Gateway processor evolution in automotive networks

Roland Lieder, Renesas Electronics Europe GmbH

Due to the upcoming requirements of multimedia applications and driver assistance systems, the volume of communication data in a car is continuously increasing. Current network architectures will soon require new topologies in order to cope with those requirements and to ensure data security.
Consequently, gateways within new topologies will change as well. This in turn impacts the functional requirements of the key component of a gateway, the gateway microcontroller and its internal system architecture. New gateway IPs with dramatically improved functionality are required.
CAN and Ethernet will come closer, and a gateway will have to handle both. Bridging CAN networks via a backbone Ethernet will also be covered.
This paper presents a view on a future gateway controller hardware architecture and its routing capabilities. The concept is compared against alternative approaches.
Interaction between a new kind of gateway IP and the associated host software provides a way to handle both CAN signal and CAN frame routing challenges.
A preliminary insight into the hardware functionality completes this contribution.

In the early days of car electronics …

…there were single wires, used as control lines between switches and the actuators, like lights and wipers. In today's cars, the amount of comfort, safety, and security functions has generated the need for network communication. Dedicated automotive interfaces like LIN, CAN and FlexRay have been invented to serve as media for digitized communication. Ethernet has been introduced to provide even higher data rates.

In the (simplified!) architecture of a today's car, all these data flows are connected in a "Central Gateway".

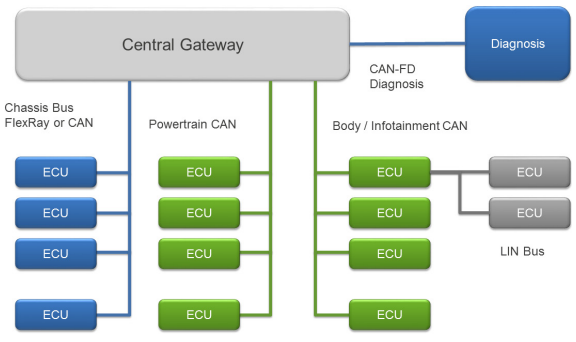

Figure 1: Car network architecture today

With increased functionality, the bus load increases and more bus systems with higher bit rates are added, so that the central gateway architecture simply comes up against its limits. At this point, a new architecture is under consideration, where the central gateway function is split into domains. These domains then are connected with each other by an Ethernet backbone.

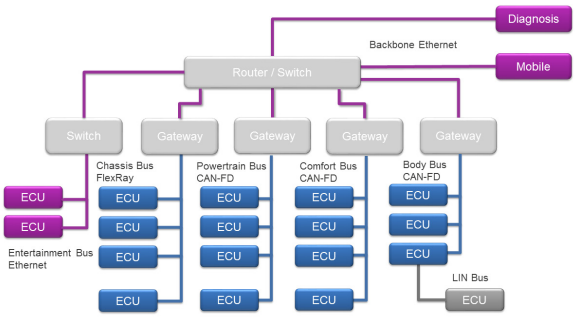

Figure 2: New car network architecture

The Ethernet backbone is formed by an Ethernet switch, which in fact maintains point-to-point connections to the domain gateways. The advantage of this architecture is that the gateway processing is split, and at the same time, the domain gateway can perform additional tasks for its local domain.

Ethernet in the car?

Initially, there were many obstacles, mostly on the physical level of Ethernet. This network type was viewed with disfavour, because the qualification of Ethernet transceivers did not fulfil automotive standards, and the existing media (cable tree) in the car was not able to serve Ethernet requirements. However, since the invention of “single-twisted-pair” Ethernet, and since there is more than one supplier for Ethernet transceivers of this media, a breakthrough was achieved. And for the qualification – this process is ongoing, and the automotive requirements can be fulfilled.

Challenges of a gateway processor

From the perspective of a semiconductor manufacturer, a new series of gateway processors and its gateway IP must fit with this new architecture. The gateway may now have various levels of complexity, depending on its position within the network architecture. It can simply be an Ethernet switch, but also a unit with several kinds of different bus systems. There are many more requirements that arise from the new architecture or are immediate consequences of it.

The new requirements can be grouped into four categories:

1. Flexibility & integration
   The gateway must have hardware routing/ switching support. Several bus protocols and adaptable bus interfaces are required. Without hardware support, the gateway processor would not be able to provide reliable latency, if it is required to handle local domain tasks in addition. The hardware feature set needs to be scalable in order to fit a device family, which can be applied to different positions within the system architecture. Security. aspects must be considered in hardware and software. This is an implicit demand of the system complexity.

2. Domain support
   On the domain level, tunnelling of data streams (like CAN over Ethernet), transport protocols (TP for software downloading) and information mirroring (for diagnosis) are required.

3. Latency
   Even though the architecture is new, there are still requirements that certain signals from a sensor must reach an actuator at a different location within the architecture within a reliable time. This concerns the total delay time, but also the variation (jitter) of the delay. For certain signals, there are fixed conditions on this, so that hardware support of priority (Time Sensitive Networking, TSN [4]) and Quality of Service (QoS, [5]) are mandatory.

4. Global time support
   The whole system should work as a common unit. This is what the “user”, who is the car driver, is expecting. In order to perform in this way, every unit in the system must be aware of a common (global) time or master clock. For Ethernet, the IEEE 1588 (PTP) [1] and 802.1AS [3] standards must be supported to allow the time synchronization of all nodes. A gateway processor must be able to perform as a time master, but also accept another time master.

Increasing network traffic at higher data rates in conjunction with the requirements mentioned above mean that the legacy architecture of a gateway processor needs to be updated. We could also talk about an alignment of the existing architecture with the new challenges. The classical architecture of a microprocessor may look like this:

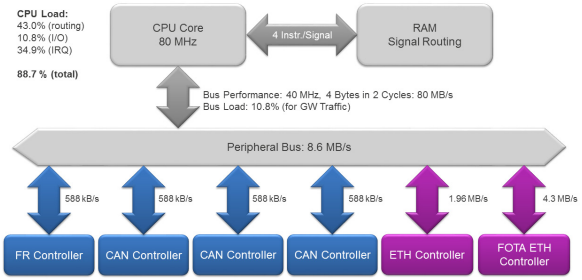

Figure 3: Legacy µC Architecture

Obviously, even though a peripheral bus system would probably be able to transfer all data of a network architecture as shown in Figure 2, the CPU system would be heavily loaded with that. Such a system tends to be unable to keep latency conditions and has, if improved with higher clock speed, much higher power consumption. At this point, a new architecture must be found, together

with new features of the communication peripherals.

The next generation architecture ...

…of gateway processors was invented with its starting point at the communication interfaces. For most of the traffic, where whole data frames are kept as they are (so called “frame routing”), the communication interfaces were enabled to process them on their own, without any support from the CPU and a long way through its software stack. To keep the system flexible, the peripherals were not purged into a huge monolithic gateway block, but are now integrated into a gateway subsystem, with its local bus, its routing engine and its local data RAM. For CPU interaction when doing “signal routing”, a dedicated interface is provided. This interface and all peripherals are now called “agents”.

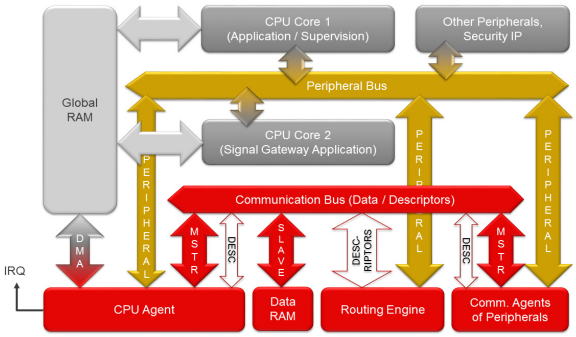

Figure 4: New gateway µC architecture

All components shown in red are part of a new “gateway IP”, which is scalable and consists of the peripheral agents, a routing engine, a dedicated, wide bus system and some attached data RAM. The classical peripheral bus system will stay; this allows the legacy usage of a peripheral, if it is deallocated from the routing engine of the gateway IP. A dual CPU core system is recommended for higher performance gateway solutions, especially if security functionality or extensive signal routing is required. The CPU cores work with data in the global RAM area when dealing with gateway routing. A DMA system with interrupt support is installed in the CPU agent, which will provide the frame and signal data from and to queues in the global RAM.
This architecture (with some restrictions) has already been created by Renesas within a prototype, in order to verify the gateway IP concept and its usability for future gateway projects in car networks.

Use cases of the gateway IP

As already mentioned, the gateway IP is scalable. It consists of sub-blocks, which can be added depending on its usage within the various gateway processor products.

For example, when looking at the car network architecture in figure 2, a gateway processor product could be used in the position of the central router/switch, which forms the backbone of the network. For this use case, the implemented gateway IP would look like this:

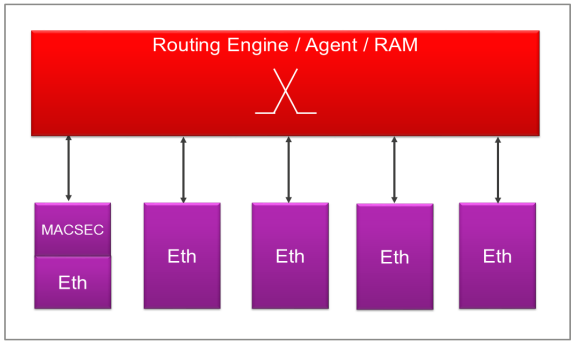

Figure 5: Gateway IP switch configuration

In this figure, CPU agent, routing engine, data RAM and communication bus were grouped in the red block to simplify the illustration. For the switch configuration, only Ethernet peripherals are added. One of them, which could be the interface to a diagnosis port, could be enhanced by a MACSEC function [3], which enables security on Ethernet MAC level.

If we are looking at the domain level, a gateway node could look like this:

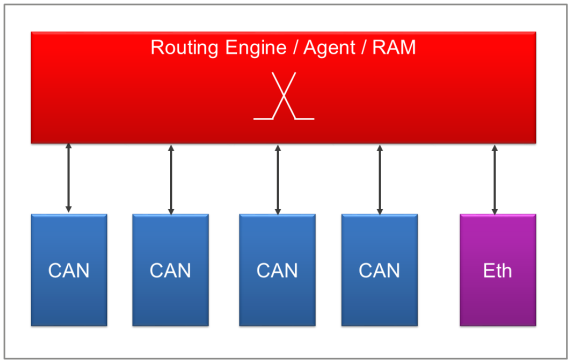

Figure 6: Gateway IP CAN domain configuration
In this type of configuration, the routing

engine would transfer CAN frames by using a tunnelling protocol through the Ethernet. For best performance, the CAN and Ethernet agents involved move the frames from and to memory by using the IEEE 1722a format [2].

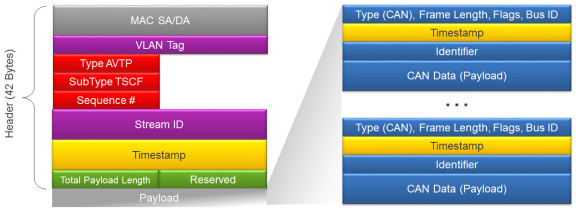

Figure 7: IEEE 1722a mapping

Using this format, several CAN frames can be joined into one Ethernet frame. As an IEEE 1722a Ethernet frame has a stream ID and a VLAN [7] tag, this kind of grouping is already part of the routing and affects the performance of the overall system. Therefore, a system architect must decide which CAN frames should go into which Ethernet stream or VLAN.

The ethernet backbone network

Ethernet streams and virtual LANs are identified by the stream ID and the VLAN tag. These are part of the Ethernet header. Whenever Ethernet frames are routed through switches, priority rules can be applied. Determined by these rules, the quality of the backbone will be seen by its latency and event jitter.
In automotive applications, these real-time conditions must be kept under control. Most use cases require well-defined reaction times and reliable overall system behaviour. This is why an Ethernet switch or router in the automotive system must support Time Sensitive Networking (TSN) and have awareness of its Quality of Service (QoS). The Renesas Gateway IP, which can form a TSN switch, supports three methods to achieve TSN and QoS requirements:

1. Audio/Video Bridging (AVB) [6]
   When using the AVB method, transmission queues of streams with dedicated IDs (see “Stream ID” above) are assigned for predefined traffic classes (A, B or C). Queues (“Qn”) with pending traffic in higher prioritized queues are serviced first. As a result, if a dedicated stream has been assigned to the highest priority, only the currently pending (active) transmission is delaying the stream. The worst case delay or jitter of an event can be the maximum length of an Ethernet message.

2. Time Aware Shaping (TAS) [8]
   In this mode, transmission time-slots “Sn” are assigned for predefined traffic classes (A, B or C). This method can be compared with TT-CAN. High jittering may occur if the transferred events are not synchronous with the time slots to which they have been assigned. However, if the overall system has the same synchronization (clock) as TAS, then this method will achieve good results. If not synchronous, the worst case delay or jitter of an event is the period of its assigned time slot.

3. AVB with Preemption [9]
   The preemption technique interrupts and splits currently transmitted frames when a high priority queue needs to transmit an event. In order to handle this, both transmitter and receivers must be able to support preemption. Apart from a minimum frame length before interruption is possible, an event can be sent almost immediately, if it has highest priority. This reduces the delay and jitter of this method to a minimum.

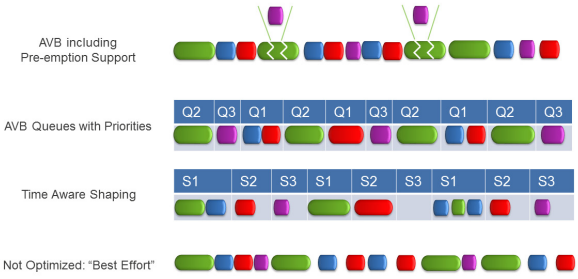

Figure 8: TSN methods on Ethernet

Hardware and software interaction

While pure hardware frame routing performance may be obvious, the interaction with software and the signal routing should be looked at in detail.
Signal routing is a kind of processing

gateway routing, where new frames need to be created from data derived from divergent sources. Such sources can be different frames from other nodes or ports, or internal gateway data, or data from an application running on the gateway's CPU. Whenever such routing is required, an efficient interface between the hardware and software of the gateway is also required.
The gateway IP in its final version will have a DMA controller, which acts as a bus master on the global RAM access, similarly to the CPU itself.
The data frames in the global RAM are organized in queues, so that prioritized processing is possible. Each frame with payload has an associated "descriptor", which contains all the information required for the routing process of the hardware.

Let's zoom into the hardware architecture at the border between the CPU agent of the gateway IP and the global RAM. This is an example configuration.

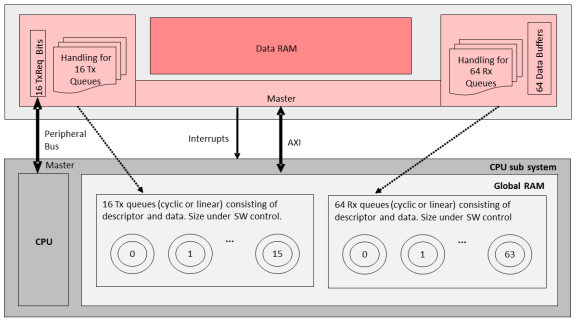

Figure 9: CPU agent interconnection

Within the global RAM, the CPU prepares the frames to be transmitted by writing into the predefined queue areas. As soon as all data is ready for the frame to be transmitted (routed signals are all inserted consistently), then the CPU triggers a transmit request via the peripheral bus to the CPU agent of the gateway IP. This is the only action that needs to be taken by the CPU. All the following is performed by the CPU agent of the gateway IP and the routing engine in it.

Once triggered, the CPU agent will copy the associated descriptor of the frame by DMA into the data RAM of the gateway IP. After having analyzed the descriptor (i.e. knowing the size of the frame), the CPU agent will then fetch the payload from the global RAM into the data RAM (again by DMA).

The queue status gets updated accordingly, and the further processing is handed over to the routing engine.

This engine is cyclically checking all configured routing sources for any updated descriptors, and then triggers the associated peripheral interfaces to handle the frames. In our case, if we want to send by Ethernet for example, the routing engine would trigger the specified Ethernet agent to fetch the frame at the specified position from the data RAM and transfer it to the Ethernet MAC. In this way, the frame gets sent out.

Similarly, a CAN agent could do the same with the frame if it was addressed. The CAN agent in addition would unpack the CAN frame(s) from the IEEE payload, before sending it by its internal transfer layer.

In the opposite direction, the reception of signals begins in the peripheral agent of the gateway IP. This agent will copy the frame data into the data RAM via the internal bus system of the gateway IP. Next, it will create a descriptor for the frame. As soon this descriptor is recognized by the routing engine, the engine will initiate the further processing. In an example case, the frame will be moved into a local data buffer, where it can be fetched by the CPU agent. As soon as the CPU agent has copied the frame data and the descriptor to the global RAM of the CPU, it will raise an interrupt. From this point, the CPU will take over the further handling and signal routing by software.

In summary, we can see that even for typical software processing tasks, the gateway IP hardware takes care of all peripheral processing. The CPU only accesses its global RAM. In this way, the speed of the processing of signal routing can be significantly increased.

Looking into an example system

The functionality of an Ethernet backbone based gateway concept can be explained by means of a simple example.

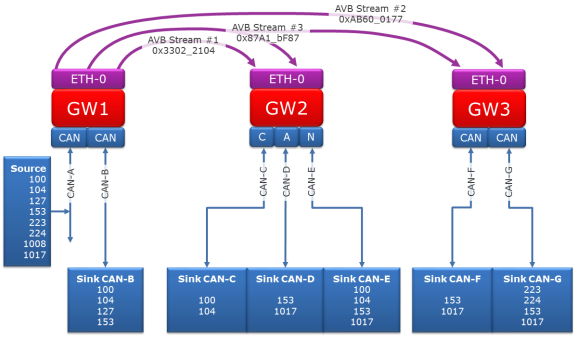

Figure 10: Example Ethernet GW System

Let's assume we have a system with three gateways, GW1 to GW3, which are interconnected by a backbone Ethernet (ETH0). This TSN based Ethernet may carry several AVB streams.

If we are looking at one source only, CAN-A, CAN frames on this bus will be routed to certain destinations, shown here as CAN-B, C, D, E, F and G. The intention is not to broadcast all frames to every bus and station, because this would have two major disadvantages. First, selective filtering would be required in every station; and second, the backbone Ethernet would be crowded unnecessarily with traffic.

Therefore, the system architect may have given the following routing rules for CAN-A:

Source CAN-A → Destinations

CAN-ID 100_H…110_H → CAN-C, E
CAN-ID 200_H…$2FF_H$ → CAN-G
CAN-ID 153_H, 1017_H → CAN-D, E, F, G
CAN-ID 100_H…$1FF_H$ → CAN-B

Along with the groups of destinations, we can set up three AVB streams with their corresponding VLAN tags and stream IDs:

Stream #1: CAN-A → CAN-C, E
Stream #2: CAN-A → CAN-G
Stream #3: CAN-A → CAN-D, E, F, G

The routing from CAN-A to CAN-B is a local route within the gateway GW1 and therefore does not require an AVB stream.
Gateway GW1 emits all three streams that comprise the corresponding CAN frames with the given IDs. Gateway GW2 will selectively receive the streams #1 and #3, so that it can forward the CAN messages with the corresponding IDs to its CAN buses C, D and E. Gateway GW3 will selectively receive the streams #2 and #3, so that it can forward the messages to its local CAN bus systems F and G, accordingly.

The principle behind this strategy is that AVB streams and associated VLANs are used to bundle data streams with the same directions. In addition, bundling is also based on priority or timing constraints. This is where the TSN based Ethernet switches will have their biggest advance. Each gateway that includes a TSN based switch can be configured accordingly, so that best performance can be achieved.

As a conclusion, Ethernet in automotive applications will have a good chance to conquer the market if the features based on TSN are implemented and effective processing in hardware supports them. Frame routing can be performed entirely by hardware, while signal and PDU based routing would require software interaction. At this point, the interface between software and hardware must become efficient, so that the gateway systems stay predictable regarding their real-time constraints.

References

[1] [1] NIST IEEE1588: http://www.nist.gov
[2] IEEE 1722a: http://www.ieee.org
[3] IEEE 802.1AE, AS: http://www.ieee.org
[4] IEEE 802.1 TSN Task Group (link see [3])
[5] IETF RFC 7567, 2212: http://www.ietf.org
[6] IEEE 802.1 http://www.ieee802.org
[7] IEEE 801.1Q (link see [3])
[8] W. Steiner, "TTEthernet Specification," TTTech Computertechnik AG, http://www.tttech.com
[9] IEEE 802.1Qbu: http://www.ieee.org

Roland Lieder
Renesas Electronics Europe GmbH
Arcadiastr. 10
DE-40472 Düsseldorf
Tel.: +49-211-6503-0
roland.lieder@renesas.com
www.renesas.com