CAN in Automotive Applications: A Look Forward

Nicolas Navet (INRIA/RealTime-at-Work), Hervé Perrault (PSA Peugeot-Citroën)

There is today more than 20 years of experience in automotive CAN applications, and CAN has certainly proven very successful as a robust, cost effective and all-around network technology. But the use of CAN in vehicles is evolving, in particular because of more complex and heterogeneous architectures with FlexRay or Ethernet networks, and because of recent needs like hybrid, electric propulsion or driver assistance that involves more stringent real-time constraints. Besides, there are other new requirements on CAN: more fine-grained ECU mode management for energy savings, multi-ECU splitted functions and huge software downloads. In parallel, safety issues request more and more mechanisms to protect against potential failures and provide end-to-end integrity. The development process is also evolving with the advent of multi-domain cooperation, Autosar, ISO2626-2 and the always shorter time-to-market requirements. In this landscape, CAN has now to be used at much higher bus load level than in the past, and there is less margin for error. What does it imply in terms of verification and validation? What are the characteristics of the communication stacks that should be paid attention to? This article is intended to shed some light and share our views on these issues.

CAN: where are we today ?

The advent of CAN

In the middle of the 80s, car makers were facing the problem of the increasing amount of wiring and connectors. This was due to data being exchanged through point-to-point links between the ECUs along with the quickly increasing need for information exchanges among electronic systems that were gradually replacing those that were purely mechanical or hydraulic. Car makers were at a turning point were electronic equipments needed to be interconnected all over the vehicle area. These issues motivated the use of multiplexed communication networks, such a VAN or CAN, for interconnecting ECUs as the engine controller, automatic gear box, junction box, body controller. Multiplexing technologies, and specifically CAN, rise up very fast, and helped to keep the wiring harness complexity under control and to satisfy the growing demand for data broadcasting.

Increased bandwidth requirements

The robustness and performance of the CAN technology, as well as the new possibilities brought by distributed software functions, have motivated engineers to use more and more bandwidth in order to improve existing Electrical and Electronic (EE) functions and introduce new ones. This trend has never decreased since then, and along with topology and functional domain constraints, has led to the use of several CAN clusters within a car, sometimes more than 4 or 5. Also, the data rates of the CAN buses are now higher (e.g., 250kbit/s for a body network when it used to be 125kbits/) and the load level has increased (e.g., greater than 50%, see §1.4).

More complex architectures

At the beginning of CAN introduction, just few ECUs were connected, and this for two main reasons: smooth technological migration and limitation of development cost. As long as just a limited number of EE functions were using CAN, with only tens or hundreds of signals, EE architectures could be designed on paper with limited tool support such as an Excel sheet for bus load evaluation and basic response time computations. Today there are thousands of signals exchanged by several tens of ECUs, with some signals having timing constraints below 5ms. Besides, the architectures are becoming complex because of gateways between the CAN buses or between a CAN bus and another networking technology (typically FlexRay). The use of several CAN clusters raises also technical issues regarding for instance fault-handling, diagnosis timing response, wake-up and sleep

synchronization. And, whatever we do, there is an overlap between the data sent on each bus, which induces a significant waste a bandwidth. To face the EE architecture complexity, and be able to push the limits of CAN, car makers have developed their own toolset as well as they have established rigorous development processes. Besides, there are now some well-suited COTS tools available on the market.

Optimizing CAN networks

When CAN was introduced, the bus loads were limited (see [15] for a typical set of messages of the years 1995-2000) and the specifications of the communication stack features, priorities and periods, etc, were defined more to handle scalability and overcome microcontroller limitations than bandwidth optimization.

Optimizing CAN networks, which includes reaching higher load levels, has now become an industrial requirement for several reasons:

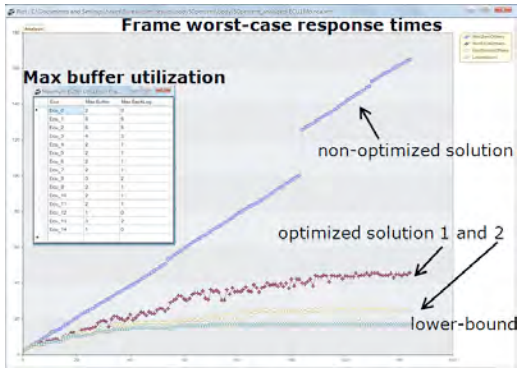
1. It helps to master the complexity of the architectures
2. It reduces the hardware costs, weight, space, consumption, etc
3. It facilitates an incremental design process,
4. It may avoid the industrial risk and the time to master new technologies,
5. It leads to better communication performances and helps to match the bandwidth needs. Sometimes, a 60%-loaded CAN network can be more efficient that two 40% CAN networks interconnected by a gateway causing delays and high jitters.

The first obvious way to optimizing a CAN is to keep the amount of data transmitted to a minimum, specifically limit the transmission frequency of the frames. This requires a rigorous identification and traceability of the temporal constraints. Given a set of signals or frames, and their associated temporal constraints (freshness, jitters, etc), they are in addition a few configuration levers than can be triggered:

1. Desynchronize the stream of frames by using offsets (see Figure 1). The reader may refer to [14] for comprehensive experiments on the large gains achieved using offsets,
2. Reassign the priorities of the frames, so that the priority order better reflects the timing constraints,
3. Re-consider the frame-packing [17] (i.e., allocation of the signals to the frames and choice of the frame periods, so as to minimize the bandwidth usage while meeting timing constraints),
4. Optimize the ECU communication stacks so as to remove all implementation choices that cause a departure from the ideal CAN behavior (see §2.3).

Configuration and verification algorithms used for 1, 2 and 3 have to guarantee the temporal behavior of the communication system, and ideally be optimal, or provide lower-bounds on their efficiency.

Figure 1: *Screenshot of NETCAR-Analyzer [1] showing maximum buffer utilization and CAN frame worst-case response times (by decreasing priority) for different offset configurations. This graph shows the typical gain one can expect with offsets.*



In our view, a bus load threshold for an “easy” CAN cluster integration is around 35-40%, and below this limit, the latencies and freshness constraints are rather easily “managed”. Overcoming this limit implies more detailed supplier specifications on the one hand, and, on the other hand, to spend more time and effort in the integration/validation phase.

Bridging the gap between models and implementations

Simulation versus analysis

Early in the development cycle, when ECUs are not available, simulation models and analytical models are the two possible verification techniques. As explained in [11], both provide complementary results and, most often, none of them alone is sufficient. On the one hand, numerous experiments (e.g., in [11,13]) suggest that simulation alone is not appropriate to find the worst case scenarios because they are too rare (see Figure 2). On the other hand, worst-case analysis cannot help to quantify how rare these events are, nor how long they last, nor what the average (or any other relevant statistics) of the response times are.

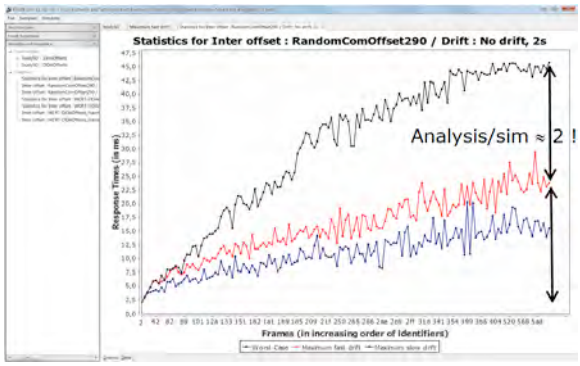
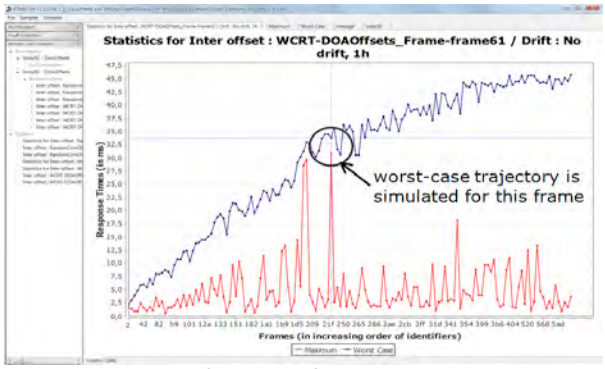


Figure 2: *Worst-case response times (by decreasing priority of the frames) obtained by analysis (black curve) versus maximum values collected during long simulation runs for two ECU clock drift values (screenshot of RTaW-Sim [2]).*

However, it is possible to derive by analysis the phasing conditions between ECUs, specific to each frame, that cause its worst case response time. Then, using a simulation tool, it becomes possible to observe for how long this situation lasts and where the ECU clock drifts lead from there. Such simulations also contribute to validate the results obtained from the analysis tool (see Figure 3), which is needed because these tools are usually commercial black boxes and, though progresses are steadily being made [3,4,9], they have to make simplifications about the hardware and the communication stack [13]. Besides, because of the complexity of the schedulability analyses, there is always the risk that the tool implementation or even the analysis itself is flawed, as it turned out to be the case with the basic CAN schedulability analysis (see [3]).

There are now COTS tools to support the verification activity, even freely available tools such as RTaW-Sim [2] for simulation and NETCAR-Analyzer [1] for schedulability analysis. For CAN, analysis consists mainly of schedulability analyses, providing upper bounds on the considered performance metrics: latencies, transmission jitters, size of the waiting queues at the ECUs and gateway levels, etc.

Figure 3: *Worst-case response times (by decreasing priority of the frames) obtained by analysis (blue curve) versus maximum values collected by simulation. The trajectory that was simulated here is the one leading to the worst-case response time for a specific frame. As the black circle shows, the worst-case response time for that frame is close to what can be obtained by simulation.*



Higher bus loads require more fine-grained models

Optimized CAN networks means higher network loads, and indeed they may now easily exceed 50% of load. But because there is less slack, there is a need for models that are more fine-grained than they were in the past. In particular, models should now account for:

- Transmission errors [15], and possibly ECU reboots,
- The use of a periodic communication task responsible for building the frame and issuing the transmission requests. In some cases, this frame may suffer delays caused by higher priority activities,
- Possible asynchronisms between the applicative level tasks that produce the signals and the communication task. Sometimes such delays can be larger than the latencies on the CAN bus,

- More fine-grained models of the hardware and communication stack (see §1.6). For instance, taking into account the ECU clock drifts may change drastically the conclusions that can be drawn from a simulation [11]. The same holds for a worst-case schedulability analysis [4,9] when explicitly modeling a FIFO waiting queue.
- Better characterization of the traffic, in particular the non-periodic part of the traffic [14] and the transmission jitters (especially for frames that are forwarded from one network to another). The non-periodic traffic is generally difficult to characterize, but if overlooked, one will tend to underestimate the frame latencies as shown in [14] which, in the worst-case, may have an impact on the safety.

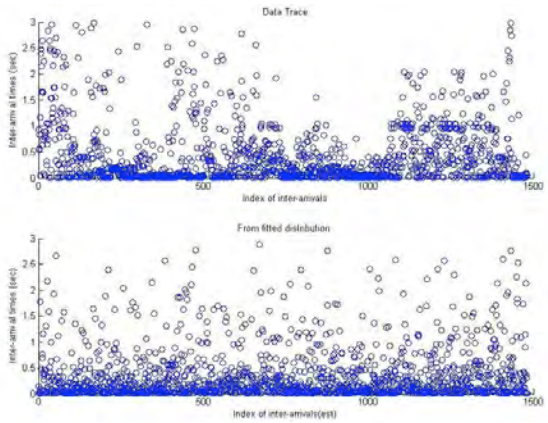


Figure 4: *On the two graphs, the X-axis shows the index of the aperiodic frames while the Y-axis is the time between two succcessive aperiodic transmissions. The upper graph is a real data trace collected while driving (only the aperiodic frames). The lower graph is an artificial data trace generated with a probabilistic model of the aperiodic frames (here Weibull interarrivals with parameters fitted with maximum-likelihood estimation using the real data trace). The probabilistic model can be used both for simulation and worst-case analysis, as done in [14].*

Departure from the ideal CAN behavior

Up to rather recently analytical models were often much simplified abstraction of reality: usually overly pessimistic (e.g. regarding the non-periodic traffic) and sometimes even optimistic, which means unsafe in our context. Indeed not all the classical assumptions made on the ideal CAN scheduling model are met by the implementations. Examples include:

- Non-abortable transmit request [7] (some communication stacks/controllers may not offer the possibility to cancel lower-priority transmission requests when a higher priority frame is released),
- Limited number of transmit buffers [5,6],
- Delays in refilling the transmit buffers [6],
- The use of a FIFO waiting queue for frames, or any other policy than the Highest Priority First (see Figure 5). The reader may refer to [4,9] for an in-depth treatment of this topic,
- Internal CAN controller message arbitration based on transmit buffer number rather than frame ID,
- Frame queuing not done in priority order (but for example by PDU index in Autosar) because of the communication stack.

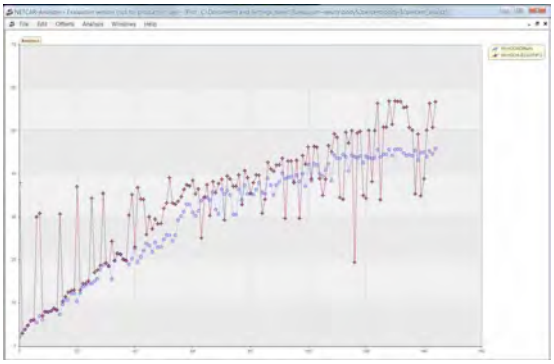


Figure 5: *Frame worst-case response times by decreasing priority on a typical body network. The blue curve shows the results when all nodes have prioritized waiting queues for the frames. The blue curve shows the actual worst-case response times when there is one station (out of 15) that possesses a FIFO waiting queue. As one can observe, in the latter case many high priority frames will suffer more delays, and potentially they may not respect their timing constraints (e.g., deadline, jitter in reception).*

Whether or not the CAN communication stacks will depart from the ideal CAN behaviour may make in practice a large difference in terms of performance and predictability. For instance, a single station with a FIFO queue can create bursts of high priority frames that will impact the latencies of the frames sent by all the other stations (see Figure 5 and experiments in [9]), possibly it may even propagate to other networks through the increased jitters of the frames that are forwarded through the gateways. In a general manner, if the integrator does not have control over the communication stacks of

all the ECUs that make up a system, he should use conservative assumptions for the validation. Fortunately, since a few years and the identification of a flaw in the original CAN schedulability analysis [8], significant progresses have been made in our view and the main issues have been identified and accounted for in the schedulability analysis (see references [3-9]).
Better adherence to the CAN priority behaviour, can be enforced by more detailed and more constraining specifications for the suppliers. Also, to some extent, the verification can be performed by tools that analyze transmission traces such as RTaW-TraceInspector.

Summary and conclusions

We can consider that when an EE architecture requires more than 3 or 4 CAN clusters, it could a better choice to introduce a new networking technology. As the most important needs for CAN bandwidth come from powertrain and chassis domains, a “natural” technology is Flexray which provides 10Mbit/s and time-triggered features. Another technology which should be considered in the future to increase bandwidth is the upcoming CAN FD from Bosch. It may provide a good trade-off between the difficulty of the migration path and additional bandwidth availability.
Nevertheless, in many cases, optimizing the standard CAN networks will help to defer the introduction of new technologies, at least for a subset of car domains. Using CAN at higher load levels requires however additional time and effort, be it for the supplier specifications or the verification. But in our view the current state of the technical literature on CAN and the COTS software tools are now mature enough to alleviate this additional work and succeed in building truly safe and optimized CAN-based communication systems.

Nicolas NAVET
INRIA / RealTime-at-Work
615, rue du Jardin Botanique
54600 Villers-les-Nancy, France
nicolas.navet@inria.fr
http://www.loria.fr/~nnavet

Hervé PERRAULT
PSA Peugeot-Citroën
DTI/DPMO/CSEO/APPT/APTI
(LG010)
18, rue des Fauvelles
92256 La Garenne-Colombes Cedex, France
herve.perrault@mpsa.com

References

[1] RealTime-at-Work, “NETCAR-Analyzer: Timing analysis and resource usage optimization for Controller Area Network,” downloadable at http://www.realtimeatwork.com/software/netcar-analyzer/, 2009.
[2] RealTime-at-Work, “RTAW-Sim: Fine-grained simulation of Controller Area Network with fault injection capabilities”, downloadable at http://www.realtimeatwork.com/software/rtaw-sim/, 2009.
[3] R.I. Davis, A. Burns, R.J. Bril, J.J. Lukkien. “Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised”. *Real-Time Systems*, Volume 35, Number 3, pp. 239-272, April 2007.
[4] R.I. Davis, S. Kollmann, V. Pollex, F. Slomka, "Controller Area Network (CAN) Schedulability Analysis with FIFO queues”. In proceedings 23rd Euromicro Conference on Real-Time Systems (ECRTS), pages 45-56, July 2011.
[5] M. Di Natale, “Evaluating message transmission times in Controller Area Networks without buffer preemption”, In *8th Brazilian Workshop on Real-Time Systems*, 2006.
[6] D.A. Khan, R.J. Bril, N. Navet, "Integrating hardware limitations in CAN schedulability analysis," IEEE International Workshop on Factory Communication Systems (WFCS) pp.207-210, 18-21 May 2010.
[7] D.A. Khan, R.I. Davis, N. Navet, “Schedulability Analysis of CAN with Non-abortable Transmission Requests”. In proceedings 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'11), Sept 5-9th, 2011.
[8] K.W. Tindell, A. Burns. “Guaranteeing message latencies on Controller Area Network (CAN)”, In *Proceedings of 1st*

International CAN Conference, pp. 1-11, September 1994.

[9] R.I. Davis, N. Navet, “Controller Area Network (CAN) Schedulability Analysis for Messages with Arbitrary Deadlines in FIFO and Work-Conserving Queues”, in submission, 2012.

[10] P. Meumeu Yomsi, D. Bertrand, N. Navet, R.I. Davis, “Controller Area Network (CAN): Response Time Analysis with Offsets“, in submission, 2012.

[11] A. Monot, N. Navet, B. Bavoux, C. Maxim, “Fine-grained Simulation in the Design of Automotive Communication Systems”, Embedded Real-Time Software and Systems (ERTS 2012), Toulouse, France, February 1-3, 2012.

[12] N. Navet, “Automotive communication systems: from dependability to security”, talk at the 1st Seminar on Vehicular Communications and Applications (VCA 2011), Luxembourg, May 2011.

[13] N. Navet, A. Monot, J. Migge, “Frame latency evaluation: when simulation and analysis alone are not enough”, 8th IEEE International Workshop on Factory Communication Systems (WFCS2010), Industry Day, May 19, 2010.

[14] D. Khan, N. Navet, B. Bavoux, J. Migge, “Aperiodic Traffic in Response Time Analyses with Adjustable Safety Level“, IEEE ETFA2009, Mallorca, Spain, September 22-26, 2009.

[15] N. Navet, Y-Q. Song, F. Simonot, “Worst-Case Deadline Failure Probability in Real-Time Applications Distributed over CAN (Controller Area Network)“, Journal of Systems Architecture, Elsevier Science, vol. 46, n°7, 2000.

[16] M. Grenier, L. Havet, and N. Navet, “Pushing the limits of CAN-Scheduling frames with offsets provides a major performance boost”, in Proc. of the 4th European Congress Embedded Real Time Software, January 2008.

[17] R. Saket, N. Navet, “Frame Packing Algorithms for Automotive Applications“, Journal of Embedded Computing, vol. 2, n° 1, pp93-102, 2006.