УДК 004.75 **DOI: 10.31799/2077-5687-2026-2-97-105**

МАРШРУТИЗАЦИЯ ЗАПРОСОВ С УЧЁТОМ ЗАВИСИМОСТЕЙ В МИКРОСЕРВИСНЫХ СИСТЕМАХ: ИМИТАЦИОННАЯ МОДЕЛЬ И МНОГОКРИТЕРИАЛЬНАЯ ОЦЕНКА

М. В. Русанов
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Разработана и исследована дискретно-временная имитационная модель для сравнительной оценки алгоритмов маршрутизации запросов в микросервисных топологиях с явными зависимостями от баз данных, кешей, брокеров, внешних API и нижележащих сервисов. Описаны модель топологии, модель задержек, сценарии деградации, механизм запаздывающей наблюдаемости и семейство алгоритмов Score-D, использующее признаки локальной конкурентности, сетевой стоимости, доли ошибок, давления зависимостей и давления совместного размещения. Проведена экспериментальная оценка на двух наборах данных: базовом полнофакторном эксперименте и расширенном эксперименте с ростом числа логических сервисов и реплик. Показано, что Score-D не является универсально лучшим по каждой отдельной метрике, однако в сценариях локализованной деградации zone-burst и partial-failure формирует более устойчивый Парето-компромисс между долей ошибок, p95-задержкой и относительной пропускной способностью.

Ключевые слова: микросервисная архитектура, маршрутизация запросов, балансировка нагрузки, дискретно-событийное моделирование, хвостовая задержка, отказоустойчивость, наблюдаемость, обнаружение сервисов, Парето-анализ, многокритериальная оптимизация.

Для цитирования:
Русанов, М. В. Маршрутизация запросов с учётом зависимостей в микросервисных системах: имитационная модель и многокритериальная оценка / М. В. Русанов // Системный анализ и логистика. – 2026. – № 2(50). – с. 97-105. DOI: 10.31799/2077-5687-2026-2-97-105.

DEPENDENCY-AWARE REQUEST ROUTING IN MICROSERVICE SYSTEMS: SIMULATION MODEL AND MULTI-CRITERIA EVALUATION

M. V. Rusanov
St. Petersburg State University of Aerospace Instrumentation

A discrete-time simulation model for comparative evaluation of request routing algorithms in microservice topologies with explicit downstream dependencies is developed and evaluated. The model includes service instances, databases, caches, message brokers, external APIs, service-to-service chains, delayed noisy observability, and controlled degradation scenarios. The Score-D family of algorithms combines local concurrency, network cost, error rate, downstream pressure, and host co-location pressure into an interpretable weighted score. The experiments include a baseline full-factorial dataset and an extended dataset with different topology sizes. The results show that Score-D does not dominate all baselines in every metric, but provides a more stable Pareto trade-off between error rate, p95 latency, and relative throughput under localized degradation scenarios such as zone-burst and partial-failure.

Keywords: microservice architecture, request routing, load balancing, discrete-event simulation, tail latency, fault tolerance, observability, service discovery, Pareto analysis, multi-objective optimization.

For citation:
Rusanov, M. V. Dependency-aware request routing in microservice systems: simulation model and multi-criteria evaluation / M. V. Rusanov // System analysis and logistics. – 2026. – № 2(50). – p. 97-105. DOI: 10.31799/2077-5687-2026-2-97-105.

Введение

В микросервисных системах [1] выбор экземпляра сервиса определяет не только сетевую задержку до выбранной реплики, но и набор нижележащих ресурсов, которые будут задействованы при обработке запроса. На практике разные реплики одного логического сервиса могут иметь различную близость к кешам, базам данных, брокерам сообщений, внешним API или другим сервисам [2]. Поэтому при асимметричной деградации простые алгоритмы *random, round-robin и least-inflight*, использующие преимущественно локальные признаки, могут хуже различать скрыто неодинаковые пути к зависимостям [3, 4, 5].

Исследование находится на пересечении балансировки нагрузки, адаптивного выбора реплик, анализа хвостовых задержек, наблюдаемости распределенных систем и имитационного моделирования. Современные L7-балансировщики и service mesh-платформы поддерживают различные стратегии выбора upstream-реплик, включая round-robin, least request, locality-aware routing и outlier detection [6, 7]. Близкая линия работ по tail latency показывает, что редкие медленные компоненты могут определять сквозную задержку распределенной системы, а адаптивный выбор реплики способен снижать хвостовые задержки [8, 9, 10]. Однако эти подходы обычно не моделируют явно структуру зависимостей выбранной реплики вниз по графу.

В данной работе предложена экспериментальная модель, в которой маршрутизатор принимает решение на основе запаздывающей и зашумленной телеметрии, а не идеального внутреннего состояния симулятора [11, 12, 13]. Это необходимо для защиты от круговой оценки: алгоритм не должен получать недоступную в реальной системе информацию. Методологически работа относится к имитационному моделированию сетевых и программных систем под высокой нагрузкой [14, 15, 16, 17]. Оценка выполняется как многокритериальная задача, поскольку минимальная p95-задержка по завершенным запросам может сопровождаться ростом доли ошибок и снижением пропускной способности [18, 19, 20].

Цель работы – проверить, улучшает ли маршрутизация Score-D, учитывающая зависимости, устойчивость относительно базовых алгоритмов при асимметричных деградациях [21, 22], сохраняется ли эффект при росте топологии и какие компромиссы возникают между долей ошибок, p95-задержкой и относительной пропускной способностью.

1. Модель микросервисной топологии

Сеть представляется ориентированным графом $G=(V,E)$. Вершины V соответствуют шлюзам, экземплярам сервисов, базам данных, кешам, брокерам, внешним API и клиентам. Каждая дуга $e=(u\to v)$ имеет задержку линии $L_e>0$, пропускную способность $C_e>0$ и абстрактную стоимость $K_e\geq 0$. Для пары вершин используется кратчайший путь по суммарной задержке [23]:

$$P^*(a,b)=\arg\min_{P:a\to b}\sum_{e\in P}L_e. \tag{1}$$

Агрегированные задержка и стоимость пути определяются как

$$L(a,b)=\sum_{e\in P^*(a,b)}L_e,\qquad K(a,b)=\sum_{e\in P^*(a,b)}K_e. \tag{2}$$

Множество логических сервисов обозначается S, а множество экземпляров сервиса s – $I(s)\subseteq V$. Каждый запрос r направлен в логический сервис $s(r)$ и должен быть маршрутизирован в один экземпляр $i(r)\in I(s(r))$. Для моделирования размещения у вершин задаются зона $z(v)$ и хост $h(v)$; эти метки используются при генерации сетевых задержек и при вычислении давления совместного размещения.

Полная задержка запроса задается суммой сетевого пути, локальной обработки и зависимостей:

$$\text{latency}_{\text{ms}}(r)=L(g,i(r))+\ell_{\text{svc}}(i(r))+\ell_{\text{down}}(r,i(r)), \tag{3}$$

где g – шлюз. Для зависимости d с базовой задержкой b_d. , вероятностью использования p_d и режимом $m_d \in \{\text{sync}, \text{async}, \text{fire}\}$ используется коэффициент $\alpha(m_d)$: 1,0 для синхронного вызова, 0,25 для асинхронного и 0,10 для fire-and-forget. Вклад конкретной цели x вычисляется как

$$\ell(i,d,x) = p_d \alpha(m_d)\big(b_d + L(i,x) + \beta(x,t)\big). \tag{4}$$

Штраф давления зависимости определяется через нормированную загрузку и ресурсные признаки:

$$\beta(x,t) = 10\big(u(x,t) + p_{\text{db}}(x,t) + p_{\text{cache}}(x,t) + p_{\text{broker}}(x,t)\big). \tag{5}$$

Для каждой вершины поддерживается число активных запросов $q(v,t)$ и номинальная емкость $\text{cap}(v)$. Насыщение и давление совместного размещения задаются выражениями

$$u(v,t) = \frac{q(v,t)}{\text{cap}(v)}, \qquad \text{host\_pressure}(v,t) = \frac{\sum_{x:h(x)=h(v)} q(x,t)}{\text{cap}(v)}. \tag{6}$$

Такая модель не претендует на точное предсказание абсолютных задержек конкретной промышленной системы. Ее назначение – контролируемое сравнение алгоритмов при одинаковых допущениях о сети, зависимостях и деградациях.

2. Наблюдаемость и алгоритмы маршрутизации

Для защиты от нереалистичного преимущества алгоритмы маршрутизации видят не истинные внутренние переменные $x(t)$, а запаздывающую и зашумленную оценку:

$$\tilde{x}(t) = x(t-\Delta) + \eta, \qquad \eta \sim \text{Uniform}(-\varepsilon, \varepsilon). \tag{7}$$

Кандидаты маршрутизации для запроса образуют множество $C = I(s(r))$. В качестве базовых алгоритмов используются *random, round-robin и least-inflight. В алгоритме least-inflight* выбирается кандидат с минимальной наблюдаемой локальной конкурентностью:

$$i(r) = \arg\min_{c \in C} \tilde{q}(c,t). \tag{8}$$

Предложенное семейство Score-D выбирает кандидата с минимальной взвешенной оценкой:

$$\text{score}(c) = \sum_{j=1}^{J} w_j f_j(c), \qquad i(r) = \arg\min_{c \in C} \text{score}(c). \tag{9}$$

В названии Score-D буква D обозначает зависимости (dependencies). Базовый вариант использует признаки сетевой задержки f_{lat}, локальной конкурентности f_{inflight}, доли ошибок f_{err}, сетевой стоимости f_{netcost}, давления зависимостей f_{down} и давления совместного размещения f_{host}:

$$\text{ScoreD} = 0.30\, f_{\text{lat}} + 0.25\, f_{\text{inflight}} + 0.15\, f_{\text{err}} + 0.10\, f_{\text{netcost}} + 0.15\, f_{\text{down}} + 0.05\, f_{\text{host}}. \quad (10)$$

Веса выбраны априорно (см. табл. 1) как фиксированная исследовательская гипотеза и не оптимизировались по финальному датасету. Поэтому они интерпретируются как один из вариантов маршрутизации с учетом зависимостей, а не как найденный оптимум.

Таблица 1 – Варианты весов Score-D

Вариант	f_lat	f_inflight	f_err	f_netcost	f_down	f_host
Score-D	0,30	0,25	0,15	0,10	0,15	0,05
local-only	0,00	0,55	0,35	0,00	0,00	0,10
local+net	0,30	0,35	0,20	0,10	0,00	0,05
local+down	0,20	0,30	0,15	0,00	0,30	0,05
no-down	0,35	0,30	0,20	0,10	0,00	0,05
no-host	0,30	0,25	0,15	0,10	0,20	0,00

3. Методика экспериментов и метрики

Эксперимент задается числом шагов T, фазой прогрева T_w, фазой дренажа T_d, интенсивностью генерации λ и начальным значением генератора случайных чисел. Новые запросы создаются до начала дренажа, а итоговая статистика учитывает только завершения и ошибки в измеряемом окне $[T_w, T - T_d)$. Такой протокол уменьшает влияние переходных процессов и исключает искажение хвостовых задержек незавершенными запросами [14, 15, 24].

Для одного прогона фиксируются число созданных, завершенных и ошибочных запросов, средняя задержка, p95/p99, число активных запросов в конце, насыщение узлов, давление хостов, всплесковость отказов и пропускная способность. Относительная пропускная способность нормируется на предложенную нагрузку:

$$\text{throughput\_ratio} = \frac{\text{throughput\_per\_step}}{\lambda}. \quad (11)$$

Для каждой группы экспериментов вычисляются среднее значение, выборочное стандартное отклонение и 95% доверительный интервал:

$$\text{CI95} = 1.96 \frac{s}{\sqrt{n}}. \quad (12)$$

Основная оценка является многокритериальной. Алгоритмы сравниваются по трем целям:

$$\text{minerror\_rate}, \quad \text{minp95\_latency}, \quad \text{maxthroughput\_ratio}.$$

Алгоритм A доминирует алгоритм B, если он не хуже по всем трем целям и строго лучше хотя бы по одной. Алгоритм находится на Парето-фронте, если нет другого алгоритма, который его доминирует. Это важно, поскольку минимальная p95-задержка по завершенным запросам не гарантирует лучшей надежности.

Проведены два набора экспериментов. Базовый датасет включает 64 800 экспериментов:

4 топологии, 9 алгоритмов, 6 сценариев, 6 уровней нагрузки и 50 начальных значений генератора. В этом датасете варианты Score-D агрегированы под техническим идентификатором *score-v1*; поэтому результат интерпретируется как сравнение семейства Score-D с базовыми алгоритмами. Расширенный датасет включает 54 000 экспериментов: 4 топологии, 3 размера по числу логических сервисов, 2 размера по числу реплик, 5 алгоритмов, 3 сценария, 3 уровня нагрузки и 50 начальных значений генератора. Он используется для проверки устойчивости результатов при росте топологии и для сравнения вариантов *score-local+network и score-no-host-pressure.*

4. Результаты экспериментов

Результаты показывают, что Score-D не является универсально лучшим по каждой метрике, но формирует более устойчивый компромисс между долей ошибок, p95-задержкой и относительной пропускной способностью в сценариях локализованной деградации.

Таблица 2 – Макроусредненные результаты базового датасета

Алгоритм	Доля ошибок	P95 задержки, мс	Относительная пропускная способность
семейство Score-D	0,02699	44,92	0,9730
round-robin	0,08314	46,18	0,9169
random	0,08832	46,15	0,9117
least-inflight	0,10810	44,35	0,8919

Из табл. 2 видно, что *least-inflight* имеет наименьшую p95-задержку, но одновременно худшую долю ошибок и относительную пропускную способность. Следовательно, вывод “минимальная p95-задержка = лучший алгоритм” для данной задачи некорректен.

Таблица 3 – Макроусредненные результаты расширенного датасета

Алгоритм	Доля ошибок	P95 задержки, мс	Относительная пропускная способность
score-local+network	0,02205	50,20	0,9779
score-no-host-pressure	0,02387	49,37	0,9761
round-robin	0,13650	51,59	0,8635
random	0,15460	51,52	0,8454
least-inflight	0,19260	50,31	0,8074

В расширенном датасете (см. табл. 3) варианты Score-D образуют лучшую группу. *score-local+network* чаще выигрывает по доле ошибок и относительной пропускной способности, а *score-no-host-pressure* чаще выигрывает по p95-задержке. Это подтверждает, что веса оценочной функции выражают целевую функцию системы, а не единственный правильный выбор.

Наиболее сильный эффект наблюдается в сценариях *partial-failure и zone-burst*. По сравнению с *random*, в расширенном датасете *score-local+network* снижает долю ошибок примерно на 88,9% в *partial-failure* и примерно на 89,4% в *zone-burst*, одновременно повышая относительную пропускную способность. На рис. 1 и 2 показаны диаграммы Парето для этих сценариев: по оси X отложена доля ошибок, по оси Y – p95-задержка, а размер точки отражает относительную пропускную способность.

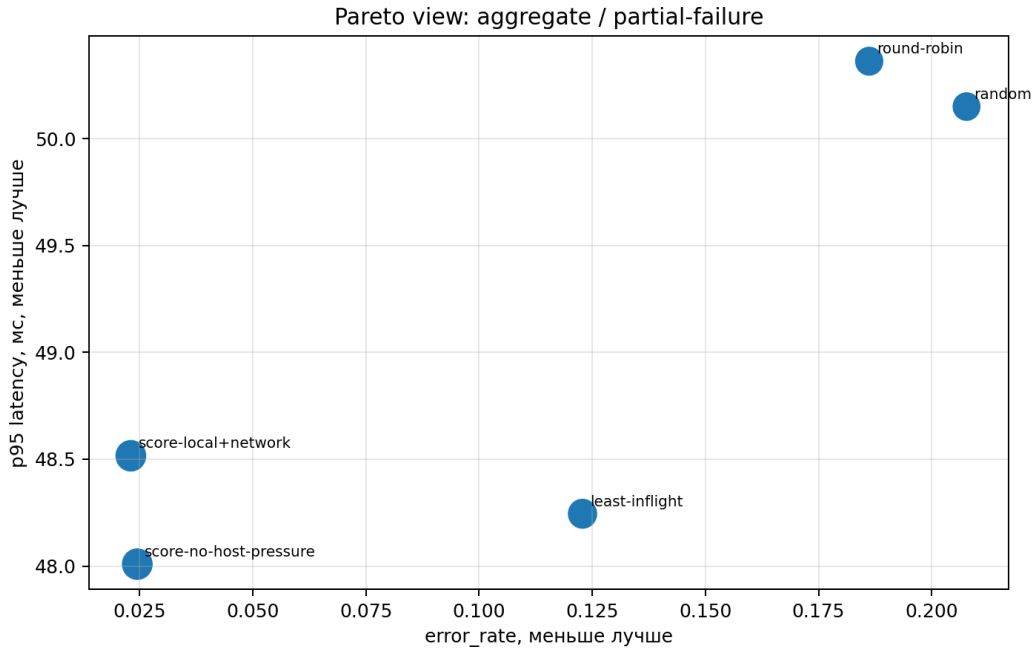

Рис. 1. Диаграмма Парето для сценария partial-failure.

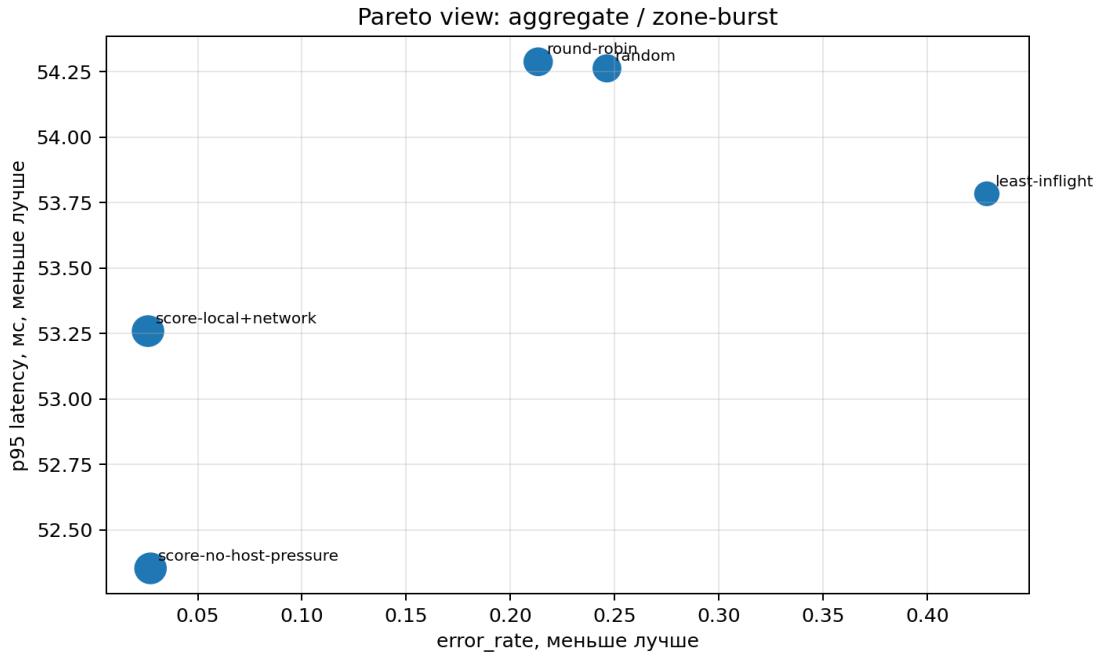

Рис. 2. Диаграмма Парето для сценария zone-burst.

Положение точек на рис. 1 и 2 демонстрирует, что варианты Score-D находятся в области меньшей доли ошибок и меньшей или сопоставимой p95-задержки по сравнению с базовыми алгоритмами. При этом *score-no-host-pressure* чаще дает меньшую p95-задержку, тогда как *score-local+network* чаще улучшает надежность и относительную пропускную способность.

5. Воспроизводимость и ограничения

Исходный код исследовательского стенда, параметры запуска, скрипты анализа и агрегированные результаты опубликованы в репозитории Micro-Net-Lab: https://github.com/rusanoph/Micro-Net-Lab. Публикационная ревизия зафиксирована в commit *ea288fd*. В каждом прогоне сохраняются *git_commit*, команда запуска, версия компилятора Rust, сведения о процессоре, параметры шардирования и диапазон начальных значений

генератора. Это позволяет связать экспериментальные артефакты с конкретной версией симулятора и скриптов анализа.

Ограничения работы следующие. Во-первых, модель давления синтетическая и низкоразмерная; абсолютные значения задержек не следует интерпретировать как прогноз для конкретной промышленной системы. Во-вторых, нагрузка в основных прогонах постоянна по шагам дискретного времени; всплесковые и суточные профили пока не исследованы. В-третьих, сеть моделируется на уровне сервисного графа без TCP, повторных передач и полного управляющего слоя service mesh. В-четвертых, в текущей версии не рассматриваются некоторые промышленные baseline-алгоритмы, включая power-of-two-choices, EWMA latency, locality-aware routing и outlier detection. Эти ограничения не отменяют сравнительного результата, но задают направления дальнейшей работы.

Заключение

В работе представлена дискретно-временная имитационная модель для исследования алгоритмов маршрутизации в микросервисных топологиях с зависимостями, неполной наблюдаемостью и контролируемыми сценариями деградации. Экспериментальная оценка показала, что семейство Score-D, учитывающее зависимости, не является универсальным доминатором по всем метрикам, однако улучшает Парето-компромисс между долей ошибок, p95-задержкой и относительной пропускной способностью в локализованных сценариях отказов. Практический вывод состоит в том, что маршрутизацию в таких системах следует рассматривать как многокритериальную задачу: ранжирование только по p95-задержке может приводить к неверной интерпретации, особенно для алгоритма *least-inflight*, который способен снижать p95 ценой худшей надежности.

Приложение А. Артефакты воспроизводимости

Для воспроизведения результатов используются следующие артефакты репозитория: *bench-results/vm/micro-net-paper-001/aggregate.csv, bench-results/vm/scaling-001/aggregate.csv, micro_net_analysis_publication.py, analysis-out/publication_tables.md, analysis-out/ci95_appendix_contexts.csv, analysis-out/scenario_focus_summary.csv, analysis-out/scenario_focus_best_score_vs_baseline.csv, analysis-out/pareto_counts.csv, analysis-out/metric_winner_counts.csv* и каталог *analysis-out/figures/*. Доверительные интервалы CI95 вынесены в машинно-читаемые таблицы приложения, чтобы не перегружать основные таблицы статьи.

СПИСОК ЛИТЕРАТУРЫ

1. *Lewis J.* Microservices: a definition of this new architectural term [Электронный ресурс] / J. Lewis, M. Fowler. // Martinfowler. 2014. 25 March. – URL: https://martinfowler.com/articles/ microservices.html (дата обращения: 20.05.2026).
2. *Карпович М. Н.* Особенности проектирования микросервисно-событийных архитектур для высоконагруженных распределенных систем обработки информаций // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. – 2023. – № 1 (266). – С. 89–95. DOI: 10.52065/2520-6141-2023-266-1-15.
3. *Ляшов Е. И.* Ресурсоэффективные алгоритмы балансировки нагрузки в распределенных микросервисных архитектурах // Вестник науки. – 2025. – Т. 1, № 2 (83). – С. 629–647.
4. *Елагин В. С.* Функциональное назначение балансировщика нагрузки в облачных микросервисных архитектурах / В. С. Елагин, В. Ф. Николаев // Информационные технологии и телекоммуникации. – 2020. – Т. 8, № 1. – С. 67–75. DOI: 10.31854/2307-1303-2020-8-1-67-75.
5. *Меркулов А. С.* Эффективная интеграция Service Discovery и балансировки нагрузки: принципы, инструменты и практические решения // Актуальные исследования. –

2025. – № 12 (247), ч. I. – С. 16–19.
6. Envoy Proxy. Supported load balancers [Электронный ресурс]: документация проекта. – URL: https://www.envoyproxy.io/docs/envoy/latest/intro/arch_overview/upstream/load_balancing/load_balancers (дата обращения: 20.05.2026).
7. Istio. Destination Rule: OutlierDetection and load balancing settings [Электронный ресурс]: документация проекта. – URL: https://istio.io/latest/docs/reference/config/networking/destination-rule/ (дата обращения: 20.05.2026).
8. *Dean J.* The Tail at Scale / J. Dean, L. A. Barroso // Communications of the ACM. – 2013. – Vol. 56, No. 2. – P. 74–80. DOI: 10.1145/2408776.2408794.
9. *Suresh L.* C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection / L. Suresh, M. Canini, S. Schmid, A. Feldmann // 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). Oakland, 2015. – URL: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/suresh (дата обращения: 20.05.2026).
10. *Wydrowski B.* Load is not what you should balance: Introducing Prequal / B. Wydrowski, R. Kleinberg, S. M. Rumble, A. Archer // 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). Santa Clara, 2024. – URL: https://www.usenix.org/conference/nsdi24/presentation/wydrowski (дата обращения: 20.05.2026).
11. *Максимов В. Ю.* Преодоление трудностей наблюдаемости в микросервисной архитектуре // Инновационная наука. – 2024. – № 2-1. – С. 30–35.
12. *Ростоцкий М. В.* Наблюдаемость (logs/metrics/traces) как часть проектирования ИС: SLI/SLO и снижение MTTR // Инженерный вестник. 2025. – URL: https://iereview.ru/index.php/IE/article/view/233 (дата обращения: 20.05.2026).
13. *Sigelman B. H.* Dapper, a Large-Scale Distributed Systems Tracing Infrastructure / B. H. Sigelman [et al.]. – Google Technical Report, 2010. – URL: https://research.google.com/pubs/pub36356.html (дата обращения: 20.05.2026).
14. *Чубейко С. В.* Дискретно-событийное моделирование сетевых компьютерных систем в условиях высоких нагрузок // Фундаментальные исследования. – 2014. – № 8-2. – С. 340–344.
15. Оценка надежности программного обеспечения методами дискретно-событийного моделирования / М. А. Бутакова [и др.] // Программные продукты и системы. – 2015. – № 4. – С. 158–165. DOI: 10.15827/0236-235X.112.158-165.
16. *Тымченко Н. С.* Проектирование СМО для оценки времени обслуживания // Материалы научной конференции. – 2025. – URL: https://mtmt.etu.ru/assets/files/2025/sbornik/07-11.pdf (дата обращения: 20.05.2026).
17. *Fujiwara K.* Speed and Accuracy of Network Simulation in the SimGrid Framework / K. Fujiwara, H. Casanova // Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools. – 2007. – 10 с.
18. *Карпенко А. П.* Модифицированный метод адаптивных взвешенных сумм в задаче многокритериальной оптимизации // Наука и образование: научное издание МГТУ им. Н. Э. Баумана. – 2013. – URL: https://cyberleninka.ru/article/n/modifitsirovannyy-metod-adaptivnyh-vzveshennyh-summ-v-zadache-mnogokriterialnoy-optimizatsii (дата обращения: 20.05.2026).
19. Университет ИТМО. Задача многокритериальной оптимизации. Multiobjectivization [Электронный ресурс] // Викиконспекты ИТМО. – URL: https://neerc.ifmo.ru/wiki/index.php?title=Задача_многокритериальной_оптимизации._Multiobjectivization (дата обращения: 20.05.2026).
20. *Курбацкий А. Н.* Лекция 5. Доверительные интервалы [Электронный ресурс] // МШЭ МГУ, 2020. – URL: https://mse.msu.ru/wp-content/uploads/2020/03/Лекция-5-

доверительные-интервалы.pdf (дата обращения: 20.05.2026).
21. *Бачурин Д. Ю.* Использование хаос-инжиниринга в системе оркестрации контейнеров Kubernetes // Вестник науки. – 2025. – № 5(86). – с. 556-560.
22. *Basiri A.* Chaos Engineering / A. Basiri [et al.]. – arXiv:1702.05843, 2017. – URL: https://arxiv.org/abs/1702.05843 (дата обращения: 20.05.2026).
23. *Кальметьев Э. И.* Анализ алгоритмов кратчайшего пути: сравнительный обзор методов маршрутизации / Э. И. Кальметьев, А. Е. Сергеева // Современные наукоемкие технологии. 2025. – URL: https://s.eduherald.ru/pdf/2025/3/21871.pdf (дата обращения: 20.05.2026).
24. *Голяндина Н. Э.* Моделирование, метод Монте-Карло. Короткий обзор [Электронный ресурс] // Санкт-Петербургский государственный университет. 2020. 1 декабря. – URL: https://statmod.ru/wiki/_media/study%3Afall2020%3Aprobmodel%3Amontecarlo2020.pdf (дата обращения: 20.05.2026).

ИНФОРМАЦИЯ ОБ АВТОРЕ

Русанов Максим Витальевич
Аспирант
Санкт-Петербургский государственный университет аэрокосмического приборостроения
Россия, 190000, Санкт-Петербург, ул. Большая Морская, д.67, лит. А
E-mail: hatrue.max@gmail.com

INFORMATION ABOUT THE AUTHOR

Rusanov Maxim Vitalievich
Postgraduate student
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: hatrue.max@gmail.com

Дата поступления: 28.05.2026
Дата принятия: 04.06.2026