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

Florian Pölzlbauer *
Robert I. Davis
Iain Bate

VIRTUAL VEHICLE Research Center is funded within the COMET – Competence Centers for Excellent Technologies – programme by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Federal Ministry of Science, Research and Economy (BMWFW), the Austrian Research Promotion Agency (FFG), the province of Styria and the Styrian Business Promotion Agency (SFG). The COMET programme is administrated by FFG.

Outline

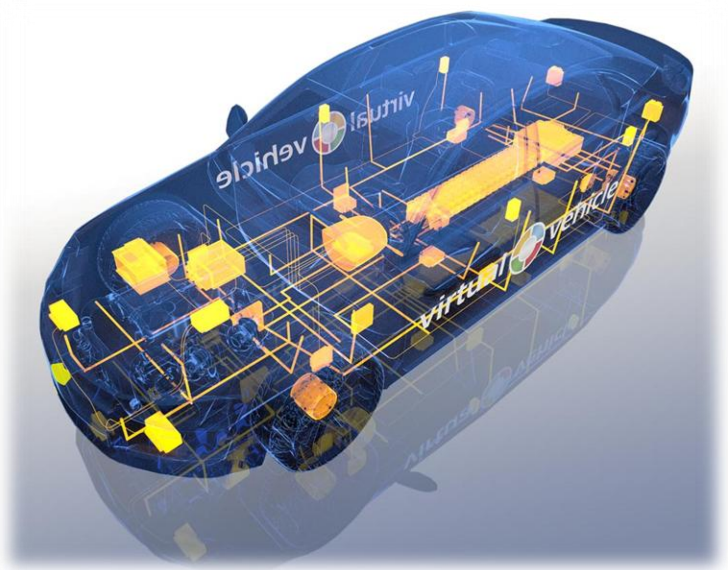
- Motivation: Automotive System Design
- Controller Area Network (CAN) Protocol

- 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

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

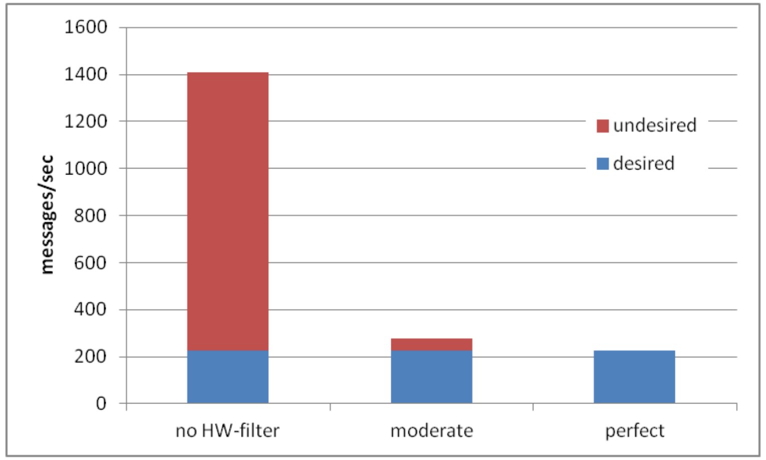
- once idle, priority-based bus arbitration: highest priority (i.e. lowest ID) wins
- non-preemptive transmission of message
- schedulability analysis (response time analysis)

$$R_m = J_m + w_m + C_m$$
$$R_m \le D_m$$
$$C_m^{11} = (55 + 10 \cdot s_m) \cdot \tau_{bit}$$
$$C_m^{29} = (80 + 10 \cdot s_m) \cdot \tau_{bit}$$
$$w_m^{n+1} = B + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil \cdot C_k$$
$$B = \max\{C_m\}$$

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

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 → 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 → min.)

How to measure filtering quality of given filter configuration?

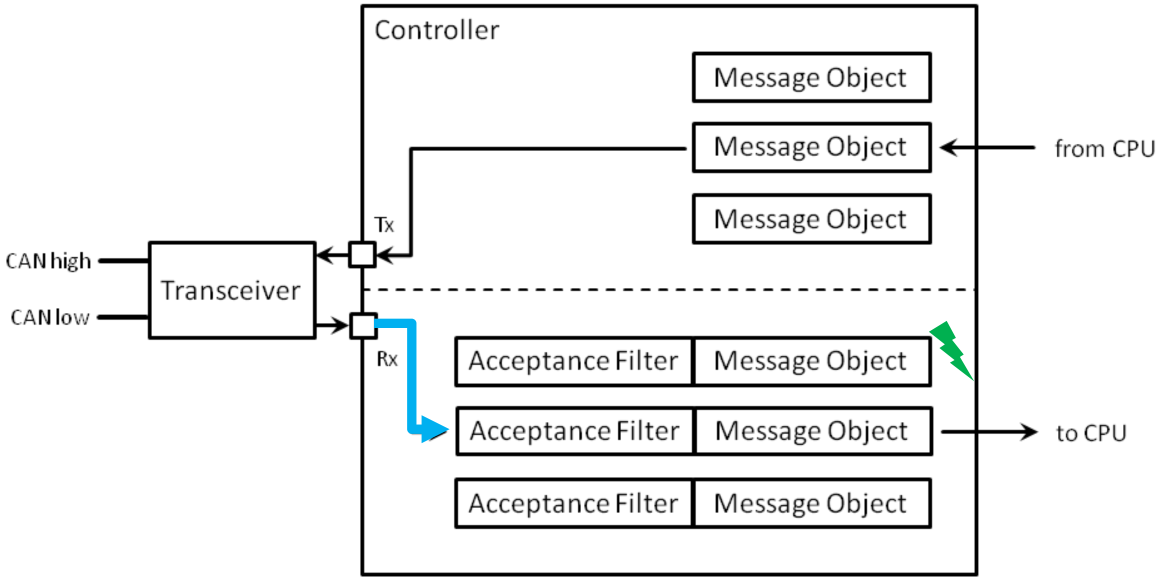
- assessment method & quality metric

How to design an optimal filter configuration?

- design optimization methodology

2 registers (per receive buffer):

- mask: which digits are checked
- tag: which digit-values are accepted




```
mask : 111 1111 0011
tag  : 000 1100 0000
--------------------
ID   : 000 0000 0100 (  4) blocked

ID   : 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"

```
mask   : 111 1111 0011
tag    : 000 1100 0000
----------------------
filter : 000 1100 xx00
```

- 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

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.

$$M^{pass}$$

$$M^{block} = M^{all} \setminus M^{pass}$$

$$M^{UB} = M^{block} \cap M^{des}$$

UB … unintended block

$$M^{UP} = M^{pass} \setminus M^{des}$$

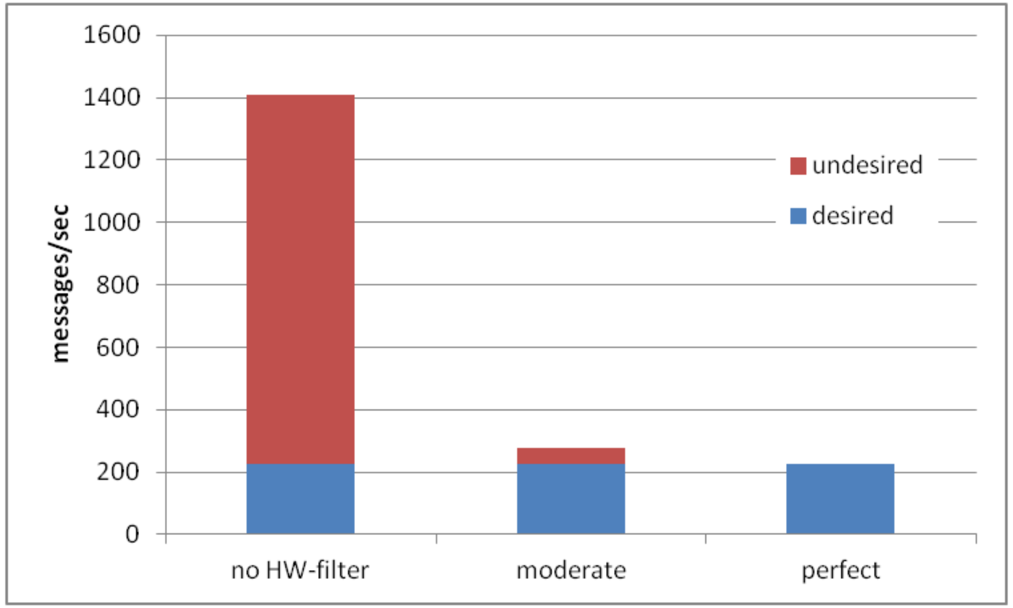
UP … unintended pass

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

$$load^{.UP} = \frac{M^{UP}}{sec.} = \sum_{M^{UP}} \frac{1}{T_m} \rightarrow \min$$



How to Design an Optimal Filter Configuration

virtual vehicle
THE UNIVERSITY of York
Inria INVENTORS FOR THE DIGITAL WORLD

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
- $load^{UP}$ → min.

Complexity:
- problem is NP-complete
- transformation to SET COVER problem
- proof in paper

3 Cases

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

Optimal Filter: f ≥ $|M^{des}|$

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

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

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

foreach $m_i \in M^{des}$ **do**

  $f_i = \mathrm{ID}(m_i)$

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 */

foreach *ID-digit* **do**
  if $\forall m_i \in M^{des}$ *the ID-digit is 0* **then**
    filter-digit = 0
  else if $\forall m_i \in M^{des}$ *the ID-digit is 1* **then**
    filter-digit = 1
  else
    filter-digit = x
  end
end

Output: F /* feasible filtering */

Optimal Filter: $1 < f < |M^{des}|$

- simulated annealing (meta-heuristic search)

- cost function:

$$\cos t_1 = \frac{M^{UB}}{M^{des}}$$ … avoid infeasible filtering

$$\cos t_2 = \frac{M^{UP} / \text{sec.}}{(M^{all} \setminus M^{des}) / \text{sec.}}$$ … 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 */
c_cur = cost(s_cur)                        /* initial cost */
repeat
    iterAtT = 0
    repeat
        iter++
        iterAtT++
        /* generate new solution */
        s_new = neighbour(s_cur)
        c_new = cost(s_new)
        /* accept move? */
        if c_new < c_best then
            /* cost is improved */
            s_cur = s_new
        else
            /* cost is not improved */
            if e^((c_cur - c_new)/t) > random(0, 1) then
                s_cur = s_new
            end
        end
        /* remember best solution */
        if c_new < c_best then
            c_best = c_new
            s_best = s_new
        end
    until iterAtT == iterAtTmax;
    t = t * coolingFactor
until iter == iterMax;
Output: s_best                             /* best solution found */
```

Optimal Filter: $1 < f < |M^{des}|$

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

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

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

Evaluation
(synthetic & real-world)

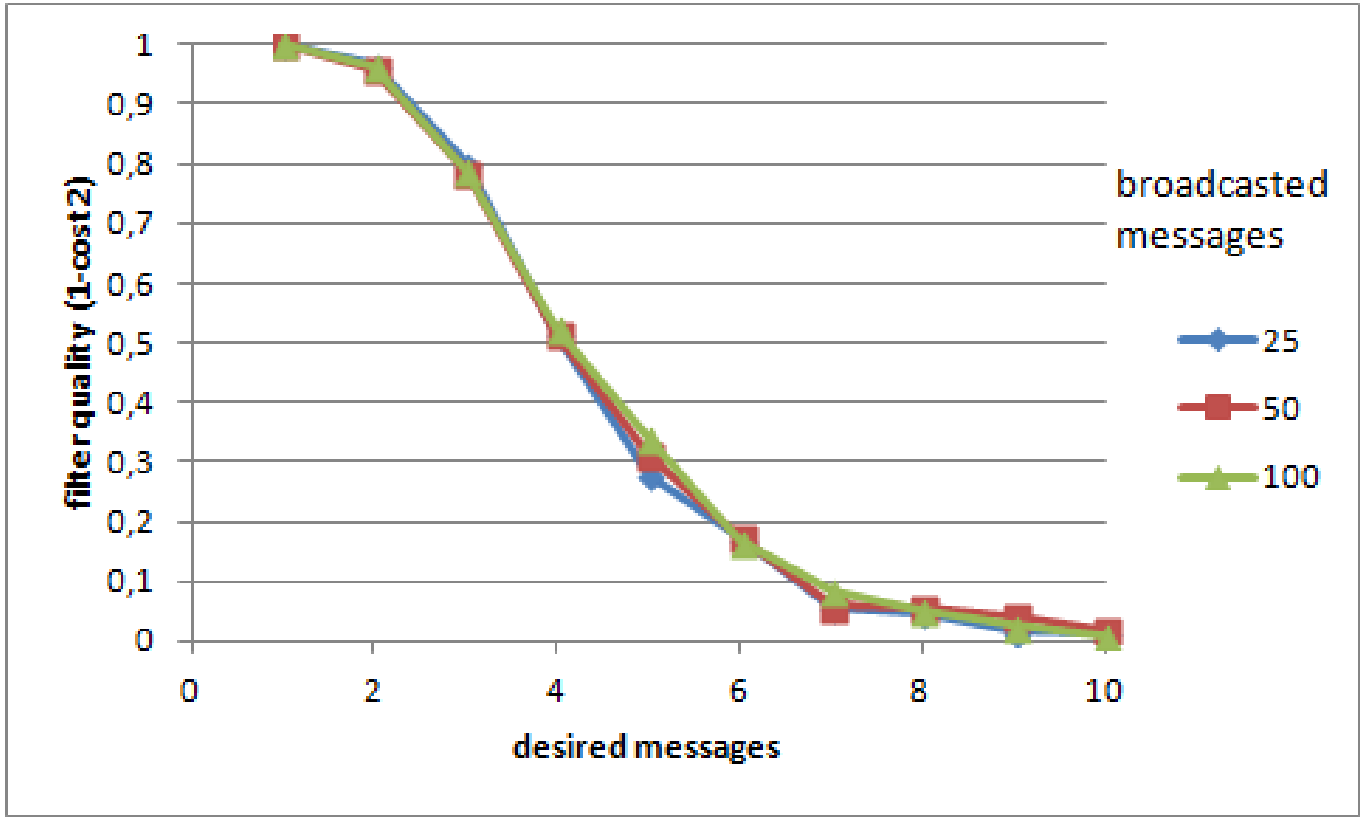
Large scale evaluation

- 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



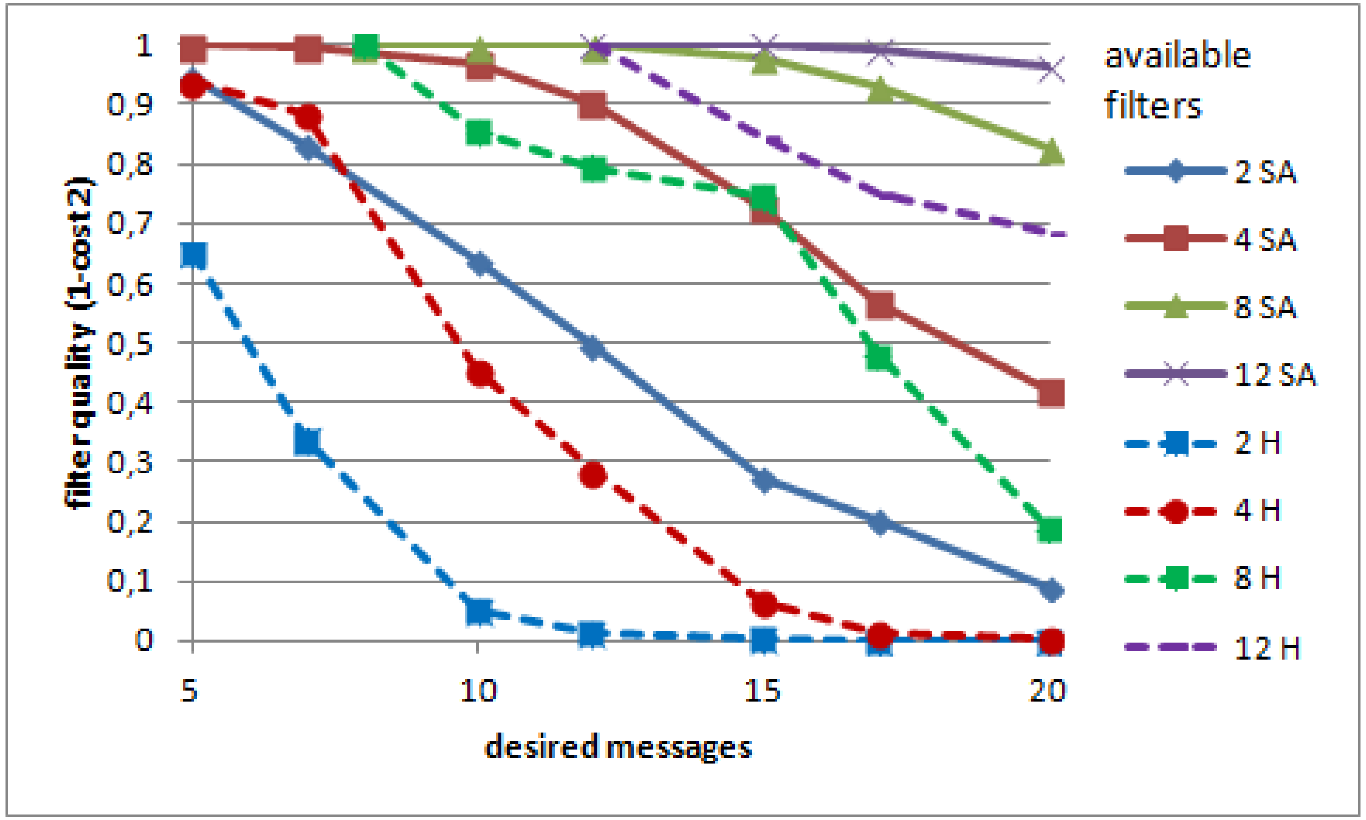
- $|M^{des}|$ increases … filter-quality drops
- “good” filtering up to 3..4 des. messages
- $|M^{all}|$ has almost no effect

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

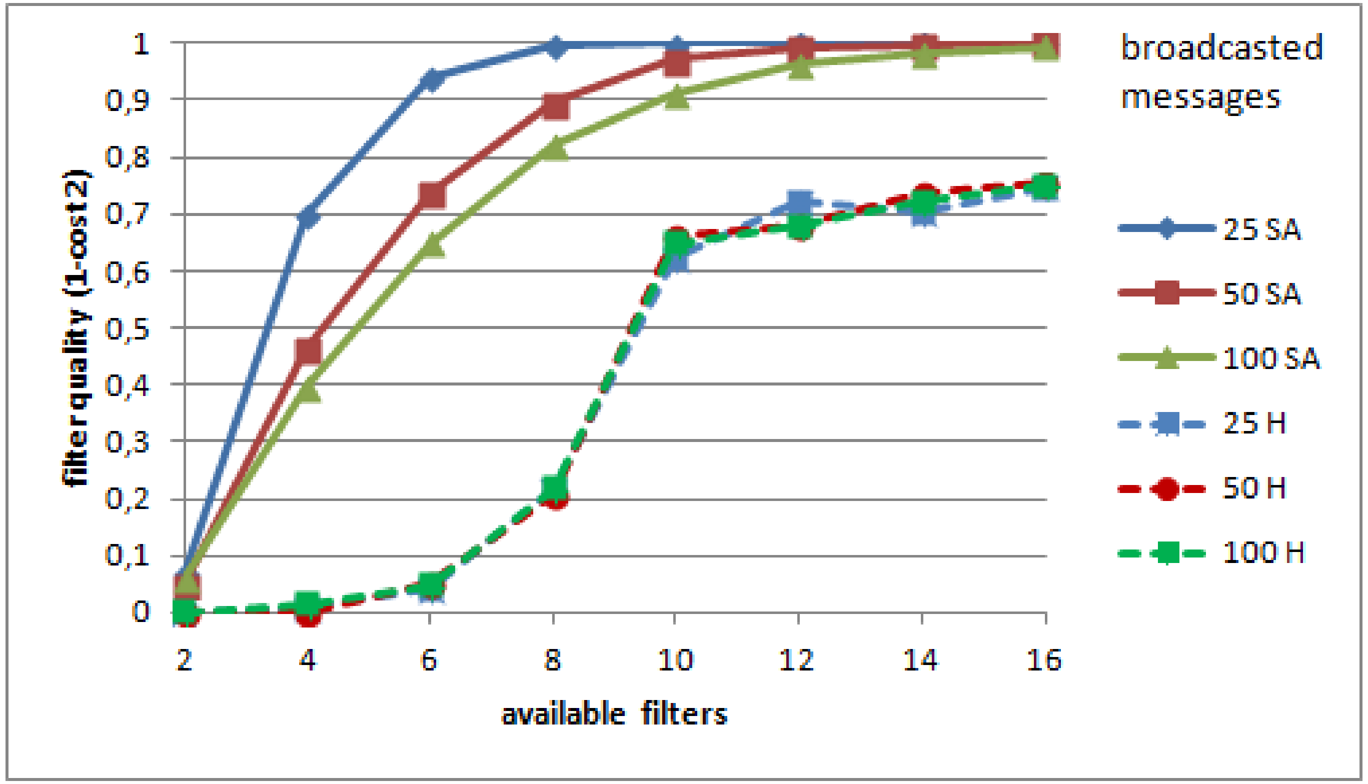


Evaluation: $1 < f < |M^{des}|$

- Fig.3 in paper

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

- |f| increases … filter-quality increases
- $|M^{all}|$ 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
- 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)

ID [hex]	T [ms]	desired	ID [hex]	T [ms]	desired
0x00A	1000		0x2F2	100	
0x010	1000	yes	0x300	1000	
0x020	100		0x350	1000	
0x060	100		0x3AC	100	
0x06A	100	yes	0x400	100	
0x06E	10		0x410	100	
0x071	100		0x411	100	
0x078	100		0x420	1000	
0x07D	100		0x425	1000	
0x081	1000	yes	0x565	500	
0x08C	100		0x610	100	
0x091	100		0x611	1000	
0x096	500		0x612	100	
0x100	10		0x613	1000	yes
0x101	10		0x614	1000	
0x102	10		0x618	100	
0x110	10		0x620	100	yes
0x120	10		0x6F0	1000	yes
0x130	10		0x6F8	1000	
0x150	10		0x700	1000	
0x160	1000		0x702	1000	
0x161	1000		0x710	1000	yes
0x200	1000		0x711	1000	
0x201	10		0x720	1000	
0x210	10	yes	0x730	1000	yes
0x220	10	yes	0x770	1000	yes
0x2F0	10		0x771	1000	
0x2F1	100				

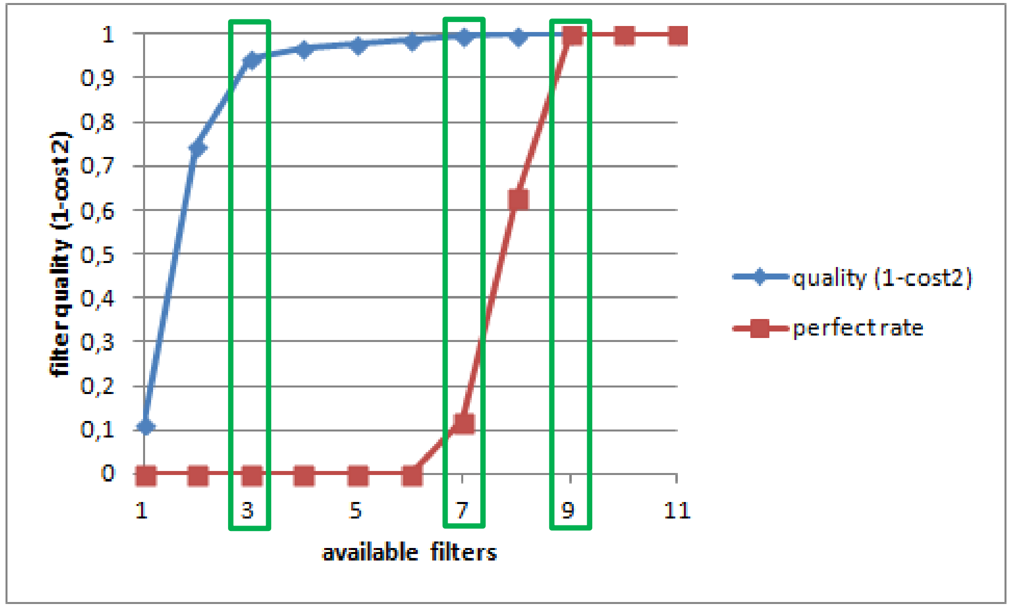
Table 2: Network Specification of Battery Electric Vehicle (BEV) available for HVAC-Controller

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

Real-World Application: Results

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

Filters	Desired [msg/sec.]	Unintended [msg/sec.]	
None	227 (11)	1180 (44)	
3	227 (11)	48 (13)	High filter quality
7	227 (11)	0 (0)	Perfect filtering (10%)
9	227 (11)	0 (0)	Perfect filtering (100%)



- 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 = 00000001010 (0x00a) UP
    ID = 00000100000 (0x020) UP
    ID = 00001100000 (0x060) UP
    ID = 00001101010 (0x06a)
    ID = 00010000001 (0x081)
---------------------------
filter = 0x000xx0000
    ID = 00000010000 (0x010)
    ID = 00000100000 (0x020) UP
    ID = 01000000000 (0x200) UP
    ID = 01000010000 (0x210)
    ID = 01000100000 (0x220)
---------------------------
filter = 11xxxxx00xx
    ID = 11000010000 (0x610) UP
    ID = 11000010001 (0x611) UP
    ID = 11000010010 (0x612) UP
    ID = 11000010011 (0x613)
    ID = 11000100000 (0x620)
    ID = 11011110000 (0x6f0)
    ID = 11100000000 (0x700) UP
    ID = 11100000010 (0x702) UP
    ID = 11100010000 (0x710)
    ID = 11100010001 (0x711) UP
    ID = 11100100000 (0x720) UP
    ID = 11100110000 (0x730)
    ID = 11101110000 (0x770)
    ID = 11101110001 (0x771) UP
```

```
filter = 00001101010
    ID = 00001101010 (0x06a)
---------------------------
filter = 00010000001
    ID = 00010000001 (0x081)
---------------------------
filter = 11000010011
    ID = 11000010011 (0x613)
---------------------------
filter = 0x000010000
    ID = 00000010000 (0x010)
    ID = 01000010000 (0x210)
---------------------------
filter = 11100x10000
    ID = 11100010000 (0x710)
    ID = 11100110000 (0x730)
---------------------------
filter = 11xx1110000
    ID = 11011110000 (0x6f0)
    ID = 11101110000 (0x770)
---------------------------
filter = x1000100000
    ID = 01000100000 (0x220)
    ID = 11000100000 (0x620)
```

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)

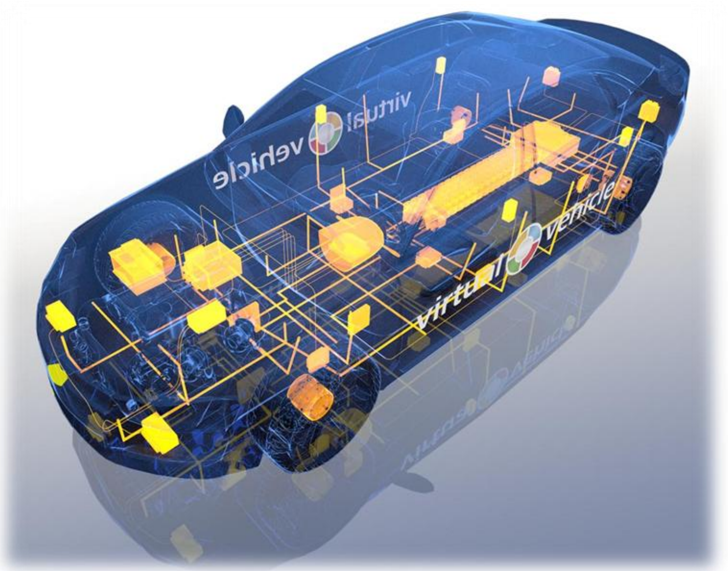
Future

- 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



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