Analysis and Optimization of Message Acceptance Filter Configurations for Controller Area Network (CAN)

Florian Pölzlbauer *
Robert I. Davis
Iain Bate
Outline

- Motivation: Automotive System Design
- Controller Area Network (CAN) Protocol

- CAN Message Acceptance Filtering – how it works
- Measuring Filtering Quality
- Optimal Filter Configuration

- Evaluation
System & Responsibilities
- many ECUs with dedicated functionality; developed by **Tier1-supplier**; (SW+HW)
- data-exchange between ECUs via networks (LIN, **CAN**, FlexRay, Ethernet)
- network design & system integration by OEM

Challenges
- cost pressure of high-volume ECUs
- constrained HW-resources (CPU clock, memory)
- efficient resource usage needed
Controller Area Network (CAN)

- asynchronous, multi-master, broadcast, serial communications bus
- each message uniquely identified by its ID, which also determines priority during arbitration phase

<table>
<thead>
<tr>
<th>ID</th>
<th>payload</th>
</tr>
</thead>
</table>

- once idle, priority-based bus arbitration: highest priority (i.e. lowest ID) wins
- non-preemptive transmission of message

- schedulability analysis (response time analysis)

Message specification:
- s: payload size
- T: period
- D: deadline
- ID-format (11 or 29 bit)

```plaintext
\[
R_m = J_m + w_m + C_m \\
R_m \leq D_m \\
C_{m}^{11} = (55 + 10 \cdot s_m) \cdot \tau_{bit} \\
C_{m}^{29} = (80 + 10 \cdot s_m) \cdot \tau_{bit} \\
w_{m}^{n+1} = B + \sum_{k \in chp(m)} \left[ \frac{w_m^n + J_k^{n} + \tau_{bit}}{T_k} \right] \cdot C_k \\
B = \max \{C_m\}
\]```

ID: 11 or 29 bits
payload: 0…8 bytes (CAN)
0…64 bytes (CAN-FD)
Receiving CAN Messages

- each node receives every message (i.e. broadcast bus)
- node looks at message-ID to determine if message is relevant
- if yes: process message
- if not: discard \(\rightarrow\) undesired message receive interrupts

- goal: minimize undesired message receive interrupt load
- solution: HW-based **acceptance filtering** (widely available)

- question: find an optimal filter-configuration
  (undesired RX interrupts \(\rightarrow\) min.)
How to measure filtering quality of given filter configuration?
- assessment method & quality metric

How to design an optimal filter configuration?
- design optimization methodology
### Message Acceptance Filter: how it works

**2 registers (per receive buffer):**
- **mask:** which digits are checked
- **tag:** which digit-values are accepted

- **mask:** 111 1111 0011
- **tag:** 000 1100 0000

[Diagram of message acceptance filter]

**ID:**
- 000 0000 0100 (4) **blocked**
- 000 1100 1000 (200) **pass**

- **if message passes HW-filter, it is put into receive buffer, and raises receive interrupt**
- **if message is blocked, nothing happens**
Message Acceptance Filter: how it works

- we use “abstract” notation (0,1,x)
  x … “don’t care”

- filters implemented in HW-logic
  inside CAN-controller → no CPU load

- Most CAN-controllers: mask + tag per buffer
- Some CAN-controllers: “shared mask”
  - tag per buffer
  - mask applied to N buffers

```
mask : 111 1111 0011

tag  : 000 1100 0000
----------------------
filter : 000 1100 xx00
```

e.g.: National’s CR16
15 buffers
1 mask: buffer[0]
1 mask: buffer[1..14]
How to measure Filtering Quality of given Filter Configuration
Given

- $M^{all}$: set of all broadcasted messages (ID, period)
- $M^{des}$: subset of messages which shall be received by node
- $F$: filter configuration (f filter-patterns)

Calc.

\[
M^{pass} = M^{all} \setminus M^{block} \\
M^{block} = M^{all} \setminus M^{pass} \\
M^{UP} = M^{pass} \setminus M^{des} \\
M^{UB} = M^{block} \cap M^{des}
\]

UB … unintended block
UP … unintended pass
Classification

- in-feasible \( M^{UB} \neq \{\} \) … some desired messages are blocked
- feasible \( M^{UB} = \{\} \) … all desired messages pass
  - perfect \( M^{UP} = \{\} \) … only desired messages pass

Quality

\[
load^{UP} = \frac{M^{UP}}{\text{sec.}} = \sum_{M^{UP}} \frac{1}{T_m} \rightarrow \min
\]
How to Design an Optimal Filter Configuration
Filter Design Problem

**Given**

- $M^{all}$ set of all broadcasted messages (ID, period)
- $M^{des}$ subset of messages which shall be received by node
- $f$ number of available filters

**Find**

- filter-pattern for each of the $f$ filters
- $load^{up} \rightarrow \text{min.}$

**3 Cases**

- $f \geq |M^{des}|$
- $f = 1$
- $1 < f < |M^{des}|$

Complexity:

- problem is NP-complete
- transformation to SET COVER problem
- proof in paper
Optimal Filter: $f \geq |M^{des}|$

- straightforward
- each filter = one message ID
- perfect filtering

Algorithm 2: Optimal Solution for $f \geq |M^{des}|$ Filters

Input: $M^{des}$ /* desired messages */

1  foreach $m_i \in M^{des}$ do
2       $f_i = \text{ID}(m_i)$
3  end

Output: $F$ /* perfect filtering */
Optimal Filter: $f = 1$

- most constrained case
- for each filter-digit, derive filter-value from desired message’s ID-digit
- ensures feasible filter

```
ID = 00011000000 (192)
ID = 00011000100 (196)
ID = 00011001000 (200)
ID = 00011001100 (204)
```

```
filter = 0001100xx00
```

---

**Algorithm 1: Optimal Solution for $f = 1$ Filter (always feasible)**

```
Input: $M^{des}$ /* desired messages */
1 foreach ID-digit do /* desired messages */
2     if $\forall m_i \in M^{des}$ the ID-digit is 0 then
3         filter-digit = 0
4     else if $\forall m_i \in M^{des}$ the ID-digit is 1 then
5         filter-digit = 1
6     else
7         filter-digit = x
8     end
9 end
Output: $F$ /* feasible filtering */
- simulated annealing (meta-heuristic search)

- cost function:
  \[
  \cos t_1 = \frac{M^{UP}}{M^{des}} \quad \text{... avoid infeasible filtering}
  \]

  \[
  \cos t_2 = \frac{M^{UP}}{(M^{all} \setminus M^{des})} \text{sec.} \quad \text{... avoid “unintended pass” load}
  \]

- problem encoding:
  - group desired messages into f groups
  - for each group: derive optimal filter (using algo.1)

- neighbour move:
  - move desired message m into another group, and re-calc filters

Algorithm 3: Simulated Annealing

```
Input: T / initial temperature */
Input: s_{cur} / initial solution */
1. c_{cur} = cost(s_{cur}) /* initial cost */
2. repeat
3. iterAtT = 0
4. repeat
5. iter++
6. iterAtT++
7. /* generate new solution */
8. s_{new} = neighbour(s_{cur})
9. c_{new} = cost(s_{new})
10. /* accept move? */
11. if c_{new} < c_{cur} then /* cost is improved */
12. s_{cur} = s_{new}
13. else /* cost is not improved */
14. if e^{\frac{c_{new} - c_{cur}}{T}} > random(0, 1) then
15. s_{cur} = s_{new}
16. end
17. end
18. /* remember best solution */
19. if s_{best} < s_{new} then
20. s_{best} = s_{new}
21. end
22. if c_{new} < c_{best} then
23. c_{best} = c_{new}
24. end
25. until iterAtT == iterAtTmax;
26. T = T * coolingFactor
27. until iter == iterMax;
```

Output: s_{best} /* best solution found */
Optimal Filter: \(1 < f < |M^{\text{des}}|\)

- heuristic
  - sort desired messages by ID
  - assign \(|M^{\text{des}}|/f\) desired messages to each filter
  - derive filter-pattern for each filter (algo.1)

- initial solution for SA
- as reference for evaluation
  (no filter design algorithm in literature)

Idea:
If IDs per filter are similar, then algo.1 should find effective filter-pattern
i.e. few x in filter-pattern
few unintended pass
Evaluation
(synthetic & real-world)
Large scale evaluation

- synthetic message-sets
  - 11-bit message ID (0 … 2047, uniform)
  - 10 … 1000 ms message period (log-uniform)
  - 25 … 100 broadcasted messages
  - 5 … 40 desired messages
  - 1 … 16 available filters
  - only schedulable message-sets considered

- f=1  30 scenarios * 100 example-instances (algo.1)
- 1<f<|M_{des}|  68 scenarios * 100 example-instances (SA, heuristic)
Evaluation: $f = 1$

- Algo.1
- Fig.2 in paper

- $|M^{\text{des}}|$ increases … filter-quality drops
- “good” filtering up to 3..4 des. messages
- $|M^{\text{all}}|$ has almost no effect
evaluation: $1 < f < |M^{des}|$

- Fig. 7 in paper

- color: nr. of filters
  - solid=SA, dashed=heuristic
  - $|M^{all}|=100$

- $|M^{des}|$ increases … filter-quality drops
- $|f|$ increased … better filter-quality
- SA always better than heuristic
Evaluation: $1 < f < |M^{\text{des}}|$ 

- Fig.3 in paper

- color: nr. of broadcast messages $|M^{\text{all}}|$ 
  solid=SA, dashed=heuristic 
  $|M^{\text{des}}|=20$

- $|f|$ increases ... filter-quality increases 
- $|M^{\text{all}}|$ increased ... filter-quality lower 
  but: only minor effect 
- SA always better than heuristic
Real-World Application: HVAC-controller of lightweight battery-electric car

- developed in EU-funded project [www.epsilon-project.eu](http://www.epsilon-project.eu)
- our tasks: design electric powertrain, implement HVAC-controller, integrate into CAN-network

- CAN network: 55 messages (1407 messages/sec.)
- HVAC controller:
  - receives 11 messages (227 messages/sec)
  - sends 4 messages
  - AT90CAN128 micro-controller (ATMEL 16MHz)
  - 15 CAN message-buffers
    - perfect filtering is possible (with 11 Rx-filters)

- can we optimize CAN filtering with fewer Rx-filters?

<table>
<thead>
<tr>
<th>ID [hex]</th>
<th>T [ms]</th>
<th>desired</th>
<th>ID [hex]</th>
<th>T [ms]</th>
<th>desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x6A</td>
<td>1000</td>
<td>yes</td>
<td>0x3F2</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>100</td>
<td></td>
<td>0x3E0</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x1A</td>
<td>100</td>
<td></td>
<td>0x3A6</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x0E</td>
<td>100</td>
<td></td>
<td>0x610</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x071</td>
<td>100</td>
<td></td>
<td>0x411</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>100</td>
<td></td>
<td>0x2A0</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x065</td>
<td>100</td>
<td></td>
<td>0x425</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x6C</td>
<td>100</td>
<td></td>
<td>0x610</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x41</td>
<td>100</td>
<td></td>
<td>0x511</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x96</td>
<td>100</td>
<td></td>
<td>0x512</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>10</td>
<td></td>
<td>0x513</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x10</td>
<td>10</td>
<td></td>
<td>0x514</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x18</td>
<td>10</td>
<td></td>
<td>0x518</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x1A</td>
<td>10</td>
<td></td>
<td>0x520</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x20</td>
<td>10</td>
<td></td>
<td>0x5F0</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x700</td>
<td>100</td>
<td></td>
<td>0x702</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x510</td>
<td>100</td>
<td></td>
<td>0x710</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x200</td>
<td>100</td>
<td></td>
<td>0x711</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x720</td>
<td>100</td>
<td></td>
<td>0x730</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x770</td>
<td>100</td>
<td></td>
<td>0x770</td>
<td>100</td>
<td>yes</td>
</tr>
<tr>
<td>0x771</td>
<td>100</td>
<td></td>
<td>0x771</td>
<td>100</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 2: Network Specification of Battery Electric Vehicle (BEV) available for HVAC-Controller
Real-World Application: Results

- find best filter-configuration for different number of filters
- solve each scenario 100 times (for “perfect ratio”)

<table>
<thead>
<tr>
<th>Filters</th>
<th>Desired [msg/sec.]</th>
<th>Unintended [msg/sec.]</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>227 (11)</td>
<td>1180 (44)</td>
</tr>
<tr>
<td>3</td>
<td>227 (11)</td>
<td>48 (13)</td>
</tr>
<tr>
<td>7</td>
<td>227 (11)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>9</td>
<td>227 (11)</td>
<td>0 (0)</td>
</tr>
</tbody>
</table>

- Rx-interrupt handler: 13.75...38.13us (avg. 31.25us)
  - 227 desired/sec: 7.1ms/sec (0.71%)
  - 48 unintended/sec: 1.5ms/sec (0.15%)
  - 1180 unintended/sec: 36.9ms/sec (3.69%)
Filter Details (3 vs. 7 filters)

filter = 000xxx0x0xx
    ID = 0000001010 (0x00a) UP
    ID = 0000100000 (0x020) UP
    ID = 0001100000 (0x060) UP
    ID = 0001101010 (0x06a)
    ID = 0010000001 (0x081)
----------------------------

filter = 0x000xx0000
    ID = 0000010000 (0x020) UP
    ID = 0000100000 (0x020) UP
    ID = 0100000000 (0x200) UP
    ID = 0100001000 (0x210)
    ID = 0100010000 (0x220)
----------------------------

filter = 11xxxxx00xx
    ID = 1100001000 (0x610) UP
    ID = 1100001001 (0x611) UP
    ID = 1100001010 (0x612) UP
    ID = 1100001011 (0x613)
    ID = 1100010000 (0x620)
    ID = 1101111000 (0x6f0)
    ID = 1110000000 (0x700) UP
    ID = 1110000010 (0x702) UP
    ID = 1110001000 (0x710)
    ID = 1110001001 (0x711) UP
    ID = 1110010000 (0x720) UP
    ID = 1110011000 (0x730)
    ID = 1110111000 (0x770)
    ID = 1110111100 (0x771) UP
----------------------------

filter = 00001101010
    ID = 00001101010 (0x06a)
----------------------------

filter = 0001000001
    ID = 0001000001 (0x081)
----------------------------

filter = 11000010011
    ID = 11000010011 (0x613)
----------------------------

filter = 0x000xx0000
    ID = 0000001000 (0x010)
    ID = 0000010000 (0x020) UP
    ID = 0100000000 (0x200) UP
    ID = 0100001000 (0x210)
    ID = 0100010000 (0x220)
----------------------------

filter = 11100x1000
    ID = 1100010000 (0x620)
    ID = 1110011000 (0x730)
----------------------------

filter = 11xx111000
    ID = 1101111000 (0x6f0)
    ID = 1110111000 (0x770)
----------------------------

filter = x100010000
    ID = 0100010000 (0x220)
    ID = 1100010000 (0x620)
Conclusion & Outlook

Conclusion
- introduced engineering-problem: CAN message acceptance filter
- method for assessing of filtering quality
- method for designing near-optimal filter-configurations
- evaluation shows effectiveness (min. undesired Rx-interrupt load)

Future
- optimize filter quality for “now” and “future extensions” (extensibility)
  - try to block “not yet defined” broadcasted messages
- combine “message ID assignment” and “message filter design”
  - assign message IDs such that messages are schedulable, and efficient acceptance-filters can be designed
- bring methods to industry (engineering tools)
Questions & Discussion
Analysis and Optimization of Message Acceptance Filter Configurations for Controller Area Network (CAN)

25th International Conference on Real-Time Networks and Systems
Grenoble, France, 4-6th October 2017

© 2017 Copyright held by the authors.
Rights to publish the paper licensed to ACM.
ISBN 978-1-4503-5286-4/17/10
DOI: https://doi.org/10.1145/3139258.3139266

Dr. Florian Pölzlbauer
VIRTUAL VEHICLE Research Center
florian.poelzlbauer@v2c2.at