Traffic Shaping to Reduce Jitter in Controller Area Network (CAN)

Robert I. Davis
Real-Time Systems Research Group, Department of Computer Science,
University of York, YO10 5DD, York, UK
rob.davis@cs.york.ac.uk

Nicolas Navet
INRIA / RTaW
615, rue du Jardin Botanique
54600 Villers-lès-Nancy (France)
nicolas.navet@inria.fr

Abstract— **When a message is transferred from one CAN bus to another via a gateway, variability in the response time of the message on the source network typically translates into queuing jitter on the destination network. This jitter inheritance accumulates across each gateway and can significantly impact the schedulability of lower priority messages. In this paper, we show that the real-time performance of the network can be enhanced by a simple method of traffic shaping that eliminates this inherited queuing jitter. This method does not require access to global time, nor does it require precise time-stamping of when messages are received at the gateway or blocking read calls. It can also be extended to account for clock drifts between networks.**

Keywords-Controller Area Network (CAN); traffic shaping; jitter; response time analysis; scheduling.

I. Introduction

In automotive applications, Controller Area Network (CAN) [1], [4] is typically used to provide high speed networks (500Kbits/s) connecting chassis and power-train Electronic Control Units (ECUs), for example engine management and transmission control. It is also used for low speed networks connecting body and comfort electronics. Data required by nodes on different networks is transferred between different CAN buses by a gateway connected to both.

Schedulability analysis for CAN [3] computes upper bounds on the worst-case response times of messages on a single network, and thus can be used to provide guarantees that messages will meet their deadlines during normal operation. When messages are transferred from one network to another by a gateway, then holistic analysis [5] can be used to determine the overall end-to-end response time of each message, and thus determine if end-to-end deadlines will be met.

If the gateway forwards messages for transmission on the destination network as soon as they have been received from the source network then this can result in those messages exhibiting significant queuing jitter on the destination network. Effectively all of the variability in the message's response time on the source network can manifest itself as queuing jitter with respect to transmission of the message on the destination network. In the worst-case, multiple instances of the same message that were originally queued periodically with only a small amount of jitter on the source network, may end up being transmitted back-to-back on a destination network, causing increased delays to lower priority messages.

Holistic analysis [5] assumes that messages inherit all of the response time up to their reception at a gateway plus the maximum delay in being processed by the gateway as queuing jitter on the destination network. This analysis can be improved by considering the time at which messages are queued on the destination network as a dynamic offset [8] which can vary between some minimum and maximum values. This model eliminates jitter equivalent to the best-case response time. Techniques for reducing variability in the length of messages caused by stuff-bits can also eliminate some jitter [12], [13]; however, the difference between best-case and worst-case response times can still be large and so significant jitter remains. The use of offset release times [11] between messages sent by the same node can reduce worst-case response times on the source network, again reducing but not eliminating queuing jitter on the destination network.

The problem is that gatewayed messages with large queuing jitter can have a significant impact on the schedulability of lower priority messages on the destination network. With two networks, the interference from gatewayed messages can easily be doubled, with two instances of a gatewayed message being sent during the response time of a lower priority message, rather than just one. Response time upper bounds for non-pre-emptive scheduling highlight this effect – see equation (33) in [9].

In this short paper, we introduce a simple traffic shaping technique that can be used in gateways to eliminate all of the jitter due to variability in message response times up to the point at which they are received by the gateway. This technique builds upon the No Global Time (NGT) method [2]. Thus it does not require the use of global time, and the source and destination networks are assumed to be unsynchronised. However, unlike NGT, it does not require information about when each message was received from the source network, nor does it assume the use of a blocking read call, necessitating the use of a separate task per gatewayed message. Instead, this technique assumes only that there is a free-running timer that may be read by a single periodic communications task in the gateway. The technique proposed is related to the *leaky bucket* / *token bucket* method [6], [10], [15] of traffic shaping. In particular, it is a greedy method of traffic shaping [14] which comes for free, in the sense that it does not introduce any additional end-to-end delays. It also has some similarities to the use of servers to shape flows on Ethernet described in [17].

Other methods of reducing jitter include synchronising networks and then using offsets to determine message release times on the destination network [16]. The work on FTT-CAN [7] is an example of this approach; however, requiring synchronisation and effectively using TDMA on top of CAN's priority based arbitration has the disadvantage of adding complexity and overheads. Decoupling source and destination networks via periodic message activation on the destination network is another way of reducing jitter [19]; however, unlike the approach taken in this paper, this reduction in jitter comes at a cost of significantly increased end-to-end latencies.

Due to space constraints, we do not describe the CAN protocol or its schedulability analysis in this paper. The interested reader is referred instead to [3].

II. Gateways and Traffic Shaping

In automotive applications, there is often a communications task that is responsible for the forwarding of gatewayed messages. Let Δ be the maximum delay between an event occurring (e.g. a message instance being received) and the communications task recognising it. Typically, Δ corresponds to the period (T_{COM}) plus worst-case response time (R_{COM}) of the communications task; assuming that it can execute as early as possible in one period and then as late as possible in the next.

With a simple immediate forwarding policy, each time the communications task runs, it checks for any received messages from the source network and queues the corresponding message on the destination network. Here, as described in [5], the message *m* on the destination network inherits queuing jitter from both the message on the source network, and the communications task:

$$J_m^{DEST} = J_m^{SRC} + R_m^{SRC} + \Delta \qquad (1)$$

Where J_m^{SRC} and J_m^{DEST} are respectively the queuing jitter on the source and destination networks, and R_m^{SRC} is the worst-case response time of the message on the source network. (Note, here we assume that R_m^{SRC} is made up of the queuing delay and transmission time of message *m* on the source network, but not its queuing jitter J_m^{SRC} which is included separately). The inherited jitter J_m^{DEST} impacts negatively schedulability on the destination network. It is therefore interesting to consider ways in which this inherited jitter can be eliminated. We note that in practice not all of the response time on the source network contributes to jitter, only the variability between the worst-case and the best-case response time; however, we use the simpler model here.

A. *Jitter reduction policies*

The No Global Time (NGT) policy introduced in [2] removes jitter, without the need for global time or clock synchronisation between different nodes.

With the NGT policy (see Algorithm 1), it is assumed that the `read` is a blocking call that waits until the next instance of the message is available from the source network, if it has not already been received. The time *t* is the local time at which the message instance was received, `period` is the message period, and `next` is the earliest time at which the next message may be queued onto the destination network.

The effect of the NGT policy is to enforce a minimum delay of `period` between the queuing of instances of the message on the destination network, thus eliminating jitter due to variability in the message response time on the source network.

```
next = 0
read msg from source network
returning the time $t$ at which it was
received
loop
    queue output to destination network
    next = max(next, $t$) + period
    delay_until next
    read msg from source network
    returning $t$
end loop
```

Algorithm 1: NGT Policy for message *m*

One of the drawbacks of the NGT policy is that it requires the time at which each message instance was received to be available. This is not necessarily possible with all CAN controller hardware. A second drawback is that the `read` call is assumed to be blocking, and so requires a separate task per forwarded message.

We now adapt the NGT policy making it more suited to gateway nodes connecting networks using CAN. We refer to the new policy as NJR for Non-blocking Jitter Reduction. With the NJR policy, instead of a blocking read call, we assume a single communications task that executes periodically and has a maximum delay of Δ as explained earlier. We do not require precise recording of the times at which message instances are received from the source network, instead this will be done approximately by the communications task. We assume that instances of gatewayed message *m* are placed in a FIFO buffer by the CAN controller or interrupt handler, ready for processing by the communications task.

In the following, we assume that the event initiating the queuing of message *m* on the source network occurs sporadically with a minimum inter-arrival time of T_m referred to as the message period. (We assume that the maximum delay Δ of the communications task is less than the message period T_m).

```
if the message $m$ FIFO buffer is empty
    return
get local time $t$
if $t < X_m$ // too early to process message $m$
    return
get instance of message $m$ from buffer
queue instance on destination network
$X_m = \max(X_m, t - \Delta) + T_m$
```

Algorithm 2: NJR Policy for message *m*

Algorithm 2 illustrates the policy implemented in the body of the periodic communications task for instances of message *m*. X_m represents the next time at which it is permissible to queue an instance of message *m* on the destination network. X_m is assumed to be initialised to zero, and the local incrementing free-running timer *t* started at zero, before the task first runs. (Note that the same communications task can deal with forwarding multiple messages).

The NJR policy operates as follows. When the first instance of message m arrives, it can be forwarded immediately, and is queued on the destination network by the communications task as soon as the task executes. As the task has a maximum delay of Δ in detecting that there is a message instance to process, the policy assumes that all of this delay may have happened, and that it is therefore permissible to queue the next instance of the message on the destination network at a time $X_m = t - \Delta + T_m$ or later. If one or more subsequent instances of message m arrive before this time, then they will wait in the FIFO buffer for later processing, with queuing of the kth subsequent instance of the message now only permitted at or after time $X_m + kT_m$.

The remaining behaviour of the NJR policy, and the fact that reading and using the local time obtained on line 3 is sufficient to correctly determine the next permissible queuing time, can be seen by considering what happens when the communications task runs and *first* finds an instance j of message m at the head of the FIFO buffer. There are two cases to consider:

Case 1: The local time $t < X_m$. In this case, we can be certain that the time at which instance j was received is not relevant for setting the earliest permitted queuing time for the next instance (j+1). Instance j will be processed by a subsequent invocation of the communications task that obtains a local time t: $X_m \le t \le X_m + \Delta$. On that invocation, line 8 will set $X_m = X_m + T_m$, without needing a value of t that reflects when instance j was actually received.

Case 2: The local time $t \ge X_m$. In this case, instance j cannot have been in the FIFO buffer when the communications task previously ran; otherwise we would have Case 1. (Trivially, it cannot have been at the head of the buffer. Further, it cannot have been behind any previous instances in the buffer, because as $\Delta < T_m$, X_m advances by more than Δ for each instance of message m processed, and so any instance that is received while a previous instance is in the buffer must belong to Case 1). Instance j must therefore have been received at some time between $t - \Delta$ and t, and so t is a valid time to use in computing the earliest permissible queuing time for instance j+1, via line 8.

The effect of the NJR policy is to ensure that the period or minimum inter-arrival time of message m on the destination network is T_m and its queuing jitter is $J_m^{DEST} = \Delta + R_{COM}$. Note, the additional R_{COM} term arises because there could be a delay of at most R_{COM} between the timer being read (i.e. time t on line 3 of Algorithm 2) and the message being queued (line 7), yet the next instance of the message may be queued at time $t + T_m - \Delta$. The longest possible delay between the event triggering the sending of an instance of message m on the source network and its processing by the communications task being permitted is $J_k^{SRC} + R_k^{SRC}$ (i.e. the same as the maximum delay between the initiating event and the reception of the corresponding message instance at the gateway). This is the case because processing of an instance of message m can only ever be delayed if it arrives less than kT_m after the kth previous instance. Hence processing of message instances with the maximum delay $J_k^{SRC} + R_k^{SRC}$ is unconstrained by the NJR policy, and processing of instances which are received after a delay of $J_k^{SRC} + R_k^{SRC} - y$ is only constrained (not permitted) for at most an interval of time y. The NJR policy therefore adds nothing to the worst-case delay with which gatewayed messages are queued onto the destination network. Recall that with immediate forwarding, such messages need to be considered as having jitter of $J_m^{DEST} = J_m^{SRC} + R_m^{SRC} + \Delta$. Instead with the NJR Policy, they can, in the worst-case, be considered as having been subject to a fixed delay of $J_m^{SRC} + R_m^{SRC} - R_{COM}$, before being queued with a period of T_m and a queuing jitter of $J_m^{DEST} = \Delta + R_{COM}$. We note that while there may be little if any difference in the overall worst-case end-to-end response time for a single gatewayed message, compared to an immediate forwarding policy, there can be significant differences in the response times of lower priority messages, including other gatewayed messages. This is because the elimination of the majority of the queuing jitter reduces the number of instances of messages that can be queued onto the destination network in a short interval of time.

B. *Jitter reduction policy experiments*

We conducted some simple experiments to demonstrate the jitter reduction that occurs with the NJR policy.

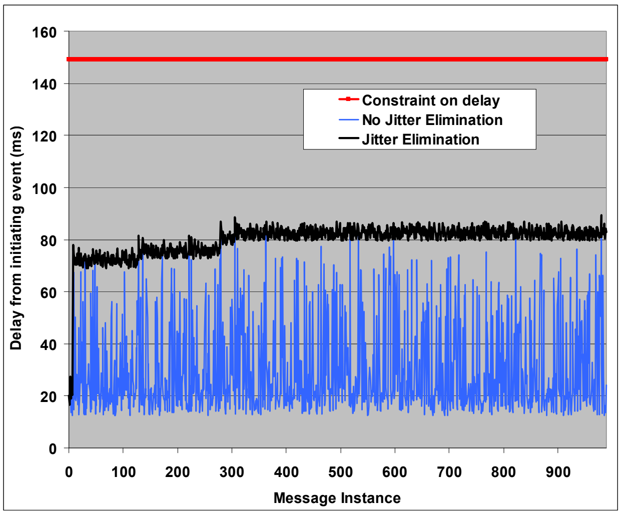

Figure 1: Overall delay in queuing messages onto the destination network

The results of one of these experiments is shown in Figure 1. This experiment is based on a typical 125Kbit/s body electronics network configuration generated by NETCARBENCH [18] with 145 messages. We selected the lowest priority message with a period of 200ms, as an example of a gatewayed message. This message has a worst-case response time of just under 140ms. The period (T_{COM}) of the communications task was set to 6ms and its worst-case response time (R_{COM}) to 3ms, giving a value for Δ of 9ms.

The bottom line in Figure 1 shows the delay from the initiating event for each message instance to the time at which it was queued on the destination network, assuming an immediate forwarding policy. The middle line shows the same delay using the NJR policy. The top line shows the bound on this time (147.5ms in this case). It is notable that the delay is far more consistent with the NJR policy.

It is noticeable in Figure 1 that there is a transient behaviour for the first 300 or so message instances, this happens as the observed delay since the initiating event increases up to a maximum value. The small overshoot evident at around 300 is due to using $t + T_m - \Delta$ as the next permitted queuing time when the communications task has not actually exhibited a delay as long as Δ.

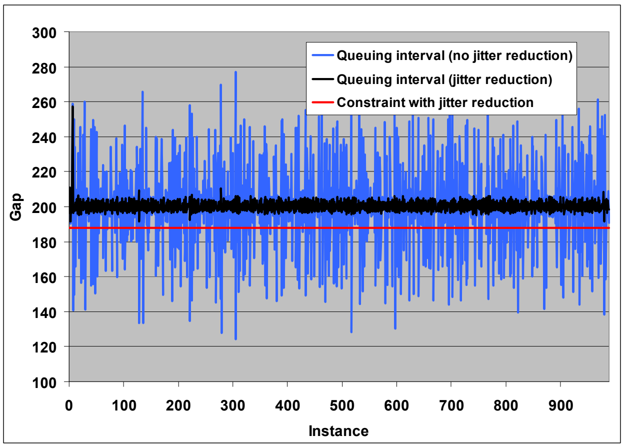

Figure 2: Time between queuing message instances (with / without jitter reduction)

Figure 2 shows the time interval between queuing two instances of the same message on the destination network with and without jitter reduction. Figure 2 illustrates how the NJR policy ensures that the queuing of $k \geq 2$ instances of message m can only take place in an interval whose length is at least $T_m - \Delta - R_{COM} + (k-2)T_m$ (i.e. respecting the period T_m and queuing jitter $J_m^{DEST} = \Delta + R_{COM}$ of the message on the destination network) This constraint is shown as a horizontal line on the graph.

III. Summary and Conclusions

The simple traffic shaping policy *Non-blocking Jitter Reduction* (NJR) introduced in this paper significantly reduces the amount of queuing jitter on gatewayed messages forwarded onto a destination network. The policy is particularly suited to CAN for the following reasons:

- It does not require the use of global time. The source and destination networks can be unsynchronised.
- It does not require the precise time-stamping of when messages are received. Instead only access to a local free-running timer is needed.
- The gatewaying of messages can be performed by a single, simple periodic task that does not block, and can therefore be implemented on a single stack operating system (e.g. OSEK BCC1).

The approach can easily be adapted to cater for clock drifts by simply assuming a slightly smaller period on the destination network, thus ensuring that the long term rate at which messages are gatewayed onto that network does not fall below the rate at which they arrive from the source network. We note that the NJR policy could also be applied to messages forwarded onto other types of network.

Acknowledgements

This work was partially funded by the UK EPSRC funded Tempo project (EP/G055548/1), the EU funded ArtistDesign Network of Excellence. The authors would like to thank Ralph Eastwood for his work on simulating the traffic shaping policy.

References

[1] Bosch. “CAN Specification version 2.0”. Robert Bosch GmbH, Postfach 30 02 40, D-70442 Stuttgart, 1991.

[2] A. Burns, Y. Chen, “Implementing Transactions in a Distributed Real-Time System without Global Time”. In proceedings Work-in-Progress session, RTSS 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] ISO 11898-1. “Road Vehicles – interchange of digital information – controller area network (CAN) for high-speed communication”, *ISO Standard-11898*, International Standards Organisation (ISO), Nov. 1993.

[5] K.W. Tindell, J.A. Clark, Holistic schedulability analysis for distributed hard real-time systems, Microprocessing and Microprogramming, Vol. 40, Issues 2-3, pp 117-134, April 1994.

[6] J.S. Turner. New directions in communications (or which way to the information age?). IEEE Communications Magazine, 24(10):8{15, October 1986.

[7] P. Pedreiras, L. Almeida, "Message routing in multi-segment FTT networks: The Isochronous Approach" In proceedings WPDRTS 2004.

[8] J.C. Palencia, M. González Harbour, “Schedulability Analysis for Tasks with Static and Dynamic Offsets”. In proceedings RTSS 1998.

[9] R.I. Davis, A. Burns. "Response Time Upper Bounds for Fixed Priority Real-Time Systems" . In proceedings RTSS 2008

[10] J. Löser and H. Härtig. Low-Latency Hard Real-Time Communication over Switched Ethernet. In proceedings ECRTS, pp 13–22, 2004.

[11] M. Grenier, L. Havet, and N. Navet. Pushing the limits of CAN - scheduling frames with offsets provides a major performance boost. In proceedings ERTS 2008.

[12] T. Nolte, H. Hansson, C. Norstrom, S. Punnekkat. Using Bit-Stuffing Distributions in CAN Analysis. In proceedings RTES 2001.

[13] T. Nolte, H. Hansson, C. Norstrom. “Minimizing CAN response-time analysis jitter by message manipulation”. In Proceedings RTAS, pp 197-206, 2002.

[14] E. Wandeler , A. Maxiaguine , L. Thiele, “Performance analysis of greedy shapers in real-time systems”, In proceedings DATE, 2006.

[15] S.-K.Kweon, K.G. Shin, “Achieving real-time communication over Ethernet with adaptive traffic smoothing”. In proceedings RTAS 2000.

[16] P. Pedreiras, L, Almeida, “Minimizing the end-to-end latency in multi-segment time triggered networks”, INCOM 2004.

[17] R. Santos, P. Pedreiras, M. Behnam, T. Nolte, L. Almeida. “Multi-level Hierarchical Scheduling in Ethernet Switches”. In proceedings EMSOFT 2011.

[18] C. Braun, L. Havet, and N. Navet, “NETCARBENCH: a benchmark for techniques and tools used in the design of automotive communication systems,” in proceedings FET 2007. Available at http://www.netcarbench.org.

[19] M. Di Natale, W. Zheng, C. Pinello, P. Giusto, A.S. Vincentelli, “Optimizing end-to-end latencies by adaptation of the activation events in distributed automotive systems”, pp. 293-302 RTAS 2007