Controller Area Network (CAN) Schedulability Analysis for Messages with Arbitrary Deadlines in FIFO and Work-Conserving Queues

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

The majority of research into schedulability analysis for CAN is based on the assumption that the highest priority message ready for transmission at each node on the network will be entered into arbitration on the bus. In practice; however, some CAN device drivers and communications stacks implement queuing policies that are not strictly priority-based, invalidating this assumption. In this paper, we introduce response time analysis for work conserving and FIFO queuing policies for messages with arbitrary deadlines.

1. Introduction

In automotive applications, Controller Area Network (CAN) [1], [12] 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.

Controller Area Network is an asynchronous multi-master serial data bus that uses Carrier Sense Multiple Access / Collision Resolution (CSMA/CR) to determine access to the bus. The CAN protocol requires that nodes wait for a bus idle period before attempting to transmit. If two or more nodes attempt to transmit messages at the same time, then the node with the message with the lowest numeric CAN Identifier will win arbitration and continue to send its message. The other nodes will cease transmitting and must wait until the bus becomes idle again before attempting to re-transmit their messages. (Full details of the CAN physical layer protocol are given in [3], with a summary in [6]). In effect CAN messages are sent according to fixed priority (FP) non-pre-emptive scheduling, with the identifier (ID) of each message acting as its priority.

In 1994, Tindell et al. [14] showed how research into fixed priority scheduling for single processor systems could be adapted and applied to the scheduling of messages on CAN. The analysis of Tindell et al. provided a method of calculating the maximum queuing delay and hence the worst-case response time of each message on the network. This seminal work lead to a large body of research into scheduling theory for CAN, and was used as the basis for commercial CAN schedulability analysis tools [5]. In 2007, Davis et al. [6] found and corrected flaws in the schedulability analysis given by Tindell et al. [14], [15].

1.1. Motivation

For the analysis derived in [6] to be applicable, it is necessary that all of the assumptions of the scheduling model are met. In particular, each CAN controller and device driver or communications stack must ensure that whenever message arbitration starts on the bus, the highest priority message queued at that node is entered into arbitration. This behaviour is essential if message transmission is to take place as if there were a single global priority queue and for the analysis to be correct. There are however many ways in which the communications stack, device driver, or CAN controller hardware can be implemented that do not match these assumptions. Issues include:

- Internal CAN controller message arbitration based on transmit buffer number rather than message ID.
- Non-abortable transmit buffers [9], [11].
- Delays in refilling a transmit buffer [10].
- The use of a FIFO queuing policy [8].

The different ECUs on a network are typically provided by a number of suppliers; with the Original Equipment Manufacturer (OEM), i.e. the car company, responsible for integration, ensuring that the messages meet their timing constraints, and that the overall system functions correctly.

One of the potential risks for OEMs composing a system from ECUs provided by a number of different suppliers is the lack of precise information about the implementation of the communications stack or device driver on each node, and how this affects the timing behaviour of messages on the network. The behaviour of different layers of the communications stack can result in an overall queuing policy for each node which is hard to quantify. Examples include:

- A priority queue that feeds messages into a set of non-abortable transmit buffers in the CAN controller. The fact that messages placed in the transmit buffers cannot be aborted means that once all of the buffers are full, then priority inversion can occur with lower priority messages transmitted ahead of higher priority ones that are waiting for a buffer to become available. This specific behaviour has been analysed in [11].
- Messages assigned to individual transmit buffers that the CAN controller enters into bus arbitration based on transmit buffer number rather than

message ID. If the transmit buffer numbers do not provide the same priority ordering as the message IDs, then the overall queuing policy will depart from being priority based. In particular, priority inversion may occur with lower priority messages transmitted ahead of higher priority ones.

In each of the cases described above, and many more that can be envisaged, the precise queuing policy becomes complex and difficult to analyse.

In this paper, we will assume that the queuing policies are nevertheless *work-conserving*. By work conserving, we mean that the communications stack on each node ensures that at any time that there are messages ready for transmission (i.e. that have been queued for transmission via an API call which has returned), and have not as yet been transmitted on the bus, then the node will always have a message ready to enter into arbitration when arbitration starts on the bus. Stated otherwise, we assume that message transmission is unaffected by *refill delays* [10].

We recognise that the system integrator may only have limited information about the queuing policy resulting from the communications stack and hardware behaviour in each node. We therefore aim to provide a broad classification of queuing policies that can be used to select appropriate forms of analysis to use in determining the schedulability of messages on the network. These are described below:

1. *Priority*: The queuing policy is work conserving and ensures a strict priority-based order to message transmission.
2. *Work conserving - re-ordering not permitted*: Any arbitrary work-conserving queuing policy that nevertheless ensures that the order in which instances of the same message are transmitted reflects the order in which they are queued. FIFO (First In First Out) is an example of such a policy.
3. *Work conserving - re-ordering permitted*: Any arbitrary work-conserving queuing policy. In this case, instances of a message that are queued later may be transmitted before instances of the same message that were queued earlier. LIFO (Last In First Out) is an example of such a policy.

We note that in practice re-ordering of message instances is highly undesirable due to the effects that it can have on the semantics of the stream of data. For example re-ordering two instances of the same message that contain a signal for 'lights on' and 'lights off' will leave the lights in a different state. Nevertheless, we provide analysis for work-conserving queues that permit re-ordering as this represents the worst possible work-conserving queuing policy in terms of timing behaviour.

Previous research by Davis et al. has provided schedulability analysis for messages with arbitrary deadlines, assuming priority queues [6], and for messages with constrained deadlines (less than or equal to their period) assuming FIFO queues [8]. In this paper, we build on this earlier work, introducing schedulability analysis applicable to messages with arbitrary deadlines, transmitted over a *heterogeneous* network containing nodes that implement priority-based, and work-conserving queuing policies. Further, we show that the more general analysis for work-conserving queues introduced in this paper reduces to that given for FIFO queues in [8], and hence that the schedulability analysis and priority assignment techniques introduced in [8] are in fact applicable to any work-conserving queuing policy, of which FIFO is just one example.

1.2. Organisation

The remainder of this paper is organised as follows: In section 2, we introduce the scheduling model, notation, and terminology used. In section 3 we recap on the analysis given in [6] for priority queued messages, and show how this can be extended to heterogeneous networks. Section 4 introduces analysis for messages in work-conserving queues. Section 5 binds the previous analysis into schedulability tests for messages on heterogeneous networks. Section 6 shows how the analysis for work-conserving queues can be simplified in the case of messages with constrained deadlines. Section 7 shows the results of a case study and experimental evaluation. Finally, section 8 concludes with a summary and recommendations.

2. System model, notation and terminology

In this section we describe the system model and notation used to analyse the worst-case response times of CAN messages. This model is adapted from [8]. Note that here we give only a high level description necessary to understand the message scheduling behaviour of CAN. Readers interested in details of the CAN protocol and its terminology are directed to section 2.1 of [6].

From an analysis perspective, a system is assumed to comprise a number of *virtual nodes* which enter messages into bus arbitration on a single network. Each virtual node is assumed to implement its own queuing policy, and to be independent of the behaviour of other virtual nodes. A number of virtual nodes may be implemented on a single physical node[1].

Virtual nodes are classified according to the queuing policy used. Thus *PQ-nodes* implement a priority-based queuing policy, whereas *WQN-nodes* implement a work-conserving queuing policy where re-ordering is not permitted, and *WQR-nodes* a work-conserving queuing policy that permits re-ordering. PQ / WQN / WQR - nodes are assumed to ensure respectively that, at any given time when bus arbitration starts, the *oldest instance of the highest priority message / the oldest instance of any arbitrary message / any arbitrary instance of any arbitrary message* queued at the node is entered into bus arbitration.

The system is assumed to contain a static set of hard real-time messages, each statically assigned to a single

[1] The concept of virtual nodes is a useful one in terms of composition. For example, a number of legacy applications each with its own queue may be implemented on a single more powerful microcontroller. Hence multiple virtual nodes may use different transmit buffers in the same physical node / CAN controller.

virtual node. Each message m has a distinct fixed Identifier (ID) and hence a unique priority. As priority uniquely identifies each message, in the remainder of the paper we will overload m to mean either message m or priority m as appropriate. We use $hp(m)$ to denote the set of messages with priorities higher than m, and $lp(m)$ to denote those with priorities lower than m. Similarly, we use $hep(m)$ to denote the set of messages with priorities higher than or equal to m, and $lep(m)$ to denote those with priorities lower than or equal to m.

Each message m has a maximum transmission time of C_m (see [6] for details of how to compute the maximum transmission time, taking into account the number of data bytes and bit-stuffing).

The event that triggers queuing of an instance of message m is assumed to occur with a minimum inter-arrival time of T_m, referred to as the message *period*. Each message m has a hard *deadline* D_m, corresponding to the maximum permitted time from occurrence of the initiating event to the end of successful transmission of the message, at which time the message data is assumed to be available on the receiving nodes that require it. The deadline of each message is arbitrary; it may be greater than, equal to, or less than its period. Each message m is assumed to be placed in a queue and available for transmission a bounded but variable amount of time between 0 and J_m after its initiating event. J_m is referred to as the *queuing jitter* of the message. The *worst-case response time* R_m of message m is defined as the maximum possible delay from the initiating event for an instance of that message, until it is received at the receiving nodes. A message is said to be *schedulable* if its worst-case response time is less than or equal to its deadline $(R_m \le D_m)$. A system is said to be schedulable if all of the messages in the system are schedulable.

The following additional notation is used to describe the properties of a set of messages that are transmitted by the same virtual node and so share a queue. The *group* $M(m)$ is the set of messages that are transmitted by the virtual node that transmits message m. The lowest priority of any message in the group is denoted by L_m.

For a message belonging to a PQ-node, the maximum time between an initiating event and the corresponding instance of that message being able to take part in priority-based arbitration is given by the queuing jitter J_m. In the case of a message m belonging to a virtual node using a work-conserving queuing policy, there can be an additional delay of up to f_m, referred to as the *buffering time*, before the message is effectively part of priority based arbitration. Hence the maximum time from the initiating event to priority-based arbitration is J_m+f_m for a message belonging to a WQN- or WQR-node. For a message belonging to a PQ-node it is J_m, and so $f_m = 0$.

As well as determining message schedulability given a particular priority ordering, we are also interested in effective priority assignment policies.

Definition 1: *Optimal priority assignment policy*: A priority assignment policy P is referred to as *optimal* with respect to a schedulability test S and a given network model, if and only if there is no set of messages that are compliant with the model that are deemed schedulable by test S using another priority assignment policy, that are not also deemed schedulable according to test S using policy P.

We note that the above definition is applicable to both sufficient schedulability tests such as those given in sections 3 and 4, as well as exact schedulability tests.

3. Analysis for Priority Queues

In this section, we recapitulate the schedulability analysis given in [6] for messages with arbitrary deadlines sent by nodes that implement priority queues. Further, we extend this analysis to take account of interference from messages sent by other nodes implementing work-conserving queuing policies, thus providing analysis for messages sent by PQ-nodes in heterogeneous networks.

Recall that a message k sent by a WQN- or WQR-node can be subject to a maximum buffering time f_k before entering priority based arbitration on the bus. Hence when considering the interference caused by such messages belonging to other virtual nodes, they can be modelled as priority queued messages with additional jitter equal to f_k. We return to the calculation of the buffering time f_k in section 5.

From [6], the worst-case response time of a priority-queued message m can be determined by examining the response time of all instances of message m that occur within a priority level-m busy period; assuming that message m and all higher priority messages are released with their maximum jitter at the start of the busy period, and then subsequently re-released as soon as possible. Further, immediately before the initial release of these messages, the longest message of lower priority than m begins transmission. B_m is the blocking factor at priority m, equivalent to the longest transmission time of any message of lower priority:

$$B_m = \max_{k \in lp(m)} (C_k) \tag{1}$$

The length of the priority level-m busy period is given by the solution to the following fixed point equation, starting with an initial value of $v_m^0 = C_m$ and ending when $v_m^{n+1} = v_m^n$.

$$v_m^{n+1} = B_m + \sum_{\substack{\forall j \in hep(m) \\ \wedge j \in M(m)}} \left\lceil \frac{v_m^n + J_j}{T_j} \right\rceil C_j + \sum_{\substack{\forall k \in hp(m) \\ \wedge k \notin M(m)}} \left\lceil \frac{v_m^n + J_k + f_k}{T_k} \right\rceil C_k \tag{2}$$

Note in (2), for ease of comparison with later equations, we have separated out the contribution from messages in $M(m)$ which are priority queued and so have $f_j = 0$. The number of instances of message m that become ready during this busy period is given by:

$$Q_m = \left\lceil \frac{v_m^{n+1} + J_m}{T_m} \right\rceil \tag{3}$$

To determine the worst-case response time of message m, it is necessary to calculate the response time

of each of the Q_m instances. The maximum of these values then gives the worst-case response time.

In the following analysis, we use the index variable q to represent an instance of message m. The first instance, released at the start of the busy period corresponds to $q = 0$. The longest time from the start of the busy period to instance q beginning successful transmission is given by the solution to the following fixed point equation:

$$w_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + f_k + \tau_{bit}}{T_k} \right\rceil C_k \quad (4)$$

Iteration starts with a value of $w_m^0(q) = B_m + qC_m$, and ends when $w_m^{n+1}(q) = w_m^n(q)$, or when $J_m + w_m^{n+1}(q) - qT_m + C_m > D_m$ in which case the message is unschedulable. The response time of instance q is given by:

$$R_m(q) = J_m + w_m(q) - qT_m + C_m \quad (5)$$

and the worst-case response time of message m by:

$$R_m = \max_{q=0..Q_m-1}(R_m(q)) \quad (6)$$

4. Analysis for Work-Conserving Queues

In this section we introduce analysis for messages with arbitrary deadlines belonging to virtual nodes implementing work-conserving queuing policies in a heterogeneous network. We first set out our basic approach, which provides a generic analysis framework. We then specialise this framework to the two different types of work-conserving queuing policy.

We are interested in determining an upper bound on the response time of message m in the group of messages $M(m)$ belonging to the same virtual node. At each stage in our analysis we will make worst-case assumptions, ensuring that the derived response time is a correct upper bound. For example, we will frame our calculation of the queuing delay by assuming that all messages in the group $M(m)$ are sent at the lowest priority L_m of any message in the group. This assumption results in response times that are upper bounds on the response time of the messages at their normal priorities. It also allows us to simplify the analysis. This abstraction means that we are effectively considering the schedulability of a network with priority queued messages only. Messages from other virtual nodes are modelled as being priority queued onto the network with additional jitter due to the buffering time f_k, while other messages in the same group $M(m)$ are assumed to be sent at the same priority L_m. Thus the worst-case scenario and analysis derives from that for priority queued messages described in section 3.

Following the analysis in section 3, we must examine the response time of each instance q of message m transmitted within a priority level-L_m busy period.

4.1. Generic analysis framework

Considering message m, the length of the priority level-L_m busy period is given by the solution to the following fixed point equation, starting with an initial value of $v_m^0 = C_m$ and ending when $v_m^{n+1} = v_m^n$. Note B_{L_m} is the blocking factor computed via (1) for priority level L_m.

$$v_m^{n+1} = B_{L_m} + \sum_{j \in M(m)} \left\lceil \frac{v_m^n + J_j}{T_j} \right\rceil C_j + \sum_{\substack{\forall k \in hp(L_m) \\ \wedge k \notin M(m)}} \left\lceil \frac{v_m^n + J_k + f_k}{T_k} \right\rceil C_k \quad (7)$$

The number of instances Q_m of message m that become ready during this busy period is then given by (3).

As per the analysis in section 3, we again use q to represent an instance of message m, starting with $q = 0$. The longest time from the start of the busy period to instance q beginning successful transmission is given by:

$$w_m^{n+1}(q) = B_{L_m} + qC_m + I_m^*(q, w_m^n) + \sum_{\forall k \in hp(L_m) \wedge k \notin M(m)} \left\lceil \frac{w_m^n + J_k + f_k + \tau_{bit}}{T_k} \right\rceil C_k \quad (8)$$

where $I_m^*(q, w_m^n)$ represents interference from messages belonging to the same virtual node as m (for example instances of other messages in $M(m)$ that are transmitted prior to instance q of message m). We will instantiate $I_m^*(q, w_m^n)$ for each queuing policy we consider. The final term on the Right Hand Side (RHS) of (8) accounts for interference from higher priority messages sent by other virtual nodes.

Iteration starts with a value of $w_m^0(q) = B_{L_m} + qC_m$, and ends when $w_m^{n+1}(q) = w_m^n(q)$, or when $J_m + w_m^{n+1}(q) - qT_m + C_m > D_m$ in which case the message is unschedulable. The response time of instance q is given by:

$$R_m^*(q) = J_m + w_m(q) - qT_m + C_m \quad (9)$$

and the worst-case response time of message m by:

$$R_m^* = \max_{q=0..Q_m-1}(R_m^*(q)) \quad (10)$$

We note that the total length of the priority level-L_m busy period, defined by (7) is the same independent of the order in which messages from $M(m)$ are transmitted, provided that the queuing policy used is work conserving. This means that the same busy period and termination condition can be used for each of the queuing policies examined in the following sections.

4.2. Analysis for work-conserving queues – re-ordering not permitted

We now instantiate the analysis introduced in section 4.1 for messages belonging to a virtual WQN-node which does not permit re-ordering of instances of the same message. In this case, we assume that all of the instances of other messages in the same group $M(m)$ that are released prior to the start of successful transmission of instance q of message m can be transmitted prior to instance q, hence:

$$I_m^{WQN}(q, w_m^n) = \sum_{j \in M(m) \wedge j \neq m} \left\lceil \frac{w_m^n + J_j + \tau_{bit}}{T_j} \right\rceil C_j \quad (11)$$

The 2nd term on the RHS of (8) represents the earlier instances of message m that are also transmitted ahead of instance q. Substituting $I_m^{WQN}(q, w_m^n)$ for $I_m^*(q, w_m^n)$ in

(8) and following the process described in section 4.1, provides analysis that can be used to compute an upper bound response time for messages sent by a WQN-node in a heterogeneous network.

We observe that if the only message in the group $M(m)$ is message *m*, then (7) and (8) reduce to (2) and (4), reflecting the fact that a work-conserving queue with only one message is also a priority queue.

4.3. Analysis for work-conserving queues – re-ordering permitted

We now instantiate the analysis introduced in section 4.1 for messages belonging to a virtual WQR-node which does permit re-ordering of instances of the same message. In this case, we assume that all of the instances of other messages in the same group $M(m)$ that are released prior to the start of successful transmission of instance q of message m can be transmitted prior to instance q. We also assume that this is true of later instances of message m which become ready before instance q is transmitted and can therefore overtake instance q, hence:

$$I_m^{WQR}(q, w_m^n) = \sum_{\substack{j \in M(m) \\ \wedge j \neq m}} \left\lceil \frac{w_m^n + J_j + \tau_{bit}}{T_j} \right\rceil C_j + \max\left(0, \left\lceil \frac{w_m^n + J_m + \tau_{bit}}{T_m} \right\rceil - (q+1)\right) C_m \quad (12)$$

Comparison of (11) and (12) suffices to show that WQN-nodes without re-ordering dominate WQR-nodes with re-ordering, in the sense that the response time of every instance q of message m belonging to a virtual WQN-node without re-ordering is no more than it would be if that node permitted re-ordering.

We note that the length v_m of the priority level-L_m busy period given by (7) can also be used to provide a simple upper bound on the worst-case response time, with: $R_m = J_m + v_m$.

5. Schedulability Tests and Priority Order

Using the approach described in section IV.C of [8] we now derive a schedulability test from the analysis given in sections 3 and 4.

The basic idea is to avoid having to consider the potentially complex interactions between the queues of different virtual nodes. This is achieved by abstracting the behaviour of messages sent by other nodes as additional jitter f_k before each message *k* becomes part of priority based arbitration. When calculating the response time of a given message, we therefore need only consider the behaviour of the virtual node that the message belongs to and the buffering times of messages sent by other nodes. An upper bound on the buffering time f_m of a message *m* belonging to a WQN- or WQR-node is given by:

$$f_m = R_m - J_m - C_m \quad (13)$$

whereas the buffering time of a message belonging to a PQ-node is always zero.

5.1. Schedulability test for arbitrary priorities

When the priorities of messages belonging to different virtual nodes implementing work-conserving queues are interleaved, this leads to a circular dependency in the response time calculations. For example, let *m* and *k* be the priorities of messages in two different groups with interleaved priorities (i.e. $k \in hp(L_m)$ and $m \in hp(L_k)$). The response time R_k of message *k*, and hence its buffering time f_k, depend on the buffering time f_m of message *m* as $m \in hp(L_k)$ and vice-versa. This problem can be solved by noting that the response times calculated via (6) and (10) are monotonically non-decreasing with respect to the buffering times, and that the buffering times given by (13) are monotonically non-decreasing with respect to the response times. Hence by using an outer loop iteration, and repeating calculations until the buffering times no longer change, we can compute correct upper bound response times and hence schedulability for all messages on the network, as shown in Algorithm 1.

```
1 repeat = true
2 initialise all f_m = 0
3 while(repeat){
4     repeat = false
5     for each priority m, highest first{
6         if (m belongs to a WQN- or WQR-node){
7             calc R_m via:
8             (7) to (10) & (11) – WQN-node,
9             (7) to (10) & (12) – WQR-node.
10            if( R_m > D_m ) {
11                return unschedulable
12            }
13            if( f_m != R_m − J_m − C_m ){
14                f_m = R_m − J_m − C_m
15                repeat = true;
16            }
17        }
18        else { // m belongs to a PQ-node
19            calc R_m via (2) to (6).
20            if( R_m > D_m ) {
21                return unschedulable
22            }
23        }
24    }
25 }
26 return schedulable
```

Algorithm 1: Schedulability Test

We note that the length of the longest busy period at the lowest priority level is the same for any work-conserving queuing policy, and therefore forms an upper bound on the worst-case response time of any message under any work-conserving queuing policy. The length of this busy period can be computed using (2), and used to limit the response times computed via Algorithm 1. (The values computed via Algorithm 1 potentially exceed this upper bound due to pessimism stemming from the fact that not all messages can achieve their maximum buffering times simultaneously).

5.2. Partial priority ordering

In this section, we consider an appropriate priority ordering for messages belonging to WQN- or WQR-nodes, given the analysis provided in the previous sections.

Definition 1: An *adjacent priority ordering* is a priority ordering whereby all of the messages belonging to the same WQN- or WQR-node are assigned adjacent priorities.

Theorem 1: If a priority ordering Q exists that is schedulable according to the schedulability analysis of Algorithm 1 then a schedulable adjacent priority ordering P also exists.

Proof: Proof follows the logic used in the proof of Theorem 1 in [8] for FIFO groups, replacing the text referring to FIFO groups / FQ-nodes / FIFO adjacent priority ordering with equivalent terms for WQN- or WQR-nodes, and adjacent priority ordering •

Theorem 1 tells us that regardless of the priority assignment applied to messages belonging to PQ-nodes, we should ensure that the messages belonging to the same WQN- or WQR-node have adjacent priorities.

5.3. Schedulability test for adjacent priorities

Following the approach used in [8], we now provide an improved schedulability test that is only valid for adjacent priority orderings. This schedulability test sets all of the buffering times (f_k) to zero. To see why this is valid, consider the interference on some message m from a higher priority message k belonging to a different WQN- or WQR-node. As message k is of higher priority than message m, then with an adjacent priority ordering so are all of the other messages in the same group (i.e. $M(k)$). Thus any message in $M(k)$ that is queued prior to the start of transmission of message m will be sent on the bus before message m, irrespective of the order in which the messages in $M(k)$ are placed in the queue. Further, immediately prior to the start of a priority level-m or -L_m busy period, there can be no messages in $M(k)$ awaiting transmission. Thus the worst-case interference from the set of messages $M(k)$ on message m is the same as it would be if the messages in $M(k)$ were in a priority queue. Hence when considering the interference from these messages on a message sent by a different virtual node, we can set $f_k = 0$.

With all of the buffering times set to zero, a single pass over the priority levels is all that is needed to determine schedulability. In other words, lines 13-16 of Algorithm 1 can be omitted when considering adjacent priority orderings. This revised schedulability test dominates the test for arbitrary priority orderings (i.e. Algorithm 1 with lines 13-16 present).

We note that the revised schedulability test is similar to that provided for FP/FIFO scheduling of flows in [13].

5.4. Optimal priority ordering

Theorem 1 tells us that messages belonging to the same work-conserving queue should be placed at adjacent priorities, thus forming a priority band.

In [8], Davis et al. showed that a variation on Audsley's Optimal Priority Ordering (OPA) Algorithm [1], [2] can be used to obtain an optimal priority ordering for priority bands each containing either a group of FIFO-queued messages with adjacent priorities, or a single priority queued message. In this section, we show that essentially the same approach can be applied in the case of messages using work-conserving queues (WQ). The pseudo code for this OPA-FP/WQ algorithm is given in Algorithm 2. Note that only one message from each WQN or WQR group is considered in the initial list, as once this message is assigned to a priority band, then so are the other messages in the same group.

```
for each priority band k, lowest first
{
    for each message msg in the initial list {
            check the schedulability of msg in priority band
            k, with all unassigned priority-queued messages
            and messages in other WQN or WQR groups
            assumed to be in higher priority bands
            if msg belongs to a WQN- or WQR-node,
            similarly check the schedulability of all other
            messages from the same group in priority band k.
            if all messages checked are schedulable
            {
                  assign them to priority band k (with adjacent
                   priorities)
                  break (continue outer loop)
            }
    }
      return unschedulable
}
return schedulable
```

Algorithm 2: Optimal Priority Assignment (OPA-FP/WQ)

In [7] Davis and Burns showed that Audsley's OPA algorithm is optimal with respect to any schedulability test that meets three specific conditions. These conditions are stated with respect to FIFO queues and priority bands in [8]. We observe that the same three conditions apply in the case of work-conserving queues and priority bands. Due to space limitations, we do not repeat the three conditions here.

Theorem 2: The OPA-FP/WQ algorithm is an optimal priority assignment algorithm with respect to the schedulability tests for messages belonging to PQ-, WQN-, and WQR-nodes in a heterogeneous network given in sections 3 and 4 with the buffering times set to zero.

Proof: It suffices to show that the three conditions (stated in [8]) hold with respect to the schedulability tests, and that the response time of message m is independent of the priority ordering of the other messages in the same priority band. Examination of the analysis equations (1) to (12), given in sections 3 and 4, shows this to be the case. *Condition 1* holds as the response time of each message m is dependent on the set of messages in higher priority bands, but not on their relative priority ordering. *Condition 2* holds as the

response time of each message *m* is dependent on the set of messages in lower priority bands via the blocking term, but not on their relative priority ordering. *Condition 3* holds as increasing the priority band of message *m* cannot result in a longer response time. •

6. Analysis for Constrained deadlines

In this section, we specialise the analysis given in section 4, simplifying it to provide sufficient analysis for messages with constrained deadlines.

For a schedulable message *m* with a constrained deadline, there can be no re-ordering of instances of the message. This is because each instance must be transmitted before the next one is queued, otherwise it would have already missed its deadline. In this section, we therefore only consider analysis for WQN-nodes.

6.1. Analysis for constrained deadlines

In the constrained deadline case, we use the term *queuing delay* y_m to denote the maximum time from an instance of message *m* being queued until the time at which it commences successful transmission.

Using the approach described in section 3.4 of [6], the analysis given in sections 4.1 and 4.2 can be reduced to the following sufficient schedulability test for messages with constrained deadlines:

$$y_m^{n+1} = \max(B_{L_m}, C_m) + \sum_{j \in M(m) \wedge j \neq m} \left\lceil \frac{y_m^n + J_j + \tau_{bit}}{T_j} \right\rceil C_j + \sum_{\forall k \in hp(L_m) \wedge k \notin M(m)} \left\lceil \frac{y_m^n + J_k + f_k + \tau_{bit}}{T_k} \right\rceil C_k \quad (14)$$

Iteration starts with a value of $y_m^0(q) = \max(B_{L_m}, C_m)$, and ends when $J_m + y_m^{n+1} + C_m > D_m$ in which case the message is unschedulable, or when $y_m^{n+1} = y_m^n$, in which case an upper bound on the worst-case response time of the message is given by:

$$R_m = J_m + y_m + C_m \quad (15)$$

The above analysis is valid for any message *m* that has a constrained deadline, and belongs to a WQN-, or WQR-node. Note other messages belonging to the same node may have arbitrary deadlines, in which case their response times need to be computed via the more general forms of analysis given in section 4.

6.2. Symmetric analysis for constrained deadlines

In [8], Davis et al. provide analysis for messages with constrained deadlines using FIFO queues. In [8], response times are measured from the time at which a message instance is queued, rather than from the initiating event. In the remainder of this section, we adopt this approach, using the response time for transmission $R_m^T = R_m - J_m$ and *transmission deadline* $E_m = D_m - J_m$ to determine schedulability. Further, we use E_m^{MIN} to denote the shortest transmission deadline of any message in the group $M(m)$, and C_m^{MAX} and C_m^{MIN} to mean the transmission times of the longest and shortest messages in the group. Finally, C_m^{SUM} is used to denote the sum of the transmission times of all of the messages in the group.

We now build up a *symmetric* schedulability test [8]. By symmetric, we mean a single schedulability test that will attribute the same upper bound queuing delay, worst-case response time for transmission, and the same schedulability test result to every message in the group $M(m)$ belonging to the same WQN-node.

To form such a symmetric test, we must make worst-case assumptions across all of the messages in $M(m)$. For example, we can assume that the blocking factor in (14) is given by $\max(B_{L_m}, C_m^{MAX})$. Further, for a message *m* to be deemed schedulable by a symmetric test, the worst-case response time for transmission must not exceed E_m^{MIN}. Thus we can adapt the iteration of (14) to start with a value of $y_m^0(q) = \max(B_{L_m}, C_m^{MAX})$, and end when $y_m^{n+1} + C_m > E_m^{MIN}$ in which case the message is deemed unschedulable, or when $y_m^{n+1} = y_m^n$, in which case the worst-case response time for transmission of the message is given by: $R_m^T = y_m + C_m$. Note that this does not as yet make (14) into a symmetric test, as we would still need to check each message *m* in $M(m)$ individually.

We observe that the ceiling function in the first summation term in (14) can only exceed 1 for some message *j* in $M(m)$ if $y_m + J_j + \tau_{bit} > T_j$. As $C_m > \tau_{bit}$, it follows that this can only occur if $R_m^T = y_m + C_m > T_j - J_j \geq D_j - J_j \geq E_m^{MIN}$. In other words, with the modified iteration described above, the ceiling function could only return a value exceeding 1 for some message *j* if message *m* had already been deemed unschedulable with the ceiling function returning 1 for every message *j*. Hence we can simplify (14), replacing the first summation term by $C_m^{SUM} - C_m$, without affecting the queuing delays or schedulability that it computes. We note that the message *m* in $M(m)$ with the smallest transmission time is then the one that gives the largest queuing delay and response time, hence we can further simplify (14) forming a single schedulability test for all of the messages in $M(m)$:

$$y_m^{n+1} = \max(B_{L_m}, C_m^{MAX}) + (C_m^{SUM} - C_m^{MIN}) + \sum_{\forall k \in hp(L_m) \wedge k \notin M(m)} \left\lceil \frac{y_m^n + J_k + f_k + \tau_{bit}}{T_k} \right\rceil C_k \quad (16)$$

Here, iteration starts with a value of $y_m^0(q) = \max(B_{L_m}, C_m^{MAX})$, and ends when $y_m^{n+1} + C_m > E_m^{MIN}$ in which case *all* of the messages in $M(m)$ are deemed unschedulable, or when $y_m^{n+1} = y_m^n$, in which case the worst-case response time for transmission of *all* of the messages in $M(m)$ is given by: $R_m^T = y_m + C_m^{MIN}$.

Equation (16) is valid for any message *m* belonging to a WQN, or WQR-node where *all* of the messages belonging to the node have constrained deadlines.

We note that (16) is precisely the analysis for FIFO queues given by Davis et al. in [8]. In this section, we have therefore shown that the FIFO-symmetric analysis given in [8] for messages with constrained deadlines, in

fact holds for arbitrary work-conserving queuing policies, not just FIFO.

7. Experimental Evaluation

In this section, we examine the performance of the schedulability tests for work-conserving queuing policies via a case study and an empirical evaluation.

7.1. Case study

The case study is based on a typical automotive network generated with NETCARBENCH [4]. There are 16 nodes exchanging a set of 80 periodic messages. The message periods are chosen from the set {20, 50, 100, 200, 1000}, and the number of data bytes in each message is between 5 and 8 bytes. The total load on the network at 500 kbit/s is 36.5%.

As is often the case in practice, there is a gateway to other networks that generates a large amount of the network load (20% or 18 messages in this case). Except for messages sent by the gateway, all message deadlines are equal to their periods and each message has a queuing jitter of 5ms. Half of the messages sent by the gateway come from another network, and have jitter equal to the message period. This jitter corresponds to the worst-case response times on the source network. The deadline for these messages is set to twice their period (i.e. their transmission deadline on the network under study is equal to their period). The blocking factor for all messages is 135 bits, the largest possible message on CAN 2.0A, since we assume the presence of non real-time traffic such as diagnostic messages.

In the following experiments, we evaluate the influence of the queuing policy together with the priority assignment policy. We evaluated 3 different configurations. In configuration #1 all of the nodes are PQ-nodes and message priorities are in Transmission Deadline Monotonic Priority Order (TDMPO) (i.e. based on the deadline minus jitter of the individual messages). In configuration #2 the priority order was again TDMPO; however, now the gateway uses a work-conserving queue (WQN-node). In this case, the priorities of messages belonging to the gateway are interleaved with those of messages belonging to other nodes, therefore the analysis was done according to Algorithm 1 with lines 13-16 present. In configuration #3, the gateway was again a WQN-node; however, this time the priority order used was TDMPO-FP/WQ where messages belonging to the same WQN-node are assigned adjacent priorities (i.e. in a priority band based on the message from that node with the shortest transmission deadline). Further, within the priority bands determined by this policy, messages belonging to the same node were assigned priorities according to their individual transmission deadlines.

Figures 1, 2 and 3 show the transmission deadlines and computed response times for transmission for each of the 80 messages in configurations #1, #2, and #3 respectively. This data is for the minimum schedulable bus speed for each configuration, which was found via binary search. The figures are best viewed online in colour. In the figures, the response times of messages that are sent by the gateway are picked out in blue. The response times represent the worst-case analytical values computed using the analysis given in this paper.

In configuration #1, the minimum schedulable bus speed was 191 kbit/s, and the corresponding maximum achievable bus utilisation 95.8%. This very high utilisation is achieved because the messages have near harmonic periods, as well as deadlines close to or greater than their periods. By contrast, in configuration #2, the interleaving of the priorities of messages belonging to the gateway with those of other messages results in significant priority inversion. As a consequence, high priority messages have long response times.

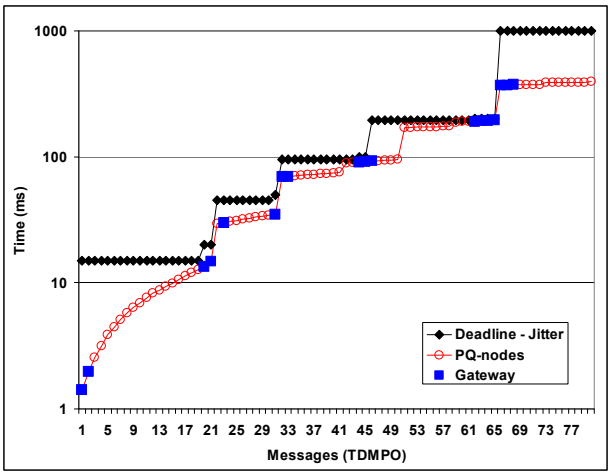

Figure 1: PQ-Nodes.

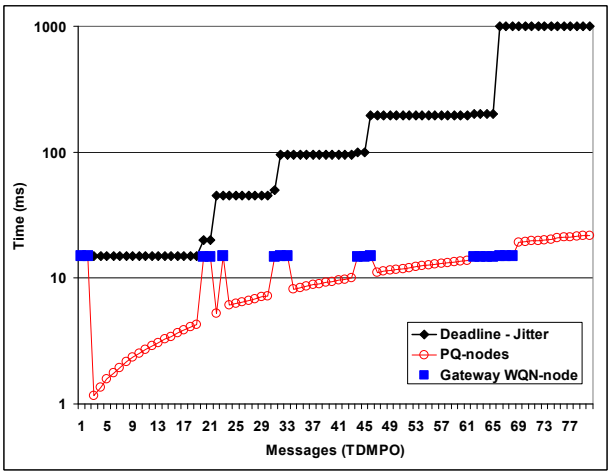

Figure 2: WQN gateway (interleaved priorities).

With configuration #2, the minimum schedulable bus speed is 624 kbit/s (more than 3 times higher than with priority queues), with a corresponding maximum achievable bus utilisation of just 29.3%. Configuration #3 shows that using an adjacent priority ordering is effective in reducing the amount of priority inversion, with a minimum schedulable bus speed of 393 kbit/s and a bus utilisation of 46.6%.

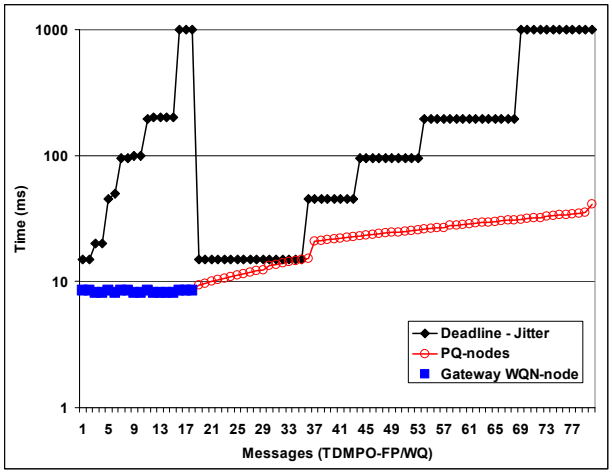

Figure 3: WQN gateway (adjacent priorities).

This case study clearly illustrates the significant detrimental impact that the use of a work-conserving queuing policy (such as FIFO) can have on the real-time performance of the network.

7.2. Empirical evaluation

In this section we further explore the effects that arbitrary work-conserving queuing policies have on the maximum achievable bus utilisation. Our experimental evaluation examined a network with 8 nodes and 80 messages connected via a single CAN bus. We considered five different configurations of this network. In configuration #1, all of the nodes used priority queues. Configurations #2, #3, and #4 increased the number of WQN- or WQR-nodes from 2, to 4 to 8 (1/4, 1/2 and all nodes respectively). Configuration #1 used TDMPO, whereas configurations #2 to #4 used TDMPO-FP/WQ priority ordering. In contrast, in configuration #5, message priorities were assigned at random, and all nodes used priority queues.

To examine the performance of these five configurations, we randomly generated 10,000 sets of messages as follows:

- The period of each message was chosen from a log-uniform distribution in the range 10-1000 ms; thus generating an equal number of messages in each time band (e.g. 10-100 ms, 100-1000 ms etc.).
- The deadline of each message was set equal to the message period.
- The jitter of each message was chosen according to a uniform distribution in the range 2.5ms to 5ms.
- Each message contained 8 data bytes.
- Each message was randomly allocated to one of the 8 nodes on the network, thus on average, each node transmitted 10 messages.
- The first node (which was a WQN- or WQR-node if such nodes were present in the configuration) was assumed to be a gateway. The deadlines and jitter of messages belonging to the gateway node were both increased by the message period. Thus deadlines for these messages equated to twice their period, with jitter representing the variation in response time on the source network.
- All messages were assumed to have 11-bit identifiers.

For each configuration, we computed the maximum bus utilisation for each message set. This was done via a binary search combined with the schedulability analysis given in section 4.

Figure 4 shows the results for WQN-nodes. The solid lines in Figure 4 illustrate the frequency distribution of the maximum bus utilisation across the 10,000 message sets for each of the five configurations. From Figure 4, it is clear that the use of arbitrary work-conserving queues rather than priority queues significantly degrades the real-time performance of the network. With all eight nodes using priority queues, the mean value of the maximum bus utilisation was 85.5%. With 2, 4, and 8 nodes using work-conserving queues this reduced to 49.9%, 38.0%, and 25.5% respectively. Worse still was random priority assignment with a mean value of just 16.4%; despite using priority queues. The results for WQR-nodes were very similar, with the mean values for maximum bus utilisation within 1% of the results for WQN-nodes.

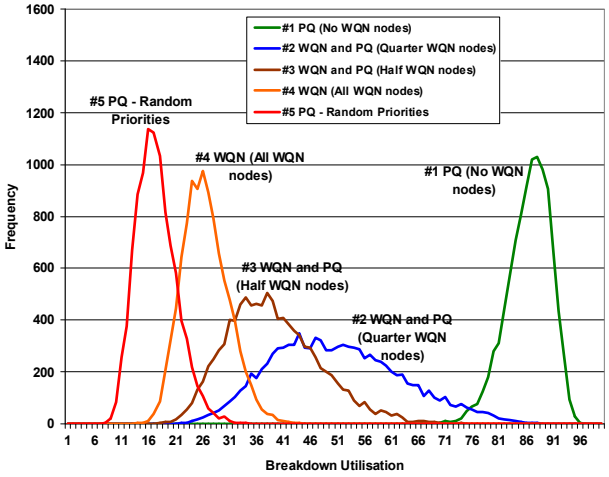

Figure 4: Freq. distribution of max. bus util.

We note that except for the modifications to messages of the gateway node, message generation was identical to that used in [8]. The results shown in Figure 4 can therefore be compared with those given in [8] for FIFO queues, in particular Figure 8 in that paper. We observe that the increase in jitter on the messages of the gateway node has a significant detrimental effect on the maximum bus utilisation that can be achieved. This is apparent in the case of priority queues where the mean value of the maximum bus utilisation reduces from 89.5% to 85.5%. However, when the gateway node is a WQN-node, then the adjacent priority assignment means that *all* of the messages belonging to this node have relatively high priorities, and due to their large queuing jitter they therefore have a significant impact on the schedulability of other messages. In this case, other messages may be subject to back-to-back interference

from two instances of *each* gatewayed message. Thus with two WQN-nodes, designating the first of them as a gateway and modifying the deadlines and jitter of its messages reduced the mean value of the maximum bus utilisation from 62.7% to 49.9%.

8. Summary and Conclusions

The major contribution of this paper is the derivation of schedulability analysis for messages with arbitrary deadlines sent on Controller Area Network, where some nodes on the network implement arbitrary work-conserving queuing policies.

We provided analysis for two classes of work-conserving queuing policy. The first class, WQN does not permit re-ordering of instances of the same message, while the second class, WQR does. FIFO is an example of a WQN policy, whereas LIFO is an example of a WQR policy. For practical reasons WQR policies are highly undesirable in automotive systems, as re-ordering instances of the same message can cause functional problems. Nevertheless, we provided analysis for WQR policies as these are the most general and have the worst timing behaviour.

Building on previous work [8], we showed that with WQN- or WQR-nodes, it is optimal to assign all of the messages belonging to the same node adjacent priorities, and that an overall, optimal priority assignment can be achieved using a modified version of Audsley's OPA algorithm.

We showed that the analysis for work-conserving queues, given in this paper, can be simplified for messages with constrained deadlines. These simplifications result in analysis that is *identical* to that given in [8] for messages in FIFO queues. Hence we showed that for messages with constrained deadlines, the analysis given in [8] is in fact applicable to any work-conserving queuing policy, not just FIFO.

Although this paper provides analysis for work-conserving queuing policies, such as FIFO and LIFO, we cannot recommend the use of such policies. They undermine the priority-based message arbitration scheme used by CAN, and significantly degrade the overall real-time performance capability of the network as shown by our case study and empirical evaluation. We therefore recommend the use of priority queues whenever possible. However, system integrators do not always have control over the queuing policies implemented in the communications stacks or devices drivers of all the ECUs that make up a system. This is where the analysis in this paper is most useful. It enables guarantees to be made on the basis of limited information about the queuing behaviour of such nodes, requiring only assurances that the policies implemented are work-conserving.

Acknowledgements

This work was partially funded by the UK EPSRC funded Tempo project (EP/G055548/1), the EU funded ArtistDesign Network of Excellence.

References

[1] N.C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", Technical Report YCS 164, Dept. Computer Science, University of York, UK, Dec. 1991.

[2] N.C. Audsley, "On priority assignment in fixed priority scheduling", Information Processing Letters, 79(1): 39-44, May 2001.

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

[4] C. Braun, L. Havet, and N. Navet, "NETCARBENCH: a benchmark for techniques and tools used in the design of automotive communication systems," in 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems, pp. 321–328, 2007. Available at http://www.netcarbench.org.

[5] L. Casparsson, A. Rajnak, K. Tindell, P. Malmberg. "Volcano - a revolution in on-board communications". *Volvo Technology Report*, 1998/1.

[6] 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.

[7] R.I. Davis, A. Burns, "Improved Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-Time Systems". Real-Time Systems, Volume 47, Issue 1, pages 1-40, 2010.

[8] 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.

[9] M. Di Natale, "Evaluating message transmission times in Controller Area Networks without buffer preemption", In *8th Brazilian Workshop on Real-Time Systems*, 2006.

[10] 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. doi: 10.1109/WFCS.2010.5548604.

[11] 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.

[12] 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.

[13] S. Martin, P. Minet, L. George, "Non pre-emptive Fixed Priority scheduling with FIFO arbitration: uniprocessor and distributed cases", Technical Report No. 5051, INRIA Rocquencourt, Dec. 2007.

[14] 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.

[15] K.W. Tindell, A. Burns, A. J. Wellings. "Calculating Controller Area Network (CAN) message response times". *Control Engineering Practice*, 3(8): 1163-1169, August 1995.