mathematics & mechanics

Разработка библиотеки динамического, перераспределения потоков данных , мультимедиа, данных, на базе SRT протокола

Автор:

1 Дедов Георгий

Научный руководитель:

Таранцев Игорь Геннадьевич, доцент кафедры АФТИ ФФ

mathematics & mechanics

В II семестре 3 курса были поставлены следующие задачи научным руководителем:

- Разработать инструмент для контролируемой потери данных и отладки
- Изучить sockets c++
- Прочитать про cpt и сделать про него презентацию своими словами
- Разобраться в srt tcp и udp

План проекта:

Базовые обязательные компоненты:

Socket: класс для работы с сокетами, включая создание, привязку, прослушивание и соединение.
SRTReceiver: класс для получения потоков данных.
SRTTransmitter: класс для отправки потоков данных.
StreamManager: класс для управления потоками и их перераспределением.
Buffer: класс для хранения и обработки данных.

Дополнительные компоненты:

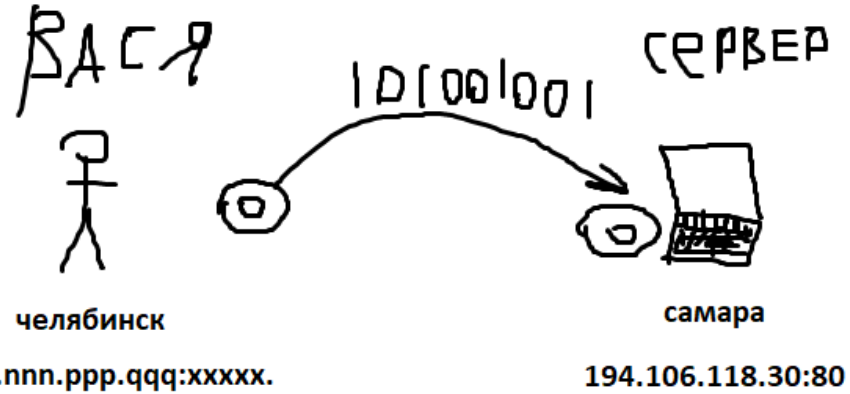
FileReceiver и **FileTransmitter**: классы для передачи файлов.
MessageReceiver и **MessageTransmitter**: классы для передачи текстовых сообщений.
TwoWayCommunication: класс для двунаправленной передачи данных.

Что такое SRT?

SRT (Secure Reliable Transport) — это протокол передачи данных, разработанный для обеспечения надежной и безопасной доставки аудио- и видеоконтента через нестабильные сети, такие как интернет.

Про сокеты и классы

Сокет сам по себе это абстрактный объект, представляющий конечную точку соединения.
Есть отправитель и есть получатель.
У каждого по своему сокету, которые имеют свои адрес и порт.
Пакет данных отправляется сокетом в сеть и забирается из сети другим сокетом, которому он был адресован.



Как SRT работает?

Классы в самом проекте:

1. **Receive:** представляет собой класс, который создает приемник для получения данных по сети с использованием протокола SRT. Он принимает поток данных через сокеты SRT и обрабатывает их. Когда приемник запущен, он ждет подключения клиентов и принимает данные от них. У него существует также реализованный аналог для обработки файлов: file-receive
2. **Send**: это класс, который создает передатчик для отправки данных по сети с использованием протокола SRT. Он соединяется с приемником через сокеты SRT и передает данные. Есть аналог для отправки файлов: file-send
3. **Forward**: позволяет клиенту и серверу обмениваться данными в обоих направлениях через установленное . Данные могут отправляться параллельно или асинхронно
4. **Buffer**: Хранилище данных, которые могут быть изменены
5. **Scheduler**: Планировщик задач и их хранилище в порядке очереди, позволяет добавлять задачи для выполнения через определенное время или запускать цикл таймера/контролировать выполнение задач по расписанию
6. **Route**: Класс маршрутизатора, который направляет данные с одного сокета на другой сокет.

Концепция инструмента для контролируемой потери данных и отладки

Изначальная концепция инструмента очень простая, я сразу же написал простой симулятор, который «моделирует» подобное поведение. С математической точки зрения мы просто теряем пакеты с фиксированной вероятностью на i-ом шаге

Концепция инструмента для контролируемой потери данных и отладки

Генератор случайных чисел выдаёт случайную величину, равномерно распределённую на интервале [a, b), то есть распределенную по функции плотности вероятности:

$$P(x|(a,b) = \frac{1}{b-a}$$

Таким образом мы получаем случайную величину от 0.000 до 1.000, которая обозначает вероятность. Если вероятность окажется в радиусе потери указанной при выполнении действия, то тогда пакет данных будет потерян.

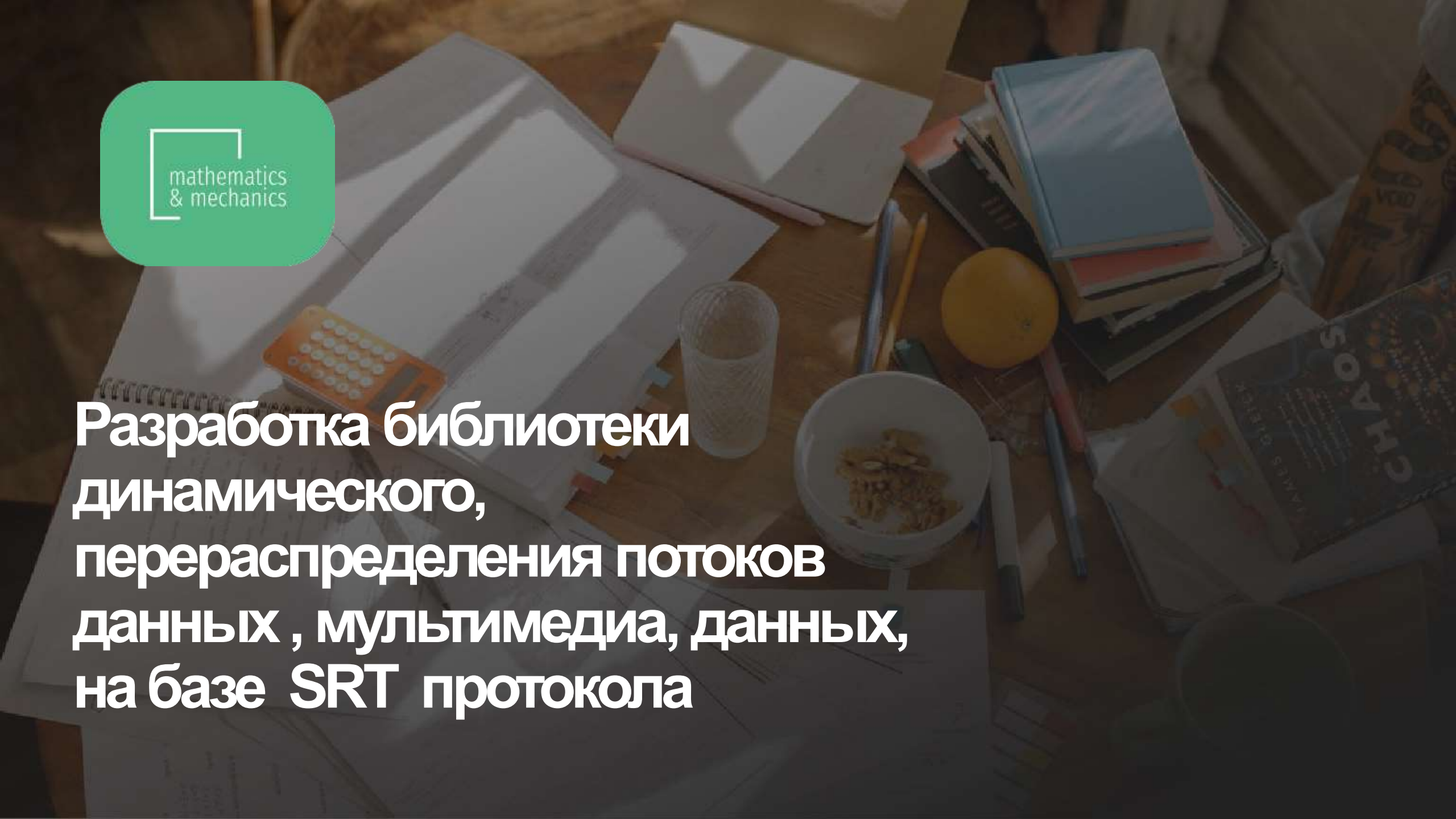
Псевдокод и то как выглядит в проекте

```
ПЕРЕМЕННЫЕ:
вероятность_потери = передать_с_конфига()
ПОКА не_завершено:
    пакет = принять_пакет_от_SRT()
    случайное_число =
генерировать_случайное_число_между(0, 1)
    ЕСЛИ случайное_число < вероятность_потери ТО
        пакет.пометить_как_потерянный()
```

```
try
{
    for (int i = 0; (num_messages < 0 || i < num_messages) && !force_break; ++i)
    {
        if (ratepacer)
        {
            ratepacer->wait(force_break);
        }

        // Проверка, соблюдена ли продолжительность отправки
        if (cfg.duration > 0 && (steady_clock::now() - start_time > seconds(cfg.duration)))
        {
            break;
        }

        // Генерация случайного числа для проверки потерь
        double random_value = distribution(generator);
        if (random_value < cfg.lossRate)
        {
            continue;
        }

        pldgen.generate_payload(message_to_send);

        target->write(const_buffer(message_to_send.data(), message_to_send.size()));
```

Что получилось в итоге?

1. Написан инструмент
2. Написана документация
3. Всё выложено на гитхаб



Разработка библиотеки динамического, перераспределения потоков данных , мультимедиа, данных, на базе SRT протокола