Fault Confinement Mechanisms on CAN : Analysis and Improvements

Bruno Gaujal and Nicolas Navet

***Abstract*— The CAN protocol possesses fault confinement mechanisms aimed at differentiating between short disturbances caused by electromagnetic interferences (EMI) and permanent failures due to hardware dysfunctioning. In this study, we derive a Markovian analysis of these mechanisms which enable to assess the risk of reaching one of the two degraded modes bus-off and error-passive defined by CAN. We identify several problems with the existing mechanisms, the major one being that the bus-off state is reached too easily. In particular it happens with bursts of EMI causing several consecutive transmission errors. We propose new mechanisms that address these drawbacks. The basic idea is to weigh the progression towards the degraded mode by the quantity of information given by the last transmission. In our experiments, these mechanisms proved to be effective: the hitting time of bus-off for non-faulty nodes increases hugely while faulty systems reach bus-off in the same amount of time. In the last part of the paper, implementation issues are discussed and different techniques for tuning the parameters of the algorithm are provided, either off-line or at run-time.**

***Index Terms*— Real-Time Systems, Fault Tolerance, Fault Confinement, Controller Area Network, Electromagnetic Interferences.**

I. Introduction

CAN (Controller Area Network) is a broadcast bus with priority based access to the medium which has become a de-facto standard for data transmission in automotive applications. On a CAN network nodes do not possess an address and no single node plays a preponderant role in the protocol. Each message has an identifier, unique to the whole system, that serves two purposes : assigning a priority for the transmission (the lower the numerical value, the greater the priority) and identifying the message for filtering upon reception. Data, possibly segmented in several frames, may be transmitted periodically, sporadically or on-demand. A minimal CAN communication profile consists of a three-layered architecture : physical layer, Data-Link Layer (DLL) and application layer. The DLL is implanted in an electronic component called a CAN controller. The ISO standards ([1] and [2]) only define the physical layer and DLL, but proposals have been made for the application layer (CAN Application Layer - CAL see [3]) or for complete profiles based on the two normalized layers (Smart Distributed Systems - SDS see [4], DeviceNet see [5] or CANopen which uses a subset of CAL see [6]).

CAN has efficient error detection mechanisms. In [7], the authors have shown the probability of undetected transmission errors during the lifetime of a vehicle to be extremely low, that is why we will further assume that all errors are correctly detected. Each station which detects an error sends an ”error flag” which is a particular frame composed of 6 consecutive dominant bits (in CAN's terminology, the dominant bit value is ”0” while ”1” is said the recessive bit value) that enables all the stations on the bus to be aware of the transmission error. The corrupted frame automatically re-enters into the next arbitration phase, which can lead to missed deadlines. The error recovery time, defined as the time from detecting an error until the possible start of a new frame, is 17 to 31 bit times (where the bit time is the time between the emission of two successive bits of the same frame).

To prevent a defective node from perturbing the functioning of the whole system (for instance by repetitively sending the so-called error frames that signal transmission errors) the CAN protocol uses fault confinement mechanisms. Their objectives are (1) to detect permanent hardware dysfunctioning and (2) to switch off defective nodes. The detailed functioning scheme of these mechanisms is described in Section II.

CAN fault confinement mechanisms are interesting features from the dependability point of view but their counterpart is that a good-functioning node may become error passive, or worse, may be bus-off just because of transmission errors. This is particularly a problem for in-vehicle networks where EMI might be very important : Bit Error Rate at order of magnitude of 10^{-3} are possible during short periods of time for instance when the vehicle is close to a high-power Radio Frequency transmitter or close to a high-voltage power supply.

Several studies were conducted to assess the impact of transmission errors on the respect of message real-time constraints on a CAN bus. In [8], [9], Tindell et al. have proposed a response time analysis that takes into account the possibility that transmission errors can occur. Their error model is deterministic in the sense that it assumes that the number of errors during any time interval can be bounded. In [10] and [11], a probabilistic fault model that can model single-bit faults as well as burst errors is adopted and it is used for analytically evaluating the probability that a message fails to meet its deadline. This approach has been made less pessimistic in [12]. To our best knowledge, no probabilistic analysis of CAN's fault confinement mechanisms has been done yet.

In Section II, CAN's fault confinement mechanisms are described. A Markovian analysis of the bus-off and error-passive hitting times is given in Section III and IV. The Section V is devoted to the proposed new fault confinement mechanisms and to the evaluation of their performances.

B. Gaujal is with the ID Laboratory, INRIA, Ensimag - Zirst 51, avenue Jean Kuntzmann, 38330 Montbonnot, France (e-mail: bruno.gaujal@imag.fr).

N. Navet is with the LORIA laboratory, INRIA, Campus Scientifique - B.P. 239, 54506 Vandoeuvre-lès-Nancy, France (e-mail: nicolas.navet@loria.fr).

Finally, implementation issues on existing hardware are addressed in Section VI.

II. CAN's fault confinement mechanisms

A CAN controller of each station possesses 2 distinct error counters :

- the Transmit Error Counter (TEC) which counts the number of transmission errors detected on the frames sent by the station,
- the Receive Error Counter (REC) which counts the number of transmission errors detected on the frames received by the station.

Each time a frame is received or transmitted correctly by a station, the value of the corresponding counter is decreased (except when its value is already zero). Similarly, each time a transmission error is detected, the value of the corresponding counter is increased. Depending on the value of both counters, the station will be in one of the 3 states defined by the protocol :

- *Error Active* (REC<128 and TEC<128) : this is the normal operating mode, the station can normally send and receive frames. This is the default state at controller initialization.
- *Error Passive* ((REC>127 or TEC>127) and TEC≤255 : the station may emit but it must wait 8 additional bits after the end of the last transmitted frame. Therefore the frames sent by the station are no longer certain to meet the worst-case response times computed through schedulability analysis.
- *Bus-off* (TEC>255) : The station is automatically switched off from the bus. In this state, the station can neither send nor receive frames. A node can leave the bus-off state after a hardware or software reset (*normal mode request*) and after having successfully monitored 128 occurrences of 11 consecutive recessive bits (a sequence of 11 consecutive recessive bits corresponding to the ACK, EOF and the intermission field of a data frame that has not been corrupted).

The rules for increasing and decreasing the TEC and the REC of a station are somewhat complex, see [1] pp 48-49. In the rest of the article, we will assume that no errors occur during the signalling of an error (no bit error in an active error flag). Furthermore, we will not consider three exceptions to the general rules listed below (see [1] pp 48-49, exceptions listed in points b) and c)).

Under these assumptions, the rules for modifying the counter value of the stations become :

1) Frame transmission successful. If the node is not the sending node : if the REC is between 1 and 127, then it is decreased by one. If the REC's value is nil, it stays unchanged. Finally, if its value is greater than 127, it randomly takes a value between 119 and 127. If the node is the sending node : if the TEC is not nil, it is decreased by one, otherwise it remains unchanged.
2) Unsuccessful transmission (transmission error detected). If the node is not the sending node : The REC is increased by one. If the node is the sending node: the TEC is increased by 8.

Whatever the result of a transmission, at most one counter is modified on a given station.

III. Bus-off hitting time

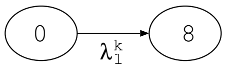
CAN fault confinement mechanisms are conceived to disconnect defective nodes from the network and prevent them from perturbing the whole network. However, under severe electro-magnetic interference conditions, one or several nodes can reach the bus-off state just because of transmission errors. It is thus important to estimate the probability of such events which can be achieved through the knowledge of the average hitting time of the bus-off state and of the variance of the bus-off hitting times. For this purpose, we model the Transmit Error Counter (TEC) with a Markov chain in continuous time (also called a Markov process).

A. Modeling

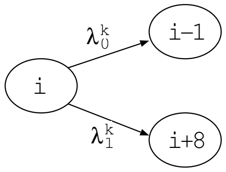
Under the assumption that state changes are exponentially distributed, the evolution of the TEC can be modeled by a Markov process. Let λ_0^k be the rate of transmission of non-corrupted messages for station k and λ_1^k be its rate of corrupted messages.

The general rule is that the TEC value is increased by 8 on the transmitting node if a frame is corrupted and that the TEC is decreased by 1 if the transmission is successful. Nevertheless, different cases have to be distinguished. The infinitesimal generator of the Markov process for the different possible values of the TEC (denoted by i) is given by the following graphs :

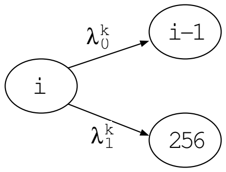
- $i = 0$:



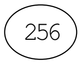
- $i \in \{1..248\}$:



- $i \in \{249..255\}$:



- $i = 256$:



The computation of λ_0^k and λ_1^k is detailed in Appendix II. The state 256, which corresponds to the bus-off state, is a so-called *absorbing* state from which it is impossible to escape and it stops the process. This is exactly the functioning scheme of the CAN protocol. When a station becomes "bus-off", it can

$$Q = \begin{array}{c|ccccccccccc}
 & 0 & 1 & 2 & \ldots & 8 & 9 & .. & 253 & 254 & 255 & 256 \\
\hline
0 & -\lambda_1^k & 0 & 0 & \ldots & \lambda_1^k & 0 & .. & 0 & 0 & 0 & 0 \\
1 & \lambda_0^k & -\lambda^k & 0 & \ldots & 0 & \lambda_1^k & .. & 0 & 0 & 0 & 0 \\
2 & 0 & \lambda_0^k & -\lambda^k & \ldots & 0 & 0 & .. & 0 & 0 & 0 & 0 \\
. & . & . & . & . & . & . & .. & . & . & . & . \\
. & . & . & . & . & . & . & .. & . & . & . & . \\
254 & 0 & 0 & 0 & \ldots & 0 & 0 & .. & \lambda_0^k & -\lambda^k & 0 & \lambda_1^k \\
255 & 0 & 0 & 0 & \ldots & 0 & 0 & .. & 0 & \lambda_0^k & -\lambda^k & \lambda_1^k \\
256 & 0 & 0 & 0 & \ldots & 0 & 0 & .. & 0 & 0 & 0 & 0
\end{array}$$

Fig. 1. Generator matrix of the bus-off stochastic process with $\lambda^k = (\lambda_0^k + \lambda_1^k)$ (the sum of each row of Q is 0).

neither send nor receive frames. With the previously exposed rules, one obtains the generator matrix of size $257 * 257$ (the Markov chain having 257 states) shown in Figure 1.

For convenience, this Markov process will be transformed in the stochastically equivalent discrete time Markov chain termed the *uniformized chain*. Let $q_i = \sum_{j \neq i} Q_{i,j}$ be the total rate out of state i and $q_{max} = \sup_{i \geq 0} q_i$. As $q_{max} < \infty$, one can uniformize the Markov process so that it is equivalent to a Markov chain with kernel P which has the following entries :

$$P_{i,j} = \begin{cases} q_{i,j}/q_{max}, & i \neq j, \\ 1 - q_i/q_{max}, & i = j \end{cases} \tag{1}$$

The steps of the Markov chain correspond to an iid process of duration exponentially distributed with parameter q_{max}. The matrix P under its "canonical form" is given below :

$$P = \begin{bmatrix} \mathcal{Z} & \mathcal{R} \\ 0 & 1 \end{bmatrix} \tag{2}$$

where $\mathcal{Z}$ is the original matrix without the 257^{th} line and the 257^{th} row. All states in $\mathcal{Z}$ are *transient* : starting from such a state, there exists a positive probability that the process may not eventually return to this state. The vector $\mathcal{R}$ is the 257^{th} column vector of P without the 257^{th} element (this latter element being the *absorbing* state that models the "bus-off" state).

One denotes by $\mathcal{T}$ the set of transient states and N_i the random variable which gives the time needed to reach for the first time the absorbing state 256 starting from a given state i. Using a classical "one-step" analysis, one obtains :

$$N_i = \begin{cases} \gamma_i + N_j, & \text{with probability} P_{i,j} \quad j \in \mathcal{T}, \\ \gamma_i, & \text{with probability} P_{i,256} \end{cases} \tag{3}$$

with $\gamma_i = 1$ if $i \neq 256$ or otherwise 0. Taking expectations, one obtains :

$$\begin{aligned} E[N_i] &= P_{i,256} E[\gamma_i] + \sum_{j \in \mathcal{T}} P_{i,j} E[\gamma_i + N_j] \\ &= \gamma_i + \sum_{j \in \mathcal{T}} P_{i,j} E[N_j] \end{aligned} \tag{4}$$

This set of 257 linear equations can easily be solved using any numerical or symbolical computation program such as Maple. $E[N_0]$ is the mean hitting time of the bus-off state for the considered station.

In a similar way, one can compute the variance of the bus-off hitting time which is by definition equal to $V[N_i] = E[N_i^2] - E[N_i]^2$. One has

$$N_i^2 = \begin{cases} (\gamma_i + N_j)^2, & \text{with probability } P_{i,j} \quad j \in \mathcal{T}, \\ \gamma_i^2, & \text{with probability } P_{i,256} \end{cases} \tag{5}$$

Taking expectations :

$$\begin{aligned} E[N_i^2] &= \sum_{j \in \mathcal{T}^C} P_{i,j} E[\gamma_i^2] + \sum_{j \in \mathcal{T}} P_{i,j} E[(\gamma_i + N_j)^2] \\ &= \gamma_i^2 + \sum_{j \in \mathcal{T}} P_{i,j} E[(N_j + \gamma_i)^2] \\ &= \gamma_i + \sum_{j \in \mathcal{T}} P_{i,j} E[N_j^2] + 2 \sum_{j \in \mathcal{T}} P_{i,j} E[N_j] \gamma_i \end{aligned} \tag{6}$$

After having solved this set of 257 linear equations, the variance of the first hitting time of the bus-off state is $V[N_0] = E[N_0^2] - E[N_0]^2$.

B. Numerical applications

To illustrate this analysis, let us consider two CAN nodes which are parts of an experimental embedded CAN-based application proposed by PSA (Peugeot-Citröen Automobiles Company) and described in [11]. Six devices exchange messages on a 250kb/s network : the engine controller, the wheel angle sensor, the AGB (Automatic Gear Box), the ABS (Anti-Blocking System), the bodywork gateway and a device y (the name of this device cannot be communicated because of confidentiality). The two considered nodes are the "engine controller" and the "bodywork network gateway" which respectively send the frames of priority $\{1, 3, 10\}$ and $\{8\}$ of periods $\{10, 20, 100\}$ ms and $\{50\}$ ms respectively. The average size of the frames for the engine controller is 118.75 bits while 105 bits for the bodywork network gateway. The characteristics of the 12 frames composing the application is given in Appendix I.

On Figure 2, one can observe that the average hitting time greatly varies depending on the Bit Error Rate (BER). For instance, it takes on average only about 40 seconds for the engine controller to reach the bus-off state with a BER of 0.001 (corresponding to a frame error rate of 11.17% for the engine controller) and more than 43360 hours with a BER of 0.0007 (to be compared to the expected cumulated utilization time of a vehicle which is about 5000 hours). In addition, the

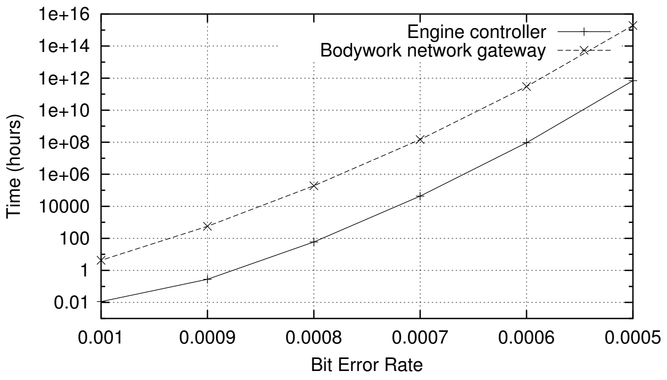


Fig. 2. Average hitting times of the bus-off state for the engine controller and the bodywork network gateway with the Bit Error Rate (BER) varying from 0.0005 to 0.001 .

curves on Figure 2 suggest that the higher the load induced by a station, the faster the station will reach the bus-off state. For instance, the average hitting time of the bodywork network gateway (which generates a nominal load of 0.84% versus 7.6% for the engine controller) is more than 4.3 hours with a BER of 0.001. It is also noteworthy that the standard deviation of the hitting times is very important, it is of the same order of magnitude as the average hitting times which in practice means that there will be a high variability among the observed hitting times. For instance, the standard deviation for the bodywork network gateway is equal to 42.84 hours for a BER of 0.001 while the average hitting time is 43.01 hours.

IV. Error-passive hitting time

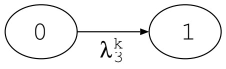
An error passive node is not disconnected from the bus. However, it must wait 8 supplementary bits after the end of the last transmitted frame before sending a frame. This may increase the worst-case response times computed through schedulability analysis. It is thus important for the application designer to assess the probability of such an event.

A station becomes error-passive if the REC is greater than 127 or if the TEC is equal to 128. The modeling through a Markov chain is straightforward : each state of the process can be identified through 2 coordinates (i, j) where for instance i is the value of the TEC and j the value of the REC. To evaluate the probability of being error passive, one just has to compute the time spent in a state such that $i > 127$ or $j = 128$ before the occurrence of ”bus-off”. The number of states of the Markov chain being $257 \cdot 128$, the probability transition matrix is of size $(257 \cdot 128)^2 \approx 1,09 \cdot 10^9$ which is too big to obtain numerical results on desktop workstations. However we can actually estimate separately the time spent in error passive due to the reception (REC$=128$) and the time due to the emission (REC> 127).

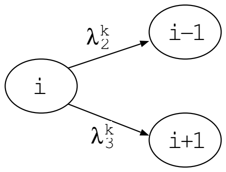
A. Error-passive due to reception

Under the assumption of exponentially distributed state changes, one can model the evolution of the REC through a Markov process. The general rule is that the REC is increased by 1 on the receiving nodes if the frame is corrupted and it is decreased by 1 if the transmission is successful. The infinitesimal generator of the Markov process for the different possible values of the REC (denoted by j) is given by the following graphs :

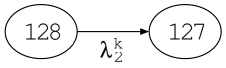
- $j = 0$:



- $j \in \{1..127\}$:



- $j = 128$:



Although the CAN standard [1] permits the REC to exceed 128, it is equivalent to consider its maximum value to be 128. Indeed, if the REC is greater or equal than 127 and a frame is successfully received then the REC is set to a ”value between 119 and 127”. We have chosen 127 which is the choice leading to the most pessimistic results from the point of view of the time spent in error-passive. Denote by λ_2^k the rate of frames successfully received by station k :

$$\lambda_2^k = \sum_{i \neq k} \lambda_0^i, \tag{7}$$

while λ_3^k is the rate of corrupted frames received by station k :

$$\lambda_3^k = \sum_{i \neq k} \lambda_1^i. \tag{8}$$

The Markov process corresponding to the above transitions is then transformed using the uniformization technique described in paragraph III-A in its stochastically equivalent Markov chain whose transition probability matrix is denoted by W. The Markov chain being *ergodic* (all states are positive recurrent, aperiodic and there exists only one communication class in the transition matrix), the stationary probability vector π can be computed :

$$\pi = \pi \cdot W, \tag{9}$$

where π_i (i^{th} component of the vector π) is the proportion of time the Markov chain spends in state i. The time spent in error-passive due to receptions is thus given by π_{128}. With a BER equal to 0.001, we obtain for the engine controller $\pi_{128} = 6.65 \cdot 10^{-131}$, with a BER equal to 0.0005 one has $\pi_{128} = 1.02 \cdot 10^{-170}$. The expected number of steps between successive visits to state 128 is $1/\pi_{128}$ or $(1/\pi_{128}) \cdot (\lambda_2^k + \lambda_3^k)$ seconds. In our example, with a BER of 0.001, the expected time between two occurrences of the error-passive state due to reception is more than 10^{124} years for the engine controller. Furthermore the probability of being in a state larger than 8 is about $7 \cdot 10^{-10}$ in the same example. This is consistent with simulation results were such a state was never reached (see paragraph IV-B). These results show that under realistic bus perturbation level, the time spent in error-passive due to reception is almost nil.

B. Error-passive due to emission

Using the Markov chain that models the evolution of the TEC and whose transition probabilities are given by the matrix P (see equation (1)), one can compute the time spent in a state greater than 127. Let M_i be the random variable which gives the number of steps spent in error-passive due to the TEC before the station enters the bus-off state. Its expectation is :

$$E[M_i] = \gamma_i + \sum P_{i,j} E[M_j], \qquad (10)$$

with $\gamma_i = 1$ if $i \geq 128$ or otherwise 0. As can been seen on Figure 3 the proportion of time spent in error passive might be very important for high BER. For instance, the engine controller spends on average 26.2% of the time in error passive with a BER of 0.001 and 4.1% for a BER of 0.0009. Logically, the lower the load induced by a station, the less important the fraction of time spent in error-passive (e.g. only 2.7% of the time in error-passive for the bodywork network gateway with BER= 0.001). The results of paragraph IV-A induce one to think that a controller almost never reaches error-passive due to reception and thus the time spent in error-passive can be estimated only considering the TEC. To verify the correctness of this statement, we simulated the evolution of the two error counters. Simulation results were collected on 250 runs where a run starts with both counters equal to zero and finishes when the bus-off state is reached. During all simulations, the maximum value of the REC never exceeded 8 before reaching bus-off. In addition, if we compare analytical results (given by equation (10)) that do not consider the REC and simulation results, the difference between simulation and exact analysis is always less than 3.3%. The results of the comparison for various BERs are shown on Figure 4.

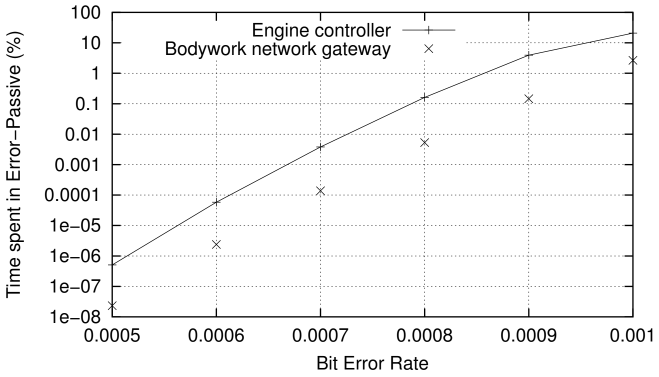


Fig. 3. Average time spent in the error passive state due to transmission for the engine controller and the bodywork network gateway with the Bit Error Rate (BER) varying from 0.0005 to 0.001 .

C. Conclusion on existing mechanisms

Experiments and computations performed under realistic assumptions on the bus perturbation level where all nodes are functioning perfectly (no hardware failure) make us think that the bus-off is reached too easily (e.g. 40 seconds with BER= 0.001). Regarding error-passive, the REC is only useful for nodes that do not emit any messages. As for emitting nodes, as shown in paragraph IV-B, error-passive is almost always reached because of the TEC. Thus, the time spent in error-passive can be estimated by computing the evolution of the TEC. In a strongly disturbed environment, the time spent in error-passive can be very important and therefore the application designers should take into account the degraded temporal behavior of the nodes in this mode.

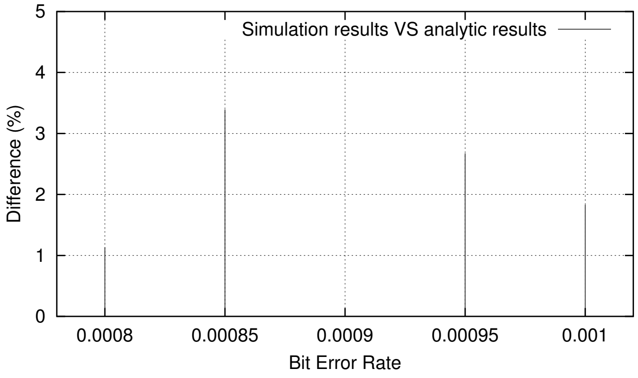


Fig. 4. Difference in percentage between analytical and simulation results regarding the time spent in error-passive. The considered node is the engine controller and the BER ranges from 0.0008 to 0.001 .

V. Improved fault confinement mechanisms

If one analyses the current fault confinement mechanisms, then two issues raise one's attention : first, all transmission errors are assumed to be independent of each other and second, the information given by correct transmissions is barely taken into account for deciding the current state. In this Section, we will provide a new proposal for deciding bus-off under more realistic assumptions :

- Assumption H1) : transmission errors can be correlated. This point is crucial since the arrival process of errors is often bursty especially in the context of in-vehicle embedded applications.
- Assumption H2.a) : faulty nodes cannot send correct frames.
- Assumption H2.b) : faulty nodes may send correct frames (according to an iid process).

Of course H2.a and H2.b are mutually exclusive and will be studied independently.

A station is said to be faulty if it has a hardware problem (e.g. defective wires). We denote by p_{k_i} the probability for the non-faulty station k to emit a frame that will be corrupted given that the last $i-1$ messages (sent by station k) were corrupted. The value of p_{k_i} can be estimated according to statistic measures taken on monitored existing systems as detailed in Section VI.

In the following, the distribution of the burst size (number of consecutive corrupted frames) will be identical for all stations. p_{k_i} will be denoted by p_i when no confusion is possible and it will be given by the modified geometric distribution proposed in [11] :

$$P[\text{error burst length on } k \ \geq i] = \alpha(r^{i-1}(i-r^i)i + r^i) \qquad (11)$$

with the typical parameters $\alpha = 0.1$ and $r = 0.5$.

A. When to decide "bus-off"?

The actual problem that one has to solve is to detect if a node is faulty only by looking at the correctness of the transmitted frames. This immediately raises another issue : when should one take a decision ? The decision will be more pertinent if it is taken after a long time since one gathered more information but on the other hand, if one waits too long, a faulty station, by successive retransmission on the bus, might lead frames of other stations to not respect their deadlines.

Our proposal is that the decision can be delayed until the suspected node might jeopardize the real-time behavior of the other stations. We denote by N_k the maximum number of retransmission of a frame of station k such that the deadlines of all frames of other stations are still respected. It seems natural that our mechanism should decide "bus-off" after N_k consecutive faulty messages. Unfortunately it is not satisfactory because on highly loaded systems where frames have a small laxity, N_k can be very small, for instance lower than 5, and with such a little information the decision to put a node in bus-off state might be wrong. We propose to decide "bus-off" after F_k consecutive faulty messages where

$$F_k = \max\{N_k, \min\{\Phi | \prod_{j=1..\Phi} p_j < \varepsilon\}\} \tag{12}$$

with ε is small enough to be considered neglectable (e.g. 10^{-12}). On highly loaded systems, where messages have a small laxity, N_k might be very small and ε should be large enough in order to keep the number of missed deadlines (of other stations) low. On such systems, transmission errors will necessarily lead some of the frames not to respect their deadline whatever the mechanisms involved. On less constraint systems, N_k will generally be larger than Φ and thus no deadline will be missed. As suggested by an anonymous referee, one can request that, for the most important nodes such as the engine controller, the decision of bus-off is taken after a longer period of time than for less important nodes and the shut-off time can be weighted with some parameter reflecting the importance of the node. This can be done by individualizing for each node the value of ε in Equation 12.

On a CAN bus, a frame m_i can be delayed by the retransmission of a frame m_j only if m_j has a higher priority (denoted $m_j \succ m_i$). To compute N_k, one has to consider the highest priority frame sent by station k since it is the frame that will delay the largest number of frames (line 4 in Figure 5). The maximum overhead induced by each retransmission is not necessarily the size of the highest priority frame since lower priority frames of the same station having a larger size may also be corrupted and delay the other stations. Thus, in the worst case, the overhead per transmission error is equal to the largest frame sent by the station (second parameter of function R_i at line 7 of Figure 5).

If station k emits the lowest priority frames of the application, it will not delay any other frame and thus N_k would be infinite in theory. In practice, the software layers on top of the communication controller have to be informed in a reasonable amount of time that the station is defective; for instance to execute some diagnostics or reboot the node. N_k has thus to be to set to a maximum value which we chose arbitrarily in this study to be 50 (around 20ms on a 250kb/s network). The algorithm for computing N_k is given in Figure 5 where D_i is the deadline of frame m_i and $R_i(n,C)$ its worst-case response time with n retransmissions of a frame of size C bits :

$$R_i(n,C) = C_i + J_i + I_i(n,C) \tag{13}$$

where J_i is the maximal jitter of m_i, and $I_i(n,C)$ is the limit when m goes to infinity of the following recurrence relation :

$$I_i^0(n,C) = 0, \quad I_i^m(n,C) = \mathcal{E}(n,C) + \max_{m_j \prec m_k}(C_j) + \sum_{m_j \succ m_k} \left\lceil \frac{I_{n,C}^{m-1} + J_j + \tau_{bit}}{T_j} \right\rceil C_j, \tag{14}$$

where $\mathcal{E}$ is the function that counts the overhead induced by n retransmissions of a frame of size C bits :

$$\mathcal{E}(n,C) = n \cdot (23\tau_{bit} + C), \tag{15}$$

with 23 bits being the maximum size of an error frame.

```
1  funct INTEGER $computeN_k$(set of messages $\mathcal{T}$)
2    INTEGER $N_k := 50$, $tmp$;
3    for $i := 1$ to $\#\mathcal{T}$ do
4        if $m_i \notin \mathcal{M}_k \wedge highestPrio\{m_j \in \mathcal{M}_k\} \succ m_i$
5          then
6                $tmp := 0$;
7                while $(R_i(tmp, \max_{j \in \mathcal{M}_k} C_j) \leq D_i) \wedge (tmp - 1 < N_k)$
8                      do $tmp++$; od
9                if $(tmp - 1 < N_k)$ then $N_k := tmp - 1$; fi
10       fi
11       return $N_k$;
12   end
```

Fig. 5. Function computing the value of N_k, the maximum number of retransmission of a frame of station k such that the deadlines of all frames of other stations are still respected. $\mathcal{M}_k$ denotes the set of tasks sent by station k.

B. Case H2.a : defect nodes cannot send correct frames

This assumption implies that whenever a station emits a correct message, we know for sure that the node is not faulty.

1) Proposal: The variable i identifies the state of the system. If the message that has been sent is correct then i is set to zero (assumption H2.a) otherwise i is increased by one. If i has reached F_k then the station becomes bus-off.

2) Markovian analysis: This mechanism can be analyzed under a Markovian model of the dynamics of the system (inter-arrivals are exponentially distributed). The corresponding Markov chain (after uniformization) is defined by the following transition probabilities $P[i+1|i] = p_i$, $P[0|i] = 1 - p_i$, $P[F_k|F_k] = 1$ and it is represented on Figure 6.

The average hitting time of bus-off is shown on Figure 7 for various BERs with a bursty error arrival process defined by equation (11) with $\alpha = 0.1$ and $r = 0.5$. With our proposal, the hitting times are much longer for high values of the BER even though the error model is now considered to be bursty. For instance, with a BER of 0.001 the hitting time for the engine controller is 221 hours versus 40 seconds with the existing mechanisms. In addition, the hitting times are less sensitive

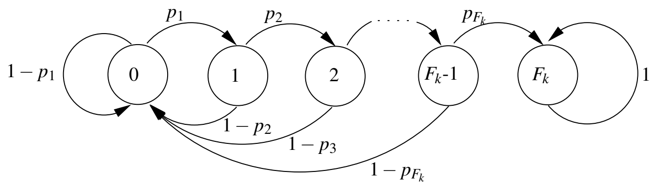


Fig. 6. Markov chain modeling mechanisms of case H2.a with $F_k = 4$.

to the value of the BER which will enable the application designer to assess the risk of bus-off in a satisfactory manner without an exact knowledge of the BER. On the contrary, the hitting time is very sensitive to the priority of the messages (due to N_k). If the application designer is ready to accept some missed deadlines, he has the possibility to increase the value of N_k.

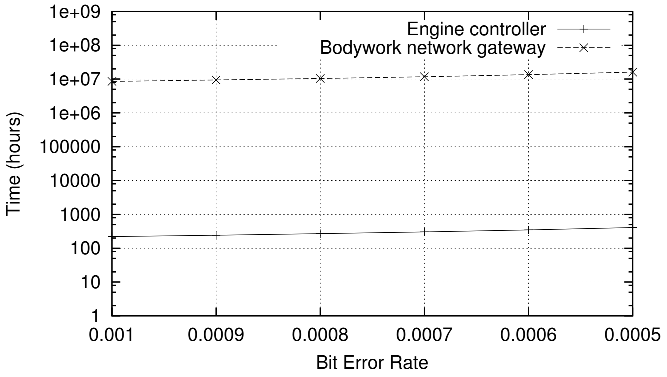


Fig. 7. Average hitting time of the bus-off state for the engine controller and the bodywork network gateway with the BER varying from 0.0005 to 0.001 and $F_k = 31$ for the bodywork network gateway and $F_k = 18$ for the engine controller (smallest value of F_k for the 6 nodes of the application).

C. Case H2.b : defect nodes can send correct frames

Here, we denote by q_k the probability that station k emits a correct frame while being faulty. It is natural to assume that emitting two consecutive correct frames while faulty are two independent events and thus has probability $(q_k)^2$.

1) Proposal: The idea is to weigh the progression towards bus-off by the quantity of information given by the last transmission. The state of the system is given by two counters (i, j) where i indicates the proximity of bus-off and j is the current number of consecutive transmission errors. The initial state is $(1,0)$ and the counters evolve according to the following rules :

- on the occurrence of an error $(i, j) \rightarrow (\lceil i/p_{k_j} \rceil, j+1)$,
- on a successful transmission $(i, j) \rightarrow (\lceil i.q_k \rceil, 0)$,
- the bus-off state is reached when $i \geq 1/\prod_{j=1..F_k} p_{k_j}$.

Imagine that the probability to emit a corrupted message is large (bursts of errors are likely), if the next transmission is unsuccessful, then the quantity of information brought by this event is small, therefore one should not approach bus-off too much. This is the same for a good transmission, imagine that a successful transmission of a faulty node is very unlikely (q_k is small), then the quantity of information is very important and it is natural to make a big step away from bus-off. It is noteworthy that when q_k goes to zero then this approach becomes more and more similar to case H2.a (the state is very close to zero on a correct message). On the other hand, when the error probabilities are independent (p_{k_i} are all equal to p_k), then this mechanism is similar to the existing scheme when one consider the logarithm of the state with steps - $\log(p_k)$ (with $log(p_k < 0)$) instead of +8 on errors and $+\log(q_k)$ (with $log(q_k < 0)$ instead of -1 on success. If one wants to mimic the existing scheme, one just has to take $q_k^8 = p_k$ (for instance $p_k = 10^{-8}$ and $q_k = 10^{-1}$). The underlying assumption in CAN current mechanisms is thus that 8 consecutive correct messages sent by a faulty node (q_k^8) has the same probability as one faulty message sent by a non-faulty node (p_k). The validity of such an hypothesis is questionable especially under heavily perturbed environments where p_k may be large. Our proposal possesses two advantages over the existing scheme : the errors are not necessarily independent and second, the parameters p_k and q_k can be set according to the system and its environment.

2) Markovian analysis: As for the previous cases, one can make a Markovian analysis of this mechanism using Poisson arrival for the frames and assuming that $\alpha_i = \log p_{k_i}$ and $\beta = \log q_k$ are integer values. The Markov chain has the following transition probabilities : $P[(i+\alpha_j, j)|(i, j)] = p_{j+1}$, $P[(i-\beta, 0)|(i, j)] = 1 - p_{j+1}$. The corresponding Markov chain is displayed in Figure 8.

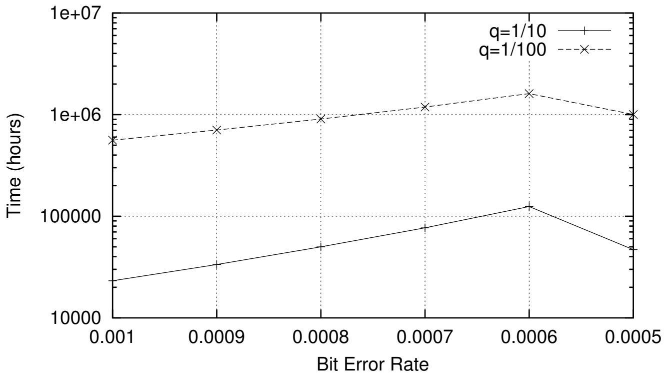


Fig. 9. Average hitting time of the bus-off state for the bodywork network gateway with the BER varying from 0.0005 to 0.001 and for $q = 1/10$ and $1/100$.

As can be seen on Figure 9, an interesting property of the proposal is that the average time to bus-off is roughly linear in q_k (because only $\log(q_k)$ is involved in the dynamics).

VI. Implementation issues

The implementation of our proposal at the communication controller level is easily feasible but it requires to redesign some parts of an existing controllers. A low-cost alternative is to bypass the existing CAN fault confinement mechanisms implemented in silicon and to take the bus-off decision at the application level. The easiest way to achieve this is to allow write access to the TEC located in the communication controller and to clear the TEC to 0 before it reaches 255.

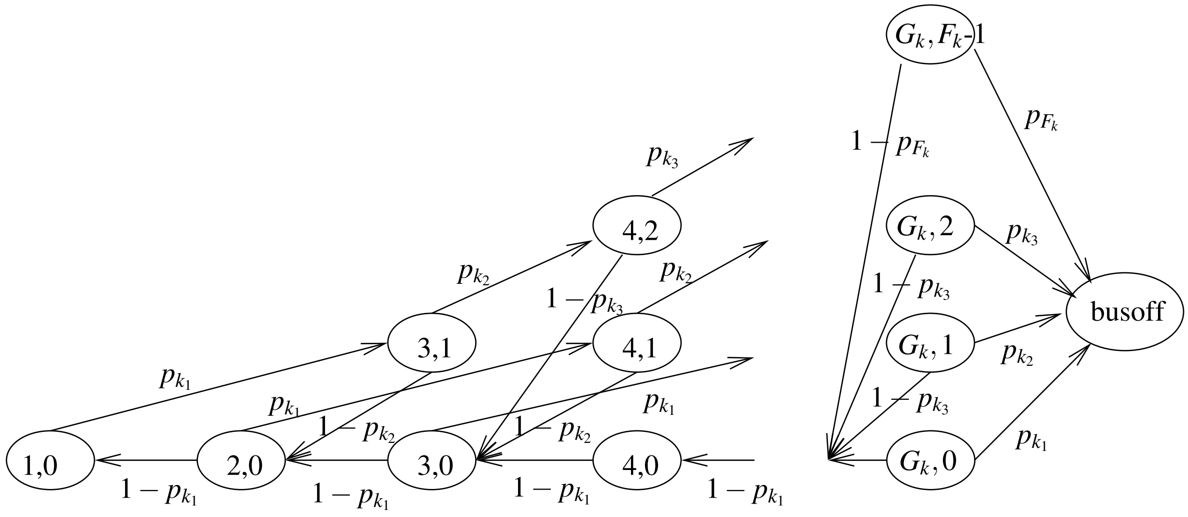


Fig. 8. Markov chain for the analysis of the proposed fault-confinement mechanisms where $\beta = 1$, $\alpha_1 = 2$ and $\alpha_2 = 1$. The value of G_k is $\sum_{j=1,\cdots,F_k-1} \alpha_j$.

To the best of our knowledge, no such controller with writing access to the TEC is available yet. However, depending on the controller, there may exist other way to clear the TEC. For instance, the popular NEC's DCAN module clears the error counters to 0 when it is switched to sleep mode ([13] pp 253). It also enables an automatic software reset (and thus clears the error counters) after the occurrence of bus-off ([13] pp 234). Although these solutions are not very convenient, they provide a way to implement our proposal on existing controllers.

In the rest of this Section, we will discuss how to set the values of the p_{k_i} which are the parameters of the error model involved in our proposal. The setting of the p_{k_i} can be done using measurements carried out on a prototype or even at run-time. Some CAN controllers such as the NEC DCAN module or the Philips SJA1000 ([14]) have interesting error-signalling features such as readable error counters or interrupt-triggering on transmission occurrences. Those features will enable the determination of an error model parameter-setting procedure that will dynamically change the parameter's values when these become improper in the light of the current bus perturbation level. Such an on-line adaptive parameter-setting procedure would be well suited for systems within which the bus perturbation level may vary greatly over time, such as automotive communication systems.

A. Off-line parameters setting

Recall that p_{k_i} is the probability for the non-faulty station k to emit at least a corrupted frame given that the last $i-1$ messages sent by station k were corrupted with p_{k_1} the probability to emit at least one corrupted frame given that the previous frame was correct. The Figure 10 represents a sample measurement taken on a prototype. On this short fragment of trajectory there exists 6 elementary events that give us information to assess the value of p_{k_1}. These events are the results of the transmission in the interval $[t_2, t_3[$, $[t_4, t_5[$, $[t_6, t_7[$, $[t_7, t_8[$, $[t_9, t_{10}[$ and $[t_{13}, t_{14}[$ (they all have in common that the transmission in the preceding interval was successful). On this sample trajectory, p_{k_1} can be estimated to $1/3$ since 2 frames out of the 6 transmitted were corrupted.

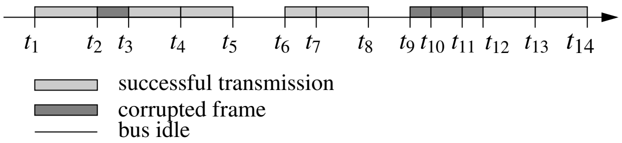


Fig. 10. A sample measurement of the frames sent by a given station k.

Denote by $R_k[i]$ the outcome of the i^{th} transmission (either successful or corrupted frame) of station k and $\#R_k$ the number of frames of the sample. The array $badOutcome[i]$ stores the number of frames that were corrupted given that $(i-1)$ successive transmission errors occurred previously while $allOutcome[i]$ stores the total number of cases where $(i-1)$ successive errors occurred. The algorithm for computing the p_{k_i} values is given on Figure 11 where *max* is the maximum size of all bursts of the sample.

B. On-line Parameters setting

Two main design goals of the parameter setting scheme are to keep the complexity low and to be robust to FER variations. Since on a fixed time interval the number of errors might be arbitrarily small, we propose to set the parameters using the last n bursts of errors. The value of n should be chosen such that the parameters actually reflect the current bus perturbation level while keeping the results statistically valid. In practice, we suggest values of n greater than 100. We consider two parameter setting procedures : one using the sample made of the last n bursts of errors and the second with a sliding-window of size n. Whatever the technique, the initial parameters should be set to "reasonable" values chosen according to measures or from the experience gained on similar systems. It is not mandatory that the computation of the parameters is performed on all nodes of the network (some CAN nodes do not even have computational capability); a chosen node can broadcast

```
INTEGER $burstSize = 0$;
INTEGER $badOutcome[max] = \{0, 0, 0, .., 0\}$;
INTEGER $allOutcome[max] = \{0, 0, 0, .., 0\}$;
for $i := 1$ to $\#R_k$ do
    if $R_k[i] = \text{corrupted}$
      then
            $burstSize{+}{+}$;
            if $i \neq 1$ /* the past is not known */
              then $badOutcome[burstSize]{+}{+}$;
                   $allOutcome[burstSize]{+}{+}$;
            fi
      else
            if $i \neq 1$
              then $allOutcome[burstSize + 1]{+}{+}$;
            fi
            $burstSize := 0$;
    fi
od
for $i := 1$ to $max$ do
    if $allOutcome[i] \neq 0$
      then
            $p_{k_i} = badOutcome[i]/allOutcome[i]$;
    fi
od
```

Fig. 11. Algorithm for computing the value of p_{k_i}.

```
if $size(newBurst) > size(oldBurst)$
  then
        for $i := size(oldBurst) + 1$ to $size(newBurst)$
            do
            $badOutcome[i]{+}{+}$;
            $allOutcome[i]{+}{+}$;
        od
  else
        if $size(newBurst) < size(oldBurst)$
          then
                for $i := size(oldBurst)$ downto $size(newBurst) + 1$
                    do
                    $badOutcome[i]{-}{-}$;
                    $allOutcome[i]{-}{-}$;
                od
        fi
fi
for $i := 1$ to $max$ do
    if $allOutcome[i] \neq 0$
      then
            $p_{k_i} = badOutcome[i]/allOutcome[i]$;
    fi
od
```

Fig. 12. Updating the value the p_{k_i}'s values after the end of a burst.

the parameters to all other nodes periodically of after each change of the values of the parameters.

1) Sampling: The parameters are estimated every n bursts of errors. The new set of p_{k_i}'s is computed with the algorithm described in Figure 11. It may replace the older p_{k_i}'s values but influence of the past can also be taken into account for instance using the exponential smoothing technique which assigns exponentially decreasing weights as the observation get older. In the latter case, if we denote $\tilde{p}_{k_i}$ as the value of p_{k_i} computed on the last n bursts of errors, the new value of p_{k_i} is given by :

$$p_{k_i} = (1 - \alpha) \cdot \tilde{p}_{k_i} + \alpha \cdot p_{k_i}$$

where the smoothing constant α can be determined on samples of measurements such as to minimize the squared errors between the forecasts and the actual observations. Two important advantages of this strategy are the low complexity of the computation and the infrequent update of the parameters.

2) Sliding window: Another strategy is to update the parameters after each burst of errors. The oldest burst of the sample is simply replaced by the new observation according to the algorithm given on Figure 12.

This technique should provide a better adaptation to the current bus perturbation than the sampling of size n bursts, its drawback being a more frequent update of the parameter.

VII. Conclusion

In this study, we proposed a Markovian analysis of the existing fault-confinement mechanisms of the CAN protocol. These results may help the application designer to assess the risk of reaching bus-off and error-passive. It also provides some evidence that the existing mechanisms has several shortages : bus-off state is reached too fast for non-faulty nodes under high perturbation, the REC is useless in nearly all cases and the parameters cannot be tuned (for instance to consider bursty errors).

We have proposed two new mechanisms that address these drawbacks. These mechanisms can mimic the original ones with adequate parameters but also show the interest of considering bursty-errors : the hitting time of bus-off for non-faulty nodes increases hugely while faulty systems reach bus-off within the same amount of time. The same scheme can be adapted easily for deciding error-passive.

The implementation issues raised by our proposals have been addressed in Section VI. Different algorithms for setting the error model parameters have been provided : this can be done off-line, using measurements carried out on a prototype, or at run-time with two strategies that induce different overheads.

Acknowledgments: The authors would like to thank the anonymous referees for their helpful comments and suggestions.

Appendix I
Description of the case study

The application considered from Section 2 is composed of 12 frames (e.g. speed and torque from the engine controller) listed in figure 13. The transmission rate of the CAN bus is 250kb/s. The Data Length Code (DLC_i) denotes the number of bytes of frame i, T_i is the period and one assumes deadlines to be equal to the periods.

Appendix II
Computation of the rates λ_0^k and λ_1^k

Let us denote by $S_{i,n}$ the size of the nth instance of message i having DLC_i data bytes. Due to CAN's bit-stuffing, all instances of the same message may not have the same size. However, $S_{i,n}$ remains bounded:

$$47 + 8DLC_i \leq S_{i,n} \leq 47 + 8DLC_i + \left\lfloor \frac{34 + 8DLC_i - 1}{4} \right\rfloor. \quad (16)$$

Priority (Id)	Transmitter node	DLC_i	T_i
1	engine controller	8	10 ms
2	wheel angle sensor	3	14 ms
3	engine controller	3	20 ms
4	AGB	2	15 ms
5	ABS	5	20 ms
6	ABS	5	40 ms
7	ABS	4	15 ms
8	bodywork gateway	5	50 ms
9	device *y*	4	20 ms
10	engine controller	7	100 ms
11	AGB	5	50 ms
12	ABS	1	100 ms

Fig. 13. Message set of the application.

If one considers the size of all instances to be equal to the upper bound, one can derive a conservative value for the unsuccessful transmission rate by using the same method as for the stochastic case below and replacing $S_{i,n}$ by the upper bound. If more information can be obtained for the $S_{i,n}$ then more accurate estimation can be computed. In the following, we assume that it is possible to estimate the distribution of the $S_{i,n}$ or at least its first J moments.

The transmission time is $C_{i,n} \overset{\text{def}}{=} S_{i,n} \cdot \tau_{bit}$ where τ_{bit} is the bit time (i.e. the time between two successive bits). The Frame Error Rate for the n^{th} instance of message i, called $FER_{i,n}$, can be estimated using the Bit Error Rate (BER) :

$$FER_{i,n} \quad = \quad 1-(1-BER)^{S_{i,n}}.$$

One denotes by $\lambda_{i,n}$ the rate of unsuccessful transmissions (i.e. corrupted frames) of the n^{th} instance of message i. One has to take into account the surcharge generated by transmission errors. To each transmission error corresponds a retransmission which can be, in its turn, corrupted (and so on). One has the following relation :

$$\begin{aligned}\lambda_1^{i,n} &= FER_{i,n}(\left(\frac{1}{T_i}\right)+\left(\frac{1}{T_i}\right)FER_{i,n}+ \\ &\quad \left(\frac{1}{T_i}\right)FER_{i,n}{}^2+\ \ldots) \\ &= \frac{FER_{i,n}}{T_i}\left(1-FER_{i,n}\right).\end{aligned}$$

The average rate of unsuccessful transmissions for message i is $\lambda_1^i = (1/T_i)\cdot\mathbb{E}[FER_{i,n}/(1-FER_{i,n})]$ and the average rate on station k is $\lambda_1^k = \sum_i \lambda_1^i$. This quantity can be computed using the distributions of $S_{i,n}$ for all i and n. Furthermore, since BER is small compared to 1, $\mathbb{E}[FER_{i,n}/(1-FER_{i,n})]$ can be approximated to

$$\mathbb{E}[1/(1-S_iBER)]-1 \approx \sum_{j=1..J} BER^j \cdot \mathbb{E}[S_i^j],$$

if the moments have sub-exponential growth.

As for the rate of successful transmission on station k, λ_0^k, it is equal to $\sum_i 1/T_i$ since all messages are successfully transmitted exactly once.

References

[1] International Standard Organization ISO, *Road Vehicles - Low Speed serial data communication - Part 2: Low Speed Controller Area Network*, ISO, 1994, ISO 11519-2.

[2] International Standard Organization ISO, *Road Vehicles - Interchange of Digital Information - Controller Area Network for high-speed Communication*, ISO, 1994, ISO 11898.

[3] CAN in Automation International Users and Manufacturers Group (CiA), "CAN application layer (CAL)," 1995, Cia/DS201-207.

[4] European Committee for Electrotechnical Standardization CENELEC, "Low voltage switchgear and controlgear - part 5: Control circuit devices and switching elements - smart distributed systems (SDS)," 1997, document CLC/TC(SEC)146 Smart Distributed Systems.

[5] Allen-Bradley, "Devicenet specification," 1994, vol. 1 & 2.

[6] CAN in Automation International Users and Manufacturers Group (CiA), "CANopen communication profile for industrial systems," 1996, CiA/DS301 (Version 3.0).

[7] J. Unruh, H.-J. Mathony, and K.-H. Kaiser, "Error detection analysis of automotive communication protocols," Tech. Rep., Robert Bosch GmbH, 1989.

[8] K. Tindell and A. Burns, "Guaranteeing message latencies on controller area network (CAN)," in *1st International CAN Conference, ICC'94*, 1994.

[9] K. Tindell and A. Burns, "Guaranteed message latencies for distributed safety-critical hard real-time control networks," Tech. Rep. YCS229, Department of Computer Science, University of York (UK), Mai 1994.

[10] N. Navet and Y.-Q. Song, "Design of reliable real-time applications distributed over CAN (Controller Area Network)," in *9th IFAC Symposium on Information Control in Manufacturing, INCOM'98*, Juin 1998.

[11] N. Navet, Y.-Q. Song, and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over CAN (Controller Area Network)," *Journal of Systems Architecture*, vol. 46, no. 7, pp. 607–618, 2000.

[12] I. Broster, A. Burns, and G. Rodríguez-Navas, "Probabilistic analysis of CAN with faults," in *23rd IEEE Real-Time Systems Symposium*, 2002, pp. 269–278.

[13] NEC Corporation, "upd789850 subseries - preliminary user's manual," April 2000, Document No. U144035J2V0UM00 (2nd edition).

[14] Philips Semiconductors, "SJA 1000 stand-alone CAN controller data sheet," January 2000.

Bruno Gaujal is a research director at INRIA Rhone-Alpes since 2003 where he is the leader of the group on large scale networks. He has held several positions in INRIA Sophia-Antipolis, Loria and ENS-Lyon before. His main interest are performance evaluation and control of discrete event dynamic systems.

Nicolas Navet received the M.S. in Computer Science from the University of Berlin (Germany) in 1994 and the PhD in Computer Science from the University of Nancy in 1999. Before joining the INRIA (LORIA Lab.) in November 2000, he was research scientist at Gemplus Software. His research interests include scheduling theory, the design of communication protocols for real-time and fault-tolerant data transmission and probabilistic risk evaluation when transient faults may occur (e.g. EMI).