# Задания по курсу “Дополнительные главы по Компьютерным Сетям”: раздел SDN/NFV. ВМК МГУ, 2015

*Лекторы: Шалимов А.В., Смелянский Р.Л.*

## Contents

# Введение

В данном документе представлены задания по курсу “Дополнительные главы по Компьютерным Сетям” – раздел SDN/NFV. Задание заключается в разработке приложения для OpenFlow контроллера Runos.

Каждому слушателю выдается свой вариант задания вида **A:{i;k;t}**, где:

- A – номер приложения, которое требуется реализовать;
- i - номер подпункта для реализации в первом списке в разделе варианты.
- k – номера подпунктов для реализаци во втором списке в разделе варианты.
- t – номер топологии, которую нужно использовать в разделе варианты.

Например, вариант **2:{1;3,4;2}** означает, что нужно выполнить задание биллинг (номер 2), идентификация клиентов происходит по MAC адресу (пункт 1), поддерживаемые протоколы Telnet и UDP (пункты 3 и 4), топология номер 2.

В качестве исключения можно придумать свое задание, предварительно его согласовав.

**Требования**:

- В ходе выполнения задания нужно реализовать и предоставить:
  - приложение для контроллера Runos,
  - скрипт на Mininet для задания, указанной в варианте топологии.
- Код должен быть универсальным, без хардкодинга, работать в большинстве случаев.
- Сдавать задания по почте лектору одним архивом tar.gz, содержащим:
  - патч с вашими изменениями к контроллеру (.patch),
  - скрипт на Mininet (.py),
  - текстовый файл, описывающий проверочные сценарии для работы приложения.
    - Пример, запустить контроллер с конфигурационным файлом номер 1, запустить скрипт topo.py, запустить h1 ping h2, оба хоста пингуют друг друга, запустить iperf h1 h2, соединение не устанавливается, потому что блокируется приложением на контроллере, что подтверждается логами.
- Использовать контроллер Runos версии 0.5:
  - Github https://github.com/ARCCN/runos.
- Вопросы по заданию лектору на почту (лучше заранее согласовать, что будете делать), а вопросы по Runos в список рассылки http://groups.google.com/d/forum/runos-ofc.
- Если нашли какой-то баг ☺, то официально описываем его на github issues https://github.com/ARCCN/runos/issues.
- Использовать Mininet версии 2.2.1, который нужно собирать из исходников.
- Можно разрабатывать в предыустановленной VM http://bit.ly/runos-vm-latest.

**Дополнительно:**

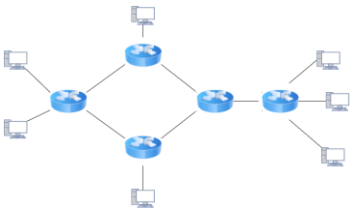
- Для генерации трафика можно использовать утилиты iperf, ping, ssh, scp, HTTP server в Mininet, произвольный udp пакет можно послать командой *echo “hello” > dev/udp/192.168.2.101/58549*.

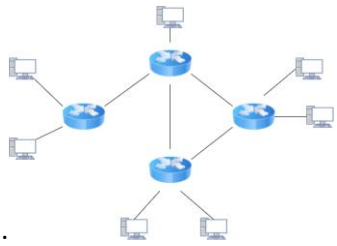
# 1. Контроль доступа

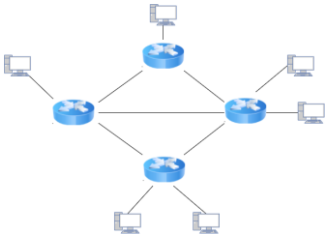
- Приложение определяет, к каким ресурсам и на каком уровне пользователь имеет доступ.
    - Для сети заранее задаются следующие политики доступа:
        - Список узлов/пользователей, которым разрешен доступ в сеть.
            - Если доступ не разрешен, нужно отправлять пользователю ICMP error.
        - По умолчанию все разрешенные пользователи имеют доступ к некоторому перечню ресурсов (список хостов).
        - Так же для каждого пользователя дополнительно задано:
            - список ресурсов, к которым он имеет дополнительный доступ и по каким протоколам (список хостов, протоколы, порты);
            - Количество одновременных сессий/потоков.

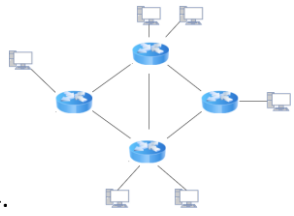
Варианты:

- способ идентификации клиентов
    1. Mac
    2. IP
- протоколы (в скобках указаны используемые ими TCP-порты)
    1. HTTP (80, 8080)
    2. SSH (22)
    3. Telnet (23)
    4. UDP (5678)
    5. ICMP
    6. ARP
- топология

**1.** 

2. 

3. 

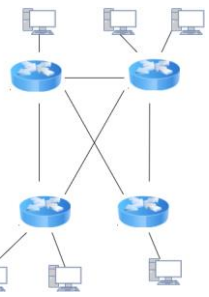
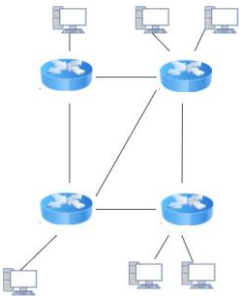
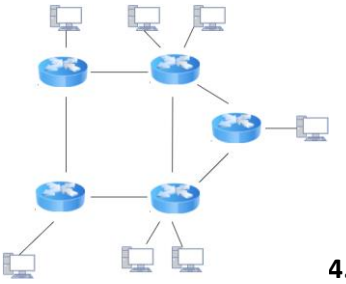
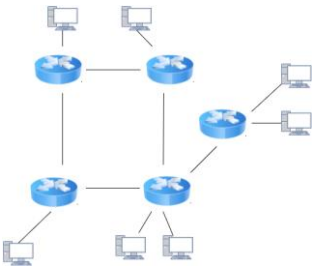
4. 

## 2. Биллинговая система

- Приложение отвечает за сбор информации об использовании телекоммуникационных услуг и их тарификацию. Хостам разрешено использование конкретных сетевых протоколов (ssh, ftp, http, etc), причем для каждого типа стоит ограничение по количеству входящего трафика. Контроллер должен осуществлять проверку по каждому типу трафика для каждого клиента и, в случае превышения лимита, ограничивать им доступ на некоторый промежуток времени (два параметра – объем разрешенного трафика (байты) и промежуток времени (секунда)).
- В конфигурационном файле для каждого клиента (или подсети) выделяется набор разрешенных протоколов и ограничения по каждому из них. Контроллер формирует соответствующие правила на ближайших к хостам коммутаторах, которые пропускают только заданный в конфигурационном файле трафик.
- Также необходимо собирать статистику по каждому типу трафика и, в случае превышения лимита, запрещать клиенту использовать этот тип трафика в течение некоторого периода времени (указанного в конфигурационном файле). Запрет использования протокола аналогичен удалению соответствующего правила на коммутаторе.

Варианты:

- способ идентификации клиентов
  1. Mac
  2. IP
- Протоколы
  1. HTTP (80, 8080)
  2. SSH/SCP (22)
  3. Telnet (23)
  4. UDP
- топология

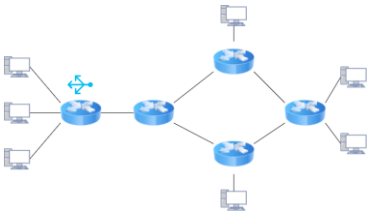
**1.**  **2.**  **3.**  **4.** 

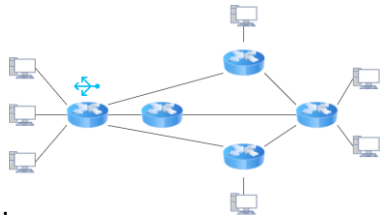
## 3. Балансировка нагрузки

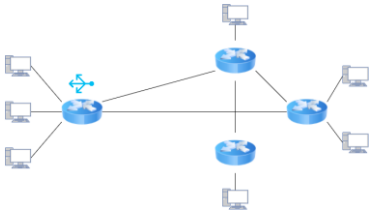
- Приложение обеспечивает распределение нагрузки между несколькими серверами с целью оптимизации использования ресурсов сервера и полосы пропускания, сокращения времени обслуживания запросов, а также обеспечения отказоустойчивости.
- В конфигурационном файле задается балансируемый сервер и порты, между которыми идет балансировка. Также в файле указывается, ip и mac адреса серверов, находящихся за каждым из балансируемых портов.
- Когда клиент обращается на один из балансируемых адресов, контроллер, согласно своим правилам балансировки, перенаправляет трафик клиента к выбранному серверу (подмену подмену mac и ip-адресов получателя на адрес выбранного сервера) и обратно (выполняя обратную замену). По факту получается, что клиент даже не знает, что его запрос обработал другой сервер. Контроллер реактивно формирует правила балансировки на коммутаторе, приходящих соединений от клиентов.
- В случае отключения порта коммутатора нужно перераспределять нагрузку. Так же отрабатывается idle_timeout, когда клиент перестал общаться с сервером. Для каждого сервера задано максимальное число обрабатываемых запросов.

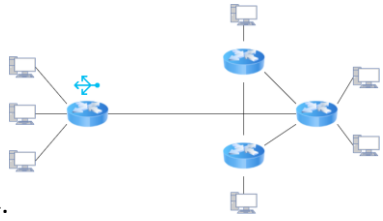
Варианты:

- балансируемые протоколы
    1. UDP
    2. TCP
- схема балансировки
    1. Weighted round-robin (для каждого сервера задано некоторое число, система перебирает сервера по кругу, отправляя на текущий сервер заданное число запросов, а потом только переходит на следующий)
    2. Weighted random (для каждого сервера задана вероятность его выбора)
    3. Dynamic round-robin (нужно обеспечить равномерное распределение нагрузки (по числу потоков), динамически поддерживать число потоков и перераспределять новые запросы на менее загруженный сервер)
- Топология (сервера находятся слева, балансировка на отмеченном свитче)

1. 

2. 

3. 

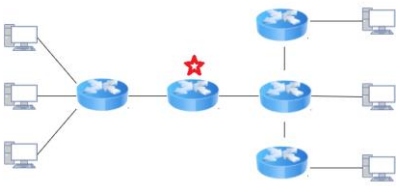
4. 

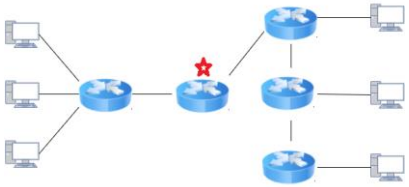
## 4. Трансляция адресов (NAT)

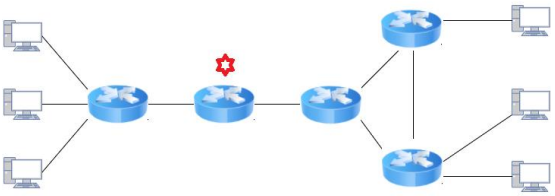
- Приложение реализует механизм преобразования локальных IP-адресов проходящих через него потоков на пул меньшего числа глобальных адресов IP-адресов.
- Примерную схему работу в OpenFlow можно описать следующим образом. Для каждого пакета из внутренней сети происходит поиск соответствия в таблице потоков. Поиск ведется по 5 полям пакета: ip_src, ip_dst, type, port_src, port_dst. В таблицы хранятся правила трансляция: указаны ip_src и port_src, которые должны быть заменены при трансляции. Если соответствие нашлось, то происходить перезапись полей пакета и отправление его на физический порт. Если соответствие не было найдено, то пакет отправляется на контроллер (PacketIn). На контроллере согласно алгоритмам трансляции происходит выбор ip_dst и port_src. Правила трансляции отправляются на контроллер (flowmod). Процесс обработки пакетов из внешней сети выглядит так же за одним исключением, что если правила в таблице потоков нет, то пакет сбрасывается.
- Конфигурационный файл включает в себя пул внешних IP адресов, схему трансляции. Приложение должно удалять неиспользуемые правила по таймаутам.

Варианты:

- Протокол, который нужно транслировать:
    1. UDP (http://tools.ietf.org/html/rfc4787)
    2. TCP (https://tools.ietf.org/html/rfc5382)
- Схема отображения (пояснения по адресу http://bit.ly/1MrsIll)
    1. Независимо от получателя
    2. В зависимости от адреса получателя
    3. В зависимости от адреса и порта получателя
- Топология (трафик идет слева, внешняя сеть справа, NAT на отмеченном свитче)

1. 

2. 

3. 

4.