electronic design
The Authority on Emerging Technologies for Design Solutions

M2M Client-Side Challenges Emerge In Mobile Embedded System Updates

Electronic Design

Chris Ault Tina Jeffrey
Chris Ault and Tina Jeffrey, QNX Software Systems

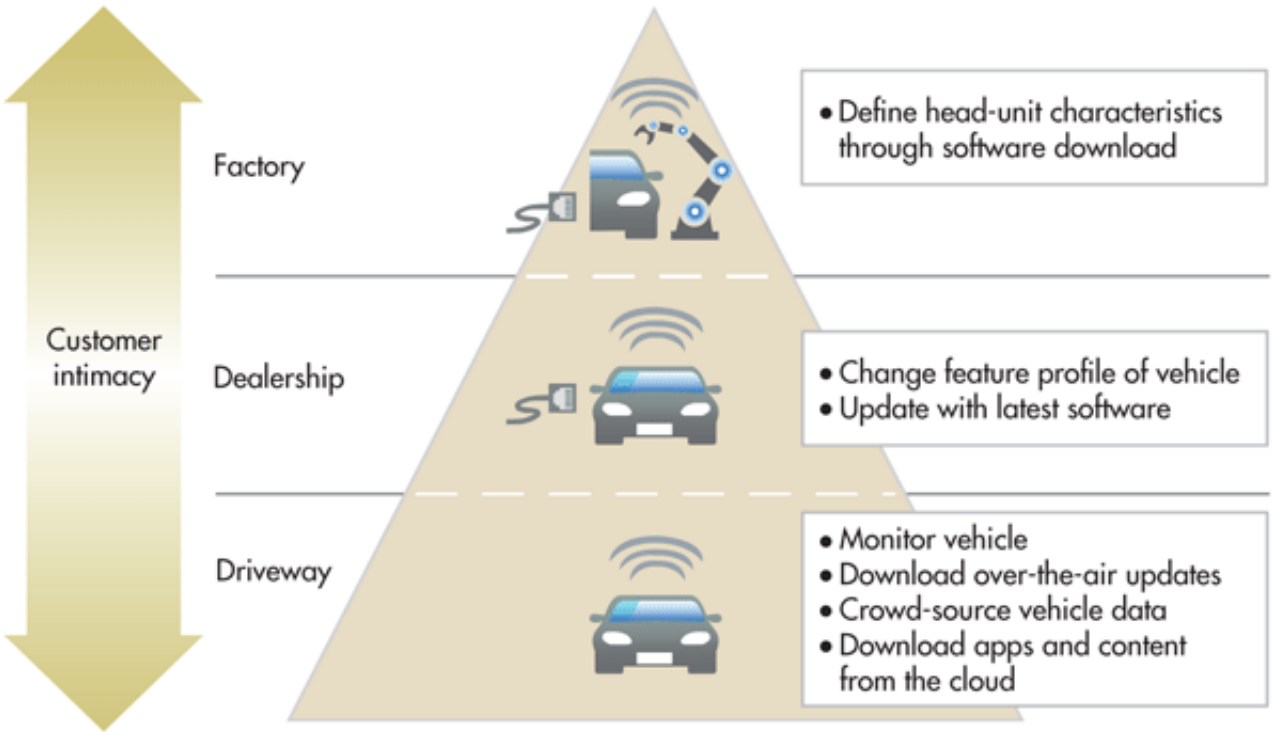
Mon, 2013-05-06 13:56

Related Articles

- Use Lua To Develop M2M Embedded Applications
- A Successful "Internet of Things" Hinges On M2M
- Compact GSM Modules Add Versatility For M2M Apps

Machine-to-machine communications (M2M) will usher in a third industrial revolution, according to Jürgen Hase, a vice president at Deutsche Telekom's M2M Competence Center.[1] M2M isn't a new concept, though. Old-fashioned telematics such as the interaction between an ATM and a bank's databases is M2M communication, as is the communication between a two-wire house thermostat and a furnace. Today, however, M2M is chiefly about communication between electronic devices over wireless networks.

Mobile platforms, from smart phones to satellites, are major drivers of this trend. M2M communications are an inextricable component of their design, used for the so-called FCAPS functions: management of faults, configuration, accounting, performance, and security.[2] Implemented as a means for updating software and firmware (FOTA, or firmware over the air), M2M is also rapidly becoming a key element in strategies to increase device longevity, reduce maintenance and renewal costs, and even generate new revenue streams *(Fig. 1)*.



Automotive Systems

The number and diversity of current and possible M2M implementations in automobiles make them an excellent paradigm for examining issues with M2M-enabled updates to mobile platforms. Automobiles present many systems in many models, trims, and configurations, which must all be served by a limited number of update strategies. All things being equal on the server side and with the network infrastructure (they are reliable and secure), M2M-enabled updates to automotive systems present three major client-side challenges:

• Safety-related components: Most cars have both non-safety-related systems, such as infotainment, and safety-related systems, such as antilock braking systems.

• Limited computing resources: Unlike network switches, for example, automotive systems don't have the luxury of redundant processors and memory.

• Connectivity: Cars are truly mobile. Their locations and time of use (and therefore connectivity) are unpredictable.

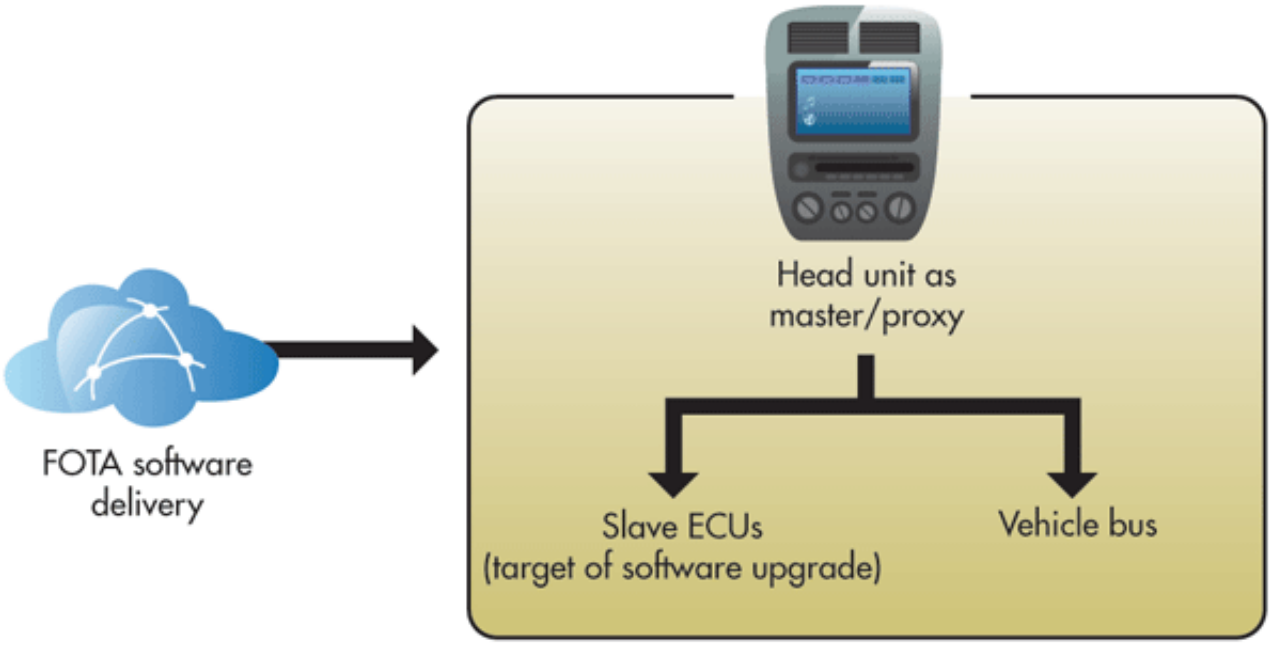
The Head Unit And TheECUs

Modern vehicles have a sophisticated infotainment system in a head unit and approximately 80 electronic control units (ECUs) for engine control, active suspension control, braking, and other functions. These components use firmware and, like the infotainment system, many ECUs are software controlled.

Already, users update many infotainment applications by accepting updates that are pushed to them. This isn't the case for ECUs, though, which require a visit to the dealership to have the module re-flashed or replaced.

In many cases, an M2M-enabled process managed by the vehicle head unit can replace this labor-intensive model by using

the vehicle identification number (VIN) and other parameters such as language preferences and subscriptions to determine which updates are relevant to that vehicle *(Fig. 2)*.



To perform this function, the supervisor processor requires:

• A remote query of the ECU module and software inventory, including information such as module type, manufacturer, serial number, manufacture date, and software version

• ECU software packaging to create an ECU software payload, validated by the head unit and distributed to the target ECU

• ECU software image signing

• Software update control and image transfer from the head unit to the ECU

Safety-Related Components

The first question that must be asked about any update, regardless of the technology used, is whether the affected device or component is safety-related. This is particularly relevant for ECUs, which may control critical vehicle functions, but it may also be relevant for components in the head unit. If the head unit is used to update ECU firmware or software, then it or its relevant components must be treated as safety-related components.

If a device or component is safety-related, then it is necessary not just to perform the update, but also to demonstrate that the updated component meets safety requirements such as IEC 61508 SIL3. Even if the updated component is not itself safety-related, proof is still required that the update doesn't compromise component isolation, as required by ISO 26262. And, the M2M-enabled update process may require approval by relevant agencies such as departments of transport in jurisdictions where it will be used.

Limited Resources

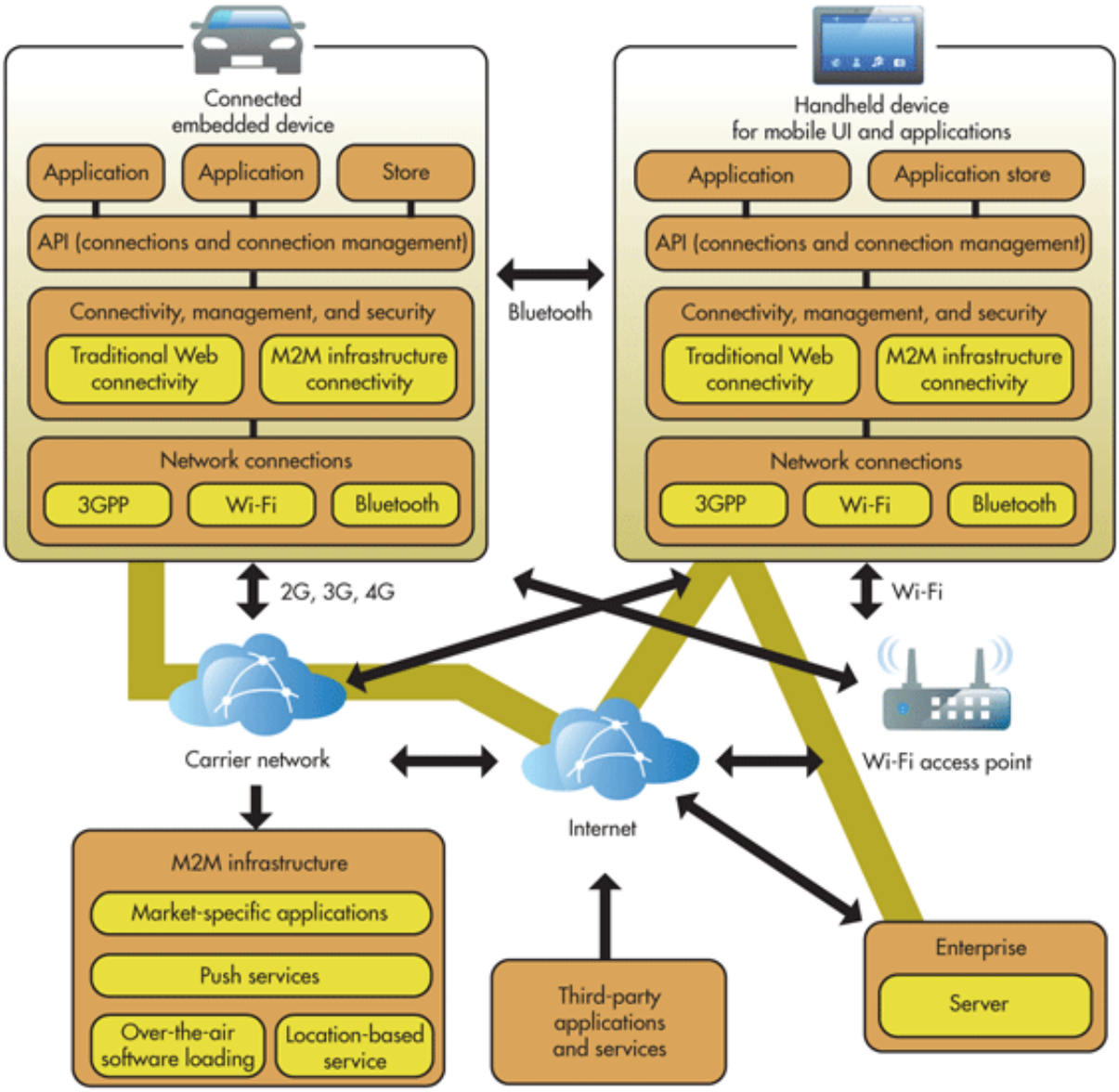
Strict limits on the capabilities of in-vehicle systems are imposed by:

• The physical constraints of their environment: There is an upper bound on the power consumption, heat production, and physical size of an ECU.

• Cost: The large number of units produced and the cost of bringing a vehicle to market is such that automakers can't pay premium prices for their processors. A 20¢ difference is significant when multiplied by 80 processors in 10 million cars.

• Time: When a car enters the market, its processors are already dated. A year is a long time in the world of computing, but cars take years from design to start of production (SOP) and must last at least a decade.

These limits mean that the in-vehicle hardware may not always have sufficient storage or memory to both keep the current version and perform an update. For ECUs, the vehicle head unit may provide the required excess capacity. In some cases, though, the head unit itself may have insufficient capacity for its own updates, especially over time as the vehicle electronics are required to handle larger updates designed for newer and more powerful processors *(Fig. 3)*.



Connectivity

Updates for sensors and actuators in, say, a factory assembly line may not need to account for a possible loss of connectivity during the procedure. For mobile platforms, however, loss of connectivity is a key issue. For many mobile platforms, the solution is simply to work with an understanding of the use patterns.

For example, movement patterns for smart phones are unpredictable, but users don't generally start updates while they're moving. Even if a phone does lose connectivity, the user can simply restart the process. At the other end of the spectrum, trains move across the countryside at high speed and may often run beyond areas of network coverage. However, the usage patterns of a train are highly predictable, so updates can be made during scheduled maintenance.

Like trains, cars move across the countryside quickly, but like phones this movement is not easily predictable. Updates performed while a car is moving run the risk of a connectivity loss when the car enters a tunnel, goes down an urban canyon, or moves outside its network area. Further, an update cannot count on the vehicle remaining stationary for any

length of time.

Imagine a car parked for the night 40 minutes from a hospital. Following its default configuration, at 2:00 a.m. the car's head unit checks for updates, finds several, and begins updating critical systems without which the car cannot run. Estimated time to completion: 17 minutes. At 2:03 the car owner, pregnant for the first time and in her seventh month, wakes up. Her water broke! No other vehicle is available.

In short, both the back-end serving up the M2M-enabled update and the client must be able to recover gracefully after a loss of connectivity or a cancelled update.

Help From The OS

If we summarize the challenges described above as isolation, footprint, and time, we see that we need an M2M-enabled update process that can:

• Isolate safety-related components from other components and from each other

• Perform updates with the limited computing resources available in the vehicle

• Complete updates without rendering the vehicle inoperable

• Gracefully pause and restart interrupted updates

The operating system (OS) alone cannot solve all the challenges of M2M-enabled updates to mobile systems, but it can provide a foundation on which to build. If dependability is essential, as is the case for automotive platforms, the OS should be a real-time OS (RTOS), which is designed to support availability and reliabilityguarantees.[3]

A microkernel OS architecture may also be an important asset for embedded systems, which don't have the luxury of redundant capacity, as might, for example, a network switch.[4] A microkernel architecture has drivers, file systems, networking stacks, and applications in separate address spaces, a design that helps ensure isolation of the update process and of new components from other components and the kernel. Equally important, the microkernel architecture can also accept small, targeted updates for individual components, including OS components, without affecting the rest of the system.

For instance, if a bug fix is available for a graphics driver, the update only requires the memory to run two graphics drivers and synchronize them. A high-availability manager can gracefully shut down the deprecated driver and switch to the new one.

Applications or services that depend on the component that is to be updated may need to be notified of the impending interruption, gracefully stopped before the deprecated component is switched out, and then restarted after the new component has been started. This approach places an additional burden on designers, but considering the number of systems in any vehicle line, this strategy may prove rewarding.

References

1. Jürgen Hase, "M2M: The third industrial revolution," *EE Times*, Nov. 19, 2012, www.embedded.com/electronics-blogs/other/4401767/M2M--The-third-industrial-revolution?cid=Newsletter+-+Whats+New+on+Embedded.com

2. Chris Hobbs, *A Practical Approach to WBEM/CIM Management*, London: Auerbach, 2004, pp. 297-98

3. Paul Leroux, "Exactly When Do You Need an RTOS?" QNX, 2012, www.qnx.com/download/feature.html?programid=8090

4. Standards such as IEC 61508 and ISO 26262 note the importance of the architecture in safety-related systems.

Chris Ault *is a product manager at QNX Software Systems, where he focuses on the medical and general embedded markets. Prior to joining QNX, he worked in various roles, including software engineering, engineering management, product management, and technical sales, at AppZero, Ciena, Liquid Computing, Nortel, and Wind River Systems.*

Tina Jeffrey *is an automotive product marketing manager at QNX Software Systems. She has more than 17 years of experience in the semiconductor and software industry. Previously, she was responsible for technical marketing of multimedia and image processors for automotive and consumer market segments at CogniVue Corporation. She holds a BSc degree in electrical engineering from the University of Ottawa.*

Source URL: http://electronicdesign.com/communications/m2m-client-side-challenges-emerge-mobile-embedded-system-updates