Using Network Calculus to optimize the AFDX network

F. Frances[1,2], C. Fraboul[2,3], J. Grieu[3]

1: ENSICA, 1 place Emile Blouin, 31056 Toulouse
2: TéSA, 14-16 port St-Etienne, 31000 Toulouse
3: ENSEEIHT/IRIT, 2 rue Charles Camichel, 31071 Toulouse Cedex 7

Abstract: This paper presents quantitative results we obtained when optimizing the setting of priorities of the AFDX traffic flows, with the objective to obtain tighter latency and queue-size deterministic bounds (those bounds are calculated by our Network Calculus tool). We first point out the fact that setting randomly the priorities gives worse bounds than using no priorities, and we then show experiments on the basis of classic optimization techniques such as a descent method and a tentative AlphaBeta-assisted brute-force approach: both of them haven't brought significantly better results. We finally present experiments based on genetic algorithms, and we show how driving these algorithms in an adequate way has allowed us to deliver a full range of priority configurations that bring tighter bounds and allow the network traffic designer to trade off average gains of 40% on all the latency bounds against focused improvement on the largest queue-size bound (up to a 30% reduction).

Keywords: AFDX, Network Calculus, optimization, genetic algorithms.

1. Overview of this paper

This paper sums up our work on the optimization of the latency and queue-size bounds of Airbus' A380's AFDX network, through the use of priorities. More precisely, it deals with the optimization problem of assigning priorities to the thousand traffic flows that are carried by this network, so that the queue-size bounds and the latency bounds calculated by our Network Calculus tool are the smallest possible. Thus, Network Calculus gives the metric that allows to compare two different priority assignments, and we present different techniques we have experimented in order to optimize the setting of priorities to the traffic flows: a variation on a Descent method, a tentative "brute-force" approach helped by Alpha-Beta reduction, and genetic algorithms.

Trying to optimize a multi-dimensional problem with a single metric can be misleading, so we have also taken benefit of the multi-criterion capabilities of genetic algorithms to demonstrate how the network traffic designer can trade-off one criterion against another.

In section 2, we present the ARINC 664's deterministic networks and the Virtual Link notion that is at the heart of the AFDX.

In section 3, we give a taste of the deterministic proof methodology we applied to the AFDX, using Network Calculus. We also mention in section 4 how we have improved the bounds calculated by our Network Calculus tool through the notion of "groups" of Virtual Links.

Then, section 5 presents the optimization problem and first attempts to address it with classic techniques. Finally, section 6 presents the results we obtained when applying genetic algorithms to this very large optimization problem.

2. The AFDX network

If Airbus hadn't chosen to have a network technology leap on the A380, the growth of communication needs between the avionics systems would have led to a tremendous amount of ARINC 429 buses [1]. By using a Full-Duplex Switched Ethernet technology for the interconnection of the essential systems (not for the truly critical systems yet, as every new technology has to prove itself first), Airbus knew that they could benefit from a huge decrease of cabling and a much more flexible connection. However, multiplexing all avionics communication flows on standard COTS Switched Ethernet would not have kept the guarantees brought by ARINC 429 buses (in terms of guaranteed bandwidth, segregation, determinism). Actually, traffic confluence inside Ethernet switches lead to variable latency, which we describe by the term of indeterminism. Moreover, without any further assumption on a limitation of the communication flows, the congestions in the switch output ports might lead to an overflow of queues, and thus to frame loss.

This is why Airbus, by committing to the process of standardizing Ethernet for aeronautical needs (namely the ARINC 664 standard), proposed "profiled networks", which are entitled to adapt the IEEE 802.3, 802.1D and "IP" (RFC 1122) standards in order to fulfil specific performance or safety needs. For instance, a subset of profiled networks, called "deterministic networks", is defined for those aircraft network domains where quality of service (including timely delivery) is paramount, and AFDX (Avionics Full-Duplex Switched Ethernet [2]) is the reference example of such networks.

ARINC 664 requires that the guaranteed service is proven; in the case of AFDX, this implies a mathematical proof firmly establishes that:

- no frame will be lost (i.e. no switch queue will overflow)
- the end-to-end delay of any frame is bounded and acceptable

As we can see, these assertions don't guarantee absolute behaviour determinism, but only a weaker form of determinism which is sufficient to bring the guaranteed service required by essential avionics systems.

Thanks to the exact bandwidth regulated traffic control of AFDX, expressed by Airbus through the notion of "Virtual Link", we have been able to develop a method for the computation of the required queue size of the switches, and of the maximum latencies suffered by the virtual links at any node of the network [3]. This VL concept is described as an example of traffic control in part #7 of the ARINC 664 [2]: actually, it can be seen as an analogy of the ARINC 429 bus, virtualized as a multicast traffic flowing through the Ethernet network. A Virtual Link is characterized by a unique identifier, one or more destination addresses, a minimal size of frames (S_{min}), a maximal size of frames (S_{max}), and a BAG (Bandwidth Allocation Gap) which defines the minimal time between transmission of two consecutive frames.

3. Determinism proof methodology

We have modelled the network switches and end-systems into elementary network entities (shapers, multiplexers, de-multiplexers, bounded-latency elements, etc.). For example, the End-System output capability is modelled this way:

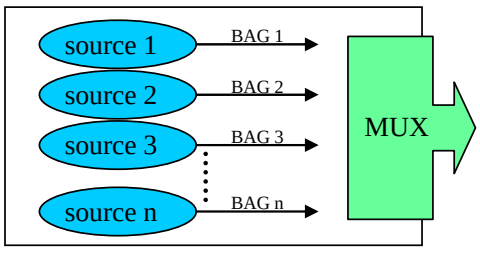

Figure 1 : End-System output model

We then use Network Calculus results [4] [5] to describe VLs by arrival curves, and further Network Calculus studies to describe the minimal service offered by network elements [6] [7] [8] [9] [10] [11]. The Calculus gives the latency bound of any elementary network entity and for those elements that have a queuing capability, a queue-size bound expressed either in a number of bits or in a number of frames (with a simple majorization using S_{min}). Given an elementary entity that offers a service curve β to an input flow constrained by an arrival curve α, the calculus also brings the arrival curve α^* of the output flow: $\alpha^* = \alpha \oslash \beta$ where $\alpha \oslash \beta$ is defined by:

$$\alpha \oslash \beta(t) = \sup_{u \geq 0} \{\alpha(t+u) - \beta(u)\} \qquad [1]$$

We developed a Network Calculus tool that propagates these results on a complete network in a dataflow way, and were thus able to compute the latency and queue-size bounds in every element of the network.

4. A Calculus optimization: the "Group" concept

In [12], we improved the Calculus of the AFDX bounds, defining "groups" of VLs that exit from the same multiplexer and enter another multiplexer together, i.e. Virtual Links that share two segments of path at least. The key issue is that the frames of those VLs are serialized once exiting the first multiplexer and thus they don't have to be serialized again in the following multiplexers. This Calculus optimization has been implemented in our tool, and it always gives tighter bounds (up to 40% better), so we never consider the simple Calculus without "groups" in this paper, and all quantitative results are rated against the bounds obtained on a network configuration with no priority assigned to the Virtual Links, but still calculated using this "group" optimization.

5. Priority Optimizations

In this paragraph, we present how we obtained better (tighter) bounds by studying different optimization techniques applied to the setting of either a low or high priority to each Virtual Link. A first important fact is that giving a priority to a Virtual Link is not supported by real-world arguments: Airbus is not willing to say some types of data are more critical than some others; all Virtual Links should provide the same guarantees. Thus, setting priorities shall only be seen as a mean to balance the load on the network: Virtual Links of "low" priority will still have to satisfy the same jitter and latency constraints. A second point to note is that splitting VLs in two low- and high-priority sets doesn't necessarily bring better bounds than having all VLs in the same high (or low) priority set (the bounds associated to this set will be used as a reference for all comparisons). With random settings of priority, we

observed that some switch ports gained a better average bound (i.e. the mean of the two bounds, one for the low-priority VLs and the other for the high-priority VLs), but other ports suffered of a degraded average bound. Moreover, trying to locally optimize the bounds in some part of the network led to overall degradation in other parts of the network.

5.1 Network model update

We changed the switch model in order to take into account two priorities. Each switch output port first transmits high-priority frames in FCFS order, and then transmits low-priority frames when all high-priority frames have been transmitted. Two queues in each output port allow implementing this static priority policy, we thus modelled output ports by multiplexers offering different service curves to the high and low priority flows. These service curves have been extensively studied, e.g. in [11]: when $R_H^*(t)$ and $R_L^*(t)$ are the arrival curves of the high and low priority flows (respectively), then the service curves offered by such a multiplexer to the high-priority flow is a β curve: $\beta_{R,T}(t)$ parameterized by $R = C$ and $T = S_{\max}^L / C$ (with $S_{\max}^L$ the maximum size of the low-priority frames, and C the output rate). And the low-priority flow is offered a service curve S_L:

$$S_L(t) = \left(C.t - R_L^*(t)\right)^+ \qquad [2]$$

From the Network Calculus tool implementation view, we had to change how groups are defined and handled: since flows of different priority are separately queued, a group can only contain flows of the same priority.

5.2 The optimization problem

Airbus' AFDX network allows assigning either a high or low priority to each VL in every switch output port. I.e. the switch configuration tables allow different priorities for a single VL in different output ports. However, it is more than likely that the same priority will be associated to a VL in every port of a given switch.

We have studied a network consisting of 1008 VLs, and restricted our study to the case a single priority is associated to a Virtual Link, i.e. a given VL always has the same priority everywhere in the network. Thus, this leaves us with "only" 1008 priorities to set; this number would approximately triple is we allowed one priority per switch for each VL. Yet, with a binary choice of low or high priority, this means a discrete optimization problem of size 2^{1008}. Of course, computing these $2^{1008} \approx 2.10^{303}$ configurations is not possible in the span of a single human life: if the bounds for each switch port of a given configuration can be calculated in one second, about 10^{295} years would be necessary for a brute-force approach. Last but not least, fixing priorities is a highly non-linear problem: changing a single VL's priority might have little or tremendous effect, and the level of this effect doesn't only depend on the considered VL. For example, the Network Calculus improvement we use (the "Group" concept [12]) is largely affected when the VL priority modification makes it exit a group of VLs and rejoin another group.

5.3 Variation on a Descent method

This classic method loops on searching a better configuration than the current one, changing only one priority setting at a time. The algorithm stops when no single-modification leads to a better configuration. The method requires a way to compare the "value" of two configurations: here we use the scalar "maximum queue-size bound" value, i.e. we take the maximum of all queue size bounds.

The descent method is known for its ability to find local optima's, but with a single priority modification at each step, it cannot exit these local optima's. We run the method a large number of times from random configurations but since we decided to reduce the time spent at each step (i.e. not to compare the 1007 single-priority modifications, which would imply spending a quarter of an hour in each step), the number of steps required to reach a local optimum was too high, because at each iterative step of the algorithm, the neighbour configuration chosen was only slightly better than the previous one, and thus we had to stop the iterative process before it reached a local optimum.

However, when starting from known "good" configurations, the method quickly converged and gave us slightly better configurations, with few priority changes from the starting configuration. Thus this method is still interesting as a "polishing" final step, when a good configuration has been obtained by another method. We think that using a true steepest descent method instead of our variation would not have brought better results; it would only have changed the time spent to reach the local optimum (i.e. more time required in each step, but less steps required).

5.4 Sorting configurations in a binary-tree and using Alpha-Beta reduction

Here we assign a sequential number to each VL (how to assign these numbers is an important issue, more on this later), so that every choice of priority leads to two branches in a tree. A full setting of priorities is now represented by a full path from the

tree root to one of the 2^{1008} leaves. A "brute force" algorithm would require comparing all these leaves: as with the previous descent method, we again use the scalar "maximum queue-size bound" metric. However, the Alpha-Beta algorithm also requires a way to give a "value" to each non-leaf node in the tree. Thus, for non-leaf nodes, we compute the queue-size bounds of a partially loaded network, i.e. a network that only carries those VLs whose priorities have been set along the incomplete path leading from the tree root to the considered node. We then build upon the fact that any node below it will have a higher value (a larger max bound), simply because the corresponding network will carry additional traffic (more VLs have their priorities set). This allows Alpha-Beta to cut large branches of the tree. However, we knew that Alpha-Beta doesn't cut enough branches to allow a complete exploration of the tree in an acceptable time. So, we used Alpha-Beta as an algorithm to explore "small" modifications of priorities, starting from an already "good" configuration (this starting configuration was heuristically build, see paragraph 6.3).

As stated earlier, the order of VLs down the tree is of primary importance: if the last VLs (those near the leaves) are such that changing their priorities has little impact, then the algorithm searches better configuration by trying priority changes of these VLs, and the gain is small. On the contrary, if the last VLs are such that their priorities greatly impact the calculated bounds, then the algorithm cut large branches (changing the priority of one of these VLs is quickly identified as giving worse bounds), so that it can quickly state that the setting of these last VLs is optimal, but it is likely it will not bring a better configuration.

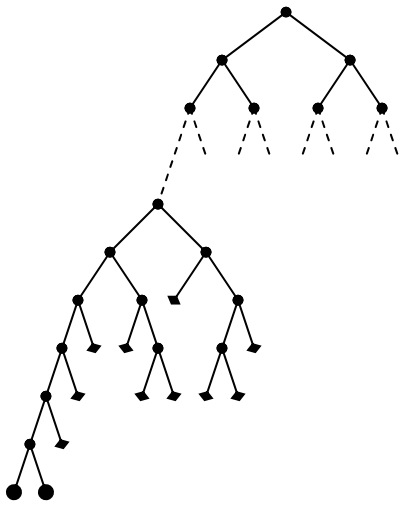

Figure 2: Alpha-Beta cuts many branches
(example figure shows cuts with diamonds)

However, it is difficult to guess if a VL's priority will largely impact the bounds (its impact also depends on the priorities of the other VLs) and thus it is difficult to sort them in "ascending" or "descending" order of influence.

In practice, we tried both cases: in a first run, we sorted the VLs in ascending S_{min} order (S_{min} is an important factor in the calculus of bounds on the number of frames). The algorithm found a better configuration than the starting one, by changing the 45 last priorities, but then it didn't find any further improvement (we stopped it after the evaluation of 5 millions of configurations). In a second run, we sorted the VLs in descending S_{min} order, and the algorithm didn't find a better configuration despite evaluating the priority change of the 104 last VLs (thanks to large branch-cuts).

In conclusion, we found that Alpha-Beta is an easily implementable variation of brute-force, but reduction is not enough to explore the full 2^{1008}–size tree. However, it allows gaining confidence that a good priority configuration is a local optimum, at least for a subset (a hundred at most) of chosen VLs: contrary to the previous method, this exhaustive exploration of a subset of priorities allows to exit from a local optimum to reach a better configuration.

6. Optimizing with Genetic Algorithms

Evolutionary algorithms [14][15][16][17] are renowned for their ability to handle very large optimization problems. Among them are Genetic Algorithms ([18][19][20]): these algorithms represent points in the optimization space by individuals, themselves represented by genes, and iterate generations of a population with reproduction and "natural" selection.

Many evolution algorithms are available, but most of them are dedicated to a specific application. This is why we chose the PISA architecture [21][22][23], which defines a simple interface that allows coupling an application field with an evolution algorithm.

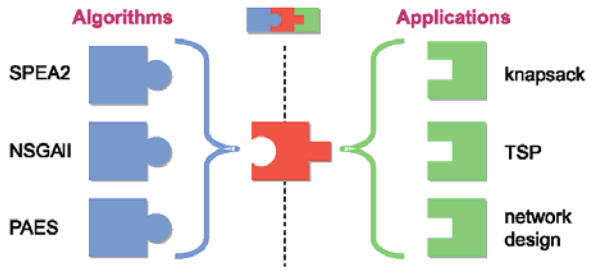

Figure 3: PISA architure

We were thus able to interface our specific Network Calculus tool to SPEA2 [24], which is one of the best multi-criterion algorithms according to [25]. SPEA2

evaluates the quality of an individual j by $Sum_i\{S(i)\}$ for all i in D(j), where D(j) is the set of individuals who dominate j (i.e. who are better than j on all criterions), and S(i) is the number of individuals in the whole population who are dominated by i.

The overall algorithm of SPEA2 is given by the following six steps:

Step #1: Initialization, setting of the initial population (P_0) and archive initially empty (A_0).

Step #2: Evaluation of the quality of each individual in P_t and A_t

Step #3: Selection: all those individuals from P_t and A_t who are not dominated are copied into A_{t+1}. The archive is either completed with dominated individuals or reduced, in order to always be of size N (nearly identical individuals are removed first if reduction is needed).

Step #4: Ending: if an end-condition is met or if the maximum number of generation (T) is reached, the algorithm ends. The result is given by the non-dominated individuals of A_{t+1}.

Step #5: Selection for reproduction. A set of parents is elected by successive tournaments in A_{t+1}.

Step #6: Variation: recombination and mutation operators are applied on the chosen set of parents. The resulting individuals make P_{t+1}. Then t is incremented and the algorithm loops back to step #2.

Of course we had to define how genetics rule our network priority world:

- An individual represents a priority configuration, i.e. a setting of priorities for all Virtual Links. The DNA sequence of an individual is a 1008-characters long string. The character located at position i reflects the priority of VLi and is either '1' for high-priority or '2' for low-priority.
- Each individual requires a full execution of our Network Calculus tool, in order to compute the queue size bounds and the delay bounds of all network multiplexers. Two criterions are used to identify dominated individuals: the largest queue-size bound and the average delay bound.
- The mutation operator modifies 2% of an individual's DNA, randomly chosen.
- Reproduction consists in concatenating the right part of a parent's DNA to the left part of the other parent's DNA: the cut point is also randomly chosen.

Finally, there are a number of global parameters for the algorithm that have been set as follows:

- Size of the initial population and of the archive: α=1000
- Number of individuals selected for reproduction: μ=100
- Number of children at each generation: λ=10
- Number of individuals in each tournament: P=3

6.1 Results obtained with a single evaluation criterion

We first tried the algorithm with a single evaluation criterion, the largest queue-size bound (i.e. the largest of all the queue-size bounds computed by the Network Calculus tool, expressed in number of frames). Figure 4 shows that at generation #10000, the population had converged to individuals having a largest queue-size bound of 727, thus bringing a 6% optimization over the reference configuration (the one with all priorities set at the same level).

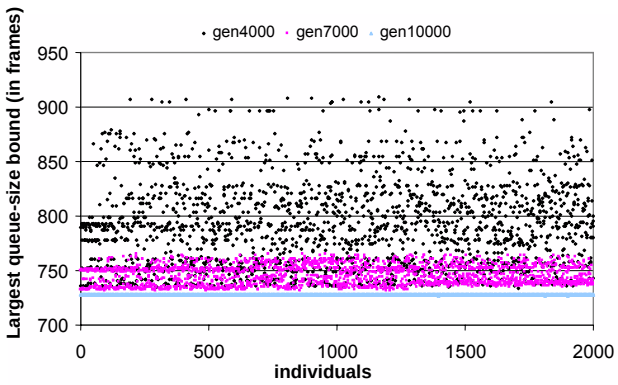

Figure 4: Generation convergence with a single evaluation criterion

However, as we didn't want that this optimization could be done at the expense of many latency bounds, we decided to exploit the multi-criterion capability of SPEA2 and had it search configurations that minimize both the largest queue-size bound and an average of the latency bounds (each latency bound being weighted by the number of VLs that suffer of this latency bound).

6.2 Bi-criterion optimization with a random initial population

Figure 5 shows how the initial random population, the 1000th generation and the 7000th generation performed on the two selected criterions: average delay bound (horizontally) and largest queue-size bound (in number of frames, vertically). One can see that the algorithm converges to a better population, both horizontally and vertically. The figure gives a good understanding of the Pareto front: points that

are not dominated on both criterions. Thus it gives an idea of the trade-off between the two criterions.

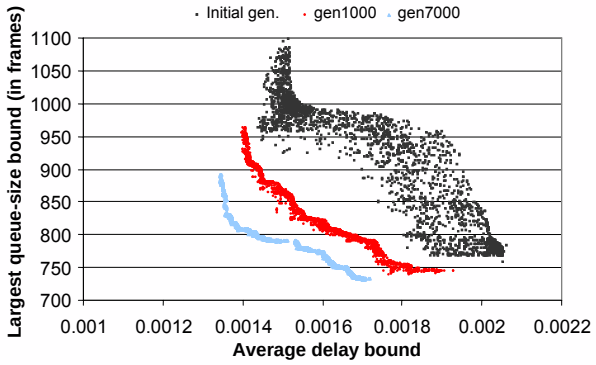

Figure 5: Bi-criterion convergence using a random initial population

However, one can notice that despite a large optimization over the initial random population, the optimization is at most only 5% over the reference configuration. We concluded that the optimization problem is of such a big size that neither the random initial population nor the random mutations can cover enough diversity, and that the whole population converges towards “attraction zones”, despite the diversity of the individuals inside a single generation.

We thus increased the mutation probability and the impact of these mutations, along with a larger number of children per generation. Figure 6 shows the results obtained with these larger parameters. Unfortunately, the results are not better: it appears that these parameters “boost” the algorithm, in the sense it converges faster, but the individuals of the last generations are not better than those obtained with the first parameters.

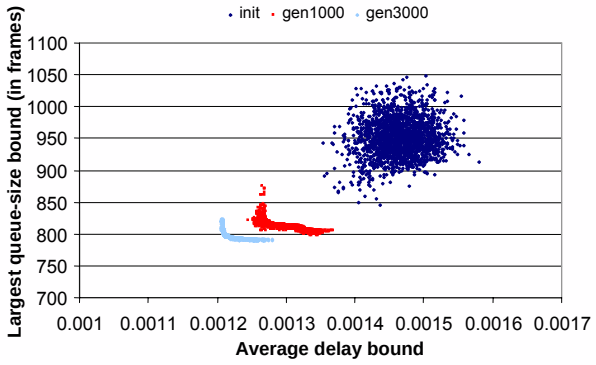

Figure 6: Evolution with bigger diversity

6.3 Results with a “heuristically-good” initial population

Our intuition is that small frames are better assigned a high priority, so that large frames do not delay small frames, which would result in a bigger number of delayed frames and thus a larger queue-size (in terms of number of queued frames). The deterministic calculus of the queue-size bounds indeed shows the influence of the S_{min} parameter (minimum size of frames for a given VL): the bound is reciprocally proportional to the minimum of all S_{min} of a given priority. Thus we ordered the VLs in ascending S_{min} order and populated the initial set of individuals with 1008 configurations ranging from the reference one (all VLs set to low-priority) to the equivalent final one (all VLs set to high-priority), each configuration having the next VL in this order set to high-priority. The 2000-sized population was completed with random configurations.

Figure 7 shows the evolution of the generations obtained with this partially-chosen initial population. Clearly, the good properties of a few good initial individuals are propagated to the whole population. The largest queue-size bound is 25% more optimized than the no-priority setting.

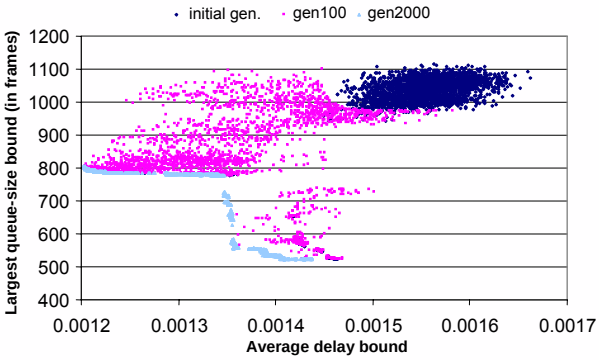

Figure 7: Evolution with a heuristically-good initial population

However, in order to better evaluate the added-value of the genetic algorithm, we show the following figure (Figure 8), where only the best individuals of the initial population are depicted, along with the complete last generation.

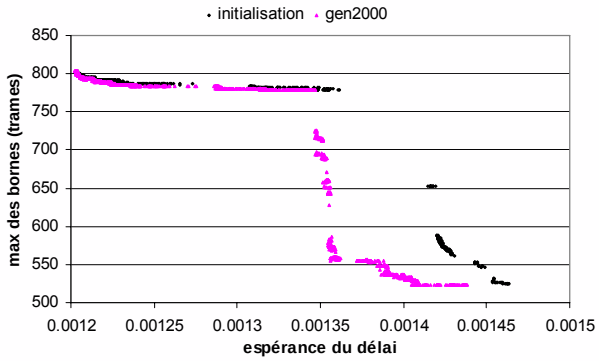

Figure 8: Added-value of genetic algorithm

This exhibits two groups of configurations: the first group consists of configurations whose average delay bound is less than 0.00135, and a largest queue-size bound ranging from 780 to 800 frames. At one end of this group, the network integrator can easily gain 41% on the average delay bound (compared to the no-priority configuration) without sacrificing much to the largest queue-size bound (only 4%). One can see that this group contains many of the initial heuristic configurations: this shows that the algorithm didn't found really better configurations.

The second group of configurations contains those configurations that are better for the largest queue-size bound criterion (here we can see configurations that have a 31% gain on the largest queue-size bound compared to the reference configuration, and still also optimize the average delay bound by another 30%). Although the algorithm didn't find a really better configuration for the largest queue-size bound criterion, it found configurations with equal value on this criterion and better value on the average delay bound.

7. Conclusion

Optimizing the deterministic bounds calculated on the AFDX network is a very large optimization problem, offering the opportunity to several research directions.

One could try to bring tighter Network Calculus results, but many of the theoretical results we used have already been proved as being optimal, i.e. the bounds can be reached. So, we think there is very little gain to hope from the Network Calculus theory itself. However, there surely are several ways to apply Network Calculus to calculate the bounds of the AFDX network. In [12], we have gained up to 40% on the bounds, by aggregating segments of flows into groups.

In this paper, we have presented another research direction that aims to get benefit from the static priority capability of the AFDX switches. This has turned to a priority setting optimization problem: how to choose either a high or low priority for each of the traffic flows. We have shown that this discrete optimization problem is so large that brute-force cannot handle it (solving it would require about 10^{295} years). We have tried classic optimization techniques such as a variation on a descent method and an AlphaBeta search, but this has brought little gain, so we consider these methods can only slightly improve an already good priority configuration.

We have also extensively used a Genetic Algorithm to search priority configurations for which our Network Calculus tool calculates tighter bounds, both from a global perspective (all the bounds are reduced in average) and from a focused perspective (the largest bound decreases too). Moreover, the multi-criterion capability of SPEA2 shows the Pareto front of our problem, and thus provides a full range of good priority configurations that help the network traffic designer to trade off one criterion against the other. On our traffic case study, we have thus been able to offer up to 41% gains on the average delay bounds, and up to 31% on the largest queue-size bound. Needless to say, these gains increase the scalability of AFDX, as they raise the potential for future additional traffic.

8. References

[1] Aeronautical Radio Inc., ARINC specification 429-ALL: Mark 33 Digital Information Transfer System (DITS) Parts 1, 2, 3, 2001.

[2] Aeronautical Radio Inc., ARINC specification 664P7, Aircraft Data Network, Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network, 2005.

[3] C. Fraboul, F. Frances, "Applicability of Network Calculus to the AFDX", contract report PBAR-JD-728.0821/2002.

[4] R. Cruz, "A calculus for network delay, Part I: Network element in isolation," IEEE Trans. Inform. Theory, vol. 37, no. 1, pp. 114-131, Jan. 1991.

[5] R. Cruz, "A calculus for network delay, Part II: Network analysis," IEEE Trans. Inform. Theory, vol. 37, no. 1, pp. 132-141, Jan. 1991.

[6] R. L. Cruz, "Quality of Service Guarantees in Virtual Circuit Switched Networks," *IEEE Journal of Selected Areas in Communication*, special issue on "Advances in the Fundamentals of Networking" (vol. 13 no. 6), August, 1995

[7] H. Sariowan, R. L. Cruz, and G. C. Polyzos, "Scheduling for Quality of Service Guarantees via Service Curves," Proceedings of the

International Conference on Computer Communications and Networks (ICCCN) 1995, Las Vegas, September 20-23, 1995, pp. 512-520.

[8] C.S. Chang, "On deterministic traffic regulation and service guarantee: A systematic approach by filtering", IEEE TIT vol 44, May 98, pp 913--931.

[9] R. Agrawal, R. L. Cruz, C. Okino and R. Rajan, " Performance Bounds for Flow Control Protocols", IEEE ToN vol 7, No3, June 99, pp 310--323.

[10] J.-Y. Le Boudec, "Application of network calculus to guaranteed service networks," IEEE Transactions on Information Theory, vol. 44, pp. 1087--1096, May 1998.

[11] J.-Y. Le Boudec and P. Thiran, "Network Calculus", Springer Verlag Lecture Notes in Computer Science volume 2050.

[12] J. Grieu, F. Frances, C. Fraboul, "Preuve de déterminisme d'un réseau embarqué avionique", CFIP (Colloque Francophone sur l'Ingénierie des Protocoles), October 2003.

[14] R. M. Friedberg, B. Dunham, and J. H. North, "A learning machine:Part II," *IBM J.,* vol. 3, no. 7, pp. 282–287, July 1959.

[15] G. E. P. Box, "Evolutionary operation: A method for increasing industrial productivity," *Appl. Statistics,* vol. VI, no. 2, pp. 81–101, 1957.

[16] J. H. Holland, "Outline for a logical theory of adaptive systems," *J.Assoc. Comput. Mach.,* vol. 3, pp. 297–314, 1962.

[17] J. H. Holland, "Adaptation in Natural and Artificial Systems". Ann Arbor, MI: Univ. of Michigan Press, 1975.

[18] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," in *Pattern-Directed Inference Systems,* D. A. Waterman and F. Hayes-Roth, Eds. New York: Academic, 1978.

[19] L. J. Fogel, "Autonomous automata," *Ind. Res.,* vol. 4, pp. 14–19, 1962.

[20] L. J. Fogel, "On the organization of intellect," Ph.D. dissertation, University of California, Los Angeles, 1964.

[21] S.H.Lu, P.R.Kumar, "Distributed Scheduling Based on Due Dates and Buffer Priorities", IEEE Transactions on Automatic Control, Vol. 36, n. 12, pp. 1406-1416, December 1991.

[22] Parekh, A. K., & Gallager, R. G., "A Generalized Processor Sharing Approach to flow control in Integrated Services Networks - The Single Node Case," IEEE/ACM Transactions on Networking, Volume 1 #3, pp 344-357, June 1993.

[23] H. Zhang, "Providing End-to-End Performance Guarantees Using Non-Work-Conserving Disciplines", Computer Communications: Special Issue on System Support for Multimedia Computing, volume 18, 10 October 1995.

[24] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks", Proc. IEEE, vol. 84, pp. 1374-1396, Oct. 1996

[25] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: "Improving the strength pareto evolutionary algorithm for multiobjective optimization". In K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, and T. Fogarty, editors, Evolutionary Methods for Design, Optimisation, and Control, pages 19--26, Barcelona, Spain, 2002. CIMNE.

8. Glossary

AFDX	Avionics Full-Duplex Switched Ethernet
DNA	Deoxyribonucleic Acid
FCFS	First Come First Served
ERTS	Embedded Real Time Software
PISA	a Platform and programming language independent Interface for Search Algorithms
SPEA	Strength Pareto Evolutionary Algorithm
VL	Virtual Link