Timing Performance of Adaptable Distributed Real-Time Control Systems

M. Dani Baba·, H. Ekiz, A. Kutlu and E. T. Powner
School of Engineering
University of Sussex
Falmer, Brighton BN1 9QT, UK.

Abstract

A distributed real-time computer system consists of several processing nodes interconnected by communication channels. In a safety critical application, the real-time system should maintain timely and dependable services despite component failures or environmental changes such as transient overloads. In this paper the imprecise computation technique is integrated with fault tolerance schemes and adopted as the Adaptable Management System (AMS) which adjusts the operating strategy of the real-time system in response to changing application environments and the internal fault patterns.

The timing response performance of the AMS for a Controller Area Network (CAN) based distributed real-time system is evaluated as to whether the timing constraints is satisfied for normal, overload, and degraded operational modes. The simulation study employs the Society of Automotive Engineers (SAE) real-time control system benchmark as the workload model. The simulation results also show that the quality of service of the AMS can be improved by varying the optional message in the workload.

1.0 Introduction

A hard real-time system is typically composed of a number of periodic and sporadic tasks which communicate their data by passing messages. In a distributed hard real-time system these messages are transferred between processing nodes via a communication network. In order to guarantee that the timing requirements of all tasks are satisfied, the total communication delay between sending and receiving a message must be bounded.

In a safety critical application, the distributed hard real-time system should maintain timely and dependable services despite component failures or overloads due to changes in the environment and system conditions. When the real-time system becomes overloaded, it's hard real-time tasks may miss their deadlines and if employed in a critical application can results in fatal consequences. When a component fails or an overload occurs, we want the system to degrade in a graceful, predictable manner. Likewise the ability to handle component faults and changes in operational modes are essential for a distributed hard real-time system, and particularly when used in a safety critical application. The next generation of complex hard real-time systems will be required to exhibit; adaptive and dynamic behaviour, resilience to software/hardware failures and graceful degradation under conditions of overload [1].

To achieve such complex systems, two conflicting requirements must be satisfied: Firstly the safety critical services must be guaranteed to maintain results of minimum acceptable quality and reliability by their deadlines. Secondly the system utility as determined

· School of Engineering, Institute Technology MARA, 40450 Shah Alam, Selangor, Malaysia.

by the timeliness, precision and confidence level of the results produced must be maximised. For a critical application, it is reasonable to argue that no event should be unpredictable and that schedulability should be guaranteed before execution. This implies the use of a static scheduling algorithm which can satisfy the first requirement. Whilst the fault tolerance techniques such as the distributed recovery blocks, N-modular redundancy and N-version programming can provide resilience to failures and some features of graceful degradation.

To meet the second requirement of maximising system utility, such techniques as Imprecise computation [3], Multiple versions [4] and Approximate processing [5] schemes can be considered. Each tasks in these schemes can be decomposed into mandatory and optional components. The mandatory components represent critical tasks which must be executed before their deadlines, whilst the timely completion of optional components enhances the utility of the system but is not essential. Typically, the complete execution of optional tasks will produce the precise desired results. However, to execute all the mandatory and optional tasks in the presence of components fault, represents a situation of extreme overload, there remains the issue of choosing the suitable operational modes for the various application environments and fault requirements.

Our approach to this issue is to propose a framework towards an Adaptable Management System (AMS) which adjusts the operating strategy of the real-time system in response to changing application environments and the internal fault patterns. In this paper the imprecise computation technique [3] is incorporated into an Adaptable Management System (AMS) which can adjust its operating strategy in response to changes in the application environment as well as changes in the internal fault pattern. The timing performance of the AMS for a Controller Area Network (CAN) based distributed real-time control system is evaluated as to whether the timing constraint is satisfied for normal, overload, and degraded operational modes. The simulation uses the Society of Automotive Engineers (SAE) automotive control system benchmark which is typically associated with real-time control system as the workload. This paper is structured as follows: The next section outlines the CAN protocol. Section 3 describes the SAE benchmark workload model. Section 4 presents the simulation model. Section 5 presents the performance evaluation results. Section 6 describes the quality of result. Finally the paper concludes with Section 7.

2.0 Controller Area Network.

The Controller Area Network (CAN) is an advanced serial communication protocol which can efficiently support distributed real-time control system with a very high level of data integrity [2]. CAN was originally designed for automotive applications where nodes such as the engine management system, anti-lock braking electronic unit, and other vehicle control systems were to be connected to single communication bus. A CAN based system is typically consists of a number of processors which are connected to the broadcast bus via the CAN controller devices. More information on CAN protocol and the operation of the network controller is available in literature [2,6]. However, some features of the protocol which are especially relevent to our study are discussed here.

2.1 Error Detection and Recovery

The CAN protocol implements powerful error detection mechanisms focusing on Cyclic Redundancy Check (CRC), bit stuffing, and both positive and negative acknowledgement. The significant feature of CAN error detection scheme is that CAN alerts all stations in the network when an error has been detected at the time the error is first recognised. As a result, non-productive bus time is minimized and data quality is ensured. The stations are provided with this information by using two complimentary acknowledge schemes. The receiver stations will provide positive acknowledge upon receiving a correct message. Negative acknowledgement will be sent out by any station at the point it detects an error in the message. The corrupted message will be rejected by all receiver stations and the transmitter station will automatically re-enter arbitration to retransmit the failed message. Additionally, if a CAN controller transmits repetitive errors, it will remove itself from the bus and notify the host processor.

3.0 Workload Model

The structure of the workload model used in the simulation contains seven control nodes. These seven automotive control nodes are connected to a real-time communication

channel which handles the SAE benchmark signals containing 53 different types of messages. Some of these messages are sporadic in nature while others are periodic control data. Since the benchmark does not specify the minimum interarrival time for some sporadic messages, then sensible values are assumed. From previous work [7], the 53 types of the benchmark signals are shown to be unschedulable using the Deadline Monotonic (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 a higher bus speed, the DM scheme has no difficulty in scheduling all the benchmark signals. To overcome the scheduling problem, a message piggybacking technique has been employed to reduce the bus utilisation. This is implemented in the form of message server which polls and collects several messages from the same source and then sends out as a single long message. The server chosen has 10ms period and a latency of 10ms. The newly transformed benchmark signals now consists of only 17 message types as shown in Table: 1. This newly transformed set of benchmark signals has been adopted as the workload model in the simulation. The bus speed used in the simulation is 125Kbits/s since a higher speed does not pose any scheduling difficulty and is also suitable for use in twisted pair cabling.

Message Number	Signals Number	Size (bytes)	Jitter (ms)	Period (ms)	Deadline (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

4.0 Simulation Model

The 17 transformed benchmark signals are mapped into the CAN standard message format. The message priority is ordered according to deadline monotonic scheme: a message with short deadline is assigned with a higher priority [7]. In this simulation the maximum message delivery delay is evaluated as in Equation 1.

$$MessageDeliveryDelay = \frac{MessageFrame}{Baudrate} + OverheadTime \quad (1)$$

The message frame as depicted in Figure 2 contains information such as the identifier, error checking, control field, and the user data. Since a stuff bit is added by the transmitting node after 5 consecutive equal bit levels, the message frame length can vary as in Equation 2. The message frame will be at its maximum length if after every 5th bit in the bit stream a stuffbit is added. This means that the maximum message frame length is 20% longer than the minimum length.

$$8 \times Data + 47bit \le MessageFrame \le 8 \times Data + 47bit + 0.2(8 \times Data + 34bit) \quad (2)$$

The overhead of a CAN message frame amounts to a total of 47 bits, out of which 34 bits are subjected to bit-stuffing. The data field (between 0 and 8 bytes) in the message frame is also being subjected to bit-stuffing. However, on the average 2 stuffbits are sufficient for most short messages. The message frame length is the most important factor that influences the delivery delay time. The overhead time comprises of software delay, controller delay and the bus access time. However, in this paper, the evaluation of the worst case message delivery delay ignores the software delay incurred when sending or receiving messages.

To ensure that the worse case scenario exists in the simulation environment, all the 17 types of messages are initially generated simultaneously. This creates 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 latency time.

4.1 Fault Tolerant Model

The distributed fault tolerant model of the real-time control system consists of a set of fail-silent computing stations connected by one or two replicated broadcast global communication channels. The simulation model adopts the approach to cluster the computing stations into Fault Tolerant Units (FTU) [8]. The FTU consists of two active but redundant computing stations also known as network nodes, namely the primary and secondary nodes. All the network nodes belonging to a particular FTU to provide the same service. They receive and process similar messages, and therefore maintain internal state equivalence. However, there is a slight difference since the primary node can always broadcast its messages on the global communication channel, whilst the secondary node can only transmit its messages whenever the primary node is faulty. Every FTU provides its specified services as long as at least one of the network nodes is functioning. Figure 1 shows the distributed fault tolerant model of the real-time control system which comprises of seven FTUs.

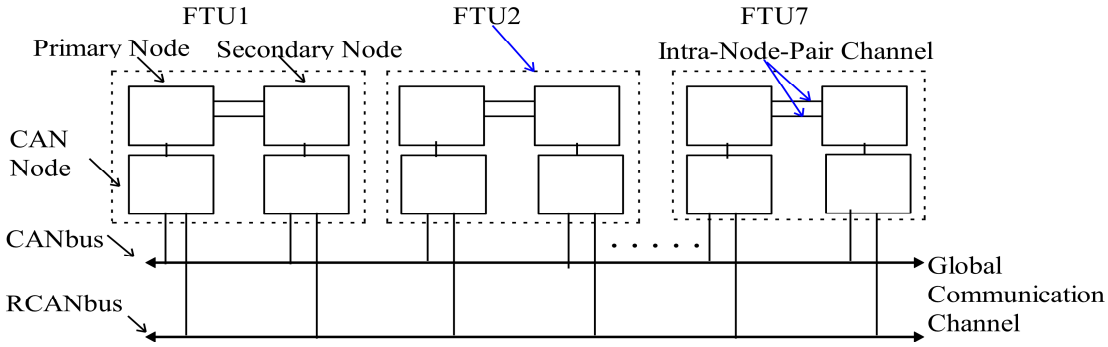

Figure 1: Distributed Fault Tolerant Model

The CAN error handling and recovery mechanisms are assumed to be integrated into the Adaptive Management System (AMS), and the AMS is distributed to all network nodes. If the AMS detects the primary node is faulty, then the AMS will disconnect the faulty node from the global bus. However, before the faulty primary node can be disconnected, it's AMS will activate the secondary node from the same FTU to begin transmitting messages. The secondary node can be initiated via the dual redundant intra-node-pair channel of the FTU. While the global communication channel consists of two active broadcast buses, and here called CANbus and its replicate the RCANbus. For improved bus efficiency, both busses are involved in the transfer of messages, and if one of them fails, then all messages will go through the working bus. In CAN protocol the valid bus status can only have one of two complementary logic values: Dominant or Recessive. For any other logic state the AMS will assume the bus is faulty. Hence the fault tolerant model implies that once the faulty components (network node or communication channel) are detected by the AMS, the recovery action takes immediate effect to initiate the respective active redundant components to continue the real-time processing.

4.2 Operational Modes

To develop an adaptive fault tolerant real-time system, the services provided should be specified with different levels of required tolerance and acceptable functionality. Figure 2

illustrates the acceptable fault tolerance and functionality for the three basic operational modes in an automotive control system when it experiences component faults and transient overloads. The noise probability percentage is used to model message transfer lost due to noise induced in the communication channels. Noise probability for both channels is assumed to be independent. A 0% noise probability indicates that there is no message lost, while 100% means all the message transferred is lost due to noise. During the normal mode operation the channel is assumed to be stable and there is no noise present to corrupt the transfer of messages. If there is noise detected in the channel, the AMS will initiate the mode change to the transient overload mode. In transient overload mode, the real-time system can tolerate noisy channel, but if any one of the two active noisy channel is faulty, then the AMS will activate the change to the degraded operational mode. When the faulty channel is restored, the system will return to the transient overload or the normal mode of operation if the channel is stable. However in this paper a study of the issue of how the mode change occur is not addressed. Instead it is assumed that when a mode change occurs, a given set of messages are to be transferred in real-time.

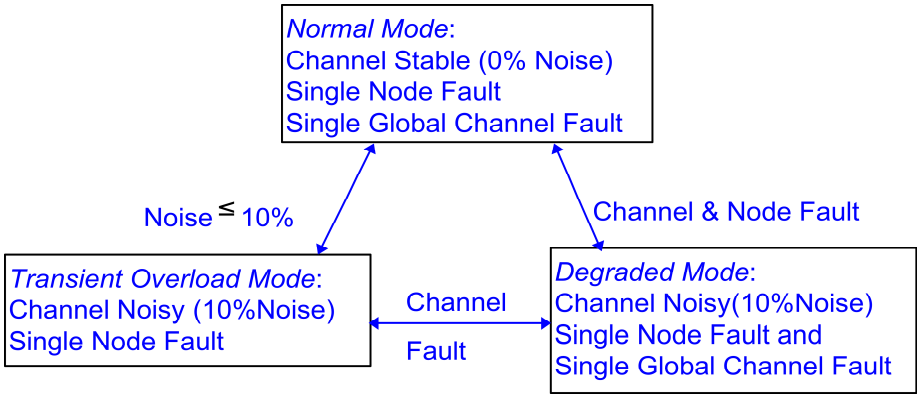

Figure 2: Adaptive Fault Tolerant Mode Space.

5.0 Simulation Results

Table 2 shows the simulation results of the worst case timing response evaluation of an adaptive fault tolerant real-time control system that employs imprecise computation technique for various modes of operation. The results listed are collected from the worst case delay evaluation from several runs each of 2 seconds simulation. For clearer presentation, the message parameters such as jitter, period, and deadline which are invariant across all the three modes are not included in Table 2, but they are available in Table 1.

Msg or Priority No.	Message Length (bytes)	Normal Mode Response R (ms)		Transient Overload R (ms) @ 10% Error		Degraded Mode R (ms) @ 10% Error	
		0% Error	5% Error	No IT	With IT	No IT	With IT
1	1	0.456	0.456	0.456	0.456	0.456	0.456
2	2	0.976	1.496	2.992	2.472	2.992	2.928
3	1	1.432	1.952	2.408	2.344	2.408	2.344
4	2 (1 for TO) (1 for DM*)	1.952	2.992	6.04*	3.08	6.096*	3.384
5	1	2.408	3.056	3.576	3.32	4.032	3.84
6	4 (2 for TO) (1 for DM*)	3.056	4.52	6.104*	4.664	8.896*	4.296
7	4	3.704	5.264	5.264	5.136	8.576	8.128
8	1	4.16	4.808	4.808	4.744	8.056	5.4
9	2	4.68	5.784	6.76	5.2	9.224	8.264
10	2	5.136	6.24	6.496	5.656	9.68	9.056
11	1	5.592	6.696	7.216	5.92	10.328	9.176
12	4	6.24	6.888	7.344	7.152	20.008	9.824
13	1	6.696	7.344	7.344	7.024	20.136	10.28

14	1	7.152	7.80	7.80	7.608	25.272	15.16
15	3	7.736	7.736	8.776	7.544	19.368	18.088
16	1	8.192	8.192	8.192	8.0	19.824	10.28
17	1	8.648	8.648	9.92	8.456	20.28	18.544

Note: TO=Transient Overload with IT, DM*=Degraded Mode with IT, *= Missing Deadline

Table 2: Worst Case Message Delivery Time for Normal, Transient Overload and Degraded Mode of operation.

5.1 Message Delivery Time Under Normal Mode

From Table 2, all the 17 types of the transformed benchmark signals are schedulable (i.e. the worst case message delay satisfy its timing constraint) with Deadline Monotonic (DM) scheme when operating in the normal operational mode. Under normal mode it is assumed that there is no noise present in the communication channel to corrupt the messages while in transfer. However when 5% noise probability is introduced in both channels, the worst case delivery time for all the messages still meet their timing constraints. This result indicates that when the system is in the normal mode it can tolerate 5% noise probability which is handled by the CAN protocol error recovery mechanism.

5.2 Message Delivery Time Under Transient Overload Mode

In transient overload mode, we assume the global communication channels are noisy and may cause some messages to be corrupted and discarded. With CAN protocol the transmitter will retransmit the lost message at its next opportunity through normal bus access arbitration. The side effect of automatic retransmission may cause some messages to miss the deadline as indicated by * in Table 2. One way of overcoming timing faults due to changing application environment is to apply the imprecise computation technique as the adaptive fault tolerant mechanism for the real-time system. With this technique, a message or a set of messages can be partitioned into compulsory and optional parts. During normal operating condition both parts of a message or set of messages are transmitted, hence resulting in the desired timely precise message. Whenever abnormality occurs, the optional part can be skipped to conserve system resources. Skipping the optional part produces an imprecise (i.e. approximate) result which can still be acceptable to the application if the controlled system remains stable.

In the simulation, the Imprecise Technique (IT) is applied to Message Nos: 4 and 6 only as these messages miss their deadline whenever transient overload develops as 10% noise is introduced in both channels. It is appropriate to apply IT to these messages as they are long messages with short deadlines and if corrupted with noise their retransmission delay could be significant and can lead to missing of their deadline. The length of the optional part for Message Nos: 4 and 6 are 1 byte and 2 byte respectively. Results from Table 2 shows that all messages satisfy their timing constraint even under transient overload after applying this technique.

5.3 Message Delivery Time Under Degraded Mode

The adaptive fault tolerant real-time control system enters the degraded operational mode whenever one of the global communication channel becomes faulty. A critical situation exists if the functioning channel also experiences excessive interference from the environment. From Table 2, Message Nos: 4 and 6 fail to satisfy their timing constraint again if the noise probability is increased to 10%. In the simulation, Message Nos: 4 and 6 is treated with the imprecise technique to increase system robustness towards a single channel fault and noisy environment.

6.0 Quality of Result (QoR)

In imprecise computation technique, both the compulsory and optional parts of a message must be transmitted completely to achieve the desired precise result. During overloads, the optional part can be skipped or discarded which produces an approximate result. The accuracy of the approximate result is proportional to the length of the discarded optional message. In the simulation the imprecise technique is applied to Message Numbers: 4 and 6 which always miss their deadline during overloads. The quality or accuracy of these messages after being treated with IT is shown in Table 3 for the three operational modes. The

results from Table 3 establish that the quality of result or the performance of the real-time system degrades as the operational mode changes from normal to transient and finally to the degraded mode.

Mode of Operation	Accuracy of Messages		
	Message No: 4	Message No: 6	Other Messages
Normal Mode	100 %	100 %	100 %
Transient Overload Mode	50 %	50 %	100 %
Degraded Mode	50 %	25 %	100 %

Table 3: Quality of Result

6.1 To Improve Quality of Result

Another approach to improve the quality of result when using the imprecise technique is to vary the workload. In the simulation the workload is varied by transmitting the optional messages at a much lower rate. For this case the optional part of Message Nos: 4 and 6 after treatment with IT are labelled as Message Nos: 18 and 19 respectively. These optional messages are not totally discarded as in the previous evaluation, but instead are sampled at lower rate. The result of such assessment is to improve the performance of the adaptive fault tolerant system for the three operational modes is shown in Table 5. The parameters which are invariant to the three operational modes are listed in Table 4.

Msg No:	Size (bytes)	Jitter (ms)	Period (ms)	Deadline (ms)	Priority DM
1	1	0.1	50	5	1
2	2	0.1	5	5	2
3	1	0.1	5	5	3
4	1	0.1	5	5	4
5	1	0.1	5	5	6
6	2	0.1	5	5	7
7	4	0.2	10	10	9
8	1	0.2	10	10	10
9	2	0.2	10	10	11
10	2	0.2	10	10	12
11	1	0.2	50	20	13
12	4	0.3	100	100	14
13	1	0.3	100	100	15
14	1	0.2	100	100	16
15	3	0.4	1000	1000	17
16	1	0.3	1000	1000	18
17	1	0.3	1000	1000	19
18	1	0.1	5	5	5
19	2	0.1	5	5	8

Table 4: The Invariant Parameters for the Three Operational Modes

Msg No:	Number of Messages Transferred	Normal Mode Response R(ms)		Transient Overload R(ms) @ 10% Error		Degraded Mode R(ms) @ 10% Error	
		0% Error	5% Error	No IT	With IT	No IT	With IT
1	40	0.456	0.456	0.864	0.456	1.536	0.456
2	400	0.976	1.952	3.904	3.448	3.928	3.904
3	400	1.432	2.344	2.344	2.344	3.272	2.36
4	400	1.888	2.408	2.80	2.80	2.816	2.80
5	400	2.80	3.32	4.36	4.296	4.632	4.332
6	400	3.32	4.36	4.88	4.136	8.392*	4.424

7	200	4.488	5.656	6.176	5.784	9.952	8.976
8	200	4.944	5.40	5.464	4.944	10.232*	5.4
9	200	5.464	5.984	6.568	5.984	15.464*	8.52
10	200	5.92	6.376	7.256	6.8	29.368*	9.432
11	40	6.376	7.064	7.064	6.376	20.232*	9.432
12	20	7.024	8.128	8.128	7.024	30.472	18.144
13	20	7.48	7.936	8.0	7.48	49.304	18.6
14	20	7.936	8.584	8.584	7.936	50.216	19.056
15	2	8.52	8.52	8.52	8.52	49.664	18.28
16	2	8.976	8.976	8.976	9.888	50.12	18.736
17	2	9.432	9.432	9.432	9.432	70.28	19.192
18	400 (34 for TO) (10 for DM*)	2.344	3.256	3.384	2.864	3.776	2.344
19	400 (34 for TO) (10 for DM*)	3.84	5.0	5.464*	4.816	8.912*	4.296

Note: TO= Transient Overload with IT, DM*= Degraded Mode with IT, *= Missing Deadline

Table 5: Worst Case Delivery Time for Normal, Transient Overload and Degraded Mode with improved QoR.

Table 5 shows the result of the worst case message delay time collected after 2 seconds simulation run for the normal, transient overload, and degraded modes of operation with improved QoR. Under the normal operational mode the real-time system barely tolerates 5% noise probability. The optional part of Message No: 6 (i.e. Message No: 19) can just satisfy the timing requirement as shown in Table 5. The increase in message delay is due to the increase in bus utilisation and node processing time since the CAN protocol incurs at least 47 bits of overhead per optional message transferred.

When operating in the transient overload mode, the optional Message No: 19 fails the timing constraint when 10% noise probability is induced into both channels. By varying the load of the optional messages (i.e. Message No:18 and 19) the system can satisfy the timing requirement. For a 2 seconds simulation run, the workload is varied by sampling the optional message at lower rate such that 34 out of 400 of these messages are transmitted.

For degraded operational mode the system with the improved QoR scheme can sustain upto six message types of timing failures. However only 10 out of 400 of the optional message load are transmitted as shown in Table 5. The overall quality of service in this approach where the workload is varied is much better when compared to the previous case for all operational modes as shown in Table 6.

	Accuracy of Messages		
Mode of Operation	Message No: 4	Message No: 6	Other Messages
Normal Mode	100%	100%	100%
Transient Overload Mode	54.25%	54.25%	100%
Degraded Mode	51.25%	51.25%	100%

Table 6: Improved Quality of Result

7.0 Conclusion

This paper has attempted to establish that imprecise computation technique can be employed as adaptive fault tolerant mechanism in distributed real-time CAN based system. From the performance evaluation, the adaptive fault tolerant real-time system that uses imprecise technique can maintain timely and dependable services despite component failures and changes in application environment. This goal is achieved by trading the quality of service for guaranteed message timing response. To improve the quality of service the optional messages treated with imprecise technique have to be adjusted such that the number of discarded or skipped messages have to be minimised.

References:

[1] R. Davis, S. Punnekkat, N. Audsley and A. Burns, "Flexible Scheduling for Adaptable Real-Time Systems", Proceedings IEEE Real-Time Technology and Applications, Illinois, USA, pp.230-239, May 1995.
[2] CAN Specification, Version 2, Philips Semiconductors, Hamburg 1991.
[3] J.W.S. Liu, W.K. Shih, K.J. Lin, R. Bettati and J.Y. Chung, "Imprecise Computation", Proceeding IEEE, Vol.82(1), pp.83-93, Jan 1994.
[4] N. Malcolm and W. Zhao, "Version Selection Schemes for Hard Real-Time Communications", Proceedings IEEE Real-Time Systems Symposium, Texas, USA, pp.12-21, Dec 1991.
[5] A. Garvey and V. Lesser, "Scheduling Satisfying Tasks with a Focus on Design-to-time Scheduling", Proceedings IEEE Workshop on Imprecise and Approximate Computation, Arizona, USA, pp.25-29, Dec 1992.
[6] ISO/DIS 11898, "Road Vehicles -- Interchange of Digital Information --Controller Area Network (CAN) for High Speed Communication", Nov. 1993.
[7] 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.
[8] Kopetz, H., and Gunter, G., “TTP - A Protocol for Fault Tolerant Real-Time Systems”, IEEE Computer, pp14-23, Jan. 1994.