PubSub-RPC (PSRPC)

PubSub-RPC (Publish-Subscribe Remote Procedure Call) is a protocol that combines the concepts of remote procedure call (RPC) and publish-subscribe messaging. It enables services or applications to communicate with each other through a publish-subscribe pattern, while still maintaining the ability to make RPCs.

In the PubSub-RPC model, a service can publish messages to a topic, and other services can subscribe to that topic to receive the messages. In addition, services can also define and expose remote procedures that can be called by other services.

When a service makes an RPC call, it includes the name of the remote procedure, along with any necessary arguments. The PubSub-RPC system then routes the call to the appropriate service, which can process the request and return a response.

PubSub-RPC is designed to be lightweight and efficient, with a focus on low latency and high throughput. It can be used in a variety of distributed systems and microservices architectures, and provides a flexible way for services to communicate with each other.

<https://github.com/livekit/psrpc>

The psrpc library is a lightweight RPC (Remote Procedure Call) framework for Go that allows developers to define service methods and endpoints, and to call those methods remotely over a network connection. The library is built on top of the standard Go net/rpc package and provides additional features such as HTTP support, pluggable codecs, and transport layers.

In the LiveKit project, psrpc is used to implement the signaling server, which is responsible for coordinating connections between clients and establishing the initial session parameters. The signaling server exposes a set of RPC methods that clients can call to join a room, leave a room, or exchange messages with other participants in the room.

Create custom protobuf-based golang RPCs built on pub/sub.

Supports:

* Protobuf service definitions
* Use Redis, Nats, or a local communication layer
* Custom server selection for RPC handling based on user-defined [affinity](https://github.com/livekit/psrpc#Affinity)
* RPC topics - any RPC can be divided into topics, (e.g. by region)
* Single RPCs - one request is handled by one server, used for normal RPCs
* Multi RPCs - one request is handled by every server, used for distributed updates or result aggregation
* Queue Subscriptions - updates sent from the server will only be processed by a single client
* Subscriptions - updates sent be the server will be processed by every client

Understanding psrpc code:

bus.go:

This file contains the implementation of the message bus interface.