ISSN 2306-1561
Automation and Control in Technical Systems (ACTS)
2015, No 2, pp. 3-12.
DOI: 10.12731/2306-1561-2015-2-1

Data Transmission over the Network Using PROTOBUF Protocol

Ayham Mhd. Hailiam

Syrian Arab Republic, Undergraduate Student, Department of «Automated Control Systems».

State Technical University – MADI, 125319, Russian Federation, Moscow, Leningradsky prospekt, 64. Tel.: +7 (499) 151-64-12. http://www.madi.ru

ayham_achami@hotmail.ru

Andrey Borisovich Nikolaev

Russian Federation, Honoris Causa, Doctor of Technical Sciences, Professor, Dean of the Faculty «Control Systems».

State Technical University – MADI, 125319, Russian Federation, Moscow, Leningradsky prospekt, 64. Tel.: +7 (499) 151-64-12. http://www.madi.ru

nikolaev.madi@mail.ru

Abstract. In this article, we present a new Protocol for transferring data over the network, different from the standard, called Protobuf. Analyzed its features, reports examples of using it. A comparison of Protobuf data types and data types in other programming languages. Results of experiments comparing protocols and Protobuf, JSON.

Keywords: protocol, package, serialization, deserialization, message, data transmission network.

ISSN 2306-1561
Автоматизация и управление в технических системах (АУТС)
2015. – № 2. – С. 3-12.
DOI: 10.12731/2306-1561-2015-2-1

УДК 004.8

Передача данных по сети с использованием протокола PROTOBUF

Хайлям Мхд. Айхам

Сирийская Арабская Республика, магистрант кафедры «Автоматизированные системы управления».

ФГБОУ ВПО «Московский автомобильно-дорожный государственный технический университет (МАДИ)», 125319, Российская Федерация, г. Москва, Ленинградский проспект, д.64, Тел.: +7 (499) 151-64-12, http://www.madi.ru

ayham_achami@hotmail.ru

Николаев Андрей Борисович

Российская Федерация, Лауреат премии правительства РФ, Заслуженный деятель науки РФ, доктор технических наук, профессор, декан факультета «Управление».

ФГБОУ ВПО «Московский автомобильно-дорожный государственный технический университет (МАДИ)», 125319, Российская Федерация, г. Москва, Ленинградский проспект, д.64, Тел.: +7 (499) 151-64-12, http://www.madi.ru

nikolaev.madi@mail.ru

Аннотация. В статье рассмотрен новый протокол передачи данных по сети, отличающийся от стандартных, который носит название Protobuf. Проанализированы его особенности, приведены примеры сообщений с его использованием. Проведено сравнение типов данных Protobuf и типов данных других языков программирования. Приведены результаты экспериментов по сравнению протоколов Protobuf и JSON.

Ключевые слова: протокол, пакет, сериализация, десериализация, сообщение, сеть передачи данных.

1. Введение

Разработка мобильных приложений на сегодняшний день является актуальной и востребованной деятельностью - этому способствует стремительное техническое развитие: мобильные устройства с каждым годом улучшаются по всем характеристикам и становятся доступнее для широкого круга людей. Почти каждый, кто имеет на руках мобильный гаджет (смартфон, коммуникатор или планшет)

пользуется приложениями: браузером, клиентом электронной почты и мгновенных сообщений, играми, бизнес или финансовыми программами. И зачастую от пользователей скрыто то, что многие из приложений взаимодействуют с удаленным сервером: обмениваются с ним данными через Интернет. На рисунке 1 можно увидеть архитектуру модели приложения клиент-сервер [1…20].

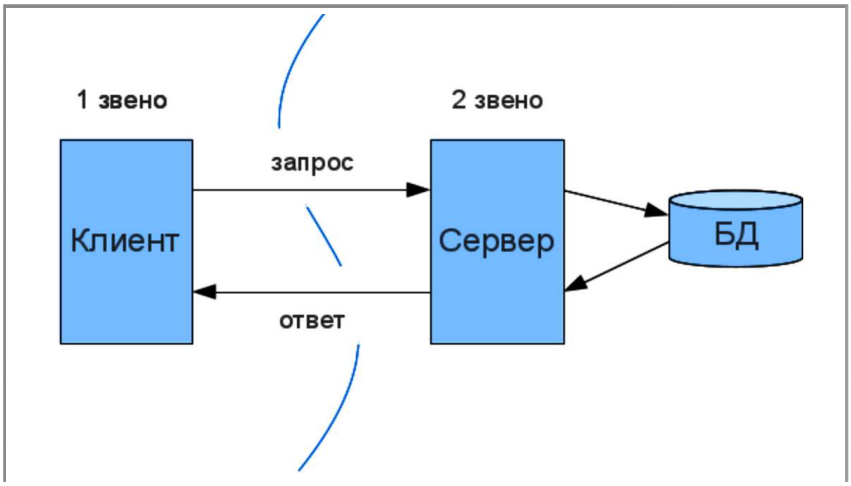

Рисунок 1 – Архитектура модели приложения сервер-клиент

Размещение и выполнение программ на стороне сервера снижает требования к аппаратному обеспечению клиентов и уменьшает проблемы обеспечения совместимости в гетерогенной сетевой среде [1…20].

2. Основные особенности Protocol Buffers

Protocol Buffers - библиотека кодирования и декодирования структурированных данных. Структура данных определяется заголовочными файлами *.proto, компилятор генерирует из них объекты на разных языках программирования (C++, Objective-C, Swift, Java и т. д.,), что обеспечивает свойство кроссплатформенности.

Такой подход, с одной стороны, позволяет заменить любой компонент системы на его реализацию на другом языке, а, с другой, позволяет удобно поддерживать различные версии объектов (message в терминологии Protobuf), добавлять новые поля и изменять существующие.

Google разработал Protobuf для использования в своих внутренних службах. Это двоичный формат кодирования, который позволяет задать схему для ваших данных с использованием спецификаций языка как показано на рисунке 2:

```
message Person {

required string name = 1;
required int32 id = 2;
optional string email = 3;
}
```

Рисунок 2 – Пример сообщения в формате Protobuf

Protobuf позволяют описывать простые структуры данных на специальном языке, который затем компилируется в классы, представляющие эти структуры. Вместе с классами идёт оптимизированный код их сериализации в компактный формат представления. А лучше всего то, что доступ к данным максимально упрощён: для доступа к каждому полю у класса имеются соответствующие методы "get" и "set", а для сериализации объекта в массив байтов или поток ввода/вывода нужно сделать всего один вызов.

3. Пример сообщения в протоколе Protobuf

Данный пример будет написан на языке C++.

Для начала необходимо определить структуру информации для сериализации посредством описания формата сообщения Protocol buffer в .proto файле. Каждое сообщение Protobuf это небольшая логическая запись информации, содержащая серию пар имя-значение. Ниже на рисунке 3 представлен простой пример .proto файла, определяющего сообщение содержащее информацию о человеке:

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }
```

Рисунок 3 – Сообщение в формате .proto

На примере видно, что формат сообщения очень простой - каждое сообщение имеет одно или несколько уникальных нумерованных полей, а каждое поле имеет имя и тип сообщения. В начале файла находится строка с именем пакета. Она определяет область видимости данных и служит для предотвращения конфликта имен. Далее следует блок с сообщениями, которые начинаются с ключевого слова message. Типом значения может быть число (целое или с плавающей запятой), логическое значение (boolean), строка или набор байтов. Каждое поле должно быть отмечено одним из

следующих ключевых слов: required, optional и repeated. Required - поле должно быть всегда инициализировано. Optional говорит protobuf-компилятору, что поле может быть не инициализировано, а repeated сообщает о возможности неоднократного повторения переменной, описанной с помощью этого спецификатора в структуре данных. Кроме того, каждый элемент имеет так называемые тэги (десятичная цифра после знака равно в конце объявления).

Следует также отметить, что каждое поле может иметь значение по умолчанию в следующем виде:

```
optional int32 result_per_page = 3 [default = 10];
```

Рисунок 4 – Значение по умолчанию в Protobuf

Этот код говорит о том, что значение поля result_per_page по умолчанию равно 10. Если поле не имеет значения по умолчанию, то значение по умолчанию для строки - пустая строка, для числовых типов - 0, для логического значения - ложь.

Иногда может понадобиться чтобы поле содержало определенный тип значения, например, номер телефона мобильного, домашнего или рабочего. Для этого необходимо использовать ключевое слово enum и указать название перечисления.

Messages могут быть вложены одно в другое и использоваться как пользовательские типы при объявлении элементов других сообщений.

После определения сообщений, необходимо запустить предназначенный для вашего языка программирования компилятор protocol buffer, который на основе .proto файлов создаст классы доступа к данным. Они предоставляют простой доступ к каждому полю (например, query() и set_query()), а так же методы сериализации/разбора всей структуры в/из бинарных данных. В нашем случае, когда мы выбрали язык C++, скомпилировав вышеприведенный пример мы получим сгенерированный класс с названием Person, который можно использовать в вашем приложении для заполнения, сериализации и передачи сообщения Person.

При использовании получится примерно такой код, как представлен на рисунке 5.

```
Person person;
person.set_name("John Doe");
person.set_id(1234);
person.set_email("jdoe@example.com");
fstream output("myfile", ios::out | ios::binary);
person.SerializeToOstream(&output);
```

Рисунок 5 – Отправление пакета

После, его можно прочитать, как показано на рисуноке 6.

```
fstream input("myfile", ios::in | ios::binary);
Person person;
person.ParseFromIstream(&input);
cout << "Name: " << person.name() << endl;
cout << "E-mail: " << person.email() << endl;
```

Рисунок 6 – Чтение пакета

В таблице 1 проведено сравнение типов данных Protobuf и типов данных других языков программирования.

Таблица 1 – Типы данных Protobuf соответствующие типу данных на других языках

.Proto типы	C++ типы	JAVA типы	Python типы
double	double	double	float
float	float	float	float
int32	int32	int	int
int64	int64	long	int/long
uint32	uint32	int	int/long
uint64	uint64	long	int/long
sint32	int32	int	int
sint64	int64	long	int/long
fixed32	uint32	int	int
fixed64	uint64	long	int/long
sfixed32	int32	int	int
sfixed64	int64	long	int/long
bool	bool	boolean	boolean
string	string	String	str/unicode
bytes	string	ByteString	str

4. Сравнение протоколов Protobuf и JSON

Отвергнув популярные технологии вроде XML, CSV и JSON, программисты Twitter выбрали в качестве формата для хранения данных бэкенда формат Protobuf.

Каждый день в базу Twitter добавляется 12 ТБ новых данных. С такими объёмами выбор правильного формата приобретает критическое значение. Комбинация Protobuf, Hadoop и смежных технологий призвана решить эту проблему.

Преимущество Protobuf перед XML становится очевидным на больших объёмах данных. По словам разработчиков Twitter, база в триллион твитов на XML может занимать примерно десять петабайт вместо одного. В JSON тоже хранится много лишней информации. На другом полюсе — CSV, где данные разделяются всего лишь запятыми. Здесь ничего лишнего, но трудно структурировать подполя.

Protobuf не имеет этих недостатков. Кроме того, в нём автоматизирован процесс воссоздания структур данных. Как сказано в руководстве по Protocol Buffers, достаточно однажды определить метод структурирования данных, после чего можно использовать специально сгенерированный код для простой т записи и чтения структурированных данных в/из различных потоков и на разных языках.

Ниже показан результат эксперимента сравнения отправки данных с помощью JSON и Protobuf. Как видно из таблицы 2 и рисунка 7, Protobuf выигрывает у JSONa по времени сериализации и дисериализации, и также Protobuf экономит размер данных информационного обмена, это видно по результатам таблицы 3 и русунка 7. Эксперимент был проведен следующим образом: в цикле было создано 1000 сообщений типа Person и отправлены клиенту.

Таблица 2 – Результат теста сериализации и дисериализации объекта Person

Название	**Среднее время**	**Минимальное время**	**Максимальное время**	**Дисперсия**
Protobuf сериализация	521	506	725	-0.03% / +0.35%
JSON сериализация	535	518	724	-0.03% / +0.39%
Protobuf дисериализация	880	861	993	-0.02% / +0.13%
JSON дисериализация	912	891	891	-0.02% / +0.25%

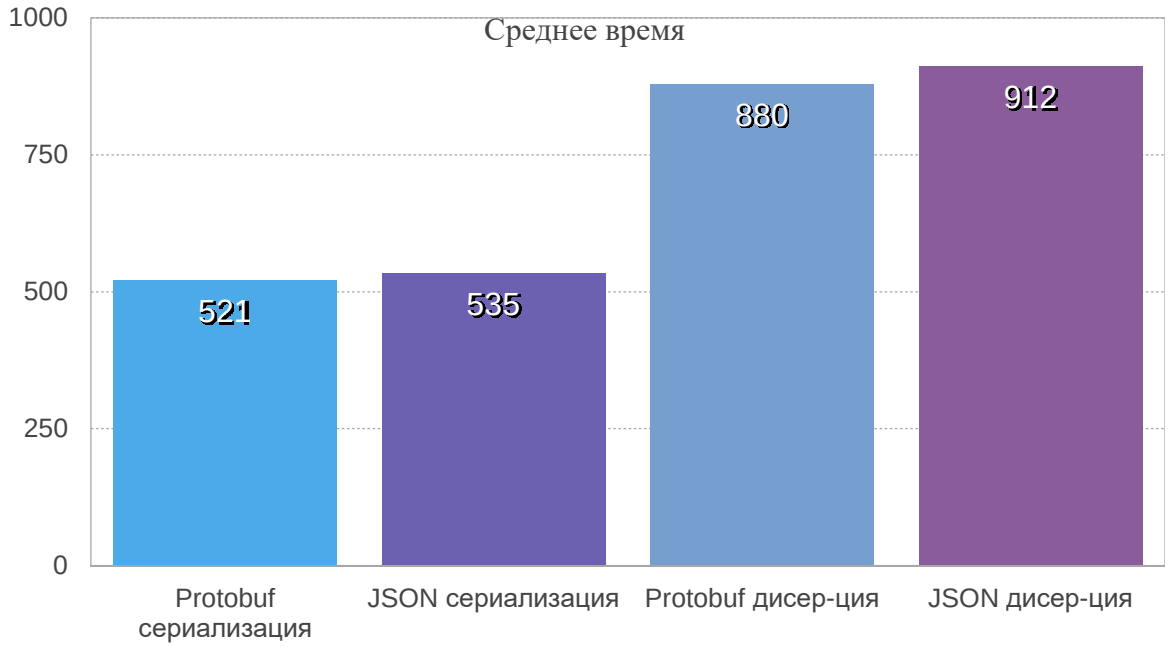

Рисунок 7 – Результаты эксперимента по сравнению среднего времени отправки данных с помощью протоколов JSON и Protobuf

Таблица 3 – Результат теста сериализации объекта Person-1

Название	Не сжатый (BYTES)	Сжатый (BYTES)	КОЭФФИЦИЕНТ СЖАТИЯ
JSON сериализация	28,064	9,747	2,88
Protobuf сериализация	18,827	9,252	2,03

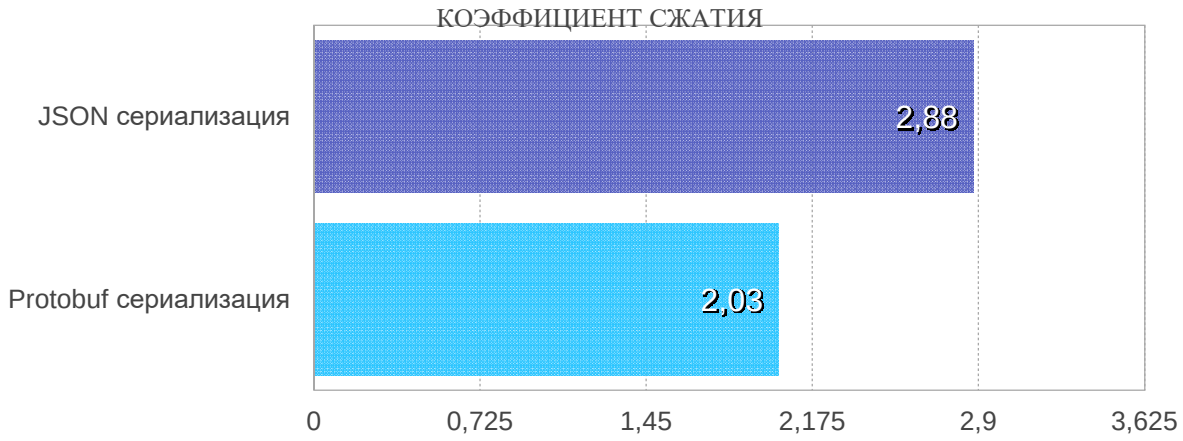

Рисунок 8 – Результат теста сериализации объекта Person-1

5. Заключение

Protocol Buffers позволят перевести данные и объекты в программе в бинарный вид и безопасно переслать их по сети или сохранить в файле для последующего использования. Основным достоинством протокола Protocol Buffers является простота использования. Достаточно сделать метаописание бизнес-объекта, после чего получается готовый код бизнес-объекта и его сериализация. Пользователю нужно просто использовать полученный код.

Стандартные средства Protocol Buffers дают возможность не только выполнять сериализацию в строку, но и в поток, что позволяет сразу вывести данные в файл, сеть или куда-то еще, что позволяет поддерживать обычные C++ потоки типа std::ostream и std::istream.

Список информационных источников

[1] Учебно-методические материалы для студентов кафедры АСОИУ www.4stud.info.

[2] Интернет ресурс для IT-специалистов www.habrhabr.ru.

[3] Интернет ресурс с публикациями для разработчиков продуктов компании Google - Google open source www.google-opensource.blogspot.ru.

[4] Интернет ресурс для разработчиков продуктов компании Google - Google Developers www.developers.google.com.

[5] Интернет ресурс для IT-специалистов www.geektims.ru.

[6] Остроух А.В. Рефакторинг баз данных. Автоматизация технологических процессов рефакторинга баз данных промышленных предприятий / А.В. Остроух, Д.А. Пшеничный, О.Б. Рогова. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 133 p. – ISBN 978-3-659-38753-1.

[7] Остроух А.В. Автоматизация управления производством. Повышение эффективности автоматизированных аналитических систем предприятий автомобильной промышленности / А.В. Остроух, Э.А. Чернов, Д.Т. Нгуен. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 285 p. – ISBN 978-3-659-34762-7.

[8] Остроух А.В. Системы планирования перевозок. Программно-технологические решения по разработке системы планирования заданий для заказных пассажирских перевозок / А.В. Остроух, А.Б. Львова, А.Р. Исмаилов. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2013. – 121 p. – ISBN 978-3-659-43619-2.

[9] Остроух А.В. Интеллектуальные системы в науке и производстве / А.В. Остроух, А.Б. Николаев. – Saarbrucken, Germany: Palmarium Academic Publishing, 2012. – 312 p. – ISBN 978-3-659-98006-0.

[10] Остроух А.В. Ввод и обработка цифровой информации: учебник для нач. проф. образования / А.В. Остроух. – М.: Издательский центр «Академия», 2012. – 288 с. – ISBN 978-5-7695-9457-1.

[11] Остроух А.В. Оперативный контроль транспортно-экспедиционной деятельности. Процессный подход к агрегированию системы показателей деятельности транспортно-экспедиционного предприятия / А.В. Остроух, А.М. Ивахненко, Н.А. Крупенский. – Saarbrucken, Germany: Palmarium Academic Publishing, 2013. – 221 p. – ISBN 978-3-659-98329-0.

[12] Исмоилов М.И. Подготовка и переподготовка персонала предприятий промышленного и транспортного комплексов с применением мобильных технологий: монография / М.И. Исмоилов, А.Б. Николаев, А.В. Остроух. – Saint-Louis, MO, USA: Publishing House Science and Innovation Center, 2013. – 166 с. – ISBN 978-0-615-67111-6.

[13] Николаев А.Б. Информационные технологии в менеджменте и транспортной логистике: учебное пособие / А.Б. Николаев, А.В. Остроух. – Saint-Louis, MO, USA: Publishing House Science and Innovation Center, 2013. – 254 с. – ISBN 978-0-615-67110-9.

[14] Остроух А.В. Системы искусственного интеллекта в промышленности, робототехнике и транспортном комплексе: монография / А.В. Остроух – Красноярск: Научно-инновационный центр, 2013. – 326 с. – ISBN 978-5-906314-10-9.

[15] Остроух А.В. Основы информационных технологий: учебник для сред. проф. образования / А.В. Остроух. – М.: Издательский центр «Академия», 2014. – 208 с. – ISBN 978-5-4468-0588-4.

[16] Суркова Н.Е. Методология структурного проектирования информационных систем: монография / Н.Е. Суркова, А.В. Остроух. – Красноярск: Научно-инновационный центр, 2014. – 190 с. – ISBN 978-5-906314-16-1.

[17] Покровский А.К. Системный анализ и компьютерные технологии при управлении транспортным обслуживанием предприятий: монография / А.К. Покровский, А.В. Остроух, А.М. Ивахненко. – М.: Известия, 2014. – 261 с. – ISBN 978-5-94280-314-7.

[18] Остроух А.В. Проектирование системы распределенных баз данных / А.В. Остроух, А.В. Помазанов. – Saarbrucken, Germany: Palmarium Academic Publishing, 2015. – 117 p. – ISBN 978-3-659-60041-8.

[19] Остроух А.В. Автоматизация управления автотранспортными предприятиями. Новый подход на основе интеллектуальных мультиагентных систем / А.В. Остроух, А.В. Воробьева, Н.Е. Суркова. – Saarbrucken, Germany: LAP LAMBERT Academic Publishing, 2015. – 117 p. – ISBN 978-3-659-47576-4.

[20] Суркова Н.Е. Профессиональные информационные системы и базы данных: методические указания к лабораторным работам / Н.Е. Суркова, А.В. Остроух, Т.И. Еремина. – Красноярск: Научно-инновационный центр, 2015. – 49 с. – ISBN 978-5-906314-23-9. DOI: 10.12731/asu.madi.ru/PISDB.2015.49.