SDN: Упражнения

SDN: Упражнения

Упражнение 1

MiniNet - это среда моделирования, которая создает реалистичную виртуальную сеть, реальный код ядра, коммутатора и приложения на одной машине (виртуальной, облачной или собственной) с помощью одной команды:

sudo mn

SDN: Упражнения

Упражнение 1

Скачать виртуальную машину **mininet VM**
http://mininet.org/download
Существует 32-битный и 64-битные образы ВМ
с предустановленной последней версией **mininet**

~800Мб

SDN: Упражнения

Упражнение 1

Скачать систему виртуализации:
Можно использовать любую систему виртуализации,
но рекомендуется установить **VirtualBox**.
Он бесплатный и работает на **Windows**, **Linux** и **OS X**.
https://www.virtualbox.org/wiki/Downloads

SDN: Упражнения

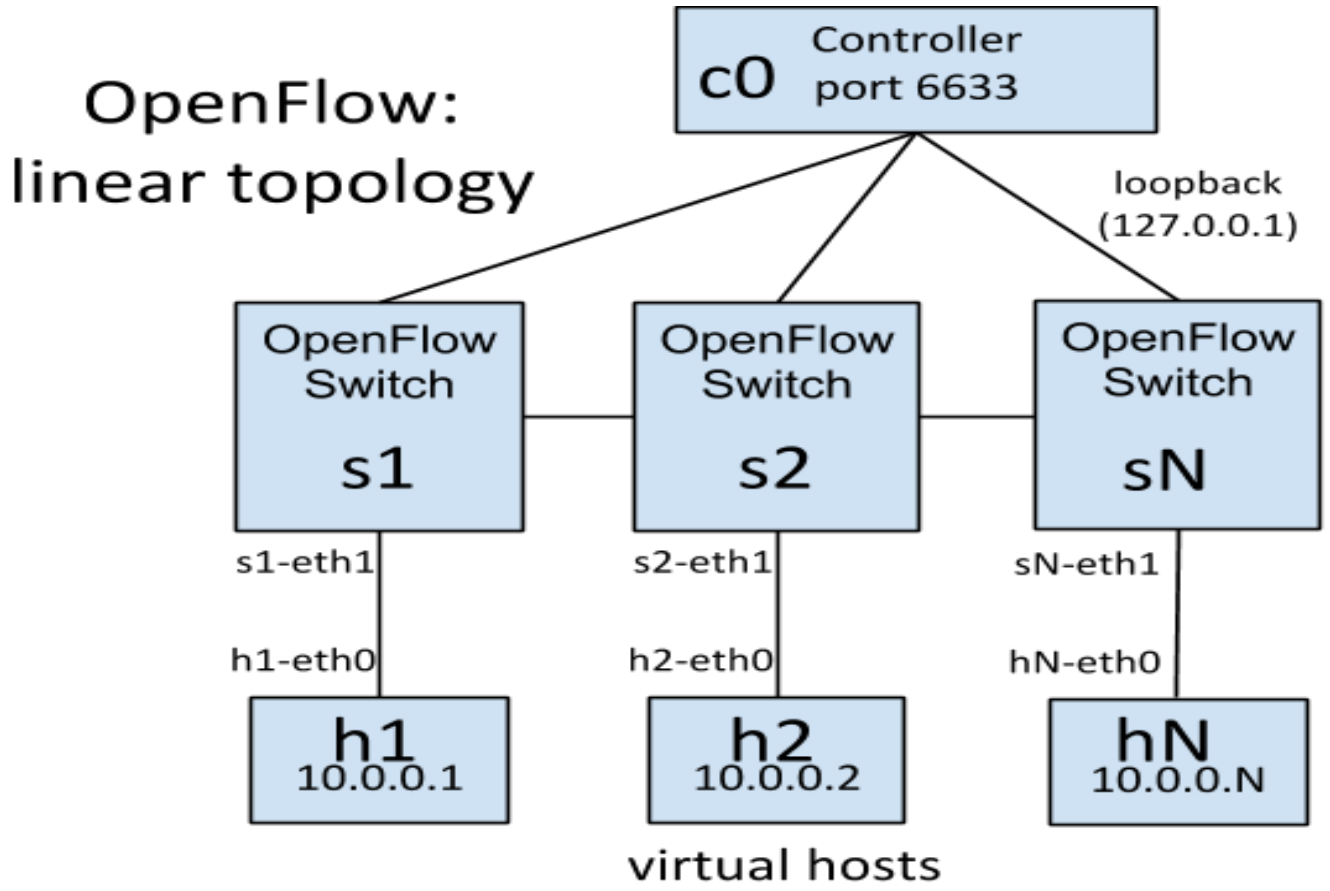
- Упражнение 2

В этом упражнении вы научитесь создавать собственные топологии с использованием **API**-интерфейса **Mininet Python**, а также, как отдельные параметры, такие как **пропускная способность**, **задержка**, **затухание** и **номер очереди** могут быть установлены индивидуально для разных связей в топологии.

Вы также узнаете, как выполнять тестирование производительности этих настраиваемые топологии с использованием **ping** и **iperf**.

SDN: Упражнения

- Упражнение 2



SDN: Упражнения

- Упражнение 2

Создание топологии

Mininet поддерживает различные **топологии**. С помощью нескольких строк кода **Python** вы можете создать гибкую **топологию**, которая может быть настроена на основе параметров, которые вы передаете в нее, и повторно использоваться для нескольких экспериментов.

Например, простая топология сети (рисунок 1) состоит из определенного числа хостов (от h1 до hN), подключенных к коммутаторам (от s1 до sN)

Linear Topology (with Performance Settings)

```
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import CPULimitedHost
from mininet.link import TCLink
from mininet.util import irange,dumpNodeConnections
from mininet.log import setLogLevel

class LinearTopo(Topo):
   "Linear topology of k switches, with one host per switch."

   def __init__(self, k=2, **opts):
       """Init.
           k: number of switches (and hosts)
           hconf: host configuration options
           lconf: link configuration options"""

       super(LinearTopo, self).__init__(**opts)

       self.k = k

       lastSwitch = None
       for i in irange(1, k):
           host = self.addHost('h%s' % i, cpu=.5/k)
           switch = self.addSwitch('s%s' % i)
```

```
            # 10 Mbps, 5ms delay, 1% loss, 1000 packet queue
            self.addLink( host, switch, bw=10, delay='5ms', loss=1,
max_queue_size=1000, use_htb=True)
            if lastSwitch:
                self.addLink(switch, lastSwitch, bw=10, delay='5ms', loss=1,
max_queue_size=1000, use_htb=True)
            lastSwitch = switch

def perfTest():
   "Create network and run simple performance test"
   topo = LinearTopo(k=4)
   net = Mininet(topo=topo,
                 host=CPULimitedHost, link=TCLink)
   net.start()
   print "Dumping host connections"
   dumpNodeConnections(net.hosts)
   print "Testing network connectivity"
   net.pingAll()
   print "Testing bandwidth between h1 and h4"
   h1, h4 = net.get('h1', 'h4')
   net.iperf((h1, h4))
   net.stop()

if __name__ == '__main__':
   setLogLevel('info')

   perfTest()
```

Some important methods and parameters:

`self.addHost(name, cpu=f)`: This allows you to specify a fraction of overall system CPU resources which will be allocated to the virtual host.

`self.addLink( node1, node2, bw=10, delay='5ms', max_queue_size=1000, loss=1, use_htb=True)`: adds a bidirectional link with bandwidth, delay and loss characteristics, with a maximum queue size of 1000 packets using the Hierarchical Token Bucket rate limiter and netem delay/loss emulator. The parameter bw is expressed as a number

```
if __name__ == '__main__':
   setLogLevel('info')

   perfTest()
```

Some important methods and parameters:

`self.addHost(name, cpu=f)`: This allows you to specify a fraction of overall system CPU resources which will be allocated to the virtual host.

`self.addLink( node1, node2, bw=10, delay='5ms', max_queue_size=1000, loss=1, use_htb=True)`: adds a bidirectional link with bandwidth, delay and loss characteristics, with a maximum queue size of 1000 packets using the Hierarchical Token Bucket rate limiter and netem delay/loss emulator. The parameter bw is expressed as a number in Mb/s; delay is expressed as a string with units in place (e.g. '5ms', '100us', '1s'); loss is expressed as a percentage (between 0 and 100); and max_queue_size is expressed in packets.

SDN: Упражнения

Упражнение 2

Запуск **Mininet**:

- Создайте сценарий **LinearTopo.py** на виртуальной машине **Mininet** и скопируйте содержимое Linear Topology (без настроек производительности).
- Сделайте скрипт исполняемым
  $ chmod u + x LinearTopo.py
- Выполните скрипт
  $ sudo ./LinearTopo.py

Output

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s1, s2) (s2, s3) (s3, s4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 4 switches
s1 s2 s3 s4
Dumping host connections
h1 h1-eth0:s1-eth1
```

```
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
Testing network connectivity
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (0/12 lost)
*** Stopping 4 hosts
h1 h2 h3 h4
*** Stopping 4 switches
s1 ...s2 ....s3 ....s4 ...
*** Stopping 1 controllers
c0
*** Done
```

SDN: Упражнения

Упражнение 2

Сети **ЦОД** обычно имеют **топологию** перевернутого дерева.

Конечные **хосты** подключаются к **коммутаторам** верхней части стойки, которые образуют листья дерева;

На верхнем уровне установлены один или несколько корневых **коммутаторов.**

Один или несколько слоев промежуточных **коммутаторов** образуют ветви дерева.

В базовой топологии дерева каждый **коммутатор** (кроме корневого) имеет родительские **коммутаторы**.

Дополнительные **коммутаторы** и **линки** могут быть добавлены для создания более сложной **топологии** дерева (например, ***fat tree***), чтобы улучшить отказоустойчивость или увеличить пропускную способность между стойками.

Задача - создать простую топологию дерева.

Предполагается, что каждый уровень, состоит из одного уровня коммутаторов / хостов с настраиваемым значением разветвления (k).

Например, простая древовидная сеть, имеющая один слой на каждый уровень и разветвление 2, выглядит следующим образом:

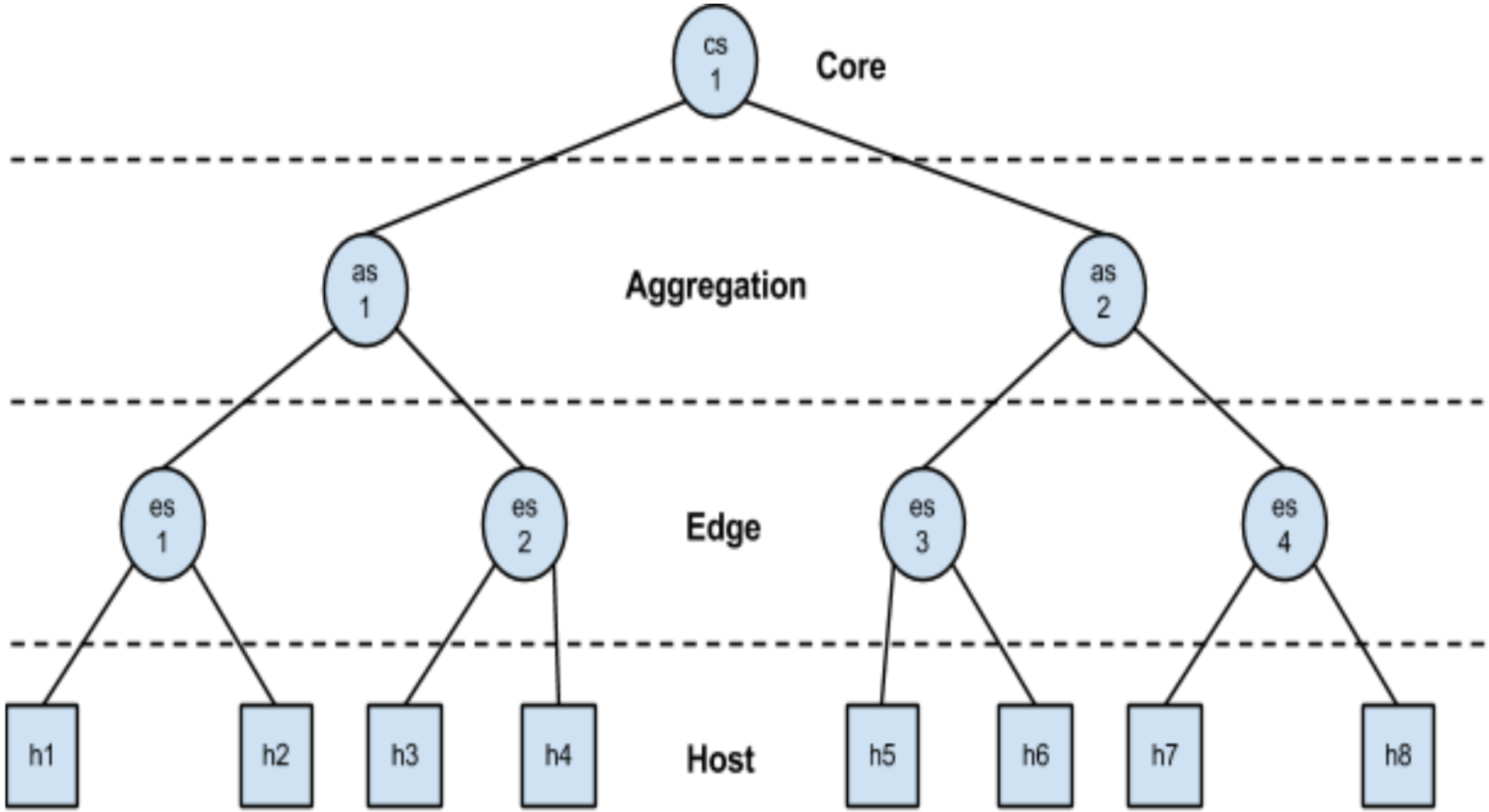


Figure 2: Simple Tree Topology with Fanout 2

SDN: Упражнения

Упражнение 2

Чтобы начать упражнение скачайте ***module3-assignment1.zip***

https://d19vezwu8eufl6.cloudfront.net/sdn/srcs/module3-assignment1.zip

Архив содержит два файла:

- **CustomTopo.py** : a sekleton class который необходимо дополнить логикой для создания топологии дерева описанной выше.
- **submit.py** : **этот файл не используется в нашем упражнении!**

SDN: Упражнения

Упражнение 2

CustomTopo.py

skeleton class может поддерживать следующие аргументы:

linkopts1 : параметр производительности для **линков** между **коммутаторами core** и **aggregation**.

linkopts2 : параметр производительности для **линков** между **коммутаторами aggregation** и **edge**.

linkopts3 : параметр производительности для **линков** между **коммутаторами edge** и **хостом**.

Fanout : параметр **fanout** означающий число **childs per node**.

Ваша логика должна содержать установку параметров **bw** и **delay** для каждого **линка**.

SDN: Упражнения

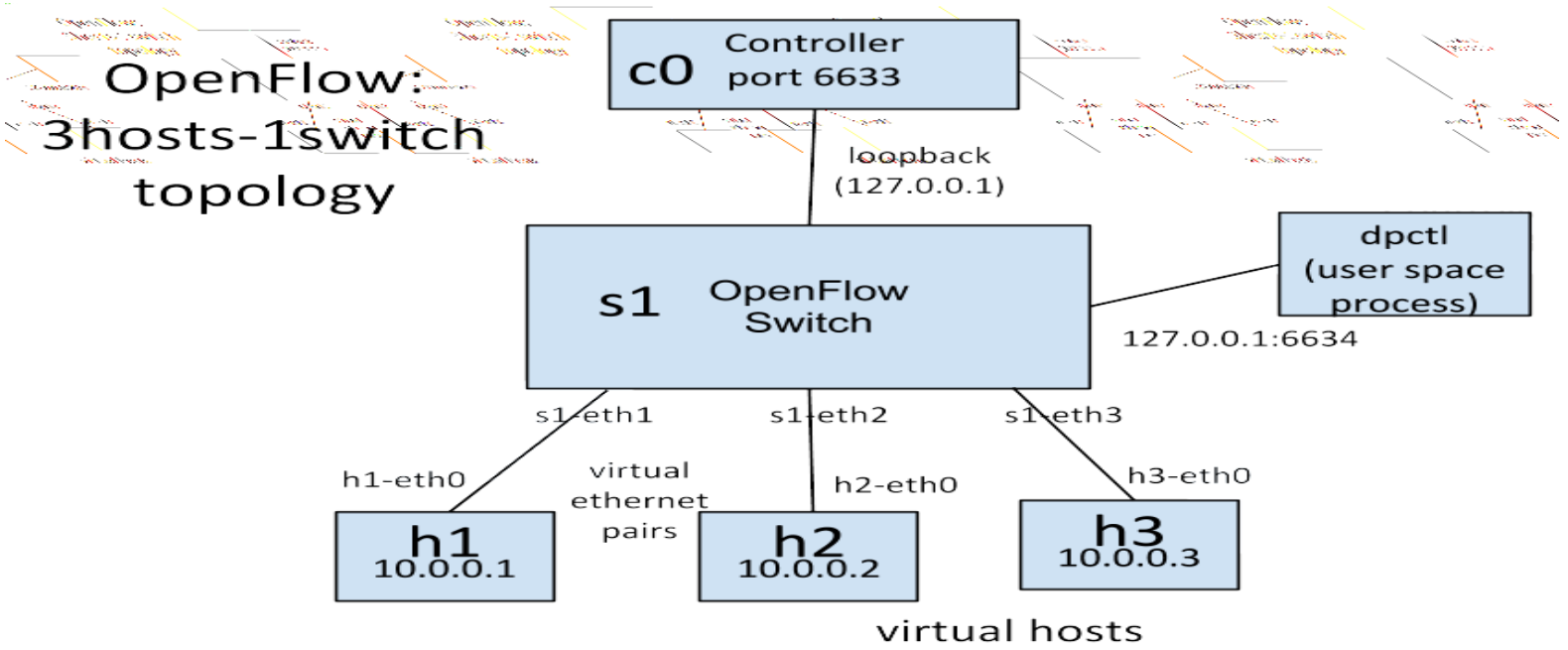
- Упражнение 3
- В этом упражнении вы узнаете о контроллере OpenFlow с открытым исходным кодом «POX».
- Вы узнаете, как писать сетевые приложения, например, Hub и Layer 2 MAC Learning и т.д., на POX и запускать их в виртуальной сети на основе Mininet.
- После этого упражнения вам будет предложено создать сетевое приложение, которое реализует L2 брандмауэр , который отключает входящий и исходящий трафик между двумя системами на основе их MAC-адреса.

SDN: Упражнения

Упражнение 3

Overview

Сеть, которую вы будете использовать в этом упражнении, включает 3 хоста и коммутатор с контроллером OpenFlow (POX):



SDN: Упражнения

Упражнение 3

POX - это платформа контроллера SDN на базе Python, ориентированная на исследования и образование.

Мы больше не будем использовать контроллер по умолчанию, который использует Mininet во время его моделирования. Убедитесь, что он не работает в фоновом режиме:

$ ps -A | grep controller

Если это так, вы должны убить его либо нажать Ctrl-C в окне, управляющем программой контроллера, либо в другом окне SSH:

$ sudo killall controller

Вы также должны запустить

sudo mn -c

SDN: Упражнения

упражнение 3

и перезапустить Mininet

mininet> exit

$ sudo mn -c

$ sudo mn --topo single,3 --mac --switch ovsk --controller remote

Контроллер POX поставляется с предустановленным образом VM.
Теперь запустите пример hub:

$ pox.py log.level --DEBUG forwarding.hub

Это заставит POX включить debug и запустить компонент hub.

Для подключения коммутаторов потребуется немного времени. Когда переключатель OpenFlow потеряет свое соединение с контроллером, он обычно увеличивает период, между которым он пытается связаться с контроллером, максимум до 15 секунд. Поскольку переключатель OpenFlow еще не подключен, эта задержка может составлять от 0 до 15 секунд. Если это слишком долго, коммутатор может быть настроен на ожидание не более N секунд с использованием параметра -max-backoff.

Кроме того, вы выходите из Mininet, чтобы удалить коммутаторы (ы), запустите контроллер, а затем запустите Mininet, чтобы немедленно подключиться.

SDN: Упражнения

упражнение 3

и перезапустить Mininet

mininet> exit

$ sudo mn -c

$ sudo mn --topo single,3 --mac --switch ovsk --controller remote

Контроллер POX поставляется с предустановленным образом VM.
Теперь запустите пример hub:

$ pox.py log.level --DEBUG forwarding.hub

Это заставит POX включить debug и запустить компонент hub.

Для подключения коммутаторов потребуется немного времени. Когда переключатель OpenFlow потеряет свое соединение с контроллером, он обычно увеличивает период, между которым он пытается связаться с контроллером, максимум до 15 секунд. Поскольку переключатель OpenFlow еще не подключен, эта задержка может составлять от 0 до 15 секунд. Если это слишком долго, коммутатор может быть настроен на ожидание не более N секунд с использованием параметра -max-backoff.

Кроме того, вы выходите из Mininet, чтобы удалить коммутаторы (ы), запустите контроллер, а затем запустите Mininet, чтобы немедленно подключиться.

SDN: Упражнения

Упражнение 3

Подождите, пока приложение покажет, что переключатель OpenFlow подключен. Когда коммутатор подключается, POX напечатает что-то вроде этого:

INFO:openflow.of_01:[Con 1/1] Connected to 00-00-00-00-00-01

DEBUG:samples.of_tutorial:Controlling [Con 1/1]

Проверить поведение хаба с помощью tcpdump

Убедитесь, что хосты могут пинговать друг друга и что все узлы видят один и тот же трафик - поведение концентратора. Для этого мы создадим xterms для каждого хоста и просматриваем трафик в каждом. В консоли Mininet запустите три xterms:

mininet> xterm h1 h2 h3

Расположите каждый xterm так, чтобы все они были на экране одновременно.

SDN: Упражнения

Упражнение 3

В xterms для h2 и h3 запустите tcpdump утилиту для печати пакетов, просматриваемых хостом:

tcpdump -XX -n -i h2-eth0

tcpdump -XX -n -i h3-eth0

В xterm для h1 запустите пинг:

ping -c1 10.0.0.2

Пакеты ping теперь идут к контроллеру, который затем посылает их на все интерфейсы, кроме отправляющего.

Вы должны увидеть идентичные ARP и ICMP-пакеты, соответствующие ping, в обоих xterms, выполняющих tcpdump.

Вот как работает хаб; он отправляет все пакеты на каждый порт в сети.

SDN: Упражнения

Упражнение 3

Теперь посмотрим, что произойдет, когда несуществующий хост не отвечает. От h1 xterm:

ping -c1 10.0.0.5

Вы должны увидеть три запроса ARP без ответа в xterms tcpdump.

Если ваш код отключен позже, три невыполненных запроса ARP - это сигнал, что вы можете случайно удалить пакеты.

Теперь вы можете закрыть xterms.

SDN: Упражнения

SDN: Упражнения

- Упражнение 4

SDN: Упражнения

- Упражнение 5