

# Relatório da Atividade 01: Tamanho de envio de pacotes

Davi de Lima Cruz  
Matrícula:474377

December 2, 2024

## 1 Introdução

Nessa atividade, foi pedido para avaliar o tamanho dos pacotes enviados por diferentes formas de envio de dados em Java.

## 2 Metodologia

### 2.1 Dados

Como pedido, avaliamos diferentes formas que o Java envia pacotes de dados. Sendo elas:

- Serialização de objetos: nesse caso apenas implementamos a interface `Serializable` e enviamos o objeto.
- Serialização de objetos Customizando: para a Customizando implementamos a escrita e leitura do objeto de acordo com o seu tipo, em tese, isso poderia diminuir o tamanho do pacote.
- Json: enviamos o arquivo em formato Json em formato de String.
- XML: enviamos o arquivo em formato XML em formato de String.
- Protobuf: usamos a classe gerada pelo protoc para enviar o arquivo.

Além disso, escolhemos dois cenários para avaliar o tamanho do pacote, sendo eles:

1. Lista de compras: contendo 5 itens onde cada item possui um nome, quantidade e unidade.
2. Agenda de contatos: contendo 20 contatos onde cada contato possui um nome, telefone, email e foto.

Foi usada a mesma foto em todos os contatos e ela foi transformada em base64 para ser enviada.



Figure 1: Foto usada nos contatos

O formato dos Json e XML fica no apêndice para facilitar a leitura. Segue o protobuf usado para o caso da lista de compras e agenda de contatos.

Listing 1: Protobuf utilizado na lista de compras.

```
1  syntax = "proto3";
2  package data;
3  option java_package = "data";
4
5  message PCompras {
6      repeated PItem lista_compras = 10;
7  }
8  message PItem {
9      string nome = 1;
10     float quantidade = 2;
11     string unidade = 3;
12 }
```

Listing 2: Protobuf utilizado na agenda de contatos.

```
1  syntax = "proto3";
2  package data;
3  option java_package = "data";
4
5  message PAgenda {
6      repeated PContato contatos = 10;
7  }
8  message PContato {
9      string nome = 1;
10     string telefone = 2;
11     string email = 3;
12     string foto = 4;
13 }
```

## 2.2 Software

Para fazer a medição dos pacotes foi criado um servidor e um cliente em Java. O client mandava em sequencia um dado de cada tipo para cada caso em portas diferentes e o servidor recebia sequencialmente. Enquanto isso, o **tshark** capturava os pacotes e salvamos a porta e o tamanho do pacote em um arquivo. A partir disso somamos o tamanho dos pacotes com o AWK.

Tipo	Porta	
	Compras	Agenda
Serialização	5100	5200
Serialização Custom.	5101	5201
Json	5102	5202
XML	5103	5203
Protobuf	5104	5204

Table 1: Portas utilizadas

## 3 Resultados

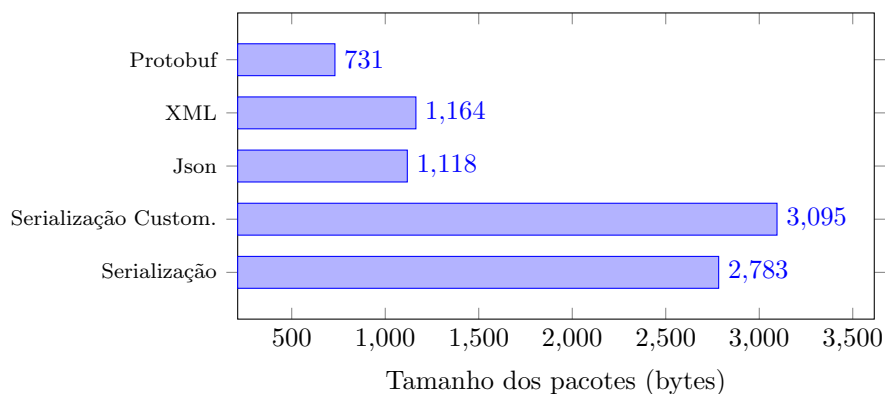


Figure 2: Lista de compras

Analisando os resultados, podemos perceber que o Protobuf é o mais eficiente em relação ao tamanho do pacote, seguido pelo Json e XML. Como já era esperado, a Serialização normal e a Customizada tiveram um tamanho de pacote maior, isso se deve ao fato de que elas enviam o objeto inteiro, enquanto o Json e XML enviam apenas os dados.

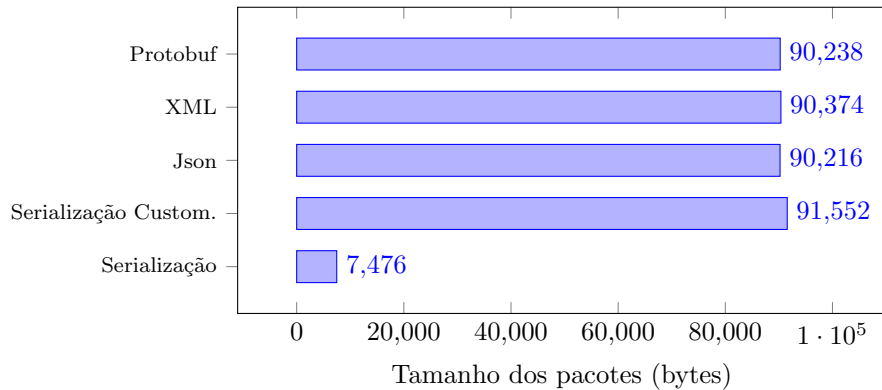


Figure 3: Agenda de contatos

Para a agenda de contatos, percebe-se que para grande volume de dados as Serializações praticamente empatam e não faz muita diferença qual usar, o único detalhe foi que a Serialização ganhou disparado de todos os outros, isso se deve ao fato de termos usado a mesma foto em todos os contatos, ou seja, o mesmo ponteiro, então provavelmente a Serialização percebeu isso e enviou só uma vez. Considerando também que a Serialização Customizada teve um tamanho dos pacotes maior, isso se deve ao fato de que os dados por serem maioria em texto, não houve uma otimização significativa, se houve muitos números a Serialização Customizada poderia ter um tamanho menor.

De modo geral, o Protobuf e o Json são os mais eficientes em relação ao tamanho do pacote, mas é importante sempre analisar o problema e ver se existe uma forma melhor de transferir os seus dados para o seu caso, como a foi a Serialização que descobriu sozinha que a foto era a mesma em todos os contatos.

## A Json e XML

Listing 3: Lista de compras Json.

```
1  {
2
3      "lista_de_compras": [
4          {
5              "item": "Leite",
6              "quantidade": 2,
7              "unidade": "litros"
8          },
9          ...
10         {
11             "item": "Peito de Frango",
12             "quantidade": 1,
13             "unidade": "kg"
14         }
```

```

15     ]
16
17 }

```

Listing 4: Lista de compras XML.

```

1  <lista_de_compras>
2    <item>
3      <nome>Leite</nome>
4      <quantidade>2</quantidade>
5      <unidade>litros</unidade>
6    </item>
7    ...
8    <item>
9      <nome>Peito de Frango</nome>
10     <quantidade>1</quantidade>
11     <unidade>kg</unidade>
12   </item>
13 </lista_de_compras>

```

Listing 5: Agenda de contatos Json. Sem a foto e alguns contatos para não ficar muito grande.

```

1  {
2    "contacts": [
3      {
4        "name": "Alice Johnson",
5        "phone": "555-1234",
6        "email": "alice.johnson@example.com",
7        "photo": ...
8      },
9      ...
10     {
11       "name": "Tina Yellow",
12       "phone": "555-5432",
13       "email": "tina.yellow@example.com",
14       "photo": ...
15     }
16   ]
17 }

```

Listing 6: Agenda de contatos XML. Sem a foto e alguns contatos para não ficar muito grande.

```

1  <contacts>
2    <contact>
3      <name>Alice Johnson</name>
4      <phone>555-1234</phone>
5      <email>alice.johnson@example.com</email>

```

```
6         <photo>...</photo>
7     </contact>
8     ...
9     <contact>
10         <name>Tina Yellow</name>
11         <phone>555-5432</phone>
12         <email>tina.yellow@example.com</email>
13         <photo>...</photo>
14     </contact>
15 </contacts>
```