SCHEDULING PERFORMANCE IN DISTRIBUTED REAL-TIME CONTROL SYSTEM.

M. Dani Baba and E. T. Powner
School of Engineering
University of Sussex
Brighton BN1 9QT, England.

ABSTRACT

Distributed real-time computer control system consists of several nodes being interconnected by communication network. Each node can execute one or more application tasks and the data generated can be transferred as messages to other nodes. For real-time application, the communication system must be able to provide predictable timing response for periodic messages that coordinate the operation of a number of control loops on different nodes. To fulfill the timing constraints, appropriate communication protocol and scheduling strategy have to be employed. This paper presents the performance evaluation of message response time using rate monotonic based scheduling on Controller Area Network (CAN) for distributed real-time control system. The SAE automotive control system benchmark is being used to evaluate the effect of faults and error handling on message response time.

1. Introduction

Real-time computer control systems have been used widely in applications such as in automated factories, robotic manipulators, and automotive systems. As the process to be controlled is usually locally distributed, several computer control nodes are interconnected via suitable communication network to coordinate their effort to achieve the desired goal.

Controller Area Network (CAN) is an advanced serial communications protocol which meet the above demand. It can efficiently supports distributed real-time computer control system with very high level of data integrity [1]. In distributed real-time control systems, communication networks play a critical roles. Data transmission must be in real-time. Late delivery of data is a fault that can results in severe consequence. To reduce message transmission delay and to achieve high communication channel utilization, CAN employs message priority scheme. Each message in the control system will be assigned a unique priority. When more than one messages are being transmitted simultaneously, messages with the higher priorities will be transmitted earlier than messages having lower priorities.

The transmission of data is greatly influenced by the intensity and distribution of messages in the network. Data transmission is also subjected to time-varying delays due to the latency of messages. Besides, messages may also be corrupted by noise in the communication link or lost due to buffer saturation at the receiving nodes, and therefore have to be retransmitted. These problems increase the message delivery delay further and impose greater difficulty for the real-time control system to satisfy its timing constraint.

To guarantee correct message delivery in real-time control system, the worst case delay of each message must be known. This delay must not exceed the allowed maximum delay or the deadline for each message in that system. If the control systems are designed using CAN based networks, the designer has to assign each message a unique priority. Different priority assignments will results in different message delays. To satisfy the timing constraint of all messages in real-time control system, a formal delay analysis and an optimum message priority assignment technique is essential.
In this paper we evaluate the performance of CAN message response time under the rate monotonic based scheduling strategy in normal and transient overload conditions. This paper is organised as follows: In Section 2, the CAN message delay analysis is described. Section 3

presents the message priority assignment. Section 4 presents the SAE benchmark workload model. Section 5 presents the simulation results. The paper concludes with Section 6.

2. Worst-Case Response Time Analysis

The worst-case response time of a message occurs when this message is generated with all other higher priority messages at the same time. And at that moment a long message of lower priority has just gained access to the bus, and while waiting to access the bus, other higher priority messages continue to arrive. The worse-case response time analysis can be derive almost directly from the theory of task scheduling by fixed priority in a single processor. In real-time communication, the link replaces the single processor as the central resource, while messages are the tasks requiring this resource. From the work of Audsley *et al* [2] and Tindell *et al* [3], the worst-case response time of a hard real-time message m are reproduced in the following equations.

$$R_m = J_m + w_m + C_m$$

Where R_m is the worst-case response time of a given message m and defined as the longest time delay between the start of a task queuing message m to the latest time that message arrives at receiving nodes. J_m is the queuing jitter of message m. While w_m is the worst-case queuing delay of message m due to both higher priority messages pre-empting message m, and a lower priority message that has already acquired the bus. C_m is the longest time taken to transmit message m. The worst-case queuing delay is given by:

$$w_m = B_m + \sum_{\forall j \in hp(m)} \left[\frac{w_m + J_j + \tau_{bit}}{T_j} \right] C_j$$

B_m is the worse-case blocking time of message m and this is equal to the time taken to transmit the longest lower priority message, and is define by:

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

Where *lp(m)* is the set of lower priority messages than message m in the system. The longest time to transmit message m for CAN based network is given by:

$$C_m = \left\{ \left(\frac{34 + 8s_m}{5} \right) + 47 + 8s_m \right\} \tau_{bit}$$

The term s_m is the number of bytes in the message and τ_{bit} is the bit time of the bus. This time delay includes the 47 bit overhead per message, and 34 bits of the overhead plus the message content are subject to bit stuffing. The stuff bit is 5 bit wide. By forming recurrence relation the term w_m can be rewritten as:

$$w_m^{n+1} = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{w_m^n + J_j + \tau_{bit}}{T_j} \right\rceil C_j$$

A value of zero can be used for w_m^0 and stop the iteration when convergence occurs (i.e. $w_m^{n+1} = w_m^n$).

3. Assigning Message Priority

The message delay analysis from the previous section does not indicate how the priority is assigned to the message. Since access to the CAN bus is controlled by the priority of competing messages, then it is essential to assign the priority which can result the maximum possible schedulable utilisation. A natural choice for ordering CAN message priority is the static priority scheduling. From the theory of tasks scheduling on single processor, the rate monotonic (RM) is the most popular static priority scheduling algorithm for periodic task [4]. In this scheme, messages with shorter period will be assigned with higher priority than those with longer period. The important assumption made in RM is message period equals its deadline, and this is not always the case. It is still possible to use RM scheduling if the assumption is relaxed but with some modification. If the deadline of a message is shorter than its period it is tempting to artificially raise the priority. A better approach is through period transformation, where the period can be shorten or lengthen [5]. Another alternative is to use a close relative of RM, the deadline monotonic (DM) algorithmn [6]. With DM, messages with tigher deadlines are assigned higher priorities. The other scheduling strategy which we also evaluate is the modified shortest job first (SJF*). In this method, higher priorities are assigned to messages which are transmitted less frequently and have shorter message length. Then adjustments are made to ensure priorities of all messages satisfy their deadline. The RM, DM and the SJF* are the static priority scheduling methods which can readily be used to ordering message priority in CAN.

Messages in real-time control system are generally generated by periodic and sporadic tasks. Hence it is necessary for the message scheduling policy used in CAN based control system to support both periodic and sporadic messages. In practice, there is message release jitter, and the worst-case message delivery time is much larger than the average delivery time. To ensure no communication channel saturation occurs can lead to very low utilisation. To achieve reasonably high bus utilisation, the scheduling scheme must tolerate transient overloads. The scheduling algorithmn is said to be stable if it can guarantee the deadlines of a set of critical messages even if the channel is overloaded, and as long as this set of critical messages are schedulable by this algorithm under worst case conditions. Unfortunately, when a set of messages are scheduled by RM, some deadlines will be missed should the channel experience transient overload. Usually channel overload occurs when the physical communication system having some difficulty such as faulty link or extreme noise interference. We will evaluate how RM, DM, and SJF* scheduling strategies perform in a CAN based real-time control system under normal and overload condition. A solution to the scheduling stability problem will also be considered.

4. Workload Model

The workload model used in our simulation is the SAE Class C benchmark which is typically associated with real-time control system. The benchmark specify the communication requirements in a distributed automotive control system. The structure of the system as shown in Figure 1 consists of seven modules: the vehicle controller (V/C), the inverter/motor controller (I/M C), the instrument panel display (Ins), the transmission control (Trans), the battery (Battery), brakes (Brakes), and the driver inputs (Driver).

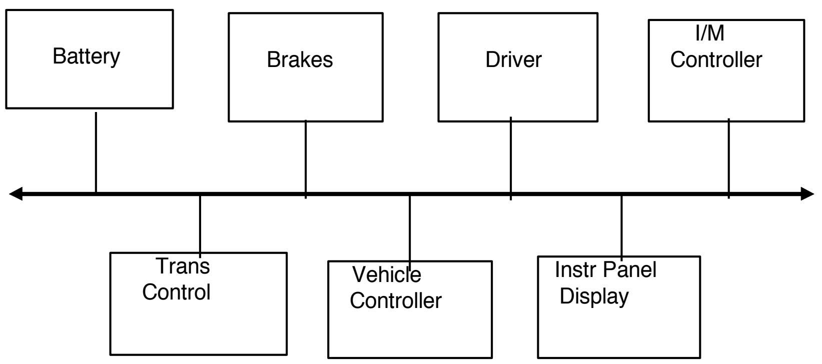

Figure 1: Structure of Automotive Control System

The seven automotive control nodes are connected to a real-time communication channel which handles 53 types of messages. These messages are listed in the Appendix. Some of these messages are sporadic in nature while others are periodically control data. Since the benchmark does not specify the minimum interarrival time for some sporadic messages, then some sensible value is assumed. From the work of Tindell *et al* [3], the 53 types benchmark signals are shown to be unschedulable using the DM scheme at 125Kbits/s bus speed. In fact only 7 types of messages satisfy their timing constraints as computed using the worse case timing response analysis. However at higher bus speed, DM has no difficulty in scheduling all the benchmark signal. To overcome the scheduling problem, they have employed message piggybacking technique to reduce the bus utilisation. This is implemented in the form of message server which polls to collect several messages from the same source and then send out as a single long message. The server chosen has 10ms period and a latency of 10ms. The newly transformed benchmark signals now consist of only 17 message types as shown in Table: 1. We have adopted this newly transformed benchmark signals as the workload model in our simulation.

Message Number	Signals Number	Size /bytes	J /ms	T /ms	D /ms	Periodic /Sporadic
1	**14**	1	0.1	50.0	5.0	**S**
2	8,9	2	0.1	5.0	5.0	P
3	7	1	0.1	5.0	5.0	P
4	43,49	2	0.1	5.0	5.0	P
5	11	1	0.1	5.0	5.0	P
6	29,30,32,42	4	0.1	5.0	5.0	P
7	**31,34,35,37,38,39 40,44,46,48,53,**	4	0.2	10.0	10.0	**S**
8	**23,24,25,28**	1	0.2	10.0	10.0	**S**
9	**15,16,17,19,20,22 26,27**	2	0.2	10.0	10.0	**S**
10	**41,45,47,50,51,52**	2	0.2	10.0	10.0	**S**
11	**18**	1	0.2	50.0	20.0	**S**
12	1,2,4,6	4	0.3	100.0	100.0	P
13	12	1	0.3	100.0	100.0	P
14	10	1	0.2	100.0	100.0	P
15	3,5,13	3	0.4	1000.0	1000.0	P
16	21	1	0.3	1000.0	1000.0	P
17	33,36	1	0.3	1000.0	1000.0	P

Table 1: Transformed Benchmark Signals

5. Simulation Results

The priority of the 17 types of newly transformed benchmark messages are assigned according to the 3 different scheduling schemes: RM, DM and SJF*. For convenience we use message number to represent the benchmark signals and priority number 1 as the highest priority. The bus speed used in all of our simulation is 125Kbit/s (or as specified) as higher speed does not pose much scheduling difficulty. Since in CAN system, once a message starts transmission it will run to completion even if higher priority messages are released during this time, hence we use the non-preemptive version of the above mentioned scheduling policy. The worst case response time of the benchmark signals using the 3 schemes were evaluated through simulation under normal and transient overload conditions. To ensure worse case scenario exist in the simulation environment, all the 17 types of messages are initially generated simultaneously. This create the situation where all messages are available and attempt to access the bus at the same time. At this critical instant the delivery of messages will have the largest response time. The criteria of merits evaluated in our scheduling performance simulation includes: level of schedulability, average delay of all schedulable messages and stability under transient overload.

5.1 Message Delivery Time Under Normal Load

Table 2 shows the simulation results of message delivery time for the 17 types of benchmark signals. Under normal load or no transmission error condition, message priority assigned according to DM and SJF* scheduling policies are schedulable. While direct message priority ordering according to RM scheme is unschedulable. We use a simple approach to address the unschedulable problem for the RM scheme by artificially raising the priority of the critical message that miss its deadline. The bus utilisation for DM, modified RM (RM*) and SJF* are the same, that is 76.33%. However the SJF* scheduling scheme is slightly better as having lower average delivery time. This is deduce as the modified SJF has shorter access delay time of 1.913 ms as compared to 1.960 ms experienced by the other two scheduling schemes.

Msg No:	Size/ bytes	J/ ms	T/ ms	D/ ms	Priority DM	Priority SJF*	Priority RM	Priority RM*	R DM	R SJF*	R RM	R RM*
1	1	0.1	50	5	1	1	10	8	0.456	0.456	5.136*	4.16
2	2	0.1	5	5	2	4	1	1	0.976	1.888	0.976	0.976
3	1	0.1	5	5	3	2	2	2	1.432	0.976	1.432	1.432
4	2	0.1	5	5	4	5	3	3	1.952	2.408	1.952	1.952
5	1	0.1	5	5	5	3	4	4	2.408	1.432	2.408	2.408
6	4	0.1	5	5	6	6	5	5	3.056	3.056	3.056	3.056
7	4	0.2	10	10	7	10	6	6	3.704	5.136	3.704	3.704
8	1	0.2	10	10	8	7	7	7	4.16	3.512	4.04	4.04
9	2	0.2	10	10	9	8	8	9	4.68	4.032	4.56	4.68
10	2	0.2	10	10	10	9	9	10	5.136	4.488	5.016	5.136
11	1	0.2	50	20	11	11	11	11	8.192	8.192	8.192	8.192
12	4	0.3	100	100	12	14	12	12	8.84	9.752	8.84	8.84
13	1	0.3	100	100	13	12	13	13	9.296	8.648	9.296	9.296
14	1	0.2	100	100	14	13	14	14	9.752	9.104	9.752	9.752
15	3	0.4	1000	1000	15	17	15	15	10.336	18.528	10.336	10.336
16	1	0.3	1000	1000	16	15	16	16	18.072	10.208	18.072	18.072
17	1	0.3	1000	1000	17	16	17	17	18.528	15.344	18.528	18.528

Table 2: Message Delivery Time Under Normal Load

Scheduler	DM	SJF	RM	RM*
Bus Utilisation	76.33 %	76.33 %	76.33 %	76.33 %
Avg Access Delay	1.960 ms	1.913 ms	1.960 ms	1.960 ms

Table 3: Bus Utilisation And Average Access Delay

5.2 Message Delivery Time Under Transient Overload

To evaluate how the DM, SJF* and RM* scheduling policies performance under transient overload we introduce noise into the communication channel. When noise present in the communication channel, some messages are corrupted or lost, and have to be retransmitted. This message retransmission will cause the bus utilisation to increase. With 10% transmission error probability, the bus utilisation approaches 80% and all the scheduling schemes evaluated: DM, SJF* or the RM* become unstable at 125Kbit/s bus speed. Table 4 shows the worst case delivery time for the benchmark messages when the communication channel experiences transient overload. All the scheduling schemes considered are unable to satisfy the timing constraint for message number 6. On closer examination, message 6 is 4 byte long and if corrupted with noise, the recovery time is significant since the message has to arbitrate the bus again with all other messages waiting to access the bus. Again there is no clear winner in these scheduling methods when overload develops.

Msg No:	Size /bytes	J /ms	T /ms	D /ms	Priority DM	Priority SJF*	Priority RM*	R DM	R SJF*	R RM*
1	1	0.1	50	5	1	1	8	1.432	0.456	5.136*
2	2	0.1	5	5	2	4	1	2.992	4.048	2.992
3	1	0.1	5	5	3	2	2	2.408	1.824	2.408
4	2	0.1	5	5	4	5	3	3.448	4.168	3.448

5	1	0.1	5	5	5	3	4	3.648	2.48	3.648
6	4	0.1	5	5	6	6	5	8.248*	5.392*	8.248*
7	4	0.2	10	10	7	10	6	8.768	9.36	8.768
8	1	0.2	10	10	8	7	7	5.136	4.032	4.888
9	2	0.2	10	10	9	8	9	9.288	7.864	9.288
10	2	0.2	10	10	10	9	10	9.744	8.712	9.744
11	1	0.2	50	20	11	11	11	9.296	9.816	8.192
12	4	0.3	100	100	12	14	12	18.328	19.696	18.328
13	1	0.3	100	100	13	12	13	10.4	10.272	10.4
14	1	0.2	100	100	14	13	14	19.432	15.408	19.432
15	3	0.4	1000	1000	15	17	15	10.336	19.176	10.336
16	1	0.3	1000	1000	16	15	16	18.072	18.136	18.072
17	1	0.3	1000	1000	17	16	17	18.528	18.592	18.528

Table 4: Message Delivery Time Under Transient Overload

Scheduler	DM	SJF*	RM*
Transmit Error	73	76	73
Avg Acces Delay	2.026 ms	1.979 ms	2.026 ms
Bus Utilisation	80.2 %	80.6 %	80.2 %

Table 5: Bus Utilisation And Average Access Delay Under Transient Overload

However, when the message transmission speed is increased to 250Kbit/s, the bus utilisation decreases significantly to around 70%. All the messages are now schedulable with the above scheduling schemes as shown in Table 6. In other words, the CAN based control system can tolerate 10% transmission error probability when operates at 250Kbit/s bus speed. From Table 7, assigning CAN message priority according to SJF* gives slightly better performance. The bus utilisation is higher and the average access delay is shorter in SJF* than the other two scheduling schemes.

Msg No:	Size /bytes	J /ms	T /ms	D /ms	Priority DM	Priority SJF*	Priority RM*	DM 250K	SJF* 250K	RM* 250K
1	1	0.1	50	5	1	1	8	0.424	0.424	4.624
2	2	0.1	5	5	2	4	1	1.76	2.64	1.76
3	1	0.1	5	5	3	2	2	2.184	1.696	2.184
4	2	0.1	5	5	4	5	3	3.192	3.712	3.192
5	1	0.1	5	5	5	3	4	3.128	2.184	3.128
6	4	0.1	5	5	6	6	5	4.72	4.624	4.72
7	4	0.2	10	10	7	10	6	8.92	7.816	8.92
8	1	0.2	10	10	8	7	7	4.624	3.584	3.648
9	2	0.2	10	10	9	8	9	5.176	5.016	5.176
10	2	0.2	10	10	10	9	10	8.4	4.984	8.4
11	1	0.2	50	20	11	11	11	8.664	8.208	8.664
12	4	0.3	100	100	12	14	12	9.576	9.576	9.576
13	1	0.3	100	100	13	12	13	9.608	8.632	9.608
14	1	0.2	100	100	14	13	14	10.032	9.056	10.032
15	3	0.4	1000	1000	15	17	15	9.088	15.048	9.088
16	1	0.3	1000	1000	16	15	16	9.512	9.544	9.512
17	1	0.3	1000	1000	17	16	17	9.936	9.968	9.936

Table 6: Message Delivery Time Under Transient Overload (250Kbit/s)

Scheduler	DM	SJF*	RM*

Transmit Error	70	75	70
Avg. Access Delay	1.690 ms	1.669 ms	1.690 ms
Bus Utilisation	70.35 %	70.67 %	70.35 %

Table 7: Bus Utilisation And Average Access Delay Under Transient Overload (250Kbit/s)

5.3 Dealing With Transient Overload

In distributed real time control system it is often incorporated with fault tolerance strategy to increase the system dependability. The imprecise computation technique [7] can be extended to enhance fault tolerance and graceful degradation during transient overload. Using this technique each time-critical message, or a set of messages, is decompose into compulsory and optional part. Under normal operating condition, the delivery of all messages including their optional part satisfy their timing constraint. During overloads, the optional part, or a portion of it can be skipped, that is the system operates in degraded mode to conserve the channel utilisation. In this paper, we do not address the issue of how the mode change is initiated. We assume when the mode change is started, we are given a list of messages to be modified. We evaluate the effectiveness of this technique in handling communication channel overload for message delivery under the 3 scheduling schemes: DM, SJF*, and RM*. From previous simulation results, message 6 is the first message to miss its deadline whenever overload occurs. Therefore, message 6 is decomposed into compulsory and optional parts. We label this optional part as message 18 and assigned its priority according to the DM, SJF*, and RM* scheduling policies. In normal operating condition, all message 18 will be delivered, but during overload some will be skipped. This is implemented by decreasing its frequency of generation. However in imprecise computation technique, all messages are delivered within their deadline regardless of operating conditions. The evaluation results for 1sec simulation run when using this technique to accommodate the overload condition are shown in the following tables.

Msg No:	Size /bytes	J /ms	T /ms	D /ms	Priority DM	Number of Msg Send		End-End Delivery	
						0% Error	10%Error	0% Error	10%Error
1	1	0.1	50	5	1	20	20	0.456	1.432
2	2	0.1	5	5	2	200	200	0.976	1.952
3	1	0.1	5	5	3	200	200	1.432	2.408
4	2	0.1	5	5	4	200	200	1.952	2.928
5	1	0.1	5	5	5	200	200	2.408	3.384
6	2	0.1	5	5	6	200	200	2.928	4.616
7	4	0.2	10	10	8	100	100	4.096	8.384
8	1	0.2	10	10	9	100	100	4.552	8
9	2	0.2	10	10	10	100	100	5.072	9.04
10	2	0.2	10	10	11	100	100	8.52	9.36
11	1	0.2	50	20	12	20	20	8.976	9.496
12	4	0.3	100	100	13	10	10	9.624	9.56
13	1	0.3	100	100	14	10	10	10.08	10.016
14	1	0.2	100	100	15	10	10	18.6	18.704
15	3	0.4	1000	1000	16	1	1	19.184	15.152
16	1	0.3	1000	1000	17	1	1	19.64	18.08
17	1	0.3	1000	1000	18	1	1	20.096	18.536
18	2	0.1	100	5	7	200	15	3.448	4.176

Table 8: Message Delivery With Imprecise Technique (DM Scheduler)

Msg No:	Size/ bytes	J /ms	T /ms	D /ms	Priority SJF*	Number of Msg Send		End-End Delivery	
						0% Error	10%Error	0% Error	10%Error
1	1	0.1	50	5	1	20	20	0.456	0.456
2	2	0.1	5	5	4	200	200	2.04	3.968
3	1	0.1	5	5	2	200	200	1.064	1.888

4	2	0.1	5	5	5	200	200	2.56	4.104
5	1	0.1	5	5	3	200	200	1.52	2.344
6	2	0.1	5	5	6	200	200	3.08	4.624
7	4	0.2	10	10	11	100	100	5.608	9.544
8	1	0.2	10	10	8	100	100	3.984	4.424
9	2	0.2	10	10	9	100	100	4.504	5.4
10	2	0.2	10	10	10	100	100	4.96	8.456
11	1	0.2	50	20	12	20	20	8.976	9.624
12	4	0.3	100	100	15	10	10	10.536	18.728
13	1	0.3	100	100	13	10	10	9.432	9.56
14	1	0.2	100	100	14	10	10	9.888	10.016
15	3	0.4	1000	1000	18	1	1	20.096	18.536
16	1	0.3	1000	1000	16	1	1	19.056	15.024
17	1	0.3	1000	1000	17	1	1	19.512	17.952
18	2	0.1	100	5	7	200	25	3.6	3.968

Table 9: Message Delivery With Imprecise Technique (SJF* Scheduler)

Msg No:	Size /bytes	J /ms	T /ms	D /ms	Priority RM*	Number of Msg Send		End-End Delivery	
						0% Error	10%Error	0% Error	10%Error
1	1	0.1	50	5	9	20	20	0.456	4.488
2	2	0.1	5	5	1	200	200	0.976	1.952
3	1	0.1	5	5	2	200	200	1.432	2.408
4	2	0.1	5	5	3	200	200	1.952	2.928
5	1	0.1	5	5	4	200	200	2.408	3.384
6	2	0.1	5	5	5	200	200	2.928	4.616
7	4	0.2	10	10	7	100	100	4.096	8.384
8	1	0.2	10	10	8	100	100	4.552	8
9	2	0.2	10	10	10	100	100	5.072	9.04
10	2	0.2	10	10	11	100	100	8.52	9.36
11	1	0.2	50	20	12	20	20	8.976	9.496
12	4	0.3	100	100	13	10	10	9.624	9.56
13	1	0.3	100	100	14	10	10	10.08	10.016
14	1	0.2	100	100	15	10	10	18.6	18.704
15	3	0.4	1000	1000	16	1	1	19.184	15.152
16	1	0.3	1000	1000	17	1	1	19.64	18.08
17	1	0.3	1000	1000	18	1	1	20.096	18.536
18	2	0.1	100	5	6	200	15	3.448	4.176

Table 10: Message Delivery With Imprecise Technique (RM* Scheduler)

Scheduler	DM		SJF*		RM*	
Overload Condition	0%	10%	0%	10%	0%	10%
Skipped Message %	0	46.25	0	43.75	0	46.25
No.Transmit Error	0	62	0	61	0	62
Avg Access Delay (ms)	2.266	2.002	2.210	1.962	2.266	2.002
Bus Utilisation %	84.17	77.85	84.17	78.28	84.17	77.85

Table 11: Imprecise Technique Performance

Tables 8, 9, and 10 illustrate imprecise technique can handle transient overload and enhance fault tolerance of real-time system. Whenever failure or transient overload occurs, it allows the system to function in degraded mode. The delivery of all important or compulsory messages are guaranteed, while some of the optional messages are disregarded. When this technique is incorporated to the 3 scheduling schemes, the SJF* performs slightly better with 43.75% optional messages being disregarded as shown in Table 11.

6. Conclusions

The most attractive features of CAN protocol is the short worst case bus access latency based on the message priority arbitration mechanism. This give CAN the potential for high performance and less missed deadlines in distributed real time control system. Thus, the message priority assignment algorithm is crucial to guarantee message latencies.

In this paper we have evaluated the scheduling stability of CAN messages under the DM, SJF*, and RM* scheduling schemes for normal and transient overload operating conditions. Through simulations we conclude that SJF* policy performs slightly better and the imprecise technique can be used to enhance system's fault tolerance.

References

[1] CAN Specification, Version 2, Philips Semiconductors, Hamburg 1991.

[2] Audsley, N., Burns, A., Richardson, M., Tindell, K., and Wellings, A., "Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling," *Software Engineering Journal* 8(5), pp.285-292, Sept. 1993.

[3] Tindell, K., "Guaranteed Message Latencies for Distributed Safety-Critical Hard Real-Time Control Network," Report YCS 229, Dept. Computer Science, Univ. of York, May 1994.

[4] Liu, C. L. and Layland, J. W., "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal Association for Computing Machinery*," Vol. 20(1), pp.46-61, Jan. 1973.

[5] Sha, L., Lehoczky, J. P. and Rajkumar, R., "Solutions for Some Practical Problems in Prioritized Pre-emptive Scheduling," *IEEE Real-Time Systems Symp.,* pp.181-191, 1986.

[6] Leung, J. Y. T. and Whitehead, J., "On the Complexity of Fixed-Priority Scheduling of Perioridic Real-Time Tasks," *Performance Evaluation* 2(4), pp.237-250, Dec. 1982.

[7] Liu, J. W. S., Shih, W. K., Lin, K. J., Bettati, R., and Chung, J. Y., "Imprecise Computation," *Proceeding IEEE*, Vol. 82(1), pp.83-93, Jan. 1994.

Appendix:

Signal No:	Signal Description	Size /bits	J /ms	T /ms	Periodic Sporadic	D /ms	From	To
1	Traction Battery Voltage	8	0.6	100	P	100	Battery	V/C
2	Traction Battery Current	8	0.7	100	P	100	Battery	V/C
3	Traction Battery Temp, Average	8	0.1	1000	P	1000	Battery	V/C
4	Auxiliary Battery Voltage	8	0.8	100	P	100	Battery	V/C
5	Traction Battery Temp, Max	8	1.1	1000	P	1000	Battery	V/C
6	Auxiliary Battery Current	8	0.9	100	P	100	Battery	V/C
7	Accelerator Position	7	0.1	5	P	5	Driver	V/C
8	Brake Pressure, Master Cylinder	8	0.1	5	P	5	Brakes	V/C
9	Brake Pressure, Line	8	0.2	5	P	5	Brakes	V/C

10	Transaxle Lubrication Pressure	8	0.2	100	P	100	Trans	V/C
11	Transaction Clutch Line Press.	8	0.1	5	P	5	Trans	V/C
12	Vehicle Speed	8	0.4	100	P	100	Brakes	V/C
13	Traction Battery Ground Fault	1	1.2	1000	P	1000	Battery	V/C
14	Hi&Lo Contactor Open/Close	4	0.1	50	S	5	Battery	V/C
15	Key Switch Run	1	0.2	50	S	20	Driver	V/C
16	Key Switch Start	1	0.3	50	S	20	Driver	V/C
17	Accelerator Switch	2	0.4	50	S	20	Driver	V/C
18	Brake Switch	1	0.3	20	S	20	Brakes	V/C
19	Emergency Brake	1	0.5	50	S	20	Driver	V/C
20	Shift Lever (PRNDL)	3	0.6	50	S	20	Driver	V/C
21	Motor/Trans Over Temperature	2	0.3	1000	P	1000	Trans	V/C
22	Speed Control	3	0.7	50	S	20	Driver	V/C
23	12V Power Ack Vehicle Control	1	0.2	50	S	20	Battery	V/C
24	12V Power Ack Inverter	1	0.3	50	S	20	Battery	V/C
25	12V Power Ack I/M Control	1	0.4	50	S	20	Battery	V/C
26	Brake Mode (Parallel/Split)	1	0.8	50	S	20	Driver	V/C
27	SOC Reset	1	0.9	50	S	20	Driver	V/C
28	Interlock	1	0.5	50	S	20	Battery	V/C
29	High Contactor Control	8	0.3	10	P	10	V/C	Battery
30	Low Contactor Control	8	0.4	10	P	10	V/C	Battery
31	Reverse & 2nd Gear Clutches	2	0.5	50	S	20	V/C	Trans
32	Clutch Pressure Control	8	0.1	5	P	5	V/C	Battery
33	DC/DC Converter	1	1.6	1000	P	1000	V/C	Battery
34	DC/DC Converter Current Cont	8	0.6	50	S	20	V/C	Battery
35	12V Power Relay	1	0.7	50	S	20	V/C	Battery
36	Traction Battery Gnd Fault Test	2	1.7	1000	P	1000	V/C	Brakes
37	Brake Solenoid	1	0.8	50	S	20	V/C	Brakes
38	Backup Alarm	1	0.9	50	S	20	V/C	Brakes
39	Warning Lights	7	1.0	50	S	20	V/C	Ins
40	Key Switch	1	1.1	50	S	20	V/C	I/M C
41	Main Contactor Close	1	0.3	50	S	20	I/M C	V/C
42	Torque Command	8	0.2	5	P	5	V/C	I/M C
43	Torque Measured	8	0.1	5	P	5	I/M C	V/C
44	FWD/REV	1	1.2	50	S	20	V/C	I/M C
45	RWD/REV Ack	1	0.4	50	S	20	I/M C	V/C
46	Idle	1	1.3	50	S	20	V/C	I/M C
47	Inhibit	1	0.5	50	S	20	I/M C	V/C
48	Shift in Progress	1	1.4	50	S	20	V/C	I/M C
49	Processed Motor Speed	8	0.2	5	P	5	I/M C	V/C
50	Inverter Temperature Status	2	0.6	50	S	20	I/M C	V/C
51	Shutdown	1	0.7	50	S	20	I/M C	V/C
52	Status/Malfunction (TBD)	8	0.8	50	S	20	I/M C	V/C
53	Main Contactor Ack	1	1.5	50	S	20	V/C	I/M C