Real-Time Scheduling and Automotive Networks

Robert Davis

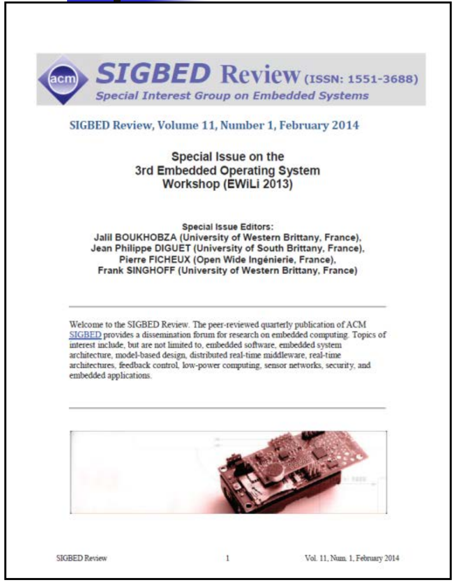
Real-Time Systems Research Group, University of York

rob.davis@york.ac.uk
http://www-users.cs.york.ac.uk/~robdavis/

Controller Area Network (CAN)

- Part 1
  - History and fundamentals of CAN
  - Original schedulability analysis
  - Revised analysis
- Part 2
  - Priority assignment - why it is so important,
  - Optimal and Robust Priority Assignment policies
- Part 3
  - FIFO queues in device drivers
  - Analysis for FIFO queued messages
  - Case study - performance effects
- Wrap up
  - Success stories and an interesting open problem

Background reading on real-time scheduling theory



SIGBED Review (ISSN: 1551-3688)
Special Interest Group on Embedded Systems

SIGBED Review, Volume 11, Number 1, February 2014

Special Issue on the
3rd Embedded Operating System
Workshop (EWiLi 2013)

Special Issue Editors:
Jalil BOUKHOBZA (University of Western Brittany, France),
Jean Philippe DIGUET (University of South Brittany, France),
Pierre FICHEUX (Open Wide Ingénierie, France),
Frank SINGHOFF (University of Western Brittany, France)

Welcome to the SIGBED Review. The peer-reviewed quarterly publication of ACM SIGBED provides a dissemination forum for research on embedded computing. Topics of interest include, but are not limited to, embedded software, embedded system architecture, model-based design, distributed real-time middleware, real-time architectures, feedback control, low-power computing, sensor networks, security, and embedded applications.

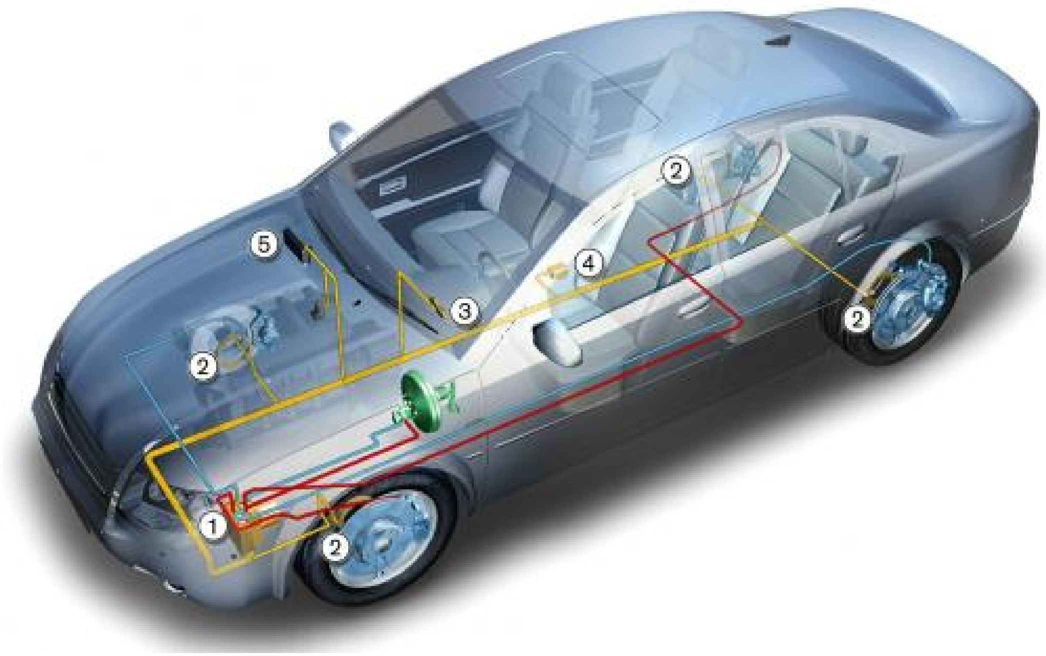
SIGBED Review 1 Vol. 11, Num. 1, February 2014

R.I. Davis "A Review of Fixed Priority and EDF Scheduling for Hard Real-Time Uniprocessor Systems ". *ACM SIGBED Review - Special Issue on the 3rd Embedded Operating Systems Workshop (Ewili 2013).* , Volume 11, Issue 1, pages 8-19, Feb 2014.
DOI: 10.1145/2597457.2597458

- Keynote presentation at ETR summer school in 2013
- Review covering scheduling theory for Fixed Priority and EDF scheduling

Controller Area Network (CAN) (Part 1)

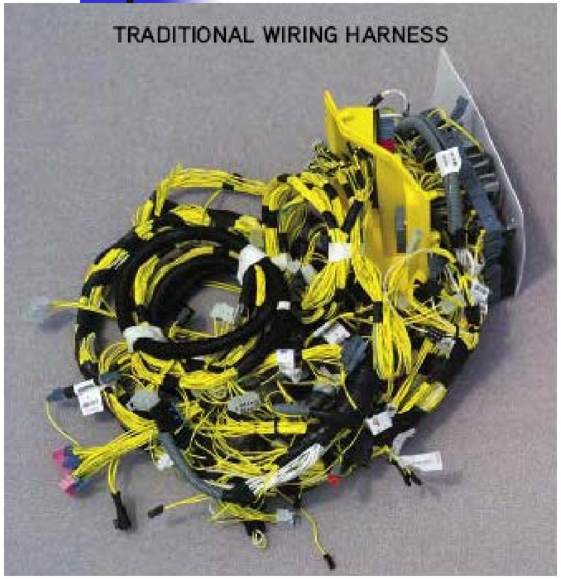
- History, Fundamentals and Schedulability analysis



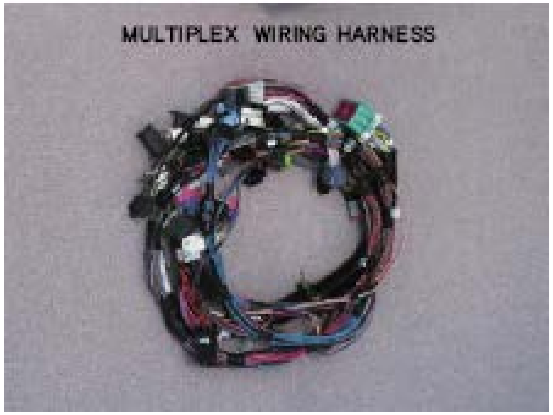
CAN History

- Controller Area Network (CAN)
  - Simple, robust and efficient serial communications bus for in-vehicle networks
- Developed by **BOSCH**
  - Starting in 1983 presented at SAE in 1986
  - Standardised by ISO in 1993 (11898)
- First CAN controller chips
  - Intel (82526) and Philips (82C200) in 1987
- First production car using CAN
  - 1991 Mercedes S-class (W140)

Multiplex v. Point-to-point Wiring



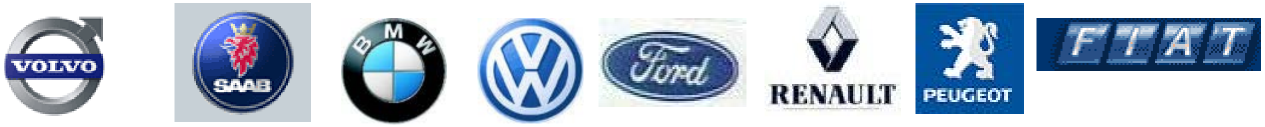
- Traditional point-to-point wiring
  - Early 1990s an average luxury car had:
    - 30Kg wiring harness
    - > 1km of copper wire
    - > 300 connectors, 2000 terminals, 1500 wires
  - Expensive to manufacture, install and maintain
    - Example: Door system with 50+ wires



- Multiplex approach (e.g. CAN)
  - Massive reduction in wiring costs
    - Example: Door system reduced to just 4 wires
  - Small added cost of CAN controllers, transceivers etc.
    - Reduced as CAN devices became on-chip peripherals

CAN in Automotive

- Other European manufacturers quickly followed Mercedes lead in using CAN

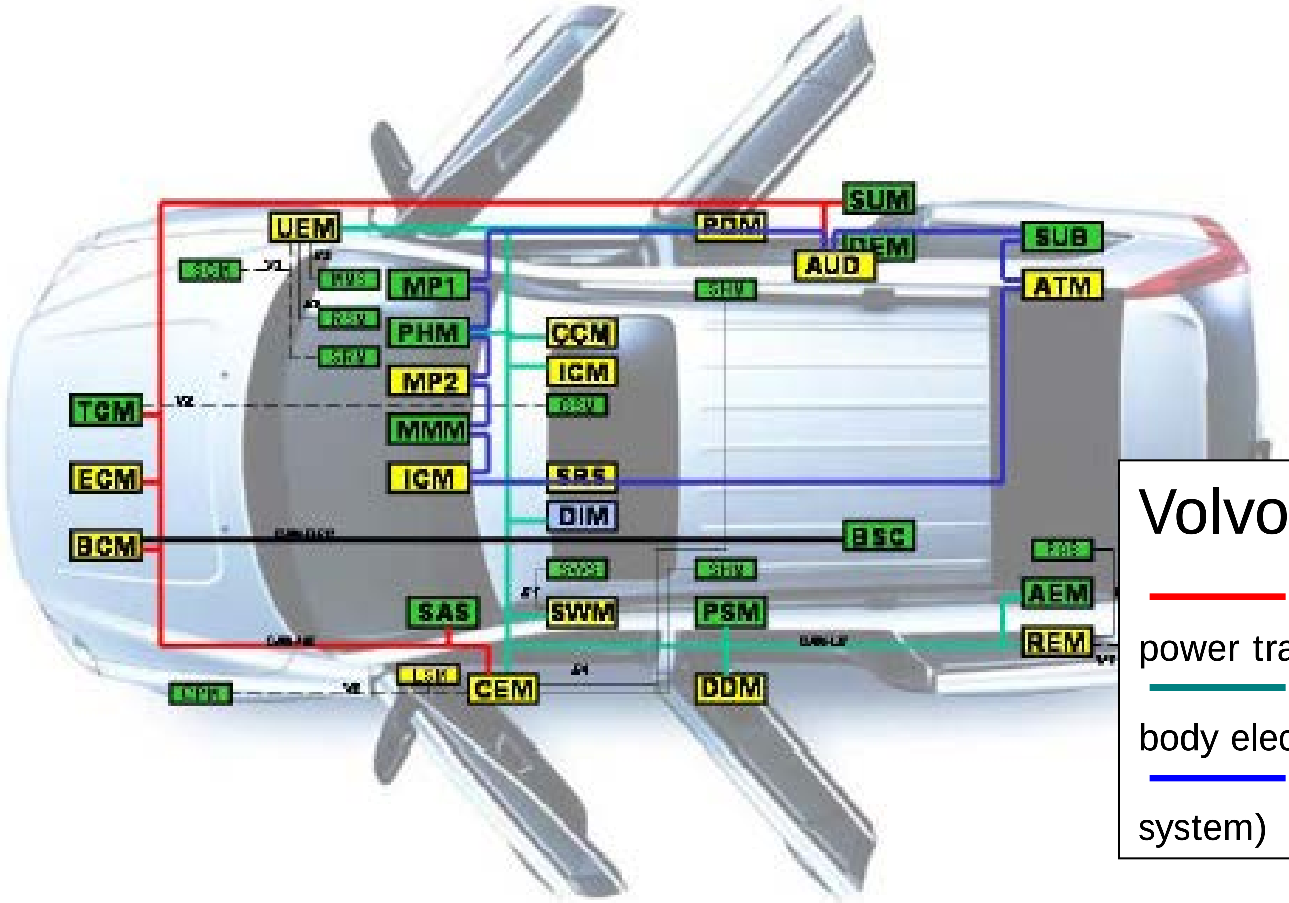


- By 2004
  - 15 different silicon vendors manufacturing over 50 different microprocessor families with on chip CAN capability
    - Analogue Devices, Atmel, Cygnal, Fujitsu, Infineon, Maxim formally Dallas, Microchip, Mitsubishi, Motorola, NEC, Phillips, Renesas, Siemens, Silicon Laboratories, and STMicroelectronics
- By 2008
  - EPA rules for On Board Diagnostics made CAN mandatory for cars and light trucks sold in the US

CAN today

- CAN is used in nearly all cars sold today
  - Approx. 1 billion CAN enabled microcontrollers sold each year
  - Typical cars today have 20 – 30 ECUs inter-connected via 2 or more CAN buses
- Multiple networks
  - High speed" (500 Kbit/sec) network connecting chassis and power train ECUs
    - E.g. transmission control, engine management, ABS etc.
  - Low speed (100-125 Kbit/sec) network(s) connecting body and comfort electronics
    - E.g. door modules, seat modules, climate control etc.
  - Data required by ECUs on different networks
    - typically "gatewayed" between them via a powerful microprocessor connected to both

Volvo XC90 Network Architecture



Volvo XC90

🟥 500 Kbit/sec CAN bus for power train

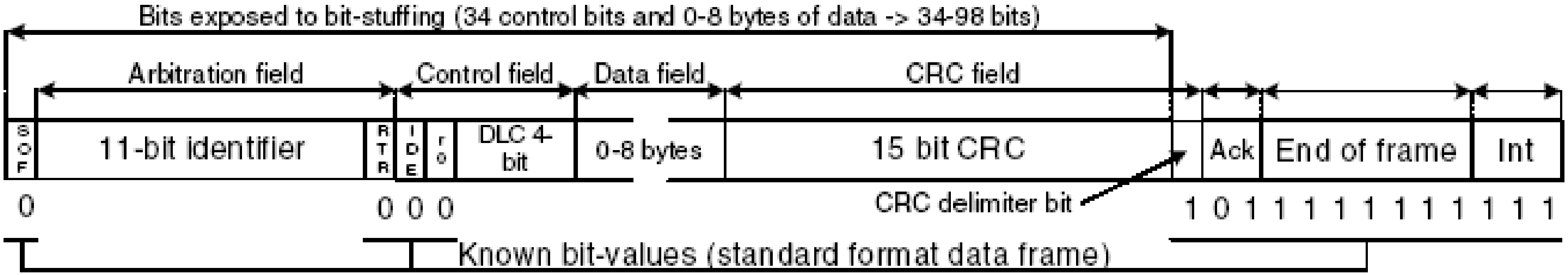
🟩 125 Kbit/sec CAN bus for body electronics

🟦 MOST (infotainment system)

Information on CAN

- CAN used to communicate *signals* between ECUs
  - Signals typically range from 1 to 16-bits of information
  - wheel speeds, oil and water temperature, battery voltage, engine rpm, gear selection, accelerator position, dashboard switch positions, climate control settings, window switch positions, fault codes, diagnostic information etc.
  - > 2,500 signals in a high-end vehicle
  - Multiple signals piggybacked into CAN messages to reduce overhead, but still 100’s of CAN messages
- Real-time constraints on signal transmission
  - End-to-end deadlines in the range 10ms – 1sec
  - Example LED brake lights

CAN Protocol: Data Frame Format



- Start of frame (synchronisation)
- **Identifier determines priority for access to bus** (11-bit or 29-bit)
- Control field (Data length code)
- **0-8 bytes useful data**
- 15-bit CRC
- Acknowledgement field
- End of frame marker
- Inter-frame space (3 bits)

CAN Protocol

- CAN is a multi-master CSMA/CR serial bus
  - Collision resolution is based on priority
  - CAN physical layer supports two states: “0” dominant, “1” recessive
- Message transmission
  - CAN nodes wait for “bus idle” before starting transmission
  - Synchronise on the SOF bit (“0”)
  - Each node starts to transmit the identifier for its highest priority (lowest identifier value) ready message
  - If a node transmits “1” and sees “0” on the bus, then it stops transmitting (lost arbitration)
  - Node that completes transmission of its identifier continues with remainder of its message (wins arbitration)
  - Unique identifiers ensure all other nodes have backed off

CAN Protocol: Message Arbitration

- Message arbitration based on priority

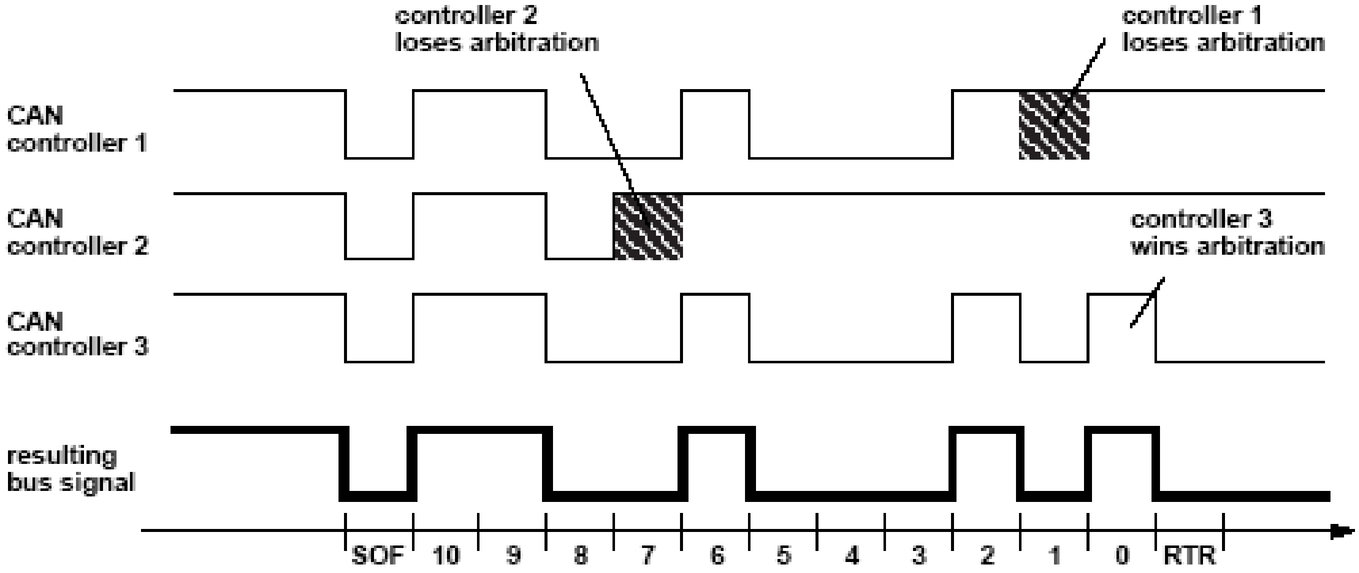
Identifiers

11001000110

11011000111

11001000101

11001000101



CAN: Schedulability Analysis

- CAN network scheduling resembles single processor fixed priority non-pre-emptive scheduling
  - Messages compete for access to the bus based on priority
  - Effectively a global queue with transmission in priority order
  - Once a message starts transmission it cannot be pre-empted

- Schedulability Analysis for CAN
  - First derived by Ken Tindell during 1993-1995 from earlier work on fixed priority pre-emptive scheduling
    - Calculates worst-case response times of all CAN messages
    - Used to check if all CAN messages meet their deadlines in the worst-case
    - Possible to engineer CAN based systems for timing correctness, rather than “test and hope”

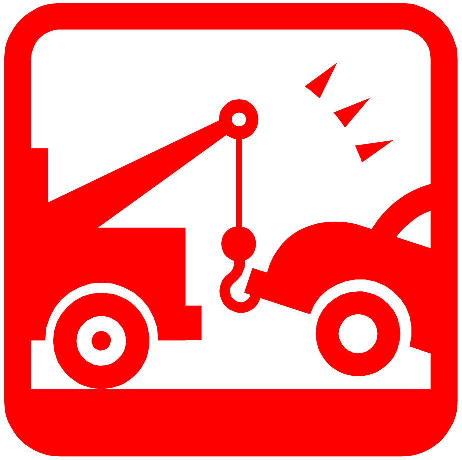
Controller Area Network

- The original analysis was:
  - Used widely in teaching
  - Referenced in over 500 subsequent research papers
  - Lead to at least two PhD Theses
  - In 1995 recognised by Volvo Car Corporation used in the development of the Volvo S80 (P23)
  - Formed basis of commercial CAN analysis tools now owned by Mentor Graphics
  - Used by many Automotive manufacturers who built millions of cars with networks analysed using these techniques
  - Enabled increases in network utilisation from 30-40% to typically 70-80%

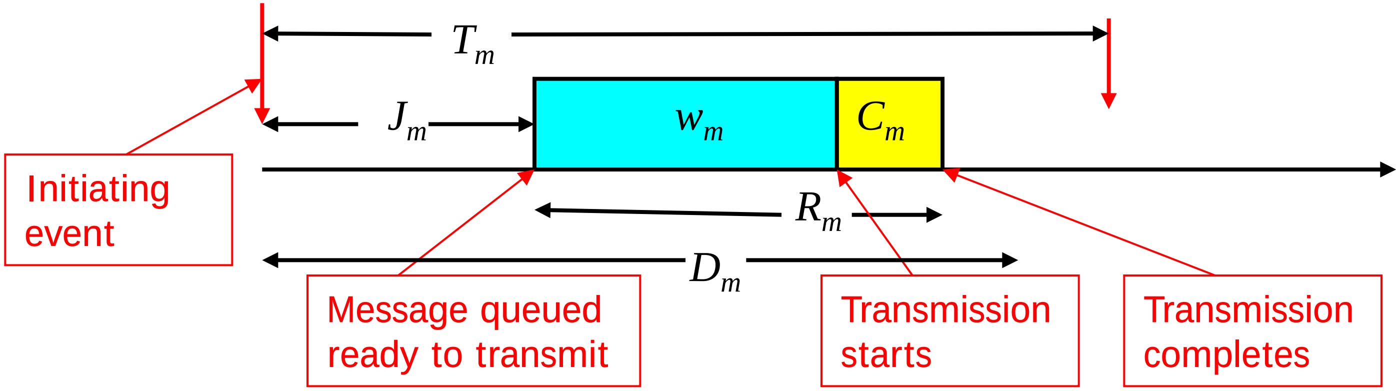
[K.W. Tindell, A. Burns, A.J. Wellings, “Calculating Controller Area Network (CAN) Message Response Times”, Control Engineering Practice, Vol 3, No 8, pp1163-1169, 1995. DOI:10.1016/0967-0661(95)00112-8]

Unfortunately...

- The original schedulability analysis for CAN was flawed...



Schedulability Analysis: Model



- Each CAN message has a:
  - Unique priority *m* (identifier)
  - Maximum transmission time C_m
  - Minimum inter-arrival time or period T_m
  - Deadline $D_m <= T_m$
  - Maximum queuing jitter J_m
  - Transmission deadline $E_m = D_m - J_m$
- Compute:
  - Worst-case queuing delay w_m
  - Worst-case response time $R_m = w_m + C_m$
  - Compare with transmission deadline $R_m \leq E_m$

Schedulability Analysis: TX Time

- Maximum transmission time
  - Bit stuffing
    - Bit patterns “000000” and “111111” used to signal errors
    - Transmitter insert 0s and 1s to avoid 6 consecutive bits of same polarity in messages
  - Increases transmission time of message

11-bit identifiers: $C_m = (55 + 10 s_m)\tau_{bit}$

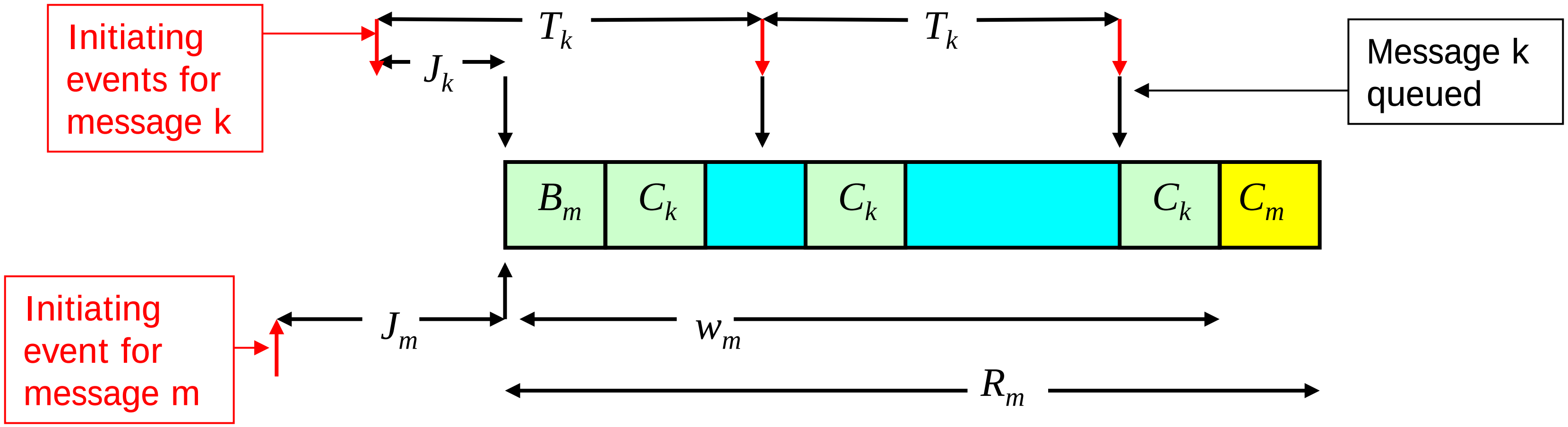
29-bit identifiers: $C_m = (80 + 10 s_m)\tau_{bit}$

Original Schedulability Analysis for CAN

- Blocking $B_m = \max_{k \in lp(m)} (C_k)$
- Queuing delay $$w_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$
- Response time $R_m = w_m + C_m$
- Deadline met? $R_m \le E_m = (D_m - J_m)$

Blocking from longest lower priority msg

Interference from multiple instances of higher priority msgs

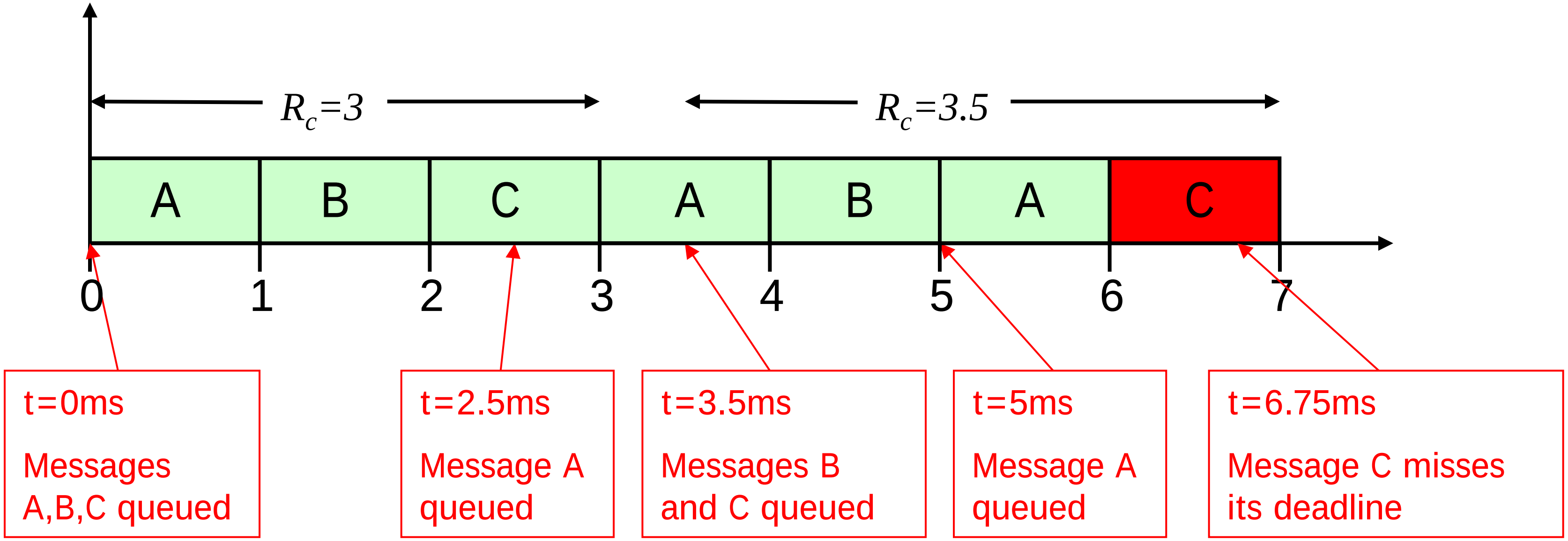


Schedulability Analysis: Example

- 125 Kbit/s bus
- 11-bit identifiers
- 3 messages with 7 data bytes each, max. 125 bits including bit stuffing

Message	Priority	Period	Deadline	TX Time	R
A	1	2.5ms	2.5ms	1ms	2ms
B	2	3.5ms	3.25ms	1ms	3ms
C	3	3.5ms	3.25ms	1ms	3ms

Response time of message C



t=0ms

Messages A,B,C queued

t=2.5ms

Message A queued

t=3.5ms

Messages B and C queued

t=5ms

Message A queued

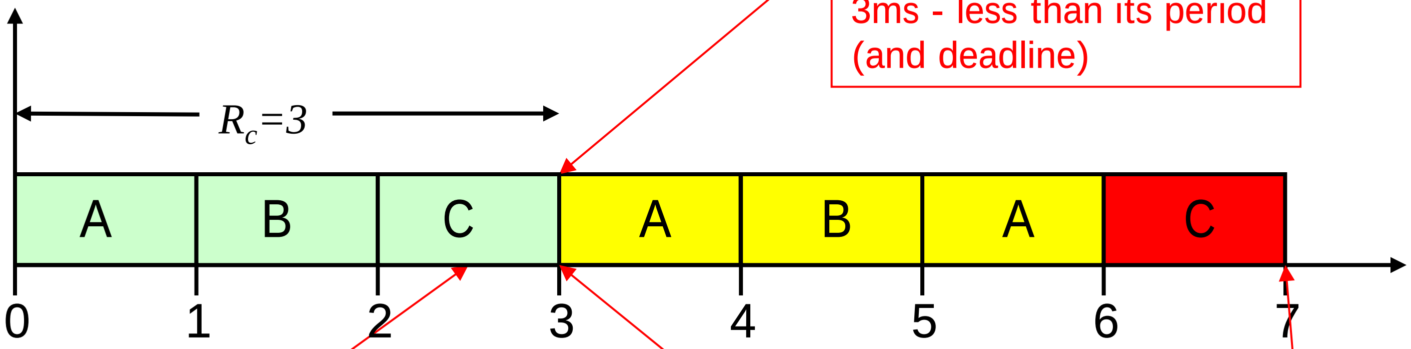
t=6.75ms

Message C misses its deadline

The original schedulability analysis gives an optimistic response time for message C: 3ms v. 3.5ms

2nd instance of message C misses its deadline

What is the flaw in the analysis?



Response time of 1st instance of message C is 3ms - less than its period (and deadline)

BUT transmission of message C is non-pre-emptive and blocks message A, pushing extra interference into next period of C

Busy period at priority of message C does NOT end with transmission of message C

Busy period ends here. Must examine all instances of message C in the busy period to find WCRT

Revised Schedulability Analysis

- Find length of longest busy period for message *m.*
  - *(Busy period includes all instances of message m and higher priority messages queued strictly before the end of the busy period)*

$$t_m^{n+1} = B_m + \sum_{\forall k \in hp(m) \cup m} \left\lceil \frac{t_m^n + J_k}{T_k} \right\rceil C_k$$

  - Starts with $t_m^0 = C_m$
- Number of instances of message *m* ready before end of busy period

$$Q_m = \left\lceil \frac{t_m + J_m}{T_m} \right\rceil$$

Revised Schedulability Analysis

- For each instance q $(q = 0$ *to* $Q_m - 1)$ of message *m* in the busy period, compute the longest time from the start of the busy period to that instance starting transmission:

$$w_m^{n+1}(q) = B_m + qC_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$

- Response time of instance q of message *m*:

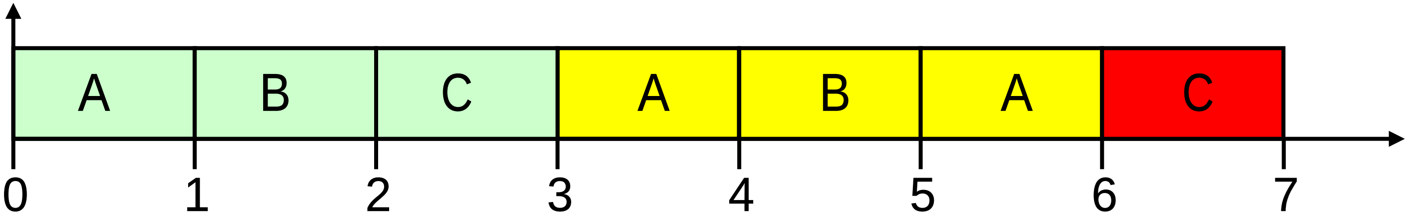
$$R_m(q) = w_m(q) - qT_m + C_m$$

- Worst-case response time of message *m*:

$$R_m = \max_{q=0..Q_m-1} (R_m(q))$$

Example Revisited

Message	Priority	Period	Deadline	TX Time
A	1	2.5ms	2.5ms	1ms
B	2	3.5ms	3.25ms	1ms
C	3	3.5ms	3.25ms	1ms



Message	Busy period	Q	R(0)	R(1)	R max
A	2ms	1	2ms	-	2ms ✓
B	5ms	2	3ms	1.5ms	3ms ✓
C	7ms	2	3ms	3.5ms	3.5ms ✓

Sufficient Schedulability Test #1

- 1st invocation of message *m:*

$$w_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$

- For messages with $D_m <= T_m$ and schedulable 1st instance, then a pessimistic view of 2nd and subsequent instances is a *critical instant* with indirect or push-through blocking of C_m from the previous instance of message *m*

$$w_m^{n+1} = C_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$

- Combined:

$$w_m^{n+1} = \max(B_m, C_m) + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$

Sufficient Schedulability Test #2

- Let maximum possible transmission time of the longest possible message on the network be: C^{MAX}

- Always assume this as the blocking factor

$$B^{MAX} = C^{MAX}$$

- As $B^{MAX} \geq \max(B_m, C_m)$

- Simple sufficient schedulability test

$$w_m^{n+1} = B^{MAX} + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$

Impact on deployed CAN systems

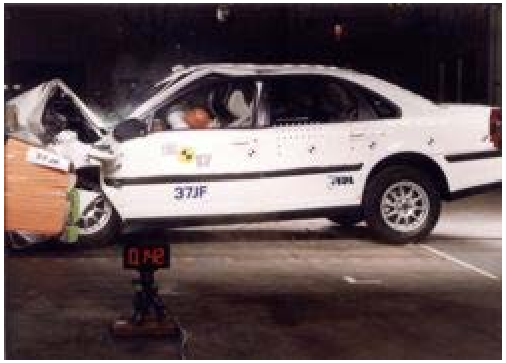
- Could the flaw in the original analysis cause problems in practice?
  - Typical systems have 8 data byte diagnostic messages:

    no problems in normal operation
  - Analysis used allows for errors:

    no issues when errors not present
  - Typically all messages have 8 data bytes:

    only lowest priority message could be affected
  - Deadline failures require worst-case phasing, worst-case bit stuffing and errors on the bus:

    very low probability of occurrence
  - Systems designed to be resilient to some messages missing their deadlines and simpler problems such as intermittent wiring faults



Commercial CAN Analysis Tools

- Volcano Network Architect
  - Commercial CAN schedulability analysis product
  - Uses the simple sufficient schedulability test #2, assuming maximum blocking factor irrespective of message priorities / number of data bytes

$$w_m^{n+1} = B^{MAX} + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$$

✓

  - Slightly pessimistic but correct upper bound on message worst-case response times
  - Used to analyse CAN systems for Volvo S80, S/V/XC 70, S40, V50, XC90 and many other cars from other manufacturers including Jaguar, Land Rover, Mazda, SAIC and others

Further reading

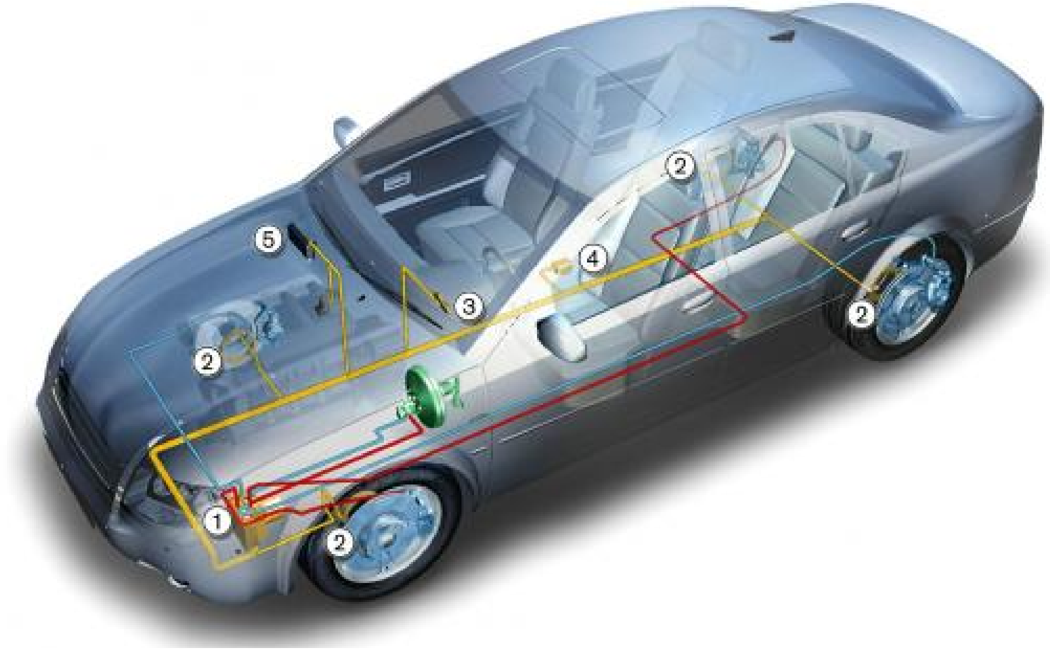
REAL-TIME SYSTEMS

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*, Volume 35, Number 3, pp. 239-272, April 2007.
DOI: 10.1007/s11241-007-9012-7

- Open access – freely available
- This is now the reference work for analysis of CAN
- Often the most downloaded paper from the journal

Controller Area Network (CAN) (Part 2)

- Priority Assignment



Priority Assignment

- With Fixed Priority Scheduling analysis is only half the story...

- What about **Priority** Assignment?

- Priority assignment is important because it has a big effect on schedulability
  - Achieve a schedulable network when it otherwise wouldn't be or provide headroom for other messages to be added
  - Provide a schedulable network at the lowest bus speed
  - Provide the maximum tolerance to errors or interference on the bus without deadlines being missed

When priority assignment goes bad!

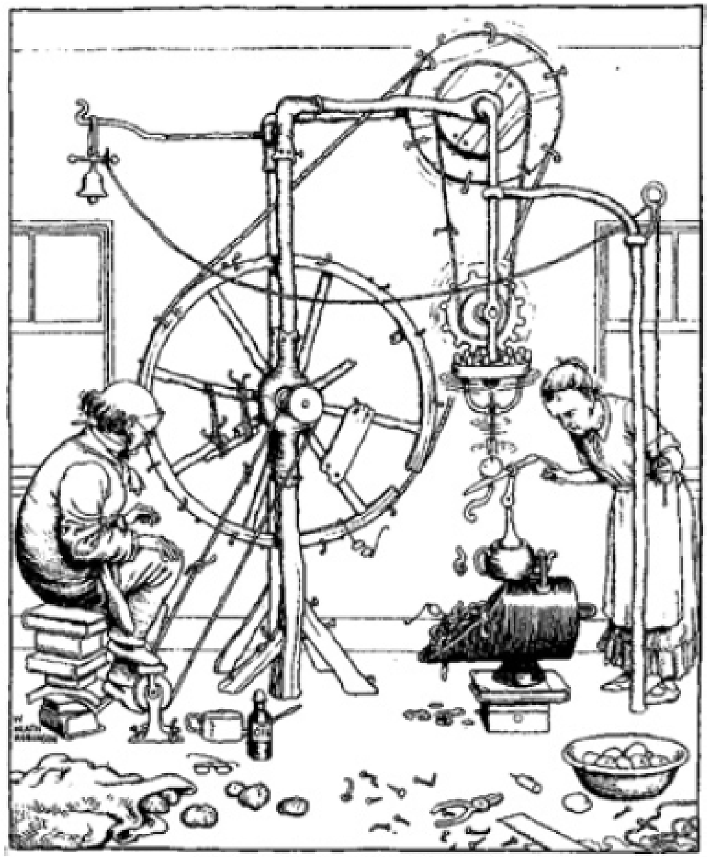
- From Darren Buttle's Keynote talk at ECRTS 2012

  The **myth** of CAN bus Utilisation – believed by many in industry

  "You cannot run CAN reliably at more than 30% utilisation[1]"

  [1] Figures may vary but not significantly

- Why?
  - Message IDs i.e. priorities assigned in an ad-hoc way reflecting data and ECU supplier (legacy issues)
  - ...as well as many other issues, including device driver implementation



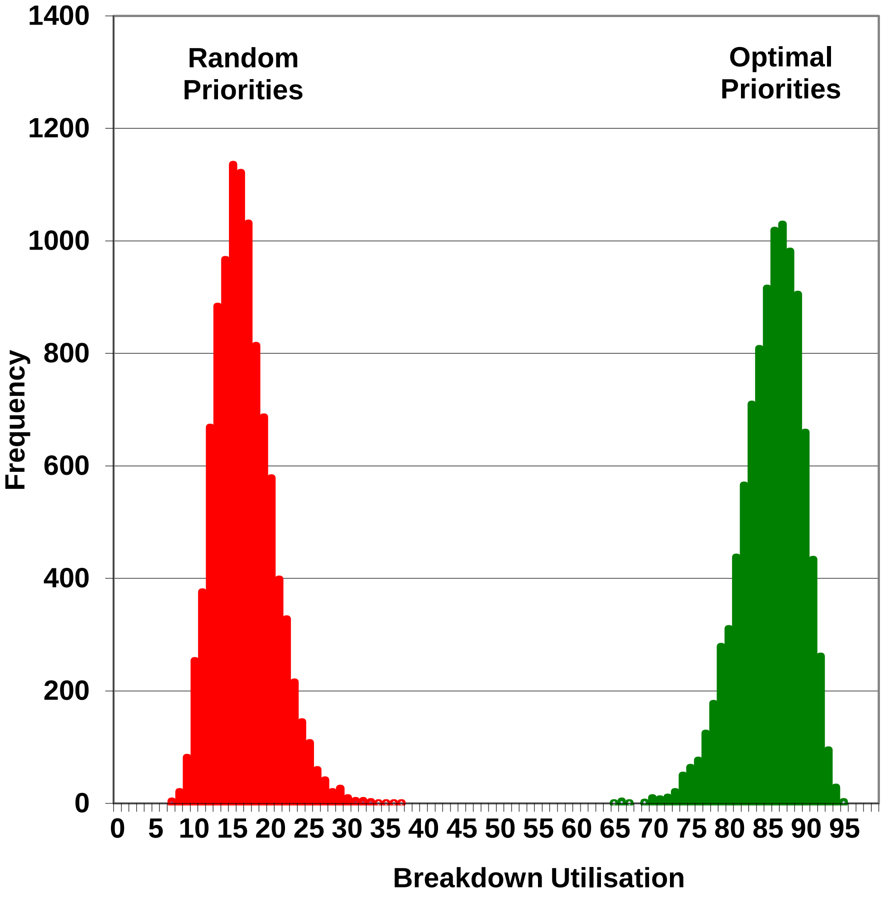
The Professor's invention for peeling potatoes.

When priority assignment goes bad!

- Example: CAN
  - Typical automotive config:
    - 80 messages
    - 10ms -1s periods
    - All priority queues
  - x10,000 message sets

- Breakdown utilisation
  - Scale bus speed to find bus utilisation at which deadlines are missed

80% v 30% or less



[R.I. Davis, S. Kollmann, V. Pollex, F. Slomka, "Schedulability Analysis for Controller Area Network (CAN) with FIFO Queues Priority Queues and Gateways". *Real-Time Systems,* 2012]

Optimal Priority Assignment

- Formal definition: Optimal priority assignment

For a given system model, a priority assignment policy P is referred to as optimal if there are no systems, compliant with the model, that are schedulable using fixed priority scheduling with another priority assignment policy that are not also schedulable using policy P.

By using an Optimal Priority Assignment policy we can schedule any system that can be scheduled using any other priority assignment policy

[N.C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", Technical Report YCS 164, Dept. Computer Science, University of York, UK, 1991.]

[N.C. Audsley, “On priority assignment in fixed priority scheduling”, Information Processing Letters, 79(1): 39-44, May 2001.]

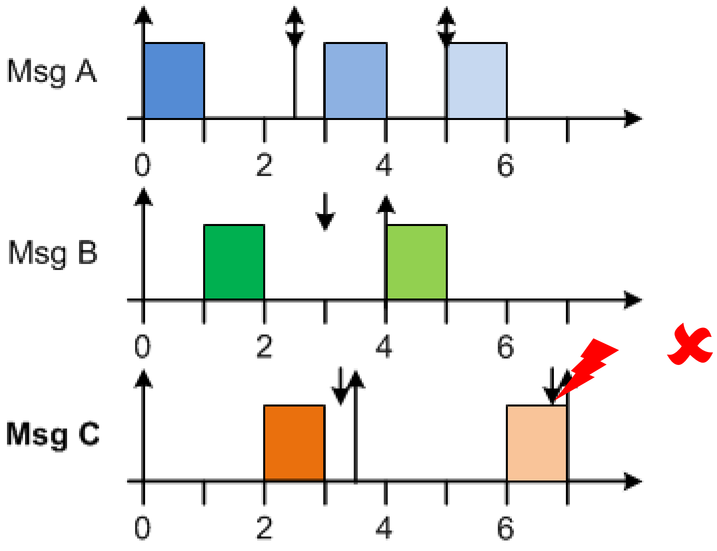
Optimal Priority Assignment for CAN

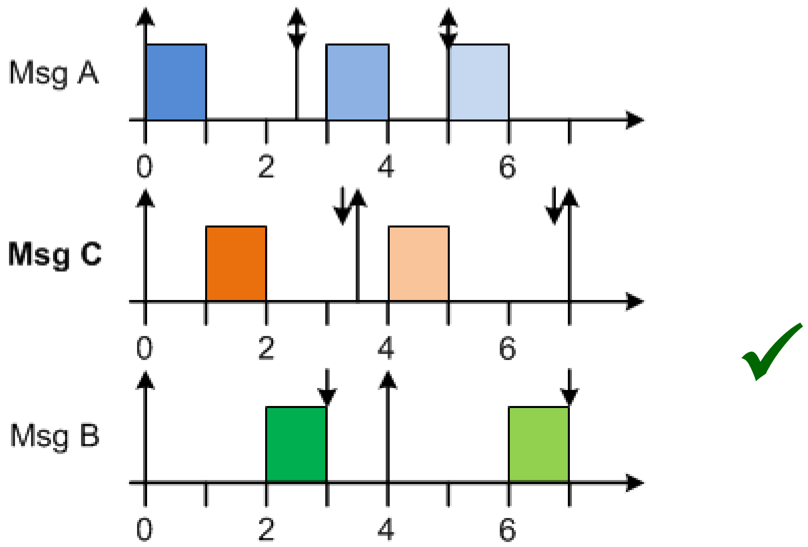
- Transmission Deadline Monotonic (D-J) ?
  - Assigns priorities based on message transmission deadlines, the shorter the transmission deadline, the higher the priority (ties broken arbitrarily)
  - Optimal for pre-emptive scheduling with constrained deadlines and jitter
  - Typically a good heuristic, but not optimal for non-pre-emptive scheduling, so not optimal for CAN

Transmission Deadline Monotonic: non-optimality

- Non-pre-emptive scheduling

Msg	Tx Time	Deadline	Period
A	1ms	2.5ms	2.5ms
B	1ms	3ms	4ms
C	1ms	3.25ms	3.5ms



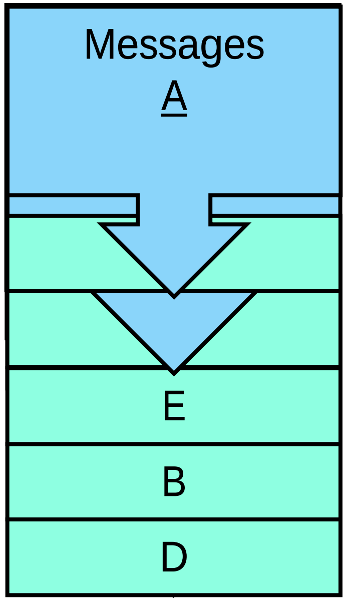


[L. George, N. Rivierre, M. Spuri, “Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling”, INRIA Research Report, No. 2966, September 1996]

Example from: [R.I. Davis and A. Burns "Robust priority assignment for messages on Controller Area Network (CAN)”. Real-Time Systems, Volume 41, Issue 2, pages 152-180, February 2009]

Audsley’s Optimal Priority Assignment algorithm

```
for each priority level i, lowest first {
    for each unassigned message msg {
         if msg is schedulable at priority i
         assuming that all unassigned messages are
         at higher priorities {
              assign msg to priority level i
              break (exit for loop)
         }
    }
    if no messages are schedulable at priority i {
         return unschedulable
    }
}
return schedulable
```



$n(n+1)/2$ schedulability tests rather than $n!$
by exploring all possible orderings
n = 25, that is 325 tests rather than 15511210043330985984000000

[N.C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", Technical Report YCS 164, Dept. Computer Science, University of York, UK, 1991.]

[N.C. Audsley, “On priority assignment in fixed priority scheduling”, Information Processing Letters, 79(1): 39-44, May 2001.]

[K. Bletsas, and N.C. Audsley, “Optimal priority assignment in the presence of blocking”. *Information Processing Letters* Vol. 99, No. 3, pp83-86, August. 2006]

Robust Priority Assignment

- Drawback of OPA algorithm
  - Arbitrary choice of schedulable messages at each priority
  - May leave the network only just schedulable – i.e fragile not robust to minor changes

- In practice want a robust priority ordering
  - As well as being optimal, a robust ordering is able to tolerate the maximum amount of additional interference of any priority ordering without missing a deadline
  - General model of additional interference $E(\alpha,w,i)$ (e.g. $=\alpha$)
  - Examples: α *gives number of errors on the bus or* α *is the number of bits of additional interference (bus unavailability) tolerated*

[R.I. Davis, A. Burns. "Robust Priority Assignment for Fixed Priority Real-Time Systems". In proceedings IEEE Real-Time Systems Symposium pp. 3-14. Tucson, Arizona, USA. December 2007.]

Robust Priority Assignment (RPA) Algorithm

```
for each priority level i, lowest first
{
    for each unassigned message msg
    {
        determine the largest value of $\alpha$ for which msg
        is schedulable at priority i assuming that all
        unassigned messages have higher priorities
    }
    if no messages are schedulable at priority i
    {
        return unschedulable
    }
    else
    {
        assign the schedulable message that tolerates the
        max $\alpha$ at priority i to priority i
    }
}
return schedulable
```

Ordering achieved is optimal and robust (tolerates the most additional interference (largest α) of any priority ordering)

Robust Priority Assignment

- Example:
  - 5 messages
  - Bus speed 125Kbits/s

Message	Period (ms)	Deadline (ms)	Number of bits	Tx Time (ms)
A	5.75	5.75	135	1.08
B	125	6.75	135	1.08
C	7.25	7.25	65	0.52
D	15.0	15.0	135	1.08
E	17.3	17.3	65	0.52

Robust Priority Assignment: maximising errors tolerated

- Computed values of α = number of errors tolerated

Priority	Message				
	A	B	C	D	E
5	0	1	0	4	**4**
4	0	1	1	**4**	-
3	1	**2**	1	-	-
2	2	-	**2**	-	-
1	**2**	-	-	-	-

- Robust priority ordering
  - (A,C,B,D,E) All messages tolerate at least **2** errors in the worst case
- Deadline monotonic: neither optimal nor robust
  - (A,B,C,D,E) All messages tolerate at least 1 error in the worst case
- OPA: may be worse still
  - Could choose (E,D,C,B,A) which is schedulable but tolerates no errors

Robust Priority Assignment: Example maximising interference

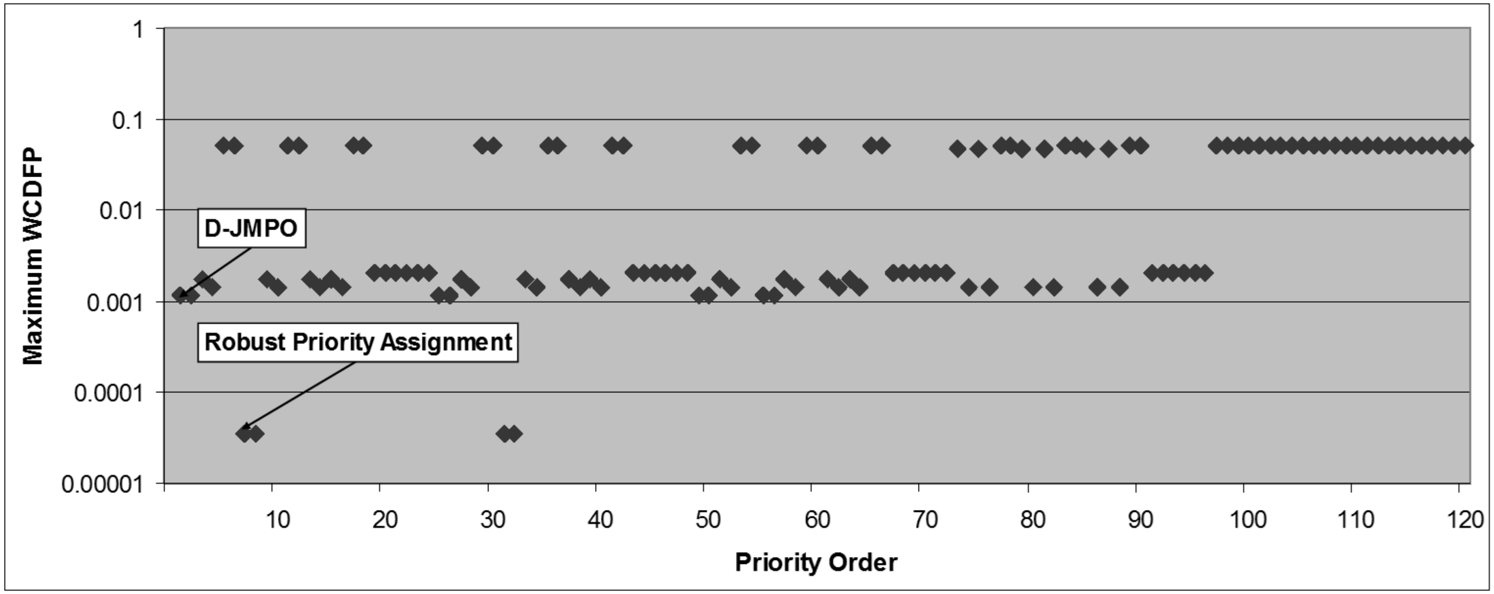
- Computed values of α = additional interference in units of τ_{bit}

Priority	Message				
	A	B	C	D	E
5	51	176	112	681	**690**
4	186	311	247	**960**	-
3	251	**376**	312	-	-
2	386	-	**447**	-	-
1	**451**	-	-	-	-

- Robust priority ordering
  - (A,C,B,D,E) All messages tolerate additional interference of **376** τ_{bit}
- Deadline monotonic: neither optimal nor robust
  - (A,B,C,D,E) All messages tolerate additional interference of 312 τ_{bit}
- OPA: may be worse still
  - Could choose (E,D,C,B,A) which tolerates only 51 τ_{bit}

RPA: minimising probability of deadline failure

- RPA can also be adapted to minimise Worst-Case Deadline Failure Probability (WCDFP) assuming an error model
    - Can reduce by orders of magnitude the WCDFP



[R.I. Davis and A. Burns "Robust priority assignment for messages on Controller Area Network (CAN)". *Real-Time Systems*, Volume 41, Issue 2, pages 152-180, February 2009.]

Further Reading

REAL-TIME SYSTEMS

R.I. Davis and A. Burns "Robust priority assignment for messages on Controller Area Network (CAN)". *Real-Time Systems*, Volume 41, Issue 2, pages 152-180, February 2009. DOI: 10.1007/s11241-008-9065-2

- Robust Priority Assignment for CAN
- How to maximise tolerance to errors, and provide maximum headroom for additional messages
- How to minimise the probability of errors on the bus causing a deadline to be missed

Priority Assignment on CAN

- How to obtain an effective priority assignment?
  - Simple heuristic: Assign priorities based on transmission deadlines
  - Best practice: Use the Robust Priority Assignment algorithm allowing for extra interference on the bus.

- To get high schedulable bus utilisation requires:
  - Appropriate allocations of message IDs (i.e. priorities)
  - Priority queues in all nodes
  - Achieved by Volvo using Volcano 15 years ago! (70-80% utilisation with headroom for errors on the bus).
  - Rarely achieved today due to:

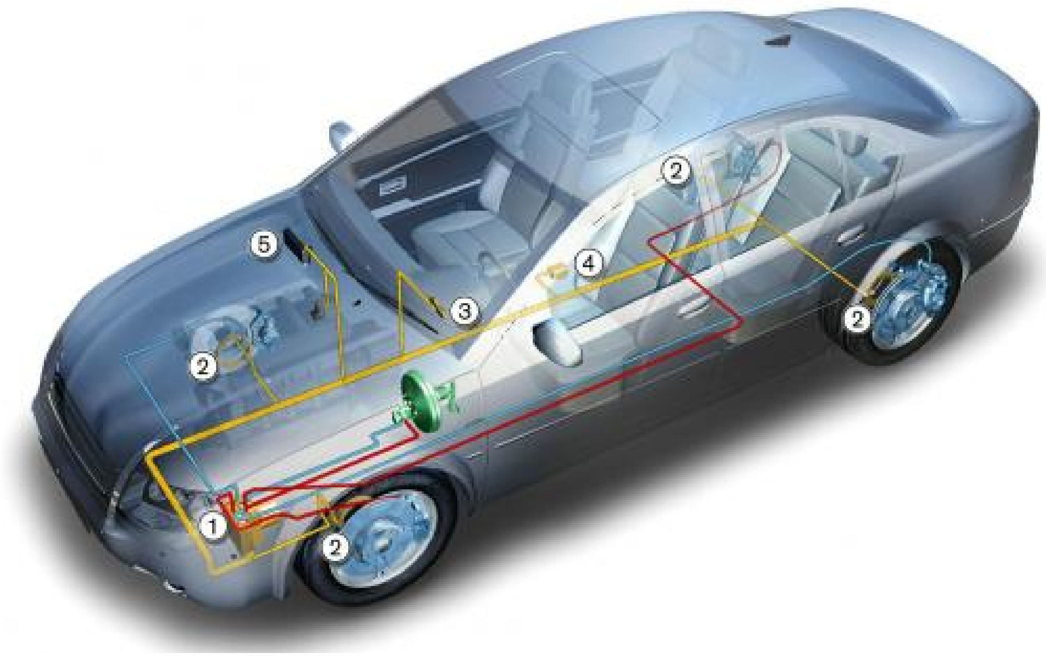
    - ad-hoc message ID allocations

    - use of FIFO queues in device drivers

[R.I. Davis and A. Burns "Robust priority assignment for messages on Controller Area Network (CAN)". *Real-Time Systems*, Volume 41, Issue 2, pages 152-180, February 2009.]

Controller Area Network (CAN) (Part 3)

- FIFO queues



FIFO queues and no Tx abort

- Classical analysis only holds if every node can always enter its highest priority ready message into bus arbitration
- This may not always be the case:
  - It may not be possible to abort a lower priority message in a transmit buffer – can be an issue if there are fewer transmit buffers than transmitted messages (e.g. in gateways)
  - Device drivers may implement FIFO rather than priority queues
    - Simpler to implement
    - Less code / lower CPU load
    - Designers may not understand the impact this can have on network performance – there is an illusion that faster queue management improves system performance
  - Hardware support for FIFO queues in BXCAN and BECAN (ST7 and ST9 microcontrollers)

Schedulability analysis: FIFO queued messages

- FIFO-symmetric analysis
  - Attributes the same upper bound response time to all messages in a group that share the same FIFO queue.
- Makes (pessimistic) worst-case assumptions
  - Considers lowest priority of any message in the group L_m
  - Indirect blocking due to longest message in the group C_m^{MAX}
  - Last message to be sent assumed to have length C_m^{MIN} allowing interference for the longest possible time
  - Messages already in the FIFO queue of max total length $C_m^{SUM} - C_m^{MIN}$

    (As all messages have $D_j \leq T_j$ then in a schedulable system, there can be at most one instance of any message in a FIFO queue at any given time)
  - E_m^{MIN} minimum transmission deadline of any message in the group
  - f_m (for hp messages) extra jitter due to buffering time spent in a FIFO queue and so not taking part in priority-based arbitration

Schedulability analysis: FIFO queued messages

- FIFO-symmetric analysis
  - Queuing delay

Delay due to other messages in same FIFO

Extra jitter due to buffering delay in FIFO queue

$$w_m^{n+1} = \max(B_{L_m}, C_m^{MAX}) + (C_m^{SUM} - C_m^{MIN}) + \sum_{\forall k \in hp(L_m) \wedge k \notin M(m)} \left\lceil \frac{w_m^n + J_k + f_k + \tau_{bit}}{T_k} \right\rceil C_k$$

Push through blocking due to longest message in same FIFO

Assume lowest priority of any message in the FIFO group

Response time $R_m = w_m^{n+1} + C_m^{MIN}$

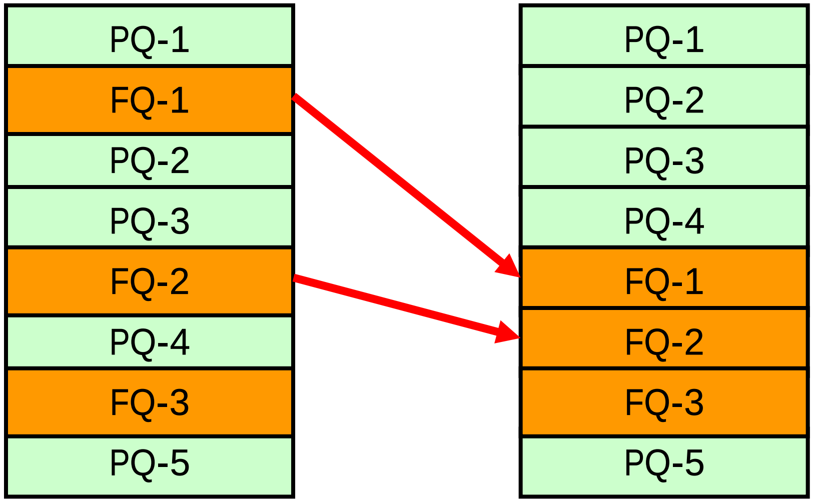
Assume min length message to get max interference

- Message *m* schedulable if $R_m \le E_m^{MIN}$

Minimum ‘transmission deadline’ (D-J) of any message sharing the FIFO

Priority Assignment

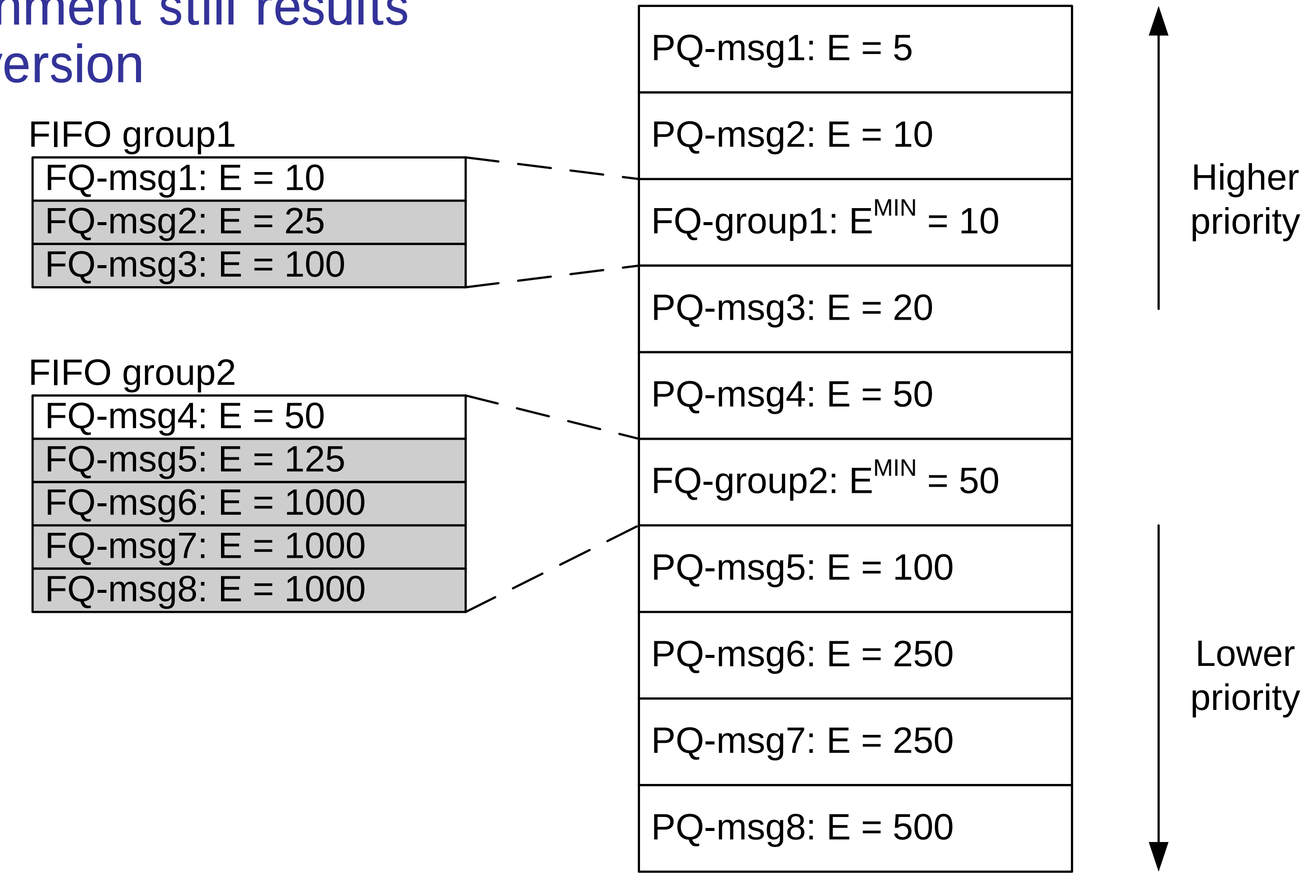
- FIFO-adjacent priority ordering
  - Optimal Partial ordering for messages within a FIFO-group have adjacent priorities – no interleaving with other messages
- Modified versions of OPA and RPA algorithms
  - Give optimal and robust priority ordering by assigning messages in each FIFO-group together in a band at adjacent priorities



- Transmission deadline monotonic priority ordering
  - Optimal w.r.t. the sufficient schedulability test when all messages have the same max. transmission time

Priority inversion

- With FIFO queues, optimal priority assignment still results in priority inversion



FIFO group1

FQ-msg1: E = 10

FQ-msg2: E = 25

FQ-msg3: E = 100

FIFO group2

FQ-msg4: E = 50

FQ-msg5: E = 125

FQ-msg6: E = 1000

FQ-msg7: E = 1000

FQ-msg8: E = 1000

PQ-msg1: E = 5

PQ-msg2: E = 10

FQ-group1: E^{MIN} = 10

PQ-msg3: E = 20

PQ-msg4: E = 50

FQ-group2: E^{MIN} = 50

PQ-msg5: E = 100

PQ-msg6: E = 250

PQ-msg7: E = 250

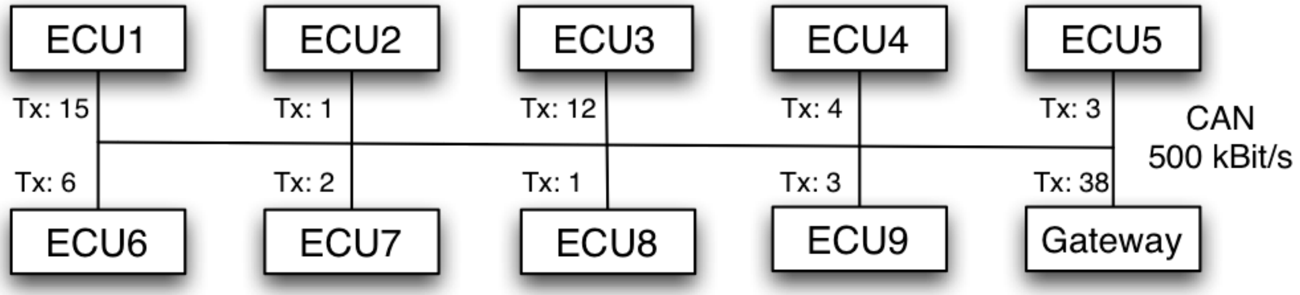
PQ-msg8: E = 500

Higher priority

Lower priority

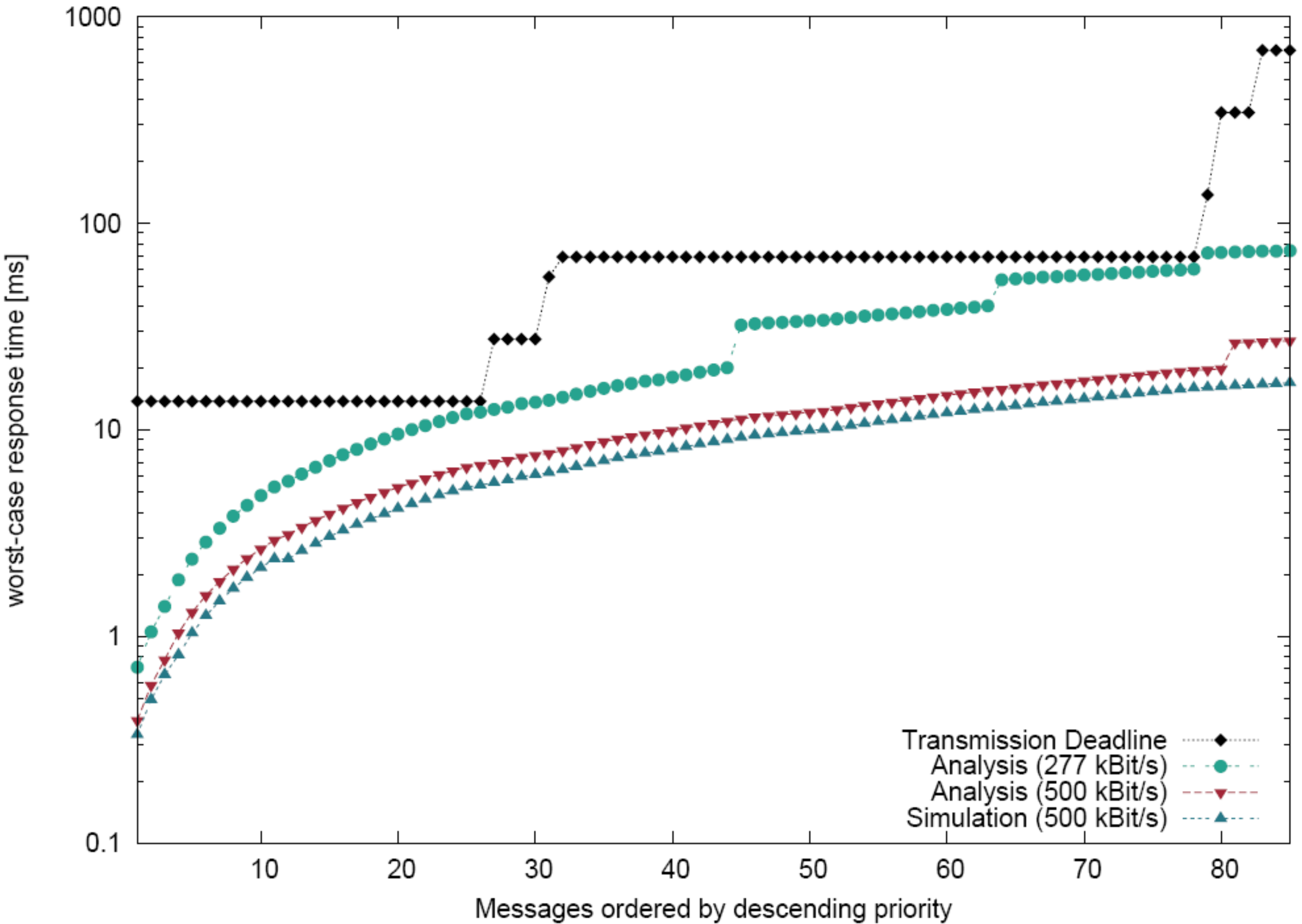
Automotive Case Study

- 10 ECUs, 85 messages



- Configuration
  - 500Kbit/s bus
  - Gateway sends 38 messages
- Experiments
  - Different numbers of FIFO queues

Expt 1: All priority queues



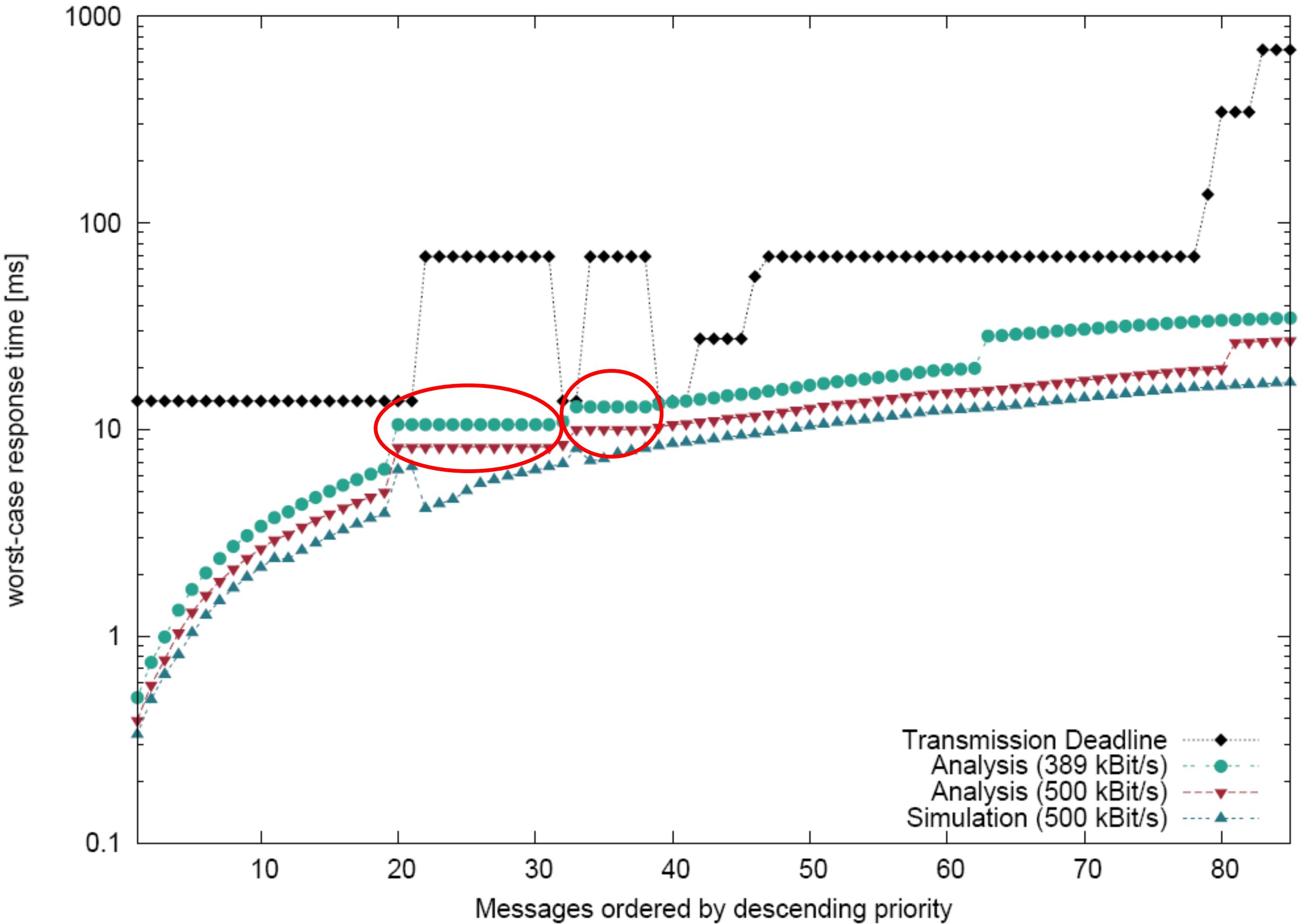
Min bus speed

277 Kbit/s

Max bus Util.

84.5%

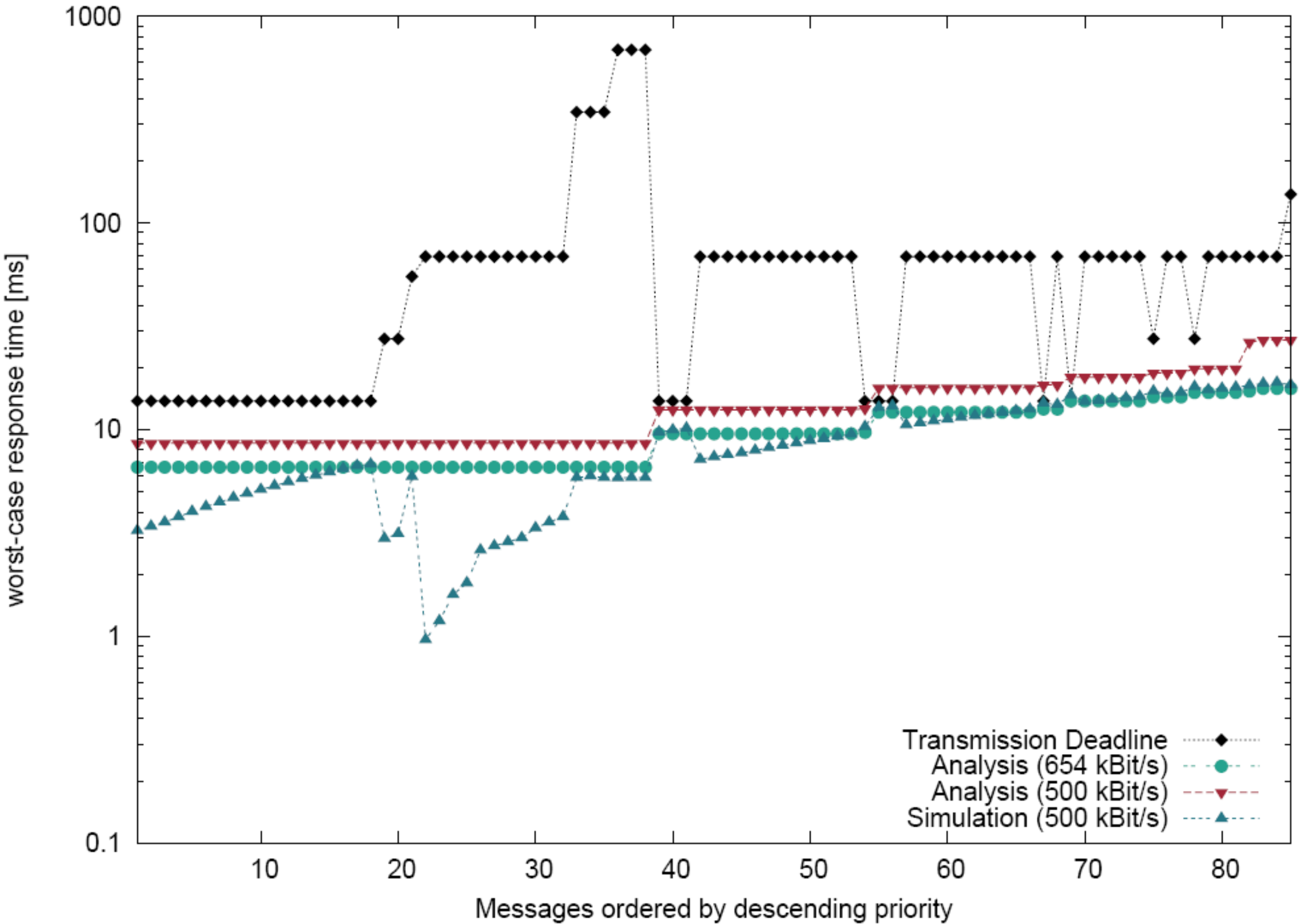
Expt 2: Two FIFO queues



Min bus speed
389 Kbit/s
(+40%)

Max bus Util.
60.1%

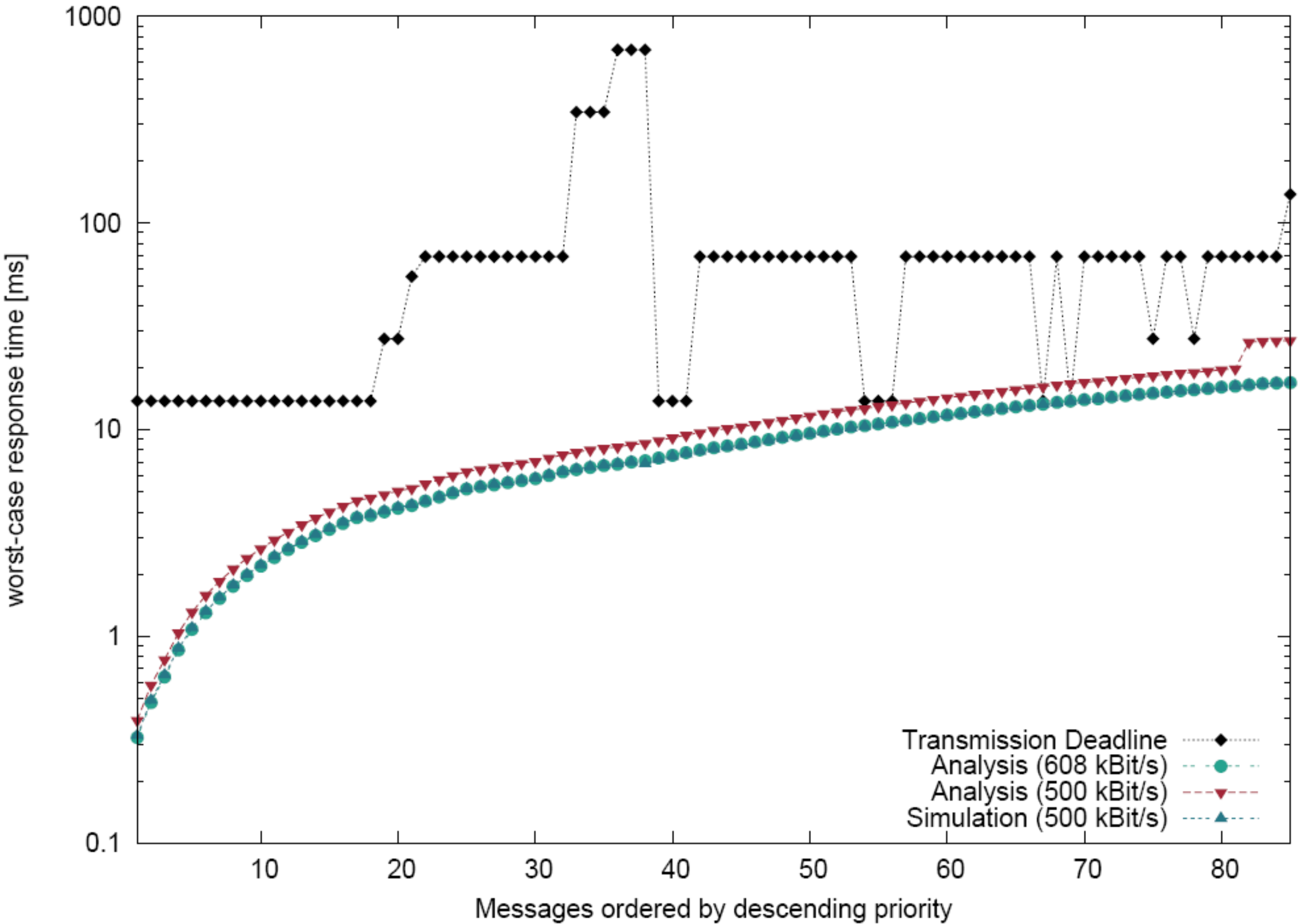
Expt 3: All FIFO queues



Min bus speed
654 Kbit/s
(+136%)

Max bus Util.
35.8%

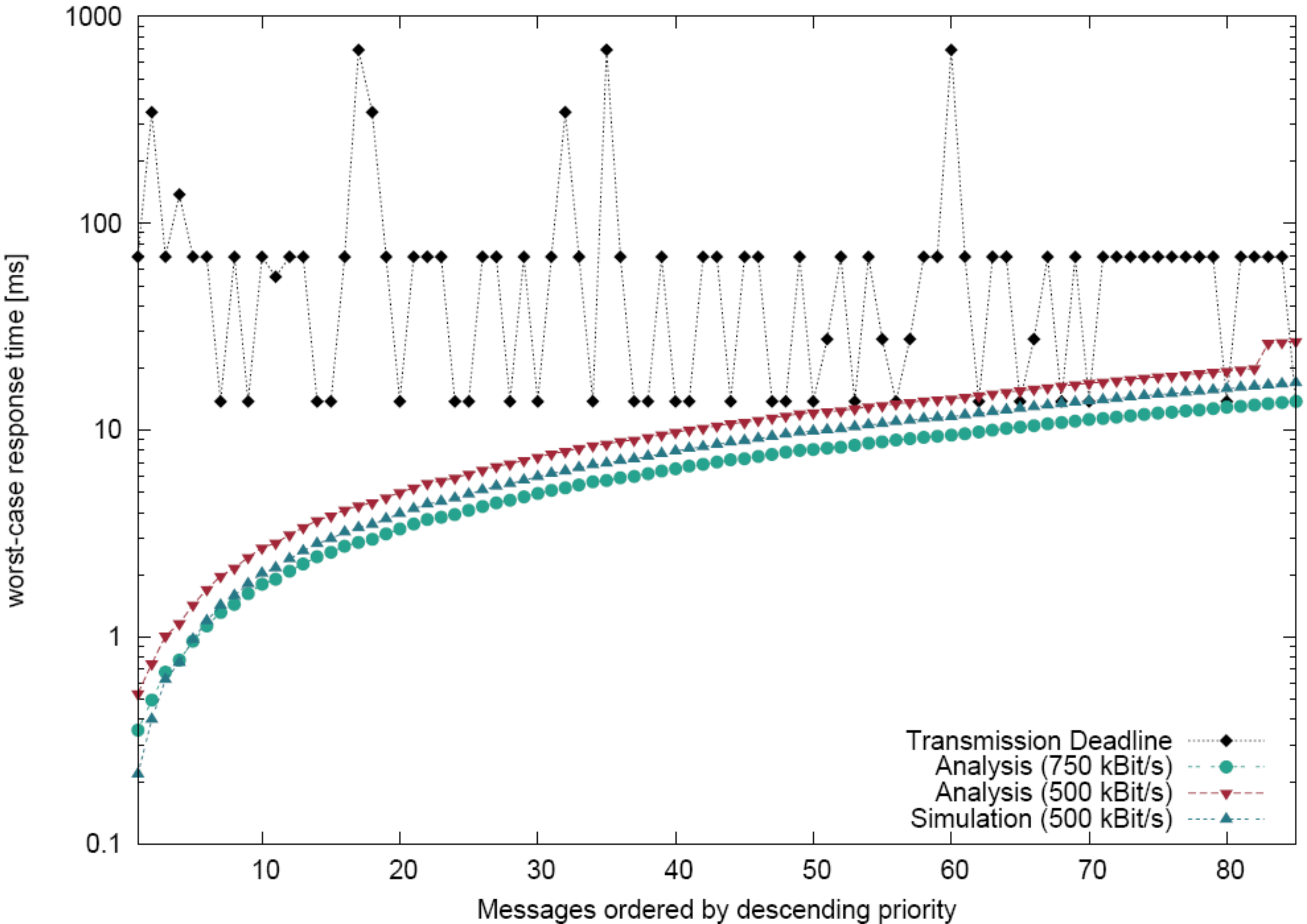
Expt 4: Priority queues: Priorities from all FIFO case



Min bus speed
608 Kbit/s
(+119%)

Max bus Util.
38.5%

Expt 5: Priority queues: Random priorities

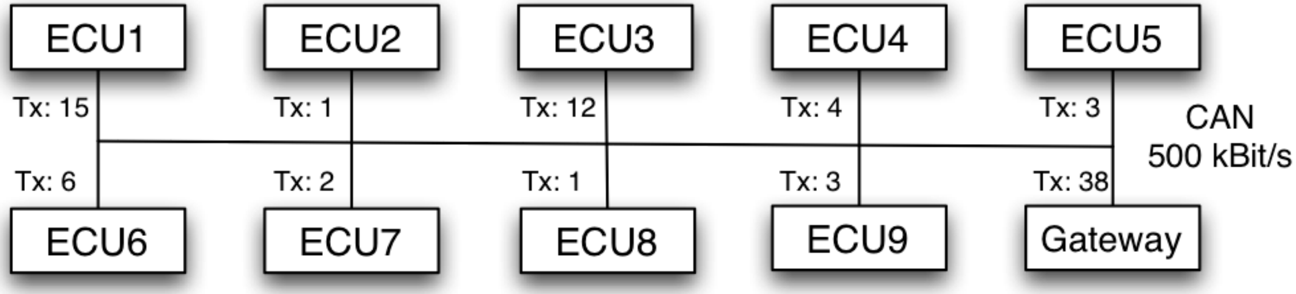


Min bus speed 732 Kbit/s (+164%)

Max bus Util. 32%

(average of 1000 random orderings)

Case Study: Summary



Expt.	Node type	Priority order	Min. bus speed	Max. bus Utilisation
1	All PQ	OPA	277 Kbit/s	84.5%
2	2 FQ, 8 PQ	OPA-FP/FIFO	389 Kbit/s (+40%)	60.1%
3	All FQ	OPA-FP/FIFO	654 Kbit/s (+136%)	35.8%
4	All PQ	From 3	608 Kbit/s (+119%)	38.5%
5	All PQ	Random	732 Kbit/s (+164%)	32.0%

Performance with FIFO queues

- Network performance with FIFO-queues
  - Significant reduction in performance – increased bus speed is required and a large decrease in max. bus utilisation (e.g. 80% down to 30%)
  - Mainly caused by unavoidable priority inversion, rather than pessimism in FIFO analysis

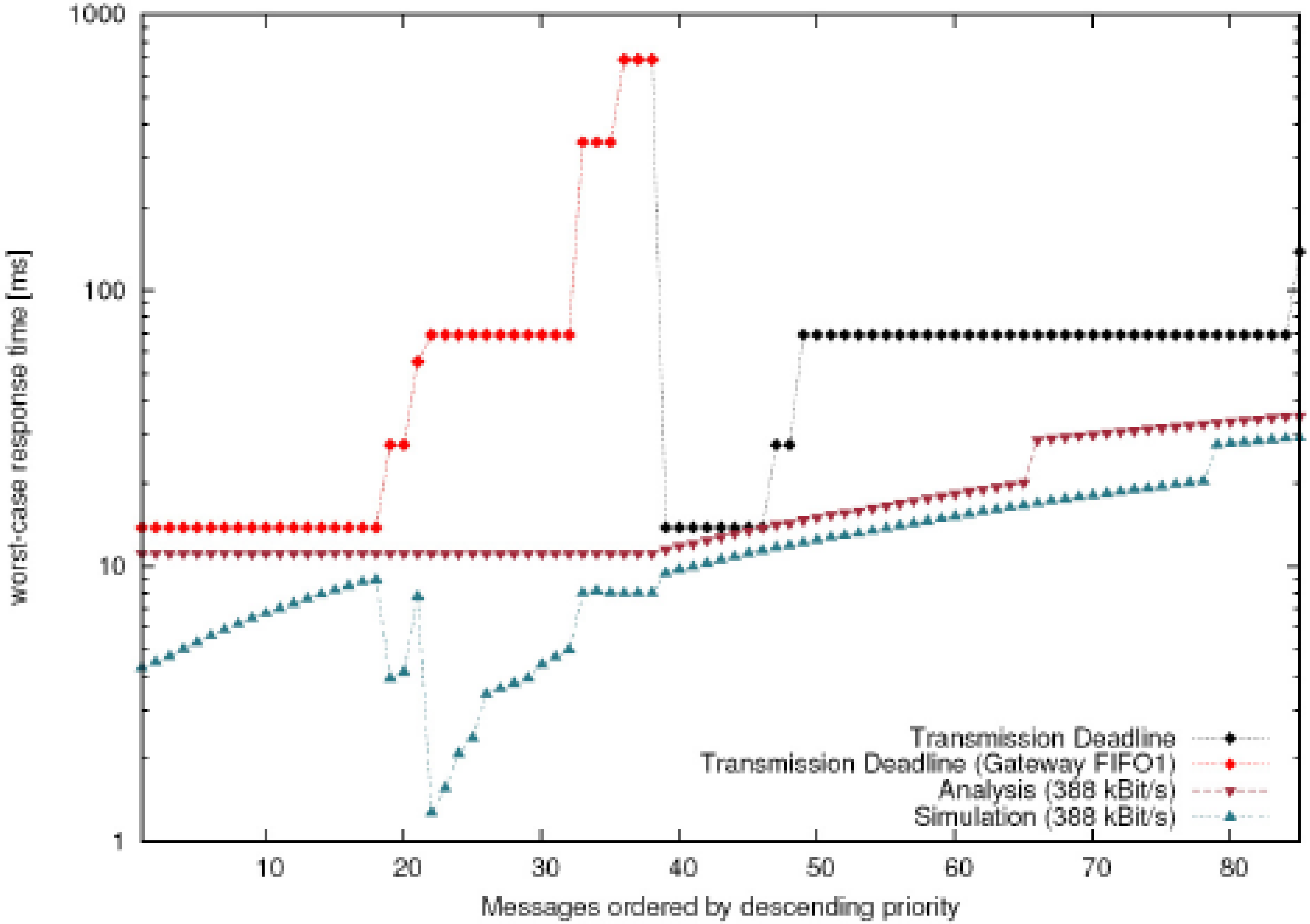
- Why are FIFO queues used
  - Make the device driver more efficient (less processor load)
  - Easier to implement
- But
  - local gain comes at a big cost – undermining priority based arbitration on CAN – inducing significant performance penalty

FIFO queues and CAN: Recommendations

- To obtain the best possible performance
  - Use an **appropriate priority ordering** (e.g. based on transmission deadlines, OPA, RPA)
  - **Avoid using FIFO queues** whenever possible
    they can cause significant performance degradation

- When might FIFO queues be acceptable?
  - Small number of messages in each FIFO, **and** those messages all have similar transmission deadlines – limits the amount of priority inversion
  - **Multiple** FIFO queues can be useful in gateway applications when there are not enough transmit buffers for one transmit buffer per message

Gateway with 1 FIFO queue



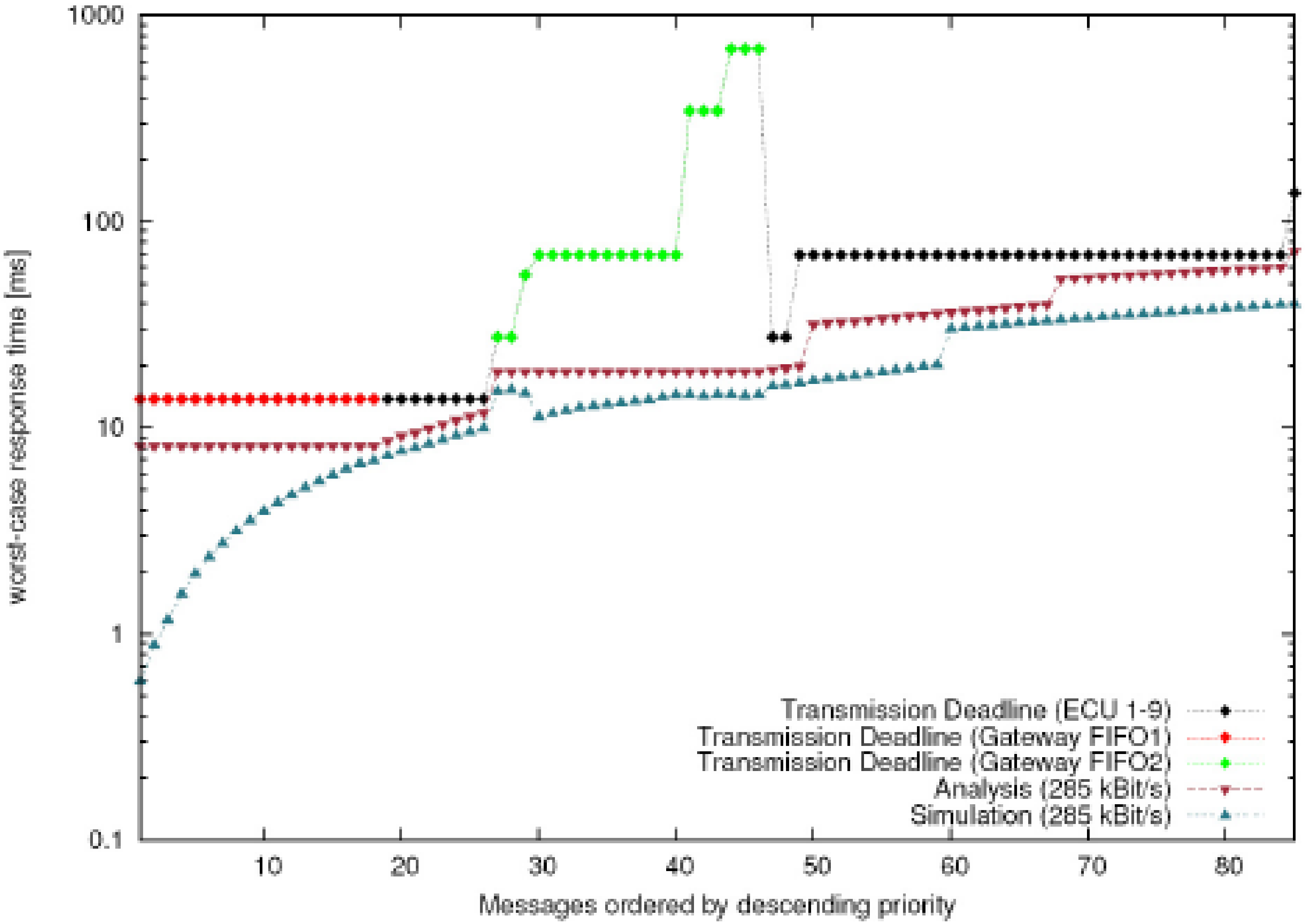
Min bus speed

388 Kbit/s

Max bus Util.

60.3%

Gateway with 2 FIFO queues



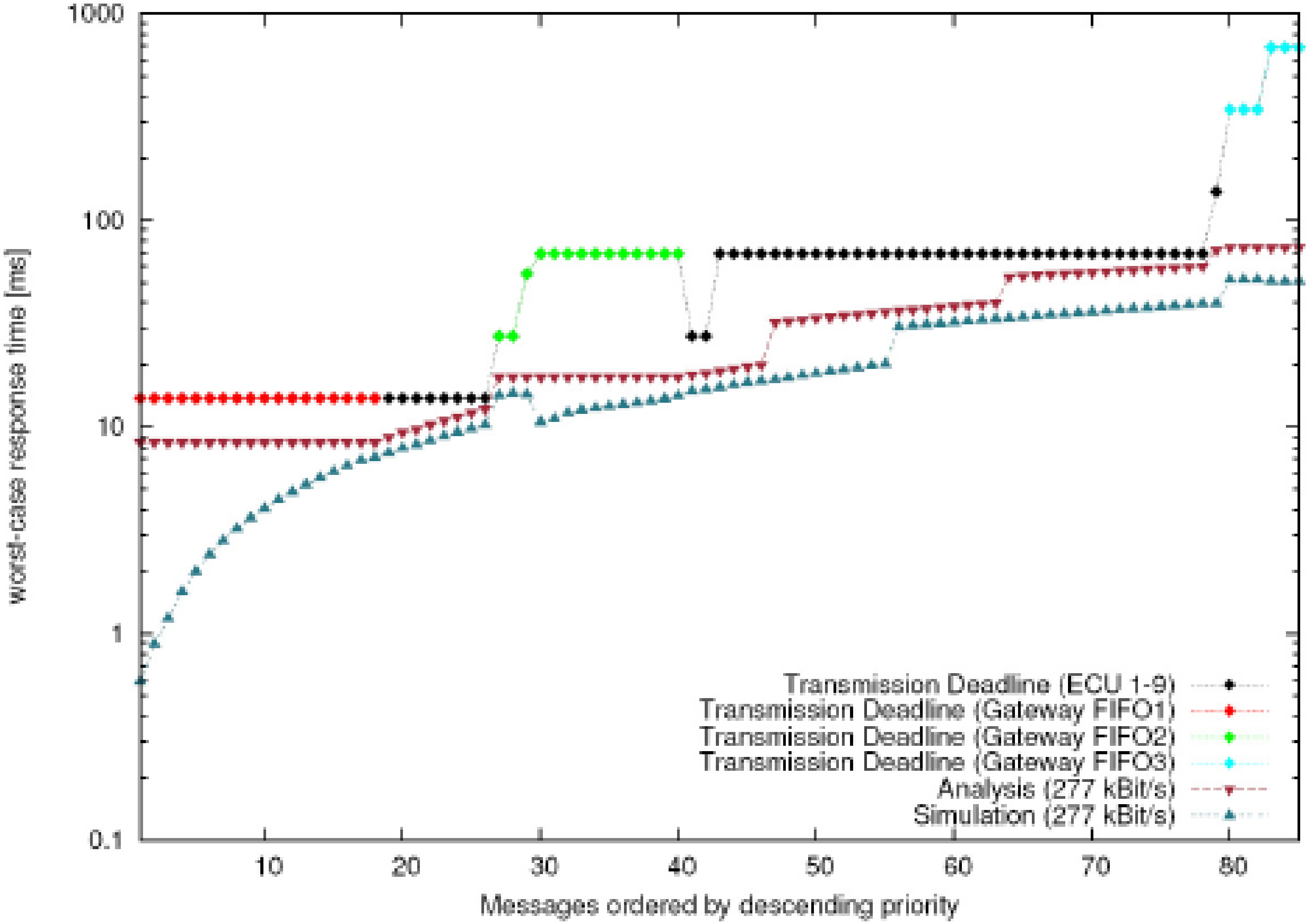
Min bus speed

285 Kbit/s

Max bus Util.

82.1%

Gateway with 3 FIFO queues



Min bus speed
277 Kbit/s
Max bus Util.
84.5%

Gave Performance equivalent to a priority queue in this case

Further Reading

REAL-TIME SYSTEMS

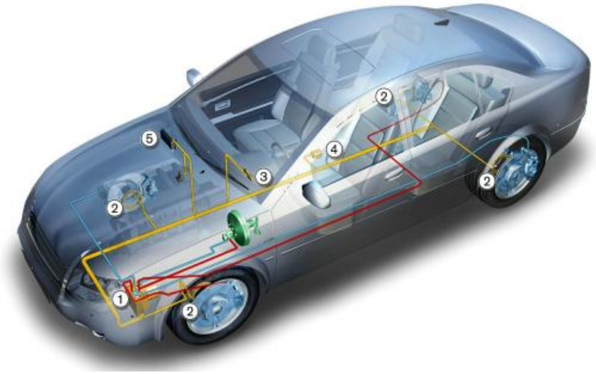
R.I. Davis, S. Kollmann, V. Pollex, F. Slomka, "Schedulability Analysis for Controller Area Network (CAN) with FIFO Queues Priority Queues and Gateways”. *Real-Time Systems,* Volume 49, Issue 1, pp. 73-116, Jan 2013. DOI: 10.1007/s11241-012-9167-8

- Analysis of Controller Area Networks with a mix of FIFO queues and priority queues
- Guidelines for configuring gateways using multiple FIFO queues

Success Stories
Fixed Priority Scheduling Theory

- Controller Area Network (CAN)
  - Response Time Analysis enables bus utilisation of up to ~70-80% compared to ~30% before
  - Involved in a start-up company NRTT that developed **Volcano** for Volvo in mid-1990s
  - Technology now owned and marketed by Mentor Graphics
  - Influenced CAN device driver HW design (MSCAN)
  - **Volcano** used in millions of cars: Volvo, Land Rover, Jaguar, Aston-Martin, Mazda, SAIC (China)

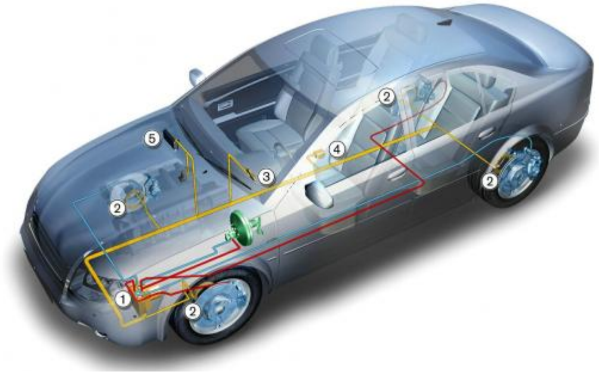
Mentor Graphics



Success Stories
Fixed Priority Scheduling Theory

- Highly effective systems can be built by adhering to the requirements of the theory
  - Priority queues in all device drivers
  - Robust priority assignment
  - OEM able to re-configure message priorities at end of production line and for upgrades

- Lack of knowledge and legacy requirements are problematic
  - FIFO queues in device drivers
  - Ad-hoc allocation and inflexibility in setting message IDs (priorities)
- Appropriate priority assignment still a (legacy) issue for industry



Success Stories
Fixed Priority Scheduling Theory

- Automotive RTOS
  - Involved in a start-up company NRTA (later LiveDevices) that developed an OSEK RTOS (1997-2003) called RTA-OSEK
  - RTOS was designed to comply with scheduling theory
  - Took advantage of FP scheduling and SRP for resource access to permit single stack operation saving memory (v. important for small microcontrollers)
  - RTOS analysable with minimal overheads
  - Supported by schedulability analysis tools
  - Company was sold in 2003 to ETAS (part of Bosch)

ETAS

- Since then
  - RTA-OS (Autosar extension), and RTA-OSEK deployments running at approx. 50 million ECUs per year...

Finally ... an interesting problem

- How to optimally assign CAN priorities when some message IDs are already fixed?
  - Common problem in industry when integrating legacy ECUs with fixed message IDs into a new system
  - New ECUs have flexibility with message IDs, legacy ones don’t

- Solution
  - If there are large enough gaps between the IDs of fixed messages we can use a variant of the Robust Priority Assignment algorithm to solve the problem optimally
- Open problem...
  - If the gaps are smaller, this does not always work - to the best of my knowledge no optimal solution is known that is also tractable (i.e. does not involve exploring all possible priority orderings)

Questions?

Further reading

R.I. Davis "A Review of Fixed Priority and EDF Scheduling for Hard Real-Time Uniprocessor Systems ”. *ACM SIGBED Review - Special Issue on the 3rd Embedded Operating Systems Workshop (Ewili 2013).* , Volume 11, Issue 1, pages 8-19, Feb 2014.

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*, Volume 35, Number 3, pp. 239-272, April 2007.

R.I. Davis and A. Burns "Robust priority assignment for messages on Controller Area Network (CAN)”. *Real-Time Systems*, Volume 41, Issue 2, pages 152-180, February 2009.

R.I. Davis, S. Kollmann, V. Pollex, F. Slomka, "Schedulability Analysis for Controller Area Network (CAN) with FIFO Queues Priority Queues and Gateways”. *Real-Time Systems,* Volume 49, Issue 1, pp. 73-116, Jan 2013.