimi hostami (*pingall*) oraz przechwyceniu pakietów za pomocą Wireshark (interfejsy urządzeń sieciowych Mininet są wirtualizowane na urządzeniu- Wireshark umożliwia ich podejrzenie). Wyniki opisane poniżej [Rysunek 14]. Zauważyć można, że każda aplikacja filtruje ruch (umożliwia tylko dozwolone przepływy – h1 pinguje h2, ale nie pinguje h3).



Rysunek 14 Zrzut wyników pomiarów aplikacji (kolejno pierwsza, druga, trzecia) z Mininet.

Dodatkowo zauważyć można, że najszybciej połączenie udało się nawiązać za pomocą aplikacji trzeciej, co potwierdza najmniejsze zużycie zasobów sieci. Ruch niedozwolony został zablokowany, co widać na logach informujących sterownika [Rysunek 15]. Sterownik informuje o niedozwolonym ruchu podając adres IP i adres MAC oraz czas zdarzenia.



Rysunek 15 Logi sterownika (kolejno aplikacje: pierwsza, druga, trzecia).

Dodatkowo zebrano statystyki (liczniki) monitorowania liczby rodzajów pakietów kontrolera. Przedstawiono je w poniższej tabeli. Łatwo można zauważyć, że najbardziej efektywną jest trzecia aplikacja, ogranicza ona w dużym stopniu ruch pakietów ARP w sieci. Duża liczba otrzymanych ARP Req wynika zaś z faktu, iż w sieci występuje niedozwolona próba połączenia (h3). Packet_In oraz Packet_Out zawierają w sobie również ramki z pakietami IP (ICMP Echo Request/Reply). W pierwszej aplikacji poprzez zalewanie całej sieci ramkami rozgłoszeniowymi, liczba otrzymanych ARP Req przez kontroler jest największa (należy także uwzględnić, że wliczane są w nie również ramki z ARP dotyczące niedozwolonego ruchu – h3). Warto zwrócić uwagę, na zabezpieczenie przed atakiem ARP Spoofing. W trzeciej aplikacji otrzymane ARP Rep wynoszą zero - w sieci nie wystąpiły pakiety ARP Reply generowane przez hosty (nie było ataku). W razie jego wystąpienia sterownik poinformowałby o tym fakcie administratora generując odpowiednie logi.

Liczniki pakietów kontrolera			
Nazwa	Aplikacja Pierwsza	Aplikacja Druga	Aplikacja Trzecia
Received Packet_In	192	186	178
Sent Packet_Out	40	35	32
Sent ARP Req	5	5	0
Sent ARP Rep	5	5	2
Received ARP Req	21	17	14
Received ARP Rep	5	5	0

Powyżej przedstawione wyniki w wystarczającym stopniu potwierdzają, iż aplikacje spełniły przyjęte dla nich założenia.

3. Podsumowanie

Artykuł wykazał, jak duże możliwości posiadają sieci definiowane programowo. Udało się opracować warianty efektywnego zarządzania ruchem ARP w sieci, zwiększając jej możliwości zarówno pod względem oszczędności zasobów (zarówno pamięciowych jak również obliczeniowych) oraz zapewnienia bezpieczeństwa (uodpornienie na ARP Spoofing, zalewanie tablic przepływów, monitorowanie ruchu). Stwierdzić można, że w sieciach SDN barierą wydaje się być tylko wyobraźnia twórcy aplikacji/administratora systemu.

Bibliografia

[1] David C. Plummer: „An Ethernet Address Resolution Protocol”, RFC 826, IETF, November 1982

[2] Open Networking Foundation „OpenFlow Switch Specification”, Version 1.0.0, ONF TS-001, December 2009

[3] https://www.opennetworking.org/

[4] https://omnetpp.org/

[5] https://mininet.org/