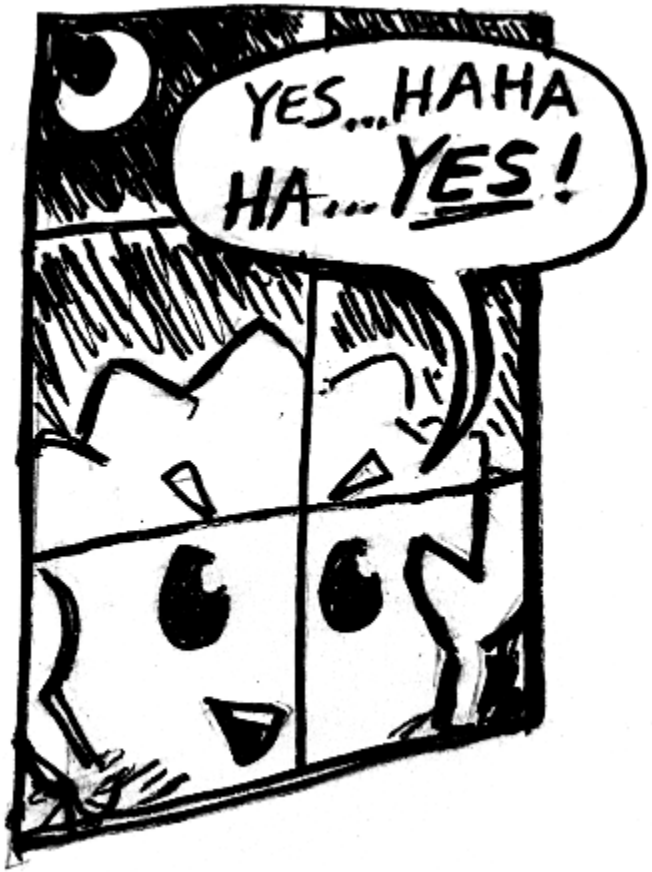
Rust, Agents and You(r Vehicle)

Safety-Critical Rust and Agentic Engineering

Eclipse SDV Community Meetup, Sunnyvale · Thursday, November 5

Pete LeVasseur
Lead @ Safety-Critical Rust Consortium
petelevasseur.com



Ferris art: @aldeka@wandering.shop, Nov 7, 2022

Extending uProtocol in Rust

Pluggable serialization

Your data → **Choose a wire format**
Protobuf · XCDRv2 · Arrow · OMG IDL
→ **Transport binding**
Zenoh · DDS · MQTT · SOME/IP

Zero-copy on a shared-memory path

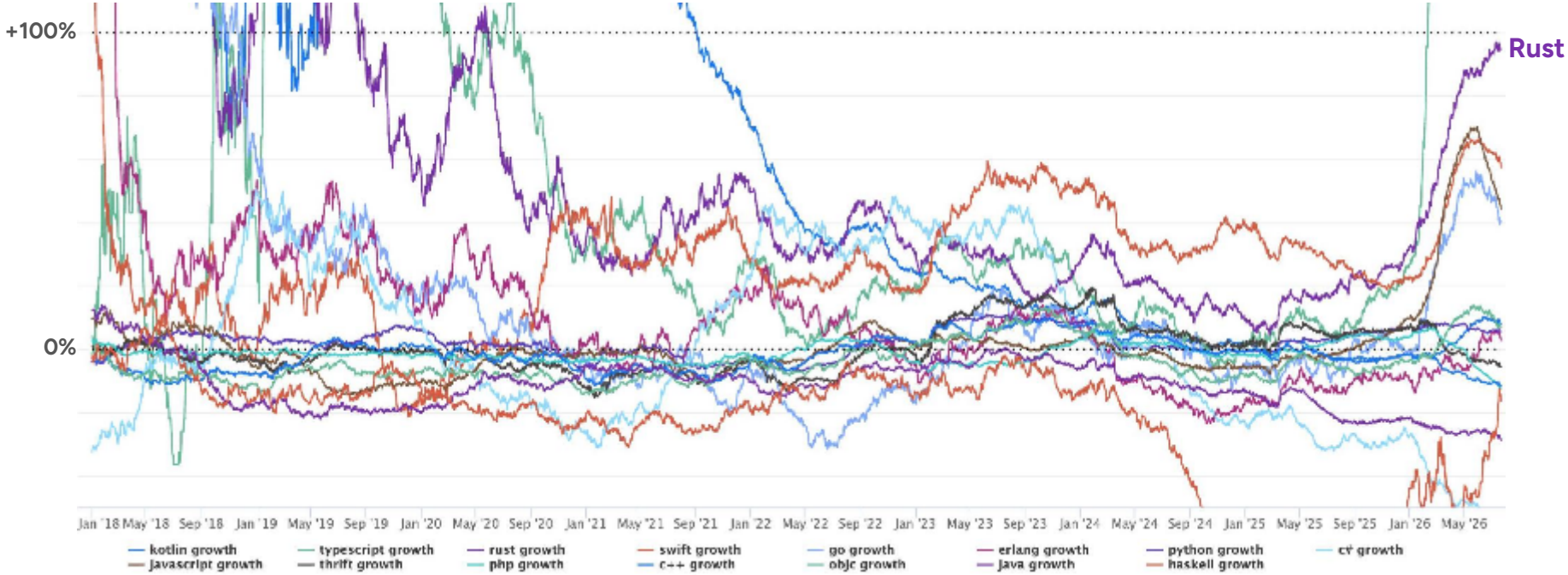
Sender: fill a loan → **Shared buffer**
e.g. iceoryx2
→ **Receiver: borrow it**

uStreamer

Keeps the selected encoding.
Copies the payload once into each outgoing native loan.

Encoding spec merged · Local integration tests passed · Implementation review is next

Rust use at Meta, 2018 to 2026



Year-over-year change in each language's share of committing developers. Guides: +0%, +100%.

Chart: David Tolnay (Meta), x.com/dtolnay, August 11, 2026.

What Google found with Rust in Android

1000×

lower memory-safety vulnerability density for Rust in Android than for Android's C and C++ code

~4×

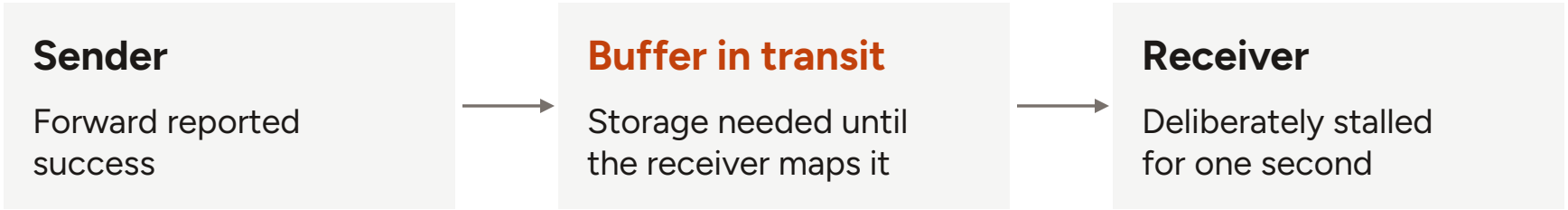
lower rollback rate for medium and large Rust changes in Android than for C++ changes

An engineer at a frontier model lab told me they use Clippy, Rust's linter, to grade generated Rust during training.

Google Security Blog, Nov 13, 2025, Android observations · Clippy detail: engineer at a frontier model lab, with permission

Missing requests between iceoryx2 and Zenoh

Streamer reported success for two RPC requests that did not reach the receiver.



0 / 31

Zenoh 1.9.0

31 / 31

Zenoh 1.10.2

Queued messages received in this test

Stalled-receiver test · Zenoh issue #2628

Receiver setup in the CloudEvents tests

We added CloudEvents to the Streamer tests.

Sender → Streamer → Receiver

The test converts at both endpoints and compares payload and metadata.

Before

Context initialized when building a message to send.

Receive-only endpoints had no conversion context.

After

Context initialized before registering the receiver.

Test: receive a native frame before sending anything.

CloudEvents adapter · Native agreement already configured · October 2026

Maintainer review of the encoding proposal

Encoding registry

We replaced the duplicate tables with one registry.

Unrelated cleanup

I removed the service-topic cleanup from this proposal.

MQTT message lifetime

The TTL fix got its own proposal.

Payload-encoding specification merged on October 26.

Public maintainer review: up-spec #336 · Separate MQTT TTL proposal: #346

A two-day Rust workshop

The engineers asked for a code review.

August 2026, with experienced automotive software engineers working in pairs with their usual coding agents.

Day 1

Teaching, then canscan

Day 2 morning

Read the code together

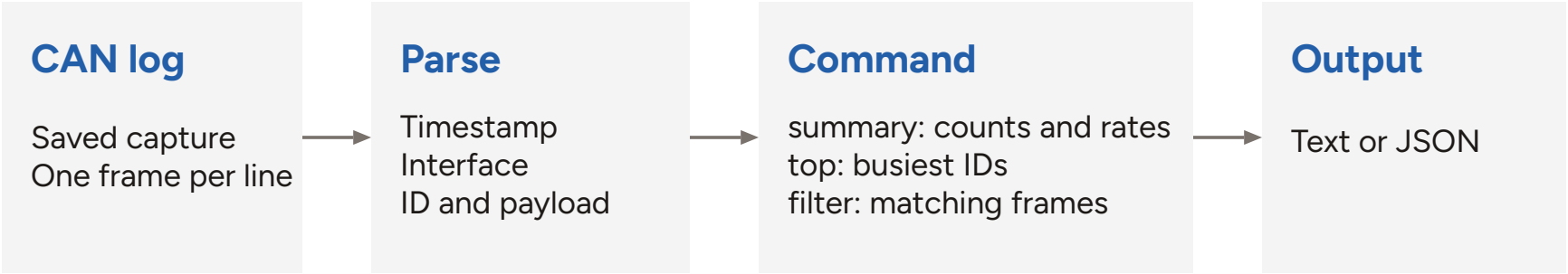
Day 2 afternoon

guardian

Workshop as run · Course and exercises: petelevasseur.com/guiding-rust/

What canscan does

All three groups had the same job.



One log line:

```
(1712345678.123456) can0 18FF0102#DEADBEEF11223344
```

Target: summarize 90 million lines (4.59 GB) in under 60 seconds.

canscan exercise specification · Each group chose its own implementation.

Three approaches to canscan

Group Layered

Before

Frame: payload bytes
FilteredFrame: hex text

After review

One frame type, with formatting in the output code.

Group Ambitious

After meeting the 60-second target, a further rewrite used unchecked conversion of file input.

Group Simple

462 source lines, one dependency
Reported runtime: 33 s

We reviewed it together and fixed a parser panic.

canscan · The groups took on different scope.

Reading the generated code together

1 Draw the architecture, without the agent

2 Trace one valid line, one malformed line

3 Find the code or test behind each claim

4 Change a behavior and check your prediction

Make your prediction before asking the agent.

Example: Group Simple

CAN ID width check changed:

`> 3` to `> 8`

Two tests failed.

Which requirement does each of those tests protect?

Was skipping UTF-8 validation worth it?

After a 20.9 s run with file-input validation, they asked for more optimization.

```
// parse_line works on &str — candump is ASCII so this is safe
let line_str = unsafe { std::str::from_utf8_unchecked(line) };
```

`FrameIter::next` uses raw file bytes without validating UTF-8.

from_utf8_unchecked requires valid UTF-8.
The parser accepted invalid UTF-8 (0xFF) into a String.

2.2%
0.35 s saved by skipping the check

13.4%
2.35 s saved by the hash-set changes

I'd keep the aggregation changes and restore UTF-8 validation.

Separate follow-up comparisons · 90M lines, M3 Max, Rust 1.98.1 · std: from_utf8_unchecked · 0xFF: one observed run

Reviewing the parser with Google's Rust skill

It asks the agent to check the safety requirements against the code.

I ran it on this parser:

3 → **2**

unchecked calls to review

file-input calls flagged as unsound

"candump is ASCII so this is safe"

Neither call validated the file bytes.

Its suggested fix: validate UTF-8 and return a parse error.

github.com/google/rust-skills/tree/main/unsafe_rust_review

Run on this parser: October 30, 2026

Recording a review finding

Here’s how I’d write up the missing UTF-8 check for another engineer.

Claim	Every unchecked conversion receives valid UTF-8.
Status	**Refuted**
Evidence	The 0xFF example, and file-input paths with no UTF-8 check.
Owner	**Name the engineer who will fix it and check the result.**

guardian kept returning Brake after reports stopped

Workshop arbiter: radar and camera reports, one decision every 50 ms.

cycle 0	cycle 1	cycles 2–4	later
radar: danger	camera: danger	both: no reports	both: moving away
Warn	**Brake**	**Brake**	**Brake**

Code Compares saved danger-report times with each other, never with the current cycle.

Tests The moving-away test starts fresh, without a prior Brake decision.

What should it do when reports stop, and how soon?

Holding Brake may be appropriate. The response and timing need a requirement.

Abridged follow-up check · ASIL not established · ISO 26262-6:2018, 6.4.2 (e), (g)

Preparing the uProtocol patches for review

Payload identity

Encoding ID and payload format

→

Frame and storage

Ownership and buffer lifetime

→

Transport and roles

Send, receive and shutdown behavior

→

Integration and guides

Transport tests and examples

Each patch includes an explanation and checks that pass at that point in the series, so reviewers can assess it on its own.

Planned review order · Local integration tests passed · Upstream review is next

I want us to use Rust in automotive

Rust gives us a strict compiler and great tooling, and agents have helped me take on more of this work.

I want the other uProtocol maintainers to be comfortable changing this code after it lands.

uProtocol: github.com/eclipse-uprotocol · Workshop: petelevasseur.com/guiding-rust/

Thank you!

Pete LeVasseur
Lead @ Safety-Critical Rust Consortium
petelevasseur.com



Ferris art: @aldeka@wandering.shop, Nov 7, 2022

Safety-Critical Rust Consortium
https://arewesafetycriticalyet.org/

Eclipse SDV WG
https://sdv.eclipse.org/

Rust Project: FLS
https://rust-lang.github.io/fls/

github.com/Safety-Critical-Rust-Consortium/safety-critical-rust-consortium

github.com/Safety-Critical-Rust-Consortium/safety-critical-rust-coding-guidelines