****

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,**

**INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

Protokoły OpenFlow i OVSDB w sieciach programowalnych SDN.  
  
OpenFlow and OVSDB in programmable Software Defined Networks.

Autor: Marek Ryznar

Kierunek studiów: Informatyka

Opiekun pracy: dr Michał Turek

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. O prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): ,,Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.'', a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): ,,Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej <<sądem koleżeńskim>>.'', oświadczam, że niniejszą pracę dyplomową wykona łem(-am) osobiście i samodzielnie i że nie korzystałem(-am)   
ze źródeł innych niż wymienione w pracy.

Strona na serdecznie dziękuje itd

**Spis Treści:**

1. **Wprowadzenie**
   1. **Cele pracy**
   2. **Zawartość pracy**
2. **Wstęp teoretyczny**
   1. **SDN**
   2. **OVS**
   3. **OpenFlow**
   4. **OVSDB**
   5. **MEF**
3. **Kryteria porównania**
   1. **Tagowanie VLAN**
   2. **Przypisywanie pakietów do EVC**
   3. **Definiowanie QoS**
4. **Rozwinięcie**
   1. **Różnice między protokołami wg dokumentacji**
   2. **Tu dalej sie zobaczy**
5. **Porównanie wyników**
6. **Wnioski**
7. **Podsumowanie**
8. **Wprowadzenie**

Współcześnie pojęcie informatyka jest tak dużym zagadnieniem, że trudno znaleść jedną osobę, która mogła by się nazwać specjalistą od wszystkich jej aspektów. W związku z tym dzieli się ona na dziedziny, skupiające się wokół konkretnych problemów. Dziedzina, którą chciałem przedstawić w tej pracy to sieci komputerowe. Jest to jedna z kwestii, która musiała być załatwiona w procesie digitalizacji, który nas doprowadził do ery komputerów w jakiej dzisiaj żyjemy.

Jednym z najważniejszych standardów zdefiniowanych w specjalizacji sieci komputerowych jest OSI/ISO RM (*ISO Open Systems Interconnection Reference Model*) czyli siedmiowarstwowy model odniesienia dla większości protokołów komunikacyjnych. Warstwy druga (*łącza danych*) i trzecia (*sieciowa*) są to miejsca w których *znajdują się* przełączniki i routery. Zestawienie i utrzymanie sieci przy użyciu wspomnianych urządzeń nigdy nie było prostym zadaniem.

Z pomocą w wykoniu tego celu wkroczyła koncepcja sieci programowalnych czyli SDN (*Software Defined Network*). Odbiorca niniejszej pracy zostanie zapoznany z ich zastowaniem oraz z głównymi protokołami komunikacyjnymi używanymi w ich działaniu.

* 1. **Cele pracy**

Celem pracy jest porówanie dwóch głównych protokołów komunikacyjnych, czyli OVSDB (*Open vSwitch Database Management Protocol*) i OpenFlow działających   
w programowalnych sieciach komputerowych. Kryterium porównania są elementy składowe usług zgodnych ze standardem Carrier Ethernet 2.0 definiowanych przez organizację MEF (*Metro Ethernet Forum).*

Na samym początku rozważań protokoły zostaną porównane na podstawie ich specyfikacji. Następnie postawiona zostanie wirtualna sieć komputerowa typu SDN pozwalająca na zestawienie możliwości protokołów w praktyce. Wyniki symulacji zostaną przedstawione za pomocą dedykowanej do tego aplikacji.

* 1. **Zawartość pracy**

TODO

1. **Wstęp teoretyczny**
   1. **SDN**

Urządzenia sieciowe są od wielu lat instalowane i zarządzane w licznych firmach dostarczających usługi sieciowe. Mostki, przełączniki i rutery od zawsze były używane   
w wielu środowiskach wykonując funkcję filtrowania i przekazywania pakietów przez sieć komputerową. Pomimo wielu dobrze działających tradycyjnych technologii, wielkość   
i skomplikowanie współczesnych zestawień sieciowych buduje potrzebę innowacji w tej dziedzinie. Głównymi powodami są wiecznie rosnące koszty posiadania i zarządzania sprzętem sieciowym oraz wzrastająca potrzeba na nowoczesne centra danych. Obecnie wiele firm sieciowych powoli odchodzi od tradycyjnych metod w kierunku bardziej wolnodostępnego i otwartego na innowacje paradygmatu SDN. [1]

Aby zrozumieć fenomen sieci programowalnych trzeba najpierw zdefiniować tradycyjną architekturę przełącznika. Przedstawić można ją w świetny sposób za pomocą trzech warstw:

1. **Data Plane** – składa się z portów odpowiedzialnych za pobieranie pakietów oraz   
   z tablicy skojarzeń (*forwarding table*). Warstwa ta jest odpowiedzialna   
   za modyfikację nagłówka ramki oraz buforowanie i przekazywanie pakietów,
2. **Control Plane** – głównym zadaniem tej warstwy jest utrzymywanie aktualnych informacji w tablicy skojarzeń, tak aby data plane mogło kierować ruchem niezależnie bez ingerecji innych warstw. Tutaj właśnie procesowane są wszystkie protokoły odpowiedzialne za zmiane rekordów w tablicy skojarzeń, czyli odpowiedzialne za zarządzanie topologią sieci (np. RIP,OSPF, BGP itd.),
3. **Management Plane** – jest to ostatnia (najwyższa) warstwa modelu, odpowiedzialna zarządzanie urządzeniem bezpośrednio przez administratora   
   (np. Poprzez protokół SNMP).

Taki model warstwowy został przedstawiony na obrazku poniżej.



Rysunek 1: Role data, control i management planes [1].

---SDN strona 8—

Warstwa control plane jest nieustannie bombardowana przez protokoły do wyznaczania ścieżek pakietów. Faktem jest, że w wielkich sieciach center danych, około 30% czasu ruterów jest wykorzystywane do śledzenia topologii sieciowej [1]. Mimo zaawansowanych protokołów trasowania, możliwe jest usprawnienie obecnych sieci, tym rozwiązaniem jest użycie sieci programowalnych. Polega to na tym, że warstwa control plane nie jest zarządzana przez protokoły trasowania, ale przez centralnie zlokalizowany *inteligentny* kontroler posiadający informacje o topologii sieciowej. Na rysunku poniżej przedstawiona jest wersja sieci zarządzana przez kontroler SDN.



Rysunek 2: Zcentralizowany kontroler zarządzający warstwą control plane [1].

---SDN strona 16---

Prostota tego rozwiązania wynika z tego, że topologia sieciowa, jest stabilna i pod ścisłą kontrolą administratora sieciowego w jednym zcentralizowanym miejscu. Dodatkowym atutem przemawiającym za tym rozwiązaniem jest szybka diagnoza potencjalnych błędów w dużej sieci komputerowej.

* 1. **OVS**

Kolejnym ważnym pojęciem, które będzie się często przewijało w niniejszej pracy jest OVS (*Open Virtual Switch*) czyli wirtualny przełącznik. Jest to oprogramowanie przedstawiające wielowarstwowy switch, udostępniany na podstawie darmowej licencji (*Open Source Apache 2 License*). Jest doskonale dostosowany aby pełnić funkcję wirtualnego przełącznika w środowisku maszyn wirtualnych. Jest stworzony do automatyzacji sieci przy czym wspiera obecne standardy i protokoły (np. NetFlow, Sflow, IPFIX, RSPAN, CLI, LACP, 802.1ag). OVS wspiera wiele bazujących na linuxie technologii takich jak Xen/XenServer, KVM czy VirtualBox [2].

OVS jest przeznaczony do wieloserwerowch środowisk. Stan urządzeń wirtualnych   
w jednostce sieciowej (np. w maszynie wirtualnej) powinien być prosty do zreprodukowania w innym miejscu. Wirtualne przełączniki to umożliwiają, poprzez możliwość migrowania tablic skojarzeń, listy kontroli dostępu (ACLs), polityki QoS itd [2].

Struktura danych wirtualnego przełącznika OVS ukazana jest na rysunku poniżej.



Rysunek 3: Struktura danych OVS'a [3].

Na powyższej ilustracji możemy zauważyć, że wirtualny przełącznik posiada trzy podstawowe elementy:

1. Port – interfejs odpowiedzialny za połączenie OVSa z innymi urządzeniami. Jego konfiguracja jest ustawiana przez protokół OVSDB. W czasie działania wirtualnego przełącznika można w dowolnym czasie dodawać nowe porty i subinterfejsy,
2. Controller – każdy wirtualny switch może być zarządzany przez jeden lub więcej kontrolerów SDN,
3. Flow\_Table – tablica regół OpenFlow, czyli odpowiednik tablicy skojarzeń   
   w tradycyjnym przełączniku (z tym, że posiada większe możliwości).

Najnowsza wersja oprogramowania OVS to 2.7.0 i ta wersja będzie wykorzystywana w zestawieniu sieci wirtualnej (TODO: przy tworzeniu upewnić się czy na pewno ta wersja czy nie starsza).

* 1. **OpenFlow**

Wraz z powstaniem sieci programowalnych, na scenę wkroczył protokół komunikacyjny OpenFlow. Został on zdefiniowany w ramach standardów wyspecyfikowanych przez organizację ONF (*Open Networking Foundation*), założoną na początku roku 2011 przez takie firmy jak Microsoft, Google, Verizon, Yahoo!, Deutshe Telecom i Facebook [4]. Protokół ten ma za zadanie komunikować warstwę danych *(data plane)* w przełączniku z warstwą *control plane* w kontrolerze. Aby kontroler mógł się porozumieć z switch’em za pomocą opisywanego protokołu, ten musi posiadać do niego wsparcie. Na teraźniejszym rynku sieci komputerowych istnieje coraz więcej przełączników wspierających protokół OpenFlow. Najbardziej znane są przełączniki wirtualne takie jak opisywany wcześniej OVS lub mniej znany *Indigo* stworzony przez firmę *BigSwitch* [1]*.* Niektóre firmy produkujące sprzęt sieciowy również wyposarzyły swoje przełączniki o wsparcie dla protokołu OpenFlow (tzw. switch’e hybrydowe). Warto tutaj wspomieć, że kontrolery SDN komunikują się z urządzeniami zarządzanymi nie tylko za pomocą protokołu OpenFlow, używają również innych protokołów by być w stanie porozumiewać się z jak największą liczbą urządzeń (SNPM, Netconf itd.).

OpenFlow powstał w celu ujednolicenia sposobu komunikacji kontrolera   
z przełącznikami. Na wysokim poziomie abstrakcji można powiedzieć, że standard OpenFlow definiuje następujące komponenty potrzebne do komunikacji kontroler - przełącznik:

1. Kontroler,
2. Interfejs OpenFlow w przełączniku,
3. Kanał do bezpiecznej komunikacji,
4. Tabele przepływów (*flow table*) zawierającą instrukcje przepływów[4].

Powyżej wymienione komponenty wizualizuje rysunek 4:



Rysunek 4: Podstawowe komponenty OpenFlow [4].

--OpenFlow strona 7—

OpenFlow sam w sobie nie potrafi manipulować właściwościami przełączników i ch portów, odpowiedzialny jest tylko za modyfikowanie rekordów w tablicach skojarzeń OpenFlow (flow tables). W tym miejscu warto więc wspomieć, że ONF stworzył prtokół, który to umożliwia, nazywa się OF-CONFIG, jednakże w tej pracy nie będzie on opisywany, gdyż w porównaniu do OVSDB ma dużo mniejsze możliwości.

Więcej właśniwości OpenFlow zostanie omówione w dalszych rozdziałach.

* 1. **OVSDB**

Jak już to wcześniej było wspomiane, kontroler SDN może komunikować się   
z przełącznikami za pomocą wielu protokołów komunikacyjnych, jednym z nich jest OVSDB (*Open vSwitch Database Management Protocol*) stworzony przez firmę Nicra (później przejętą przez VMware). Został on stworzony wraz z stworzeniem wirtualnych przełączników OVS jako protokół ich zarządzania.

Istnieje ogólne przekonanie, że kontroler może porozumiewać się z wirtualnym przełącznikiem tylko za pomocą protokołu OpenFlow, ale jeżeli chodzi o konfigurację,   
tu wkracza OVSDB. Za jego pomocą można dodawać, modyfikować i usuwać wirtualne przełączniki OVS, ich porty i interfejsy. Możliwe jest zarządzanie urządzeniem OVS, wyłącznie przy użyciu protokołu OVSDB [5].

Na poniższym rysunku widać interfejsy wirtualnego przełącznika OVS, gdzie *ovsdb-server* to serwer bazy danych protokołu OVSDB, *ovs-vswitchd* to deamon vswitch’ai *openvswitch.*ko, czyli moduł kernela wykonujący szybkie przekierowywanie ścieżek.



Rysunek 5: Interfejsy przełącznika OVS [6]

//TODO: zmienić na aktualny rysunek!

Wraz z pojawieniem się na rynku OVSDB, niektóre firmy postanowiły wprowadzić wsparcie dla tego protokołu. Do niniejszych firm należą min. *Cumulus, Arista i Dell* [5]*.* Więcej o OVSDB zostanie powiedziane w późniejszych rozdziałach.

* 1. **MEF**

Aby dopełnić wstęp teoretyczny do niniejszej pracy, trzeba powiedzieć czym jest MEF. Jest to skrót od *Metro Ethernet Forum* i jest to międzynarodowa organizacja zrzeszająca ponad 220 przedsiębiorstw zajmujących dostarczaniem usług sieciowych, produkcją sprzętu sieciowego itd. Zajmuje się ona przyśpieszeniem zastosowania sieci komputerowych i usług typu *Carrier-class Ethernet* [7].

Organizacja MEF definiuje w jaki sposób dostawcy usług sieciowych powinni tworzyć usługi. Zanim wymienione zostaną typy usług, trzeba wyjaśniona musi być terminologia używana przy ich definicji. Najważniejsze pojęcia to:

1. CEN (*Carrier Ethernet Network*) – sieć dostawcy usług,
2. CE (*Customer Edge equipment*) – sprzęt po stronie subskrybenta usługi, podłączona do CEN,
3. UNI (*User Network Interface*) - to punkt fizycznego rozgraniczenia pomiędzy domenami klienta i dostawcy usługi,
4. EVC (*Ethernet Virtual Connection*) – logiczna reprezentacja połączenia w usłudze pomiędzy dwoma lub więcej punktami końcowymi UNI.

Znając terminologię opisującą usługi, możemy wymienić trzy podstawowe typy usług definiowane przez standard MEF:

1. E-Line – usługa definiująca bezpośrednie połaczeniem między dwoma punktami końcowymi UNI,
2. E-LAN – usługa umożliwiająca na więcej niż dwa UNI, gdzie każdy z nich może przekazywać ramki eternetowe do każdego innego UNI,
3. E-Tree – usługa łącząca wiele UNI, posiada strukturę drzewiastą, przy czym zapobiega przekazywaniu ramek eternetowych w kierunku od interfejsów użytkownika o roli Liść *(Leaf)* do tych o roli Korzeń (*Root)*. Przekazywanie ramek w odwrotnym kierunku jest możliwe.

Na rysunku poniżej znajduje się przykładowa usługa E-Line.



Rysunek 6: Podstawowa usługa MEF [7].

--MEF strona 24--

MEF do wszystkich wymienionych wyżej usług definiuje atrybuty, takie jak jakość usługi (*Quality of Sevice*), ramki usług na podstawie ramek Ethernet itd. Właśnie tego typu atrybuty będą wybrane jako kryterium porównania protokołów OpenFlow i OVSDB   
w niniejszej pracy i zostaną one dokładniej opisane w następnym rozdziale.

1. **Kryteria porównania**

Przy każdym porównaniu dwóch lub więcej obiektów potrzebne są kryteria względem których nastąpi zestawienie. W tej pracy kryteriami porównania są atrybuty usług zdefiniowanych przez standard CarrierEthernet 2.0 MEF. Wybrane atrybuty są zwięźle opisane w poniższych podpunktach.

* 1. **Tagowanie VLAN**

Usługi MEF są definiowane jako niezależne od obecnych technologii sieciowych pozwalając dostawcom usług na ich stworzenie przy użyciu dowolnych, odpowiadających im technologii. Jednakże definicja usług sieciowych według MEF wymaga aby połączenie pomiędzy klientem (*CE*) a siecią dostawcy usługi (*CEN*) była zgodna ze standardem IEEE 802.3 [7].

MEF definiuje ramkę danych jako ramkę eternetową warstwy drugiej modelu OSI/ISO. Tak zdefiniowana ramka nazywa się ramką usługi (*service frame*) i posiada wszystkie pola od docelowego adresu MAC (*Destination MAC Address*) do pola FCS (*Frame Check Sequence*), lecz nie zawiera komponentów ramki z warstwy pierwszej. Opisana ramka znajduję się na rysunku numer 7.



Rysunek 7: Ramka usługi [7]

--MEF strona 52—

Według standardu MEF ramki można rozróżniać między innymi na podstawie pola C-TAG (*Customer TAG*). Pole to zajmuje 4 bajty i składa się z następujących pól:

1. TPID (*Tag protocol identifier*) – 16 bitowe pole definiujące typ tagu,
2. TCI (*Tag control information*) – pole składające się z 3 składowych:
   1. PCP (*Priority code point*) – 3 bitowe reprezentujące priorytet na skali od 0 do 7,
   2. DEI (*Drop Eligibility Indicator*) – 1 bit mówiący czy ramka nadaje się do odrzucenia czy nie,
   3. VID (VLAN Identifier) – 12 bitów reprezentujących 4094 wartości identyfikujący do jakiej sieci VLAN ramka należy.

MEF 10.3 definiuje trzy formaty ramek:

1. Ramka nieotagowana – nie posiadająca żadnego tagu eternetowego,
2. Ramka tagowana przez VLAN – ramka C-TAG, gdzie TPID=8100, a VID jest różny od 0,
3. Ramki priorytetowe - ramka C-TAG, gdzie TPID=8100, a VID jest równy 0.

Oprócz ramek tagowanych przez klienta (C-Tagged) MEF rozpoznaje również dodatkowe pole S-TAG czyli tag VLAN należący do usługi. Pole to składa się z takich samych komponentów co pole C-TAG. Tak tagowane ramki są rozpoznawane przez MEF jako ramki usługi, ale niestety jak narazie nie mają zdefiniowanego zachowania. Poniższy rysunek w dobry sposób ukazuje rozpoznawanie taga VLAN ze wzglęgu   
na pole TPID [7].



Rysunek 8: Rodzaj tagu w zależności od TPID [7].

--MEF strona 53--

Podsumowując, ramka może posiadać jedno lub więcej pole VLAN defiujące   
do jakiej podsieci wirtualnej przynależy ramka. Tag ten może być definiowany przez klienta (C-TAG) lub przez dostawcę usługi (S-TAG). Porównując protokoły niniejszym kryterium sprawdzana będzie *umiejętność* protokołów do nadawania tagu VLAN ramce, filtrowania ramek na jego podstawie, edytowania i usuwania. Ponadto sprawdzone zostanie, czy za pomocą wybranego protokołu można nadać więcej niż jeden tag VLAN, czyli czy wspierają mechanizm IEEE 802.1 Q-in-Q zazwyczaj używany do nadania tagu VLAN przez dostawcę usługi (S-TAG).

* 1. **Przypisywanie pakietów do EVC**

Ramki usługi są przypisywane do EVC na dwa podstawowe sposoby. Pierwszy   
z nich jest bazowany na portach (*port-based*), co znaczy że wszystkie ramki usługi przy każdym z UNI są przypisywane do jednego EVC. Drugi sposób polega na wybieraniu wyjściowego EVC na podstawie tagu VLAN (*VLAN-based*) [7]. W tym kryterium protokoły będą porównywane na podstawie ich możliwości do przypisywania pakietów do EVC na podstawie portów i na używając tagów VLAN.

* 1. **Definicja QoS**

QoS jest skrótem od *Quality of Service* czyli jakość usługi. Według standardu MEF termin ten odnosi się do wydajności dostarczania ramek usługi, co zostało zdefiniowane w SLS (*Service Level Specification*) używając atrybutów EVC dotyczących wydajności usługi. Te atrybuty to dostępność, elastyczność, współczynnik strat ramki i opóźnienia ramek [7].

Aby uzyskać dany QoS, trzeba skorzystać z zarządzania ruchem (*traffic managemnt*) czyli zbioru mechanizmów, narzędzi i polityk używanych przez dostawcę usługi. Jednym z mechanizmów używanych do uzyskania QoS są profile przepustowości (*Bandwidth profiles*). Używane są one do ograniczenia obciążenia usługi *Carrier Ethernet.*

W tym kryterium protokoły będą głównie porównywane pod względem stopnia implementacji profili przepustowości. Jeżeli któryś z protokołów będzie posiadał możliwość ustawienia jakiegoś parametru QoS, zostanie to porównane do możliwości drugiego protokołu w tym zakresie.

1. **Rozwinięcie**

W niniejszym rozdziale znajduje się porównanie protokołów OpenFlow oraz OVSDB, gdzie najpierw zostanie ono wykonane według specyfikacji protokołów a następnie   
w rzeczywistych przypadkach użycia. Zestawienie ich będzie wykonane według kryteriów zdefiniowanych w poprzednim rozdziale.

* 1. **Dokładniejszy opis protokołów**

Aby być w stanie porównać działanie protokołów według danych kryteriów, trzeba przedstawić to, w jaki sposób protkoły wykonują podstawowe funkcje. W tym podrozdziale OpenFlow i OVSDB zostaną opisane w nieco szerszym zakresie niż było to zrobione we wstępie teoretycznym.

* + 1. **OpenFlow**

OpenFlow został stworzony przez organizację ONF, która wydaje wiele dokumentów specyfikujących ten protokół. Niniejszy opis protokołu jest bazowany na podstawie dokumentu *OpenFlow Switch Specification*, który opisuje OpenFlow, jako protokół służący komunikacji kontrolera SDN z przełącznikiem. Dodatkowo określa wymagania, co do switch’a, z którym się komunikuje. ONF wydał już kilka wersji tej specyfikacji, w tej pracy wykorzystana zostanie najnowsza 1.5.0 (0x06) wydana 19 Grudnia 2014 roku [8].

Jak było to wspomniane we wstępie teoretycznym, podstawowe komponenty OpenFlow to switch i kontroler. O ile kontroler jest istotny w architekturze SDN, tak w architekturze OpenFlow jest drugoplanowy, ponieważ protokół nie ma nic do powiedzenia w kwestii decyzji, jakie instrukcje są przesyłane do przełącznika, jest to rozstrzygane na poziomie kontrolera lub na wyższych warstwach.

Przełącznik jest miejscem, w którym dzieje się większość akcji z punktu widzenia protokołu. Przetwarzanie pakietów odbywa się przez tabele: *flow tables, group table* i *meter table*. Rekordy w tabelach inicjalizowane są poprzez instrukcje podawane przez kontroler poprzez kanał OpenFlow. Pakiety wchodzą i wychodzą przez porty, tak jak w tradycyjnym przełączniku [4].

OpenFlow definiuje 3 podstawowe typy portów:

1. Porty fizyczne – odwołują się bezpośrednio do interfejsów sprzętowych,
2. Porty logiczne – w tym terminie zawierają się wszystkie logiczne interfejsy napotykane w tradycyjnych przełącznikach, np. interfejs loopback, interfejs null, MPLS LSP itd. Z perspektywy OpenFlow port logiczny jest traktowany tak samo jak fizyczny,
3. Porty zarezerwowane – używane są do wewnętrznego przetwarzania pakietów. Porty te mogą być wymagane lub opcjonalne.   
   Porty wymagane:

* ALL - oznacza wszystkie porty oprócz wejściowego,
* CONTROLLER – port do bezpiecznego kanału łączącego przełącznik z kontrolerem,
* TABLE – wejściowy port do tzw. *pipeline, czyli* miejsca, w którym znajdują się tabele,
* IN\_PORT – używany jest, kiedy pakiet musi być przesłany do portu, którym przyszedł,
* ANY – może reprezentować jakikolwiek port lub wszystkie.

Porty opcjonalne:

* LOCAL – wejściowy i wyjściowy port należący do lokalnego stosu przełącznika,
* NORMAL – wspierany tylko w switch’ach hybrydowych i jest to wyjściowy port przekierowujący pakiety do normalnego przetwarzania przełącznika,
* FLOOD – tak jak NORMAL, wspierany tylko przez switch’e hybrydowe. Przekazuje pakiet na wszystkie interfejsy fizyczne lub na pewną ich podgrupę[4].

Przełącznik OpenFlow komunikuje się z kontrolerem poprzez tzw. *Secure Channel*, który korzysta z TCP na porcie 6653, co zostało zdefiniowane przez IANA (*Internet Assigned Numbers Authority*). Switch inicjalizuje połączenie z kontrolerem podczas startu lub restartu, które zazwyczaj jest szyfrowane poprzez protokół TLS (*Transport Layer Security*). [4]

Jak już było to wspomniane, przetwarzanie pakietów odbywa się poprzez 3 tabele, pierwszą z nich jest *flow table.* Składa się ona z następujących pól:

1. *Match fields* – określa, pod jakimi warunkami pakiet jest dopasowywany. Może to być kombinacja portów wchodzących, nagłówek Ethernet lub IPv4, metadane itd.,
2. *Priority* – definiuje kolejność rekordów,
3. *Counter* – zapisuje statystyki dopasowanych pakietów,
4. *Instruction* – specyfikuje, jakie akcje mają być wykonane po dopasowaniu pakietu,
5. *Timeouts* – specyfikuje maksymalny czas istnienia rekordu (jeśli nie jest używany),
6. *Cookie ­*– używany przez kontroler do filtrowania rekordów,
7. *Flags* – służy do zmieniania sposobu dodawania rekordów [4].

Format rekordu tabeli przepływu zilustrowany jest na rysunku poniżej:



Rysunek 9: Format rekordu tabeli przepływu [4].

--OpenFlow str 51 3.5—

W pierwszej wersji protokołu (1.0.0) przełącznik OpenFlow mógł posiadać tylko jedną tabelę przepływu. Od wersji 1.1.0 switch może posiadać więcej takich tabel. Wraz z tym został wprowadzony mechanizm o nazwie *pipeline processing* odpowiadający za tworzenie hierarchii przetwarzania tabel przy użyciu prostych warunków „if-then-go”. Wszystkie przychodzące pakiety muszą być przetworzone przez tabele od id równym 0, po czym mogą być przekazane do kolejnych tabel. W dobry sposób przedstawia to rysunek 10.



Rysunek 10: Przetwarzanie pakietu przez wiele tabel [4].

--OpenFlow str 54 3.8—

Oprócz tabel przepływu OpenFlow definiuje tzw. *group tables*, w których rekordy reprezentują grupę pakietów, które powinny być traktowane w ten sam sposób. Tworząc pakiet, jako członek pewnej wybranej, zdefiniowanej wcześniej grupy, akcje mogą być wykonywane w bardziej wydajny sposób w porównaniu do tradycyjnego sposobu przepływu [4].

Ostatnią z opisywanych tabel jest wprowadzona w wersji 1.3.0 *meter table,* odpowiedzialna za tworzenie prostych mechanizmów QoS, które mierzą szybkość przepływu i na jego podstawie nakładają pewien limit. Tabela ta zostanie lepiej omówiona w dalszych podpunktach [4].

Wszystkie wiadomości OpenFlow są enkapsulowane wraz z nagłówkiem. Pola w nim zawarte to:

1. *Version –* wersja według standardu *OpenFlow Switch Specification*,
2. *Type –* typ wiadomości, dzieli się na trzy kategorię: *Controller-to-Switch* (np. FEATURE REQUEST, PACKET OUT, FLOW MOD, TABLE MOD, GROUP MOD, METER MOD)*, Asynchronous* (np. PACKED IN, ERROR)i *Symmetric* (np. HELLO, ECHO REQUEST, ECHO REPLY),
3. *Length –* długość wiadomości,
4. *Transaction ID (XID) –* id wiadomości[4].

Istnieje kilka struktur używanych w wiadomości OpenFlow opisujących elementy przełącznika:

1. *Port Structures –* definiuje porty,
2. *Queue Structures –* każda kolejka składa się z:

* 32 bitowego numeru portu,
* 32 bitowego id kolejki,
* Minimalna i maksymalna szybkość przepływu na porcie,
* Stuktura dopasowywania przepływu (*Flow Match),* która opisuje dopasowywanie pakietów lub rekordów przepływu (*flow entries*) [4].

Po opisaniu najważniejszych komponentów, można przystąpić do objaśnienia tego, w jaki sposób przepływy są przetwarzane w przełączniku OpenFlow. Warunki dopasowania w rekordzie tabeli przepływu dzielą się na trzy kategorie:

1. *Flow Match ­*– przepływ jest identyfikowany, jako kombinacja parametrów takich jak port wchodzący, adres źródłowy i docelowy, bity CoS (*Class of Service*) i porty wyższych warstw. Bazując na tych parametrach przepływ może być dopasowywany,
2. *Header Match* – przepływ może być również dopasowywany na podstawie zawartości nagłówków pakietów (warstwy 2,3 i 4),
3. *Pipeline Match* – dopasowywanie na podstawie pól dodanych do pakietu, innych niż nagłówek pakietu. Te pola to IN\_PORT (wejściowy port), IN\_PHY\_PORT (wjściowy port fizyczny), METADATA (metadane przekazywane między tabelami) oraz TUNNEL\_ID (metadane portu logicznego) [4].

Gdy rekord tabeli przepływu zostanie dopasowany, polecenia zawarte w polu *Instructions* zaczynają być wykonywane. Każde pole *Instructions* zawiera zbiór instrukcji (*Instruction set*), a każda instrukcja może (lecz nie musi) zawierać listę akcji (*Action List*). Dopiero akcja oznacza jakieś konkretne polecenie. Opisane relacje w dobry sposób zilustrowane są na poniższym rysunku.



Rysunek 11: Relacje instrukcji i akcji [4].

--OpenFlow str 67—

Pole *Instruction* ze zbioru instrukcji (*Instruction Set)* może być typu:

1. Meter *meter-id* (Opcjonalne) – wysyła pakiet do *meter table* wyspecyfikowanej przez *meter-id,*
2. Apply-Actions *actions(s)* (Wymagane) – Wykonuje podaną listę akcji, bez zmieniania zbioru akcji,
3. Clear-Actions (Opcjonalne) – usuwa wszystkie akcje ze zbioru akcji (*action set*),
4. Write-Actions *action(s)* (Wymagane) *–* dodaje listę akcji do zbioru akcji,
5. Write-Metadata *metadata/mask* (Opcjonalne) – zapisuje metdane do pola metadanych dla *pipeline processing,*
6. Goto-Table *next-table-id* (Wymagane) – przejście do następnej tabeli przepływu [4].

W zbiorze instrukcji, każda z nich jest wykonywana w takiej kolejności, w jakiej zostały wymienione w powyższej liście. Dodatkowe określenie, które nie zostało do tej pory objaśnione to zbiór akcji (*action set*). Jest on inicjalizowany na początku *pipeline processing* i składa się z list akcji dodawanych przez instrukcję Write-Actions. Wszystkie listy ze zbioru akcji wykonywane są w momencie osiągnięcia przez pakiet ostatniej tabeli przepływu.

Po długim wstępie można w końcu wyjaśnić, w jaki sposób dodawane są operacje wyjściowe dla dopasowanych już pakietów w przełącznikach OpenFlow. Dzieje się to za pomocą wykonania akcji. Mogą one specyfikować wyjściowy port (OUTPUT *port*), dodawać nowe pole VLAN (PUSH\_VLAN *ethertype*) i wykonywać inne akcje, które zostaną opisane dokładniej później.

* + 1. **OVSDB**

Specyfikacja tego protokołu jest tworzona m.in. w dokumencie *The Open vSwitch Database Management Protocol* przez organizację IETF. Dokument ten mówi o tym, w jaki sposób protokół porozumiewa się z bazą danych przełącznika OVS   
w celu jego konfiguracji. Ostatnia wersja dokumentacji została stworzona w Październiku 2014 roku.

Jak to było wcześniej wspomniane, switch OVS zawiera bazę danych (*ovsdb-server*), deamon’a urządzenia (*ovs-vswitchd*) i opcjonalnie moduł wykonujący szybkie przekierowywanie ścieżek, czyli moduł OpenFlow. Operacje wykonywane przez protokół OVSDB to:

1. Tworzenie, modyfikacja i usuwanie wirtualnych mostków (*bridges*), których może być wiele w pojedynczej instancji OVS,
2. Konfiguracja połączeń z kontrolerami dla każdego z mostków (według dokumentacji inna nazwa na mostek to *OpenFlow datapath*),
3. Konfiguracja menadżerów, do których podpięty jest serwer OVSDB,
4. Tworzenie, modyfikacja i usuwanie portów na mostkach,
5. Tworzenie, modyfikacja i usuwanie interfejsów tunelowych (np. GRE) dla mostków,
6. Tworzenie, modyfikacja i usuwanie kolejek,
7. Konfiguracja polityki QoS i podpinanie jej do wcześniej zdefiniowanych kolejek,
8. Zbieranie statystyk [6].

OVSDB używa formatu JSON, jako schemat danych i przy przekazywaniu danych, bazuje na JSON-RPC 1.0. We wspomnianym dokumencie oprócz opisanych już informacji znajduje się opis metod RPC i działania na bazie danych [6].

Oprócz powyżej opisanego dokumentu, warto zaznajomić się ze schematem bazy danych urządzenia OVS [9]. Tabele tam zawarte to:

1. Open\_vSwitch – konfiguracja OVSa,
2. Bridge – konfiguracja mostka,
3. Port – konfiguracja portu,
4. Interface – fizyczne urządzenie sieciowe podpięte pod port,
5. Flow\_table – konfiguracja tabeli przepływów,
6. QoS – konfiguracja jakości usługi,
7. Queue – wyjściowa kolejka QoS,
8. Mirror – port mirroring,
9. Controller – konfiguracja kontrolera OpenFlow,
10. Manager – konfiguracja połaczenia OVSDB,
11. NetFlow – konfiguracja NetFlow,
12. SSL – konfiguracja SSL,
13. sFlow – konfiguracja sFlow,
14. IPFIX – konfiguracja IPFIX,
15. Flow\_Sample\_Connector\_Set – konfiguracja Flow\_Sample\_Connector\_Set.

Rysunek 12 ilustruje relacje pomiędzy tymi tabelami.



Rysunek 12: Relacje pomiędzy tabelami z bazie danych OVS [9].

W dalszej części dokumentu opisywane są wszystkie wymienione tabele. Wybrane z nich będą opisane w dalszych podpunktach w zależności od potrzebnych informacji.

* 1. **Różnice między protokołami według dokumentacji**

* + 1. **Tagowanie VLAN**

1. **OpenFlow**

Operowanie na tagu VLAN w OpenFlow jest możliwe od wersji 1.1.0   
i jest wykonywane za pomocą trzech akcji:

* PUSH\_VLAN *ethertype* – Dodaje nowy nagłówek VLAN do pakietu, jeżeli tag już istanieje, nowy zostaje najbardziej zewnętrzym tagiem.  
  Atrybut *ethertype* musi wynosić 0x8100 lub 0x88a8, czyli oznacza odpowiednio tagi C-TAG lub S-TAG według standardu MEF,
* POP\_VLAN – Ściąga najnowszy tag VLAN,
* SET\_FIELD *field –* Akcja ustawia wartości dla najbardziej zewnętrznego pola z nagłówka pakietu wyspecyfikowanego przez nazwę pola (*field*).

W tym przypadku jest to albo identyfikator VLAN (VLAN\_VID) lub priorytet (VLAN\_PCP) [4].

Przy założeniu, że używany przełącznik w pełni implementuje *OpenFlow Switch Specification* w wersji nie niższej niż 1.1.0, przy pomocy protokołu OpenFlow możemy dodawać, modyfikować i usuwać zarówno tagi VLAN dla klienta (C-TAG) jak i dla konkretnej usługi (S-TAG).

1. **OVSDB**

Konfiguracja VLAN w przypadku OVSDB może być ustawiana w tabeli *Port*. Operacje są wykonywane za pomocą tworzenia i modyfikacji wartości z następujących kolumn:

* *vlan-mode –* konfiguracja VLAN na porcie może być jednym z trzech typów: *access, native-tagged, native-untagged* lub *trunk*,
* *tag –* liczba całkowita od 0 do 4095,
* *trunks –*zbiór tagów, przy czym maksymalna ich liczba to 4096,
* *other\_config:**priority-tags* **–** wartość *true* lub *false***.**

Działanie portu z ustawioną konfiguracją VLAN zależy od trybu konfiguracji, czyli od jednej z czterech wartości z kolumny *vlan-mode*:

* *trunk* – port trunk przepuszcza pakiety z nagłówkiem 802.1Q, których identyfikatior VLAN znajduje się w kolumnie *trunks*, wszystkie inne pakiety są odrzucane.
* *access* – przepuszcza pakiety oznaczone VLAN tagiem ustawionym   
  w kolumnie *tag.* Wszystkie inne są odrzucane.
* *native*-*tagged* – przypomina port *trunk*, z tą różnicą, że przychodzące pakiety bez nagłówka 801.1Q znajdują się w sieci *native VLAN*, o numerze wyspecyfikowanym w kolumnie *tag* (dodawany jest nagłówek z danym VLAN ID)*,*
* *native*-*untagged –* jest podobny do *native*-*tagged,* z tą różnicą, że pakiety wychodzące do sieci *native* nie są tagowane [9].

Podsumowując, jeśli przełącznik posiada zaimplementowaną funkcjonalność przedstawioną w schemacie bazy danych OVS, to protokół OVSDB potrafi tworzyć różne typy portów VLAN filtrujące ruch. Niestety, jeżeli chodzi o dodawanie nagłówków 802.1Q, OVSDB potrafi tylko w przypadku gdy port jest typu ­*native-tagged* i to w bardzo ograniczonym zakresie, podczas gdy OpenFlow nie posiada w tym temacie żadnych obostrzeń.

* + 1. **Przypisywanie pakietów do EVC**

1. **OpenFlow**

Przypisywanie pakietów do konkretnej ścieżki jest głównym zadaniem OpenFlow, więc w tym przypadku OpenFlow nie posiada żadnych ograniczeń. Pakiety mogą być dopasowywane do konkretnych akcji na podstawie wielu parametrów. Przykładami mogą być port wejściowy (IN\_PORT), adresy źródłowy (ETH\_SRC, IPV4\_SRC, IPV6\_SRC) i docelowy (ETH\_ DST, IPV4\_ DST, IPV6\_DST), identyfikator VLAN (VLAN\_VID) i wielu innych.

Każdy dopasowany już pakiet, jeżeli nie został odrzucony, musi być przekierowany na jakiś port wyjściowy. W OpenFlow realizowane jest to za pomocą akcji „OUTPUT *port”,* gdzie *port* specyfikuje port wyjściowy. OpenFlow realizuje kryterium przypisywania pakietów do EVC zarówno w trybie *port-based* jak i ­*vlan-based.*

1. **OVSDB**

OVSDB nie jest protokołem służącym do przypisywania pakietów do konkretnych ścieżek, ma jednakże możliwość tworzenia tuneli za pomocą GRE (*Generic Routing Encapsulation*). Stworzenie takich tuneli odbywa się dzięki modyfikacji rekordów w tabeli *Interfaces,* która definiuje podpięty pod port interfejs. Kolumny dotyczące tunelowania to:

* *options : remote\_ip –* Adres końcowy tunelu (IPv4),
* *options : local\_ip –* Adres lokalny,
* *options : in\_key ­–* klucz, który musi być zawarty w pakiecie wejściowym,
* *options : out\_key -* klucz ustawiany przy wysyłaniu pakietów,
* *options : key –* skrót do ustawienia klucza wejściowego i wyjściowego za jednym razem,
* *options : tos –* wartość type-of-service,
* *options : ttl –* ustawiany czas życia enkapsulowanego pakietu,
* *options : df\_default –* jeżeli ustawione na *true,* do nagłówka pakietu dodawane są bity *Don’t fragment,*
* *options : csum –* obliczona suma kontrolna GRE dla wychodzących pakietów.

Aby stworzyć taki tunel, użytkownik musi tylko ustawić pole *remote\_ip*, pozostałe są opcjonalne [9]. Dzięki stworzonemu tunelowi, każdy pakiet, który będzie przechodził przez dany port, zostanie przetransportowany na drugi jego koniec. Można to nazwać sposobem przypisania pakietu do EVC, ale na pewno nie efektywnym i wygodnym w konfiguracji. Tego typu tunele tworzy się raczej pomiędzy maszynami wirtualnymi, więc funkcjonalność tego rozwiązania pozostawia wiele do życzenia.

* + 1. **Definicja QoS**

1. **OpenFlow**

W celu definicji jakości usługi za pomocą protokołu OpenFlow, trzeba skorzystać z wcześniej już wspominanego komponentu *meter table*, wprowadzonego w wersji 1.3.0. Tabela ta pozwala na mierzenie szybkości przepływu i nakładanie odpowiednich ograniczeń. Każdy rekord tabeli *meter table* powiązany jest w rekordem z tabeli przepływu [4].

Pola z rekordu tabeli *meter* są następujące:

* *Meter identifier –* id reprezentowany przez 32 bitową liczbę całkowitą,
* *Meter Bands* – nieuporządkowana lista ograniczeń (*Meter Band*),
* *Counters –* Licznik, który jest zwiększany za każdym razem, kiedy przez tabelę przechodzi pakiet.

Pole *Meter Band* składa się z:

* *Band Type –* Definiuje jak pakiet ma być przetwarzany,
* *Rate –* Specyfikuje najwolniejszą szybkość jaką *meter band* może aplikować,
* *Burst –* definiuje ziarnistość *meter band,*
* *Counters -* licznik
* *Type Specific Arguments –* używany z niektórymi typami ograniczeń do podania dodatkowych instrukcji [4].

Przekierowanie do tabeli *meter* z rekordu tabeli przepływu następuje w chwili rozpoczęcia wykonywania instrukcji związanych z danym rekordem. Za pomocą *meter table* można ograniczyć szybkość przepływu pakietów, czyli możliwa w wykonaniu jest jedna z podstawowych funkcji QoS.

1. **OVSDB**

Definicję jakości usługi w przypadku OVSDB definiuje się za pomocą dodawania rekordów w tabeli QoS, która przypisywana jest do każdego portu. Posiada następujące pola:

* *Type –* typ implementowanego QoS. Na ten moment istnieje wsparcie dla *linux-htb* (Linux hierarchy token bucket classifier) oraz *linux-hfsc* (Linux "Hierarchical Fair Service Curve" classifier),
* *Queues –*mapa z rekordami w postaci (*integer, Queue*), gdzie drugi argument przedstawia tabelę kolejek,
* *other\_config : max-rate –* określa maksymalną szybkość łącza. Domyślna wartość to 100 Mbps.

Tabela kolejek (*Queues*) wygląda następująco:

* *other\_config : min-rate –* Minimalna gwarantowana przepustowość, w bit/s,
* *other\_config : max-rate –* Maksymalna dopuszczalna przepustowość,
* *other\_config : burst –* Maksymalna ilość „kredytu” podczas gdy kolejka jest bezczynna,
* *other\_config : priority* – priorytet [9].

OVSDB ma dość spore możliwości, jeżeli chodzi o definicję QoS, ale niestety jest ograniczony do systemów opartych na linuxie, ponieważ mechanizmy definiujące jakość usługi bazują na *linux-htb* i *linux-hfsc*.

* + 1. **Podsumowanie**

W celu lepszego zwizualizowania różnic pomiędzy protokołami, ich zestawienie zostało przedstawione w poniższej tabeli.

|  |  |  |
| --- | --- | --- |
| **Kryterium porównania** | **OpenFlow** | **OVSDB** |
| Tagowanie VLAN | - Zrealizowane za pomocą akcji,  - Możliwe wszystkie operacje (dodawanie, edycja i usuwanie),  - Możliwość dodawania więcej niż jednego taga. | - Możliwość tworzenia różnych rodzajów portów, zezwalających na filtrowanie ruchu na różne sposoby,  - Dodawanie pola VLAN do nieotagowanego pakietu możliwe tylko w określonych scenariuszach. |
| Przypisywanie pakietów do EVC | - Przekierowywanie pakietów na daną ścieżkę możliwe przy pomocy zarówno tagów VLAN jak i bazując na portach. | - Możliwość stworzenia tunelu do miejsca docelowego. |
| Definicja QoS | - Możliwe ograniczanie prędkości przepływu. | - Możliwość ograniczenia prędkości na danych portach. |

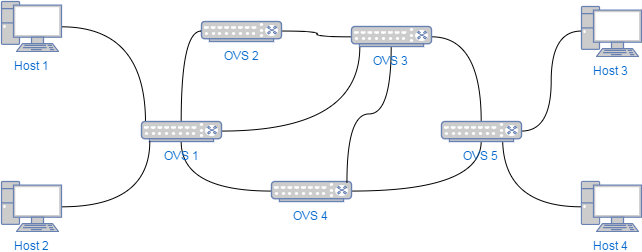
Rysunek 13: Teoretyczne porównanie protokołów

Z porównania wynika, że oba protokoły potrafią wykonywać wszystkie operacje zdefiniowane przez kryteria. Trzeba jednak zauważyć, że przypisywanie pakietów do EVC w wykonaniu protokołu OVSDB nie jest dobrym rozwiązaniem i nie pokrywa większości scenariuszy, jakie mogłyby być zdefiniowane przez dostawce usługi. W przypadku kryterium Tagowania VLAN również protokół OpenFlow oferuje większe możliwości, gdyż nie ma ograniczeń, co do konkretnych rozwiązań typu porty *access* czy *trunk.* W przypadku ostatniego kryterium w teorii oba protokoły wypadają w zestawieniu podobnie, gdyż oba potrafią ograniczyć ruch, tylko za pomocą innych sposobów. Podsumowując, oba protokoły mają duże możliwości, ale w tym konkretnym ich zestawieniu to OpenFlow przedstawia lepsze perspektywy.

* 1. **Różnice między protokołami w praktyce**
     1. **Mininet**

W celu porównania protokołów zasymulowana zostanie wirtualna sieć komputerowa za pomocą programu *mininet.* Wszystkie tworzone w ten sposób przełączniki są urządzeniami OVS. W tej pracy został użyty *mininet* w wersji 2.2.1, uruchomiony na wirtualnej maszynie Ubuntu 14.04 przygotowanej przez twórców tego oprogramowania.

W niniejszej pracy to porównania protokołów wymyślona została topologia przedstawiona na rysunku 14.



Rysunek 14: Topologia sieciowa

W programie topologię sieci definiuje się za pomocą skryptu napisanego w języku Python. Skrypt definiujący powyższą topologie wygląda następująco:

classMgrTopo(Topo):

"Topology prepared for POC"

def**\_\_init\_\_**(self):

# Initialize topology

Topo.**\_\_init\_\_**(self)

# Add hosts and switches

h1=self.addHost('h1')

h2=self.addHost('h2')

h3=self.addHost('h3')

h4=self.addHost('h4')

s1=self.addSwitch('s1')

s2=self.addSwitch('s2')

s3=self.addSwitch('s3')

s4=self.addSwitch('s4')

s5=self.addSwitch('s5')

# Add links

self.addLink(h1,s1)

self.addLink(h2,s1)

self.addLink(h3,s5)

self.addLink(h4,s5)

self.addLink(s1,s2)

self.addLink(s1,s3)

self.addLink(s1,s4)

self.addLink(s2,s3)

self.addLink(s3,s4)

self.addLink(s3,s5)

self.addLink(s4,s5)

Uruchomienie takiej topologii odbywa się za pomocą poniższego skryptu:

if**\_\_name\_\_**=='\_\_main\_\_':

controller\_ip='127.0.0.1'

OVSSwitch13=partial(OVSSwitch,protocols='OpenFlow13')

topo=PocTopo()

net=Mininet(topo,switch=OVSSwitch13,controller=partial(RemoteController,ip= controller\_ip,port=6633))

net.start()

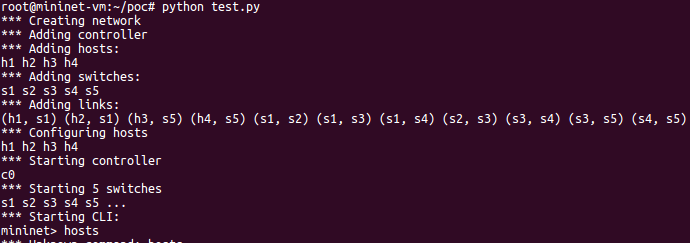
CLI(net)

net.stop()

Przy tworzeniu takiej sieci, trzeba podać adres IP kontrolera odpowiedzialnego za inicjalizowanie reguł OpenFlow. Ponadto wyspecyfikowana jest wersja OpenFlow, ponieważ bez tego domyślnie używana jest najniższa czyli 1.1.0. Tak przygotowane skrypty uruchamia się za pomocą polecenia:

~/mgr# python skrypt.py

Przy uruchamianiu skrypty mininet tworzy wirtualne przełączniki, komputery i połączenia między nimi oraz ustanawia połączenie z kontrolerem SDN. Po wszystkim uruchamiana jest konsola programu, dzięki której można bezpośrednio konfigurować wszystkie wirtualne urządzenia. Proces uruchamiania jest zwizualizowany na rysunku 15.

****

Rysunek 15: Start programu mininet.

Wirtualne komputery emulują działanie systemu, na jakim działa mininet, więc można je konfigurować za pomocą takich samych komend jak w systemie na którym uruchomiony jest mininet. Przykładowo, w celu zobaczenia konfiguracji pierwszego z emulowanych komputerów trzeba wykonać następującą komendę w konsoli:

mininet> h1 ifconfig

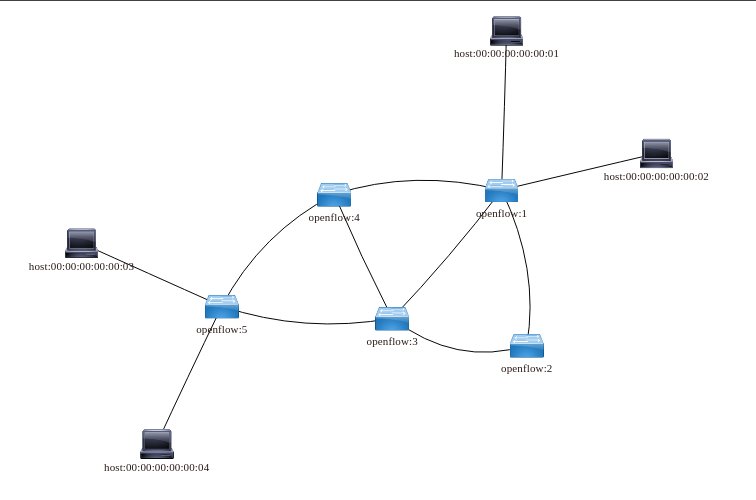
Oprócz tego można generować ruch za pomocą komendy ping, aby sprawdzić czy kontroler zainicjalizował w dobry sposób regóły OpenFlow.

* + 1. **Dostępne narzędzia do konfigurowania przełączników OVS**

Najprostrzym sposobem konfiguracji urządzeń OVS jest użycie programów dostarczanych wraz z nim. Używa się ich z linii poleceń. Te programy to:

* Ovs-vsctl – konfiguracja poprzez OVSDB,
* Ovs-ofctl – konfiguracja poprzez OpenFlow,

//TODO: Skończyć powyższe (co można za ich pomocą zrobić) oraz dalej trzeba napisać, że głównym sposobem komunikacji z przełącznikami są kontrolery SDN opisane w kolejnych rozdziałach.

* + 1. **OpenDayLight**//TODO: Krótki wstęp czym jest ODL, że jest postawiony na osgi (karaf), jakie posiada najlepsze funkcję, po czym jak się konfiguruje   
       //opis konfiguracji OVS przez OpenFlow i OVSDB  
       Po połączeniu sieci mininet z ODLem można sobie ją zwizualizować za pomocą komponentu dlux w przeglądarce internetowej:  
         
       
    2. **ONOX lub FloodLight**//TODO: Tutaj muszę się jeszcze zastanowić, którego kontrolera użyje, po rozpoznaniu, który ma lepsze wsparcie dla urządzeń OVS  
       //opis konfiguracji OVS przez OpenFlow i OVSDB
    3. **Podsumowanie**  
       Podsumowanie różnic między protokołami za pomocą kontrolerów

1. **Aplikacja wizualizująca wyniki**

W tym rozdziale opisze aplikację, która będzie wizualizowała wyniki. Najprawdopodobniej będzie to aplikacja napisana w Javie, postawiona na Springu, komunikująca się z urządzeniami OVS poprzez kontrolery SDN (za pomocą zawołań REST).

1. **Porównanie wyników**Porównanie (może tabelaryczne) protokołów OpenFlow i OVSDB na podstawie porównania teoretycznego i praktycznego.
2. **Podsumowanie**

Podsumowanie.

**Bibliografia**

# Paul Goransson, Chuck Black. Software Defined Networks: A Comprehensive Approach. Elsevier, 2014.

# Open vSwitch. Getting Started. <http://docs.openvswitch.org/en/latest/intro>, 2016. [dostęp: 2017-05-20].

# Rajdeep Dua. Open vSwitch deep dive. <https://www.slideshare.net/rajdeep/openvswitch-deep-dive>, 2013. [dostęp: 2017-05-20].

# D. Marschke, J. Doyle, P. Moyer. SDN: Anatomy of OpenFlow Volume I. Lulu, 2015.

# SdxCentral. What is Open vSwitch Database or OVSDB? <https://www.sdxcentral.com/cloud/open-source/definitions/what-is-ovsdb/>, [dostęp: 2017-05-21]

# B. Davie, The Open vSwitch Database Management Protocol, https://tools.ietf.org/html/draft-pfaff-ovsdb-proto-04, 2013, [dostęp: 2017-05-21]

# J. Kieffer, R. Satitoro, MEF-CECP Study Guide for Carrier Ethernet Professionals, Fujitsu Network Communications Inc., 2015.

# Open Networking Foundation, OpenFlow Switch Specification, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>, 2014, [dostęp: 2017-05-23]

# Open vSwitch, Open vSwitch Database Schema, <http://openvswitch.org/ovs-vswitchd.conf.db.5.pdf>, 2015, [dostęp: 2017-05-24]