Лабораторная работа 3. «Моделирование задержек в распределённой системе IoT»

(Методические указания по проекту STUDY_FOG)

ВВЕДЕНИЕ

Цель лабораторной работы: понять принципы работы распределённых систем, научиться моделировать и анализировать задержки передачи данных в IoT-системах на примере «умного дома».

Распределённая система – совокупность устройств, работающих через сеть для достижения общей цели. В нашем случае — «умный дом» с компонентами: робот-датчик, Fog-обработчик, робот-курьер, смартфон и ноутбук-сервер.

Цели симуляции:

- Проследить путь данных от датчика до сервера
- Измерить сквозную задержку (end-to-end delay)
- Проанализировать буфер смартфона
- Экспериментировать с параметрами системы

Необходимые навыки: базовое знакомство с Python, понимание принципов сетевого взаимодействия.

1. ПОДГОТОВКА РАБОЧЕЙ СРЕДЫ ДЛЯ ВЫПОЛНЕНИЯ ПРОЕКТА

1.1 Скачайте проект с GitHub репозитория robots-fog-mini-sim https://github.com/SA9Z/STUDY_FOG

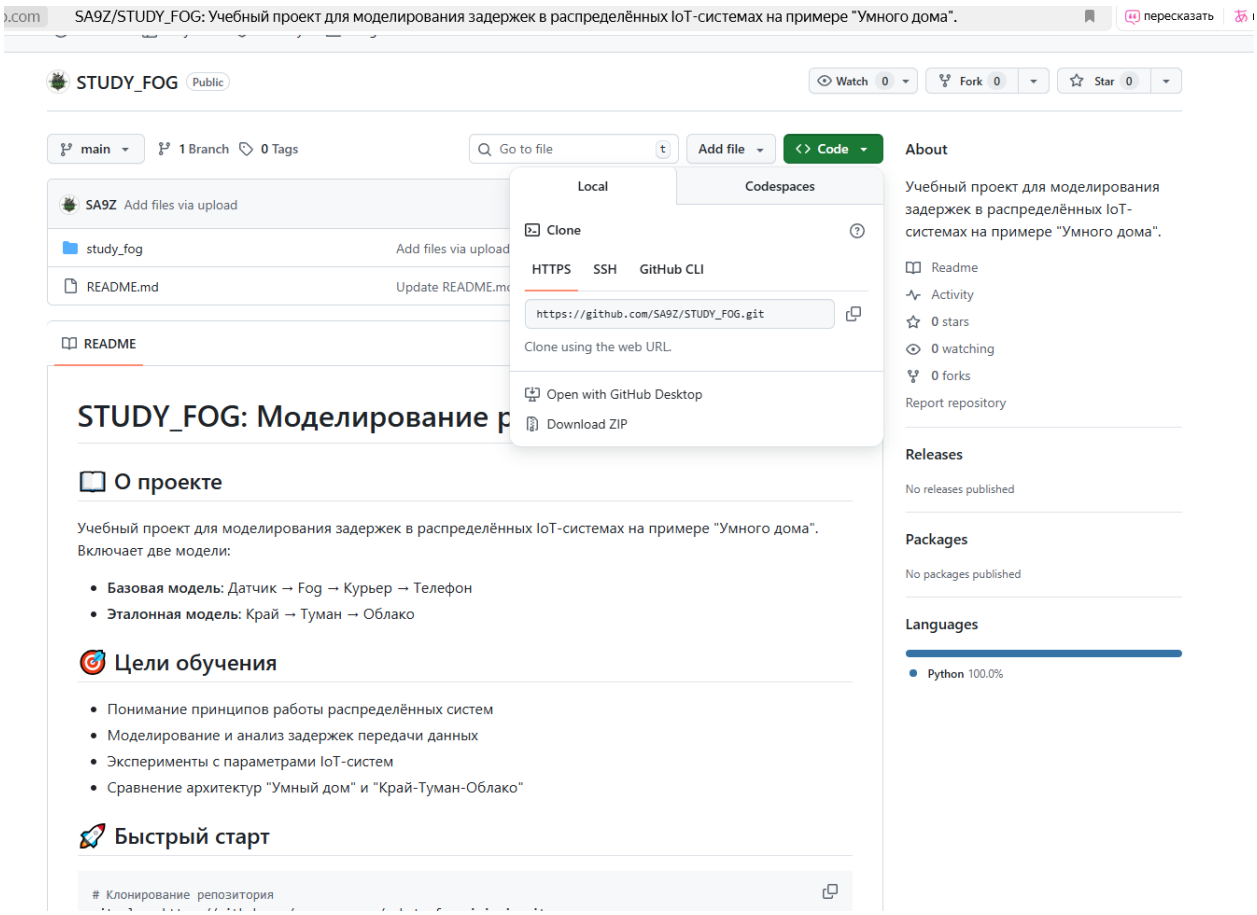

Рисунок 1 – Скачивание репозитория в zip

1.2 Установите зависимости:

>>> pip install -r requirements.txt

1.3 Основные файлы проекта:

- *fog_standard.py* — главный файл симуляции

2. КОМПОНЕНТЫ СИСТЕМЫ

Система состоит из пяти компонентов, распределенных по уровням вычислений:

Уровни архитектуры:

- **Краевой уровень (Edge): Робот-датчик** — сбор сырых данных
- **Туманный уровень (Fog): Fog-обработчик** — локальная обработка

- **Облачный уровень (Cloud): Ноутбук-сервер** — аналитика и хранение

Связующие компоненты:

- **Робот-курьер** — транспортировка данных
- **Смартфон** — буферизация и передача"

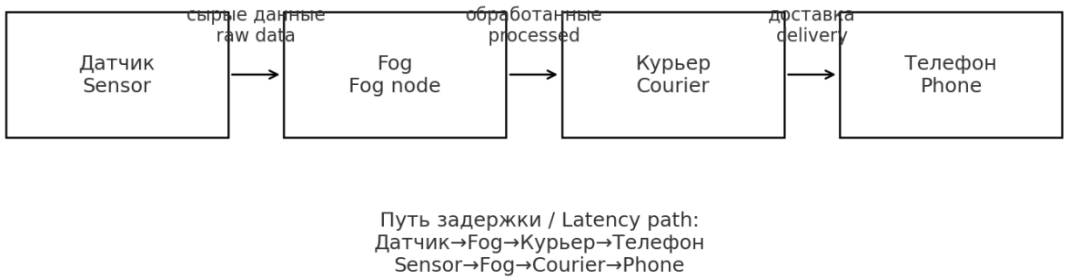

Рисунок 2 – схема работы компонентов системы

Следующая схема представляет собой логическую модель компонентов распределенной системы "Умный дом". Она показывает основные функциональные блоки и направление потока данных между ними. Каждый компонент выполняет строго определенную роль в цепочке обработки информации: от сбора данных (робот-датчик) до их конечного анализа (ноутбук-сервер).

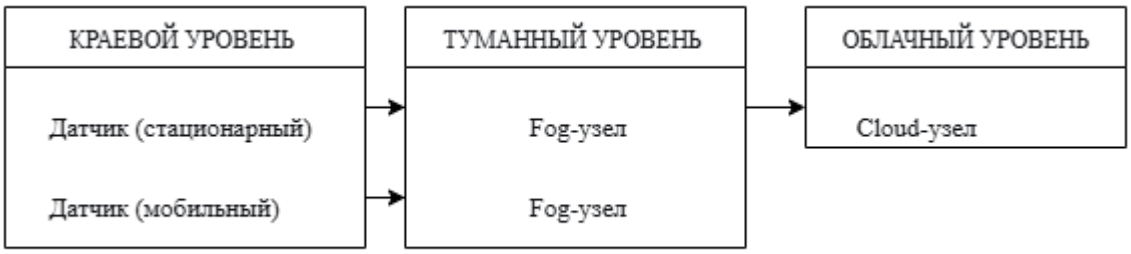

Рисунок 3 - Схема архитектуры системы

Схема на рисунки 4 является **технической реализацией** логической модели из Рисунка 3. Она детализирует, как именно компоненты взаимодействуют на практике, показывая сетевые интерфейсы, протоколы передачи данных и физическую структуру системы. Здесь наглядно отображено разделение на сетевые сегменты (например, локальная сеть датчиков и облачный сервер).

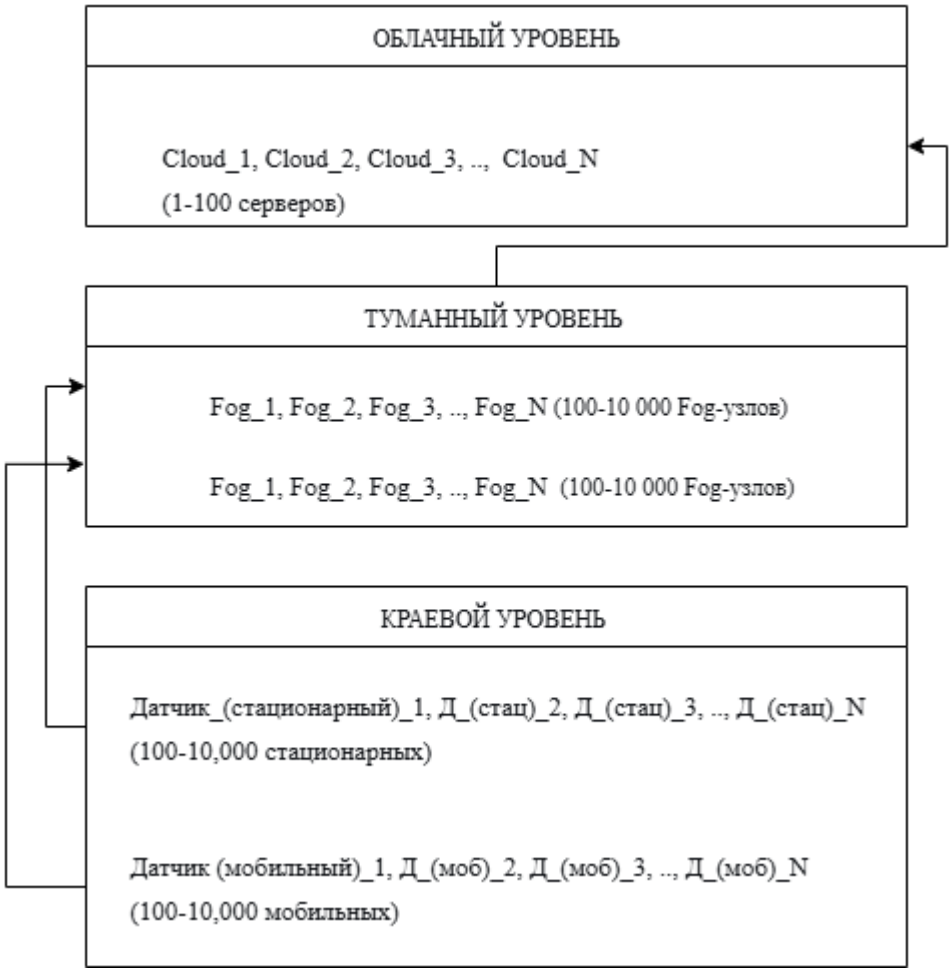

Рисунок 4 – Схема эталонной архитектуры системы

Взаимосвязь и взаимодействие между схемами

Обе схемы описывают одну и ту же систему, но на разных уровнях абстракции:

- **Рисунок 3** отвечает на вопрос **"ЧТО делает система?"** — показывает основные роли и поток данных.

- **Рисунок 4** отвечает на вопрос **"КАК система это делает?"** — показывает техническую реализацию и сетевые взаимодействия.

Краткое описание взаимодействия:

1. **Робот-датчик** собирает данные и передает их **Fog-обработчику** по локальному сетевому протоколу (например, Wi-Fi или Bluetooth)
2. **Fog-обработчик** выполняет первичную обработку и передает результат **роботу-курьеру** через локальную сеть
3. **Робот-курьер** доставляет данные **смартфону** через интернет (мобильная сеть 4G/5G)
4. **Смартфон** буферизует данные и передает их **ноутбуку-серверу** через облачное API или прямое сетевое соединение

3 ОСНОВЫ МОДЕЛИРОВАНИЯ ЗАДЕРЖЕК

3.1. Типы задержек в системе

Основные типы задержек:

- **Сетевая:** время передачи между узлами (пример: передача от Fog к смартфону)
- **Обработки:** время обработки на узле (пример: анализ данных на Fog-роботе)
- **Очереди:** время ожидания в буфере (пример: ожидание в буфере смартфона)
- **Сквозная:** сумма всех задержек (пример: от датчика до сервера)

Формула: Сквозная_задержка = Задержка_датчика + Задержка_Fog + Задержка_курьера + Очередь_смартфона

3.2 Буфер смартфона

Буфер смартфона (или очередь сообщений) — это временное хранилище данных в памяти смартфона. Его основная задача — согласовать разные скорости поступления и потребления данных в системе.

Принцип работы в симуляции:

1. **Поступление:** Робот-курьер доставляет обработанные сообщения на смартфон. Сообщения приходят в случайные моменты времени (определяются задержкой курьера).

2. **Накопление:** Каждое пришедшее сообщение помещается в буфер (очередь), увеличивая его размер на 1.

3. **Чтение (Отправка на сервер):** Смартфон не отправляет каждое сообщение мгновенно. Вместо этого он проверяет буфер с фиксированным интервалом, который задается параметром read_interval_ms (например, 120 мс). При каждой такой проверке он извлекает из буфера и отправляет на сервер **все накопленные на данный момент сообщения**, после чего буфер полностью очищается (buf = 0).

Зачем это нужно?

- **Сглаживание пиковой нагрузки:** Если курьеры одновременно пришлют много сообщений, буфер не даст им потеряться, а сервер успеет их обработать.
- **Энергоэффективность:** Постоянная отправка мелких пакетов данных разряжает батарею смартфона. Групповая отправка данных раз в определенный интервал более эффективна.
- **Устойчивость к разрывам связи:** Если связь с сервером пропадает, данные не теряются, а накапливаются в буфере до ее восстановления.

4. Подготовка к проведению экспериментов

4.1. Запуск базовой симуляции

1. Откройте файл *viz_phone_pipeline_bilingual.py*
2. Запустите симуляцию:

>>>python viz_phone_pipeline_bilingual.py

3. Проанализируйте полученные графики:

- График сквозной задержки по задачам
- График динамики буфера смартфона

4.2. Анализ кода моделирования

Изучите функцию *simulate()* в основном файле:

```
def simulate(n_tasks=30, seed=7):
    random.seed(seed)
    # Задержки для каждого компонента (в миллисекундах)
    sensor  = [random.randint(20, 60) for _ in range(n_tasks)]
    fog     = [random.randint(30, 80) for _ in range(n_tasks)]
    courier = [random.randint(10, 40) for _ in range(n_tasks)]

    # Расчёт сквозной задержки
    latencies = [s + f + c for s, f, c in zip(sensor, fog, courier)]

    # Моделирование буфера смартфона
    read_interval_ms = 120
    time = 0
    buffer_sizes = []
    buf = 0
    # ... (логика обработки очереди)
```

Ключевые параметры:

- n_tasks — количество сообщений для симуляции
- read_interval_ms — интервал чтения сообщений смартфоном
- Диапазоны задержек для каждого компонента

5. Проведение экспериментов (моделирование)

5.1. Эксперимент 1: Оптимизация Fog-узла

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

Шаги выполнения:

1. Найдите в коде строку, которая отражает время задержки для Fog-узла: fog = [random.randint(30, 80) for _ in range(n_tasks)]. Запишите это начальное время задержки для Fog-узла в отчет.

Запустите симуляцию для начального времени задержки и отразите в отчёте полученные результаты симуляции (значение и график средней сквозной задержки системы). В результате у вас должны получиться графики средней сквозной задержки системы, похожие на график на рисунке 5.

Рисунок 5 – Оптимизированная средняя сквозная задержка системы

2. Затем измените в коде значение времени задержки Fog-узла на оптимизированное:

fog = [random.randint(10, 40) for _ in range(n_tasks)]. Запишите оптимизированное время задержки для Fog-узла в отчет.

3. Снова запустите симуляцию и запишите рассчитанные оптимизированные значение и график средней сквозной задержки в отчёт.

Сравните оптимизированную среднюю сквозную задержку с неоптимизированной. Проверьте, достигли ли вы о**жидаемого результата по** уменьшению сквозной задержки на 25-30%. Если нет, то снова измените время задержки Fog-узла как указано на шаге 3 и повторите шаги 1-5.

5.2 Эксперимент 2: Настройка буфера смартфона

Цель: исследовать влияние частоты чтения сообщений на размер буфера смартфона.

Шаги выполнения:

1. Найдите параметр *read_interval_ms* в функции *simulate()*
2. Измените значение:

- Для ускорения обработки: *read_interval_ms = 60 (уменьшите значение)*
- Для замедления обработки: *read_interval_ms = 200 (увеличьте значение)*

3. Запустите симуляцию для каждого случая
4. Проанализируйте график буфера смартфона

Проверьте, достигли ли вы следующего ожидаемого результата, когда ри уменьшении интервала чтения буфер будет расти медленнее или чаще опустошаться. Запишите полученные результаты в отчет.

Пример заполнения буфера показан на рисунке 6.

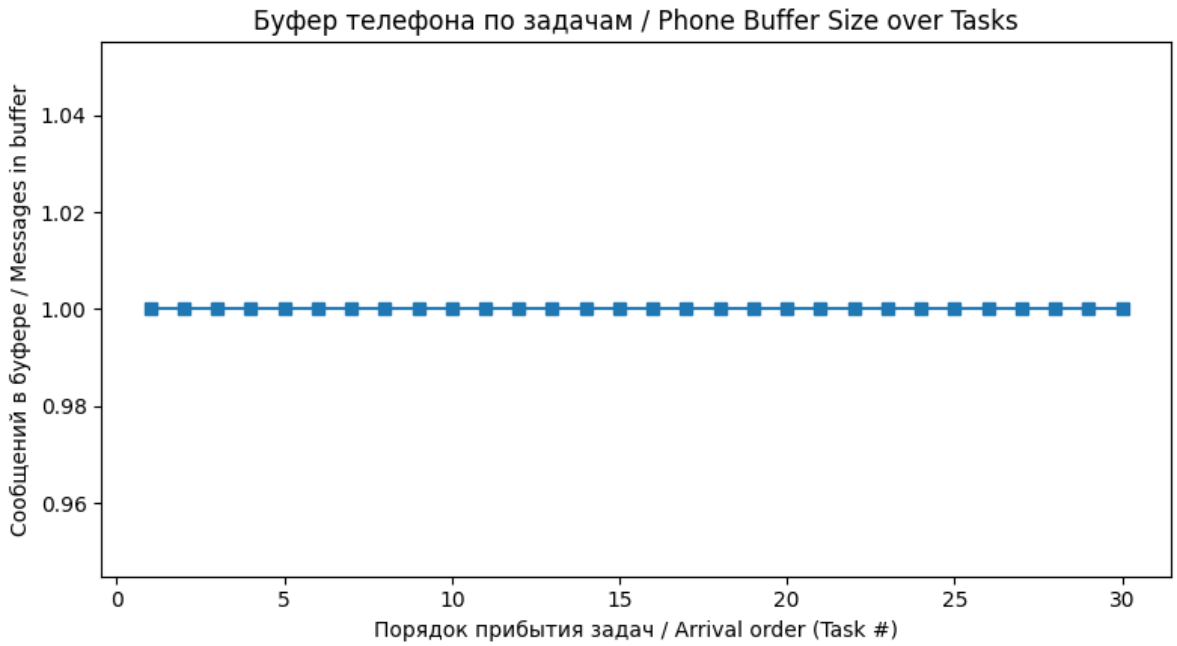

Рисунок 6 – Оптимальное значение для буфера телефона по задачам

6 СОСТАВЛЕНИЕ ОТЧЁТА

Заполните отчёт по выполнению лабораторной работы. Сделайте выводы.