Система Передачи Мгновенных Сообщений с Шифрованием Данных

Дмитрий Русу, Дориан Саранчук, Татьяна Скороходова
Департамент Программной Инженерии и Автоматики
Технический Университет Молдовы
Кишинев, Республика Молдова
dumitru.rusu@ati.utm.md
dorian.saranciuc@ati.utm.md
tatiana.scorohodova@ati.utm.md

***Abstract*— In this paper the software for instant messaging with encription is presented. The analysis and comparison of the existing products on the market is done. The requirements for the system being developed are formulated.The architecture of the developed system and the advantages of such systems over other methods of information transmission are described.**

***Index Terms*— Messaging, mobile, transfer system, Android, open source**

I. Введение

На сегодняшний день, когда компьютерные технологии проникают во все сферы человеческой деятельности, в том числе учебные заведения, они переходят на использование компьютерных технологий для автоматизации своей работы, так как преимущества от этого очевидны: это и скорость обработки данных, и расширяемость функций, и гибкость индивидуальной настройки. Для всего этого используется программное обеспечение, которое и выполняет часть функций человека, заменяет бумажные хранилища данных и ускоряет работу.

В современном обществе информационные технологии прочно вошли в жизнь каждого человека. Во всем мире наблюдается бурное развитие средств и технологий, связанных с обменом информацией. В последние годы темпы роста телекоммуникационного рынка увеличились на 30-40% и можно ожидать, что тенденция сохраниться и в среднесрочной перспективе. Сегодня в обществе «информационного бума» завоевали важное значение различные технологии информационного обмена. Реалии времени диктуют все ускоряющийся темп обмена информацией и ведут к появлениею новых технологий. Идея создания сервиса обмена короткими текстовыми сообщениями (Short Message Service) возникла еще в 1984 году, а первое SMS-сообщение было отправлено в 1992 в сотовой сети Vodafone.

Сегодня же, для личной переписки люди пользуются различными мессенджерами, которые для связи используют Интернет. Уже привычный метод SMS, хоть и остается популярным способом связи, но свои позиции сдает достаточно быстро. За последние несколько лет появилось много приложений, позволяющих пользователям не просто переписываться между собой текстом, но и общаться по видеосвязи, обмениваться файлами, создавать групповые чаты и прочее. Разрабатываемое нами средство обмена мгновенными сообщениями ставит своей задачей предоставить пользователям удобный сервис и защиту их конфиденциальной информации. Целью данного проекта является разработка системы, отличающейся от своих конкурентов и являющейся полностью открытой. Преимуществами открытого (Open Source) проекта [1] перед проприетарным программным обеспечением (ПО) является следующее: возможность быстро адаптироваться; время вывода продукта на рынок значительно ниже; над open-source проектами работают энтузиасты.

Приложения для обмена мгновенными сообщениями обладают рядом преимуществ:

- Мобильный номер, используемый для идентификации пользователя в приложении, является гораздо более надёжным и долгосрочным контактом, чем электронная почта, т.к. её проще сменить;
- Мгновенные push-уведомления. Можно быть уверенным, что уведомление дойдёт до клиента и будет прочитано;
- Приватность и персональность — в отличие от переписки с клиентами в соц. сетях или на открытых площадках, мессенджеры обеспечивают достаточный уровень приватности для личных обращений;
- Разнообразный контент. Мессенджеры позволяют делиться не только текстом, но и фото, видео, геопозицией. Также через них можно звонить, причём бесплатно.

II. Анализ Рынка

Наиболее близким по концепции аналогом разрабатываемой системы является приложение Telegram.

Telegram – это бесплатный кроссплатформенный мессенджер для смартфонов и других устройств, позволяющий обмениваться текстовыми сообщениями и медиа файлами различных форматов (Рис. 1).

Рис. 1. Telegram на платформе Android

Telegram позиционирует себя как защищенный мессенджер для обмена текстовыми сообщениями (Рис. 2). Для шифрования данных используется криптографический протокол MTProto [2], разработанный компанией Telegram. Основным недостатком Telegram-а в плане информационной безопасности является то, что по умолчанию end-to-end шифрование [3] отключено и для каждого чата его нужно включать вручную. Также, во многих странах он запрещен на законодательном уровне из-за нежелания сотрудничать с правоохранительными органами.

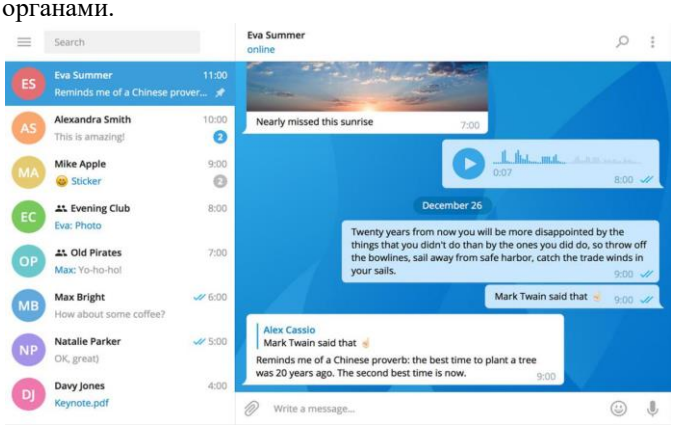

Рис. 2. Telegram на PC

Главным отличием Telegram-а от своих конкурентов являются боты. Боты в Telegram - это разновидность чат-ботов. По своей сути - это те же пользовательские аккаунты, которыми вместо людей управляют программы. Боты помогают выполнять разные действия: переводить и комментировать, обучать и тестировать, искать и находить и т.д. Взаимодействие с ботом происходит посредством текстовых команд. Любой желающий может создать свои боты, которые будут выполнять необходимые действия.

Несмотря на то, что Telegram считается самым защищенным приложением для передачи сообщений и медиа файлов, это не совсем так. Приватные ключи, необходимые для расшифровки содержания сообщений, хранятся в 3-х разных странах, и при желании они могут быть предоставлены соответствующим органам.

III. Требования к Разрабатываемой системе

После анализа конкурентов на рынке и сферы в целом были сформированы следующие требования к разрабатываемой системе:

- Система должна содержать необходимую информацию, механизм своевременной актуализации содержания и базовый набор сервисов работы с информацией, обеспечивающий требуемую полноту информационных и иных услуг, предоставляемых пользователю;
- Структура представления информационных ресурсов и пользовательские интерфейсы по доступу к ресурсам и сервисам должны быть интуитивно понятны широкому кругу пользователей;
- Разработка методов мониторинга состояния, использования и периодичности обновлений информационных ресурсов и сервисов системы, а также сбора статистических данных в динамичной содержательной части отдельных разделах, и обеспечения удобства их представления в Интернете;
- Необходимость включения в инфраструктуру системы механизмов поддержания постоянного диалога и взаимодействия между пользователями системы по актуальным темам и вопросам;
- Система должна обеспечивать конфиденциальность пользовательских данных;
- Базовое программное обеспечение системы должно быть проверено на отсутствие известных уязвимостей к атакам на отказ и на несанкционированный доступ;
- Приватные ключи хранятся только на устройствах самих пользователей;
- При переустановке приложения приватный ключ пользователя меняется, что означает что старые сообщения более не могут быть расшифрованы и будут удалены.

IV. Реализация Системы

Важной частью всей системы является хранилище, которое в нашем случае представляет собой REST-сервис. REST (Representational State Transfer — «передача состояния представления») – это набор архитектурных принципов и стиль проектирования приложений, ориентированный на создание сетевых систем, в основе которых лежат механизмы для описания и обращения к ресурсам. Примером такой системы может служить World Wide Web. В REST определяется строгое разделение ответственности между компонентами клиент-серверной системы, облегчающее реализацию необходимых актеров (actors). Другой целью REST является упрощение семантики взаимодействия компонентов сетевых систем, что позволяет улучшить масштабируемость и повысить производительность.

В основу REST заложен принцип автономности запросов, означающий, что запросы, обрабатываемые клиентом или сервером, должны включать всю контекстную информацию, необходимую для их понимания. При работе REST-систем для обмена данными стандартных медиа-типов используется минимальное количество запросов. REST-системы используют URI (универсальные идентификаторы ресурсов) для поиска и получения доступа к представлениям необходимых ресурсов. В течении последних нескольких лет разработчики создавали REST сервисы для своих .NET-приложений, используя самые разнообразные технологии. Архитектура REST отличается своей простотой, требуя от приложений обеспечить только возможность приема сообщений с HTTP заголовками. Эта функция легко реализуется простыми контроллерами в шаблоне MVC.

Для моделирования работы системы использован язык UML, который является стандартным инструментом для создания «чертежей» программного обеспечения.

На Рис. 3 изображена диаграмма развертывания разрабатываемой системы. Клиентское приложение представляет собой пакет, установленный в системе под управлением Android, который реализует интерфейс, определенный серверной частью и протоколом передачи данных. Веб сервер представляет собой приложение, работающее на удаленной машине. Он основан на архитектуре MVP и разделен на 3 слоя. Для хранения данных используется система управления базой данных PostgreSQL, которая находится на другом сервере. Взаимодействие между веб-сервером и базой данных происходит посредством JDBC соединения.

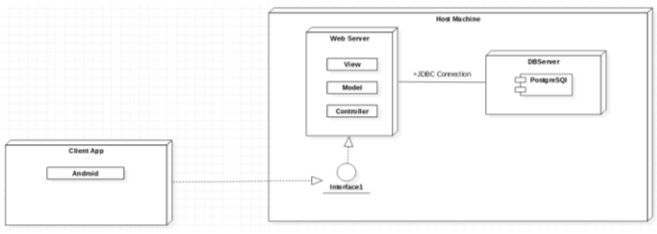

Рис. 3. Диаграмма развертывания системы

На Рис. 4 изображена Use case диаграмма взаимодействия 2-х пользователей посредством системы обмена мгновенными сообщениями. Пользователь может посредством установленного на телефоне приложения создать новую группу, подключиться к удалённому серверу для получения информации, присоединиться к уже существующей группе. После выбора одного из перечисленных действий пользователь может принимать и отправлять сообщения другим пользователям из того же чата [4].

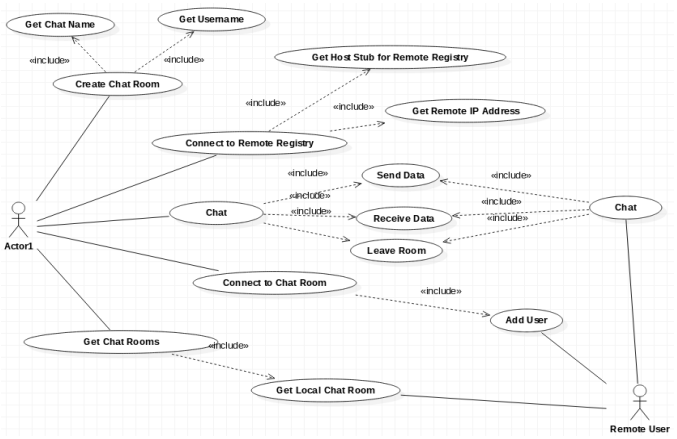

Рис. 4. Диаграмма взаимодействия пользователей

На Рис. 5 изображена диаграмма последовательности, описывающая процесс добавления другого пользователя в друзья. Только пользователи, находящиеся друг у друга в друзьях, могут общаться между собой, видеть статусы друг друга (онлайн/офлайн). После отправки предложения о дружбе принимающая сторона вправе отклонить запрос либо же подтвердить его и продолжить общение. После подтверждения либо после отказа пользователь, отправивший запрос, будет уведомлен о результате операции.

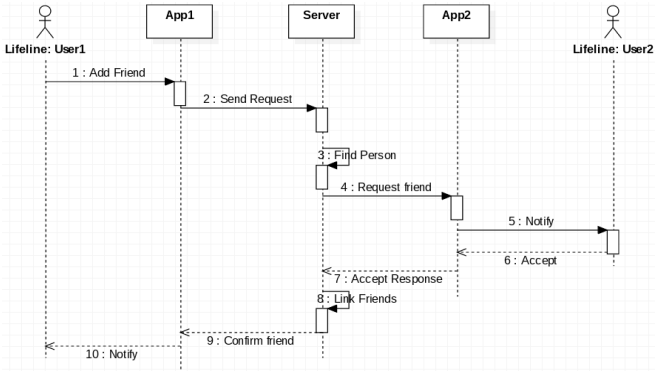

Рис. 5. Диаграмма последовательности добавления пользователя

В качестве базы данных используется NoSQL база данных под названым Tarantool [5]. В некотором смысле, Tarantool можно сравнить с Nginx, который разрабатывался в Rambler, и только через некоторое время

обрел заслуженную славу и признание сообщества, будучи открытым проектом.

Tarantool — это open-source проект. Исходный код открыт для всех и распространяется бесплатно согласно лицензии BSD license. Поддерживаемые платформы: GNU/Linux, Mac OS и FreeBSD.

Создателем Tarantool'а, а также его основным пользователем, является компания Mail.Ru Group— крупнейшая Интернет-компания России (30 млн пользователей, 25 млн электронных писем в день, веб-сайт в списке top 40 международного Alexa-рейтинга). Tarantool используется для обработки самых «горячих» данных Mail.Ru, таких как данные пользовательских онлайн-сессий, настройки онлайн-приложений, кеширование сервисных данных, алгоритмы распределения данных и шардинга, и т.д. Tarantool также используется во всё большем количестве проектов вне стен Mail.Ru. В Tarantool включены патчи от большого числа сторонних разработчиков. Усилиями сообщества разработчиков Tarantool'а были написаны (и далее поддерживаются) библиотеки для подключения модулей на внешних языках программирования. А сообщество Lua-разработчиков предоставило сотни полезных пакетов, большинство из которых можно использовать в качестве расширений для Tarantool'а.

API для функционала СУБД позволяет хранить Lua-объекты, управлять коллекциями объектов, создавать и удалять вторичные ключи, делать атомарные изменения, конфигурировать и мониторить репликацию, производить контролируемое переключение при отказе (failover), а также исполнять код на Lua, который вызывается событиями в базе. А для прозрачного доступа к удаленным (remote) экземплярам баз данных разработан API для вызова удаленных процедур.

В архитектуре серверной части СУБД Tarantool'а реализована концепция «движков» базы данных (storage engines), где в разных ситуациях используются разные наборы алгоритмов и структуры данных. В Tarantool'е есть два встроенных движка: in-memory движок, который держит все данные и индексы в оперативной памяти, и двухуровневый движок для B-деревьев, который обрабатывает данные размером в 10-1000 раз больше того, что может поместиться в оперативной памяти. Все движки в Tarantool'е поддерживают транзакции и репликацию, поскольку они используют единый механизм упреждающей записи (WAL = write ahead log). Это механизм обеспечивает согласованность и сохранность данных при сбоях. Таким образом, изменения не считаются завершенными, пока не проходит запись в лог WAL. Подсистема логирования также поддерживает групповые коммиты.

Tarantool поддерживает работу с составными ключами в индексах (Рис. 6). Возможные типы ключей: HASH, TREE, BITSET и RTREE. Также, Tarantool поддерживает асинхронную репликацию — как локальную, так и на удаленных серверах. При этом, репликацию можно настроить по принципу мастер-мастер, когда несколько узлов могут не только обрабатывать входящую нагрузку, но и получать данные от других узлов.

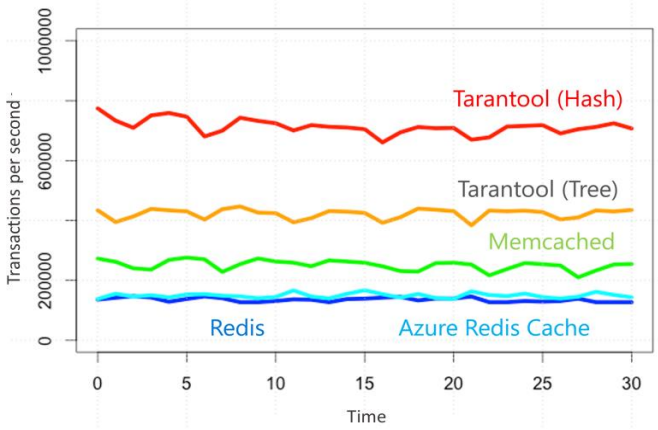

Рис. 6. Анализ скорости обработки транзакций различных систем

Для создания новой таблицы используется метод create из пакета box.schema.space. В качестве аргументов метод принимает названия полей, их тип и тип индекса. В зависимости от потребностей, типы индексов варьируются.

```
1. messages = box.schema.space.create('messages', {id=
   6})
2.    messages:create_index('primary', {parts = {1, 'strin
   g'}})
3.    messages:create_index('status', {type='tree', unique
    = false, parts = {3, 'string', 4, 'unsigned'}})
4.    messages:create_index('sender_id', {type='tree', un
   ique = false, parts = {2, 'string'}})
5.    messages:create_index('receiver_id', {type='tree', u
   nique = false, parts = {3, 'string'}})
```

Все изменения в Tarantool происходят при помощи функций, написанных на языке программирования Lua. Использования полноценного языка программирования добавляет гибкости и позволяет писать запросы к БД любой сложности.

```
1. function addUser(phone)
2.    return box.space.profile:insert({phone, {}, {}, 0})
3. end
```

Для передачи данных используется протокол под названием MQTT (Message Queuing Telemetry Transport) – протокол для передачи последовательности сообщений с телеметрическими данными.

Основные черты протокола MQTT:

- Обмен сообщениями происходит по принципу "издатель-подписчик" (Pub-Sub), размер заголовка сообщения составляет 2 байта, а полезная нагрузка может варьировать от 1 байта до 260 Мбайт;
- В протоколе заложена возможность выбора одного из трех уровней обслуживания.

Отличительной особенностью принципа "издатель-подписчик" от клиент-серверного подхода является то, что клиенты, посылающие сообщения (издатели, Publisher), и клиенты, принимающие сообщения (подписчики, Subscriber), как правило, разделены. Разделение может быть организовано в трех плоскостях:

- Пространство – издатель и подписчик не обязаны знать друг друга;
- Время – издатель и подписчик не должны быть включены в одно и то же время;
- Синхронизация – операции на обеих сторонах не должны приостанавливаться в течение публикации или получения информации.

Издатель и подписчик не передают друг другу сообщения напрямую, не устанавливают прямой контакт, могут не знать о существовании друг друга. Координирует и управляет передачей сообщений от издателя к подписчику и от подписчика к издателю брокер (Broker). Распараллеливание операций на брокере является второй важной особенностью принципа взаимодействия "издатель-подписчик".

Для реализации такой функции необходимы как минимум три типа клиентов и один брокер, обслуживающий этих клиентов с использованием протокола MQTT (Рис. 7).

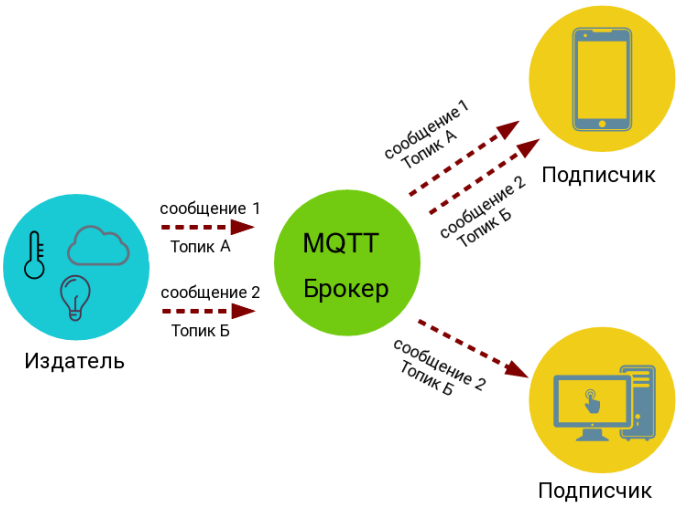

Рис. 7. Принцип работы протокола MQTT

Для отправки сообщений по протоколу MQTT необходимо в первую очередь подключиться к брокеру. В настройках подключения указывается имя пользователя и пароль, остальные значения являются опциональными. В случае неполадки при попытке подключиться к брокеру ошибка будет записана в лог-файл.

```
1.    public void connect() {
2.        try {
3.            MqttConnectOptions options = new
    MqttConnectOptions();          String brokerUser =
    config.getProperty("mqttUser");
    options.setUserName(brokerUser);
    options.setPassword(generateJwt().toCharArray());
    options.setAutomaticReconnect(true);
    options.setCleanSession(true);
    mqttClient.connect(options);
4.        } catch (Exception e) {
5.            LOG.error("Could not connect to mqtt
    broker", e);
6.            throw new RuntimeException("Could not
    connect to mqtt borker", e);
7.        }
8.    }
```

Для отправки сообщения нужно указать topic в который будет отправлено сообщение. Все сообщения для пользователей публикуются в topic '/message/user_phone', где user_phone – телефонный номер пользователя, под которым он авторизовался в приложении. Само сообщение будет передано в формате Json, и впоследствии десериализованно в клиентском приложении в соответствующую структуру. Необходимо также указать QoS о котором было подробно расписано выше.

```
1.  public void publishMessage(String recipient,
    IncomingMessage message) {
2.        MqttMessage mqttMessage = new
    MqttMessage();
3.        byte[] payload = new
    Gson().toJson(message).getBytes();
4.        mqttMessage.setQos(0);
5.        mqttMessage.setPayload(payload);
6.        try {
7.            String topic = "message/" + recipient;
8.            LOG.debug(String.format("Publishing
    message %1$s to %2$s", message.id, topic));
9.            mqttClient.publish(topic, mqttMessage);
10.       } catch (Exception e) {
11.           LOG.warn("Could not publish message", e);
12.       }
13. }
```

Минимальный объем служебной информации, наличие классов обслуживания и иерархическая структура тем являются неоспоримыми достоинствами протокола MQTT, что подтверждается большим разнообразием как клиентского, так и серверного ПО, в том числе открытого ПО.

Быстрый доступ к данным позволяет пользователям приложения получать сообщения почти мгновенно. Благодаря использованию протокола MQTT и NoSql базы данных Tarantool, задержка между отправкой и получения сообщения сводится к минимуму. Использования данного стэка технологии позволяет так же быстро передавать аудио-фото файлы между клиентами приложения.

Экран чата разработанного Android приложения для передачи мгновенных сообщений с шифрованием данных показан на Рис. 8.

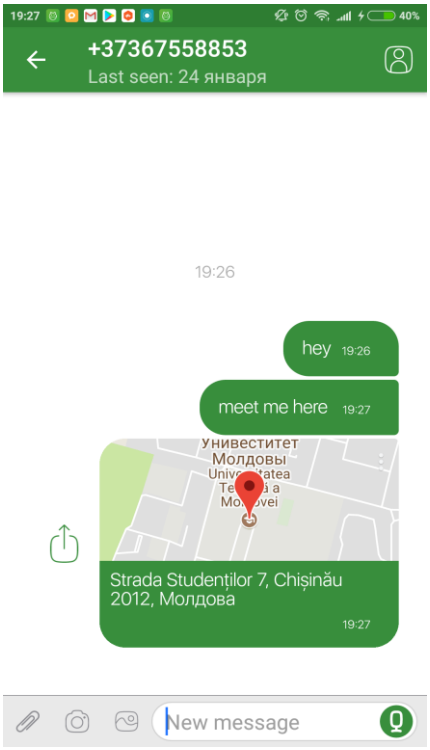

Рис. 8. Экран чата приложения для передачи мгновенных сообщений

V. Заключение

Разработана система передачи мгновенных сообщений с шифрованием данных.

В рамках проекта был проведен анализ основных сервисов обмена мгновенными сообщениями. Дано описание и общее сравнение крупнейших конкурирующих систем передачи мгновенных сообщений. Описаны их преимущества и недостатки.

Описаны возможности программы на пользовательском уровне и требования для серверной реализации. Для более глубокого понимания основ работы программы проведено исследование протоколов, используемых конкурентами. Проведен анализ взаимодействий клиент-сервер, приведены примеры с образцами дампов сетевых пакетов. Освещены вопросы безопасности, спама и хакерства в современных приложениях обмена мгновенными сообщениями.

Все полученные результаты явно указывают на важность проведения исследований систем мгновенных сообщений в целях развития перспективного направления коммуникации. Данная технология является передовой и многообещающей. Учитывая тенденции и ускорения темпов взаимодействия пользователей, необходимо уделять больше внимание изучению технологии мгновенных сообщений.

Литература

[1] Ron Goldman. Innovation Happens Elsewhere: Open Source as Business Strategy, MORGAN KAUFMAN PUBL, 2005.

[2] Telegram.org, Документация протокола MtProto [Электронный ресурс] – Главная страница [Режим доступа] https://core.telegram.org/mtproto.

[3] Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы построение протоколы. Учебник для вузов. 3-е издание. - СПб.: Питер, 2006.

[4] Вильям С. Компьютерные системы передачи данных. 6-е издание. - М.: Издательский дом «Вильямс», 2002.

[5] Tarantool.org, Документация БД Tarantool, [Электронный ресурс] – Главная страница [Режим доступа] https://tarantool.org/ru/doc/1.7/index.html