Existing offset assignments are near optimal for an industrial AFDX network

Xiaoting Li, Jean-Luc Scharbarg, Christian Fraboul
INP-ENSEEIHT IRIT, Université de Toulouse
Toulouse, France
{*Xiaoting.Li, Jean-Luc.Scharbarg, Christian.Fraboul*}*@enseeiht.fr*

Frédéric Ridouard
LISI-ENSMA, University of Poitiers
Poitiers, France
frederic.ridouard@ensma.fr

***Abstract*—Avionics Full DupleX Switched Ethernet (AFDX) has been developed for modern aircraft such as Airbus 380. Due to the non-determinism of switching mechanism, a worst-case delay analysis of the flows entering the network is a key issue for certification reasons. Up to now most existing approaches (such as Network Calculus) consider that all the flows are asynchronous and they do not take into account the scheduling of flows generated by the same end system. It is then pessimistic to take into account such a synchronous scenario. Each end system can be considered as an offset free system, thus the main objective of this paper is to evaluate existing offset assignments in the context of an industrial AFDX network. Existing offset assignments are adapted to take into account specific characteristics of an AFDX network. Worst-case delay results are obtained according to these offset heuristics. It is shown that some existing heuristics are not efficient while some are near optimal for the studied industrial AFDX network.**

***Keywords*-AFDX network, offset assignment, worst-case delay**

I. Introduction

Avionic Full DupleX Switched Ethernet (AFDX [1]) has been proposed in order to satisfy the growing requirements of avionics application. Such a network is defined based on static network configuration and routing. The demonstration of a determined upper bound for end-to-end (ETE) communication delays on such a real-time network plays a key role. Different methods [2]–[5] have been presented for the worst-case delay analysis on the AFDX network. Among them, the Network Calculus [6] has been used for the certification of Airbus 380.

Since each end system of the AFDX network schedules its flows according to a local clock, it is pessimistic to consider that all frames arrive simultaneously (synchronous scenario) on this network. This issue has been addressed in [7], in which a computation method integrating the offsets of flows based on the Network Calculus approach has been developed. However, only one offset assignment originally designed for the CAN network in [8] was applied to an industrial AFDX network. It is interesting to consider other existing offset assignments [9], [10] in order to find the best algorithm for an industrial AFDX network. Moreover, the existing algorithms can be adapted in order to take into account specific characteristics of an AFDX network.

The objective of this paper is to evaluate and compute offset assignment algorithms for an industrial AFDX network. The goal of the evaluation is to measure the gap between offset assignment based on heuristics and the optimal assignment, which is intractable on an industrial AFDX network. An upper bound on this gap is computed, based on an optimal scenario.

This paper is organized as follows. Section II shortly introduces the context of the studied industrial AFDX network and existing offset assignments. Section III derives an ideal offset assignment which gives an optimal scenario for the scheduled flows. In Section IV, new heuristics integrating the AFDX characteristics are proposed. The existing and proposed offset assignments are applied to the industrial AFDX network, and their results are compared and analyzed in Section V. Section VI concludes and indicates directions for future research.

II. Context

A. Introduction of the industrial AFDX network

An AFDX network [1] is composed of end systems and switches. The inputs and outputs of the AFDX network, called $end\ systems\ (ES)$, are connected by several interconnected AFDX switches. Each end system can be connected to only one port of an AFDX switch and each port of an AFDX switch can be connected at most to one end system. Links between switches work in full-duplex mode.

A $Virtual\ Link\ (VL)$ standardized by ARINC-664 is a concept of virtual communication channel, which statically defines the flows. A connection defined by a Virtual Link is unidirectional, including one source end system and one or more paths leading to different destination end systems (multicast nature). A VL is characterized by:

- $Bandwidth\ Allocation\ Gap\ (BAG)$, the minimum delay between two consecutive frames of corresponding VL ranging in powers of 2 from $1\ ms$ to $128\ ms$, and
- S_{min} and S_{max}, the minimum and maximum frame length which respect the standard Ethernet frame.

An AFDX network architecture is illustrated by Figure 1. According to this architecture, there are five end systems and two AFDX switches. On the example, v_1 has a unique path $\{e_1 - S_1 - S_2 - e_4\}$ and v_5 has multi-paths $\{e_3 - S_2 - e_4\}$ and $\{e_3 - S_2 - e_5\}$.

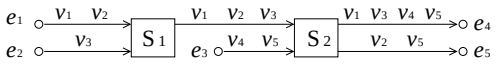

Figure 1. Example of an AFDX configuration

The industrial AFDX network interconnects aircraft functions in the avionics domain. It is composed of two redundant networks. Each network includes 123 end systems, 8 switches, 984 Virtual Links and 6412 VL paths (due to VL multicast characteristics). The left part in Table I gives the dispatching of VLs among BAGs. The right part in

BAG (ms)	Number of VL
2	20
4	40
8	78
16	142
32	229
64	220
128	255

Frame length (bytes)	Number of VL
0-150	561
151-300	202
301-600	114
601-900	57
901-1200	12
1201-1500	35
$>$ 1500	3

Table I
BAGS AND FRAME LENGTHS

Table I gives the dispatching of VLs among frame lengths, considering the maximum length S_{max}. The majority of VLs considers short frames. Table II shows the number of VL paths per length (i.e. the number of crossed switches).

Nb of crossed switches	Number of paths
1	1797
2	2787
3	1537
4	291

Table II
VL PATHS LENGTHS

This industrial AFDX network works at 100 Mb/s and the technological latency of an AFDX switch is 16 μs. The overall workload (utilization) of the industrial network is about 10%. Actually, the industrial AFDX network is lightly loaded in order to guarantee that buffers will never overflow. Both sporadic VLs and periodic VLs exist on the AFDX network, and offsets can be assigned to periodic VLs. There is no global clock in an AFDX network. Consequently, frame releases of different end systems are independent. However, each end system schedules its flows. This scheduling can be integrated in the worst-case delay analysis thanks to offsets. The next paragraph gives an overview of existing offset assignments.

B. Existing offset assignments

The offset assignment has been studied in [9] in the context of periodic task sets executed in a uniprocessor. Each task τ_i is characterized by a period T_i, a hard deadline D_i, a processing time C_i and an offset O_i. In the context of uniprocessor, the systems can be classified into three classes in terms of offset:

- Synchronous system: all the tasks have the same fixed offsets, i.e., at time 0, all the tasks generate one request;
- Asynchronous system: an offset is allocated to each task due to application constraints;
- Offset free system: any offset can be allocated to each task in order to improve the system schedulability.

For the third class, a key point is the choice of an offset assignment. The number of possible offset assignments is exponential.

In [9], an *optimal offset assignment* is proposed to exhaust all possible non-equivalent offset assignments. Although this method reduces significantly the number of combinations, the number remains exponential. *Dissimilar offset assignment*, denoted GCD, is then defined in order to reduce computational complexity in the comparison with the *optimal offset assignment* by providing a single offset assignment for a task set. This method tries to move from the synchronous case as much as possible. It considers a minimal distance $\lfloor \frac{gcd(T_i,T_j)}{2} \rfloor$ between two requests of τ_i and τ_j, where $gcd(T_i, T_j)$ is the greatest common divisor of T_i and T_j. This method treats task pairs (τ_i, τ_j) by decreasing value of $gcd(T_i, T_j)$.

Near-optimal offset assignment heuristics are derived in [10] based on the study of GCD. This assignment considers four alternative offset allocations when GCD fails to generate a schedulable asynchronous situation. Since both these two approaches assign offsets to VLs pair by pair, they are called *PairAssign* in this paper. Besides the decreasing value of $gcd(T_i, T_j)$, other heuristics are proposed considering criteria like utilization rate, i.e., $\frac{C_i}{T_i}$, and the value of $-gcd(T_i, T_j)$ to decide the order of flow pairs. These four heuristics are denoted and defined as follows:

- $RateAdd$: $\frac{C_i}{T_i} + \frac{C_j}{T_j}$,
- $RAGCD$: $(\frac{C_i}{T_i} + \frac{C_j}{T_j}) \times gcd(T_i, T_j)$,
- $RMGCD$: $max(\frac{C_i}{T_i}, \frac{C_j}{T_j}) \times gcd(T_i, T_j)$;
- $GCDMinus$: $-gcd(T_i, T_j)$;

In [8], the authors addressed that the offset assignments mentioned above are not efficient when applied to the scheduling of automotive message, and an offset assignment algorithm is tailored for automotive CAN network. This algorithm, called *SingleAssign* in this paper, aims at choosing offsets to maximize the distance between frames. For n flows emitted by one source node, sort them by increasing value of their periods and calculate $T_{max} = \max_{i \in [1,n]}\{T_i\}$. The assignments start with the flow having smallest period and process one flow after another. For a flow τ_k ($k \in [1, n]$), its offset O_k is decided as follows:

- first search for the least loaded interval in $[0, T_k)$;
- then set O_k in the middle of this interval;
- finally record all the frames of τ_k released in $[0, T_{max})$.

III. Optimal scenario of scheduled flows over the AFDX network

Considering an industrial AFDX configuration with about 1000 flows, the *optimal offset assignment* proposed in [9] is intractable. Thus approaches based on heuristics have to be used. Then, the evaluation of the gap between the *optimal offset assignment* and the assignment generated by each heuristics is an important issue. For a given flow, this gap can be defined as the difference between the worst-case ETE delays obtained by, on the one hand considering the *optimal offset assignment*, on the other hand considering the offset assignment based on a heuristic. On a whole configuration, the gap is the average of the gaps obtained for the flows. Obviously, it is not possible to compute the gap for the *optimal offset assignment* on an industrial AFDX configuration, since the *optimal offset assignment* is intractable. Then, a first idea is to compute an upper bound on this gap.

This upper bound can be obtained by considering an ideal offset assignment, which minimizes the worst-case ETE delay for all the flows. This ideal assignment may not exist for a given configuration, but it is sure that it gives worst-case delays which are not higher than the ones obtained by the *optimal offset assignment*. This ideal assignment, denoted *IdealAssign*, minimizes the maximum waiting delay of every frame in each output port it crosses. It corresponds to the following scenario:

- At its source ES, a frame f_i of a VL v_i is not delayed by any other frames emitted by the same ES, i.e., the frame f_i is transmitted immediately after its release;
- At each switch output port of its path, the frame f_i crosses VLs generated by several ESs. f_i can be delayed by exactly one frame coming from each of these ESs. The frame with the largest size S_{max} is considered. The delay encountered by f_i at each switch output port takes into account the serialization effect (i.e., two frames cannot be received at the same time from an input link, see [3] for details).

Indeed, since there is no common clock among the end systems, there is no relationship between the releases of two frames from different end systems. Consequently, there exist scenarios where the two frames arrive at their first common switch output port at the same time.

Let us illustrate this scenario on the example depicted in Figure 2. This sample network has 4 VLs v_1 and v_2 emitted by the ES e_1 as well as v_3 and v_4 emitted by the ES e_2. The network works at $100\ Mb/s$. The temporal characteristics of each VL are listed in Table III.

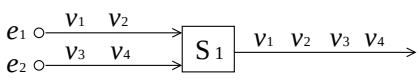

Figure 2. A sample AFDX network

v_i	BAG_i (ms)	S_{max_i} $(Byte)$	C_i (μs)
v_1	8	1000	80
v_2	8	1000	80
v_3	8	1000	80
v_4	8	500	40

Table III
The Configuration of the sample example in Figure 2

The VL v_1 is focused on. The *IdealAssign* leads to scenario illustrated in Figure 3 where the arrow represents the frame arrival of VL v_i, a_i^h is the frame arrival of v_i at the node h, and the $\boxed{\text{i}}$ means the transmission of a frame of VL v_i. At the ES e_1, the frame f_1 is transmitted as soon as it is released due to the separation from v_2. Since the ESs are not synchronized, at the output port of the switch S_1, the frame f_1 of v_1 can arrive at the same time as the frame f_3 of v_3 and it is delayed by f_3, i.e., $a_1^{S_1} = a_3^{S_1}$. Only one frame (f_3) from the ES e_2 delays the frame f_1 at the output port of S_1 since v_3 and v_4 are separated far away from each other, and f_3 is considered due to the frame size $S_{max_3} > S_{max_4}$.

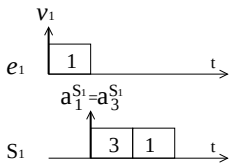

Figure 3. Scenarios of the VL v_1

The *IdealAssign* gives an upper bound on the reduction which can be obtained by an offset assignment algorithm. The next section proposes some offset assignment heuristics tailored for the AFDX network.

IV. Offset assignments in the context of AFDX network

In the context of a uniprocessor, a set of tasks shares a unique resource, i.e., the processor. The situation is different in the context of a switched Ethernet network, like the AFDX network, where a set of flows shares a set of output ports. Actually, each port is shared by a subset of all the flows. Consequently, the load can be different for each output port. The worst waiting time of a frame in an output port increases when the load of the output port increases. Then, it could be interesting to take into account the load of the output port in the offset assignment. This is illustrated in the example in Figure 4, where six VLs v_i ($i \in [1, 6]$) are transmitted over the network. The temporal characteristics of each VL are given in Table IV. The network works at $100\ Mb/s$ and the technological latency of switch is null.

The offset assignment *SingleAssign* is applied to this example network. The three VLs emitted by the ES e_1 are considered. The offsets are assigned to these three VLs in

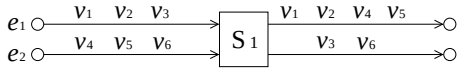

Figure 4. A small example of AFDX network

v_i	BAG_i (μs)	S_{max_i} ($Byte$)	C_i (μs)
v_1	400	500	40
v_2	800	750	60
v_3	400	750	60
v_4	400	500	40
v_5	800	750	60
v_6	400	750	60

Table IV
THE CONFIGURATION OF THE SMALL EXAMPLE IN FIGURE 4

order: $O_1 = 0\ \mu s$, $O_3 = 200\ \mu s$ and $O_2 = 100\ \mu s$. This case is drawn in part e_1 in Figure 5. Similar case at the end system e_2 is depicted in part e_2 in Figure 5. v_2 is focused on whose first frame f_2 is released at $O_2 = 100\ \mu s$. At the output port of the switch S_1 where v_2 visits, v_4 and v_5 from e_2 join the path of v_2 while v_3 has left. Then one possible scenario at this output port is depicted in part S_1 in Figure 5. It can be seen that when the frames f_1 and f_2 arrive at S_1, they are still separated far enough to avoid delaying each other. Similarly, the frames f_4 and f_5 from v_4 and v_5 are separated far enough when they arrive at S_1, consequently only one frame f_5 delays the studied frame f_2. Since the frame f_2 is released at the ES e_1 at time $O_2 = 100\ \mu s$ and the transmission of frame f_2 is finished at the switch S_1 at time $280\ \mu s$, the delay of the frame f_2 is $R_2 = 280 - 100 = 180\ \mu s$.

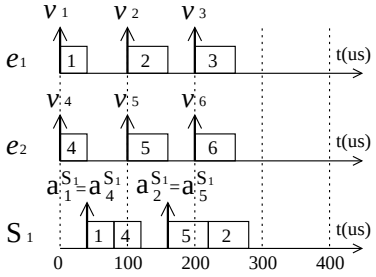

Figure 5. Illustration of the *SingleAssign* with low workload

The illustration in Figure 5 shows an example where the offset assignment *SingleAssign* succeeds to distribute the workload even in the output port of a switch. It is interesting to demonstrate the case when the workload increases. The example AFDX network in Figure 4 is under study and the maximum frame sizes of VLs v_1 and v_4 are increased to $S_{max_1} = S_{max_4} = 750\ Bytes$ ($C_1 = C_4 = 60\ \mu s$). According to the *SingleAssign*, the releases of frames at e_1 and e_2 are depicted in Figure 6 (same as in Figure 5). One possible scenario at the output port of S_1 is exhibited in part S_1 in Figure 6, where the studied frame f_2 finishes its transmission at time $300\ \mu s$. The delay of the frame f_2 is $R_2 = 300 - 100 = 200\ \mu s$, higher than the case in Figure 5 ($180\ \mu s$). It increases due to the fact that when the frame f_2 arrives at S_1 at time $a_2^{S_1} = 160\ \mu s$, the transmission of frame f_4, delayed by the transmission of frame f_1, is not completed which delays the transmission of frame f_2. For this case the *SingleAssign* could not separate frames at a crossed switch.

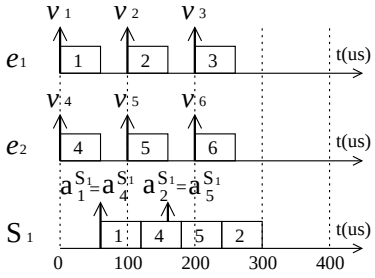

Figure 6. Illustration of the *SingleAssign* with high workload

Note that v_1, v_2 and v_3 emitted by e_1 visit three output ports: e_1 with the utilization $U_{e_1} = \sum(\frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3}) = 0.375$; the upper output port of S_1 with the utilization $U_{S_1} = \sum(\frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_4}{T_4} + \frac{C_5}{T_5}) = 0.45$; and the lower output port of S_1 with the utilization $U_{S_1'} = \sum(\frac{C_3}{T_3} + \frac{C_6}{T_6}) = 0.3$. Consequently, for these three VLs, the most loaded port is the upper output port of S_1, followed by e_1 and the lower output port of S_1. We could first assign offsets to v_1 and v_2 which visit the most loaded port of S_1, then pass to the v_3, leading to the offsets: $O_1 = 0\ \mu s$, $O_2 = 200\ \mu s$ and $O_3 = 100\ \mu s$. This case is illustrated in part e_1 in Figure 7. Similar case for the VLs emitted by e_2 is shown in part e_2 in Figure 7. Then one possible scenario for the frame f_2 at S_1 is identified in part S_1 in Figure 7, indicating that the delay of this frame is $R_2 = 380 - 200 = 180\ \mu s$, which is smaller than the one obtained by *SingleAssign* ($200\ \mu s$). The reason is that at the most loaded output port of S_1 the workload is further evenly distributed to reduce the waiting time in the buffer.

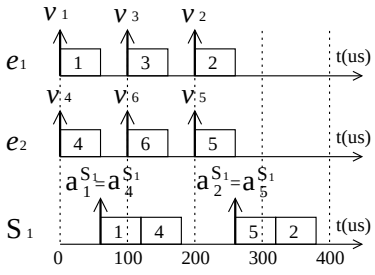

Figure 7. Illustration of the *MostLoadSA* with high workload

A proposed algorithm considers separating the VLs by

decreasing utilization of the output ports they share. The offsets are first assigned to the flows visiting the most loaded port using the assignment *SingleAssign*, then to the flows which are not yet handled in the secondly most loaded port till all the flows of one ES are assigned with offsets. This algorithm is developed based on the assignment *SingleAssign* and denoted *MostLoadSA*.

For the *PairAssign*, a similar heuristic is proposed to consider the load of the output ports. This heuristic, denoted $MostLoad$, sorts the VL pairs (v_i, v_j) by decreasing values of $Ld_i + Ld_j$, where Ld_i is the workload (utilization) of most loaded switch port crossed by v_i.

Due to the nature of the switched Ethernet, flows in one set can share several output ports. When flows share several common switches, the minimum interval between two frames decreases, which can increase the waiting time of a frame in the output port. Then the number of crossed switches can be considered in the offset assignments. For the *PairAssign*, a heuristic, denoted $CrossedS$, is proposed. It sorts the VL pairs (v_i, v_j) by decreasing values of $cs(v_i, v_j)$, where $cs(v_i, v_j)$ is the number of common switches crossed by v_i and v_j. For the *SingleAssign*, a similar heuristic, denoted *CrossedSSA*, is proposed which orders the VLs in one set by decreasing values of maximum number of crossed switch.

Besides the four new proposed heuristics, the existing offset assignment heuristics presented in Section II-B are applied to the AFDX network with the value of BAG as the period. The evaluation on each offset assignment is processed in the next section.

V. Obtained results

The existing and proposed offset assignments introduced in Section IV are applied to the industrial AFDX network presented in Section II-A. In this evaluation, all the VLs are assumed to be strictly periodic. The computation is processed using the Network Calculus approach integrating the offsets, which has been developed in [7]. The computed ETE delay upper bounds of each offset assignment are compared with those obtained from the network without offset constraints. The statistic reductions on ETE delay upper bounds of each algorithm are listed in Table V. The columns *Average*, *Max* and *Min* give the average, maximum and minimum reductions, respectively.

The *SingleAssign* as well as its extended algorithms *MostLoadSA* and *CrossedSSA* outperform the *PairAssign* heuristics. Indeed, the average reductions obtained with the *PairAssign* heuristics are 23% (*GCD*) and 32% (*RateAdd*, *RAGCD*, *RMGCD*, *GCDMinus*, *MostLoad* and *CrossedS*). It is 49% for the *SingleAssign* and 51% for the *SingleAssign* based algorithms adapted to the AFDX network. On the considered example, the *SingleAssign* based algorithms are close to the *IdealAssign*, which gives an average reduction of 53%.

Heuristics	Average %	Max %	Min %
IdealAssign	53.48	83.29	21.00
GCD	23.00	70.24	4.01
RateAdd	32.89	73.50	5.08
RAGCD	32.51	72.99	8.85
RMGCD	32.29	70.77	9.99
GCDMinus	32.95	70.06	8.83
MostLoad	32.12	70.06	8.84
CrossedS	32.32	73.03	8.90
SingleAssign	49.67	83.29	18.84
MostLoadSA	51.32	82.94	18.84
CrossedSSA	51.29	82.94	18.84

Table V
THE COMPARATIVE RESULTS

The *PairAssign* heuristics are not efficient in the studied context due to the limited different values of BAG, which lead to same values of $gcd(BAG_i, BAG_j)$ for different VL pairs. Here is a small example in Figure 8. Considering VLs v_1, v_2 and v_3 with $BAG_i = 4\ ms$ ($i \in [1, 3]$) of e_1, there are three pairs: (v_1, v_2), (v_1, v_3) and (v_2, v_3). They have the same value of $gcd(BAG_i, BAG_j) = 4\ ms$ ($1 \leq i < j \leq 3$). *GCD* leads to $O_1 = 0\ ms$, $O_2 = O_1 + \frac{gcd(BAG_1, BAG_2)}{2} = 2\ ms$, and $O_3 = O_1 + \frac{gcd(BAG_1, BAG_3)}{2} = 2\ ms$ ($O_2 = O_3$). The releases of the first frames for both v_2 and v_3 overlap, and the frames have to wait in the queue. This case is depicted in Figure 9.

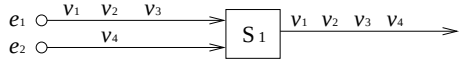

Figure 8. A small example of AFDX

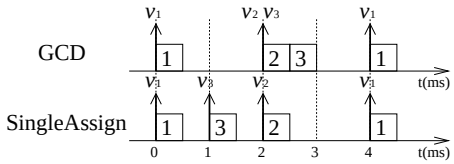

Figure 9. Comparison of GCD and SingleAssign

The situation is different when applying the offset assignment *SingleAssign* (Figure 9). With the same configuration, the offsets are set in order: $O_1 = 0\ ms$, $O_2 = 2\ ms$ and $O_3 = 1\ ms$. In this way, no frame has to wait in the output queue of e_1.

The analyzed problem of *GCD* for the industrial AFDX network exists for all the *PairAssign* heuristics because the computation of offsets mainly concerns the value of $gcd(BAG_i, BAG_j)$ even if the order of pairs varies based on different criteria.

The results are further studied by a normalized method. For one path $\mathcal{P}_x$, the computed ETE delay upper bound without offset assignment is considered as the reference

(denoted rf_x) and normalized as 100. The computed result with one offset assignment (denoted cp_x) is taken as the comparison and normalized as Ncp_x:

$$Ncp_x = 100 + (\frac{cp_x - rf_x}{rf_x} \times 100)$$

All the 6412 VL paths are sorted by increasing order of Ncp_x. Three offset assignments are taken into account: *IdealAssign*, *SingleAssign*, and *MostLoadSA*. The comparative results are presented in Figure 10.

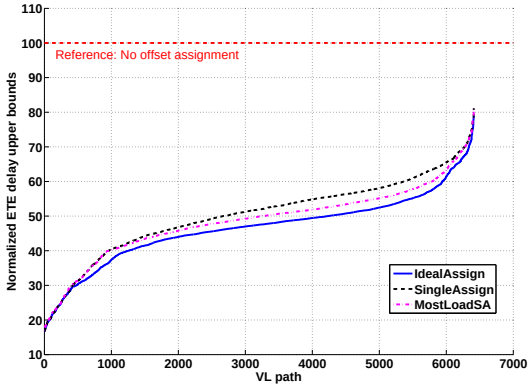

Figure 10. Comparative results of *IdealAssign*, *SingleAssign* and *MostLoadSA*

It can be seen in Figure 10 that the *MostLoadSA* curve is close to the *IdealAssign* curve, which reveals that this algorithm taking into account the AFDX properties works well on this industrial AFDX network. The gap between the *SingleAssign* curve and the *IdealAssign* curve is also small (although bigger than the gap with *MostLoadSA* curve). It suggests that a simple algorithm could be efficient to separate the flows of the industrial AFDX network.

Further evaluations have been conducted, leading to the same conclusions. They consider the same industrial AFDX architecture described in Section II-A and the overall workload 10% is kept. For each VL, the S_{min} and S_{max} are randomly chosen from 72 $bytes$ to 1526 $bytes$, and the BAG value is randomly chosen from 1 ms to 128 ms as the powers of 2. The results show that the average ETE delay reduction brought by the $IdealAssign$ is 45%. The $PairAssign$ heuristics bring average reductions ranging from 24% to 31%, which are far from the $IdealAssign$. The algorithms based on the $SingleAssign$ bring average reductions ranging from 39% to 40%, which are closer to the $IdealAssign$.

VI. Conclusion

In this paper, the offset assignments for the industrial AFDX network are studied. Since the *optimal offset assignment* is intractable in this context, an optimal scenario is built based on a presumed ideal assignment in order to upper bound the gap between the *optimal offset assignment* and each offset assignment heuristic. New heuristics considering the AFDX characteristics are proposed. Using the Network Calculus approach, the improvement on ETE delay upper bound bought by each heuristic is compared to the ideal algorithm. It is demonstrated that *PairAssign* heuristics are not efficient when applied to the industrial AFDX network due to the limited different values of BAG. The *SingleAssign* turns out a near optimal algorithm in the studied context. Although the heuristics integrating specific AFDX characteristics bring slight improvements in contrast to the *SingleAssign*, they are of increased complexity.

The industrial AFDX network considered in this paper is lightly loaded. The offset assignment for a switched Ethernet with heavier workload remains an open question, which is the subject of our ongoing work.

References

[1] *ARINC 664*, ACCE Std. 664, 2002-2008.

[2] F. Frances, C. Fraboul, and J. Grieu, "Using network calculus to optimize the AFDX network," in *Proc. 3rd Embedded Real Time Software Conference (ERTS'06)*, Toulouse, Jan. 2006.

[3] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach," *IEEE Trans. Ind. Informat.*, vol. 6, no. 4, pp. 521–533.

[4] J.-L. Scharbarg, F. Ridouard, and C. Fraboul, "A probabilistic analysis of end-to-end delays on an AFDX avionic network," *IEEE Trans. Ind. Informat.*, vol. 5, no. 1, pp. 38–49, Feb. 2009.

[5] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Method for bounding end-to-end delays on an AFDX network," in *Proc. the 18th Euromicro Conference on Real-Time Systems (ECRTS'06)*, Germany, Jul. 2006, pp. 192–202.

[6] J.-Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2001, vol. 2050, iSBN: 3-540-42184-X.

[7] X. Li, J.-L. Scharbarg, and C. Fraboul, "Improving end-to-end delay upper bounds on an AFDX network by integrating offsets in worst-case analysis," in *Proc. IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'10)*, Bilbao, Spain, Sep. 2010, pp. 1–8.

[8] M. Grenier, L. Havet, and N. Navet, "Pushing the limits of CAN - scheduling frames with offsets provides a major performance boost," in *4th European Congress on Embedded Real Time Software*, Toulouse, France, Jan. 2008.

[9] J. Goossens, "Scheduling of offset free systems," *Real-Time Systems*, vol. 24, no. 2, pp. 239–258, Mar. 2003.

[10] M.Grenier, J. Goossens, and N. Navet, "Near-optimal fixed priority preemptive scheduling of offset free systems," in *Proc. 14th International Conference on Real Time Network and Systems (RTNS'06)*, Poitiers, France, May 2006.