WHITE PAPER

A NEW CONCEPT IN OTA UPDATING FOR AUTOMOTIVE

OTA Updates are not a new concept. They first became a wide-spread technology for remote updates with the introduction of 3G networks and mobile phones. With the advent of connectivity in automobiles we are seeing OTA Updates start to be used in place of recalls. In fact, the early suppliers of OTA Update solutions to the automotive industry merely repurposed technology they had invented for mobile phones. Is technology that was first developed for Nokia Series 40, Motorola P2K and Android suitable for embedded systems such as brake systems?

Most of the existing solutions need a client to extract the delta, using an algorithm that requires at least a CPU with 200Mhz horsepower to perform in a reasonable time frame. This might be acceptable for a modern smartphone but does the airbag module have this fancy HW?

A car is fundamentally different from a smartphone in that it has multiple internal networks and multiple independent computing end units, from multiple vendors. To build a solution that works in such an embedded environment standardized file formats (ELF file, S record, or Intel Hex format) need to be used. Proprietary BIN files should be avoided as these would force the vendor to design a client on the ECU to extract it, which in turn would create new dependencies in software distribution systems.

Current solutions need five memory-hungry procedures, such as (1) reading blocks to the RAM from the flash, (2) patch-write the delta files on the RAM, (3) burn-write the patched block to a redundant memory block, (4) erase the flash sectors, and then (5) burn-write the patched block back to the image flash.

Can current OTA Update solutions rollback to a previous safe and certified software version should a problem occur with the new version, while avoiding re-distribution of the previous software versions, and writing them again on the flash?

Is it possible to update new functionality while ensuring uptime of the embedded system and a seamless user experience for the driver?

Can the solution support different car ECUs: Infotainment, Safety, Powertrain, ADAS, Chassis, Body & Comfort, today and in the future? Using the same technology?

Can the OTA Update solution meet the reflash time and power requirements of the production line? Enabling all the ECUs to be updated in parallel with the latest software versions prior to leaving the factory and without causing disruption on the production line?

Does the OTA solution require client integration work by all the ECU vendors in the vehicle eco-system?

INTRODUCING AURORA LABS' 3D-DIFF™ OTA UPDATE

At Aurora Labs we have introduced an In-Vehicle Software Management solution that has been designed for the Automotive industry with these questions in mind. Our 3D-Diff™ algorithm creates the industry's smallest update files and the clientless solution uses standard programming protocols to remove the need for integration on the target ECU.

STANDARD PROGRAMMING TOOLS

Aurora Labs' delta.bin files are standardized Intel Hex, S Records that bind uncoupled, non-related changes between binary files as shown in Figure 1.

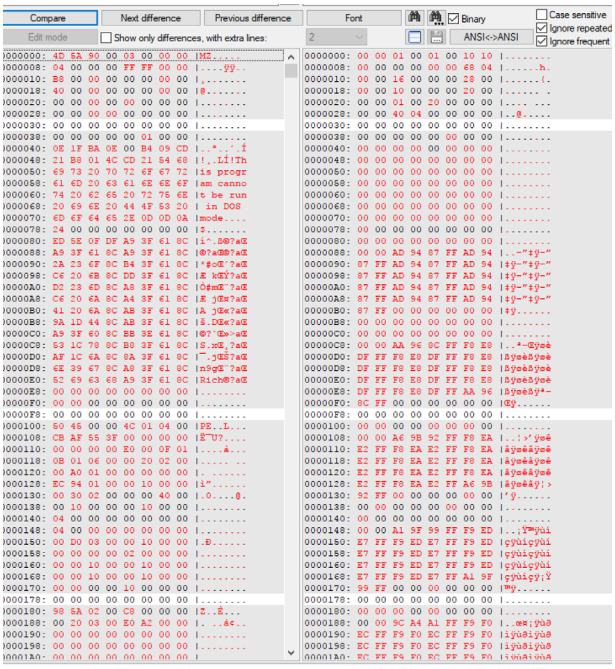

Figure 1, illustrates a diff between two bin files

Those changes binded into one file - ELF - a standard file that is ready for distribution as an S19 record. In POCs, we have shown how we create a delta.bin file or an S19 record and burn it to the flash without reprogramming the entire flash, all in a single write process with zero downtime!

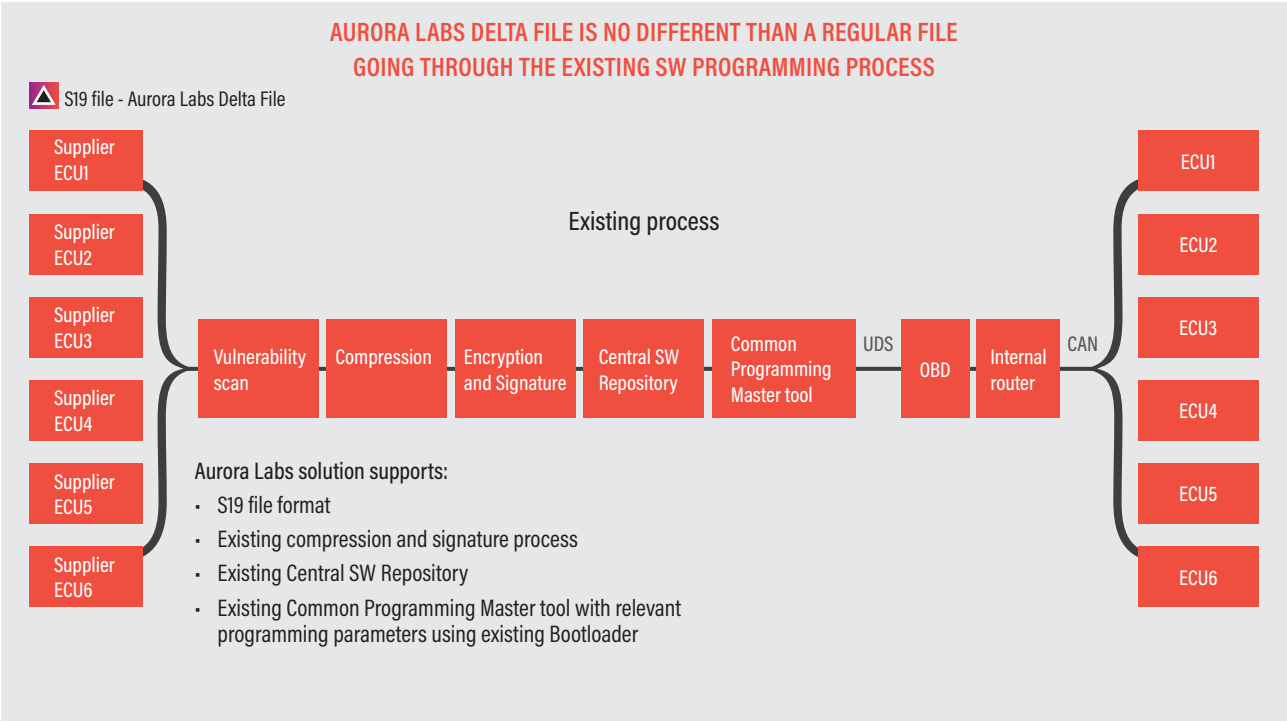

Figure 2, illustrates a common OEM SW distribution channel

One technology barrier that we solved was finding a way to burn the delta out of the original program body on the memory. We do not patch the original program. We use a standard bootloader—JTAG, TRACE32, and the UDS protocol.

Our S19 records are ready to be patched into the next free space on the memory, which can be any kind of memory technology (Flash, RAM, etc), as described in Fig 3.

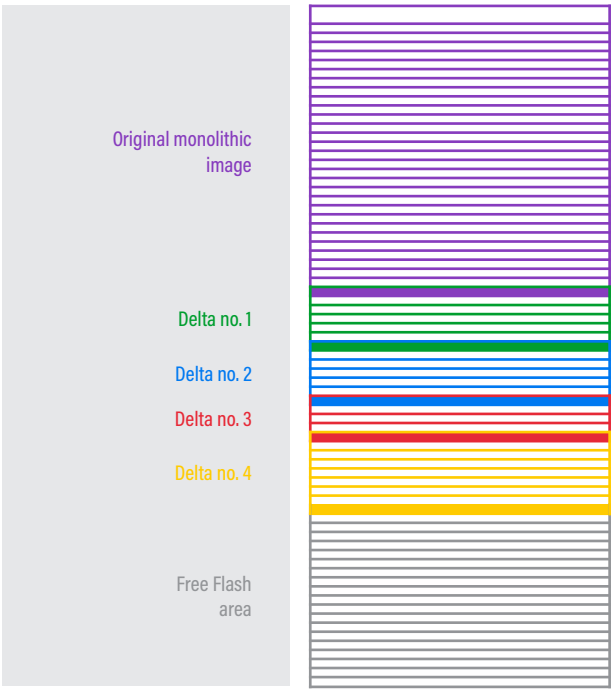

Figure 3, illustrates the Aurora Labs delta out of the original program body on the flash without reprogramming the memory

UPDATE TIME, SIZE AND RESOURCE REQUIREMENTS

By this technique, we avoid patching the original program, which would mean going through an extensive sequence of reading the original program, patching it on the RAM, erasing the flash, and then reprogramming the flash with the patch.

As we do not need to erase the existing image, we can update the ECU without taking it offline.

In a POC we ran with a leading European OEM, our technology was compared (Figure 4) with a full reflash in a garage/workshop, an online full reflash and other existing differential update technologies. The Aurora Labs 3D-Diff technology created a smaller delta, required less flash sector writes and significantly reduced the offline time to only the reboot that occurs when the car is next switched off and started again.

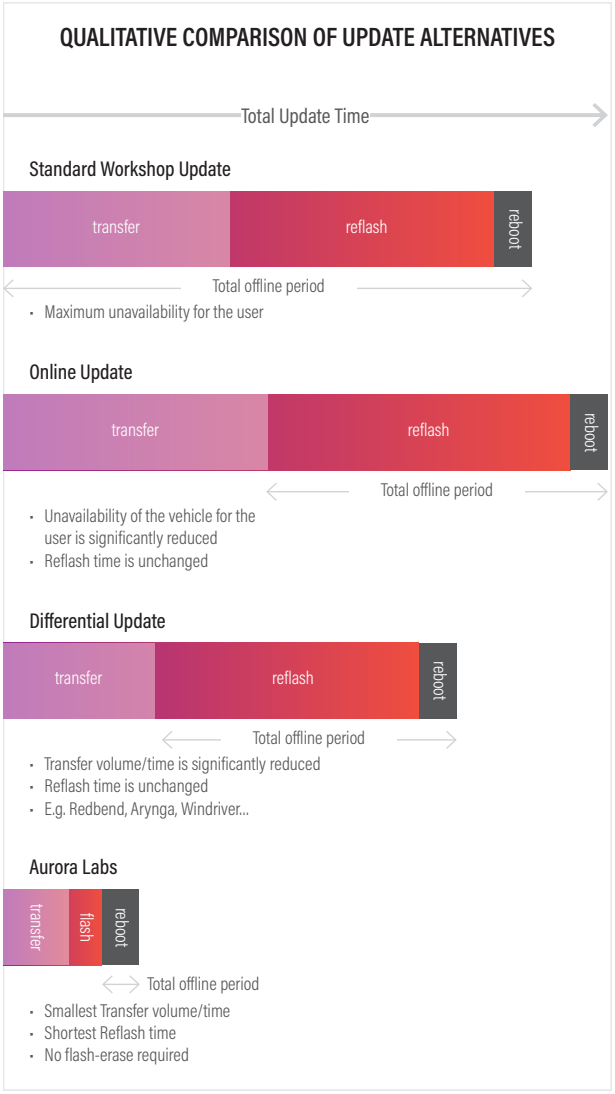

Figure 4, Qualitative Comparison of Update Alternative as reported by a leading European OEM

Our 3D-Diff algorithm encapsulated uncoupled differences in the bin files, as shown in Figure 1, into a small virtual file system (VFS). In addition to enabling fast, online updates, this also gives us the capability to rollback between the deltas that have been written to the flash.

Each delta is, in fact, a build, or an independent software version, as shown in Figure 3.

The size of the VFS is less than 1% of the flash memory. Going through the VFS requires only 5 CPU cycles, it is thus very lean. To put this in perspective, a basic mathematical divide operation (x/y) requires 30 CPU cycles on an MPC5748g ECU by NXP (a well-known body controller unit). Power savings such as these will have a huge impact on the production line where the fear of discharging the battery and of missing the allocated window for software updates of an unpredictable amount of software is causing great concern.

We also tested the delta update technique on a small 8 bit 16 Mhz sensor. In this test, the application size was 43KB, and Aurora Labs' VFS footprint was only 273 bytes (0.63%).

On a larger project with an application size of 5MB, our VFS was in the range of only 10KB (0.2%).

A POC we conducted with another leading European OEM led to the following independent and verified results:

- 3D-Diff between AutoSAR v3 to AutoSAR v4 is 79 Kbytes VS BSdiff 180 Kbytes - less than half as big
- Diff in S Record
- Zero downtime, no need for A/B memory
- Clientless
- Rollback with zero downtime no A/B memory usage
- Single write/burn, without using a client to extract the delta.bin
- 20 times faster than current solutions
- 25 times power consumption reduction
- 10 times flash memory friction reduction, using far fewer write cycles on the flash memory, increasing its lifespan

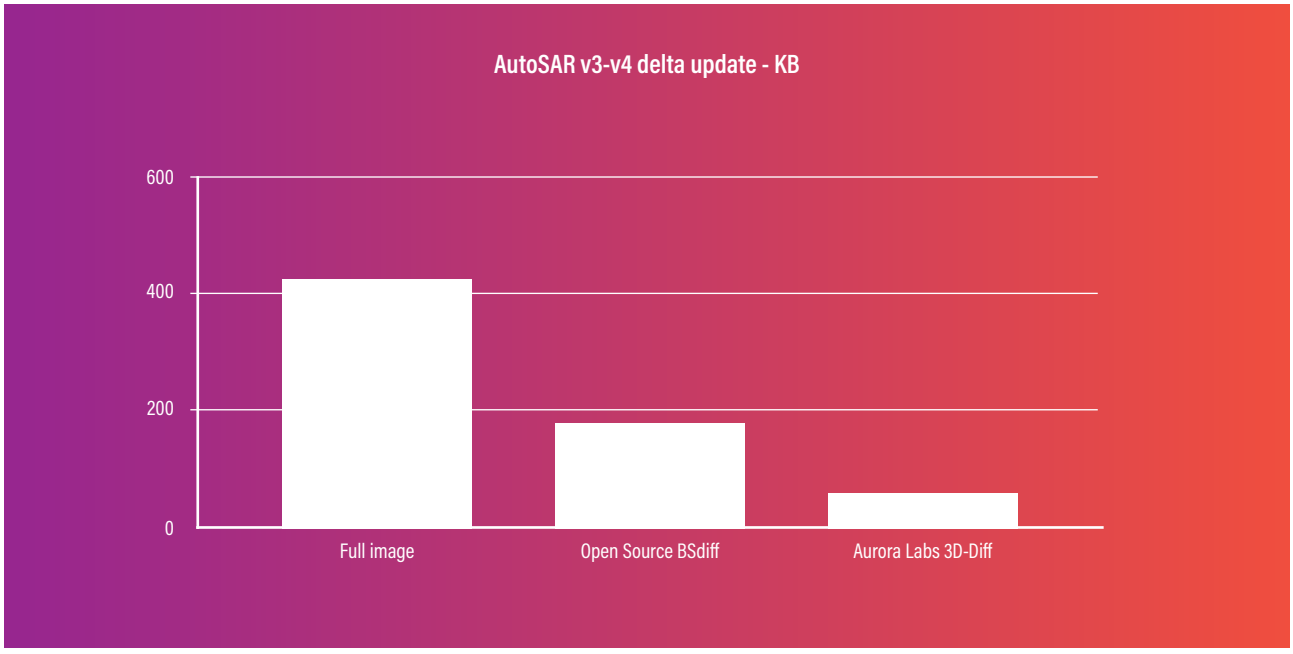

Figure 5, depicts the results of the OEM verified POC

FROM VFS TO VCU (VIRTUAL ECU).

Electric Vehicles (EVs) have opened the market to new players. While new automakers such as Tesla have a lot to learn from the history of traditional OEMs, they are also free to drop old technology and develop a clean version of the connected car, as Tesla has shown. Further, new EV projects within traditional OEMs see the opportunity to change the concepts of the previous architectures, with new concepts such as Service architectures, ECU Domains, and Virtual Machines (VM). VMs and Hypervisors form the foundation and infrastructure for VCU - Virtual ECU.

Moving from ECUs to VCUs will reduce the number of physical ECUs, lower BOM costs and offer the ability to work with virtual flash environments and multiple operating systems (OS) such as AutoSAR, Linux, and others.

Aurora Labs thus designed our solution to support these future architectures, agnostic to memory technology or OS, while cracking the embedded technology barrier and supporting today's architectures.

SUMMARY

Current OTA solutions, imported from the world of traditional IT and mobile phones is not best suited for the challenges of the connected car, not now and not in the future.

At Aurora Labs, we took into consideration all the limitations and challenges of the embedded automotive architectures and developed our 3D-Diff™ technology to address them all.

Without the need for a dedicated client, OEMs can use our solution to retrofit existing products for on-the-road cars and for new projects while investing minimal engineering and integration time.

As delta-based OTA Updates become the norm for the automotive industry, significantly improving software recalls, solving software security vulnerabilities, and enabling the introduction of new software and OTA features we believe that the Aurora Labs In-Vehicle Software Management solution is the best fit for the industry, now and in the future.

ABOUT AURORA LABS

Aurora Labs is pioneering Self-Healing Software for connected cars to enable automotive manufacturers to proactively support to future vehicle software architectures, processes, and services. Aurora Labs' Line-Of-Code Maintenance™ technology is the foundation of its In-Vehicle Software Management solution. Using machine learning algorithms to uniquely address all four stages - detect, fix, update and validate - of a software management solution, Aurora Labs future-proofs the next generation of software-driven automotive features. From detecting line-of-code faults to predict downtime events, fixing errors on-the-go to provide a safety-net for new software rollouts, enabling reliable and cost-effective rollouts of new automotive features to all ECUs in the vehicle without any downtime for the user and validating changes to the software to facilitate homologation, Aurora Labs is paving the way for the age of the self-healing car.