Heuristic Algorithm for Virtual Link Configuration in AFDX Networks

YoungJun Cha and Ki-Il Kim
Department of Informatics, Engineering Research Institute
Gyeongsang National University
Jinju, 660-701, Korea
Email: kikim@gnu.ac.kr

***Abstract*—As the AFDX networks have been increasingly employed for airborne networks, much research works have been conducted to support real-time service in a deterministic way. However, since they assumed the preconfigured networks where all involved parameters were already determined, the impact of configuration algorithm is not well explored. To solve this problem, in this paper, we focus on how to reduce the required bandwidth by configuring virtual link which logically consists of at least one or more application flows. To achieve this, new heuristic algorithms have been proposed by applying well-known greedy approach while taking essential constraints of AFDX networks into account. To evaulate the performance of proposed scheme, diverse case studies for airborne application flows are concerned and their number of virtual links as well as required bandwidth are compared.**

***Keywords*—AFDX, Virtual Link, Configuration**

I. Introduction

Recently, the airborne networks demand high transmission rate and real-time service due to increase in avionics equipments such as diverse sensors and multimedia devices. However, since existing avionics networks such as ARNIC(Aeronautical Radio, Incorporated)-429, MIL-1553 can not meet above requirements well, a new network technology, AFDX (Avionics Full-Duplex Switched Ethernet), has been proposed. AFDX networks are based on IEEE 802.3 Ethernet due to its strong compatibility and popularity in communication technology. Moreover, this protocol is standardized as a PART 7 in ARINC 664[1].

In the point of technology, since typical Ethernet is not designed to support airborne networks which aim to provide deterministic service as well as real-time service, diverse new features have been introduced in AFDX. These include functions to support QoS (Quality of Service) such as admission control, shaping and policing. In addition, scheduling and duplicated transmission are another functions to meet requirement of AFDX networks. These mentioned functions are provided by new terminology, that is, VL (Virtual Link) between source and destination. Through this VL, deterministic behaviors of flows are guaranteed by two major parameters, BAG (Bandwidth Allocation Gap) and MTU (Maximum Transfer Unit). So, determining virtual link properties and configuring network environments become one of great tasks.

But, despite of its importance and demand, a few research works have been conducted in this area so impact of configuration and relationship between parameters still need to be explored. Especially, recent work for AFDX networks has focused on the system analysis [2-8]. In their works, the AFDX network analysis was accomplished by one of following scheme, that is, queuing networks, network calculus, and model checking. Throughout the analysis schemes, the impact of parameters has been analyzed including end-to-end delays, worst case latencies, and so on. But, as far as the authors know, there is few research work to address configuration problem with the exception of our previous works. In addition, the authors proposed how to set the transmission parameters of virtual links so as to minimize the reserved bandwidth while transmitting the data within their maximum delivery times[7]. Rather, in our previous work, we derive algorithm how to find the possible BAG and MTU pair for one VL in AFDX networks timely[9]. Even though our proposed algorithm is able to get feasible pairs for (BAG, MTU) to minimize the bandwidth requirement, it does not address how to group applications in one virtual link as well as the number of virtual links. That is, we also assumed the preconfigured networks in previous work like others.

To solve this problem, in this paper, we propose new heuristic algorithms for virtual link configuration to reduce the required bandwidth. The proposed algorithm has two different operations according to metric for grouping and it is based on greedy scheme while grouping. Basically, two VLs can be merged into one VL if bandwidth requirement of the merged VL is less than sum of two separate VLs. Otherwise, a VL remains without merging. This procedure will continue until there is no available VL left for merging. For this, two different metrics, bandwidth and ratio for payload per period are defined. Finally, two algorithms are evaluated by the diverse scenarios and analyzed in the point of amount of required bandwidth and number of VLs.

The rest of this paper is organized as follows. In section II, we briefly present the AFDX networks and their important properties. The new algorithms are described and explained by example in section III. The experiment results are presented in section IV. And then, conclusion and further work will be given in section V.

II. AFDX Networks

AFDX networks consists of end system and switching system as shown in Figure 1. In AFDX networks, a LRU sends a frame to other LRU connected by one or multiple switches. The physical point-to-point link of existing ARNIC-429 is replaced by the VL which is time division multiplexed at

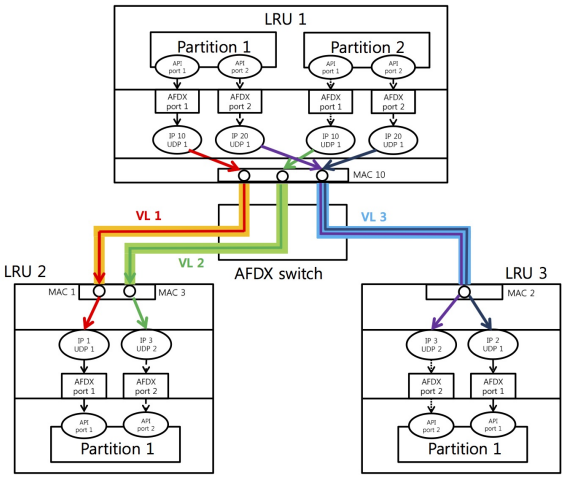

Fig. 1. An example of virtual links in an AFDX switch

the end system. Each VL is guaranteed as a specific maximum bandwidth and end-to-end maximum latency for QoS (Quality of Service). Actually, each VL can have maximum four sub-VLs. Fig. 1 shows an AFDX networks with three VLs 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 Fig. 1, two application messages are shared in the virtual link VL_3.

To guarantee bandwidth, each VL is assigned a transmission time slot, BAG. A VL can transmit a frame within this BAG. It confines the virtual links bandwidth by defining the minimum gap time between two consecutive frames. Another important parameter to affect the bandwidth is MTU. It is defined as the maximum size of message to be transmitted in each frame. Both latency and jitter are controlled by traffic shaping function at the end system. The jitter allows the end system flexibility when transmitting simultaneous frames from different VLs. In AFDX, jitter is defined as the time between the beginning of the BAG interval and the first bit of the frame to be transmitted in the BAG. The maximum allowed jitter for ADFX networks is limited to 500 μs.

III. System Model and Problem Definition

A. System Model

Since we deal with 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 MTC (Message Transmit Cycle) 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 through AFDX networks.

In AFDX networks, 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 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 small frames. Therefore, a virtual link VL_i is defined by (BAG_i, MTU_i, F_i) as follows.

- BAG_i: bandwidth allocation gap or period of VL_i in a value of 2^k msec where $k = 0, 1, \ldots, 7$.
- MTU_i: maximum transfer unit or message size of VL_i in bytes.
- F_i: a set of flow id in VL_i, where the message flow with flow id j is denoted as $f_j = (l_j, p_j)$.

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, Eq. (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 as defined in [1].

$$\sum_{j \in F_i}^{n_i} \frac{\lceil l_j / MTU_i \rceil}{p_j} \leq \frac{1}{BAG_i} \tag{1}$$

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. Since the total bandwidth of VLs should not exceed the network bandwidth, the following bandwidth constraint should be met.

$$8 \sum_{i=1}^{n} \frac{MTU_i + 67}{BAG_i} \times 10^3 \leq B \tag{2}$$

The last constraint of virtual link scheduling is about jitter. The maximum allowed jitter on each virtual link in the ARINC 664 specification requires 500 μsec [**?**]. Thus, the following equation tells the jitter constraint, where 40 μsec is the typical technological jitter in hardware level to transmit an Ethernet frame. These three equations are defined and required as a standard in [1].

$$40 + \frac{8 \sum_{i=1}^{n} (67 + MTU_i)}{B} \leq 500 \tag{3}$$

B. Problem Definition

When there are n flows represented by $\xi = \{f_1, f_2, ..., f_n\}$ between end system and switch, our solution is to construct F_i in VL_i while reducing the required bandwidth as compared to the case that any specific algorithm is not given. To achieve this goal, we propose two heuristic algorithms depending on grouping metrics such as required bandwidth or flow property. By the proposed algorithms, two or more flows are grouped in one VL only if the required bandwidth for VL being merged flows is less than sum of bandwidth of merging flows. The detail procedures are described in Algorithm 1.

Algorithm 1 shows the configuration scheme based on required bandwidth. To run this algorithm, a new set B composed with required bandwidth for f_i is defined. At

Algorithm 1 Grouping Algorithm with Required Bandwidth

1: $B = \{B_1, B_2, ..., B_n\}$ ▷ B_i = Required Bandwidth for f_i
2: $Found \leftarrow True$, if any f_i is grouped with other flow
3: $MIN \leftarrow \infty$

4: **for** $i \leftarrow 1, n$ **do**
5: $F_i \leftarrow \{i\}$
6: (BAG_i, MTU_i) ← Find_Feasible_BAG_MTU(F_i)
7: $B_i \leftarrow$ Find_Feasible_Configuration(BAG_i, MTU_i)
8: **end for**

9: **for** $count \leftarrow n-1, 1$ **do**
10: Sort B_i in ascending order
11: $l \leftarrow \underset{B_i \in B}{\operatorname{argmin}}(B_i)$
12: $Found \leftarrow False$
13: **for** $j \leftarrow 2, |B|-1$ **do**
14: $m \leftarrow \underset{B_i \in B, i>j}{\operatorname{argmin}}(B_i)$
15: (BAG_t, MTU_t) ← Find_Feasible_BAG_MTU($F_l \bigcup F_m$)
16: $B_t \leftarrow$ Find_Feasible_Configuration(BAG_t, MTU_t)

17: **if** ($B_t < B_l + B_m$ and $B_t < MIN$) **then**
18: $F_{temp} \leftarrow F_l \bigcup F_m$
19: $B_{temp} \leftarrow B_t$
20: $MIN \leftarrow B_t$
21: $Found \leftarrow True$
22: **end if**
23: **end for**
24: **if** ($Found$ == $True$) **then**
25: $F_l \leftarrow F_{temp}$
26: $B_l \leftarrow B_{temp}$
27: $B \leftarrow B$ - $\{B_m\}$
28: **else**
29: $B \leftarrow B$ - $\{B_l\}$
30: **end if**
31: **end for**

32: **return** F_i

TABLE I. THE EXAMPLE OF FLOW INFORMATION

Flow Id	Payload	MTC
f_1	200	80
f_2	180	65
f_3	165	100
f_4	140	10
f_5	135	20
f_6	120	40
f_7	115	90
f_8	100	55

initial phase represented by line 4 $\sim$ 8, each respective flow is assigned to one VL. Also, two parameters, (BAG_i, MTU_i) are computed by calling **Find_Feasible_BAG_MTU()** function introduced in our previous works[9]. After getting feasible sets by calling above function, minimum required bandwidth for each VL is done by another function, **Find_Feasible_Configuration()**. This two functions are slighted modified in our previous works.

TABLE II. STEPS FOR ALGORITHM 1

	F_i	B_i
Step 1	1	33.375
	2	30.875
	3	29
	4	207
	5	101
	6	46.75
	7	22.75
	8	41.75
$\sum B_i$	512.5	

	F_i	B_i
Step 2	1	33.375
	2	30.875
	4	207
	5	101
	6	46.75
	7, 3	45.5
	8	41.75
$\sum B_i$	506.25	

	F_i	B_i
Step 3	1	33.375
	2	30.875
	4	207
	5	101
	6	46.75
	7, 3	45.5
	8	41.75
$\sum B_i$	506.25	

	F_i	B_i
Step 4	1, 8	66.75
	2	30.875
	4	207
	5	101
	6	46.75
	7, 3	45.5
$\sum B_i$	497.875	

	F_i	B_i
Step 7	1, 8	66.75
	2	30.875
	5	101
	6	46.75
	7, 3, 4	232
$\sum B_i$	477.375	

After initializing the variable, the main procedure for grouping starts sorting the set of bandwidth for each VL in ascending order. And then, we choose flow with the smallest required bandwidth and keep this flow id to temporary variable. Next, we choose the possible flow id for pair with a selected one by comparing the required bandwidth. Two selected flows can be possible to be grouped into one VL only if the required bandwidth of VL which assumes two flows grouping in it is less than sum of two current VLs bandwidth. If this condition is true, we mark it and continue searching possible flow id to minimize required bandwidth. This condition is represented by line 17 $\sim$ 22 in algorithm 1. By comparing the minimum value, it is possible to find the best flow id to minimize the required bandwidth. Finally, If we find the adequate two flow id to minimize the required bandwidth, the grouped flow replace the existing VL and update the required bandwidth for further comparison. In this procedure, grouped flow id is excluded in B set. Otherwise, if there is no possible flow to be grouped with current flow, it occupies one VL and maintains it until end of algorithm. This procedure is employed to exclude the corresponding flow from B as shown in line 29. That is, the proposed algorithm do not take this flow into account any more for grouping. As a result, a flow can find the best pair for grouping or remain one VL by this proposed algorithm. This procedure will continues as much as total number of flows (n) - 1 times since only one flow is determined. Table 2 illustrate the example of operations of algorithm 1 when the example flow information with different payload and MTC is given in

Table 1.

For the given flow, in the first step, required bandwidth for each flow is computed at initialization phase. In Step 2, our algorithm starts looking for possible grouping pair for flow 7 because its bandwidth is minimum among all flows. Since the grouping condition is met by flow 3, two flows are merged into one VL and flow 3 is no more possible pair for others. Next step is for flow 2. However, since there is no flow to satisfy grouping condition with flow 2, no grouping happens at Step 3 and corresponding VL is composed of that flow. At the Step 4, flow 1 is grouped with flow 8 depending like above procedure. This procedure continues until Step 7 is accomplished. Finally, five VLs are created by our algorithm and around 7 percents bandwidth for the case that all VLs is configured by one flow is saved in our example.

Even though algorithm 1 is to find the possible configuration method to reduce bandwidth, its operation is mostly dependent on the required bandwidth. However, since the required bandwidth is computed by the Equation (1) $\sim$ (3), it is feasible to analyze the equations in order to reduce the required bandwidth. Specially, in Equation (2), we can identify that both great BAG and less MTU contributes to reduce bandwidth. Since (MTU_i/ BAG_i) represents throughput for link, algorithm 2 takes throughput of end system represented by (l_i/p_i) instead of required bandwidth. Moreover, due to grouping pattern starting from the smallest value to the largest value, the two largest values are generally grouped in the last step. But, if these two flows is grouped into one VL, the high bandwidth is required due to fragmentation and simultaneous transmission in a BAG. So, it can be possible that this increasing bandwidth by two flows is greater than total reduced bandwidth obtained by grouping the small bandwidth. Thus, it is necessary to group the flow with the largest value at early step by changing the order of operation. To implement above demand, new grouping procedure introduces two functions at the same step. One is to group with two flows with the smallest value and the other is to accomplish grouping flow with the largest value and suitable flow among remaining flows. This new heuristic algorithm is explained below as Algorithm 2.

Algorithm 2 begins initialization phase like algorithm 1. And then, it sorts new metric, BP_i, in ascending order. And then, two flows with the smallest BP_i are chosen for grouping as indicated in line 9 $\sim$ 22. This procedure is the same as algorithm 1. When the grouping is done with the smallest value, related variables are updated. Upon updating, a flow id with the largest value is chosen in line 24. Next step is to chosen the suitable pair flow id with the smallest values in BP set. If the grouping condition becomes true, two flows are grouped. Through this algorithm, maximum four flows are grouped into two VLs at one step. So, less steps are demanded than one of algorithm 1. Table 3 shows the example of grouping steps in algorithm 2 with the same flows in Table 1.

Table III shows the detail steps for grouping by algorithm 2. At first, initialization procedures including B_I computation and assigning each flow to one VL is accomplished. Flow 7 and 3 are grouped in one VL and removed from BP. After that, flow 4 with the largest value starts searching until finding flow 8 which is the smallest value in current BP. Finally, all procedures are done after flow 1 and 5 are merged into one

Algorithm 2 Grouping Algorithm with Flow Parameter of Each Flow

1: BP = $\{BP_1, BP_2, .., BP_n\}$ ▷ BP_i = ratio of payload per period in f_i and represented as l_i/p_i
2: B_i = Required Bandwidth for f_i

3: **for** $i \leftarrow 1, n$ **do**
4: $F_i \leftarrow \{i\}$
5: (*BAG*$_i$, *MTU*$_i$) $\leftarrow$ Find_Feasible_BAG_MTU(F_i)
6: $B_i \leftarrow$ Find_Feasible_Configuration(*BAG*$_i$, *MTU*$_i$)
7: **end for**

8: **for** $count \leftarrow \log_2 |BP|, 1$ **do**
9: Sort BP_i in ascending order
10: $k \leftarrow \underset{BP_i \in L}{\operatorname{argmin}}(BP_i)$
11: **for** $j \leftarrow 1, |L| - 1$ **do**
12: $m \leftarrow \underset{BP_i \in BP, i > j}{\operatorname{argmin}}(BP_i)$
13: (*BAG*$_t$, *MTU*$_t$) $\leftarrow$ Find_Feasible_BAG_MTU($F_k \bigcup F_m$)
14: $B_t \leftarrow$ Find_Feasible_Configuration(*BAG*$_t$, *MTU*$_t$)
15: **if** $((B_t < B_k + B_m)$ and $(Found \neq True)$ **then**
16: $F_k \leftarrow F_k \bigcup F_m$
17: $BP_k \leftarrow (BP_k + BP_m)/2$
18: $B_k \leftarrow B_t$
19: $BP \leftarrow BP - \{BP_m\}$
20: $Found \leftarrow True$
21: **end if**
22: **end for**

23: $Found \leftarrow False$

24: $k \leftarrow \underset{BP_i \in BP}{\operatorname{argmax}}(BP_i)$
25: **for** $j \leftarrow 2, |BP| - 1$ **do**
26: $m \leftarrow \underset{BP_i \in BP, i > j}{\operatorname{argmin}}(BP_i)$
27: (*BAG*$_t$, *MTU*$_t$) $\leftarrow$ Find_Feasible_BAG_MTU($F_k \bigcup F_m$)
28: $B_t \leftarrow$ Find_Feasible_Configuration(*BAG*$_t$, *MTU*$_t$)
29: **if** $((B_t < B_k + B_m)$ and $(Found \neq True)$ **then**
30: $F_k \leftarrow F_k \bigcup F_m$
31: $BP_k \leftarrow (BP_k + BP_m)/2$
32: $B_k \leftarrow B_t$
33: $BP \leftarrow BP - \{BP_m\}$
34: $Found \leftarrow True$
35: **end if**
36: **end for**
37: **end for**

38: **return** F_i

TABLE III. STEPS FOR ALGORITHM 2

	F_i	BP_i	B_i
Step 1	1	2.50	33.375
	2	2.77	30.875
	3	1.65	29
	4	14.00	207
	5	6.75	101
	6	3.00	46.75
	7	1.28	22.75
	8	1.82	41.75
$\sum B_i$	512.5		

	F_i	BP_i	B_i
Step 2	1	2.50	33.375
	2	2.77	30.875
	4, 8	15.82	207
	5	6.75	101
	6	3.00	46.75
	7, 3	2.93	45.5
$\sum B_i$	464.5		

	F_i	B_i	BP_i
Step 3	1, 5	9.25	133.5
	2	2.77	30.875
	4, 8	15.82	207
	6	3.00	46.75
	7, 3	2.93	45.5
$\sum B_i$	463.625		

VL. As a result, around 9 percents of bandwidth is reduced as compared to one in Step 1 in Table III.

IV. EXPERIMENT RESULTS

In this section, we provide the experiment results for proposed scheme through diverse flows sets. In the experiments, we assume eight message flows. The payload of a message is randomly generated from 20 to 80 bytes. The MTC or period of a message is randomly selected among five different intervals. Five intervals range from 10 to 260 msec and the duration of each interval is 50 msec. The network bandwidth is set as 6Mbps. In order to model eight flows, three different data sets are assume in Table 1. Each set has different rule for payload and MTC where Fixed represented the same value for parameter and Varied does changing one depending on experiment setting explained above. Through these three sets, we can identify the impact of each parameter and relationship between them.

TABLE IV. PROPERTY OF EACH DATA SET

	Payload	MTC
Data Set 1	Varied	Fixed
Data Set 2	Fixed	Varied
Data Set 3	Varied	Varied

For each data set, we generate 100 cases and measure the average required bandwidth in each algorithm. The comparative method is initialization scheme which each flow occupies one VL as shown in initialization phase in our algorithm. We compare the required bandwidth of each algorithm as well as how many virtual links are made.

Fig. 2 shows the experiment result for required bandwidth of each algorithm. The Y-axis in Fig. 2 represents the ratio for required bandwidth as a base of required bandwidth computed by algorithm 2. So, the ratio is computed by ($\sum Bi$ in algorithm1 or initialization / $\sum Bi$ in algorithm 2) and shown

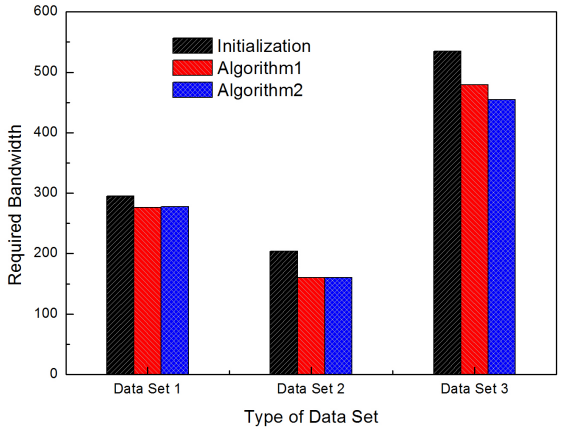

Fig. 2. Comparison of ratio for required bandwidth as a base algorithm 2

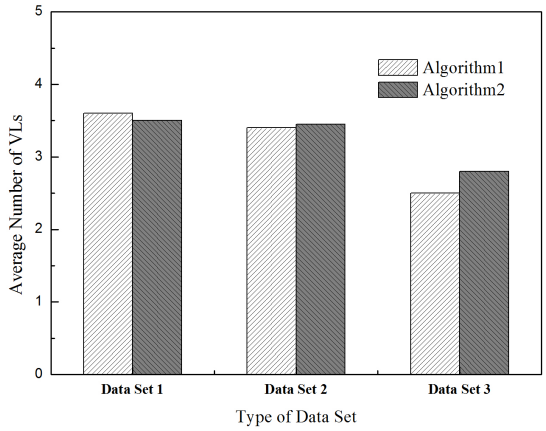

Fig. 3. Comparison of average number of VLs

according to type of data set. In this figure, it is very easy to identify the reduced bandwidth of the proposed scheme rather than initialization algorithm. For all cases, two proposed algorithms outperform comparative scheme. Moreover, slight difference is observed between two proposed algorithms when data set 1 and 2 are tested. This is because bandwidth and throughput are very closely related to each other. Thus, grouping results for given data set through two algorithms are almost identical even though order of grouping flow is slightly different. This mean that sub-VLs for one VL is the almost identical without regard to algorithms which determines the grouping order for each node.

Unlike mentioned two data sets, we can recognize the difference between two algorithms for data set 3. According to varied flow properties, different grouping order and number of VLs are observed. Furthermore, algorithm 2 outperforms algorithm 1 slightly. This difference is determined by whether flow with the largest value is merged at early time or last time. By this procedure, bandwidth difference among VLs in algorithm 2 become smaller than algorithm 1. This means

that huge amount of bandwidth is not observed in algorithm 1. Specially, when data flow consists of extremely inclined property, noticeable difference is observed in algorithm 1. This case include the following example that two flows have the largest difference but other six flows shows the almost the similar property. On the other hand, for the case where all flows is divided into two categories or all flows have the similar property, the difference between two algorithm is not great.

The average number of VLs for respective data set are compared in Fig. 3. Compared to reduced bandwidth, it shows similar patterns because bandwidth is very closely related to the number of VLs. However, two algorithms have the slight difference for data set 3. Algorithm 1 demands the less number of VL than algorithm 2. The number of VLs increases in algorithm 2 because more VLs consisting of one flow id are demanded. This implies that a few flow is feasible to be grouped with flow that has the largest throughput. As a summary, the number of VLs is mostly dependent on variation of payload and MTC. If the variation between flows become minimum, least number of VLs are required. But, since the experiment is conducted for eight flows, it is very hard to see the great difference in them.

V. Conclusion

In this paper, we proposed two heuristic algorithms to configure the number of VL and sub-VLs in ADFX networks when flow information is given. The proposed algorithms are to reduce the required bandwidth and perform grouping procedure by comparing the required bandwidth as well as new defined metrics. Moreover, they need less bandwidth than initialized configuration method without regard to metric and grouping order in two algorithm. This reduced bandwidth can be used to accommodate more flows in AFDX networks.

Related to this work, more extended experiments are needed to evaluate the proposed scheme. It will include increased number of flows and more data set. Also, we will evaluate the how much time is taken for each algorithm. Another work is to devise more optimal solution to reduce bandwidth. One solution can be top down approach which splits one VL to multiple ones to reduce bandwidth unlike current bottom up approach.

Acknowledgment

This work was supported by Basic Science Research Program (NRF-2013R1A1A2A10004587) through the National Research Foundation of Korea (NRF) funded by the Ministry of Education and the MSIP(Ministry of Science, ICT & Future Planning), Korea, under the ”SW master’s course of a hiring contract” support program (NIPA-2014-HB301-14-1014) supervised by the NIPA(National IT Industry Promotion Agency).

References

[1] ARINC AIRCRAFT DATA NETWORK PART 7 AVIONICS FULL-DUPLEX SWITCHED ETHERNET NETWORK ARINC SPECIFICATION 664 P7. Sep 2009.

[2] J. J. Gutirrez, J. Carlos Palencia, M. G. Harbour, *Holistic schedulability analysis for multipacket messages in AFDX networks*, Journal of Real-Time Systems, vol. 50, no. 2, pp. 230-269, Mar. 2014.

[3] S. Wang, J. Shi, D. Sun, M. Tomovic, *Time delay oriented reliability analysis of Avionics Full Duplex Switched Ethernet*, In Proc. of IEEE Industrial Electronics and Applications (ICIEA), pp. 982 - 987, June. 2013.

[4] S. Dong, Z. Xingxing, D. Lina, H. Qiong, The design and implementation of the AFDX network simulation system, In Proc. of Multimedia Technology (ICMT), pp. 1 - 4, Oct. 2010.

[5] H. Charara, J. Scharbarg, J. Ermont, and C. Fraboul, *Methods for bounding end-to-end delays on an AFDX network*, In Proc. Of 18th Euromicro Conference on Real-time Systems, pp. 193-202, July 2006.

[6] H. Bauer, J. Scharbarg, and C. Fraboul, *Worst-case end-to-end delay analysis of an avionics AFDX network*, In Proc. of Design, Automation & Test in Europe Conference & Exhibition, pp. 1220-1224, March 2010.

[7] M. Tawk, X. Liu, L. Jian, G. Zhu, *Optimal scheduling and delay analysis for AFDX end-systems*, SAE Technical Paper 2011-01-2751, 2011,

[8] Y. Hua, X. Liu, *Scheduling heterogeneous flows with delay-aware deduplication for avionics applications*, IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 9, pp. 1790-1802, Sept., 2012

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