Software, Over the Air

Leon Anavi
Konsulko Group under contact by ATS
AGL Face to Face Technical Meeting
25 – 27 May, Vannes, France

Konsulko Group

Advanced Telematic SYSTEMS

SOTA

- **Updating software is important.**

  It’s even more important when it’s software running on 3 metric tons of fast-moving steel.

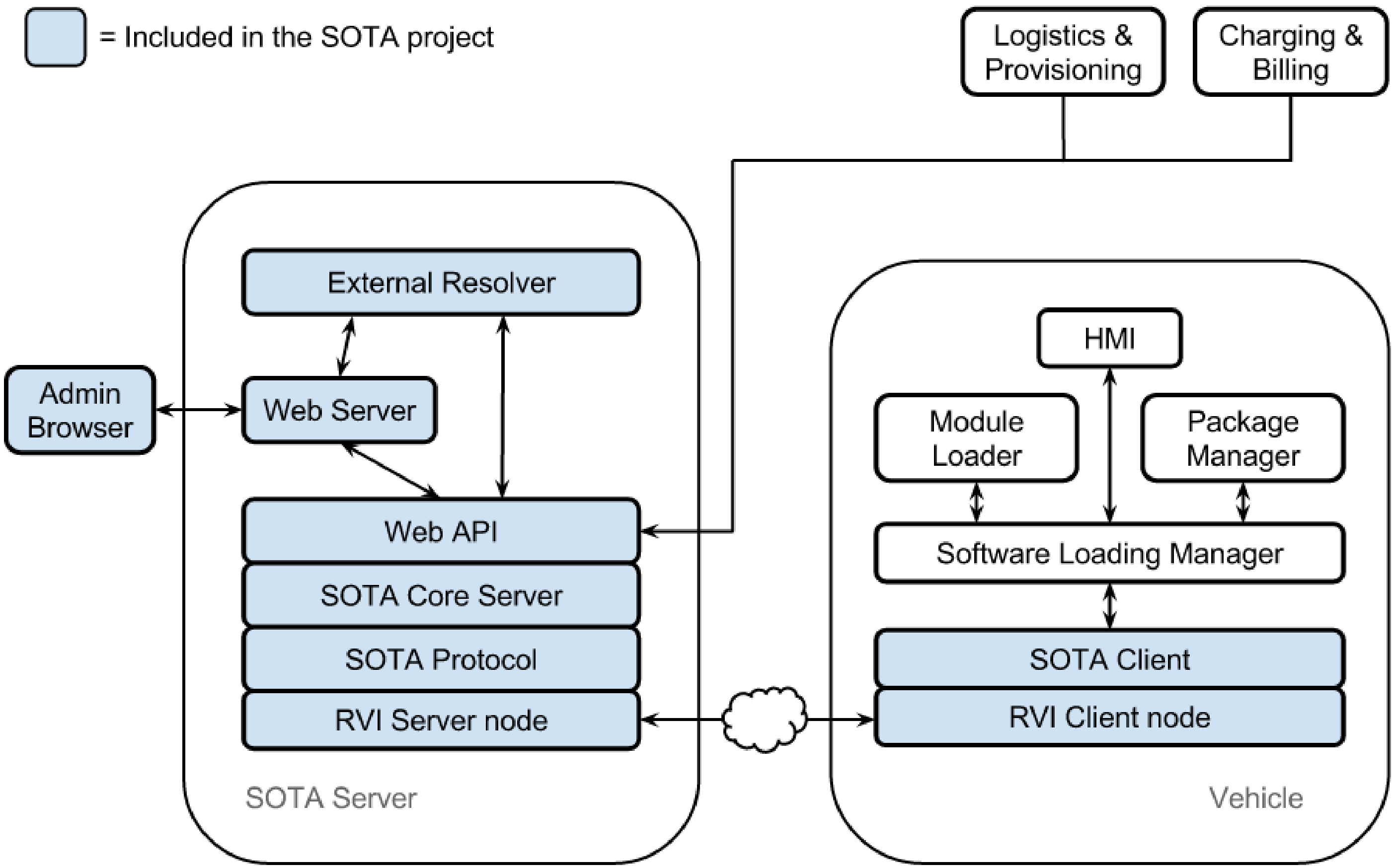
- **Getting your users to update software is hard.**

  It’s even harder when your users are running software on a thing they don’t really think of as a computer.

GENIVI SOTA Project

- A complete suite for uploading, managing, queueing, transmitting, validating, and deploying software updates remotely to a fleet of vehicles
- Server + Client
- Open source repositories in GENIVI GitHub

SOTA Architecture



SOTA Server

- ❑ Web Server
- ❑ SOTA Core Server
- ❑ External Resolver

SOTA Client

- SOTA client implementation written in the Rust programming language (ATS is also investigating a C implementation)
- Remote Vehicle Interaction (RVI) and/or HTTPS communication based on JSON-RPC
- Integration of RVI SOTA Client in Automotive Grade Linux (AGL) and GENIVI Development Platform (GDP) through Yocto/OE recipes and layer meta-rust

SOTA Client & AGL

- Layer meta-rust in AGL-repo which provides recipes for building Rust and Cargo
- Recipe rvi-sota-client_git.bb in layer meta-agl which builds and deploys RVI SOTA client and its systemd service

Running SOTA client

- Add RVI SOTA client to the image by appending the following line to conf/local.conf: IMAGE_INSTALL_append = " rvi-sota-client "
- Run Docker images for RVI server and client node
- On the target device add IP of rvi-client in /etc/hosts
- On the target device restart systemd service rvi-sota-client and verify that it is ready to accept connections

How does it work?

- SOTA client downloads the binaries from the server, and then hands them over to the system for installation
- Dbus interfaces for communication with to any software compatible with GENIVI Software Management APIs
- WebSocket interface for direct communication with HTML5-based applications
- Documentation: https://genivi.github.io/rvi_sota_server

Installation Strategy

- Whitepaper about software update management on AGL devices by Matt Porter (CTO of Konsulko Group), sponsored by Advanced Telematic Systems (ATS): http://bit.ly/25cZVJZ
- Join the discussion in AGL mailing list

Installation Strategy

Requirements (in priority order):

1. Atomic software release update
2. On failure, deploy previous working bootloader, kernel and configuration, and filesystems on AGL device
3. Update of bootloader, kernel and configuration data, and filesystems on AGL device
4. Support for OpenEmbedded-based builds
5. Support for updating both the AGL device and any ECU devices

Installation Strategy

6. Flexible delivery of software image(s) with QoS controls and supporting arbitrary interfaces

7.Support for signing of images and verification of images on installation

8. Support trusted boot and execution of software update in a trusted application environment leveraging the platform’s hardware TPM and/or TEE features

9. Enable/disable a specific feature and apply/rollback system updates incrementally

OSS Update Tools

- SWUpdate
- Mender
- Resin
- swupd
- OSTree
- Other

Trusted Zone

- Execute update process in Trusted Execution Environment (TEE)
- Support OP-TEE using an ARM QEMU target for PoC