Appendix for Multi-variant scheduling of critical time-triggered communication in incremental development process: Application to FlexRay

Jan Dvořák*†, Zdeněk Hanzálek†
*Department of Control Engineering, Faculty of Electrical Engineering, CTU in Prague
†Industrial Informatics Research Center, Czech Institute of Informatics, Robotics and Cybernetics, CTU in Prague

***Abstract*—The portfolio of models offered by car manufacturing groups often includes many variants (i.e., different car models and their versions). With such diversity in car models, variant management becomes a formidable task. Thus, there is an effort to keep the variants as close as possible. This simple requirement forms a big challenge in the area of communication protocols. When several vehicle variants use the same signal, it is often required to simultaneously schedule such a signal in all vehicle variants. Furthermore, new vehicle variants are designed incrementally in such a way as to maintain backward compatibility with the older vehicles. Backward compatibility of time-triggered schedules reduces expenses relating to testing and fine-tuning of the components that interact with physical environment (e.g., electromagnetic compatibility issues). As this requirement provides for using the same platform, it simplifies signal traceability and diagnostics, across different vehicle variants, besides simplifying the reuse of components and tools.**

This paper proposes an efficient and robust heuristic algorithm, which creates the schedules for internal communication of new vehicle variants. The algorithm provides for variant management by ensuring compatibility among the new variants, besides preserving backward compatibility with the preceding vehicle variants. The proposed method can save about 20% of the bandwidth with respect to the schedule common to all variants. Based on the results of the proposed algorithm, the impact of maintaining compatibility among new variants and of preserving backward compatibility with the preceding variants on the scheduling procedure is examined and discussed. Thanks to the execution time of the algorithm, which is less than one second, the network parameters like the frame length and cycle duration are explored to find their best choice concerning the schedule feasibility. Finally, the algorithm is tested on benchmark sets and the concept proved on the FlexRay powered hardware system.

Note: This document containts only the Appendix for the original paper:
J. Dvorak and Z. Hanzalek, "Multi-variant scheduling of critical time-triggered communication in incremental development process: Application to FlexRay," in IEEE Transactions on Vehicular Technology (doi: 10.1109/TVT.2018.2879920).
Thus, the figures or tables refer to the original paper.

APPENDIX A
BENCHMARK INSTANCE GENERATION PROCEDURE

In this section, the process of the multi-variant benchmark instance generation is described. The process is depicted from a high-level in Algorithm 1.

Input : Instance parameters
Output: Multivariant benchmark instance
Read the instance parameters;
for *each signal* s_i *in* S **do**
  Generate the signal period;
  Generate the signal payload;
  Generate the signal deadline;
  Generate the signal release date;
end
Assign the transmitting ECU to commom signals;
Assign the transmitting ECU to specific signals;
Assign the transmitting ECU to other signals;
Generate variant matrix $V_{i,j}$;
Repair instance $V_{i,j}$;

Algorithm 1: Scheduling of unscheduled/new signals

At the beginning, the required instance parameters are read. These parameters consist of distributions presented in Fig. 11 or Fig. 12, parameters from Table III, the number of signals and the number of variants to generate, the multi-variant coefficients α and β for signals and similar coefficients for ECUs. Note that ECUs can be common to all variants (so-called common ECUs) or specific to just one variant (so-called specific ECUs) in the same way as signals can be.

Subsequently, the basic signal parameters are generated for each signal. The periods and payloads follow the requested distributions. Deadlines and release dates are generated only for the requested portion of the signals determined by the instance parameters. The deadline is set to the end of a randomly chosen communication cycle from the last third of the signal period. The release date is set to the beginning of the communication cycle also randomly chosen from the first six communication cycles.

After generation of basic signal parameters, the transmitting ECUs are assigned to the signals. Firstly, the ECUs are assigned to the common signals. The common signals can be transmitted from the common ECUs only. The common ECUs similar to the common signals have to be included in all variants. Secondly, the ECUs are assigned to the specific signals. Inversely to the case with common signals, specific ECUs are allowed to transmit specific signals only. Otherwise, the specific ECUs would be forced to appear in more than one variant. In the end, the ECUs are assigned to the rest of

the signals that are shared, and it is assured that each ECU transmits at least one signal.

The generation of variant matrix $V_{i,j}$ is divided into two steps. The first step is deciding which ECUs are used in which variant. The common ECUs are used in all variants, and the specific ECUs are used only in one randomly chosen variant. The rest of the ECUs are distributed to the random subset of variants. In the second step, the signals are assigned to the variants. All the common signals are assigned to all the variants. The specific signals that are transmitted by the specific ECUs are assigned to the same variant as the specific ECUs. The rest of the specific signals are assigned to randomly chosen variants to which the transmitting ECU is assigned. For the case of shared signals, random probability from 30 to 70 % is chosen for each variant. This probability determines whether it is rather luxurious or economy variant. With this probability, the shared signals are assigned to the particular variant if the transmitting ECU is used in the variant.

According to this strategy of assigning signals to variants, situations can occur when some signal is not assigned to any variant. Thus, it is necessary to repair such issues in matrix $V_{i,j}$. Each signal in $V_{i,j}$ is checked whether the signal is assigned to some variants. If it is not, the signal is assigned to a random subset of variants assigned to its transmitting ECU. Finally, the admissible multi-variant instance is generated that satisfies all the requested instance parameters.

However, the described process does not take into account any predecessor benchmark instance and, thus, it is useful only for the generation of the benchmark instance for the first iteration of incremental scheduling. The generation of subsequent iterations follows a process depicted in Algorithm 2. The

Input : Instance parameters
Instance for the previous incremental iteration
Output: Incremental multivariant benchmark instance
Read the instance parameters;
Read the instance for the previous incremental iteration;
for *each new signal* s_i *in* $S \setminus \tilde{S}$ **do**
  Generate the signal period;
  Generate the signal payload;
  Generate the signal deadline;
  Generate the signal release date;
end
Assign the transmitting ECU to the new signals ;
Add the new variant to variant matrix $V_{i,j}$;

Algorithm 2: Scheduling of unscheduled/new signals

generation starts with the reading of the requested instance parameters. Then, the instance of the previous incremental iteration is read. The new instance is going to be based on this so-called original instance. All the signals and ECUs from the original instance will be present in the new instance with the unchanged basic parameters.

Then the basic parameters are generated for each new signal. The generation process is the same as in case of non-incremental instance generation. However, in this case, it cannot be assured that the new instance will follow the requested instance parameters because the parameters distribution in the original instance can vary significantly from the requested instance parameters.

In the next step, the new signals are assigned to its transmitting ECUs. If there is no new ECU, then the signals are uniformly distributed among all ECUs. However, if there are some new ECUs, 70 % of the new signals are distributed among these new ECUs, and the rest is uniformly distributed among all the ECUs. Moreover, it is assured that all new ECUs are used in the new instance.

Finally, a new variant is added to the variant matrix $V_{i,j}$. No variant used in the original instance is changed. The new variant is based on a randomly chosen variant (so-called original variant) from the original instance, and all new signals and new ECUs are assigned to it. Because it is not often the case, in practice, that the new variant just adds new signals and ECUs to the original one, part of the variant matrix $V_{i,j}$ copied from the original variant is mutated. The signals from the original instance are processed one by one. Each signal has a 70 % chance that it will not be passed to the mutation stage at all. Once the signal reaches the mutation stage, the following mutation rules are employed:

- If the signal appears in the original variant, it has a 35 % chance that it will not appear in the new variant.
- If the signal does not appear in the original variant and its transmitting ECU appears in original variant, it has 65 % chance that it will appear in the new variant.
- If the signal and its transmitting ECU does not appear in the original variant, it has $\frac{1}{3}$ % chance that it will appear in the new variant. In this case, the transmitting ECU is added to the original variant also.

After all these steps, the new incremental multi-variant benchmark instance is ready.

List of Symbols and Abbreviations

MS^{ECU}	Unit multischedule
MS^{O}	Original multischedule
$MS_{i,j}$	Multiframe in cycle i and slot j
SL^{N}	Ordered set of new and dummy signals
SL^{N}	Set of new signals
$\tilde{o_i}$	o_i from original assignment
$\tilde{S}$	Subset of signals used in the original multischedule
$\tilde{t_i}$	t_i from original assignment
$\tilde{y_i}$	y_i from original assignment
c_i	Payload length of signal s_i
CG	Conflict graph
d_i	Deadline of signal s_i
e_i	Index of ECU transmiting slot l_i
E_{CG}	Edges from conflict graph
G_{SLOT}	Graph for Slot scheduling
H	Number of cycles in the hyperperiod
i	Index
j	Index
l_i	Slot with index i
M	Duration of one communication cycle
n_i	Identifier of the transmitting ECU of signal s_i
N_{CG}	Nodes from conflict graph
o_i	Offset in the frame of signal s_i

p_i	Period of signal s_i
r_i	Release date of signal s_i
S	Set of all signals
s_i	Signal i
S_{MIS}	Set of signals from the Maximal independent set
t_i	Identifier of the slot
$V_{i,j}$	Binary matrix of the signal-to-variant assignment
W	Maximal frame payload length
w_i	Weight of node i in MWIS
y_i	Identifier of the communication cycle
EEM	ECU Mutual Exclusion Matrix
MS	Multischedule
SEM	Signal Mutual Exclusion Matrix
D	Set of dummy signals
freeBits	Set of unallocated bits in the slot
MIS	Maximal independent set
MWIS	Maximal weighted independent set problem