Genetic Algorithm Based ARINC 664 Mixed Criticality Optimization Using Network Calculus

Eyüp Can Akpolat1,2, Ömer Faruk Gemici3, M. Selim Demir3, İbrahim Hökelek1,3, Sinem Coleri4, and Hakan Ali Çırpan1

1Department of Telecommunication Engineering, Istanbul Technical University, Istanbul, Turkey
2Turkish Aerospace Industries Inc., Istanbul, Turkey
3Research Center for Advanced Technologies on Informatics and Information Security, TUBITAK BILGEM, Kocaeli, Turkey
4Department of Electrical and Electronics Engineering, Koc University, Istanbul, Turkey

Abstract—ARINC 664 is an Ethernet based deterministic networking standard providing data transmission with bounded delays among avionics sub-systems. This paper presents a Genetic Algorithm (GA) based ARINC 664 network delay optimization using the network calculus (NC), where the GA is used to effectively search the mapping of Virtual Links (VLs) to priority levels using the extended priority scheme. While there are only two priority levels in the ARINC 664 standard, the extended priority concept increases the number of priority levels to improve the schedulability of VLs. For each possible assignment of the VLs to the priority levels, the NC analysis provides the worst-case delay results for all VLs. We define three different fitness functions aiming to minimize the maximum, the average, and the standard deviation of the worst-case VL delays, respectively. The results demonstrate that the extended priority concept improves the schedulability of VLs and the GA optimization approach can successfully achieve the desired objectives for the VL delays if the appropriate cost function is selected.

Index Terms—deterministic network, ARINC 664, network calculus, genetic algorithm, mixed criticality

I. INTRODUCTION

The Integrated Modular Avionics (IMA) is characterized by common hardware and software modules hosting multiple avionics applications [1], [2]. The amount of exchanged data among the IMA systems becomes enormous as the number of integrated functions within an aircraft increases. Since such a growth in the number of communicating nodes cannot be handled properly by traditional avionics communication standards such as MIL-STD 1553 and ARINC 429 [3], the Ethernet based deterministic networking technologies, namely ARINC 664 [4], Time Triggered Ethernet (TTEthernet) [5], and IEEE Time Sensitive Networking (TSN) [6], have been developed.

ARINC 664 is a deterministic data network, where the bandwidth is shared among the communicating nodes to provide guaranteed Quality of Service (QoS). End Systems (ES) exchange data frames through unidirectional connection from one source to one or more destinations using Virtual Links (VLs). The desired bandwidth resource for each VL is allocated by setting two parameters, namely Bandwidth Allocation Gap (BAG) and the maximum frame length \( L_{\text{max}} \). Traffic shaping at the transmitting ES regulates the outgoing traffic according to the VL profile. The scheduler plays an important role to determine the delay and jitter bounds in ES with multiple VLs by deciding the order of transmission for incoming regulated flows. Each ARINC 664 switch is responsible for switching, filtering, and policing to make sure that only valid incoming frames are forwarded to the correct physical ports. A full duplex switched Ethernet avoids message collisions but the conflict may still arise when the messages compete for the same resource at the same time resulting in delay variations. Traditionally, the scheduler with high and low priority classes are utilized to determine the delay and jitter bounds for the VLs passing through a specific output port of the switch.

ARINC 664 has been widely used due to its lower complexity; however, its mixed criticality support is limited since it is an asynchronous protocol and defines only two priority classes (i.e., high priority - HP and low priority - LP). TTEthernet and IEEE TSN provide better support for mixed time criticality applications sharing a single physical network since they utilize time synchronization information for determining transmission schedules. The ARINC 664 standard does not specify the details of the scheduler algorithm which is an active research topic. In this study, we extend the priority scheduler concept in ARINC 664 such that there are multiple priority levels within the HP and LP classes. This priority extension concept significantly improves the schedulability of VLs to achieve lower delay and jitter bounds for higher priority VLs; and hence, the ARINC 664 with the extended priority scheduler results in better mixed criticality support.

The offline network planning is mandatory for ARINC 664 to provide the desired delay and jitter bounds for the VLs. For a given network with the communication needs of applications, the number of VLs together with the associated parameters (e.g., routing path, BAG and \( L_{\text{max}} \)) for each VL should be specified by making sure that the rate, delay and jitter requirements of the applications are satisfied [7], [8]. The ARINC 664 network planning requires the calculation of the worst-case end to end delays for all VLs, which has been traditionally performed using two alternative methods, namely Network Calculus (NC) [9] or Trajectory Approach [10]. The
foundation of network calculus lies in the mathematical theory of min-plus algebra and it provides deep insights into the delay analysis problems in networking such as integrated services networks [11], flow control, scheduling, and delay dimensioning. NC is powerful approach to obtain the worst-case delay bounds of avionics network including ARINC 664. In [12], the Genetic Algorithm (GA) is utilized to calculate the maximum frame length that optimizes the worst-case end to end delay using the NC analysis for the ARINC 664 network with cascaded switches. In another study, the priority levels of VLs are determined either HP or LP to minimize the worst-case end to end delay [13].

In this study, we first introduce the extended priority concept by increasing the number of priority levels within the HP and LP classes to improve the schedulability of VLs for ARINC 664 network. Secondly, we propose a GA based ARINC 664 network optimization using the NC, where the GA is used to effectively search the mapping of VLs to the extended priority levels. For each possible assignment of the VLs to the priority levels, the NC analysis provides the worst-case delay results for all VLs. We define three different fitness functions aiming to minimize the maximum, the average, and the standard deviation of the worst-case delays. The results demonstrate that the GA optimization approach can successfully achieve the desired objective when the corresponding fitness function is selected.

The remainder of this paper is organized as follows. The system model and the delay analysis using the network calculus are presented in Section II. The genetic algorithm based delay optimization and the experiment results are described in Sections III and IV, respectively. Finally, Section V concludes the paper.

II. SYSTEM MODEL AND NETWORK CALCULUS DELAY ANALYSIS

In this section, we present the proposed system model as depicted in Fig. 1 and the corresponding network calculus worst case delay analysis. The system model is generic to support any number of priority levels; however, for the sake of presentation, the system model is described using eight priority levels. The flows in the HP class (group) \( G_{HP} \) can be assigned into four levels from P1 to P4 and the flows in the LP class (group) \( G_{LP} \) can be assigned into four levels from P5 to P8. The strict priority scheduler resolves these eight queues such that their priority levels decrease from P1 to P8, where P1 has the highest and P8 has the lowest priority.

Let \( N \) represent the number of flows \( f_1, f_2, \ldots, f_N \) passing through an output port of ARINC 664 ES or switch. We assume that there is one to one mapping from a flow \( f_n \) to \( VL_n \) such that \( L_{max,n} \) and \( BAG_n \) represent the maximum frame size and bandwidth allocation gap of the \( VL_n \), respectively. The regulated flow \( f_n \) can be represented by the leaky bucket arrival curve \( a_n(t) = r_n t + s_n \), where \( s_n = L_{max,n} + 160 \) shows the maximum frame size in bits and \( r_n = s_n / BAG_n \) indicates the arrival rate.

Flows are ordered from the highest to the lowest priority levels. For an ARINC 664 node (i.e., ES or switch), the flow list that shares the same output port with \( f_n \) can be represented as \( f_1, f_2, \ldots, f_{n-k}, f_{n-k+1}, \ldots, f_{n-1}, f_{n+1}, \ldots, f_{n+m}, f_{n+m+1}, \ldots, f_N \), where \( I_{hp}^n \) and \( I_{lp}^n \) represent the flow lists having higher and lower priority levels than \( f_n \), \( I_{hp}^n \) represents the flow list having the same priority level with \( f_n \). Considering the priority levels, the NC delay bounds for each flow can be calculated by adding the delay of the transmitting ES (\( D_{ES}^{f_n} \)) and the delay of the switches (\( D_{SW_k}^{f_n} \)) through which the flow passes:

\[
D_{n} = D_{ES}^{f_n} + \sum_{k \in SW/f_n} D_{SW_k}^{f_n}
\]

where \( SW/f_n \) represents the switch list that \( f_n \) passes.

A. ES Delay Analysis

At the transmitting ES, the worst-case delay of the \( n^{th} \) flow (i.e., virtual link) \( f_n \) supporting multi-level priority can be calculated as:

\[
D_{ES}^{f_n} = \frac{\gamma \max(s_i) + s_n + \sum_{i \in I_{hp}^n} s_i}{R} + \sum_{i \in I_{lp}^n} \frac{s_i}{R}\left[\frac{BAG_n}{BAG_i}\right]
\]

where \( R \) shows the ES link capacity and \( [\cdot]\) is the ceiling operator. Here, \( \max(s_i) \) represents the maximum \( s_i \) value of the flows having lower priority than \( f_n \). The variable \( \gamma \) indicates whether the preemption is enabled or disabled in the network configuration such that it is 0 when the preemption is enabled and 1 when the preemption is disabled. Note that, for each flow \( f_n \), the worst-case delay value is expected to be smaller than or equal to \( BAG_n \).
B. Switch Delay Analysis

In this subsection, we provide the NC delay analysis for multiple flows crossing a switch as depicted in Fig. 2. The delay analysis of each flow requires the residual service curve information. There are two different approaches to calculate the residual service curve, namely first in first out (FIFO) and strict service element [14]. For flows having the same priority levels, the FIFO assumption is used, while for flows with different priority levels, the strict service assumption is used.

The service curve for each flow guaranteeing the individual remaining service capability is defined as the residual service curve. At the $k^{th}$ switch, the service curve of an output port can be represented by $\beta_k(t) = R(t-T)^+$, where $R$ is the service capacity and $T$ is the technological latency of the switch. The residual service curve $\beta_n(t)$ can be calculated as the remaining service capability of $f_n$ after subtracting the other flows’ arrival curves from the total service curve as

$$\beta_n(t) = \left[ \beta(t) - \sum_{i \in I_{k,n}^h} (\alpha_i(t)) \right]^+$$

The residual service curve for $f_n$ can be represented as $\beta_k(t) = R_k(t-T_k)^+$ at the switch $k$. Here, $R_k$ and $T_k$ represent the residual service capacity and the technological latency of $f_n$ at the $k^{th}$ switch. Considering the multiple priority levels, $\beta_n(t)$ can be expressed as:

$$\beta_n(t) = \left( R - \sum_{i \in (I_{k,n}^h \cup I_{k,n}^p)} r_i \right) \left[ t - \left( RT + \frac{\sum_{i \in (I_{k,n}^h \cup I_{k,n}^p)} s_i + \gamma \max_i (s_i)}{R - \sum_{i \in I_{k,n}^h} r_i} \right) \right]^+$$

The worst-case delay for $f_n$ at the $k^{th}$ switch output port ($D_{SW_k}^{f_n}$) can be calculated by obtaining the maximum horizontal distance between the residual service curve $\beta_n(t)$ and the arrival curve $a_n(t)$ [14]:

$$D_{SW_k}^{f_n} = \frac{RT + \sum_{i \in (I_{k,n}^h \cup I_{k,n}^p)} s_i + \gamma \max_i (s_i)}{R - \sum_{i \in I_{k,n}^h} r_i}$$

When $f_n$ crossed a switch, the arrival curve is modified and can be calculated as $a_n^*(t) = a_n(t) \odot \beta_n(t)$, where $\odot$ represents the min-plus deconvolution operator [14]. Note that the modified arrival curve $a_n^*(t)$ is used for the delay analysis of the next switch and can be given as:

$$a_n^*(t) = r_n t + s_n + r_n \frac{RT + \sum_{i \in (I_{k,n}^h \cup I_{k,n}^p)} s_i + \max_i (s_i)}{R - \sum_{i \in I_{k,n}^h} r_i}$$

C. Improved NC Bounds for Tandem Switches

When a flow travels through multiple cascaded switches, the “Pay The Burst Only Once” (PBOO) phenomenon is used in the delay bound analysis [14]. The PBOO phenomenon suggests that if the first switch introduces a delay, it also smooths the arrivals to the second switch. Thus, if a burst is delayed at the first switch, when the last bit of the data of this burst arrives in the second switch, the rest of the burst has already started being processed by the second switch. As a result, the last bit of data does not have to pay again the delay induced by the burst. Therefore, the network topology given in Fig. 3 can be simplified as in Fig. 4.

A network is generally composed of several flows and multiple switches. The multiple flows sharing the same switch output port can be transmitted through different output ports of the next switch. In addition, in the next switch, there can be some other flows to be transmitted at the same output ports. In that case, the network calculus delay analysis requires the analysis of each path of the flows. For each flow, the individual service curve and modified arrival curve at each switch output port needs to calculated considering the service element type, priority and preemption assumptions. Delay bound of a given flow can be determined by combining the residual service curve ($\beta_n,C(t)$) of each crossed switches as:

$$\beta_n,C(t) = \beta_{n1}^1(t) + \beta_{n2}^2(t) + \cdots$$

where $\odot$ represents the min-plus convolution operator. $\beta_n,C(t)$ can be shown as $\beta_n,C(t) = R_{n,C}(t-T_{n,C})^+$ where $R_{n,C} = \min_{k \in SW_f} (R_k)$ and $T_{n,C} = \sum_{k \in SW_f} T_k$. 

III. DELAY OPTIMIZATION USING GENETIC ALGORITHM

The optimum value of priority selection can be obtained by evaluating all possible priority combinations of VLs in the ARINC 664 network using exhaustive search. However, the exhaustive search method is prohibitively expensive when the number of VLs is significantly high. Let $N_{G_{HP}}$ and $N_{G_{LP}}$ represent the number of HP and LP VLs in the network. Considering 4 additional priority levels for each priority class (i.e., HP and LP), there are $4^{N_{G_{HP}}} + 4^{N_{G_{LP}}}$ possible solutions to assign priority levels. Therefore, such a brute-force method is computationally prohibitive for most practical purposes. Here, we use genetic algorithm (GA), a bio-inspired population-based powerful meta-heuristic algorithm [15] to efficiently explore the large search spaces. In this section, we describe how the GA can be utilized to find the priority configurations for a given ARINC 664 network.

A. Chromosome Structure and Initialization

The chromosome structure of the GA consists of priority levels of VLs for the HP class, $G_{HP}$. For a network including 6 HP VLs, one of the candidate solutions (P1, P3, P2, P2, P4, P1) for Parent 1 is shown in Fig. 6. In each chromosome, there are six genes corresponding to six VLs and each gene can have four different priority levels. Hence, each gene takes a value from P1 to P4 uniquely representing a priority level. Assuming that there are 100 VLs, there are $4^{100}$ different VL to priority level assignments. A GA is used to explore this huge search space in a time efficient manner to find a reasonably fit solution. Fig. 5 shows the flowchart of the GA approach. The algorithm starts by creating $K$ individuals (i.e., $K$ chromosomes) randomly, where each individual represents a candidate solution. The algorithm in the main loop updates the chromosomes until the stopping criterion is met. The fitness value of each candidate chromosome is calculated to determine how preferable the candidate solution is. The elites which are the best candidates with higher fitness values are directly transferred to the next generation. The elite rate $\mu$ defines the ratio of the elites in the population. Then, the diversity in the population is provided by cross-over and mutation functions. The number of individuals to be selected as the parents of the cross-over operation is determined by the cross-over rate $\eta$ and the offspring are passed to the next generation. The remaining individuals with the ratio of $1 - \eta - \mu$ are used to create mutants with the mutation operation. This procedure continues until the stopping criteria is met. In this study, we select the stopping criteria as the maximum number of generations.

B. Cross-over and Mutation

In GA, the solution diversity is provided by cross-over and mutation operations. The cross-over operation is employed for the randomly selected $K\eta$ parents from the population. An example of the cross-over operation for 6 VL network is given in Fig. 6. Parent 1 and Parent 2 are transformed into two offspring with the randomly selected single point switching point. The mutation operation in GA is used to provide diversity by avoiding the local optimums. The number of the individuals to be employed in the mutation operations are $K(1 - \eta - \mu)$. An example of the single point mutation operation is shown in Fig. 7. The 4th gene of the parent is changed from P2 to randomly selected value of P1 with the mutation operation.

C. Fitness (Cost) Function

At each generation, the fitness values of $K$ individuals are calculated, and the best solutions are transferred to the next generation. We utilize the network calculus theory to find the worst-case delay bounds of each VL for each configuration scenario defined by the candidate solution (individual). Further, using the worst-case delay bound of each VL, the following fitness (cost) function $C_1$ can be used to determine the maximum delay of HP class as the objective of the GA algorithm:

$$C_1 = \max_{n \in G_{HP}} (D_f^n)$$  \hspace{1cm} (8)
Alternatively, the second fitness (cost) function $C_2$ can be used to determine the average delay of $G_{HP}$ as follows:

$$C_2 = \frac{\sum_{n \in G_{HP}} D_{f_n}}{N_{G_{HP}}},$$

(9)

where $N_{G_{HP}}$ represents the number of flows belong to the HP group.

The third fitness (cost) function $C_3$ can be used to calculate the standard deviation of the VL delays in the HP class as:

$$C_3 = \sqrt{\frac{N_{G_{HP}} \sum_{n=1}^{N_{G_{HP}}} (D_{f_n} - C_2)^2}{N_{G_{HP}} - 1}}$$

(10)

Our proposed GA approach utilizes one of the above fitness functions and yields a solution aiming to minimize the selected cost function.

D. GA Convergence Experiments

The GA parameters play an important role to determine the accuracy and the speed of the convergence to the desired objective [7]. In this section, we investigate the effects of elites, mutation, and cross-over rates on the GA convergence as depicted in Fig. 8.

In the first set of experiments, the elites rate is kept constant as 5% while the cross-over rate is increased from 25% to 75% (the mutation rate is decreased from 70% to 20% using $1 - \eta - \mu$). The best result among three experiments is obtained when the cross-over is set to 75% (the mutation is %20). In the second set of experiments, the elites rate goes up from 5% to 15%, while the cross-over is fixed to 75% (the mutation rate is decreased from 20% to 10% using $1 - \eta - \mu$). The convergence becomes degraded such that the worst result is obtained when the elites rate is the highest (15%). In this paper, for the rest of the experiments, the elites, cross-over, and mutation rates are set to %5, %75, and %20, respectively while the population size ($K$) is 50.

IV. NUMERICAL EXPERIMENTS

In this section, we present numerical experiments to demonstrate that the proposed GA approach can be effectively used for the ARINC 664 network planning optimization.

The network topology including 3 switches, 7 ESs and 9 links are used for all experiments as depicted in Fig. 9. All link capacities are 100Mbps, the technological latency in each switch is 100$\mu$s and the service element type is assumed to be FIFO. In this scenario, there are 49 VLs going from 5 transmitting ESs to 2 receiving ESs ($ES_6$ and $ES_7$). The first 14 VLs belong to the HP class while the remaining 35 VLs belong to the LP class. This network scenario with the corresponding BAG and $L_{max}$ values are used from Ref. [8]. For all experiments, the PBOO and pre-emption mechanisms are enabled. Note that the $L_{max}$ values of the VLs in the LP class are generated as uniform distributed random variables with the mean of 750 bytes. Although there are only two priority classes in the original scenario, we first assign 4 priority levels to the 14 VLs in the HP class using the GA optimization approach.

Table I shows the maximum (Max.) and average (Avg.) end-to-end (E2E) delays of the VLs in the HP class and their standard deviations (Std. Dev.) with and without the GA optimization. Note that “without GA” corresponds to the scenario using the default parameters with only two priority classes in [8]. The results demonstrate that the GA optimization approach can successfully achieve the desired objective if the appropriate cost function is selected. For example, the maximum delay of VLs in the HP class is the smallest (reduced from 1000.95 $\mu$s to 840.01 $\mu$s) when the cost function $C_1$ is used. The smallest average delay of 600.06 $\mu$s is achieved when the cost function $C_2$ is utilized. The smallest standard deviation of 57.09 $\mu$s is obtained using the cost function $C_3$.

The same approach can be applied to optimize the LP class after completing the GA optimization of the HP class. Table II shows the delay results of the VLs in the LP class with and without the GA optimization. Without loss of generality, the GA based approach always provides better results for all three performance measures. Similar to the HP class results, the
TABLE I HIGH PRIORITY E2E DELAY

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. (µs)</th>
<th>Avg. (µs)</th>
<th>Std. Dev. (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>without GA</td>
<td>1000.95</td>
<td>869.06</td>
<td>109.47</td>
</tr>
<tr>
<td>GA with C₁</td>
<td>840.01</td>
<td>746.09</td>
<td>142.28</td>
</tr>
<tr>
<td>GA with C₂</td>
<td>974.20</td>
<td>600.06</td>
<td>213.33</td>
</tr>
<tr>
<td>GA with C₃</td>
<td>865.44</td>
<td>756.86</td>
<td>57.09</td>
</tr>
</tbody>
</table>

TABLE II LOW PRIORITY E2E DELAY

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. (µs)</th>
<th>Avg. (µs)</th>
<th>Std. Dev. (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>without GA</td>
<td>6862.59</td>
<td>3998.85</td>
<td>1212.15</td>
</tr>
<tr>
<td>GA with C₁</td>
<td>4172.54</td>
<td>3147.80</td>
<td>779.04</td>
</tr>
<tr>
<td>GA with C₂</td>
<td>4998.64</td>
<td>2836.86</td>
<td>1176.41</td>
</tr>
<tr>
<td>GA with C₃</td>
<td>4191.19</td>
<td>3248.02</td>
<td>632.24</td>
</tr>
</tbody>
</table>

GA optimization approach can successfully achieve the desired objective for the LP class if the appropriate cost function is selected. For instance, the smallest average delay is achieved when the cost function C₂ is utilized.

In Fig. 10, the performance gains of the HP class using different number of priority levels compared to the single priority case as in the standard [4] are shown. When the number of priority levels increases from 2 to 16, the performance gain increases for all three cost functions. For instance, when the number of priority levels (PLs) within the HP class is set to 2, the performance gains are 6.9%, 23.38%, and 37.56% for C₁, C₂, and C₃, respectively. We observe that as the number of PLs increases beyond 4, the performance gains become limited particularly for C₁ and C₂. The reason for this behaviour may be due to the network scenario, where the number of nodes and VLs are relatively small. It is worth to mention that each queue requires the additional memory resources; and hence, the trade-off between the number of priority levels and the memory requirement should be considered for the practical implementation.

V. CONCLUSION

In this paper, we propose a priority extension concept to improve the mixed criticality support for the ARINC 664 network. Using the extended priority concept, the Genetic Algorithm (GA) based network delay optimization is utilized to minimize one of the following objectives, namely the maximum, the average, and the standard deviation of the worst-case delays by considering all VLs in the network. Numerical results show that the VL delays are significantly reduced for the selected delay objectives. For example, the maximum, the average and the standard deviation of the worst-case delays of HP VLs are reduced about 17%, 35%, and 63%, respectively. Future work will investigate the effectiveness of the proposed GA approach using more complex network scenarios.

REFERENCES