Envoy Mobile:
From Server to Multiplatform Library

KubeCon - November 2019

Michael Schore
@goaway

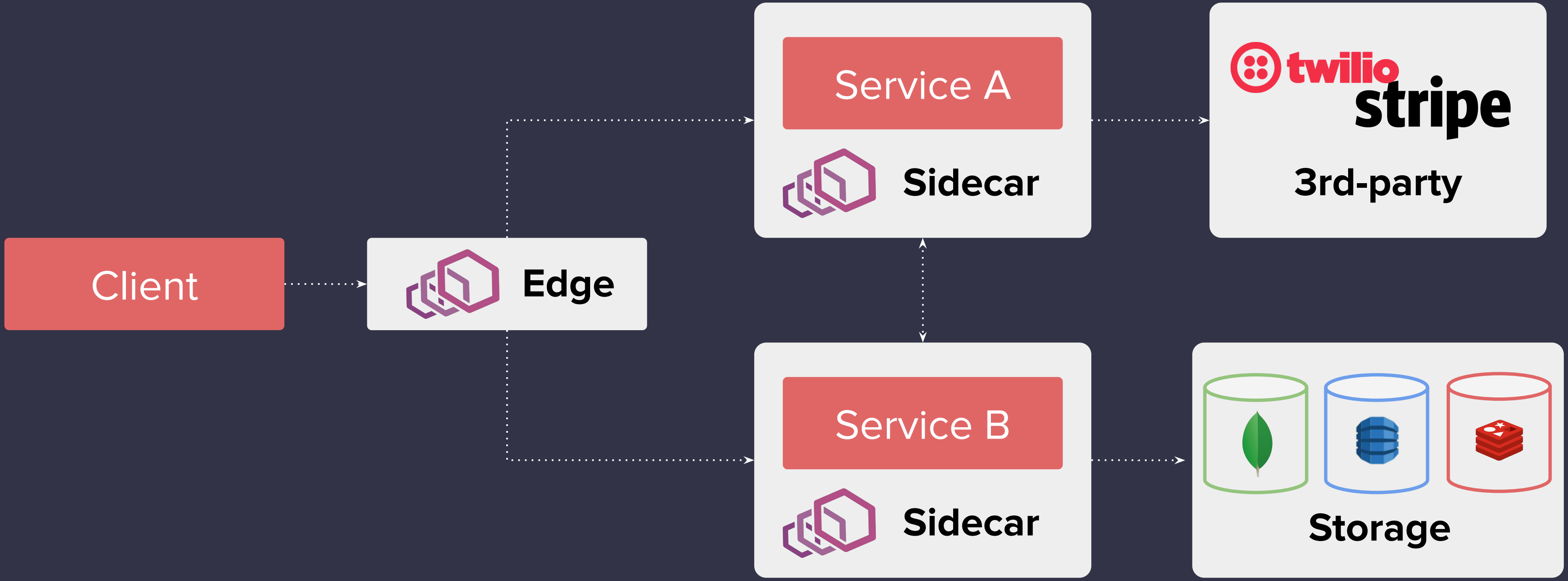
Jose Nino
@junr03

Agenda

- Why bring Envoy to Mobile?
- Envoy as a Library
- Where are we now?
- Onwards!

Why bring Envoy ...to Mobile?

Topology 2.0: Universal Network Primitive

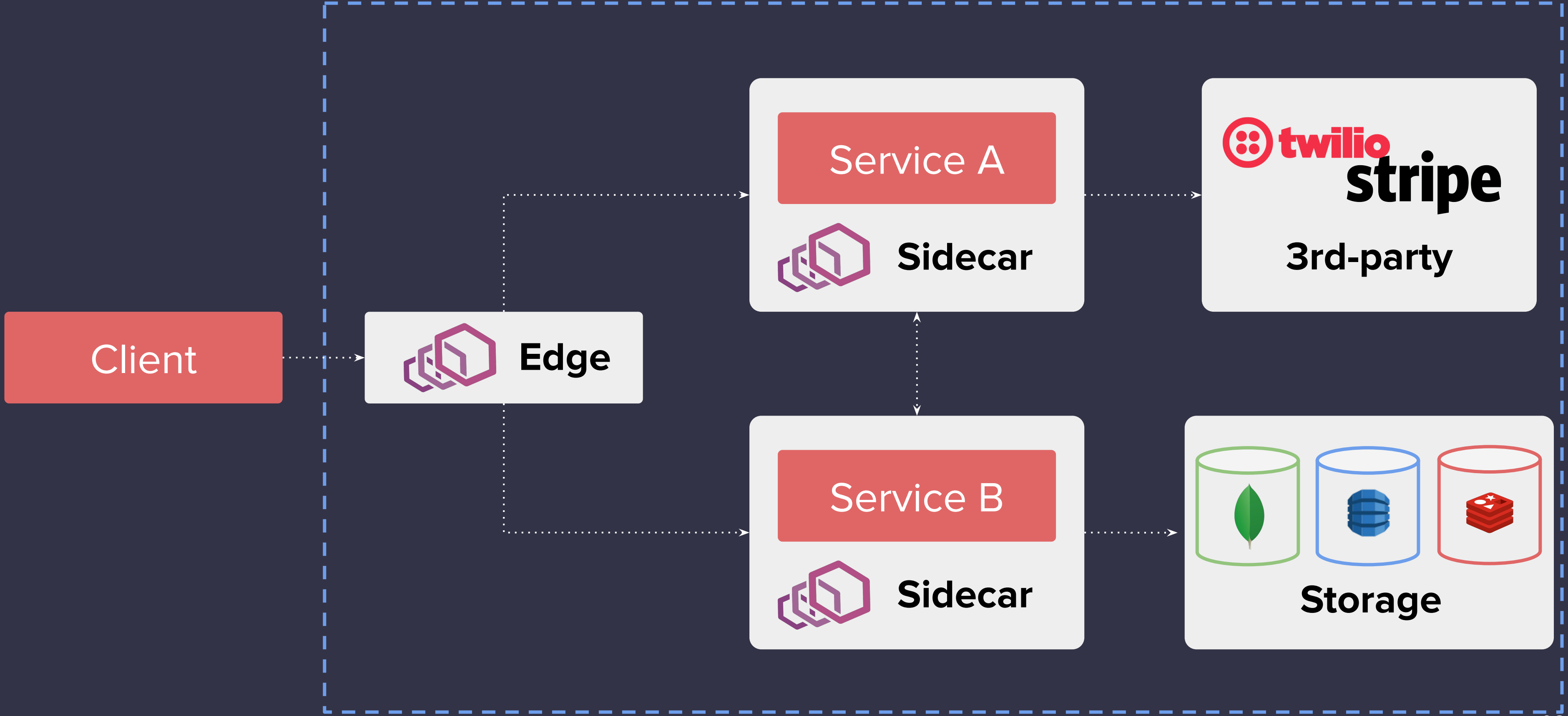


What are we solving for?

Three 9s at the server-side edge is meaningless if the user of a mobile application is only able to complete the desired product flows a fraction of the time.

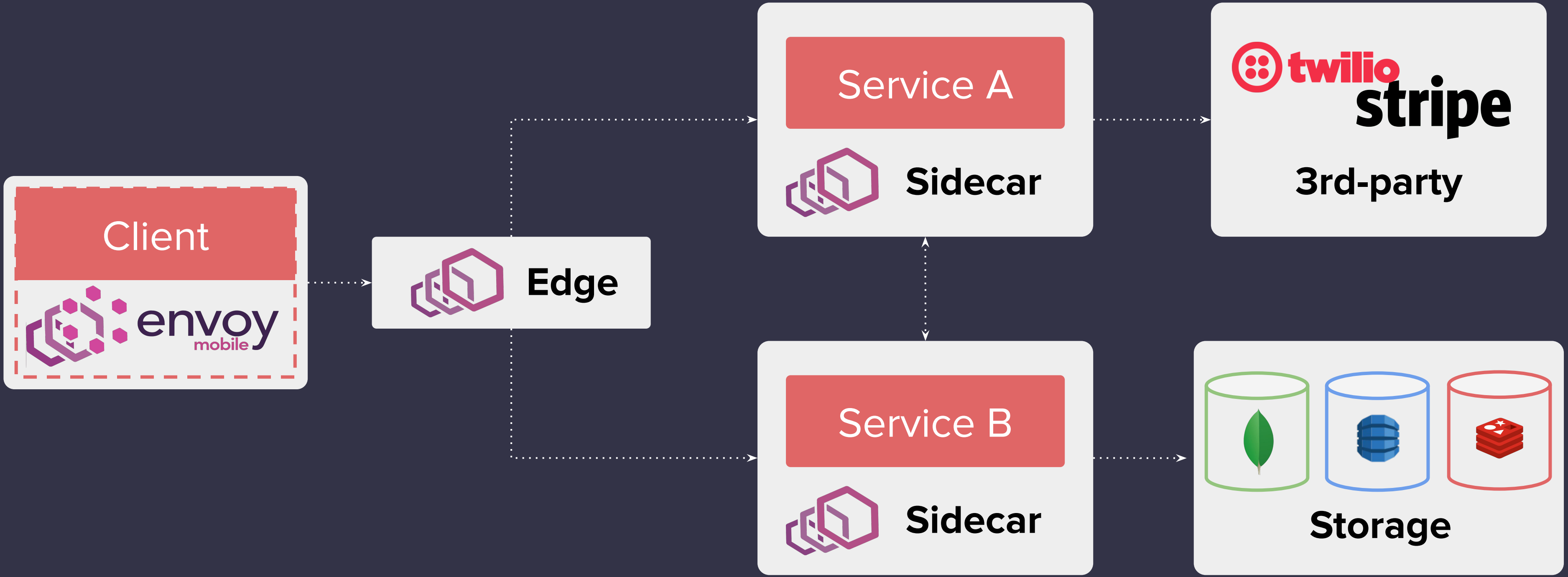
Performance	?	✓
Reliability	?	✓
Extensibility	?	✓
Observability	?	✓
Configuration API	?	✓

Topology 2.0: ~~Universal~~ Network Primitive



@goaway @junr03

Topology 3.0: Universal Network Primitive



@goaway @junr03

Standardizing infrastructure

envoy mobile

envoy

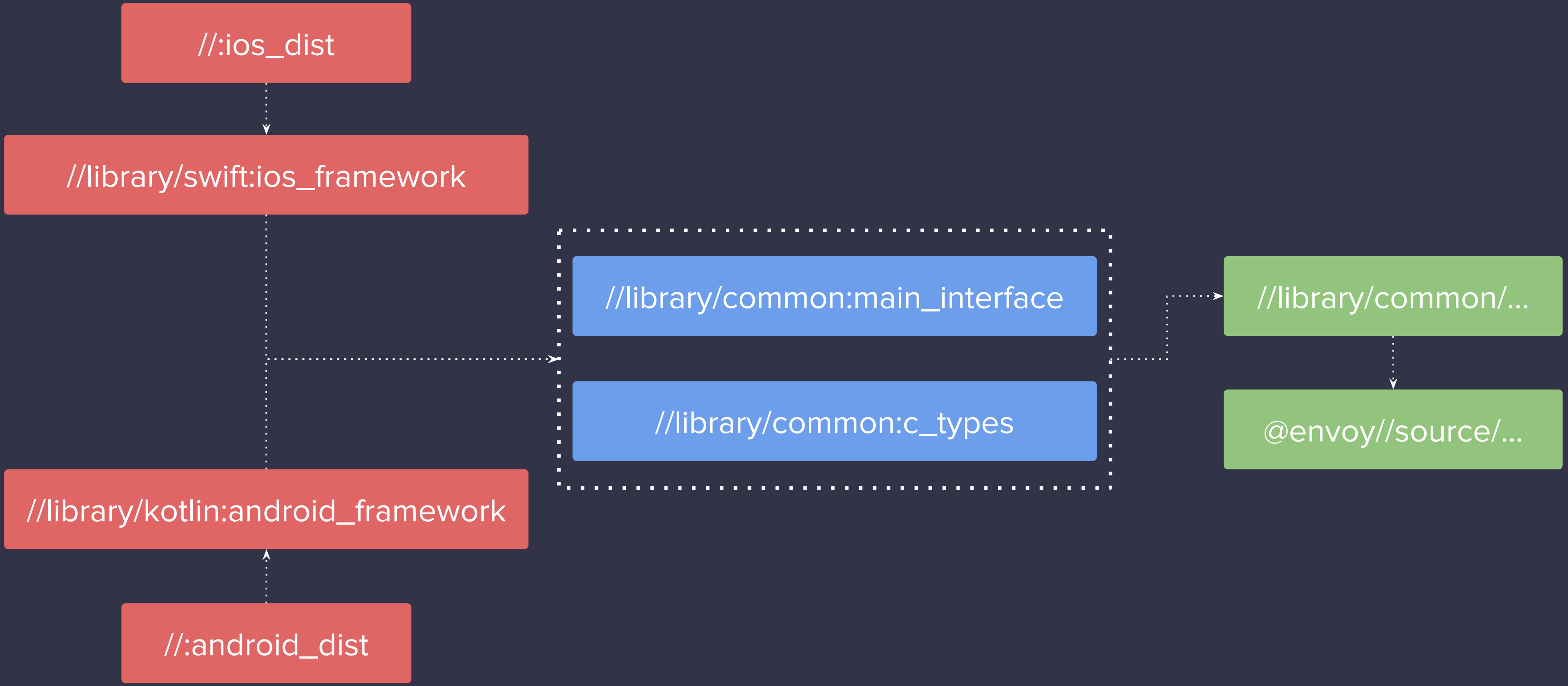
@goaway @junr03

Why is ~~world domination~~ standardization useful?

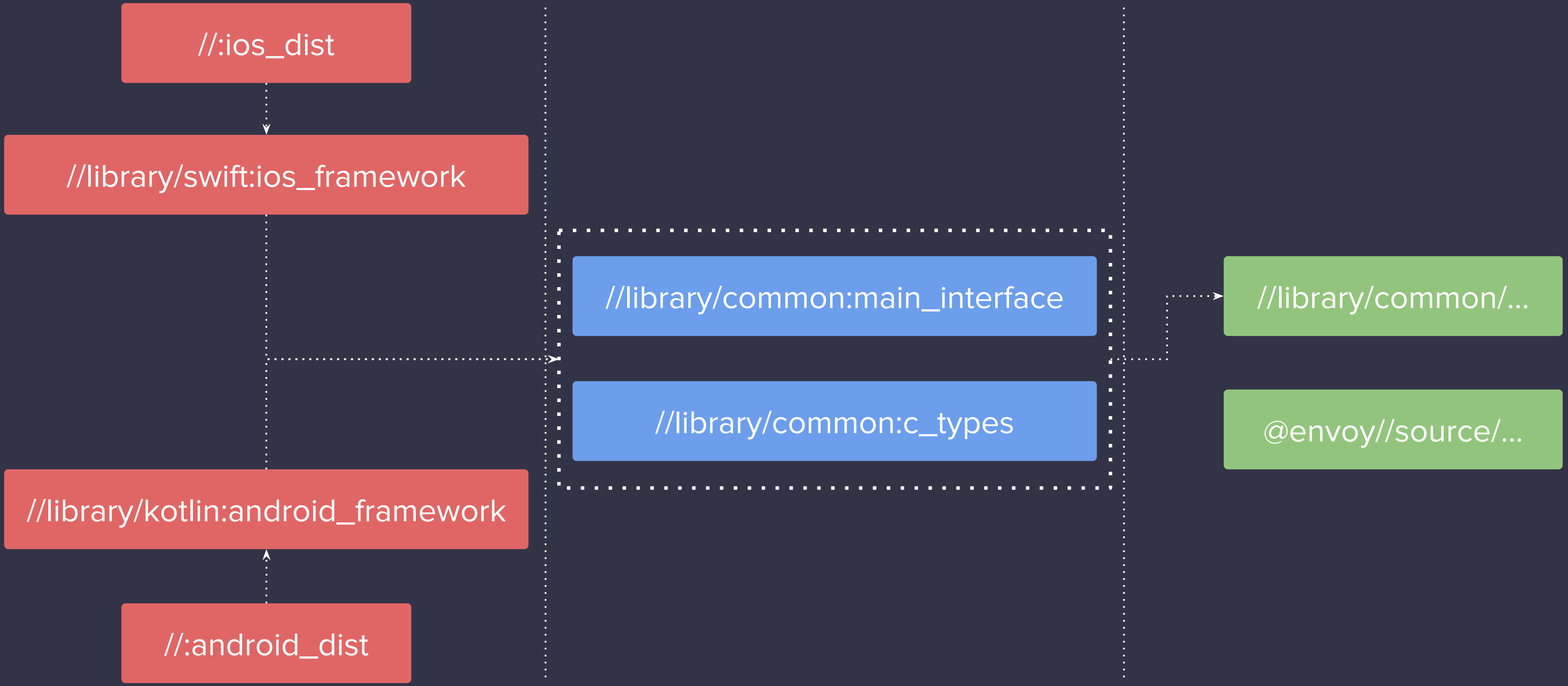
- Write once, deploy everywhere
- Common tooling for common problems
- Reduce cognitive load

Envoy as a Library

Build System


//:ios_dist
//library/swift:ios_framework
//library/common:main_interface
//library/common:c_types
//library/common/...
@envoy//source/...
//library/kotlin:android_framework
//:android_dist

Build System


//:ios_dist
//library/swift:ios_framework
//library/common:main_interface
//library/common:c_types
//library/common/...
@envoy//source/...
//library/kotlin:android_framework
//:android_dist

API - Layered Design

Platform (iOS/Android)	Bridge (C)	Native (C++/Envoy)
Thin platform code	bridging over C bindings	leveraging C++ native code

How to run a process in an app?

picture of an engine (a very fast one)

Threading contexts

Application Threads

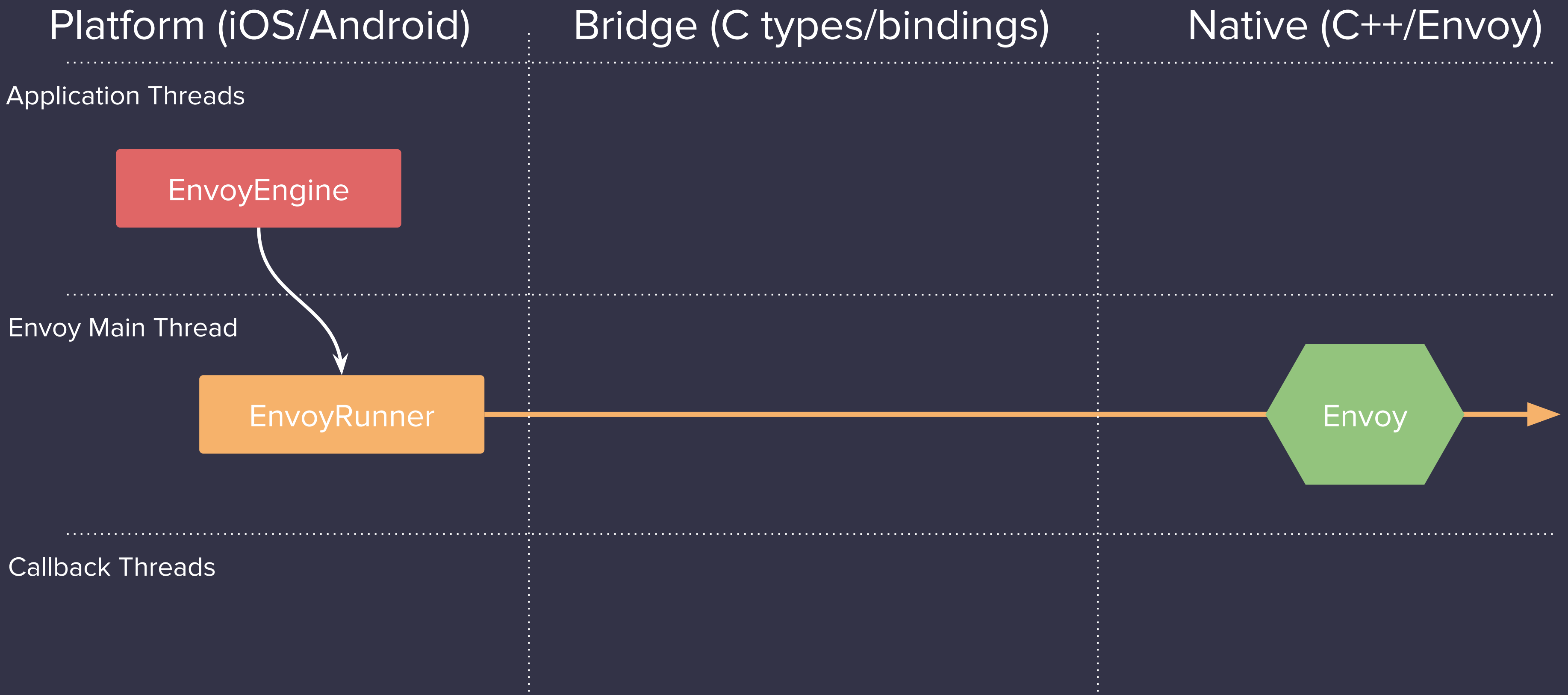
Envoy Main Thread

Callback Threads

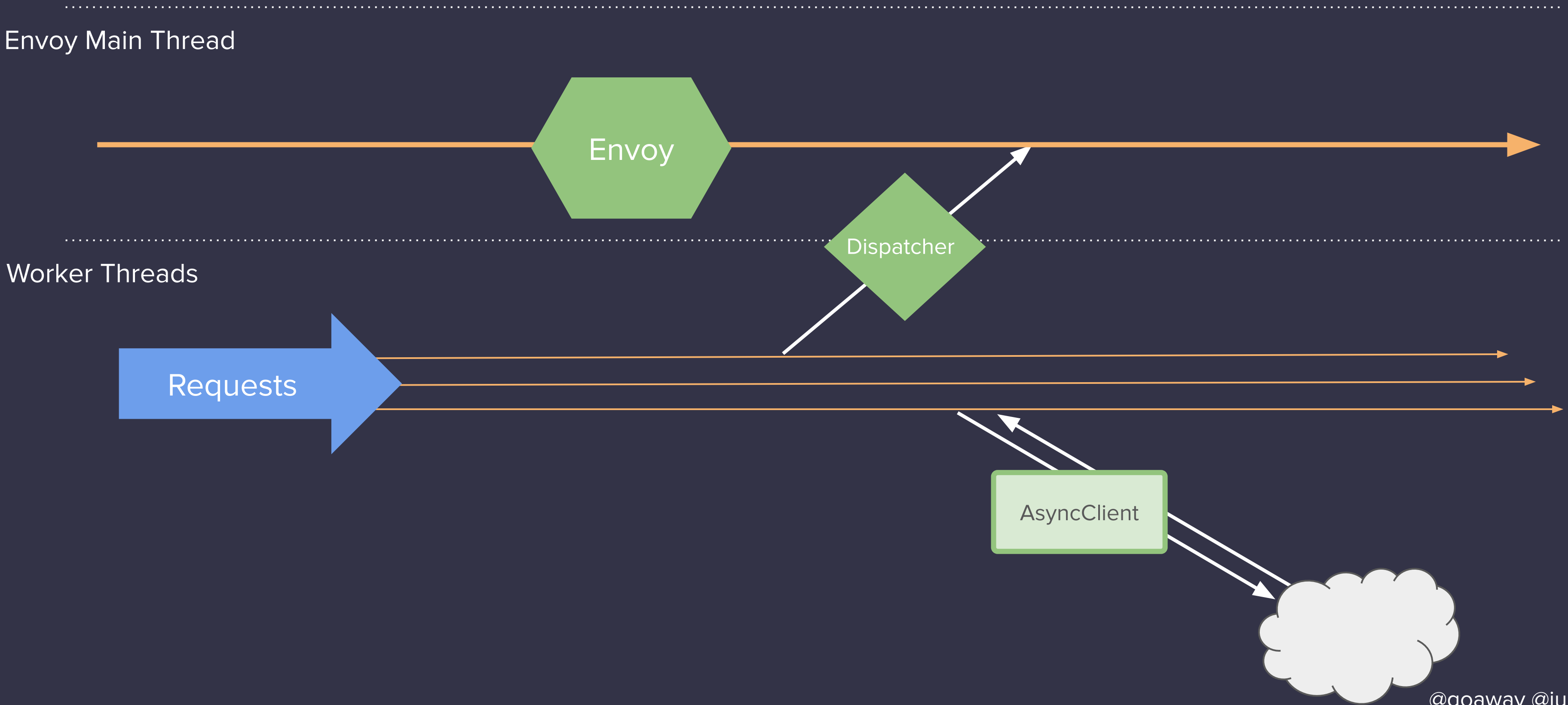
Library Matrix

	Platform (iOS/Android)	Bridge (C types/bindings)	Native (C++/Envoy)
Application Threads			
Envoy Main Thread			
Callback Threads			

Library Lifecycle - Running Envoy

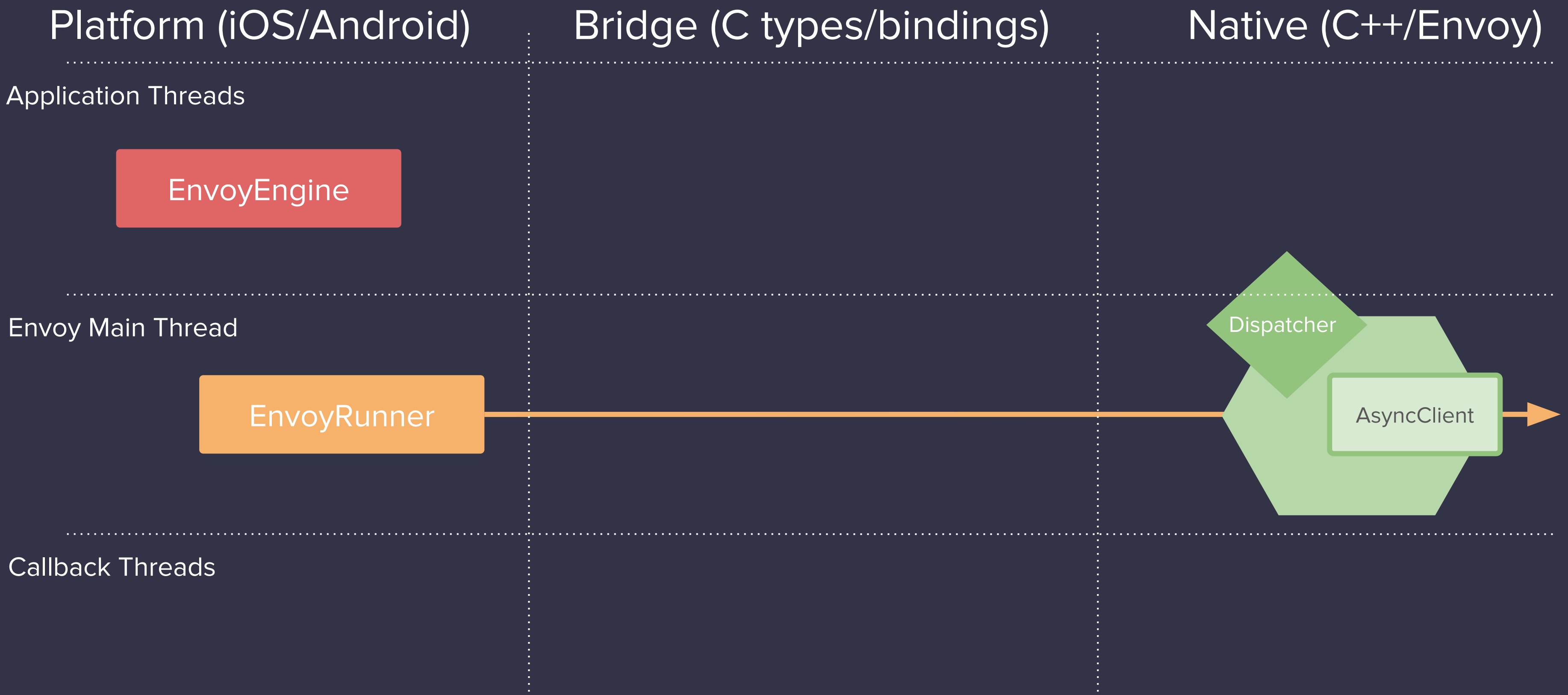


Server Envoy

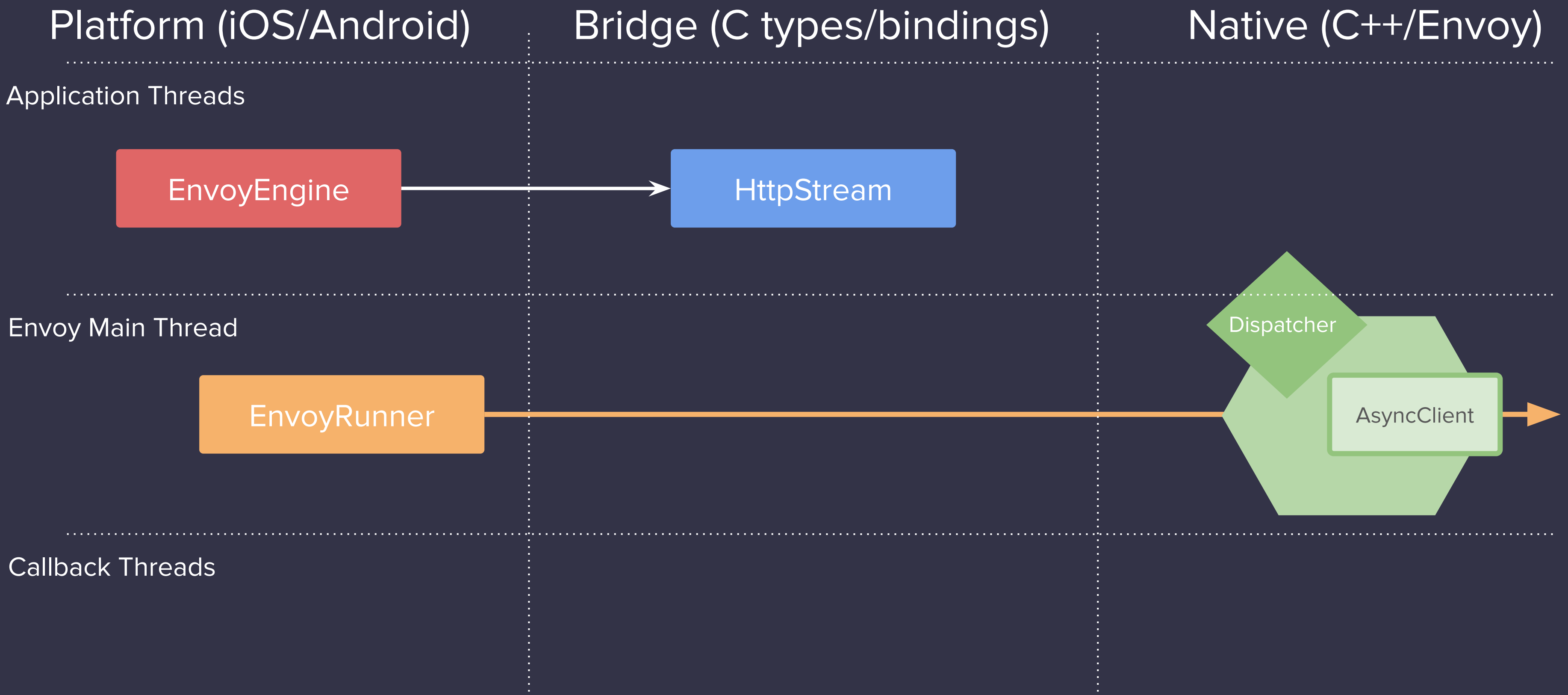


@goaway @junr03

Library Lifecycle - using Envoy Constructs



Library Lifecycle - starting a stream

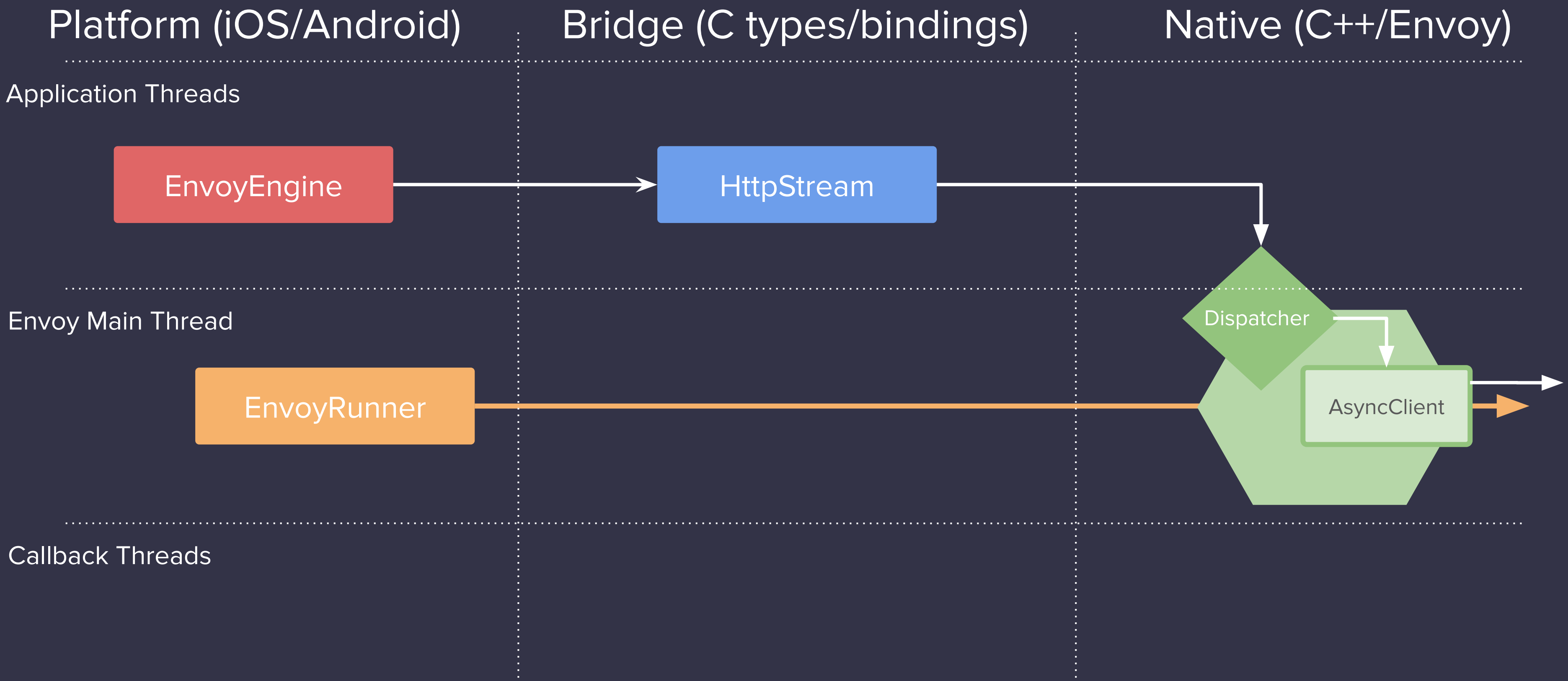


@goaway @junr03

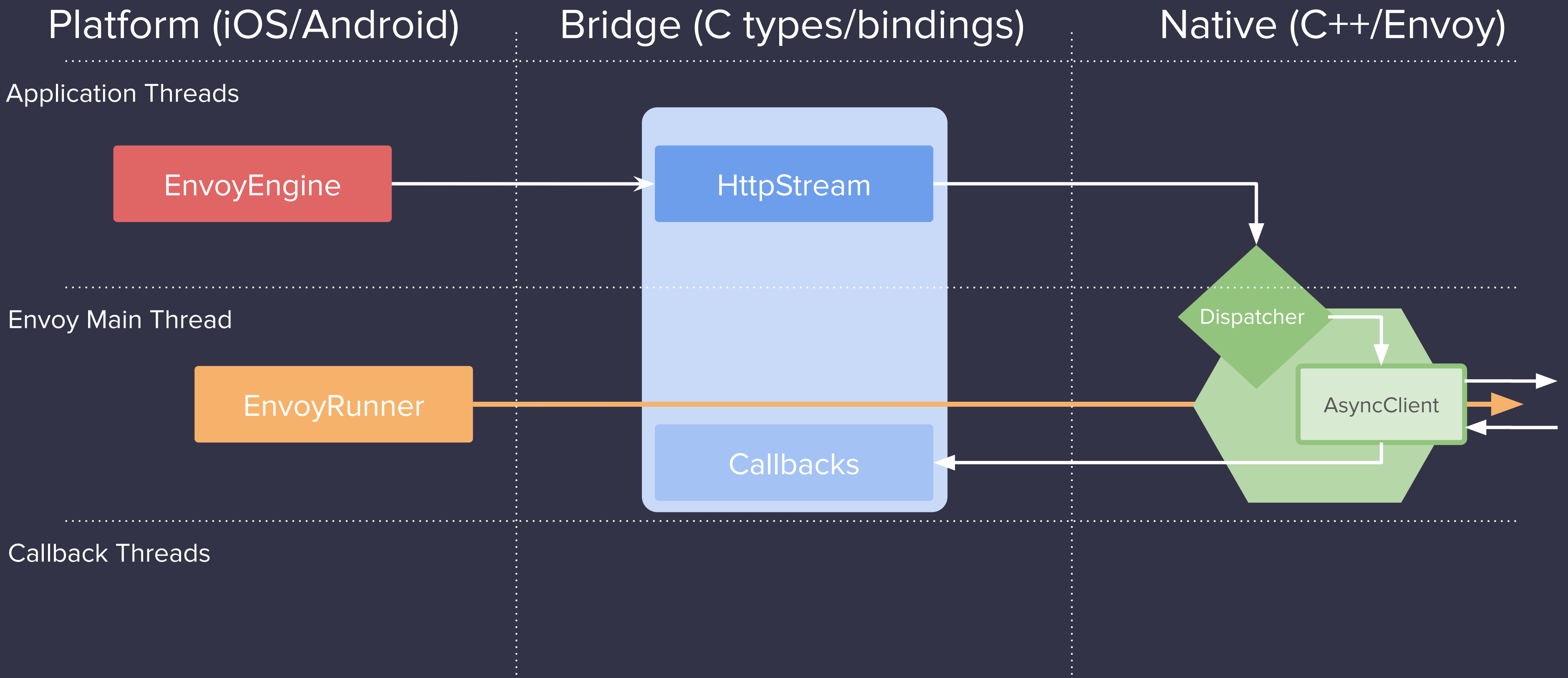
Memory Management

```
/**
 * Holds raw binary data as an array of bytes.
 */
typedef struct {
  size_t length;
  const uint8_t* bytes;
  envoy_release_f release;
  void* context;
} envoy_data;

/**
 * Callback indicating Envoy has drained the associated buffer.
 */
typedef void (*envoy_release_f)(void* context);
```

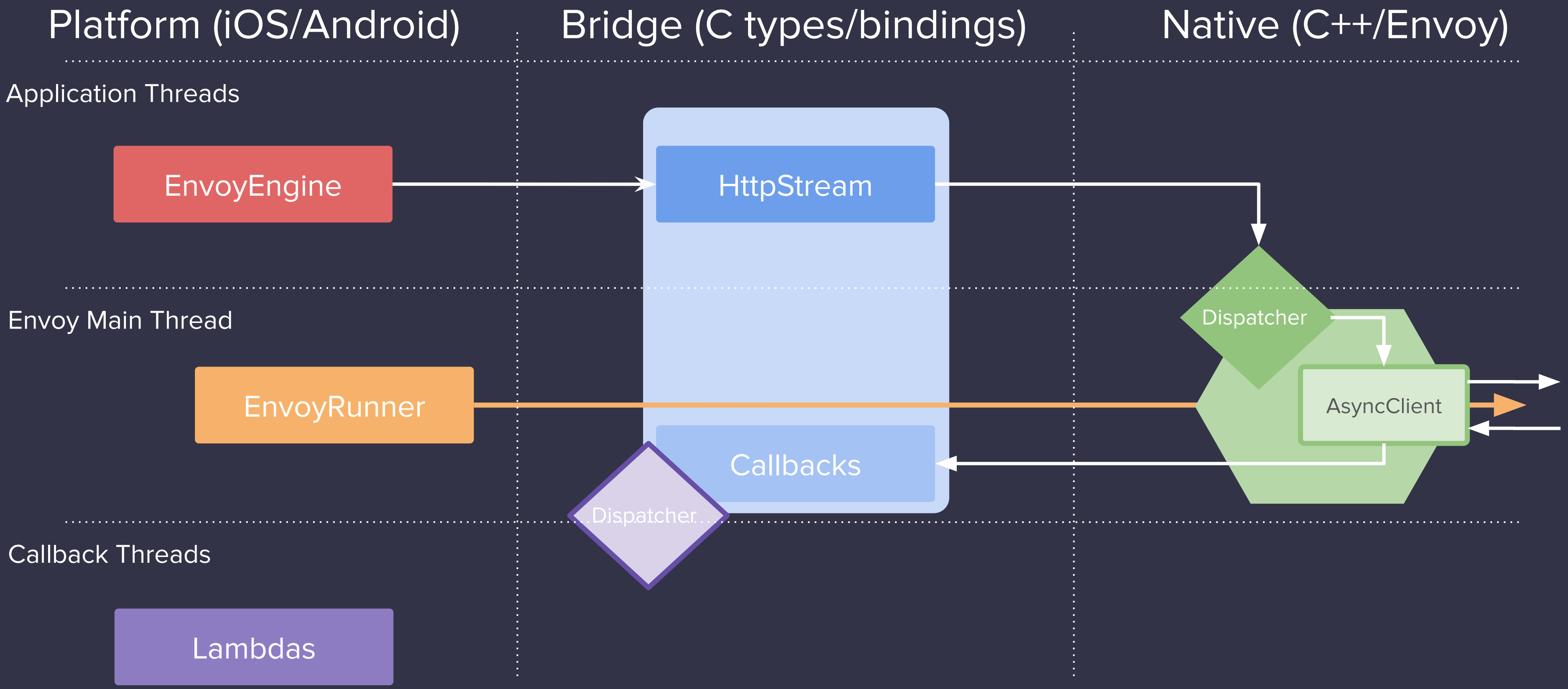
Library Lifecycle - dispatching a stream



Library Lifecycle - callbacks



Library Lifecycle - platform callbacks

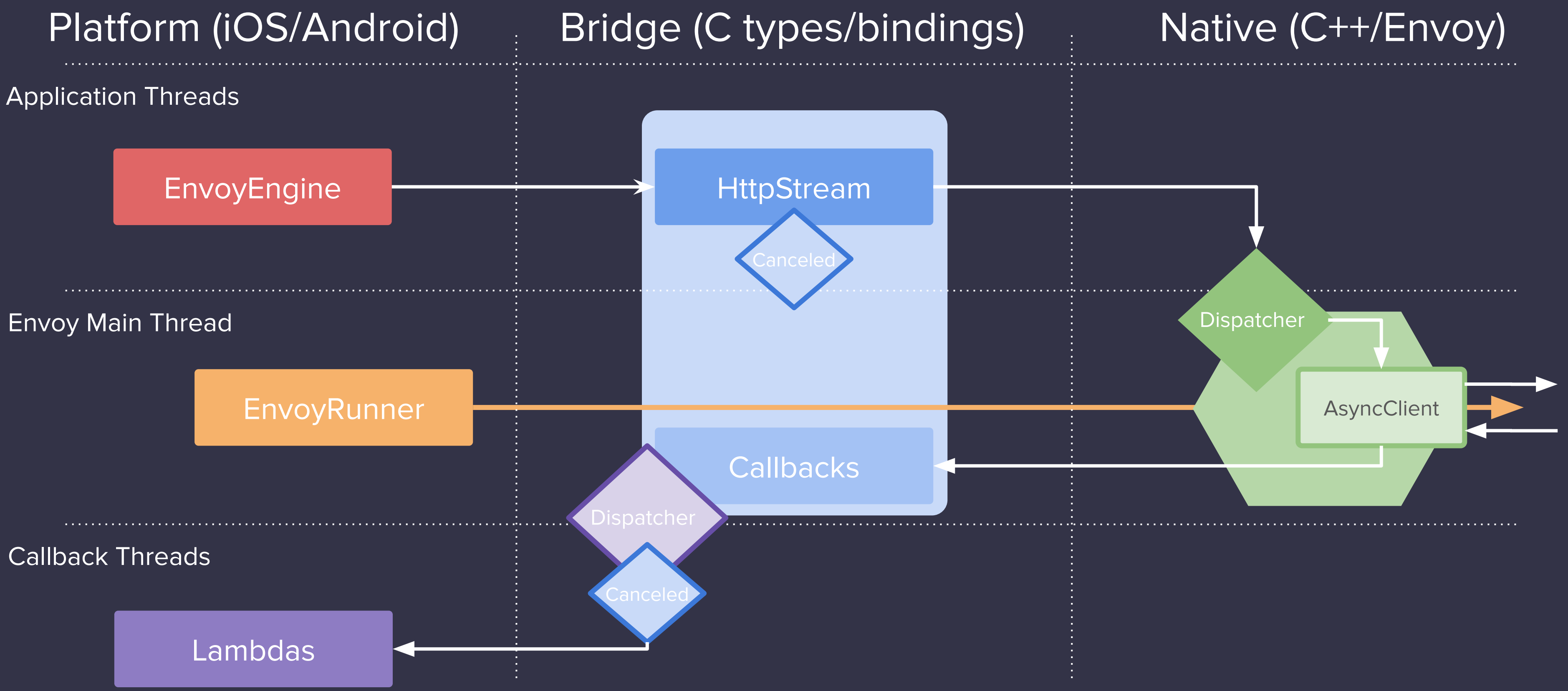


@goaway @junr03

Platform Callbacks

```
typedef struct {
  envoy_on_headers_f on_headers;
  ...
  // Will be passed through to callbacks to provide
  // dispatch and execution state.
  void* context;
} envoy_http_callbacks;

/**
 * Called when all headers get received on the async HTTP stream.
 */
typedef void (*envoy_on_headers_f)(envoy_headers headers, bool
end_stream, void* context);
```

Library Lifecycle - cancellation



@goaway @junr03

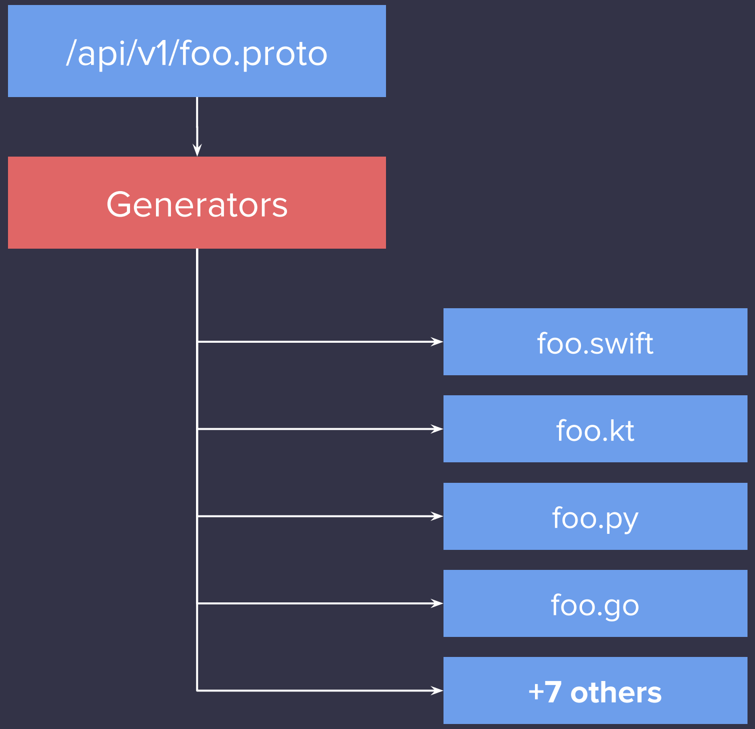
Where are we Now?

Alpha App at Lyft!

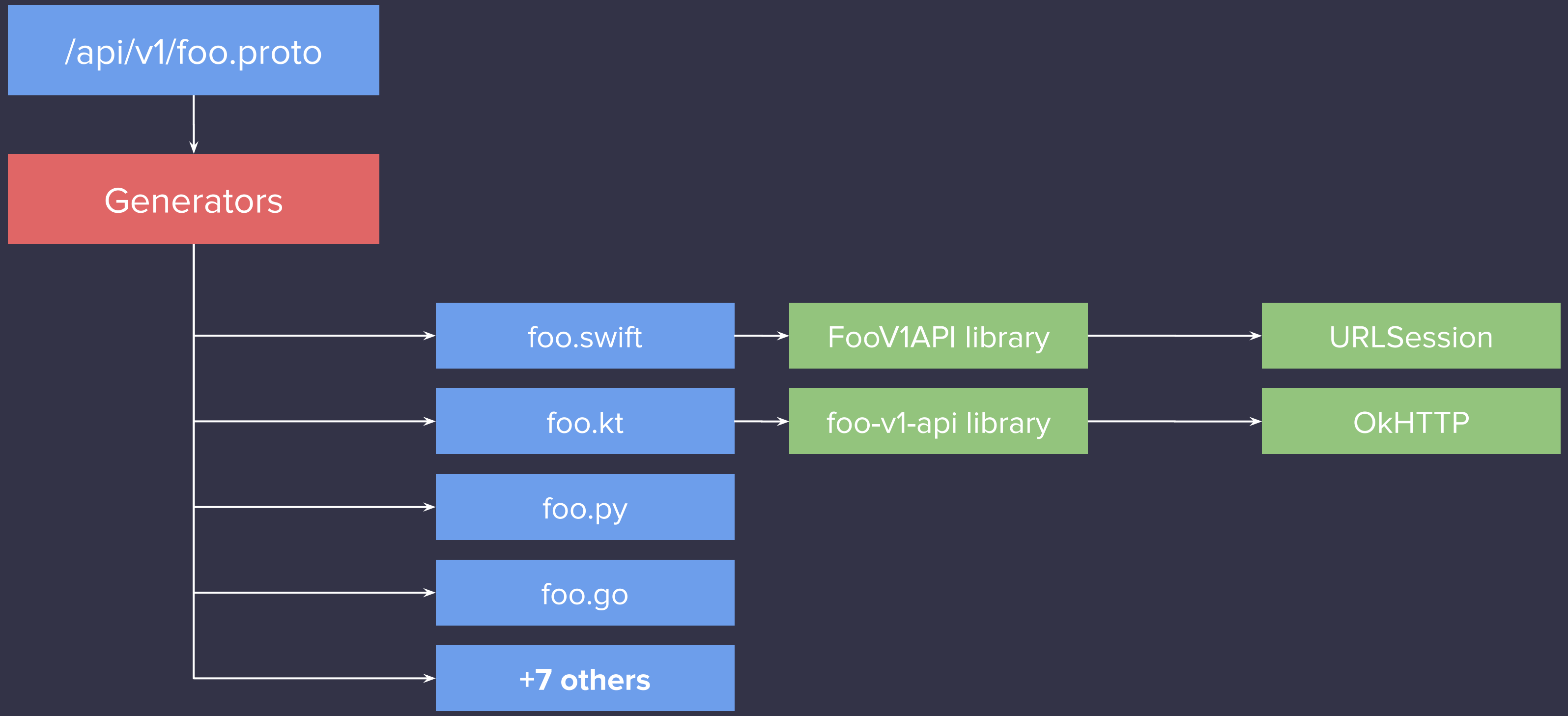
IDL pipeline

/api/v1/foo.proto

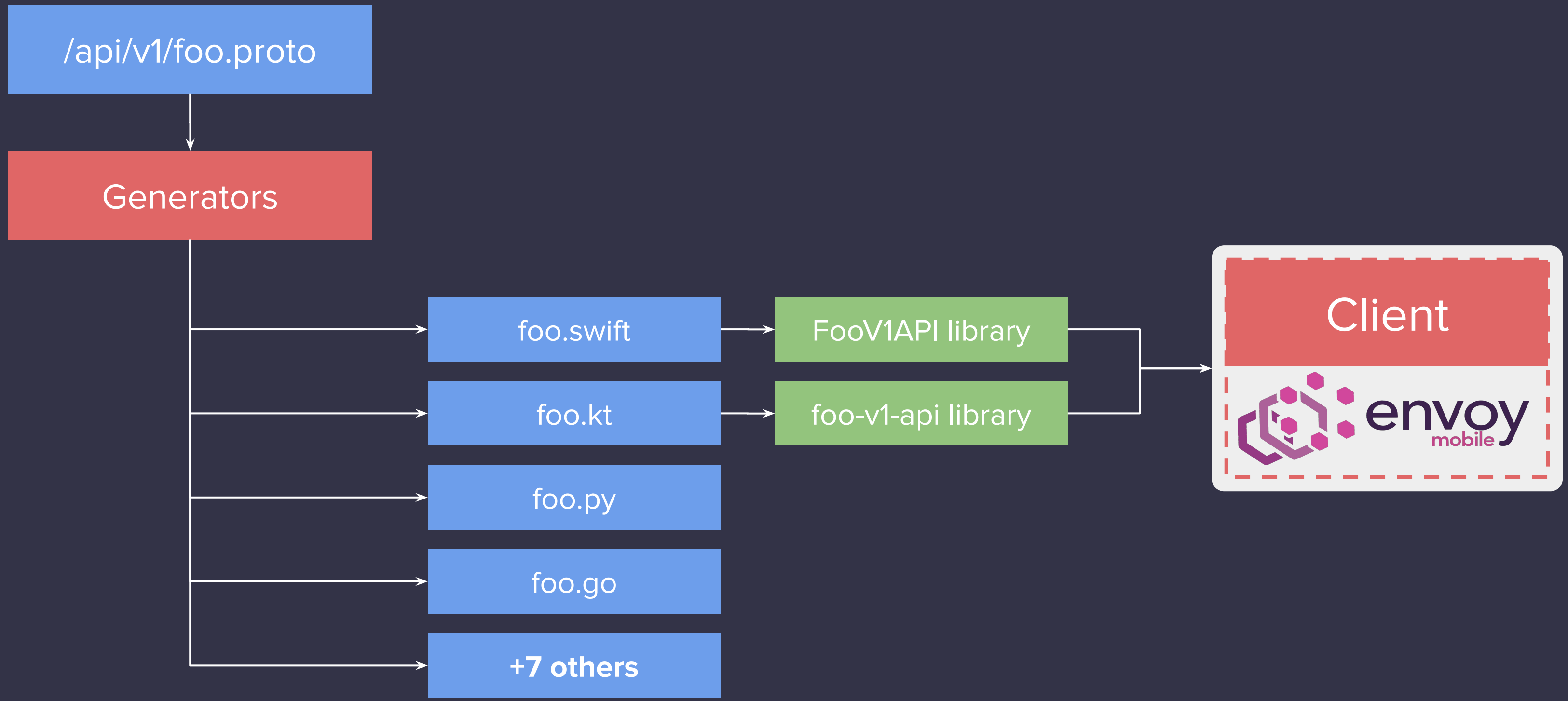
IDL pipeline



IDL pipeline



IDL pipeline



Build an Engine

```
let envoy = try EnvoyClientBuilder(domain:
"api.envoyproxy.io")
   .addLogLevel(.warn)
   .addStatsFlushSeconds(60)
   .build()

val envoy = EnvoyClientBuilder(
Domain("api.envoyproxy.io"))
   .addLogLevel(LogLevel.WARN)
   .addStatsFlushSeconds(60)
   ...
   .build()
```

Build an Engine

```
let envoy = try EnvoyClientBuilder(domain:
"api.envoyproxy.io")
  .addLogLevel(.warn)
  .addStatsFlushSeconds(60)
  .build()
```

Build an Engine

```
let envoy = try EnvoyClientBuilder(domain:
"api.envoyproxy.io")
   .addLogLevel(.warn)
   .addStatsFlushSeconds(60)
   .build()
```

Build a Request

```
let request = RequestBuilder(path:
"/pb.api.v1.Foo/GetBar")
   .addHeader(name: "x-custom-header", value: "foobar")
   .addRetryPolicy(RetryPolicy(...))
   .build()
```

Build a Request

```
let request = RequestBuilder(path:
"/pb.api.v1.Foo/GetBar")
   .addHeader(name: "x-custom-header", value: "foobar")
   .addRetryPolicy(RetryPolicy(...))
   .build()
```

Build a Response Handler

```
let handler = ResponseHandler()
  .onHeaders { headers, status, _ ->
    ...
  }
  .onData { data ->
    // Deserialize message data here
  }
  ...
```

Build a Response Handler

```
let handler = ResponseHandler()
  .onHeaders { headers, status, _ ->
    ...
  }
  .onData { data ->
    // Deserialize message data here
  }
  ...
```

Make a request

```
envoy.send(request, responseHandler)
   .sendData(message)
   .sendData(message)
   .close()
```

Make a request

```
envoy.send(request, responseHandler)
   .sendData(message)
   .sendData(message)
   .close()
```

Drop in Replacement

- Expose compatible bindings to classic network libraries: NSURL, OkHTTP

What are we solving for?

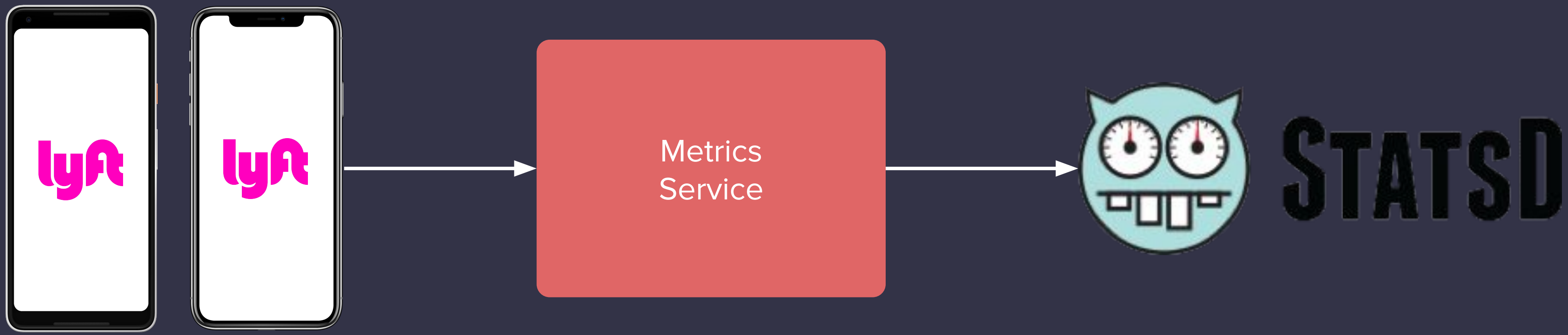
Three 9s at the server-side edge is meaningless if the user of a mobile application is only able to complete the desired product flows a fraction of the time.

Performance	✓	✓
Reliability	✓	✓
Extensibility	✓	✓
Observability	✓	✓
Configuration API	✓	✓

Observability

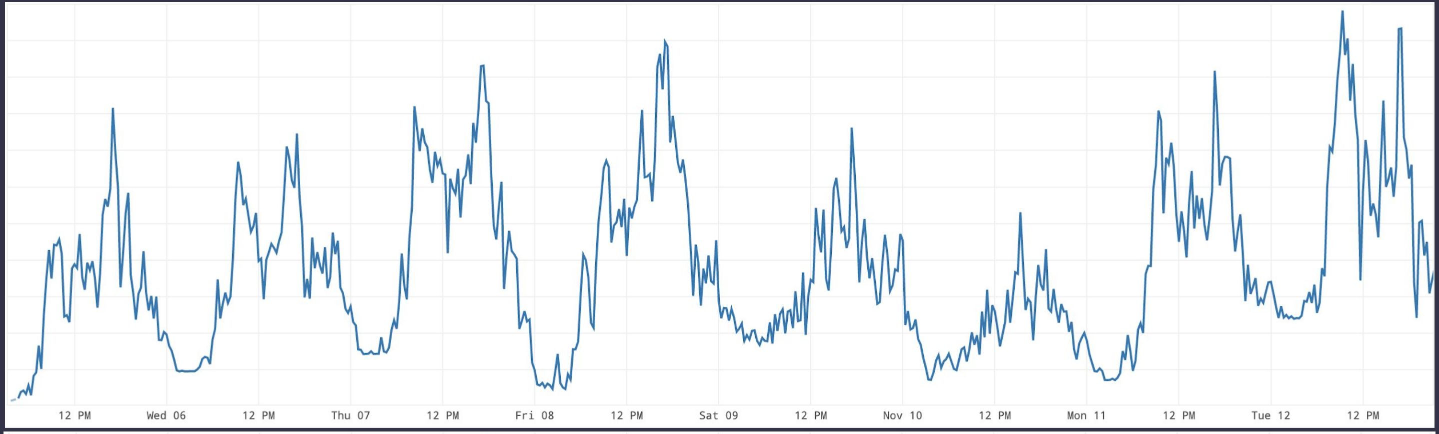
```
ts(envoy_mobile.cluster.api.upstream_rq.count)


ts(envoy_edge.cluster.*.upstream_rq.count)
```

Time-series Metrics

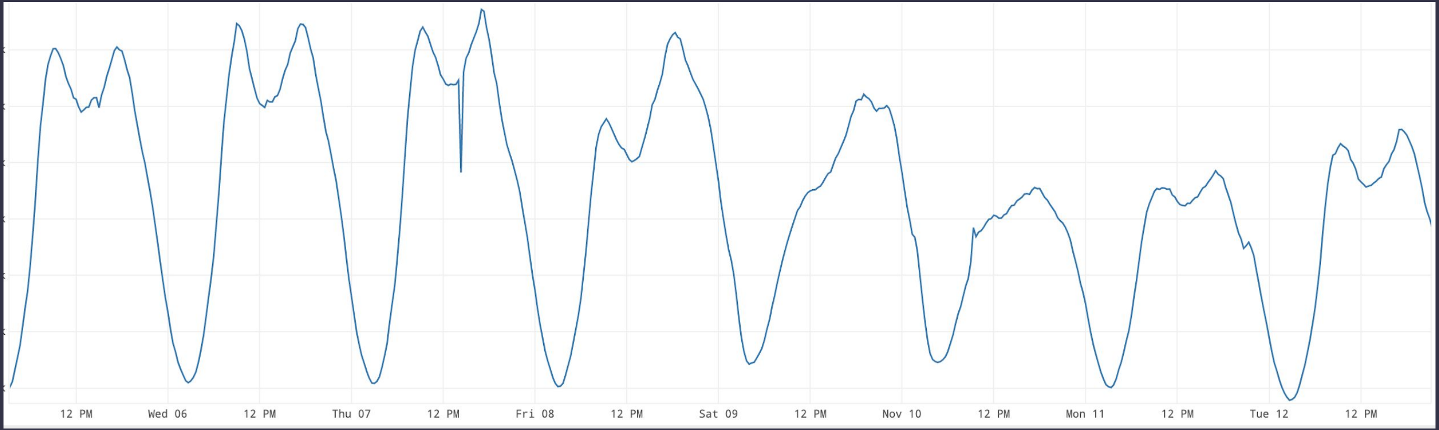


Dashboards!

`ts(envoy_mobile...)`



`ts(envoy_edge...)`



Onwards!

Onwards!

- Protocol Experimentation
- API Listener - Filter stack
- Intelligent network behavior
- Annotated APIs
- Dynamic configuration
- Beyond mobile phones!

Community

This is the beginning, join us!

Michael Schore

@goaway

Jose Nino

@junr03

envoy mobile

envoy-mobile.github.io