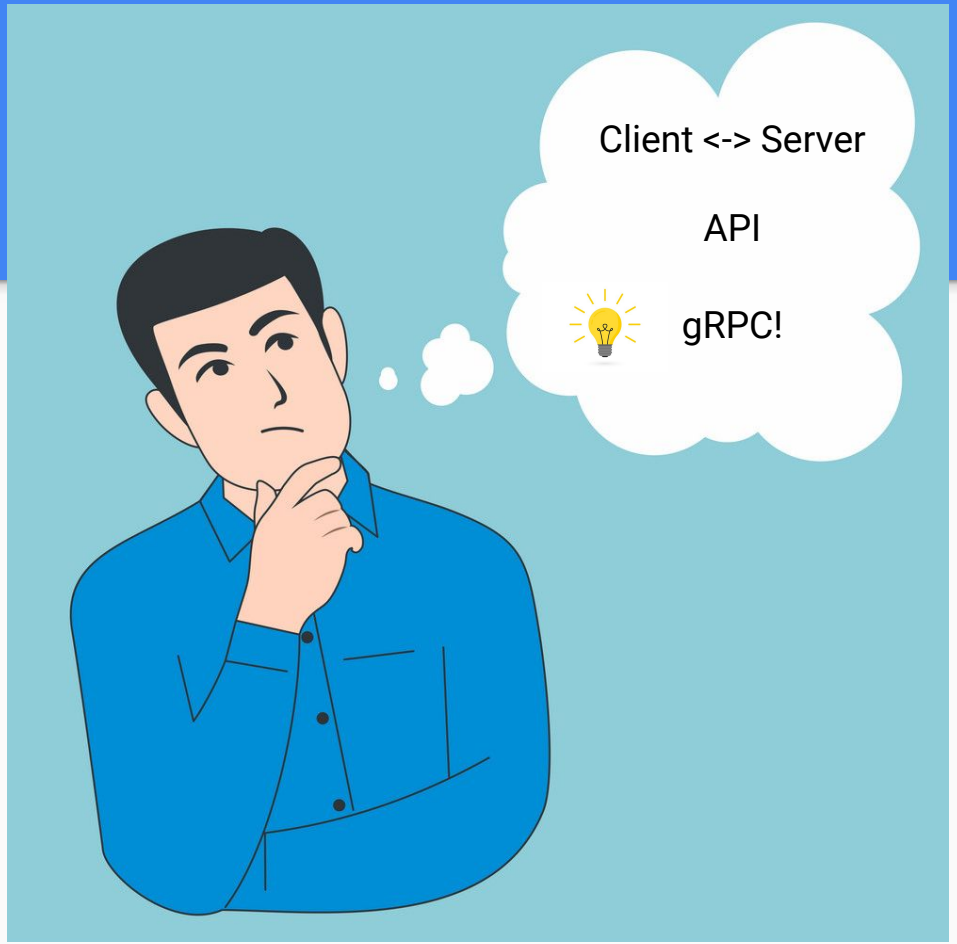
Proto3 transport formats and protocols

Anatoly Rugalev, Backend Teamlead at Stream
with support of Alexei Kozlenok, Director of Engineering at Stream


Client <-> Server
API
gRPC!

WRONG!

Or is it..?

Why is it good?

- Source of truth
- Generated ready-to-go clients
- Source of documentation
- Quick development cycle

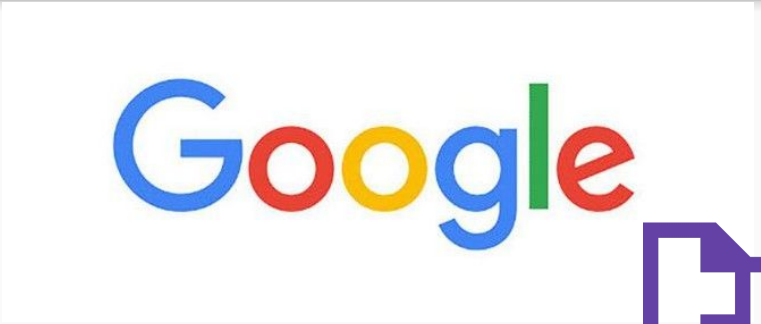
```
// The greeter service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
message HelloReply {
  string message = 1;
}
```

Why stop here?

- Not ideal for web
- Load balancing caveats

What options do we have?

- gRPC
- Twirp
- dRPC



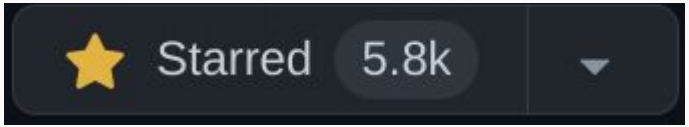
STORJ

Twirp

- Supports `http/1.1` and `http/2`
- Standard `net/http` server
- Supports `application/json` out of the box
- 13 languages supported
- Does not support streaming

twirp

https://github.com/twitchtv/twirp



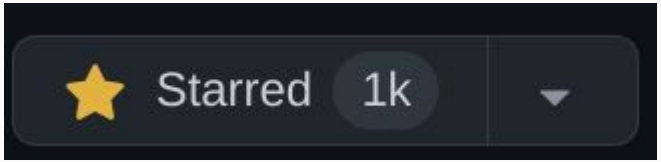
dRPC

- Simplified wire protocol
- Supports http/1.1 and `http/2`
- Supports `application/json` out of the box
- Supports `grpc-web` clients
- Supports `Twirp` clients
- Go only

STORJ

DRPC

https://github.com/storj/drpc



Features

	http/1.1	http/2	Streams	Web	OpenAPI gen
gRPC	No	Yes	Yes	With proxy	Yes
Twirp	Yes	No	No	Yes	Yes
dRPC	Yes	Yes	Yes	Yes	Yes

Cross Compatibility

		Client			
		gRPC	Twirp	dRPC	grpc-web
Server	gRPC	Yes	No	No	No
	Twirp	No	Yes	No	No
	dRPC	No	Yes	Yes	Yes

Conclusion

- Twirp and dRPC have less friction
  - Built in JSON support
  - http/1.1 support
- dRPC is especially a good choice for Go-centric environments
- You don’t always need gRPC to build proto3 defined API