Dynamic Agents Readout

April 28, 2016 | Genivi AMM

Anson Fan
Data Architect
Jaguar Land Rover

Outline

Problem Description

As cars become more connected users will shift to the best user experience that can get them to the content they want the fastest.

Biggest challenge for Automotive is that once a vehicle is in the field we don’t know what features are being utilized.

Being a connected vehicle that will most likely be upgradable over the air in the future, being locked into static data won’t solve the issue.

Purpose of Dynamic Agents

Dynamic

- Send scripts at will over the air through RVI to a targeted fleet of vehicles
- Receive newly defined data streams straight to your existing infrastructure
- Agents can have an expiration date or be remotely terminated

Pre-processing of data

- Take advantage of a full fledged scripting language in a sandboxed environment to pre-process data

Built leveraging RVI features

- By leveraging key features of RVI such as security and message persistence, creates most reliable data transmission

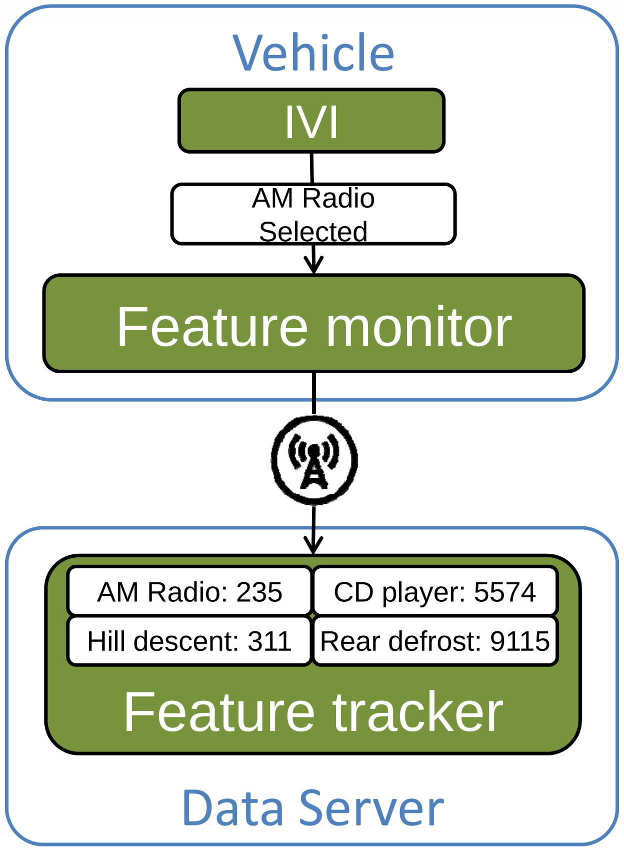
Re-Usability

- Write one script, use for multiple make and model year of vehicles.

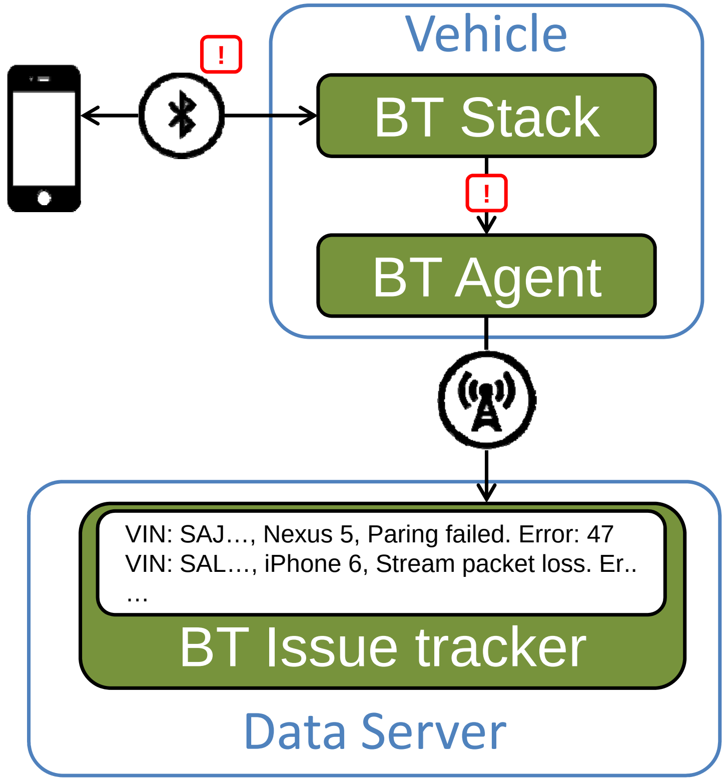
Use Cases

- Feature Usage
- BT Stack Issues
- Driver/Vehicle Anomaly

Use Case: Feature Usage

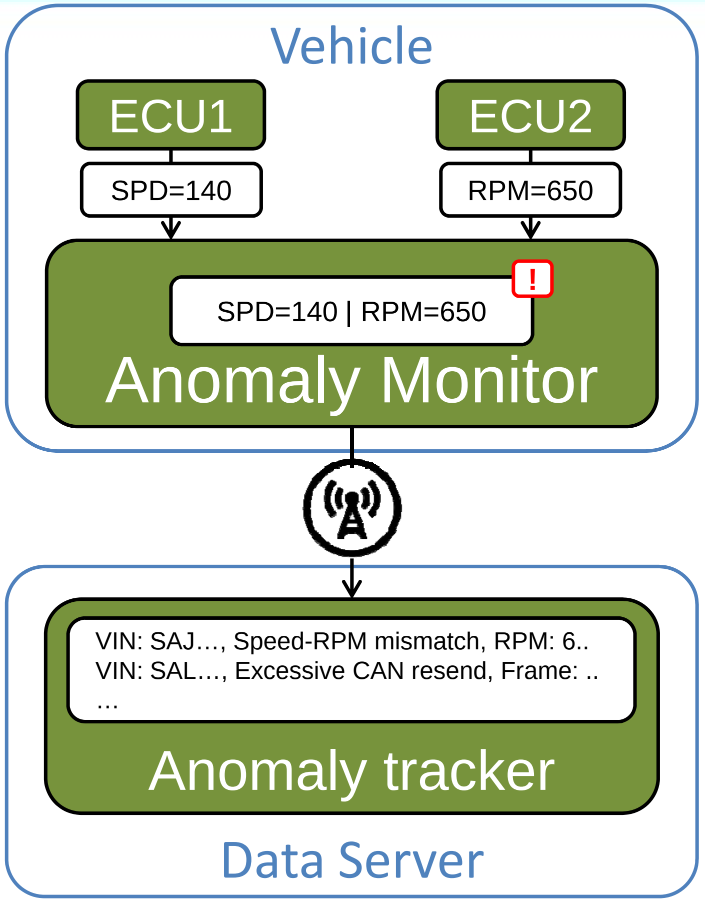


Use Case: BT Stack Issue

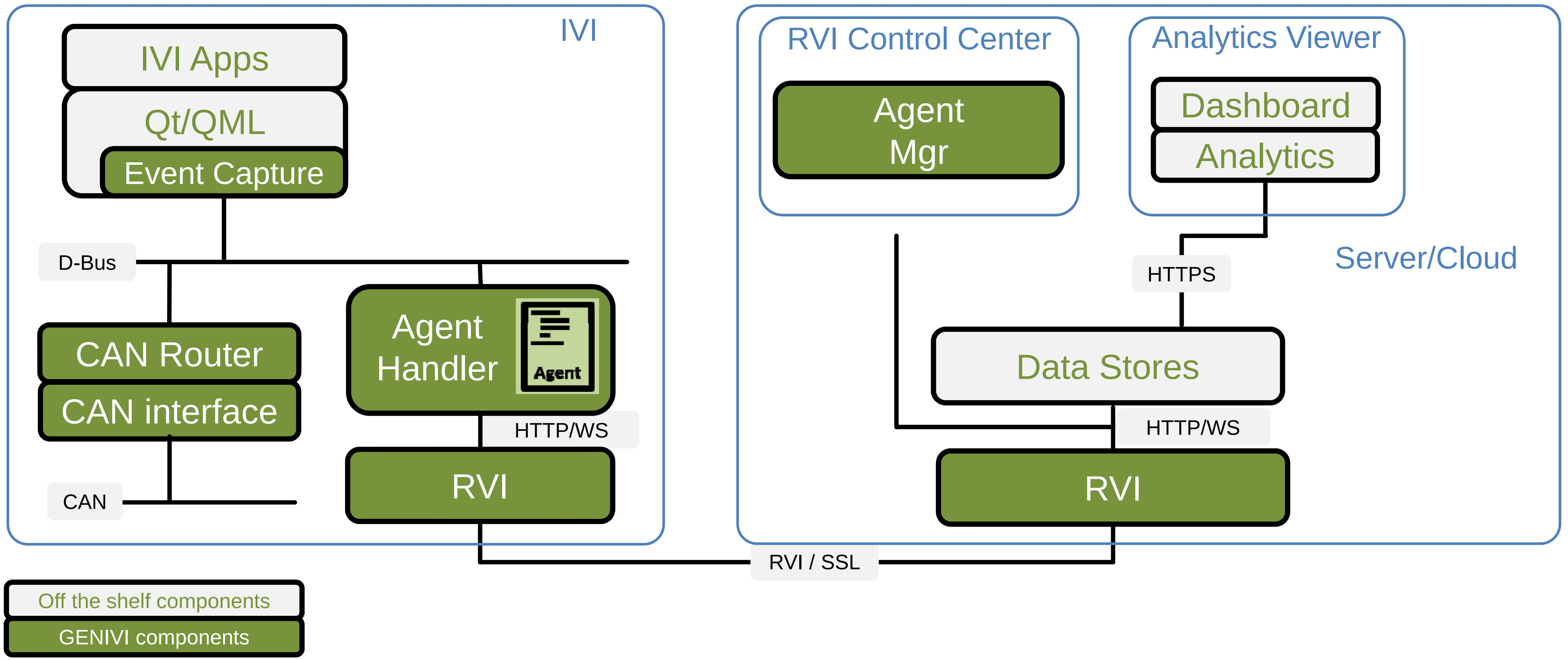


VIN: SAJ…, Nexus 5, Paring failed. Error: 47
VIN: SAL…, iPhone 6, Stream packet loss. Er..
…

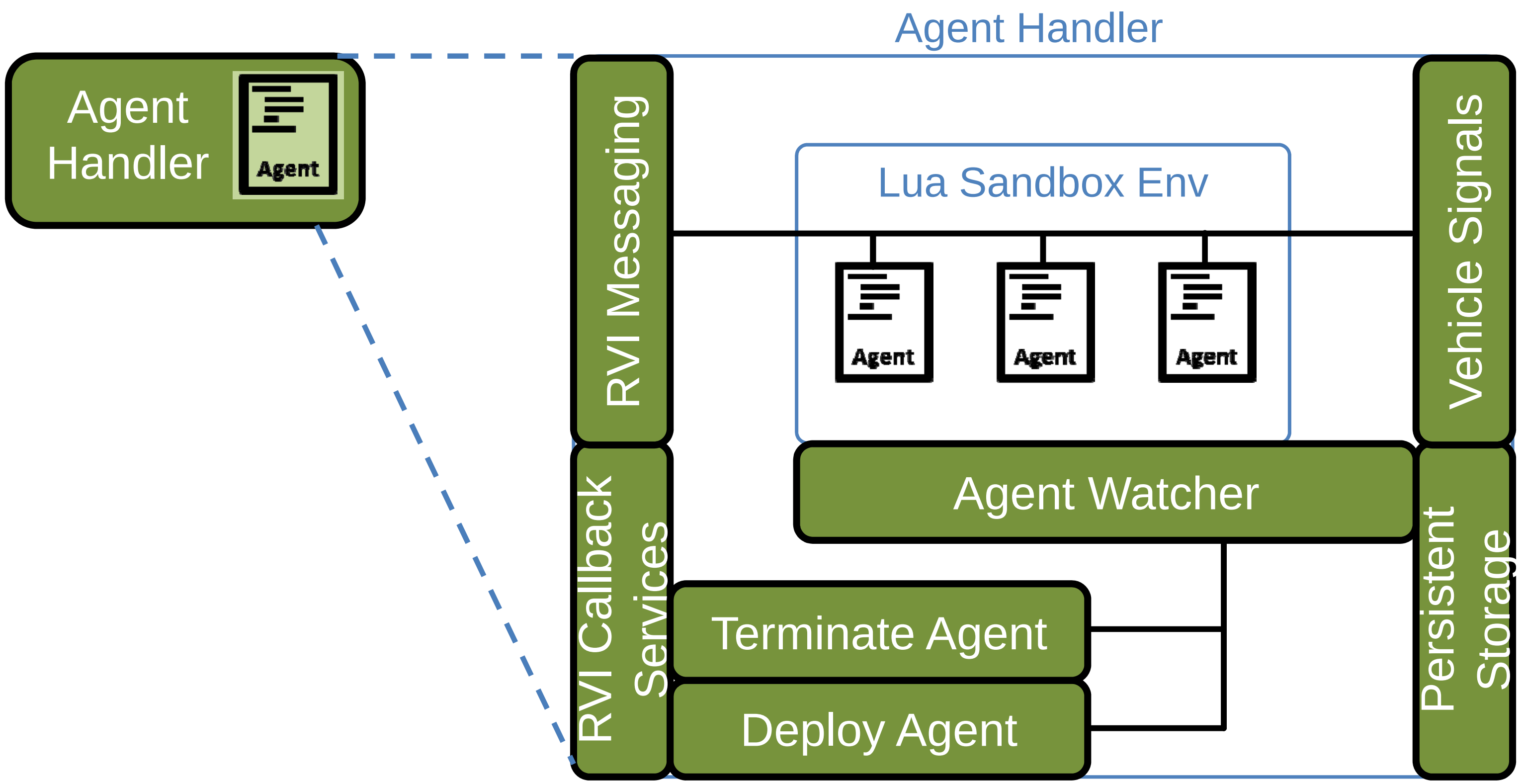
Use Case: Driver/Vehicle Anomaly



High Level Architecture



High Level Architecture Cont.



RVI Communication Transport Protocol

Connectivity

- Utilize a wide array of data links to setup communication to and from vehicle, either P2P or via backend serve
- Provide encryption for secrecy, non repudiation, replay attack protection, etc
- Work with OMA, IEEE, and other organizations to standardize RVI and integrate existing communication standards

Authentication

- Prove the identity of communicating parties
- Use best-of-breed open source technologies to drive peer-reviewed security

Authorization

- Prove to remote parties the right to discover and invoke their services.

Service Discovery

- Announce services available to remote parties

Service Invocation

- Invoke services and report the result over unreliable data links that may change during execution
- Support retry and store & forward of service invocations to alleviate transient connectivity

Lua Scripting Environment

Why did we choose Lua?

- **Easily modifiable runtime environment**
  The Lua runtime environment is actually just a regular table that is referenced by the global variable "_G."
- **Lightweight**
  Having a barebones Lua parser/compiler/interpreter can weigh in under 100kb
- **Simple API to communicate with native C code**
  Lua is an extension language and has easy to use API for communication between C and Lua scripts
- **High level syntax similar to that of Python**
  Things such as automatic type casting and the idea of tables which is equivalent to dictionary and list in Python

Lua Scripting Environment Cont.

```
local to_load = {}
to_load["time"] = true
to_load["cjson"] = true
to_load["rvi"] = true
to_load["agent"] = true

local white_list = {}
white_list["setmetatable"] = true
white_list["ipairs"] = true
white_list["utf8"] = true
white_list["rawequal"] = true
white_list["pairs"] = true
```

```
for key, value in pairs(to_load) do
   load("_G." .. key .. " = require(\"" .. key .. "\")")()
end
```

```
for key, value in pairs(_G) do
   if white_list[tostring(key)] then

   else
      load("_G." .. tostring(key) .. " = nil")()
   end
end
```

Future Roadmap + Tasks

Common API	• Make any communication that goes over the D-Bus run over Common API
Lua API	• Flush out the agent API so that scripts can be much more configurable and provide more standard libraries to use.
VSI/VSS	• D-Bus TP/S not as high as we'd like and has a relatively high CPU load • Have agents be designed for standardized signal set for greatest reusability
Better Persistent Storage	• Replace current write directly to JSON object to use SQLite for data persistence
Testing Utilities	• Create virtual vehicle simulator to allow for easier testing of dynamic agents before actually pushing to fleet

Conclusion

- Send new remote probes over the air and remotely terminate ones that aren’t needed anymore
- Pre-process, capture, and off board only relevant data from your fleet of vehicles
- Re-use the same script for a whole fleet of vehicles of different makes and models
- Github Repo:
  https://github.com/PDXostc/rvi_dynamic_agents
- Questions?