**Programowalne Sieci Komputerowe**

Dokumentacja końcowa projektu

Piotr Wieżel  
Natalia Jakubiak

# Różnice między dokumentacją wstępną a końcową

Różnice są opisane poprzez skreślenie zdania z poprzedniej dokumentacji i dopisania aktualizacji ukazującej stan rzeczywisty.

## Treść zadania

* + Skonstruować równoważnik obciążenia zgodnie ze specyfikacją podaną w następnych podpunktach:
  + Sieć składa się z dwóch hostów: h1 i h2 oraz 4 przełączników: s1, s2, s3, s4. Przełączniki sterowane są zewnętrznym kontrolerem Ryu. Hosty h1 i wysyłają do siebie nawzajem zapytania ICMP. Pakiety są kierowane albo górną, albo dolną ścieżką.
  + ~~Środkowe przełączniki s4 i s3 liczą pakiety, które przez nie przechodzą.~~ Przełączniki wysyłają do sterownika powiadomienia o przyjściu pakietu, a sterownik zlicza pakiety, które przyszły do przełączników s3 i s4. Gdy liczniki osiągną określony poziom, ~~przełączniki wysyłają powiadomienie do kontrolera, który w reakcji na to~~ sterownik wysyła aktualizacje tablicy przepływów do przełączników s1 i s2 tak, aby zapytania były kierowane alternatywną ścieżką.
  + Hosty h1 i h2 używają adresacji IPv6.

## Przyjęte założenia

* + Przełączniki nie posiadają żadnej zerowej wiedzy, jak mają kierować pakiety. Dowiedzą się o tym dopiero od kontrolera przy nadejściu pakietu (routing reaktywny).
  + Hosty h1 i h2 nie znają topologii sieci. Znają adresy IP drugiej strony.
  + Kontroler Ryu i emulowana sieć Mininet znajdują się na osobnych maszynach wirtualnych
  + Przełączniki komunikują się z kontrolerem zewnętrznym, korzystając z połączenia sieciowego pomiędzy dwiema maszynami wirtualnymi (Mininet-Ryu)

## Wybrane środowisko narzędziowe:

* emulator: **Mininet** do emulowania sieci SDN,
* sterownik: **Ryu**
* środowisko wirtualizacji: **Oracle VirtualBox**
* repozytorium kodu: **GitHub**

## Architektura rozwiązania

Topologia sieci Mininet:

  
rys.1 Topologia sieci Mininet

Struktura logiczna systemu:

rys.2 Połączone maszyny wirtualne Mininet i kontrolera Ryu

Oracle VirtualBox

sieć

wewnętrzna

mininet-VM

ryu-VM

Mininet i Ryu są uruchomione jako osobne maszyny wirtualne **Oracle VirtualBox**’a i znajdują się się w tej samej podsieci. Mininet będzie korzystał z Ryu jako sterownika zewnętrznego. Możliwe jest dostanie się do maszyny z mininetem z maszyny z kontrolerem przez ssh.

## Sposób demonstracji wyników

* + Scenariusze:
    - Wysłanie 20 zapytań ICMP z hosta h1 do hosta h2, przełączenie na alternatywną ścieżkę po zliczeniu 5 pakietów przez środkowe przełączniki
    - Wysłanie 20 zapytań ICMP z hosta h2 do hosta h1, przełączenie na alternatywną ścieżkę po zliczeniu 5 pakietów przez środkowe przełączniki
  + Prezentacja wyników:
    - Logi zebrane z kontrolera Ryu, mówiące o m.in. przełączaniu ścieżek na alternatywne
    - Obserwacja pakietów programem śledzącym lub wyświetlanie ich w analogiczny sposób w terminalu
    - Podejrzenie stanów tablic przepływów danych przełączników przy pomocy poleceń OpenVSwitch’a, np. ovs-ofctl

## Wyniki do zaprezentowania w czasie odbioru częściowego

* + Pokazanie logów z kontrolera Ryu podczas prób wysyłania zapytań ICMP z hosta h1 do h2 i odwrotnie
  + Wykonanie podstawowych komend ovs-ofctl: ręczne uzupełnianie/usuwanie wpisów w tablice przepływów przełączników

## Podział prac w zespole

* + Natalia Jakubiak – skonfigurowanie środowiska. Sprawienie, że hosty h1 i h2 będą posługiwały się adresacją IPv6
  + Piotr Wieżel – stworzenie topologii sieci. Napisanie części aplikacji dla kontrolera Ryu, polegającej na równoważeniu obciążenia i zliczaniu pakietów danego typu.

# Projekt końcowy

## Opis konstrukcyjny i użytkowy zbudowanych modułów.

W skład projektu wchodzą dwa moduły:

* Aplikacja Mininet, która emuluje całą sieć wczytuje jej topologię z pliku topo-square.py. Napisany przez nas plik powoduje stworzenie w emulatorze topologii takiej, jak na Rys.1. Składa się z jednej klasy: MyTopo, która dziedziczy po klasie Topo z modułu mininet.topo.
* Aplikacja ping.py uruchomiona na kontrolerze Ryu. Zawiera ona w sobie jedną klasę ICMPResponder, a w niej logikę kontrolera razem ze statyczną tablicą routingu. W tablicy routingu znajdują się wpisy podstawowe dla wszystkich przełączników oraz dodatkowe wpisy dla przełączników s1 i s2, gdy ruch przeładowywany jest do przełącznika s4.

Na początku należy uruchomić aplikację kontrolera poleceniem ryu-manager ping.py.

Następnie trzeba włączyć emulator z żądaną topologią poleceniem:  
 sudo mn --custom ./topo-square.py --topo squaretopo --controller remote,ip=192.168.34.3 --mac --switch ovsk   
z konsoli maszyny Mininet, w katalogu, gdzie znajduje się plik topo-square.py.

Sterownik wykonuje wówczas następujące czynności:

* Wymienia z przełącznikami wstępne informacje dotyczące parametrów połączenia z kontrolerem, oraz jego zdolności poprzez reakcję na zdarzenie ofp\_event.EventOFPSwitchFeatures wywołaniem funkcji switch\_features\_handler.
* W funkcji switch\_features\_handler kontroler odsyła do przełącznika, który nadesłał mu pakiet, polecenie wstawienia przepływu do tablicy przepływów: „kieruj wszystkie pakiety do kontrolera” (patrz: reguła nr 1)
* Następnie przy próbie wysłania pingów (czy to IPv4 czy IPv6) przez hosty h1 lub h2, przekazywane są na początku żądania protokołu ARP do zmapowania adresów IP na adresy MAC. Przełączniki przekazują te żądania do kontrolera, a ten odpowiada im poleceniem likwidacji przepływu nr 1 (funkcja remove\_controller\_flow) oraz wpisuje nowy przepływ, nr 2 (funkcja add\_mac\_src\_flow). Dzieje się to w reakcji na zdarzenie ofp\_event.EventOFPPacketIn, które powoduje wywołanie funkcji packet\_in\_handler.
* Kontroler ustawia flagę dla każdego przełącznika osobno, mówiącą o wysłaniu początkowych wpisów do tablic przepływów.
* Przełączniki wysyłają pakiety do kontrolera, a ten zlicza je dla przełączników s3 i s4. Gdy jeden z liczników przekroczy wartość 5, jest zerowany, a do przełączników s1 i s2 są wysyłane polecenia modyfikacji przepływów (funkcja change\_offload) zgodnie ze statyczną tablicą routingu.

Podjęcie decyzji o przeładowywaniu ruchu na inny port dla przełączników s1 i s2 jest wykonywane podczas obsługi zdarzenia ofp\_event.EventOFPPacketIn. Wybierany jest wówczas port wyjściowy (funkcja choose\_output\_port), na który przełączniki powinny przekazać przychodzące pakiety.

## Opis reprezentatywnych reguł zapisywanych w tablicach przepływów przełączników

Reguły zostały podane w tabeli.

|  |  |  |  |
| --- | --- | --- | --- |
| L.p. | Dopasowanie | Akcje | Wysyłane do przełączników: |
| 1 | każdy pakiet | prześlij pakiet do kontrolera | s1, s2, s3, s4 |
| 2 | źródłowy adres MAC = src\_mac | przekaż pakiet do kontrolera &  przekaż pakiet na odpowiedni port wyjściowy | s1, s2, s3, s4 |

## Opis przeprowadzonych testów.

W ramach sprawdzania wykonaliśmy 3 testy:

* poleceniem pingall w Mininecie sprawdziliśmy podstawową osiągalność hostów po IPv4
* polecenie h1 ping h2 –c 20 oraz h2 ping h1 –c 20 wykonały po 20 zapytań ICMP (po IPv4)
* polecenie h1 ping6 –I h1-eth0 –c 20 <adres\_IPv6\_h2> oraz h2 ping6 –I h2-eth0 –c 20 <adres\_IPv6\_h1>

## Refleksje na temat realizacji projektu

Projekt pozwala na zapoznanie się ze szczegółami sterownika Ryu i ugruntowanie w praktyce teoretycznej wiedzy zdobytej na wykładzie przedmiotu. W trakcie realizacji pojawiły się pewne przeszkody:

* Początkowo zainstalowana wersja emulatora Mininet działała na OpenVSwitch wersji 1.9, która obsługiwała tylko protokół OpenFlow wersji 1.0, w której nie występują mierniki. Konieczne było zaktualizowanie Mininet’a do wersji z OpenFlow v1.3, co sprowadziło się do aktualizacji OpenVSwitch’a do wersji 2.3.
* Okazało się, że mierniki nie pozwalają mierzyć i wysyłać powiadomienia o liczbie pakietów przesłanych przez dany port, ale dane związane z obecną szybkością ruchu na portach itd. Postanowiliśmy więc przesyłać każdy pakiet do kontrolera i w ten sposób je zliczać.

Ryu ma dość duży próg wejścia, ponieważ w dokumentacji mało jest przykładów (przydałoby się więcej, niż 1 do każdego podpunktu dokumentacji, aby uczący się zrozumiał, jak używać określonych modułów). Tym bardziej, napisanie podstawowej aplikacji w nim i przejście przez proces uczenia się tego kontrolera daje dużą satysfakcję.

## Oszacowanie czasochłonności projektu w roboczogodzinach

* instalacja i konfiguracja środowiska: 12
* nauka środowiska i narzędzi: 20
* tworzenie rozwiązania: 9
* testowanie rozwiązania: 2
* opracowywanie dokumentacji: 3

## Wykaz załączonych plików źródłowych.

1. ping.py – Aplikacja sterownika Ryu
2. topo-square.py – Topologia emulatora Mininet