Client-side Challenges of M2M-enabled Updates for Mobile Embedded Systems

Tina Jeffrey, Product Manager, Automotive (tjeffrey@qnx.com)
Chris Ault, Product Marketing Manager (cault@qnx.com)

What is M2M?

According to Jürgen Hase, a vice president at Deutsche Telekom's M2M Competence Center, M2M (machine-to-machine communication) will usher in a third industrial revolution.[1] M2M is of course not a new concept. It has been in use at least since the 1990s, when it was known by such old-fashioned names as "telematics". For perspective, it does us no harm to remember that the interaction between an automatic teller and a central banking system is M2M communication, as is, strictly speaking, the communication between an old-fashioned two-wire house thermostat and a furnace. In its more current manifestations, however, M2M is chiefly about communication between electronic devices over wireless networks—though it does not expressly exclude landline communications.

Thus, M2M refers to networks in which diverse software and hardware systems communicate data, information, decisions, and instructions. It assumes that some (and often many) of the component systems are embedded devices, and that a portion of the communication is carried over wireless technologies. Automotive telematics provides a good example of established M2M systems.

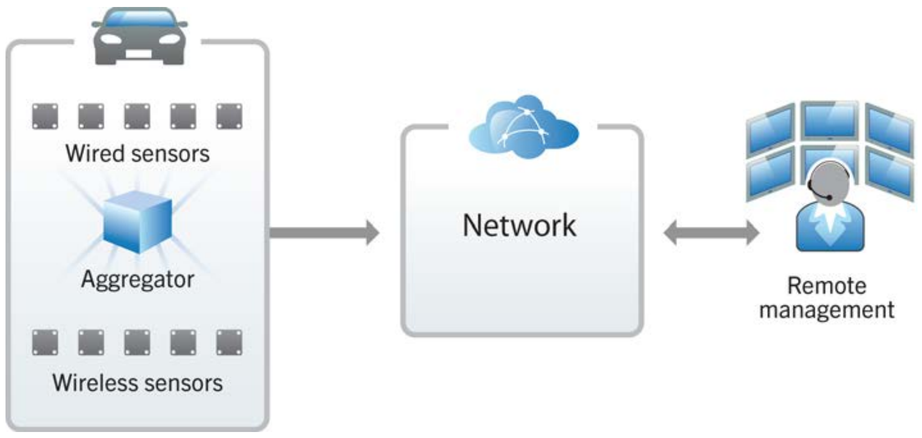

Figure 1. A simple M2M network connecting sensors to an aggregator and a decision-making system.

Figure 1 shows a very simplified view of an M2M network, where:

1. A device, such as a remote sensor, sends data to an aggregating system.
2. The aggregating system aggregates data from multiple sensors, transforms it into information, and sends it to a decision-making system.
3. The decision-making system transforms the information into knowledge and either communicates this knowledge to a person for a decision or action, or independently makes the decision or initiates an action.

In the inverse direction, an M2M network communicates decisions or required actions from a person or machine back to a remote device, which executes the required actions.

Depending on the requirements of each implementation, the M2M network may include multiple levels of aggregation and decision-making systems (such as consoles and mobile computing devices), or the aggregation and decision-making systems may be one and the same (a centralized hub). The system that ultimately makes the decision depends on the needs of the application.

FCAPS

Whatever the specific purpose of an M2M implementation, it will likely involve all or most of the so-called FCAPS functions: [2]

- Fault management—in particular, the uploading and restarting of software and firmware after a fault has been detected and corrected.
- Configuration management—auto-discovery of components and capabilities, and configurations to permanent or changing parameters, from subscriptions to services to localizations.
- Accounting management—collection, and secure storage and communication of billing and other financial information.
- Performance management—collection, storage and communication of performance data that can be used to monitor and improve performance of the system and its various components and devices.
- Security management—authentication and control of access to the system and its parts by humans and machines.

A new market

Recent interest in M2M and the recent growth in implementations can be attributed to a number of factors, including:

- the extensive and reliable mobile coverage provided by mobile networks in the populated regions of industrialized countries
- the decreasing costs of wireless connectivity and of suitable components (micros, SoCs, 2G/3G/4G modules)
- economic pressures to reduce costs and increase efficiencies in areas ranging from supply-chain management to traffic flow
- legislative requirements for devices such as smart energy meters and services such as in-vehicle emergency connectivity
- consumer demand for new products and services such as cruise control in automobiles that can adjust speeds according to actual traffic flow

Hase notes that today “more than 100 million vending machines, vehicles, containers and other devices are already connected with each other by a mobile wireless link”, and presents several estimates for the near-term worldwide growth of M2M: 360 million connections by 2016 (Berg Insight), 453 million connections by 2017 (ABI Research), and 12.5 billion connections with global sales worth €743 billion by 2020 (Machina Research).[3]

Pyramid Research divides this growing M2M market into three elements and estimates their share of projected revenues as follows: hardware modules (2%), network access and service management (33%), and systems integration and application service provisioning (65%).[4]

Important market segments for M2M development include automotive, consumer electronics, healthcare, intelligent buildings, smart grid energy, and logistics and movement of materials. In fact, M2M has a central role to play in what some have called the "Internet of Things" where objects communicate with other objects independently of people.[5]

M2M for remote updates

M2M communications is already in use for both software and firmware (FOTA, or firmware over the air) updates for a wide range of devices. M2M offers a considerable range of benefits, whose importance vary depending on the devices involved. These benefits include:

- Device longevity—this is especially important for devices which are difficult to access, very numerous, or which must last a long time. The ability to push updates out to a device such as a sensor at a remote weather station, to mobile phones, or to automobile head units can significantly extend the useful lives of these devices.
- Speed—updates can be pushed out immediately; there is no delay while devices are brought into the shop, or while service personnel go out to the field to perform the updates.
- Cost reductions—eliminating the need for human intervention to deliver and install new firmware or software can translate into significant financial savings for the manufacturer, service provider and consumer.
- Revenue streams—device vendors and service providers can offer updates or new paid applications and services, after a device has been sold; for instance, a subscriber with a mobile phone can download and run an application that did not even exist when the phone was purchased.

Figure 2 below shows a possible overall architecture for M2M-enable software and firmware updates. Note that M2M can be a viable update mechanism for all sorts of devices and systems, not just automobiles and mobile phones.

Obligatory and optional M2M-enabled updates

It can be useful to think of software and FOTA updates as belonging to two categories. Depending on the device and its implementation, M2M-enabled updates can be an obligatory or an optional strategy:

- Obligatory—satellites, spacecraft and similar devices, which cannot be accessed by a technician; and more common devices, such as mobile phones, whose large numbers make it impossible to manage updates without M2M.
- Optional—M2M offers efficiencies and savings. Automotive systems are just this sort of system, though as the number and complexity of embedded components in automobiles increases, reliance on M2M may become a necessity.

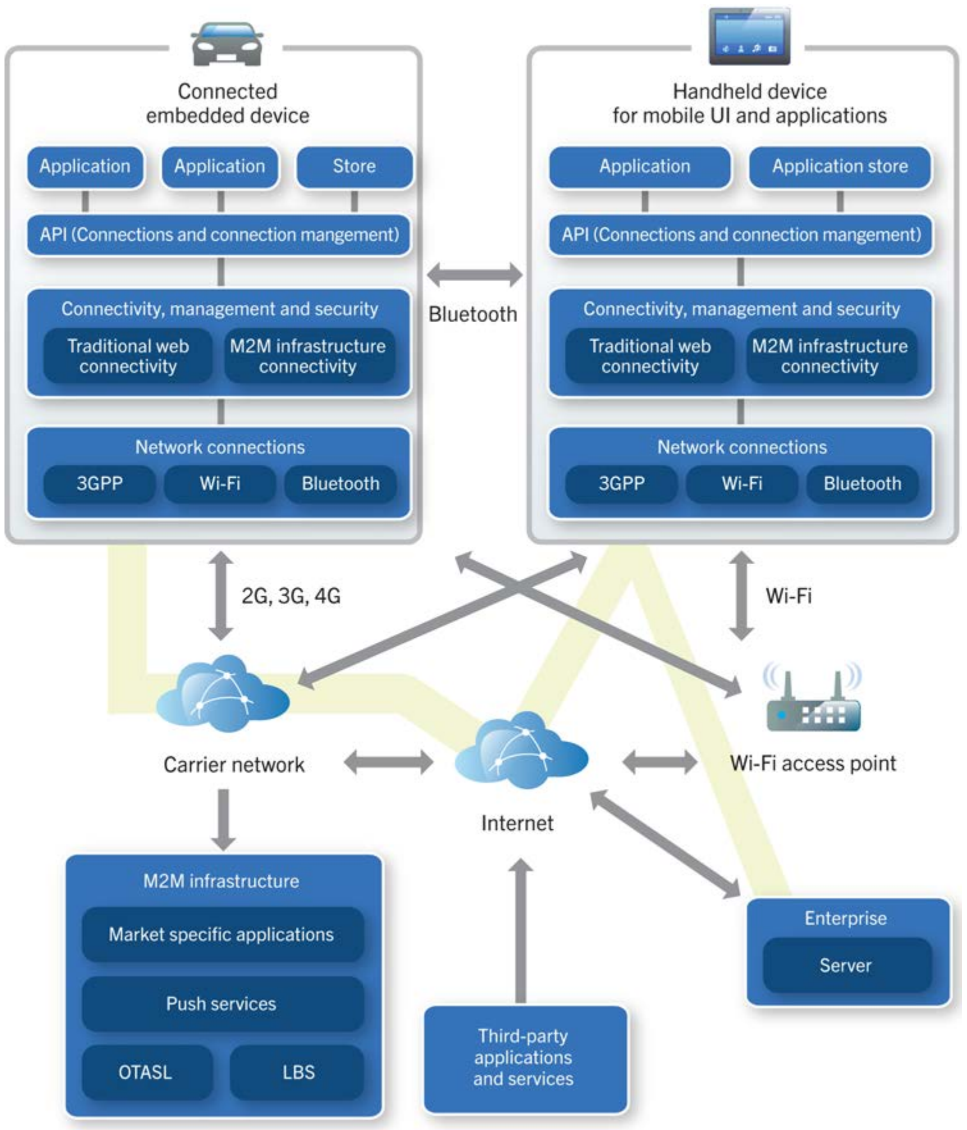
Connected embedded device
Application
Application
Store
API (Connections and connection mangement)
Connectivity, management and security
Traditional web connectivity
M2M infrastructure connectivity
Network connections
3GPP
Wi-Fi
Bluetooth
Bluetooth
Handheld device for mobile UI and applications
Application
Application store
API (Connections and connection mangement)
Connectivity, management and security
Traditional web connectivity
M2M infrastructure connectivity
Network connections
3GPP
Wi-Fi
Bluetooth
2G, 3G, 4G
Wi-Fi
Carrier network
Internet
Wi-Fi access point
M2M infrastructure
Market specific applications
Push services
OTASL
LBS
Third-party applications and services
Enterprise
Server

Figure 2. Overall architecture for M2M-enabled updates, including the vehicle and handheld device.

M2M in automotive

The recent joint resolution by the European Parliament and the Internal Market and Transport Committees calling on EU legislators to mandate that by 2015 all new vehicles be equipped with eCall capabilities is just one example of the importance of M2M in the automotive industry.[6] Implementations of M2M for automotive go far beyond emergency services like eCall, however. They can involve everything from cloud services for infotainment systems to intelligent cruise control, with vehicles and roadside infrastructure sharing data about speed, road conditions, traffic patterns, and so on.

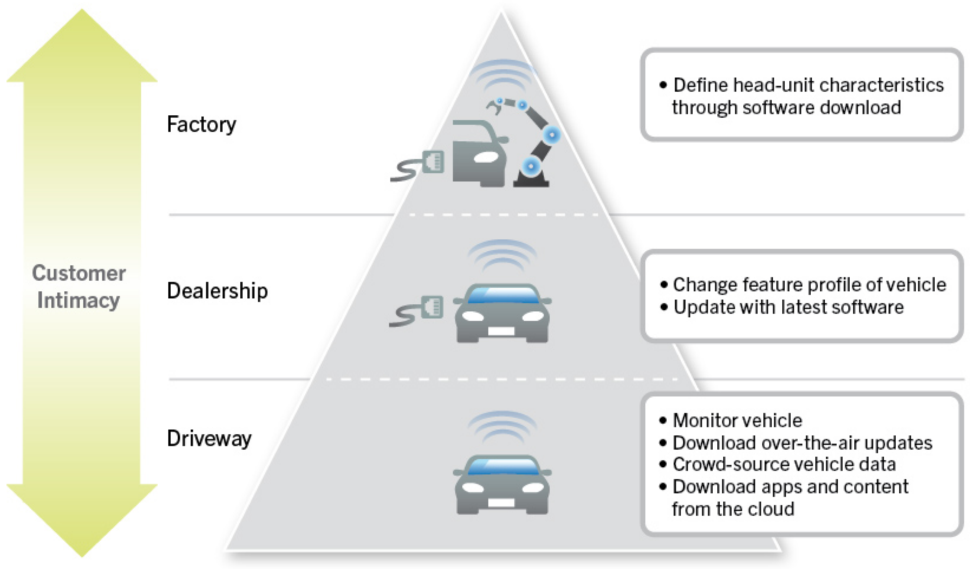
Customer Intimacy
Factory
Dealership
Driveway
• Define head-unit characteristics through software download
• Change feature profile of vehicle
• Update with latest software
• Monitor vehicle
• Download over-the-air updates
• Crowd-source vehicle data
• Download apps and content from the cloud

Figure 3. M2M-enabled updates could be used at multiple stages of the vehicle lifecycle.

The number and diversity of current and possible M2M implementations in cars make them an excellent paradigms for examining issues of software and firmware updates to mobile and embedded platforms. Automobiles present many systems in many models, trims and configurations, which must all be served by a limited number of update strategies.

Client-side challenges

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 (e.g. infotainment) and safety-related systems (e.g. ABS braking systems).
- limited computing resources—unlike networks switches, for example, automotive systems don't have the luxury of redundant processors and memory.
- connectivity—cars are truly mobile and their locations and time of use (and therefore connectivity) are unpredictable.

In short, the requirements for M2M software and firmware updates for automobiles are sufficiently diverse and demanding that a discussion of these updates is likely to be valuable, not just to the automotive sector, but also to any industry in which M2M has or may soon have a role.

The head unit and the ECUs

Modern vehicles contain a complex and 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. These components will likely require firmware updates from time to time to correct faults, keep them current, or even implement new features and capabilities.

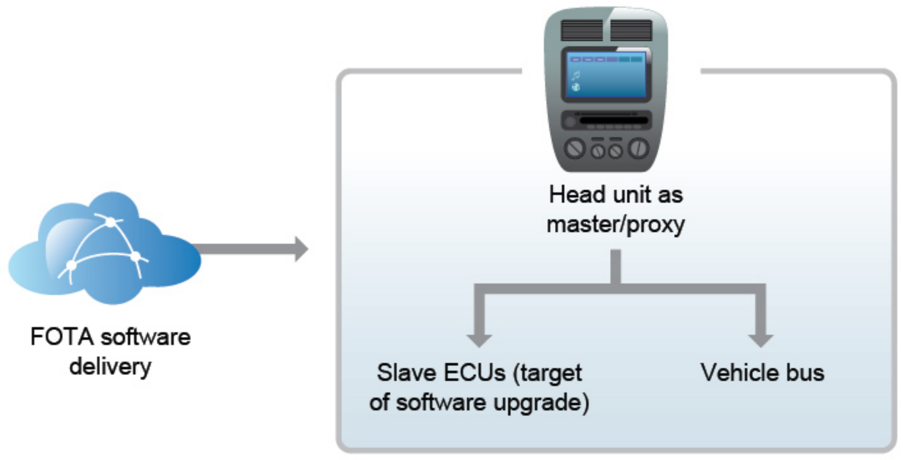
Head unit as master/proxy
FOTA software delivery
Slave ECUs (target of software upgrade)
Vehicle bus

Figure 4. The vehicle head unit used as a controller for the ECU upgrade process

Already, many infotainment applications can be updated by users accepting updates pushed through to them. This is not currently the case for ECUs, however. ECU firmware and software is updated either through a complete module replacement or through re-flashing the module at a dealership.

This labor-intensive, time-consuming and, therefore, costly model could in many cases be replaced by an M2M-enabled process managed by the vehicle head unit, which could use the vehicle identification number (VIN) and other parameters, such as user language preferences and subscriptions, to determine which updates are relevant to that vehicle. Figure 4 above shows how a head unit could supervise ECU image updates, unpack the software updates, and control the software upgrade process for the connected ECUs within the vehicle. To perform this function, the supervisor processor would require:

- remote query of the ECU module and software inventory, including information such as module type, manufacturer, serial number, manufacture date, 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 needs to be asked about any upgrade, regardless of whether it is an M2M-enabled upgrade or an upgrade by some other method, is whether the affected device or component is safety-related. This is particularly relevant for ECUs, which may control critical vehicle functions, but it is also relevant for at least some components in the head unit. If, for instance, the head unit is used to upgrade ECU firmware or software, then the head unit or some parts of it become safety-related components.

If a device or component is safety-related, then the issue becomes not just a question of performing the upgrade and demonstrating that the upgraded component meets all safety requirements, such as IEC 61508 Safety Integrity Level 3

(SIL3). Even if the device or component being upgraded is not itself safety-critical but interacts with safety-critical components in the vehicle, then the manufacturer will need to demonstrate that the upgrade maintains component isolation, as required by ISO 26262.

Add to this the possibility that drivers accustomed to bringing their cars in to a garage for service may be leery of updates to safety-related systems performed without direct human intervention—even if they are completely aware that humans may in fact be the weakest, that is, the most error-prone element in the entire update process. Finally, there is the issue of safety certifications. The upgrade process may require approval by the relevant agencies (departments of transport and the like) in all jurisdictions where it will be used.

Limited resources

In-vehicle embedded systems generally have limited capacities. These limitations 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 from design to dealer is such that automakers cannot afford to pay premium prices for their processors; a 20¢ difference in price is significant when it is multiplied by 80 processors in 10 million cars
- time—vehicles must last at least a decade, while processors and memory increase in computing power and capacity continuously; in addition, hardware must be thoroughly proven before it can go into a vehicle, which takes time, so when a car rolls of the assembly line, its processors are, unfortunately, already dated

These limitations mean that the hardware on which updates must be performed may not always have sufficient storage or memory to both keep the current version and perform the upgrade. For ECUs, the vehicle head unit may provide the excess capacity required. 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.

Connectivity

M2M-enabled updates to mobile platforms face a challenge that is for the most part irrelevant to stationary platforms and devices. The update procedure for sensors and actuators used in, say, a building management system or factory assembly line may not need to take into account a possible loss of connectivity during the procedure. For mobile devices and systems, 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 smartphones are unpredictable, but users do not generally start updates while moving; and 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 as with phones this movement is not easily predictable. Updates may be interrupted when a car enters a tunnel, goes down an urban canyon or moves outside its network area; they may also be forced to pause if the car is unexpectedly required.

Imagine, for instance, a car that has been parked for the night at a house 30 minutes from the nearest neighbor and 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.

Help from the OS

If we summarize the challenges described above as isolation, footprint and time, we can formulate an approach to these challenges as follows. We need a system that can:

a) maintain a demonstrable isolation of safety-related components from non-critical components and from each other

b) perform the updates with the limited CPU and memory resources available in the vehicle, especially in ECUs

c) complete updates rapidly enough and at such time that they never render the vehicle inoperable

d) gracefully pause and restart interrupted updates

The 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), because these are designed to support availability and reliability guarantees.[7]

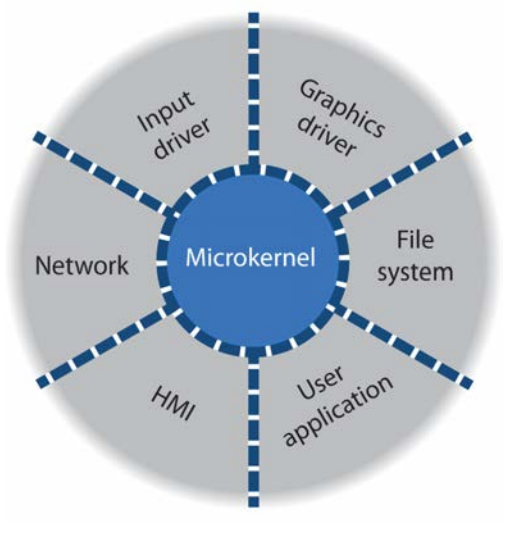

Figure 5. The microkernel OS architecture: components are isolated from each other, and a fault in one component can't percolate across the system.

A microkernel architecture[8] 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. A microkernel architecture has drivers, filesystems, 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 example, if a bug fix is available for a graphics driver, the M2M-enabled upgrade would require sufficient memory to run a duplicate graphics driver for a short period. A high-availability manager could be used to gracefully shut down the deprecated driver and start up 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, 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.

Notes

[1] Jürgen Hase , "M2M: The third industrial revolution", *EE Times*, 19 Nov. 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] *Ibid.*

[4] Pyramid Research, *Global Telecom Insider*, 4(3) March 2012, p. 4.

[5] Kevin Ashton, "That 'Internet of Things' Thing", *RFID Journal*, 22 June 2009. <www.rfidjournal.com/article/view/4986>

[6] Ludovic Privat, "eCall Mandate Pushed Forward by EU Parliament", GPS Business News, 20 June 2012. <http://www.gpsbusinessnews.com/eCall-Mandate-Pushed-Forward-by-EU-Parliament_a3705.html>

[7] See Paul Leroux, "Exactly When Do You Need an RTOS?", QNX, 2012. <www.qnx.com/download/feature.html?programid=8090>

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