Hindawi Publishing Corporation
International Journal of Aerospace Engineering
Volume 2015, Article ID 329820, 9 pages
http://dx.doi.org/10.1155/2015/329820

Research Article

Optimal Configuration of Virtual Links for Avionics Network Systems

Dongha An, Kyong Hoon Kim, and Ki-Il Kim

Department of Informatics, Research Center for Aerospace Parts Technology, Gyeongsang National University, Jinju-daero 501, Jinju 660-701, Republic of Korea

Correspondence should be addressed to Kyong Hoon Kim; khkim@gnu.ac.kr

Received 28 June 2015; Accepted 18 October 2015

Academic Editor: Linda L. Vahala

As the bandwidth and scalability constraints become important design concerns in airborne networks, a new technology, called Avionics Full Duplex Switched Ethernet (AFDX), has been introduced and standardized as a part 7 in ARNIC 664. However, since previous research interests for AFDX are mainly bounded for analyzing the response time where flows information is given, configuration problem for both Maximum Transmission Unit (MTU) and Bandwidth Allocation Gap (BAG) over virtual links in AFDX networks has not been addressed yet even though it has great impact on required bandwidth. Thus, in this paper, we present two configuration approaches to set MTU and BAG values on virtual links efficiently while meeting the requirement of AFDX. The first is to search available feasible configuration (MTU, BAG) pairs to satisfy application requirements as well as AFDX switch constraints, and the second is to get an optimal pair to minimize required bandwidth through well-known branch-and-bound algorithm. We analyze the complexity of the proposed algorithm and then evaluate the proposed algorithm by simulation. Finally, we prove that the proposed schemes are superior to general approach in the aspects of speed and required bandwidth in AFDX networks.

1. Introduction

Due to increasing demands in Aircraft Data Networks (ADNs), such as high available bandwidth, minimum wiring to reduce the weight, and low development cost, typical communications technologies are expected to be replaced by new one. In other words, since ARNIC 429, MIL-STD-1553, and ARNIC 629 cannot accommodate the mentioned demands completely, a new technology, called Avionics Full Duplex Switched Ethernet (AFDX), has been implemented and then standardized for new ADN [1–3]. As a new technology for ADN, reliable and deterministic property as well as implementation cost should be considered. As a result, AFDX was introduced and recently deployed in data networks on the Airbus 380 aircraft.

In the point of technical issue, the AFDX follows original Ethernet for compatibility and scalability properties and extends it to ensure deterministic behavior and high reliability. In order to comply with the stringent requirements of ADNs as well as ensuring them, two new functions are introduced and implemented in AFDX. One is traffic control by guaranteeing the bandwidth of each application, and the other is duplicated transmission for reliability along dual redundant channels. While the former aims to bound the jitter and transmit latency within the deadline, the latter transmits the same data stream over disjoint networks concurrently. To achieve this goal, virtual links are introduced and created for each (source, destination) pair in AFDX. By controlling these virtual links, deterministic behaviors are completely guaranteed. Thus, determining flows for the same virtual link and its properties become the network designer's great task when it comes to configuring AFDX networks.

Two parameters of virtual links in AFDEX networks are important in terms of real-time requirements: BAG (Bandwidth Allocation Gap) and MTU (Maximum Transfer Unit). BAG is the time duration or period between two consecutive frames, where the value of BAG is 1 and 128 msec in a form of power of 2. MTU is the size of message in each frame. Thus, the AFDX configuration problem is to determine these two parameters of each virtual link without missing the deadline.

In [4], they provided several algorithms and integer-linear programming formulation in order to solve the transmission parameters of virtual links as well as data routing over links in the AFDEX networks. They provided optimal solutions to minimize reserved bandwidth as well as actually consumed bandwidth, respectively. In [5], modeling method for AFDX frame management was introduced to ascertain the reliability properties of design. Another important property of AFDX, reliability through redundant transmissions, was analyzed by formal method in [6].

Another related work is to analyze the system metric such as response time of AFDX networks. In [7], they applied network calculus, queuing networks simulation and model checking into evaluating bounding end-to-end delays on AFDX networks. In [8], they showed that Trajectory approach which analyzes the worst-case delays throughout message flows outperforms the network calculus method under industrial configuration. In [9], they obtain worst-case latency and output jitter for the network messages by defining a real-time model for a communications network based on AFDX. Another approach in [10] is performance evaluation system through simulation with popular NS-2 and analysis of impact of several system parameters such as scheduling algorithm.

Since most of existing research work has assumed the preconfigured networks in advance, configuration problem should be mentioned prior to deployment. To achieve this goal, in this paper, we focus on two main problems: (i) finding feasible BAG and MTU parameters of virtual links in an AFDX switch for given virtual links of messages and (ii) providing an optimal solution in feasible pairs to minimize the total bandwidth. We define two problems formally and then solve the problems using the branch-and-bound technique. While our earlier work [11] only provided an algorithm to solve the first problem, this paper improves and extends it to include optimal solution for the purpose of minimizing the total bandwidth. Finally, complexity of the proposed scheme and experimental results through diverse simulation scenarios are given to demonstrate the suitability of the proposed scheme.

The rest of this paper is organized as follows. The system model and the problem definition are provided in Section 2. The solution of a feasible configuration problem is dealt with in Sections 3 and 4, while the optimal algorithm is provided in Section 5. Performance evaluations are shown in Section 6. Finally, conclusion and further work followed in Section 7.

2. System Model and Problem Definition

2.1. System Model. In this paper, we assume that an avionics system is composed of many computing or sensing units and AFDX network switches to connect other computing units. Thus, an AFDX message is uniquely defined by UDP source and destination ports, as shown in Figure 1. Sine we focus on real-time AFDX messages, a message flow f_i is defined by (l_i, p_i), where l_i is the payload of the message in bytes and p_i is Message Transmit Cycle (MTC) of the message in msec. That is, a message of l_i bytes is generated every p_i time units and is delivered to the destination application.

The basic communication unit in AFDEX networks is defined as a *virtual link* (VL). For example, Figure 1 shows three virtual links among LRUs. These virtual links sharing physical links are scheduled in AFDX network switches. Furthermore, multiple applications transmit real-time messages throughout a common virtual link if their source and destination units are the same. In the example of Figure 1, two application messages are shared in the virtual link VL_3.

There are two important parameters in a virtual link. The first is Bandwidth Allocation Gap (BAG) to specify a periodic frame. In AFDX switches, a BAG is defined by a value of 2^k msec, where $k = 0, 1, \ldots, 7$. As all BAGs are 2^k msec, virtual links are multiplexed in AFDX switches. The second parameter is Maximum Transfer Unit (MTU) of the message in bytes at each frame. Payloads of applications in a virtual link are transmitted within maximum MTU bytes in a single frame. If the size of a payload is greater than the MTU, it is fragmented into multiple frames. Therefore, a virtual link VL_i is defined by (BAG_i, MTU_i, F_i) as follows:

(i) BAG_i: Bandwidth Allocation Gap or period of VL_i in a value of 2^k msec where $k = 0, 1, \ldots, 7$.

(ii) MTU_i: maximum transfer unit or message size of VL_i in bytes.

(iii) F_i: a set of message flows in VL_i, where the jth message flow is denoted as $f_{i,j} = (l_{i,j}, p_{i,j})$.

2.2. Problem Definition. For a given virtual link VL_i, MTU and BAG are configured so as to meet all the real-time requirements of message flows in the link. If the payload of a message is greater than the MTU size, it is transmitted in multiple fragmented packets. Since all BAGs of VLs are harmonic, the schedulability analysis is easily derived by utilization analysis. Thus, (1) tells the message constraint of VL_i with n_i messages to guarantee the real-time requirement of all message flows in the link [2]:

$$\sum_{j=1}^{n_i} \frac{\lceil l_{i,j}/\mathrm{MTU}_i \rceil}{p_{i,j}} \leq \frac{1}{\mathrm{BAG}_i}. \tag{1}$$

Let us assume that the system has N VLs on an AFDX switch with B bandwidth in bps. Each VL_i is configured with (MTU_i, BAG_i), so that MTU_i bytes are transmitted every BAG_i msec. In addition, each VL message requires the overhead of 67 bytes as shown in Figure 2. Since the total bandwidth of VLs should not exceed the network bandwidth, the following bandwidth constraint should be met:

$$8\sum_{i=1}^{n} \frac{\mathrm{MTU}_i + 67}{\mathrm{BAG}_i} \times 10^3 \leq B. \tag{2}$$

The last constraint of virtual link scheduling is about jitter. The maximum allowed jitter on each virtual link in

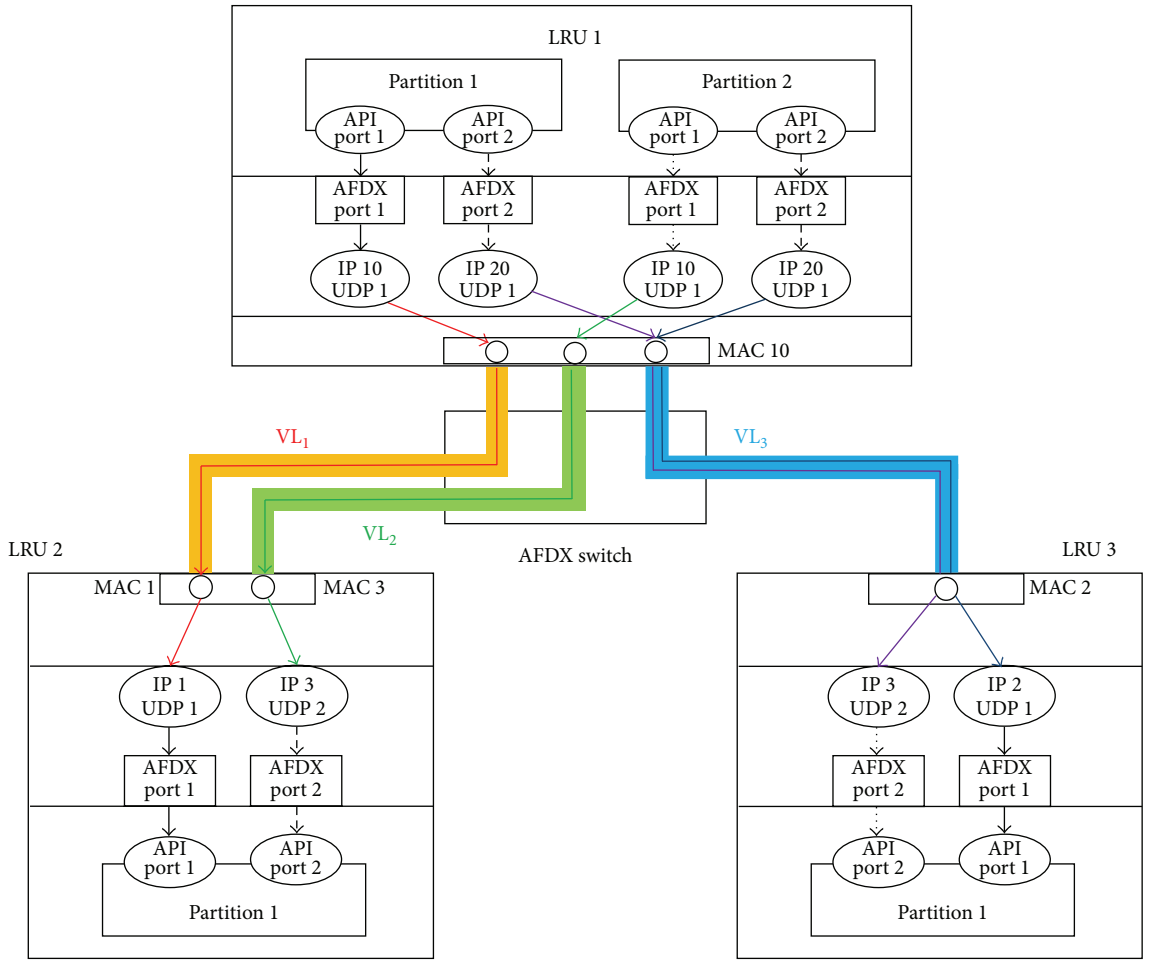


Figure 1: An example of virtual links in an AFDX switch.

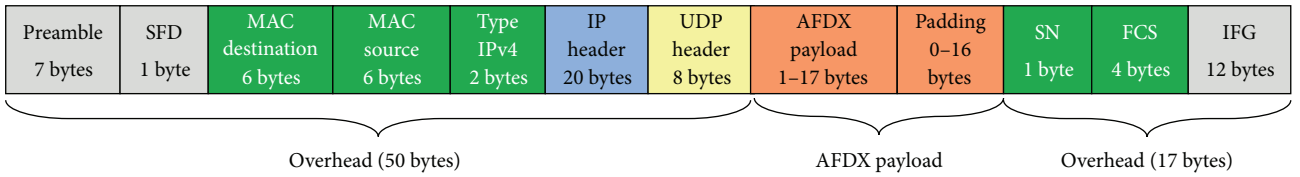


Figure 2: The AFDX frame structure and its overhead.

the ARINC 664 specification requires 500 μsec [2]. Thus, the following equation tells the jitter constraint, where 40 μsec is the typical technological jitter in hardware level to transmit an Ethernet frame:

$$40 + \frac{8\sum_{i=1}^{n}(67 + \mathrm{MTU}_i)}{B} \leq 500. \quad (3)$$

In this paper, we define two problems related to configuration of virtual links of an AFDX switch. The first problem is to find a feasible configuration of BAG and MTU pairs of virtual links, which satisfies three constraints of (1), (2), and (3). This is useful and important when an administrator finds a feasible configuration of a given set of real-time messages in the AFDX switch. The following tells the formal definition of the problem.

Definition 1. For a given set of virtual links $V = \{\mathrm{VL}_i \mid i = 1, \ldots, N\}$, the problem of *AFDX-CONF* is to determine $(\mathrm{BAG}_i, \mathrm{MTU}_i)$ of each VL_i so as to satisfy three constraints of (1), (2), and (3), where $\mathrm{BAG}_i \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ and $\mathrm{MTU}_i \in \{1, 2, \ldots, 1471\}$.

The second one is to find an optimal configuration of virtual links for the purpose of minimizing the total bandwidth. The bandwidth remaining after reserving real-time flows is generally used for non-real-time network traffic. Thus, it is important to find a configuration with the least bandwidth as

Algorithm Find_Feasible_BAG_MTU (VL_i)
(1) $N_{step} \leftarrow \varnothing$
(2) **for each** message $f_{i,j}$ in VL_i **do**
(3) $frag \leftarrow \lceil l_{i,j}/(\lceil l_{i,j}/p_{i,j} \rceil) \rceil$
(4) **while** $frag \geq 1$ **do**
(3) $\lceil m \leftarrow l_{i,j}/\text{frag} \rceil$
(5) $N_{step} \leftarrow N_{step} \cup \{m\}$
(6) $frag \leftarrow frag - 1$
(7) **endwhile**
(8) **endfor**
(9) **for** k **from** 0 **to** 7 **do**
(10) $MTU_{i,k} \leftarrow 0$
(11) $MTU_{i,k} \leftarrow$ the least $m \in N_{step}$ s.t. $\sum_{j=1}^{n_i} (\lceil l_{i,j}/m \rceil / p_{i,j}) \leq 1/2^k$
(12) **endfor**

Algorithm 1: Algorithm of feasible BAG and MTU pairs of a VL.

long as the configuration is feasible. The following tells the second problem definition.

Definition 2. For a given set of virtual links $V = \{VL_i \mid i = 1, \ldots, N\}$, the problem of *AFDX-BOPT* is to find (BAG_i, MTU_i) of each VL_i so as to minimize the total bandwidth and satisfy three constraints of (1), (2), and (3), where $BAG_i \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ and $MTU_i \in \{1, 2, \ldots, 1471\}$.

We solve both problems of *AFDX-CONF* and *AFDX-BOPT* in two steps. The first step is to find the list of (BAG_i, MTU_i) which guarantees the schedulability of message flows in VL_i. Each (BAG_i, MTU_i) should be selected such that it satisfies the constraint of (1) (in Section 3). Then, we find the solution of given virtual links with consideration of two constraints of (2) and (3) (in Sections 4 and 5).

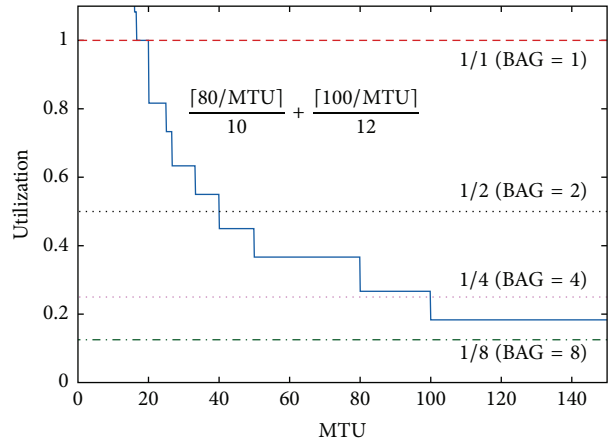


Figure 3: An example of feasible BAG and MTU of a virtual link.

3. Schedulable BAG and MTU Pairs of a VL

In AFDX networks, multiple applications should send their messages to a virtual link queue in the system, so that the scheduling issue for those messages is required. Let us consider a virtual link VL_1 with two message flows of $f_{1,1}(80, 10)$, $f_{1,2}(100, 12)$ as an example. The values of BAG and MTU of VL_1 are set to satisfy (1) in order to meet the real-time requirement of two messages. The left side of (1) is shown in Figure 3 as a step function, while 1/BAG is also drawn in the figure for different BAG values.

For a given BAG_i, there exist many MTUs which satisfy (1). For example, when $BAG_1 = 1$, all MTUs can be used if MTU $\geq$ 17, as shown in Figure 3. Since a longer MTU size requires more bandwidth and jitter, the smallest value should be selected. Thus, MTU_1 of the example VL_1 is 17 bytes when BAG_1 is 1 msec. Similarly, MTUs of VL_1 for BAGs with 2 msec and 4 msec are given by 40 bytes and 100 bytes in each, as shown in Figure 3.

When the MTU size is greater than the maximum payload size of messages, the required utilization is not changed. For example, the lower bound of the utilization of VL_1 is given by about 0.1834 at MTU = 100. This implies that there is no MTU which guarantees the schedulability of two messages if BAG is greater than or equal to 8 msec. Therefore, the feasible solutions (BAG_1, MTU_1) of VL_1 are given by (1, 17), (2, 40), and (4, 100).

The pseudocode of Algorithm 1 describes how to obtain the set of feasible BAG and MTU pairs of a given virtual link VL_i. The first part of the algorithm gathers all step integers at which the utilization function begins a new piecewise constant due to the ceiling function. We denote the set of such step integers as N_{step}. For each message $f_{i,j}$, such step points are derived and added into N_{step} (lines 1–8).

Then, for each 2^k value, we find the minimum MTU which satisfies (1) (lines 9–12). We denote $MTU_{i,k}$ by the feasible MTU in case of $BAG_i = 2^k$ for a virtual link VL_i. If there is no feasible MTU, then $MTU_{i,k} = 0$. For a given n_i flows, the time complexity of Algorithm 1 is $O(n_i \cdot |N_{step}|)$ since we have to find and check the feasibility at each step point of messages.

4. Feasible BAG and MTU Pairs of VLs

4.1. Problem Definition. The problem of finding feasible BAG and MTU pairs of a given set of virtual links is not trivial. For

Table 1: An example of virtual links (B = 1 Mbps).

	Flows ($f_{i,j}$)	Payload ($l_{i,j}$)	MTC ($p_{i,j}$)	Feasible BAG and MTU pairs $s_{i,k}(\text{bag}_{i,k}, \text{MTU}_{i,k})$
VL_1	$f_{1,1}$	200	80	(1, 5), (2, 9), (4, 17), (8, 34), (16, 67), (32, 200)
	$f_{1,2}$	250	160	
VL_2	$f_{2,1}$	250	220	(1, 6), (2, 12), (4, 25), (8, 50), (16, 100), (32, 200)
	$f_{2,2}$	200	40	

example, let us consider the example of two virtual links of Table 1 where the network speed (B) is given by 1 Mbps. For each virtual link, the feasible BAG and MTU pairs are derived by Algorithm 1, as shown in the last column of Table 1. Now, a new problem arises about selecting appropriate BAG and MTU pairs of two virtual links so as to meet both constraints of (2) and (3).

There are some tradeoffs among feasible BAG and MTU pairs of a virtual link VL_i. Solutions with smaller BAG provide less jitter due to smaller MTU size, while they require more bandwidth due to overhead of fragmentation. For example, if we select (1, 5) and (1, 6) as (BAG, MTU) of two VLs of Table 1, it does not meet the bandwidth constraint of (2). On the contrary, if (2, 9) and (2, 12) are selected as (BAG, MTU) of two VLs, this configuration does not meet the jitter constraint of (3). The selection of (1, 5) and (2, 12) of VL_1 and VL_2 satisfies both constraints so that all messages in VLs meet their real-time requirements.

Let us assume that $\text{MTU}_{i,k}$ is derived from Algorithm 1 for each i and k. And, let us denote it by

$$X_{ik} = \begin{cases} 1 & \text{if BAG and MTU of } VL_i \text{ are set as } 2^k \text{ and } \text{MTU}_{i,k} \text{ in each,} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Then, the problem of *AFDX-CONF* can be stated as the following linear integer problem with $n \times 8$ binary variables and three constraints: to find X_{ik}

subject to

$$\sum_{k=0}^{7} X_{ik} = 1$$

$$\sum_{i=1}^{n} \left(\sum_{k=0}^{7} X_{ik} \cdot \frac{\text{MTU}_{i,k} + 67}{2^k} \right) \le \frac{B}{8000} \tag{5}$$

$$\sum_{i=1}^{n} \left(\sum_{k=0}^{7} X_{ik} \cdot (67 + \text{MTU}_{i,k}) \right) \le 460 \cdot \frac{B}{8}.$$

4.2. Algorithm. For given N virtual links, the exhaustive search of the problem *AFDX-CONF* takes $O(8^N)$ since each virtual link might have maximum eight solutions. In this paper, we provide an enhanced branch-and-bound algorithm proposed in [11] in order to find a feasible solution for given N virtual links with their feasible BAG and MTU pairs derived by Algorithm 1. In [11], we used the cumulative bandwidth and jitter for the pruning condition. In this work, however, we define the bound functions of bandwidth and jitter for the remaining levels in the search tree in order to find the pruning node as early as possible. The following is the pruning condition and branch-and-bound strategy of the proposed algorithm.

(i) Pruning Condition. The pruning condition is two constraints of (2) and (3). The algorithm examines whether the solutions in the subset satisfy both constraints by using the bound function of minimum values of remaining levels. The algorithm stops the search of the subset which already violates one of two constraints.

(ii) Branch-and-Bound Strategy. The algorithm searches a feasible solution in a leaf node in depth-first-search (DFS) manner. This algorithm finds a feasible solution when it reaches any leaf node in the search tree.

Algorithm 2 finds the set of feasible BAG and MTU pairs of each virtual link by calling the function Find_Feasible_BAG_MTU ( ) proposed in Algorithm 1 (lines 1, 2). Let us denote σ_i by the set of feasible BAG and MTU pairs ($s_{i,k}$) of virtual link i. Then, the bounds of bandwidth and jitter in virtual link i are found by selecting the minimum values (lines 4–7).

The function *EDFS_BandB* in Algorithm 2 is the recursive implementation at level i in the search tree. Two values of B_{prev} and J_{prev} are two bounds of bandwidth and jitter of sub-solutions from VL_i to VL_N. For each $s_{i,k} = (\text{bag}_{i,k}, \text{MTU}_{i,k})$, two constrains of (2) and (3) are checked including a new solution of VL_i (lines 14–16). If either of the two constraints is not satisfied, it is pruned. Otherwise, the depth-first-search is continued with two updated bound values (line 17).

When the search reaches a leaf node, the function returns *true* (line 12). The return value of calling *EDFS_BandB* is *true*; the final solution S is updated as to include $s_{i,k}$ (line 19) and the function returns *true*. Thus, the problem of *AFDX-CONF* is solved by Algorithm 2. If the return value of *DFS_BandB* (B_0, J_0, 1, S) is *true*, a feasible solution is stored

```
Algorithm Find_Feasible_Configurations (V)
  /* V = {VL_i | i = 1, ..., N} */
(1)  for i from 1 to N do
(2)     call Find_Feasible_BAG_MTU (VL_i)
(3)  S ← ∅
(4)  for i from 1 to N do
(5)     B_i ← min_{s_{i,k} ∈ σ_i} {(MTU_{i,k} + 67)/bag_{i,k}}
(6)     J_i ← min_{s_{i,k} ∈ σ_i} {MTU_{i,k} + 67}
(7)  endfor
(8)  B_0 ← Σ_{i=1}^{N} B_i
(9)  J_0 ← Σ_{i=1}^{N} J_i
(10) result ← EDFS_BandB (B_0, J_0, 1, S)
(11) return S
  Function EDFS_BandB (B_prev, J_prev, i, S)
(12) if i = N + 1 then return true
(13) for each s_{i,k} of VL_i do
(14)     B_curr ← B_prev − B_i + (MTU_{i,k} + 67)/bag_{i,k}
(15)     J_curr ← J_prev − J_i + MTU_{i,k} + 67
(16)     if B_curr ≤ B/8000 and J_curr ≤ 460 · B then
(17)        result ← EDFS_BandB (B_curr, J_curr, i + 1, S)
(18)        if result = true then
(19)           S ← S ∪ {s_{i,k}}
(20)           return true
(21)        endif
(22)     endif
(23) endfor
(24) return false
```

Algorithm 2: The proposed algorithm for *AFDX-CONF*.

in S. Otherwise, the empty set is returned, which implies no feasible configuration is found for a given set of virtual links.

5. Optimal Solution of BAG and MTU Pairs for Reducing Bandwidth

In this section, we focus on finding an optimal configuration of virtual links in terms of the used bandwidth of the configuration. It is important to minimize the total bandwidth of virtual links for real-time traffic because we can use the remaining bandwidth for non-real-time network traffic in avionics systems. The problem of *AFDX-CONF* finds a feasible solution which satisfies three constraints of AFDX switch as soon as possible, so that it does not minimize the bandwidth. However, *AFDX-BOPT* finds a feasible solution which minimizes the total bandwidth, so that the problem is rewritten as follows:

to minimize

$$8000 \sum_{i=1}^{n} \left(\sum_{k=0}^{7} X_{ik} \cdot \frac{\mathrm{MTU}_{i,k} + 67}{2^k} \right) \tag{6}$$

subject to

$$\sum_{k=0}^{7} X_{ik} = 1 \tag{7}$$

$$\sum_{i=1}^{n} \left(\sum_{k=0}^{7} X_{ik} \cdot \frac{\mathrm{MTU}_{i,k} + 67}{2^k} \right) \le \frac{B}{8000} \tag{8}$$

$$\sum_{i=1}^{n} \left(\sum_{k=0}^{7} X_{ik} \cdot \left(67 + \mathrm{MTU}_{i,k}\right) \right) \le 460 \cdot \frac{B}{8}. \tag{9}$$

For example, there are two feasible configurations in Table 1: {(1, 5), (2, 12)} and {(2, 9), (1, 6)} under the bandwidth constraint of 1 Mbps. The first configuration requires 892 kbps, while the second one uses 888 kbps. Thus, in this case, {(2, 9), (1, 6)} is the optimal configuration of two virtual links.

In order to solve the problem, we provide a branch-and-bound algorithm. The bound function is the same as one in Algorithm 2, so that we use the minimum bandwidth and jitter for bound values of each node. Thus, the pruning condition is two constraints of (8) and (9) using the bound functions. The branch strategy is different from DFS. Since the algorithm finds an optimal solution as soon as possible, we select a node with the minimum bandwidth bound function and branch its child nodes. The pseudocode of this is shown in Algorithm 3. Since the branch-and-bound technique searches all possible cases in the worst-case, the time complexity of Algorithm 3 is $O(2^N)$ for given N virtual links.

In Algorithm 3, a *node* has four data: *node.level* for the node level, *node.B* for the bandwidth bound, *node.J* for the jitter bound, and *node.S* for the selected configuration from the root to the node. We use a priority-queue for managing live nodes in the order of bandwidth bound. The function *insert_node_in_Q* (*i*, *B*, *J*, *S*) makes a node with four data and inserts it into the priority queue. Then, the function *get_first_in_Q* ( ) returns the minimum-bandwidth node in currently live nodes in the queue. When the algorithm reaches a leaf node (line 12), it finds an optimal solution and returns the configuration (line 13).

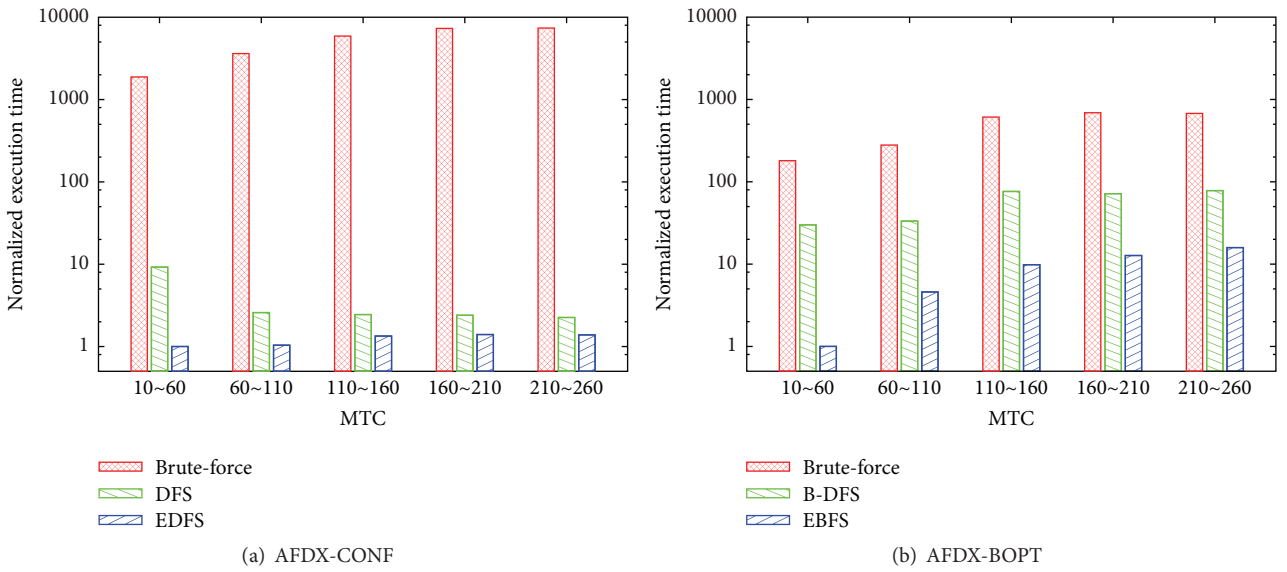
The branch strategy is implemented by the priority queue. The function call of *get_first_in_Q* ( ) selects such node to branch among live nodes. For all child nodes of the selected node, the bound functions of bandwidth and jitter are updated (lines 16-17). The pruning condition is checked in line 18. If a new child node is not pruned, it is added in the priority queue (line 19). The algorithm repeats this procedure until it finds an optimal solution or there is no live node in the queue. If there is no feasible configuration for virtual links, it returns the empty set for notifying no feasible solution (line 22).

6. Performance Evaluations

In this section, we show performance evaluation of the proposed algorithm. First, we evaluate the execution time of the proposed algorithms compared with the brute-force search and the previous work [11]. In the experiments, we generate five virtual links with two message flows in each virtual link. The payload of a message is randomly generated from 20 to 80 bytes. The MTC or period of a message is

Algorithm Find_Minimum_Bandwidth_Configuration (V)
(1) **for** i **from** 1 **to** N **do**
(2) call **Find_Feasible_BAG_MTU** (VL_i)
(3) **for** i **from** 1 **to** N **do**
(4) $B_i \leftarrow \log_{s_{i,k} \in \sigma_i}\{(MTU_{i,k} + 67)/bag_{i,k}\}$
(5) $J_i \leftarrow \log_{s_{i,k} \in \sigma_i}\{MTU_{i,k} + 67\}$
(6) **endfor**
(7) $B_0 \leftarrow \sum_{i=1}^{N} B_i$
(8) $J_0 \leftarrow \sum_{i=1}^{N} J_i$
(9) *insert_node_in_Q* $(0, B_0, J_0, \varnothing)$;
(10) **while** *is_empty_Q* ( ) = *false* **do**
(11) *node* ← *get_first_in_Q* ( );
(12) **if** *node.level* = N **then**
(13) **return** *node.S*;
(14) $i \leftarrow$ *node.level* + 1
(15) **for each** $s_{i,k}$ of VL_i **do**
(16) $B_{curr} \leftarrow$ *node.B* $- B_i + (MTU_{i,k} + 67)/bag_{i,k}$
(17) $J_{curr} \leftarrow$ *node.J* $- J_i + MTU_{i,k} + 67$
(18) **if** $B_{curr} \le B/8000$ **and** $J_{curr} \le 460 \cdot B$ **then**
(19) *insert_node_in_Q* $(i, B_{curr}, J_{curr}, node.S \cup \{s_{i,k}\})$;
(20) **endfor**
(21) **endwhile**
(22) **return** $\varnothing$

Algorithm 3: The *Bounded_BFS_BandB* algorithm.



(a) AFDX-CONF (b) AFDX-BOPT

Figure 4: Normalized execution times of algorithms.

randomly selected among five different intervals, as shown in Figure 4. The network bandwidth is set as 6 Mbps.

For each case of Figure 4, we generate 5000 random sets of five virtual links and measure the average execution time of algorithms. Figure 4(a) shows the normalized execution times of the brute-force search, *DFS* in [11], and *EDFS* in this paper for solving *AFDX-CONF*. Similarly, Figure 4(b) shows the performance of the proposed algorithm *EBFS* compared to the brute-force search and the modified version of DFS [11] for solving *AFDX-BOPT*.

As shown in Figure 4, the proposed algorithms run much faster than the exhaustive search algorithm due to the branch-and-bound technique. Let us note that the y-axis in Figure 4 is log-scale value. Table 2 shows the average number of BAG and MTU pairs per virtual link. As the MTCs become larger, the number of possible solutions of each virtual link is increased, which requires more execution time, as shown in Figure 4. Since we use bound functions for bandwidth and jitter, the proposed ones are faster than the previous algorithm in [11].

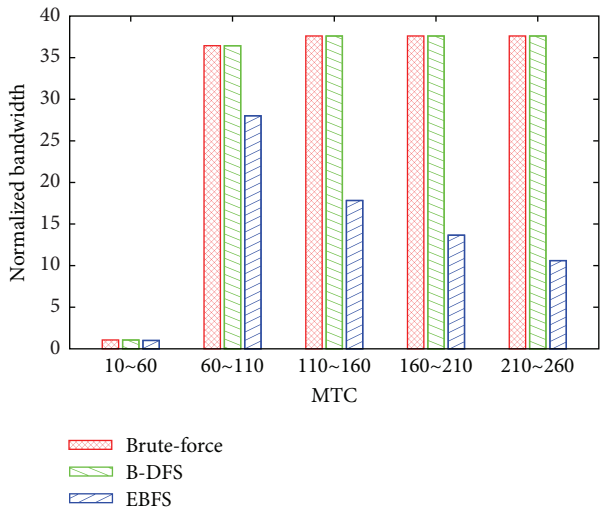


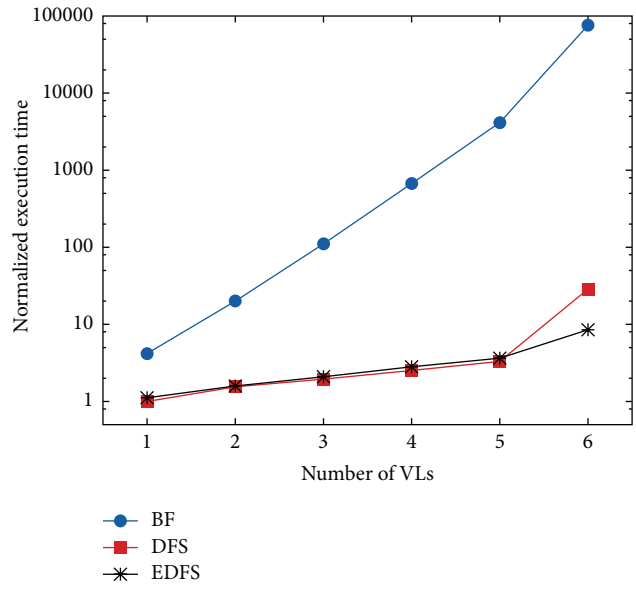
Figure 5: Bandwidth comparison.

Table 2: The average number of BAG and MTU pairs per VL.

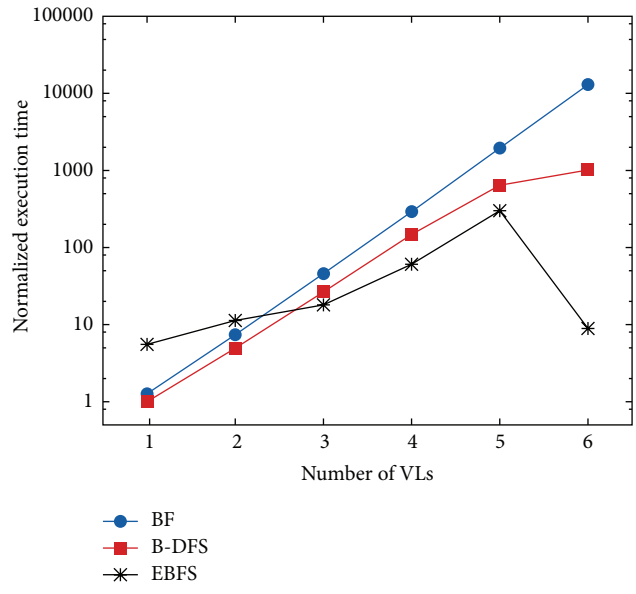
MTC	10~60	60~110	110~160	160~210	210~260
BAG and MTU pairs per VL	4.7	6.0	6.8	7.0	7.0

Next, we compared the bandwidth reserved by the solutions of *AFDX-BOPT* with feasible solutions of *AFDX-CONF*. As shown in Figure 5, the proposed algorithm for *AFDX-BOPT* reduces much bandwidth, so that the remaining bandwidth can be used for other non-real-time network traffic.

We also analyzed the effect of algorithm execution times in terms of the number of virtual links. In Figure 6, we increased the number of virtual links from 1 to 6, where each virtual link includes two messages. As shown in Figure 6, the execution time of brute-force search increases exponentially since the search space is $O(2^N)$. Let us note that y-axis in Figure 6 is log-scale. Although the time complexity of the proposed algorithm is $O(2^N)$, both *EDFS* and *EBFS* are much faster than the previous algorithms in [11]. This is mainly because we use efficient bound functions in the algorithms. The execution time of *EBFS* in case of 6 virtual links is decreased since the algorithm decides that there is no solution in the upper-layer of solution trees.



(a) AFDX-CONF



(b) AFDX-BOPT

Figure 6: Normalized execution times with respect to the number of virtual links.

7. Conclusions

In this paper, we provided an efficient algorithm to find feasible configuration of an AFDX switch for the purpose of meeting the real-time requirements of all messages in avionics. Two important parameters of BAG and MTU of virtual links are considered in the configuration problem. The proposed algorithm is based on branch-and-bound technique so that the simulation results show that it is faster than the exhaustive search algorithm of the previous work.

We also provide an optimal solution of the problem for the purpose of reducing the total bandwidth of configuration. Throughout simulations, we showed that the algorithm runs fast and reduces much bandwidth compared with the algorithm to find a feasible set. The proposed algorithm is useful when the remaining bandwidth for real-time traffic is used for other network traffic.

Since the AFDX network configuration becomes an important issue in avionics systems, we will investigate many problems based on the results of this paper. For example, we will extend the problem into multiple AFDX switches or discuss the routing issues through the networks.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (no. NRF-2015R1A1A1A05001369), and the BK21 Plus Program (Research Team for Software Platform on Unmanned Aerial Vehicle, 21A20131600012) funded by the Ministry of Education (MOE, Korea) and National Research Foundation of Korea (NRF).

References

[1] R. L. Alena, J. P. Ossenfort, K. I. Laws, A. Goforth, and F. Figueroa, "Communications for integrated modular avionics," in *Proceedings of the IEEE Aerospace Conference*, pp. 1–18, IEEE, Big Sky, Mont, USA, March 2007.

[2] AFDX Tutorial, http://www.techsat.com/fileadmin/media/pdf/infokiosk/TechSAT_TUT-AFDX-EN.pdf.

[3] AFDX/ARNIC 664 Tutorial, http://www.cems.uwe.ac.uk/~ngunton/afdx_detailed.pdf.

[4] A. Al Sheikh, O. Brun, M. Chéramy, and P.-E. Hladik, "Optimal design of virtual links in AFDX networks," *Real-Time Systems*, vol. 49, no. 3, pp. 308–336, 2013.

[5] I. Saha and S. Roy, "A finite state modeling of AFDX frame management using spin," in *Formal Methods: Applications and Technology*, vol. 4346 of *Lecture Notes in Computer Science*, pp. 227–243, Springer, Berlin, Germany, 2007.

[6] J. Täubrich and R. von Hanxleden, "Formal specification and analysis of AFDX redundancy management algorithms," in *Computer Safety, Reliability, and Security*, vol. 4680 of *Lecture Notes in Computer Science*, pp. 436–450, Springer, Berlin, Germany, 2007.

[7] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an AFDX network," in *Proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS '06)*, pp. 193–202, Dresden, Germany, July 2006.

[8] M. Tawk, X. Liu, L. Jian, G. Zhu, Y. Savaria, and F. Hu, "Optimal scheduling and delay analysis for AFDX end systems," SAE Technical Paper 2011-01-2751, SAE International, 2011.

[9] J. J. Gutierrez, J. C. Palencia, and M. G. Harbour, "Response time analysis in AFDX networks with sub-virtual links and prioritized switches," in *XV Jornadas de Tiempo Real*, Santander, 2012.

[10] D. Song, X. Zeng, L. Ding, and Q. Hu, "The design and implementation of the AFDX network simulation system," in *Proceedings of the International Conference on Multimedia Technology (ICMT '10)*, pp. 1–4, Ningbo, China, October 2010.

[11] D. An, H. W. Jeon, K. H. Kim, and K. I. Kim, "A feasible configuration of AFDX networks for real-time flows in avionics systems," in *Proceedings of the International Workshop on Real-Time and Distributed Computing in Emerging Applications (REACTION '13)*, Vancouver, Canada, Decebmer 2013.