When is CAN the Weakest Link? A Bound on Failures-In-Time in CAN-Based Real-Time Systems

Arpan Gujarati and Björn B. Brandenburg
Max Planck Institute for Software Systems (MPI-SWS)

***Abstract*—A method to bound the *Failures In Time (FIT)* rate of a CAN-based real-time system, *i.e.*, the expected number of failures in one billion operating hours, is proposed. The method leverages an analysis, derived in the paper, of the probability of a correct and timely message transmission despite host and network failures due to electromagnetic interference (EMI). For a given workload, the derived FIT rate can be used to find an optimal replication factor, which is demonstrated with a case study based on a message set taken from a simple mobile robot.**

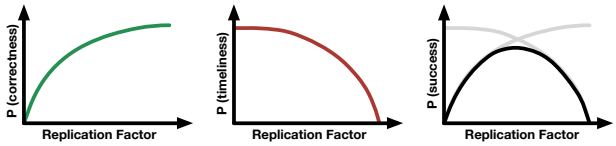
I. Introduction

Automotive embedded systems are surrounded by spark plugs and electric motors. Industrial embedded systems are often deployed in close vicinity to high-powered machinery. Autonomous robots may need to operate in radiation-prone environments. Each of these examples is a safety-critical real-time system exposed to electromagnetic interference (EMI).

To withstand the effects of EMI—including hangs, crashes, or incorrect outputs due to faults on the hosts, and, in networked systems, also message corruption during transmissions—critical tasks are often *replicated* on independent, networked hosts, and network stacks typically detect and *retransmit* corrupted messages. For example, in a *Controller Area Network* (CAN), CAN controllers automatically queue messages for retransmission if any host signals a transmission fault. Similarly, classic replication approaches such as *triple modular redundancy* (TMR) are commonly used to tolerate host failures.

In a distributed real-time system, however, these two techniques—spatial vs. temporal redundancy—are fundamentally at odds, as they both require spare network bandwidth, a scarce resource. More precisely, a *low* network load is favorable to ensure that deadlines are not violated due to retransmissions (*i.e.*, the more slack, the better), but active replication *increases* the network load as more messages are sent (*i.e.*, it reduces the available slack). Thus, beyond a certain point, adding replicas actually hinders a system's ability to meet all deadlines, and can ultimately decrease its overall reliability—see Fig. 1 for a graphical illustration of the problem.

For a systems reliability engineer, this poses tough design questions. Clearly, replicating the most-critical tasks to *some* extent is a good idea, but what is the "right" number of replicas? Should one choose TMR, or would two replicas suffice to achieve the desired level of reliability? Perhaps five replicas would be much better? Or would this already be detrimental from a timing-reliability point of view? If there are multiple critical tasks, but only limited slack available, how should it be allocated? Favor higher-rate tasks (as they suffer from greater exposure to transient faults), or favor lower-rate tasks



(a) (b) (c)

Fig. 1. *Replication factor* denotes the number of independent replicas of critical tasks. The illustration shows that: **(a)** probability of correct execution improves with replication; **(b)** probability of timely execution degrades with replication; and **(c)** probability of correct *and* timely execution initially increases with replication due to improved resilience to host failures, but then starts degrading with replication due to increasing bus load.

(which induce less additional load if replicated)? None of these questions admits an obvious, quick answer.

And further integration challenges exist. For instance, hardening the network subsystem (or individual hosts), be it by adding extra shielding to attenuate EMI, or by adding a second bus to split the load, is of little value *if the network is not actually the weakest link*. But when *is* the network, and the hosts connected to it, the weakest link? In short, how can the timeliness vs. correctness tradeoff be *quantified*, and ultimately *optimized*, in the context of system-level reliability goals?

This paper. As a first step, to render these questions more approachable to future study, we propose to recast the timeliness vs. correctness question as the problem of bounding a distributed real-time system's *Failures-In-Time* (FIT) rate, or its inverse, the *mean time between failures* (MTBF), as a function of both system load and EMI strength.

FIT, which is the expected number of failures in one billion operating hours, is a standard metric that is widely used in the semiconductor, electronics, and defense industries to characterize both component and product reliability. FIT rates are a convenient metric because they are compositional, that is, they are usually determined for low-level parts such as individual capacitors or memory cells (*e.g.*, see [33] for a tutorial), from which the FIT rate of larger components (such as memory modules or power supplies) and even entire systems (say, an on-board computer) can be derived. By expressing a distributed real-time system's likelihood of violating its logical and temporal specification as a FIT rate, a direct comparison of its relative impact on overall system reliability becomes possible—the weakest link, if any, becomes apparent.

Contributions. Our specific focus in this paper is CAN-based systems, due to CAN's attractive timing properties and its prevalence in distributed real-time systems. Inspired by prior work by Broster *et al.* [8], the main contribution of this paper is the (to our knowledge) first bound on the FIT rate of CAN-

based distributed real-time systems (§VI), which in turn is based on the first probabilistic analysis of CAN that takes both host and network faults into account (§V). Our analysis applies to both systems with and without synchronized clocks, and to both sporadic and periodic message arrivals. An empirical evaluation (§VII) shows our analysis to closely track simulation results. Most interestingly, the analysis results confirm the effect illustrated in Fig. 1: in certain scenarios, adding just two more replicas, *i.e.*, from six to eight, can drive up the FIT rate by roughly twenty orders of magnitude (Fig. 4(b)).

We begin with a brief overview of CAN and relevant fault types, and then establish essential definitions and notations.

II. System Model and Assumptions

CAN is a broadcast bus designed for simple yet efficient communication in embedded systems. It has a linear topology and the bus arbitration policy results in priority-based non-preemptive scheduling. A CAN controller packs each message into a frame along with a cyclic redundancy checksum (CRC), which ensures that hosts detect virtually all transmission errors. In case of an error, the corresponding message is automatically queued for retransmission (see [4] for details).

Faults. As a result of the error detection and correction mechanism in CAN, *raw transient faults* (*i.e.*, EMI-induced bit-flips) may result in automatic retransmissions, which we refer to as *transmission failures*.

Regarding the effect of raw transient faults affecting hosts, we classify them into two broad categories, crash failures and commission failures. *Crash failures* occur if the system suffers an EMI-induced transient corruption that causes an exception to be raised and the system to be rebooted. Although bit flips can cause unbounded hangs in principle (*e.g.*, loops that never terminate due to a bit flip in the termination condition), we assume that each host is equipped with a watchdog timer that reboots the system in case of a hang. A crashed system may remain unavailable for some time while it reboots and thus causes an interval in which messages are continuously omitted. If a system requires application state to be restored after a reboot, then any delay due to state re-synchronization is included in this unavailability interval; how application state is re-synchronized (if needed) is beyond the scope of this work.

A *commission failure* occurs if a message is corrupted during preparation before the CAN controller computes the payload checksum included in the CAN header. For example, bit flips in registers or memory of the CAN controller may manifest as commission failures. We refer to the interval during which a message is at risk of corruption as its *exposure interval*.

The exposure interval of a message depends on the application design, *i.e.*, it depends on whether the message computation relies solely on the current input or also on application state that could be affected by *latent faults* (*i.e.*, state corruptions that have not yet been detected). Therefore, we classify tasks (or message preparations) as either *stateless* or *stateful*.

The exposure interval of a message that is prepared by a stateless task is trivially defined as its scheduling window (*i.e.*, the time from its arrival until its deadline). The exposure interval of a message from a stateful task, however, depends on the mechanisms in place to tolerate (or avoid) latent faults.

We consider two scenarios: if the hardware platform is built from *Error-Correcting Code* (ECC) memory and processors using *lockstep* execution (common in safety-critical systems), then the built-in protections suppress latent faults; otherwise, if no such architectural support is available, then any relevant state can be protected with software-based ECC or checksums.

In the first case, ECC memory guarantees that the latent errors are prevented either by silently fixing bit flips, or by converting unfixable errors into crashes (by raising an exception). Combined with lockstep execution, the hardware guarantees that a message can only be corrupted while it is being prepared or while it is residing in device memory prior to CRC computation. Thus, as with stateless messages, it suffices to consider the scheduling window as the exposure interval.

For systems that use software techniques such as checksums and error-correcting codes to detect (and possibly correct) latent faults (*e.g.*, see [28]), we assume the presence of a *data integrity checker* task that periodically checks the checksums of all relevant data structures (and that reboots the system in the case of any mismatch). In this case, a message may be corrupted due to host faults occurring either at any time during its scheduling window, or at any time since the data integrity was last checked. Therefore, the exposure interval of the message includes its scheduling window and (in the worst case) an entire period of the data integrity checker task. As is the case with ECC memory, this mechanism avoids data corruption at the expense of an increased crash rate.

We emphasize that our notion of "commission failures" does *not* refer to software bugs or malicious (or Byzantine) agents.

System model. Consider a task system T where tasks communicate with each other via messages. Considering tasks to be vertices and messages to be edges, the task system forms a directed, acyclic graph G. The task system T is deployed over a distributed system H that consists of multiple independent hosts that we assume to be networked with a single CAN bus. (This is not a major restriction: although systems with multiple, bridged CAN buses can be found in practice, each CAN instance can be analyzed individually in such cases.)

Our analysis is primarily concerned with messages transmitted on the CAN bus; task execution on hosts is considered implicitly. We therefore do not model tasks in detail and focus on the message set instead.

We let $M = \{M_1, M_2, \ldots, M_n\}$ denote the set of all messages. Let $T_{send_i}, T_{recv_i} \in T$ denote the tasks that send and receive message M_i, respectively. We assume that task T_{send_i} sends message M_i either *periodically* or *sporadically* with a *period* or *minimum inter-arrival time* P_i.

Let $M_{i,k}$ denote the k^{th} runtime activation (*i.e.*, instance) of message M_i. For each instance $M_{i,k}$, let $a_{i,k}$ denote its *arrival time* and $d_{i,k}$ denote its *absolute deadline*, *i.e.*, the time by which it must have been successfully received by the recipient. Let D_i denote the *relative deadline* of message M_i, *i.e.*, $d_{i,k} = a_{i,k} + D_i$ for each instance $M_{i,k}$.

If all messages meet their respective deadlines, then any end-to-end deadlines on G are considered satisfied. In addition, let J_i denote the *maximum jitter* of message M_i. We assume that J_i accounts for all processing delays and scheduling delays

experienced by the task T_{send_i} while preparing a message $M_{i,k}$ before $M_{i,k}$ is queued for transmission.

Replicated system model. To improve system reliability, critical tasks in T are deployed on multiple hosts. For example, if T_{send_i} is a critical task, we may choose to deploy two active copies of T_{send_i} (say $T^a_{send_i}$ and $T^b_{send_i}$) on separate hosts (say H_a and H_b, respectively). Tasks $T^a_{send_i}$ and $T^b_{send_i}$ both send separate messages M^a_i and M^b_i to task T_{recv_i}. Although both messages M^a_i and M^b_i should ideally always contain the same payload, replication guarantees that if either H_a or H_b crashes, T_{recv_i} can still receive at least one of the two messages in time. We refer to tasks $T^a_{send_i}$ and $T^b_{send_i}$ as *replicas* of task T_{send_i}, and messages M^a_i and M^b_i as *replicas* of message M_i.

More formally, let r_i denote the number of replicas of message M_i, also known as its *replication factor*. The r_i replicas of message M_i are denoted as $M^1_i, M^2_i, \ldots, M^{r_i}_i$, and the runtime messages corresponding to each replica M^j_i are denoted $M^j_{i,1}, M^j_{i,2}, \ldots$; the parameters D_i and P_i remain the same for all replicas of message M_i. The arrival time $a_{i,k}$ and the absolute deadline $d_{i,k}$ also remain the same for all replicas of the runtime message $M_{i,k}$. However, since the jitter experienced by each replica is host-specific, we let J^j_i denote the replica-specific maximum jitter for each message M^j_i. Finally, the replicated system model differs in that task T_{recv_i} may receive multiple messages $M^1_i, M^2_i, \ldots, M^{r_i}_i$ instead of a single message M_i. Therefore, for each iteration k, having received one or more messages $M^1_{i,k}, M^2_{i,k}, \ldots, M^{r_i}_{i,k}$, task T_{recv_i} is assumed to execute a *receiver protocol* in order to derive the *final message value*, which is denoted $M^{final}_{i,k}$. We discuss two possible receiver protocols in §IV.

Assumptions. In the proposed analysis, we derive guarantees assuming only a single replica of task T_{recv_i} and that task T_{recv_i} never fails. In case task T_{recv_i} also has multiple replicas, the proposed analysis can be applied to each replica separately. In addition, given any message $M^j_{i,k}$, we assume that parameters i, j, and k can be identified from the message frame. We also assume that any clock synchronized to some source of physical time may lag behind the physical time by at most Δ_{clk} units of time. For clocks that are not synchronized, Δ_{clk} is not defined.

We also require that "babbling idiot" failures, *i.e.*, cases where a host attempts to monopolize the bus, are detected and mitigated by *bus guardians* [5, 8] that cut off (and reboot) "babbling" hosts, and further assume that crash and commission failures across multiple hosts are mutually independent.

We revisit these assumptions in §VIII and Appendix A, where we discuss their implications and potential alternatives.

III. Fault Model

As mentioned in §I, since safety-critical real-time systems are susceptible to EMI, this work focusses on transient faults (also known as *soft errors* or *single-event upsets*) as the primary considered source of failures. We first model the raw EMI-induced transient faults, and then consider the resulting program-visible effects.

A. EMI Model, Derating Factors, and Exposure Intervals

Let λ_{CAN} and λ_{H_i} denote the average rate of raw transient faults occurring on a CAN bus and inside the hardware components of each host $H_i \in H$, respectively. In practice, these rates are empirically determined with measurements or from environmental modeling assuming worst-possible operating conditions, and typically include safety margins as deemed appropriate by reliability engineers or domain experts.

Given the rates λ_{CAN} and λ_{H_i}, we model the raw transient faults as random events following a Poisson distribution (other possible fault models are discussed in §VIII).

Let $Poisson(n, \delta, \lambda)$ denote the probability mass function of the Poisson distribution, *i.e.*, the probability that n events occur in time δ given that their mean rate of occurrence is λ:

$$Poisson(n, \delta, \lambda) = \frac{e^{-\delta \cdot \lambda} \left(\delta \cdot \lambda\right)^n}{n!}.$$

The probability that n raw transient faults occur on a CAN bus in any interval of length δ is given by $Poisson(n, \delta, \lambda_{CAN})$; and the probability that n raw transient faults occur inside the hardware of host H_i in any interval of length δ is given by $Poisson(n, \delta, \lambda_{H_i})$.

As in Broster et al.'s analysis [6], we assume that every transient fault on the CAN bus causes a retransmission. Thus, the CAN bus retransmission-rate is also bounded by λ_{CAN}, and the probability that n retransmissions occur in any interval of length δ is bounded by $Poisson(n, \delta, \lambda_{CAN})$. This is an overestimation because a transient fault may occur when the bus is idle, because multiple transient faults may result in a single retransmission, and because the number of retransmissions in practice is limited by the CAN bandwidth, whereas a Poisson distribution has a non-zero probability for any number of events.

Next, we consider transient faults in hosts. Prior studies have shown that the majority of transient faults do not affect program execution [34]. Therefore, in order to derive the average rate of commission failures and crash failures on any host H_i, we assume a *derating factor* that accounts for masked transient faults, which can be determined empirically [22]. Let $f_{H_i,com}$ and $f_{H_i,crash}$ denote the host-specific derating factors for commission failures and crash failures, respectively; the average rate of commission failures and crash failures on host H_i is then given by $\lambda_{H_i,com} = f_{H_i,com} \cdot \lambda_{H_i}$ and $\lambda_{H_i,crash} = f_{H_i,crash} \cdot \lambda_{H_i}$, respectively.

Like the raw transient faults, we also model commission and crash failures as random events following a Poisson distribution. Thus, the probability that n commission failures occur in any interval of length δ is $Poisson(n, \delta, \lambda_{H_i,com})$; and the probability that n crash failures occur in any interval of length δ is $Poisson(n, \delta, \lambda_{H_i,crash})$.

With respect to commission failures, as discussed in §II, we assume for each message a finite exposure interval. We let $e^j_{i,k}$ denote the maximum length of the exposure interval of message $M^j_{i,k}$. For stateless messages, and for stateful message that are prepared on ECC- and lockstep-protected hosts, $e^j_{i,k} = d_{i,k} - a_{i,k} = D_i$. For stateful messages that are prepared on hosts protected by a data integrity checker task, the exposure

interval length is given by $e^j_{i,k} = D_i + P^j_{checker}$, where $P^j_{checker}$ denotes the period of the checker task.

Finally, we let ζ_{H_i} denote the maximum time that host H_i remains unavailable after a crash.

B. Successful and Unsuccessful Message Transmissions

What does it mean to successfully transmit a message M_i in the replicated system model? To illustrate different scenarios of successful and unsuccessful message transmission, we next discuss three simple examples.

Suppose that T_{send_i} is a critical task, of which three active copies, denoted $T^1_{send_i}$, $T^2_{send_i}$, and $T^3_{send_i}$, are deployed in parallel on hosts H_1, H_2, and H_3, respectively. The receiving task T_{recv_i}, which is not replicated, is deployed on host H_4.

We assume that task T_{recv_i} follows the following straw-man protocol in each iteration k: among the messages $M^1_{i,k}$, $M^2_{i,k}$, and $M^3_{i,k}$, the first message received by T_{recv_i} is assigned as the final value $M^{final}_{i,k}$, and the remaining two messages (if received later) are ignored. We discuss three different scenarios in the following, assuming that host H_4 executes fault-free.

Example 1. *Hosts H_1 and H_2 are unavailable due to crash failures. Host H_3 executes fault-free. The worst-case response time of message M^3_i is less than or equal to D_i.* In this scenario, T_{recv_i} receives M^3_i on time. Since H_3 executes fault-free, M^3_i is not corrupted by a commission failure. Thus, despite crash failures, $M^{final}_{i,k}$ is assigned a message that is not corrupted and the assignment happens before deadline $d_{i,k}$. Hence, M_i is considered to be transmitted successfully.

Example 2. *Host H_1 is unavailable due to a crash failure. Messages from host H_2 are corrupted due to commission failures. Host H_3 executes fault-free. The worst-case response times of messages M^2_i and M^3_i are less than or equal to D_i.* In this scenario, T_{recv_i} is guaranteed to infer $M^{final}_{i,k}$ on time, *i.e.*, on or before $d_{i,k}$. However, in the worst case, M^2_i is corrupted due to commission failures on host H_2. As a result, if M^2_i is received before M^3_i, T_{recv_i} infers a corrupted value, *i.e.*, $M^{final}_{i,k} = M^2_i$. Therefore, although M_i is guaranteed to be transmitted on time, it is not guaranteed to be transmitted successfully due to a lack of correctness guarantees.

Example 3. *All hosts execute fault-free. Due to transmission failures, each message transmission is delayed due to at most x retransmissions. After incorporating delays due to x retransmissions, all messages M^1_i, M^2_i, and M^3_i have worst-case response times greater than D_i.* In this scenario, T_{recv_i} always infers a correct value, since any message that is received first and assigned to $M^{final}_{i,k}$ is guaranteed to be not corrupted by any commission failure. However, all the messages may be received after deadline $d_{i,k}$ and $M^{final}_{i,k}$ may not be inferred on time. Thus, although logical correctness is guaranteed, M_i is not guaranteed to be transmitted successfully due to the lack of a timeliness guarantee.

In summary, a successful transmission of message M_i requires both timely inference (*i.e.*, T_{recv_i} must infer $M^{final}_{i,k}$ before its deadline $d_{i,k}$) and correct inference (*i.e.*, the final value $M^{final}_{i,k}$ must be based on a message that was not corrupted due to commission failures on any host).

We next describe two receiver protocols that a message recipient can use in order to mask as many corrupted messages as possible without violating its temporal constraints.

Algorithm 1 Receiver protocol assuming that message arrivals are periodic and hosts have synchronized clocks.

```
1:  for each job J_{i,0} do
2:      set timer Δ_{i,0} = d_{i,0} − Δ_clk
3:  procedure ReceiveMessage(M^j_{i,k})
4:      if M^j_{i,k} is stale then
5:          return
6:      append M^j_{i,k} to vector V_{i,k}
7:  end procedure
8:  procedure TimerExpires(Δ_{i,k})
9:      M^{final}_{i,k} ← most frequent message in V_{i,k}
10:             (breaking ties uniformly at random)
11:     set timer Δ_{i,k+1} = d_{i,k+1} − Δ_clk
12: end procedure
```

IV. Receiver Protocols

The protocol followed by the receiver task T_{recv_i} in the preceding examples, namely to use the first message and to disregard any later-arriving messages, is not a good protocol as it does not maximize the chances of masking commission failures. Instead, the receiver task should take into account as many messages as possible, while adhering to the respective message deadline. We next describe two protocols to address this issue.

Algorithm 1 provides the first protocol, which assumes periodic message transmissions and hosts with synchronized clocks. According to the protocol, all replicas of message $M_{i,k}$ are collected in a vector $V_{i,k}$. A timer is set to fire at time $d_{i,k} - \Delta_{clk}$, which accounts for the maximum clock error Δ_{clk} w.r.t. the physical time. When the timer fires, the message value that is occurring with maximum frequency in $V_{i,k}$ is inferred as the final message value and assigned to $M^{final}_{i,k}$. The protocol guarantees timeliness if at least one message is received by time $d_{i,k} - \Delta_{clk}$. It also maximizes the probability of masking message corruptions due to commission failures, since (within the timing limitations) as many messages as possible are used to estimate the final message value $M^{final}_{i,k}$.

Algorithm 1 assumes that the receiver task T_{recv_i} knows the absolute deadline $d_{i,k}$ of each message $M^j_{i,k}$, which is generally true only for periodic message arrivals, and that Δ_{clk} is defined for the host on which task T_{recv_i} executes. However, these assumptions do not hold for all systems. In particular, if the host executing task T_{recv_i} does not have a reliably synchronized clock, Δ_{clk} might not exist. For sporadic message arrivals, the receiver task does not know the absolute deadlines of any messages received, *i.e.*, $d_{i,k}$ is not known.[1] For such circumstances, we describe a second protocol (Algorithm 2)

[1]While one might imagine a plausible protocol in which the absolute deadline is embedded in each message, we do no consider such approaches here since the limited payload sizes of CAN makes it difficult to transmit a high-resolution timestamp in addition to the actual payload.

Algorithm 2 Receiver protocol for sporadic message arrivals and/or hosts without synchronized clocks.

for each task T_i **do**
  $r'_i \leftarrow \lfloor \frac{r_i}{2} \rfloor + 1$
procedure RECEIVEMESSAGE($M^j_{i,k}$)
  if $M^j_{i,k}$ is stale **then**
    return
  append $M^j_{i,k}$ to vector $V_{i,k}$
  if max. frequency of messages in $V_{i,k} \geq r'_i$ **then**
    $M^{final}_{i,k} \leftarrow$ most frequent message in $V_{i,k}$,
      (breaking any ties uniformly at random)
end procedure

that caters to systems with sporadic message arrivals and/or hosts with clocks that are not synchronized.

According to Algorithm 2, the receiver task waits for at least $r'_i = \lfloor \frac{r_i}{2} \rfloor + 1$ messages with the same message value before estimating $M^{final}_{i,k}$. Given the way r'_i is defined, the final message value $M^{final}_{i,k}$ always reflects the value of the majority quorum, *i.e.*, the largest set of messages with the same message value. However, the protocol does not *enforce* that $M^{final}_{i,k}$ is inferred on time. Rather, this depends on the timing of messages $M^1_{i,k}, \ldots, M^{r_i}_{i,k}$ (Lemma 7 in §V-D).

In the next section, we derive a lower bound for each protocol on the probability that any message M_i is successfully transmitted, *i.e.*, that all messages $M_i = \{M_{i,1}, M_{i,2}, \ldots\}$ are successfully transmitted at runtime.

V. PROBABILISTIC ANALYSIS

In the following, we establish separate lemmas to reason about transmission failures, crash failures, and commission failures (in §V-A, §V-B, and §V-C, respectively). These results are then used to define the final probabilistic analysis in §V-D.

A. Accounting for Transmission Failures

Recall from §I that the CAN protocol detects messages corrupted by transmission failures and automatically queues them for retransmission, which significantly affects message latencies. The problem of determining worst-case response times for *individual* messages has been thoroughly investigated in prior work, *e.g.*, see [8, 14, 18, 23, 25, 32]. We first review the basics of this retransmission-aware response-time analysis, before proposing a new analysis for the replicated system model.

Response-time analysis. (From prior work [8, 14, 32].) Let $R^j_i(n)$ denote the worst-case response time of message M^j_i assuming that it is delayed due to n retransmissions. $R^j_i(n)$ is defined as $R^j_i(n) = J^j_i + \phi^j_i$, where $\phi^j_i = B^j_i + C_i + I^j_i(\phi^j_i) + E^j_i(n)$, and where J^j_i denotes the maximum jitter, B^j_i denotes the blocking due to lower-priority messages, $I^j_i(\phi^j_i)$ denotes the maximum interference due to higher-priority messages in any interval of length ϕ^j_i, C_i denotes the worst-case transmission time assuming zero transmission errors (which is the same for all replicas), and $E^j_i(n)$ denotes the maximum error overhead due to n retransmissions. Definitions of the delay values B^j_i, C_i, $I^j_i(\phi^j_i)$, and $E^j_i(n)$ are based on the specification of the CAN protocol and can be found in prior work [8, 14, 32].

In contrast to prior work, however, we require aggregate timing guarantees for a *set* of messages. That is, we require a guarantee that, given a set of messages, at least a certain fraction of messages is received on time, *e.g.*, in Example 3, at least one out of three messages must reach on time.

Therefore, we build upon the retransmission-aware response time analysis summarized above to derive a new aggregate timing analysis for sets of messages. In particular, we derive the probability that, given any set of messages $M'_i \subseteq M_i$, each message in M'_i has a worst-case response time less than or equal to its relative deadline, and each message in $M_i \setminus M'_i$ has a worst-case response time exceeding its relative deadline. We impose the condition on messages in $M_i \setminus M'_i$ since our analysis in §V-C is conditioned on the exact set of messages guaranteed to be received by task T_{recv_i} in the worst case.

We introduce a slightly refined notation to denote the worst-case response time. Recall from §II that a crashed system experiences an interval in which messages are continuously omitted. Thus, given a set of crashed hosts H', we assume that, for each task $T^j_{send_i}$ executing on one of the hosts in H', **(i)** message M^j_i is omitted and it's response time is considered to be infinity, and **(ii)** omitted messages do not interfere with the transmission of other messages on the CAN bus.

To reflect both (i) and (ii), we use the modified notation $R^j_i(H', n)$ to denote the worst-case response time of message M^j_i, assuming that M^j_i is delayed by n retransmissions and that hosts in H' are unavailable due to crash failures at the time of release of M^j_i and during the transmission window, *i.e.*, each host in H' crashed and is unavailable during the entire time that M^j_i is queued for transmission.

Based on $R^j_i(H', n)$, we can bound the probability of a timely transmission by enumerating all possible H'.

Lemma 1. *Given a set of hosts $H'_i \subseteq H$ that are unavailable due to crash failures, a set of messages $M'_i \subseteq M_i$, and a relative deadline $\delta \leq D_i$, the probability that (in the worst case and despite retransmissions) all messages in M'_i have worst-case response times less than or equal to δ, whereas all messages in $M_i \setminus M'_i$ have worst-case response times greater than δ, is given by*

$$P_{timely}(H'_i, M'_i, \delta) = \sum_{n \in \mathbb{N} \cup \{0\} \wedge \Lambda_n = M'_i} Poisson(n, \delta, \lambda_{CAN}),$$

where $\Lambda_n = \{M^j_i \mid R^j_i(H'_i, n) \leq \delta\}$.

Proof. Let $n \in \mathbb{N} \cup \{0\}$ denote the number of retransmissions occurring in an interval of length δ. We evaluate each value of n separately as an independent case.

Let α denote the conditional probability given n that all messages in M'_i have worst-case response times less than or equal to δ, whereas all messages in $M_i \setminus M'_i$ have worst-case response times greater than δ. Assuming that each message $M^j_i \in M_i$ is delayed by n retransmissions, the set of messages in M_i with worst-case response times less than or equal to δ is given by $\Lambda_n = \{M^j_i \mid R^j_i(H'_i, n) \leq \delta\}$. Thus, if $\Lambda_n = M'_i$, $\alpha = 1$; otherwise, $\alpha = 0$.

Recall from §III that n retransmissions occur during an interval of length δ with a probability of at most

$Poisson(n, \delta, \lambda_{CAN})$. Using the theorem of total probability,

$$P_{timely}(H'_i, M'_i, \delta) = \sum_{n \in \mathbb{N} \cup \{0\}} \left(Poisson(n, \delta, \lambda_{CAN}) \cdot \alpha\right)$$

{ignoring the case where $\alpha = 0$}

$$= \sum_{n \in \mathbb{N} \cup \{0\} \wedge \Lambda_n = M'_i} \left(Poisson(n, \delta, \lambda_{CAN}) \cdot 1\right). \quad \square$$

B. Accounting For Crash Failures

In the analysis of message $M_{i,k}$, suppose that any task $T^j_{send_i}$ (which transmits message $M^j_{i,k}$) is executing on host H_a. If host H_a becomes unavailable due to a crash failure before message $M^j_{i,k}$ is queued for transmission, we assume that message $M^j_{i,k}$ is omitted in the worst case. The following lemma derives the probability of this case.

In particular, the following lemma derives the probability for a more generic scenario, *i.e.*, the probability that all hosts in $H' \subseteq H$ are unavailable during any interval of length δ due to crash failures, whereas all hosts in $H \setminus H'$ are available during this interval. Recall that ζ_{H_i} denotes the maximum duration for which host H_i remains unavailable after a crash failure.

Lemma 2. *The probability that, during any interval of length δ, hosts $H' \subseteq H$ are unavailable due to crash failures, whereas hosts $H \setminus H'$ are available during this interval, is given by*

$$P_{crash}(H', \delta) = \left(\prod_{H_i \in H'} \Phi^i_{crash}\right) \cdot \left(\prod_{H_i \in H \setminus H'} \Phi^i_{not\text{-}crash}\right),$$

$$\textit{where} \quad \Phi^i_{not\text{-}crash} = Poisson(0, \zeta_{H_i} + \delta, \lambda_{H_i, crash}),$$
$$\textit{and} \quad \Phi^i_{crash} = 1 - \Phi^i_{not\text{-}crash}.$$

Proof. Consider an arbitrary interval $(t, t + \delta]$ of length δ. Consider host H_i. Assume that host H_i remains unavailable for exactly ζ_{H_i} time units immediately after a crash (a worst-case assumption). Thus, host H_i is unavailable during the interval $(t, t + \delta]$ if it crashed at least once in the interval $(t - \zeta_{H_i}, t + \delta]$. The probability of this event is given by $\Phi^i_{crash} = \sum_{n \geq 1} Poisson(n, \zeta_{H_i} + \delta, \lambda_{H_i, crash})$. Similarly, if host H_i does not crash during the interval $(t - \zeta_{H_i}, t + \delta]$, it is available during the interval $(t, t + \delta]$. The probability of this event is $\Phi^i_{not\text{-}crash} = Poisson(0, \zeta_{H_i} + \delta, \lambda_{H_i, crash})$, where $\Phi^i_{crash} = 1 - \Phi^i_{not\text{-}crash}$ for each host H_i.

Per-host probabilities can be safely multiplied assuming that EMI-induced crash failures are independent across hosts. $\square$

C. Accounting for Commission Failures

In the following, we derive the probability that message $M^j_{i,k}$ is corrupted due to commission failures on host H_a, where H_a denotes the host executing task $T^j_{send_i}$.

Lemma 3. *Suppose that H_a denotes the host executing task $T^j_{send_i}$. An upper bound on the probability that any message $M^j_{i,k}$ that is transmitted on or before its deadline $d_{i,k}$ is corrupted due to one or more commission failures is given by*

$$P_{corrupt}(M^j_{i,k}) = 1 - Poisson(0, e^j_{i,k}, \lambda_{H_a, com}).$$

Proof. As discussed in §II, message $M^j_{i,k}$ can be corrupted due to host commission failures occurring any time during its exposure interval of length $e^j_{i,k}$. Thus, from §III, the probability that at least one commission failure occurs on host H_a in the exposure interval is given by $\sum_{n \geq 1} Poisson(n, e^j_{i,k}, \lambda_{H_a, com})$. In the worst case, each commission failure results in the corruption of message $M^j_{i,k}$. Thus, $\sum_{n \geq 1} Poisson(n, e^j_{i,k}, \lambda_{H_a, com})$ also bounds the probability that message $M^j_{i,k}$ is corrupted due to one or more commission failures, where $\sum_{n \geq 1} Poisson(n, e^j_{i,k}, \lambda_{H_a, com}) = 1 - Poisson(0, e^j_{i,k}, \lambda_{H_a, com})$. $\square$

Next, we derive the probability that, given the set of messages received by task T_{recv_i} on time, T_{recv_i} assigns to $M^{final}_{i,k}$ a message value that was not corrupted by a commission failure. We assume that task T_{recv_i} is executing either of the two protocols illustrated in Algorithms 1 and 2, and derive a probability for each protocol separately.

Lemma 4. *Suppose task T_{recv_i} executes Algorithm 1. Let M'_i denote the set of messages received by task T_{recv_i} on time out of the messages in M_i (i.e., each message $M^j_{i,k} \in M'_i$ is received on or before time $d_{i,k} - \Delta_{clk}$). A lower bound on the probability that the final message value inferred by T_{recv_i} is not corrupted by a commission failure is*

$$P^{algo\text{-}1}_{correct}(M'_i) = \sum_{M''_i \subseteq M'_i} \left(\Phi_{incorrect} \cdot \Phi_{correct} \cdot \alpha\right),$$

$$\textit{where} \quad \Phi_{incorrect} = \prod_{M^j_{i,k} \in M''_i} P_{corrupt}(M^j_{i,k}),$$

$$\Phi_{correct} = \prod_{M^j_{i,k} \in M'_i \setminus M''_i} \left(1 - P_{corrupt}(M^j_{i,k})\right),$$

$$\textit{and} \quad \alpha = \begin{cases} 1 & \textit{if } |M''_i| < \left\lfloor \frac{|M'_i|}{2} \right\rfloor + 1, \\ \frac{1}{2} & \textit{if } |M''_i| = \frac{|M'_i|}{2}, \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. By Lemma 3, each message $M^j_{i,k}$ is corrupted by a commission failure with probability at most $P_{corrupt}(M^j_{i,k})$. Under the assumption that each commission failure is independent, the worst-case probability that messages in M''_i are corrupted, whereas messages in $M'_i \setminus M''_i$ are not corrupted, is given by $\Phi_{incorrect} \cdot \Phi_{correct}$.

Recall from Algorithm 1 that, when the timer expires, the message value occurring with the maximum frequency is selected as the final message value (with ties broken uniformly at random). Based on this maximum-frequency procedure, for all $M''_i \subseteq M'_i$, we let α denote a lower bound on the conditional probability that a correct inference is made given M'_i.

Case (i): The inferred message value is guaranteed to be correct if there are fewer corrupted messages than correct messages, *i.e.*, if $0 \leq |M''_i| < \left\lfloor \frac{|M'_i|}{2} \right\rfloor + 1$. Therefore, $\alpha = 1$.

Case (ii): If there are as many corrupted messages as there are correct messages, and if all the corrupted messages happen to have the same message value (a worst-case assumption), then

there is a tie. In this case, since ties are broken uniformly at random, the inferred message value is correct with a probability of $\frac{1}{2}$. Hence, if $|M_i''| = \frac{|M_i'|}{2}$, then $\alpha = \frac{1}{2}$.

Case (iii): If there are more corrupted messages than correct messages, we assume that the inferred message value is corrupted (a worst-case assumption). Thus, $\alpha = 0$. Since we are deriving a lower bound, this is safe, but (likely) pessimistic.

Using the theorem of total probability, the probability of correct inference is computed by summing over the probability of correct inference for all $M_i'' \subseteq M_i'$. □

Lemma 5. *Suppose task T_{recv_i} executes Algorithm 2. Let M_i' denote the set of messages received by task T_{recv_i} on time (out of the messages in M_i). The probability that the final message value inferred by T_{recv_i} is not corrupted due to any commission failure is*

$$P_{correct}^{algo\text{-}2}(M_i') = \sum_{\substack{M_i'' \subseteq M_i' \\ |M_i''| \leq |M_i'| - r_i'}} \left(\Phi_{incorrect} \cdot \Phi_{correct}\right),$$

$$\textit{where} \quad \Phi_{incorrect} = \prod_{M_{i,k}^j \in M_i''} P_{corrupt}(M_{i,k}^j)$$

$$\textit{and} \quad \Phi_{correct} = \prod_{M_{i,k}^j \in M_i' \setminus M_i''} \left(1 - P_{corrupt}(M_{i,k}^j)\right).$$

Proof. Recall from Algorithm 2 that the first r' messages received by task T_{recv_i} with the same message value are used to select the final message value.

Since $r_i' = \lfloor \frac{r_i}{2} \rfloor + 1$ and $r_i' + r_i' > r_i$, *i.e.*, since more than half of the messages are needed to constitute a quorum (*i.e.*, a group of messages with the same message value) of size at least r_i', there can either be a quorum of r_i' correct messages, or a quorum of r_i' corrupted messages, but not a quorum for both cases simultaneously. Therefore, if at least r_i' messages in M_i' are correct, it is guaranteed that no other group of messages will constitute the first r' messages received by T_{recv_i} with the same message value. In other words, in order to guarantee that the final value inferred by task T_{recv_i} is correct, it is sufficient to guarantee that at least r' messages in set M_i' are correct. We derive the probability for this case in the following.

For at least r_i' messages in M_i' to be correct, there must be at most $|M_i'| - r_i'$ corrupted messages in M_i'. Let M_i'' denote the set of corrupted messages in M_i': then $|M_i''| \leq |M_i'| - r_i'$. Since $P_{corrupt}(M_{i,k}^j)$ is known for each message $M_{i,k}^j$, and assuming that commission failures are independent, the probability that messages in M_i'' are corrupted and messages in $M_i' \setminus M_i''$ are correct is computed trivially. $P_{correct}^{algo\text{-}2}(M_i')$ is computed by summing over all cases $M_i'' \subseteq M_i'$.

Note that if $|M_i'| < r_i'$, the probability of a correct inference is trivially zero, because the protocol in Algorithm 2 does not decide on a final message value, *i.e.*, the condition in line 7 is false. Since $|M_i'| < r_i'$ implies that $|M_i''| \leq |M_i'| - r_i'$ is false (as $|M_i'| - r_i' < 0$), this case is accounted for by considering only those subsets M_i'' where $|M_i''| \leq |M_i'| - r_i'$. □

D. Probability of a Successful Message Transmission

In the following, using Lemmas 1–5, we derive a bound on the probability that a message M_i is successfully transmitted by considering the worst-case scenario for any message $M_{i,k} \in M_i$. Recall from Example 1 that message $M_{i,k}$ is successfully transmitted only if both task T_{recv_i} infers $M_{i,k}^{final}$ not later than its deadline $d_{i,k}$ (*i.e.*, timely inference) and the message value assigned to $M_{i,k}^{final}$ was not derived from a corrupted message (*i.e.*, correct inference). We derive separate bounds for Algorithms 1 and 2, respectively.

Lemma 6. *Suppose task T_{recv_i} executes Algorithm 1. Let J_i^{max} denote the maximum jitter experienced by any message replica M_i^j. A lower bound on the probability that an instance of message M_i is successfully transmitted is given by*

$$P_{success}^{algo\text{-}1}(M_i) = \sum_{H' \subseteq H} \Phi_{crash} \cdot \sum_{M_i' \subseteq M_i} \left(\Phi_{timely} \cdot \Phi_{correct}\right),$$

$$\begin{aligned} \textit{where} \quad & \Phi_{crash} = P_{crash}(H', J_i^{max}), \\ & \Phi_{timely} = P_{timely}(H', M_i', D_i - \Delta_{clk}), \\ \textit{and} \quad & \Phi_{correct} = P_{correct}^{algo\text{-}1}(M_i'). \end{aligned}$$

Proof. Consider any arbitrary message $M_{i,k}$. Replicas of $M_{i,k}$ may be omitted due to crash failures: if task T_i^j is executing on host H_a that is unavailable during $(a_{i,k}, a_{i,k} + J_i^j]$ due to crash failures, $M_{i,k}^j$ is omitted. Therefore, we first perform a case analysis considering each $H' \subseteq H$. From Lemma 2, the probability that hosts in H' are unavailable due to crash failures during $(a_{i,k}, a_{i,k} + J_i^{max}]$, whereas hosts in $H \setminus H'$ are available during $(a_{i,k}, a_{i,k} + J_i^{max}]$, is given by $P_{crash}(H', J_i^{max})$. The conditional probability given H' that $M_{i,k}$ is successfully transmitted is derived next.

According to Algorithm 1, the receiver task infers the final message value $M_{i,k}^{final}$ when timer $\Delta_{i,k}$ expires. Since $\Delta_{i,k} = d_{i,k} - \Delta_{clk}$, all messages transmitted later than time $d_{i,k} - \Delta_{clk}$ are ignored. In other words, any message with a worst-case response time greater than $d_{i,k} - \Delta_{clk} - a_{i,k} = D_i - \Delta_{clk}$ may be ignored. Therefore, we perform another case analysis over all subsets $M_i' \subseteq M_i$, assuming that messages in M_i' have worst-case response times of at most $D_i - \Delta_{clk}$ and messages in $M_i \setminus M_i'$ have worst-case response times exceeding $D_i - \Delta_{clk}$.

From Lemma 1, the probability for each case is given by $\Phi_{timely} = P_{timely}(H', M_i', D_i - \Delta_{clk})$. From Lemma 4, the conditional probability given M_i' that the message value with the maximum frequency (ties broken uniformly at random) is not corrupted is given by $\Phi_{correct} = P_{correct}^{algo\text{-}1}(M_i')$. Therefore, by the theorem of total probability, the conditional probability given H' that M_i is successfully transmitted is $\sum_{M_i' \subseteq M_i} (\Phi_{timely} \cdot \Phi_{correct})$. □

Lemma 7. *Suppose task T_{recv_i} executes Algorithm 2. Let J_i^{max} denote the maximum jitter experienced by any message replica M_i^j. A lower bound on the probability that all messages in*

M_i are successfully transmitted is given by:

$$P_{success}^{algo\text{-}2}(M_i) = \sum_{H' \subseteq H} \Phi_{crash} \cdot \sum_{\substack{M'_i \subseteq M_i \\ |M'_i| \geq r'_i}} \left(\Phi_{timely} \cdot \Phi_{correct}\right),$$

$$\begin{aligned} \textit{where} \quad & \Phi_{crash} = P_{crash}(H', J_i^{max}), \\ & \Phi_{timely} = P_{timely}(H', M'_i, D_i), \\ \textit{and} \quad & \Phi_{correct} = P_{correct}^{algo\text{-}2}(M'_i). \end{aligned}$$

Proof. The proof is similar to the proof of Lemma 6. Consider any arbitrary message $M_{i,k}$. Replicas of $M_{i,k}$ may be omitted due to crash failures: if task T_i^j is executing on host H_a that is unavailable during $(a_{i,k}, a_{i,k} + J_i^j]$, then $M_{i,k}^j$ is omitted. Therefore, we first perform a case analysis considering each $H' \subseteq H$. From Lemma 2, the probability that hosts in H' are unavailable due to crash failures during $(a_{i,k}, a_{i,k} + J_i^{max}]$, whereas hosts in $H \setminus H'$ are available during $(a_{i,k}, a_{i,k} + J_i^{max}]$, is given by $P_{crash}(H', J_i^{max})$. The conditional probability given H' that $M_{i,k}$ is successfully transmitted is derived next.

According to Algorithm 2, the receiver task infers the final message value $M_{i,k}^{final}$ if at least r'_i messages with the same message value have been received. For a successful transmission, $M_{i,k}^{final}$ must be inferred no later than by the deadline $d_{i,k}$. Thus, only the cases where at least r'_i messages have worst-case response times less than or equal to $d_{i,k} - a_{i,k} = D_i$ can guarantee a successful transmission. Therefore, we perform another case analysis considering each subset $M'_i \subseteq M_i$ such that $|M'_i| \geq r'_i$, assuming that messages in M'_i have worst-case response times of at most D_i and that messages in $M_i \setminus M'_i$ have worst-case response times exceeding D_i.

By Lemma 1, the probability for each such case is given by $\Phi_{timely} = P_{timely}(H', M'_i, D_i)$. By Lemma 5, the conditional probability given M'_i that at least r'_i messages in M'_i are not corrupted by any commission failures is given by $\Phi_{correct} = P_{correct}^{algo\text{-}2}(M'_i)$. As argued in the proof of Lemma 5, if at least r'_i messages are not corrupted and have the same message value, there cannot be another quorum of size greater than or equal to r'_i. Thus, $\Phi_{correct}$ also denotes the conditional probability given M'_i that the receiver task makes a correct inference. Therefore, by the theorem of total probability, the conditional probability given H' that M_i is successfully transmitted is $\sum_{M'_i \subseteq M_i \text{ s.t. } |M'_i| \geq r'_i} (\Phi_{timely} \cdot \Phi_{correct})$. □

Complexity. Computing $P_{success}^{algo\text{-}1}(M_i)$ or $P_{success}^{algo\text{-}2}(M_i)$ requires enumerating potentially all subsets of H and M_i. With regard to H, a host is relevant only if at least one of the replicas of M_i originates from it. Thus, since $|M_i| = r_i$ and assuming that all message replicas originate on distinct hosts, the analysis requires 2^{r_i} outermost iterations, and further 2^{r_i} innermost iterations. In total, the analysis thus requires $2^{(2 \cdot r_i)}$ iterations of the computation of $\Phi_{timely} \cdot \Phi_{correct}$ (which each involves response-time analysis). However, given that replication factors are typically small, the analysis remains practical despite its nominally high computational complexity.

Lemmas 6 and 7 only help us to understand the chances that a message M_i is transmitted on time and correctly. In the next section, we derive a system-wide reliability metric to assess if the entire system executes successfully.

VI. Failure-in-Time of the Task System

Recall that the FIT rate is the expected number of failures in one billion operating hours. The procedure to compute the FIT rate for a system with multiple components is as follows: **(i)** for each component, compute its probability density function $f(t)$ that denotes the probability that the component fails for the first time at time t; **(ii)** using $f(t)$, compute the MTBF for each component, where $MTBF = \int_0^\infty t \cdot f(t)\, dt$; **(iii)** given each component's MTBF in hours, its FIT rate is given by $\frac{10^9}{MTBF}$; and, **(iv)** assuming that each component fails independently, the FIT rate for the entire system is given as the sum of all the component-wise FIT rates (see [33] for an industry tutorial and real-world example calculations).

In the following, we derive the FIT rate assuming that all messages arrive periodically and that hosts have synchronized clocks, using the four-step process described above. Thereafter, we briefly discuss the remaining cases.

Step (i): For each message M_i, we derive the probability density function as a function of the number of runtime activations x of the message, rather than as a function of time. From Lemma 6, the worst-case probability that any particular instance of message M_i is successfully transmitted is given by $P_{success}^{algo\text{-}1}(M_i)$. Thus, the probability that $x-1$ runtime activations of message M_i, *i.e.*, messages $M_{i,1}, M_{i,2}, \ldots, M_{i,x-1}$, are successfully transmitted whereas the x^{th} instance of message M_i, *i.e.*, message $M_{i,x}$, is not successfully transmitted is bounded by $f_i(x) = \left(P_{success}^{algo\text{-}1}(M_i)\right)^{x-1} \cdot \left(1 - P_{success}^{algo\text{-}1}(M_i)\right)$.

Steps (ii) and (iii): Let $MABF_i$ denote the *Mean number of Activations Between Failures* for message M_i (similar to the MTBF, but in terms of the number of runtime activations rather than time). Like the MTBF, $MABF_i$ is defined as follows: $MABF_i = \int_0^\infty x \cdot f_i(x)\, dx$. Assuming that the period P_i of messages in M_i is given in milliseconds, the MTBF for the successful transmission of M_i is defined as $MTBF_i = (MABF_i \times P_i)/(1000 \times 3600)$ hours. The FIT rate of message M_i is trivially given by $FIT_i = 10^9/MTBF_i$.

Step (iv): For any two messages $M_a, M_b \in M$, the probabilities that they are successfully transmitted, *i.e.*, $P_{success}^{algo\text{-}1}(M_a)$ and $P_{success}^{algo\text{-}1}(M_b)$, respectively, are derived using worst-case assumptions and are thus not dependent on each other. Therefore, the respective FIT rates for messages M_a and M_b, which are derived from the failure probabilities, can also be considered to be mutually independent. Based on this assumption, the FIT rate for the entire task system is defined as the sum of all the message-specific FIT rates, as follows: $FIT_{system} = \sum_{M_i \subseteq M} FIT_i$.

Our approach yields an upper bound on the actual FIT rate since it is based on the probabilistic analysis proposed in §V, which is defined under worst-case assumptions (which has the benefit of ensuring the required independence). However, at runtime, not every message instance incurs worst-case conditions in practice. Therefore, the actual observed FIT rate can be expected to be lower than predicted by our analysis. Given that we seek to bound failures of possibly safety-critical components,

a safe, but inexact bound is acceptable, and certainly preferable to a potentially optimistic FIT bound.

Generally speaking, the FIT rate computation derived assuming periodic messages and hosts with synchronized clocks can also be repeated for any system that uses Algorithm 2, by using $P_{success}^{algo\text{-}2}(M_i)$ in place of $P_{success}^{algo\text{-}1}(M_i)$. However, for the scenario where messages arrive sporadically, the derived FIT rate could be even more pessimistic. This is because the equation $MTBF_i = (MABF_i \times P_i)/(1000 \times 3600)$ hours assumes that runtime activations of messages happen with maximum frequency, whereas the average frequency of sporadic messages may be much lower in practice. Nonetheless, our procedure still yields a safe bound on the actual FIT rate.

VII. Experiments

As discussed in §I, the FIT rate is a standard industry metric used to characterize both component and product reliability. While it is our hope that the proposed FIT-rate bound will prove useful in the analysis and benchmarking of various real-time system design issues (*e.g.*, such as how to allocate slack, or how to pick optimal replica counts), the focus of our evaluation is to assess the proposed FIT derivation itself, and not the utility of a FIT rate. In particular, we want to evaluate whether the derived bound on the FIT rate works, in the sense of yielding non-obvious insights into a practical task system, or whether the proposed analysis is too coarse-grained to be useful.

To this end, we based our experiments on a message set extracted from a simple mobile robot, which was previously also used as a benchmark by Broster et al. [7]. The message set consists of six messages, with a total bus utilization of roughly 40%. We denote the original message set as $M^{40pc} = \{M_1, M_2, \ldots, M_6\}$, with periods $2ms$, $4ms$, $4ms$, $8ms$, $12ms$, and $24ms$, respectively. We consider message M_1, which is sent by the "MotorCtrl" unit, to be a critical task. Without breaking (deterministic) schedulability guarantees of lower-priority messages, M_1 can be replicated up to $r_i^{40pc} = 4$ times. To evaluate larger replica counts, we derived a second message set M^{30pc}, with roughly 30% total bus utilization (prior to replication) and periods $4ms$, $6ms$, $6ms$, $8ms$, $12ms$, and $24ms$, respectively. The second message set allows replicating message M_1 up to $r_i^{30pc} = 9$ times.

To assign message priorities, we used Davis and Burns's robust priority assignment algorithm [13] to maximize the number of tolerated retransmissions, which resulted in M_1 receiving the highest priority. Note that Davis and Burns's algorithm does not assume task replication. While their proof of optimality [13] thus does not extend to our replicated system model (where not necessarily all messages in the message set need to be received in time), we believe that it is still a good heuristic. Finding an optimal priority assignment in the presence of replicated messages, *i.e.*, a priority assignment that minimizes the overall FIT, is an interesting open problem.

A. Experiment 1 – Analysis vs. Simulation

In the first experiment, we determined the probability that a message transmission fails both analytically (using the proposed analysis) and empirically (using a simulator). For messages M_1 and M_4 in message set M^{30pc}, Fig. 2 shows how the probability of failure varies as the replication factor of message M_1 is increased. The curves for the remaining lower-priority messages are similar to that of M_4 and have been omitted from the graph to avoid clutter. The probability of failure assuming message set M^{40pc} also follows a similar trend.

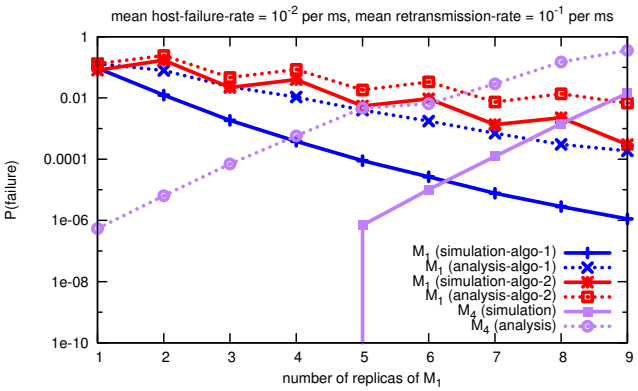


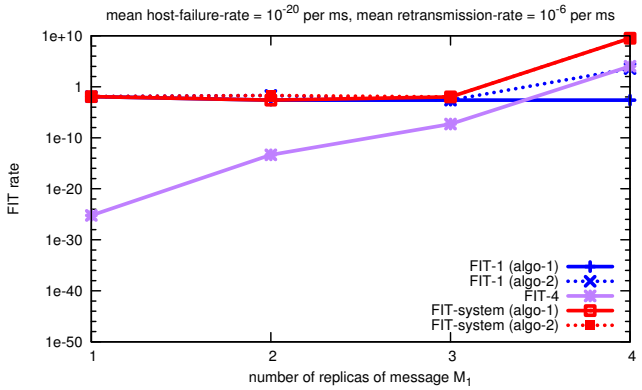
Fig. 2. Probability of transmission failure for messages M_1 and M_4 in message set M^{30pc} as a function of M_1's replication factor. Dashed curves show probabilities derived analytically; solid curves indicate simulation results. Separate curves are plotted for message M_1 assuming Algorithms 1 and 2, respectively. Exceptionally high failure rates $\lambda_{host} = 10^{-2}$ faults/ms and $\lambda_{CAN} = 10^{-1}$ faults/ms were used in this experiment as sampling limitations prevented the simulation from measuring probabilities lower than 1^{-09}.

There are two main observations from this experiment. First, the proposed analysis is *safe*, in that it overestimates the probability of failure (the dashed analysis curves are always above the corresponding solid simulation curves). And second, the proposed analysis is *sufficiently accurate* to track the simulation results, in particular for higher-priority messages. In general, given that we have made several worst-case assumptions (*e.g.*, in Lemmas 6 and 7, and by assuming that EMI on the bus *always* causes a retransmission), the analysis and simulation diverges somewhat when analysis results of multiple messages are taken into account (*e.g.*, at high replica counts, or for lower-priority messages that are delayed by all higher-priority retransmissions).

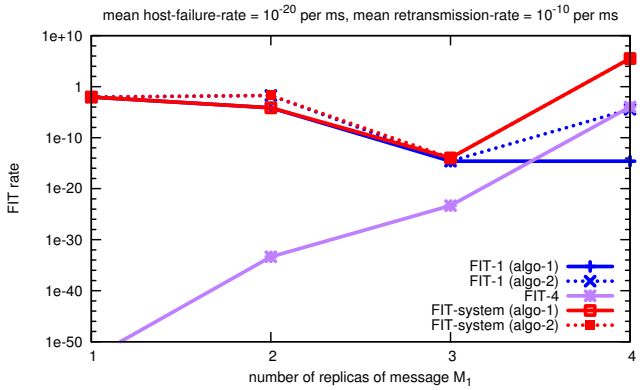
Another interesting trend in Fig. 2 is the distinct sawtooth pattern that the curves for message M_1 exhibit when Algorithm 2 is used by the receiver task. This can be attributed to two key properties of Algorithm 2: **(i)** the probability of a correct inference depends only on the set of r_i' messages with identical payload received first by the receiver task (and not directly on r_i), and **(ii)** since $r_i' = \lfloor \frac{r_i}{2} \rfloor + 1$, the probability to receive r_i' out of r_i messages is higher for odd values of r_i than it is, relatively speaking, for even values of r_i. For example, the receiver is more likely to receive four out of seven messages on time than it is to receive four out of six messages on time. In other words, when waiting to receive an unambiguous *majority* of identical messages, it is beneficial if the total number of messages is odd. At the same time, the bus load grows with increasing r_i, which causes the uptick in failure probability for even values for r_i.

B. Experiment 2 – FIT Analysis

In the second set of experiments, we determined message-specific FIT rates and the system-wide FIT rate as a function

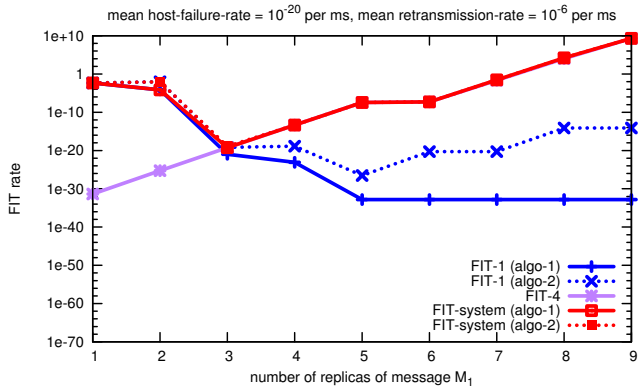


(a) $\lambda_{CAN} = 10^{-6}$ faults/ms, $\lambda_{host} = 10^{-20}$ faults/ms

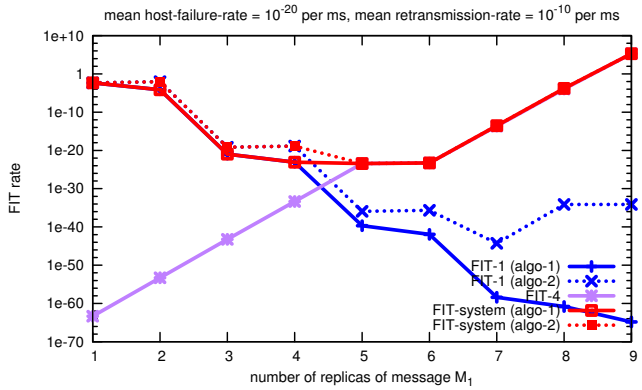


(b) $\lambda_{CAN} = 10^{-10}$ faults/ms, $\lambda_{host} = 10^{-20}$ faults/ms

Fig. 3. FIT rates for messages M_1 and M_4 (labelled FIT-1 and FIT-4, respectively) and for the entire system (labelled FIT-system) for message set M^{40pc} as a function of r_1. Solid curves show FIT rates using Algorithm 1; dashed curves correspond to Algorithm 2.



(a) $\lambda_{CAN} = 10^{-6}$ faults/ms, $\lambda_{host} = 10^{-20}$ faults/ms



(b) $\lambda_{CAN} = 10^{-10}$ faults/ms, $\lambda_{host} = 10^{-20}$ faults/ms

Fig. 4. FIT rates for messages M_1 and M_4 (labelled FIT-1 and FIT-4, respectively) and for the entire system (labelled FIT-system) for message set M^{30pc} as a function of r_1. Solid curves show FIT rates using Algorithm 1; dashed curves correspond to Algorithm 2.

of the replication factor of message M_1. We repeated the experiments for both message sets and assuming different values for the host failure and retransmission rates. Representative example graphs are shown in Fig. 3 for message set M^{40pc} and in Fig. 4 for message set M^{30pc}.

While Figs. 3(a) and 4(a) reflect a high retransmission rate of $\lambda_{CAN} = 10^{-6}$ faults/ms, Figs. 3(b) and 4(b) correspond to a lower retransmission rate of $\lambda_{CAN} = 10^{-10}$ faults/ms. These rates are in accordance with prior studies on the bit-fault rate in CAN, *e.g.*, according to Ferreira et al. [15] and Rufino et al. [27], typical values of λ_{CAN} range from 10^{-4} faults/ms in aggressive environments to 10^{-10} faults/ms in lab conditions. All graphs assume a host failure rate of $\lambda_{host} = 10^{-20}$ faults/ms for all hosts and for both crash failures and commission failures; we assume that failure rates in ECC- and lockstep-protected hosts are typically orders of magnitude smaller than the bit-fault rate in CAN (since cabling can be expected to be more exposed to EMI). As in Fig. 2, results for messages other than the highest-priority message M_1 and the lower-priority message M_4 have been omitted for clarity.

Optimal replication factor for message M_1. First of all, all graphs clearly resemble the conceptual graph in Fig. 1(c) (note that the "U" shape is flipped, *i.e.*, Fig. 1(c) shows probability of success, whereas Figs. 3 and 4 show FIT). The optimal replication factor for M_1 can thus be readily identified: while for the message set M^{40pc}, all graphs indicate that $r_1 = 3$ yields the minimal system-wide FIT rate, in the case of message set M^{30pc}, the result varies with the failure parameters. Case in point, Fig. 4 indicates that the optimal number of replica of message M_1 is on the higher side (*i.e.*, around 5 or 6) if the mean rate of retransmission is lower.

Range of derived FIT rates for M. An interesting observation can be made in Fig. 4: there can be a difference of more than 20 orders of magnitude between the maximum and minimum system-wide FIT rates when the replication factor for M_1 is varied from 1 to 7 (note the log scale). In general, the results strongly suggest that a systematic approach towards system reliability, *e.g.*, using the proposed FIT rate analysis, can make a large difference. For instance, in the case of $\lambda_{CAN} = 10^{-10}$ faults/ms shown in Fig. 4(b), the difference between a good system design (*i.e.*, selecting $r_1 = 5$ or $r_1 = 6$) vs. a bad system design (*i.e.*, selecting $r_1 = 1$ or $r_1 = 9$) results in substantial variations of the overall failure rate.

Our results show that the derived FIT-rate bound is accurate enough to discuss key system design questions, and to yield non-obvious results (*e.g.*, that $r_1 = 3$ is optimal for M^{40pc}, but

that $r_1 = 5$ can be preferable for M^{30pc}, depending on EMI strength). In future work, we plan to carry out more extensive studies using a larger set of benchmarks.

VIII. Discussion and Related Work

In this section, we elaborate on our choices, *e.g.*, the failure types and corresponding models, model assumptions, *etc.*, and justify our approach in the context of prior studies.

Recall that we considered three types of failures: transmission failures, crash failures, and commission failures. While transmission failures on a CAN bus are well documented in the literature [8, 14, 18, 23, 25, 32], there are no generic studies available on host failures that justify crashes and commission faults as the right abstraction for host failures. Therefore, we refer to studies based on specific types of hosts in order to justify these abstractions. For example, Frantz et al. [16] evaluated the effects of soft errors in a network-on-chip router and showed that they can manifest in the form of lost packets, routing errors, payload errors, packet formation errors, and router crash. Rebaudengo et al. [26] and Azkarate-askasua et al. [2] studied the effects of transient faults on processor caches and on time-triggered system-on-chip architectures, respectively, and showed that they can manifest in the form of exceptions, timeouts, stalls, and wrong answers. We believe that crash failures and commission failures cover a majority of these fault effects under a simple abstraction.

The probabilistic fault models described in §III follow recent work. Like Broster et al. [8], and as in all prior probabilistic analyses of the CAN bus [6, 23], we assume independence among transient faults. Thus, their inter-arrival times can be modeled with an exponential density function, forming a Poisson distribution for the faults [1]. We use a similar model for EMI-induced host failures as well. This choice is reasonable since real-time tasks are repeated, short workloads; thus, any job is equally likely to be affected by a commission failure, and a host is equally likely to be crashed at any point in time (see [19] for a mathematical basis for this argument).

When describing the system model in §II, we made an implicit assumption that task replication is a useful mechanism to tolerate transient faults on the hosts. This view is supported by prior studies (*e.g.*, [3]), which have noted that system failure rates can be reduced to near-zero levels with active replication, providing the necessary reliability for some long-term remote or mission-critical applications.

Nonetheless, it is interesting to note that prior work, *e.g.*, [9, 17, 21, 24], has often assumed a system model based on *temporal redundancy*, *i.e.*, similar to the retransmission mechanism on a CAN bus, where any job affected by a fault is re-executed. However, research has shown that immediate detection of erroneous tasks is not always feasible, and that faults propagate and corrupt memory [10, 11, 20]. Recently, Song and Parmer [29] proposed an efficient and predictable system-level monitoring framework to tackle the problem of undetected errors. Their solution is in line with prior work that focuses on building robust operating systems as a means to improve fault-tolerance, *e.g.*, [12, 30, 31].

In general, these techniques are *complementary* to the active replication studied herein: active replication helps provide another layer of redundancy (and also helps against permanent faults, against which host-level approaches are powerless), which is generally desirable and aids with implementing a defense-in-depth approach. Therefore, we see host-level approaches as a mechanism that (greatly) reduces the derating factor (which reflects a host's vulnerability to EMI).

IX. Conclusions and Future Work

We have proposed a method to derive a FIT rate for CAN-based distributed real-time systems. To the best of our knowledge, this is the first such attempt. We believe that FIT rates, a standard reliability metric used widely in industry, are beneficial to assess the reliability, both in a logical and in a timeliness sense, of embedded real-time systems in the context of system-level reliability goals.

In future work, we seek to investigate ways to reduce the remaining pessimism in our analysis (*e.g.*, by analyzing entire hyperperiods as a whole). Furthermore, we believe that the proposed FIT rate analysis is an excellent tool to investigate many diverse CAN-related system design challenges.

References

[1] R. B. Ash, *Basic probability theory*. Courier Corporation, 2012.

[2] M. Azkarate-askasua, I. Martinez, X. Iturbe, and R. Obermaisser, "Dependability assessment of the time-triggered SoC prototype using FPGA fault injection," in *IECON*, 2011.

[3] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, 2005.

[4] C. Bosch, "Specification version 2.0," *Published by Robert Bosch GmbH (September 1991)*, 1991.

[5] I. Broster and A. Burns, "An analysable bus-guardian for event-triggered communication," in *RTSS 2003*.

[6] I. Broster, A. Burns, and G. Rodríguez-Navas, "Probabilistic analysis of CAN with faults," in *RTSS 2002*.

[7] I. Broster, A. Burns, and G. Rodriguez-Navas, "Comparing real-time communication under electromagnetic interference," in *ECRTS*, 2004.

[8] I. Broster, A. Burns, and G. Rodríguez-Navas, "Timing analysis of real-time communication under electromagnetic interference," *Real-Time Systems*, 2005.

[9] A. Burns, R. Davis, and S. Punnekkat, "Feasibility analysis of fault-tolerant real-time task sets," in *ECRTS*, 1996.

[10] S. Chandra and P. M. Chen, "How fail-stop are faulty programs?" in *FTCS*, 1998.

[11] P. Chevochot and I. Puaut, "Experimental evaluation of the fail-silent behavior of a distributed real-time run-time support built from COTS components," in *DSN*, 2001.

[12] F. M. David, E. Chan, J. C. Carlyle, and R. H. Campbell, "CuriOS: Improving reliability through operating system structure." in *OSDI*, 2008.

[13] R. I. Davis and A. Burns, "Robust priority assignment for messages on controller area network (CAN)," *Real-Time Systems*, 2009.

[14] 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*, 2007.

[15] J. Ferreira, A. Oliveira, P. Fonseca, and J. Fonseca, "An experiment to assess bit error rate in CAN," in *RTN*, 2004.

[16] A. P. Frantz, L. Carro, E. Cota, and F. L. Kastensmidt, "Evaluating SEU and crosstalk effects in network-on-chip routers," in *IOLTS*, 2006.

[17] S. Ghosh, R. Melhem, and D. Mosse, "Enhancing real-time schedules to tolerate transient faults," in *RTSS*, 1995.

[18] H. A. Hansson, T. Nolte, C. Norstrom, and S. Punnekkat, "Integrating reliability and timing analysis of CAN-based systems," *IEEE Transactions on Industrial Electronics*, 2002.

[19] X. Li, S. V. Adve, P. Bose, and J. A. Rivers, "Architecture-level soft error analysis: Examining the limits of common assumptions," in *DSN*, 2007.

[20] H. Madeira and J. G. Silva, "Experimental evaluation of the fail-silent behavior in computers without error masking," in *FTCS-24*, 1994.
[21] P. Mejía-Alvarez and D. Mossé, "A responsiveness approach for scheduling fault recovery in real-time systems," in *RTAS*, 1999.
[22] S. S. Mukherjee, C. Weaver, J. Emer, S. K. Reinhardt, and T. Austin, "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," in *MICRO-36*, 2003.
[23] N. Navet, Y.-Q. Song, and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over controller area network," *Journal of Systems Architecture*, 2000.
[24] S. Punnekkat and A. Burns, "Analysis of checkpointing for schedulability of real-time systems," in *RTCSA*, 1997.
[25] S. Punnekkat, H. Hansson, and C. Norstrom, "Response time analysis under errors for CAN," in *RTAS*, 2000.
[26] M. Rebaudengo, M. Sonza Reorda, and M. Violante, "An accurate analysis of the effects of soft errors in the instruction and data caches of a pipelined microprocessor," in *DATE*, 2003.
[27] J. Rufino, P. Verissimo, G. Arroz, C. Almeida, and L. Rodrigues, "Fault-tolerant broadcasts in CAN," in *FTCS*. IEEE, 1998.
[28] P. Shirvani, N. Saxena, and E. McCluskey, "Software-implemented EDAC protection against SEUs," *IEEE Transactions on Reliability*, vol. 49, no. 3, pp. 273–284, 2000.
[29] J. Song and G. Parmer, "C'MON: a predictable monitoring infrastructure for system-level latent fault detection and recovery," in *RTAS*, 2015.
[30] J. Song, J. Wittrock, and G. Parmer, "Predictable, efficient system-level fault tolerance in C^3," in *RTSS*, 2013.
[31] M. M. Swift, B. N. Bershad, and H. M. Levy, "Improving the reliability of commodity operating systems," in *SOSP*, 2003.
[32] K. Tindell and A. Burns, "Guaranteeing message latencies on control area network (CAN)," in *International CAN Conference*, 1994.
[33] J. Trinh and D. Ly, "FIT data and MTTF/MTBF," TDK Corp., Whitepaper, available at http://product.tdk.com/capacitor/mlcc/en/faq/pdf/24_fit_data_and_mttfmtbf.pdf, 2013.
[34] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. Patel, "Characterizing the effects of transient faults on a high-performance processor pipeline," in *DSN*, 2004.

APPENDIX

A. Correlated Host Failures

The analysis in §V fundamentally assumes that failures on different hosts are uncorrelated. In this appendix, we briefly elaborate on alternatives if this assumption is invalidated.

Fundamentally, there are three different failure scenarios: **(i)** independent host faults without any correlation (*e.g.*, due to atmospheric background EMI); **(ii)** external events that cause potentially catastrophic damage to the entire system and thus affect all hosts at the same time (*e.g.*, a massive power surge or a small UAV hit by a lightening strike); and **(iii)** intermediate cases, where failures affecting groups of one or more hosts may be correlated (*e.g.*, a large plane subject to a lightening strike, or a system-on-a-chip in which several logically independent hosts are located on the same silicon die).

The analysis in §V covers case (i); we consider cases (ii) and (iii) in the following. Correlated failures of type (ii) can be simply handled as a separate failure process. To derive a whole-system FIT, all sources of failures must be considered. Whereas we have focused on the CAN bus in this paper, in a real product, other concerns such as failure of power supplies, material fatigue, thermal wear-out, *etc.* must be taken into account—a typical embedded system may fail for a large number of different reasons. As discussed in §I and §VI, this can be addressed by deriving a FIT rate for each failure source and by then adding the individual rates. Thus, correlated whole-system effects do not have to be integrated into our analysis for *localized* EMI effects, but rather should be dealt with by deriving the FIT (or its inverse, the MTBF) for catastrophic events, which can then be directly considered in the whole-system reliability analysis.

Correlated failures of type (iii) are more challenging to address. First, we focus on specific design-induced correlations, *e.g.*, when two or more (logically independent) hosts are located on the same silicon die, or when they share the same power source, or when different hosts share a single CAN controller or CAN bridge, in which case a fault in the CAN controller or bridge makes all connected hosts unavailable simultaneously, *etc.* To handle such cases, we partition the set of all hosts into *equivalence classes* such that hosts in the same class may have correlated failures, whereas hosts in different classes are expected to not be correlated. We then make a simplifying worst-case assumption that, if a host crashes, then all hosts in the equivalence class may crash at the same time.

The equivalence classes are denoted as $E_1, E_2, \ldots$ such that $\forall i, j, E_i \cap E_j = \emptyset$ and $\bigcup_i E_i = H$. Based on this assumption, it is possible to formulate an alternative to Lemma 2. We assume that the mean rate of crash failures for each class E_i can be derived empirically and is denoted as $\lambda_{E_i,crash}$; and that the maximum duration for which any host in E_i may remain unavailable due to a crash failure is denoted as ζ_{E_i}.

Lemma 8. *The probability that, during any interval of length δ, hosts $H' \subseteq H$ are unavailable due to crash failures, whereas hosts $H \setminus H'$ are available during this interval, is given by*

$$P_{crash}(H', \delta) = \left(\prod_{E_i \subseteq H'} \Phi^i_{crash} \right) \left(\prod_{E_i \subseteq H \setminus H'} \Phi^i_{not\text{-}crash} \right) \beta,$$

$$\textit{where} \quad \Phi^i_{not\text{-}crash} = Poisson(0, \zeta_{E_i} + \delta, \lambda_{E_i,crash}),$$
$$\Phi^i_{crash} = 1 - \Phi^i_{not\text{-}crash}, \ \textit{and} \ \beta = 1 - \prod_{\substack{E_i \cap H' \neq \emptyset \\ E_i \cap H \setminus H' \neq \emptyset}} 1.$$

Proof. Similar to Lemma 2, except for the factor of β. For any equivalence class E_i, since all hosts in E_i are assumed to crash together, it is not possible that some hosts in E_i are in H' and some hosts in E_i are in $H \setminus H'$. Thus, if any E_i intersects both H' and $H \setminus H'$, then $\beta = 0$. □

Lemma 8 addresses host crashes; correlated message corruption failures can be accounted for similarly based on equivalence classes. The corresponding changes in Lemmas 4 and 5 are omitted due to space constraints.

Finally, one may consider correlation due to physical proximity. For example, consider a large plane. In the case of a lightening strike, or when intersecting a powerful radar beam, it is plausible to assume that physically close hosts are more likely to exhibit correlated EMI effects than physically distant hosts. If a proper correlation matrix for such events can be established—a non-trivial data collection challenge—then it would be interesting to extend the proposed analysis to integrate correlation into the probabilistic model, which we leave as future work.