gRPC

A shorter way to Microservice Architecture?

Balazs Nagy

Senior Field Applications Engineer at NI

CLA, CLED, CTD

balazs.nagy@ni.com

Microservice Architecture – Intro

gRPC – Intro

gRPC – As architectural building block

gRPC – Use cases

gRPC – Use in LabVIEW

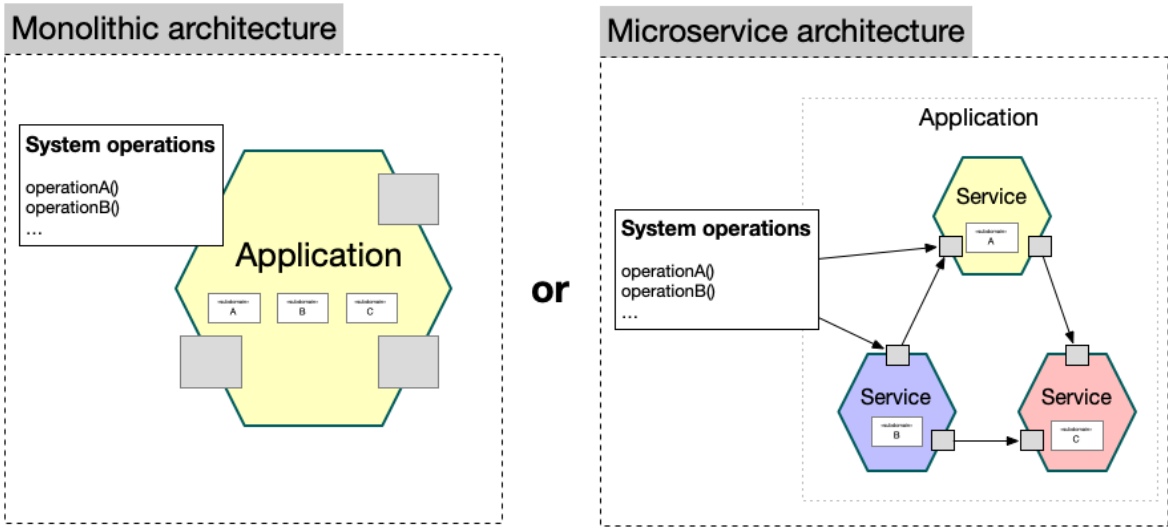
Microservice Architecture – Intro

When you outgrow your monolithic architecture

Sometimes an application can outgrow its monolithic architecture and become an obstacle to rapid, frequent and reliable software delivery.

This typically happens when the application becomes large and complex and is developed by many teams. For example, its deployment pipeline become a bottleneck.

When this occurs, you should consider migrating to microservices.



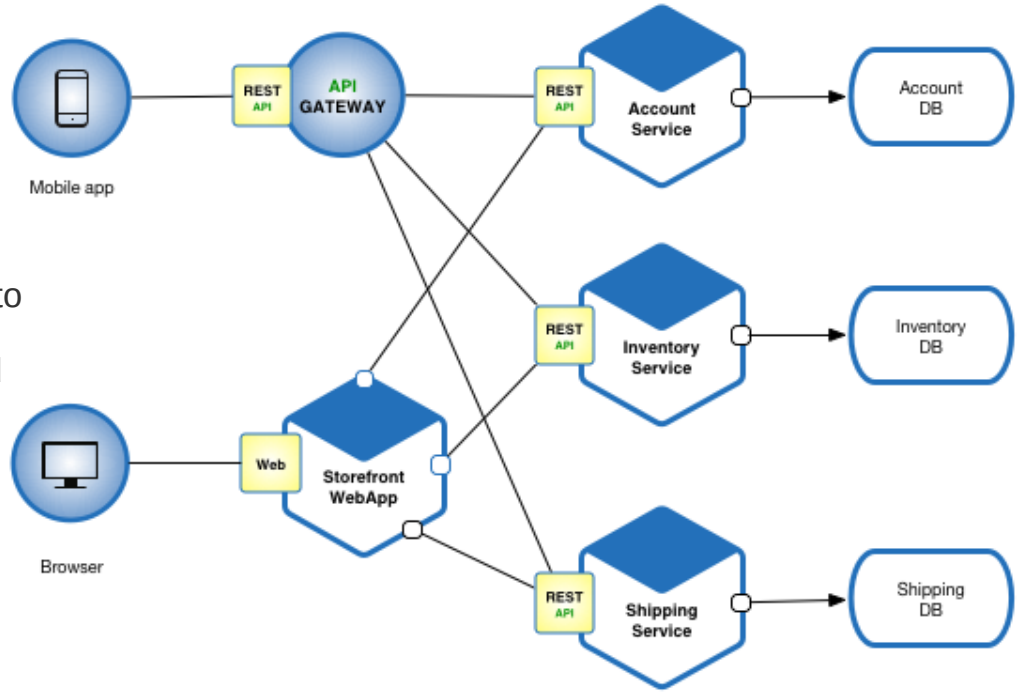
https://microservices.io/

What are microservices?

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are:

- Independently deployable
- Loosely coupled
- Organized around business capabilities
- Owned by a small team

The microservice architecture enables an organization to deliver large, complex applications rapidly, frequently, reliably and sustainably - a necessity for competing and winning in today's world.



https://microservices.io/

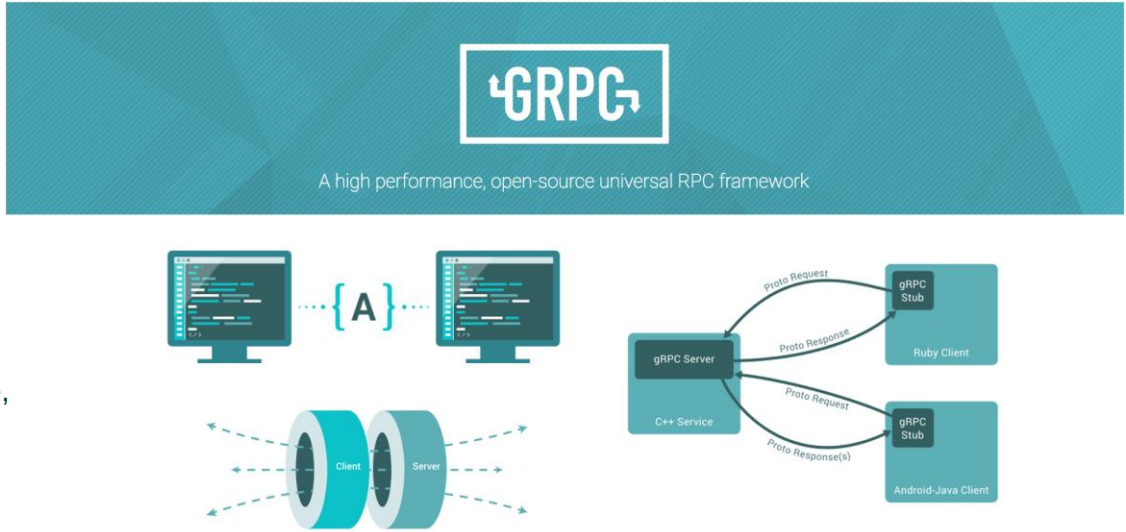
gRPC – Intro

gRPC Overview

gRPC is a Google-invented, open-source, remote procedure technology built for cloud process communication.

It offers many advantages such as:

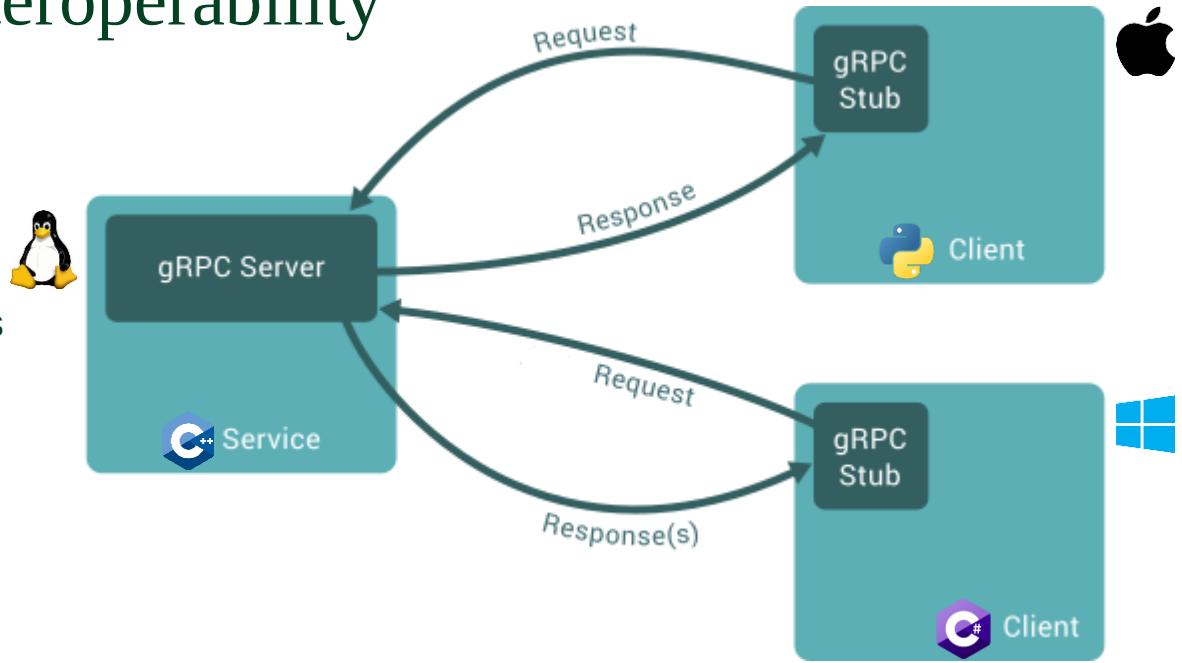
- OS-Agnostic
- Language Agnostic
- Transport Layer Agnostic (ethernet, PCIe, Reflective Memory, etc.)
- High Bandwidth Streaming Enabled
- Secure (HTTP2)
- Open Source
- Can Directly Call Drivers or Application SW
- Works with native error handling



https://grpc.io/docs/what-is-grpc/introduction/

Key Feature - Interoperability

- Communicate across processes or systems
- Client and server can be on different platforms and written in different languages
- Supported Languages
  - C#
  - C++
  - Dart
  - Go
  - Java
  - Kotlin
  - LabVIEW
  - Node
  - Objective-C
  - PHP
  - Python
  - Ruby

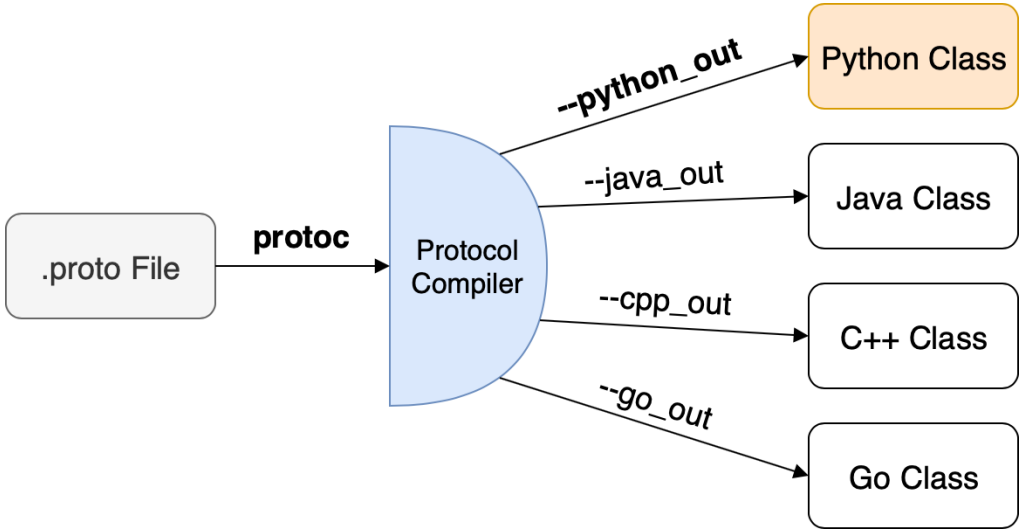


https://grpc.io/docs/languages/

Key Feature - Cross-language

.proto file compiles to language of choice for client/server stubs, message definitions, etc.

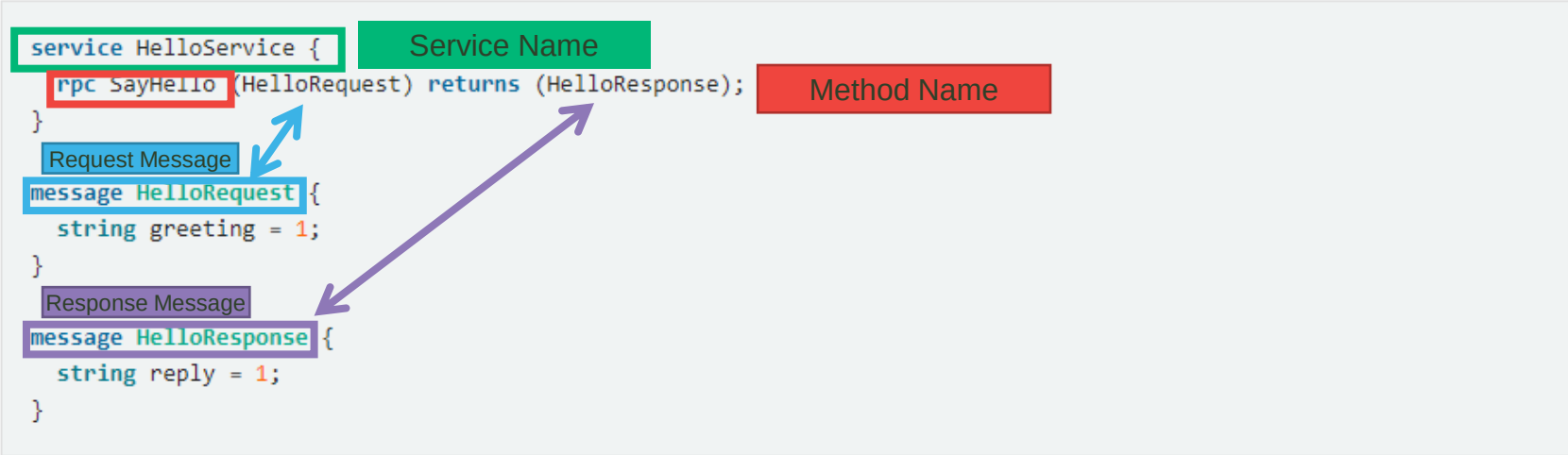
.proto file compiles to different languages for client(s) and server



https://www.grpcgenerator.com/

What is a Proto File?

The ProtoBuf interface describes the structure of the data to be sent. Payload structures are defined as “messages” in a .proto file.



Review the gRPC **Core concepts, architecture and lifecycle** document here [https://grpc.io/docs/what-is-grpc/core-concepts/] for more details.

High-Level Workflow of gRPC

gRPC Server(s)

Transport layer

gRPC Client(s)

Services
Methods
Request + Response

LabVIEW

.proto File

gRPC

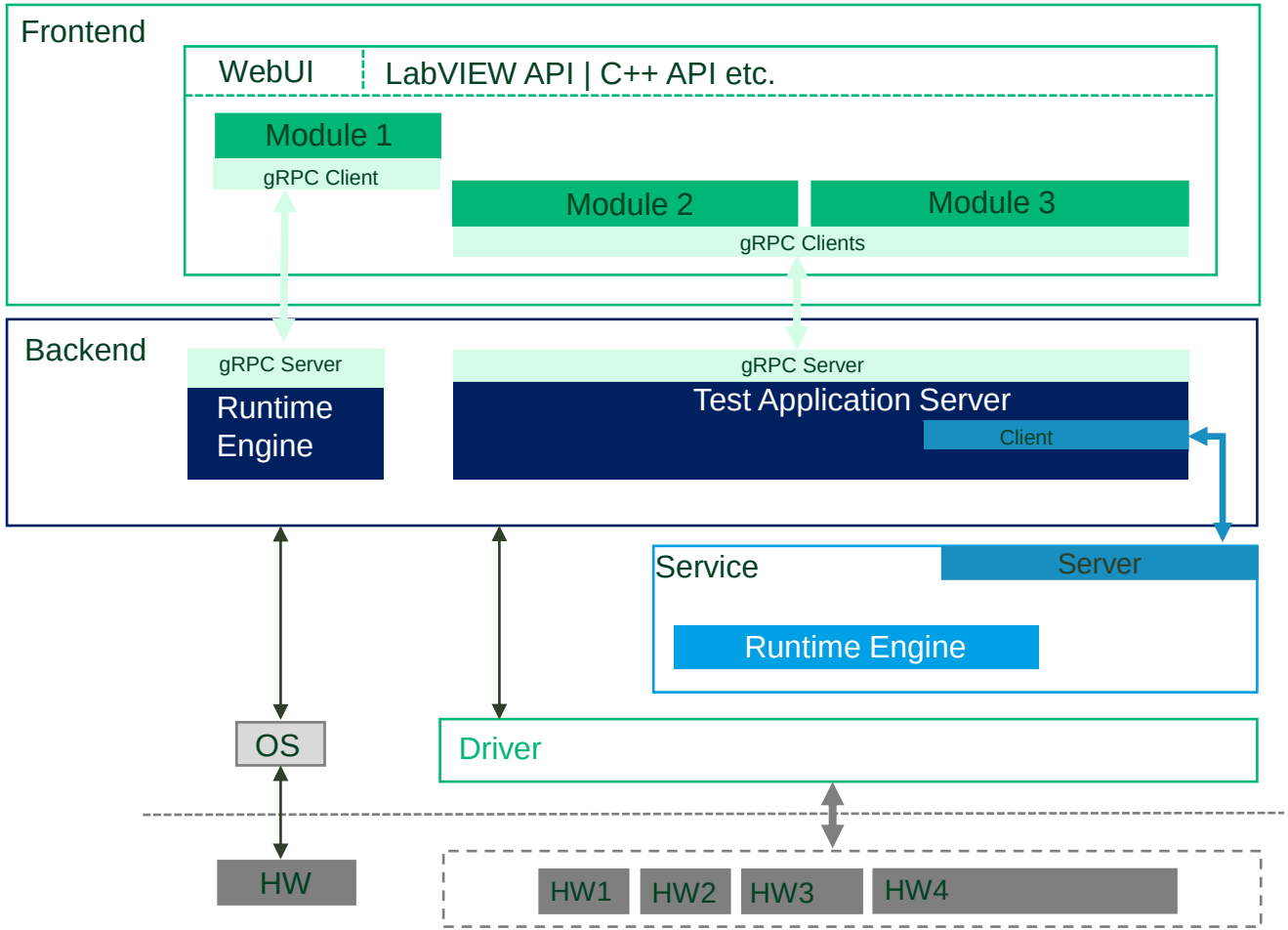
Contains all the details about the commands/queries/messages (called methods and messages)

Auto-creates a **full client API** ready for that programming language

gRPC – As architectural building block

gRPC – Use cases

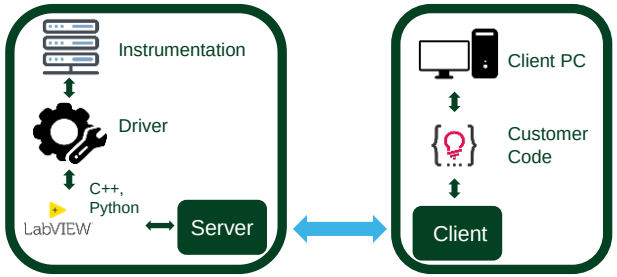
gRPC – As architectural building block



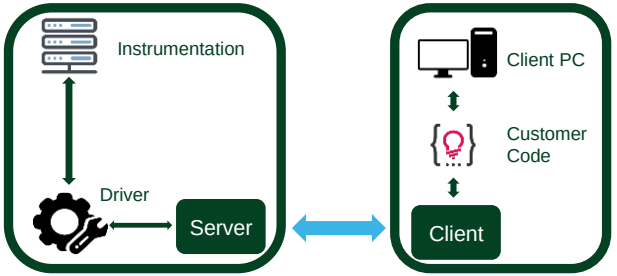
gRPC – Use cases

gRPC – Applications Examples

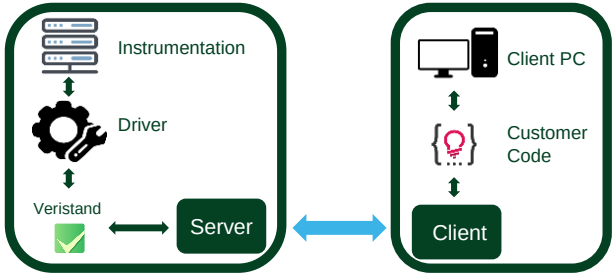
Workflow 1: Allow to control test execution and **pass data** through LabVIEW **between Tester and Client machine**.



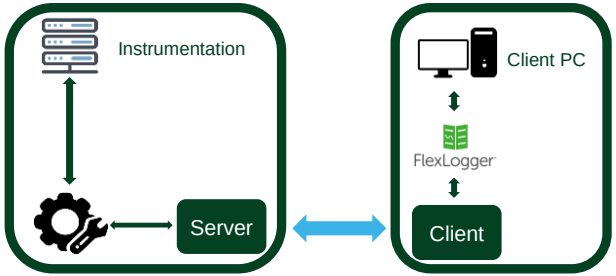
Workflow 2: Allow to make **driver calls remotely**.



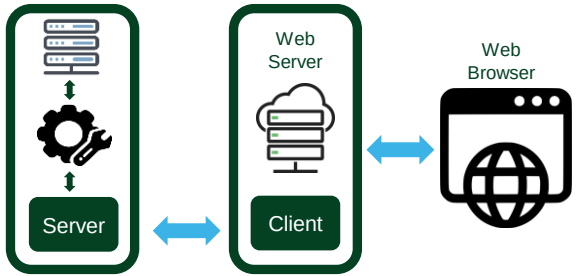
Workflow 3: Allow to **remotely control App Software**



Workflow 4: Allow to **remotely control instrumentation** from App Software

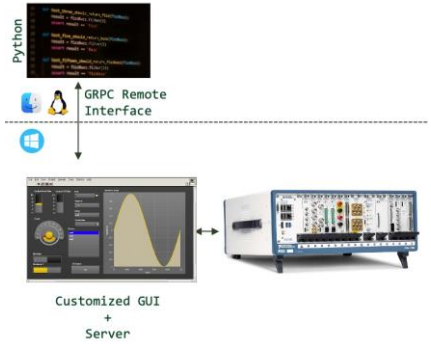


Workflow 5: **Web-based instrument control**



gRPC – Detailed Applications Examples

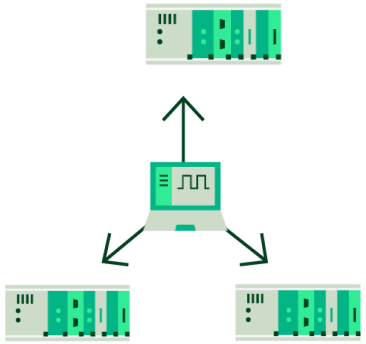
LabVIEW Server Support



Examples:
Remote interface for LV Application

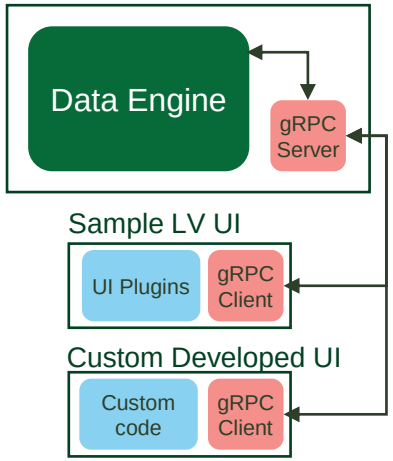
Benefits:
- Easily accessible through e.g. Python
- Will open the door for fully remote applications.

Device Driver Support



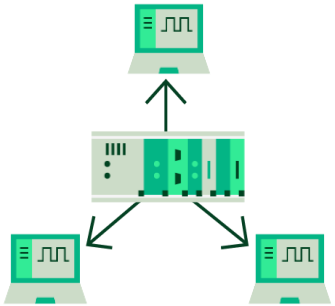
Examples:
Multiple instruments being controlled by a single test machine

Benefits:
- No drivers installed on test machine
- Can integrate NI instruments without rework



Examples:
gRPC Server is added to the Datalogger to expose a set of functionality to users.

Benefits:
- Light-weight client.
- Flexibility due to range of supported OS and Languages.
- Quick timeline turnaround due to minimal rework



Examples:
Different groups are sharing the same equipment in a lab.

Benefits:
- Single Driver install on server
- Groups can use different languages and OS's

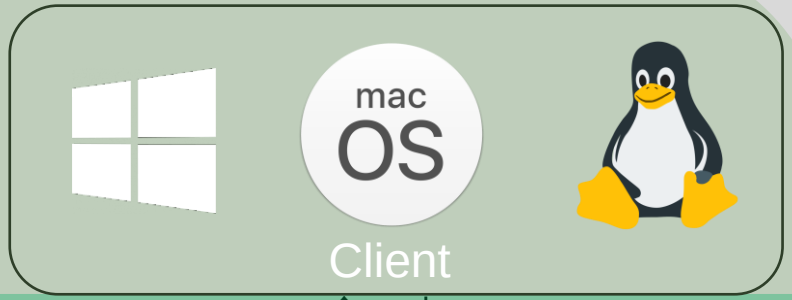
gRPC App. Ex. 1: NI gRPC Device Server and Client APIs

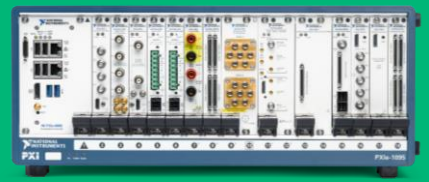
Use test systems from anywhere, with any language, on any OS

OS, Language Agnostic

Transport layer Agnostic

Low Latency, Highly Scalable



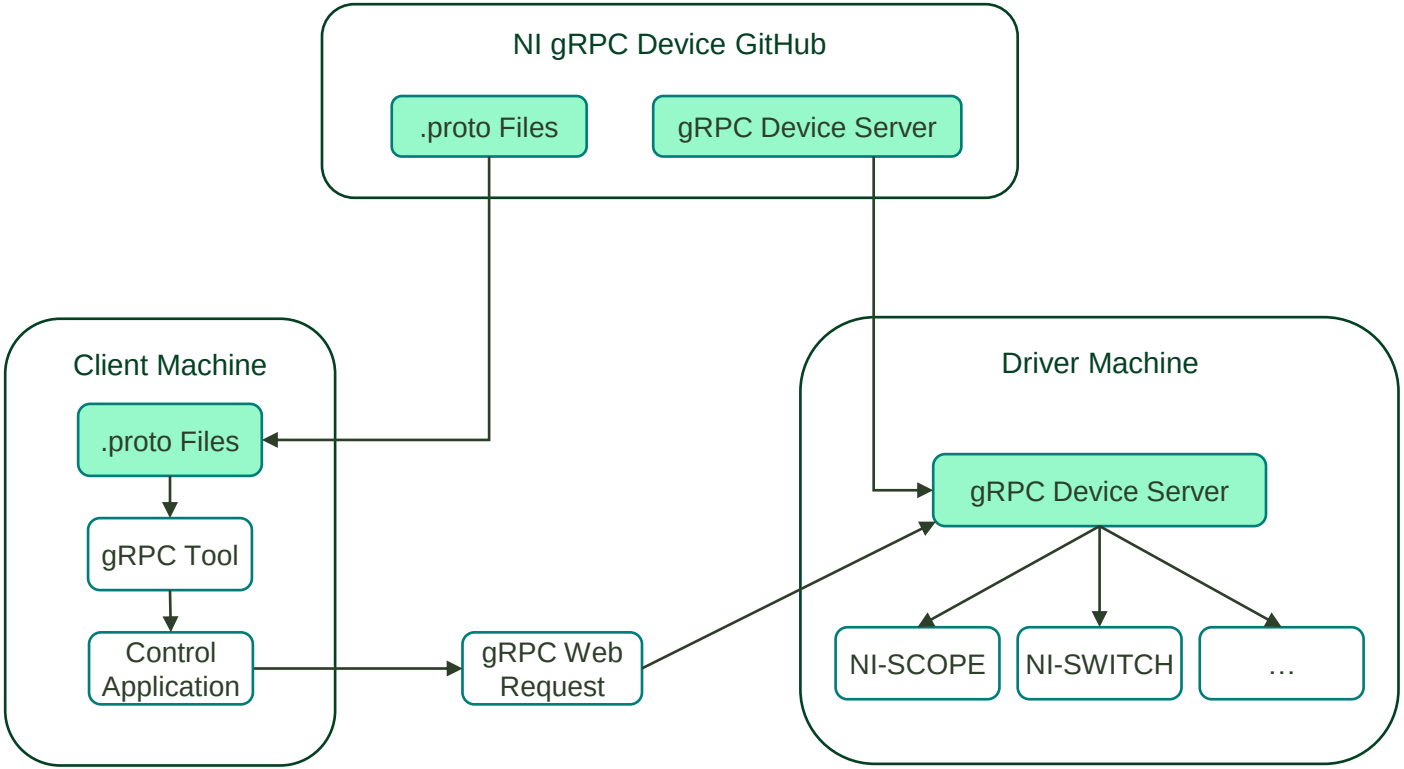


https://github.com/ni/grpc-device

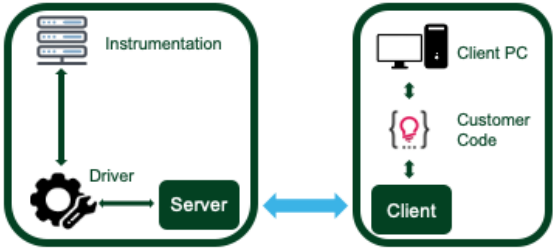
Benefits

- Remotely control NI Hardware and Software
- Minimize time to first measurement
- Leverage existing workflows
- Avoid driver installation on client

NI gRPC Device Server and Client APIs



NI Provided Components for gRPC Device Server and Client APIs



- gRPC Server
  - The gRPC Server is installed on the machine directly connected to the test instrumentation.
  - The gRPC Server provides remote clients with a link to the actual instrument drivers installed on the tester.
  - A single gRPC Server can support multiple instrument drivers.
- Remote gRPC API for each supported driver
  - NI will provide a gRPC .proto file that describes the API client machines can use in order to interact with their instrumentation. This API will mirror the C API of the driver.
  - The .proto file can be used to generate the remote API into any programming language you prefer.
    - Ex.) The .proto file for the NI Scope driver will allow you to use the remote NI Scope API in C#, Python, Go, and many other languages.
- Basic instrument discovery capability
  - User on client side will be able to programmatically query for instruments connected to the gRPC Server.

Current Support

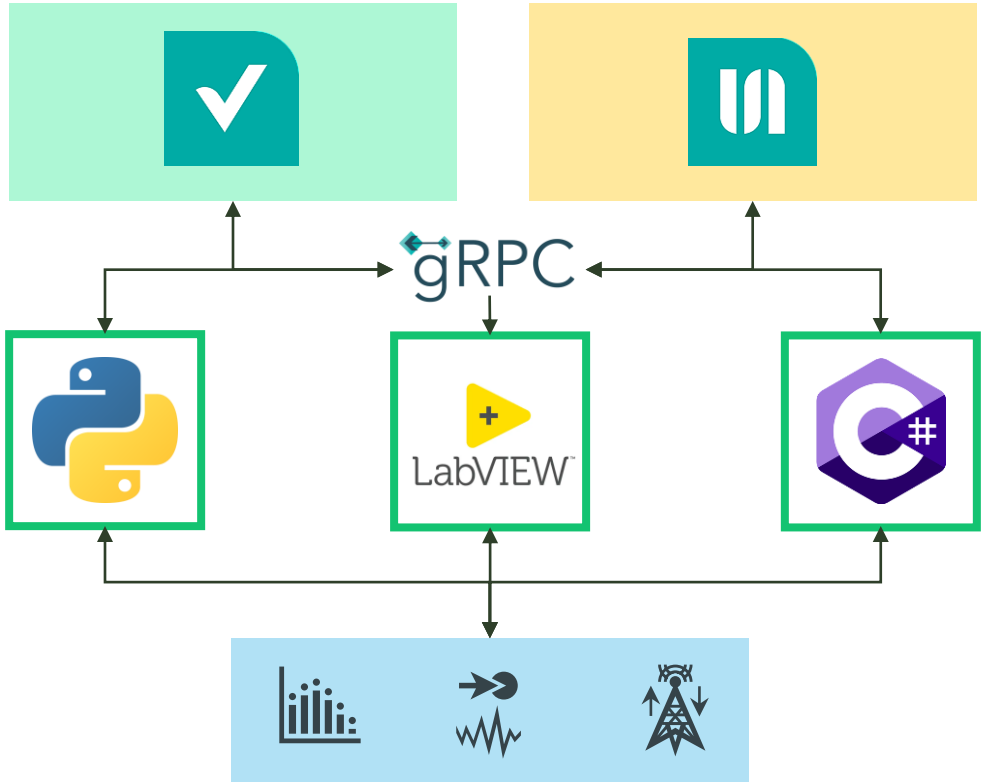
- NI-DAQmx (in-progress)
- NI-DCPower
- NI-Digital Patter
- NI-DMM
- NI-FGEN
- NI-SCOPE
- NI-SWITCH
- NI-Sync
- NI-TClk

Future Support

- RF Drivers
- NI-XNet
- NI VRTS

NI MeasurementLink, NI InstrumentStudio, NI TestStand

gRPC App. Ex. 2: Plug-ins Shared Between Applications

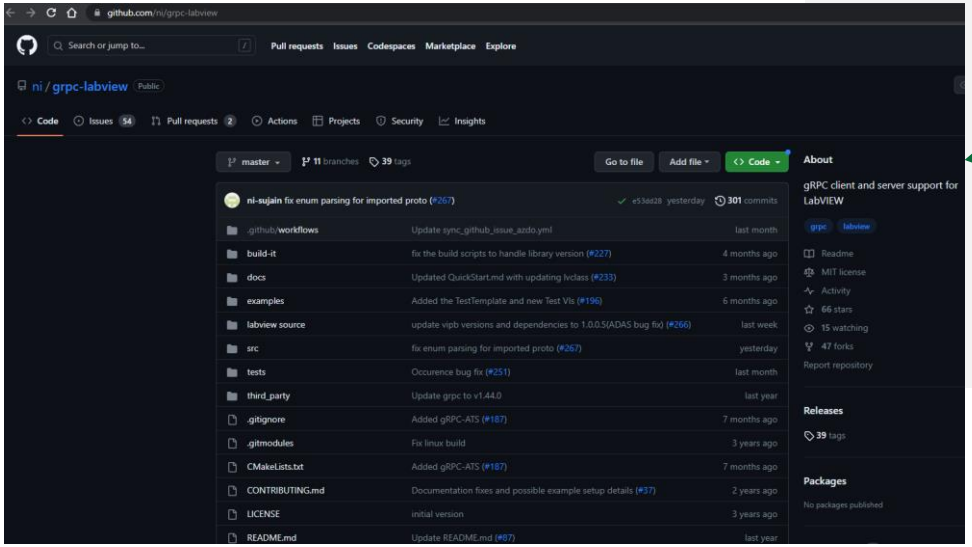


gRPC – Use in LabVIEW

Custom LV gRPC Server: Development Workflow

1. Download and Install the gRPC tools for LabVIEW: https://github.com/ni/grpc-labview
2. Define Messages for communication between Sever and Client
3. Create and Validate Protofile
4. Generate LV gRPC Server Interface
5. Generate Client Interface
6. Integrate LV gRPC Server Interface into existing LabVIEW Codebase
7. Integrate Client Interface into Client Codebase

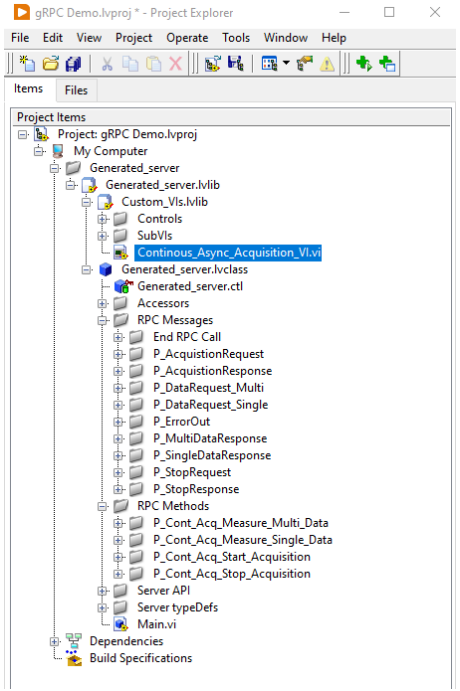
Where to get gRPC for LabVIEW?

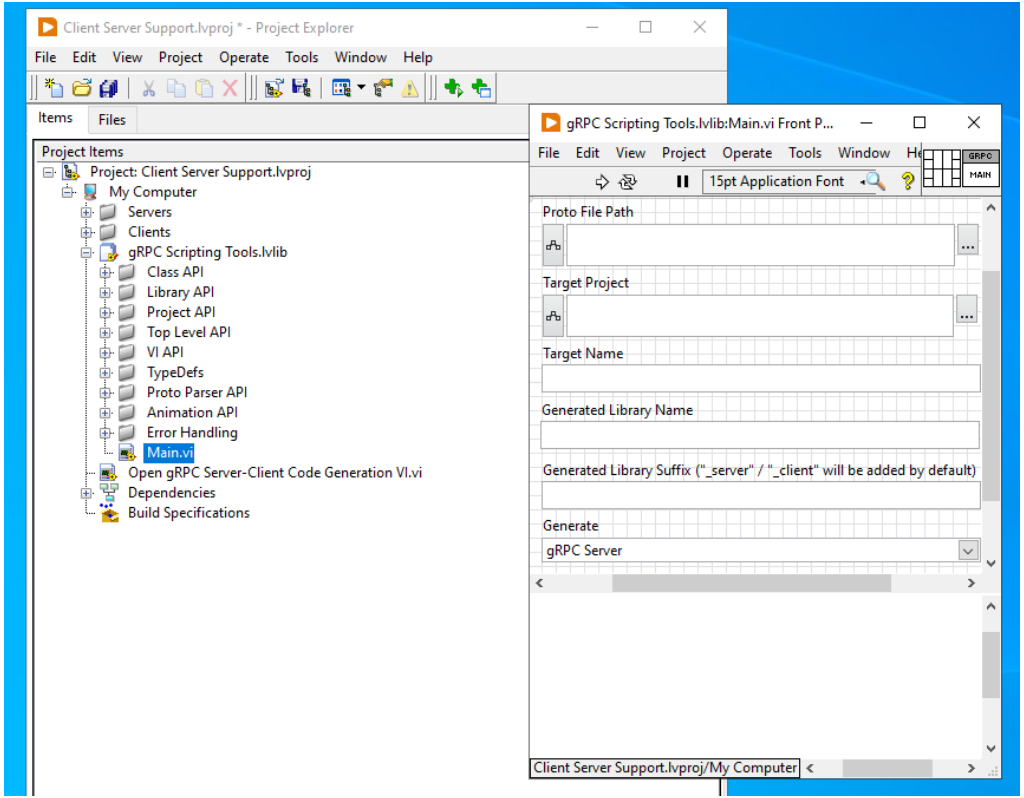


https://github.com/ni/grpc-labview

https://www.vipm.io/package/ni_lib_grpc_labview/

LabVIEW Server / Client code generation based on .proto file with Scripting Tool

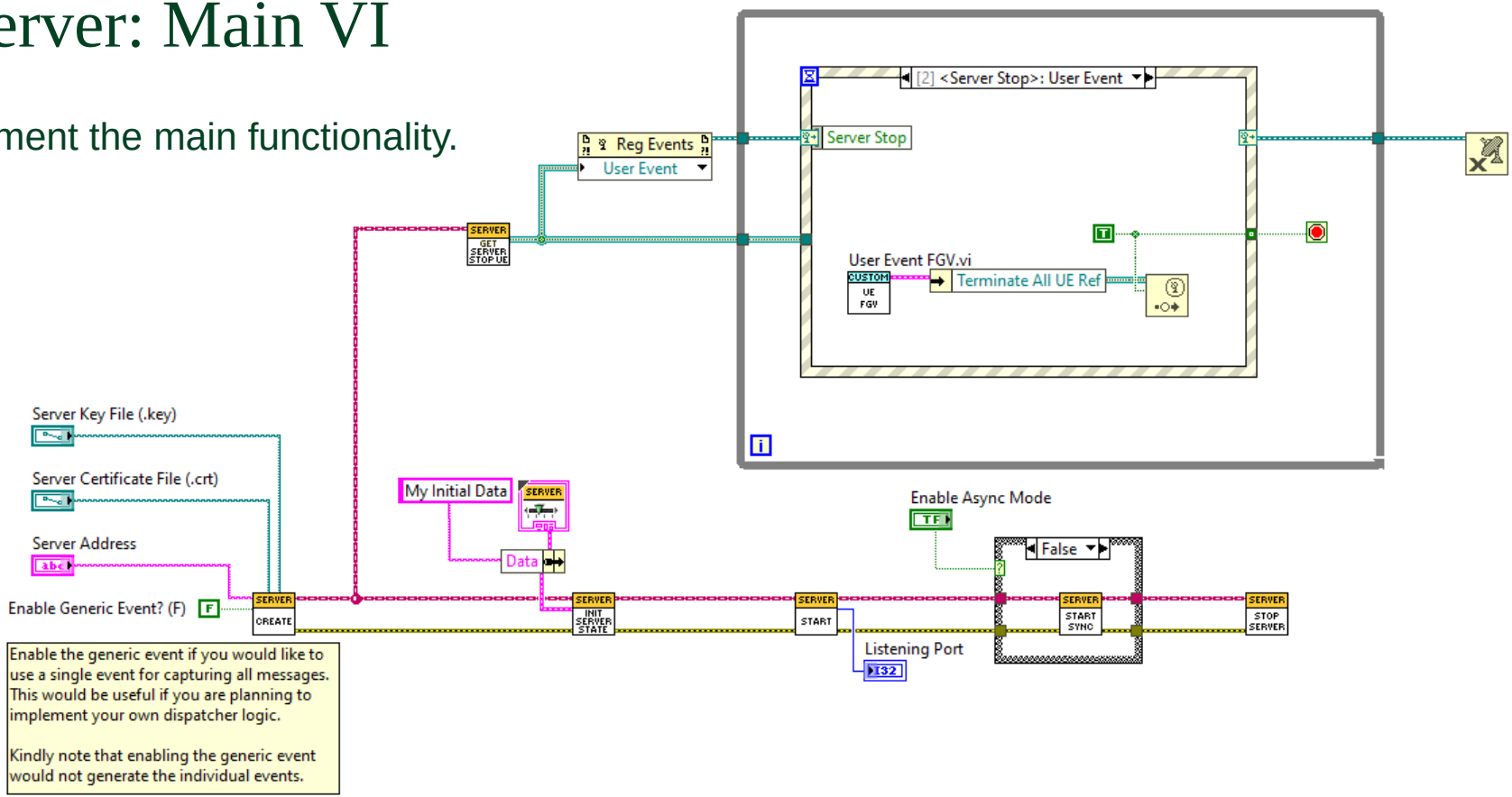




https://github.com/ni/grpc-labview

LV Server: Main VI

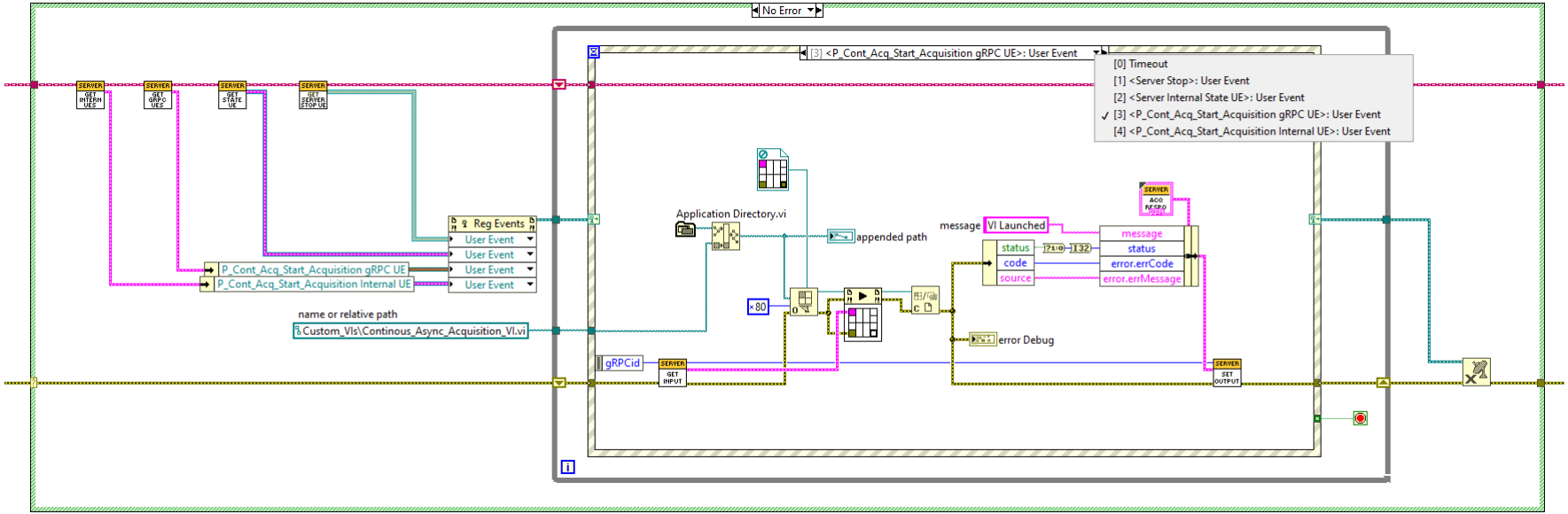
To implement the main functionality.



LV Server: Method VI

To implement the functionality when a gRPC client makes a request.

e.g. send a start acqusition request to the suitable thread, send response the client.



Additional References

- NI gRPC-Device Repo
  - https://github.com/ni/grpc-device
- gRPC Repo
  - https://github.com/grpc/grpc
- NI gRPC Device Wiki:
  - https://github.com/ni/grpc-device/wiki
- NI Device gRPC Server and Client API Wiki:
  - https://github.com/ni/grpc-device/wiki
- gRPC Overview
  - https://grpc.io/
- Proto language guide
  - https://developers.google.com/protocol-buffers/docs/proto3
- Getting started with GoogleTest
  - http://google.github.io/googletest/

Thank you!

Questions?