**Технический Университет Молдовы**

**Факультет вычислительной техники, информатики и микроэлектроники**

**Департамент Программной Инженерии и Автоматики**

**Лабораторная работа №1**

по дисциплине: «Программирование распределенных приложений»

на тему: «Message Broker»

Выполнили: ст. гр. TI-185 Жуля Артём

Пласкин Андрей

Козленко Юрий

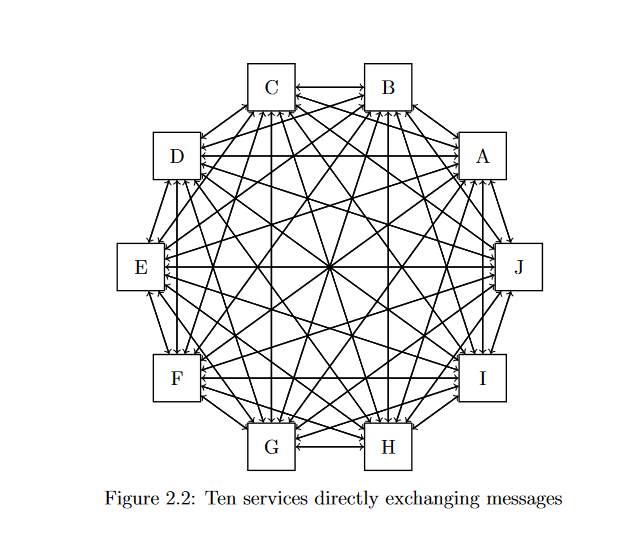
Проверил: Fiștic Cristofor

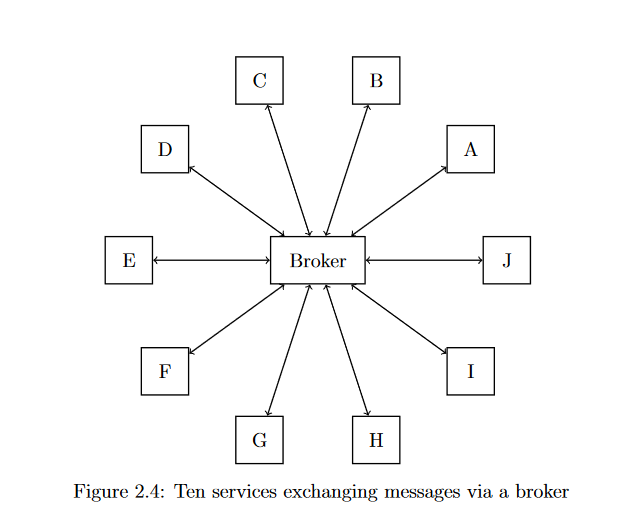
Кишинёв-2021

**Брокер сообщений** — архитектурный паттерн в распределённых системах; приложение, которое преобразует сообщение по одному протоколу от приложения-источника в сообщение протокола приложения-приёмника, тем самым выступая между ними посредником. Кроме преобразования сообщений из одного формата в другой, в задачи брокера сообщений также входит:

1. проверка сообщения на ошибки;
2. маршрутизация конкретному приемнику(ам);
3. разбиение сообщения на несколько маленьких, а затем агрегирование ответов приёмников и отправка результата источнику;
4. сохранение сообщений в базе данных;
5. вызов веб-сервисов;
6. распространение сообщений подписчикам, если используются [шаблоны типа издатель-подписчик](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C-%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%87%D0%B8%D0%BA_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Издатель-подписчик (шаблон проектирования)).

Использование брокеров сообщений позволяет разгрузить веб-сервисы в распределённой системе, так как при отправке сообщений им не нужно тратить время на некоторые ресурсоёмкие операции типа маршрутизации и поиска приёмников. Кроме того, брокер сообщений для повышения эффективности может реализовывать стратегии упорядоченной рассылки и определение приоритетности, балансировать нагрузку и прочее.





**Сокет —** название [программного интерфейса](https://ru.wikipedia.org/wiki/API" \o "API) для обеспечения обмена данными между [процессами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o "Процесс (информатика)). Процессы при таком обмене могут исполняться как на одной [ЭВМ](https://ru.wikipedia.org/wiki/%D0%AD%D0%92%D0%9C" \o "ЭВМ), так и на различных ЭВМ, связанных между собой [сетью](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C" \o "Компьютерная сеть).

Следует различать [клиентские](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o "Клиент (информатика)) и [серверные](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)" \o "Сервер (программное обеспечение)) сокеты. Клиентские сокеты грубо можно сравнить с конечными аппаратами [телефонной сети](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C_%D0%BE%D0%B1%D1%89%D0%B5%D0%B3%D0%BE_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F" \o "Телефонная сеть общего пользования), а серверные — с [коммутаторами](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D1%82%D0%B0%D1%82%D0%BE%D1%80" \o "Телефонный коммутатор). Клиентское приложение (например, [браузер](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80" \o "Браузер)) использует только клиентские сокеты, а серверное (например, [веб-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80" \o "Веб-сервер), которому браузер посылает запросы) — как клиентские, так и серверные сокеты.

[Интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Интерфейс) сокетов впервые появился в [BSD Unix](https://ru.wikipedia.org/wiki/BSD" \o "BSD). [Программный интерфейс](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Программный интерфейс) сокетов описан в стандарте [POSIX](https://ru.wikipedia.org/wiki/POSIX" \o "POSIX).1 и в той или иной мере поддерживается всеми современными [операционными системами](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Операционная система).

**gRPC** — это система [удалённого вызова процедур](https://ru.wikipedia.org/wiki/%D0%A3%D0%B4%D0%B0%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80" \o "Удалённый вызов процедур) (RPC) с открытым исходным кодом, первоначально разработанная в [Google](https://ru.wikipedia.org/wiki/Google" \o "Google) в 2015 году. В качестве транспорта используется [HTTP/2](https://ru.wikipedia.org/wiki/HTTP/2" \o "HTTP/2), в качестве языка описания интерфейса — [буферы протоколов](https://ru.wikipedia.org/wiki/Protocol_Buffers" \o "Protocol Buffers). gRPC предоставляет такие функции как [аутентификация](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F" \o "Аутентификация), двунаправленная потоковая передача и управление потоком, блокирующие или неблокирующие привязки, а также отмена и тайм-ауты. Генерирует кроссплатформенные привязки клиента и сервера для многих языков. Чаще всего используется для подключения служб в микросервисном стиле архитектуры и подключения мобильных устройств и браузерных клиентов к серверным службам.

Сложное использование HTTP/2 в gRPC делает невозможным реализацию клиента gRPC в браузере - вместо этого требуется прокси.

Выбрали протокол TCP, потому что не хотим терять пакеты. И быть уверенным, что все сообщения дойдут до получателя.

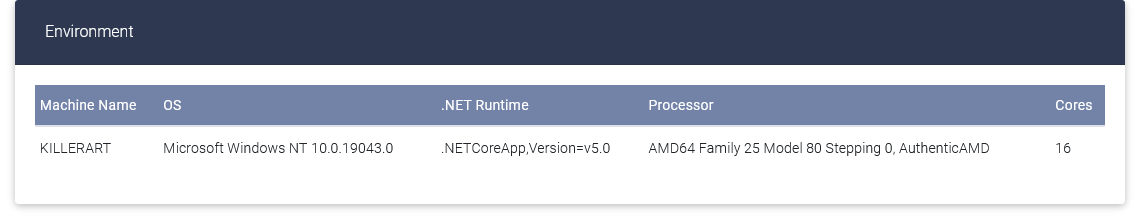
Сравнение socket и gRPC:

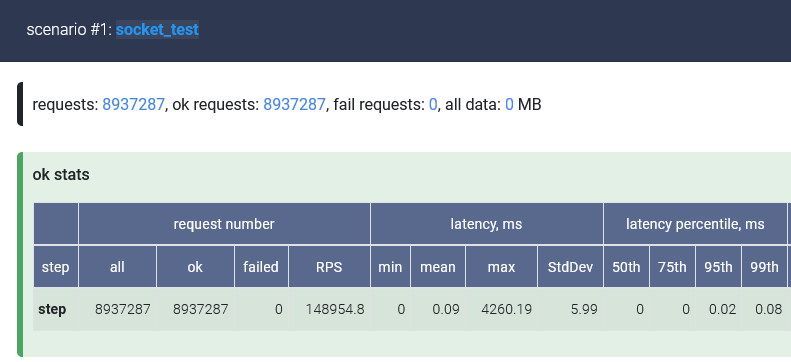
В сокетах нужно придумать свой протокол обработки пакетов и сообщений. Для этого используется много памяти и времени на парсинг сообщения.

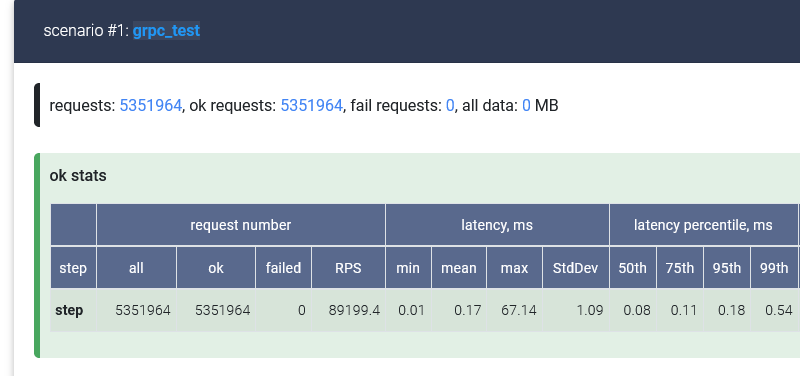
gRPC же имеет заранее заданные в proto-файлах виды сообщний и виды удаленных процедур, что сильно упрощает вызов процедур и парсинг сообщений.

Тесты:

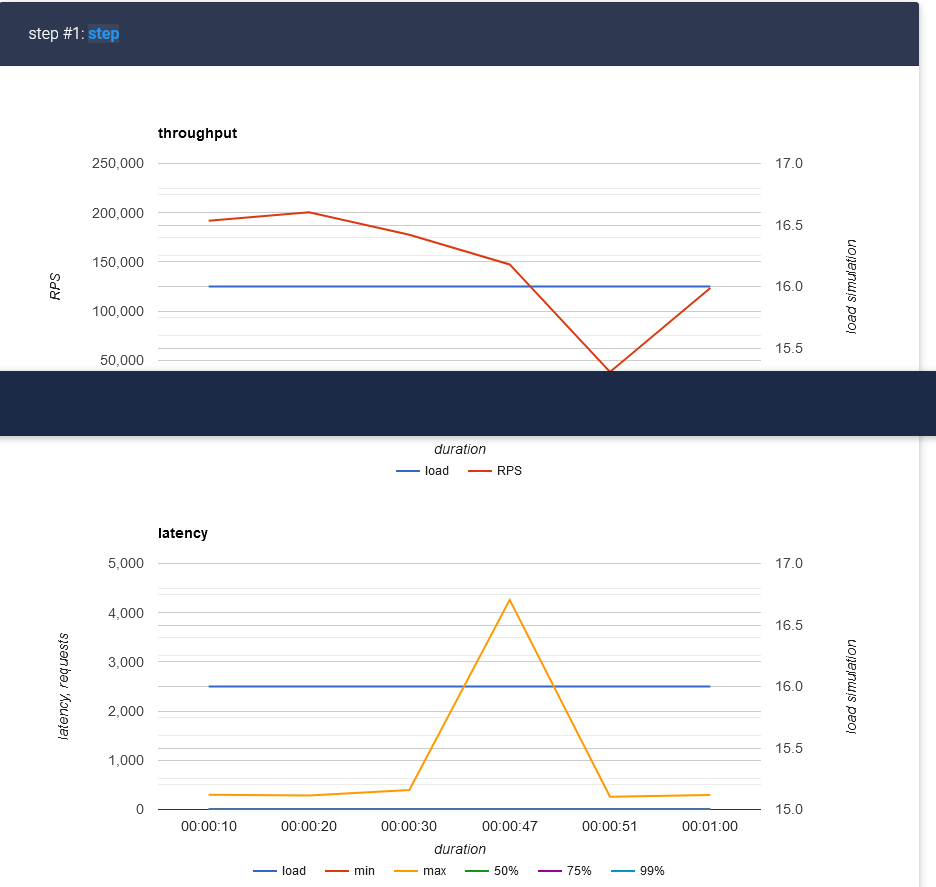
Машина для тестирования: Ryzen 9 5900HX (8 cores, 16 threads), 16GB RAM







socket



gRPC



На графиках видно, что пропускная способность у TCP-сокетов выше, чем у gRPC, но максимальная задержка ответов на запрос у сокетов выше, из-за сборщика мусора и сложности парсинга сообщений. У gRPC сообщения имеют заранее заданную в proto-файле форму, что упрощает и ускоряет парсинг сообщений.

Ссылка на проект на GitHub:

https://github.com/killerart/PAD-Lab1