ДИНАМИЧЕСКОЕ УПРАВЛЕНИЕ НАГРУЗКОЙ В ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЯХ

В.С. Елагин[1*]

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

Информация о статье
УДК 004.942
Язык статьи – русский

Ссылка для цитирования: Елагин В.С. Динамическое управление нагрузкой в программно-конфигурируемых сетях // Труды учебных заведений связи. 2017. Т. 3. № 3. С. 60–67.

Аннотация: *В связи с непрерывно усложняющейся топологией сетевой инфраструктуры ЦОД и предъявляемыми к ней требованиями производительности, надежности и отказоустойчивости трудность управления сетевой инфраструктурой ЦОД так же непрерывно растет. Вот почему возникает потребность в замене ручной конфигурации сетевой инфраструктуры ЦОД автоматизированными средствами. Одним из перспективных направлений модернизации существующей архитектуры сети является концепция SDN. Программно-определяемый ЦОД – это виртуализированная структура, направленная на предоставление услуги. Именно в рамках этой концепции необходимо развивать методы управления трафиком и балансировки внутри сетей ЦОД. В статье представлена математическая модель динамического алгоритма балансировки нагрузки в сетях ЦОД.*

Ключевые слова: *ЦОД, программно-конфигурируемые сети, SDN, дата-центр, динамическая балансировка нагрузки.*

Введение

В управлении сетевой инфраструктурой центров обработки данных (ЦОД) можно выделить ряд отдельных задач, которые должна решать система управления сетевой инфраструктурой. Поэтому для сравнения существующих средств управления сетевой инфраструктурой ЦОД был выбран набор функциональности, который покрывает все основные аспекты управления сетями ЦОД. Каждое рассматриваемое средство далее оценивается по наличию или отсутствию поддержки указанной функциональности из приведенного списка.

Перечень рассматриваемой функциональности, по которому оценивались средства управления сетевой инфраструктурой ЦОД, следующий:

- Автоматическое обнаружение сетевых устройств и динамическое построение карты сети (autodiscovery).
- Наличие автоматизации процессов конфигурации сетевых устройств и управления версиями программного обеспечения.
- Проверка соблюдения нормативных и регуляционных требований, обеспечение безопасности сетевой инфраструктуры.
- Автоматизированное управление потоками трафика в сети, его контроль и мониторинг в режиме реального времени.
- Сбор и анализ событий со всей сетевой инфраструктуры, включая физические и виртуальные устройства.
- Поддержка различных протоколов и технологий передачи данных, сетевого оборудования сторонних производителей, управления географически удаленными локациями, открытого API и возможность внедрения сторонних сервисов.

Из количественных характеристик производительности и эффективности для исследования были выбраны следующие:

- Характеристики, отражающие масштабируемость системы управления сетевой инфраструктурой для поддержки сетей разного размера и природы, а именно:
  - максимальный поддерживаемый размер сети (в виде количества узлов, входящих в сеть);
  - число различных поддерживаемых моделей сетевых устройств от различных производителей (коммутаторов, маршрутизаторов, межсетевых экранов и другого оборудования).

- Характеристики, отражающие возможности системы по реагированию на изменения в сети, а именно:
  - количество событий различной природы, анализируемых в единицу времени;
  - количество источников событий, с которых поддерживается одновременный сбор и анализ сообщений.
- Характеристики, отражающие управление трафиком в сети, а именно:
  - максимальное количество потоков трафика за единицу времени, для которых обеспечивается управление;
  - максимальное количество потоков трафика, для которых обеспечивается сбор и анализ статистики;
  - количество поддерживаемых виртуальных сегментов, устройств, туннелей различной природы, маршрутизаторов и т. п.

В связи с повышенной нагрузкой и неоднородностью трафика стала актуальной проблема эффективного использования серверных мощностей. Одним из аспектов данной проблемы является эффективное планирование и распределение нагрузки внутри распределенных вычислительных систем ЦОД с целью оптимизации использования ресурсов и сокращения времени вычисления. Эти и другие задачи решаются при помощи технологий виртуализации. В контексте виртуализации сетевых функций, в качестве примера возможных задач этого типа можно привести: адаптивное увеличение или уменьшение требуемых клиенту ресурсов, например, пропускной способности канала, перераспределение трафика между серверами, – то есть, реализацию балансировки нагрузки как услуги (LBaaS, от англ. Load-Balancing-as-a-Service).

Программно-определяемый ЦОД (SDDC, от англ. Software Defined Data Center) – это ЦОД, в котором инфраструктура виртуализирована и предоставляется в виде услуги. Из определения видно, что в основе этого решения должны лежать технологии виртуализации, получившие свое развитие в концепции виртуализации сетевых функций (NFV, от англ. Network Function Virtualization). Эта концепция, в свою очередь, лежит в основе программно-определяемых сетей (SDN, от англ. Software Defined Networking)[1, 2]. Вкупе с виртуализацией системы хранения данных (SDS, от англ. Software Defined Storage) мы получаем виртуализированную инфраструктуру дата-центра, и, в конечном счете, некоторую абстрактную модель программно-определяемого ЦОД. Далее будет рассмотрено алгоритмическое обеспечение функции LBaaS для SDDC.

Разработке алгоритмов управления, ориентированных на тип трафика, посвящены работы Zhang Q., Morselli R., Riska A. В работах Chandra A., Sahu S., Pacifici G. [3–5] описаны динамические алгоритмы, которые собирают информацию о загрузке серверов. Таким образом, в научной литературе уделяется внимание вопросам управления входным потоком и связанными с ними аспектами, что подчеркивает важность и актуальность решаемой научно-технической задачи.

Однако известные алгоритмы управления входным потоком имеют ряд недостатков: неравномерное распределение нагрузки между серверами, низкая производительность, недостаточный учет динамики системы, большие накладные расходы ресурсов, значительное время ответа [6]. Указанные недостатки требуют разработки обновленных методов и алгоритмов управления процессом распределения запросов между узлами сети.

Модель динамического алгоритма балансировки нагрузки

Анализируя недостатки классических алгоритмов, можно сделать вывод о целесообразности разработки динамического контентозависимого алгоритма с модифицированной обратной связью. Концепцию предлагаемого алгоритма можно представить в виде схемы, представленной на рисунке 1. Предлагается разработать алгоритм, который будет учитывать тип запроса и динамическое состояние сервера. При поступлении запроса на балансировщик нагрузки определяется тип трафика, и направляется на сервер, который обрабатывает запросы данного класса (типа). При перегрузке одного из серверов, запрос направляется на низкозагруженный сервер, если такой существует, или на наименее загруженный.

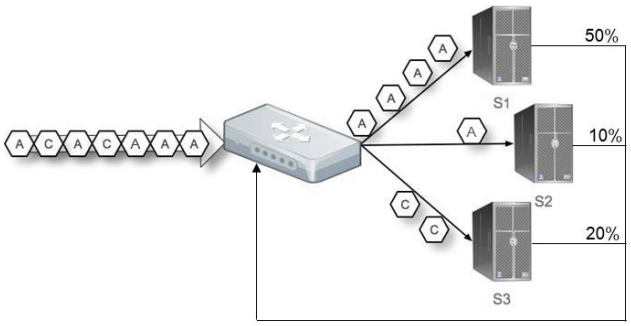

Рис. 1. Схема распределения входящих запросов

Состояние и доступность сетевых узлов (серверов) контролируется путем периодического обмена служебной информацией между всеми серверами и контроллером с приложением системы балансировки нагрузки [7]. Рассматриваемая система состоит из группы серверов, коммутатора OpenFlowSwitch [8] и контроллера [9]. Предварительно трафик попадает на систему DPI (от англ. Deep packet inspection – технология накопления статистических данных, проверки и фильтрации сетевых пакетов по их содержимому), которая

идентифицирует тип сессии в рамках соединения; коммутатор выполняет только функции передачи данных. Подсистема балансировки нагрузки и подсистема управления и мониторинга, которые тесно взаимодействуют друг с другом в рамках данной концепции, реализованы в виде соответствующих приложений на контроллере.

Динамический алгоритм управления входным потоком, который разрабатывается в работе, умеет использовать прогностические оценки интенсивности входящего потока запросов каждого класса. Это позволит предотвратить перегрузки и отслеживать тенденции загрузки отдельных узлов. Совершенствование методики краткосрочного прогноза позволит системе быстрее реагировать на флуктуации во входящем потоке. Для сокращения количества служебной информации и более точного отслеживания изменений во входящем потоке необходимо разработать адаптивный алгоритм мониторинга [10].

Алгоритм должен обеспечивать высокие показатели производительности, пропускной способности и отказоустойчивости (автоматически обнаруживая сбои узлов и перераспределяя поток данных среди оставшихся) и низкое время отклика [11, 12].

Математически задачу динамического управления потоком запросов можно представить следующей зависимостью (рисунок 2):

$$X = f(N, G, \overline{v}), \qquad (1)$$

где X – матрица распределения запросов в сети; N – множество серверов кластера $N = \{N_j\}$; G – множество характеристик серверов $G = \{U_{j\max}, U_j\}$, $U_{j\max}$ – максимальная загруженность сервера, U_j – текущая загруженность сервера; $\overline{v}$ – характеристика входящего потока запросов.

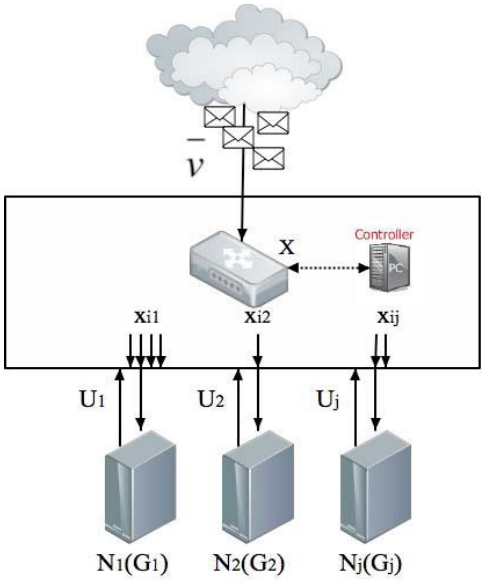

Рис. 2. Динамическое управление потоком запросов

Телекоммуникационную сеть ЦОД можно представить в виде открытой системы массового обслуживания со следующей совокупностью характеристик:

1) множеством серверов сети $\{N_1, N_2, \ldots, N_j\}$, которые образуют кластер, где $j = 1, N$;

2) интенсивностью входящего потока $\lambda_i (i = 1, M)$, где i – класс запроса, $\lambda_i^*(k)$ – прогнозируемое количество запросов i-го класса на k-ом шаге;

3) матрицей распределения запросов $X(k) = [x_{ij}]$, $(i = 1, M;\ j = 1, N)$, где x_{ij} – доля $\lambda_i^*(k)$-х запросов, которые обрабатываются на j-ом сервере;

4) законами распределения времени обслуживания $F_1(t), F_2(t), \ldots, F_j(t)$ и дисциплинами обслуживания запросов на серверах;

5) длиной очереди на серверах Q_j.

Запросы описываются вектором:

$$\overline{v} = (t, i, j, z_{ij}, m_{resource\_i}),$$

где t – время поступления запроса к контроллеру, i – класс запроса, j – номер сервера, который обрабатывает запрос, z_{ij}– загруженность j-го сервера (создается i-м классом запросов), $m_{resource\_i}$ – количество обращений i-го запроса к ресурсу, который обрабатывает.

Как только входящий поток будет создавать нагрузки, которые будут превышать максимальную загруженность сервера, то есть не будет выполняться условие (2), то на сервере будут возникать очереди и связанные с ними задержки:

$$U_j > U_{j\max}, \qquad (2)$$

где $U_{J\,\max}$ – максимальная загруженность сервера.

С целью внедрения верхней границы подобных задержек на узлах сети общую буферную емкость ограничивают, то есть для каждого сервера определяют текущую и максимальную емкость, обозначив их q_j и $q_j^{\max}$ соответственно. Как только $q_j > q_{j\,\max}$, запросы считают потерянными.

Для обеспечения динамического распределения запросов по серверам на k-ом шаге определяется матрица распределения запросов в сети:

$$X(k) = [x_{ij}], \qquad (i = 1, M;\ j = 1, N). \qquad (3)$$

По результатам расчета коэффициентов матрицы $X(k)$ рассчитывают значение загрузки j-го сервера на k-ом шаге – $U_j(k)$:

$$U_j(k) = U_j(k-1) + \sum_{i=1}^{M} \lambda_{ij}^*(k) \cdot x_{ij}(k) \cdot z_{ij}. \qquad (4)$$

Алгоритм управления входным потоком должен распределять запросы по серверам так, чтобы отклонение по загруженности серверов было минимальным, то есть:

$$s = \frac{\sum_{j=1}^{N}(\overline{U} - U_j(k))^2}{N} \to \min, \qquad (5)$$

где

$$\begin{cases} U_1(k-1) + \sum_{i=1}^{M} \lambda_i^*(k) \cdot x_{i1}(k) \cdot z_{i1} = U_1(k) \\ U_2(k-1) + \sum_{i=1}^{M} \lambda_i^*(k) \cdot x_{i2}(k) \cdot z_{i2} = U_2(k) \\ \dots \\ U_j(k-1) + \sum_{i=1}^{M} \lambda_i^*(k) \cdot x_{ij}(k) \cdot z_{ij} = U_j(k) \\ \overline{U} = \frac{\sum_{j=1}^{N} U_j(k)}{N} \end{cases};$$

$$\sum_{j=1}^{N} x_{ij}(k) = 1, \qquad i = (\overline{1, M});$$

$$\sum_{j=1}^{N} \lambda_{ij}^*(k) = \lambda_i^*, \qquad i = (\overline{1, M});$$

$$x_{ij}(k) > 0;$$

$$z: i \times j, \qquad i = (\overline{1, M}), \qquad j = (\overline{1, N}).$$

Алгоритм управления входным потоком

Основной идеей разрабатываемого алгоритма является распределение запросов пользователей на основании прогноза входящего потока по его классам. При этом горизонт прогноза на каждом шагу меняется в зависимости от изменений входящего потока. В работе алгоритма учитывается не только текущее состояние серверов, но и их производительность. Идею разрабатываемого алгоритма можно представить в виде блок-схемы, приведенной на рисунке 3.

Алгоритм выполняется циклически: в результате оптимизации в каждом цикле определяется матрица распределения запросов по узлам сети. Алгоритм представляет собой последовательность следующих шагов.

Шаг. 1. Собирается статистическая информация:

а) об интенсивности входящего потока запросов $\lambda(k-2), \lambda(k-1)$;

б) о состоянии серверов (утилизация CPU – U_j; загруженность j-го сервера, создаваемая i-м классом запросов – Z_{ij}; среднее время обслуживания запросов i-го класса – W_{ij}).

Шаг 2. На основании статистических данных шага 1 рассчитывается коэффициент пульсации входящего потока запросов – $b_m(k-1), b_m(k)$.

Шаг 3. Рассчитывается длина шага (далее – интервал) мониторинга и длина горизонта прогноза $d(k)$ по формуле (7).

Шаг 4. Прогнозируется интенсивность входящего потока для каждого класса запроса λ_i^* на длину горизонта $d(k)$.

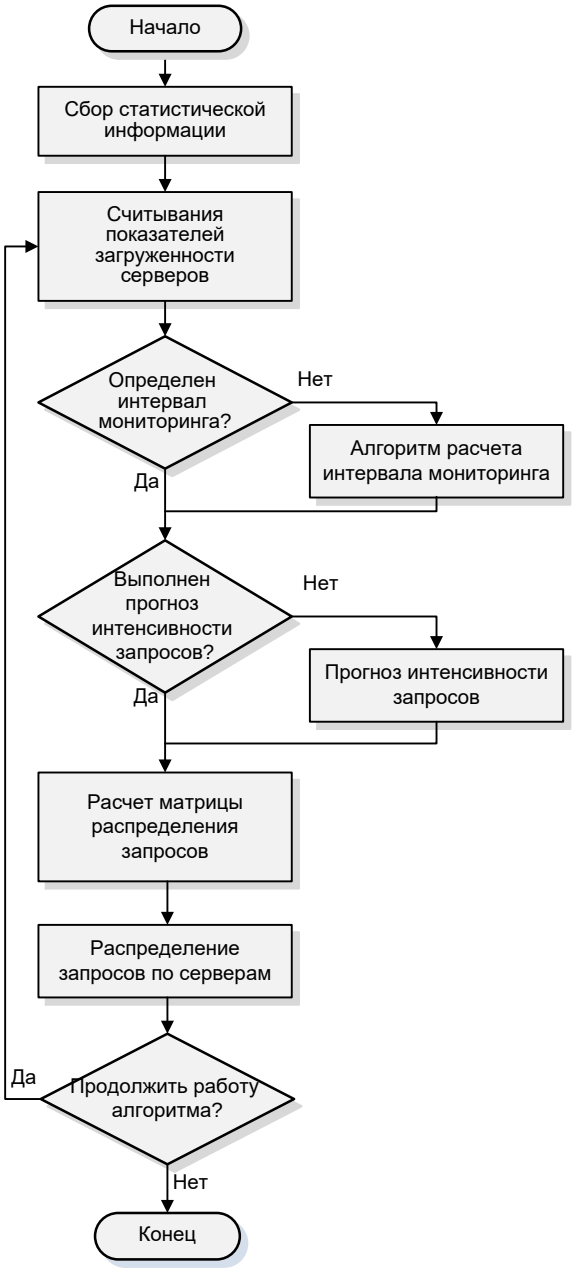

Рис. 3. Блок-схема алгоритма управления входным потоком

Шаг 5. Определяется матрица распределения запросов по узлам сети $\overline{x}(k)$ по формуле (3) с учетом класса запроса и загруженности сервера. На основании полученных данных прогнозируется загрузка серверов на следующем интервале.

Шаг 6. Запросы направляются на серверы в соответствии с заданным алгоритмом в пределах каждого класса запросов.

Шаг 7. Осуществляется ревизия прогнозируемого количества запросов $\lambda_i(k) - \lambda_i^*(k)$, которые распределяются согласно алгоритму на наименее загруженный сервер. При этом переоценка прогноза $\lambda_i^*(k) - \lambda_i(k)$ не учитывается алгоритмом, как не вносящая существенных изменений в нагрузку.

Шаг 8. Снимаются данные о загруженности серверов $U_j(k)$ и передаются на контроллер, для расчета нового распределения запросов $\bar{x}(k+1)$.

Для эффективной работы разрабатываемого алгоритма необходимо обеспечить мониторинг состояния серверов. Основной из существующих проблем мониторинга является его точность: при попытке достижения высокой точности служебная информация приобретает избыточный характер.

Шаг 9. Мониторинг состояния серверов, который можно осуществить тремя способами:

1) после каждого поступившего запроса;

2) в фиксированные промежутки времени, определяемые статическим алгоритмом;

3) в нефиксированные промежутки времени, определяемые динамическим алгоритмом.

Информация, полученная **первым** способом, является наибольшей по объему, т. к. измерения проводятся после каждого поступившего запроса. При **втором** способе количество информации постоянно, но требуется определить интервал съема информации, чтобы объем информации не был избыточным и недостаточным. При **третьем** способе количество информации зависит от частоты интервалов контроля, который должен приспосабливаться к потоку поступающих запросов.

На основании вышесказанного, использование динамически меняющегося интервала является наиболее приемлемым с точки зрения уменьшения избыточности данных. При таком способе частота мониторинга будет зависеть от количества всплесков (пульсации) входящего потока. Наличие отдельных всплесков в интернет-трафике является одной из его особенностей. Интервал мониторинга должен сокращаться, если во входящем потоке обнаружен всплеск & vv.

Адаптивный алгоритм мониторинга

Для решения задачи оценки интервала мониторинга предложен соответствующий алгоритм, блок-схема которого приведена на рисунке 4.

Алгоритм представляет собой последовательность следующих шагов:

Шаг 1. Собирается статистическая информация об интенсивности входящего потока запросов $\lambda(k-2), \lambda(k-1)$ за два предыдущих интервала и о длине прошлого интервала мониторинга $d(k)$.

Шаг 2. Определяется коэффициент пульсации $b_m(k-1), b_m(k)$, используя формулу (6).

Шаг 3. Для расчета продолжительности интервала мониторинга сравнивается $b_m(k-1)$ и $b_m(k)$. Если $b_m(k) > b_m(k-1)$, то продолжительность интервала уменьшается, в случае $b_m(k) < b_m(k-1)$ – продолжительность интервала увеличивается по сравнению с прошлой итерацией.

Шаг 4. Рассчитывается продолжительность очередного интервала $d(k+1)$ по формуле (7).

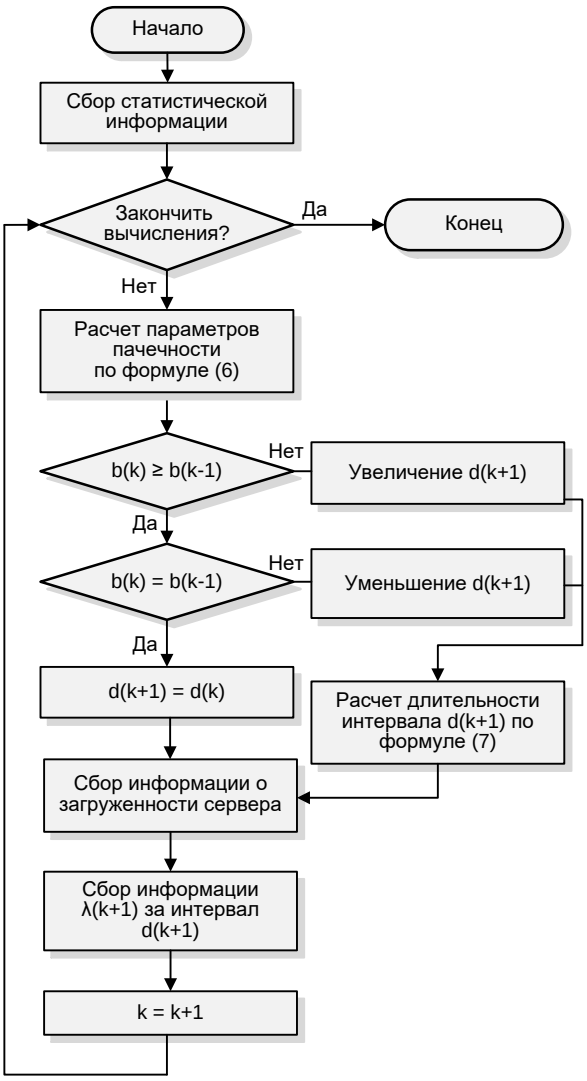

Рис. 4. Блок-схема алгоритма мониторинга

Шаг 5. Собирается статистическая информация об интенсивности входящего потока запросов $\lambda(k-1), \lambda(k)$ за два предыдущих интервала и о длине прошлого интервала мониторинга $d(k-1)$.

Шаг 6. Определяется коэффициент пульсации $b_m(k-1), b_m(k)$ по формуле (6), используя следующую методику:

1) Подсчитываем количество http или других запросов – L, попавших в интервал длиной d.

2) Интенсивность поступления запросов в интервале определяем, как $\lambda = \frac{L}{d}$.

3) Интервал разбиваем на n равных подинтервалов, длиной $h = \frac{d}{n}$.

4) Определяем $L(h)$ – количество запросов, попавших в интервал h.

5) Находим λ_h – интенсивность поступления запросов за интервал длиной h, как $\lambda_h = \frac{L(h)}{h}$.

6) Вычисляем L^* – общее количество запросов в интервалах, что удовлетворяют условию $\lambda_h > \lambda$.

7) Определяем g – количество интервалов, удовлетворяющих условию $\lambda_h > \lambda$.

8) Рассчитываем коэффициент пульсации по формуле:

$$b_m = \frac{g}{n} \cdot \left(1 + \frac{\lambda(k) - \lambda(k-1)}{\lambda(k-1)}\right); \qquad (6)$$

9) Так как продолжительность каждого слота d меняется в зависимости от значения пульсаций входящего потока, то продолжительность интервала мониторинга d (k +1) определяем как:

$$d(k+1) = \frac{b_m(k-1)}{b_m(k)} \cdot d(k). \qquad (7)$$

Шаг 7. Для расчета продолжительности интервала мониторинга сравниваем $b_m(k-1)$ и $b_m(k)$. Если $b_m(k) > b_m(k-1)$ – продолжительность интервала уменьшается, в случае $b_m(k) < b_m(k-1)$ – продолжительность интервала увеличивается по сравнению с прошлой итерацией.

Шаг 8. Рассчитывается продолжительность следующего интервала *d(k)* по формуле (7).

Предложенный алгоритм адаптирован к флуктуациям во входящем потоке, что позволяет сократить количество служебной информации и предотвратить перегрузку за счет качества оценок прогнозирования. Горизонт прогнозирования на k-ой итерации зависит от значений пульсаций, то есть, длительность следующего слота уменьшается, если был обнаружен всплеск & vv.

Динамический алгоритм балансировки нагрузки в сетях ЦОД

Представленные выше алгоритмы и методику необходимо объединить в единый динамический алгоритм балансировки нагрузки (с учетом специфики реализации на базе инфраструктуры SDDC). При этом стоит отметить, что анализ состояния загруженности серверов и принятие решения по адресации трафика лежит на контроллере, который должен быть установлен в ЦОД и иметь интерфейсы мониторинга ко всем активным его серверам. Алгоритм, блок-схема которого приведена на рисунке 5, представляет собой последовательность следующих шагов:

Шаг 1. Клиент обращается к одному из серверов (зная его IP-адрес, к примеру, IP_1).

Шаг 2. На граничный с серверами коммутатор приходит пакет от клиента.

Шаг 3. Коммутатор видит, что клиент хочет обратиться к одному из серверов в первый раз, то есть, у него еще нет сессии с сервером.

Шаг 4. Коммутатор отсылает пакет клиента на контроллер.

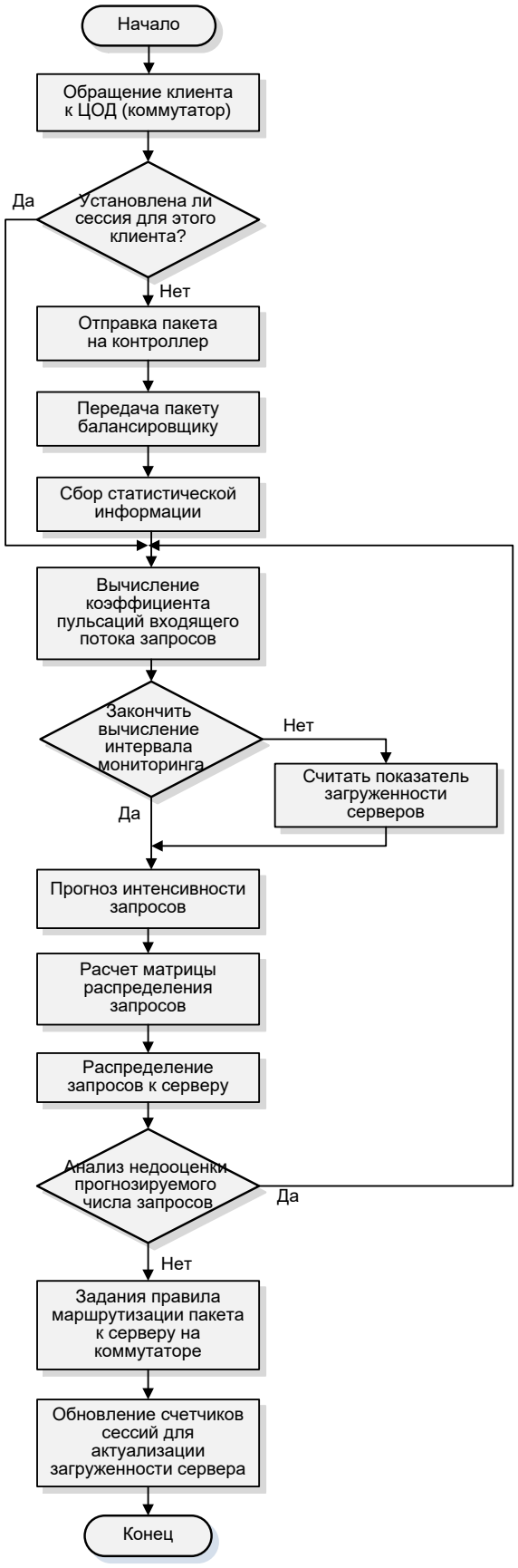

Рис. 5. Блок-схема динамического алгоритма балансировки нагрузки

Шаг 5. Контроллер видит, что IP-адрес назначения – один из балансируемых серверов, поэтому передает обработку пакета приложению по балансировке нагрузки.

Шаг 6. Приложение мониторинга производит сбор и анализ статистической информации: об интенсивности входящего потока запросов $\lambda(k-2)$, $\lambda(k-1)$, о состоянии серверов (утилизация CPU – U_j; загруженность j–го сервера, создаваемая i-м классом запросов – Z_{ij}; среднее время обслуживания запросов i-го класса – W_{ij}) за несколько итераций, а также о длине прошлого интервала мониторинга $d(k)$.

Шаг 7. Вычисляется коэффициент пульсаций входящего потока запросов $b_m(k-2), b_m(k-1)$ по формуле (6) на основании статистических данных шага 6.

Примечание. Коэффициент пульсации $b_m \in [0; 1]$ – это доля времени, в течение которого мгновенная интенсивность поступления запросов превышает среднюю интенсивность.

Шаг 8. Рассчитывается интервал мониторинга загруженности серверов и горизонт прогноза $d(k)$ по формуле (7).

Шаг 9. Прогнозируется интенсивность входящего потока по каждому типу запросов λ_i^* на длину горизонта $d(k)$, определяемую на шаге 8.

Шаг 10. Приложение балансировки производит расчет матрицы распределения запросов по узлам сети $\overline{x}(k)$ с учетом класса запроса и загруженности сервера. На основании полученных данных прогнозируется загрузки серверов на следующей итерации.

Шаг 11. Осуществляется распределение запросов по серверам согласно алгоритму в пределах каждого класса.

Шаг 12. Осуществляется ревизия прогнозируемого количества запросов $\lambda_i(k) - \lambda_i^*(k)$, которые распределяются согласно алгоритму балансировки на наименее загруженный сервер.

Шаг 13. Приложение определяет, на какой из серверов передать обработку запроса клиента (например, IP_2).

Шаг 14. На граничном коммутаторе задается правило, что если от клиента приходит пакет с адресом назначения IP_1, то адрес назначения подменяется на IP_2, и пакет передается на порт, соответствующий серверу IP_2.

Шаг 15. Также на коммутаторе задается правило, что если от сервера IP_2 приходит пакет с адресом назначения клиента, то адрес источника подменяется на IP_1 и передается на порт клиента.

Шаг 16. В приложении обновляются значения счетчиков сессий серверов для сохранения актуальности загруженности серверов $U_j(k)$ и расчета нового распределения запросов $\bar{x}(k+1)$.

Шаг 17. Если на граничный коммутатор приходит пакет о закрытии соединения клиентом или сервером, то он также передается контроллеру, который передает управление приложению по балансировке нагрузки, которое декрементирует значение счетчика активированных сессий на сервере.

Алгоритм балансировки нагрузки должен распределять запросы по серверам так, чтобы отклонение загруженности серверов от среднего значения было минимальным, то есть:

$$s = \frac{\sum_{j=1}^{N}(\overline{U} - U_j(k))^2}{N} \to \min, \qquad (8)$$

Алгоритм вычисляет загруженность j-го сервера на k-ом шаге, основываясь на сумме его загруженности до этого момента $U_j(k-1)$ и нагрузке, создаваемой долей $x_{ij}(k)$ прогнозируемой интенсивности потока $\lambda_i^*(k)$ i-го класса, которая будет обрабатываться на данном сервере.

В качестве искомой матрицы выступает матрица распределения запросов:

$$X(k) = \left[x_{ij}\right], \qquad (i = 1, M;\ \ j = 1, N).$$

В результате расчета обеспечивается динамическое распределение нагрузки по серверам в каждом цикле.

Разработанный алгоритм отвечает всем требованиям к балансировке, соответствует концепции SDN, и адаптирован к изменениям интенсивности входящего потока для уменьшения времени ответа, вероятности потерь и повышения производительности системы.

Заключение

В заключение стоит уточнить, что построение и эксплуатация ЦОД не ограничивается выбором алгоритма балансировки трафика между серверами, необходимо учесть индивидуальные технические параметры серверов, варианты резервирования, синхронизации и другие технологические аспекты, которые могут влиять на условия применения адаптивных алгоритмов.

При этом применение SDN не позволяет автоматизировать решение этих вопросов, но оставляет возможность для интеграции рассмотренных алгоритмов в том сегменте ЦОД, где это необходимо и целесообразно.

Список используемых источников

1. Vladyko A., Letenko I., Lezhepekov A., Buinevich M. Fuzzy Model of Dynamic Traffic Management in Software-Defined Mobile Networks // Lecture Notes in Computer Science. 2016. Vol. 9870. PP. 561–570.
2. Amelyanovich A., Shpakov M., Muthanna A., Buinevich M., Vladyko A. Centralized Control of Traffic Flows in Wireless Lans Based on the SDN Concept // Systems of Signal Synchronization, Generating and Processing in Telecommunications (SINKHROINFO). 2017. PP. 1–5.
3. Pacifici G., Spreitzer M., Tantawi A., Youssef A. Performance Management for Cluster-Based Web Services // IEEE Journal on Selected Areas in Communications. 2005. 23 (12). PP. 2333–2343.
4. Urgaonkar B., Pacifici G., Shenoy P., Spreitzer M., Tantawi A. An Analytical Model for Multi-Tier Internet Services and its Applications // Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. 2005. PP. 291–302.
5. Urgaonkar B., Chandra A. Dynamic Provisioning of Multi-tier Internet Applications // Proceedings of the 2nd International Conference on Automatic Computing. 2005. PP. 217–228.
6. Елагин В.С., Зобнин А.А. Аспекты реализации системы законного перехвата трафика в сетях SDN // Вестник связи. 2016. № 12. С. 6–9.
7. Елагин В.С. Подходы к моделированию систем законного перехвата трафика в SDN // Актуальные проблемы инфотелекоммуникаций в науке и образовании: сборник научных статей V международной научно-технической и научно-методической конференции. СПБГУТ. 2016. С. 353–358.
8. ONF TS-025. OpenFlow Switch Specification. Version 1.5.1. 2015.
9. Елагин В.С., Сорокин В.А. Исследование технологических возможностей внедрения СОРМ в SDN // Труды учебных заведений связи. 2017. Т. 3. № 1. С. 28–35.
10. Labes S., et al. Standardization Approaches within Cloud Computing: Evaluation of Infrastructure as a Service Architecture // Federated Conference on Computer Science and Information Systems. 2012. PP. 923–930.
11. Al-Fares M., Radhakrishnan S., Raghavan B., Huang N., Vahdat A. Hedera: Dynamic Flow Scheduling for Data Center Networks // Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation. USENIX Association. April 2010. PP. 19–19.
12. Parfenov D., Bolodurina I., Shukhman A. Efficient access to multimedia resources in distributed systems of distance learning // Proceedings of the IEEE Global Engineering Education Conference (EDUCON). 2013. PP. 1228–1231.

* * *

DYNAMIC LOAD BALANCING IN SOFTWARE-DEFINED NETWORK

V. Elagin[1]

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

Article info:
Article in Russian

For citation: Elagin V. Dynamic Load Balancing in Software-Defined Network // Proceedings of Telecommunication Univercities. 2017. Vol. 3. Iss. 3. PP. 60–67.

Annotation: *In regard with the increasingly complex topology of the network infrastructure of data centers and its requirements of performance, reliability and fault tolerance, the complexity of managing network infrastructure of the data center are also constantly growing. In this regard, there is a need to modification manual configuration of the network infrastructure of data centers by automated means. One of the perspective directions of modernization of the existing network architecture is the concept of SDN. Software-defined data center (Software Defined Data Center, SDDC) is a virtualized structure aimed at providing services. Within this concept it is necessary to develop methods of traffic management and balancing within the data center networks. The article presents a mathematical model of the dynamic load balancing algorithm in data center networks.*

Key words: *Data center, software defined networking, SDN, dynamic load balancing.*