Probabilistic Scheduling Guarantees in Distributed Real-Time Systems under Error Bursts

Hüseyin Aysan[1], Radu Dobrin[1], Sasikumar Punnekkat[1], and Julián Proenza[2]
[1]Mälardalen University, Västerås, Sweden
[2]University of the Balearic Islands, Palma de Mallorca, Spain
{huseyin.aysan, radu.dobrin, sasikumar.punnekkat}@mdh.se, julian.proenza@uib.es

ABSTRACT

Dependable communication is becoming a critical factor due to the pervasive usage of networked embedded systems that increasingly interact with human lives in many real-time applications. However, these systems are often subject to faults that manifest as error bursts and affect the timing properties of the messages used in the communication. Controller Area Network (CAN) has gained wider acceptance as a standard in a large number of distributed industrial and control applications, mostly due to its cost effectiveness, efficient bandwidth utilization, ability to provide real-time guarantees, as well as its fault-tolerant capability. Research so far has focussed on rather simplistic error models which assume only singleton errors separated by a minimum interarrival time. However, error bursts of various lengths during message transmissions have an adverse effect on the message response times that needs to be accounted for. In this paper we propose a methodology which enables the provision of appropriate probabilistic real-time guarantees in distributed real-time systems under error bursts. The proposed approach introduces a comprehensive probabilistic error model together with appropriate schedulability analysis for the particular case of real-time message scheduling on CAN.

Keywords

Distributed Real-time Systems, CAN, Dependability, Fault tolerance, Time redundancy.

1. INTRODUCTION

Networked embedded systems are deployed ubiquitously in applications that interact and control our lives including in safety critical applications. These systems are increasingly interacting with each other in a distributed manner and providing reliable communications in such contexts is an important research question. In order to be able to provide accurate analysis of such systems, it is essential to use a realistic error model that takes into account not only the severity, but also the duration of errors. Controller Area Network (CAN) has been widely used in the automotive and automation industries due to its ease in use, low cost and provided reduction in wiring complexity. The priority-based message scheduling used in CAN has a number of advantages, some of the most important being the efficient bandwidth utilization, flexibility, simple implementation and small overhead. Moreover, CAN provides for real-time guarantees as well as fault-tolerance for messages under transient errors. However, error bursts typically affect several message retransmission attempts and contribute to potentially large response time that may deem the system unschedulable. Additionally, the existing schedulability analysis on CAN does not take into account the interplay between the minimum interarrival time between bursts, minimum interarrival time between errors within a burst and the burst duration.

CAN was designed in the 1980s at Robert Bosch GmbH ([15]) with a special focus on automotive real-time requirements. The most important feature of CAN from the real-time perspective is its predictable behavior. CAN provides means for prioritized control of the transmission medium by using an arbitration mechanism which guarantees that the highest priority message that enters an arbitration will be transmitted first. This makes CAN amenable to response time analysis akin to those performed on fixed priority task sets. Volcano methodology used by Volvo [8] is an example of the acceptance of such analysis by the industry.

The model underlying the basic CAN analysis assumes an error free communication bus, i.e. all messages sent are assumed to be correctly received, which may not always be true. For instance, in applications such as automobiles, the systems are often subjected to high degrees of Electro Magnetic Interference (EMI) from the operational environment which can potentially cause transmission errors. The common causes for such interference include cellular phones and other radio equipments inside the vehicle and electrical devices like switches and relays, radio transmissions from external sources and lightning in the environment. Electro Magnetic Compatibility (EMC) has been seriously considered by the automotive industry for more than 40 years, and several legislations and directives are in effect to tackle the interference problem [16]. However, even today it has not been possible to completely eliminate the effects of EMI since exact characterization of all such interferences defy comprehension. Though usage of an all-optical network could greatly eliminate EMI problems, it is not favored by the cost-conscious automotive industry.

These interferences cause errors in the transmitted data, which could indirectly lead to catastrophic failures. To reduce the risks due to erroneous transmissions, CAN designers have provided elaborate error checking and error confinement features in the protocol. Basic philosophy of these features is to identify an error as fast as possible and then retransmit the affected message. This implies that in systems without spatial redundancy of communication medium/controllers, the fault-tolerance (FT) mechanism employed is time redundancy which could have an adverse impact on the latencies of message sets; potentially leading to violation of timing requirements.

Majority of the earlier research efforts were based on a simplified error model assumption that only singleton errors can occur in the systems and that they are separated at least by a known minimum interarrival time. However, error bursts of varying lengths are not uncommon during message transmissions and they have an adverse effect on the message response times. Hence the versatility and applicability of the existing models are limited, in the sense that they are incapable of representing complex scenarios and interdependent errors, thus potentially resulting in inaccurate schedulability analyses.

In this paper we propose a generalized parametric error model which is essential to provide an accurate representation of faults and associated errors, and provide a probabilistic schedulability analysis for distributed real-time tasks. We instantiate the proposed framework to real-time message scheduling on CAN and extend the existing CAN response time analysis [20, 5] to cope with burst errors modeled with an improved accuracy that enables the specification of a range of new parameters including e.g., burst length and intensity.

The remainder of the paper is organized as follows. In the next section, we present the real-time system model and in Section 3, we present our error model. Section 4 gives a brief summary of the Controller Area Network. Section 5 describes our proposed methodology together with an illustrative example, and finally Section 6 concludes the paper.

2. REAL-TIME SYSTEM MODEL

We assume a distributed real-time architecture consisting of sensors, actuators and processing nodes communicating over CAN. The communication is performed via a set of periodic messages, $\Gamma = \{M_1, M_2, \ldots\}$. For the sake of generality, we assume that a message consists of one or more frames. However, the analysis presented in this paper applies to the particular case of single frame messages as well. While the CAN network communication is non-preemptive during the frame transmissions, messages composed of more than one frame can preempt each other at frame boundaries. Additionally, the non-preemptiveness of message frames may cause a higher priority message to be blocked by a lower priority message for at most one frame length, if the high priority message is released during the transmission of a lower priority frame. This priority inversion phenomenon can affect all messages except the lowest priority one, and only once per message period, before the transmission of the first message frame ([11]).

Each CAN message M_i has a period T_i, a relative deadline D_i which is assumed to be equal to the period, a priority P_i (defined by the message identifier), the number of frames N_i that forms the message and a worst case transmission time C_i of the message in an error-free scenario:

$$C_i = N_i * f^{max} * \tau_{bit} \tag{1}$$

where f^{max} is the maximum frame size in number of bits, and τ_{bit} is the time it takes to transmit a single bit on CAN.

3. ERROR MODEL

Safety-critical embedded systems typically work in harsh environments where they are exposed to frequent transient faults such as power supply jitter, network noise and radiation. Pizza et. al. [17] observed from published statistics that the ratio between the frequencies of transient and permanent faults varies from 4 to 1000. We follow the dependability concepts presented by Laprie [14] and Avizienis et. al.[3], and assume that systems are exposed to faults with probabilities depending on the characteristics of the systems and the environments that they are operating in.

Once an error occurs, it is likely that the fault causing this error will be in effect for a certain duration and will cause more errors during that period. Burton and Sullivan [7] defined error bursts consisting of errors that are occurring during the period that a fault is in effect and if two successive errors within that duration does not exceed a certain maximum error-free period. As the errors in a burst are caused by a single fault source, they will have a different probability of occurrence than the errors caused by independent faults. This probability depends on several factors, such as the type and the severity of the fault, the resistance of the hardware to the fault, and the reaction of the fault detection and fault tolerance mechanisms to the fault. Furthermore, the error bursts can have different durations due to various reasons. For example, if we imagine a vehicle as our system under observation, which passes through a field with strong electromagnetic interference, the duration of the exposure to this fault is related to the area of this field as well as the velocity of the vehicle. Ferreira [12] shows that 90% of the errors occurred on a CAN network are in the form of error bursts with an average length of $5\mu sec$ in an aggressive environment (factory conditions). However, the probability distribution of the burst length is highly dependent to the environment and more experimental studies are required in order to determine valid distributions for different domains. An example of such a study was performed by [7] for telecommunication systems.

In our work, we assume that, each frame failure is detected as soon as it occurs by the built in CAN error detection mechanisms and upon each frame failure, an identical frame to the failed frame is scheduled for re-transmission following the error frame.

Our *error model* consists of the following parameters:

1. T_E**:** The minimum interarrival time between independent error bursts.

2. T_E^{burst}**:** The minimum interarrival time between errors *within* a burst.

3. l**:** The length of the error burst.

Consequently, we obtain the following probability functions:

1. $Pr_{error}(t)$**:** The probability of error occurrence within a time interval of length t can be calculated by using the Poisson probability distribution as described by [6]. The errors can occur either in form of single errors or error bursts. Poisson distribution is a discrete probability distribution used for finding the probability of a number of events occurring in a fixed time period, assuming that the events occur at a constant rate and their occurrences are independent. In our case, the events are error occurrences, hence the error occurrence rate for transient errors is assumed to be constant. This rate (the expected number of events in a unit time as denoted by λ) not only depends on the system but also on the type of environment. For a given system, the common values for λ range from 10^2 errors per hour in aggressive environments to 10^{-2} errors per hour in lab conditions as presented by [12] and [19].

   The probability of m events during a time period of t is calculated as shown below.

   $$Pr_m(t) = \frac{e^{-\lambda t}(\lambda t)^m}{m!}$$

   If we assume that the event is an error, then the probability of no error during the lifetime or mission time (L) of the system is given by

   $$Pr_{no\_error}(t = L) = e^{-\lambda L}$$

   Thus, the probability of at least one error during L is

   $$Pr_{at\_least\_one\_error}(t = L) = 1 - e^{-\lambda L}$$

   The lifetime or mission time of a system can vary largely depending on the domain, typically ranging from 1 hour for a plane to take a short trip to 15 years for a satellite to complete its lifetime.

   In this paper, we are interested in the probabilities of the messages meeting their deadlines under the error rate assumptions.

2. $f(l)$**:** The probability mass function for the error burst length l which is a function that gives the probability that an error burst length is exactly equal to some value.

3. $Pr_{error|burst}(t)$**:** The probability of an error under an error burst during a time interval of length t which is a function of the error burst length l and λ^{burst}.

4. CONTROLLER AREA NETWORK (CAN)

CAN is a broadcast bus designed to operate at speeds of up to 1 Mbps. Data is transmitted in messages containing between 0 and 8 bytes of data. An 11 bit identifier is associated with each message frame. There is also an extended CAN format with a 29 bit identifier, but since this format is identical in all other respects, it will not be considered here. The identifier is required to be unique, in the sense that two simultaneously active message frames originating from different sources must have distinct identifiers. The identifier serves two purposes: (1) assigning a priority to the message frame, and (2) enabling receivers to filter message frames. A station filters message frames by only receiving message frames with particular bit patterns. The use of the identifier as priority is the most important part of CAN with respect to real-time performance.

CAN is a collision-detect broadcast bus, which uses deterministic collision resolution to control access to the bus. The basis for the access mechanism is the electrical characteristics of CAN bus: if multiple stations are transmitting concurrently and one station transmits a '0' then all stations monitoring the bus will see a '0'. Conversely, only if all stations transmit a '1' will all processors monitoring the bus see a '1'. This behavior is used to resolve collisions: each station waits until the bus is idle. When silence is detected, each station begins to transmit the highest priority message frame held in its output queue whilst monitoring the bus. The identifier is the first part of the message frame to be transmitted; the identifier is transmitted from the most-significant to the least-significant bit. If a station transmits a recessive bit ('1'), but monitors the bus and sees a dominant bit ('0'), then it stops transmitting since it knows that its message frame is not the highest priority message frame currently being transmitted in the system. Because identifiers are deemed unique within the system, a station transmitting the last bit of the identifier without detecting a collision must be transmitting the highest priority queued message frame, and hence can start transmitting the body of the message frame.

The CAN message frame format contains 47 bits of protocol control information (the identifier, CRC data, acknowledgement and synchronization bits, etc.). The data transmission uses a bit stuffing protocol which inserts a stuff bit after five consecutive bits of the same value. The frame format is specified such that only 34 of the 47 control bits are subject to bit stuffing. Hence, the maximum number of stuff bits in a message frame with n bytes of data is $\lfloor\frac{(n*8+34-1)}{4}\rfloor$ (since the worst case bit pattern is '0000011110000...'). This means that a message frame is transmitted with between 0 and 24 stuff bits. Hence, the size of a transmitted CAN message frame, denoted by f, is between 47 and 135 bits:

$$f = (n * 8 + 47 + \lfloor\frac{(n * 8 + 34 - 1)}{4}\rfloor) \qquad (2)$$

where n is the number of data bytes.

4.1 Response Time Analysis of CAN

In [20] the authors present analysis to calculate the worst-case latencies of CAN messages. This analysis is based on the standard fixed priority response time analysis for CPU scheduling proposed by [2], and later refined by [10]. Calculating the response times requires a bounded worst case queuing pattern of messages. The standard way of expressing this is to assume a set of traffic streams, each generating messages with a fixed priority. The worst case behavior of each stream is to periodically queue messages. In analogy with CPU scheduling, we obtain a model with a set of messages (corresponding to CPU tasks).

For an ideal CAN controller (the non-ideal case is presented by [21]) the worst-case latency R_i of a CAN message M_i is defined by

$$R_i = J_i + q_i + C_i \tag{3}$$

where J_i is the queuing jitter of message M_i, inherited from the sender task which queues the message. We have assumed that the minimum delay from the point in time t, relative to the time message M_i is queued, is 0 (t is typically the start of the period). In other cases we need to add a term $J_i^{smallest}$ to Equation 3, since jitter is defined as the difference between the biggest and smallest delay from t. The worst-case queuing delay q_i is given by,

$$q_i = B_i + \sum_{j \in hp(i)} \left\lceil \frac{q_i + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{4}$$

where B_i, in the general case, is either the non-preemptive transmission of a lower priority message frame, or the non-preemptive transmission of a message frame belonging to the previous instance of the message M_i [10]. When using the system model presented in this paper, this is equivalent to the worst-case blocking time of the longest possible message frame (i.e., the worst-case transmission time of a CAN message frame with 8 bytes of data and worst-case bit stuffing). Moreover, $hp(i)$ is the set of messages with priorities higher than that of M_i, J_j is the queuing jitter of message M_j, and τ_{bit} caters for the difference in arbitration start times at the different nodes, due to propagation delays and protocol tolerances.

In [18] the authors extended the above analysis and presented an approach to schedule messages in a fault-tolerant manner using fixed priority scheduling (FPS). Broster et. al. [5] addressed the reliability of message transmission on CAN assuming probabilistic fault models. Bartolini et. al. [4] presented an approach to reduce the response time of multi-frame messages in CAN by using the Priority Inheritance Protocol. Our work extends the existing approaches by providing a more generalized error model as well as incorporating probabilistic schedulabiliy analysis.

4.2 Error Handling Features in CAN

In CAN, errors may occur due to different sampling points or switching thresholds in different nodes, or due to signal dispersion during propagation. To handle these, the CAN protocol provides elaborate error detection and self-checking mechanisms as presented by [9], specified in the data link layer of [13]. The error detection is achieved by means of transmitter-based-monitoring, bit stuffing, Cyclic Redundancy Check (CRC) message frame format check, and frame acknowledgment.

To make sure that all nodes have a consistent view, errors detected in one node must be globalized. This is achieved by allowing the detecting node to transmit an error flag containing 6 bits of same polarity. Upon reception of an error frame, each node will discard the erroneous message, which then will be automatically re-transmitted by the sender. Note that, the re-transmitted message could be subjected to arbitration during re-transmission. This implies that if any higher priority messages gets queued during the transmission and error signaling of the current message, then those messages will be transmitted before the erroneous message is re-transmitted.

Specification documents of CAN claim that the error detection mechanisms can detect and globalize all transmitter errors. Bursts are guaranteed to be detected on the receiver side up to a length of 15 (which is equal to the degree of f(x) in CRC sequence). Most longer error bursts are also detected. Even though there is a positive probability for undetected errors, we shall assume that all errors are detected. The probability for undetected errors is negligibly small, as indicated by the following quote from the CAN specification documents: "with an operating time of eight hours per day on 365 days per year and an error rate of 0.7 s, one undetected error occurs every thousand years (statistical average)".

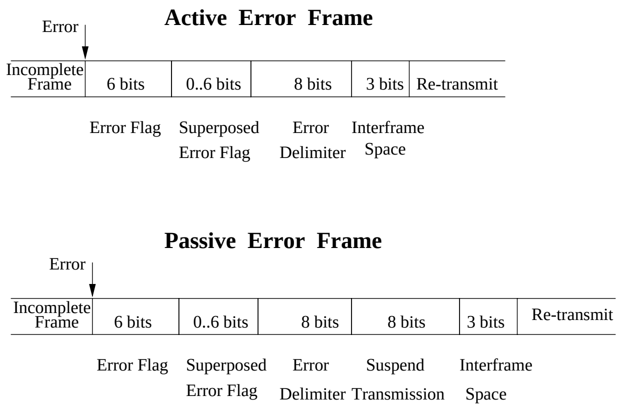


Figure 1: Error frame formats in CAN

Error signaling is done with an error frame that is between 17 to 31 bits long. Figure 1, shows formats of the CAN error frames (details are given in [1]). The error flags are different depending on whether the node that signals the error is in the error active or the error passive state, since a node in the error passive state is not guaranteed to be able to ensure globalization of errors. Note that this is very important, since it may cause inconsistencies. This happens when some nodes in the passive state decide that a frame has errors and, since they are unable to enforce the error detection by the remaining active nodes, which eventually accept the affected frame. As a result, some nodes receive a frame that is not received by others. This lack of consistency is a real problem in distributed systems and hence it is advisable to change a node to bus-off before it gets into the error passive state (when the error warning notification is issued). In this way a node is either ensuring consistency or disconnected, but not causing undetected inconsistencies.

Going into the details, a node in the passive state signals errors with passive error flags consisting of six consecutive recessive bits. However, it should be noted that only if the node is the transmitter of the original data frame, the structure of the error frames corresponds to what is shown in Figure 1 as passive error frame. This is so because only in that case the transmission of a passive error flag can force the other nodes to detect additional errors. Moreover, even if the transmitter is the first to detect an error, if the detection occurs in the EOF field, the transmission of the passive error flag will not be detected by the other nodes and no globalization will be achieved. In any case, it is true that if the passive error flag is detected by all nodes, what is shown in Figure 1 would be the maximum interference caused by the errors. Furthermore, suspend transmission only causes the corresponding additional delay in case the error passive transmitter of the frame affected by the errors is the one that is going to win the arbitration for the next message (because only said transmitter suspends transmission not the other nodes). However again adding suspend transmission to the error flag though it corresponds to the worst case, is an additional source of pessimism.

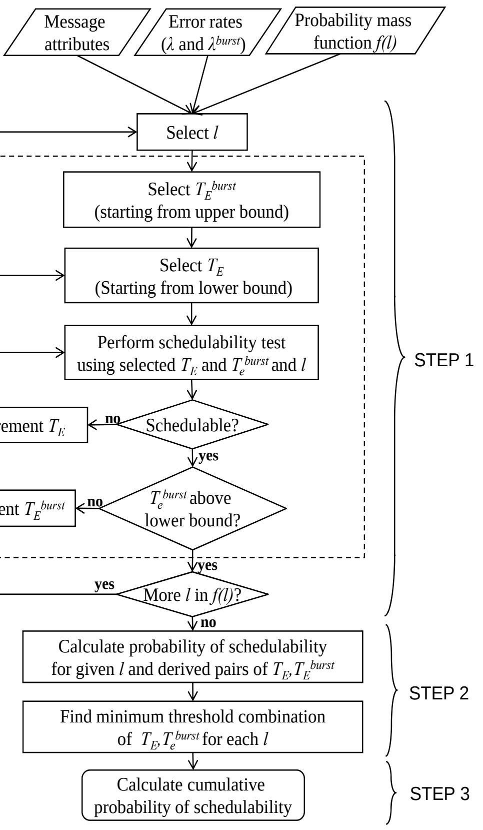
We assume that all nodes are in the error active state (and then only consider active error frames) because when a significant number of errors are detected (but lower than the number required to go to error passive) nodes are directly sent to the bus-off state in order to prevent the potential inconsistencies that such state may cause. At the end real-time systems must be reliable (even more when proposing fault-tolerant scheduling) and message inconsistencies are a fundamental source of unreliability in distributed systems (e.g. replica non-determinism). However, mechanisms for taking care of those inconsistencies are beyond the scope of this paper and we assume that the maximum delays caused by the signalling of errors are those shown in Figure 1.

5. METHODOLOGY

Our ultimate goal is to find the probability that the message set is schedulable. Our methodology for achieving this goal is outlined in the following steps, and illustrated in Figure 2. In this paper we assume that lower bound for the the error burst minimum interarrival time, T_E, as well as both the lower and upper bounds for the minimum interarrival time between errors within a burst, T_E^{burst} are known.

STEP 1: Sensitivity analyses: In this step, a series of sensitivity analyses are performed for each l in the probability mass function $f(l)$ in order to derive combinations of the minimum interarrival times of error bursts (T_E) and the minimum interarrival times of errors under error bursts (T_E^{burst}) that renders the message set schedulable. The schedulability test proposed in this paper is used as a tool for performing these sensitivity analyses.

STEP 2: Probability calculations of the shortest error minimum interarrival times: This step involves the usage of statistical approaches to find the probability of the errors occurring with interarrival times larger than or equal to the T_E and T_E^{burst} thresholds from the λ, and λ^{burst} values together with the mission time L. The T_E and T_E^{burst} threshold combination that gives the largest probability of having no anomalies is defined as the *minimum threshold combination.*



Message attributes
Error rates (λ and λ^{burst})
Probability mass function $f(l)$
Select l
Sensitivity Analysis
Select T_E^{burst} (starting from upper bound)
Select T_E (Starting from lower bound)
Perform schedulability test using selected T_E and T_e^{burst} and l
Increment T_E
no
Schedulable?
yes
Decrement T_E^{burst}
no
T_e^{burst} above lower bound?
yes
yes
More l in $f(l)$?
no
Calculate probability of schedulability for given l and derived pairs of T_E, T_E^{burst}
Find minimum threshold combination of T_E, T_e^{burst} for each l
Calculate cumulative probability of schedulability
STEP 1
STEP 2
STEP 3

Figure 2: Methodology overview

STEP 3: Calculation of the cumulative probability of schedulability: Finally, the probabilities of having no anomalies under the minimum threshold combination for each discrete burst length l, and the probabilities of the burst lengths are used to derive the cumulative probability of schedulability.

In the scope of this paper, we present a schedulability analysis under error bursts which is the main tool to perform the sensitivity analyses mentioned in the first step above.

5.1 Response Time Analysis under Error Bursts

The response time analysis given in this section will show us if the message set is schedulable under a combination of error interarrival time thresholds (minimum interarrival time of independent errors T_E and errors within a burst T_E^{burst}) and a burst length (l).

The worst-case response-time calculations will differ in the following scenarios depending on the relationship between the error burst length l, minimum interarrival time of the independent errors T_E and the message periods. Figure 3 shows the threshold between different scenarios based in the relationship between l and T_E. In case l exceeds the threshold, no frame can be guaranteed to be successfully transmitted before its deadline.

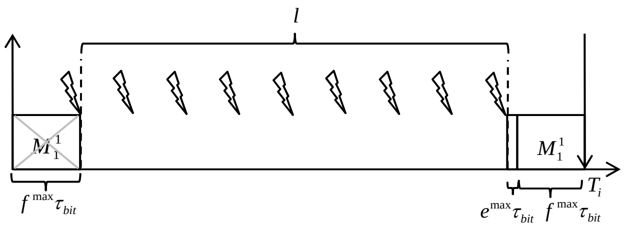


Figure 3: Interplay between l and T_i

SCENARIO 1: $l < T_E$ *and* $l \leq T_i - (e^{max} + 2f^{max})\tau_{bit}$; The burst length is less than the minimum interarrival time of the independent errors, and *at least one* message frame can be successfully transmitted before its deadline.

SCENARIO 2: $l < T_E$ *and* $l > T_i - (e^{max} + 2f^{max})\tau_{bit}$; Burst length is less than the minimum interarrival time of the independent errors, but the error burst length exceeds the threshold, such that no frame can be guaranteed a successful transmission before its deadline.

SCENARIO 3: $l \geq T_E$; Burst length is greater than or equal to the minimum interarrival time of the independent errors. In this case, a new error burst can occur before the current burst finishes, therefore the worst case response time analysis assumes that an error burst can affect the transmission of the message instances from the start of the queueing to the completion of the message transmission.

In these scenarios the worst-case response time calculations are similar to response time analysis of CAN under periodic messages and sporadic faults introduced by Tindell et al. [20], where an additional term, E_i, for the maximum error interference is added to Equation 4:

$$q_i = E_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{q_i + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{5}$$

In Scenario 1 error bursts can affect only parts of the response time. A simple example is shown in Figure 4 where two bursts occur with a separation of T_E and three errors occur with a separation of T_E^{burst} during each burst.

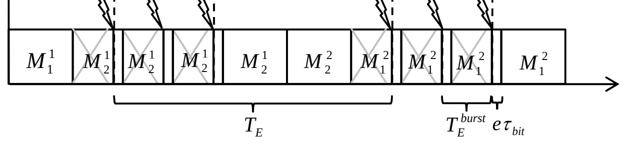


Figure 4: FT execution under error bursts

In this Scenario, the worst-case response-time calculations will differ also in the following cases depending on the minimum interarrival time of the errors within an error burst T_E^{burst}:

CASE 1: $T_E^{burst} < (e^{max} + f^{max})\tau_{bit}$: In this case, if the errors within an error burst occur with a separation of T_E^{burst}, it may not be possible to transmit any frame between any two consecutive errors during the burst (Figure 5). Therefore, the worst case error overhead E_i in Equation 5 becomes:

$$E_i = \left\lceil \frac{q_i + C_i}{T_E} \right\rceil ((f^{max} + e^{max})\tau_{bit} + l) \tag{6}$$

The left hand product term of Equation 6 gives the maximum number of error bursts that can occur until the end of the message transmission (i e., the message response time). The right hand product term includes the transmission time of the largest frame in the worst case scenario, i e., when the first error in the burst hits its last bit. The other components of the right hand product term are the transmission time of the largest error frame and the whole length of the error burst, since in the worst case, no frame can be transmitted during this time. Figure 5 shows an example scenario in Case 1. The largest message frame and the largest error frame in Equation 6 are the frames before and after the error burst respectively.

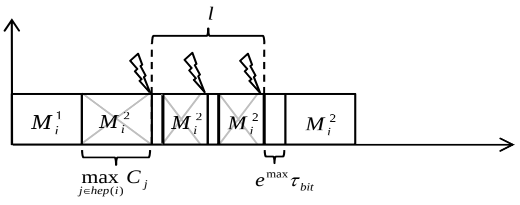


Figure 5: Worst case error overhead in Case 1

CASE 2: $T_E^{burst} \geq (e^{max} + f^{max})\tau_{bit}$: In this case, one or more frames can successfully be transmitted between two errors within an error burst. Therefore only certain sections of the error burst length contribute to the error induced overhead. The worst case overhead due to error bursts, E_i, in this case, is given by:

$$E_i = \left\lceil \frac{q_i + C_i}{T_E} \right\rceil ((f^{max} + e^{max})\tau_{bit} + p) \tag{7}$$

The left hand product term of Equation 7 similarly gives the maximum number of error bursts that can occur until the end of the message transmission. The right hand product term includes the transmission time of the largest frame, the largest error frame, and the error overhead (denoted by p) during l. Note that in this case, the error overhead p is strictly less than the burst length l.

$$p = \left\lfloor \frac{l}{T_E^{burst}} \right\rfloor (e^{max}\tau_{bit} + r) \tag{8}$$

The left hand product term of Equation 8 gives the maximum number of errors that can occur during an error burst, minus the last error (which only contributes to the error overhead E_i with its corresponding error frame). The right hand product term gives the worst case error overhead caused by one error during the burst, and includes the transmission time of the largest error frame, as well as the frame hit by the next error, and denoted by r (e g., M_2^3 in Figure 6. In our calculations we assume that all the successfully transmitted frames between two errors are of the maximum frame size.

$$r = (T_E^{burst} - e^{max}\tau_{bit})(mod\ f^{max}\tau_{bit}) \tag{9}$$

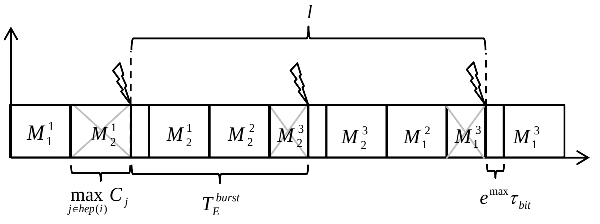


Figure 6: Worst case error overhead in Case 2

Figure 6 shows an example scenario in Case 2. The largest message frame and the largest error frame in Equation 7 are the frames before and after the error burst respectively. The error frames and the remainders in Equation 8 are the ones that come after the first two errors in the error burst.

In Scenarios 2 and 3, the maximum number of errors that can occur from the time that the first message frame is queued to the completion time of the last message frame is given by:

$$\left\lceil \frac{q_i + C_i}{T_E^{burst}} \right\rceil$$

The same logic behind using Equation 9 applies here as well. Therefore, except for the first error, the worst case overhead induced by each remaining error is equal to $e^{max}\tau_{bit} + r$ where r is given in Equation 9. Hence,

$$E_i = (f^{max} + e^{max})\tau_{bit} \left\lfloor \frac{q_i + C_i}{T_E^{burst}} \right\rfloor (e^{max}\tau_{bit} + r) \tag{10}$$

5.2 Probabilistic Schedulability Bounds

In this paper, we make an assumption similar to the one in [6], which states that, during a mission time, if the actual shortest time interval between any two errors is less than the assumed minimum interarrival time, then the system is unschedulable. In detail, for the two cases described in the previous section, we assume:

1. Case 1 ($T_E^{burst} < (e^{max} + f^{max})\tau_{bit}$): If the actual shortest time interval between any two error bursts W, during mission time L, is less than the assumed minimum interarrival time T_E, then the message set is unschedulable. In this case, the probability of unschedulability is:

$$Pr(U) \equiv Pr(W < T_E)$$

2. Case 2 ($T_E^{burst} \geq (e^{max} + f^{max})\tau_{bit}$): If the actual shortest time interval between any two error bursts W, during mission time L, is less than the assumed minimum interarrival time between two error bursts T_E, or if the actual shortest time interval between any two errors within a burst W^{burst}, during all the bursts with length l, within mission time L, is less than the assumed minimum interarrival time between errors within a burst T_E^{burst}, then the message set is unschedulable. In this case, the probability of unschedulability is:

$$Pr(U) \equiv Pr((W < T_E)\ or\ (W^{burst} < T_E^{burst}))$$

Assuming that L is an even integer multiple of T_E, Burns et al., [6], presented the upper bound for $Pr(W < T_E)$ as shown below:

THEOREM 1. *If $L/(2T_E)$ is a positive integer then*

$$Pr(W < T_E) <$$

$$1 + [e^{-\lambda T_E}(1 + \lambda T_E)]^{\frac{L}{T_E}-1} - 2[e^{-2\lambda T_E}(1 + 2\lambda T_E)]^{\frac{L}{2T_E}} \tag{11}$$

PROOF. Presented in [6] □

Assuming that L is an even integer multiple of T_E, and l is an even integer multiple of T_E^{burst}, we present the upper bound for $Pr((W < T_E)\ or\ (W^{burst} < T_E^{burst}))$ as follows:

THEOREM 2. *If $L/(2T_E)$ and $l/(2T_E^{burst})$ are both positive integers then*

$$Pr((W < T_E)\ or\ (W^{burst} < T_E^{burst})) <$$

$$1 + [e^{-\lambda T_E}(1 + \lambda T_E)]^{\frac{L}{T_E}-1} - 2[e^{-2\lambda T_E}(1 + 2\lambda T_E)]^{\frac{L}{2T_E}}$$

$$+1 + [e^{-\lambda T_E^{burst}}(1 + \lambda T_E^{burst})]^{\frac{l\left\lceil\frac{L}{T_E}\right\rceil}{T_E^{burst}}-1}$$

$$-2[e^{-2\lambda T_E^{burst}}(1 + 2\lambda T_E^{burst})]^{\frac{l\left\lceil\frac{L}{T_E}\right\rceil}{2T_E^{burst}}} \tag{12}$$

PROOF. In the proposed approach we assume that the cause(s) of two bursts occurring within a time interval shorter than T_E and the cause(s) resulting in two errors occurring within a time interval shorter than T_E^{burst} are independent of each other. For example, driving a car near police stations or airports, and thus exposing it to electromagnetic interference (EMI), is the cause of having error bursts. On the other hand, the intensity of the EMI, as well as the distance to the source, are the factors that contribute to potential occurrence of errors within a burst. Thus, the logical "OR" in the probability function is represented by the addition operator:

$$Pr((W < T_E) \ or \ (W^{burst} < T_E^{burst})) \equiv$$
$$Pr(W < T_E) + Pr(W^{burst} < T_E^{burst}) \qquad (13)$$

While the representation of the first term in the probability function, i.e., $Pr(W < T_E)$, is formalized in [6], we focus on the derivation the second term, i.e., the probability of mishap caused by errors occurring within a time interval shorter than T_E^{burst}: $Pr(W^{burst} < T_E^{burst})$. The proof is similar to the one of Theorem 1, with the following differences:

- The minimum interarrival time between bursts T_E is substituted with the minimum interarrival time between errors *within* bursts T_E^{burst}
- The mission time L is substituted with the cumulative length of the maximum number of bursts $(\left\lceil \frac{L}{T_E} \right\rceil)$ of length l that can occur within L: $l * \left\lceil \frac{L}{T_E} \right\rceil$

Hence, upper bound for the probability $Pr(W^{burst} < T_E^{burst})$ is represented as:

$$Pr(W^{burst} < T_E^{burst}) < 1 + [e^{-\lambda T_E^{burst}}(1+\lambda T_E^{burst})]^{\frac{l\left\lceil \frac{L}{T_E}\right\rceil}{T_E^{burst}}-1}$$
$$-2[e^{-2\lambda T_E^{burst}}(1+2\lambda T_E^{burst})]^{\frac{l\left\lceil \frac{L}{T_E}\right\rceil}{2T_E^{burst}}} \qquad (14)$$

Combining 11, 13 and 14, the theorem is proved. □

5.3 Illustrative Example

We consider a distributed embedded system where 4 devices exchange messages over a CAN network. The message set consists of 4 messages (1 message per device) as shown in Table 1 where columns P, N, T and D represent the priority, number of message frames, period and the deadline respectively. Priorities are ordered from 1 to 4 where 4 is the lowest priority. The time unit is *milliseconds*. Each message con-

Task	P	N	T	D
A	1	8	4	4
B	2	12	4	4
C	3	16	12	12
D	4	12	16	16

Table 1: Example message set

sists of 8 data bytes, hence $f^{max} = 135$ and $e^{max} = 31$ bits. We assume a mission time of 1 hour ($L = 1h$), a bus speed of 1Mbit/s ($\tau_{bit} = 1\mu s$), and a discrete probability distribution for burst length l as shown in Figure 7. Expected number of error bursts and errors within a burst in unit time are assumed to be $\lambda = 10^{-1}$ and $\lambda^{burst} = 10^2$ respectively.

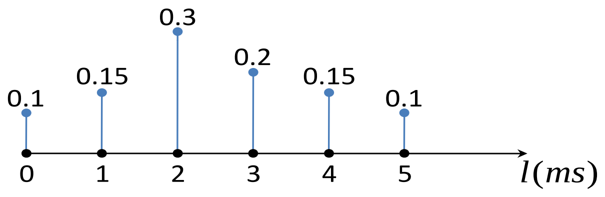


Figure 7: Probability mass function $f(l)$

To find the probability of schedulability under error bursts, we first used the proposed schedulability test to perform a series of sensitivity analyses for each error burst length l in the probability mass function $f(l)$. As we are assuming that l is an even integer multiple of T_E^{burst}, for each l, we started from the largest T_E^{burst} that satisfied the assumed condition ($T_E^{burst} = \frac{l}{2}$) and reduced it until we reached the T_E^{burst} value that also satisfied the condition for Case 2 in Subsection 5.1 ($T_E^{burst} < (e^{max} + f^{max})\tau_{bit}$). Then, for each T_E^{burst} for the given l, we found the minimum T_E at which the message set is guaranteed to be schedulable. As a result, the analyses gave us a number of T_E and T_E^{burst} combinations for each error burst length l. We used the statistical formulas presented in Subsection 5.2 and derived the probabilities of unschedulability for each T_E and T_E^{burst} combination as shown in Table 2.

T_E	T_E^{burst}	l	$Pr(U)$
1.501	0	0	6.2542x10^{-9}
3.4	0.25	0.5	1.5319x10^{-4}
6.674	0.125	0.5	2.7808x10^{-8}
3.36	0.5	1	6.1989x10^{-4}
6.634	0.25	1	1.5704x10^{-4}
13.181	0.125	1	5.4921x10^{-8}
2.719	0.75	1.5	1.7228x10^{-3}
6.594	0.375	1.5	3.5541x10^{-4}
6.594	0.1875	1.5	1.7773x10^{-4}
-	0.0937	1.5	1
2.679	1	2	3.1067x10^{-3}
6.554	0.5	2	6.3560x10^{-4}
13.101	0.25	2	1.5906x10^{-4}
-	0.125	2	1
2.5	1.25	2.5	5.1975x10^{-3}
4.511	0.675	2.5	1.5577x10^{-3}
4.511	0.3125	2.5	7.2142x10^{-4}
-	0.1577	2.5	1
4.471	1.5	3	4.1866x10^{-3}
4.471	0.75	3	2.0951x10^{-3}
13.021	0.375	3	3.5999x10^{-4}
13.021	0.1875	3	1.8004x10^{-4}
-	0.0937	3	1

Table 2: Probabilities of unschedulability

Finally we selected the *minimum threshold combinations* that gave the highest probabilities of schedulability $P(S) = 1 - P(U)$, and based on these probabilities as well as the probability of each l extracted from $f(l)$, we calculated the cumulative probability of schedulability $P(S)^{cumulative}$ as shown on Figure 3.

	l	$f(l)$	$P(S)$
	0	0.1	0.99999999374583
	0.5	0.15	0.99999997219166
	1	0.25	0.99999994507913
	1.5	0.2	0.99982226780869
	2	0.15	0.9998409355277
	2.5	0.1	0.99927857698501
	3	0.05	0.99981996174267
$P(S)^{cumulative}$			0.99985943114964

Table 3: Minimum threshold combinations

This analysis showed that the example message set is schedulable with a probability of 0.99985943114964 during a 1 hour mission where $\lambda = 10^{-1}$, $\lambda^{burst} = 10^2$ and burst length probabilities are as given by the $f(l)$.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a methodology which enables the provision of probabilistic real-time guarantees in distributed real-time systems under error bursts. The proposed approach introduces a comprehensive probabilistic error model that has the capability of modeling independent errors as well as errors within a burst, together with an appropriate schedulability analysis for the particular case of real-time message scheduling on CAN. The fault tolerance technique considered in this paper is redundancy in the temporal domain as it is the often preferred method in many dependable embedded applications to recover from the most common transient and intermittent errors.

Our ongoing research includes consideration of multiple criticality levels of messages for efficient usage of resources.

7. REFERENCES

[1] CAN in Automation, CAN Specifications. *http://www.can-cia.org.*

[2] N. C. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[3] A. Avizienis, J. Laprie, and B. Randell. Fundamental Concepts of Dependability. *Research Report N01145, LAAS-CNRS*, 2001.

[4] C. Bartolini, G. Lipari, and L. Almeida. Using Priority Inheritance Techniques to Override the Size Limit of CAN Messages. 7th *IFAC International Conference of Fieldbuses and Networks in Industrial and Embedded Systems (FET)*, 2007.

[5] I. Broster. Flexibility in Dependable Real-time Communication. *Department of Computer Science, University of York*, 2003.

[6] A. Burns, S. Punnekkat, L. Strigini, and D. Wright. Probabilistic Scheduling Guarantees for Fault-Tolerant Real-Time Systems. *Dependable Computing for Critical Applications 7, 1999*, pages 361–378, Nov 1999.

[7] H. Burton and D. Sullivan. Errors and Error Control. *Proceedings of the IEEE*, pages 1293–1301, 1972.

[8] L. Casparsson, A. Rajnak, K. Tindell, and P. Malmberg. Volcano - a Revolution in On-board Communication. *Volvo Technology Report 98-12-10*, 1998.

[9] J. Charzinski. Performance of the Error Detection Mechanisms in CAN. *1st International CAN Conference, Mainz*, September 1994.

[10] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien. Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised. *Real-Time Systems*, 35(3):239–272, 2007.

[11] M. Di Natale. Scheduling the CAN Bus with Earliest Deadline Techniques. *IEEE Real-Time Systems Symposium*, pages 259–268, November 2000.

[12] J. Ferreira. An Experiment to Assess Bit Error Rate in CAN. *3rd International Workshop of Real-Time Networks*, pages 15–18, 2004.

[13] ISO-11898. Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for High Speed Communication. 1993.

[14] J.-C. Laprie. Dependable Computing and Fault-Tolerance: Concepts and Terminology. *International Symposium on Fault-Tolerant Computing, ' Highlights from Twenty-Five Years'.*, 1995.

[15] N. Navet. Controller Area Networks: CAN's use within Automobiles. *IEEE Potentials*, pages 12–14, October/November 1998.

[16] I. Noble. EMC and the Automotive Industry. *IEE Electronics & Communication Engineering Journal*, pages 263–271, October 1992.

[17] M. Pizza, L. Strigini, A. Bondavalli, and F. D. Giandomenico. Optimal Discrimination between Transient and Permanent Faults. *3rd IEEE International Symposium on High-Assurance Systems Engineering*, pages 214–223, 1998.

[18] S. Punnekkat, H. Hansson, and C. Norström. Response Time Analysis under Errors for CAN. 6th *IEEE Real-Time Technology and Applications Symposium*, May-June 2000.

[19] J. Rufino, P. Verissimo, G. Arroz, C. Almeida, and L. Rodrigues. Fault-Tolerant Broadcasts in CAN. *Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing, 1998. Digest of Papers.*, pages 150–159, 1998.

[20] K. Tindell, A. Burns, and A. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 3:1163–1169, 1995.

[21] K. W. Tindell, A. H. Hansson, and A. J. Wellings. Analysing Real-Time Communications: Controller Area Network (CAN). *IEEE Real-Time Systems Symposium*, pages 259–265, December 1994.