REST vs gRPC

WHAT IS REST?

REST (Representational State Transfer) is an architectural style that provides guidelines for designing web APIs.
It uses standard HTTP 1.1 methods like GET, POST, PUT, and DELETE to work with server-side resources. Additionally, REST APIs provide pre-defined URLs that the client must use to connect with the server.

What Is gRPC?

gRPC (Remote Procedure Call) is an open-source data exchange technology developed by Google using the HTTP/2 protocol. It uses the Protocol Buffers binary format (Protobuf) for data exchange. Also, this architectural style enforces rules that a developer must follow to develop or consume web APIs.

Guidelines vs. Rules

REST is a set of guidelines for designing web APIs without enforcing anything. On the other hand, gRPC enforces rules by defining a .proto file that must be adhered to by both client and server for data exchange.

Underlying HTTP Protocol

REST provides a request-response communication model built on the HTTP 1.1 protocol. Therefore, when multiple requests reach the server, it is bound to handle each of them, one at a time. However, gRPC follows a client-response model of communication for designing web APIs that rely on HTTP/2. Hence, gRPC allows streaming communication and serves multiple requests simultaneously. In addition to that, gRPC also supports unary communication similar to REST.

Data Exchange Format

REST typically uses JSON and XML formats for data transfer. However, gRPC relies on Protobuf for an exchange of data over the HTTP/2 protocol.

Serialization vs. Strong Typing

REST, in most cases, uses JSON or XML that requires serialization and conversion into the target programming language for both client and server, thereby increasing response time and the possibility of errors while parsing the request/response. However, **gRPC** provides strongly typed messages automatically converted using the Protobuf exchange format to the chosen programming language.

Latency

REST utilizing HTTP 1.1 requires a TCP handshake for each request. Hence, REST APIs with HTTP 1.1 can suffer from latency issues.

On the other hand, gRPC relies on HTTP/2 protocol, which uses multiplexed streams. Therefore, several clients can send multiple requests simultaneously without establishing a new TCP connection for each one. Also, the server can send push notifications to clients via the established connection.

Browser Support

REST APIs on HTTP 1.1 have universal browser support. However, gRPC has limited browser support because numerous browsers (usually the older versions) have no mature support for HTTP/2. So, it may require gRPC-web and a proxy layer to perform conversions between HTTP 1.1 and HTTP/2. Therefore, at the moment, gRPC is primarily used for internal services.

Code Generation Features

REST provides no built-in code generation features. However, we can use third-party tools like Swagger or Postman to produce code for API requests.

On the other hand, **gRPC**, using its protocol compiler, comes with native code generation features, compatible with several programming languages.

Conclusion

In this article, we compared two architectural styles for APIs, REST and gRPC.
We conclude that REST is handy in integrating microservices and third-party applications with the core systems.
However, gRPC can find its application in various systems like IoT systems that require lightweight message transmission, mobile applications with no browser support, and applications that need multiplexed streams.