

go-grpc-channelz

@rantav

Gophercon IL Feb 2020

whoami?



Quick recap

- Efficient and robust
- Polyglot
- Feature rich
- Protocol Buffers
- HTTP/2

Connecting to a gRPC server

```
import "google.golang.org/grpc"
```

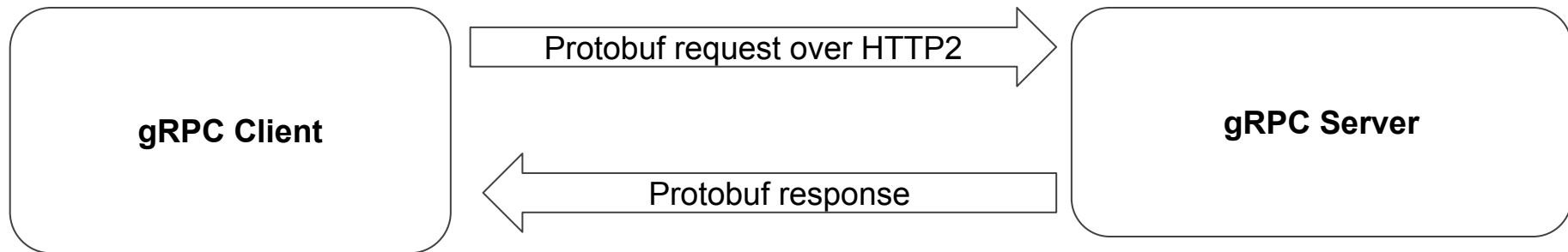
```
grpc.Dial("localhost:9999")
```

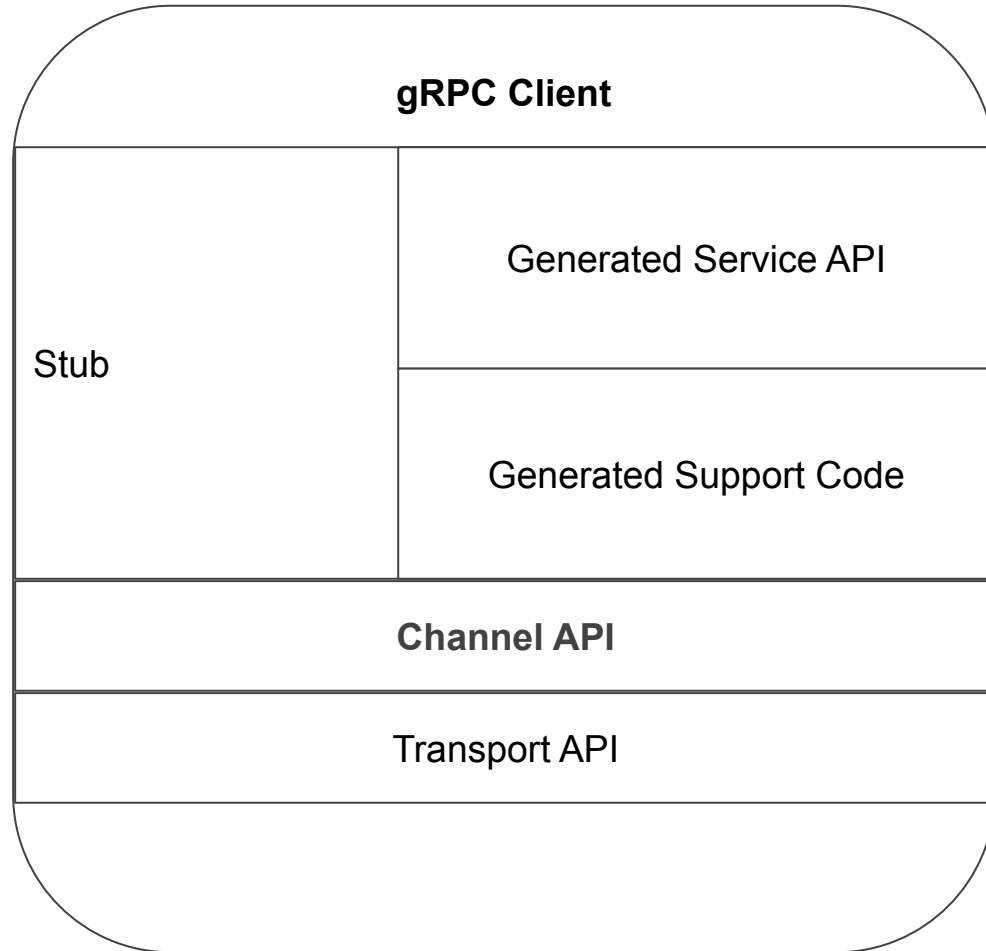
```
grpc.Dial("x.dns.af.com:9999")
```

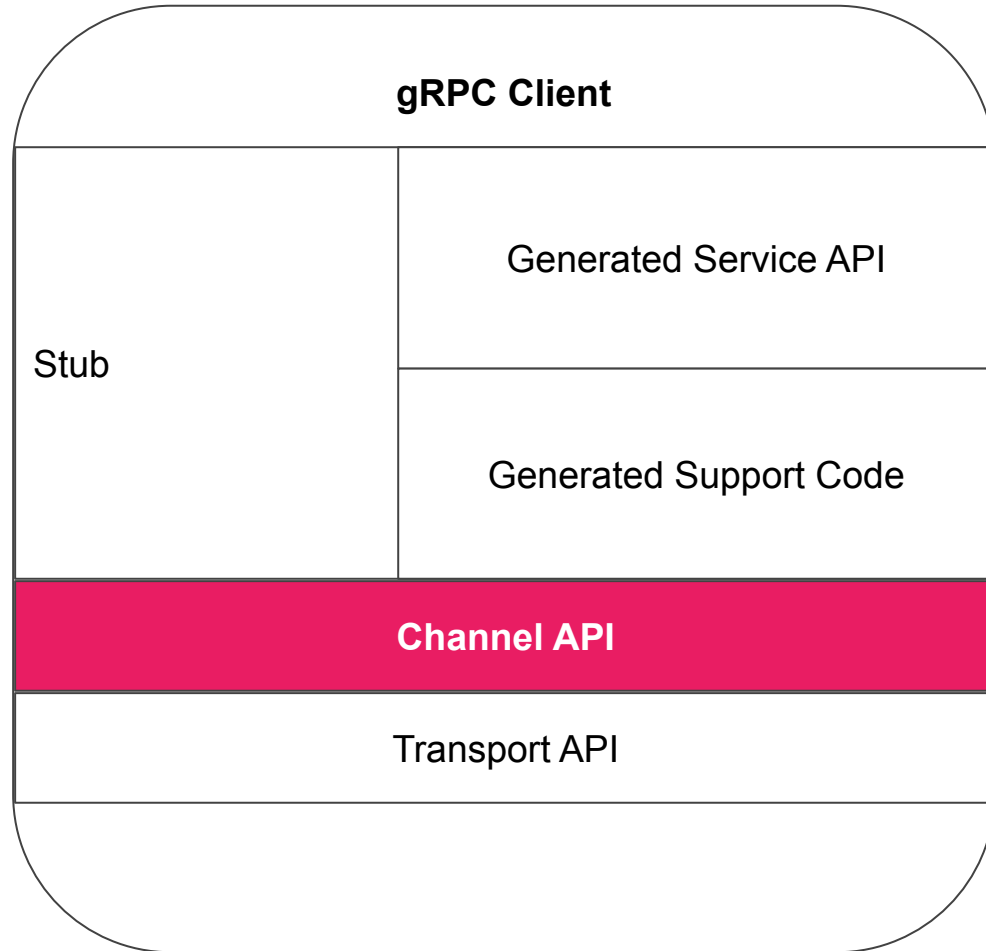
```
grpc.Dial("consul://127.0.0.1:8500/my-service")
```

```
grpc.Dial("consul://127.0.0.1:8500/my-service?tag=t&dc=us1")
```

gRPC Channels







gRPC Client

Stub

Generated Service API

Generated Support Code

Channel API

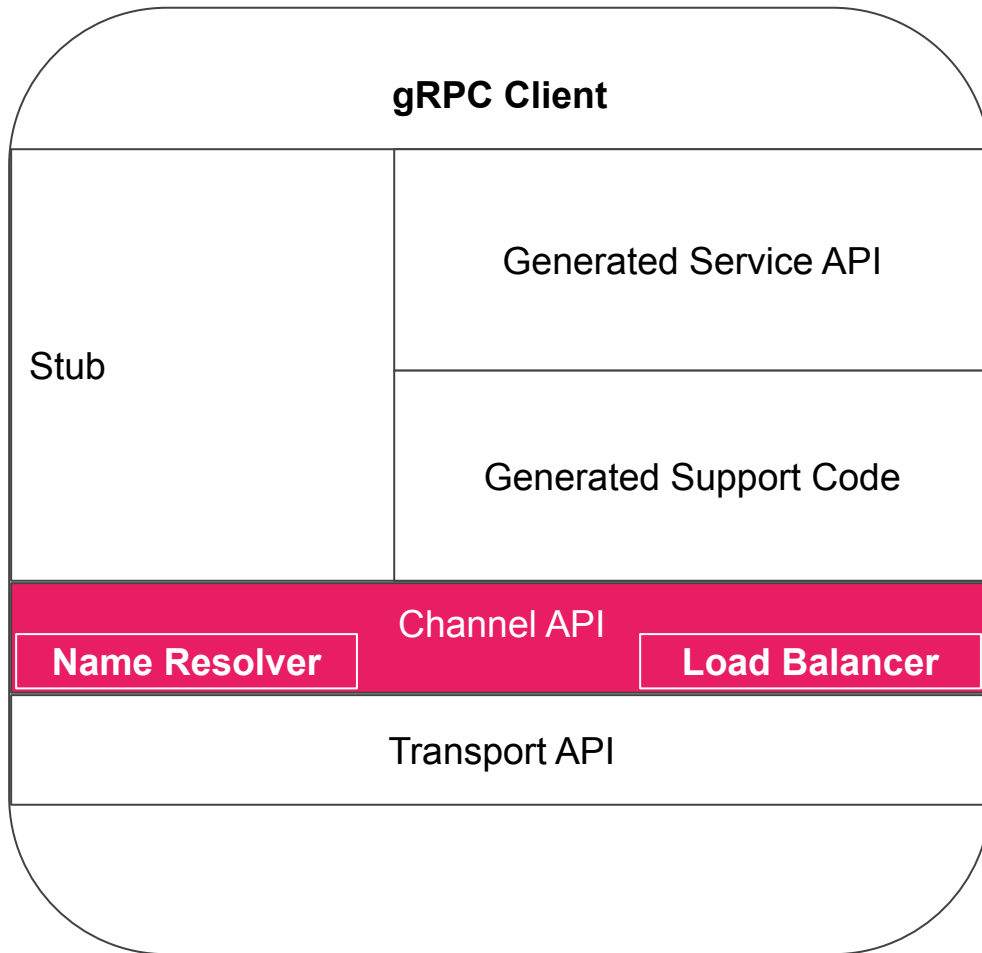
Name Resolver

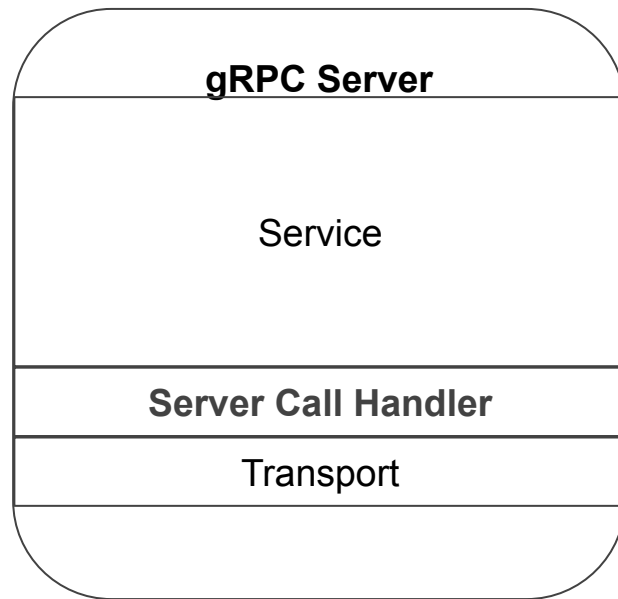
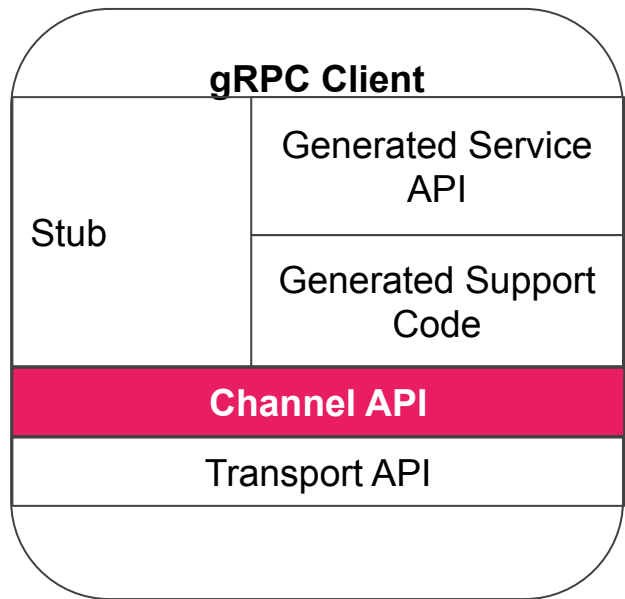
Load Balancer

Transport API

Pluggable
name resolver

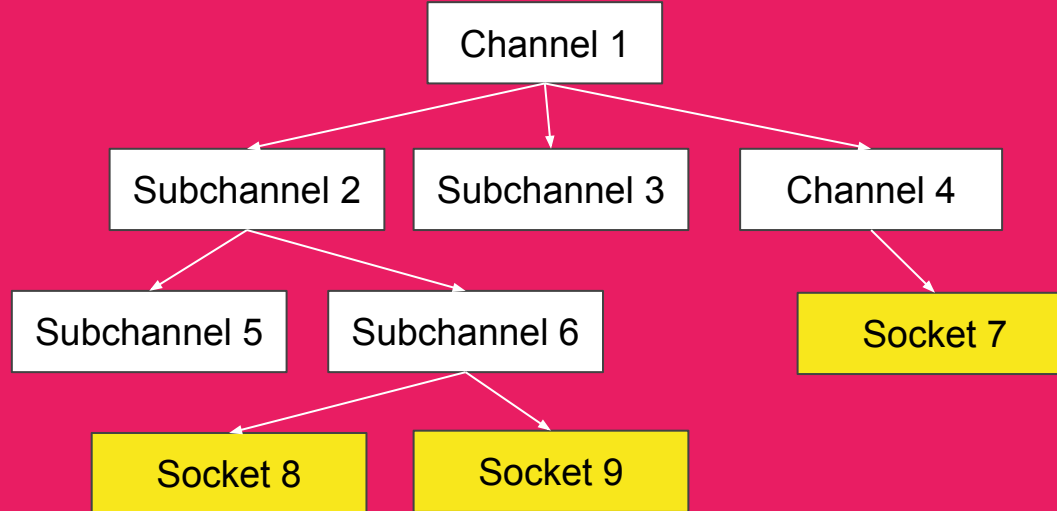
Pluggable
load balancer



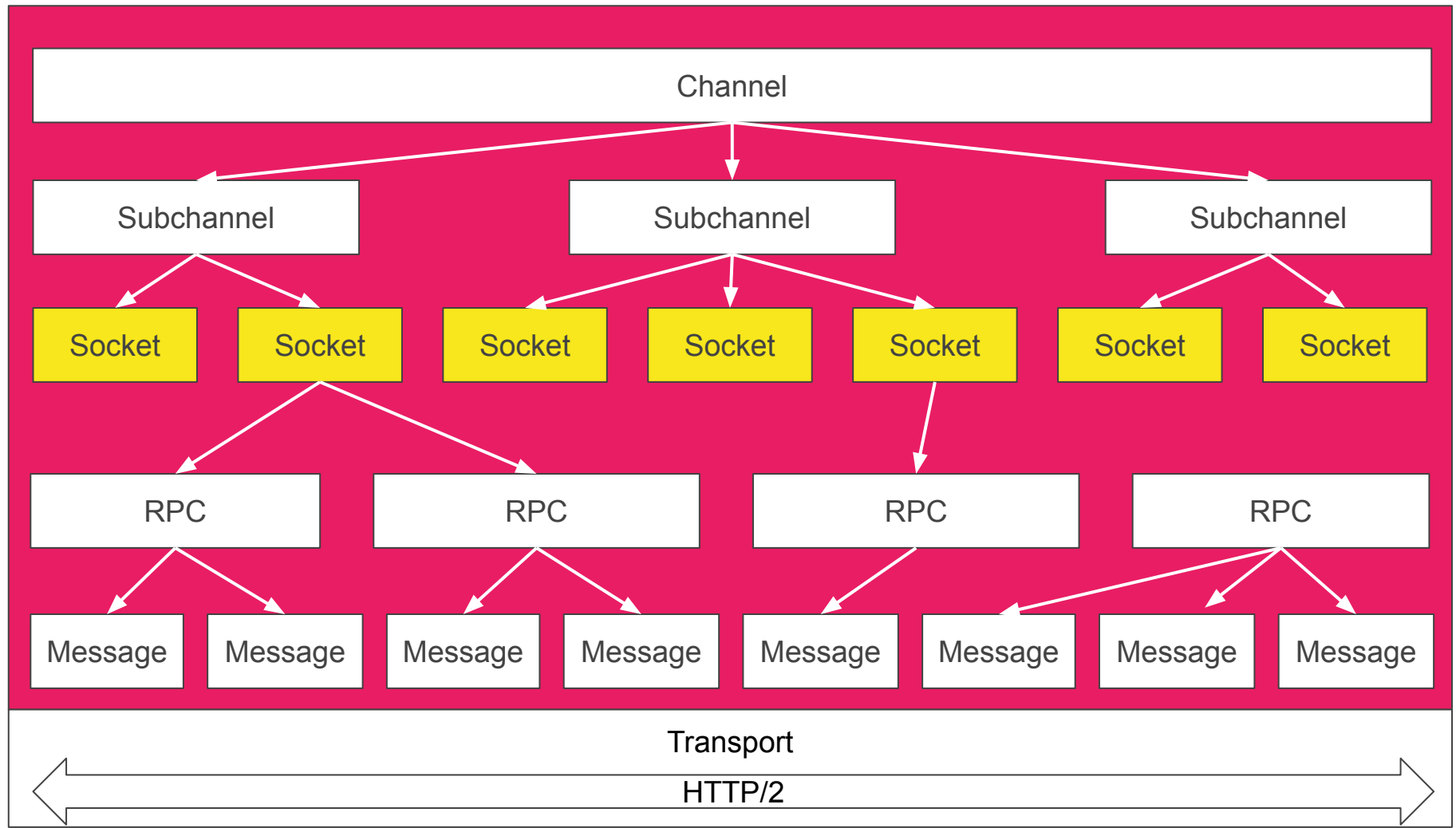


Stub

Channel API



Transport API



Gosh, that's complicated!

[illegible]



localhost:8080/_/zpages/tracez



Trace Spans

Span Name	Running	Latency Samples										Error Samples
		[>0s]	[>10μs]	[>100μs]	[>1ms]	[>10ms]	[>100ms]	[>1s]	[>10s]	[>1m0s]		
DeleteRecords	<u>0</u>	<u>3</u>	<u>3</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	
GetClicksFromDB	<u>0</u>	<u>3</u>	<u>3</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	
GetTransactionIdsFromDB	<u>0</u>	<u>0</u>	<u>0</u>	<u>3</u>	<u>3</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	
attr_endpoint.EngagementLookup.Healthcheck	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	
attr_endpoint.EngagementLookup.LocalLookupAndDelete	<u>0</u>	<u>0</u>	<u>0</u>	<u>3</u>	<u>3</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	
attr_endpoint.EngagementLookup.LookupAndDelete	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>3</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	

Channelz to the rescue

Channelz service

```
// Channelz is a service exposed by gRPC servers that provides detailed debug
// information.
service Channelz {
    // Gets all root channels (i.e. channels the application has directly
    // created). This does not include subchannels nor non-top level channels.
    rpc GetTopChannels(GetTopChannelsRequest) returns (GetTopChannelsResponse);
    // Gets all servers that exist in the process.
    rpc GetServers(GetServersRequest) returns (GetServersResponse);
    // Returns a single Server, or else a NOT_FOUND code.
    rpc GetServer(GetServerRequest) returns (GetServerResponse);
    // Gets all server sockets that exist in the process.
    rpc GetServerSockets(GetServerSocketsRequest) returns (GetServerSocketsResponse);
    // Returns a single Channel, or else a NOT_FOUND code.
    rpc GetChannel(GetChannelRequest) returns (GetChannelResponse);
    // Returns a single Subchannel, or else a NOT_FOUND code.
    rpc GetSubchannel(GetSubchannelRequest) returns (GetSubchannelResponse);
    // Returns a single Socket or else a NOT_FOUND code.
    rpc GetSocket(GetSocketRequest) returns (GetSocketResponse);
}
```

But we want a UI

Pretty Please 🙏

Introducing:

go-grpc-channelz

ChannelZ Stats

Clients

Top Channels: 2

Channel	State	Target	Subchannels	Child Channels	Sockets	CreationTimestamp	CallsStarted	CallsSucceeded	CallsFailed
2 localhost:8080	state:READY	localhost:8080	4			2020-01-07T14:47:45Z	6	6	0
7 localhost:8080	state:READY	localhost:8080	8			2020-01-07T14:48:51Z	1	0	0

Servers

Servers: 1

Server	CreationTimestamp	CallsStarted	CallsSucceeded	CallsFailed	LastCallStartedTimestamp	Sockets
1		8	7	0	2020-01-07T14:48:51Z	3 [::]:8080

[Channelz Spec](#)

ChannelZ channel 2

ChannelId	2
Channel Name	localhost:8080
State	state:READY
Target	localhost:8080
Subchannels	<u>4</u>
Child Channels	
Sockets	
CreationTimestamp	2020-01-07T14:47:45Z
CallsStarted	112
CallsSucceeded	112
CallsFailed	0
LastCallStartedTimestamp	2020-01-07T15:07:35Z
Events	<pre>CT_INFO [2020-01-07T14:47:45Z]: Channel Created CT_INFO [2020-01-07T14:47:45Z]: Resolver state updated: {Addresses:[{Addr:localhost:8080 CT_INFO [2020-01-07T14:47:45Z]: Channel switches to new LB policy "pick_first" CT_INFO [2020-01-07T14:47:45Z]: Subchannel(id:4) created CT_INFO [2020-01-07T14:47:45Z]: Channel Connectivity change to CONNECTING CT_INFO [2020-01-07T14:47:45Z]: Channel Connectivity change to READY</pre>

ChannelZ subchannel 4

Subchannel	4
State	state:READY
Target	localhost:8080
CreationTimestamp	2020-01-07T14:47:45Z
CallsStarted	119
CallsSucceeded	119
CallsFailed	0
LastCallStartedTimestamp	2020-01-07T15:08:45Z
Child Channels	
Child Subchannels	
Socket	<u>5 [::1]:50066 -> [::1]:8080</u>
Events	<pre>CT_INFO [2020-01-07T14:47:45Z]: Subchannel Created CT_INFO [2020-01-07T14:47:45Z]: Subchannel Connectivity change to CONNECTING CT_INFO [2020-01-07T14:47:45Z]: Subchannel picks a new address "localhost:8080" to connect CT_INFO [2020-01-07T14:47:45Z]: Subchannel Connectivity change to READY</pre>

ChannelZ socket 9

SocketId	9
Socket Name	:::1]:50071 -> :::1]:8080
Socket Local -> Remote	tcpip_address: -> tcpip_address:
StreamsStarted	7
StreamsSucceeded	6
StreamsFailed	0
MessagesSent	7
MessagesReceived	6
KeepAlivesSent	0
LastLocalStreamCreated	2020-01-07T15:09:21Z
LastRemoteStreamCreated	0001-01-01T00:00:00Z
LastMessageSent	2020-01-07T15:09:21Z
LastMessageReceived	2020-01-07T15:09:19Z
LocalFlowControlWindow	65535
RemoteFlowControlWindow	65528
Options	
Security	

ChannelZ server 1

ServerId	1
Server Name	
CreationTimestamp	
CallsStarted	134
CallsSucceeded	133
CallsFailed	0
LastCallStartedTimestamp	2020-01-07T15:09:54Z
Sockets	<u>3 [::]:8080</u>

Simple integration

— — —

```
// Activate.
```

```
channelzservice.RegisterChannelzServiceToServer(grpcServer)
```

```
// Register web handler
```

```
http.Handle("/", channelz.CreateHandler("/_", grpcBindAddress))
```

Summary and Key Takeaways

- gRPC is robust and mature
- Visibility is not ~~optional~~, it is **required**
- go-grpc-channelz provides a much needed visibility

github.com/rantav/go-grpc-channelz