Fault confinement mechanisms on CAN: analysis and improvements

Bruno Gaujal, Nicolas Navet

FAULT CONFINEMENT MECHANISMS ON CAN : ANALYSIS AND IMPROVEMENTS

B.Gaujal N. Navet
LORIA - INPL
ENSEM - 2, Avenue de la forêt de Haye
F-54516 Vandoeuvre-lès-Nancy

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 and identify several problems. We then propose new mechanisms in order to address them.

Keywords: real-time systems; local area networks; fault tolerance.

1. CAN'S FAULT CONFINEMENT MECHANISMS

CAN has very efficient error detection mechanisms. In Unruh et al. (1989), 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 error frames, the CAN protocol includes fault confinement mechanisms whose objectives are (1) to detect permanent hardware dysfunctioning and (2) to switch off defective nodes. For this purpose, a CAN controller possesses 2 distinct error counters :

- the Transmit Error Counter (TEC) which counts the number of transmission errors detected on the frames that the station sends,
- the Receive Error Counter (REC) which counts the number of transmission errors detected on the frames that the station receives.

Each time a frame is correctly received or transmitted by a station, the value of the corresponding counter is decreased (except when the 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$<$128 and TEC$<$128) : this is the normal operating mode, the station can normally send and receive frames. This is the default state at controller initialization.
- *Error Passive* (REC$>$127 or TEC$>$127) and TEC$\leq$255) : the station may emit but it must wait 8 supplementary bits after the end of the last transmitted frame. Furthermore, the station is not allowed to send an *active error flag* upon the detection of a transmission error, instead it will send a *passive error flag* which is made of 6 recessive bits and has thus no influence on the electric level of the bus. In this state, because of the 8 supplementary bits before sending, the frames sent by the station are no longer certain to respect the worst-case response times computed through schedulability analysis.
- *Bus-off* (TEC$>$255) : The station is automatically switched off from the bus. In this state, the station can neither send or 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 ISO (1994) 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 ISO (1994) pp 48-49, exceptions listed in points b) and c)), two of them are only useful during the initialization phase of the system where only one node may be on-line. Given these assumptions, the rules for modifying the counter value of the stations are :

(1) Frame transmission successful. 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. The node is the sending node : if the TEC is not nil, this is decreased by one, otherwise it remains unchanged.
(2) Unsuccessful transmission (transmission error detected). The node is not the sending node : The REC is increased by one. The node is the sending node: the TEC is increased by 8.

Whatever the result of transmission, there is no more than one counter whose value is modified on a given station.

2. BUS-OFF HITTING TIME

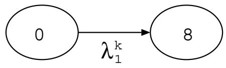
CAN fault confinement mechanisms are conceived to disconnect defective nodes from the network and prevent them from perturbing the whole network. However, under severe electro-magnetic interference conditions, one or several nodes can reach the bus-off state just because of transmission errors. It is 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, one model the Transmit Error Counter (TEC) with a Markov chain in continuous time (also called a Markov process).

2.1 *Modeling*

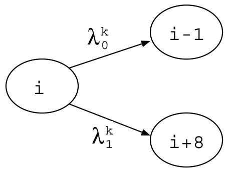
Under the assumptions that state changes are exponentially distributed, the evolution of the TEC can be modeled by a Markov process. Let λ_0^k be the rate of transmission of non-corrupted messages for station k and λ_1^k be its rate of corrupted messages.

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 depending on the TEC value (denoted by i) :

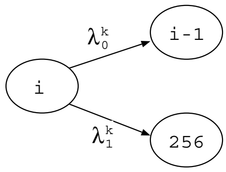
- $i = 0$:



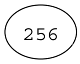
- $i \in \{1..248\}$:



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



- $i = 256$:



The computation of λ_0^k and λ_1^k is detailed in Appendix A. The state 256, which corresponds to the bus-off state, is a so-called *absorbing* state from which it is impossible to escape and that stops the process. This is exactly the functioning scheme of the CAN protocol. When a station becomes "bus-off", it can neither send nor receive frames. With the previously exposed rules, one obtains the following generator matrix of size 257∗257 (the Markov chain having 257 states) :

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

with $\lambda^k = (\lambda_0^k + \lambda_1^k)$ (the row sums of $\mathcal{Q}$ is 0).

Because it is much more convenient to handle, this Markov process will be transformed in the stochastically equivalent discrete time Markov chain termed the *uniformized chain*. Let $q_i = \sum_{j \neq i} \mathcal{Q}_{i,j}$ the total rate out of state i and $q_{max} = \sup_{i \geq 0} q_i$. Since $q_{max} < \infty$ we can uniformize the Markov process so that it is equivalent to a Markov chain denoted by 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 matrix P under its "canonical form" is :

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

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 :

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

with $\gamma_i = 1$ if $i \neq 256$ and 0 otherwise. Taking expectations, we get :

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

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 times 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 per definition equal to $V[N_i] = E[N_i^2] - E[N_i]^2$. One has

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

Taking expectations :

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

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

2.2 *Numerical applications*

To illustrate this analysis, one will consider two CAN nodes part of an experimental embedded CAN-based application proposed by PSA (Peugeot-Citröen) Automobiles Company and described in Navet et al. (2000). Six devices exchange messages on a 250kbit/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 being 105 bits for the bodywork network gateway. The characteristics of the 12 frames composing the application is given in Appendix A.

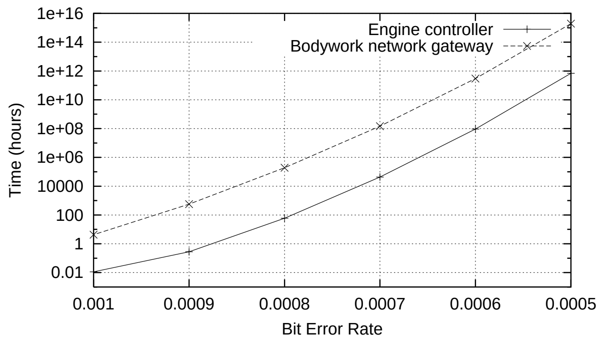


Fig. 1. Average hitting times of the bus-off state for the engine controller and the bodywork network gateway with the Bit Error Rate (BER) varying from 0.0005 to 0.001 .

On Figure 1, one can observe that the average hitting time greatly varies depending on the Bit Error Rate (BER). For instance, it takes in 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 curves on Figure 1 suggest that the more important 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 than the average hitting times which in practice means that there will be a high variability among the observed hitting times.

3. 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.

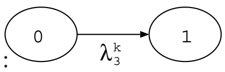
A station becomes error-passive if the REC is greater than 127 or if the TEC is equal to 128. The modeling through a Markov chain is straightforward : each state of the process can be identified through 2 coordinates (i, j) where for instance

i is the value of the TEC and j the value of the REC. In order 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 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).

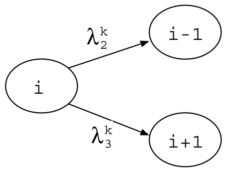
3.1 *Error-passive due to reception*

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. One distinguishes the following cases depending on the REC (denoted by j) :

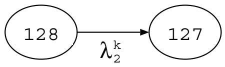
- $j = 0$:



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



- $j = 128$:



Although the CAN standard ISO (1994) 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". For the latter value, 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. The computation of λ_2^k and λ_3^k is detailed in Appendix B.

The Markov process corresponding to the above transitions is then transformed using the uniformization technique described in paragraph 2.1 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 :

$$\pi = \pi \cdot W \tag{7}$$

π_i (i^{th} component of the vector π) gives us 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 $\pi_{128} = 6.656581622 \cdot 10^{-131}$, with a BER equal to 0.0005 on has $\pi_{128} = 1.027266912 \cdot 10^{-170}$. 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 $7\ 10^{-10}$ in the same example. This is consistent with simulation results were such a state was never reached (see paragraph 3.2). These results shows that under realistic bus perturbation level, the time spent in error-passive due to reception is almost nil.

3.2 *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 (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 step spent in error-passive due to the TEC before the station enters the bus-off state. Its expectation is :

$$E[M_i] = \gamma_i + \sum P_{i,j} E[M_j], \tag{8}$$

with $\gamma_i = 1$ if $i \geq 128$ and 0 otherwise. As can

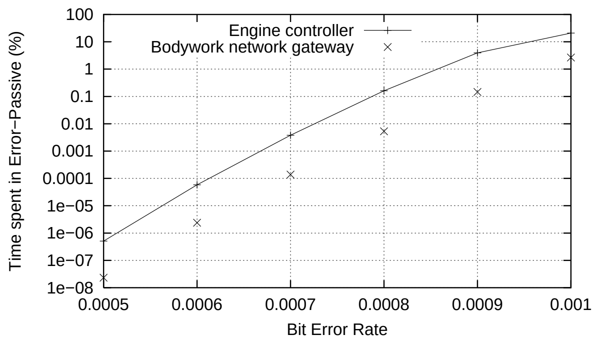


Fig. 2. Average time spent in the error passive state due to transmission for the engine controller and the bodywork network gateway with the Bit Error Rate (BER) varying from 0.0005 to 0.001 .

been seen on Figure 3.2 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 less important 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 3.1 induce 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. 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 (8)) 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 3.2.

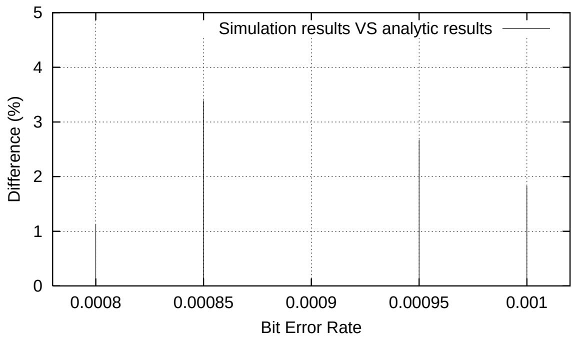


Fig. 3. Difference in percentage between analytical and simulation results regarding the time spent in error-passive. The considered node is the engine controller and the BER ranges from 0.0008 to 0.001 .

3.3 *Conclusion on existing mechanisms*

Experiments and computations performed under realistic assumptions on the bus perturbation level were all nodes are functioning perfectly (no hardware failure) make us think that the bus-off is reached too easily (eg. 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 3.2, 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.

4. 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 an iid process).
Of course H2.a and H2.b are mutually exclusive and will be studied independently.

A station is said faulty if it has a hardware problem (e.g. defect wires). We denote by p_{k_i} the probability for the non-faulty station k to emit a corrupted frame 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. Starting from the measures of the length of the bursts, one can estimate $p_{k_i} = P[\text{error burst length on } k \ \geq i]/P[\text{error burst length on } k \geq i-1]$ for $i \geq 2$ and $p_{k_1} = FER_k/B_k$ (where B_k is the average size of a burst). This choice for p_{k_1} assures that the average FER_k is respected. In the following, the distribution of the burst size will be identical for all stations (and p_{k_i} will be denoted by p_i when no confusion is possible) and given by the modified geometric distribution proposed in Navet et al. (2000) :

$$P[\text{error burst length on } k \ \geq i] = \alpha(r^{i-1}(i-r^i)i+r^i) \tag{9}$$

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

4.1 *When to decide "bus-off"?*

The actual problem we want to solve is to detect if a node is faulty only by looking at the correctness of the transmitted frames. This raises immediately another issue : when should one take a decision ? We believe that the decision can be delayed until the suspected node may 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 is still respected (see Appendix C for the algorithm). It seems natural that our mechanism should decide "bus-off" after F_k consecutive faulty messages where

$$F_k = \max\{N_k, \min\{\Phi | \prod_{j=1..\Phi} p_j < \epsilon\}\} \tag{10}$$

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 such as to keep the number of missed deadlines (of other stations) low. On such systems, transmission errors will necessarily lead 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.

4.2 *Case H2.a : defect nodes cannot send correct frames*

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

4.2.1. *Proposal*

With the variable i that identifies the state of the system, the algorithm for deciding bus-off after a transmission is given in Figure 4.

```
if sent message = corrupted
  then i := i + 1;
       if i = F_k then BUS-OFF fi
  else i := 0;
fi
```

Fig. 4. Deciding bus-off after a transmission.

4.2.2. *Markovian analysis*

This mechanism can be analysed 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 represented on Figure 5.

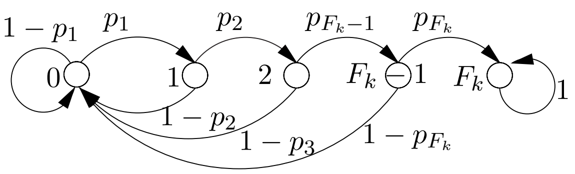


Fig. 5. Markov chain modeling mechanisms of case H2.a with $F_k = 4$.

The average hitting time of bus-off is shown on figure 6 for various BERs with a bursty error arrival process defined by equation (9) 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 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.

4.3 *Case 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 seems natural to assume that emitting two consecutive

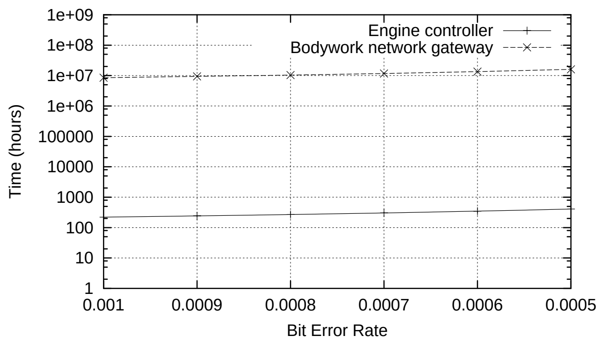


Fig. 6. Average hitting time of the bus-off state for the engine controller and the bodywork network gateway with the BER varying from 0.0005 to 0.001 and $F_k = 31$ for the bodywork network gateway and $F_k = 18$ for the engine controller (smallest value of F_k for the 6 nodes of the application).

correct frames while faulty are two independent events and thus has probability $(q_k)^2$.

4.3.1. *Proposal*

The idea is to weight 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 approaches 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 has 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.

4.3.2. *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 7.

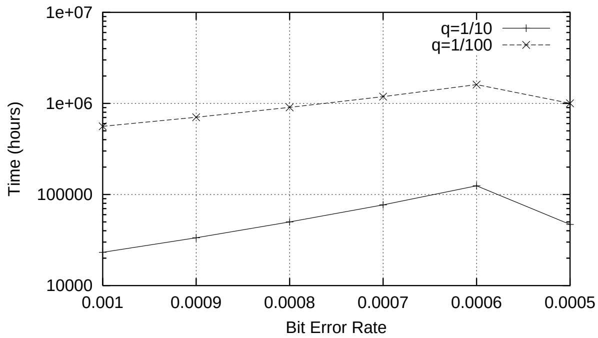


Fig. 8. Average hitting time of the bus-off state for the bodywork network gateway with the BER varying from 0.0005 to 0.001 and for $q = 1/10$ and $1/100$.

As can be seen on Figure 8, 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).

5. 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 in the same amount of time. The same scheme can be adapted easily for deciding error-passive.

Appendix A. COMPLEMENTS TO SECTION 2

The application considered from Section 2 is composed of 12 frames (e.g. speed and torque from the engine controller) listed in figure A.1. The transmission rate of the CAN bus is 250kbit/s. The Data Length Code (DLC) denotes the number of bytes of each frame and deadlines equal periods. One denotes by ρ_k the load induced by station k.

Priority (Id)	Transmitter node	DLC	Period
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. A.1. Message set of the application

One has to take account of 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 :

$$\rho_k = \left(\sum_{m_i \in \mathcal{M}_k} \frac{C_i}{T_i} \right) / \left(1 - FER_k\right),$$

where $\mathcal{M}_k$ is the subset of messages sent by station k, m_i is the message of identifier i and FER_k is the Frame Error Rate for station k which can be estimated with the Bit Error Rate (BER) common to all the stations of the network :

$$FER_k = 1 - \sum_{m_i \in \mathcal{M}_k} \left(\frac{(1-BER)^{S_i}}{T_i} / \left(\sum_{m_j \in \mathcal{M}_k} \frac{1}{T_j} \right) \right), \tag{A.1}$$

with $C_i = S_i \cdot \tau_{bit}$ where τ_{bit} is the bit time (i.e. the time between two successive bits) and S_i is the maximal size of the message m_i (having d_i data bytes) :

$$S_i = 47 + 8d_i + \left\lfloor \frac{34 + 8d_i - 1}{4} \right\rfloor . \tag{A.2}$$

One notes the average size of the frames transmitted by station k :

$$\tilde{S}_k = \left(\sum_{m_i \in \mathcal{M}_k} \frac{S_i}{T_i} \right) / \left(\sum_{m_i \in \mathcal{M}_k} \frac{1}{T_i} \right), \tag{A.3}$$

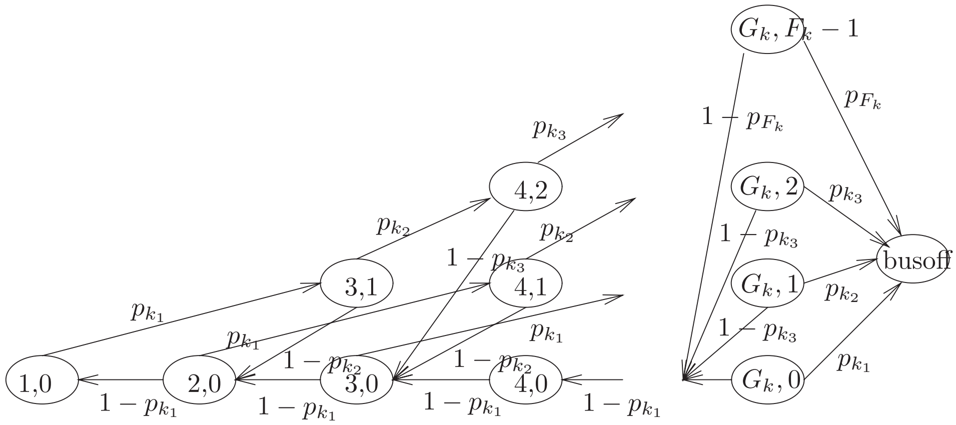


Fig. 7. Markov chain where $\beta = 1$, $\alpha_1 = 2$ and $\alpha_2 = 1$. The value of G_k is $\sum_{j=1,\cdots,F_k-1} \alpha_j$.

λ_1^k is the rate of unsuccessful transmissions (i.e. corrupted frames), one has :

$$\lambda_1^k = \rho_k FER_k / (\tilde{S}_k \cdot \tau_{bit}) = \left(\sum_{m_i \in \mathcal{M}_k} \frac{S_i}{T_i} / (1 - FER_k) \right) FER_k / \tilde{S}_k, \tag{A.4}$$

while λ_0^k, the rate of successful transmissions is :

$$\lambda_0^k = \rho_k \cdot (1 - FER_k)/(\tilde{S}_k \tau_{bit}) = \sum_{m_i \in \mathcal{M}_k} \frac{S_i}{T_i} / \tilde{S}_k. \tag{A.5}$$

Appendix B. COMPLEMENT TO SECTION 3

One denotes λ_2^k the rate of frames successfully received by station k :

$$\lambda_2^k = \sum_{i \neq k} \lambda_0^i, \tag{B.1}$$

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

$$\lambda_3^k = \sum_{i \neq k} \lambda_1^i. \tag{B.2}$$

Appendix C. COMPLEMENTS TO SECTION 4

N_k denotes the maximum number of retransmissions of a frame of station k such that all other frames of the application will respect their deadline. Note that a frame m_i may be delayed by the retransmission of a frame m_j only if m_j has a higher priority (denoted $m_j \succ m_i$). If station k emits the lowest priority frames of the application, it will not delay any other frame and N_k is set to a maximum value that we chose to be 50. The algorithm for computing N_k is given in Figure C.1 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 :

$$R_i(n, C) = C_i + J_i + I_i(n, C) \tag{C.1}$$

```
funct INTEGER computeN_k(set of task T)
  INTEGER N_k := 50, tmp;
  for i := 1 to #T do
    if m_i ∉ M_k ∧ highestPrio{m_j ∈ M_k} ≻ m_i
      then
          tmp := 0;
          while (R_i(tmp, max_{j∈M_k} C_j) ≤ D_i) ∧ (tmp − 1 < N_k)
              do tmp + +; od
          if (tmp − 1 < N_k) then N_k := tmp − 1; fi
    fi
    return N_k;
end
```

Fig. C.1. Function computing the value of N_k.

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

$$I_i^0(n, C) = 0, \quad I_i^m(n, C) = \mathcal{E}(n, C) + \max_{m_j \prec m_k} (C_j) + \sum_{m_j \succ m_k} \left\lceil \frac{I_{n,C}^{m-1} + J_j + \tau_{bit}}{T_j} \right\rceil C_j, \tag{C.2}$$

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

$$\mathcal{E}(n, C) = n \cdot (23\tau_{bit} + C), \tag{C.3}$$

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

References

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

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*, 46(7):607–618, 2000.

J. Unruh, H.-J. Mathony, and K.-H. Kaiser. Error detection analysis of automotive communication protocols. Technical report, Robert Bosch GmbH, 1989.