УДК 378.2

# ПРОЕКТИРОВАНИЕ И ПРОГРАММИРОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ КАЧЕСТВОМ ПРОГРАММНО-ОПРЕДЕЛЯЕМЫХ СЕТЕЙ

В.А. ВИШНЯКОВ, Б.А. МОНИЧ

*Учреждение образования «Белорусская государственная академия связи», ул. Ф. Скорины, 8/2, Минск, 220114, Беларусь*

*Поступила в редакцию 2 марта 2020*

Приведено понятие качества программно-определяемых сетей в облачных средах. Определены требования к качеству их обслуживания. Рассмотрен подход к проектированию системы управления качеством в облачной среде. Приведены диаграммы использования и сущность-связь для системы нее. Рассмотрена структура интерфейса программного средства системы управления качеством в облачной среде, даны особенности программирования этой системы, рассмотрены вопросы ее тестирования.

*Ключевые слова:* программно-определяемая сеть, качество обслуживания, диаграммы, структура, программирование, тестирование.

## Введение

Технологии программно-конфигурируемых сетей (Software-Defined Networking − SDN) выполняют согласованное быстрое управление сетями, разрешая управлять всей сетью с одной консоли управления [1]. Необходимо решать вопросы обеспечения качества сервиса (QoS), среди его параметров наиболее важные: Bandwidth (BW) – полоса пропускания, которая описывает номинальную пропускную способность среды передачи информации, определяет ширину канала; Delay – задержка при передаче пакета; Jitter – вариация задержки при передаче пакетов; Packet Loss – потери пакетов [2].

Основа реализации качества обслуживания базируется на контроле входа и выхода пакета из оконечных устройств. Контроль над прохождением пакетов через сеть доступен только в пределах центра обработки данных (ЦОД), за пределами ЦОД вся ответственность ложится на провайдеров телекоммуникационных услуг. Протокол OpenFlow позволяет программному обеспечению SDN взаимодействовать с элементами сети − маршрутизаторами и коммутаторами через открытые интерфейсы API. QoS предназначен для решения перегрузки каналов и потери пакетов, которые приводят к снижению производительности приложений [2].

Согласно концепции SDN вся логика управления выносится в контроллеры OpenFlow, которые способны отслеживать работу всей сети [3].

В работе [4] определена спецификация OpenFlow для поддержки функционала в области QoS, показаны примеры настройки качества программно-определяемых сетей. Обсуждено взаимодействие модуля прототипа обеспечения QoS с сетевой облачной средой (ОС). Рассмотрена схема обеспечения QoS в OpenFlow-коммутаторе. Рассмотрим вопросы проектирования и программирования управления качеством в облачной среде (УКвОС).

## Требования к качеству обслуживания

Основа реализации качества обслуживания базируется на контроле входа и выхода пакета из устройства. Перегрузка каналов и потери пакетов могут произойти по ряду причин, например, превышение максимальной пропускной способности, высокий коэффициент загрузки (это делает их неспособными к эффективной обработке входящих пакетов), а также при неправильной конфигурации сетевых устройств. Некоторые типы QoS, например,

ограничение полосы пропускания, могут помочь в решении проблемных ситуаций. Управление полосой пропускания может уменьшить проблемы, связанные с ограниченной пропускной способностью каналов связи. Вне контекста приложений QoS редко дает заметное улучшение в производительности с точки зрения конечного пользователя. Контекст является необходимым для того, чтобы применять правильные методы и политику в нужное время, чтобы обеспечить оптимальную производительность для пользователей конкретного приложения.

Используются два основных типа QoS услуг: ограничение скорости на входе и гарантирование минимальной пропускной способности на выходе. Первая обычно делается на основе измерений скорости входного порта или потока, после превышения определенной скорости пакеты отбрасываются в соответствии с некоторым алгоритмом. Ограничение минимальной гарантированной полосы пропускания означает, что каждому потоку будет доступна определенная часть от имеющейся пропускной способности. OpenFlow поддерживает восемь очередей. На каждый порт можно подключить одну или несколько очередей. Каждый поток ассоциируется с конкретной очередью. Обработка в очереди происходит в соответствии с настройкой конкретной очереди. По умолчанию все очереди имеют настройку на предоставление минимальной гарантированной пропускной способности.

Для корректной настройки параметров QoS необходима точная информация о трафике в сети. В протокол OpenFlow встроена возможность анализа проходящего трафика средствами измерений. Все измерения записываются в таблицу, которая состоит из записей измерений, определяющих измерения для каждого потока. Измерения для каждого потока позволяют OpenFlow реализовывать различные операции QoS, такие как ограничение скорости, понижение\повышение приоритета пакета.

Контроллер должен постоянно проводить мониторинг всей сети с целью обеспечения глобального управления ресурсами. При необходимости контроллер может рассчитать места внедрения или удаления политик QoS, заново рассчитывать оптимальные маршруты с учетом применения QoS и требований приложений. Кроме того, предлагается регулярно делать копию таблиц OpenFlow с коммутаторов и проводить их анализ для выявления несоответствий. Дополнительно во многих коммутаторах предоставляется возможность запроса QoS конфигурации. Для обеспечения функции глобального управления производительностью и качеством обслуживания сети необходима единая модель, в которой будут учитываться топология, полоса пропускания и настройки QoS для каждого потока. Предлагаемая модель применения политик QoS должна основываться на базовых возможностях коммутаторов в области качества обслуживания – ограничителях скорости и статических очередях с приоритетами. Ограничитель скорости в OpenFlow коммутаторах реализован с некоторыми отличиями от Received Signal Code Power (RSCP) в обычных коммутаторах**.**

## Проектирование функциональной модели системы

Представим список задач, которые должны решаться разрабатываемым программным средством:

– управлять виртуализированным сервером;

– централизованно управлять сетью виртуальных машин, настраивать параметры для обеспечения надлежащего качества сети;

– поддерживать работу сети на основе заданных ранее параметров;

– повысить отказоустойчивость сети.

Виртуализация позволяет сократить количество физических серверов и распределить ресурсы по загруженности, а также снизить энергопотребление: четыре физических сервера со средней загруженностью в 15 % будут расходовать больше энергии, чем один сервер со средней загруженностью даже в 75 %. При виртуализации есть возможность удаленного доступа к консоли виртуальных серверов и изменения аппаратных характеристик: добавление ядер процессора и дискового пространства. Виртуализация сервера позволяет быстро и удаленно менять параметры, которые напрямую влияют на качество сети.

Диаграмма вариантов использования в моделировании программного обеспечения считается одной из обязательных наряду с конструкциями диаграмм классов, последовательности и развертывания. Она позволяет описать проектируемую систему на

абстрактных концептуальных уровнях, учитывая актеров и варианты ее применения. Диаграмма вариантов использования представлена на рис. 1. В качестве актеров выступают администратор (настраивает сеть, управляет сервером и виртуальными машинами), пользователь (использует ресурсы сети), контроллер (проверяет качество управления в сети).

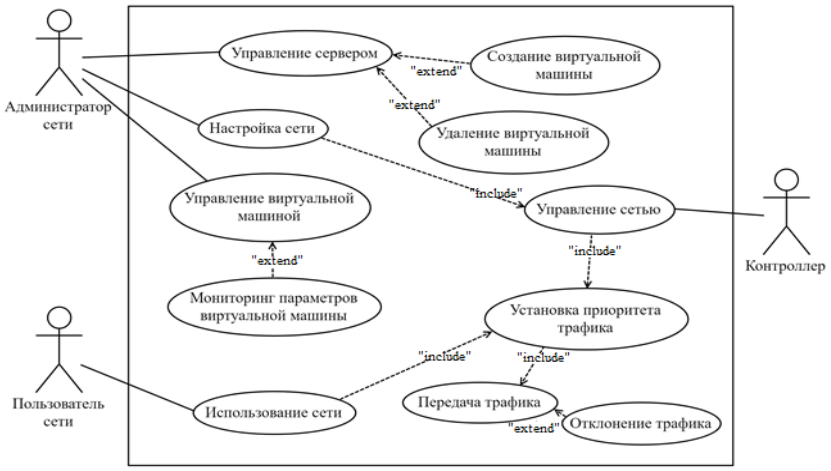


Рис. 1. Диаграмма вариантов использования УКвОС

**Проектирование диаграммы сущность-связь.** Базовыми понятиями модели сущность-связь являются: сущность, атрибут и связь. Основные понятия диаграммы сущность-связь:

– Сущность – это класс однотипных объектов, информация о которых должна быть учтена в модели. Примерами сущностей могут быть такие классы объектов, как «Коммутатор», «Сервер», «Пакет».

– Экземпляр сущности – это конкретный представитель данной сущности. Например, конкретный представитель сущности – «Коммутатор OpenFlow».

– Атрибут сущности – это именованная характеристика, являющаяся некоторым свойством сущности. Примерами атрибутов сущности, «Коммутатор» могут быть такие атрибуты, как «Модель», «Номер в сети», «Приоритет» и т. п.

– Ключ сущности – это неизбыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. На диаграмме ключевые атрибуты отображаются подчеркиванием.

– Связь – это отношение одной сущности к другой или к самой себе. Связи между сущностями могут выражаться следующими фразами: «КОНТРОЛЛЕР может иметь несколько КОММУТАТОРОВ», «ПАКЕТ должен иметь один ПРИОРИТЕТ». Графически связь изображается линией, соединяющей две сущности. На рис. 2 показана диаграмма сущность-связь УКвОС.

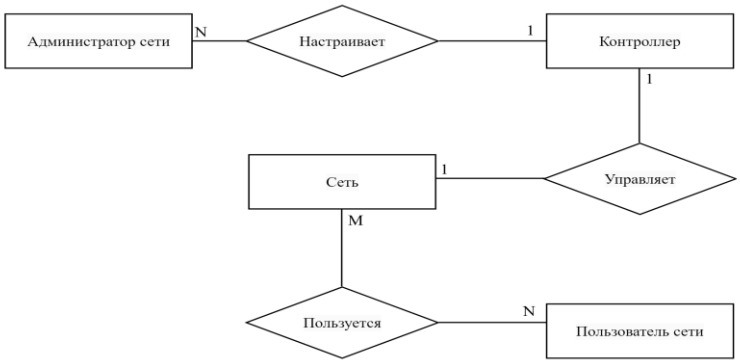


Рис. 2. Диаграмма сущность-связь УКвОС

### Структура интерфейса программного средства

Рассмотрим структуру интерфейса разрабатываемой программы УКвОС (рис. 3), которая включает окно авторизации, окно управления виртуальными машинами на сервере, окно мониторинга параметров выбранной виртуальной машины, окно управления сетью.

При входе в программу первое, что видит пользователь, – окно авторизации. Должно быть два поля для ввода данных: имя пользователя, пароль. В окне управления виртуальными машинами должны быть наглядно показаны запущенные на данном сервере виртуальные машины, в том числе название виртуальной машины; нагрузка, оказываемая на ресурсы сервера виртуальной машиной.

В окне мониторинга параметров виртуальной машиной отображается информация о состоянии выбранной виртуальной машины: название виртуальной машины, время работы виртуальной машины, загрузка процессора виртуальной машины, оперативная память виртуальной машины; дисковое пространство виртуальной машины, скорость входящего и исходящего потока данных.


Рис. 3. Структура интерфейса программного средства УКвОС

В окне управления сетью представлены пункты интерфейса, позволяющие настраивать сеть: схема представленной сети, состояние каждого из ее элементов, возможность настроить коммутатор; возможность отключения хостов от сети. В каждом из окон программы есть возможность выйти из сети, в таком случае у пользователя откроется окно авторизации.

### Разработка программной системы

В качества языка программирования выбран Pyton [5]. Для создания сети использован компонент MiniNet [6], применены особенности управления программно-управляемым средством (ПУС) на базе технологии OpenFlow [7]. Команда net покажет схему соединения устройств (рис. 4). Sudo mn запустит MiniNet, topo single, 4 означает, что будет создана сеть с простой топологией, в которой будет четыре хоста и один коммутатор.

Sudo mn – – topo single,4 – controller remote.

```
sudo mn --topo single,4 --controller remote
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0
```

Рис. 4. Результат выполнения команды создания сети

Список команд, доступных в MiniNet, приведен в табл. 1.

Таблица 1. Список команд MiniNet

| Команда | Описание |
|---|---|
| dump | отображает все IP-адреса устройств, их интерфейсы |
| intfs | отображает список интерфейсов устройств |
| iperf | измеряет производительность обмена по TCP между первым и последним хостом |
| iperfudp | измеряет производительность обмена по UDP между первым и последним хостом |
| net | показывает схему соединений устройств |
| link | позволяет создать новую связь «link устройство1 устройство2 [up\|down]» |
| nodes | показывает список устройств |
| noecho | выполняет команду, но не отображает информацию о результате |
| pingpair | посылает ICMP-пакеты между первыми двумя хостами |
| pingall | посылает несколько ICMP-пакетов между всеми хостами и показывает результат |
| py | выполняет любую команду на языке Python |
| quit, exit | закрывает MiniNet |
| source | читает список команд из указанного после команды файла |
| sh | выполняет произвольную команду в операционной системе |
| ping | выполняет команду ping на указанном хосте, ее синтаксис «хост1 ping хост2» |

login.py – авторизация, это простая часть программы для авторизации, имеет два поля для ввода и кнопку. Происходит сравнение введенных данных с данными учетной записи администратора, при совпадении пользователь работает в виртуальной среде.

server.py – управление виртуальными машинами. Создание виртуализированного сервера позволяет наиболее эффективно использовать ресурсы реального сервера, это повышает качество обслуживания.

В приведенном ниже коде описывается создаваемая виртуальная среда, выделяются ресурсы.

Определение названия сервера:

```
   vm_name = models.name
max_length=128,
verbose_name='Название сервера'
```

Определение модели (количества ядер) процессора виртуальной машины.

```
cpu = models.ForeignKey
      'ResourceCPU',
related_name='Процессор'
```

Определение количества памяти, выделенной виртуальной машине от сервера:

```
      'ResourceRAM',
related_  name='всегоRAM',
```

Выделение дискового пространства:

```
hdd = models.ForeignKey
      'ResourceHDD,
related_ name='Всегопамяти'
```

В окне программы выводится статистика о виртуальных машинах, работающих на сервере. Отсюда можно перейти в управление виртуальной машиной. Важная особенность и несомненный плюс виртуализации серверов – администратор видит все виртуальные машины, но сами виртуальные машины, расположенные на одном сервере, не видят друг друга.

monitoring.py – мониторинг и управление виртуальной машиной. В данной части программы представлена подробная информация о ресурсах, выделенных выбранной виртуальной машине. По умолчанию информация обновляется один раз в секунду, но в программе реализована возможность выбора интервала обновления данных. В условиях высокой загруженности сети у администратора будет возможность вернуться к окну управления виртуальными машинами на сервере и добавить вычислительные ресурсы данной

виртуальной машине, что является огромным плюсом в обеспечении высокого качества обслуживания. Предусмотрена и возможность выключения, перезагрузки виртуальной машины.

sdncontrol.py – управление сетью. Представлена общая схема сети, выводится статистика о коммутаторах, их состоянии, а также информация о состоянии контроллера. В случае обнаружения проблем программа сразу же оповестит пользователя.

Рассмотрим подробнее очереди OpenFlow. В соответствии с очередями выполняется автоматическое управление трафиком, пакеты с более высоким приоритетом будут отправлены первыми, а также в соответствии с политикой управления более высокая очередь имеет более высокую минимальную пропускную способность. Если коммутатор будет загружен на 100 %, каждая очередь получит свой процент скорости. Пакетам присваивается очередь в соответствии с приоритетами. Классификация трафика является первым шагом для обеспечения QoS и основывается на DSCP-значениях (Differentiated Services Code Point) в записях, установленных в таблицах OpenFlow-коммутаторов.

DSCP – это точка кода дифференцированных услуг, элемент архитектуры компьютерных сетей, описывающий простой масштабируемый механизм классификации, управления трафиком и обеспечения качества обслуживания. DSCP может применяться, например, для снижения задержки чувствительного сетевого трафика вроде голосовой связи и потокового мультимедиа, тогда как остальной трафик будет идти без приоритизации.

Каждому DSCP-значению соответствует различное действие, например: отбрасывать пакеты, перемещать пакеты на указанный порт, назначить трафик в указанную очередь. DSCP использует 6-битное поле 8-битного IP-заголовка Differentiated Services (DS). Для IPv4 используется пространство устаревшего заголовка Type of Service (ToS), который разделен на поля DSCP и Explicit Congestion Notification (ECN). В терминологии Internet Protocol version 6 (IPv6) заголовок называется TrafficClass, логика его организации совершенно идентична.

Соотнесем категории доступа с DSCP-значениями для организации классификации трафика на коммутаторах SDN и разделим DSCP-значения по приоритетам, организуя в соответствии с этими приоритетами 8 очередей на портах OpenFlow-коммутатора (табл. 2).

Таблица 2. Тип трафика и соответствующее DSCP-значение

| Тип трафика | DSCP-значение | DSCP-приоритет | Очередь OpenFlow |
|---|---|---|---|
| контроль сети | 56 | 7 | 7 |
| межсетевое управление | 48 | 6 | 6 |
| голосовой | 46 | 5 | 5 |
| интерактивное видео | 34 | 4 | 4 |
| потоковое видео | 32 | 4 | 4 |
| критически важные трафики | 26 | 3 | 3 |
| сигнализации вызова | 24 | 3 | 3 |
| транзакционный | 18 | 2 | 2 |
| сетевое управление | 16 | 2 | 2 |
| большой массив данных | 10 | 1 | 1 |
| негарантированная доставка данных | 0 | 0 | 0 |
| опционный класс – Internet/scavenger | 8 | 1 | 1 |

После процесса классификации и назначения приоритета, контроллер будет устанавливать разные правила для разных типов трафика в зависимости от конкретных QoS-политик. С использованием данного механизма можно решить задачу управления мультисервисным трафиком: отправить трафик данных по кратчайшим путям, а мультимедиатрафик – по путям с маленькой задержкой и гарантируемой пропускной способностью. Достоинство этого решения заключается в легкости применения новых QoS-политик через программные приложения, установленные в SDN-сети.

Первое, что видит пользователь при открытии программы, – это окно авторизации. Стандартные данные учетной записи администратора: имя пользователя admin, пароль root. После успешного входа у пользователя появляется возможность управлять ресурсами сервера, создавать и удалять виртуальные машины (рис. 5).

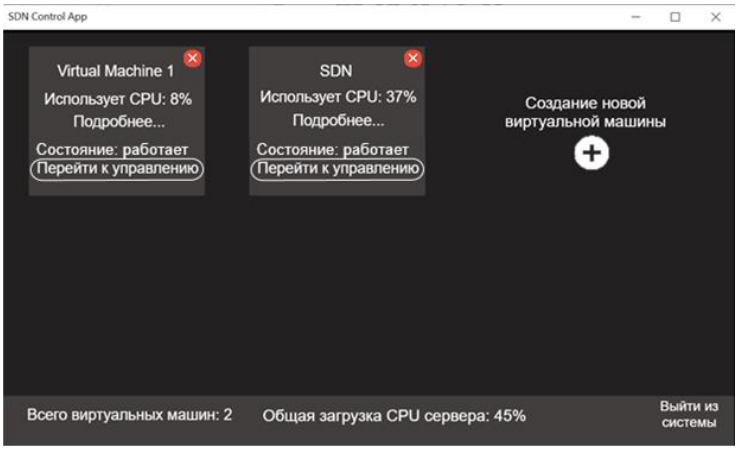


Рис. 5. Интерфейс управления виртуальными машинами на сервере

Перейдем к управлению виртуальной машиной «SDN». В появившемся окне осуществляется мониторинг за основными ресурсами, выделенными виртуальной машине. Контроль за температурой позволяет избежать порчи серверного оборудования (рис. 6). Виртуальную машину можно выключить, для этого необходимо нажать на кнопку «Выключение». В программе реализована защита от случайного выключения: после нажатия кнопки необходимо ввести пароль администратора в качестве подтверждения.

В окне управления сетью представлена схема сети, которая была создана ранее через MiniNet (рис. 7). Слева сконцентрированы элементы управления и мониторинга. По умолчанию установлен режим управления сетью «Гарантирование минимальной пропускной способности на выходе». Ниже показан статус коммутатора, в данном случае он один, но если их будет больше, они все будут показываться в этом окне. Отображается наличие ошибок и загруженность коммутатора. Отображается состояние контроллера сети, также выводится состояние и количество ошибок. Так как контроллер программно определяемой сети – самая важная часть сети, состояние контроллера дополнительно выделено цветом. Настройка хостов показана на рис. 8.

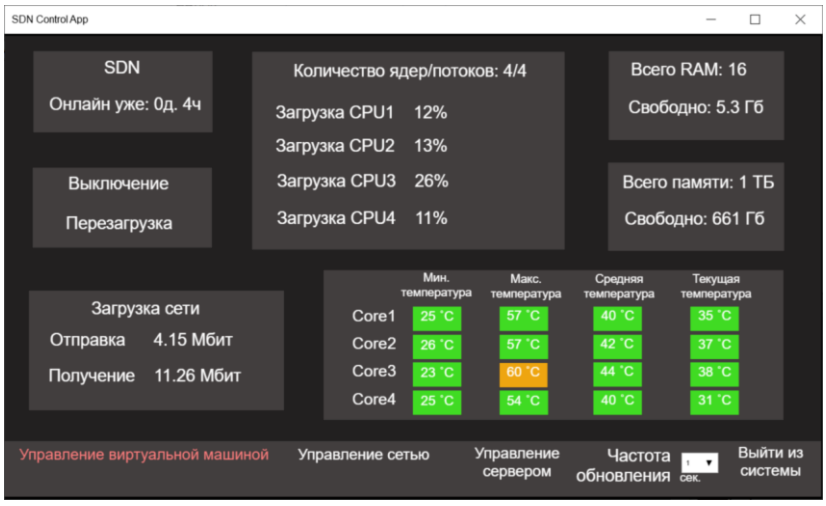


Рис. 6. Интерфейс мониторинга за состоянием виртуальной машины

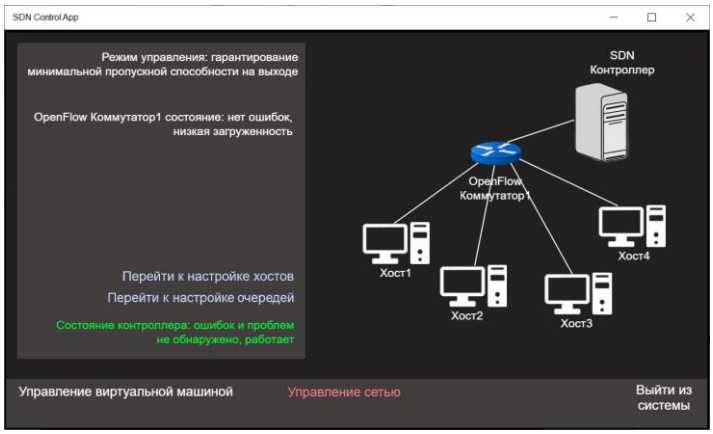


Рис. 7. Интерфейс окна «Управление сетью»

Представлена возможность назначить высокий приоритет одному из хостов (рис. 8). В таком случае трафик от этого хоста будет обрабатываться коммутатором в первую очередь, даже если от других хостов будет поступать трафик с более высоким приоритетом. Устанавливать приоритет необязательно, если он не установлен (именно такая настройка выбрана по умолчанию), то трафик обрабатывается по обычным правилам – в соответствии с приоритетами пакетов.

В окне «Настройка очередей» можно настроить 8 очередей протокола управления. Для изменения необходимо нажать на нужную очередь и ввести значение. Общее значение на всех очередях не может превышать 100 %. При нажатии кнопки «Перейти к настройке очередей», откроется сразу настройка очередей единственного коммутатора – OpenFlow Коммутатор1. Если бы коммутаторов в сети было больше одного, то сначала появился бы выбор коммутатора.

**Тестирование программного продукта УКвОС.** Выполнено ручное тестирование по стратегии «черного ящика». Начинается тестирование ПО с проверки окна авторизации. При введении неверных учетных данных ожидаемый результат – ошибка входа. Проверялась защита от случайного выключения виртуальной машины, результат – виртуальная машина выключится только после ввода верного пароля.

Проверялось, как поведет себя программа при отключении коммутатора. Полученный результат: программа сразу же оповестила администратора о проблеме, четко выделив проблемные элементы в интерфейсе.

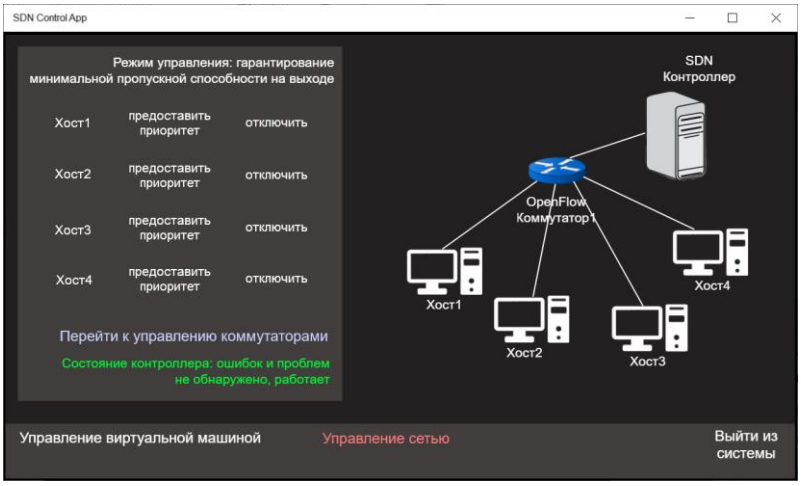


Рис. 8. Интерфейс «Настройка хостов»

Проведено тестирование пункта программы «Настройка хостов». В программе имеется возможность выбрать приоритетный хост. Сделан приоритетным «Хост1», а после – «Хост2».

Полученный результат: при установке приоритета «Хост1» кнопка стала зеленой, при установке приоритета «Хост2» зеленой стала уже кнопка «Хост2», а приоритет «Хост1» автоматически отключился. Проведен аналогичный тест с отключением хоста от сети.

Проверялся пункт «Управление очередью», в руководстве пользователя описано, что программа не разрешит сохранить введенные данные, если сумма всех очередей будет больше 100 %. При задании очереди 4 минимальной пропускной способности 20 % вместо 11 % программа не разрешает сохранить введенное значение, выведено предупреждающее сообщение. Если отменить ввод, то программа вернет то значение, которое было до редактирования.

## Заключение

1. Даны основные положения управления качеством для ПУС. Для программной реализации проекта управления качеством в ПУС разработаны требования к системе, представлены диаграммы вариантов использования и сущность-связь, приведена структура интерфейса.

2. Приведено описание программной реализации управления качеством в ПУС с использованием языка Pyton. Для создания сети использован компонент моделирования MiniNet, применены особенности управления ПУС на базе технологии OpenFlow.

3. В ходе тестирования программы управления качеством в ПУС были проверены все основные функции программного средства, ошибки не выявлены, все тестовые случаи были проверены успешно. Программа выводит дополнительную информацию о возникающих проблемах, что является плюсом для конечного пользователя. Разработанное программное средство может использоваться в учебном процессе кафедры ТКС академии связи.

## DESIGN AND PROGRAMMING OF A QUALITY MANAGEMENT SYSTEM FOR SOFTWARE-DEFINED NETWORKS

U.A. VISHNIAKOU, B.A. MONICH

### Abstract

The concept of quality (QoS) of software-defined networks (SDN) in cloud area is given. Requirements for quality of SDN are defined. The approach to the design of the quality management system in cloud area (QAMiC) is given. Diagrams of use and nature communication for QAMiC system are discused. The structure of the interface of QAMiC tool is considered. Features of programming this system are given, and testing issues are considered.

### Список литературы

1. Программно определяемые сети (Software Defined Networks): настоящее и будущее [Электронный ресурс]. – Режим доступа : https://habr.com/en/company/hpe/blog/160531/. − Дата доступа : 15.11.2019.

2. Дубинин, В. Программно-определяемые сети: от концепции к технической реализации / В. Дубинин // ITWeek. – 2016. – № 5. – С. 15–20.

3. SDN: кому и зачем это надо [Электронный ресурс]. − Режим доступа : https://www.osp.ru/lan/ 2012/12/13033012/. − Дата доступа : 15.11.2019.

4. Вишняков, В. А. Модели и управление качеством программно-определяемых сетей / В. А. Вишняков, Б. А. Монич // Проблемы инфокоммуникаций. – 2019. – № 1. – С. 42–47.

5. PyCharm IDE для Python программистов [Электронный ресурс]. – Режим доступа : https://python-scripts.com/pycharm-download. – Дата доступа : 03.11.2019.

6. Mininet – эмулятор компьютерной сети [Электронный ресурс]. – Режим доступа : https://ivirt-it.ru/mininet/. – Дата доступа : 04.11.2019.

7. Протокол OpenFlow, его применение и оценка безопасности [Электронный ресурс]. – Режим доступа : http://sci-article.ru/stat.php?i=1454985611/. – Дата доступа : 09.11.2019.