**What is the paper about?**

Paper is about *API architecture* bases and styles, which were developed throughout life. It describes 4 basic API, explaining strong and weak sides, and highlight the scenarios where each of them fits best.

**What is a server / client stub, in the context of RPC?**

The server stub and client stub take care of the serialization and deserialization of the *parameters*. RPC uses GET to fetch information and POST for everything else. The mechanics of the interaction between a server and a client come down to calling an endpoint and getting a response.

**What does it mean to be integrated with WS-security protocols?**

One of the soap pros is *the number of security extensions*. Integrated with the WS-Security protocols, SOAP meets an enterprise-grade transaction quality. It provides privacy and integrity inside the transactions while allowing for encryption on the message level. WS-Security protocols need to be flexible enough to accommodate new security mechanisms and provide alternative mechanisms if an approach is not suitable. Because SOAP-based messages traverse multiple intermediaries, security protocols need to be able to identify fake nodes and prevent data interpretation at any nodes. WS-Security combines the best approaches to tackle different security problems by allowing the developer to customize a particular security solution for a part of the problem.

**How do you understand HATEOAS?**

It is short for *Hypertext As The Engine of Application State.* Basically, it means that with each response, a REST API provides metadata linking to all the related info about how to use the API. That’s what enables decoupling the client and the server. As a result, both API provider and API consumer can evolve independently without hindering their communication.

**What are subscriptions? Why would we need them?**

GraphQL has subscriptions allowing for real-time notifications from the server.

Subscriptions are a way to push data from the server to the clients that choose to listen to real-time messages from the server. Subscriptions are similar to queries in that they specify a set of fields to be delivered to the client, but instead of immediately returning a single answer, a result is sent every time a particular event happens on the server without the client needing to resend that request. This can be achieved when the client opens a bi-directional communication channel with the server and sends a subscription query that specifies which event it is interested in. When an event is triggered, the server executes the stored GraphQL query, and the result is sent through the same communication channel opened by the client. One way to achieve this is by using WebSockets, which establish a two-way interactive communication session between the user’s browser and a server. The client can unsubscribe by sending a message to the server. The server can also unsubscribe at any time due to errors or timeout. Subscriptions create a ton of possibilities for creating truly dynamic applications.

**Order the API patterns by message size.**

1. **RPC**
2. **GraphQL**
3. **Rest**
4. **SOAP**

**Which API pattern would be best fit your laboratory work? Why?**

I think the best approach would be to use the REST API and RPC for my laboratory work. The main reason is that code for each service can evolve at its own pace, without affecting other services. Therefore, REST is an especially good fit for my project. It easy to develop and clearly for anyone what happened when you want to send one request from one service to another. It very useful that I can receive the response, due this I can understand what I should do next in order to achieve one or another result.