Оригинальная статья / Original article

https://doi.org/10.21869/2223-1560-2022-26-3-98-111

Балансировка нагрузки данных в распределенной сети через прокси-сервер Nginx

К. И. Никишин [1] ✉

[1] Пензенский государственный университет
ул. Красная, д. 40, г. Пенза 440026, Российская Федерация

✉ e-mail: nkipnz@mail.ru

Резюме

Цель исследования: *исследование программного способа балансировки данных в распределенной сети через прокси-сервер Nginx.*

Методы. *Балансировка нагрузки данных в компьютерной сети является важным параметром сети. Из-за балансировки нагрузки в сети может уменьшаться или увеличиваться задержка передачи, разброс от среднего значения джиттера. Таким образом, балансировка нагрузки в сети влияет на временные характеристики и пропускную способность сети. Управление и оптимизацию балансировки нагрузки можно осуществлять как программным, так и аппаратным способами. В статье уделяется внимание балансировке нагрузки данных на прикладном уровне приложений. Кратко рассмотрена аппаратная балансировка нагрузки, которая решается в рамках непосредственно сетевого оборудования, к примеру, в коммутаторах. Это решается диспетчером управления очередями в коммутаторе Ethernet, который управляет полосой пропускания и очередями. Описаны циклические алгоритмы, а также алгоритм с временной селекцией кадров в диспетчерах коммутатора, которые реализуют эффективную аппаратную балансировку нагрузки. Рассмот-рена программная балансировка нагрузки данных в сети. В качестве программной балансировки нагрузки использовался веб-сервер и обратный прокси-сервер Nginx, на сервере было запущено 3 Docker контейнера, сделанного на основе Asp.net приложения, запущенных на разных окружениях.*

Результаты. *Произведена настройка, конфигурация сети и использовался циклический алгоритм балансировки нагрузки RR в сервере Nginx. Было проведено исследование сети с разным количеством окружений в сети, веб-серверов, запросов данных. Циклический алгоритм балансировки нагрузки в Nginx является более эффективным по сравнению со случайным алгоритмом, это было показано в ходе экспериментов.*

Заключение. *Были рассмотрены и исследованы аппаратные и программные алгоритмы балансировки нагрузки в распределенной сети. Циклические алгоритмы балансировки данных позволили повысить пропускную способность сети, ее эффективность и быстродействие.*

Ключевые слова*: балансировка нагрузки; распределенные сети; Ethernet; задержка передачи; циклические алгоритмы балансировки; прокси-сервер Nginx; Docker; сети Петри; CPN Tools.*

Конфликт интересов: *Автор декларирует отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Для цитирования: Никишин К. И. Балансировка нагрузки данных в распределенной сети через прокси-сервер Nginx // Известия Юго-Западного государственного университета. 2022; 26(3): 98-111. https://doi.org/10.21869/2223-1560-2022-26-3-98-111.

Поступила в редакцию 22.06.2022 *Подписана в печать 20.07.2022* *Опубликована 30.09.2022*

Load Balancer of Data in a Distributed Network via Nginx Proxy Server

Kirill I. Nikishin [1] ✉

[1] Penza State University
40 Krasnaya str., Penza 440026, Russian Federation

✉ e-mail: nkipnz@mail.ru

Abstract

Purpose of research. *Is investigation of a software method for balancing data in a distributed network via an Nginx proxy server.*

Methods. *In a computer network load balancer of data is an important network parameter. Due to load balancer in the network, the transmission delay may decrease or increase, the spread from the average jitter value. Thus, load balancer in the network affects the time characteristics and network bandwidth. Load balancer can be managed and optimized in both software and hardware ways. The article focuses on load balancer of data at the application level of applications. Hardware load balancer, which is solved within the framework of network equipment itself, for example, in switches, is briefly considered. This is handled by the queue manager in the Ethernet switch, which manages the bandwidth and queues. Cyclic algorithms are described, as well as an algorithm with time selection of frames in dispatcher of switch that implement effective hardware load balancer. Software load balancer of data in the network is considered. A web server and an Nginx reverse proxy server were used as software load balancer, 3 Docker containers based on Asp.net applications running on different environments.*

Results. *The network was configured and the cyclic load balancer algorithm was used in the Nginx server. A research of a network with a different number of environments in the network, web servers, data requests was conducted. The cyclic load balancer of data in Nginx is more efficient than the random algorithm, this has been shown during experiments.*

Conclusion. *Hardware and software load balancer algorithms in a distributed network were considered and investigated. Cyclic load balancer of data has made it possible to increase the network bandwidth, its efficiency and performance.*

Keywords*: load balancer; distributed networks; Ethernet; transmission delay; cyclic balancing algorithms; Nginx proxy server; Docker; Petri Nets; CPN Tools.*

Conflict of interest. *The author declare the absence of obvious and potential conflicts of interest related to the publication of this article.*

For citation: Nikishin K.I. Load Balancer of Data in a Distributed Network via Nginx Proxy Server. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2022; 26(3): 98-111 (In Russ.). https://doi.org/ 10.21869/2223-1560-2022-26-3-98-111.

Received 22.06.2022 *Accepted 20.07.2022* *Published 30.09.2022*

***

Введение

В настоящее время распределенные сети нашли широкое применение и применяются различных сферах деятельности. Распределенные сети находят применение в промышленности, так и у конечного пользователя. Основным достоинством распределенных сетей яв-

ляется отказоустойчивость и создание резервных копий данных, системы в случае ошибок.

Распределенные сети являются новой парадигмой в случае с классическими компьютерными сетями, основанными на Ethernet [1] и понятии «качества обслуживания» (Quality of Service QoS) согласно формату IEEE 802.1 [2-3]. К распределенным компьютерным сетям можно отнести программно-конфигурируемые сети (ПКС) [4-8], облачное хранилище и сеть [9], технологию Time-Triggered Ethernet [10-12].

Критически важным параметром в компьютерных сетях является задержка передачи данных, разброс на выходе коммутатора среднего значения задержки (джиттер). Согласно стандарту семиуровневой модели OSI верхним уровнем является прикладной, передача данных в сети Ethernet – на сетевом и каналом уровне.

Многие исследования в области уменьшения и прогнозирования задержек данных в компьютерной сети основываются в первую очередь на этих уровнях модели OSI. Одним из вариантов уменьшения задержки является балансировка нагрузки данных, трафика.

В статье уделяется внимание балансировке нагрузки данных на прикладном уровне, поскольку прикладной уровень играет существенную роль в конфигурировании и управлении распределенной сети.

К примеру, в ПКС выделяют три уровня управления сетью, и важным уровнем является уровень приложений [13], поскольку администратор сети распределяет оптимальным образом маршрут данных, принятие решения в случае исключительных, ошибочных ситуаций, когда контроллер ПКС не может самостоятельно принять данное решение. Частым способом обмена информацией в распределённой сети является JSON.

Материалы и методы

Балансировка нагрузки данных в компьютерной сети является важным параметром сети. Управление и оптимизацию балансировки нагрузки можно осуществлять как программным, так и аппаратным способами. На рис. 1 представлена обобщенная передача данных по сети Интернет от клиента к серверу и с учетом балансировки нагрузки.

Аппаратная балансировка нагрузки может решаться в рамках сетевого и телекоммуникационного оборудования (коммутаторы, маршрутизаторы). В коммутаторах основным элементом управления является диспетчер очередей коммутатора.

Балансировка нагрузки в коммутаторе решается с помощью управления полосой пропускания и управления очередями. Как раз таковым является диспетчер управления очередями в коммутаторе Ethernet.

Рис. 1. Схема передача данных с учетом балансировки нагрузки в сети

Fig. 1. The scheme of data transmission using load balancer in the network

Исследования в этой области основываются на создании и разработке эффективных алгоритмов диспетчеризации с учетом многокритериальных параметров управления диспетчерами коммутатора. К наиболее эффективным алгоритмам относятся различные циклические алгоритмы (Round Robin RR), взвешенные справедливые (Weighted Fair Queuing WFQ) [14-15].

Циклические алгоритмы делятся на взвешенный циклический алгоритм (Weighted Round Robin WRR) и дефицитный циклический алгоритм (Deficit Round Robin DRR). К новому эффективному алгоритму относится алгоритм с временной селекцией кадров (Time Selection Service TSS), в статьях [16-17] описано подробное описание алгоритма, моделирование и показтели его эффективности при управлении очередями. Основное отличие учитываются время ожидания кадров в очередях и ограничение времени обслуживания очередей, что в свою очередь приводит к уменьшению разброса среднего значения джитера в сети.

Эффективным средством исследования компьютерных сетей является математический аппарат сетей Петри в пакете CPN Tools [18-19]. В данном пакете можно исследовать не только функционирование компьютерной сети, а также временные задержки, джиттер, вести и учитывать статистику, создавать кортежи с помощью различных структур и цветов, декомпозировать модель на более мелкие модели, программировать на языке CPN ML, верифицировать модели на тупиковые переходы, вычислять пространства состояний [20-21]. Сеть Петри, моделирующая алгоритм с учетом времени ожидания кадров (диспетчер очередей TSS), представлена на рис. 2.

Перейдем к рассмотрению программной балансировки нагрузки данных в сети. В качестве программной балансировки нагрузки (load balancer) используется веб сервер и обратный прокси-сервер Nginx, на сервере запущено 3 Docker контейнера, сделанного на основе Asp.net приложения, запущенных на разных окружениях.

Чтобы веб-сервер смог распределять нагрузку необходимо создать его конфигурацию и находится */nginx/conf/nginx.conf. Создается группа веб-серверов, для того чтобы по ним распределять запросы сервером. Для тега upstreams задаются имя servers и добавляются адреса веб-серверов.

```
upstream servers{
        server localhost:8001;
        server localhost:8002;
        server localhost:8003;
    }
```

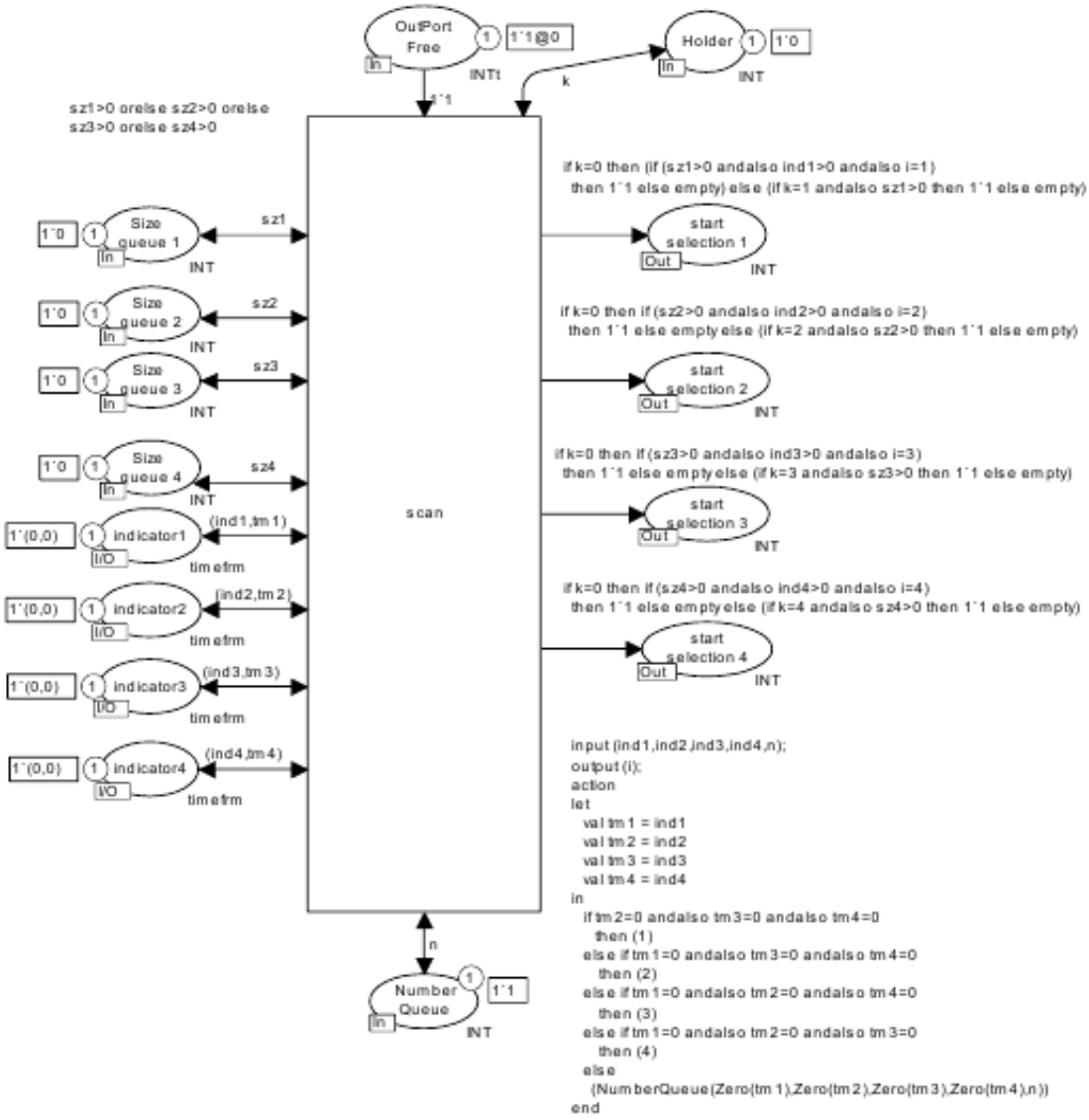

Рис. 2. Сеть Петри диспетчера очередей коммутатора по алгоритму TSS

Fig. 2. The Petri network of dispatcher of switch using the TSS algorithm

Затем необходимо в настройках виртуального домена создать прокси-запросы на созданный ранее upstream. Настройка выглядит следующим образом:

```
server{
        listen 8888;
        location / {
         proxy_pass http://servers;
         proxy_set_header Host   $host;
           }
     }
```

Параметр proxy_pass указывает, что все запросы должны переводиться на upstream servers, а он, в свою очередь, будет перенаправлять запросы на указанные сервера. Параметр proxy_set_header продублирует заголовок Host пришедший на веб-сервер в Docker. При настройке серверов можно указывать опции: наличие резервного сервера backup, временная приостановка запросов down, распределение весов и их учете при прохождении прокси-запроса weight.

```
upstream servers{
        server localhost:8001 weight
= 100 max fails=2 fail_timeout =90s;
        server localhost:8002 weight
= 200 max_conns=1000;
        server localhost:8003 backup;
        server localhost:8003 down;
     }
```

Nginx считает сервер недоступным после одной неудачной попытки отправки на него запроса и в течение 10 секунд не работает с ним. Балансировка нагрузки регулируется с помощью следующих параметров: количество неудачных попыток, после которых сервер считается недоступным, max_fails и время недоступности сервера fail_timeout.

В сервере Nginx могут использоваться следующие алгоритмы балансировки нагрузки: циклический (RR), на основе хэш-функций, случайный. В статье рассматривается взвешенный циклический алгоритм, таким образом, с учетом распределения весов обслуживаются и управляются очереди. Файл конфигурации веб-сервера представлен ниже:

```
events {{
http{
  upstream servers{

   server localhost:8001 weight = 100
max fails=2 fail_timeout =90s;
     server localhost:8002 weight =
200 max_conns=1000;
     server localhost:8003 backup;
     server localhost:8003 down;
    }
    server{
      listen 8888;
      location / {
       proxy_pass http://servers;
       proxy_set_header Host $host;
            }
       }
  }
```

В качестве утилиты для отправки HTTP запросов используется Postman. В качестве серверов используется WEB API решение на фреймворке Asp.net. Шесть серверов запущены на условно разных окружениях (Environment_1, Environment_2, Environment_3). Для исследования и распределения балансировки нагрузки было разработано приложение, предназначенное для анализа получения погоды на ближайшие 5 дней.

На запрос /Api/GetWeather методом GET приходит ответ из ResponseModel: enviroment – имя окружения, на котором запущен сервис типа string, lead_time - время длительности получения погоды, weather – перечисление дней и погоды на каждый день типа коллекции элементов типа WeatherForecast: date – дата типа datatime, temperatureC – средняя температура за день в градусах Цельсия, temperatureF – средняя температура за день в градусах Фаренгейта, summary - общая характеристика погоды на день.

Для развертывания приложения используется Docker. Команда создания образа приложения - docker build --pull -t apiweater:1.0.

Команды создания контейнеров на соответствующих портах и разных переменных окружения ASPNETCORE_ENVIRONMENT:

```
docker run --name Environment1 --env "ASPNETCORE_ENVIRONMENT=Environment_1" -it -p 8001:80 apiweater:1.0
docker run --name Environment2 --env "ASPNETCORE_ENVIRONMENT=Environment_2" -it -p 8002:80 apiweater:1.0
docker run --name Environment3 --env "ASPNETCORE_ENVIRONMENT=Environment_3" -it -p 8003:80 apiweater:1.0
```

Для запуска серверов также можно использовать стандартную утилиту Windows – PowerShell. Необходимо перейти в директорию с приложением и выполним команду: dotnet run --launch-profile "имя окружения для запуска". Запуск окружения Environment_1 представлен на рис. 3.

Запуск веб-сервера с созданной конфигурацией представлен на рис. 4.

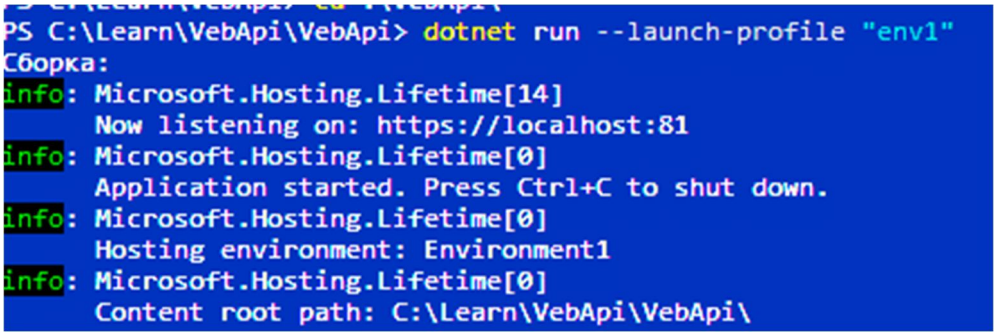

Рис. 3. Запуск окружения Environment_1

Fig. 3. Run of Environment_1

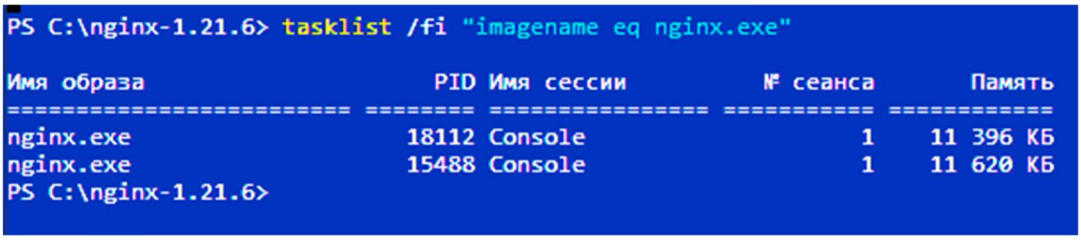

Рис. 4. Запуск веб-сервера

Fig. 4. Run of web server

Таким образом, запущено три идентичных приложения на различных окружениях.

Результаты и их обсуждение

После запуска веб-сервера на порту 8888 и приложений на портах 8001, 8002, 8003 отправляются 10 запросов по адресу: http://localhost:8888/Api/GetWeather. Каждый запрос будет проходить через прокси на разные запущенные сервера согласно циклическому алгоритму RR с разным временем обработки запроса на сервере.

Формат ответа на запрос приходит в JSON формате и вычисляется время обработки на сервере в зависимости от окружения, доступности самого веб-сервера. Например, ответ на запрос для окружения 2 представлен ниже и на рис. 5:


```
{  "environment": "Environment_2",
   "lead_time": 4.2125794,
   "weather": [
      {
         "date": "2022-06-
12T00:17:10.6409921+00:00",
         "temperatureC": 54,
         "temperatureF": 129,
         "summary": "Scorching"
      },
      {
         "date": "2022-06-
14T00:17:10.6409986+00:00",
         "temperatureC": -15,
         "temperatureF": 6,
         "summary": "Hot"
      },
      {
         "date": "2022-06-
16T00:17:10.640999+00:00",
         "temperatureC": 42,
         "temperatureF": 107,
         "summary": "Warm"
      }
   ]
}
```


Рис. 5. Ответ на один из запросов для окружения 2

Fig. 5. Response to one of the requests for environment 2

Для тестирования производилось выключение веб-серверов для проверки работоспособности сети и корректной отправки запросов согласно циклическому алгоритму RR между рабочими веб-серверами и для необходимости запасного веб-сервера, указанного в конфигурации сети. На рис. 6 представлено одновременное включение веб-серверов 1-2, вычисление балансировки нагрузки с учетом поступивших к ним запросов.

Циклический алгоритм RR отправляет запросы по очереди на каждый доступный веб-сервер. При поступлении запросов 4-6 был недоступен веб-сервер 1, при поступлении запросов 7-8 были

недоступны веб-сервера 2-3. Таким образом, Nginx способен прокси-запросы на разные сервера, распределяя нагрузку между всеми доступными в данный момент серверами.

Был проведен сравнительный анализ балансировки нагрузки данных на распределенную сеть с учетом различных алгоритмов обработки. На рис. 7 представлена диаграмма обработки запросов, можно увидеть, что циклический алгоритм балансировки нагрузки обладает большей эффективностью и уменьшению среднего времени обработки запроса на стороне сервера, повышается быстродействие в сети в среднем на 25-28% в отличие от алгоритма случайного характера в балансировке нагрузки.

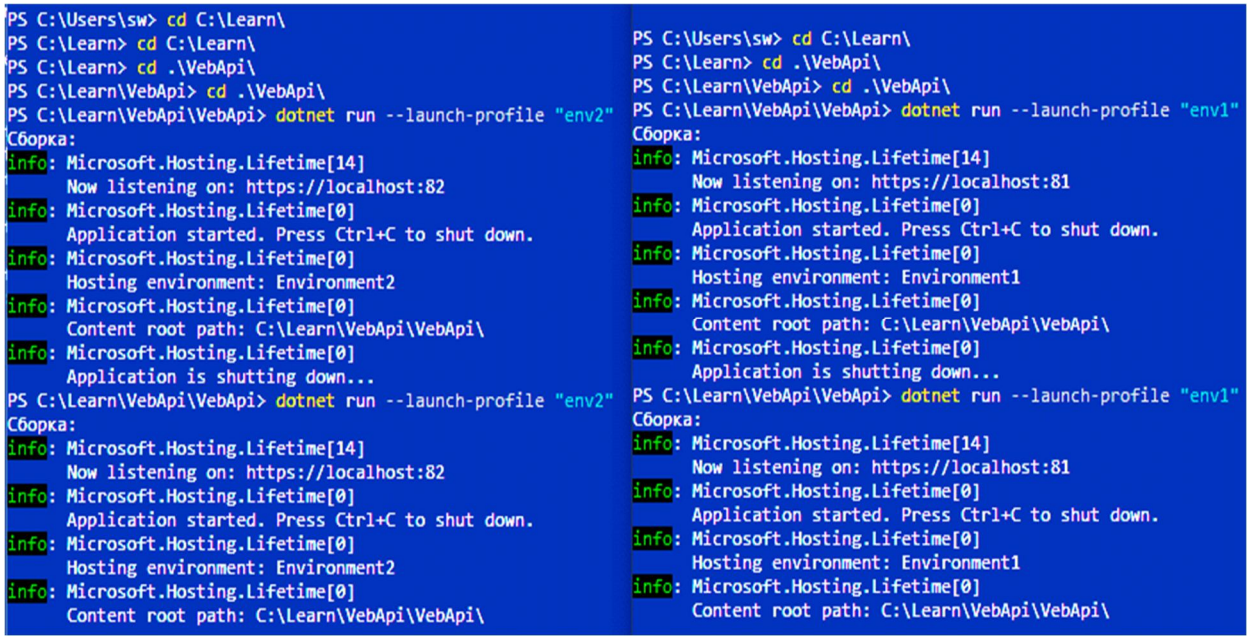

Рис. 6. Одновременный запуск веб-серверов 1-2

Fig. 6. Simultaneous launch of web servers 1-2

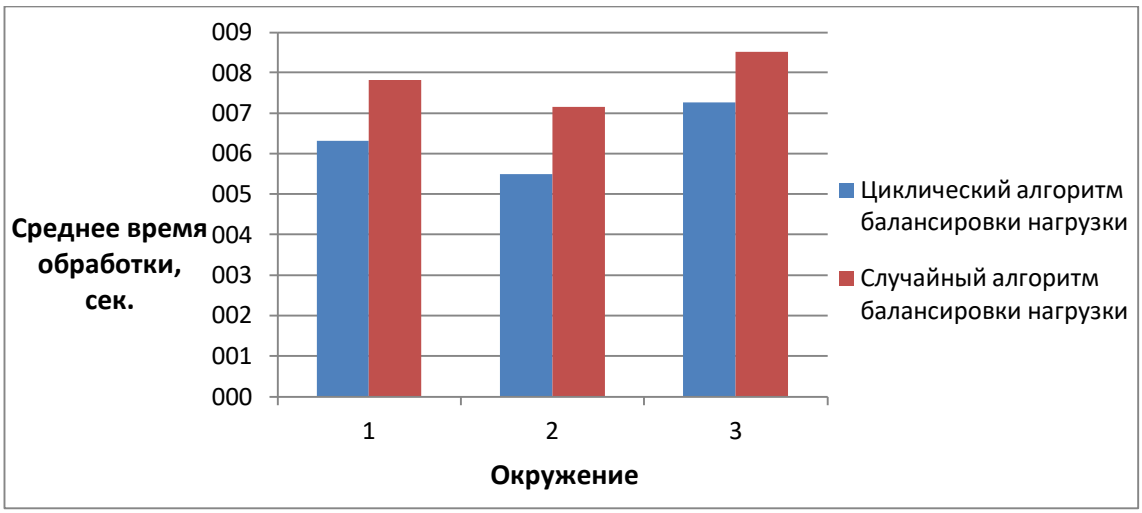

Рис. 7. Результаты экспериментов балансировки нагрузки в сети

Fig. 7. Results of network load balancing experiments

Кроме этого алгоритм, основанный на случайном характере, может приводить к потере запросов. Такая ситуация может возникнуть в случае, если сеть будет случайно переключать запросы на неработающие веб-серверы, в таком случае конечный пользователь не сможет получить необходимые данные. Это существенный недостаток в отличие от циклической балансировки нагрузки.

Выводы

Балансировка нагрузки данных в компьютерной сети является важным параметром сети. Из-за балансировки нагрузки в сети может уменьшаться или увеличиваться задержка передачи, разброс от среднего значения джиттера. Таким образом, балансировка нагрузки в сети влияет на временные характеристики и пропускную способность сети.

Управление и оптимизацию балансировки нагрузки можно осуществлять как программным, так и аппаратным способами. В статье уделяется внимание балансировки нагрузки данных на прикладном уровне приложений.

Кратко рассмотрена аппаратная балансировка нагрузки, которая решается в рамках непосредственно сетевого оборудования, к примеру, в коммутаторах. Это решается диспетчером управления очередями в коммутаторе Ethernet, который управляет полосой пропускания и очередями. Описаны циклические алгоритмы WRR и DRR, а также алгоритм с временной селекцией кадров TSS в диспетчерах коммутатора, которые реализуют эффективную аппаратную балансировку нагрузки.

Рассмотрена программная балансировка нагрузки данных в сети. В качестве программной балансировки нагрузки использовался веб-сервер и обратный прокси-сервер Nginx, на сервере было запущено 3 Docker контейнера, сделанного на основе Asp.net приложения, запущенных на разных окружениях. Произведена настройка, конфигурация сети и использовался циклический алгоритм балансировки нагрузки RR в сервере Nginx.

Было проведено исследование сети с разным количеством окружений в сети, веб-серверов, запросов данных. Циклический алгоритм балансировки нагрузки в Nginx является более эффективным по сравнению со случайным алгоритмом, это было показано в ходе экспериментов.

Таким образом, аппаратные и программные алгоритмы балансировки нагрузки в распределенной сети повышают пропускную способность сети, ее эффективность и быстродействие.

Список литературы

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. 4-е изд. СПб.: Питер, 2010. 943 с.

2. Описание стандарта IEEE 802.1q. URL: https://ru.wikipedia.org/wiki/IEEE_802.1Q (дата обращения 18.06.2022).

3. Никишин К. И. Механизм управления трафиком реального времени в коммутаторе Ethernet // Вестник компьютерных и информационных технологий. 2015. № 10. С. 32–37.

4. McKeown N., Anderson T., Balakrishnan H. et al. Openflow: enabling innovation in campus networks // ACM SIGCOMM Computer Communication Review, 2008, vol. 38, no. 2, pp. 69–74.

5. Kobayashi M., Seetharaman S., Parulkar G., Appenzeller G., Little J., Van Reijendam J., McKeown N. Maturing of OpenFlow and Software-Defined Networking Through Deployments // Computer Networks. 2014. Vol. 61. P. 151–175.

6. Корячко В. П., Перепелкин Д. А. Программно-конфигурируемые сети. М.: Горячая линия – Телеком, 2020. 288 с.

7. Shalimov A. et al. Advanced study of SDN/OpenFlow controllers // Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia. ACM, 2013.

8. Перепелкин Д. А., Бышов В. С. Балансировка потоков данных в программно-конфигурируемых сетях с обеспечением качества обслуживания сетевых сервисов // Радиотехника. 2016. № 11. С. 111–119.

9. Никульчев Е. В., Паяин С. В., Плужник Е. В. Динамическое управление трафиком программно-конфигурируемых сетей в облачной инфраструктуре // Вестник Рязанского государственного радиотехнического университета. 2013. № 3 (45). С. 54–57.

10. Kopetz H. Real-time systems design principles for distributed embedded applications. New York: Springer, 2011. 396 p.

11. The Time-Triggered Ethernet (TTE) design / H. Kopetz, A. Ademaj, P. Grillinger, K. Steinhammer // International Symposium on Object-oriented Real-time Distributed Computing. 2005. P. 22–33.

12. Никишин К. И., Коннов Н. Н., Пащенко Д. В. Моделирование систем на базе технологии Time-Triggered Ethernet // Информационные технологии и математическое моделирование: материалы XV Междунар. конф. имени А. Ф. Терпугова. Томск: Изд-во Томск. ун-та, 2016. Ч. 2. С. 117–122.

13. Перепелкин Д. А. Концептуальный подход динамического формирования трафика программно-конфигурируемых телекоммуникационных сетей с балансировкой нагрузки // Информационные технологии. 2015. Т. 21. № 8. С. 602–610.

14. Механов В.Б., Кизилов, Е.А. Моделирование цветными сетями Петри обслуживания очередей алгоритмом WRR // Труды IX Международной научно-технической конференции «Новые информационные технологии и системы». Пенза: Изд-во ПГУ, 2010. Ч. 1. С. 67 – 73.

15. Артемов И. В., Коннов М. Н., Никишин К. И. Анализ эффективности адаптивного алгоритма формирования виртуального таймслота в сетевом коммутаторе // Тру-

ды Международного симпозиума «Надежность и качество». Пенза: Изд-во ПГУ, 2020. Т.2. С. 298–302.

16. Scheduling queues in the Ethernet switch, considering the waiting time of frames / E. Kizilov, N. Konnov, K. Nikishin, D. Pashchenko, D. Trokoz // MATEC Web of Conferences. 2016. Vol. 44. P. 01011-p.1–01011-p. 5.

17. Учет времени поступления кадров для управления очередями в коммутаторе / Е. А. Кизилов, Н. Н. Коннов, В. Б. Механов, К. И. Никишин // Телематика-2014: тр. XXI Всерос. науч.-метод. конф. СПб. : СПбГУ ИТМО, 2014. С. 134–136.

18. Jensen K. An Introduction to the Practical Use of Coloured Petri Nets // Lectures on Petri Nets II: Applications. Berlin: Springer, 1998. P. 237-292.

19. Jensen K., Kristensen L.M. Coloured Petri Nets. Modelling and Validation of Concurrent Systems. Berlin: Springer, 2009. 384 p.

20. Nikishin K., Konnov N. Schedule Time-Triggered Ethernet // International Conference on Engineering Management of Communication and Technology, EMCTECH 2020. DOI: 10.1109/EMCTECH49634.2020.9261540.

21. Никишин К. И., Коннов Н. Н. Генератор трафика Ethernet на основе цветных сетей Петри // Модели, системы, сети в экономике, технике, природе и обществе. 2016. № 1 (17). С. 299–307.

References

1. Olifer V. G., Olifer N. A. *Komp'yuternye seti. Printsipy, tekhnologii, protokoly* [Computer networks. Principles, technologies, protocols]. St. Petersburg, 2010. 943 p.

2. Opisanie standarta IEEE 802.1q [Description of the IEEE 802.1q standard]. Available at: https://ru.wikipedia.org/wiki/IEEE_802.1Q (accessed 01.06.2022).

3. Nikishin K. I. Mekhanizm upravleniya trafikom real'nogo vremeni v kommutatore Ethernet [The mechanism of management real-time traffic in the switch Ethernet]. *Vestnik komp'yuternykh i informatsionnykh tekhnologii = Herald of Computer and Information Technologies*, 2015, no. 10, pp. 32–37.

4. McKeown N., Anderson T., Balakrishnan H. et al. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 2008, vol. 38, no. 2, pp. 69–74.

5. Kobayashi M., Seetharaman S., Parulkar G., Appenzeller G., Little J., Van Reijendam J., McKeown N. Maturing of OpenFlow and Software-Defined Networking Through Deployments. *Computer Networks*, 2014, vol. 61, pp. 151–175.

6. Koryachko V. P., Perepelkin D. A. *Programmno-konfiguriruemye seti* [Software defined networks]. Moscow, Goryachaya liniya – Telekom Publ., 2020, 288 p.

7. Shalimov A. et al. Advanced study of SDN/OpenFlow controllers. *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*. ACM, 2013.

8. Perepelkin D. A., Byshov V. S. Balansirovka potokov dannykh v programmno-konfiguriruemykh setyakh s obespecheniem kachestva obsluzhivaniya setevykh servisov [Balancing data flows in software defined networks with ensuring the quality of service of network services]. *Radiotekhnika = Radio Engineering*, 2016, no. 11, pp. 111-119.

9. Nikulchev E. V., Payin S. V., Pluzhnik E. V. Dinamicheskoe upravlenie trafikom programmno-konfiguriruemykh setei v oblachnoi infrastrukture [Dynamic traffic management of software defined networks in cloud infrastructure]. *Vestnik Ryazanskogo gosudarstvennogo radiotekhnicheskogo universiteta = Vestnik of Ryazan State Radio Engineering University*, 2013, no. 3 (45), pp. 54-57.

10. Kopetz H. Real-time systems design principles for distributed embedded applications. New York: Springer, 2011, 396 p.

11. Kopetz H., Ademaj A., Grillinger P., Steinhammer K. The Time-Triggered Ethernet (TTE) design. *International Symposium on Object-oriented Real-time Distributed Computing*, 2005, pp. 22–33.

12. Nikishin K. I., Konov N. N., Pashchenko D. V. [Modelling of systems using Time-Triggered Ethernet]. *Informatsionnye tekhnologii i matematicheskoe modelirovanie: materialy XV Mezhdunar. konf. imeni A. F. Terpugova* [Information technologies and mathematical modeling. Materials of the XV International Conf. named after A. F. Terpugov]. Tomsk, Tomsk University Publ., 2016, pt 2, pp. 117-122 (In Russ.).

13. Perepelkin D. A. Kontseptual'nyi podkhod dinamicheskogo formirovaniya trafika programmno-konfiguriruemykh telekommunikatsionnykh setei s balansirovkoi nagruzki [Conceptual approach of dynamic traffic generation of software defined telecommunication networks with load balancing]. *Informatsionnye tekhnologii = Information Technologies*, 2015, vol. 21, no. 8, pp. 602-610.

14. Mekhanov V. B., Kizilov E. A. [Modeling of queue dispatcher by the WRR algorithm with colored Petri nets]. *Trudy IX Mezhdunarodnoi nauchno-tekhnicheskoi konferentsii "Novye informatsionnye tekhnologii i sistemy"* [Proceedings of the IX International Scientific and Technical Conference "New Information technologies and Systems"], Penza, 2010, pp. 67-73 (In Russ.)

15. Artemov I. V., Konnov M. N., Nikishin K. I. [Analysis of the effectiveness of an adaptive algorithm for forming a virtual timeslot in a network switch]. *Trudy Mezhdunarodnogo simpoziuma «Nadezhnost' i kachestvo»* [Proceedings of the International Symposium "Reliability and quality"]. Penza, 2020, vol.2, pp. 298-302 (In Russ.).

16. Kizilov E., Konnov N., Nikishin K., Pashchenko D., Trokoz D. Scheduling queues in the Ethernet switch, considering the waiting time of frames. *MATEC Web of Conferences*, 2016, vol. 44, pp. 01011-p.1–01011-p. 5.

17. Kizilov E. A., Konnov N. N., Mekhanov V. B., Nikishin K. I. [Accounting of the delivery time for control of queue dispatcher in the switch]. *Telematika-2014*»: *tr. XKhI Vse-*

ros. nauch.-metod. konf. ["Telematics-2014"Proceedings of the XXI Scientific and Technical Conference]. St. Petersburg : St. Petersburg State University ITMO Publ., 2014, pp. 134-136. (In Russ.)

18. Jensen K. An Introduction to the Practical Use of Coloured Petri Nets. Lectures on Petri Nets II: Applications. Berlin: Springer, 1998, pp. 237-292.

19. Jensen K., Kristensen L.M. Coloured Petri Nets. Modelling and Validation of Concurrent Systems. Berlin: Springer, 2009, 384 p.

20. Nikishin K., Konnov N. Schedule Time-Triggered Ethernet. *International Conference on Engineering Management of Communication and Technology*, EMCTECH 2020. DOI: 10.1109/EMCTECH49634.2020.9261540.

21. Nikishin K. I., Konnov N. N. Generator trafika Ethernet na osnove tsvetnykh setei Petri [The traffic generator of switch Ethernet using colored Petri nets]. *Modeli, sistemy, seti v ekonomike, tekhnike, prirode i obshchestve = Models, Systems, Networks in Economics, Technology, Nature and Society*, 2016, no. 1 (17), pp. 299–307.

Информация об авторе / Information about the Author

Никишин Кирилл Игоревич, кандидат технических наук, старший преподаватель кафедры «Вычислительная техника», Пензенский государственный университет, г. Пенза, Российская Федерация, e-mail: nkipnz@mail.ru, ORCID: https://orcid.org/0000-0001-7966-7833

Kirill I. Nikishin Cand. of Sci. (Engineering), Senior Lecturer of Computer Engineering Departament, Penza State University, Penza, Russian Federation, e-mail: nkipnz@mail.ru, ORCID: https://orcid.org/0000-0001-7966-7833