Software Defined Networking

Dlaczego SDN (Software Defined Networking) ?

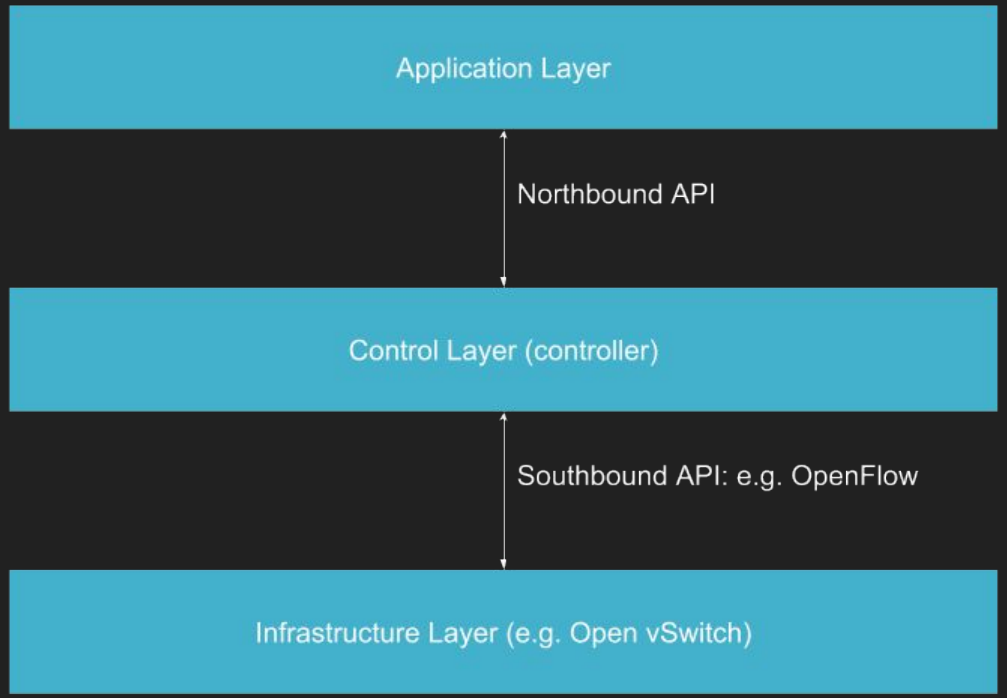
- Tradycyjne zarządzanie siecią jest skomplikowane, wymaga pamiętania o wielu rzeczach, znajomości wielu zagadnień.
- W SDN, centralizacja przez gromadzenie wielu informacji w jednym miejscu umożliwia podjęcie lepszych decyzji niż te podejmowane przez urządzenia sieciowe na podstawie danych lokalnych. Mówi się o programowaniu tzw. Globalnego widoku sieci (ang. Global view).
- SDN pozwala obniżyć koszty, jest elastyczne i skalowalne
  - Działanie na prostych (i tanich) urządzeniach
  - Uniwersalne urządzenia mogą spełniać różne funkcje (funkcja nie jest na stałe przypisana do danego urządzenia)
    - Wirtualizacja funkcji sieciowych (NFV - Network Function Virtualization)
    - Dana, zwirtualizowana funkcja (VNF - Virtual Network Function)

Docelowe warstwy w SDN

- Control program (SDN controller) - określa zachowanie komponentów, bazując na abstrakcyjnym opisie. Abstrakcyjnym tzn. niezależnym od producenta sprzętu obecnego w konfiguracji.
- Net virtualization (wirtualizacja sieci) - przekłada abstrakcyjny opis na globalny widok sieci.
- Network operating system (sieciowy system operacyjny) - przekłada globalny widok sieci na fizyczne urządzenia.

https://youtu.be/YHeyuD89n1Y?t=1469

Czym jest SDN?



Northbound - od kontrolera do aplikacji

Southbound - od kontrolera do infrastruktury (fizycznych urządzeń)

OpenFlow

- Protokół służący do wymiany informacji między kontrolerem, a urządzeniami w warstwie infrastruktury
- Urządzenia znają adres IP kontrolera, po uruchomieniu witają się z kontrolerem (wiadomość hello), uzgadniają numer wersji protokołu, którą będą się posługiwać w komunikacji między sobą. Później kontroler może modyfikować tabele zawierające reguły dotyczące pakietów które pojawiają się na urządzeniu. Tabele te nazywane są Flow tables.
- Wpis w tabeli - proaktywny, a reaktywny
  - Reaktywny - kontroler dynamicznie reaguje (podejmuje decyzję) co zrobić z danym pakietem
  - Proaktywny - kontroler z góry mówi urządzeniu co ma zrobić w danym przypadku
- Wpisy w tabeli analizowane są do pierwszego dopasowania, w kolejności wg priorytetu (atrybut liczbowy w rekordzie)
- Można zdefiniować wiele tabel (przydatne dla skomplikowanych reguł). Przeskok do innej tabeli jest realizowany przez dopasowanie do reguły w bieżącej tabeli, która zawiera instrukcję goto table x.
- Specyfikacje dla wszystkich wersji protokołu: https://www.opennetworking.org/software-defined-standards/specifications/

Środowisko laboratoryjne

Do symulacji sieci zarządzanej programowo wykorzystane jest środowisko wirtualizacyjne VirtualBox z dwoma maszynami wirtualnymi. Na jednej znajduje się Mininet, służący do symulacji warstwy infrastruktury (urządzenia sieciowe, hosty, połączenia między nimi) oraz program Wireshark (do monitorowania pakietów poruszających się w sieci). Na drugiej maszynie wirtualnej znajduje się kontroler SDN.

Instalacja Mininet na Virtualbox

- Instrukcje instalacji dostępne są również w oficjalnej dokumentacji http://mininet.org/download/
- Pobierz obraz maszyny wirtualnej z https://github.com/mininet/mininet/wiki/Mininet-VM-Images
- W VirtualBox (VBox): File -> Import Appliance (utworzenie maszyny o nazwie mininetVM)
- VBox: tools -> network -> create (zapamiętać nazwę utworzonej sieci)
- VBox: mininetVM -> settings -> network -> adapter 2: dodać host-only adapter (do utworzonej sieci) i sprawdzić adres MAC tego adaptera (advanced)
- VBox: uruchomić mininetVM (hasło/login = mininet/mininet)
- Sprawdzić czy interfejs z zapamiętanym adresem MAC ma przydzielony adres IP, jeśli nie: sudo dhclient [iinterfejs]
- Sprawdzić łączność host->mininetVM (ping) ; ssh -X -Y mininet@[adres IP] ; sprawdzić X forwarding (wireshark &)
- W razie problemów zainstalować środowisko graficzne na mininetVM (np. sudo apt-get install xinit lxde). Uruchomienie środowiska graficznego: startx
- Zainstalować guest additions i dopasować rozdzielczość do rozmiaru okna
    - Najprościej: sudo apt-get install virtualbox-guest-utils
    - Alternatywnie: przez ściągnięcie obrazu .iso
- VBox: mininetVM -> Snapshots -> take (ustawienie punktu przywracania)

Instalacja kontrolera (ryu)

- Ściągnąć obraz dysku instalacyjnego dla wybranego systemu operacyjnego (np. Ubuntu server lub Ubuntu desktop)
- VBox: Machine->New (nowa maszyna ryuVM)
- …
- Zdefiniować host-only adapter analogicznie jak w przypadku mininetVM
- Zainstalować Python: sudo apt-get python-dev python-pip
- Zainstalować kontroler: sudo pip install ryu
- VBox: ryuVM -> Snapshots -> take (ustawienie punktu przywracania)

Instalacja kontrolera (OpenDaylight)

- Ściągnąć obraz dysku instalacyjnego dla wybranego systemu operacyjnego (np. Ubuntu server lub Ubuntu desktop)
- VBox: Machine->New (nowa maszyna odlVM)
- …
- Zdefiniować host-only adapter analogicznie jak w przypadku mininetVM
- Ściągnąć i rozpakować zip z kontrolerem ze strony https://docs.opendaylight.org/en/latest/downloads.html
- Zainstalować java8 (ważne! Problem z nowszymi wersjami Javy)
  - sudo add-apt-repository ppa:webupd8team/java
  - sudo apt-get update
  - sudo apt-get install oracle-java8-installer
  - sudo apt-get install oracle-java8-set-default
- Zrestartować maszynę, uruchomić kontroler: cd ścieżka_do_rozpakowanego_archiwum , ./bin/karaf
  - feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-all
- VBox: odlVM -> Snapshots -> take (ustawienie punktu przywracania)

Weryfikacja działania środowiska

- Sprawdzić łączność host -> mininetVM ; mininetVM -> ryuVM
- ryuVM:
  - cd /usr/local/lib/python2.7/dist-packages/ryu/app
  - ryu-manager gui_topology/gui_topology.py simple_switch.py --observe-links
- Host:
  - Przeglądarka internetowa: adres IP ryuVM:8080
- mininetVM:
  - sudo mn
  - Exit
  - sudo mn -c
  - sudo mn --controller=remote,ip=[ip kontrolera] --topo=linear,5 --mac

Mininet c.d.

Sprawdź działanie komend (na mniejszych topologiach): help, nodes, net, dump

Jeżeli coś pójdzie nie tak, można wyczyścić mininet wykonując: sudo mn -c (cleanup)

Wykonanie polecenia na danej maszynie: [nazwa maszyny] [polecenie]. Np. h1 ip a, h1 ping h2

Wygenerowanie ruchu między wszystkimi urządzeniami (może być potrzebne do odkrycia topologii): pingall

Tylko sieć podlega wirtualizacji w Mininet, system plików i przestrzeń procesów są współdzielone.

Porównaj: h1 ps -a , h2 ps -a , s1 ps -a ; h1 ls, h2 ls , s1 ls

Konfiguracja połączeń między urządzeniami: link h1 s1 up ; link h1 s1 down

Wybranie przełącznika i protokołu: --switch ovsk,protocols=OpenFlow13

Jak wyglądają różne topologie

Obserwuj zmiany w interfejsie graficznym kontrolera i sprawdzaj stan sieci poleceniami dump oraz net.

--topo=linear,5 --topo=linear,40

--topo=tree,depth=3,fanout=3 --topo=tree,depth=10,fanout=4

--topo=minimal

--topo=single --topo=single,10

--topo=reversed --topo=reversed,7

--topo=torus,3,3 --topo=torus,4,4

Monitorowanie ruchu aplikacyjnego

- Uruchom wireshark na mininetVM (intefejs: any, filtr udp)
- Uruchom topologię z kilkoma maszynami
- Uruchom w tle program xterm na maszynach h1 i h2
- Na h1 uruchom nasłuchiwanie na udp (port 1234): nc -l -u 1234
- Sprawdź adres ip h1
- Na h2 połącz się z serwerem udp na h1: nc -u [adres ip h1] 1234
- Prześlij dane w obie strony, obserwuj pakiety w Wiresharku

Monitorowanie ruchu OpenFlow

- Uruchom wireshark na mininetVM (intefejs: any, filtr of)
- Uruchom topologię z kilkoma maszynami

Jakie pakiety OpenFlow są widoczne?

Ile pakietów hello zostało przesłanych, dlaczego właśnie tyle?

Polecenia OpenVSwitch

OpenVSwitch: openvswitch.org

Na mininetVM:

- sudo ovs-vsctl show
- sudo ovs-ofctl dump-flows s1 //Openflow 1.0
- sudo ovs-ofctl dump-ports s1 //Openflow 1.0
- sudo ovs-ofctl -O OpenFlow13 dump-flows s1 //Openflow 1.3
- sudo ovs-ofctl -O OpenFlow13 dump-ports s1 //Openflow 1.3

Korzystając z powyższych poleceń zaobserwuj zmiany w tabelach Openflow na urządzeniach (zaraz po uruchomieniu topologii, po wykonaniu kilku poleceń ping).

Zaawansowane topologie w Mininet

Mininet umożliwia tworzenie dowolnych topologii, wykorzystując API w języku Python.

https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

- Skopiuj kod z pierwszego przykładu.
- Uruchomienie funkcji testującej: sudo python [nazwa_pliku.py]
- Kluczowe informacje:
  - Nowa topologia rozszerza klasę Topo (mininet.topo)
  - Tworzenie topologii zaczyna się w metodzie build (parametr n to tutaj liczba hostów, które będą wygenerowane)
  - Dodanie nowych urządzeń/hostów (self.addSwitch, self.addHost)
  - Dodanie nowego połączenia (self.addLink)
  - Uruchomienie własnej topologii:
    - Dodaj słownik topos, (np. Na końcu pliku py): topos = { ‘nazwa’ : NazwaKlasyTopo }
    - Uruchom mininet: sudo mn --custom plik.py --topo nazwa
- Korzystając z informacji dostępnych w powyższym linku, zaimplementuj topologie wskazane przez prowadzącego.

Implementacja własnego kontrolera (ryu)

Lub innymi słowy, aplikacji w Ryu network operating system.

Dokumentacja: https://ryu.readthedocs.io/en/latest/index.html

- Przeanalizuj prosty przykład ze strony https://ryu.readthedocs.io/en/latest/writing_ryu_app.html
- Kluczowe informacje:
  - Nowy pakiet (typu packet_in) jest przechwytywany przez metodę packet_in_handler, zwróć uwagę na zastosowany dekorator (@set_ev_cls), to właśnie on powoduje wywołanie tej metody przy zdarzeniu packet_in.
  - Zobacz jak w przykładzie został skonstruowany pakiet wychodzący oraz jak można taki pakiet wysłać
  - Pamiętaj, że ryu jest aplikacją jednowątkową, która działa w oparciu o zdarzenia. Uwaga na wywoływanie operacji blokujących!
- Zaimplementuj:
  - Filtrowanie ruchu danego typu (np. Blokowanie udp)
  - Gromadzenie prostych statystyk (np. Liczby otrzymanych wiadomości hello)
  - Udostępnienie listy połączonych przełączników i hostów