Scheduling for a TTCAN network with a stochastic optimization algorithm

José Fonseca*, Fernanda Coutinho**, Jorge Barreiros**
* DET / IEETA – Universidade de Aveiro, Portugal
** Instituto Superior de Engenharia de Coimbra, Portugal

The TTCAN protocol is a new development in CAN technology that specifies a time-slot based communication mechanism. TTCAN technology avoids the transmission collisions commonly found in standard CAN networks. In TTCAN, all the message instances are transmitted only on previously allocated time-slots. No other instance may be transmitted on an allocated time-slot, so transmission collisions are avoided.
Before transmitting a message set over a TTCAN network, it is necessary to create a valid scheduling table that specifies how the time is discretized into time-slots and how the message instances are allocated into those time-slots. Building a valid optimized scheduling table that follows the TTCAN specification isn't trivial. Additionally, several distinct scheduling tables may be built for the same message set, so it is necessary to use some quality criterion to select one among all. Typical message sets include a large number of messages, making manual scheduling a very hard and error-prone process. This means it is desirable to use automated tools that compute a feasible scheduling table automatically for a given message set. In this paper, we present a scheduling tool for TTCAN networks that generates a valid scheduling table using a stochastic optimization algorithm. The tool attempts to optimize the scheduling table so it will generate solutions with low jitter. The tool was applied to two well known message sets used in the automotive industry and results are positive.

1. Introduction

The CAN fieldbus has been used in automotive industry for real-time distributed systems embedded in vehicles since several years. Recently, the interest of the industrial and research communities in communication based on the time-triggered paradigm [1] has increased substantially and a standardization process led to the definition of a new session layer for CAN. The future standard, known as time-triggered CAN (TTCAN) [4] uses a scheduling table that must be built off-line, prior to the system start of operation. In this paper, we present an early version of a tool that can be used to simplify the construction of the required scheduling table. We begin with a brief presentation of the TTCAN protocol, and then we follow by describing the scheduler and optimizer algorithms. We briefly describe the tool operation and some of its screens. We show some results obtained from applying the tool to a modified PSA message set [3] with messages having harmonic periods. The results are briefly analyzed, some conclusions are extracted and some future work is identified.

2. TTCAN – Time-triggered CAN

During the last years, CAN – Controller Area Network has become the main fieldbus used in the time critical parts of automotive systems. Although CAN operates following the event-triggered paradigm, recent academic studies [8] pointed to the possibility of using a time-triggered approach in part of the operation of CAN based distributed systems. This feature could improve the bandwidth utilization in CAN-based real-time systems by separating the periodic traffic from the aperiodic one and keeping the timeliness guarantees of the former. From the industrial side, the interest of using the time-triggered approach in CAN has also been recognized since some time and, as a consequence, a standardization task force (ISO/TC 22/SC 3/WG 1/TF 6) was launched under International Standards Organization in order to achieve a definition for a session layer for CAN. This standard, which will be numbered

11898-4, is already known as TTCAN, from Time-Triggered Controller Area Network.
In TTCAN, a special node, the time master, is responsible for the transmission of a systolic message, the reference message, which is used to achieve synchronization between the fieldbus nodes. The reference message marks the start of a time slot called the Basic Cycle (BC). The BC may be divided in different types of windows, namely, the exclusive windows and the arbitrating windows. An exclusive window is used to transmit a specific periodic message. An arbitrating window may be used to transmit any message, provided it gains the bitwise arbitrating process as in normal CAN operation and provided it finishes transmission before the end of the window, thus guaranteeing temporal isolation between the different types of traffic.
The complete traffic pattern in a TTCAN system consists in a fixed number of consecutive BCs and is named the System Matrix or Matrix Cycle (SM). In figure 1, an example of a system SM is presented. There, it is possible to notice one of the major restrictions in the SM construction, which is the column like organization. All the windows of the same column in every BC must be of the same size. With this organization it is possible to define a set of trigger instants (offsets from the reference message) which is kept constant from BC to BC all along the SM, thus simplifying the TTCAN controllers hardware. Once the MC is defined, it is possible to merge two (or more) consecutive arbitrating windows in the same BC in order to facilitate the transmission of normal CAN messages. Finally, The other major restriction is in the number of BCs per SM, which must be an integer power of two.

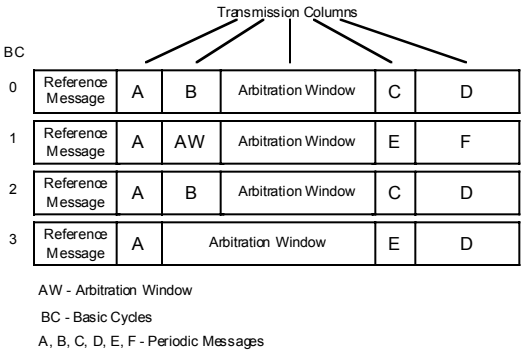

Figure 1 Example of a Matrix Cycle in TTCAN.

In TTCAN, the system matrix must be defined off-line. All the nodes must have stored the correspondent information prior to the start of operation. Details about TTCAN operation modes, synchronization techniques and other specification details can be found in [4], [5] and in the draft standard.

3. Building a schedule

The process of off-line scheduling a message set in the case of a TTCAN network consists in building the system matrix (SM). It includes:

- The determination of the number of columns.
- The determination of the number of rows.
- The definition of the duration of each column.
- The indication of the message to be transmitted in each cell (row, column).

This must be done respecting the messages' period and duration. In what concerns the period, the first approach is to keep the average period of each message identical to the respective instantaneous period. This is done using the appropriate number of message instances in the system matrix. The average period is equal to the duration of SM divided by the number of message instances. In what concerns the messages' duration, this determines the column width.
In spite of these restrictions and of the one imposed to the number of Basic Cycles as indicated in section 2, it is usually possible to build several distinct system matrixes for the same message set. Consequently, it is possible to assess the quality of the schedule in the System Matrix according to some pre-defined criterion.
In the present case we use a cost function based on the sum of the message jitter values. Jitter is determined for every instance of every message all over the system matrix. The cost function is weighted by the system matrix duration. The correspondent expression is the following:

$$Jitter = \frac{1}{M} \sum_{p} \sum_{i} \left| e_i^p - a_i^p \right|$$

where e_i^p is the expected beginning time of transmission of instance i of message p, a_i^p is the actual beginning time and M is the duration of the system matrix.

Considering now the automatic tool operation, the required phases needed to build the system matrix are the following:

- Scheduling.
- Optimization.

The first corresponds to the generation of a feasible set of distinct system matrixes (SMs). The second does the selection of the best SMs according to the cost function above.
In order to be able to maintain the average period of every message, the SMs duration, M, must be the least common multiple of the messages' periods (it could also be an integer multiple of it). The average period is kept with M/P_i instances of every message i of period P_i during the SM.

The first phase, scheduling, includes the following steps:

1. Determination of the maximum number of lines of the system matrix.
2. Message allocation.
3. Free time redistribution.

Ignoring the restriction about the number of basic cycles, it is obvious that the maximum number of lines in the SM is bounded by:

$$L_{\max} = \frac{M}{T_{\max}}$$

where $T_{\max}$ is the maximum transmission time of all the messages in the set.
Before starting the allocation process, the automatic tool generates an ordered set I which includes every instance of every message in the initial set. This set is organized by decreasing order of the message transmission time T:

$$I = \left\{ I_1^1, I_1^2, ..., I_1^{K_1}, I_2^1, ..., I_2^{K_2}, ..., I_n^1, ..., I_n^{K_n} \right\}$$

with n being the number of different messages in the initial set and:

$$K_i = \frac{M}{P_i} \text{ and}$$

$$T_1 = T_{\max} > T_2 > ... > T_i$$

The next step in the operation of the tool consists in defining a random number of lines:

$$L \le L_{\max}$$

Then, the tool removes the first L instances in I and allocates them in the first column of the system matrix. The tool repeats this last step until all instances in I are removed, thus obtaining a SM with the following number of columns, $\#C$:

$$\#C = \left\lceil \frac{\#I}{L} \right\rceil$$

As the longer messages are taken into consideration first, $\#C$ is minimum.
At this moment it is possible to determine the minimum duration of the basic cycle for this particular system matrix:

$$D_{BC} = \sum_{i=1}^{\#C} D_i$$

with D_i being the duration of each column.
With this value it is possible to determine if the set is schedulable which happens when:

$$D_{BC} \le \frac{M}{L}$$

If the set is schedulable then there is some free time available in each basic cycle:

$$t_{free} = \frac{M}{L} - D_{BC}$$

The tool inserts a free column between each two occupied columns and redistributes the free time randomly between the first ones. This is the only random factor in the construction of the first set of system matrixes.

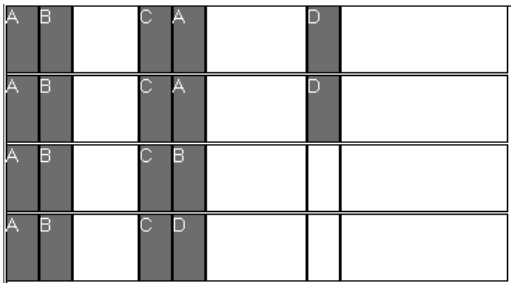

Figure 2 - Example of a first System Matrix, after an initial random transformation.

3.1 Optimization process

The optimization process uses a set of feasible system matrixes built as described previously. These matrixes, considered the initial population in a similar way to genetic algorithms, will be subject to random transformations. These transformations must guarantee that the matrixes are still feasible afterwards. The cost function defined above determines the quality of the matrixes.
The complete set of steps in this process is the following:

1. Generation of the initial population;
2. Diversification of the population;
3. Random selection of a matrix;
4. Application of a random transfor-mation to the selected matrix;
5. Evaluation of the cost function for this matrix.
6. If this matrix is worse than the worst matrix in the population then it is eliminated, otherwise it replaces the latter;
7. Repeat steps 4 to 6 till the pre-defined maximum number of iterations is attained.

Considering step 1, the number of elements that constitute the initial population is defined previously by the user. The scheduler generates then a feasible system matrix as it was explained before.
In step 2 the tool applies to the elements of the population a random transformation without caring about quality. This originates some diversity in the initial population. This operation ends the initialization process.
Now the main body of the optimization algorithm is attained. The algorithm is based on a steady state genetic algorithm [6], [7]. However, we called it a stochastic algorithm, as we couldn't find any simple process to apply crossover keeping the system matrixes feasible.
Another issue regarding the completion of the algorithm is the number of iterations. In fact, it is impossible to define an absolute value for the cost function that could be used in every message set. Jitter depends on the transmission load and on the relationship between the messages' periods. The number of iterations must then be pre-imposed by the user. It is easy to choose an adequate value after running a couple of optimization processes.

4. A brief look at the tool and some results

Before starting the operation of the tool, the optimization settings must be initialized (figure 3). These are the number of runs, the number of iterations, the size of the initial population and the probabilities of each transformation.

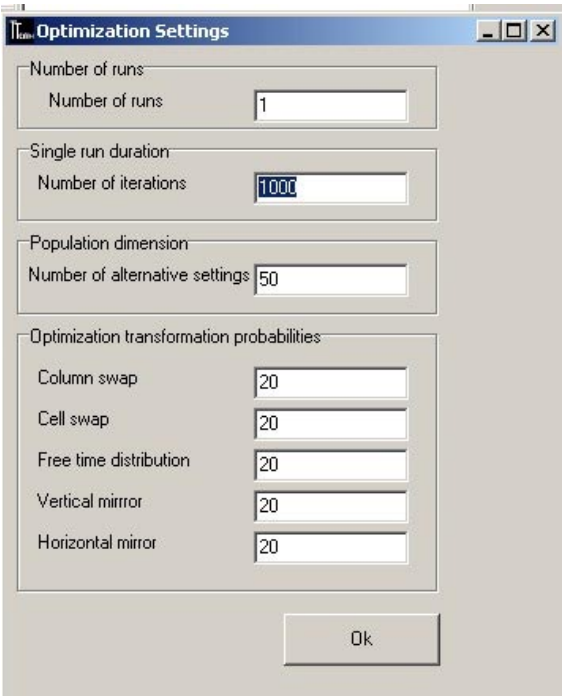

Figure 3 – Initialization of parameters.

After, a message set must be chosen from a file or a new one must be created. The tool has also the adequate editing screen (figure 5).

When this is done, it is possible to run the optimization tests. When the message set has not too many messages, the user can get a flavor of the final optimized schedule directly from the main tool screen (figure 6). The accurate results can be obtained from an output file. An example is shown in figure 4.

In [9] it was shown that the optimization tool was able to schedule the well known SAE [2] and PSA [3] message sets which are often used as benchmarks for fieldbus based distributed systems. The messages in the sets have non-harmonic periods so, it is only possible to reduce jitter, which was achieved by the tool, but it is impossible to eliminate it.

In order to verify if the tool is able to find a zero jitter solution, we decided to use a modified PSA message set in which the messages' periods were slightly reduced in order to become harmonic. The messages' parameters are the ones in figure 5.

```
Config
            Pop size:10
    Nr of iterations:20000
Probabilities
        column swap:20
          cell swap:20
  free time distrib:20
      horiz mirror :20
  vertical mirror  :20
Rows:8
Columns:12
Column duration:
808 736 1040 1040 1096 844 584 1427
260 736 773 656
Scheduling matrix:
(-1 represents an empty cell, other
values are the index of the message
(with the same order that was
defined in the input file))
4 6 0 10 -1 -1 3 -1 -1 8 -1 1
5 6 0 9 -1 -1 3 -1 -1 2 -1 1
4 6 0 -1 -1 -1 3 -1 -1 8 -1 1
11 6 0 7 -1 -1 3 -1 -1 2 -1 1
4 6 0 10 -1 -1 3 -1 -1 8 -1 1
5 6 0 -1 -1 -1 3 -1 -1 2 -1 1
4 6 0 -1 -1 -1 3 -1 -1 8 -1 1
-1 6 0 7 -1 -1 3 -1 -1 2 -1 1
Jitter:0
```

Figure 4 – Schedule with 0 jitter (output from the tool).

Some tenths of simulations were then made with that set. Several zero jitter solutions such as the one in figure 4 were obtained. We noticed also many solutions with a very reduced jitter and some with jitter in two or three messages. It was clear that the number of iterations was enough to arrive to a steady population, so we think that the optimization often discovers just a local minimum. However, as the time to execute is just a few seconds in a normal PC, it is quick to make a reasonable number of optimizations and it is almost sure that a good schedule is found.

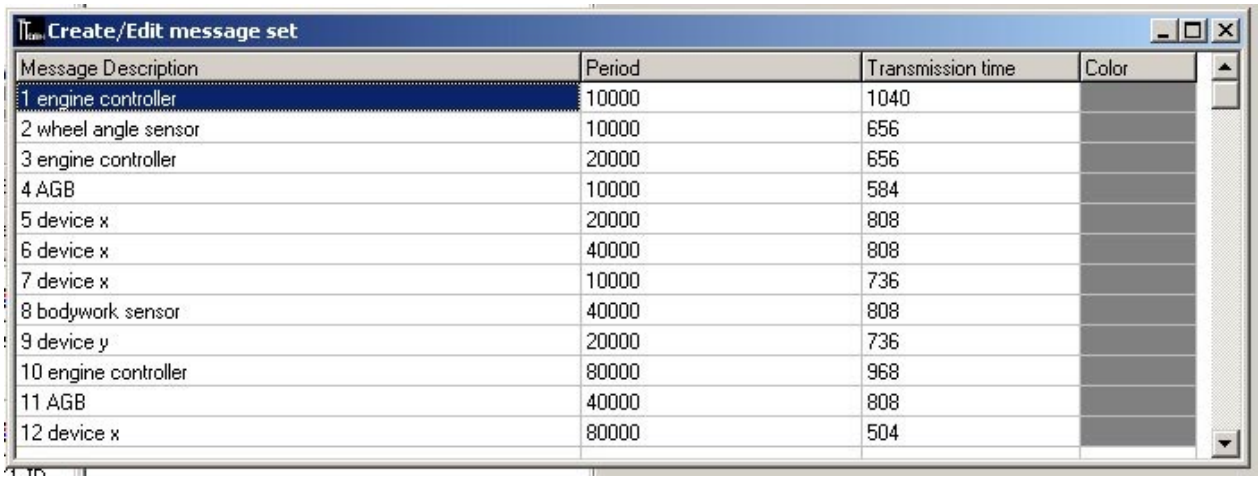

Figure 5 – View of the Create/Edit message screen with the modified PSA set.

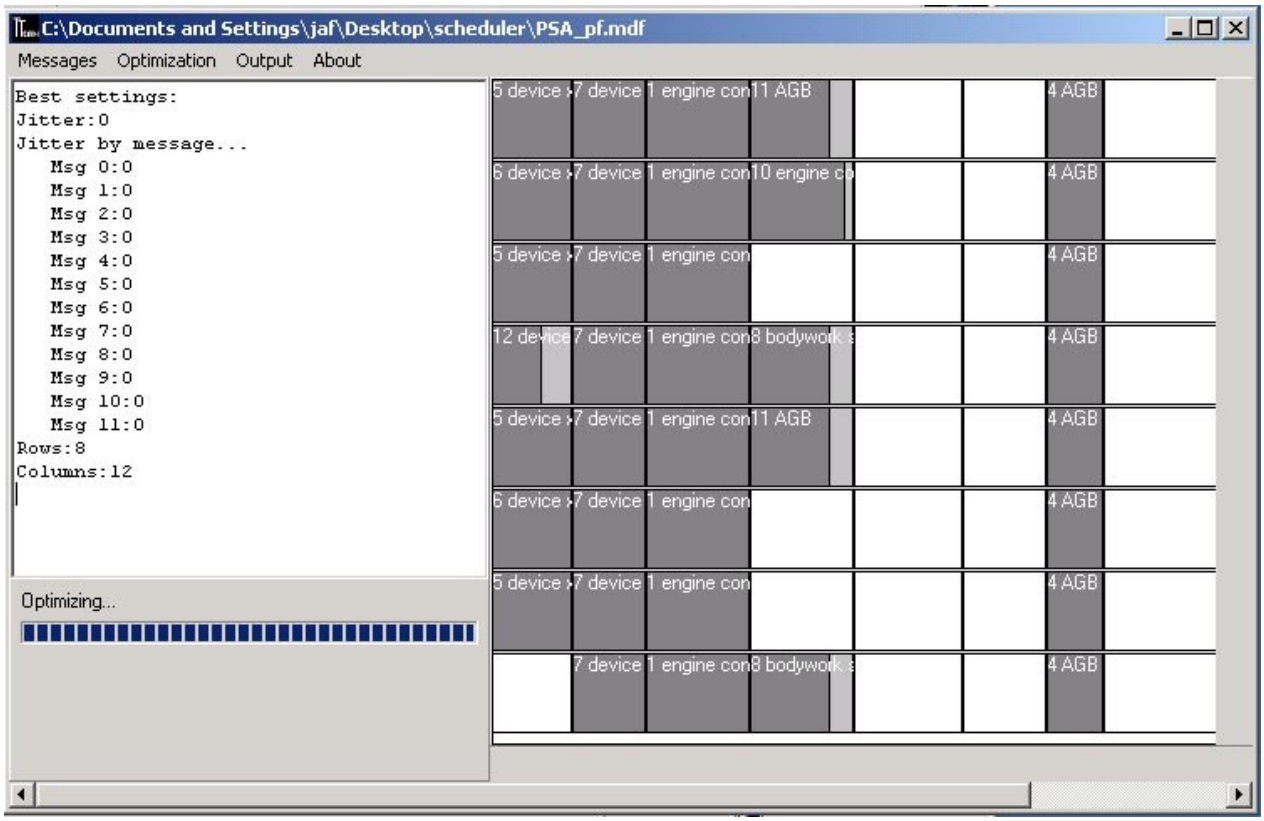

Figure 6 – Main screen of the tool showing a schedule with 0 jitter (output from the tool).

5. Conclusions

We developed a scheduling and optimization tool capable of building the so-called system matrix for a TTCAN based network given a set of periodic messages. The tool, based on a stochastic algorithm similar to a genetic one, always generates a valid scheduling solution for feasible sets. An optimization phase applies different transformations to the schedule in order to refine the solution so that message jitter can be reduced or, if possible, eliminated. The tool has been tried with SAE and PSA message sets and could find several zero jitter schedules with a modified version of this last set.

However, there is yet some research work to do on it. First, we would like to tune some initial parameters such as the percentages of the optimization transformations. Also, we would like to find more of those transformations and to study the possibility of applying a transformation similar to the crossover operation in genetic algorithms.

References

[1] Kopetz H., "Real Time Systems - Design Principles for Distributed Embedded Applications", Kluwer Academic Publishers, 1997.

[2] Tindel K., Burns A., "Guaranteeing Message Latencies on Control Area Network (CAN)", Proceedings of the ICC'94, Mainz, Germany, 1994.

[3] Navet N., Song Y.-Q., "Performance and Fault Tolerance of Real Time Applications Distributed over CAN", CiA – CAN in Automation Research Award, 1997.

[4] Führer T., et al, "Time-triggered Communication on CAN (Time-triggered CAN-TTCAN)", Proceedings of ICC'2000, Amsterdam, The Netherlands, 2000.

[5] Hartwich F., et al, "CAN Network with Time-triggered Communication", Proceedings of ICC'2000, Amsterdam, The Netherlands, 2000.

[6] Coutinho F., et al, "Using Genetic Algorithms to Reduce Jitter in Control Variables Transmitted over CAN", Proceedings of ICC'2000, Amsterdam, The Netherlands, 2000.

[7] Michalewicz Z., "Genetic Algorithms + Data Structures = Evolution Programs (3rd, revised and extended edition)", Springer-Verlag. Berlin, 1999.

[8] Almeida L., Fonseca J., Fonseca P., "A Flexible Time-Triggered Communication System Based on the Controller Area Network" Proceedings FeT '99 - Fieldbus Systems and their Applications Conference, Magdeburg, Germany, September 23-24, 1999.

[9] Coutinho F., Barreiros J., Fonseca J., "Scheduling for a TTCAN Network with a Stochastic Optimization Algorithm", proceedings of 4th FET'2001, IFAC Conference on Fieldbus Systems and their Applications, Nancy, France, November 15-16, 2001.

José A. Fonseca
DET/IEETA, Universidade of Aveiro
Campus Universitário de Santiago
P-3810-193 Aveiro – PORTUGAL
Phone: +351 234 370 330
Fax: +351 234 381 128
Email: jaf@det.ua.pt