ИССЛЕДОВАНИЕ ПРИНЦИПОВ РАБОТЫ ПРОТОКОЛА OPENFLOW В ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЯХ

И.А. Альшаев[1*], **А.В. Красов**[1], **И.А.Ушаков**[1]

[1]Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, Санкт-Петербург, 193232, Российская Федерация
*Адрес для переписки: alshaev51@gmail.com

Информация о статье
УДК 004.056.52
Язык статьи – русский

Ссылка для цитирования: Альшаев И.А., Красов А.В., Ушаков И.А. Исследование принципов работы протокола OpenFlow в программно-конфигурируемых сетях // Труды учебных заведений связи. 2017. Т. 3. № 2. С. 16–27.

Аннотация: *Несмотря на возможность использования программно-конфигурируемых сетей, протоколы, которые обеспечивают корректное и безопасное управление этими сетями, постоянно подвергаются доработке. Принципы работы этих протоколов демонстрируют порядок обработки трафика в сетевых устройствах, уровень управления которых централизован на выделенном контроллере, а также, процесс передачи трафика от одного устройства к другому. В данной статье представлено исследование принципов работы протокола управления программно-конфигурируемых сетей – OpenFlow. Детально рассмотрены механизмы передачи сетевого трафика в OpenFlow-коммутаторы и процесс обработки полученной информации в самом коммутаторе.*

Ключевые слова: *программно-конфигурируемые сети, OpenFlow-коммутатор, OpenFlow-порт, пайплайн, таблица потоков, классификатор, исполнительное действие.*

RESEARCH OF THE WORKING PRINCIPLES OF THE OPENFLOW PROTOCOL IN THE SOFTWARE-DEFINED NETWORKS

I. Alshaev[1], **A. Krasov**[1], **I. Ushakov**[1]

[1]The Bonch-Bruevich Saint-Petersburg State University of Telecommunication, St. Petersburg, 193232, Russian Federation

Article info
Article in Russian

For citation: Alshaev I., Krasov A., Ushakov I. Research of the Working Principles of the Openflow Protocol in the Software-Defined Networks // Proceedings of Educational Institutes of Communication. 2017. Vol. 3. Iss. 2. PP. 16–27.

Abstract: *Despite on the fact that the SDN is actually in using, the protocols are always have experiencing some modernizations and completions. The work principles of this protocols are clearly*

show how the SDN working and the order of traffic handling in the network devices. This paper describes the researches of working SDN management protocol – OpenFlow. The article has a detail investigation of the network traffic transferring mechanisms in the OpenFlow – switches and processing of the received information in the OpenFlow – switch by itself.

Keywords: *software-defined networking, OpenFlow-switch, OpenFlow-port, pipeline, flow table, classifier, an action.*

Введение

На сегодняшний день самым распространённым решением для реализации управления сетевых устройств в программно-конфигурируемых сетей является протокол OpenFlow. По определению, OpenFlow – это протокол взаимодействия между сетевыми устройствами, такими как коммутаторы и маршрутизаторы, и централизованным контроллером, который представляет собой сетевую операционную систему [1], установленную на выделенном физическом сервере, в программно-конфигурируемой сети. Такое управление может заменить или дополнить работающую на сетевом устройстве функцию, осуществляющую построение маршрутов, создание таблицы коммутации и т. д. OpenFlow-контроллер используется для управления таблицами потоков коммутаторов, на основании которых принимается решение о передаче принятого пакета на конкретный порт коммутатора. Таким образом, в сети формируются прямые сетевые соединения с минимальными задержками передачи данных и необходимыми параметрами. На рис. 1 представлена схема взаимодействия между контроллером и сетевыми устройствами с помощью различных протоколов управления уровнем передачи данных, к которому относятся эти устройства.

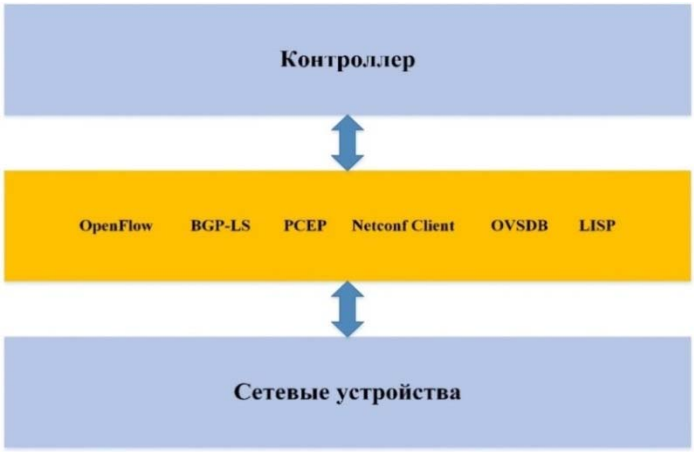

Рисунок 1. Взаимодействие между контроллером и коммутатором

Основные проблемы протокола

На сегодняшний день реализация протокола OpenFlow имеет некоторые открытые вопросы. Централизация управления сетью может стать настоящей проблемой, которая приводит к некоторым сомнениям в развертывании программно-конфигурируемой сети в принципе.

Основными угрозами при внедрении данного протокола являются:

1) В случае отсутствия связи между контроллером и устройствами сети коммутаторы переходят в автономное состояние, и уровень передачи данных становится просто-напросто неуправляемым или вовсе перестает работать.

2) Ошибка в программировании контроллера может привести к серьезным проблемам на всей сети, которую он обслуживает.

3) Контроллер становится основной точкой уязвимости и главной целью для атак со стороны злоумышленников.

4) Пока такого рода управление, вынесенное отдельным уровнем (control plane), делается как проприетарное программное обеспечение, то имеют место такие понятия как «ответственность», «техническая поддержка» и т. п. А если в качестве решения выступает открытое программное обеспечение, то конечный пользователь не сможет обратиться за поддержкой к какому-либо вендору.

Контроллеры и коммутаторы

При такой концепции всей системы особенно важным становится выбор контроллера OpenFlow. Именно его быстродействие станет решающим фактором при возникновении любых экстренных ситуаций. Именно его алгоритмы будут определять эффективность работы всей сети и его сбой с большой степенью вероятности повлияет на работу всей сети.

Ingress Port	Ingress MAC	Dest MAC	Ether Type	VLAN ID	VLAN Priority	IP SRC	IP DEST	IP Protocol	IP TOS	TCP/UDP SRC	TCP/UDP DEST
1	2	3	4	5	6	7	8	9	10	11	12

Рисунок 2. Базовые поля в OpenFlow-пакете

Основная идея предполагает, что если мы отделяем уровень управления от уровня передачи трафика в коммутаторах, то они, как минимум, должны стать проще. Но тем не менее, новый протокол, а точнее огромное количество учитываемых по умолчанию полей в таблицах маршрутизации, выдвигает повышенные требования к объемам памяти на коммутаторах. На рис. 2 показаны поля стандартного OpenFlow-пакета.

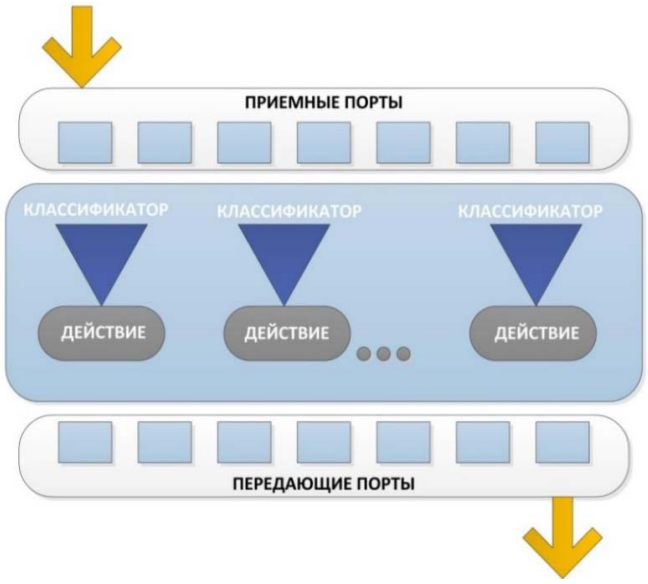

Рисунок 3. Схема прохождения OpenFlow-пакета в коммутаторе

Уровень передачи данных коммутатора состоит из следующих компонентов, который программируются OpenFlow-контроллером [1]: OpenFlow-порты (Ports), потоки информации (Flows), таблицы потоков (Flow-Tables), классификаторы (Matches), исполнительные действия (Actions). Приходящие пакеты на OpenFlow-порт распределяются по потокам в таблицах потоков с помощью классификаторов. Каждый поток состоит из набора действий (Actions), приме-

няющихся к каждому пакету, который должен соответствовать определённому правилу. На рис. 3 представлена схема прохождения пакета в OpenFlow-коммутаторе.

OpenFlow-порты

OpenFlow-порты выполняют те же функции, что и порты стандартного коммутатора. С точки зрения входящих и исходящих потоков трафика, нет никакой разницы с обычным L2-коммутатором традиционной IP-сети [2]. Входящий порт одного потока может быть исходящим для другого. Процесс коммутации пакетов в OpenFlowвыглядит следующим образом. Пакеты, обрабатываемые протоколом OpenFlow, принимаются на входящий порт, обрабатываются в пайплайне коммутатора и направляются далее в исходящий порт. По определению, пайплайном считается программный конвейер, в котором происходит последовательный процесс обработки данных. Например, значение входящего порта является одним из свойств пакета, проходящего через пайплайн коммутатора, и показывает на каком OpenFlow-порту этот пакет был принят в обработку [3]. Пайплайн в протоколе OpenFlow является самой важной частью, так как именно в нем принимается решение о том, отправлять пакеты на исходящий порт или нет, при этом устанавливая параметры действий (actions). А параметры этих действий в свою очередь будут определять, как именно пакет будет возвращён обратно в сеть.

Исследуемый протокол определяет набор стандартных портов: физические, логические и локальные зарезервированные, если поддерживаются OpenFlow-коммутатором. Физическими портами (physical) являются непосредственно порты самого коммутатора. Логические (logical), это порты напрямую непривязанные к физическим интерфейсам оборудования, как и в традиционных сетях, например, VLAN, Tunnel и Null интерфейсы [4, 5]. OpenFlow-коммутатор может создавать определённое число OpenFlow-портов, доступных для использования в локальных процессах коммутатора, такие порты называются зарезервированными (reserved). Зарезервированные порты используются для внутренней обработки трафика и для гибридных типов внедрения, например, OpenFlow-порты в связке с портами традиционной сети. Зарезервированные порты могут быть обязательными или опциональными. Обязательные порты включают порты типа ALL, CONTROLLER, TABLE, IN_PORT, ANY, UNSET. В то время как опциональные порты – LOCAL, NORMAL и FLOOD.

Коммутаторы, которые разработаны только для работы с OpenFlow не поддерживают порты типа NORMALи FLOOD, в то время как гибридные коммутаторы поддерживают. Передача трафика в порты типа FLOOD зависит от того, как сконфигурирован и для чего предназначается коммутатор. Здесь можно упомянуть об использовании группы ALL, которая позволяет контроллеру более гибко реализовывать потоковую рассылку. А если, например, понадобится создать туннель между двумя конечными точками, это будет невозможно сделать с помощью протокола OpenFlow. Подобные задачи можно

выполнить с помощью других протоколов, например, OF-CONFIG. Порты могут быть добавлены или удалены из конфигурации коммутатора с помощью OF-CONFIG [5], но не с помощью OpenFlow.

OpenFlow таблицы

При получении пакета из него извлекаются метаданные (например, ingress port) и другие поля пакета. Затем эти поля сравниваются с записями в таблице потоков. Каждая запись в таблице имеет соответствующее поле «protocol», по которому идет сравнение. Результат с наивысшим приоритетом считается лучшим, и обработка пакета с таким приоритетом начинается первой. Каждая запись в таблице потоков должна иметь различный приоритет, а запись с наивысшим приоритетом определяет то, как дальше будет обработан пакет. На рис. 4 наглядно представлена стандартная таблица потоков, через которую проходит пакет, а на рис. 5 показан процесс обработки пакета в пайплайне, который состоит из нескольких таких таблиц.

Рисунок 4. Стандартная таблица потоков

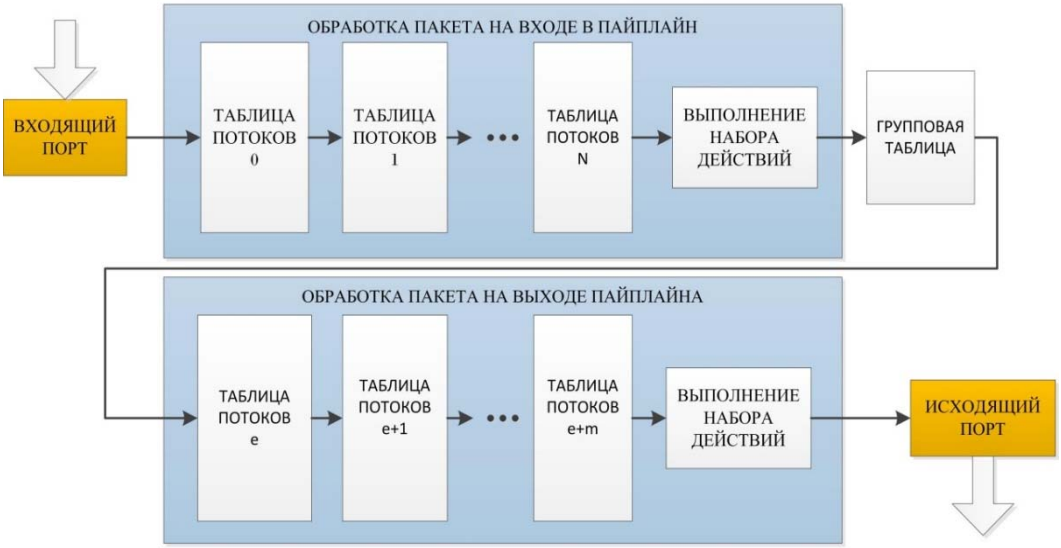

Рисунок 5. Процесс обработки пакетов в пайплайне OpenFlow-коммутатора

Как только выбрано правило (Match), к пакету применяется определенное действие (Action). Возможными действиями могут быть: «отправить в порт X», «отбросить» или «отправить на контроллер». Пример такого процесса показан на рис. 6. По умолчанию протокол OpenFlow со времён версии 1.0 отправляет на контроллер все пакеты, не соответствующие ни одному правилу. В более поздних версиях протокола этот механизм был изменен, т. к. может подвергнуть DoS-атакам на контроллер со стороны злоумышленников [6].

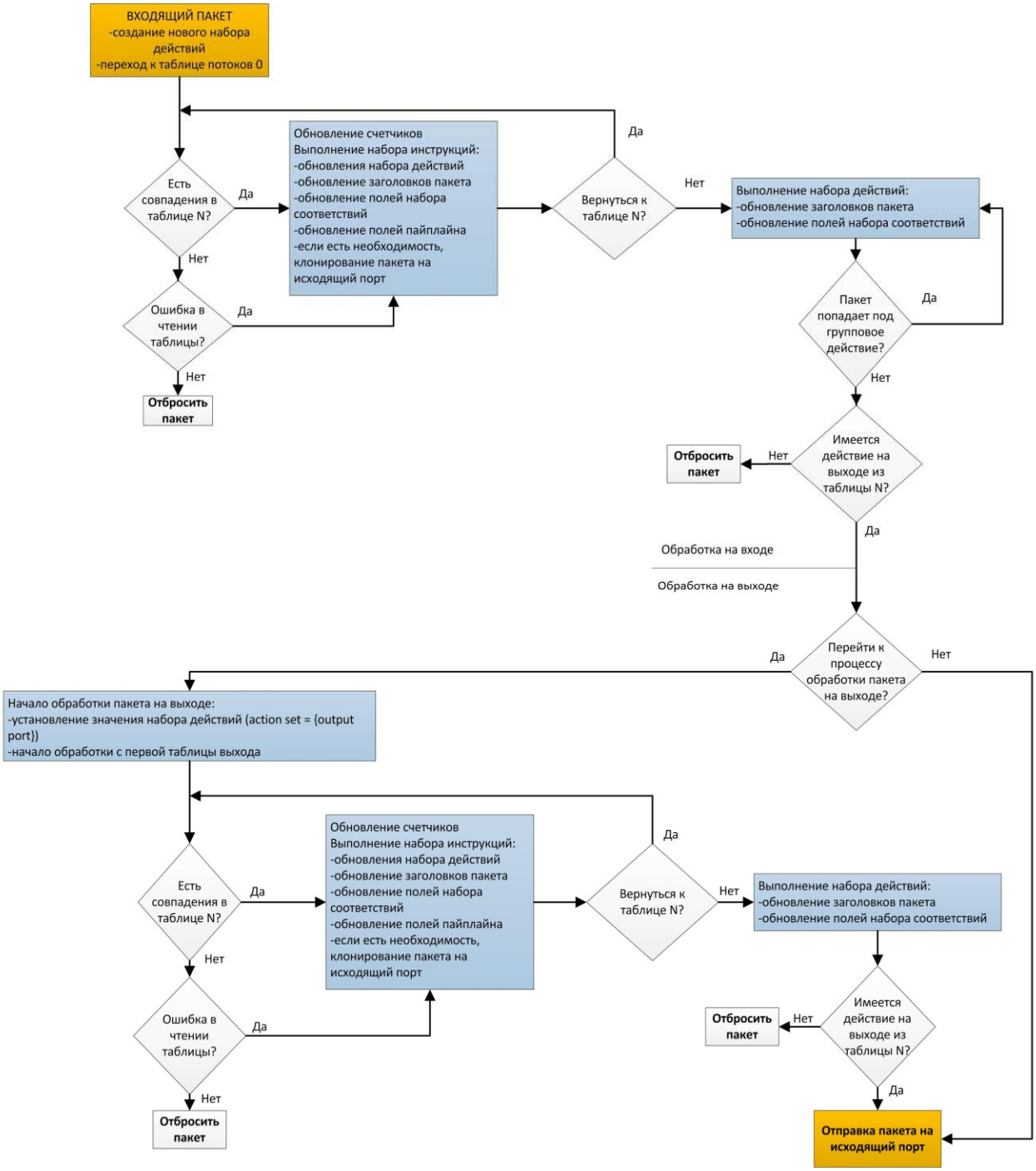

Рисунок 6. Детальная блок-схема прохождения пакетов в OpenFlow-коммутаторе

Первоначальная спецификация протокола OpenFlow 1.0 предполагала лишь возможности «перенаправить на другой порт» или «отправить на контроллер», но позже пришли к выводу, что в определенных случаях требуется модифицировать поля в пакете, например, уменьшить TTL или добавить тег VLAN. Версия 1.0 была признана неполной, т. к. необходима возможность выполнения более одного действия с конкретным пакетом.

Ограничение на количество таблиц в последующих версиях протокола было изменено. Но в то же время разрушило идею о том, что OpenFlow может использоваться на недорогих коммутаторах. Наличие нескольких таблиц накладывает дополнительные требования к ресурсам коммутаторов.

Исследование и примеры создания OpenFlow-записей в таблицах

Рассмотрим методики создания топологий сетей, которые будут содержать в себе коммутатор, с поддержкой протокола OpenFlow, OpenFlow-контроллер и несколько хостов для реализации простейшей локальной сети, а также исследуем воздействие OpenFlow-записей на трафик в коммутаторе. Виртуальный стенд развернут в эмуляторе сетей mininet, который позволяет работать с виртуальной сетью таким образом, чтобы желаемая топология могла содержать различные параметры хостов и коммутаторов. Для запуска эмулятора mininet использовалась операционная система Ubuntu 16.10. Для управления OpenFlow-коммутатором использовались команды – ovs-ofctl, которые позволяют создать политики обработки трафика и управлять его передачей в сети.

На рис. 7 показан скриншот запущенного эмулятора mininet с параметром –mac: sudo mn --topo=single,3 --controller=none –mac, что позволяет создать топологию с 1 коммутатором и 3 хостами.

```
pigga@ubuntu:~$ sudo mn --topo=single,3 --controller=none --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=5091>
<Host h2: h2-eth0:10.0.0.2 pid=5093>
<Host h3: h3-eth0:10.0.0.3 pid=5095>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=5100>
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
```

Рисунок 7. Создание простейшей топологии сети с OpenFlow-коммутатором

Параметр --controller=none исключает OpenFlow-контроллер из топологии в данном примере. Отсюда следует, что если мы изменим значения параметра --topo на 2 и 4, то топология будет состоять из двух OpenFlow-коммутаторов и каждый из коммутаторов будет иметь по 4 хоста. Команда dump позволяет вывести информацию о всех узлах сети и хостах с параметрами интерфейсов. Команда net демонстрирует как соединены хосты и узлы.

Созданная сеть будет выглядеть следующим образом (рис. 8). Параметры виртуальных сетевых адаптеров получаем с помощью широкоизвестной команды ifconfig. В интерпретаторе mininet это будет выглядеть следующим образом: h1 (h2 & h3) ifconfig. Open vSwitch имеет свойство обозначать свои порты для того, чтобы их использование в дальнейшем было упрощено. Каждый хост будет закреплен под определенным номером OpenFlow-порта. Для удобства, номер хоста соответствует номеру порта на OpenFlow-коммутаторе.

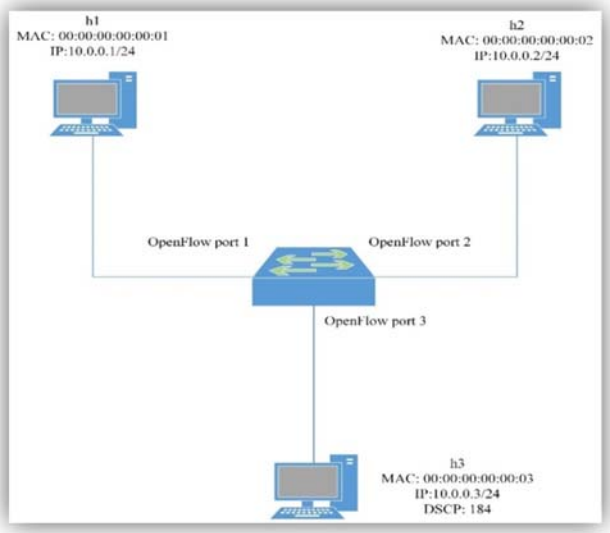

Рисунок 8. Топология сети
и параметры конфигурации

Рассмотрим пример (рис. 9), где sh ovs-ofctl show s1 позволяет запускать команду из терминала операционной системы.

```
mininet> sh ovs-ofctl show s1
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_nw_s
rc mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(s1-eth1): addr:6e:a6:70:f0:32:31
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s1-eth2): addr:e2:3d:89:b0:99:92
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s1-eth3): addr:22:85:b1:3c:2f:3c
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s1): addr:7e:fd:5b:4e:b7:4d
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
mininet>
```

Рисунок 9. Параметры OpenFlow-коммутатора. Привязка OpenFlow-портов
к стандартным портам коммутатора

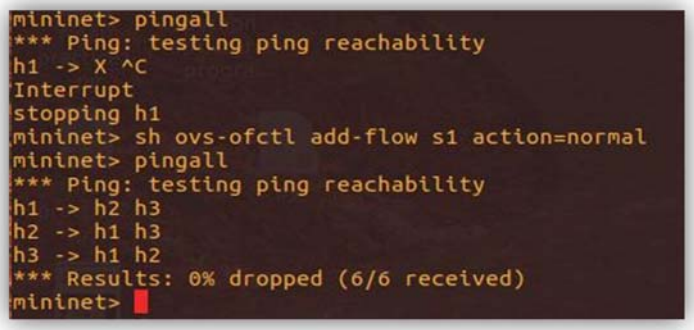

Рисунок 10. Добавление записи в OpenFlow-таблицу

Далее для проверки связи между хостами нужно добавить OpenFlow-запись (рис. 10).

Здесь можно увидеть, что, если не задать ни одного правила на OpenFlow-коммутаторе, тосвязи в локальной сети не будет. Для того, чтобы создать стандартную OpenFlow-запись, нужно воспользоваться командой: sh ovs-ofctl add-flow s1 action= normal.

Для того чтобы получить данные об OpenFlow-записи, используем следующую команду: sh ovs-ofctl dump-flows s1. И получим (см. рис. 11):

Рисунок 11. OpenFlow-таблица

Чтобы удалить все записи для потоков, необходимо воспользоваться командой sh ovs-ofctl del-flows s1.

Создадим правило в OpenFlow-таблице, которое будет указывать коммутатору, на какой из портов отправлять трафик, который пришел с указанного порта. Такая запись называется Layer 1 matching (см. рис. 12).

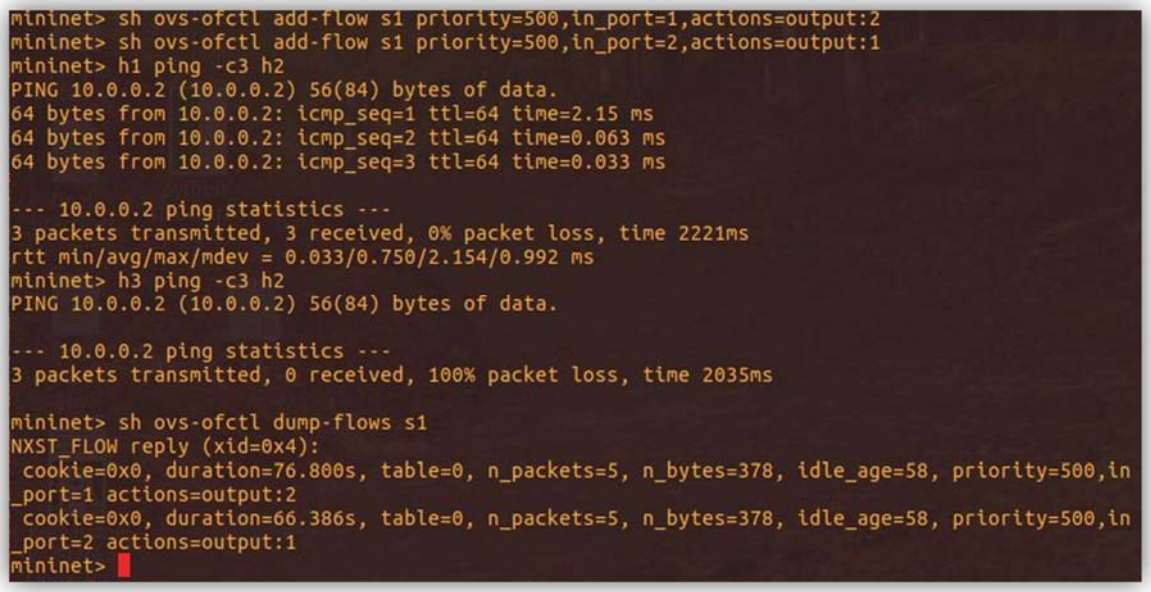

Рисунок 12. OpenFlow-таблица № 1. Layer 1 matching

Теперь OpenFlow-коммутатор может применить определённые ему правила при передаче трафика с порта 1 на порт 2 и наоборот. Как видно на рис. 12, связи с хостом h3 нет как раз из-за отсутствия правил для порта, к которому он подключен (порт 3). После повторного применения команды sh ovs-ofctl dump-flows s1 видим, как изменилась таблица OpenFlow-записей.

Управлять правилами в таблицах OpenFlow-записей позволяет поле приоритета (priority). Посмотрим, как влияет добавление правила с большим значением priority на передачу трафика, а затем удалим его, используя параметр --strict: (см. рис. 13).

```
mininet> sh ovs-ofctl add-flow s1 priority=32768,actions=drop
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2030ms

mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=20.085s, table=0, n_packets=6, n_bytes=420, idle_age=7, actions=drop
 cookie=0x0, duration=1100.334s, table=0, n_packets=5, n_bytes=378, idle_age=1082, priority=500,in_port=1 actions=output
:2
 cookie=0x0, duration=1089.920s, table=0, n_packets=5, n_bytes=378, idle_age=1082, priority=500,in_port=2 actions=output
:1
mininet> sh ovs-ofctl del-flows --strict
ovs-ofctl: 'del-flows' command requires at least 1 arguments
mininet> sh ovs-ofctl del-flows s1 --strict
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1151.669s, table=0, n_packets=5, n_bytes=378, idle_age=1133, priority=500,in_port=1 actions=output
:2
 cookie=0x0, duration=1141.255s, table=0, n_packets=5, n_bytes=378, idle_age=1133, priority=500,in_port=2 actions=output
:1
mininet>
```

Рисунок 13. Приоритет записей в OpenFlow-таблицах

Следующее правило будет сравнивать MAC-адрес источника трафика и MAC-адрес назначения, и отправлять трафик, согласно значению параметра output. Такая запись называется Layer 2 matching (см. рис. 14).

```
mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,actions=ou
tput:2
mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01,actions=ou
tput:1
mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,actions=flood
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X
h2 -> h1 X
h3 -> X X
*** Results: 66% dropped (2/6 received)
mininet>
```

Рисунок 14. OpenFlow-таблица № 2. Layer 2 matching

Для корректной работы L2 необходимо прописать правило, которое будет контролировать ARP-запросы. Первые 2 строчки таблицы описывают передачу трафика согласно физическим адресам, но для того, чтобы OpenFlow-

коммутатор получил сведения о том, за каким IP закреплён нужный MAC-адрес, нужно описать правило, которое разрешает широковещательный трафик. В данном примере действие flood даёт возможность отправлять ARP-REQUEST на все порты, кроме того, с которого запрос был сгенерирован.

Третья таблица OpenFlow-записей описывает политику передачи трафика на основе IP адресации и ToS – Layer 3 matching. Запись: sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.3,actions=mod_nw_tos:184,normal. В данной записи описывается ToS, который позволяет маркировать трафик специальной меткой DSCP (Differenciated Services Code Point), что в свою очередь определяет его приоритет. Если обратить внимание на саму OpenFlow-запись, то параметр priority отличается от DSCP тем, что задает порядок сравнения пакетов передаваемого трафика с записями в OpenFlow-таблице. Если повысить значение priority в записи для ToS, то пакет сначала пройдет сравнение по этой записи, а потом по всем остальным. На рисунке 15 показан пример создания OpenFlow-таблицы Layer 3 matching. Такая структура таблиц очень похожа на работу списков доступа в процессе маршрутизации трафика в традиционных сетях – ACL.

```
mininet> sh ovs-ctl add-flow s1 priority=500,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=normal
/bin/sh: 1: ovs-ctl: not found
mininet> sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24,actions=norm
al
mininet> sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.3,actions=mod_nw_tos:184,normal
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.1,actions=output:1
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.2,actions=output:2
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.3,actions=output:3
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=48.844s, table=0, n_packets=4, n_bytes=168, idle_age=13, arp,arp_tpa=10.0.0.1
actions=output:1
 cookie=0x0, duration=41.844s, table=0, n_packets=4, n_bytes=168, idle_age=13, arp,arp_tpa=10.0.0.2
actions=output:2
 cookie=0x0, duration=23.328s, table=0, n_packets=4, n_bytes=168, idle_age=13, arp,arp_tpa=10.0.0.3
actions=output:3
 cookie=0x0, duration=123.928s, table=0, n_packets=12, n_bytes=1176, idle_age=18, priority=500,ip,nw
_src=10.0.0.0/24,nw_dst=10.0.0.0/24 actions=NORMAL
 cookie=0x0, duration=87.500s, table=0, n_packets=0, n_bytes=0, idle_age=87, priority=500,ip,nw_src=
10.0.0.3 actions=mod_nw_tos:184,NORMAL
mininet>
```

Рисунок 15. OpenFlow-таблица № 3. Layer 3 matching

Заключение

Таким образом, проведенное исследование показало, что использование OpenFlow позволяет упростить множество задач, а реализация таблицы обработки потоков данных обеспечивает многоуровневую безопасность. В приведенном исследовании показано, как с помощью OpenFlow-контроллера управлять коммутатором с помощью внутреннего языка командной строки Mininet. Для более приближенной демонстрации работы SDN данный макет необходимо реализовать на оборудовании, которое будет поддерживать работу OpenFlow. На реальном макете можно будет доказать значимость проблем протокола

OpenFlow, которые проявляются в незащищённости передаваемых сообщений с информацией о политике передачи трафика. На сегодняшний день, наиболее известным и бюджетным вариантом является использование оборудования MikroTik. Реализация данного макета находится на стадии тестирования.

В дальнейшем планируется провести более детальное исследование работы программно-конфигурируемых сетей на базе протокола OpenFlow и выполнить работу по внедрению механизмов защиты от перехвата трафика, используя TLS-шифрование.

Список используемых источников

1. Open Networking Foundation. OpenFlow Switch Specification Version 1.5.1 (Protocol version 0x06) // Open Networking Foundation. 2015. URL: https://www.opennetworking.org /images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

2. Красов А.В., Левин М.В., Цветков А.Ю. Управление сетями передачи данных с изменяющейся нагрузкой // Всероссийская научная конференция по проблемам управления в технических системах. 2015. № 1. С. 141–146.

3. Дубровин Н.Д., Ушаков И.А., Чечулин А.А. Применение технологии больших данных в системах управления информацией и событиями безопасности // Актуальные проблемы инфотелекоммуникаций в науке и образовании: сборник научных статей V международной научно-технической и научно-методической конференции. СПбГУТ. 2016. С. 348–353.

4. Алейников А.А., Билятдинов К.З., Красов А.В., Левин М.В. Контроль, измерение и интеллектуальное управление трафиком: монография. СПб.: Центр «Астерион», 2016. 92 с.

5. Open Networking Foundation. OF–CONFIG 1.2 OpenFlow Management and Configuration Protocol // Open Networking Foundation. 2014. URL: https://www.opennetworking.org /images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf.

6. Алейников А.А., Билятдинов К.З., Красов А.В., Кривчун Е.А., Крысанов А.В. Технические аспекты управления с использованием сети интернет. СПб.: Центр «Астерион», 2016. 305 с.