GOSIM®
GLOBAL OPEN-SOURCE INNOVATION MEETUP

GOSIM CHINA 2024

From Service-*Mess* to Service-*Mesh* with

uProtocol
CONNECTING AUTOMOTIVE APPS AND SERVICES, EVERYWHERE

Pete LeVasseur
Oxidation Partners

GOSIM®
GLOBAL OPEN-SOURCE INNOVATION MEETUP

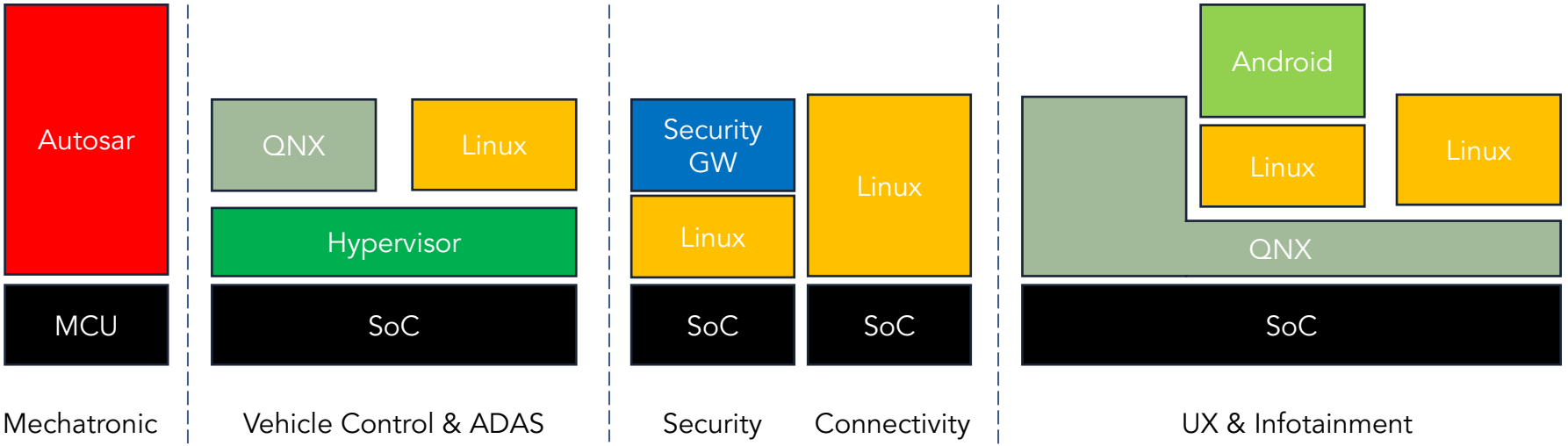
GOSIM CHINA 2024

SDV™
Eclipse Software Defined Vehicle

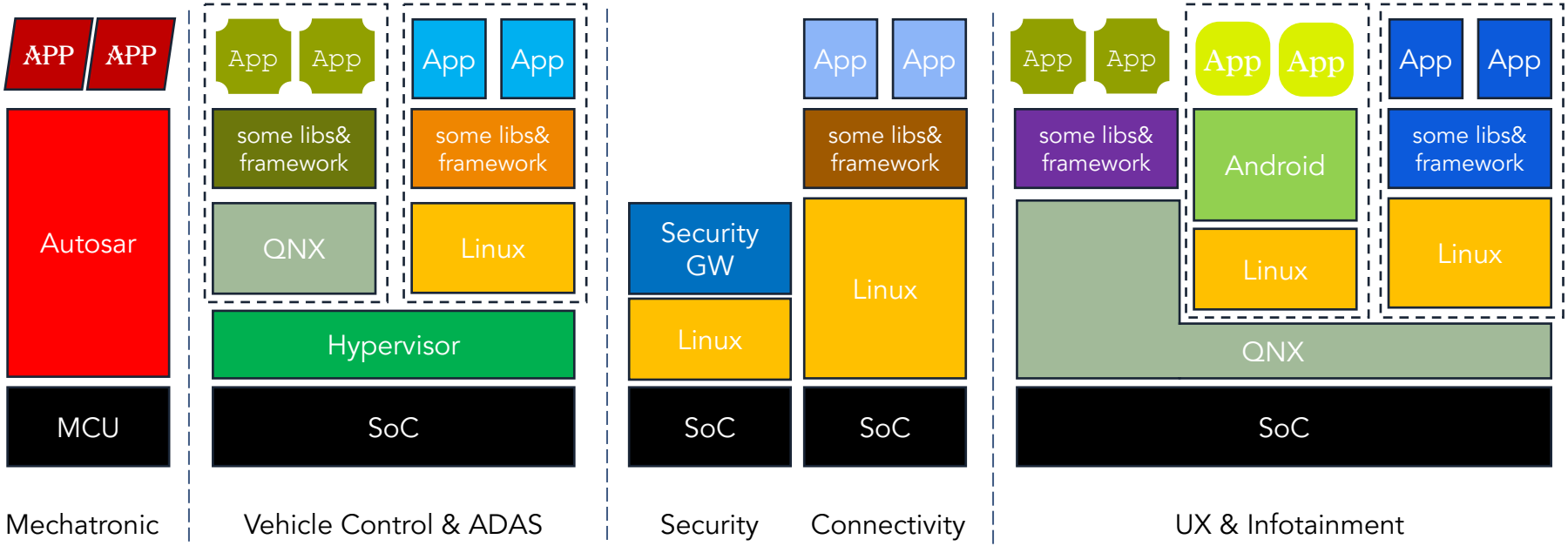
uProto-why?

Middleware, why do we care?

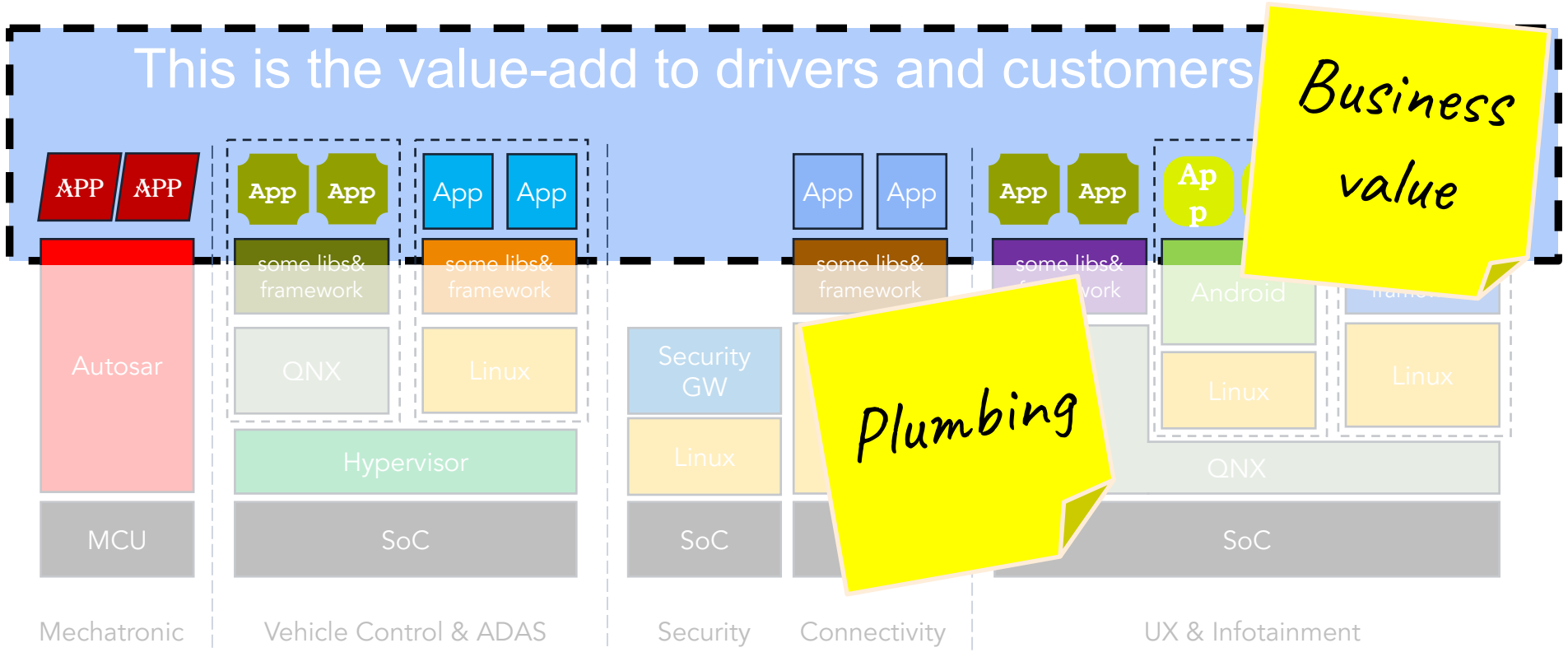
We have computers…



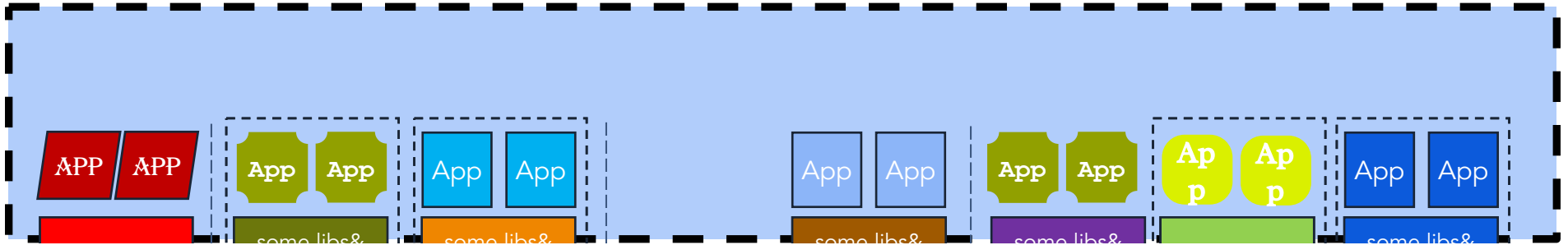
… now we want to run things!



Where is the customer value, what is just plumbing?



The deal with Apps & Services



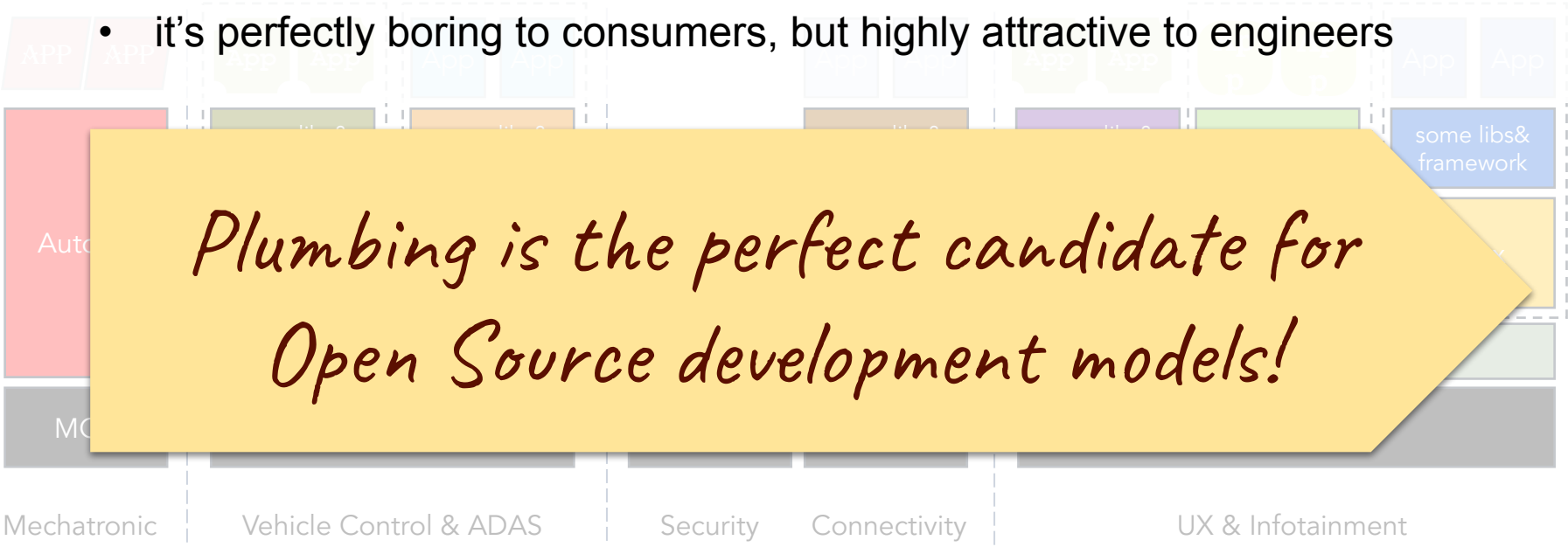
- Apps/Services represent the “vehicle surface” that users experience,
- they are built by very different kinds of organization (departments etc),
- who need to align around a common view of the vehicle,
- and provide long-term continuity of the business value they embody.

BUT: every tech stack brings its own plumbing

(programming model, vehicle view, communication paradigm, etc)

The deal with Plumbing (aka Software Infrastructure)

- Plumbing is mission critical, but not really differentiating
- it's ubiquitous, but invisible
- it's perfectly boring to consumers, but highly attractive to engineers



Plumbing is the perfect candidate for Open Source development models!

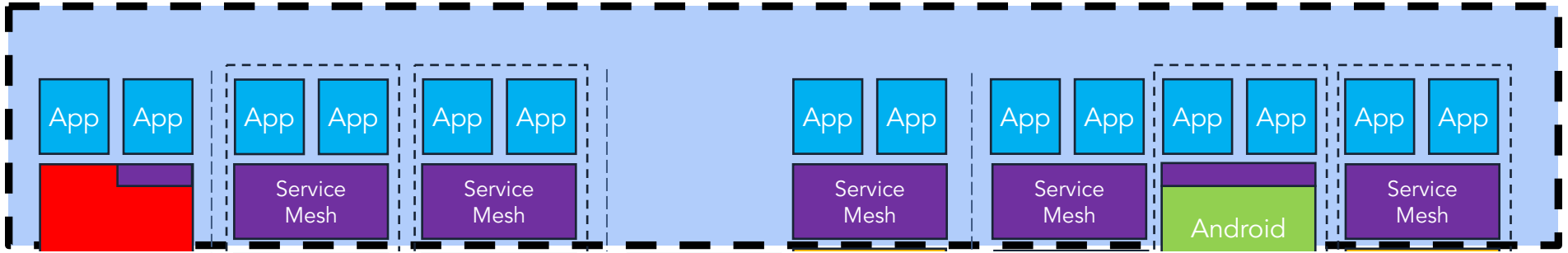
GOSIM®
GLOBAL OPEN-SOURCE INNOVATION MEETUP

GOSIM CHINA 2024

SDV™
Eclipse Software Defined Vehicle

uProto-what?

How do we want to do ”Apps and Services”?



Plumbing needs to:

make communication simple and ubiquitous

→ Unified addressing, inter-domain pub/sub, transparent message routing

foster efficient long-term maintenance & evolution

→ Decouple programming model from proprietary, platform-specific libs&protocols

make App & Service development simple

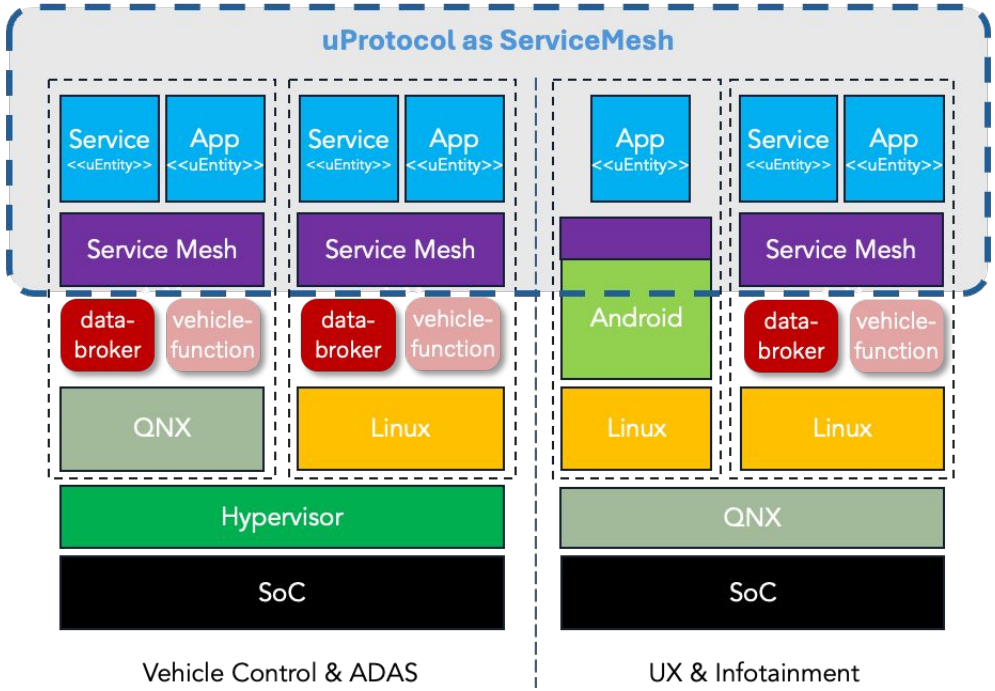
→ Need ecosystem that caters to all kinds of programming languages, protocols, etc

There exist scores of OSes, communication stacks, middlewares, pubsub implementations, transport protocols, etc

We need something on top of all of that, providing us with:

- Unified Addressing
- Inter-domain message forwarding
- X-domain subscription tracking
- Cross-domain discovery
- Communication abstraction and programming model

→ We need an ***Open Source automotive Service Mesh***



GOSIM®
GLOBAL OPEN-SOURCE INNOVATION MEETUP

GOSIM CHINA 2024

SDV™
Eclipse Software Defined Vehicle

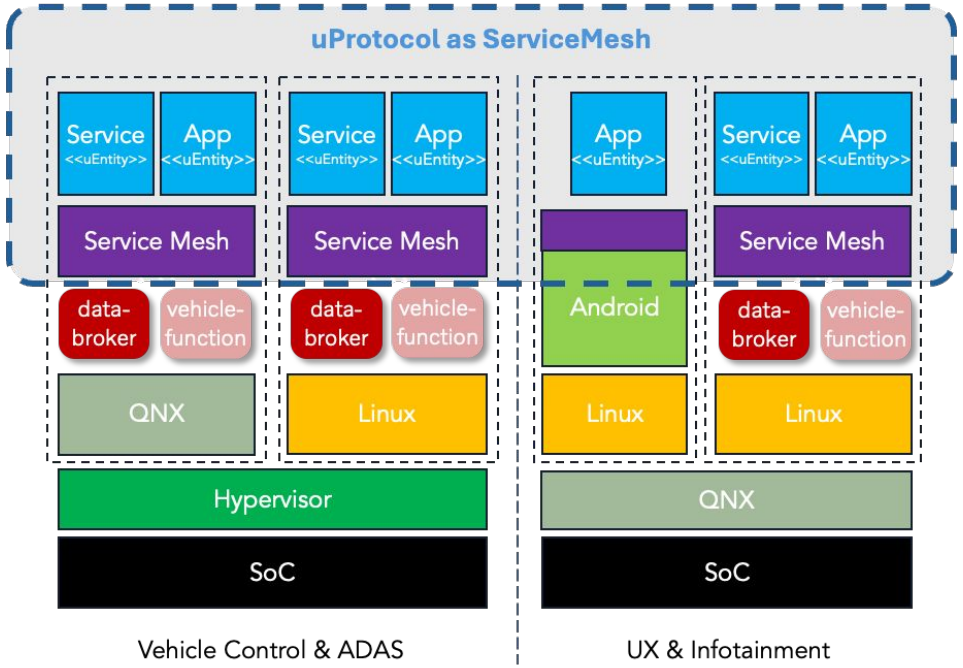
uProto-how?

uProtocol

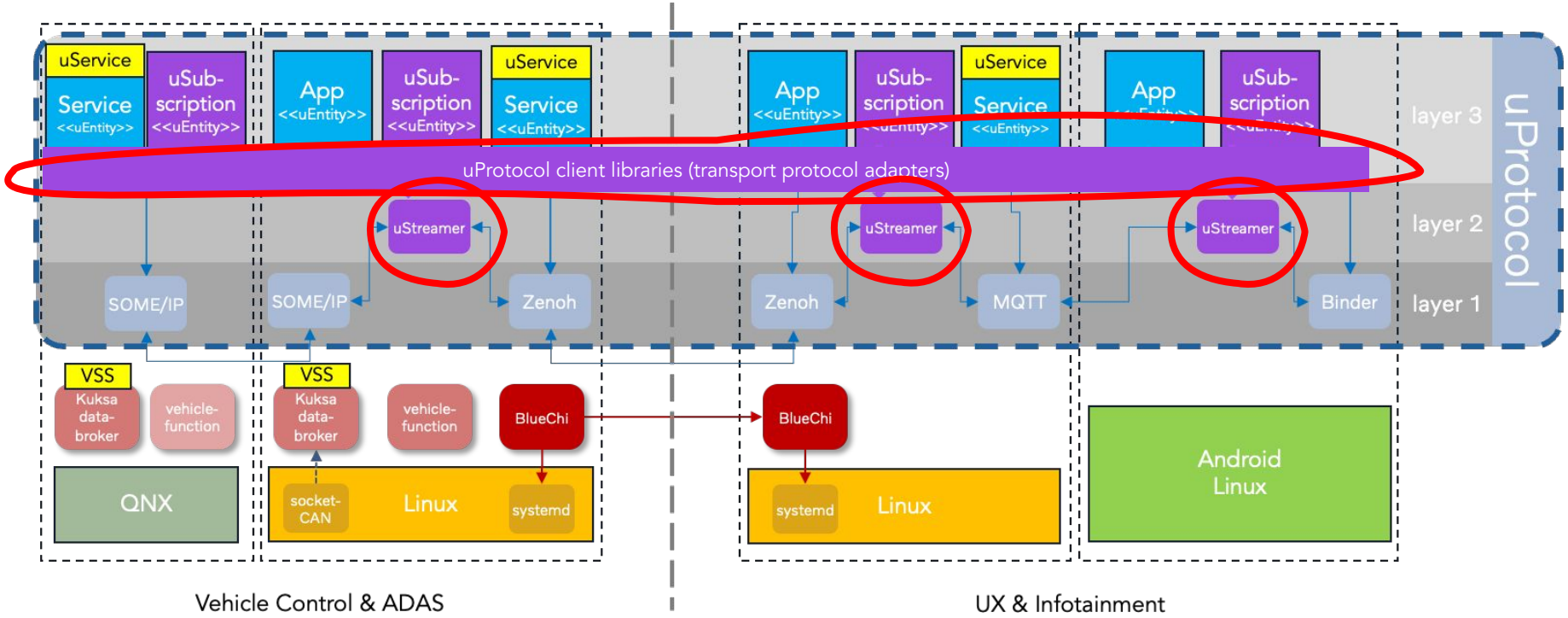
CONNECTING AUTOMOTIVE APPS AND SERVICES, EVERYWHERE

Feature spotlights

- Unified Addressing
- Inter-domain message forwarding
- X-domain subscription tracking
- Support for multiple programming languages and transport protocols
- Early adopter of Eclipse quality process initiative

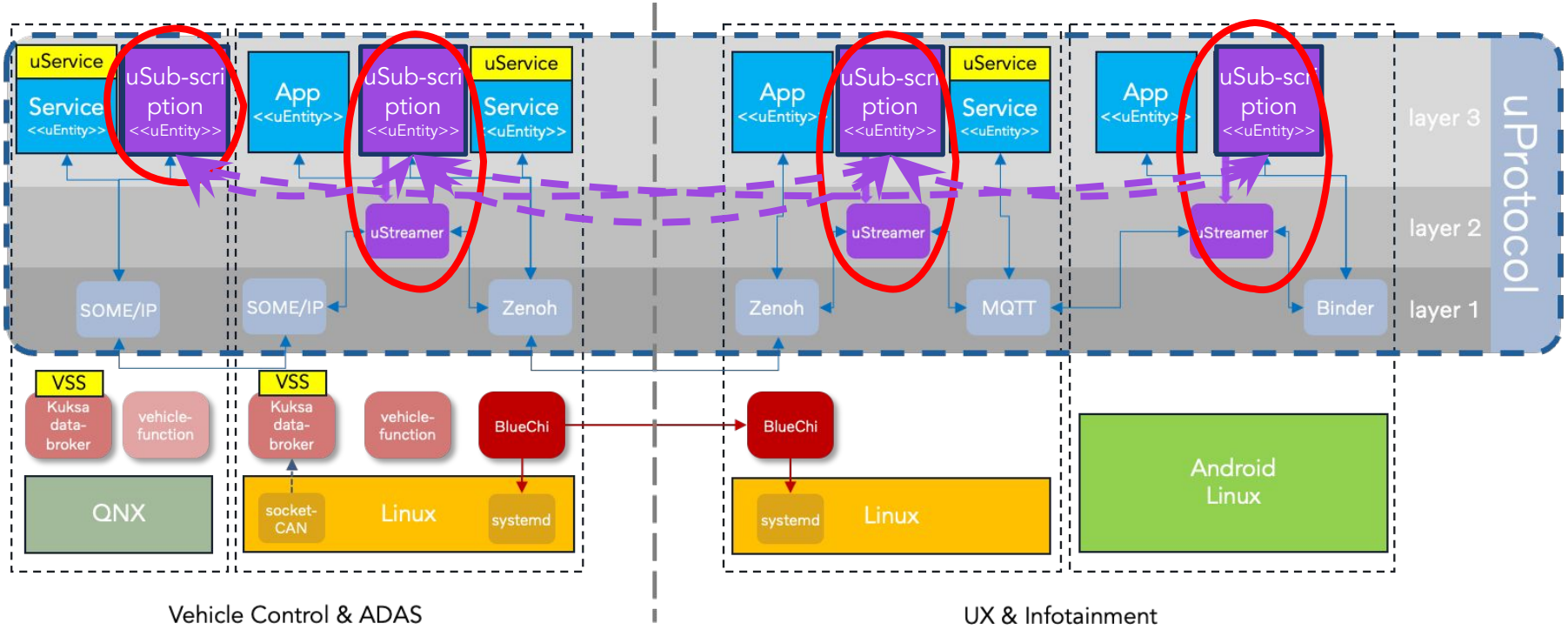


Unified addressing & Inter-domain message forwarding



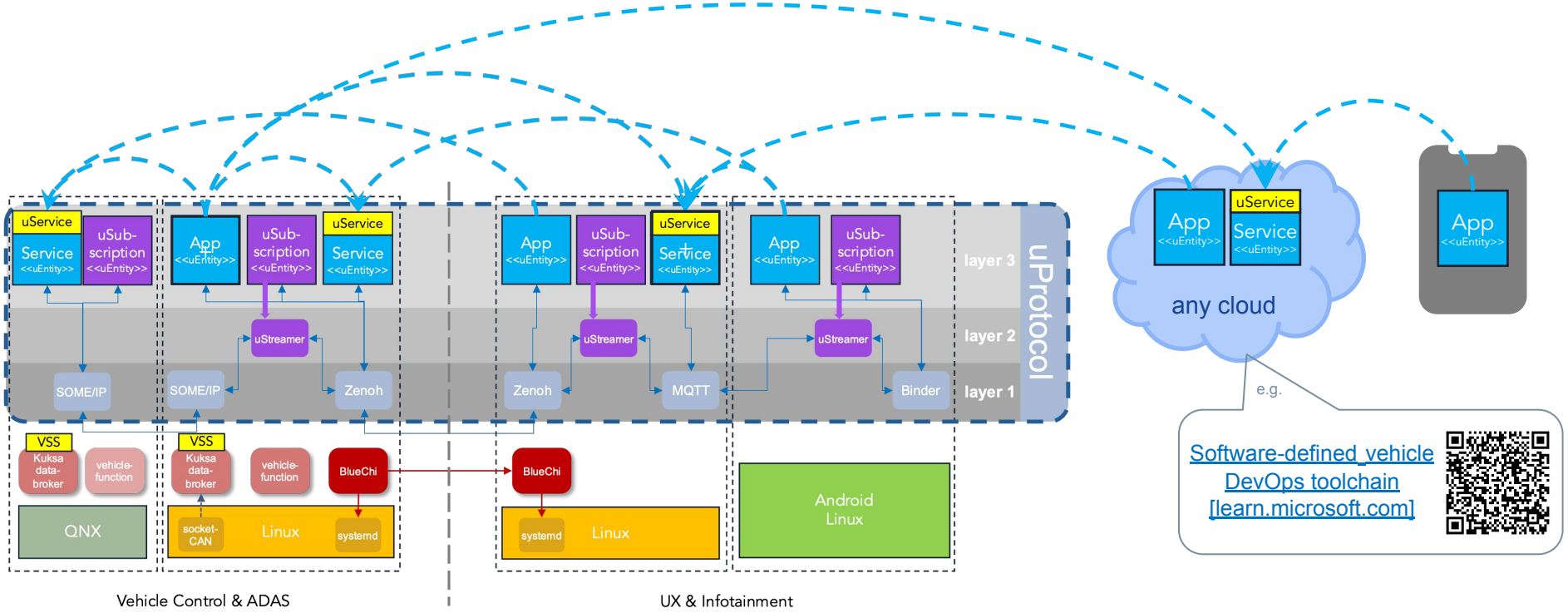
https://github.com/eclipse-uprotocol

X-domain subscription tracking



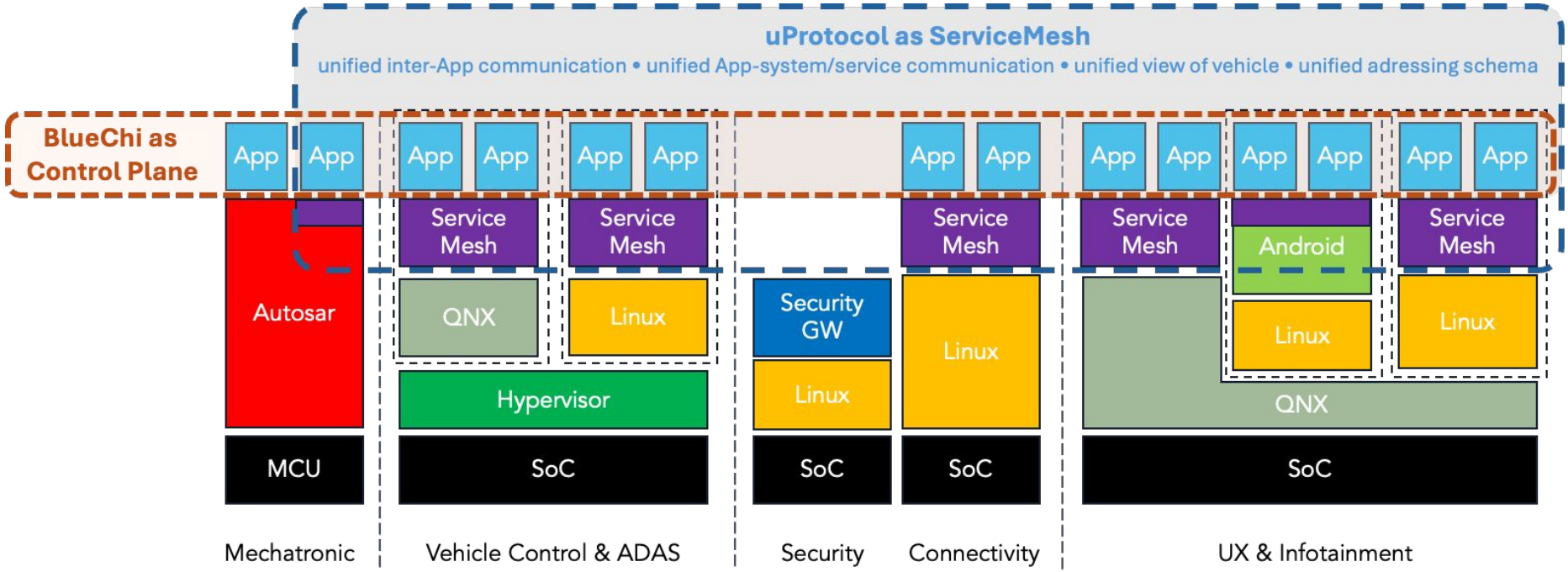
https://github.com/eclipse-uprotocol

Also: extends to cloud/backend, mobile, etc



https://github.com/eclipse-uprotocol

So: launching an SDV lighthouse blueprint!



https://github.com/eclipse-uprotocol

GOSIM®
GLOBAL OPEN-SOURCE INNOVATION MEETUP

GOSIM CHINA 2024

SDV™
Eclipse Software Defined Vehicle

uProto-whoa!

Rust usage for Infrastructural Services

- Rust benefits: memory safety, crates ecosystem, cargo build system
- Ready for Automotive? Yes! Ferrocene
- Flipping entire departments overnight to Rust? No…
- Pareto principle: Put Rust into the 20% or so of critical infra to reap 80% of the benefits for Eclipse uProtocol robustness

Rust uStreamer Route Forwarding API

```rust
#[derive(Clone)]
pub struct Endpoint {
    pub(crate) name: String,
    pub(crate) authority: String,
    pub(crate) transport: Arc<dyn UTransport>,
}
```

```rust
pub struct UStreamer { /* ... */ }

impl UStreamer {
    pub async fn add_forwarding_rule(
        &mut self,
        r#in: Endpoint,
        out: Endpoint,
    ) -> Result<(), UStatus> { /* ... */ }
}
```

Rust uStreamer Endpoint Configuration

```rust
let host_transport: Arc<dyn UTransport> = Arc::new(
            UPTransportZenoh::new(/* ... */)
    );

let host_endpoint = Endpoint::new(
        "host_endpoint",
        &config.streamer_uuri.authority,
        host_transport.clone(),
    );
```

```rust
let someip_transport: Arc<dyn UTransport> = Arc::new(
        UPTransportVsomeip::new_with_config(/* ... */)
    );

let mechatronics_endpoint = Endpoint::new(
        "mechatronics_endpoint",
        &config.someip_config.authority,
        someip_transport.clone(),
    );
```

Rust uStreamer Configuration

```rust
streamer
    .add_forwarding_rule(mechatronics_endpoint.clone(), host_endpoint.clone())
    .await
    .expect("Unable to add mechatronics -> host forwarding rule");

streamer
    .add_forwarding_rule(host_endpoint.clone(), mechatronics_endpoint.clone())
    .await
    .expect("Unable to add host -> mechatronics forwarding rule");

```

GOSIM®
GLOBAL OPEN-SOURCE INNOVATION MEETUP

GOSIM CHINA 2024

SDV™
Eclipse Software Defined Vehicle

uProto-when?

Facts & Status

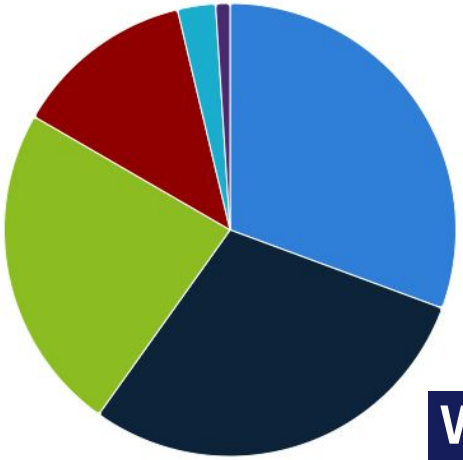
Languages

MQTT SOME/IP

Transports

Organization Contribution Activity

Commits on this project by supporting organization over the last three months.



- Contributor
- Robert Bosch GmbH
- ETAS GmbH
- General Motors GTO LLC
- ZettaScale Technology SARL
- Other Committer

November 1.6.0-alpha Release:

MQTT

uProtocol project status and tech coverage

	Java	C++	Rust	Python
client library	✔	✔	✔	✔
uStreamer			✔	
uSubscription	✔		✔	
uDiscovery	✔			
Zenoh adapter		✔	✔	✔
MQTT adapter	✔		✔	✔
Android Binder adapter	✔			
SOME/IP adapter		✔	✔	

- uProtocol specification and API definitions available in dedicated repo
- Varying maturity: Java and Rust likely ahead
- Additional tooling available: e.g. **Test Compatibility Kit**
- Rust and Java components available on bespoke registries (crates.io, Maven Central)
- Partial implementation of traceability requirements (up-rust, up-subscription-rust)

https://github.com/eclipse-uprotocol

GOSIM®
GLOBAL OPEN-SOURCE INNOVATION MEETUP

GOSIM CHINA 2024

SDV™
Eclipse Software Defined Vehicle

Thank you!

Pete LeVasseur

Oxidation Partners