Controller Area Network (CAN) Schedulability Analysis with FIFO queues

Robert Davis[1], Steffen Kollmann[2], Victor Pollex[2], Frank Slomka[2]

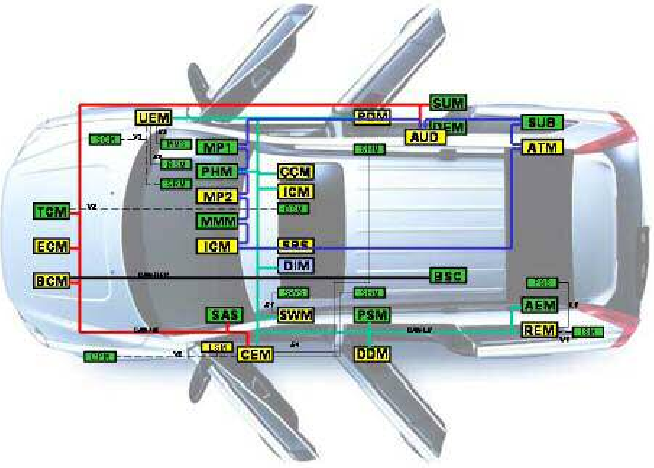
[1]Real-Time Systems Research Group, University of York
[2]Institute of Embedded Systems / Real-Time Systems Ulm University

Outline

- Controller Area Network (CAN)
  - Background
- Scheduling model
  - Recap analysis with priority queues
- Schedulability analysis with FIFO queues
- Optimal priority assignment
  - …and unavoidable priority inversion
- Automotive case study
  - Impact of FIFO queues
- Empirical investigation
- Summary and conclusions
- Recommendations

CAN Background

- Controller Area Network (CAN)
  - Simple, robust and efficient serial communications bus for in-vehicle networks
  - Developed originally by BOSCH in 1983, standardised in 1993 (ISO 11898)
  - Average family car now has approx 25-35 Electronic Control Units (ECUs) connected via CAN
  - CAN mandatory for cars and light trucks sold in USA since 2008 (On Board Diagnostics)
  - Today almost every new car sold in Europe uses CAN
  - Sales of microprocessors with CAN capability – approx 750 million in 2010.

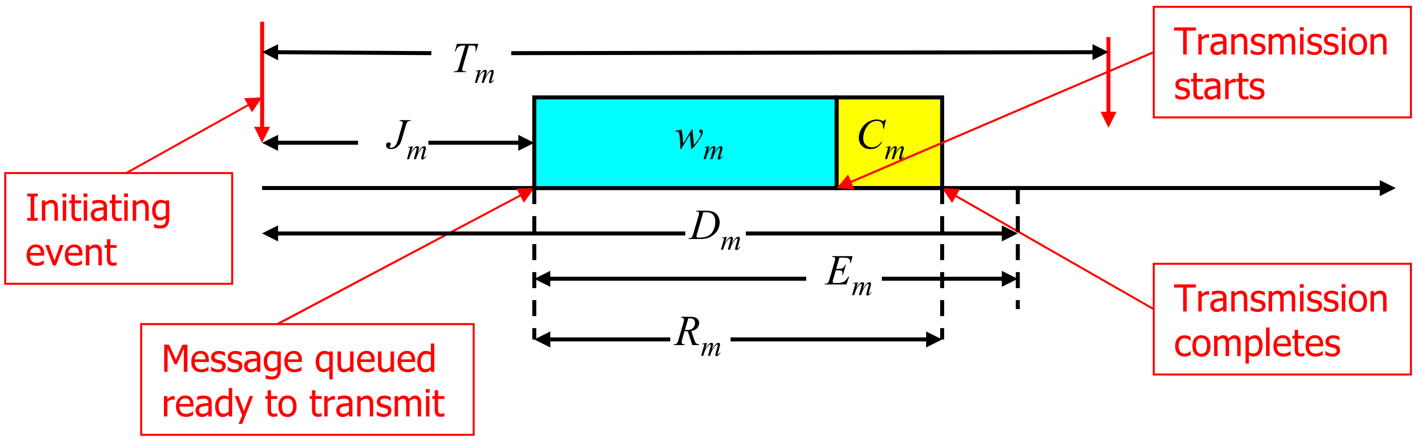


Scheduling model

- CAN Scheduling
  - Messages compete for access to the bus based on priority
  - With each node implementing a priority queue, network can be modelled as if there was a single global queue
  - Once a message starts transmission it cannot be pre-empted
  - Resembles single processor fixed priority non-pre-emptive scheduling

- Schedulability Analysis for CAN (assuming priority queues)
  - First derived by Tindell in 1994 [31, 32, 33] from earlier work on fixed priority pre-emptive scheduling
    - Calculates worst-case response times of all CAN messages
    - Used to check if all messages meet their deadlines in the worst-case
  - Significant flaws in the original analysis corrected by Davis et al. [11] in 2007.

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 $D_m \leq T_m$
  - 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: Priority queues only

- Sufficient schedulability test for priority queued messages [11]:
  - Blocking $B_m = \max_{k \in lp(m)} (C_k)$
  - Queuing delay $w_m^{n+1} = \max(B_m, C_m) + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k$
  - Response time $R_m = w_m + C_m$
- Message m schedulable if $R_m \leq E_m = D_m - J_m$

Motivation: FIFO queues

- Previous 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
  - 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 *"illusion that faster queue management improves system performance"* – de Natale 2008
  - Hardware support for FIFO queues in BXCAN and BECAN (ST7 and ST9 microcontrollers)

Scheduling model: FIFO queues

- Additional notation:
  - FIFO-group $M(m)$ the set of messages transmitted by the node that transmits message m
  - L_m lowest priority of any message in FIFO-group $M(m)$
  - C_m^{MIN} and C_m^{MAX} shortest and longest max. transmission times of messages in FIFO-group $M(m)$
  - C_m^{SUM} sum of the transmission times of messages in $M(m)$
  - E_m^{MIN} minimum transmission deadline of any message in $M(m)$
  - f_m buffering time – longest time that message m can take from being queued to being able to enter into priority based arbitration ($f_m = 0$ for priority queued messages)

Impact of FQ messages on PQ messages

- High priority FIFO-queued messages delayed from entering priority based arbitration can impact schedulability of priority queued messages
  - Such a message k effectively has additional jitter equal to the maximum buffering time f_k
  - Queuing delay

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

  - Response time $R_m = w_m + C_m$
  - Message m schedulable if $R_m \le E_m$

Schedulability analysis: FQ messages

- ***FIFO-symmetric*** analysis
  - Attributes the same upper bound response time to all messages in a FIFO queue.
- Make (pessimistic) worst-case assumptions:
  - Consider lowest priority of any message in the FIFO-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 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)

Schedulability analysis: FQ messages

- ***FIFO-symmetric*** analysis:
  - Queuing delay

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

  - Response time $R_m = w_m^{n+1} + C_m^{MIN}$
  - FIFO group schedulable if $R_m \le E_m^{MIN}$

Schedulability analysis: FQ messages

- Buffering times (FIFO):
  - Upper bound given by

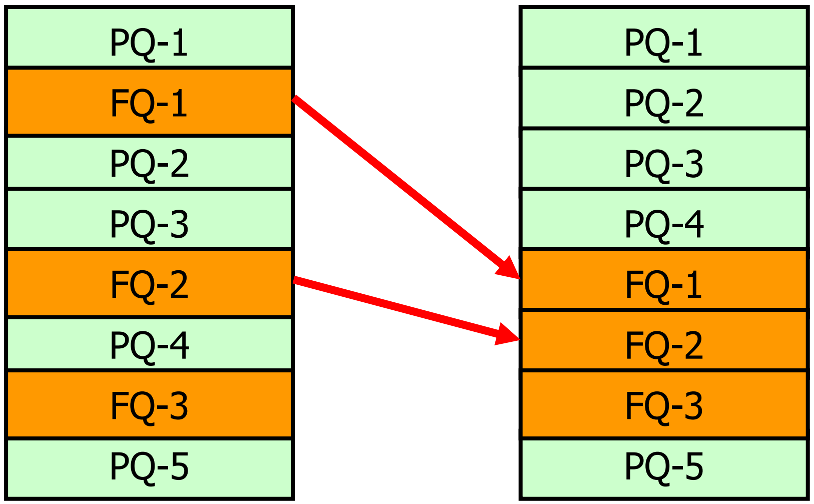
    $$f_m = R_m - C_m^{MIN}$$

  - Problem – if priorities of FIFO groups are interleaved, then buffering time of one message can depend on the response time of another message and vice-versa
  - Resolved by noting that buffering times are monotonically non-decreasing w.r.t. response times and vice-versa

```
repeat = true
initialise all f_k = 0
while(repeat){
    repeat = false
    for each priority m, highest first{
        if (m is FIFO-queued){
            calc R_m for FIFO-queued message
            if( R_m > E_m^MIN ) {
                return unschedulable
            }
            if( f_m != w_m ){
                f_m = w_m
                repeat = true;
            }
        }
        else {
            calc R_m for priority-queued message
            if( R_m > E_m ) {
                return unschedulable
            }
        }
    }
}
return schedulable
```

FIFO-adjacent priority ordering

- ***FIFO-adjacent*** priority ordering:
  - Messages within a FIFO-group have adjacent priorities – no interleaving with other messages
  - Optimal partial ordering: If a priority ordering Q exists that is schedulable according to the FIFO-symmetric schedulability test, then a schedulable FIFO-adjacent priority ordering also exists
  - Regardless of the priority ordering of PQ-messages, all messages sharing a FIFO queue should have adjacent priorities (but not necessarily consecutive values)



FIFO-adjacent priorities

- With ***FIFO-adjacent*** priorities:
  - No need to account for buffering time so $f_m = 0$ for all FIFO-queued messages
  - This is because if a FIFO-queued message *m* is of higher priority than message *k*, then crucially, so are <u>all</u> of the other messages that share the FIFO queue with *m*, hence all contribute to the queuing delay of message *k*, and the order in which they are actually sent on the bus is irrelevant
  - Setting $f_m = 0$ for all messages:
    - simplifies the analysis (no repeats of the while loop – just calculate the message response times)
    - Removes a significant amount of pessimism

Optimal priority assignment

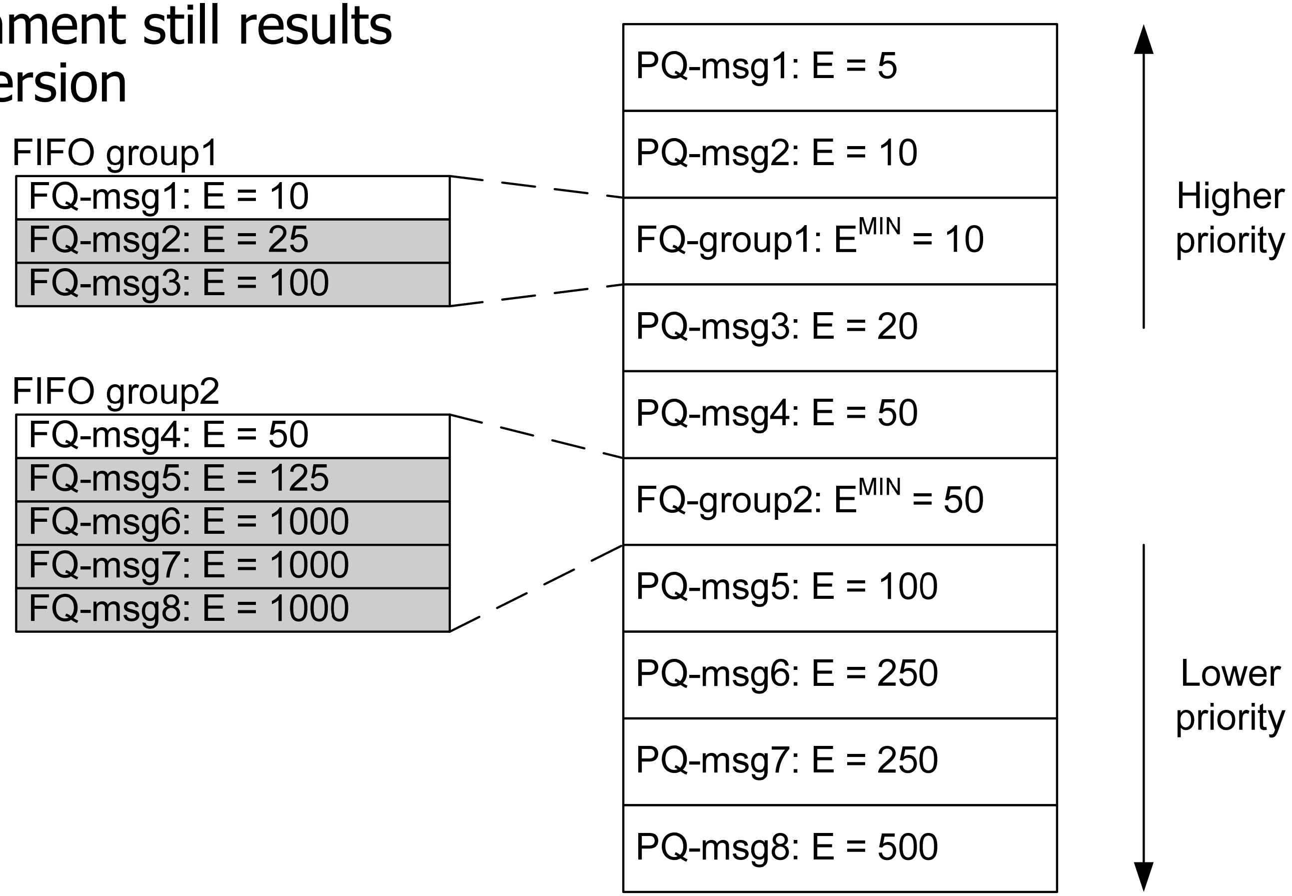
- OPA-FP/FIFO algorithm
  - Based on Audlsey's greedy Optimal Priority Assignment (OPA) algorithm
  - Optimal for networks with a mix of priority-queued and FIFO-queued messages w.r.t. the FIFO-symmetric schedulability test

```
for each priority band k, lowest first
{
    for each message msg in the initial list {
        if msg is schedulable in priority band k according to
        schedulability test S with all unassigned priority-
        queued messages / other FIFO groups assumed to be
        in higher priority bands {
            assign msg to priority band k
            if msg is part of a FIFO group {
                assign all other messages in the FIFO group
                to adjacent priorities within priority band k
            }
            break (continue outer loop)
        }
    }
    return unschedulable
}
return schedulable
```

- Transmission deadline monotonic priority ordering
  - Optimal when all messages have the same max. transmission time
  - Use E_m^{MIN} to represent the transmission deadline of all messages in a FIFO- group (and adjacent priorities within the group)

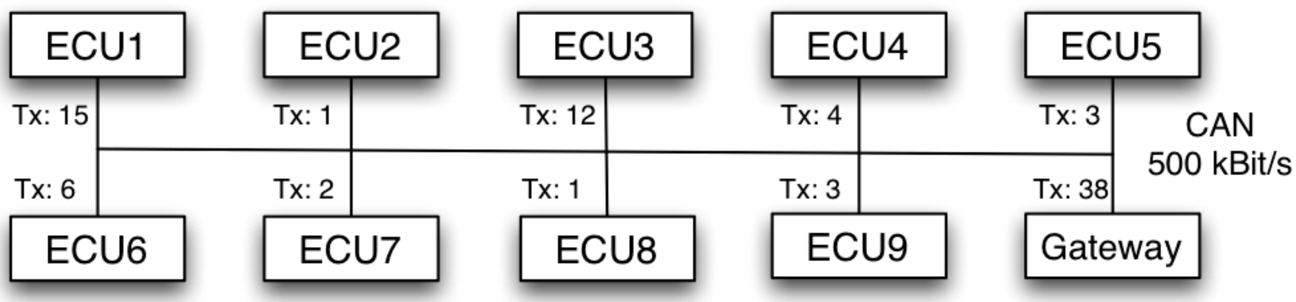
Priority inversion

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



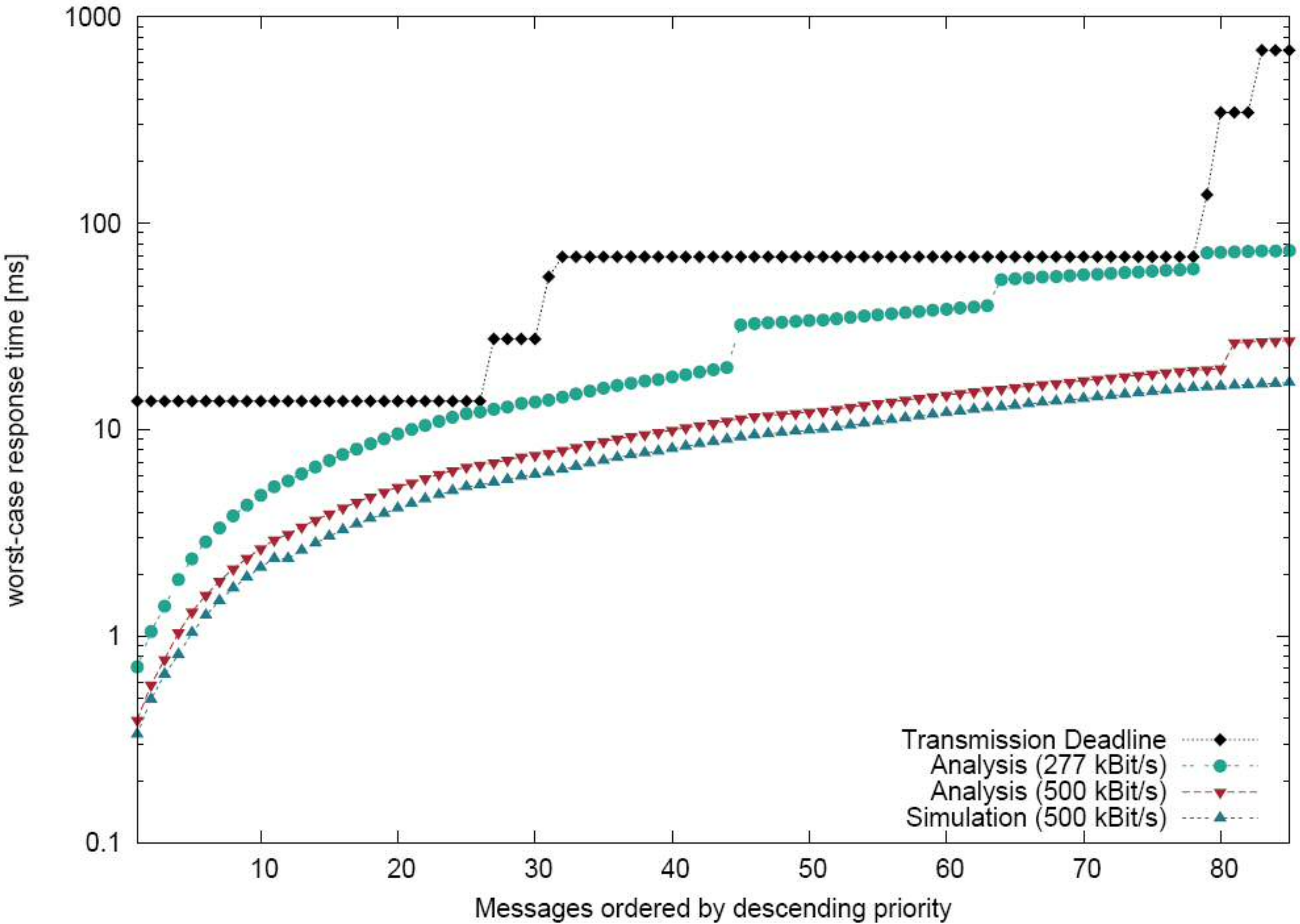
Case Study: Automotive

- 10 ECUs, 85 messages



- Experiments
  - *Expt. 1*: All ECUs used priority queues
  - *Expt. 2*: ECU3 (12 msgs) and ECU6 (6 msgs) used FIFO queues
  - *Expt. 3*: All ECUs used FIFO queues
  - *Expt. 4*: All ECUs used priority queues, priority ordering from Expt 3
  - *Expt. 5*: All ECUs used priority queues, random priority ordering

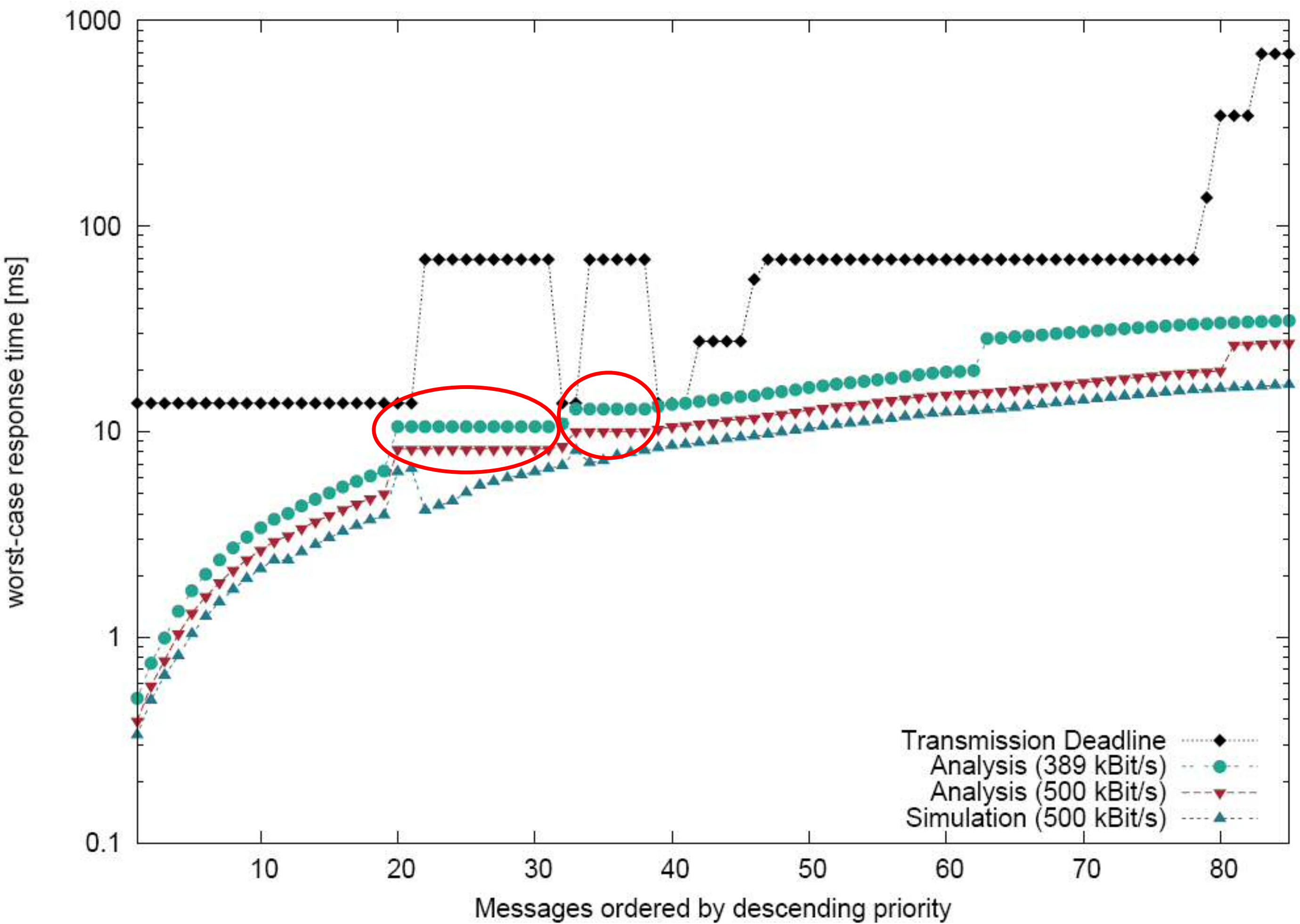
Expt 1: All priority queues



Min bus speed
277 Kbit/s

Max bus Util.
84.5%

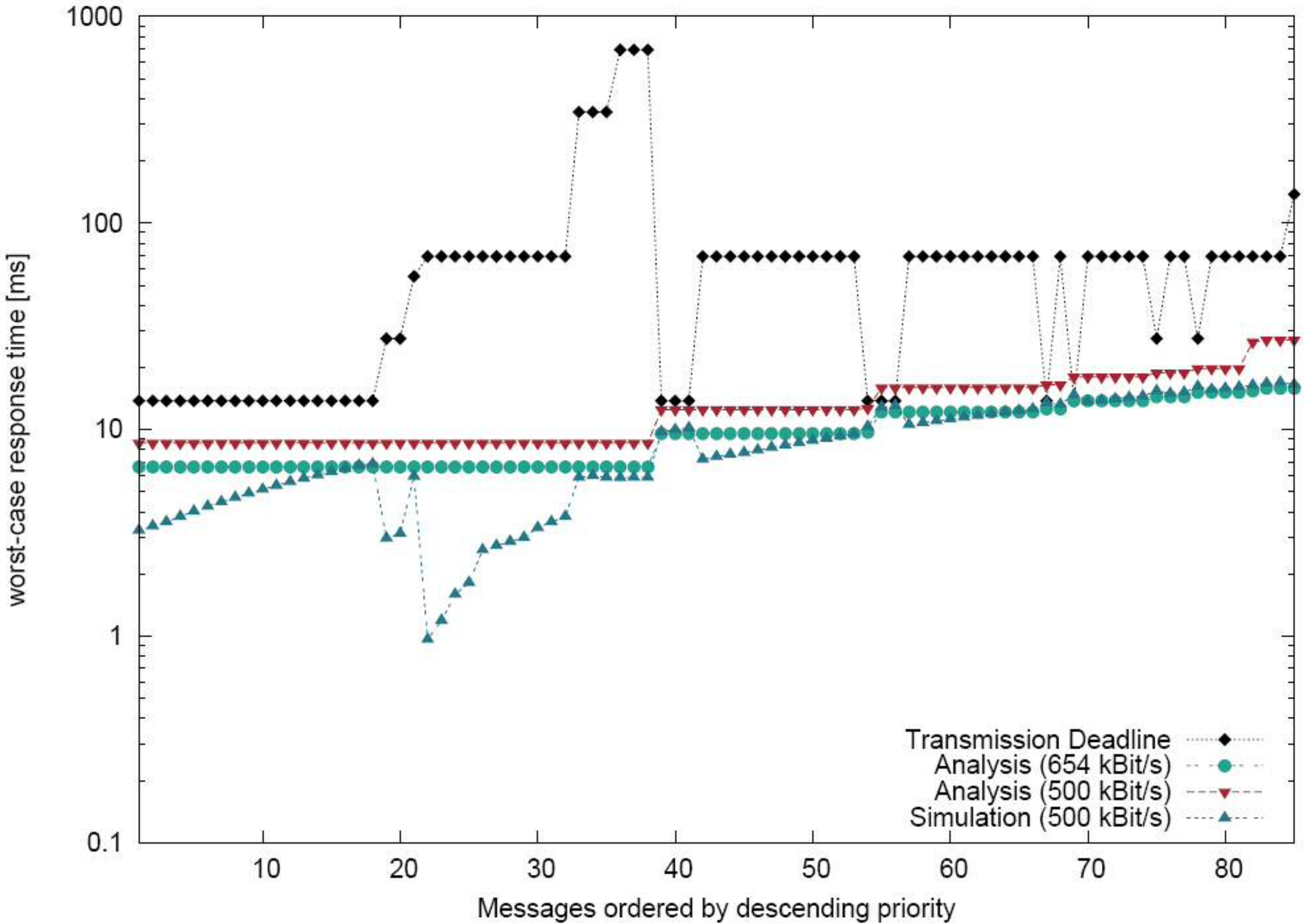
Expt 2: Two FIFO queues



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

Max bus Util.
60.1%

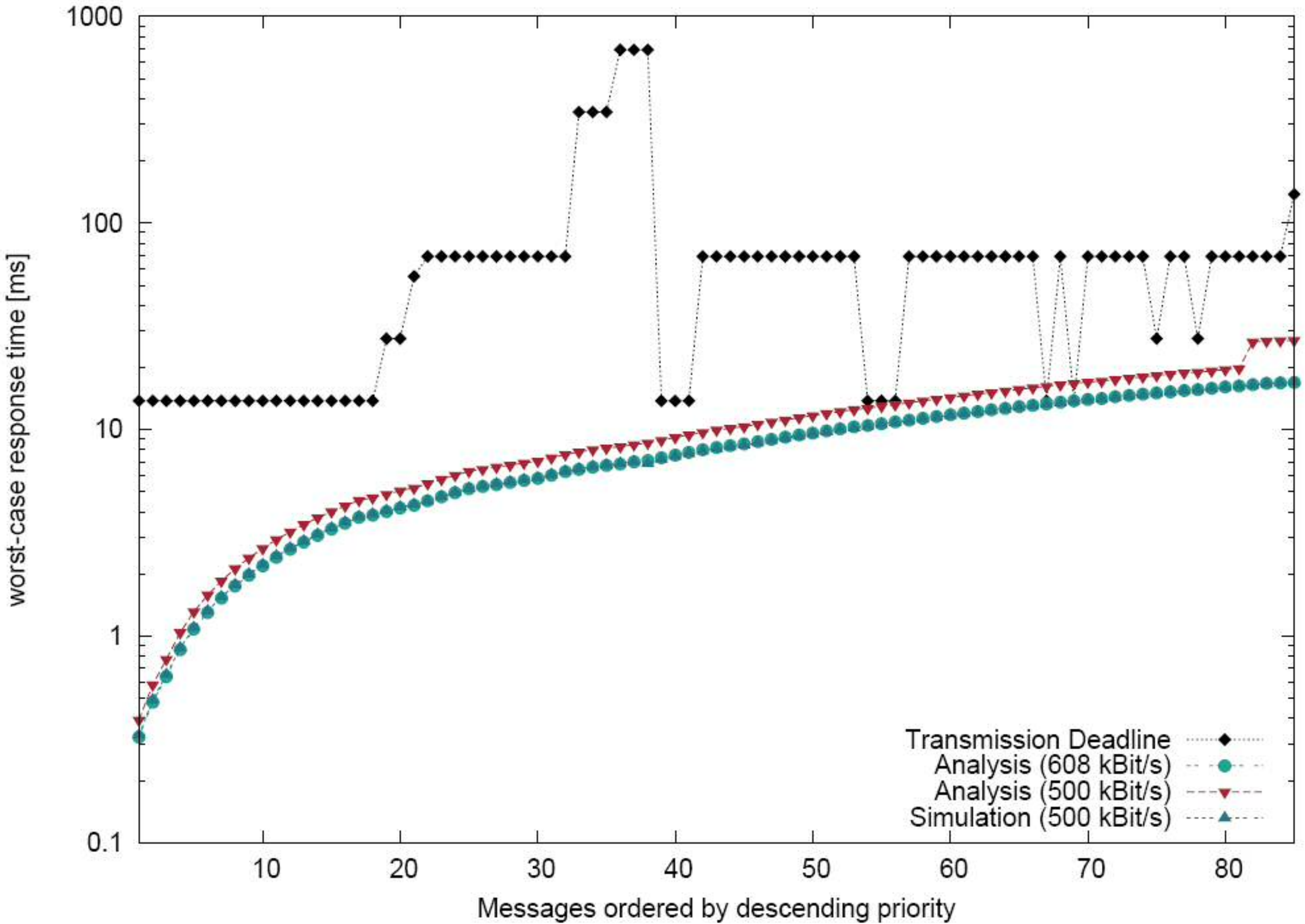
Expt 3: All FIFO queues



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

Max bus Util.
35.8%

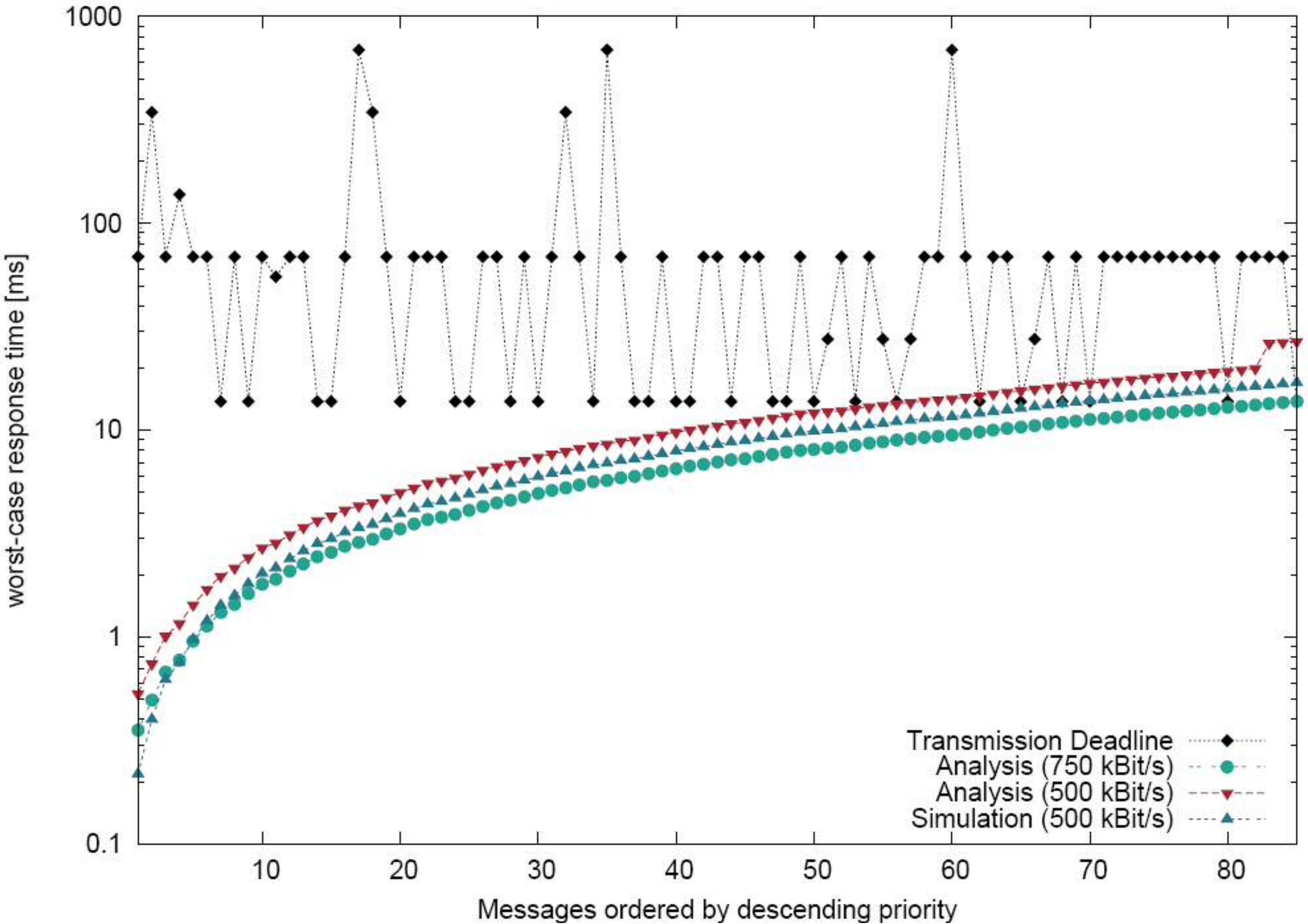
Expt 4: Priority queues: priorities from all FIFO case



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

Max bus Util.
38.5%

Expt 5: Priority queues: random priorities

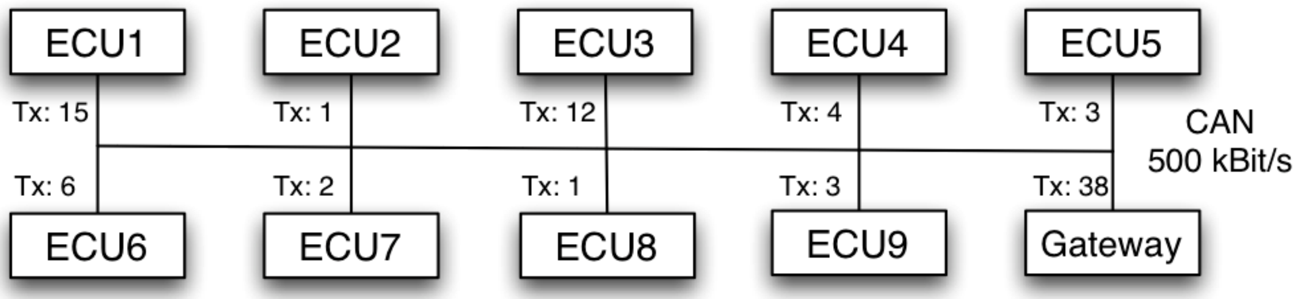


Min bus speed
732 Kbit/s
(+164%)

Max bus Util.
32%

(average of 1000 random orderings)

Case Study: Summary

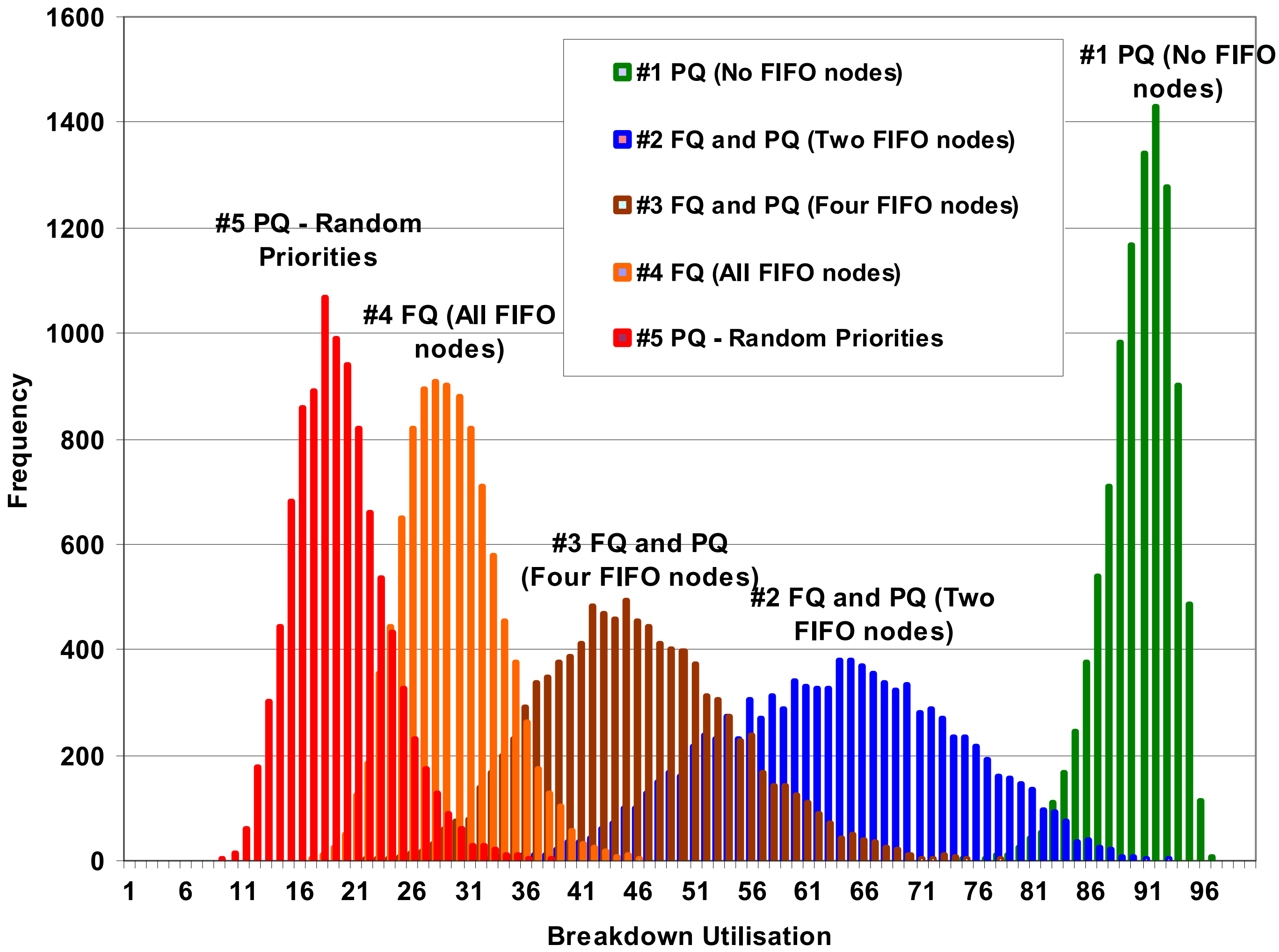


Expt.	Node type	Priority order	Min. bus speed	Max. bus Utilisation
1	All PQ	OPA	277 Kbit/s	84.5%
2	2 FQ, 8 PQ	OPA-FP/FIFO	389 Kbit/s (+40%)	60.1%
3	All FQ	OPA-FP/FIFO	654 Kbit/s (+136%)	35.8%
4	All PQ	From 3	608 Kbit/s (+119%)	38.5%
5	All PQ	Random	731 Kbit/s (+164%)	32.0%

Empirical evaluation

- Examined 10,000 randomly generated sets of messages:
  - 80 messages in each set, 8 data bytes per message
  - 8 nodes on the network
  - Random allocation of messages to nodes
  - Log-uniform distribution of message periods 10ms – 1000ms
  - Message deadline = period
  - Jitter (uniform distribution in range 2.5 – 5ms)
  - 11-bit identifiers
- Configurations
  - *Config. 1*: All PQ nodes - TDMPO
  - *Config. 2*: Two FQ nodes – TDMPO-FP/FIFO
  - *Config. 3*: Four FQ nodes – TDMPO-FP/FIFO
  - *Config. 4*: All FQ nodes – TDMPO-FP/FIFO
  - *Config. 5*: All PQ nodes – random priorities

Empirical results



Evaluation

- Empirical evaluation of 10,000 message sets
  - 8 nodes, 80 messages, 8 data bytes per message
  - periods 10-1000ms (log uniform distribution)
  - jitter 2.5-5ms (uniform distribution)

Config.	Node type	Priority order	Average Max. bus utilisation
1	All PQ	TDMPO	**89.5%**
2	2 FQ, 6 PQ	TDMPO-FP/FIFO	62.7%
3	4 FQ, 4PQ	TDMPO-FP/FIFO	44.9%
4	All FQ	TDMPO-FP/FIFO	**28.4%**
5	All PQ	Random	18.4%

Summary and Conclusions

- Introduced sufficient schedulability test for CAN networks with a mix of nodes using FIFO and priority queues
  - *FIFO-symmetric* analysis – attribute same upper bound response time to all messages in a FIFO queue.
  - With FIFO-symmetric analysis, *FIFO adjacent priority ordering* is optimal within each FIFO group
  - Modified OPA algorithm to provide optimal priority ordering (w.r.t. our analysis) for a mix of FIFO queued and priority queued messages
  - Nevertheless priority inversion is unavoidable with FIFO queues

Summary and Conclusions

- Examined performance of FIFO-queues / analysis via case study and empirical evaluation
  - 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 cost – undermining priority based arbitration on CAN – significant performance penalty

Recommendations

- To obtain the best possible performance
  - Use an **appropriate priority ordering** (e.g. based on transmission deadlines)
  - **Avoid using FIFO queues** whenever possible

- FIFO queues can cause significant performance degradation
  - When there are many messages in a FIFO, with a range of transmission deadlines that interleave with those of other messages on the network – result is significant priority version

Recommendations

- When FIFO queues might just be acceptable
  - Small number of messages in each FIFO, ***and*** those messages all have similar transmission deadlines – limits the amount of priority inversion
  - Multiple small FIFO queues could be useful in gateway applications when there are not enough transmit buffers for one transmit buffer per message
  - Schedulability tests and priority assignment techniques now available to explore this

- Future work
  - Non-abortable transmission buffers, FIFO queues, and message / priority assignment to FIFO queues in gateway applications

Questions?