Quantifying the Resiliency of Fail-Operational Real-Time Networked Control Systems

Arpan Gujarati, Mitra Nasri, Björn B. Brandenburg
Embedded systems are susceptible to environmentally-induced **transient faults**

- **Harsh environments**
  - Robots operating under **hard radiation**
  - Industrial systems near **high-power machinery**
  - **Electric motors** inside automobile systems

- **Bit-flips in registers, buffers, network**
Embedded systems are susceptible to environmentally-induced transient faults

- Harsh environments
  - Robots operating under hard radiation
  - Industrial systems near high-power machinery
  - Electric motors inside automobile systems

- Bit-flips in registers, buffers, network

Example*


- One bit-flip in a 1 MB SRAM every $10^{12}$ hours of operation
- 0.5 billion cars with an average daily operation time of 5%
- About 5,000 cars are affected by a bit-flip every day
Failures and errors due to transient faults in distributed real-time systems
Failures and errors due to transient faults in distributed real-time systems

- Transmission errors
  - Faults on the network

- Omission Errors
  - Fault-induced kernel panics

- Incorrect computation Errors
  - Faults in the memory buffers
Failures and errors due to transient faults in distributed real-time systems

- Transmission errors
  - Faults on the network

- Omission Errors
  - Fault-induced kernel panics

- Incorrect computation Errors
  - Faults in the memory buffers

Failures in:
- value domain (incorrect outputs)
- time domain (deadline violations)

E.g., safety-critical control system

- Incorrect, delayed, or skipped
Mitigating the effects of transient faults in distributed real-time systems
Mitigating the effects of transient faults in distributed real-time systems

- **Transmission errors**
  - Faults on the network
  - Retransmissions at the network layer

- **Omission Errors**
  - Fault-induced kernel panics
  - Dual modular redundancy (DMR)

- **Incorrect computation Errors**
  - Faults in the memory buffers
  - Triple modular redundancy (TMR)
Mitigating the effects of transient faults in distributed real-time systems

How can we objectively compare the reliability offered by different mitigation techniques?

- **Omission Errors**
  - Fault-induced kernel panics

- **Incorrect computation Errors**
  - Faults in the memory buffers

- **Retransmissions at the network layer**

- **Dual modular redundancy (DMR)**

- **Triple modular redundancy (TMR)**
Mitigating the effects of transient faults in distributed real-time systems

- Transmission errors ➡ Faults on the network
- Omission Errors ➡ Fault-induced kernel panics
- Incorrect computation Errors ➡ Faults in the memory buffers

How does the real-time requirement affect system reliability? When does it really become a bottleneck?

- Dual modular redundancy (DMR)
- Triple modular redundancy (TMR)
Mitigating the effects of transient faults in distributed real-time systems

How does the real-time requirement affect system reliability? When does it really become a bottleneck?

What if the system is weakly-hard real-time, i.e., it can tolerate a few failures?
Problem: **Reliability analysis of networked control systems**
Problem: Reliability analysis of networked control systems

Given

1. Networked control system (messages, period)
2. Robustness specification (weakly-hard constraints)
3. Active replication scheme (DMR, TMR, others)
4. Peak transient fault rates (for the network and the hosts)
Problem: **Reliability analysis of networked control systems**

<table>
<thead>
<tr>
<th>Given</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>① Networked control system (messages, period)</td>
<td>A safe upper bound on the failure rate of the networked control system</td>
</tr>
<tr>
<td>② Robustness specification (weakly-hard constraints)</td>
<td></td>
</tr>
<tr>
<td>③ Active replication scheme (DMR, TMR, others)</td>
<td></td>
</tr>
<tr>
<td>④ Peak transient fault rates (for the network and the hosts)</td>
<td></td>
</tr>
</tbody>
</table>
Problem: **Reliability analysis of networked control systems**

**Given**

1. Networked control system (messages, period)
2. Robustness specification (weakly-hard constraints)
3. Active replication scheme (DMR, TMR, others)
4. Peak transient fault rates (for the network and the hosts)

**Objective**

A safe upper bound on the failure rate of the networked control system

**Failures-In-Time (FIT)** = Expected number of failures in one billion operating hours
Outline

Analysis of a Controller Area Network (CAN) based networked control system

System Model

Analysis

Evaluation

\[ \int_{0}^{\infty} t \cdot f(t) \, dt \]
Analysis of a Controller Area Network (CAN) based networked control system

System Model

\[ \int_{0}^{\infty} t \cdot f(t) \, dt \]

Analysis

Evaluation
Fault tolerant single-input single-output (FT-SISO) networked control loop
Fault tolerant single-input single-output (FT-SISO) networked control loop
Fault tolerant single-input single-output (FT-SISO) networked control loop

Physical sensor → Controlled plant → Physical actuator

Sensor task replicas: S1, S2, S3

Controller task replicas: C1, C2, C3

Actuator task
Fault tolerant single-input single-output (FT-SISO) networked control loop

The diagram illustrates a networked control loop with the following components:

- **Physical sensor**
- **Controlled plant**
- **Physical actuator**

**Sensor task replicas:**
- S1
- S2
- S3

**Controller task replicas:**
- C1
- C2
- C3

**Actuator task:** A

**CAN bus***

* Controller Area Network
Fault tolerant single-input single-output (FT-SISO) networked control loop

- **Physical sensor**
  - Sensor task replicas: S1, S2, S3
  - Sensor message replicas
  - CAN bus*

- **Controlled plant**

- **Physical actuator**
  - Actuator task: A
  - Controller task replicas: C1, C2, C3

* Controller Area Network
Fault tolerant single-input single-output (FT-SISO) networked control loop

Physical sensor -> Controlled plant -> Physical actuator

Sensor task replicas: S1, S2, S3
Control command replicas: C1, C2, C3
Controller task replicas: A

CAN bus

* Controller Area Network
Fault tolerant single-input single-output (FT-SISO) networked control loop

- Physical sensor
- Controlled plant
- Physical actuator

- Sensor task replicas: S1, S2, S3
- Control command replicas: C1, C2, C3
- Controller task replicas

- CAN bus*

* Controller Area Network
Fault tolerant single-input single-output (FT-SISO) networked control loop

Physical sensor → Controlled plant → Physical actuator

Sensor task replicas: S1, S2, S3
Control command replicas: C1, C2, C3
Controller task replicas: A

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast

* Controller Area Network
Failures and errors in a FT-SISO networked control loop
Failures and errors in a FT-SISO networked control loop

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast

Omission

Can bus

Sensor task replicas

Control command replicas

Controller task replicas

Physical sensor

Controlled plant

Physical actuator

Sensor message replicas

Actuator task

Failures and errors in a FT-SISO networked control loop

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast

Omission

Can bus

Sensor task replicas

Control command replicas

Controller task replicas

Physical sensor

Controlled plant

Physical actuator

Sensor message replicas

Actuator task
Failures and errors in a FT-SISO networked control loop

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast
Failures and errors in a FT-SISO networked control loop

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast

Deadline violation
Failures and errors in a FT-SISO networked control loop

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast
Failures and errors in a FT-SISO networked control loop

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast
Failures and errors in a FT-SISO networked control loop

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast

1. How often does the final actuation deviate from an error-free scenario (iteration failure)?
Failures and errors in a FT-SISO networked control loop

1. How often does the final actuation deviate from an error-free scenario (iteration failure)?

2. What is the likelihood of a complete control failure?

Assumptions:
- Physical plant reliable
- Simple majority voting
- Clock synchronization
- Atomic broadcast

Failures and errors in a FT-SISO networked control loop
1. Modeling control loop iteration failures

Control loop iterations $l_1 \ l_2 \ l_3 \ \cdots \ l_{n-1} \ l_n \ l_{n+1} \ \cdots$
1. Modeling control loop iteration failures

Control loop iterations: $I_1 I_2 I_3 \ldots I_{n-1} I_n I_{n+1} \ldots$

1. Final actuation is successful

Success
1. Modeling control loop iteration failures

Control loop iterations: $I_1, I_2, I_3, \ldots, I_{n-1}, I_n, I_{n+1}, \ldots$

- **1.** Final actuation is successful
- **2.** Final actuation failed (differs from 1)

Flow diagram:
- Success: $I_n \rightarrow \text{Success}$
- Failure: $I_n \rightarrow \text{Failure}$
- Error-free: $I_n \rightarrow \text{Error-free}$
- Erroneous: $I_n \rightarrow \text{Erroneous}$
1. Modeling control loop iteration failures

Control loop iterations

1. Final actuation is successful
2. Final actuation failed (different from 1)
3. Final actuation is successful (same as 1) despite the errors

Success

Failure

Error-free

Erroneous
1. Modeling control loop iteration failures

Control loop iterations: $I_1, I_2, I_3, \ldots, I_{n-1}, I_n, I_{n+1}, \ldots$

1. Final actuation is successful
2. Final actuation failed (different from 1)
3. Final actuation is successful (same as 1 despite the errors)

Explicitly account for fault tolerance
2. Modeling control failure based on the \((m, k)\)-firm constraint
2. Modeling **control failure** based on the \((m, k)\)-firm constraint

Control loop iterations

\[ \text{Success} \quad \text{Failure} \]

\[ S \quad S \quad S \quad F \quad S \quad S \quad S \quad S \quad F \quad S \quad S \quad F \quad S \quad S \quad S \quad S \quad S \quad S \quad S \]
2. Modeling control failure based on the \((m, k)\)-firm constraint

Control loop iterations

<table>
<thead>
<tr>
<th>Success</th>
<th>Failure</th>
</tr>
</thead>
</table>

Hard constraint

Control failure upon first iteration failure
2. Modeling control failure based on the \((m, k)\)-firm constraint

- **Hard constraint**: Control failure upon first iteration failure
- **(2, 3) constraint**: Control failure when less than 2 iterations successful in 3 consecutive iterations
Analysis of a Controller Area Network (CAN) based networked control system

\[ \int_0^\infty t \cdot f(t) \, dt \]
Analysis steps

Peak fault rates

Upper-bound the control failure rate
Analysis steps

1. Peak fault rates
2. Upper-bound message error probabilities
3. Upper-bound the control failure rate
Analysis steps

Peak fault rates

Upper-bound message error probabilities

Upper-bound iteration failure probability

Upper-bound the control failure rate
Analysis steps

1. Peak fault rates
2. Upper-bound message error probabilities
3. Make the upper bound safe for all possible fault rates

Upper-bound iteration failure probability

Upper-bound the control failure rate
Analysis steps

1. Peak fault rates
2. Upper-bound message error probabilities
3. Upper-bound iteration failure probability
4. Make the upper bound safe for all possible fault rates

Upper-bound the control failure rate
Upper-bounding the message error probabilities

Using Poisson model for fault arrivals
Upper-bounding the message error probabilities

Peak fault rates

Using poisson model for fault arrivals

Based on the message parameters

\[ P_1 \geq P \text{ (msg. is omitted at time } t ) \]
\[ P_2 \geq P \text{ (msg. is incorrectly computed) } \]
\[ P_3 \geq P \text{ (msg. is misses its deadline) } \]
Analysis steps

1. Peak fault rates

2. Upper-bound iteration failure probability

3. Make the upper bound safe for all possible fault rates

4. Upper-bound the control failure rate

Message error probability bounds $P_1, P_2, P_3, ...$
Upper-bounding the
iteration failure probabilities

Accounting for
➡ all possible error scenarios
➡ error propagation and correlation
➡ voting protocol

Upper bounds on message error probabilities

\[ P_1 \geq P(\text{msg. is omitted at time } t) \]
\[ P_2 \geq P(\text{msg. is incorrectly computed}) \]
\[ P_3 \geq P(\text{msg. is misses its deadline}) \]
Upper-bounding the iteration failure probabilities

Accounting for:
- all possible error scenarios
- error propagation and correlation
- voting protocol

\[ V_n (P_1, P_2, P_3, \ldots) \geq P (I_n = F) \]

Upper bounds on message error probabilities

\[ P_1 \geq P (\text{msg. is omitted at time } t) \]
\[ P_2 \geq P (\text{msg. is incorrectly computed}) \]
\[ P_3 \geq P (\text{msg. is misses its deadline}) \]
Analysis steps

1. Peak fault rates
2. Message error probability bounds $P_1, P_2, P_3, ...$
3. $V_n (P_1, P_2, P_3, ...) \geq P (I_n = F)$
4. Make the upper bound safe for all possible fault rates

Upper-bound the control failure rate
Is the upper bound $V_n ( P_1, P_2, P_3, ... )$ safe for all possible fault rates?

Let's look at a simple example!
Is the upper bound $V_n (P_1, P_2, P_3, \ldots)$ safe for all possible fault rates?
Is the upper bound $V_n(P_1, P_2, P_3, ...)$ safe for all possible fault rates?

$P_1, P_2, P_3 ...$ defined such that:
- $M_1$ is omitted
- $M_2$ is incorrectly computed
- $M_2$ misses its deadline

Simple majority (ties broken randomly)
Is the upper bound $V_n (P_1, P_2, P_3, ...)$ safe for all possible fault rates?

- **P_1, P_2, P_3 ...** defined such that:
  - $M_1$ is omitted
  - $M_2$ is incorrectly computed
  - $M_2$ misses its deadline

Simple majority (ties broken randomly)

- Omission of $M_1$
- Incorrect computation & deadline violation of $M_2$
- Only $M_3$ participates in the voting process
Is the upper bound $V_n (P_1, P_2, P_3, \ldots)$ safe for all possible fault rates?

$P_1, P_2, P_3 \ldots$ defined such that:
- $M_1$ is omitted
- $M_2$ is incorrectly computed
- $M_2$ misses its deadline

$V_n (P_1, P_2, P_3, \ldots) = 0$

Simple majority (ties broken randomly)

Voter

Message replica $M_1$ (Omission)

Message replica $M_2$ (Incorrect computation & deadline violation)

Message replica $M_3$ (Only $M_3$ participates in the voting process)
Is the upper bound $V_n(P_1, P_2, P_3, ...)$ safe for all possible fault rates?

In practice, there may be no deadline violations!

$V_n(P_1, P_2, P_3, ...) = 0$

- Omission
  - $M_1$ is omitted
  - $M_2$ is incorrectly computed
  - $M_2$ misses its deadline

Simple majority (ties broken randomly)

- Message replica $M_1$
- Message replica $M_2$
- Message replica $M_3$

$P_1, P_2, P_3 ...$ defined such that:

- The peak fault rates are just upper bounds
Is the upper bound $V_n( P_1, P_2, P_3, ... )$ safe for all possible fault rates?

- Omission
- Incorrect computation
- Deadline violation

Message replica $M_1$
Message replica $M_2$
Message replica $M_3$

$P_1, P_2, P_3 ...$ defined such that:
- $M_1$ is omitted
- $M_2$ is incorrectly computed
- $M_2$ misses its deadline

Simple majority (ties broken randomly)

$V_n( P_1, P_2, P_3, ... ) = 0$

In practice, there may be no deadline violations!
- The peak fault rates are just upper bounds
Is the upper bound $V_n ( P_1, P_2, P_3, ... )$ safe for all possible fault rates?

$V_n ( P_1, P_2, P_3, ... ) \geq P ( I_n = F )$

Safe if $V_n$ is monotonic in $P_1, P_2, P_3, ...$
Is the upper bound $V_n(P_1, P_2, P_3, \ldots)$ safe for all possible fault rates?

$$V_n(P_1, P_2, P_3, \ldots) \geq P(I_n = F)$$

+ A fudge factor $\Delta$ is added to ensure monotonicity*

Safe if $V_n$ is monotonic in $P_1, P_2, P_3, \ldots$

---

Is the upper bound \( V_n(P_1, P_2, P_3, \ldots) \) safe for all possible fault rates?

\[
V_n(P_1, P_2, P_3, \ldots) \geq P(I_n = F) + \Delta
\]

A fudge factor \( \Delta \) is added to ensure monotonicity*

\[
U_n(P_1, P_2, P_3, \ldots) \geq P(I_n = F)
\]

Safe if \( V_n \) is monotonic in \( P_1, P_2, P_3, \ldots \)

Analysis steps

1. Peak fault rates
2. Message error probability bounds $P_1, P_2, P_3, ...$
3. $V_n (P_1, P_2, P_3, ... \geq P(I_n = F))$
4. Upper-bound the control failure rate

Safe
Upper-bounding the control failure rate

(Failures-In-Time or FIT)

\[ U_n (P_1, P_2, P_3, \ldots) \geq P (I_n = F) \]
Upper-bounding the control failure rate
(Failures-In-Time or FIT)

\[ U_n(P_1, P_2, P_3, \ldots) \geq P(I_n = F) \]

\[ \text{FIT} = \frac{10^9}{MTTF} \text{ (in hours)} \]
\[ = \frac{10^9}{\int_0^\infty t \cdot f(t) \, dt} \text{ (Mean Time To first control Failure)} \]
\[ = \frac{10^9}{\int_0^\infty f(t) \, dt} \text{ (probability density function)} \]
Upper-bounding the control failure rate
(Failures-In-Time or FIT)

\[ U_n(P_1, P_2, P_3, \ldots) \geq P(I_n = F) \]

FIT (expected # failures in 1 billion hours)
\[ \text{FIT} = \frac{10^9}{\text{MTTF}} \text{ (in hours)} \]
\[ \text{FIT} = \frac{10^9}{\int_0^\infty t \cdot f(t) \, dt} \text{ (probability density function)} \]

\[ f(t) = P(\text{first control failure at time } t) \]
\[ = P(\text{first violation of (2, 3)-firm constraint at time } t) \]
\[ = P(\text{first instance of FSF I FFS I SFF I FF at time } t) \]
Upper-bounding the control failure rate (Failures-In-Time or FIT)

\[ U_n \left( P_1, P_2, P_3, \ldots \right) \geq P \left( I_n = F \right) \]

Using prior work*

\[ f(t) = P \left( \text{first control failure at time } t \right) \]
\[ = P \left( \text{first violation of (2, 3)-firm constraint at time } t \right) \]
\[ = P \left( \text{first instance of FSF I FFS I SFF I FF at time } t \right) \]

\[ \text{FIT} = \frac{10^9}{\text{MTTF}} \text{ (in hours)} \]
\[ = \frac{10^9}{\int_0^\infty t \cdot f(t) \, dt} \text{ (probability density function)} \]

Upper-bounding the control failure rate
(Failures-In-Time or FIT)

\[ U_n (P_1, P_2, P_3, \ldots) \geq P (I_n = F) \]

Scalable and numerical, but sound, analysis

\[ f(t) = P \text{ (first control failure at time } t \text{) }\]
\[ = P \text{ (first violation of (2, 3)-firm constraint at time } t \text{) }\]
\[ = P \text{ (first instance of } \text{FSF} | \text{FFS} | \text{SFF} | \text{FF at time } t \text{) }\]

\[ \text{FIT = } \frac{10^9}{\text{MTTF}} \text{ (in hours)} \]
\[ = \frac{10^9}{\int_0^\infty t \cdot f(t) \, dt} \]

(probability density function)

\[ \text{(expected # failures in 1 billion hours)} \]

\[ \text{MTTF (in hours)} = \frac{10^9}{\int_0^\infty f(t) \, dt} \]

Analysis steps

Peak fault rates

Message error probability bounds $P_1, P_2, P_3, ...$

$V_n (P_1, P_2, P_3, ...)$ \geq P (I_n = F)

$U_n (P_1, P_2, P_3, ...)$ \geq P (I_n = F)

FIT$_{UB}$ for a single control loop

Safe
Analysis steps

Peak fault rates

\( \text{FIT}_{UB} \) for a single control loop

\( \text{FIT}_{UB} \) for \( L_1 \)
\( \text{FIT}_{UB} \) for \( L_2 \)
\( \cdots \)
\( \text{FIT}_{UB} \) for \( L_n \)

Compute \( \text{FIT}_{UB} \) bounds for all control loops in the networked control system
Analysis steps

Peak fault rates

Upper bound on the FIT rate of the entire networked control system

\[ \sum \text{FIT}_{UB} \text{ for } L_1 \text{, } L_2 \text{, } ... \text{, } L_n \]

Compute FIT bounds for all control loops in the networked control system

FIT_{UB} for a single control loop
Analysis of a Controller Area Network (CAN) based networked control system

\[ \int_{0}^{\infty} t \cdot f(t) \, dt \]
Evaluation overview
Evaluation overview

- How accurate is the analysis?
  - Comparison with simulation results
Evaluation overview

☐ How accurate is the analysis?
  ➡ Comparison with simulation results

☐ Case study: FIT vs. (m, k) constraints vs. replication schemes
**CAN-based active suspension workload**

- Four control loops $L_1$, $L_2$, $L_3$, $L_4$ to control the four wheels with magnetic suspension

<table>
<thead>
<tr>
<th>Messages</th>
<th>Length</th>
<th>Period (ms)</th>
<th>Deadline (ms)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock sync.</td>
<td>1</td>
<td>50</td>
<td>50</td>
<td>High</td>
</tr>
<tr>
<td>Current mon.</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>$L_1$ messages</td>
<td>3</td>
<td>1.75</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>$L_2$ messages</td>
<td>3</td>
<td>1.75</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>$L_3$ messages</td>
<td>3</td>
<td>1.75</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>$L_4$ messages</td>
<td>3</td>
<td>1.75</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>Logging</td>
<td>8</td>
<td>100</td>
<td>100</td>
<td>Low</td>
</tr>
</tbody>
</table>

## CAN-based active suspension workload

Four control loops L₁, L₂, L₃, L₄ to control the four wheels with magnetic suspension.

<table>
<thead>
<tr>
<th>Messages</th>
<th>Length</th>
<th>Period (ms)</th>
<th>Deadline (ms)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock sync.</td>
<td>1</td>
<td>50</td>
<td>50</td>
<td>High</td>
</tr>
<tr>
<td>Current mon.</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>L₁ messages</td>
<td>3</td>
<td>1,75</td>
<td>1,75</td>
<td></td>
</tr>
<tr>
<td>L₂ messages</td>
<td>3</td>
<td>1,75</td>
<td>1,75</td>
<td></td>
</tr>
<tr>
<td>L₃ messages</td>
<td>3</td>
<td>1,75</td>
<td>1,75</td>
<td></td>
</tr>
<tr>
<td>L₄ messages</td>
<td>3</td>
<td>1,75</td>
<td>1,75</td>
<td></td>
</tr>
<tr>
<td>Logging</td>
<td>8</td>
<td>100</td>
<td>100</td>
<td>Low</td>
</tr>
</tbody>
</table>

This talk: Control loop L₁'s tasks were replicated.

---

## CAN-based active suspension workload*

- **Four control loops** $L_1$, $L_2$, $L_3$, $L_4$ ➡ to control the four wheels with magnetic suspension

<table>
<thead>
<tr>
<th>Messages</th>
<th>Length</th>
<th>Period (ms)</th>
<th>Deadline (ms)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock sync.</td>
<td>1</td>
<td>50</td>
<td>50</td>
<td>High</td>
</tr>
<tr>
<td>Current mon.</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>$L_1$ messages</td>
<td>3</td>
<td>1.75</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>$L_2$ messages</td>
<td>3</td>
<td>1.75</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>$L_3$ messages</td>
<td>3</td>
<td>1.75</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>$L_4$ messages</td>
<td>3</td>
<td>1.75</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>Logging</td>
<td>8</td>
<td>100</td>
<td>100</td>
<td>Low</td>
</tr>
</tbody>
</table>


This talk: Control loop $L_1$'s tasks were replicated

In the paper: Experiments with all replica schemes
How accurate is the analysis?

Iteration failure probability bound

\[ U_n (P_1, P_2, P_3, \ldots) \geq P(\text{I}_n = \text{F}) \]

Discrete event simulation of a CAN-based system

- Poisson process for CAN bus faults
- Poisson process for faults on Host 1
  ... and so on
How **accurate** is the analysis?

Iteration failure probability bound

\[ U_n (P_1, P_2, P_3, ... ) \geq P (I_n = F) \]

Discrete event simulation of a CAN-based system

Poisson process for CAN bus faults

Poisson process for faults on Host 1

... and so on

Simulation is not safe
Analysis versus simulation

Lower implies better reliability
Analysis versus simulation

![Graph showing the comparison between Analysis and Simulation](image)
Analysis versus simulation

Failure probability decreases with increasing replication
Analysis versus simulation

Pessimism incurred stays within an order of magnitude for up to four replicas.
Analysis versus simulation

At high network utilization, worst-case response-time analysis affects the analysis accuracy.
Case study
Case study

FIT analysis for different \((m, k)\)-firm constraints

- \((9, 100) \sim 9\%\)
- \((19, 20) \sim 95\%\)
- \((99, 100) \sim 99\%\)
- \((9999, 10000) \sim 99.99\%\)
Case study

- FIT analysis for different (m, k)-firm constraints
  - (9, 100) ~ 9%
  - (19, 20) ~ 95%
  - (99, 100) ~ 99%
  - (9999, 10000) ~ 99.99%

- Replication factor of loop L_1's tasks varied from 1 to 5
Case study

- FIT analysis for different (m, k)-firm constraints
  - (9, 100) ~ 9%
  - (19, 20) ~ 95%
  - (99, 100) ~ 99%
  - (9999, 10000) ~ 99.99%

- Replication factor of loop L₁'s tasks varied from 1 to 5

- What should be the replication factor to achieve FIT under $10^{-6}$?
FIT rate vs. replication factor vs. $(m, k)$ parameters

Lower implies better reliability
FIT rate vs. replication factor vs. \((m, k)\) parameters

![Graph showing FIT rate vs. replication factor vs. \((m, k)\) parameters]
FIT rate vs. replication factor vs. \((m, k)\) parameters

FIT rate decreases with increasing replication
FIT rate vs. replication factor vs. \((m, k)\) parameters

If the desired FIT rate is under 10\(^{-6}\), prefer three replicas.
Summary

A safe Failures-In-Time (FIT) analysis for networked control systems

\[\rightarrow\] CAN-based networked control system model
Summary

- A safe Failures-In-Time (FIT) analysis for networked control systems
  - CAN-based networked control system model

- Focus on failures and errors due to transient faults
  - omission errors
  - incorrect computation errors
  - transmission errors
Summary

A safe Failures-In-Time (FIT) analysis for networked control systems
- CAN-based networked control system model

Focus on failures and errors due to transient faults
- omission errors
- incorrect computation errors
- transmission errors

... and on robust systems that can tolerate a few iteration failures
- (m,k)-firm model for control failure
A safe Failures-In-Time (FIT) analysis for networked control systems

- CAN-based networked control system model

Focus on failures and errors due to transient faults

- omission errors
- incorrect computation errors
- transmission errors

Future work: Byzantine errors + BFT protocols

... and on robust systems that can tolerate a few iteration failures

- (m,k)-firm model for control failure
A safe Failures-In-Time (FIT) analysis for networked control systems
- CAN-based networked control system model

Focus on failures and errors due to transient faults
- omission errors
- incorrect computation errors
- transmission errors

Future work: Byzantine errors + BFT protocols

... and on robust systems that can tolerate a few iteration failures
- (m,k)-firm model for control failure

Accounting for other robustness criteria