Optimal Static Scheduling of CAN Bus Communication

Vojtěch Michal
Thursday, 9:15 a.m.
Cybernetics and Robotics
michavo3@fel.cvut.cz

I. Assignment

A. Motivation

The Controller Area Network (CAN) bus is a time-proven differential digital communication network widely spread in the automotive industry. It has been designed and further refined, emphasising low latency and safety, offering reliability and error detection via CRC. The physical layer of its high-speed variant makes use of two wires that are either actively driven apart by a transmitting electronic control unit (ECU) to represent dominant logical zero, or passively pulled together by a pair of bus terminating resistors to represent a recessive logical one. The CAN bus naturally employs a wired logic AND, i.e. the bus is the recessive state only when no ECU is transmitting a dominant level, whereas the dominant state is attained whenever any ECU is transmitting a dominant zero.

Bits of information are carried by messages (or frames). Every message has a globally unique numeric identifier – either 11 or 29 bits long – that simultaneously acts as its priority. During the initial phase of every transmission (most notably when transmitting the ID), each ECU simultaneously actively monitors the state of the bus. Should it observe a dominant state whilst transmitting the recessive level itself, it understands there is another ECU with a lower-ID (higher-priority) message, accepts the loss of arbitration and yields the right to transmit to the message with the lowest ID. This ingenious medium access control algorithm is known as Carrier Sense Multiple Access / Collision Resolution (CSMA/CR).

Although this principle guarantees a smooth resolution of all conflicts, it may cause delays of messages with lower priority in times of high bus load. Even without significant delays, the transmission period of low-priority messages experiences significant jitter as there is no prior guarantee of maximal transmission delay. Furthermore, without centralized synchronization and scheduling, the communication is liable to burst transmissions, where the bus transitions between longer periods of idle state followed by short periods of full bandwidth utilization. This report aims to investigate possible solutions to the problem of static communication scheduling with the goal of offering several optimization approaches, namely to spread the bus utilization evenly and guarantee maximal temporal consistency of the whole scheduled system. As the research is motivated by the very real problem of CAN communication scheduling for a Formula Student Electric built by the team *eForce FEE Prague Formula*, emphasis is laid on practical implementation aspects.

B. Properties of CAN communication

A vehicle contains multiple ECUs connected by independent CAN buses and competing for access to the shared network, attempting to transmit a predefined set of distinct messages. As far as scheduling needs are concerned, three types of messages can be distinguished:

1) **High-priority messages** with low transmission periods and high demands on deadline satisfaction. These are typically driver inputs (pedals and steering wheel), state of the high voltage accumulator, inertial measurements or commands for the electric powertrain.
2) **Low-priority messages** with large periods, predominantly carrying auxiliary data. These include diagnostic messages such as information about the ECU firmware version, the current ambient temperature or lighting.
3) **Aperiodic messages** bound to an occurrence of trigger event rather than to a fixed period, such as the request to change parameters of vehicle dynamics controllers. The release time of transmissions is completely random.

The distinction between high and low-priority messages is rather arbitrary and is made only on the basis of educated guess of how important the inclusion of said message in the static schedule is.

Further details of CAN physical and link layers, such as the bit stuffing algorithm, are not crucial for the rest of this work. An interested reader can learn more in the CAN standard itself or the overview presented by [3]. After the application of all transformations, such as the channel coding, the length in bits l_m of a standard CAN message m with $n \leq 8$ data bytes is bounded from above by

$$l_m \leq 8n + 44 + \lfloor \frac{34 + 8n - 1}{4} \rfloor, \qquad l_m \leq 132 \text{ for } n = 8. \tag{1}$$

Since there are no preemptions (message transmission cannot be interrupted after a victorious arbitration), one has to reserve a sufficiently large window from time to time to account for messages outside of the static schedule. The most prevalent message periods are 5, 10, 20, 50, 100 and 200 ms.

The typical solution to static scheduling of periodic tasks is to divide time into periodically repeating intervals – usually called *hyper-periods* – representing the lowest common multiple of periods of all scheduled tasks. Each hyper-period is further divided into intervals called *time quanta* $Q \in \mathbb{N}$ representing the shortest duration (expressed using a number of bit times) manipulated by the scheduling algorithm as a single object to account for the real-world limitation of the ECU system clock resolution. Let $H \in \mathbb{N}$ denote the hyper-period length in quanta.

C. Problem Statement

Given a bus baud rate in bits per second b, the length of time quantum Q, the length of hyper-period H and the set of messages $M = \{m_1, \cdots\}$, each with fixed period in quanta T_m, length in bits l_m and numeric identifier i_m, the initial problem is to schedule the transmission of all messages whilst respecting the rules of CAN bus communication (i.e. to find any feasible schedule). Only the integral message identifier (priority) can be relied on to achieve finer scheduling within each window.

The initial problem is later extended to search for an optimal schedule minimizing either the maximal jitter of TX periods of all messages, the peak bus utilization or the maximal number of messages transmitted by a single ECU during a single time quantum. The optimization problem will be solved using the framework of ILP due to its expressiveness.

II. Related works

An interesting approach to CAN communication scheduling is presented in [4], where authors use the natural process of CAN message arbitration to implement the Earliest Deadline First (EDF) algorithm directly in hardware. A system-wide time window called an *epoch* is defined with a clear resynchronization event at the beginning. Each message ID is then constructed by concatenating the code for its deadline relative to the start of the epoch and its unique number. This approach ensures transmission of all real-time messages in time before respective deadlines but struggles when scheduling distinct messages with very different periods. An improvement is introduced through the logarithmic encoding of the relative deadline (permitting the designer to express a wider range of relative deadlines). However, this does not lend itself to a simple extension to handle release times and significantly different message periods. If the disadvantages mentioned are not an issue for a concrete application, this approach has favourable performance characteristics due to the reduction of CPU load, as demonstrated in [5].

The constraint programming approach to static communication scheduling is taken in [6], where the authors deal with static scheduling for an automotive FlexRay network. In contrast to the CAN communication analyzed in this work, FlexRay supports static time scheduling by design in the form of time-triggered static segments, where individual messages are assigned to individual slots in advance. Authors aimed to minimize the number of used slots within each static segment in order to facilitate the future extensibility of the system. However, such an objective function does not lend itself to simple utilization in CAN scheduling. It was nevertheless an inspiration for one of the utilized optimization criteria to minimize the maximal number of messages transmitted by each ECU during a single quantum.

III. Problem solution

A. Definitions and observations

During each hyper-period H, each message $m \in M$ will be transmitted $N_m = H/T_m$ times, where T_m is the transmission period of message m expressed in time quanta. Let $J_{m,i}$ for $i \in \{0, 1, \cdots, N_m - 1\}$ denote the i-th transmission of message m within the hyper-period. Define a binary decision variable

$$x_{m,i,j} = \begin{cases} 1 & \text{iff } J_{m,i} \text{ is scheduled for transmission during the time quantum } j \in \{0, 1, \cdots, H-1\}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Using these definitions, one can attempt to express the start time of transmission $J_{m,i}$ as

$$t_{m,i} = \sum_{j=1}^{H} j \cdot x_{m,i,j}, \tag{3}$$

this equality, however, holds exactly only for the transmission of the highest-priority message within each time quantum, whereas all other transmissions are incrementally delayed. To improve the scheduling algorithm and achieve more accurate control over the TX period jitter, the exact start time of transmission $J_{m,i}$ within the time quantum j is defined as

$$\bar{t}_{m,i,j} = j \cdot x_{m,i,j} + \sum_{\bar{m} \in M_m} \sum_{k=0}^{N_{\bar{m}}-1} l_{\bar{m}} \cdot x_{\bar{m},k,j}, \tag{4}$$

where $M_m = \{\bar{m} \in M \mid i_{\bar{m}} < i_m\}$ is the set of all messages with higher priority than m.

B. Constraints

In order for the set of variables to describe a valid communication schedule, the values of individual decision variables have to be related by a list of constraints.

1) Each transmission shall be scheduled for exactly one time quantum:

$$\sum_{j=1}^{H} x_{m,i,j} = 1 \qquad \text{for every message } m \in M \text{ and its } i\text{-th transmission.} \tag{5}$$

Note that the natural constraint

$$\sum_{i=0}^{N_m-1} x_{m,i,j} \leq 1 \qquad \text{for every message } m \in M \text{ and quantum } j \tag{6}$$

prohibiting scheduling of multiple transmissions $J_{m,i}$ of the same message m for a single time quantum j is not necessary as section III-B2 presents an even stronger condition.

2) Coarse upper bound on transmission period jitter: Using the simpler definition of transmission start time (3), one can construct the following constraint: for every message m and for every pair of consecutive transmissions $J_{m,i}$ and $J_{m,i+1}$, it must hold that the transmission period jitter (i.e. the absolute difference between the ideal interval T_m and real interval $t_{m,i+1} - t_{m,i}$) must be bounded from above. The precise formulation in the language of ILP is

$$-J \leq t_{m,i+1} - t_{m,i} - T_m \leq J \qquad \text{for every message } m \text{ and } i \in \{0, 1, \cdots, N_m - 2\}, \tag{7}$$

where J is the upper bound on both the positive jitter (i.e. $t_{m,i+1} > t_{m,i} + T_m$) as well as negative jitter ($t_{m,i+1} < t_{m,i} + T_m$) expressed in time quanta.

Because of the repetition of static schedule during consecutive hyper-periods, one also has to constrain the start time of the last transmission J_{m,N_m-1} with respect to the first transmission $J_{m,0}$ for every message m, i.e. require that

$$-J \leq H + t_{m,0} - t_{m,N_m-1} - T_m \leq J \qquad \text{for every message } m. \tag{8}$$

This constraint is further improved in section III-B5.

3) Upper bound on the bus utilization: In order to facilitate the transmission of aperiodic or low-priority messages ignored by the static schedule, a sufficient amount of time has to be reserved in each quantum. This can be achieved by restricting the combined length of messages scheduled for individual time quanta. The time taken by i-th transmission of message m during time quantum j is $l_m \cdot x_{m,i,j}$. Since there is – in all generality – no prior knowledge about what transmission of what message will take place during any given time quantum j, the summation has to capture all options, i.e.

$$\sum_{m \in M} \sum_{i=0}^{N_m-1} l_m \cdot x_{m,i,j} \leq T, \tag{9}$$

where $T = Q - R$ is the maximal number of bits used by static schedule in each time quantum to guarantee at least R bits are always reserved for sporadic or event-triggered messages.

4) Minimize memory footprint: The low-level implementation of static scheduling in the ECU firmware will use a simple array `msg_id schedule[H][F]`, where `schedule[j][k]` is the identifier of k-th message scheduled for transmission by the given ECU during time quantum j. The size of the second array dimension $F \in \mathbb{N}$ is the maximal number of message IDs stored for each time quantum, significantly impacting the number of bytes occupied in memory. To reduce the memory footprint of this implementation, F has to be reasonably low. For simplicity of automatic code generation, F will be a single constant for all ECUs.

To formulate the constraint precisely, the definition of a new mathematical object is needed, namely of a unit u representing a single ECU with the set of transmitted messages $M_u \subseteq M$. In every time quantum j, there shall be an upper bound F on the number of messages each ECU u will transmit, i.e.

$$\sum_{m \in M_u} \sum_{i=0}^{N_m-1} x_{m,i,j} \leq F \qquad \text{for every unit } u \text{ and time quantum } j \in \{0, 1, \cdots, H-1\}. \tag{10}$$

5) Fine upper bound on transmission period jitter: The coarse constraint on upper bound on transmission period jitter constructed in section III-B2 can be significantly improved using the exact definition of the transmission start time (4). It is possible to enable sub-quantum control over the maximal period jitter using the Big M method in conjunction with the following series of observations. Since $\bar{t}_{m,i,j}$ is the start time of i-th transmission of message m within the time quantum j, one can require that

$$-J \leq \bar{t}_{m,i+1,k} - \bar{t}_{m,i,j} - T_m \leq J, \tag{11}$$

but these inequalities are only meaningful when $J_{m,i}$ is scheduled for transmission in quantum j and $J_{m,i+1}$ is scheduled in quantum k, i.e. iff $x_{m,i,j} + x_{m,i+1,k} = 2$ using the set of binary decision variables. Combining these ideas together creates a constraint

$$| \bar{t}_{m,i+1,k} - \bar{t}_{m,i,j} - T_m | \leq J + (2 - x_{m,i+1,k} - x_{m,i,j}) \cdot H \qquad \text{for } m \in M,\ i \in \{0, 1, \cdots, N_m - 2\} \text{ and } j, k \in \{0, 1, \cdots, H-1\}, \tag{12}$$

where H acts as the Big M – the constraint is inactive unless both $x_{m,i+1,k}$ and $x_{m,i,j}$ have the value 1 as the difference between two timestamps within a single hyper-period clearly cannot be greater than the hyper-period length H itself.

The equation (12) can be seen as an extension of (7) facilitating a sub-quantum control over the transmission period jitter. It can be easily rewritten in the form suitable for a linear program using the transformation

$$| a | \leq b \qquad \Rightarrow \qquad -b \leq a \text{ and } a \leq b. \tag{13}$$

In practice, due to the complexity of equation (12) caused by the large number of optimization variables participating in the expression for transmission start time $\bar{t}_{m,i,j}$ (note that (4) is a linear combination with one term for every transmission of every higher-priority message), using (12) yields an order of

$$(\underbrace{\sum_{m \in M} N_m}_{\text{No. of all } J_{m,i}})^2 \cdot H^2. \tag{14}$$

new constraints.

This makes the problem hardly tractable or entirely intractable, and the Python script execution was interrupted after six hours of runtime. Just the creation of the optimization model took approximately 14 minutes, yielding a model with 406134 rows, 49800 columns and 377 million nonzeros. Significant improvement was achieved by imposing constraint (12) not for every messages m and every pair $(j, k) \in \{0, 1, \cdots, H-1\}^2$, but rather only for sensible j and k. It is clear that when optimizing to minimize the TX period jitter, all transmissions of a single message should ideally spread equidistantly over the whole hyper-period. One could either use lazy constraints to dynamically restrict the search space when needed or use a limited number of traditional constraints based on this prior knowledge. For each message transmission $J_{m,i}$, a notion of a *sensible interval* is introduced to capture the general part of hyper-period the transmission should be scheduled in. For example, when scheduling a message with period $T_m = 10$ quanta over the hyper-period of length $H = 100$, a sensible start time of the first transmission belongs to the interval $[0, 2T_m)$ rather than $[2T_m, H)$. This sensible interval was calculated for all messages and transmissions, and subsequently (12) was only specified for j and k belonging to sensible intervals of corresponding transmissions, reducing the quadratic term H^2 in (14) to a lower number proportional to T_m^2.

C. Optimization goals

The first part of this semestral work aimed to obtain any feasible schedule considering imposed requirements. Therefore, many symbols defined above in the problem statement were assigned constant values calculated using the following thought process.

Since the typical ECU system clock used in eForce's electric formulae has $1\,\mathrm{ms}$ resolution and the CAN bus baud rate is 1 Mbitps, it is reasonable to select the length of a single time quantum as $Q = 1000$ bits or $1\,\mathrm{ms}$. Furthermore, since in our case 98.3 % of strictly periodic communication on the CAN bus belongs to messages with transmission periods $T_m \leq 100\,\mathrm{ms}$, hyper-period length $H = 100$ was chosen. It is a reasonable tradeoff between the complexity of the underlying optimization problem (the number of decision variables x – as defined in (2) – is, in fact, quadratic in the hyper-period length H) and its expressiveness. If only $H = 50$ was chosen, the static schedule would not describe any message with $T_m = 100\,\mathrm{ms}$, of which there are many.

Regarding the maximal memory footprint of the static scheduling table in the ECU memory, a reasonable decision is to permit the size of `msg_id schedule[H][F]` at most $1\,\mathrm{kB}$. Since `sizeof(msg_id)` $= 2$, this leaves us with the reasonable value $F = 1000/2/H = 5$. The maximal allowed bus utilization was chosen by reserving $R = 200$ bits such that the static schedule guarantees enough bandwidth for at least three sporadic or event-triggered messages per each pair of consecutive time quanta (from (1), three messages amount to at most 396 bits, whereas two consecutive time quanta combined will reserve 400 bits).

In the second part of the semestral work, optimization was carried out with one of J, T or F as the objective to minimize while others were assigned fixed values.

D. Implementation

The optimization problem defined above was turned into a Python script parameterized with H and Q and paths to one input and one output text file. All computations were executed using *Gurobi Optimizer version 10.0.1 build v10.0.1rc0 (win64)* running on AMD Ryzen 7 5700U with Radeon Graphics (instruction set [SSE2—AVX—AVX2]) with 8 physical cores and 16 logical processors.

The input file has structure shown in listing 1. The first line contains a single number n – the number of distinct CAN messages. Each of the following n lines describes a single message in five whitespace-separated columns containing the name of the transmitting unit u (in this case either the *Accumulator Management System* or *Inertial Navigational System*), the name of the message m (e.g. *ACP_STATISTICS*), its identifier i_m, period T_m in time quanta and length of data in bytes (used to obtain l_m using the equation 1), respectively.

Listing 1: Format of the input file

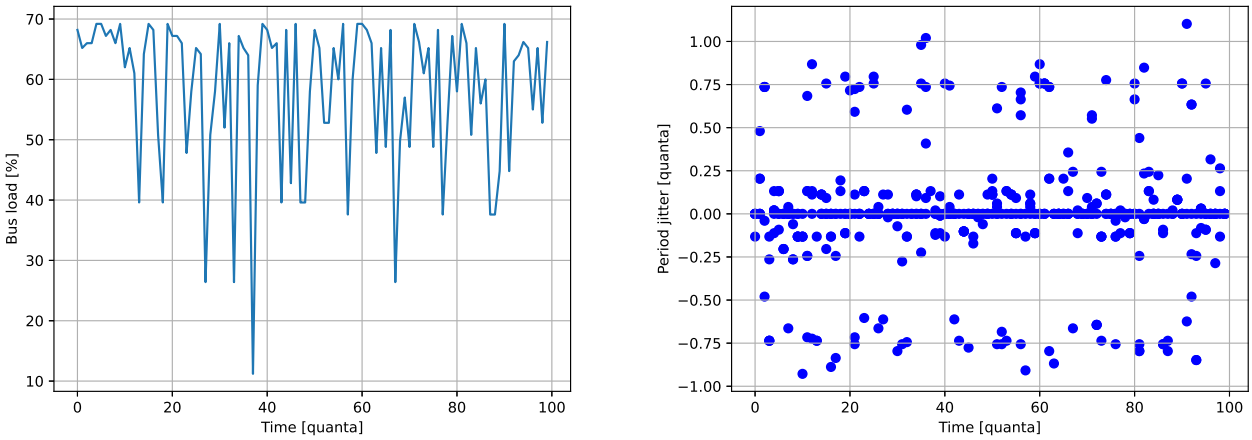
```
118
AMS     ACP_MEAS          912    10    8
AMS     ACP_STATISTICS    914    10    2
INS     EKF_EULER         306     5    6
INS     EKF_ORIENT_ACC    307     5    6
[...]
```

The program output is structured as shown in listing 2. The first line contains three integers – the number of messages n, the hyper-period length in quanta H and the length of each time quantum in bits Q. Subsequent n lines each describe a schedule of one message m by specifying its identifier i_m and number of transmissions per hyper-period N_m followed by N_m integers $t_{m,i}$ – zero-based indices of time quanta individual transmissions $J_{m,i}$ of m are scheduled for.

Listing 2: Format of the output file

```
118  100  1000
912  10    7  16  26  35  46  57  68  78  88  97
914  10    8  17  26  34  45  54  64  75  86  97
306  20    2   7  13  17  22  26  31  35  39  45  51  56  61  65  71  76  82  87  93  97
307  20    0   6  10  15  19  23  28  34  39  45  50  55  61  65  70  74  78  84  89  94
[...]
```

Apart from the solution to ILP, auxiliary scripts were created to visualise and present scheduling results using *matplotlib*. They were used to generate all figures displayed in this report.



(a) Bus utilization in time over one hyper-period

(b) TX period jitter in time over one hyper-period

Fig. 1: Optimal schedule with minimal peak bus utilization of $T \approx 69$ %

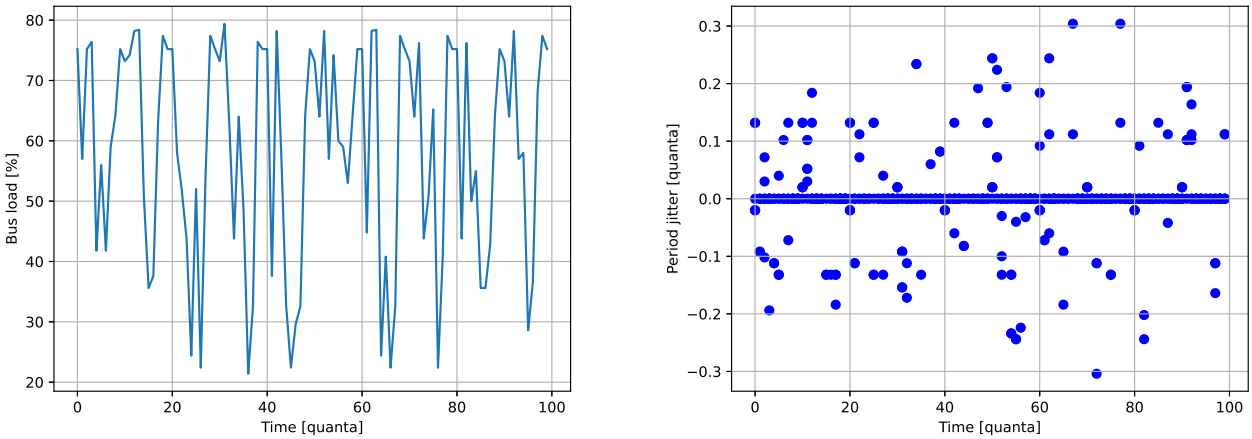
IV. Experiments

A. Results

Metrics for comparison of various schedules – as described in preceding sections (mainly I-C) – are based on the TX period jitter and the bus load. First, multiple indicators such as the maximum or RMS value may be utilized to assess the jitter. Figures in this report show the true jitter in time to show both the general trend and possible outliers should any appear. Second, regarding the bus utilization, not only the peak (maximal) utilization but also the variance (or standard deviation) is important as one of the goals stated in I-A is to eliminate transmission bursts (interleaving periods of low and high bus utilization).

1) Minimal peak bus load: To search for the optimal schedule with respect to peak bus load, values of $J = 1.2$ and $F = 5$ were used in order not to impose too restrictive constraints. The optimization achieved $T \leq 70$ %, as shown in Fig. 1.

2) Minimal jitter: The optimal jitter schedule was searched for with $T = 80$ % and $F = 5$. Thanks to the sub-quantum resolution of the message jitter, the optimization pushed J as low as 0.306. Metrics of this optimal schedule are shown in Fig. 2.



(a) Bus utilization in time over one hyper-period

(b) TX period jitter in time over one hyper-period

Fig. 2: Optimal schedule with minimal TX period jitter of $J \approx 300$ bits

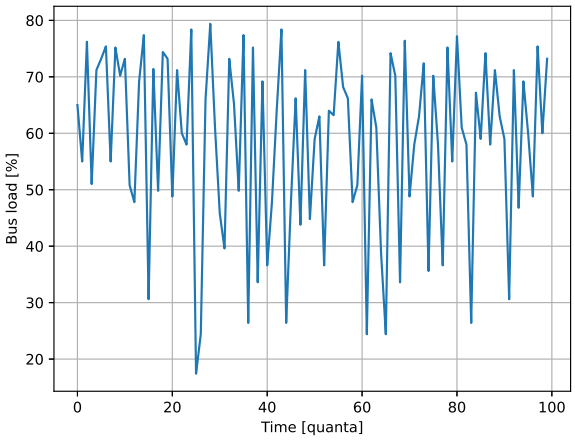
3) Memory footprint optimization: After the initial calculation of acceptable value $F = 5$ (resulting in cca 1 kB memory footprint of static schedule in the ECU firmware), a better value was searched for by converting the constant F into an integer optimization variable and using it as the model objective for minimization. An optimal solution with $F = 3$ was found; its metrics are shown in Fig. 3.

B. Discussion

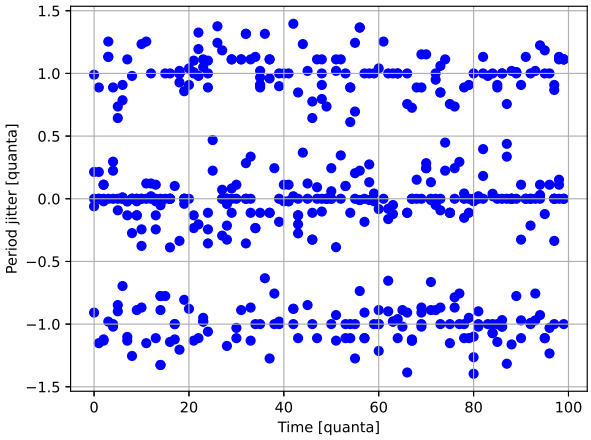
Results from individual experiments vary dramatically based on the optimization criterion as well as the set of experiment parameters. Optimizing for one specific criterion typically leads to significant deterioration in values of the other criteria that can only be mitigated either by imposing strict requirements (i.e. hard constraints restricting the problem search space) or possibly using softer constraints, such as using a linear combination of F, T and J as the objective function. This decision has to be made using application-specific knowledge of the concrete communication network, as well as a grain of educated guess since there is generally no clear indicator of what values the "correct" weight vector c in objective function $c^T \cdot [J,\ T,\ F]^T$ should contain. This would become yet another optimization problem – an optimization problem over the set of hyper-parameters of the optimization problem studied in this report.

The optimization with respect to T has a clear visual explanation – the solver draws a horizontal line bounding the graph in Fig. 1a from above and attempts to push the boundary as low as possible. Since the total number of bits to transmit is constant, this process, in some sense, pours communication from peaks into trenches of the bus utilization graph and thus has the side effect of decreasing bus load variance. Comparing Fig. 1a with Fig. 2a clearly shows that the communication is more uniform when optimizing for T rather than for J at the expense of worse (but still tolerable) jitter.

On the other hand, optimization for the lowest peak jitter has pretty much the opposite effect. Although the bus utilization is far from uniform, the benefit of negligible period jitter outweighs all drawbacks. This optimization criterion appears to be the



(a) Bus utilization in time over one hyper-period



(b) TX period jitter in time over one hyper-period

Fig. 3: Optimal schedule with minimal memory footprint in the ECU memory (best $F = 3$)

most useful as it can potentially improve the performance of various advanced mathematical methods (such as Kalman filtering) relied on by the electric vehicle's control systems by ensuring near-ideal periodicity of data reception.

The minimization of F resulted in the deterioration of both jitter and bus utilization metrics, reaching worst-case period jitter of almost 1.5 time quanta in both negative and positive directions. Nevertheless, the result can be useful in the case of ECUs with very low-cost memory-starved microcontrollers. Allowing at most three messages per unit per time quantum bounds the maximal memory footprint from above at $600\,\mathrm{B}$ which may be a game-changing improvement when the complete software has to fit into only a handful of KiB of RAM.

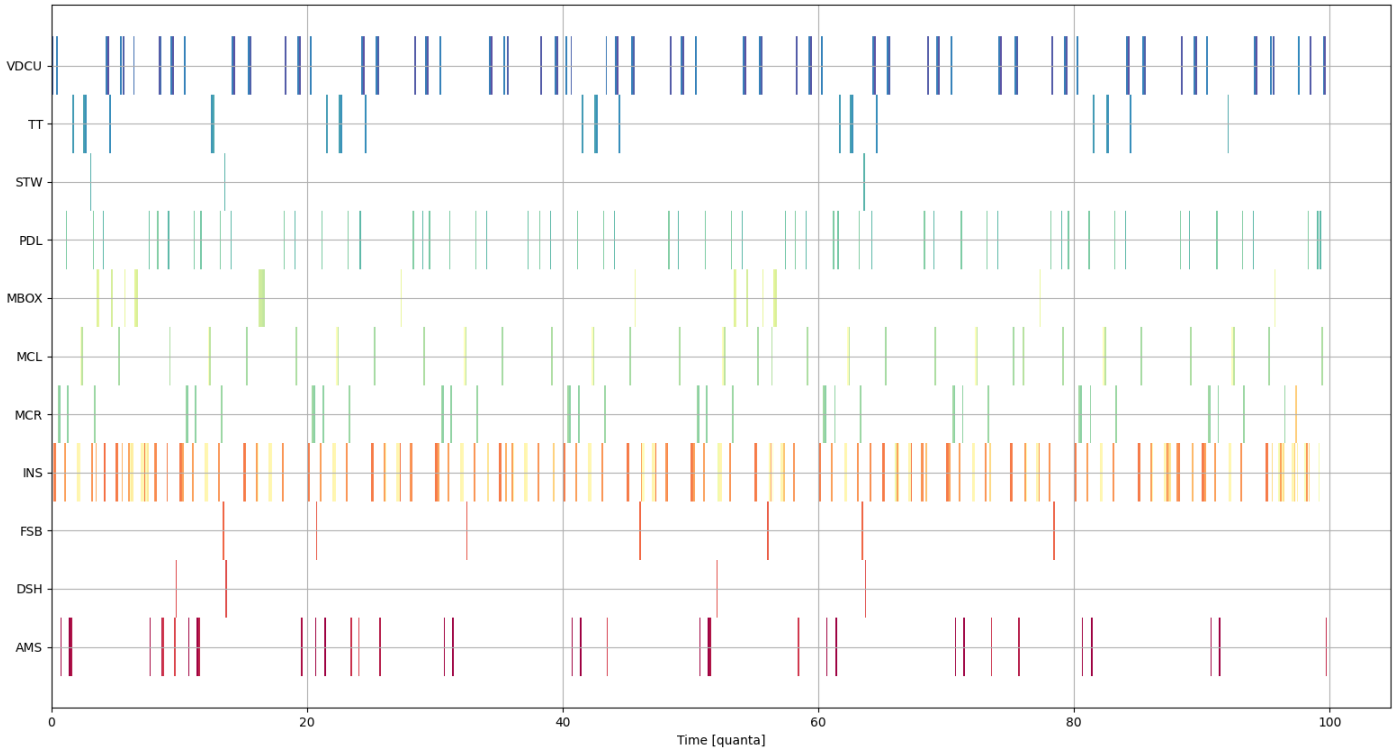


Fig. 4: Visualization of complete schedule for a $100\,\mathrm{ms}$ hyper-period.
The vertical axis lists all ECUs present on the CAN bus

V. Conclusion

The problem of static CAN bus communication scheduling for approximately one hundred twenty messages was successfully solved using the means of integer linear programming, yielding schedules such as the one shown in Fig. 4, optimal with respect to various optimization criteria. The actual implementation of the described algorithm is highly parametric, allowing swift modifications. Several optimization objectives have been tested, each yielding interesting results. The user can decide for optimization for the lowest possible message transmission period jitter resulting in a schedule with J as low as $300\,\mu s$, sufficient even for the most demanding control algorithms such as autonomous driving. Alternatively, the schedule can be optimized for the lowest possible memory footprint in the ECU software, yielding results as low as $600\,B$ of requires storage. Finally, optimization for minimal peak bus utilization yields results that guarantee sufficient communication bandwidth and low latency to messages of purchased ECUs, whose software can't be modified to implement the static schedule.

Subsequent work should investigate whether the presented algorithm is usable for scheduling CAN FD communication immediately or whether it could be further improved by exploiting features added by the newer communication technology. Another possible improvement in the optimization algorithm would be to look into the incorporation of variable message identifiers in order to allow the optimization to suggest modification of message identifiers to achieve perfect control over communication during individual time quanta.

This semestral project with both the written Python code as well as the report has laid a basis for further improvements, but it is already usable in the present state in practice when statically scheduling CAN bus communication in the first Czech Formula Student Electric, designed and constructed by the team eForce representing the Faculty of Electrical Engineering of Czech Technical University in Prague.

References

[1] Surma, D.R. and Sha, E.H.-M., *"Static communication scheduling for minimizing collisions in application specific parallel systems,"*, Proceedings of International Conference on Application Specific Systems, Architectures and Processors: ASAP '96, 1996, DOI 10.1109/ASAP.1996.542819

[2] R. Dobrin and G. Fohler, *"Implementing off-line message scheduling on controller area network (CAN),"* ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597), Antibes-Juan les Pins, France, 2001, pp. 241-245 vol.1, doi: 10.1109/ETFA.2001.996374

[3] Wikipedia, Wikimedia Foundation, *"CAN Bus."*, en.wikipedia.org/wiki/CAN_bus. Accessed 26 May 2023.

[4] H. Shokry, M. Shedeed, S. Hammad, M. Shalan and A. Wahdan, *"Hardware EDF scheduler implementation on controller area network controller,"* 2009 4th International Design and Test Workshop (IDT), Riyadh, Saudi Arabia, 2009, pp. 1-6, doi: 10.1109/IDT.2009.5404095.

[5] K. M. Zuberi and K. G. Shin, *"Design and implementation of efficient message scheduling for controller area network,"* in IEEE Transactions on Computers, vol. 49, no. 2, pp. 182-188, Feb. 2000, doi: 10.1109/12.833115.

[6] Z. Sun, H. Li, M. Yao and N. Li, *"Scheduling Optimization Techniques for FlexRay Using Constraint-Programming,"* 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, Hangzhou, China, 2010, pp. 931-936, doi: 10.1109/GreenCom-CPSCom.2010.111.