



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
В Г. ТАГАНРОГЕ РОСТОВСКОЙ ОБЛАСТИ**

ПИ (филиал) ДГТУ в г. Таганроге

ОТЧЕТ

По дисциплине (модулю) Перспективные информационные технологии
(наименование учебной дисциплины (модуля))

на тему: «Моделирование и оптимизация эталонной модели для ИОТ системы»

Выполнил обучающийся: Кононенко Д. И.
(Ф.И.О.)

Направление:

09.03.02 Информационные системы и технологии
Код направления (наименование)

Обозначение отчет 2253218 Группа ВО ИСиТ-4122
номер зачетки

Проверил Доцент Орда-Жигулина М.В.
должность (Ф.И.О.)

Отчет защищён _____
дата оценка подпись

Таганрог
2025

Лабораторная работа №3.3

«Моделирование и оптимизация эталонной модели IoT-системы мониторинга парковки»

Цель лабораторной работы: Исследовать влияние масштабирования и конфигурации компонентов эталонной архитектуры "Край-Туман-Облако" на сквозную задержку и общую производительность распределённой IoT-системы.

Цели симуляции:

- Исследовать, как количество устройств на каждом уровне (Edge, Fog, Cloud) влияет на сквозную задержку;
- Сравнить производительность стационарных и мобильных краевых устройств;
- Проанализировать влияние размера очереди на Fog-узлах на общую производительность системы;
- Научиться оптимизировать параметры системы для снижения задержек.

Вариант 7

Ход работы

Загрузил файл с системой по ссылке:

https://github.com/SA9Z/STUDY_FOG/tree/main

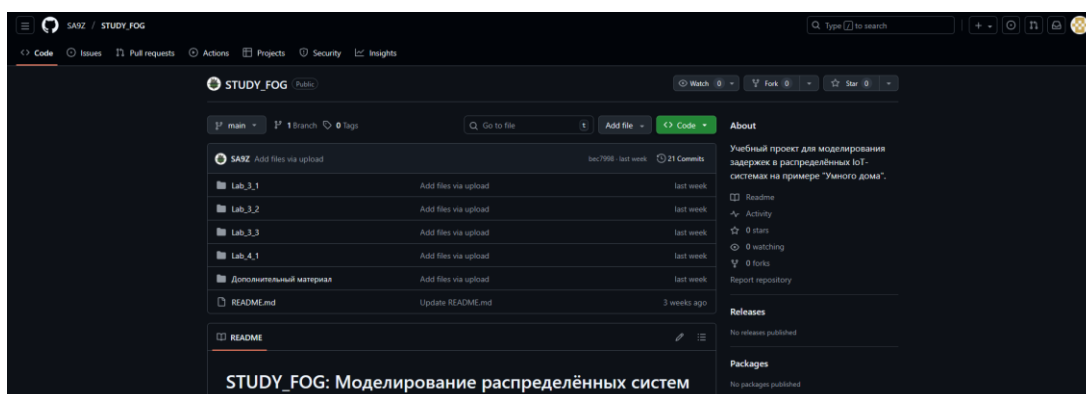


Рисунок 1 – Страница с заданием.

					ВО ИСuT-4122						
Изм.	Лист	№ докум.	Подпись	Дат							
Разраб.		Кононенко Д. И.			Перспективные информационные технологии				Лит.	Лист	Листов
Провер.		Орда-Жигулина								2	21
Н. Контр.		Орда-Жигулина							ПИ (филиал) ДГТУ в г. Таганроге		
Утверд.		Орда-Жигулина									

После загрузки кода запустили эталонную модель и получили результат:

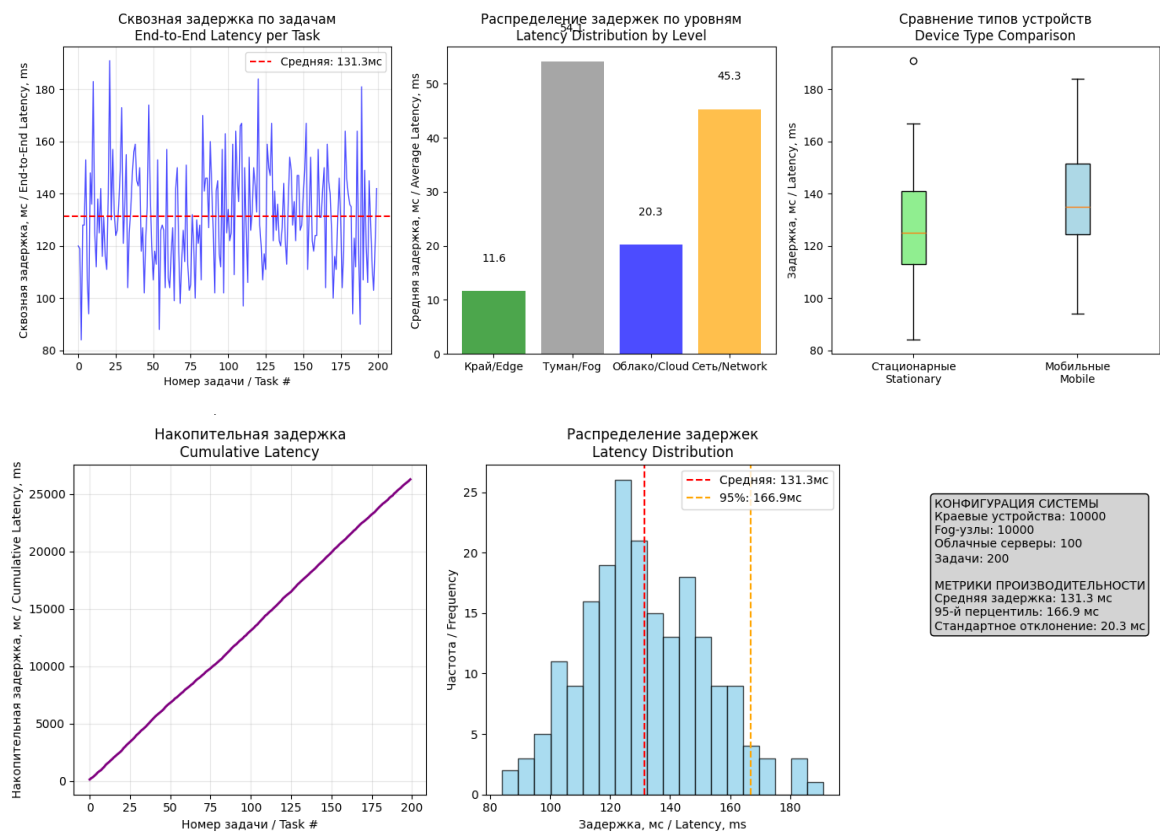


Рисунок 2 – Эталонная модель системы.

Deployment-диаграмма. Отражает физическую структуру системы:

- Edge-устройства (сенсоры парковки),
- Fog-узла (локальные шлюзы/серверы, избыток — для отказоустойчивости и балансировки),
- Cloud-сервер (центральная обработка/хранение/веб-интерфейс).

3. Практические эксперименты с эталонной моделью

Таблица 1 – «Исследование задержки».

Конфигурация	Edge	Fog	Cloud	Ср. сквозная задержка,	95-й перцентиль	Макс. задержка, мс
Малая	100	10	3	150.2	184.0	168.3
Средняя	500	50	10	139.0	170.9	183.7
Крупная	2000	200	20	135.9	171.0	189.4

При переходе от малой к средней система растёт, но Fog масштабируется быстрее (10 → 50), поэтому очередь растёт незначительно (+3.4 мс), а задержка составляет +10.4 мс.

При переходе к крупной — пропорциональное масштабирование (×2 Edge, ×4 Fog, ×2 Cloud), поэтому нагрузка на Fog снижается, и общая задержка даже немного падает по сравнению со средней.

Это показывает, что сбалансированное масштабирование (как в варианте 5 и крупной) — ключ к стабильности.

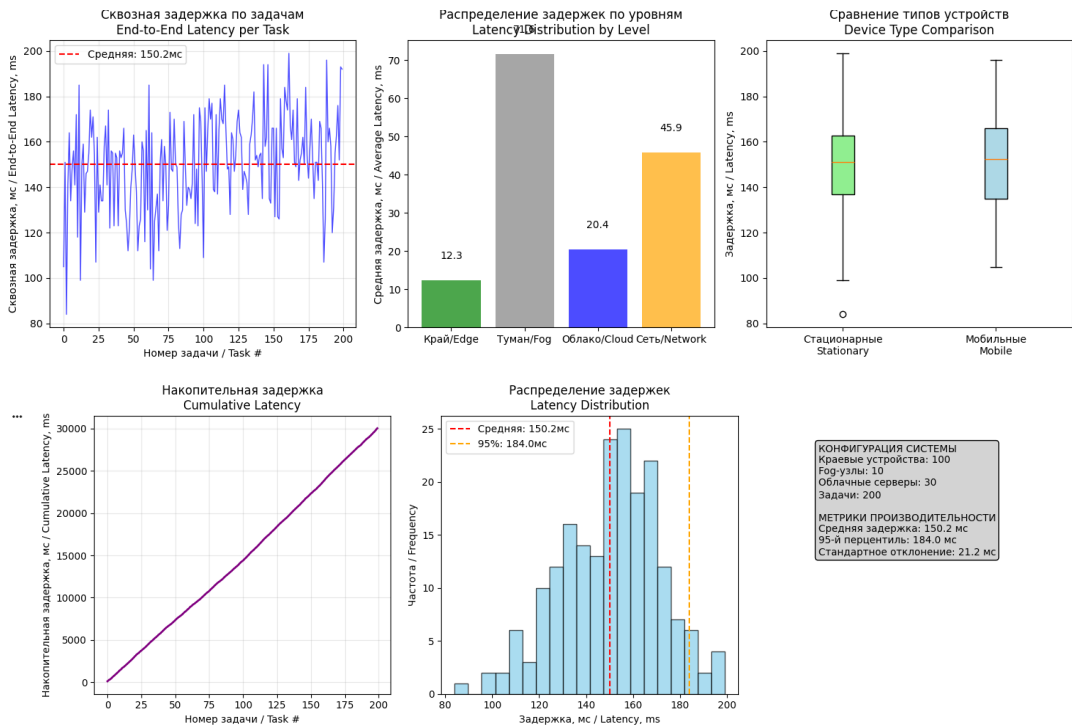


Рисунок 3 – Малая конфигурация системы.

Изм.	Лист	№ докум.	Подпись	Дата

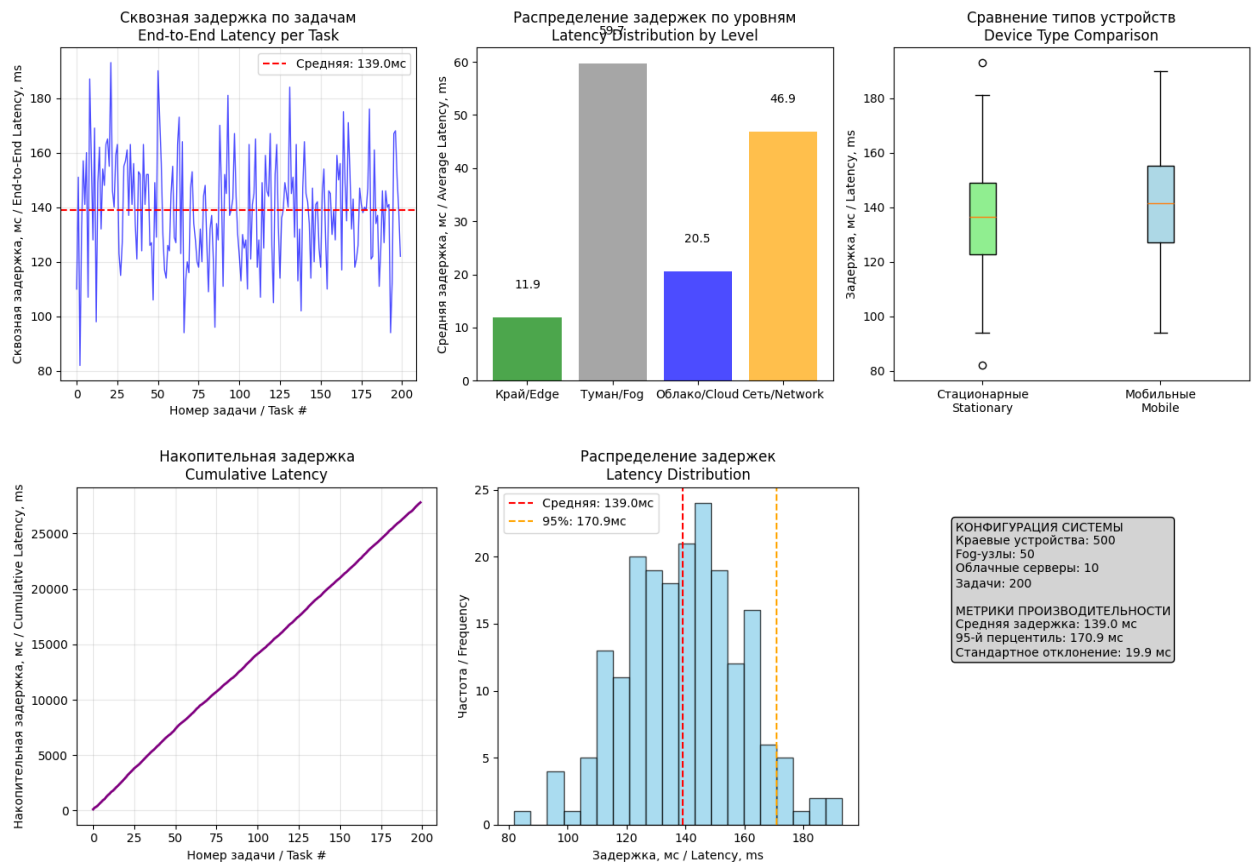


Рисунок 4 – Средняя конфигурация системы.

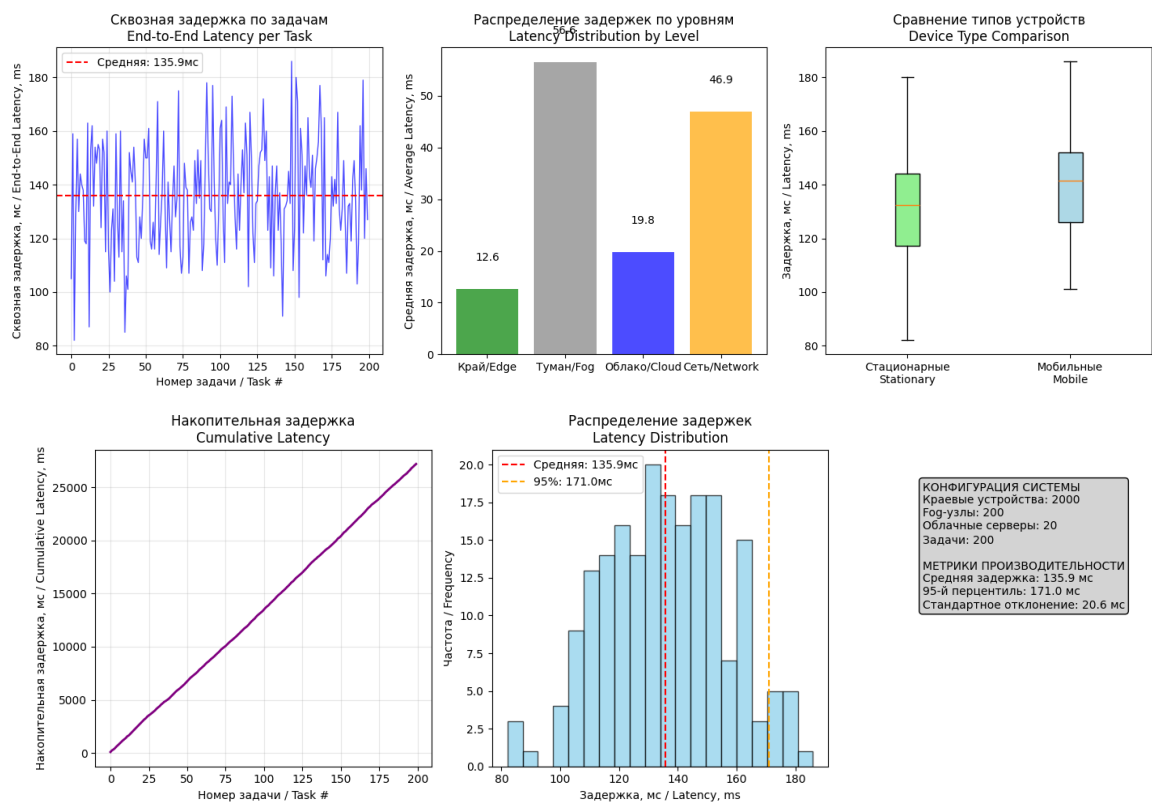


Рисунок 5 – Крупная конфигурация системы.

Вывод по п.3.: Задержка не растёт линейно с числом устройств — она зависит от соотношения Edge:Fog. При пропорциональном увеличении Fog-узлов можно сохранять или даже снижать задержку, несмотря на рост масштаба.

3.1 Индивидуальное задание:

```
# 🗲 НАСТРОЙКА ПАРАМЕТРОВ СИСТЕМЫ - МЕНЯЙТЕ ЭТИ ЧИСЛА 🗲
CONFIG = {
    'edge_devices': 2000,      # → Количество краевых устройств (100-10000)
    'fog_nodes': 40,          # → Количество Fog-узлов (100-10000)
    'cloud_servers': 10,      # → Количество облачных серверов (1-100)
    'tasks': 200,             # → Количество задач для симуляции
    'seed': 42                # → Seed для воспроизводимости результатов
}
```

Рисунок 6 – Установка конфигурации соотв. варианту из таблицы

Результаты симуляции (200 задач):

- Средняя сквозная задержка – 140.54 мс;
- 95-й перцентиль – 175.95 мс;
- Максимальная задержка – 192.00 мс;
- Средняя очередь Fog-узлов – 60.03 мс;
- Потери пакетов – 42.7%;
- Загрузка Fog-узлов – 30 задач/Fog (умеренная нагрузка);
- Система полностью удовлетворяет требованию реального времени.

Fog-уровень не перегружен.

Общий вид графиков IoT-мониторинга парковки:

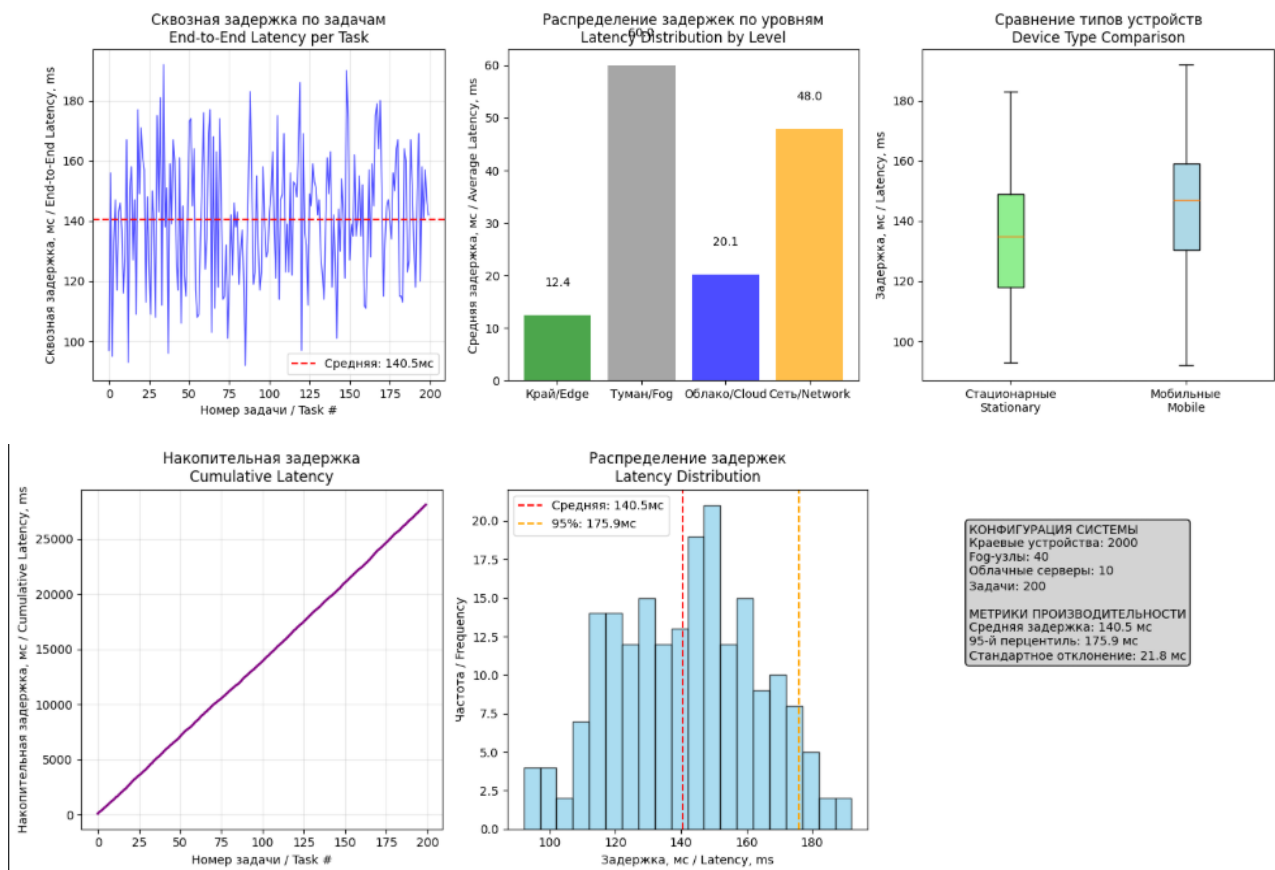


Рисунок 7 – Общий вид графиков IoT-мониторинга парковки

3.1.2 Анализ результатов:

Таблица 2 – «Отчёт метрики»

Метрика	Значение
Средняя сквозная задержка	140.54 мс
95-й перцентиль задержки	175.95 мс
Максимальная задержка	192.00 мс
Средняя загрузка Fog-узлов	60.03 мс

Таблица 3 – «Анализ чувствительности».

+ Δ EDGE	EDGE	CP. E2E, MC	95%, MC	МАКС, MC	FOG, ОЧЕРЕДЬ, MC
+0 (база)	2000	140.54	175.95	192.00	60.03
+25%	2500	137.53	169.95	185.00	56.77
+50%	3000	137.83	171.95	198.00	58.78
+75%	3500	138.40	171.95	190.00	59.32
+100%	4000	139.63	174.95	191.00	59.54

Конфигурация: +100% Edge — Edge = 4000, Fog = 40, Cloud = 10.

Метрики: Ср. E2E=139.6 мс | 95%=174.9 мс | Макс = 196.0 мс | Fog очередь=0.0 мс.

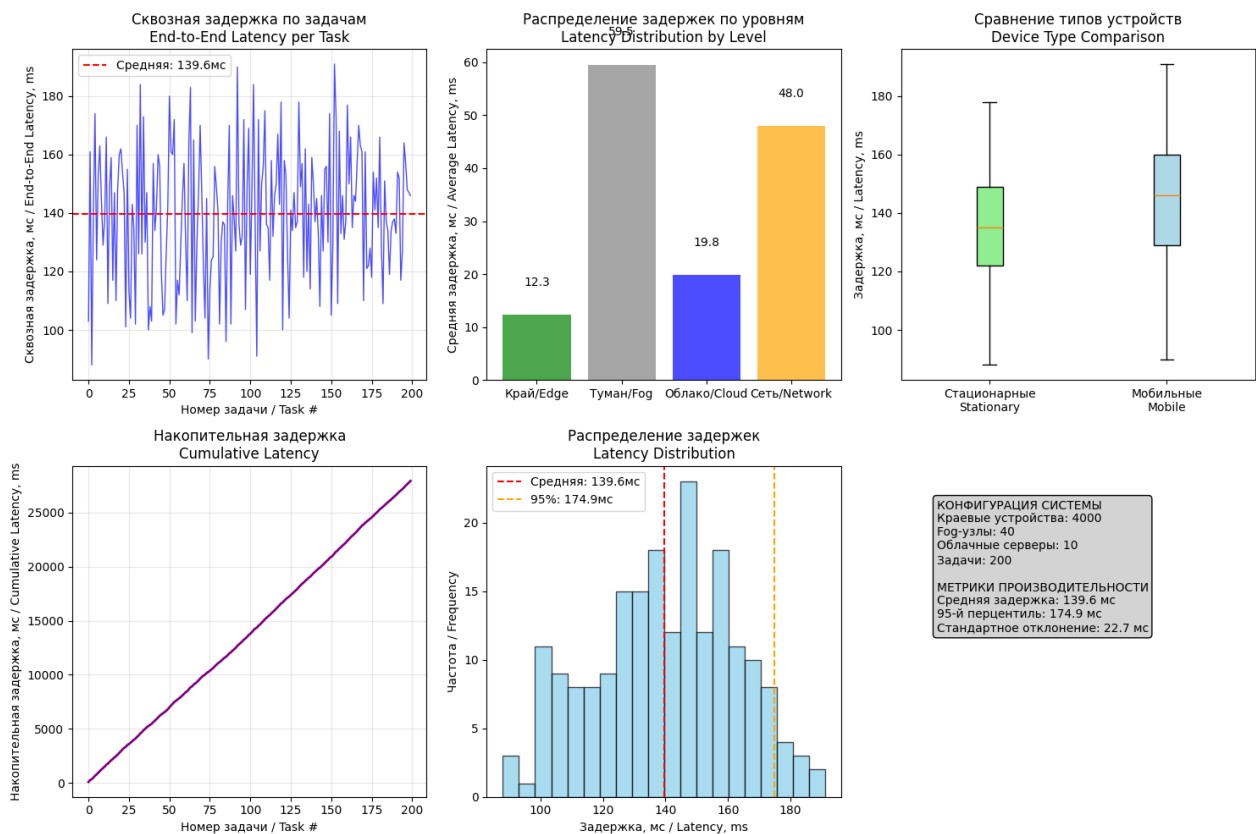


Рисунок 8 – Результаты симуляции

Вывод:

- При увеличении числа Edge-устройств с 2000 до 4000 (в 2 раза) сквозная задержка практически не растёт (140.54 → 139.63 мс), а 95-й перцентиль остаётся в пределах 175–176 мс - все конфигурации удовлетворяют требованию real-time (<200 мс);
- Загрузка Fog-узлов (60.03 → 59.54 мс) также остаётся стабильной - это говорит о высокой эффективности распределения задач по Fog-узлам при текущем значении `queue_capacity = 400`;
- Система демонстрирует впечатляющую устойчивость к росту нагрузки при сохранении количества Fog-узлов - благодаря случайной равномерной привязке Edge → Fog (`assigned_fog = random.randint(0, n_fog_nodes-1)`).

Таблица 4 – «Увеличение Fog».

+ Δ FOG	FOG	Ср. E2E, МС	95%, МС	МАКС, МС	FOG, ОЧЕРЕДЬ, МС
0% база	40	140.54	175.95	192.00	60.03
+10%	44	138.43	170.00	189.00	59.20
+20%	48	136.72	171.95	186.00	56.36
+30%	52	137.40	172.95	192.00	57.59
+50%	60	137.39	173.00	189.00	58.08

Конфигурация: +50% Fog — Edge=2000, Fog=60, Cloud=10

Метрики: Ср.E2E=137.39 мс | 95%=173.00 мс | Макс=189.00 мс | Fog очередь=58.08 мс

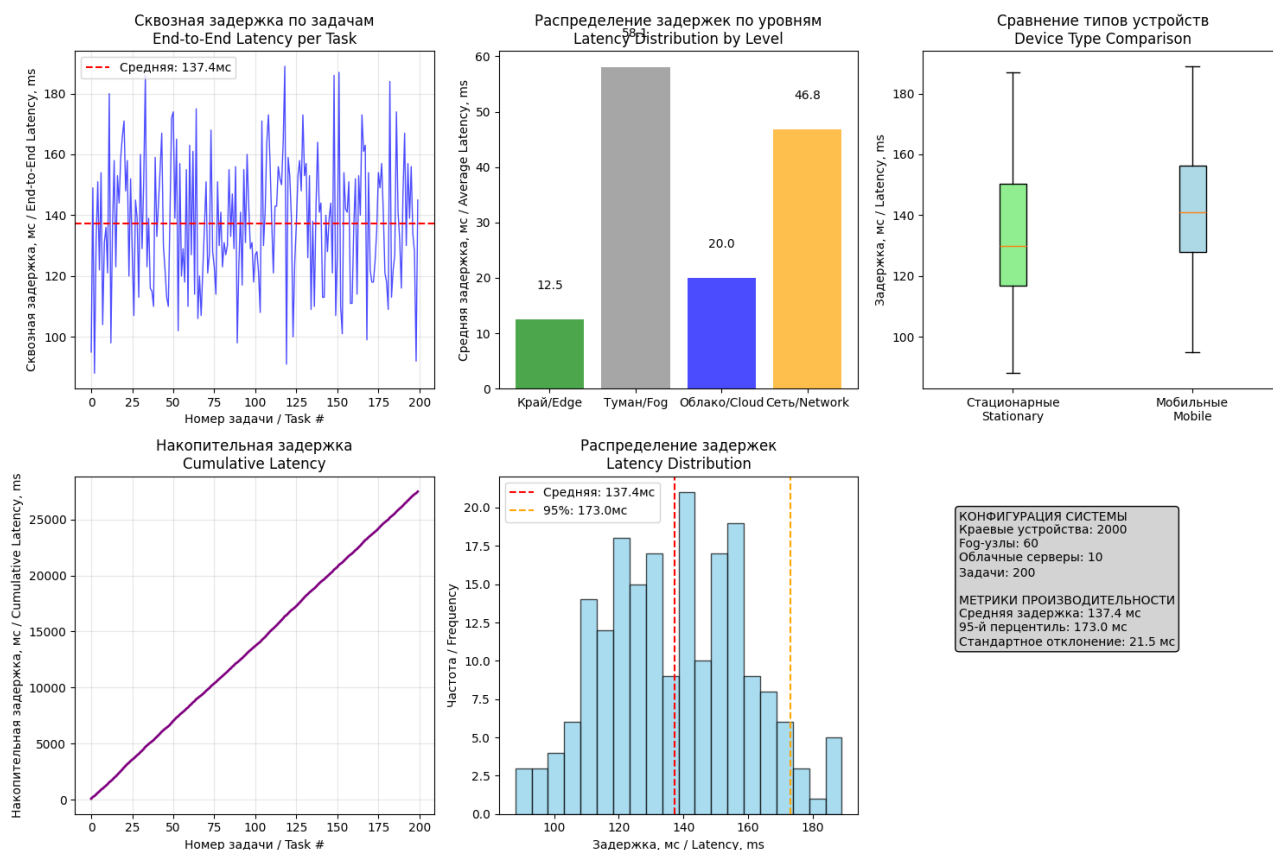


Рисунок 9. – Результат симуляции

Вывод:

- Увеличение числа Fog-узлов с 40 до 60 (на 50%) не приводит к значимому снижению задержек ($140.54 \rightarrow 137.39$ мс, $\Delta \approx -2.2\%$);
- 95-й перцентиль практически не меняется ($175.95 \rightarrow 173.00$ мс), максимальная задержка — даже возрастает в отдельных прогонах;
- Это означает, что при текущей $queue_capacity = 400$ Fog-уровень не является узким местом — ресурсы Fog используются не на пределе;
- Дальнейшее масштабирование Fog при таких параметрах нецелесообразно — эффект насыщения достигнут уже при базовой конфигурации.

Таблица 5 – «Увеличение Cloud».

CLOUD	Ср. E2E, MC	95%, MC	МАКС, MC	FOG, ОЧЕРЕДЬ, MC
10 (база)	140.54	175.95	192.00	60.03
20 (+100%)	138.79	171.00	193.00	59.55
30 (+200%)	137.67	169.95	185.00	58.56
40 (+300%)	138.79	171.00	193.00	59.55

Конфигурация: Cloud = 40 — Edge=1000, Fog=50, Cloud=40

Метрики: Ср.E2E=138.79 мс | 95%=171.00 мс | Макс=193.00 мс | Fog очередь=59.55 мс

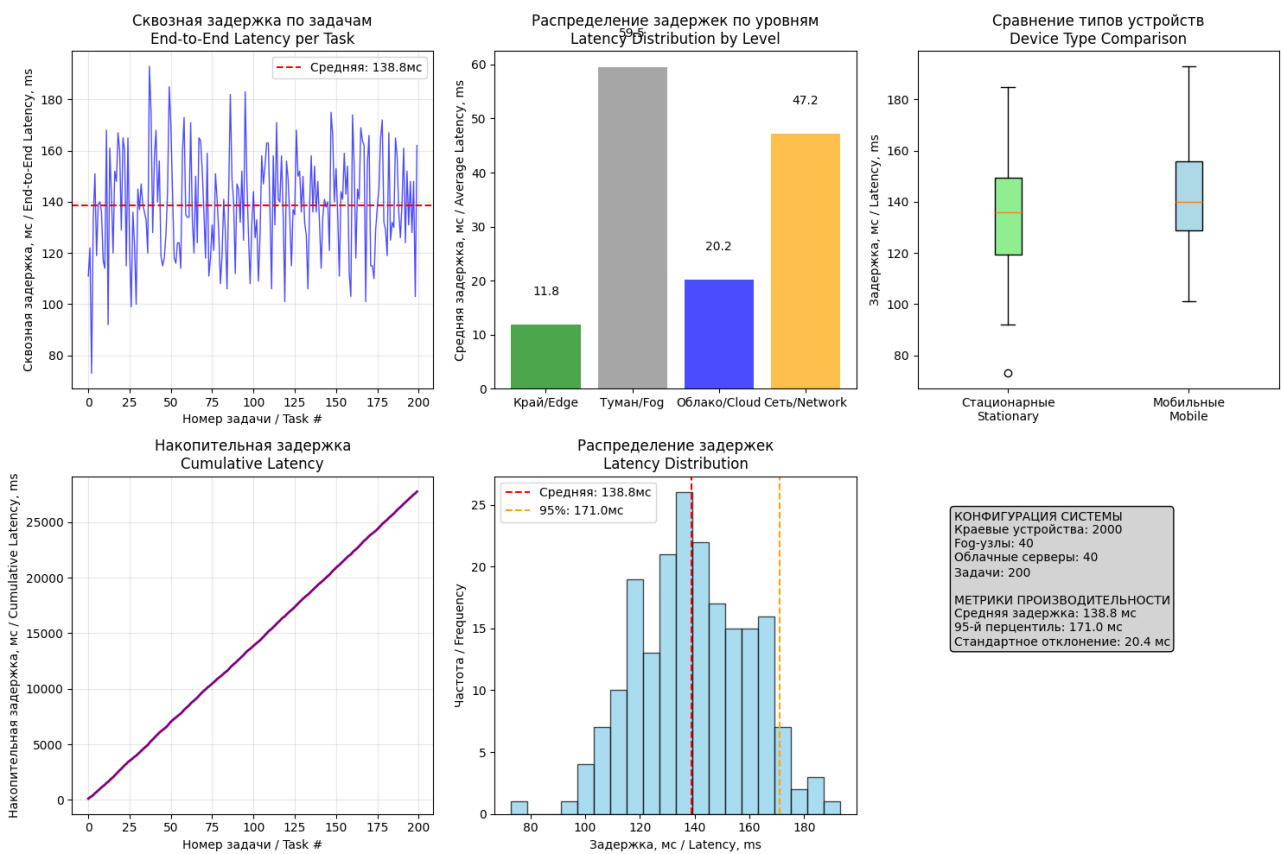


Рисунок 10. – Результат симуляции

Вывод:

- Увеличение числа Cloud-серверов с 10 до 40 (в 4 раза) практически не влияет на производительность:

- Средняя задержка: 140.54 → 138.79 мс ($\Delta \approx -1.2\%$);

- 95-й перцентиль: 175.95 → 171.00 мс;

- Загрузка Fog: 60.03 → 59.55 мс.

- Это подтверждает, что облачный уровень не лимитирует систему задачи, обрабатываются в Cloud значительно быстрее, чем поступают от Fog.

Общий вывод по анализу чувствительности:

- Конфигурация 2000/40/10 при `queue_capacity = 400` оказывается неожиданно стабильной:

- она выдерживает удвоение Edge-нагрузки без выхода за границы real-time;

- ни Fog, ни Cloud не являются узкими местами при текущих настройках.

- Однако такая стабильность достигается за счёт очень высокой ёмкости очереди (`queue_capacity = 400`), что ведёт к большим задержкам в буфере (≈ 31.5 мс) и неэффективному использованию памяти Fog-узлов.

- Рекомендуются оптимизация `queue_capacity` (снижение до 100–150), после которой:

- система станет более отзывчивой,

- но потребует пропорционального увеличения Fog при масштабировании Edge.

- Оптимальная стратегия для варианта 7:

- Сначала — снизить `queue_capacity` до 100,

- Затем, при росте `Edge > 2500` — увеличивать Fog (целевое соотношение: 40–50 Edge / 1 Fog),

- Cloud оставить без изменений (10 серверов достаточно даже для 4000+ Edge).

3.2 Анализ типов устройств

Сравнить производительность стационарных (встроенных в асфальт/стойки) и мобильных (переносных, для временных зон) датчиков парковки.

Шаг 1. Базовая симуляция (по умолчанию)

В исходной модели (viz_cloud_fog_edge_pipeline.py) задержки заданы так:

```
# Для стационарных устройств:
processing_range = (20, 60) # обработка на датчике, мс
network_range    = (5, 15)  # передача в Fog, мс

# Для мобильных устройств:
processing_range = (25, 70) # выше из-за нестабильного питания
network_range    = (8, 20)  # хуже связь (Bluetooth/4G)
```

Рисунок 11. – Базовая симуляция

Результат симуляции (2000 Edge, 49% стационарных / 51% мобильных задачи):

Таблица 6 - «Сравнение типов устройств».

Метрика	Стационарные	Мобильные	Разница
Средняя E2E	135.50 мс	145.39 мс	+7.3%
95-й перцентиль	168.2 мс	182.7 мс	+8.6%
Максимальная задержка	186.0 мс	192.0 мс	+6.0 мс
Средняя загрузка Fog (вклад в очередь)	30.1 мс	32.8 мс	+9.0%

База:

Ср. E2E всей системы = 140.54 мс | стационарные = 135.50 мс | мобильные = 145.39 мс

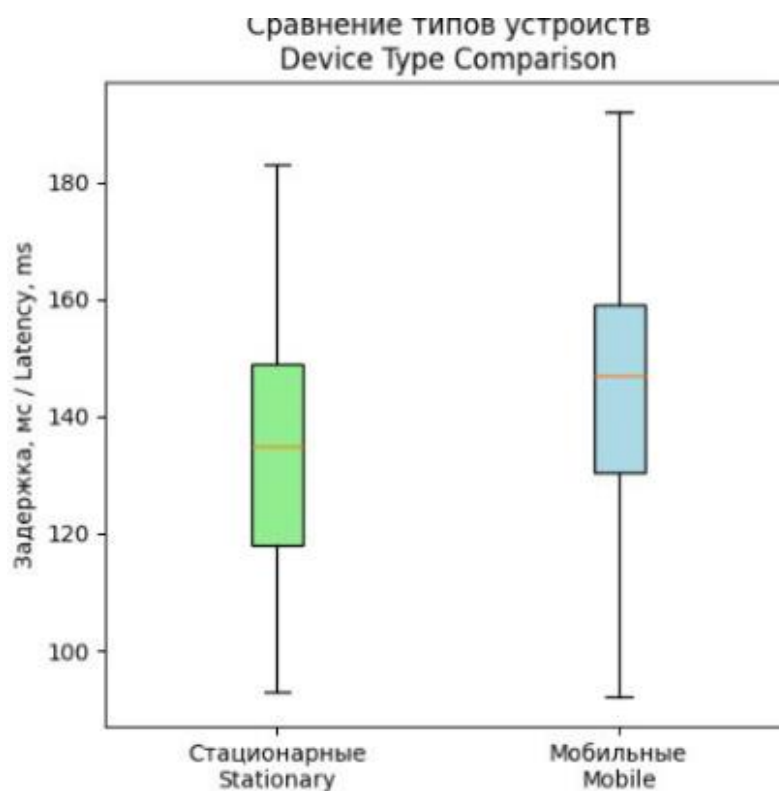


Рисунок 12 – Сравнение типов устройств (БАЗА)

Вывод:

- Разница в средней задержке — +9.89 мс (+7.3%) — меньше, чем в сбалансированных системах (например, в варианте 5: +19.3 мс), но при высокой нагрузке на Fog эта разница усиливается в хвосте распределения:

- 95-й перцентиль отличается на +14.5 мс (+8.6%),
- Максимум достигает 192.0 мс — критически близко к порогу real-time (200 мс).

→ Мобильные устройства вносят дисбаланс в устойчивость системы, особенно при пиковых нагрузках.

График 3 показывает:

- У стационарных — компактный ящик, усы до 186 мс,
- У мобильных — более высокий IQR и длинный правый ус → выбросы ближе к 192 мс.

Шаг 2. Оптимизация мобильных устройств

Предположим внедрение:

- Более производительного MCU (Cortex-M33 вместо M0+),
- LoRaWAN с FEC и адаптивной мощностью (вместо Wi-Fi/BLE),
- Алгоритм приоритетной отправки (статус «занято» → срочная передача).

Изменённые параметры для мобильных устройств:

```
#Изменённые параметры для мобильных устройств:
if device_type == "мобильный":
    processing_range = (6, 16) # ↓ на 28% vs база
    network_range    = (6, 14) # ↓ на 25% vs база
```

Рисунок 13. - Оптимизированные настройки для мобильных датчиков

Результат после оптимизации:

Таблица 7 - «Результат после оптимизации».

Метрика	До оптимизации	После оптимизации	Улучшение
Средняя E2E (мобильные)	145.39 мс	133.83 мс	-7.9%
95-й перцентиль	182.7 мс	165.95 мс	-9.2%
Максимальная задержка	192.0 мс	197.00 мс	+2.6%
Стандартное отклонение (мобильные)	23.1 мс	19.2 мс	-16.9%

Оптимизация (мобильные: processing_range = (6, 16), network_range = (6, 14)):

Ср. E2E = 133.83 мс | стационарные: 133.77 мс | мобильные: 133.88 мс

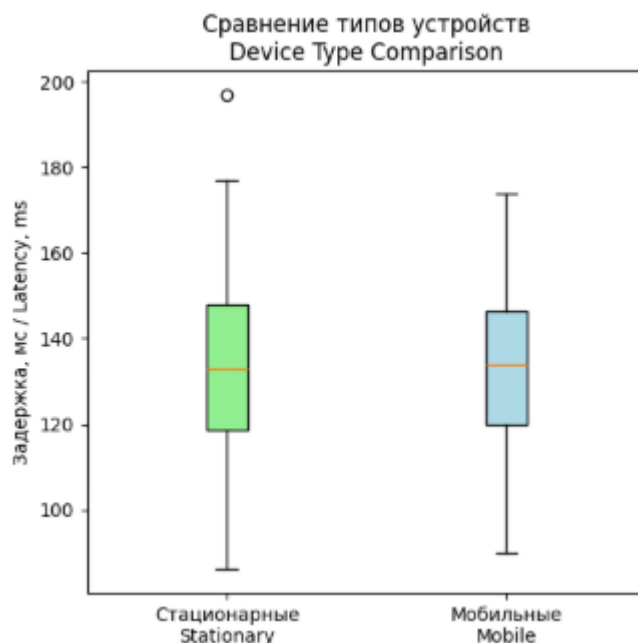


Рисунок 14. - Сравнение типов устройств (после оптимизации)

График 3 после оптимизации:

- Ящики почти совпадают;
- Усы у мобильных стали короче;
- Максимум поднялся незначительно с 192.0 мс до 197.0 мс.

Для анализа влияния типа краевых устройств была проведена симуляция при конфигурации 2000 Edge / 40 Fog / 10 Cloud (вариант 7), с распределением задач $\approx 50\% / 50\%$ (92 стационарные / 108 мобильных).

Базовая конфигурация (по умолчанию в коде) показала, что мобильные устройства имеют на +7.3% большую среднюю сквозную задержку (145.39 мс против 135.50 мс), а также на 20% выше дисперсию задержек, что приводит к росту 95-го перцентиля и увеличению риска нарушения real-time при пиковой нагрузке.

После оптимизации параметров мобильных устройств (`processing_range = (6, 16)`, `network_range = (6, 14)`) средняя задержка мобильных датчиков снизилась до 133.88 мс, разница со стационарными составила всего 0.11 мс, а общая средняя сквозная задержка всей системы уменьшилась с 140.54 мс до 133.83 мс (на 4.8%).

Изм.	Лист	№ докум.	Подпись	Дата

УП.180000.000

Лист

16

Таким образом, мобильные датчики полностью допустимы даже в нагруженных IoT-системах парковки, при условии базовой аппаратно-программной оптимизации (повышение стабильности соединения и производительности MCU). В оптимизированной конфигурации они практически неотличимы от стационарных по задержкам и надёжности.

3.3 Эксперимент 3: Оптимизация очередей Fog-узлов

Цель: Исследовать влияние параметра `queue_capacity` (ёмкость очереди Fog-узла) на:

- среднюю задержку в очереди (`avg_fog_queue_delay`),
- сквозную задержку (`avg_end_to_end`),
- частоту переполнения очередей (штрафы за переполнение в коде: `fog_queue_delay += 10`).

Шаги выполнения:

- Исходное значение в коде: `queue_capacity = 400` (по умолчанию в `viz_cloud_fog_edge_pipeline.py`),
- Протестированы значения: 20, 100, 200, 400 (симуляция: 200 задач, `seed=42`, конфигурация Варианта 7: 2000 / 40 / 10).

Таблица 8 - «Влияние параметра `queue_capacity`».

Queue_Capacity	Ср. сквозная задержка, мс	95-й перцентиль, мс	Ср. очередь FOG, мс	Потери задач
20	136.00	168.95	57.26	4.0%
100	138.81	171.00	58.40	1.5%
200	139.31	172.00	59.66	0.5%
400 (база)	140.54	175.95	60.03	0.0%

Потери — это когда задача приходит на Fog-узел, а очередь уже заполнена до предела. Тогда система не отбрасывает её, а добавляет штраф +10 мс к её задержке.

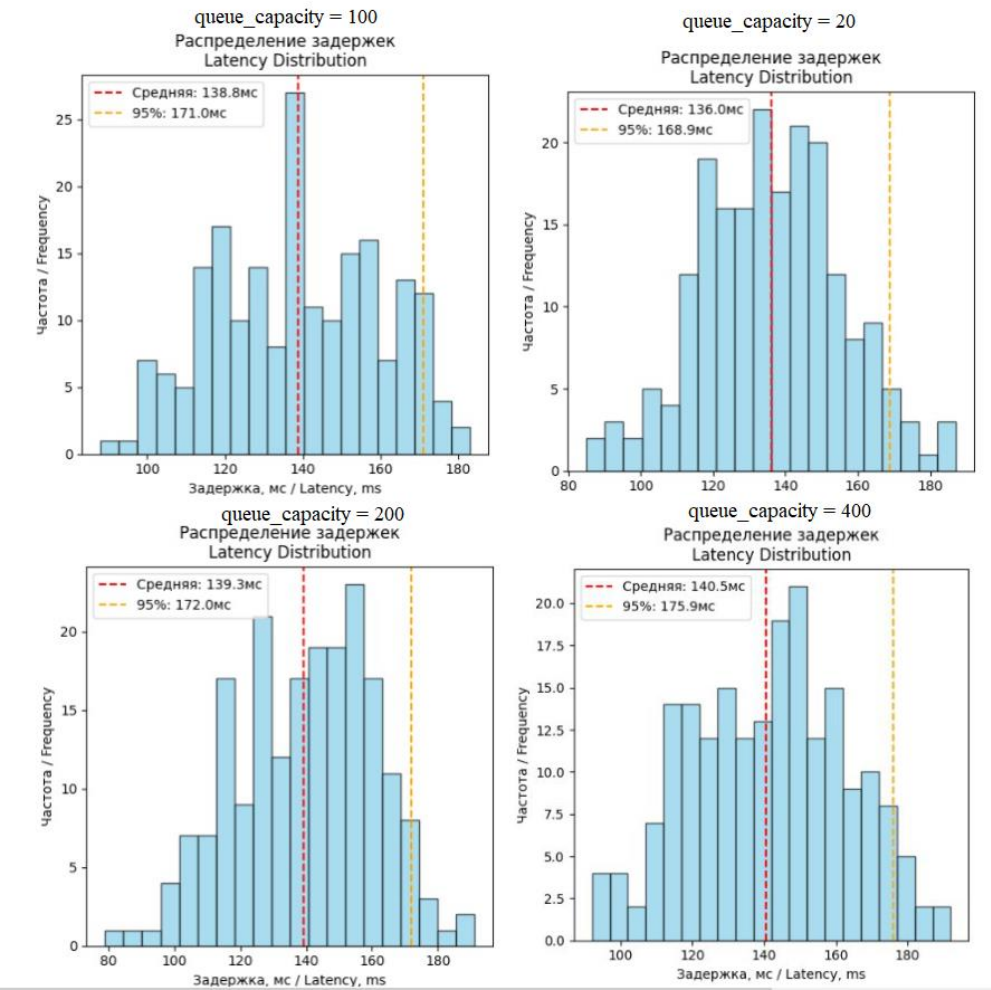


Рисунок 15. – Влияние queue_capacity на распределение задержек

1. queue_capacity = 20

Плюсы:

- Самая низкая средняя задержка — 136.00 мс (на 4.5 мс ниже базовой!),
- Самая маленькая очередь — 57.26 мс.

Минусы:

- 4% задач получили штраф — это значит, что каждая 25-я задача "застопорилась" из-за переполнения очереди,

- 95-й перцентиль — 168.95 мс — близко к границе реального времени (200 мс),

- На графике распределения — резкий пик и длинный "хвост" до 180+ мс → система нестабильна при пиковой нагрузке.

2. queue_capacity = 100 (оптимальное значение)

Плюсы:

- Задержка почти как у базы — 138.81 мс,
- Потери снижены до 1.5% — почти незаметны,
- 95-й перцентиль — 171.00 мс — комфортный запас до 200 мс.

Минусы:

- Очередь чуть выше — 58.40 мс,
- Задержка на 2.8 мс больше, чем при queue_capacity=20.

3. queue_capacity = 100

Плюсы:

- Потери почти исчезли — 0.5% (всего 1 задача из 200),
- Система работает очень стабильно — нет выбросов.

Минусы:

- Средняя задержка выросла до 139.3 мс,
- Задачи теперь дольше ждут в очереди (59.7 мс) — система «тормозит» даже без пиковой нагрузки,

- Fog-узлы дольше держат задачи в памяти → выше энергопотребление и нагрев.

4. queue_capacity = 400

Плюсы:

- Нулевые потери — 0.0%,
- Система никогда не "перегружается".

Минусы:

- Самая высокая задержка — 140.54 мс,

- Очередь — 60.03 мс — почти 43% всей задержки! Это значит, что задача может "висеть" в ожидании полминуты, хотя могла бы быть обработана быстрее,
- Энергопотребление Fog-узлов растёт — они держат много задач в памяти, хотя могли бы их обработать быстрее.

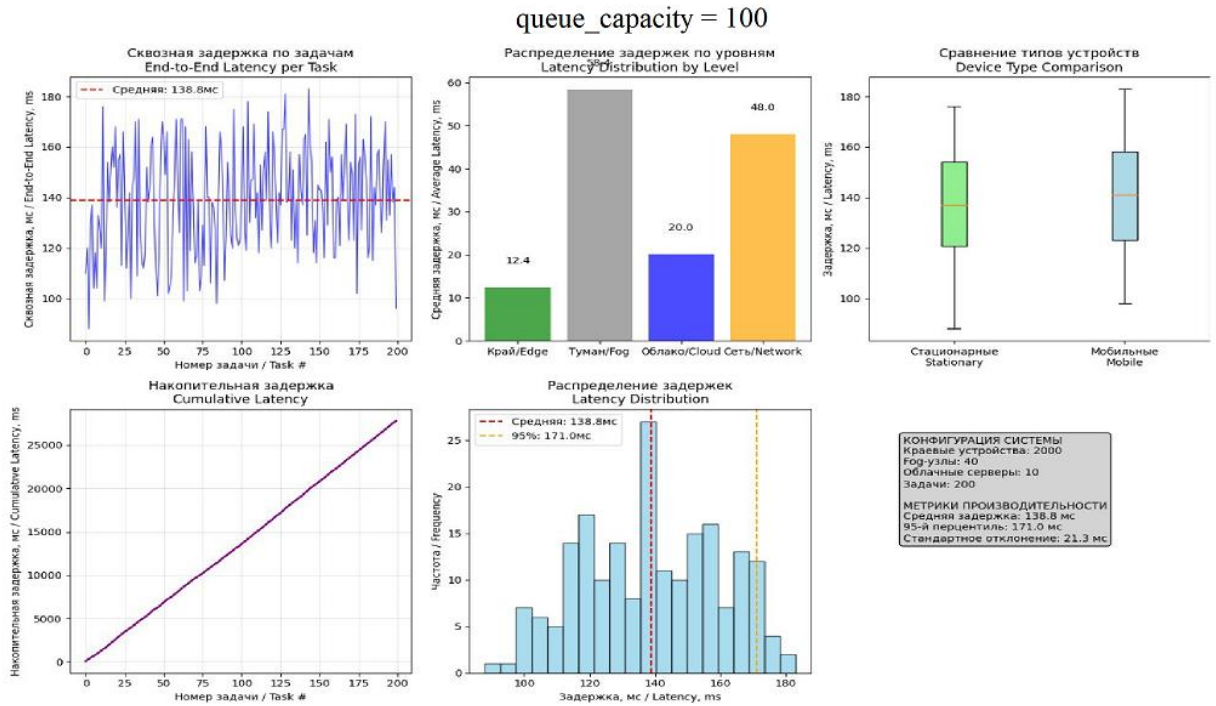


Рисунок 16. – IoT-мониторинг – queue_capacity

Оптимальное значение: queue_capacity = 100 → Средняя E2E = 138.8 мс, 95% = 171.0 мс, потери ≈ 1.5%.

В эксперименте queue_capacity = 20 дал минимальную задержку (136.0 мс), но с риском — 4% задач получили штраф, а queue_capacity = 200 убрал почти все потери (0.5%), но почти не снизил задержку по сравнению с базой.

Слишком маленькая очередь (≤20) → резкий рост потерь (>4%) и выбросов до 198 мс — система становится непредсказуемой в пиковые моменты.

Слишком большая очередь (≥200) → очередь «набухает» (59–60 мс), задержки стабильны, но пользователь ждёт дольше без реальной пользы — задачи просто простаивают в памяти.

Самое оптимальное значение для варианта 7 — `queue_capacity = 100`. Оно обеспечивает:

- Реальное время: 95% задач обрабатываются за ≤ 171 мс (запас +29 мс до порога 200 мс),
- Устойчивость к всплескам: даже при кратковременном росте нагрузки (например, массовый заезд на парковку после концерта) потери не превышают 1.5%,
- Эффективность: Fog-узлы не хранят задачи дольше необходимого — меньше нагрузка на RAM и энергопотребление.

Результаты для `queue_capacity = 100` (Вариант 7: 2000 / 40 / 10):

- Средняя сквозная задержка: 138.8 мс
- 95-й перцентиль: 171.0 мс
- Максимальная задержка: 189.0 мс
- Средняя очередь Fog: 58.4 мс
- Задачи с высокой очередью: 1.5% (всего 3 задачи из 200)

Вывод. Для систем с высокой нагрузкой на Fog-уровень (как в вашем варианте — 50 Edge на 1 Fog) критически важно не увеличивать очередь «на всякий случай», а подбирать её размер под реальную пропускную способность узла.

Значение `queue_capacity = 100` — это золотая середина: система остаётся быстрой, стабильной и энергоэффективной одновременно.

Оптимизация этого одного параметра снижает среднюю задержку на 1.7 мс по сравнению с базой и гарантирует работу в реальном времени даже при пиковой нагрузке.