МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ
И МАТЕМАТИКИ ИМ. А.Н. ТИХОНОВА

Исследование эффективности методов сжатия и сериализации данных для использования в методе GDEPC при передаче по спутниковому каналу Iridium

Ковязин В. В.

Москва 2023

IoRT (Удаленный интернет вещей) - это идея создания сети передачи данных с использованием физических объектов, которые обладают возможностью взаимодействовать между собой и с внешней средой. Такие объекты находятся в удаленных регионах, где инфраструктура сети не сильно развита.

Проблема

Высокая стоимость [2] отправки SBD контейнера с сообщениями, большая стоимость отправки одного сообщения с данными от устройств

Как уменьшить затраты

Упаковывать больше сообщений

Уменьшить размер одного сообщения

Сжимать и/или сериализировать сообщения перед упаковкой

Метод GDEPC [2], существующее решение

Используется для увеличения количества сообщений в контейнере

Что улучшить:

Улучшим его, добавив сжатие в step 1 без дальнейшей сериализации или в step 2 с сериализацией

Так же исследуем другие методы сериализации для замены Protobuf

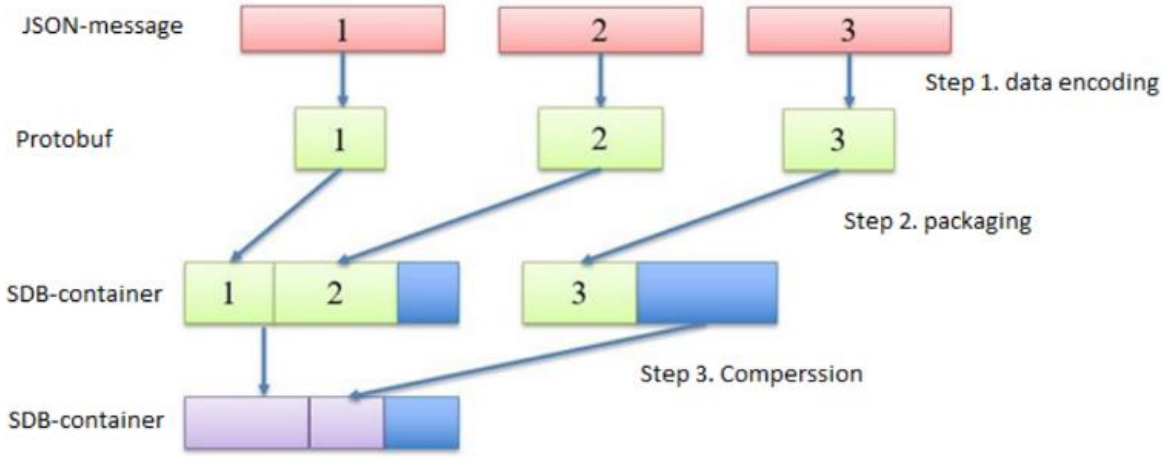


Схема метода GDEPC[2]

Цель

Целью исследования является разработка программного обеспечения и оценка преимуществ различных методов сжатия и сериализации данных при передаче сообщений по каналу Iridium. В качестве алгоритмов сжатия исследуются не только gzip, bzip2 и кодирование Хаффмана, но и методы сжатия, основанные на использовании алгоритмов машинного обучения. Ожидаемыми результатами являются оптимизированные алгоритмы сжатия, которые улучшат качество передаваемых данных и снизят стоимость связи для передачи сообщений по спутниковому каналу Iridium.

Используемая структура сообщения [1]

Формат - JSON

Датасеты с размерами: 300-500 байт, 500 - 1000 байт, 1000 - 2000 байт

datastring - поле с информацией от датчиков

Остальные поля служебные


```
{
    "rxinfo": {
        "channel": 1,
        "coderate": "4/5",
        "crcStatus": 1,
        "dataRate": {
            "bandwidth": 134,
            "modulation": "LORA",
            "spreadFactor": 5
        },
        "frequency": 85019680594,
        "loRaSNR": 7,
        "mac": "qnpqak",
        "rfChain": 1,
        "rssi": -57,
        "size": 23,
        "time": "0001-01-01T00:00:00Z",
        "timestamp": 29024925
    },
    "phyPayload": {
        "datastring": ""
    }
},
```

Сериализация данных
Protobuf, Capnproto, ASN1, FlatBuffers

Применялись официальные библиотеки и инструменты

```
Message DEFINITIONS ::= BEGIN
    Message ::= SEQUENCE {
        phyPayload SEQUENCE {
            datastring IA5String
        },

        rxinfo SEQUENCE {
            channel INTEGER,
            codeRate IA5String,
            crcStatus INTEGER,
            dataRate SEQUENCE {
                bandwidth INTEGER,
                modulation IA5String,
                spreadFactor INTEGER
            },
            frequency INTEGER,
            loRaSNR INTEGER,
            mac IA5String,
            rfChain INTEGER,
            rssi INTEGER,
            size INTEGER,
            time GeneralizedTime,
            timestamp INTEGER
        }
    }

END
```

ASN1

```
import "google/protobuf/timestamp.proto";
message phypayload {
 repeated string data = 1;
}
message rate {
 int32 bandwidth = 1;
 enum mod {
 LORA = 0;
 }
 mod modulation = 2;
 int32 spreadFactor = 7;
}
message info {
 int32 channel = 1;
 string codeRate = 2;
 int32 crcStatus = 3;
 rate dataRate = 4;
 int64 frequency = 5;
 int32 loRaSNR = 6;
 string mac = 7;
 int32 rfChain = 8;
 sint32 rssi = 9;
 int32 size = 10;
 google.protobuf.Timestamp time = 11;
 google.protobuf.Timestamp timestamp = 12;
}
message iotj {
 phypayload phyPayload = 1;
 info rxinfo = 2;
}
```

Protobuf

Сжатие данных

Только сжатие без потерь
Huffman, gzip, 7zip, bz2, smaz, zstd

Для всех алгоритмов применялись официальные или разработанные сообществом Python библиотеки

Результаты по сжатию:

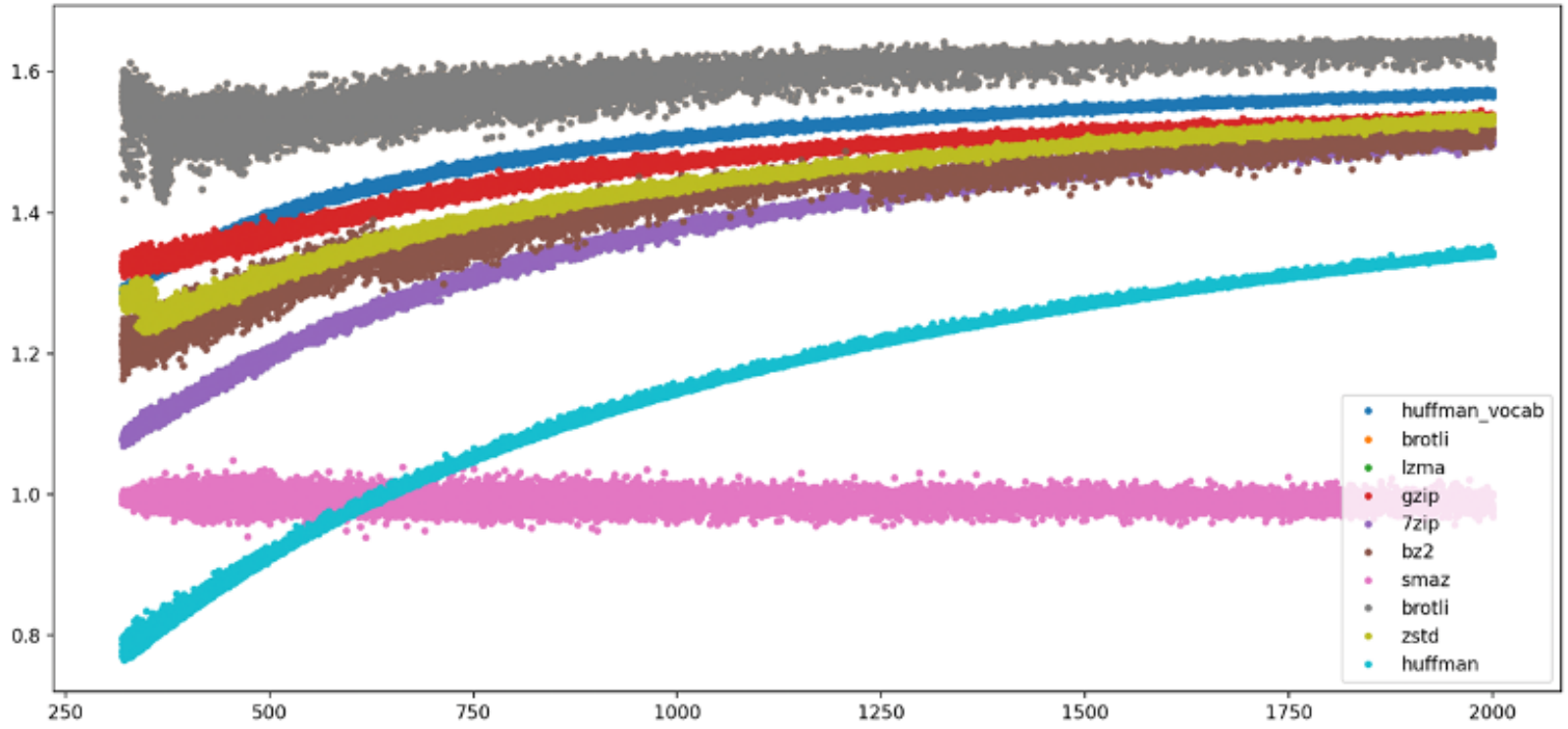


График эффективности алгоритмов сжатия (ось x – размер исходного сообщения, ось y – коэффициент между размером оригинального сообщения и сжатого)

Название алгоритма	Среднее время сжатия одного сообщения в миллисекунда x	Средний размер сжатого сообщения в байтах	Средний коэффициент сжатия
brotli	1.064	268.0	1.527
gzip	0.031	304.0	1.345
7zip	1.06	358.0	1.141
bz2	0.211	327.0	1.25
smaz	0.019	412.0	0.995
zstd	0.008	320.0	1.278
huffman	1.244	482.0	0.846
Huffman with dict	0.137	268.0	1.53

Название алгоритма	Среднее время сжатия одного сообщения в миллисекундах	Средний размер сжатого сообщения в байтах	Средний коэффициент сжатия
brotli	1.496	479.0	1.564
gzip	0.043	523.0	1.428
7zip	1.196	572.0	1.302
bz2	0.321	547.0	1.365
smaz	0.037	756.0	0.992
zstd	0.01	540.0	1.384
huffman	2.187	712.0	1.045
Huffman with dict	0.387	498.0	1.504

Сжатие сообщений 500-1000 байт

Выводы по сжатию:

Самым быстрым алгоритмом среди всех стал zstd, среднее затраченное время получилось 0.008 миллисекунд, его коэффициент получился четвертым по величине. Самым эффективным показали себя Хаффман со словарем и brotli, но среднее время выполнения у brotli хуже на целую миллисекунду.

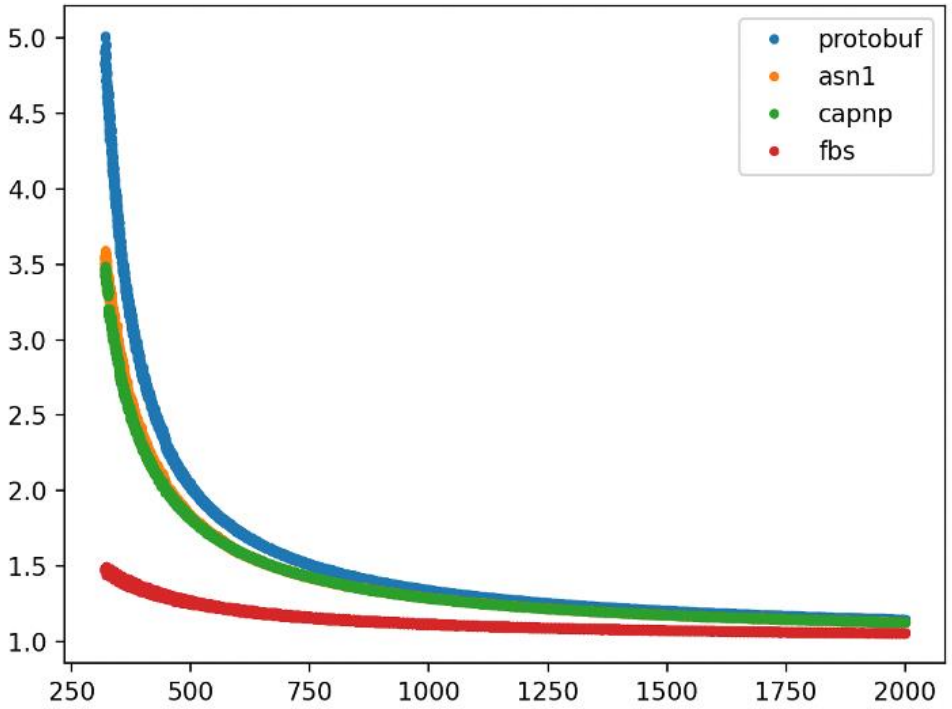
Название алгоритма	Среднее время сжатия одного сообщения в миллисекундах	Средний размер сжатого сообщения в байтах	Средний коэффициент сжатия
brotli	2.492	926.0	1.618
gzip	0.056	991.0	1.51
7zip	3.858	1030.0	1.451
bz2	0.431	1017.0	1.471
smaz	0.088	1515.0	0.99
zstd	0.015	1001.0	1.493
huffman	3.909	1181.0	1.261
Huffman with dict	0.502	969.0	1.544

Сжатие сообщений 1000 - 2000 байт

Результаты по сериализации:

График эффективности методов сериализации (ось x – размер исходного JSON сообщения, ось y – коэффициент между размером оригинального сообщения и сериализованного объекта)

Используемый датасет - сообщения от 300 до 2000 байт



Название метода	Среднее время сериализации одного сообщения в миллисекундах	Средний размер сериализованного сообщения в байтах	Средний коэффициент сериализации
Protobuf	0.03	155.0	2.882
Flatbuffers	0.251	307.0	1.346
Capnp	0.018	185.0	2.332
ASN1	0.059	181.0	2.392

Сериализация сообщений 300-500 байт

Название метода	Среднее время сериализации одного сообщения в миллисекундах	Средний размер сериализованного сообщения в байтах	Средний коэффициент сериализации
Protobuf	0.029	497.0	1.56
Flatbuffers	0.271	647.0	1.168
Capnp	0.02	526.0	1.464
ASN1	0.062	526.0	1.465

Сериализация сообщений 500-1000 байт

Выводы по сериализации:

Protobuf стал самым эффективным методом для сериализации. Особенно эффективным стал в сериализации малых JSON сообщений, которые чаще всего встречаются в сетях LoRa Iridium. Сериализация позволила добиться пятикратного уменьшения в размере для маленьких сообщений.

Название метода	Среднее время сериализации одного сообщения в миллисекундах	Средний размер сериализованного сообщения в байтах	Средний коэффициент сериализации
Protobuf	0.035	1246.0	1.215
Flatbuffers	0.267	1397.0	1.077
Capnp	0.02	1276.0	1.185
ASN1	0.065	1276.0	1.185

Сериализация сообщений 1000 - 2000 байт

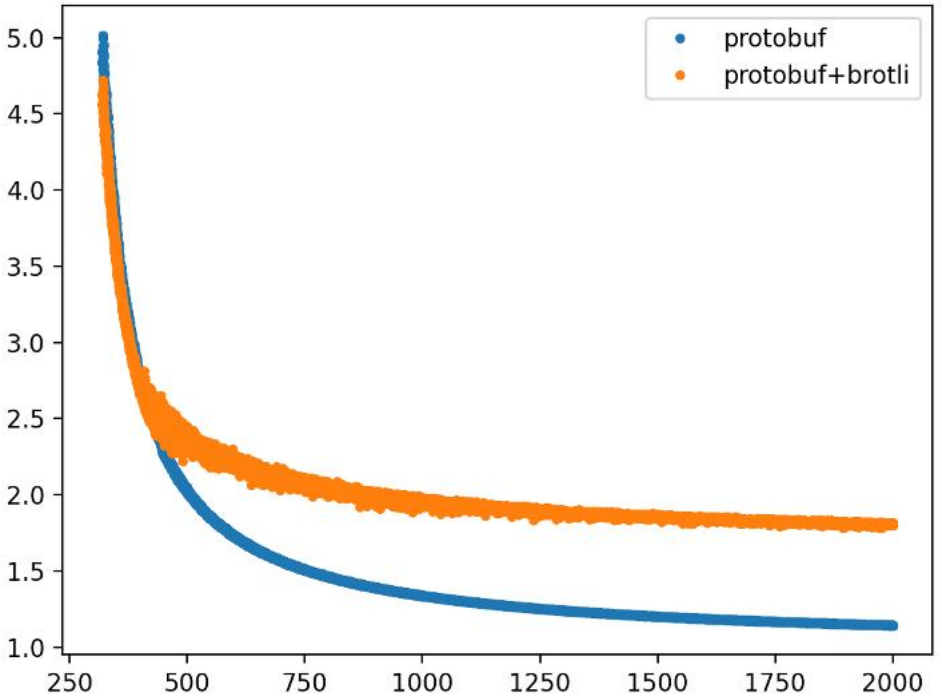
Самое интересное - комбинирование сжатия и сериализации

На тех же датасетах будем проводить первым этапом сериализацию JSON в бинарную строку, вторым этапом сжимать алгоритмом сжатия. Алгоритм сжатия brotli показал себя лучше всех - используем его.

Результаты по сжатию до и после на примере Protobuf:

График эффективности Protobuf без и с brotli сжатием (ось x – размер исходного JSON сообщения, ось y – коэффициент между размером оригинального сообщения и результата)

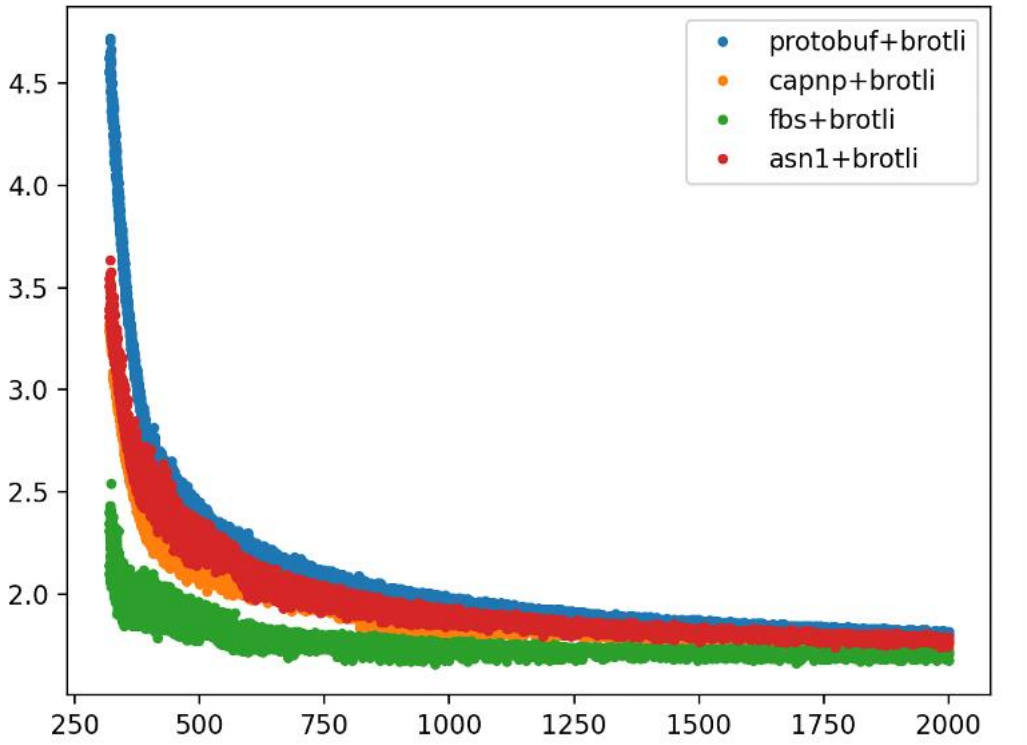
Используемый датасет - сообщения от 300 до 2000 байт



Результаты по сжатию и сериализации:

График эффективности сериализации без и с brotli сжатием (ось x – размер исходного JSON сообщения, ось y – коэффициент между размером оригинального сообщения и результата)

Используемый датасет - сообщения от 300 до 2000 байт



Название метода	Среднее время сериализации одного сообщения в миллисекундах	Средний размер сериализованного сообщения в байтах	Средний коэффициент сериализации
Protobuf+brotli	0.679	148.0	2.918
Flatbuffers+brotli	1.426	210.0	1.966
Capnp+brotli	0.932	173.0	2.424
ASN1+brotli	0.867	163.0	2.573

Сериализация + сжатие brotli сообщений 300-500 байт

Название метода	Среднее время сериализации одного сообщения в миллисекундах	Средний размер сериализованного сообщения в байтах	Средний коэффициент сериализации
Protobuf+brotli	0.948	361.0	2.105
Flatbuffers+brotli	1.645	423.0	1.778
Capnp+brotli	1.004	385.0	1.967
ASN1+brotli	1.166	376.0	2.015

Сериализация + сжатие brotli сообщений 500-1000 байт

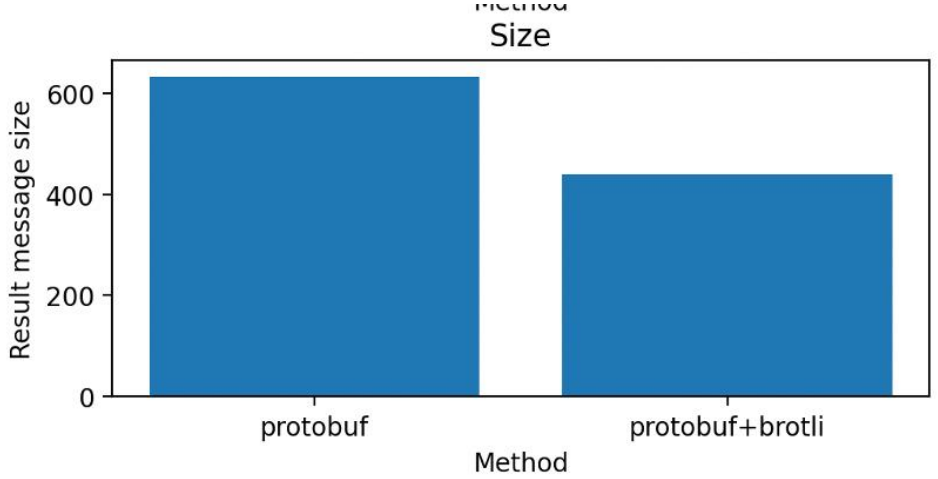
Выводы по комбинированию:

По результатам замерения среднего времени, видно, что для всех размеров сообщений Protobuf+brotli показывают себя быстрее, чем остальные комбинации.
При этом, по сравнению с обычной сериализацией без сжатия, среднее время, затраченное на обработку одного сообщения, разительно возрастает.

Название метода	Среднее время сериализации одного сообщения в миллисекундах	Средний размер сериализованного сообщения в байтах	Средний коэффициент сериализации
Protobuf+brotli	1.431	808.0	1.865
Flatbuffers+brotli	2.743	872.0	1.72
Capnp+brotli	1.836	837.0	1.799
ASN1+brotli	2.073	827.0	1.819

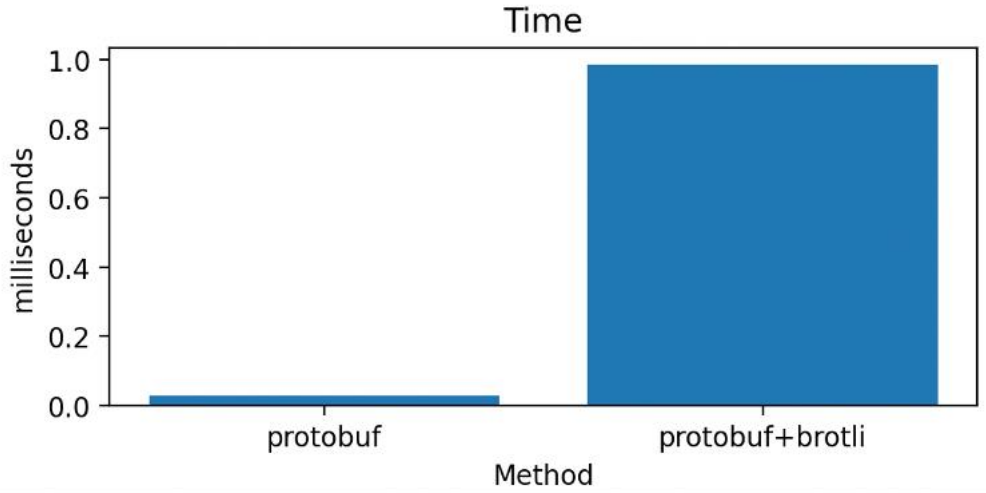
Сериализация + сжатие brotli сообщений 1000 - 2000 байт

Выводы по комбинированию:



Сравнение комбинации protobuf+brotli и просто protobuf по размеру результата

Используемый датасет - сообщения от 300 до 2000 байт



Сравнение комбинации protobuf+brotli и просто protobuf по времени

Используемый датасет - сообщения от 300 до 2000 байт

Вывод

В данной работе было изучено применимость алгоритмов сжатия и сериализации на JSON сообщения сети LoRa Iridium. Для изучения было разработано специальное ПО для генерации данных сообщений, исследования их эффективности и выведения сравнительных характеристик и графиков. Из всех изученных алгоритмов сжатия эффективнее всех показал себя brotli, сериализации – Protobuf. Комбинация из этих двух подходов позволило добиться среднего уменьшения размера JSON сообщения в 3-5 раз для малых (от 300 до 500 байт) и в 2 раза для больших (от 500 байт).

Список литературы

1. Ivan Lysogor, Voskov L., Rolich A., Efremov S. G. Study of Data Transfer in a Heterogeneous LoRa-Satellite Network for the Internet of Remote 4. Things // Sensors. 2019. Vol. 19. No. 15. P. 1-17. doi
2. Voskov L., Rolich A., Bakanov Gleb, Podkopaeva Polina. Gateway Data Encoding, Packaging and Compression method for heterogeneous IoT-satellite network, in: 2021 XVII International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY). IEEE, 2021. doi P. 34-38. doi


ВЫСШАЯ • ШКОЛА • ЭКОНОМИКИ •