Научная статья

УДК 004.75
DOI 10.25205/1818-7900-2025-23-3-23-31

# Анализ методов балансировки сетевого трафика в высоконагруженных сетях

**Надежда Валерьевна Красовская**

Новосибирский государственный университет
Новосибирск, Россия

n.krasovskaya@g.nsu.ru

*Аннотация*

В статье проводится комплексный анализ методов балансировки сетевого трафика в высоконагруженных сетях. Рассматриваются фундаментальные принципы распределения нагрузки, включая требования к сохранению целостности TCP-сессий и особенности идентификации сетевых потоков. Детально исследуются современные алгоритмы балансировки: от классического ECMP до перспективных решений на основе согласованного хеширования, HRW и Maglev. Особое внимание уделяется выбору оптимальных хеш-функций для идентификации сетевых потоков и сравнительному анализу характеристик различных методов. Представленные результаты позволяют выработать рекомендации по выбору стратегии балансировки для различных сценариев эксплуатации высоконагруженных сетевых инфраструктур.

*Ключевые слова*

балансировка сетевого трафика, высоконагруженные сети, согласованное хеширование, распределение потоков, Maglev

*Для цитирования*

*Красовская Н. В.* Анализ методов балансировки сетевого трафика в высоконагруженных сетях // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 3. С. 23–31. DOI 10.25205/1818-7900-2025-23-3-23-31

# Analysis of Network Traffic Load Balancing Techniques in High-Load Networks

**Nadezhda V. Krasovskaya**

Novosibirsk State University,
Novosibirsk, Russian Federation

n.krasovskaya@g.nsu.ru

*Abstract*

This article provides a comprehensive analysis of network traffic balancing methods in high-load networks. It examines fundamental load distribution principles, including requirements for maintaining TCP session integrity and specific aspects of network flow identification. The study offers a detailed investigation of modern load balancing algorithms – from classical ECMP to advanced solutions based on consistent hashing, HRW, and Maglev. Particular attention is given to the selection of optimal hash functions for network flow identification and comparative analysis of different methods' characteristics. The presented results enable the development of recommendations for choosing balancing strategies in various operational scenarios of high-load network infrastructures.

© Красовская Н. В., 2025

*Keywords*
network load balancing, high-load networks, consistent hashing, flow distribution, Maglev

*For citation*
Krasovskaya N. V. Analysis of Network Traffic Load Balancing Techniques in High-Load Networks. *Vestnik NSU. Series: Information Technologies*, 2025, vol. 23, no. 2, pp. 23–31 (in Russ.) DOI 10.25205/1818-7900-2025-23-3-23-31

## Введение

Современные высоконагруженные сети, включая центры обработки данных, облачные платформы и телекоммуникационные системы, требуют эффективного управления сетевым трафиком для обеспечения стабильной работы и высокой производительности. Одной из ключевых задач в таких системах является балансировка сетевого трафика, которая заключается в оптимальном распределении нагрузки между серверами, маршрутизаторами и другими сетевыми узлами. Ее цель – минимизация задержек, предотвращение перегрузки критических элементов инфраструктуры и повышение отказоустойчивости системы в целом.

Задача балансировки сетевого трафика включает не только равномерное распределение запросов, но и динамическую адаптацию к изменениям нагрузки, учет топологии сети, пропускной способности каналов и состояния серверов. С ростом объемов данных и усложнением сетевых архитектур традиционные подходы к балансировке могут становиться недостаточно эффективными, что стимулирует разработку новых алгоритмов и методов управления трафиком.

К основным требованиям к методам балансировки сетевого трафика относятся:

- масштабируемость – способность системы сохранять эффективность при увеличении числа узлов и объема передаваемых данных;
- отказоустойчивость – минимизация влияния сбоев на доступность сервисов;
- эффективность – снижение задержек и оптимальное использование сетевых ресурсов;
- гибкость – адаптация к изменениям нагрузки и конфигурации сети;
- простота управления – баланс между сложностью алгоритма и удобством его практической реализации.

В данной статье проводится анализ современных методов балансировки сетевого трафика на сетевом (L3) и транспортном (L4) уровнях сетевой модели OSI (Open Systems Interconnection), оцениваются их преимущества и ограничения.

## Принципы балансировки сетевых потоков

Существенный объем трафика современных сетей использует протокол TCP, критически зависимый от порядка доставки пакетов. Нарушение этого порядка при балансировке (когда пакеты одного соединения обрабатываются разными узлами) приводит к серьезным проблемам: повторным передачам, росту задержек и разрывам соединений [1]. Для предотвращения этих ситуаций применяется принцип сохранения целостности потока, требующий направления всех пакетов TCP-сессии на один сервер.

Для идентификации потоков используются два основных метода. Базовый двухкортежный подход анализирует только IP-адреса источника и назначения (L3 уровень), что обеспечивает минимальные накладные расходы, но приводит к объединению всех соединений между парой узлов в единый поток. Более совершенный пятикортежный метод (L4 уровень) дополнительно учитывает порты и тип протокола, позволяя различать отдельные соединения между теми же узлами. Это особенно важно для сред NAT (Network Address Translation), где множество устройств используют один внешний IP-адрес [2, с. 721].

Выбор между методами представляет собой компромисс: двухкортежный обеспечивает высокую скорость обработки при низкой вариативности IP-адресов, тогда как пятикортежный дает более точное распределение нагрузки ценой увеличения вычислительных затрат. Современные решения часто используют гибридный подход, комбинируя оба метода для оптимального баланса производительности и качества. Окончательный выбор узла балансировки выполняется на основе хеш-функции, применяемой к выбранным параметрам потока.

## Существующие реализации

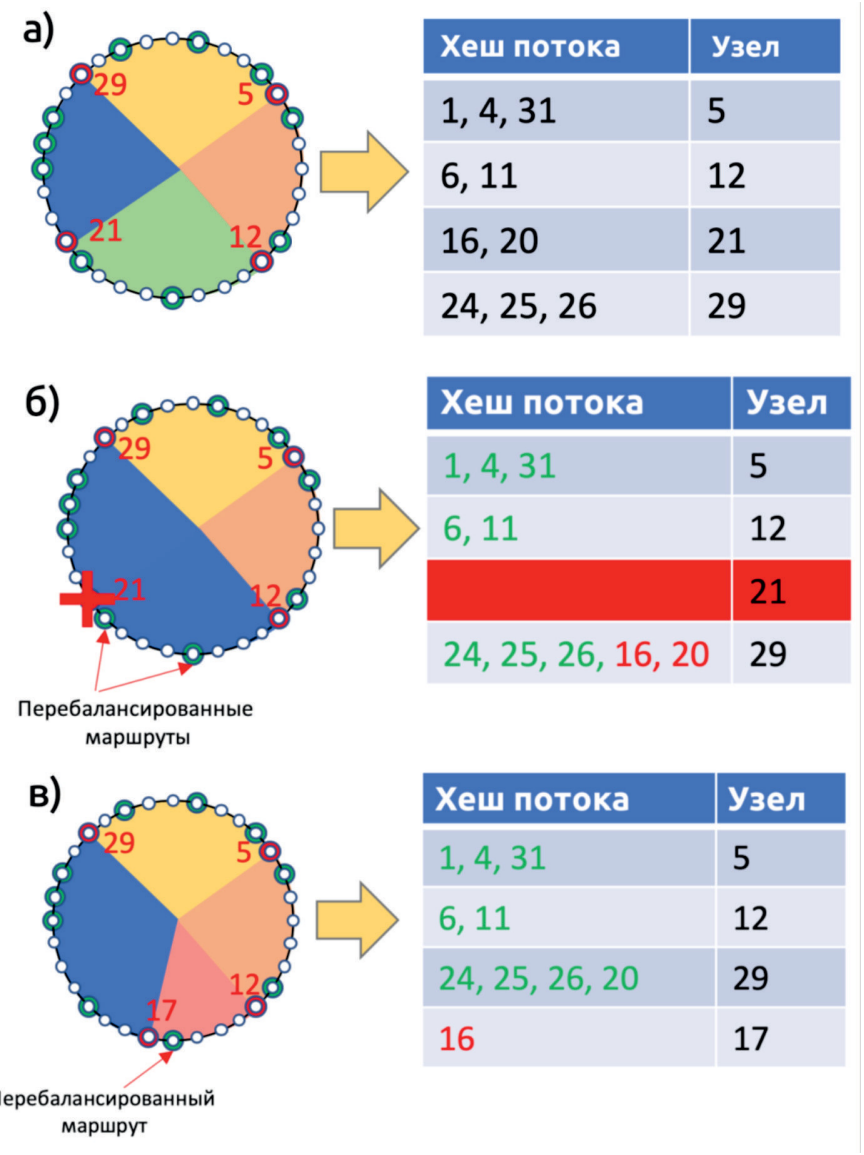
Широко распространенный в современных центрах обработки данных подход ECMP (Equal-Cost Multi-Path) [3] реализует балансировку на основе попотокового распределения, используя хеширование заголовков пакетов (обычно пятикортежный подход) для равномерного распределения трафика по доступным путям. Хотя этот подход отличается простотой реализации, он обладает рядом существенных ограничений, особенно критичных в высоконагруженных сетях. Главная проблема ECMP заключается в его неспособности обеспечить равномерное распределение нагрузки – при малом количестве потоков или использовании хеш-функций с низким лавинным эффектом возникает значительный дисбаланс. Кроме того, ECMP реализован на основе сетевого стека Linux, который имеет ряд недостатков, существенно влияющих на производительность, таких как контекстные переключения, копирования данных между пространствами ядра и пользователя, а также ограничений буферизации [4].

## Алгоритмы балансировки

Критически важным местом является алгоритм выбора узла по значению хеш-функции. Он должен равномерно распределять нагрузку по узлам и не должен иметь высокую алгоритмическую сложность. Наиболее простой метод – прямое хеширование с операцией взятия остатка. В этом случае для каждого сетевого потока вычисляется хеш-функция от ключа, после чего номер целевого узла определяется как остаток от деления полученного хеша на количество доступных узлов. Главное преимущество этого подхода – его предельная простота и минимальные вычислительные затраты. Однако при изменении количества узлов (например, при выходе одного из них из строя) происходит массовое перераспределение потоков – практически все существующие соединения будут перенаправлены на новые узлы, что нарушает стабильность работы системы.

Более совершенный метод – согласованное хеширование – решает проблему массовой перебалансировки. Его фундаментальное отличие от прямого хеширования с взятием остатка заключается в использовании абстрактного хеш-кольца – одномерного замкнутого пространства, на котором независимо размещаются как узлы, так и ключи. Размер кольца выбирается исходя из требуемой точности распределения. В практических реализациях обычно используется кольцо размером $2^M$ (например, $M = 32$ или $M = 64$), что позволяет равномерно распределять узлы и минимизировать коллизии. Пространство кольца нормализуется в диапазон [0, 2M – 1]. Проецирование узлов на кольцо осуществляется с помощью устойчивой хеш-функции, которая преобразует идентификатор узла (IP-адрес или уникальное имя) в некоторое значение. Точка на кольце определяется путем взятия остатка от деления на размер кольца. Размещение ключей (потоков) выполняется аналогично: хеш-функция применяется к идентификатору потока (полям заголовка), и полученное значение проецируется на кольцо. Ключ ассоциируется с ближайшим узлом в направлении обхода кольца (по часовой стрелке) [5].

При исключении узла из балансировки его ключи перераспределяются на его следующего соседа по часовой стрелке по кольцу (рисунок, *б*). При добавлении узла (рисунок, *в*) он проецируется на хеш-кольцо согласно его значению хеш-функции, после этого все узлы, находящиеся между добавленным узлом и предыдущим на кольце, будут балансироваться на добавленный.




Отображение узлов и потоков на хеш-кольцо при согласованном хешировании:
*а* – исходное состояние, *б* – удаление узла из балансировки, *в* – добавление нового узла в балансировку.
Красными точками обозначены узлы, зелеными – потоки.
Mapping nodes and streams to a hash ring with consistent hashing: *а* – the initial state, *б* – the removal of a node from balancing, *В* – the addition of a new node to balancing
The red dots represent nodes, and the green dots represent streams.

Такой метод позволяет добиться минимальной перебалансировки в случаях сбоев или добавления новых узлов, однако вызывает неравномерность балансировки, особенно при небольшом числе узлов. Для повышения равномерности распределения каждый физический узел может быть представлен несколькими виртуальными точками (обычно 100–200 на узел), что статистически снижает дисперсию нагрузки на узел. Однако даже с виртуальными узлами сохраняется вероятность некоторой неравномерности, особенно при небольшом количестве физических узлов. Вычислительная сложность получения узла при согласованном хешировании: $O(\log N)$, где $N$ – число узлов.

Наиболее совершенным с теоретической точки зрения является метод хеширования рандеву (Rendezvous Hashing), также известный как Highest Random Weight (HRW) хеширование [6].

В этом подходе для каждого потока вычисляется вес относительно каждого доступного узла с помощью детерминированной хеш-функции, комбинирующей идентификатор потока и идентификатор узла. Поток направляется на тот узел, для которого вычислен наибольший вес.

Главное преимущество этого метода – исключительная стабильность распределения. При изменении количества узлов перераспределяются только те потоки, которые были закреплены за исчезнувшим узлом или которые должны теперь закрепиться за новым узлом. Все остальные потоки остаются на своих прежних узлах. При добавлении и удалении узлов перебалансировка происходит естественным образом. Кроме того, этот метод обеспечивает практически идеальное равномерное распределение нагрузки даже без использования виртуальных узлов.

Однако за эти преимущества приходится платить более высокой вычислительной сложностью – для каждого потока требуется вычислить хеш для каждого доступного узла, т. е. сложность данного алгоритма $O(N)$. Для систем с большим количеством узлов и для высоконагруженных систем это может стать существенной дополнительной нагрузкой.

Развитием подхода согласованного хеширования является алгоритм Maglev [7], предложенный Google в 2016 г. для высоконагруженных сетевых балансировщиков. В отличие от классического согласованного хеширования, где узлы размещаются на виртуальном кольце, Maglev использует детерминированное заполнение таблицы перестановок, что обеспечивает минимальную перебалансировку при изменении топологии и идеальное распределение нагрузки.

Основная идея Maglev заключается в двухэтапном построении таблицы соответствия потоков узлам. На первом этапе каждый узел генерирует список предпочтительных позиций в таблице на основе псевдослучайной перестановки, определяемой хеш-функцией от идентификатора узла. На втором этапе происходит итеративное заполнение таблицы: каждый узел последовательно занимает ближайшую свободную позицию из своего списка. В результате формируется таблица, где каждый поток, определяемый некоторым ключом, однозначно сопоставляется с узлом через $O(1)$ поиск по индексу.

Алгоритмическая сложность Maglev определяется размером таблицы $M$, выбираемым как простое число порядка 65 537. Инициализация требует $O(M \cdot N)$ операций, где $N$ – число узлов, но выполняется однократно при старте системы. Память расходуется на хранение таблицы ($O(M)$ элементов), что при типичных реализациях составляет ~512 КБ ($M$ = 655 37, 32-битные индексы).

Важным свойством Maglev является детерминированность перебалансировки: изменение топологии затрагивает только те потоки, чьи позиции в таблице перекрываются с модифицированными узлами. Это достигается за счет того, что перестановки для каждого узла остаются неизменными, а обновление таблицы происходит атомарно. Экспериментальные результаты Google показали, что даже при обработке 10 млн пакетов в секунду на узел Maglev обеспечивает равномерность распределения с отклонением менее 0,3 %, что делает его предпочтительным выбором для решения задачи балансировки сетевого трафика в высоконагруженных сетях.

## Выбор хеш-функции для отображения ключа на хеш-кольцо

Требования к хеш-функциям в согласованном хешировании:

- Детерминированность. Одинаковые ключи всегда дают одинаковые значения:

$$\forall\, k : h(k) = \text{const}.$$

- Выходные значения должны равномерно покрывать пространство $[0, 2^M)$:

$$P\left(h(k) \in [a,b]\right) \approx \frac{(b-a)}{2^M} \forall a, b.$$

- Устойчивость к коллизиям. Вероятность коллизии для двух произвольных ключей:

$$P\left(h\left(k_1\right)=h\left(k_2\right)\right)\approx 1/2^{\text{Ì}} .$$

- Лавинный эффект. Малое изменение входа вызывает значительное изменение выхода:

$$diff\left(k_1,k_2\right)=1\text{ бит}\Rightarrow diff\left(h\left(k_1\right),h\left(k_2\right)\right)\approx\frac{\text{Ì}}{2}\text{бит}.$$

Всем этим требованиям удовлетворяет семейство хеш-функций MurmurHash, в частности, функция MurmurHash3 [8]. Первая хеш-функция MurmurHash была разработана в 2008 году австралийским разработчиком Остином Эпплби (Austin Appleby) для решения конкретных задач в области распределенных систем и хеш-таблиц. Позднее появилась улучшенная версия MurmurHash2, имеющая более высокий лавинный эффект, а после – MurmurHash3, оптимизированная под x86/x64, что обеспечивало более высокую производительность.

**Сравнительная таблица методов балансировки сетевого трафика**

| Параметр сравнения | Прямое хеширование | Согласованное хеширование | Рандеву (HRW) | Maglev |
|---|---|---|---|---|
| Принцип работы | Остаток от деления хеша на число узлов | Хеш-кольцо с виртуальными узлами | Выбор узла с максимальным весом для каждого ключа | Таблица перестановок |
| Равномерность распределения | Низкая (зависит от хеш-функции) | Средняя (улучшается с виртуальными узлами) | Высокая | Очень высокая (>99,7 %) |
| Перебалансировка при изменении топологии | Полная (почти все потоки перераспределяются) | Частичная (только соседние потоки) | Минимальная (только затронутые узлы) | Минимальная (детерминированная) |
| Вычислительная сложность | $O(1)$ | $O(\log N)$ | $O(N)$ | $O(1)$ после инициализации |
| Память | $O(1)$ | $O(\text{V}\cdot N)$, $V = 100–200$ вирт. узлов | $O(1)$ | $O(M)$, $M \approx 65537$ |
| Устойчивость к асимметрии нагрузки | Низкая | Средняя | Высокая | Очень высокая |
| Хеш-функция | Любая простая | MurmurHash3 | MurmurHash3 | MurmurHash3 |
| Использование в реальных системах | Базовые маршрутизаторы | CDN, распределенные хранилища | Специализированные балансировщики | Высоконагруженные системы |
| Поддержка взвешенных узлов | Нет | Да (через количество виртуальных узлов) | Да | Да (через частоту в таблице) |
| Лучший сценарий применения | Статичные среды с малым числом узлов | Средние нагрузки, где важна стабильность | Системы с высокими требованиями к доступности | Очень высокие нагрузки (миллионы пакетов в секунду) |

MurmurHash3 обеспечивает почти идеальное равномерное распределение благодаря нескольким ключевым особенностям:

- Эффективное перемешивание бит: функция использует каскадные умножения и XOR-операции, которые разрушают любые паттерны во входных данных. Умножение на «магические» числа (хорошо подобранные константы) рассеивает биты, XOR + сдвиги дополнительно перемешивают хеш [8].
- Использование лавинного эффекта: MurmurHash3 спроектирована так, что изменение даже одного бита во входных данных меняет ~50 % бит в выходном хеше [9].
- Равномерно заполняет все пространство хеш-кольца.

Помимо этого, функция MurmurHash3 обладает высокой производительностью относительно других хеш-функций, что важно для высоконагруженных систем [9; 10]. Все это делает MurmurHash3 идеальной хеш-функцией для балансировщиков сетевого трафика, использующих алгоритмы согласованного хеширования (см. таблицу).

## Заключение

Проведенный анализ методов балансировки сетевого трафика в высоконагруженных сетях позволяет сделать следующие выводы.

- Ключевые требования к современным системам балансировки включают не только равномерное распределение нагрузки, но и сохранение целостности TCP-сессий, минимальную перебалансировку при изменениях топологии, а также высокую производительность обработки трафика.
- Методы идентификации потоков (двухкортежный и пятикортежный) представляют собой компромисс между точностью распределения и вычислительной сложностью. Выбор конкретного подхода должен определяться характеристиками сетевой инфраструктуры и особенностями трафика.

Современные алгоритмы балансировки демонстрируют существенную эволюцию – от простого ECMP к более совершенным методам:

- согласованное хеширование решает проблему массовой перебалансировки;
- алгоритм Maglev обеспечивает практически идеальное распределение нагрузки;
- HRW-хеширование предлагает оптимальное сочетание равномерности и стабильности.

Критически важным компонентом является выбор хеш-функции, где MurmurHash3 представляет собой оптимальное решение, сочетающее высокую производительность, равномерность распределения и выраженный лавинный эффект.

Перспективными направлениями развития технологий балансировки являются:

- гибридные подходы, адаптивно сочетающие разные методы;
- использование машинного обучения для прогнозирования нагрузки.

Представленные в статье методы и алгоритмы составляют основу для построения эффективных систем балансировки, способных удовлетворять растущим требованиям современных высоконагруженных сетевых инфраструктур.

## Список литературы


1. **Yasin W., Ibrahim H., Abdul Hamid N. A. W., Udzir N. I.** Performance Analysis of Transport Control Protocol Flavours in the Existence of Packet Reordering Phenomena // Digital Enterprise and Information Systems, 2011. P. 569–579. DOI: 10.1007/978-3-642-22603-8_50.
2. **Олифер В. Г., Олифер Н. А.** Компьютерные сети, принципы, технологии, протоколы: учебник для вузов. 3-е изд. СПб.: Питер, 2007.

3. **Hopps C.** Analysis of an Equal-Cost Multi-Path Algorithm, RFC Editor, Nov. 2000. DOI 10.17487/rfc2992.
4. **Ларин Д. В., Гетьман А. И.** Средства захвата и обработки высокоскоростного сетевого трафика // Тр. ИСП РАН. 2021. № 4. С. 49–68. DOI 10.15514/ispras-2021-33(4)-4
5. **Karger D., Lehman E., Leighton T., Panigrahy R., Levine M., Lewin D.** Consistent hashing and random trees // Proceedings of the twenty-ninth annual ACM symposium on Theory of computing – STOC '97, 1997. P. 654–663, DOI 10.1145/258533.258660
6. **Thaler D. G., Ravishankar C. V.** Using name-based mappings to increase hit rates // IEEE/ACM Transactions on Networking. 1998. Vol. 6. No. 1. P. 1–14. DOI 10.1109/90.663936
7. **Eisenbud D. E., Yi C., Contavalli C., Smith C., Kononov R., Mann-Hielscher E. et al.** Maglev: a fast and reliable software network load balancer // Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI'16). 2016. Usenix Association. P. 523–535.
8. **Appleby A.** Murmurhash3 // Github. URL: https://github.com/aappleby/smhasher/wiki/MurmurHash (date of access: 17.04.2025).
9. **Suparn P., Mamta R.** Evaluation and Categorization of Hashing Algorithms Based on Their Applications // IAENG International Journal of Applied Mathematics. 2025. No 55. P. 540–552.
10. **Akoto-Adjepong V., Asante M., Okyere-Gyamfi S.** An Enhanced Non-Cryptographic Hash Function // International Journal of Computer Applications. 2020. Vol. 176. No. 15. P. 10–17. DOI 10.5120/ijca2020920014

## References

1. **Yasin W., Ibrahim H., Abdul Hamid N. A. W., Udzir N. I.** Performance Analysis of Transport Control Protocol Flavours in the Existence of Packet Reordering Phenomena. *Digital Enterprise and Information System*s, 2011, pp. 569–579. DOI: 10.1007/978-3-642-22603-8_50
2. **Olifer V. G., Olifer N. A.** Computer Networks: Principles, Technologies and Protocols, 3rd Edition, Saint Petersburg, Piter publ., 2007.
3. **Hopps C.** Analysis of an Equal-Cost Multi-Path Algorithm. *RFC Editor*, Nov. 2000. DOI: 10.17487/rfc2992.
4. **Larin D. V., Getman A. I.** High-speed network traffic capturing and processing tools. *Proceedings of the Institute for System Programming of the RAS*, 2021, vol. 33, no. 4, pp. 49–68. DOI: 10.15514/ispras-2021-33(4)-4. (in Russ.)
5. **Karger D., Lehman E., Leighton T., Panigrahy R., Levine M., Lewin D.** Consistent hashing and random trees. *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing – STOC '97*, 1997, pp. 654–663. DOI: 10.1145/258533.258660
6. **Thaler D. G., Ravishankar C. V.** Using name-based mappings to increase hit rates. *IEEE/ACM Transactions on Networking*, 1998, vol. 6, no. 1, p. 1–14. DOI: 10.1109/90.663936
7. **Eisenbud D. E., Yi C., Contavalli C., Smith C., Kononov R., Mann-Hielscher E., et al.** Maglev: a fast and reliable software network load balancer. *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI'16)*, 2016, Usenix Association, pp. 523–535.
8. **Appleby A.** Murmurhash3. Github. URL: https://github.com/aappleby/smhasher/wiki/MurmurHash (date of access: 17.04.2025)
9. **Suparn P., Mamta R.** Evaluation and Categorization of Hashing Algorithms Based on Their Applications. *IAENG International Journal of Applied Mathematics*, 2025, no. 55, pp. 540–552.
10. **Akoto-Adjepong V., Asante M., Okyere-Gyamfi S.** An Enhanced Non-Cryptographic Hash Function. *International Journal of Computer Applications*, 2020, vol. 176, no. 15, pp. 10–17, DOI: 10.5120/ijca2020920014

## Информация об авторе

**Красовская Надежда Валерьевна,** студентка магистратуры

## Information about the Author

Nadezhda V. Krasovskaya, Master's Student

*Статья поступила в редакцию 30.04.2025;*
*одобрена после рецензирования 27.05.2025; принята к публикации 27.05.2025*

*The article was submitted 30.04.2025;*
*approved after reviewing 27.05.2025; accepted for publication 27.05.2025*