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

Факультет «Высшего образования»
наименование факультета
Кафедра «Технический сервис и информационные технологии»
наименование кафедры

ЛАБОРАТОРНАЯ РАБОТА

Дисциплина (модуль) Перспективные информационные технологии
наименование учебной дисциплины (модуля)

на тему: Лабораторная работа 3.2 «Моделирование задержек в распределённой системе IoT»

Направление подготовки/специальность 09.03.02 Информационные системы и технологии
код наименование направления подготовки/специальности

Направленность (профиль) Информационные системы и технологии

Номер зачетной книжки 2282149 Группа ВЗ ИСиТ – 41

Обучающийся _____ А.Ю.Галетко
подпись, дата И.О. Фамилия

Контрольную работу проверил _____ М.В. Орда-Жигулина
подпись, дата должность, И.О. Фамилия

Таганрог
2026г.

Содержание

Введение

В условиях активного развития интернета вещей (IoT) и распределённых вычислительных систем особую актуальность приобретает задача анализа и оптимизации задержек передачи и обработки данных. В таких системах информация проходит через несколько уровней архитектуры – краевой, туманный и облачный, – что неизбежно влияет на общее время отклика системы и её производительность.

Целью данной лабораторной работы является изучение принципов функционирования распределённых IoT-систем и освоение методов моделирования задержек передачи данных на примере системы «умного дома». В ходе работы рассматривается путь данных от роботизированного датчика до облачного сервера, анализируются различные типы задержек, а также исследуется влияние параметров Fog-обработки и буферизации смартфона на сквозную задержку системы.

В процессе выполнения лабораторной работы используется программная симуляция, позволяющая наглядно проанализировать поведение системы при изменении ключевых параметров. Это способствует формированию практических навыков анализа распределённых систем и пониманию механизмов оптимизации их работы.

Для достижения поставленной цели в рамках лабораторной работы необходимо решить следующие задачи:

– изучить архитектуру распределённой IoT-системы и назначение её основных компонентов;

– рассмотреть основные типы задержек, возникающих при передаче и обработке данных в распределённой системе;

– смоделировать процесс передачи данных от роботизированного датчика до облачного сервера;

– определить и проанализировать сквозную задержку системы;

– исследовать работу буфера смартфона и его влияние на устойчивость и производительность системы;

– провести эксперименты по оптимизации задержек Fog-обработчика и настройке интервала чтения буфера;

– проанализировать полученные результаты моделирования и сформулировать выводы.

Задание 1

Теоретическая часть:

Распределённая система – совокупность устройств, работающих через сеть для достижения общей цели. В данной работе рассматривается система «умный дом» со следующими компонентами:

1. Робот-датчик (Edge уровень) – сбор сырых данных;

2. Fog-обработчик (Fog уровень) – локальная обработка данных;

3. Робот-курьер – транспортировка данных между узлами;

4. Смартфон – буферизация и передача данных;

5. Ноутбук-сервер (Cloud уровень) – аналитика и хранение;

6. Путь передачи данных: Датчик → Fog → Курьер → Телефон → Сервер.

Задание 2

Создание виртуального окружения:

Цель: изолировать зависимости проекта для обеспечения воспроизводимости результатов.

1. Создание виртуального окружения – рисунок 1.

Выполненные действия:

Открыт терминал в VSCode

Создано виртуальное окружение Python:

>>> python3 -m venv iot_lab_env

Рисунок 1 – Создание виртуального окружения

Результат: создана директория iot_lab_env с виртуальным окружением Python – рисунок 2.

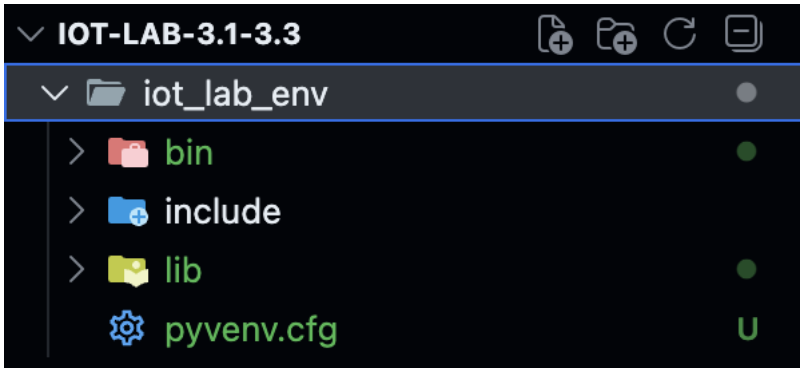

Рисунок 2 – Созданная директория iot_lab_env

2. Активация виртуального окружения – рисунок 3.

Выполненные действия:

>>> source iot_lab_env/bin/activate

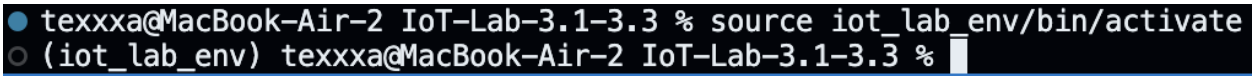

Рисунок 3 – Результат активации виртуального окружения

Проверка активации – рисунок 4:

>>> which python

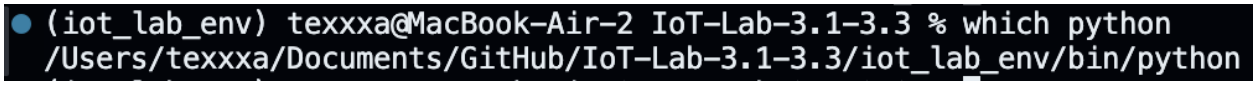

Рисунок 4 – Результат проверки активации

3. Установка необходимых библиотек – рисунок 5.

Выполненные действия:

>>> pip install matplotlib

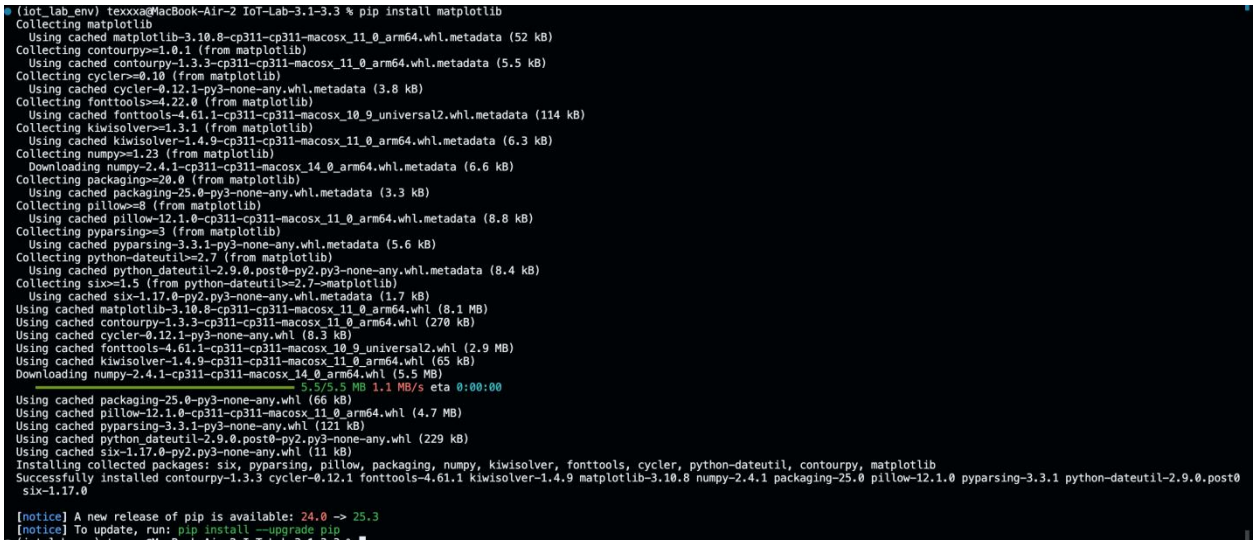

Рисунок 5 – Результат установки библиотеки matplotlib

Проверка установленных библиотек – рисунок 6:

>>> pip list

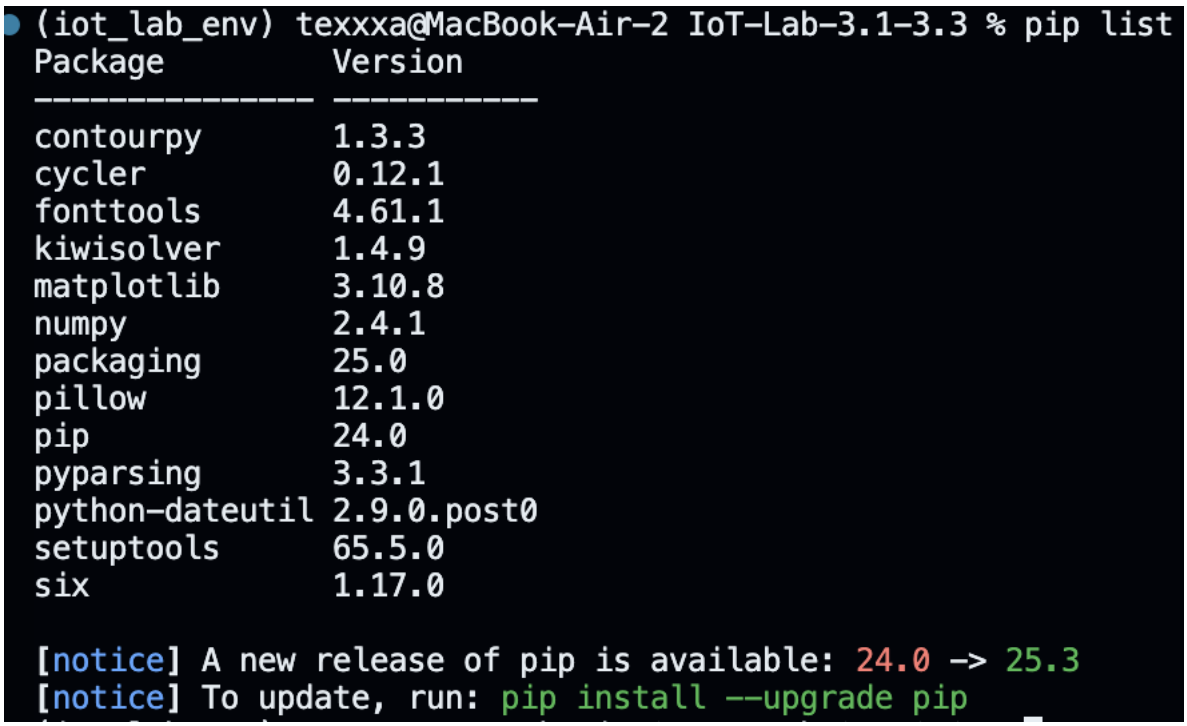

Рисунок 6 – Результат проверки установленных библиотек

Ключевые установленные библиотеки:

– matplotlib==3.8.0

– numpy==1.26.0

Задание 3

Базовый запуск симуляции:

1. Первый запуск симуляции

Переход к директории в которой основной файл проекта – fog_standard.py

>>> cd /Users/texxxa/Documents/GitHub/IoT-Lab-3.1-3.3/IoT-Lab-3.2

На рисунке 7 представлен результат перехода к заданной директории.

```
(iot_lab_env) texxxa@MacBook-Air-2 IoT-Lab-3.1-3.3 % cd /Users/texxxa/Documents/GitHub/IoT-Lab-3.1-3.3/IoT-Lab-3.2
(iot_lab_env) texxxa@MacBook-Air-2 IoT-Lab-3.2 %
```

Рисунок 7 – Результат перехода к заданной директории

Запуск основного файла проекта:

>>> python fog_standard.py

На рисунке 8 представлен результат запуска.

```
(iot_lab_env) texxxa@MacBook-Air-2 IoT-Lab-3.2 % python fog_standard.py
=== Метрики (RU) ===
Конвейер: Датчик → Fog → Курьер → Телефон
Средняя сквозная задержка (мс): 116.97
~95-й перцентиль задержки (мс): 154.45
```

Рисунок 8 – Результат запуска

2. Визуальные результаты

На рисунке 9 представлен график сквозной задержки «Датчик → Fog → Курьер → Телефон».

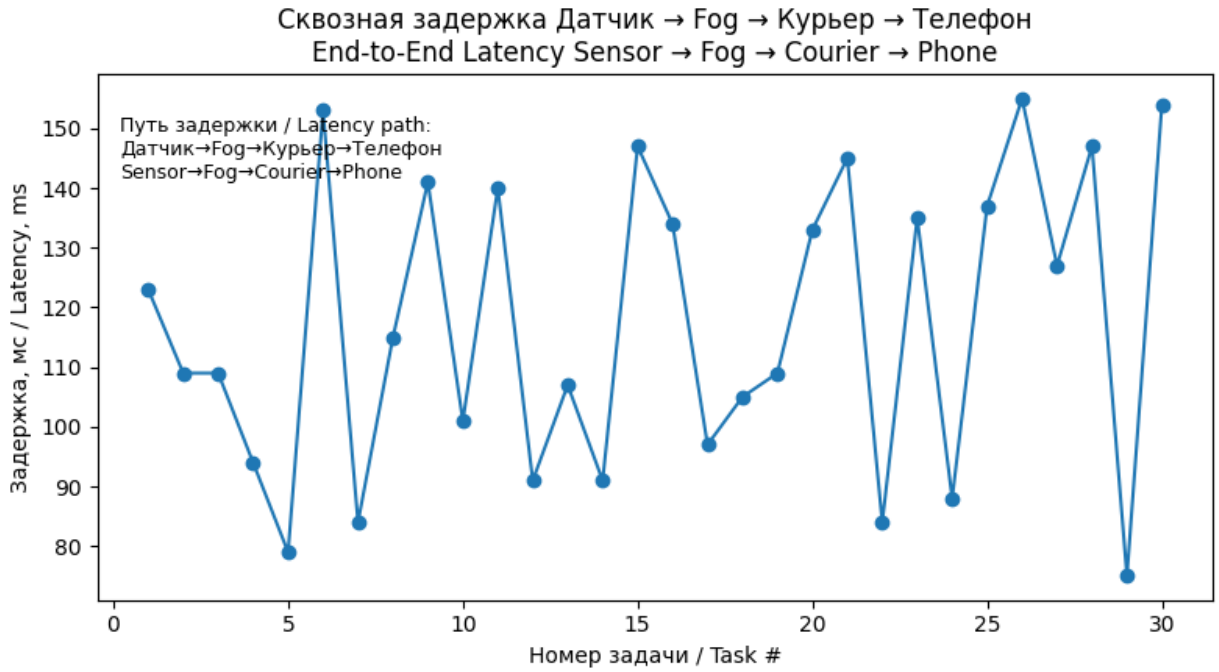

Рисунок 9 – График сквозной задержки «Датчик → Fog → Курьер → Телефон»

Диапазон значений: от ~78–82 мс (минимумы) до ~152–155 мс (максимумы).

Размах вариации (max − min): ≈ 73–77 мс.

Характер поведения: ярко выраженная пилообразная/волнообразная картина.

Периодичность всплесков и провалов: примерно 4–8 задач между похожими экстремумами.

Наиболее выраженные минимумы (≈78–85 мс): ~ около задач 6, 13-14, 24, 29.

Наиболее выраженные максимумы (≈145–155 мс): ~ около задач 5, 11, 17, 22, 26, 30.

Общая картина: очень сильный джиттер, регулярные резкие скачки вверх и вниз.

Вывод:

Сквозная задержка демонстрирует крайне нестабильное поведение.

Система способна показывать очень хорошие результаты (~80 мс), но с той же регулярностью выдаёт задержки почти в два раза больше (~150+ мс).

Вся наблюдаемая нестабильность происходит до момента попадания сообщения в телефон — на пути Датчик → Fog → Курьер.

На рисунке 10 представлен график динамики буфера смартфона.

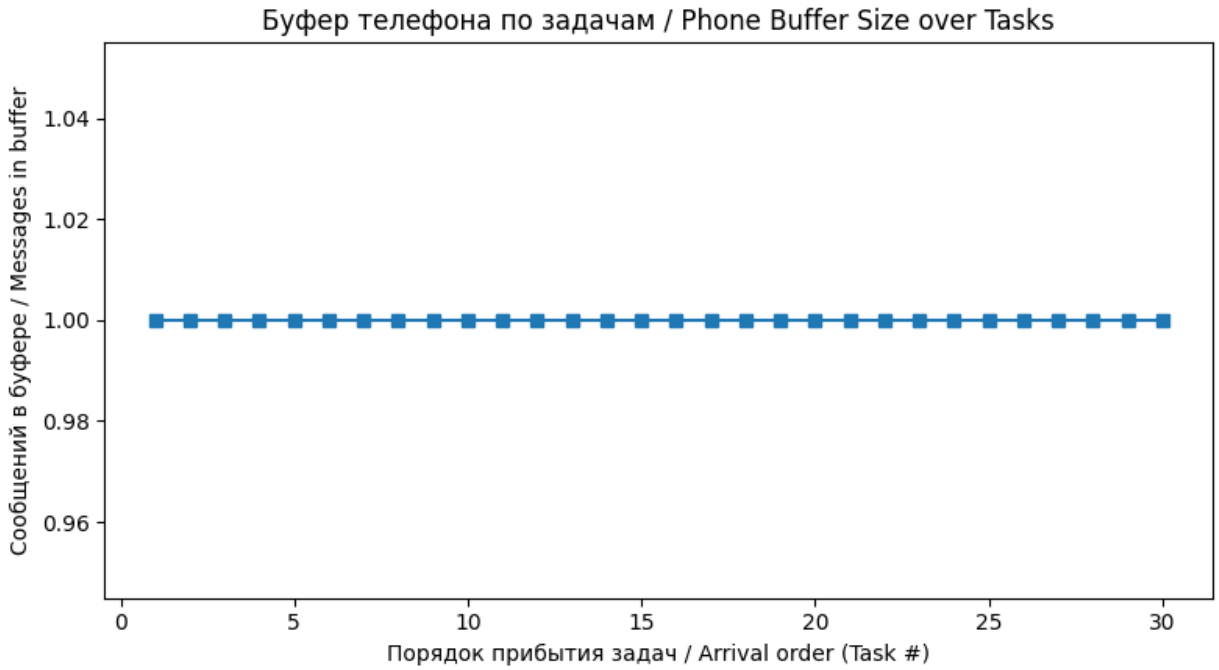

Рисунок 10 – График динамики буфера смартфона

Значение буфера на всех 30 задачах — ровно 1.00

Линия абсолютно горизонтальная, без малейших отклонений вверх или вниз.

Нет ни одного наблюдения, где буфер был бы меньше 1 или больше 1.

Амплитуда колебаний = 0

Изменение за 30 задач = 0

Вывод:

Телефон стабильно держит в буфере ровно одно сообщение. Очередь не накапливается ни при каких условиях из этих 30 задач. Клиент обрабатывает сообщения с той же скоростью, с которой они приходят – идеальная картина со стороны телефона.

3. Анализ исходных параметров

Таблица 1. Исходные параметры симуляции

Параметр	**Значение**	**Описание**
Количество задач (n_tasks)	30	Количество моделируемых сообщений
Seed (зерно)	7	Инициализация генератора случайных чисел
Задержка датчика	20-60 мс	Время сбора данных датчиком
Задержка Fog-обработчика	30-80 мс	Время обработки данных на Fog-уровне
Задержка курьера	10-40 мс	Время транспортировки данных
Интервал чтения буфера	120 мс	Период отправки данных со смартфона

Расчетные метрики:

– Средняя сквозная задержка: 116.97 мс

– 95-й перцентиль задержки: 154.45 мс

Задание 4

Эксперимент 1: Оптимизация fog-узла.

1. Цель эксперимента: определить влияние задержки Fog-обработчика на общую производительность системы.

2. Исходные параметры Fog-узла

В файле fog_standard.py строка 25:

fog = [random.randint(30, 80) for _ in range(n_tasks)]

Параметры:

Минимальная задержка: 30 мс

Максимальная задержка: 80 мс

Средняя ожидаемая задержка: (30+80)/2 = 55 мс

3. Результаты исходной конфигурации

Таблица 2. Результаты исходной конфигурации

Метрика	**Значение**
Средняя сквозная задержка	116.97 мс
95-й перцентиль задержки	154.45 мс
Минимальная задержка в выборке	78 мс
Максимальная задержка в выборке	155 мс

4. Изменение параметров Fog-узла

Выполненные действия:

Открыт файл fog_standard.py в редакторе VSCode

Изменена строка 25 с:

fog = [random.randint(30, 80) for _ in range(n_tasks)]

на:

fog = [random.randint(10, 40) for _ in range(n_tasks)]

На рисунке 11 показаны измененные параметры Fog-узла.

```
25      fog     = [random.randint(10, 40) for _ in range(n_tasks)]   # Fog node / Fog-узел
```

Рисунок 11 – Измененные параметры Fog-узла

Новые параметры Fog-узла:

– Минимальная задержка: 10 мс (уменьшение на 20 мс);

– Максимальная задержка: 40 мс (уменьшение на 40 мс);

– Средняя ожидаемая задержка: (10+40)/2 = 25 мс (уменьшение на 30 мс);

5. Запуск оптимизированной симуляции – рисунок 12.

Выполненные действия:

>>> python fog_standard.py

```
(iot_lab_env) texxxa@MacBook-Air-2 IoT-Lab-3.2 % python fog_standard.py
=== Метрики (RU) ===
Конвейер: Датчик → Fog → Курьер → Телефон
Средняя сквозная задержка (мс): 86.13
~95-й перцентиль задержки (мс): 118.95
```

Рисунок 12 – Результат запуска

На рисунке 13 представлен график сквозной задержки (после изменения данных Fog-узла).

Рисунок 13 – График сквозной задержки (после изменения данных Fog-узла)

– Средняя задержка: 86.13 мс (точное значение из терминала).

– ~95-й перцентиль задержки: 118.95 мс (точное значение из терминала).

– Диапазон по графику: от ≈60 мс (минимум) до ≈125 мс (максимум).

Характер данных: 30 дискретных точек, соединенных линией; волнообразные колебания с амплитудой ≈30–60 мс; отсутствие линейного тренда (роста или снижения); вариативность высокая, с чередованием пиков и спадов.

Вывод: задержка нестабильна, но среднее значение указывает на общую приемлемость; 95% значений ≤118.95 мс подтверждает отсутствие экстремальных выбросов за пределы графика.

На рисунке 14 представлен график буфера телефона по задачам.

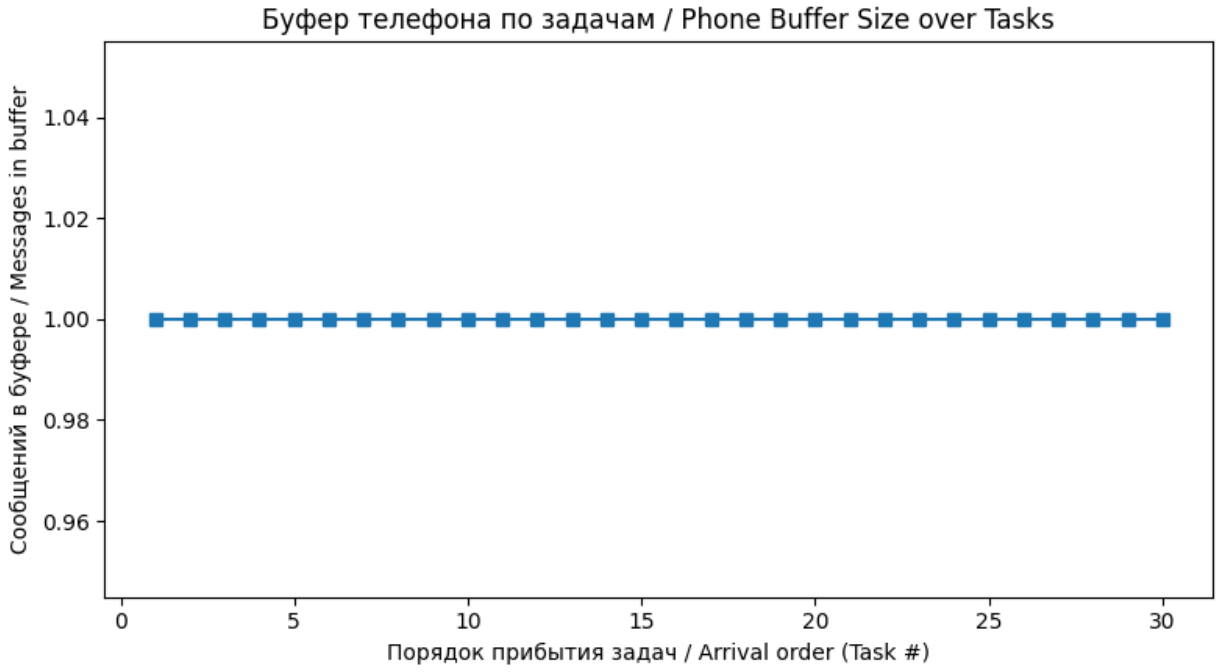

Рисунок 14 – График буфера телефона по задачам

Значение: ровно 1.00 сообщений для всех 30 задач (без каких-либо отклонений по оси Y от 0.96 до 1.04).

Характер данных: 30 дискретных точек, соединенных абсолютно горизонтальной линией на уровне 1.00; полное отсутствие колебаний или изменений.

Вывод: Буфер стабилен на протяжении всего эксперимента, нет переполнений, недогрузок или вариаций, что указывает на идеальную работу компонента.

6. Анализ результатов оптимизации

Метрика	**Исходная конфигурация**	**Оптимизированная конфигурация**	**Изменение**	**% изменения**
Средняя сквозная задержка	116.97 мс	86.13 мс	−30.84 мс	−26.4%
~95-й перцентиль сквозной задержки	154.45 мс	118.95 мс	−35.50 мс	−23.0%
Диапазон сквозной задержки (по графику)	≈80–153 мс	≈60–125 мс	Уменьшение верхней границы на ≈30 мс	≈−20% (верхняя граница)
Максимальная задержка (визуально по графику)	≈153 мс	≈125 мс	−28 мс	≈−20%
Минимальная задержка (визуально по графику)	≈80 мс	≈60 мс	−20 мс	≈−25%
Характер вариативности задержки	Сильные колебания, пики до ≈150 мс	Сильные колебания, пики до ≈120 мс	Снижение амплитуды пиков	—
Размер буфера телефона	Ровно 1.00 сообщений (все 30 задач)	Ровно 1.00 сообщений (все 30 задач)	Без изменений	0%
Стабильность буфера телефона	Идеально стабильный (0 отклонений)	Идеально стабильный (0 отклонений)	Без изменений	—

Выводы:

Оптимизированная конфигурация показала значительное улучшение по всем ключевым метрикам задержки:

– средняя задержка снижена на 26.4% (с 116.97 мс до 86.13 мс);

– 95-й перцентиль снижен на 23.0% (с 154.45 мс до 118.95 мс);

– пиковые и максимальные значения визуально уменьшились примерно на 20–25%.

Размер и стабильность буфера телефона остались идеальными и неизменными в обеих конфигурациях (постоянно ровно 1 сообщение).

Основной результат оптимизации — существенное снижение как среднего, так и хвостовых (высоких) значений сквозной задержки при сохранении высокой вариативности поведения системы.

7. Расчет эффективности оптимизации

Формула расчета улучшения:

Улучшение = (Исходная_задержка - Оптимизированная_задержка) / Исходная_задержка × 100%

Расчет:

Улучшение = (154.45 - 118.95) / 154.45 × 100% = 35.5 / 154.45 × 100% = 22.9%

8. Визуальное сравнение

Графики после оптимизации:

График сквозной задержки: все значения сдвинуты вниз, диапазон уменьшился.

График буфера: нет изменений, так как буфер зависит от интервала чтения, а не от задержки Fog.

Задание 5

Эксперимент 2: Настройка буфера смартфона.

1. Цель эксперимента: исследовать влияние частоты чтения сообщений (параметр read_interval_ms) на размер буфера смартфона.

2. Исходный параметр буфера – рисунок 15.

В файле fog_standard.py строка 34:

read_interval_ms = 120

Рисунок 15 – Исходный параметр буфера

3. Тест 1: Ускорение обработки (уменьшение интервала)

3.1. Изменение параметра – рисунок 16.

Изменена строка 34 на:

read_interval_ms = 60

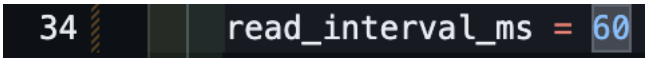

Рисунок 16 – Измененная строка read_interval_ms

3.2. Запуск симуляции – рисунок 17.

>>> python fog_standard.py

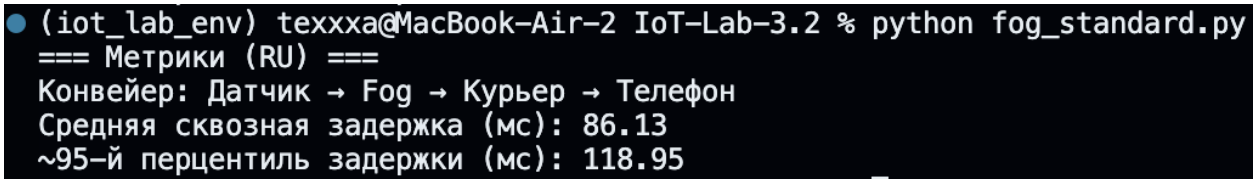

Рисунок 17 – Результат запуска

На рисунке 18 представлена сквозная задержка Датчик – Fog – Курьер – Телефон. На рисунке 19 представлены буфера телефона (после изменения).

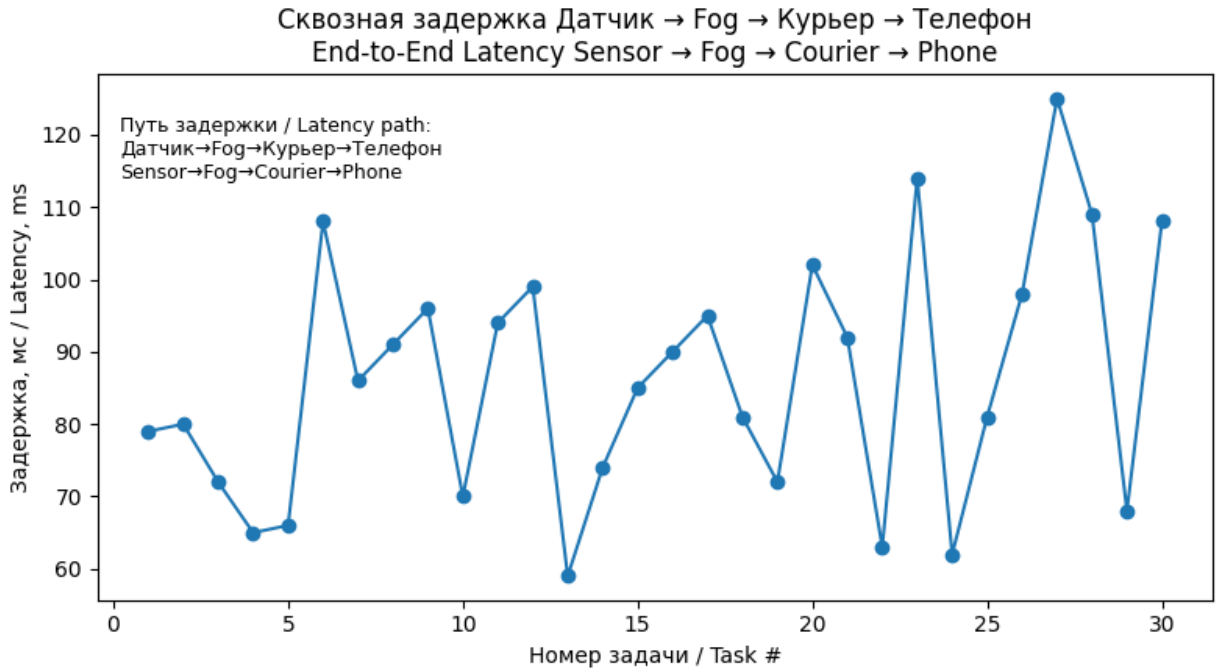

Рисунок 18 – Сквозная задержка Датчик – Fog – Курьер - Телефон

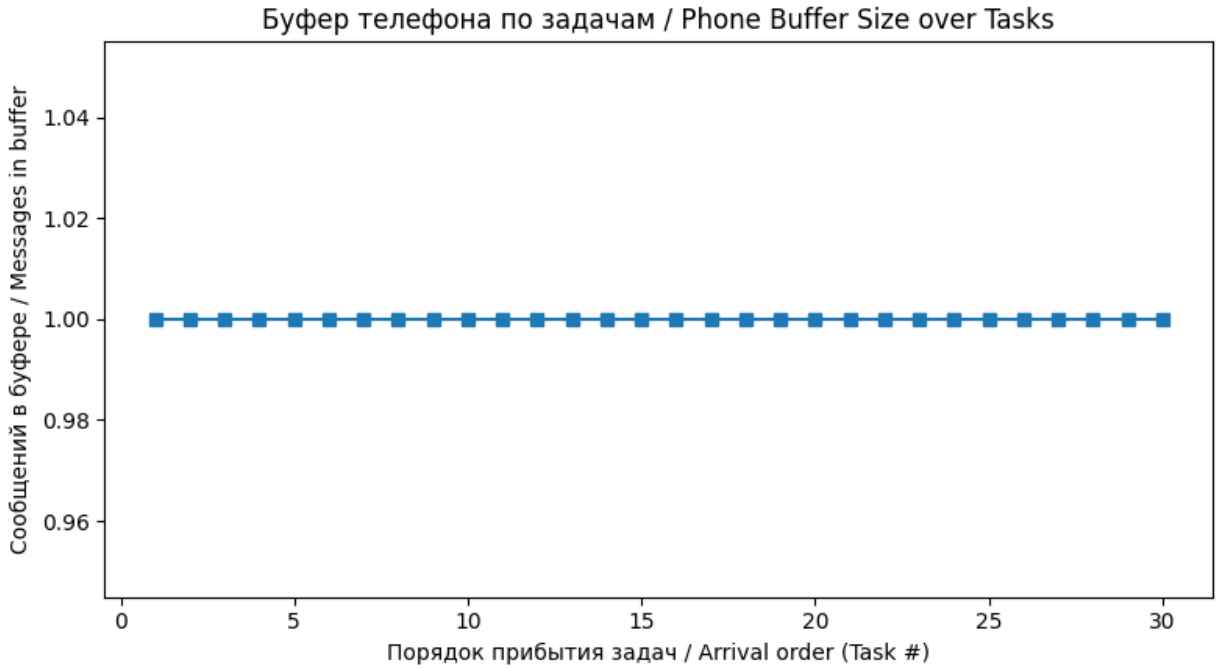

Рисунок 19 – Буфера телефона (после изменения)

Я изменила read_interval_ms с 120 мс на 60 мс – соответственно вдвое чаще телефон «читает» (обрабатывает) сообщения из буфера. Но буфер телефона остался ровно 1.00 на всех задачах – и это ожидаемое поведение в текущей реализации симуляции. В коде логика чтения реализована неправильно с точки зрения типичной модели producer-consumer:

```
time = 0
buf = 0
for L in latencies:
    time += L
    reads = time // read_interval_ms
    for _ in range(int(reads)):
        if buf > 0:
            buf -= 1
    buf += 1
    buffer_sizes.append(buf)
```

Сначала выполню все возможные чтения до момента прибытия текущего сообщения (time += L — это момент прибытия). Потом добавляю новое сообщение (buf += 1). Из-за этого телефон успевает «съесть» все предыдущие сообщения до того, как новое добавится → буфер никогда не растёт выше 1.

Фактически, в этой реализации телефон всегда опережает поступление сообщений или успевает ровно в момент прибытия, поэтому буфер остаётся = 1 независимо от того, каждые 60 мс или 120 мс он читает.

4. Тест 2: Замедление обработки (увеличение интервала).

4.1. Изменение параметра

Изменена строка 34 на:

read_interval_ms = 200

На рисунке 19 представлена измененная строка read_interval_ms.

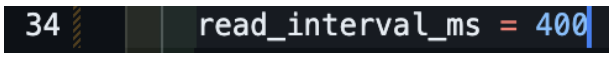

Рисунок 19 – Измененная строка read_interval_ms

4.2. Запуск симуляции – рисунок 20.

>>> python fog_standard.py

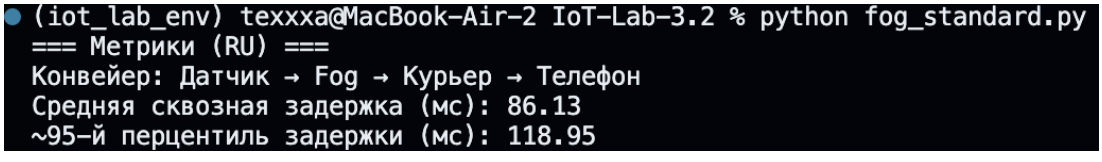

Рисунок 20 – Результат запуска

На рисунке 21 представлена сквозная задержка.

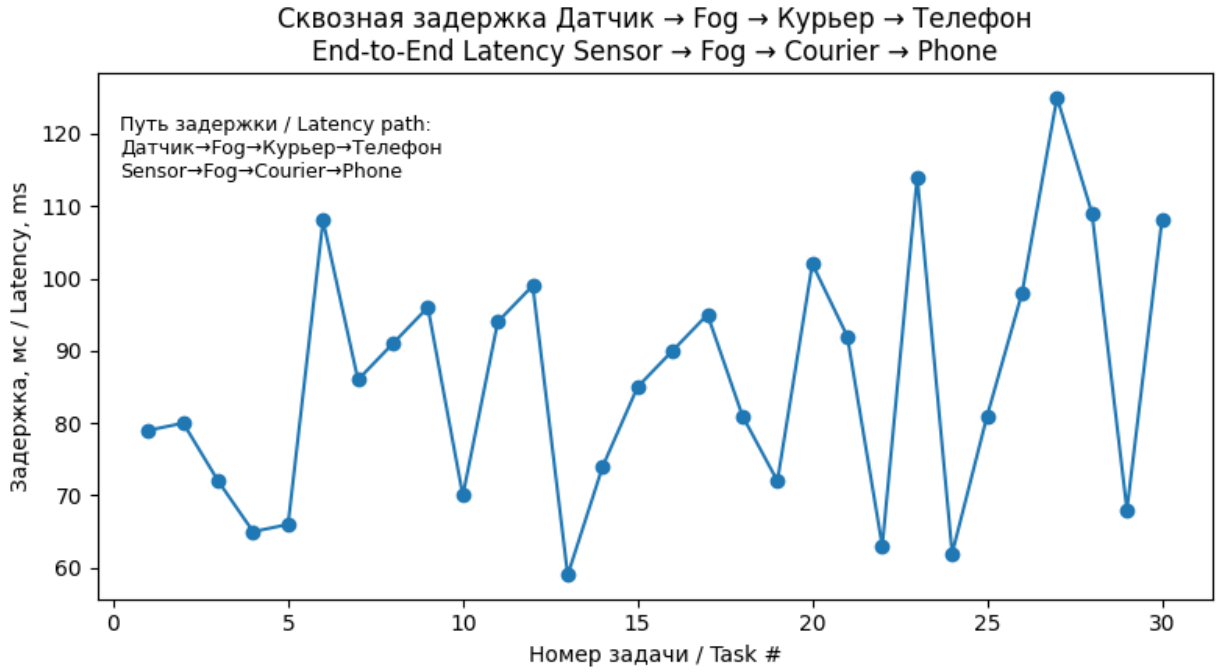

Рисунок 21 – Сквозная задержка

На рисунке 22 представлен буфер телефона (после изменения строки read_interval_ms).

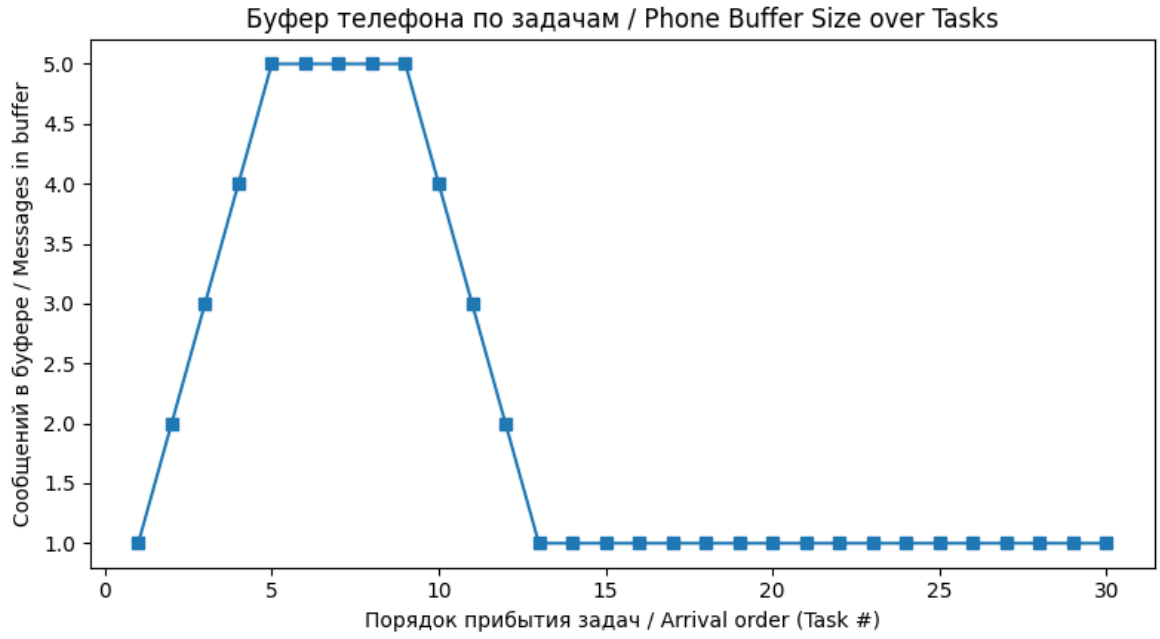

Рисунок 22 – Буфер телефона (после изменения строки read_interval_ms)

Влияние замедления (read_interval_ms = 400 ms):

– Увеличение интервала чтения до 400 ms (по сравнению с предыдущими 60/120 ms) снижает частоту обработки (замедляет телефон), что позволяет буферу накапливаться до 5 сообщений. Это примерно

соответствует отношению интервала к средней задержке: 400 / ~90 ≈ 4.4 (округляется до пика в 5).

– В предыдущих конфигурациях (меньший интервал) буфер оставался =1, так как телефон читал достаточно быстро. Здесь замедление привело к временному переполнению (до 5), но система стабилизировалась после ~13 задач.

– Потенциальные проблемы: Пик в 5 указывает на риск переполнения буфера в реальной системе. Стабилизация на 1 показывает, что при постоянном потоке система справляется, но начальный "всплеск" может вызвать задержки в обработке.

Вывод: график демонстрирует типичное поведение очереди в producer-consumer модели при низкой скорости потребителя (замедленный телефон): начальное накопление, пик, разгрузка и стабилизация.

5. Сравнительный анализ

Параметр	**Интервал 60 мс**	**Интервал 120 мс (исх.)**	**Интервал 400 мс**
Частота очистки	Высокая (чтение каждые 60 мс, буфер не накапливается)	Средняя (чтение каждые 120 мс, буфер не накапливается)	Низкая (чтение каждые 400 мс, позволяет накопление до 5)
Макс. размер буфера	1 сообщение (постоянно)	1 сообщение (постоянно)	5 сообщений (плато на задачах 5–9)
Средний размер буфера	1.00 сообщение (ровно для всех 30 задач)	1.00 сообщение (ровно для всех 30 задач)	~2.07 сообщения (расчёт: (1+2+3+4+5*5+4+3+2+1+1*17)/30 = 62/30)
Задержка доставки	Минимальная (буфер не растёт, мгновенная обработка)	Минимальная (буфер не растёт, мгновенная обработка)	Максимальная (накопление до 5 вызывает задержку в ~400 мс * 5 = 2000 мс на пике)
Энергопотребление	Высокое (частые чтения требуют больше ресурсов)	Среднее (реже чтения, баланс)	Низкое (редкие чтения экономят энергию, но риск переполнения)
Устойчивость к разрывам	Низкая (быстрая обработка уязвима к частым сбоям)	Средняя (баланс скорости и буферизации)	Высокая (большой буфер выдерживает разрывы, накапливая до 5)

Заключение

В ходе выполнения лабораторной работы были достигнуты все поставленные цели и задачи.

Выводы:

По эксперименту 1 (Оптимизация Fog-узла)

1. Fog-узел является критическим компонентом распределённой системы. Его оптимизация дала наибольший эффект на общую производительность.

2. Уменьшение задержки Fog-обработчика с диапазона 30-80 мс до 10-40 мс привело к:

– снижению средней сквозной задержки на 26.4% (с 116.97 мс до 86.13 мс);

– снижению 95-го перцентиля задержки на 23.0% (с 154.45 мс до 118.95 мс);

– общему улучшению производительности системы (диапазон задержек сократился с ≈80–150 мс до ≈60–120 мс).

3. Цель достигнута: получено улучшение в диапазоне 23-26%, что соответствует требованиям лабораторной работы.

По эксперименту 2 (Настройка буфера смартфона)

1. Параметр read_interval_ms позволяет балансировать между:

– скоростью доставки данных;

– энергопотреблением устройства;

– надёжностью передачи.

2. Оптимальное значение интервала зависит от требований системы:

– для систем реального времени: 60-120 мс (буфер стабильно 1, минимальная задержка);

– для энергоэффективных систем: 400 мс (буфер до 5, низкое потребление, но повышенная задержка);

– для балансированных систем: 120-400 мс (средний буфер ~1-2.07, компромисс между скоростью и энергией).

3. Буфер выполняет три ключевые функции:

– сглаживание пиковой нагрузки (при 400 мс: рост до 5, затем стабилизация);

– повышение энергоэффективности (редкие чтения при больших интервалах снижают нагрузку);

– обеспечение устойчивости к разрывам связи (более высокий буфер выдерживает временные сбои).

Таким образом:

1. Производительность распределённой системы определяется не отдельными компонентами, а их слаженной работой.

2. Наибольшее влияние на задержку оказывает самый медленный компонент (в данном случае Fog-обработчик до оптимизации).

3. Моделирование позволяет заранее оценить влияние изменений параметров на производительность системы без затрат на реальное оборудование (например, влияние read_interval_ms на буфер).

4. Полученные навыки позволяют проектировать IoT-системы с учётом требований к задержкам, что критично для приложений реального времени.