ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
СПбГУТ))) Декабрь 2019 Том 7 № 4 ISSN 2307-1303 http://www.itt.sut.ru
TELECOM IT
December 2019 Vol. 7 Iss. 4 ISSN 2307-1303 http://www.itt.sut.ru

МОДЕЛИРОВАНИЕ ОБЛАЧНОЙ СИСТЕМЫ 5G С БАЛАНСИРОВЩИКОМ НАГРУЗКИ

С. С. Полевич*, О. А. Симонина

Санкт-Петербургский государственный университет телекоммуникаций
им. проф. М. А. Бонч-Бруевича,
Санкт-Петербург, 193232, Российская Федерация

*Адрес для переписки: sergeypolevich@mail.ru

Аннотация
Работа направлена на организацию виртуализации и разработку балансировки нагрузки в сети 5G. На основе существующих программных и аппаратных решений построена виртуальная сеть, отражающая общую концепцию организации облачного RAN. Описан базовый функционал балансировки нагрузки в виде перенаправления запросов в зависимости от типа запрашиваемого трафика и расширенный – ограничения доступа к каналу/ресурсу для нескорых видов абонентов. Показано, что такая система может быть реализована на основе ОС Linux, сервера Nginx с модулем upstream. Для наглядности эксперимента используется Docker-контейнер, поддерживающий структуру слоев, на каждом из которых запущен PHP-обработчик, определяющий работу балансировщика.

Ключевые слова
5G, виртуализация, облако, балансировка нагрузи, фильтрация трафика.

Информация о статье
УДК 621.396
Язык статьи – русский.
Поступила в редакцию 23.03.2019, принята к печати 30.12.19.

Ссылка для цитирования: Полевич С. С., Симонина О. А. Моделирование облачной системы 5G с балансировщиком нагрузки // Информационные технологии и телекоммуникации. 2019. Том 7. № 4. С. 30–36. DOI 10.31854/2307-1303-2019-7-4-30-36.

SIMULATION OF A 5G CLOUD SYSTEM WITH A LOAD BALANCER

S. Polevich *, O. Simonina

The Bonch-Bruevich Saint-Petersburg State University of Telecommunications,
St. Petersburg, 193232, Russian Federation

*Corresponding author: sergeypolevich@mail.ru

Abstract—The work is aimed at organizing virtualization and developing load balancing in a 5G network. Based on existing software and hardware solutions, a virtual network is built that reflects the general concept of organizing a cloud RAN. The basic functionality of load balancing is described in the form of redirecting requests depending on the type of requested traffic and advanced – restrictions on access to the channel / resource for several types of subscribers. It is shown that such a system can be implemented on the basis of Linux OS, Nginx server with upstream module. To illustrate the experiment, a Docker container is used that supports the structure of layers, on each of which a PHP handler is launched that determines the operation of the balancer.

Keywords—5G, virtualization, cloud, load balancing, traffic filtering.

Article info
Article in Russian.
Received 23.03.2019, accepted 30.12.19.

For citation: Polevich S., Simonina O.: Simulation of a 5G Cloud System with a Load Balancer // Telecom IT. 2019. Vol. 7. Iss. 4. pp. 30–36 (in Russian). DOI 10.31854/2307-1303-2019-7-4-30-36.

Известно, что сеть 5G будет построена на основе виртуализации сетевых элементов и сервисов, посредством технологии Network Functions Virtualization[1]. Общая схема перехода от классической сети к сети, построенной посредством технологии NVF показана на рис. 1 (см. ниже).

Предполагается реализовать централизованную систему управления трафиком и облачную структуру, организованную посредством технологии Software Defined Networking [1]. Таким образом, обработка всех запросов возлагается на облачную сеть, по которой требуется распределить запросы и сбалансировать потоки трафика, тем самым оптимизировав нагрузку на сеть [2].

Целью балансировки нагрузки является распределение запросов между вычислительными облачными системами в зависимости от типа запрашиваемого трафика (услуг), так, чтобы избежать перегрузки, сократить задержки и максимизировать пропускную способность [3]. Аналогично реализована работа веб-сервера Nginx, когда запросы анализируются и распределяются по облачным слоям [4].

В качестве начальных условий предположим наличие физического сервера, функционирующего на основе ядра Linux с установленным docker-контейнером и PHP-обработчиком, моделирующими расслоение сети[2].

[1] 3GPP Release 15 Specification #: 21.915 SA#83 V:1.1.0 2019-06-21.
[2] Nginx: документация URL: https://nginx.org/ru/docs/

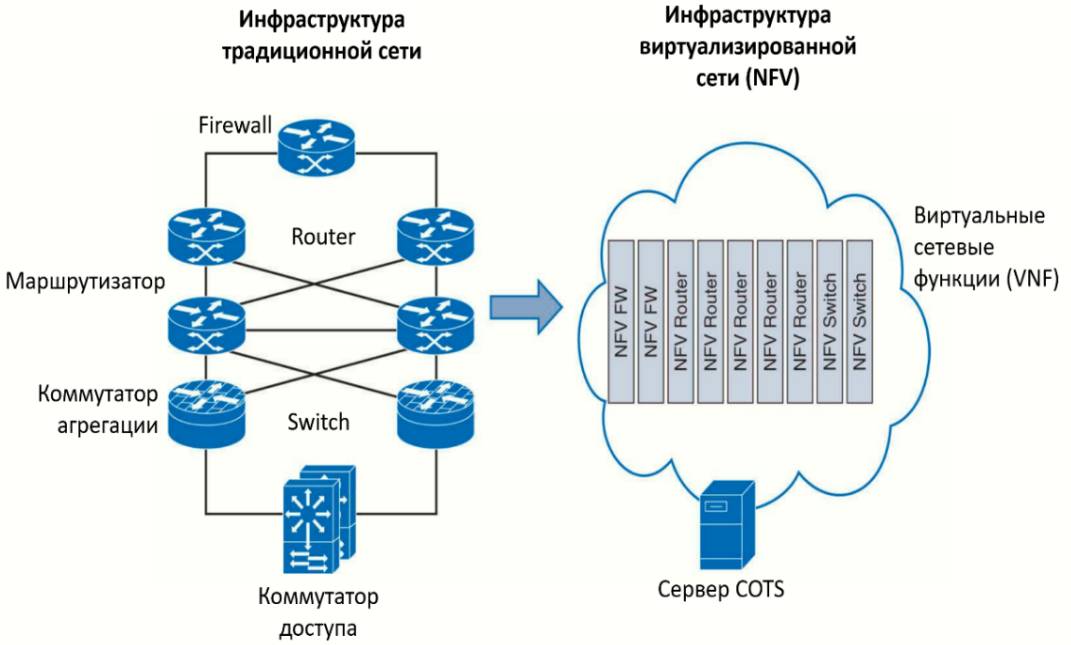

Рис. 1. Общая схема перехода от классической сети к сети, построенной посредством технологии NVF[3]

Каждому PHP-обработчику будет соответствовать свой вид трафика, например, такие:

1. запросы к online-RPG видеоиграм, которым будет соответствовать производительный обработчик с версией 7.3;
2. запросы к обычным web-играм, для которых установим слой с менее производительной версией 7.2;
3. запросы к приложениям, которые должны обрабатываться на еще менее производительной версии 7.1.

Таким образом, разрабатываемая система примет вид согласно рис. 2.

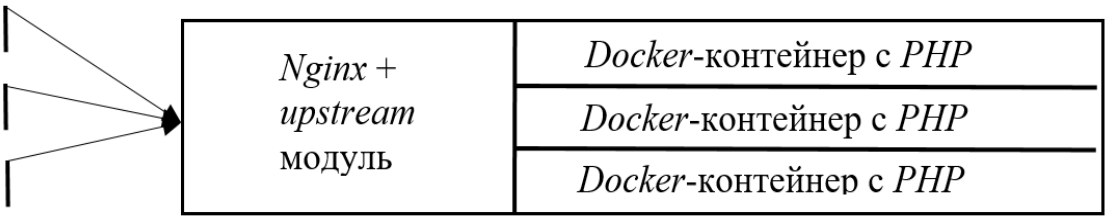

Рис. 2. Схема проектируемой модели

Чтобы nginx узнал о направленном к нему запросе, нужно прослушивать запросы на определённые *IP*-адрес и порт, для чего служит директива listen.

listen 87.236.16.227:80 http1;

Настройку балансировщика следует начать с определения типа трафика, который запрашивается на входе, и адреса обрабатывающего его слоя. Определить адреса всех слоев, чтобы определить направление передачи обработки данных,

[3] 3GPP Release 15 Specification #: 21.915 SA#83 V:1.1.0 2019-06-21.

можно посредством модуля upstream, первым параметром которого является название сервера обработчика, а вторым – его IP-адрес [4]:

```
upstream sergeyt0__onlinerpg.reshi-ticket.ru {server 127.0.0.73:80;}
upstream sergeyt0__web-game.reshi-ticket.ru { server 127.0.0.72:80;}
upstream sergeyt0__web-applications.reshi-ticket.ru {server 127.0.0.71:80;}
```

Все адреса являются локальными, что соответствует облачной структуре, то есть каждая обрабатывающая запросы вычислительная мощность работает в отдельном облаке, но в рамках одного физического сервера[4].

Таким образом создано адресное пространство имен сущностей, которым будет передан запрос при вызове директивы location, которая участвует в определении типа запрашиваемого трафика. Определив location, запрос нужно передать нужному облаку используя директиву proxy_pass. Далее следует задать некоторые переменные модуля proxy_pass, чтобы успешно передать запрос в нужный контейнер. Делается это посредством директивы proxy_set_header, которая позволяет переопределять или добавлять поля заголовка запроса, передаваемые облаку. В качестве значения можно использовать текст, переменные и их комбинации. Директивы наследуются с предыдущего уровня [4].

Завершение настройки следует за дублированием записей, для всех типов определяемого трафика, при этом переменные, переданные через заголовки proxy_set_header остаются без изменений, а меняются лишь поля location и proxy_pass. Пример:

```
location /WEB-applications/ {
proxy_pass http://sergeyt0__web-applications.reshi-ticket.ru;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Server-Addr $server_addr;
        proxy_set_header X-Forwarded-Proto $proxy_scheme;
       proxy_set_header X-Real-IP $remote_addr; }
```

Директива «proxy_http_version 1.1;» задаёт версию протокола HTTP, по которому будет передан запрос. По умолчанию используется версия 1.0. Полученное в заголовках запроса, поле Host запишется в переменную $host, которая будет передана на следующий слой для обработки, как и другие значения. Чтение данной конфигурации nginx будет производить последовательно, записывая значения переменных в виртуальное адресное пространство [4].

Чтобы дополнить модель, следует создать фильтр, который будет ограничивать доступ к ресурсам и сети для роботов, не несущих полезной нагрузки, с целью освобождения канала и вычислительных мощностей. Проверку можно производить по наличию поддержки cookies и javascript. Для этого будет введена переменная $cookie_test:

[4] Docker Documentation URL: https://docs.docker.com/

if ($cookie_test != "bonch") { set $test_cookie "${test_cookie}Y"; }

В данном условии добавлена переменная $test_cookie принимающая значения либо Y, либо N. Из условия видно, что применяется значение Y, если проверка на тестовую cookies не выполнена, т. е. $cookie_test != "bonch", и значение N, если проверка пройдена.

Если $test_cookie = Y, то проверка запроса еще не производилась и требуется передать в запрос тестовую cookies, для проверки поддержки указанных методов. Проверка будет осуществляться через скрипт, на который потребуется перенаправить запрос:

if ($test_cookie = Y) { rewrite ^(.*)$ /cookiestest.html last; break; }

Директива rewrite, в ответ на запрос, будет возвращать код скрипта cookiestest.html, который заранее будет создан в каталоге с конфигурационным файлом nginx отдельным файлом. После отдачи данного файла обработка на стороне сервера завершается, и со стороны балансировщика дополнительных мер не требуется.

Исходные данные для скрипта: формат html и возможность проверить поддержку cookies клиентом javascript.

1. Так как скрипт – это html-файл, то начнем с открытия и закрытия тегов html внутри которых и будет расположен код [5]:

<html><head><script></script></head><body></body></html>

2. Внутри тегов требуется разместить javascript передающий coockies. Для этого воспользуемся функцией function set_cookie() [6]:

<script> function set_cookie() {
var now = new Date();
var time = now.getTime();
*time += 19360000 * 1000;*
now.setTime(time);
document.cookie='test=bonch'+'; expires='+now.toGMTString()+'; path=/';}
</script>

В функцию в качестве входных параметров передается текущая дата и время сервера, а также новый параметр времени, который определяет время жизни cookies. После этого в параметр document.cookie передаем значение cookies, которое в последствии будет проверяться сервером, а также временные параметры [6].

Значение функции function set_cookie() требуется передать клиенту, а также дополнить параметрами, которые говорят клиенту о том, что требуется повторить запрос с уже установленной cookies. Сделать это можно внутри тегов <script> и </script> посредством следующих функций:

set_cookie();location.reload();

Здесь set_cookie() передает ранее определенные значения, а location.reload() обязывает клиента повторить запрос. Итоговый алгоритм работы системы, представлен на рис. 3.

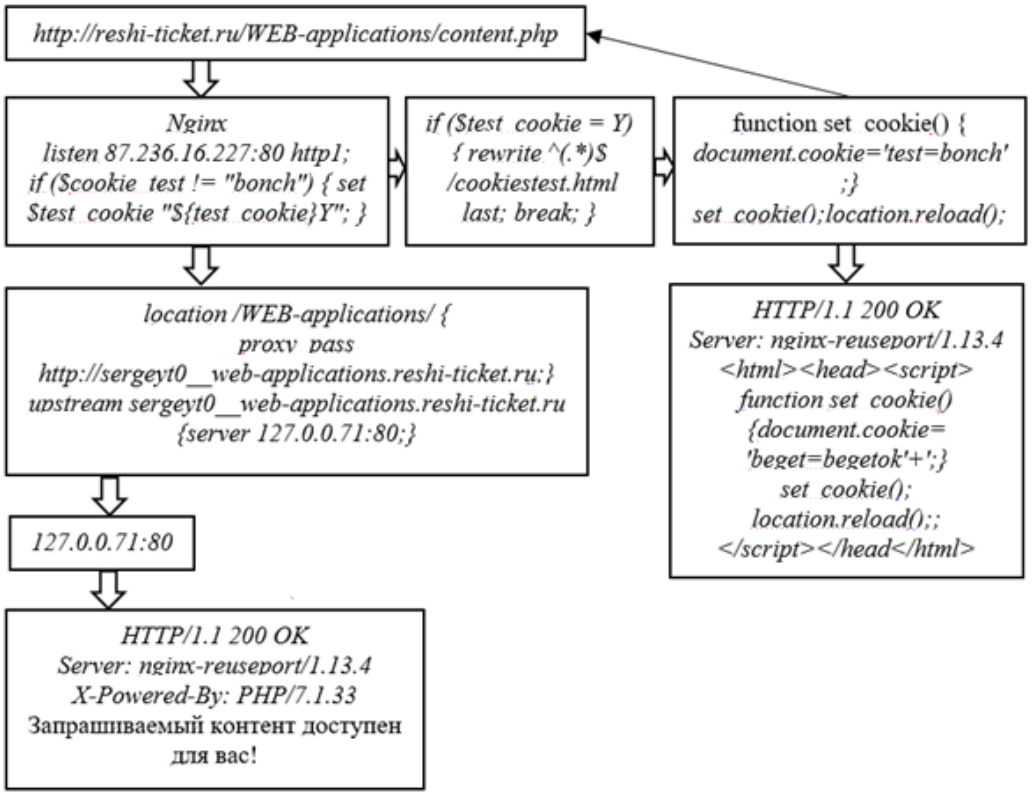

Рис. 3. Итоговый алгоритм работы системы, выполняющей часть функций ядра сети 5G

В заключение следует отметить, что построенная система представляет собой облачную серверную архитектуру, выполняющую часть функций ядра сотовой связи 5G. Используя данную модель как фундамент, в дальнейшем предполагается разработать полную архитектуру, выполняющую все функции ядра.

Литература

1. Степутин А. Н., Николаев А. Д. Мобильная связь на пути к 6G. – М., Вологда: ООО «Издательство «Инфра-Инженерия», 2017. 380 с. ISBN: 978-5-9729-0182-1.

2. Thomas D. Nadeau and Ken Gray. SDN: Software Defined Networks. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. URL: https://www.academia.edu/5481877/SDN_Software_Defined_Networks

3. Rajendra Chayapathi, Syed Farrukh Hassan, Paresh Shah. Network Functions Virtualization (NFV) with a Touch of SDN. – Boston, Columbus, Indianapolis, New York, San Francisco, Amsterdam, Cape Town, Dubai, London, Madrid, Milan, Munich, Paris, Montreal, Toronto, Delhi, Mexico City, São Paulo, Sidney, Hong Kong, Seoul, Singapore, Taipei, Tokyo: Addison-Wesley. First printing: November 2016. ISBN-13: 978-0-13-446305-6. ISBN-10: 0-13-446305-6. URL: http://ptgmedia.pearsoncmg.com/images/9780134463056/samplepages/9780134463056.pdf

4. Керриск М. Linux API. Исчерпывающее руководство. – СПб.: Питер, 2018. 1248 с.: ил. ISBN 978-5-496-02689-5

5. Муссиано Ч., Кеннеди Б. HTML и XHTML Подробное руководство. – СПб.: Символ-Плюс, 2008. 752 с. ISBN-10: 5-93286-104-5

6. Хавербеке М. Выразительный JavaScript. Современное веб-программирование: пер. с англ. Е. Сандицкой. – СПб.: Питер, 2019. 480 с.

References

1. Steputin A. N., Nikolayev A. D. Mobilnaya svyaz na puti k 6G. – M., Vologda: OOO «Izdatelstvo «Infra-Inzheneriya». 2017. 380 s. ISBN: 978-5-9729-0182-1.

2. Thomas D. Nadeau and Ken Gray. SDN: Software Defined Networks. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. URL: https://www.academia.edu/5481877/SDN_Software_Defined_Networks

3. Rajendra Chayapathi, Syed Farrukh Hassan, Paresh Shah. Network Functions Virtualization (NFV) with a Touch of SDN. – Boston, Columbus, Indianapolis, New York, San Francisco, Amsterdam, Cape Town, Dubai, London, Madrid, Milan, Munich, Paris, Montreal, Toronto, Delhi, Mexico City, São Paulo, Sidney, Hong Kong, Seoul, Singapore, Taipei, Tokyo: Addison-Wesley. First printing: November 2016. ISBN-13: 978-0-13-446305-6. ISBN-10: 0-13-446305-6. URL: http://ptgmedia.pearsoncmg.com/images/9780134463056/samplepages/9780134463056.pdf

4. Kerrisk M. Linux API. Ischerpyvayushcheye rukovodstvo. – SPb.: Piter. 2018. 1248 s.: il. ISBN 978-5-496-02689-5

5. Mussiano Ch., Kennedi B. HTML i XHTML Podrobnoye rukovodstvo. – SPb.: Simvol-Plyus. 2008. 752 s. ISBN-10: 5-93286-104-5

6. Khaverbeke M. Vyrazitelnyy JavaScript. Sovremennoye veb-programmirovaniye: per. s angl. E. San-ditskoy. – SPb.: Piter. 2019. 480 s.

Полевич Сергей Сергеевич – студент, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, sergeypolevich@mail.ru

Симонина Ольга Александровна – кандидат технических наук, доцент, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, sergf7@mail.ru

Polevich Sergey – Student, SUT, St. Petersburg, 193232, Russian Federation, sergeypolevich@mail.ru

Simonina Olga – Candidate of Engineering Sciences, Associate Professor, SUT, St. Petersburg, 193232, Russian Federation, sergf7@mail.ru