https://doi.org/10.26160/2474-5901-2025-48-42-48

РАЗРАБОТКА ПРОГРАММНОЙ СРЕДЫ ДЛЯ АНАЛИЗА ПОВЕДЕНИЯ EDGE-СИСТЕМ НА ОСНОВЕ WEBSOCKET

Хабаров С.П.
Санкт-Петербургский государственный лесотехнический университет им. С.М. Кирова, Санкт-Петербург, Россия

Кл**ючевые слова:** edge computing, WebSocket, распределённые системы, программная среда, задержка сообщений, мониторинг систем.
Аннотация. В статье описывается создание экспериментальной Edge-системы, предназначенной для анализа качества передачи данных в условиях ограниченных вычислительных ресурсов и нестабильного соединения. Система реализует модель взаимодействия сенсоров и клиентов через WebSocket/HTTP-сервер. Особое внимание уделено разработке модуля, используемого для сбора и анализа временных метрик: задержек, интервалов между сообщениями и аномалий. Представлена UML-диаграмма взаимодействия компонентов, обсуждаются выявленные проблемы, пути их решения.

DEVELOPMENT OF A SOFTWARE ENVIRONMENT FOR ANALYZING THE BEHAVIOR OF EDGE SYSTEMS BASED ON WEBSOCKET

Khabarov S.P.
Saint-Petersburg State Forest Technical University, Saint-Petersburg, Russia

Keywords: edge computing, WebSocket, distributed systems, software environment, message delay, system monitoring.
Abstract. The article describes the development of an experimental Edge system designed to analyze data transmission quality under limited computing resources and unstable network conditions. The system implements a sensor-client interaction model via a WebSocket/HTTP server. Special attention is given to the development of a module for collecting and analyzing temporal metrics, including message delays, intervals between messages, and anomalies. A UML diagram of component interactions is presented, and identified issues along with potential solutions are discussed.

Введение. Edge-вычисления становятся ключевой парадигмой построения современных распределённых систем, где обработка данных осуществляется близко к источнику. Такой подход снижает задержки и нагрузку на центральные серверы [1]. Для организации обмена данными в реальном времени в Edge-инфраструктуре всё чаще применяют протокол WebSocket, обеспечивающий двустороннюю передачу с минимальной латентностью, что делает его целесообразным выбором для приложений с частым обновлением данных [2, 3].

В рамках исследования была разработана тестовая Edge-система, реализующая модель "N сенсоров – M клиентов", где данные о состоянии объектов (температура, влажность) генерируются имитаторами датчиков и рассылаются через WebSocket-сервер всем подключённым клиентам, которые представлены веб-интерфейсами для визуализации текущих показаний сенсоров и алертов.

Диаграмма, представленная на рисунке 1, отражает ключевые аспекты функционирования системы: периодическую отправку данных сенсорами, HTTP-доставку интерфейса клиентам и WebSocket-обмен информацией.

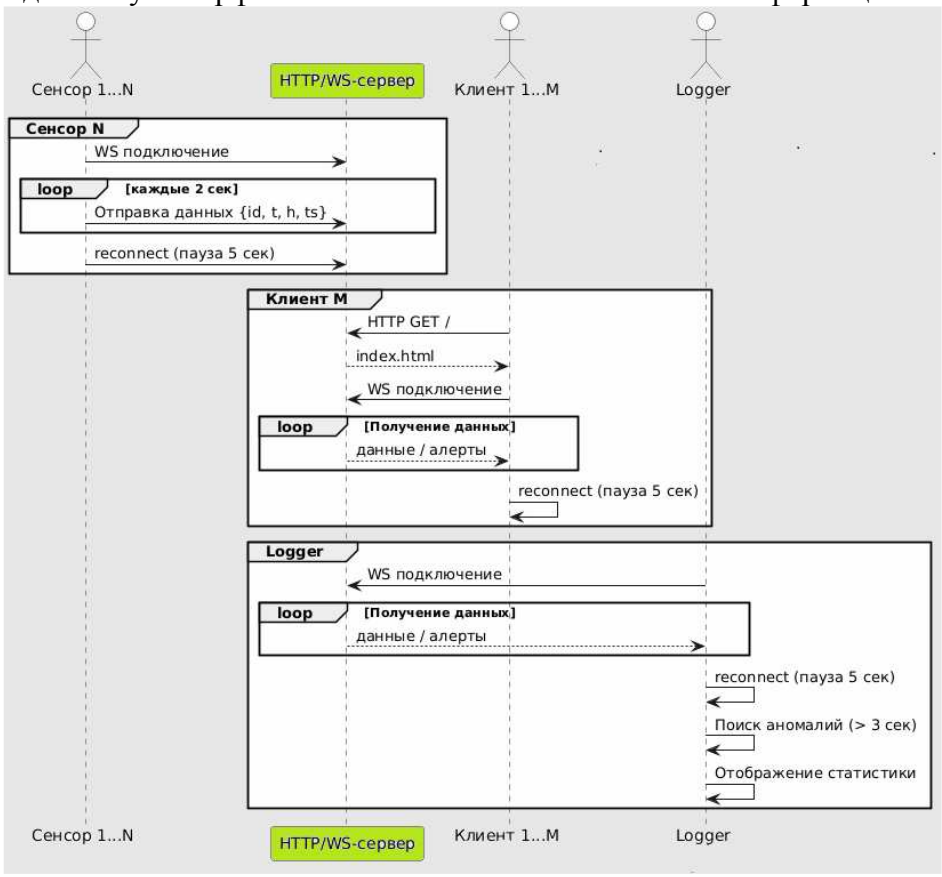

Рис. 1. Диаграмма взаимодействия компонентов Edge-системы

Постановка задачи. Исходная реализация Edge-системы позволяет отслеживать состояние сети в реальном времени, но не предоставляет количественной оценки ключевых метрик – задержки доставки сообщений, стабильности обмена данными и поведения при сетевых сбоях. Все это затрудняет анализ надёжности и производительности, особенно при тестировании на одном вычислительном узле, где возможны ресурсные конфликты и эффекты синхронизации.

Для устранения данного ограничения был разработан и внедрён исследовательский клиент logger.html, предназначенный исключительно для сбора временных метрик. Этот модуль работает параллельно с пользовательскими интерфейсами, не влияя на логику системы, и регистрирует следующие параметры: задержки доставки сообщений, интервалы между ними, аномалии (>3 с), а также события разрыва и восстановления соединений. Это позволило создать эмпирическую основу для анализа качества передачи данных и устойчивости системы к внешним воздействиям.

Инструментарий исследования. Для анализа поведения Edge-системы был разработан программный инструментарий, позволяющий регистрировать метрики доставки данных, моделировать нагрузку и воспроизводить сетевые сбои. Основу инструментария составляют:

1. Исследовательский клиент (logger.html). Он предназначен для диагностики состояния системы и сбора временных метрик. Работает независимо от клиентских интерфейсов, не влияет на логику работы Edge-сети, обеспечивая объективность измерений. Подключаясь к WebSocket-серверу, выполняет следующие функции:

– мониторинг соединения: при потере связи автоматически пытается восстановить подключение с задержкой в 5 секунд;

– приём и обработка сообщений: извлекает из JSON-сообщений идентификатор сенсора (id), временную метку (ts) и значения датчиков, определяет задержку как разницу между временем получения и отправки;

– обновление статистики: каждую секунду считает минимальную, максимальную и среднюю задержку, стандартное отклонение, количество аномалий (с задержкой >3 с) и выводит результаты на страницу.

UML-диаграмма последовательности (рис. 2) показывает, что logger.html работает независимо от пользовательских клиентов и собирает метрики без влияния на систему. Это позволяет использовать его для анализа производительности и диагностики Edge-архитектуры.

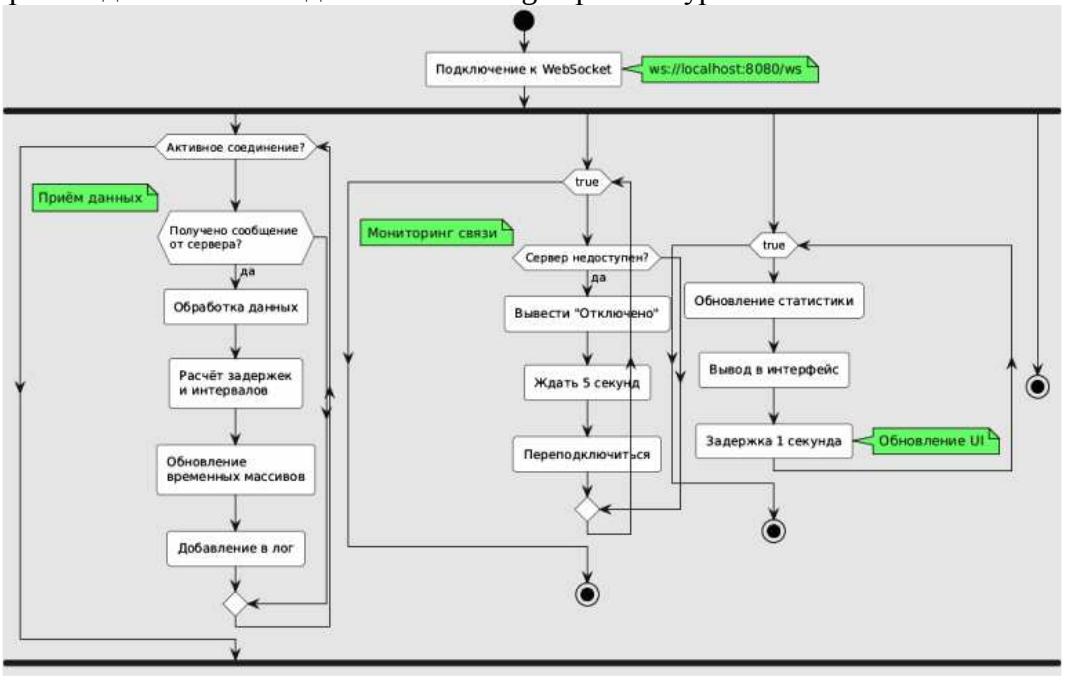

Рис. 2. Логика JScript обработки данных в модуле logger.html

2. Сценарий запуска системы (start_system.bat). Он предназначен для автоматического развертывания тестируемой Edge-системы, На языке командной строки Windows имеет вид:

```
@echo off
:: Запуск сервера
start "Server" cmd /k python ws_server.py
timeout /t 2 >nul
:: Количество запусков сенсоров
set N=10
for /L %%i in (1,1,%N%) do (
   start "Sensor %%i" cmd /k python sensor.py --id %%i
   timeout /t 1 >nul  )
:: Количество запусков копий msEdge
set COUNT=3
for /L %%i in (1,1,%COUNT%) do (
   start msedge --new-window http://127.0.0.1:8080
   timeout /t 1 >nul  )
```

Код последовательно запускает WebSocket-сервер (ws_server.py), тестируемое число сенсоров и веб-клиентов (экземпляров Microsoft Edge).

Он обеспечивает воспроизводимость условий эксперимента и строгий контроль начальной конфигурации системы перед началом измерений.

3. Скрипт циклического запуска сервера. Обеспечивает имитацию нестабильного сетевого окружения для оценки устойчивости системы к частым перезапуском сервера. Реализован в виде Python-скрипта, который периодически перезапускает сервер:

```
import time
import datetime

def log_event(message):
   timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
   with open("history.log", "a", encoding="utf-8") as f:
      f.write(f"{timestamp} - {message}\n")

def main():
   while True:
      log_event("Запуск сервера")
      proc = subprocess.Popen(["python", “ws_server.py”])
      time.sleep(20)
      proc.terminate()                  # отправляем сигнал завершения
      try:
            proc.wait(timeout=2)        # ждём, пока процесс завершится
      except subprocess.TimeoutExpired:
            proc.kill()                 # если не завершился, убиваем жестко
      log_event("Сервер остановлен")
      time.sleep(2)                     # пауза перед следующим запуском

 main()
```

При каждом цикле сервер запускается, и работает 20 секунд, после чего принудительно завершается. С паузой в 2 сек. процесс повторяется. Все события логируются в файл history.log. Этот инструмент позволяет проверить способность клиентов, включая logger.html, восстанавливать соединение и продолжать сбор данных после кратковременных сбоев.

Результаты исследования. Полученные данные подтверждают работоспособность разработанного инструментария и его пригодность для анализа поведения Edge-системы. Модуль logger.html обеспечил сбор ключевых метрик: задержек доставки сообщений, периодичности обмена данными и событий потери соединения. Это позволило не только оценить качество функционирования системы, но и выявить её слабые места.

В одном из тестов модуль logger.html регистрировал время передачи пакетов, которые генерировались 10 сенсорами с частотой 1 секунда, обрабатывались WebSocket-сервером и передавались клиентам (рис. 3).

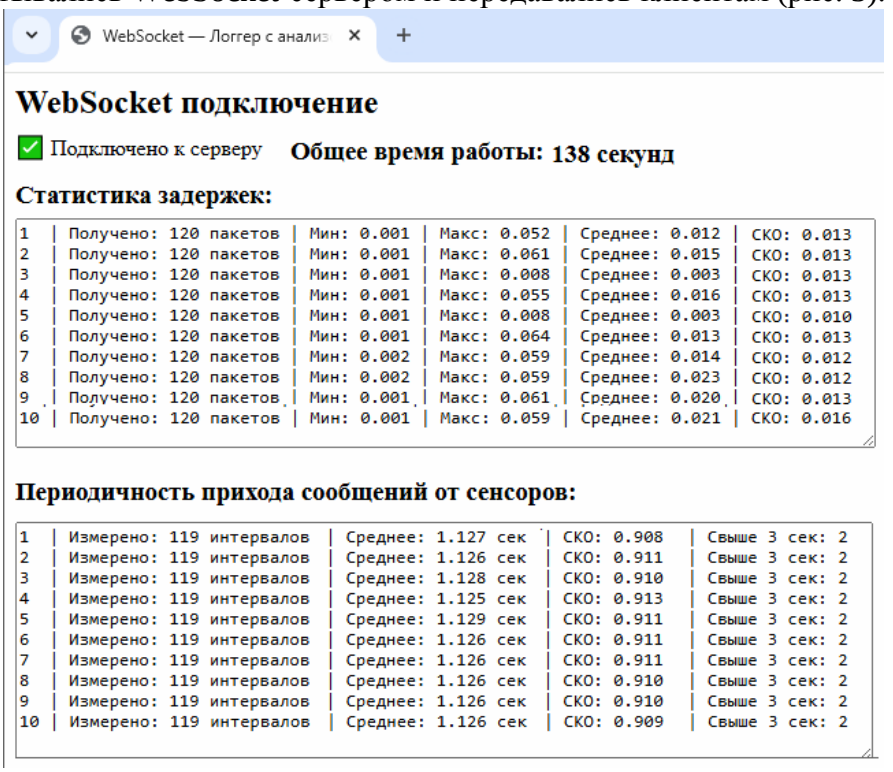

Рис. 3. Статистка, полученная модулем logger.html

1. Анализ задержки доставки сообщений. В ходе эксперимента были зафиксированы временные задержки между отправкой сообщения сенсором и его получением клиентом. Средняя задержка составила 0,015 ± 0,013 секунды, максимальная – не более 0,064 секунды. Низкие значения стандартного отклонения свидетельствуют о стабильной работе системы в большинстве случаев. Такие показатели находятся в допустимых пределах для большинства Edge-приложений, требующих обработки данных в реальном времени.

2. Периодичность приёма сообщений. Анализ интервалов между последовательными сообщениями показал, что средний период равен 1,126 секунды, что близко к заявленному интервалу отправки сенсоров. Однако наблюдается значительное стандартное отклонение (около 0,91 секунды), а также 2 случая аномальных интервалов. Эти события совпадают по времени у всех сенсоров, что указывает на глобальные сбои в системе, а не на локальные ошибки отдельных компонентов.

Факт синхронного возникновения аномалий позволяет исключить вероятность аппаратных или программных сбоев в отдельных сенсорах. Вместе с тем, это указывает на наличие проблем в центральном элементе системы – WebSocket-сервере или в условиях выполнения процессов на общем вычислительном ресурсе.

3. Устойчивость к сетевым сбоям. Для тестирования устойчивости системы к кратковременным перебоям связи был применён скрипт принудительного перезапуска сервера каждые 20 секунд. В результате:

– модуль logger.html корректно обнаруживал разрывы соединения,

– восстанавливал связь спустя 5 секунд после каждого отключения,

– продолжал сбор метрик без потери целостности данных.

Это подтверждает надёжность реализации исследовательского клиента и его способность функционировать в условиях нестабильной сети, что особенно важно для Edge-систем, работающих в сложных внешних условиях.

4. Анализ вычислительной нагрузки. Тестирование проводилось на единой вычислительной платформе, где сразу запускались: WebSocket-сервер (ws_server.py), несколько экземпляров имитаторов сенсоров (sensor.py), пользовательские клиенты (index.html) и исследовательский модуль (logger.html).

Наблюдения показали: периодическое увеличение загрузки CPU и паузы сборщика мусора (GC) в браузере при обновлении интерфейса, Эти факторы приводят к внутрисистемным задержкам и аномальным интервалам между сообщениями. Таким образом, тестовая конфигурация выявила ресурсное ограничение на одном хосте, которое может служить источником нестабильности даже при корректной реализации протоколов передачи данных.

Выводы. В ходе исследования был разработан и апробирован программный инструментарий для эмпирического анализа поведения Edge-системы. Основным элементом стал исследовательский клиент logger.html, позволяющий регистрировать временные метрики доставки данных, выявлять аномалии и отслеживать устойчивость системы к сетевым сбоям. Вспомогательные скрипты обеспечили автоматизацию запуска и моделирование нагрузки.

Полученные в результате исследования данные подтвердили работоспособность инструментария и позволили выявить ключевые проблемы функционирования Edge-сети, включая ресурсные ограничения на общем хосте и синхронные сбои в передаче данных. Это демонстрирует его

пригодность для диагностики и оптимизации распределённых систем, построенных на базе WebSocket и работающих в условиях ограниченных ресурсов.

Список литературы

1. An Introduction to Edge Computing in the Era of Connected Devices – [Электронный ресурс]. – Режим доступа: https://www.cavliwireless.com/blog/nerdiest-of-things/edge-computing-for-iot-real-time-data-and-low-latency-processing#one.
2. Хабаров С.П., Заяц А.М. Использование технологии websocket в клиент-серверных экспертных системах // Леса России: политика, промышленность, наука, образование: Материалы второй МНТК. – СПб: СПбГЛТУ им. С.М. Кирова, 2017. – С. 278-280.
3. Хабаров С.П. Использование утилиты WebSocketD для удаленного выполнения программ // Информационные системы и технологии: теория и практика: Сборник научных трудов. Выпуск 9. – СПб: СПбГЛТУ им. С.М. Кирова, 2017. – С. 90-104.

References

1. An Introduction to Edge Computing in the Era of Connected Devices. [Electronic resource]. – Access mode: https://www.cavliwireless.com/ blog/nerdiest-of-things/edge-computing-for-iot-real-time-data-and-low-latency-processing#one.
2. Khabarov S.P., Zayats A.M. Using WebSocket Technology in Client-Server Expert Systems // Forests of Russia: politics, industry, science, education: Materials of the second ISTC. – SPb.: SPbSFTU, 2017. – P. 278-280.
3. Khabarov S.P. Using the WebSocketD Utility for Remote Program Execution // Information systems and technologies: theory and practice: Collection of scientific papers. Issue 9. – SPb.: SPbSFTU, 2017. – P. 90-104.

Хабаров Сергей Петрович – кандидат технических наук, старший научный сотрудник, доцент, доцент кафедры информационных систем и технологий	**Khabarov Sergey Petrovich** – candidate of technical sciences, senior researcher, associate professor, associate professor of the Department of information systems and technologies
Serg.Habarov@mail.ru	

Received 19.06.2025