𝗪𝗵𝗲𝗻 𝘁𝗼 𝘂𝘀𝗲 𝗚𝗿𝗮𝗽𝗵𝗤𝗟, 𝗴𝗥𝗣𝗖, 𝗮𝗻𝗱 𝗥𝗘𝗦𝗧  
  
Developers can pick from a variety of client-server communication protocols when it comes to designing an application. Utilizing GraphQL, gRPC, and REST is rather common in contemporary projects. Each protocol can provide a variety of advantages depending on the requirements of your application.  
  
𝗚𝗿𝗮𝗽𝗵𝗤𝗟 is a flexible approach for making data requests that focuses on specific requests and provides only what is necessary. The fact that GraphQL is client-driven distinguishes it from other APIs. Instead of handling it the standard way, where the client makes all the decisions. Its 𝗮𝗱𝘃𝗮𝗻𝘁𝗮𝗴𝗲𝘀 are that it is language agnostic, requests are made through a single endpoint, and it is strongly typed, as it has schemas.  
  
𝗥𝗘𝗦𝗧 is the most popular one. It is a great fit when a domain can be described as a set of resources. REST is a stateless architecture for data transfer. Some 𝗮𝗱𝘃𝗮𝗻𝘁𝗮𝗴𝗲𝘀 of REST are that is a well-established standard, it is simple to use, and have good caching support.  
  
𝗴𝗥𝗣𝗖 is a method that offers a lightweight and rapid system for obtaining data. Here, the primary distinction is in the way it describes its contract negotiations. It relies on contracts; the architecture is not what governs the negotiation; it is the relationship between the server and the client. While handling and calculations are delegated to a remote server housing the resource, the majority of power is used on the client side. Its main 𝗮𝗱𝘃𝗮𝗻𝘁𝗮𝗴𝗲𝘀 are that it has lightweight clients, it is high efficient as it uses protocol buffers to send/receive data, and its open source too.  
  
So, 𝘄𝗵𝗲𝗻 𝘁𝗼 𝗰𝗵𝗼𝗼𝘀𝗲 each of those protocols:  
  
✅ Use 𝗥𝗘𝗦𝗧 if you're building a CRUD-style web application or you work with well-structured data.  
  
✅ Use 𝗴𝗥𝗣𝗖 if your API is private and it's about actions. Also, if performances are important.  
  
✅ Use 𝗚𝗿𝗮𝗽𝗵𝗤𝗟 if you are having public API which needs to be flexible customizing requests and you want to add data from different sources into a public API.  
  
As you can see, each of these choices has specific uses and benefits. In this case, there is no clear winner, so what you should use—or, more , what you want to use—depends on your objectives and strategy.  
  
Image credits: LogRocket.  
  
\_\_\_\_\_\_\_  
If you like my posts please follow me [Dr. Milan Milanović](https://www.linkedin.com/in/ACoAAAC60lcBKfZlzaG-ANOWfpqXrwps720RKpc) and hit the 🔔 on my profile to get a notification for all my new posts.  
  
Grow with me 🚀!  
  
[#programming](https://www.linkedin.com/feed/hashtag/?keywords=programming&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7003297055443939328) [#softwareengineering](https://www.linkedin.com/feed/hashtag/?keywords=softwareengineering&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7003297055443939328) [#apidesign](https://www.linkedin.com/feed/hashtag/?keywords=apidesign&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7003297055443939328) [#techworldwithmilan](https://www.linkedin.com/feed/hashtag/?keywords=techworldwithmilan&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7003297055443939328) [#api](https://www.linkedin.com/feed/hashtag/?keywords=api&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7003297055443939328)

