МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И КОМПЬЮТЕРНЫХ СЕТЕЙ

УДК 004.724

ИССЛЕДОВАНИЕ СЕТИ ВИРТУАЛЬНОЙ ИНФРАСТРУКТУРЫ ЦЕНТРА ОБРАБОТКИ ДАННЫХ С ГИБРИДНОЙ ПРОГРАММНО-КОНФИГУРИРУЕМОЙ КОММУТАЦИЕЙ

М. В. Ушакова, старший преподаватель кафедры ГКН ОГУ, Оренбург, Россия;
orcid.org/0000-0003-4462-9946, e-mail: m.v.ushakova@mail.ru
Ю. А. Ушаков, к.т.н., доцент кафедры ГКН ОГУ, Оренбург, Россия;
orcid.org/0000-0002-0474-8919, e-mail: unpk@mail.ru

Современные тенденции развития информационных технологий привели к тому, что вся инфраструктура постепенно становится программно-управляемой. Современные гиперконвергентные решения используют программно-конфигурируемые сети и программные коммутаторы для сетевой подсистемы гипервизоров. ***Целью работы*** *является исследование обработки трафика в гиперконвергентных структурах с программной коммутацией на основе традиционных подходов и протокола OpenFlow. Рассмотрены особенности реализации сетевой инфраструктуры гиперконвергентных решений, описаны подходы к изучению программно-управляемых сред. Предложена модель внутренней структуры обработки трафика конвергентного узла, совмещающего функции гипервизора, системы хранения и коммутатора. Разработаны модели интерфейсов, воспроизводящие коммутацию трафика обычным способом и с вышестоящей обработкой OpenFlow. Описываются подходы к реализации разработанных моделей на основе экспериментальных исследований сетевого оборудования. Приводятся результаты экспериментального исследования узла сети и синтезированной модели, показывающие возможность реализации предложенных подходов в рамках заданной точности.*

Ключевые слова: *моделирование, OpenFlow, коммутатор, трафик, QoS.*

DOI: 10.21667/1995-4565-2021-75-34-43

Введение

Сегодня облачные инфраструктуры требуют все более гибких подходов и технологий, которые предполагают переход к более распределенным приложениям, микросервисной архитектуре и гибким IT-моделям. Современные тенденции развития облачных сервисов приводят к широкому распространению программно-управляемой инфраструктуры, хранилищ, сетей, а также к внедрению конвергентных и гиперконвергентных сервисов и оборудования [1]. Конвергенция в сетях совмещает трафик разнородных сервисов на едином стеке технологий с аппаратной составляющей (традиционные коммутаторы с поддержкой приоритезации, сегментации на уровнях 2 – 4, совмещения нескольких протоколов уровня 2), в то время как гиперконвергенция основана на программно-ориентированной архитектуре. Гиперконвергенция не исключает использование конвергентных технологий в сетевом стеке, но единообразное управление сетью как в гипервизорах, системах хранения, виртуальных коммутаторах и маршрутизаторах, так и в оборудовании позволяет намного повысить качество управления потоками, логической сегментацией, безопасностью и приоритезацией. Использование виртуальных выделенных каналов от источника до получателя с гарантированными предельными характеристиками на базе программно-конфигурируемых сетей с коммутацией пакетов позволяет обеспечивать качество обслуживания трафика и безопасность с гораздо меньшими

организационными и конфигурационными затратами, чем на базе традиционных конвергентных сетей [2].

Основной особенностью гиперконвергентных решений разных производителей является совмещение управления всеми сетевыми устройствами в единообразную программную систему, причем большинство производителей добавляют в гиперконвергентные решения поддержку виртуальных наложенных технологий (Overlay Transport), таких как сети VXLAN (Virtual Extensible LAN), Cisco OTV (Overlay Transport Virtualization), а также различные наложенные сети гипервизоров виртуальной инфраструктуры (VMWare NSX-T, Microsoft HNV, OVN, Cisco ACI) и контейнерных оркестраторов (swarm overlay network, flannel, k-vswitch, OVN) [3]. Открытые облачные решения (OpenStack, OpenNebula) и контейнерные решения (Docker Swarm, Rancher, Kubernetes, OpenShift) могут использовать наложенные сети через множество технологий посредством системы плагинов, но самым распространенным является использование VXLAN решений и управление передачей через Open vSwitch.

Использование Open vSwitch в связке с технологиями программно-конфигурируемых сетей (как по протоколу OpenFlow, так и с использованием ovs-ofctl, ovs-vsctl) позволяет объединять виртуальный коммутатор и оборудование в единую управляемую сеть, а также сделать универсальное управление трафиком виртуальных машин и сервисов, работающих непосредственно на оборудовании (bare-metal), например, программно-определяемых систем хранения данных (CEPH, drbd9, gluster, NFS,VMWare vSAN).

В то же время изучение работы таких сетей в связке с гиперконвергентной инфраструктурой является открытой проблемой, слишком много факторов может оказывать влияние на каждый элемент инфраструктуры и тем самым на параметры производительности. Одним из способов всестороннего охвата всех факторов для исследования сетей может выступать моделирование [4], однако оно усложняется высоким уровнем вложенности и взаимной зависимостью компонентов.

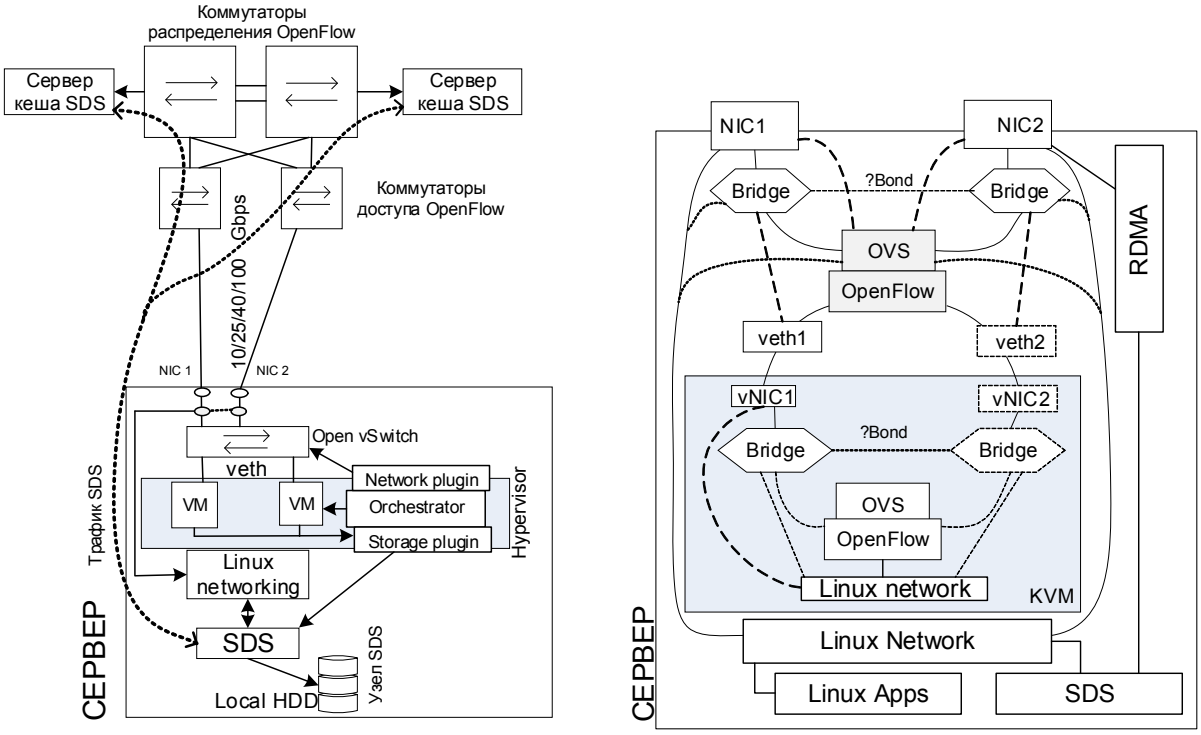
Теоретическая часть

Рассмотрим один узел гиперконвергентной инфраструктуры на основе наиболее распространенных открытых решений – гипервизор KVM, виртуальная коммутация Open vSwitch и Linux Bridge, SDS (software defined storage), программно-определяемое хранилище (блочное, объектное или файловое), основанное на ресурсах тех же серверов, на которых выполняется виртуализация (CEPH, Gluster, ScaleIO, SwiftStack, HDFS и подобные).

Каждый сервер гиперконвергентной инфраструктуры содержит вычислительный узел (процессор, память, диски), операционную систему Linux, сетевую часть (сетевые карты, коммутатор Open vSwitch, мосты Linux Bridge, виртуальные сетевые карты veth), узел виртуализации под управлением внешнего оркестратора, узел сервера SDS с хранением на локальных дисках, интегрированных с общей подсистемой хранения (рисунок 1, *а*). Поскольку вариантов внутренних соединений много (рисунок 1, *б*), необходимо предусмотреть и исследовать все возможные варианты при создании модели [5].

Основной проблемой моделирования сетей большим количеством технологий обработки трафика и обеспечения качества обслуживания является огромное количество возможных сочетаний различных параметров, видов и настроек очередей, политик обработки трафика. Рассмотрим вариант создания сетевого интерфейса с включенной функциональностью VLAN, классификатором в обоих направлениях, входящим и исходящим ограничением скорости через очереди и фильтрацией трафика в обоих направлениях. При этом классификатор и очереди используют DSCP, политикой управления перегрузками выбран WRED (Weighted Random Early Detection), а приоритезация происходит через WRR (Weighted Round Robin) для классов кроме ef, для которого используется LLQ (Low Latency Queue). Ограничение объема входящего трафика также происходит через WRED удаление пакетов из очереди с разными весами разных классов (Class-based policing) [6]. Классификатор настроен на маркировку некоторых пакетов классами af11, af13, af21, af23, ef, be. Вход в сеть происходит через

двунаправленный модуль mac, выход и вход верхних уровней через фильтры (ACL). Данная последовательность действий на входе из сети (декапсуляция, классификация, работа с полосой пропускания, маркировка, фильтрация) и на выход из сети (фильтрация, маркировка, работа с полосой пропускания, очереди, инкапсуляция) создана на основе описания последовательности действий Cisco IOS [7] (рисунок 2).



а(a) **б(b)**

Рисунок 1 – Схема внутренней организации сервера: а – схема соединений компонентов; б – схема возможных внутренних сетевых соединений

Figure 1 – Server internal organization diagram: a – component connection diagram; b – diagram of possible internal network connections

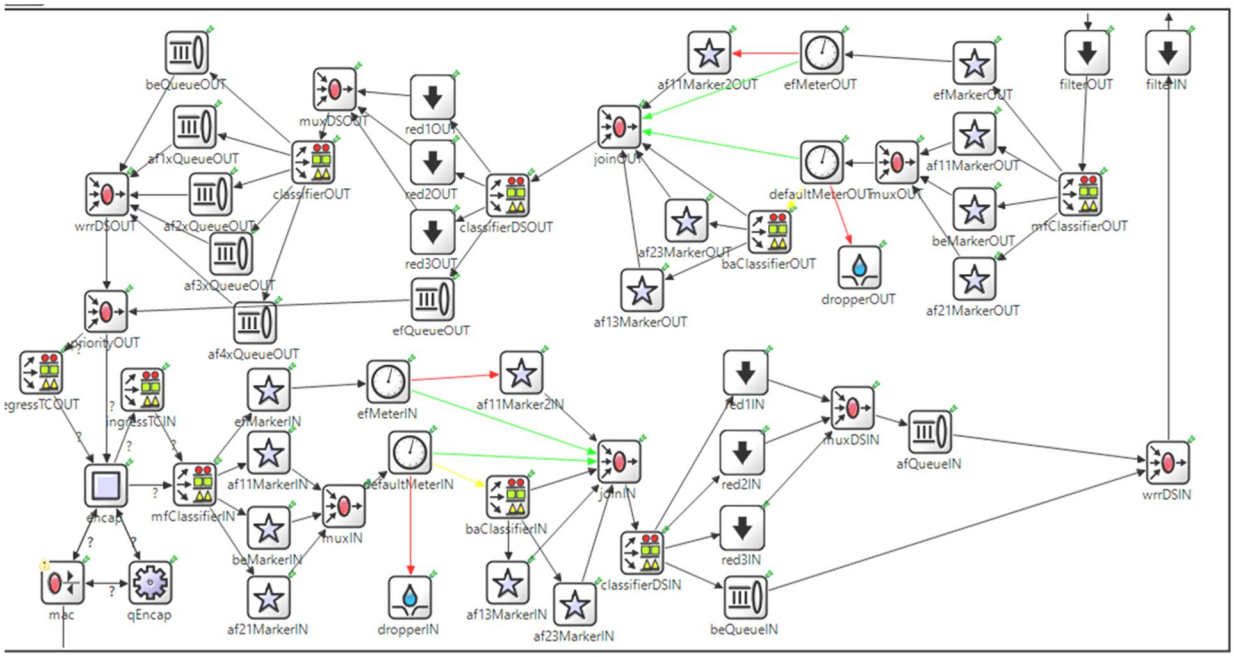


Рисунок 2 – Схема прохождения пакета сквозь сетевой интерфейс

Figure 2 – The scheme of the packet passing through the network interface

Для создания реалистичной задержки использовался созданный модуль типа TrafficConditioner для помещения на место ingressTCIN и egressTCOUT компонентов. Он используется для внесения задержки во входящем направлении, соответствующей времени обработки пакета коммутатором на реальной сети. При этом использование других сочетаний настроек очередей, их маркировки и логики соединений (например, вложенные очереди WRR) может существенно усложнить модель интерфейса.

Использование OpenFlow для обработки трафика в общем случае не влияет на схему интерфейса в части работы с очередями, но убирает часть классификации и маркировки, которая теперь выполняется на уровне OpenFlow Data Plane. В общем случае QoS реализован в OpenFlow только как классификация и маркировка пакетов и измерение текущих показателей потоков [8]. Управление параметрами очередей, ограничителями трафика, политиками приоритезации в большинстве реализаций оставлено для традиционной настройки через механизмы операционной системы коммутатора (или программу управления, как в случае с Open vSwitch). Некоторые контроллеры (например, OpenDaylight, ONOS, Floodlight) имеют механизмы настройки очередей на некотором наборе коммутаторов через NETCONF или напрямую командами (например, для OVSDB). Но это не решает проблему создания модели такого оборудования. На рисунке 3 показана схема работы интерфейса в OpenFlow режиме, вся маркировка и ограничение скорости происходит в ядре OpenFlow, однако очереди и работа с ними, в том числе приоритезация, остались прежними.

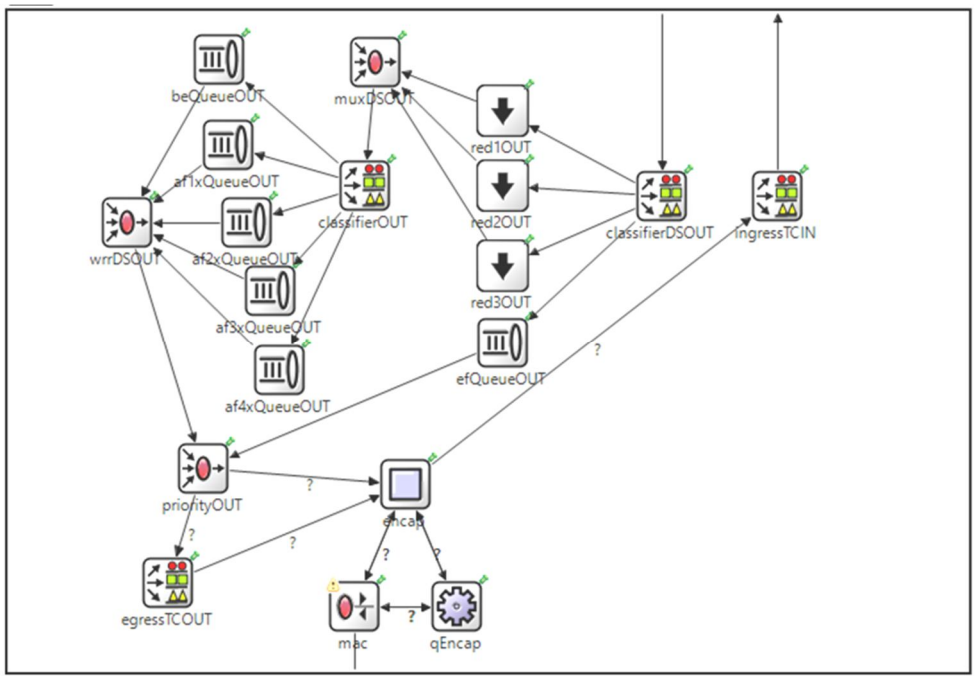


Рисунок 3 – Схема прохождения пакета сквозь сетевой интерфейс OpenFlow с поддержкой QoS
Figure 3 – The scheme of the packet passing through OpenFlow network interface with QoS support

Для изучения параметров работы программно-конфигурируемого устройства на промежутке времени T обозначим заданные входные параметры, описывающие входящий трафик и текущие настройки устройства, как $Z(t)=\{Traffic(t), Device(t)\}, \forall t \in T,$ а измеренные выходные параметры, включающие в себя временные характеристики, параметры входных и выходных очередей, статистику по очередям и слепок метрик устройства, как $Y(t)=\{Latency(t), Qout_{ij}(t), S_{ij}(t), State'_i(t)\}, \forall t \in T$.

Тогда функция отображения входных параметров на выходные примет вид

$$F(t): Z(t) \to Y(t), \forall t \in T. \quad (1)$$

Представим дерево входных параметров $Z(t)$ в виде линейной развертки

$$Z(t) = \{z_1(t), z_2(t) \ldots z_n(t)\}, \forall t \in T,$$

где $n \in N$ – количество входных параметров в развертке.

Аналогично представим дерево выходных параметров $Y(t)$ в виде линейной развертки

$$Y(t) = \{y_1(t), y_2(t)...y_m(t)\}, \forall t \in T,$$

где $m \in N$ – количество выходных параметров в развертке.

Входное множество параметров $Z(t_i)$ при $\forall t_i \in [T_{i-1}..T_i) \in T$ не должно изменяться, а длительность интервала времени $[T_{i-1}..T_i)$ должна быть достаточна для требуемого количества измерений выходных параметров и достижения стационарности вероятностных закономерностей выходных параметров $\{y_1(t_i), y_2(t_i)...y_m(t_i)\}$.

Тогда отображение (1) можно привести к виду $F(t) = \{f_1(t), f_2(t)...f_m(t)\}$ так, чтобы

$$\begin{cases} f_1(t_i) : \{z_1(t_i), z_2(t_i)...z_n(t_i)\} \to y_1(t_i) \\ f_2(t_i) : \{z_1(t_i), z_2(t_i)...z_n(t_i)\} \to y_2(t_i) \\ ... \\ f_k(t_i) : \{z_1(t_i), z_2(t_i)...z_n(t_i)\} \to y_k(t_i). \end{cases}$$

При моделировании необходимо обеспечить соответствие получаемых выходных параметров изучаемого устройства $Y(t_i)$ и получаемых выходных параметров модели $Y^*(t)$ при одних и тех же наборах входных параметров. Для этого найдем такое отображение $F(t) : Z(t) \to Y^*(t), \forall t \in T,$ что

$$\left| \frac{y_j(t_i) - y_j{}^*(t_i)}{y_j(t_i)} \right| < \varepsilon, \forall j \in [1..m],$$

где $y_j^*(t_i) \in Y^*(t_i)$ – измеренные выходные параметры модели; ε – заданная точность.

Исследование устройства при всех сочетаниях входных параметров $Z(t)$ осложняется тем, что количество всех возможных сочетаний параметров зависит от количества и диапазонов параметров конкретных устройств с конкретными версиями системного ПО и в общем виде является NP-полной задачей [9]. Поэтому для построения модели будет использоваться метод определения показателей при граничных значениях и некотором количестве промежуточных значений каждого параметра в отдельности, а также будет исследоваться зависимость между параметрами при максимально возможных значениях. Это позволит получить интерполяционную оценку значений показателей для модели с точностью, обусловленной количеством промежуточных измерений параметров и формой кривых зависимостей.

Экспериментальные исследования

Изучение работы модельных устройств, объединенных в сеть, проводилось на примере нового экспериментального сегмента центра обработки данных (ЦОД) ОГУ. Виртуальные машины в ЦОД делятся на группы, между которыми должен быть исключен сетевой доступ. Таких групп в зависимости от времени и потребностей может быть более ста, причем в группах могут быть свои маршрутизаторы, DHCP-серверы, повторяющиеся VLAN и VXLAN сети. Поэтому вместо создания наложенных сетей посредством встроенных в OpenNebula возможностей принято решение изолировать трафик через OpenFlow на физическом и канальном уровнях.

В качестве среды моделирования использовался инструмент OMNeT++ с фреймворком INET, являющийся наиболее подходящим инструментом для моделирования сетей с поддержкой ПКС. Особенностью OMNeT++ является декомпозиция модели до уровня 1 модели OSI с максимальной детализацией всех используемых технологий, протоколов, форматов данных [10].

Сегмент реализован для обеспечения высокопроизводительной обработки данных и создания частного облака для учебных и производственных задач, в том числе для записи и обработки видеопотоков, сохранения дампов трафика Wi-Fi клиентов, создания учебных вир-

туальных машин, задач обработки больших данных. В состав сегмента ЦОД входят два коммутатора, играющих роль уровня распределения. Эти коммутаторы соединяются каналами 10 GbE с каждым из семи коммутаторов доступа. Каждый сервер соединяется с двумя коммутаторами доступа двумя каналами 10 GbE (сегмент хранения данных и сегмент коммутации) и двумя каналами 1 GbE (управление). Поскольку система хранения построена на базе CEPH, основанной на дисках тех же серверов, что используются для запуска всех виртуальных машин, требования к сегмента хранения приводят к необходимости обеспечить пиковую скорость узла не менее 10 – 15 Гбит/с при обмене на канальном уровне и задержке. При записи в хранилище происходит дупликация на несколько серверов, кроме этого, происходит регулярная перебалансировка и шардинг данных, что существенно увеличивает трафик. Сегмент коммутации используется для межсетевого взаимодействия виртуальных машин с внешними потребителями и между собой. При использовании распределенных нейросетей и обращении к хранилищам на базе Hadoop, которые физически хранят данные на отдельных накопителях тех же серверов, требования к производительности сегмента коммутации также имеют ограничения снизу.

На рисунке 4 показана топология экспериментального сегмента ЦОД.

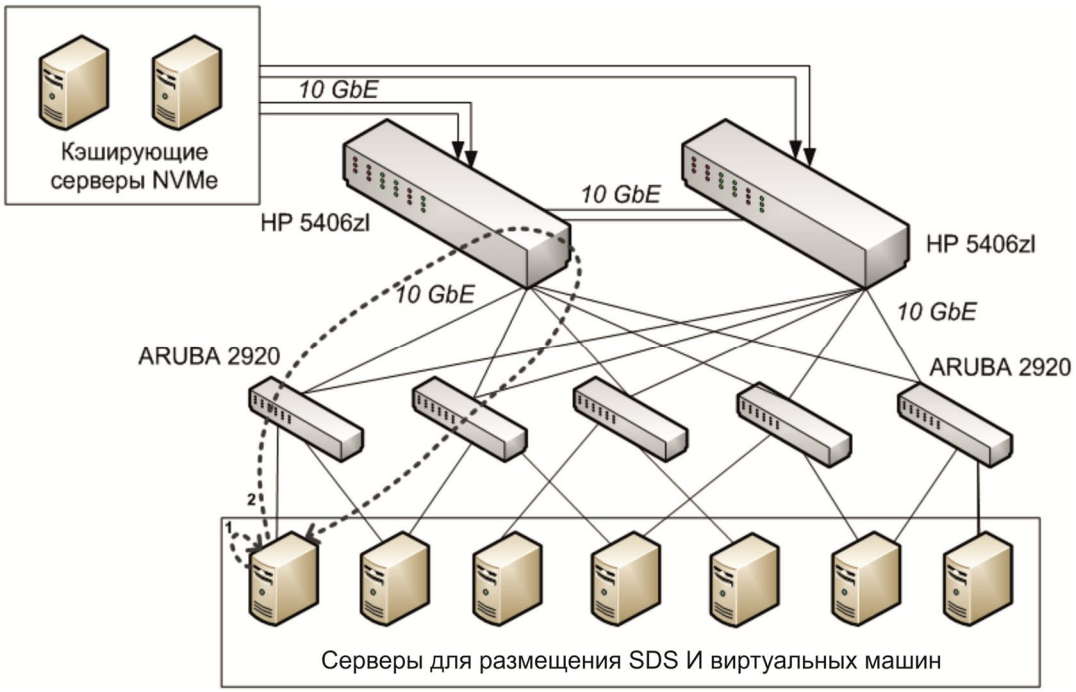


Рисунок 4 – Топология экспериментального сегмента ЦОД
Figure 4 – Experimental segment topology

На момент исследований в экспериментальный сегмент включались семь серверов (Xeon Gold, 128 Гб ОЗУ, NVMe SSD, четыре сетевые платы 10 Гбит/с с аппаратной обработкой пакетов) для размещения системы хранения данных и виртуальных машин, два сервера для работы только в качестве систем хранения, пять коммутаторов доступа (HPE ARUBA 2920) и два коммутатора распределения (HP ProCurve Switch 5406zl с пятью модулями HP J9309A ProCurve 4-Port 10GbE SFP+ zl). Виртуальные машины работают под управлением системы оркестрации OpenNebula, в качестве виртуальных коммутаторов используется Open vSwitch в режиме гибридной коммутации с проактивным режимом OpenFlow (правила устанавливаются на локальный серверный коммутатор оркестратором при выполнении действий с виртуальными машинами или сетями).

С помощью скриптов, автоматизирующих соединение сетевых компонентов (рисунок 1, *б*) исследованы задержки в экспериментальном сегменте ЦОД при значениях интенсивности генерации трафика *Traffic*(*t*) в различные промежутки времени *t*. Для этого после

запуска всех необходимых компонентов скрипт запускает перехват трафика tcpdump с записью на tmpfs диск, а также генерацию трафика утилитой trafgen на различных интенсивностях. Во время эксперимента происходило снятие копий состояния устройства по протоколу SNMP и OpenFlow метрикам.

На основе топологии, показанной на рисунке 4, в системе моделирования OMNeT++ сформирована модель данной сети с использованием стандартных средств и проведено исследование задержек в модельной сети (рисунок 5).

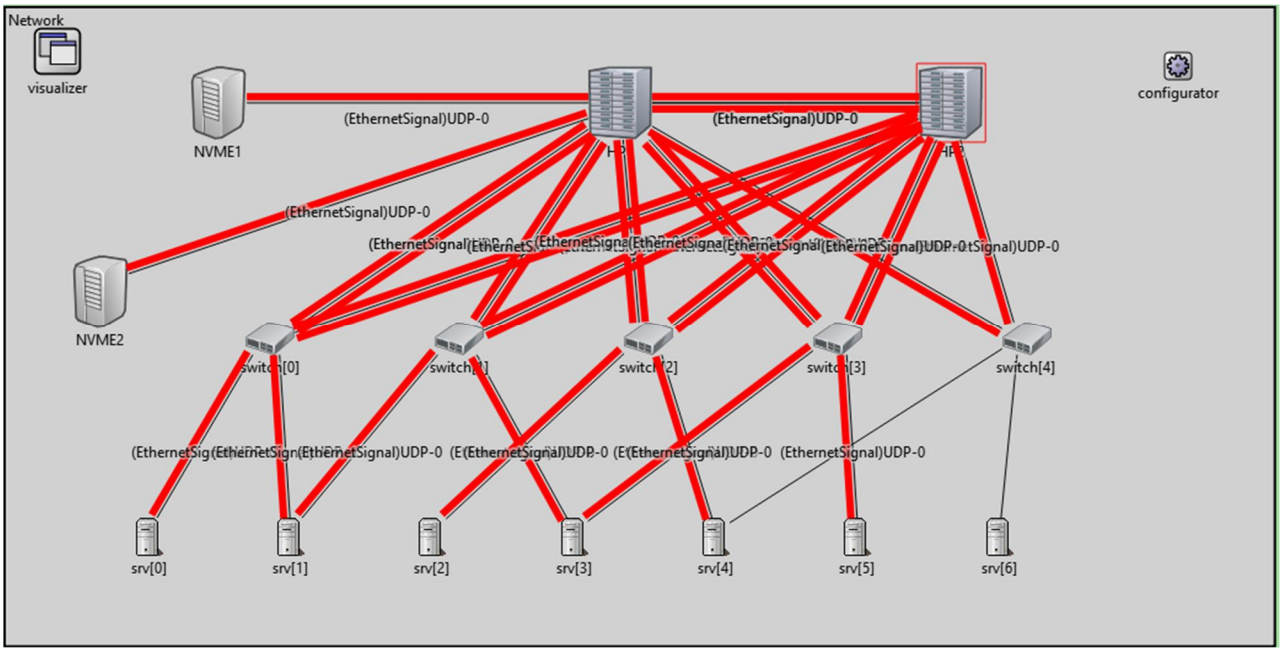


Рисунок 5 – Модель сети в OMNeT++
Figure 5 – Network model in OMNeT ++

С помощью разработанных скриптов автоматизации эксперимента проведено исследование всех моделей сетевых устройств и серверов, входящих в состав исследуемой сети, и синтезированы модели этих устройств.

Для исследования задержек на модели сети был синтезирован план генерации трафика, аналогичный экспериментам на оборудовании. На рисунке 6 приведена графическая интерпретация результатов исследования задержки в сети без использования очередей (на оборудовании очереди были минимизированы до 1 пакета на очередь, мультиплексирование в один канал исключено), ее модели, полученной с помощью стандартных средств системы моделирования OMNeT++, а также модели, полученной средствами разработанного программно-аппаратного комплекса. Исследование проводилось при различных сценариях работы (различных путях, показанных на рисунке 1, *б*) для различных интенсивностей генерации трафика.

Задержки вносились модулем ingressTCIN, интерполировавшим полученные времена обработки пакетов при экспериментальном исследовании на текущее состояние коммутатора и имеющиеся включенные настройки (VLAN, очереди, классификаторы, маркировщики).

В таблице показаны сводные результаты измерения задержек в реальной сети, модели сети, полученной стандартными средствами системы моделирования, и модели сети.

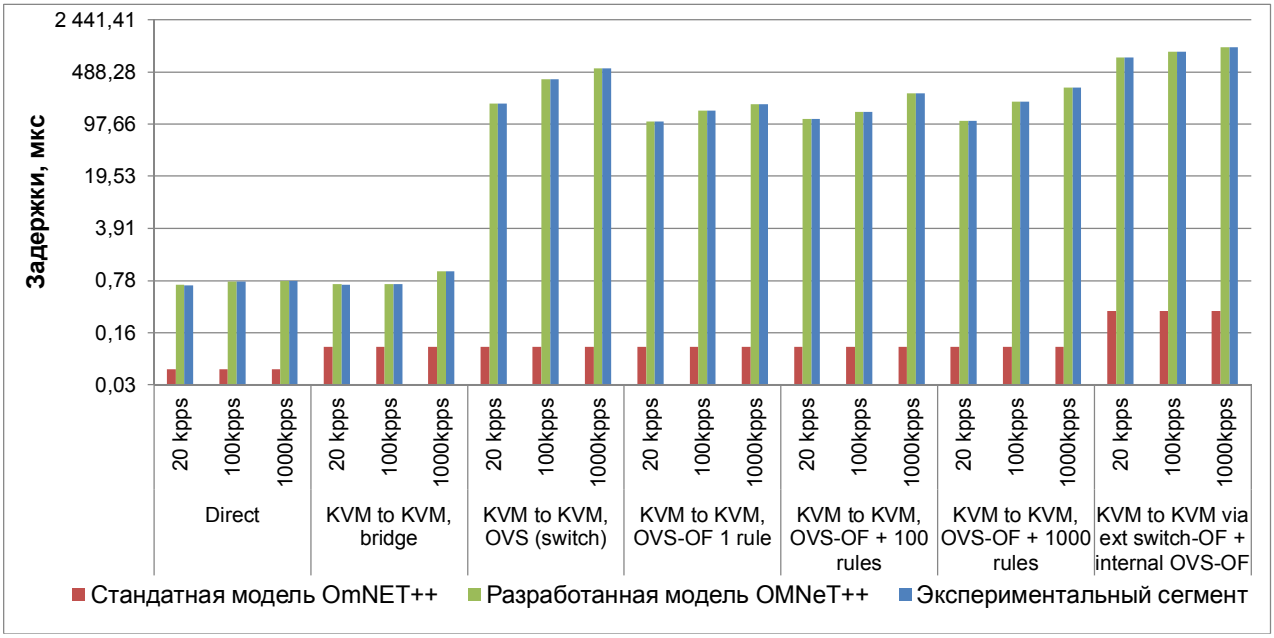


Рисунок 6 – Результаты исследования задержки в сети и ее моделях
Figure 6 – Results of studying the latency in a network and its models

Таблица – Средняя задержка, мкс
Table – Average latency, µs

Режим работы	Интенсивность	Реальная сеть	Стандартная модель сети OMNeT++	Разработанная модель сети OMNeT++
Прямое соединение сетевых карт сервера	20 kpps	0,679	0,051	0,684
	100 kpps	0,757	0,051	0,762
	1000 kpps	0,769	0,051	0,774
Между виртуальными машинами через Linux bridge	20 kpps	0,687	0,102	0,697
	100 kpps	0,696	0,102	0,706
	1000 kpps	1,042	0,102	1,052
Между виртуальными машинами через OVS в режиме обычной коммутации (switch)	20 kpps	185,019	0,102	185,029
	100 kpps	390,493	0,102	390,504
	1000 kpps	548,669	0,102	548,679
Между виртуальными машинами через OVS в режиме OpenFlow коммутации (pro-active) с 1 правилом	20 kpps	104,913	0,102	104,923
	100 kpps	146,284	0,102	146,294
	1000 kpps	179,147	0,102	179,157
Между виртуальными машинами через OVS в режиме OpenFlow коммутации (pro-active) + 100 правил	20 kpps	114,155	0,102	114,165
	100 kpps	140,678	0,102	140,689
	1000 kpps	253,802	0,102	253,812
Между виртуальными машинами через OVS в режиме OpenFlow коммутации (pro-active) + 1000 правил	20 kpps	107,267	0,102	107,277
	100 kpps	193,589	0,102	193,599
	1000 kpps	298,667	0,102	298,677
Между виртуальными машинами через внешний коммутатор OpenFlow и внутренний OVS в режиме OpenFlow коммутации (pro-active) + 4 правила на каждом	20 kpps	754,384	0,306	754,394
	100 kpps	920,220	0,306	920,230
	1000 kpps	1044,120	0,306	1044,130

Согласно полученным результатам, при использовании входящих в поставку OMNeT++ моделей коммутатора не представляется возможным исследовать влияние времени обработ-

ки фрейма коммутатором на общую задержку передачи пакета в сети. Сравнение результатов исследования модели, полученной с использованием средств разработанного программно-аппаратного комплекса, с задержками экспериментального сегмента ЦОД показало совпадение на аналогичных интенсивностях генерации фреймов в пределах 5 %.

Заключение

В статье предложена методика исследования процессов прохождения трафика в гиперконвергентных структурах с программной коммутацией на основе традиционных подходов и OpenFlow таблиц. Созданы модели интерфейсов для коммутации трафика обычным способом и с вышестоящей обработкой OpenFlow. Для получения вероятностно-временных характеристик обработки пакетов оборудованием предложен метод первоначального синтеза модели оборудования с разработанной моделью интерфейса, также учтена возможность прохождения трафика по различным виртуальным путям внутри виртуальной инфраструктуры. Полученные результаты исследования и моделирования показали существенное влияние настроек и размеров таблиц OpenFlow элементов виртуальной инфраструктуры на время обработки пакета, в том числе нелинейную форму зависимости. Использование интерполяции результатов экспериментального исследования в модели для обеспечения требуемой формы зависимости времени обработки пакета показало высокую точность моделирования на схожих условиях и интенсивностях генерации трафика в пределах 5 %.

Работа выполнена при финансовой поддержке гранта РФФИ (*проект №20-57-53019*) *и гранта Президента РФ для государственной поддержки ведущих научных школ Российской Федерации* (*проект № НШ-2502.2020.9*).

Библиографический список

1. **Melo C., Dantas J., Maciel P., Oliveira D. M., Araujo J., Matos R., Fé I.** Models for hyperconverged cloud computing infrastructures planning // International Journal of Grid and Utility Computing. 2020, no. 11 (2), pp. 196-208.

2. **Bhaumik S., Chanda K., Chakraborty S.** Poster: Controlling Quality of Service of Container Networks in a Hyperconverged Platform // 2020 IFIP Networking Conference (Networking). IEEE, 2020, pp. 661-663.

3. **Meneses-Narvaez S. E., Dominguez-Limaico P., Rosero-Montalvo D., Narvaez-Pupiales S., Vizuete M. Z., Peluffo-Ordónez D. H.** Design and Tests to Implement Hyperconvergence into a DataCenter: Preliminary Results // Advances in Emerging Trends and Technologies. 2019, no. 1, vol. 1066, pp. 54-67.

4. **Кутузов О. И., Татарникова Т. М.** К анализу парадигм имитационного моделирования // Научно-технический вестник информационных технологий, механики и оптики. 2017. Т. 17. №. 3. С. 552-558.

5. **Ушакова М. В., Ушаков Ю. А., Болодурина И. П., Тарасов В. Н., Бахарева Н. Ф.** Исследование параметров производительности виртуальных коммутаторов с поддержкой OpenFlow // Инфокоммуникационные технологии. 2020. Т. 18, № 4.

6. **Bök P. B., Noack A., Müller M., Behnke D.** Quality of Service. In Computernetze und Internet of Things // Springer Vieweg, Wiesbaden. 2020, pp. 251-291.

7. **Szigeti T., Zacks D., Falkner M., Arena S.** Cisco Digital Network Architecture: Intent-based Networking for the Enterprise. Cisco Press. 2018.

8. **Karakus M., Durresi A.** Quality of service (QoS) in software defined networking (SDN): A survey // Journal of Network and Computer Applications. 2017, vol. 80, pp. 200-218.

9. **Lei Y., Tai K.C.** In-parameter-order: A test generation strategy for pairwise testing //Proceedings Third IEEE International High-Assurance Systems Engineering Symposium. 1998. С. 254-261.

10. **Хабаров С. П.** Моделирование Ethernet сетей в среде OMNeT++ INET framework // Научно-технический вестник информационных технологий, механики и оптики. 2018. Т. 18. № 3. С. 462-472.

UDC 004.724

RESEARCH OF VIRTUAL INFRASTRUCTURE NETWORK OF DATA PROCESSING CENTER WITH HYBRID SOFTWARE-CONFIGURABLE SWITCHING

M. V. Ushakova, Lecturer, OSU, Orenburg, Russia;
orcid.org/0000-0003-4462-9946, e-mail: m.v.ushakova@mail.ru
Y. A. Ushakov, Associate Professor, OSU, Orenburg, Russia;
orcid.org/0000-0002-0474-8919, e-mail: unpk@mail.ru

Modern trends in the development of information technology have led to the fact that entire infrastructure is gradually becoming software-controlled. Modern hyper-converged solutions use software-defined networking and soft switches for a hypervisor networking subsystem. ***The aim of the work*** *is to study traffic processing in hyperconverged structures with software switching based on traditional approaches and OpenFlow. The features in the implementation of hyperconverged solutions network infrastructure are considered, approaches to the study of software-controlled environments are described. A model of internal structure of converged node traffic processing combining the functions of hypervisor, storage system and switch is proposed. Interface models that reproduce traffic switching in usual way and with higher-level OpenFlow processing are developed. The approaches to the implementation of developed models based on experimental studies of network equipment are described. The results of an experimental study of network node and a synthesized model showing the possibility of implementing the proposed approaches within the specified accuracy are presented.*

Key words: *modeling, OpenFlow, switch, traffic, QoS.*

DOI: 10.21667/1995-4565-2021-75-34-43

References

1. **Melo C., Dantas J., Maciel P., Oliveira D. M., Araujo J., Matos R., Fé I.** Models for hyper-converged cloud computing infrastructures planning. *International Journal of Grid and Utility Computing. 2020*, no. 11 (2), pp. 196-208.

2. **Bhaumik S., Chanda K., Chakraborty S.** Poster: Controlling Quality of Service of Container Networks in a Hyperconverged Platform. *2020 IFIP Networking Conference (Networking). IEEE*, 2020, pp. 661-663.

3. **Meneses-Narvaez S. E., Dominguez-Limaico P., Rosero-Montalvo D., Narvaez-Pupiales S., Vizuete M. Z., Peluffo-Ordónez D. H.** Design and Tests to Implement Hyperconvergence into a DataCenter: Preliminary Results. *Advances in Emerging Trends and Technologies*. 2019, no. 1, vol. 1066, pp. 54-67.

4. **Kutuzov O. I., Tatarnikova T. M.** K analizu paradigm imitatcionnogo modelirovaniya. *Nauchno-technicheskiy vestnik informatcionnyh technologiy, mekhaniki i optiki*. 2017, vol. 17, no. 3, pp. 552-558. (in Russian).

5. **Ushakova M. V., Ushakov Y. A., Bolodurina I. P., Tarasov V. N., Bakhareva N. F.** Issledovanie parametrov proizvoditelnosty virtualnyh kommutatorov s poddergkoy OpenFlow. *Infokommunikatcionnye technologii*. 2020. vol. 18, no. 4. (in Russian).

6. **Bök P. B., Noack A., Müller M., Behnke D.** Quality of Service. *In Computernetze und Internet of Things. Springer Vieweg, Wiesbaden*. 2020, pp. 251-291.

7. **Szigeti T., Zacks D., Falkner M., Arena S.** *Cisco Digital Network Architecture: Intent-based Networking for the Enterprise*. Cisco Press. 2018.

8. **Karakus M., Durresi A.** Quality of service (QoS) in software defined networking (SDN): A survey. *Journal of Network and Computer Applications*. 2017, vol. 80, pp. 200-218.

9. **Lei Y., Tai K.C.** In-parameter-order: A test generation strategy for pairwise testing. *Proceedings Third IEEE International High-Assurance Systems Engineering Symposium.* 1998, pp. 254-261.

10. **Khabarov S. P.** Modelirovanie Ethernet setey v srede OMNeT++ INET framework. *Nauchno-technicheskiy vestnik informatcionnyh technologiy, mekhaniki i optiki*. 2018, vol. 18, no. 3, pp. 462-472. (in Russian).