

Distributed Systems

Higher Institute for Applied Sciences and Technology

Teacher: Mohammad Bashar Dasouki
Student: Daaa Hanna
Date:2/11/2025

gRPC

Nov 2 2025

Q1: What is Protobuf?

Lab definition: the definition i caught from the lab is that protobuf is the protocol then runs gRPC and can run in other different technologies such as REST

It is responsible for serialization and deserialization of data

It does that in a compressed way such that it saves data being transferred compared to json

Reference definition: Protobuf, which is short for “Protocol Buffers,” is an efficient, language-agnostic data serialization mechanism. It enables developers to define structured data in a `.proto` file, which is then used to generate source code that can write and read data from different data streams(

Gbadebo Bello,2024).

Reference Link : <https://blog.postman.com/what-is-protobuf/>

Q2: What are the benefits of using gRPC?

gRPC offers several benefits :

Efficiency: Protobuf serializes data into a binary format, which is much more compact than equivalent data in text-based formats like JSON or XML. This compactness translates to reduced storage and bandwidth usage. Additionally, Protobuf's serialization and deserialization processes are typically faster when compared to JSON.

Cross-language support: Protobuf provides support for multiple programming languages, which makes it easy to integrate across polyglot microservice architectures.

Strong typing with a clear schema: Protobuf requires developers to define a clear schema for data in .proto files. This schema-based approach ensures that the data structure is explicitly defined, which leads to better consistency, easier maintenance, and early detection of errors.

Backward and forward compatibility: Protobuf is designed to handle changes in the schema without breaking compatibility with older versions. This means that you can add new fields to your data structures without affecting existing code, which is crucial for long-term maintenance of a system.

Efficient network usage: Protobuf's compact binary format makes it an excellent choice for network communication, especially in environments where bandwidth is limited, such as mobile networks or IoT devices.

The Reference that is used to get gRPC advantages is
“<https://blog.postman.com/what-is-protobuf/>” by Gbadebo Bello

Q3: What is the difference between REST and gRPC why should one use one over the other?

Aspect	gRPC	REST
Protocol	Uses HTTP/2, which is binary and multiplexed	Typically uses HTTP/1.1 or HTTP/2, which is text-based
Data Format	Uses Protocol Buffers (binary format)	Typically uses JSON (text-based format)
Language Support	Wide range of languages with official libraries	Language-agnostic, works with any language
Performance	Known for high performance, especially in scenarios with low latency and high throughput	Offers good performance but may not match gRPC in some scenarios
Flexibility	Provides strongly typed interfaces for better validation and error handling	More flexible in terms of data formats and endpoint design
Use Cases	Well-suited for internal microservices communication in cloud-native environments	Widely used for public APIs and integrations with web services
Tooling	Well-established tooling and libraries for various languages	Mature tooling with a wide range of libraries and frameworks
Complexity	Can be more complex due to strong typing and binary format	Generally simpler due to its stateless nature and text-based format

Source of the Photo if geeks for geeks from the url
“<https://www.geeksforgeeks.org/blogs/grpc-vs-rest/>”

One might ask why not always use gRPC if it is better in 99% of use cases

There are two main reasons for that

The first reason we talked about in class

1. gRPC require a predefined common interface between the server and the client

While in Rest u can have the only need endpoint for ur application

2. Second good reason to use Rest if because of its ease of use and flexibility

And it has a huge advantage for people who are already familiar with http web requests.

Q4: What is the purpose of the .proto file in the protobuf context

a .proto file is a text file that defines the structure of your data and, optionally, the services that operate on that data. It serves as the schema for structured data that you intend to serialize, deserialize, and exchange between different systems or applications.

Q5: What is the purpose of the protoc compiler? What does it do?

This compiler turns a .proto file which contains a structured data interface into a language specific code that contains the implementation and methods in that specific language.

Q6: explain the Builder design pattern

This question is the hardest so far to me

Because it is super new topic

From this video "https://www.youtube.com/watch?v=M7Xi1yO_s8E"

Builder is a creational design pattern that lets us construct complex objects step by step. The pattern allows you to produce different types and representations of an object using the same construction code.

The video clearly demonstrate the why would someone use such pattern thus the design choices of protobuf (builder design pattern is justified

)

Q7: how does protobuf reduce the data transferred?

Binary vs Text:

- JSON is a **text-based format**. Every piece of data is stored as readable text, including field names and values.
- Protocol Buffers (Protobuf) are **binary**, meaning the data is encoded in a compact binary form. Binary encoding uses fewer bytes to represent the same information.

No Repeating Field Names:

- In JSON, the field names (like `"name": "Alice"`) are repeated for every object, which takes extra space.
- In Protobuf, field names are replaced by small **numeric tags**. So instead of sending `"name"`, Protobuf might just send `1` to represent that field.

Efficient Data Types:

- Protobuf uses compact encodings for numbers, booleans, and other types. For example, small integers can take just 1 byte, whereas JSON always writes them as text (like `"123"` = 3 bytes, plus quotes).

Optional and Default Fields:

- Protobuf can **omit fields that have default values**, while JSON usually includes all fields, even if empty.

Q8: what is the importance of the field flags?

These numbers are called field tags.

They uniquely identify each field in the binary encoding of a Protocol Buffers message.

The numeric tags are used instead of field names when the message is serialized, which makes the data smaller and more efficient to send over the network

Q9: What is the purpose of the repeated keyword when defining a field in a .proto file?

The repeated keyword is used to define a field that can appear multiple times, essentially representing a list or array of values.

For example, repeated string emails = 3; means a user can have multiple email addresses.

Q10: What are the features of HTTP/2 and how are they used with gRPC?

HTTP/2 Features:

1. Multiplexing: Multiple requests/responses can be sent over a single connection simultaneously.
2. Binary framing: Data is sent in a compact binary format instead of text, reducing overhead.
3. Header compression: HTTP headers are compressed to save bandwidth.
4. Server push: The server can send data proactively to the client.

gRPC Usage:

- 1.gRPC uses HTTP/2 as its transport layer to take advantage of multiplexing, low-latency streaming, and efficient binary communication.
- 2.This allows gRPC to support bi-directional streaming, long-lived connections, and high-performance RPC calls.