Analysis of TDMA Crossbar Real-Time Switch Design for AFDX Networks

Lei Rao*†, Qixin Wang*§, Xue Liu‡, Yufei Wang*

***Abstract*—The rapid scaling up of modern avionics is forcing its communication infrastructure to evolve from shared medium toward multi-hop switched real-time networks. This prompts the proposal of *avionics full-duplex switched Ethernet* (AFDX) standard. Since its publication, AFDX has been well-received, and is deployed or to-be-deployed in state-of-the-art aircrafts, such as Airbus A380/A400M/A350, Boeing 787, Bombardier CSeries etc. On the other hand, AFDX standard only specifies the behavior that an underlying switch must follow, but leaves the architecture design open. This creates an open market for switch vendors. Among the different candidate designs for this market, the TDMA crossbar real-time switch architecture stands out as it complies with and even simplifies many mainstream switch architectures, hence lays a smooth evolution path toward AFDX. In this paper, we focus on analyzing this switch design for AFDX networks. We first prove that TDMA crossbar real-time switch architecture complies with the AFDX specifications; and derive closed-form formulae on the corresponding AFDX networks' traffic characteristics and end-to-end real-time delay bound. Then we prove the resource planning problem in the corresponding AFDX networks is NP-Hard. To address this NP-Hard challenge, we re-model the problem. Based upon the re-modeling, we propose an approximation algorithm.**

I. Introduction

Modern avionics systems are rapidly scaling up to support various advanced computing demands, such as fly-by-wire, auto pilot, head-up display, helmet mounted display, digital combat systems etc.[1][2]. As an example, Airbus A380 is already deploying hundreds of processors onboard. This forces the avionics communication infrastructure to evlove from single-hop shared medium to multi-hop switched real-time networks. As a result, *avionics full-duplex switched Ethernet* (AFDX), a.k.a. ARINC 664 Part 7, is published as a standard for such networks [3][4].

AFDX is well-received by the avionics industry, and is deployed or to-be-deployed in many state-of-the-art aircrafts, such as Airbus A380/A400M/A350, Boeing 787, Sukhoi Superjet 100, AgustaWestland AW101, Irkut MS-21, Bombardier CSeries etc. [5].

As most avionics network flows are for safety/mission-critical control loops (to keep the aircraft in the air, or to conduct combat etc.), the key demand to AFDX networks is to guarantee end-to-end *real-time*, i.e., every packet's end-to-end delay is upper bounded by a preconfigured constant.

To support end-to-end real-time, AFDX mimics a virtual point-to-point connection, a.k.a. *virtual link* (VL), between any source-end destination-end pair. We can regard each VL conducts one unicast flow (support of flow aggregation [3] is beyond the scope of this paper) from a source-end to a destination-end. Along the VL's route, before entering each AFDX switch, the VL flow f must behave as if policed by a *token bucket* [6]. The token bucket is defined by a bucket size $\sigma_f \stackrel{\mathsf{def}}{=} L_f^{\mathsf{max}}(1 + J_f/\mathsf{BAG}_f)$ and a bucket refilling rate $\rho_f \stackrel{\mathsf{def}}{=} L_f^{\mathsf{max}}/\mathsf{BAG}_f$, where L_f^{max} is the maximum packet bit length of f; $\mathsf{BAG}_f \in \mathbb{R}^{>0}$ is the *bandwidth allocation gap* of f; and $J_f \in \mathbb{R}^{\geq 0}$ is the *maximum admissible jitter* of f. For this paper, it is enough to know L_f^{max}, BAG_f, and J_f are all constants throughout VL's runtime life cycle (interested readers can refer to [3] for their intuitive meanings).

With the above per hop token bucket policing and proper switch architecture design, we can guarantee end-to-end real-time for each VL. AFDX standard leaves the switch architecture design open, so that vendors can propose their own.

In other words, given a switch architecture and a network of such switches, if the aforementioned per-flow token bucket policing and end-to-end real-time are supported, then the flow is considered running in a VL, and the switched network is *AFDX compliant*.

The openess of the AFDX standard/market attracts switch vendors. However, the challenge is, most vendors want to reuse their legacy switch architectures, particularly the non-real-time Internet switch architectures, instead of a complete redesign.

To address this challenge, this paper tries to build AFDX networks using a (if not "the") popular real-time switch architecture, which we call "*time division multiple access* (TDMA) crossbar real-time switch" [7][8][9][10][11]. The merit of this architecture lies in its compliance with (even simplifications to) many mainstream non-real-time switch architectures [7][9][11], particularly iSLIP [12], a de facto standard for Internet switches. Such backward compatibility lays a smooth evolution path for non-real-time switch vendors toward AFDX.

In summary, our main contributions include: *i*) we proved that TDMA crossbar real-time switched network is AFDX compliant, and derived the formulae for its traffic pattern and end-to-end real-time delay bound; *ii*) we proved the corresponding AFDX network's resource planning problem is NP-Hard; *iii*) we proposed a re-modeling approach, upon which, we proposed an approximation algorithm.

In the following, Section II introduces the TDMA crossbar real-time switch architecture; Section III proves TDMA crossbar real-time switched networks are AFDX compliant;

* Dept. of Computing, The Hong Kong Polytechnic University
† Present Address: School of Computer Science, McGill University
‡ School of Computer Science, McGill University
§ Corresponding Author Email: csqwang@comp.polyu.edu.hk

Section IV proves the resource planning in such AFDX networks is NP-Hard; Section V re-models the resource planning problem, and finds it an approximation algorithm; Section VI discusses related work; and Section VII concludes the paper.

II. Background

Let us first introduce the TDMA crossbar real-time switch architecture.

Switch architectures generally fall into two categories: output-queueing and input-queueing. In output-queueing, only *output ports* (simplified as "*outputs*" in the following) can buffer packets. Once a packet enters a switch, it is immediately routed to its corresponding output to be buffered there. This simple scheme, however, has a fatal shortcoming: if N packets arrive at the switch's N *input ports* (simplified as "*inputs*" in the following) simultaneously, and all route to the same output O, then the switch internal fabric at the entrance of O must be N times faster than the input port capacity (assume all inputs have the same capacity) [13]. Due to this shortcoming of output-queueing, input-queueing instead becomes the de facto standard for high performance switches [7][12].

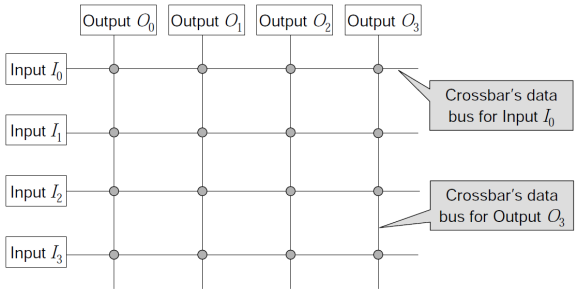
In input-queueing, a *crossbar* fabric connects a switch's inputs with outputs (see Fig. 1(a)). Packets are only buffered at the inputs. When a packet enters an input, it is immediately buffered at a proper queue in the input (see Fig. 1(b)). At scheduled time, the destination output will fetch the packet through the crossbar and directly forward it to the next hop without further buffering (see Fig. 1(c)).

To facilitate scheduling, inside an input-queueing switched network, all packets are fragmented into same-size units called *cells*. Inside of a switch, outputs fetch/forward cells synchronously and periodically. The period is called a *cell-time*. At the beginning of a cell-time, the switch decides a *one-to-one matching* between its inputs and outputs. During the cell-time, each output fetches/forwards a cell (if there is one) from its matched input via the crossbar. One-to-one matching is necessary due to the crossbar constraint that at any time, one input can only connect to one output and vice versa.

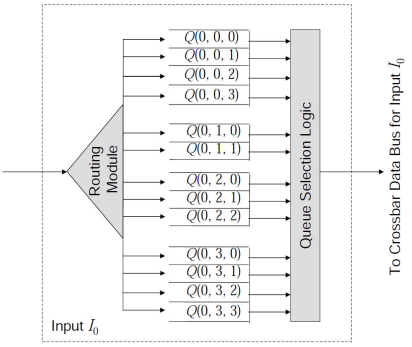
Depending on the detailed queueing and matching schemes, various input-queueing switch architectures exist. Among them is the TDMA crossbar real-time switch architecture [7][9][10][11] (in [10], the switch architecture is called "Birkhoff-von Neumann" architecture, as it is inspired by Birkhoff [15] and Von Neumann [16]'s math models). Its deterministic TDMA scheduling allows end-to-end real-time guarantee; meanwhile, it also complies with (or even simplifies) mainstream non-real-time switch architectures [7][9][11], particularly the iSLIP architecture[12], which happens to be the industry de facto standard for Internet switches. This backward compliance lays a smooth evolution path for switch vendors toward real-time switched networks, such as AFDX.

The details of TDMA crossbar real-time switch architecture are as follows.

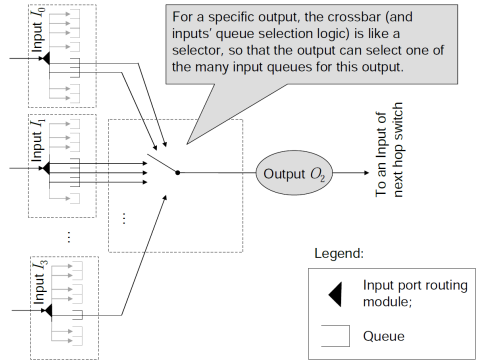
In such a switch, each input carries out per-flow queueing, and each output runs a *static* TDMA schedule of M cell-time, a.k.a. an *M-slot frame* (note static scheduling is



(a) crossbar fabric, which connects inputs with outputs; each input connects to a data bus (the horizontal line segments) that intersects with each output's data bus (the vertical line segments); the intersections (grey dots) can be connected/disconnected during runtime by scheduler(s); note at any time, one input can connect to at the most one output, and vice versa.



(b) an input port: packet routing and queueing are carried out in it; in input i, the kth queue buffering packets to output j is denoted as $Q(i, j, k)$.



(c) an output port: at different time slot, the output fetches packets from different input queues according to the switch scheduling scheme.

Fig. 1. Input Queueing Switch Architecture (quoted from [14])

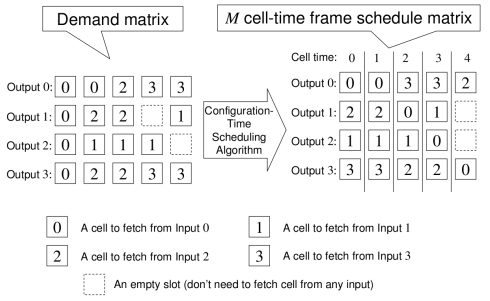


Fig. 2. Conflict free schedule for TDMA crossbar real-time switch (quoted from [14]): in this example, the switch has $N = 4$ inputs and outputs, frame size is $M = 5$ slots (note in reality, M is in the order of $10^3 \sim 10^5$); each row of the "schedule matrix" is a conflict free schedule for its corresponding output, which means at any time slot (i.e., any column of the "schedule matrix"), no two outputs contend for the same input (for different input queues).

possible because real-time network flows are for persistent sensing/actuating, and are configured offline; e.g., avionics control loop flows persist throughout each flight from taking off to landing). The kth ($k = 0, 1, \ldots, M-1$) slot of the frame specifies from which input per-flow queue shall the output fetch cell at the kth (modulo M) cell-time of the switch. Note since the TDMA schedule is static, if the corresponding input queue happens to be empty, the output just fetches/forwards nothing during the cell-time.

To ease narration, in the following, we will use the term "*M-slot frame*" and "*frame*" interchangeably; and the term "*slot*" and "*cell-time*" interchangeably.

One rule is that at any cell-time of the switch, no two outputs can fetch from the same input (i.e., the aforementioned *one-to-one matching* rule). If the switch outputs' M-slot frame schedules all respect this rule, we say the switch has a *conflict free* schedule (see Fig. 2). An important thoerem on conflict free schedulability is as follows (quoted from [7]):

Theorem 1 (Schedulability)**.** *For an N inputs N outputs TDMA crossbar real-time switch described above, if in every M-slot frame, each output needs to fetch no more than M cells, and each input needs to send no more than M cells, then we can always derive a* conflict free *schedule for the switch with $\mathcal{O}(N^4)$ time cost.*

The corresponding $\mathcal{O}(N^4)$ scheduling algorithm is in [7]. Note as real-time networks carry out scheduling offline, such complexity is satisfactory.

III. AFDX Compliance of TDMA Crossbar Real-Time Switched Networks

In the following, we shall show how to build a basic AFDX network with TDMA crossbar real-time switches.

As mentioned in Section I, to make a switched network AFDX compliant, the core challenges are to ensure each flow *i*) behaves as if being policed by a token bucket before entering each switch; *ii*) has end-to-end real-time guarantee, i.e., end-to-end delay is upper bounded.

To address these challenges, we first look at the features of AFDX network flows. AFDX network flows are mostly periodical sensing/actuating/video flows. Even if there are a few non-real-time flows, their source-ends can send data in a periodical pattern. Therefore, we can assume the following:

Assumption 1 (Stable Periodical Traffic Source)**.** *Let F be the set of all flows in an AFDX network. For each flow $f \in F$, its source-end injects no more than $L_f^{\mathsf{src}} \in \mathbb{Z}^{\geq 0}$ bits of data into network at and only at time instances $\phi_f + kP_f^{\mathsf{src}}$ (second), where $k \in \mathbb{Z}^{\geq 0}$, $P_f^{\mathsf{src}} \in \mathbb{R}^{>0}$ and $\phi_f \geq 0$ (second) is the* phase *(i.e., initialization time) of flow f. We call L_f^{src} (bit) and P_f^{src} (second) the* source-end maximum packet length *and* source-end period *of flow f respectively.*

L_f^{src} and P_f^{src} use the units of "bit" and "second" respectively. We can adapt the units into "cell" and "cell-time" to more directly match the switch architecture.

Specifically, let τ (second) be the duration of a cell-time, then each M-cell-time frame lasts

$$T \stackrel{\mathsf{def}}{=} M\tau \text{ (second)}.$$

In practice, switches with wire capacity of 10Gbps are already in mass production (in fact, wire capacity of 100Gbps and beyond may emerge in in near future [17]), and the standard cell size ℓ is around 500 bits. This implies τ is at the magnitude of $5 \sim 50$ nanoseconds. So even if M is as big as $2,000 \sim 20,000$, the corresponding T remains at the magnitude of 0.1 millisecond. On the otherhand, avionics sensing/actuating/video periods usually range within $1 \sim 1000$ millisecond. This allows another assumption:

Assumption 2. $T << P_{\mathsf{min}}^{\mathsf{src}} \stackrel{\mathsf{def}}{=} \mathsf{min}_{\forall f \in F}\{P_f^{\mathsf{src}}\}$.

Then for each $f \in F$, we can adapt its source-end maximum packet length and source-end period into following:

$$L_f \stackrel{\mathsf{def}}{=} \lceil L_f^{\mathsf{src}}/\ell \rceil \text{ (cell)}; \quad (1)$$

$$P_f \stackrel{\mathsf{def}}{=} \lfloor P_f^{\mathsf{src}}/(\tau M) \rfloor M = \lfloor P_f^{\mathsf{src}}/T \rfloor M \text{ (cell-time)}; \quad (2)$$

where constant ℓ (bit) is the standard cell size. We call L_f and P_f flow f's *in-network maximum packet length* and *in-network period*.

Note L_f and P_f's units are now cell and cell-time respectively. Also note due to Assumption 2, $P_f/M = \lfloor P_f^{\mathsf{src}}/T \rfloor \in \mathbb{Z}^{>0}$. This guarantees $P_f/M \neq 0$, hence for each flow f, we can define its *per-frame throughput* θ_f as

$$\theta_f \stackrel{\mathsf{def}}{=} \lceil L_f/(P_f/M) \rceil \text{ (cell)} . \quad (3)$$

When $\theta_f = 0$ (i.e., $L_f = 0$), flow f is *inactive*. When $\theta_f > 0$, we have:

Theorem 2 (AFDX Compliance)**.** *If a flow f with per-frame throughput of $\theta_f > 0$ cells routes through a set of TDMA crossbar real-time switches. If each switch can schedule $C_f \geq \theta_f$ ($C_f \in \mathbb{Z}^{\geq 0}$) slots per frame to forward f, then*
Claim 1*) f behaves as if being policed by a token bucket of bucket size $\sigma_f = L_f(1 + H_f M/P_f)$ (cell), and token refilling rate $\rho_f = L_f/P_f$ (cell/cell-time), where H_f is the total* hop count *of f, i.e., the number of switches that f passes excluding the source and destination ends;*
Claim 2*) for each of the H_f switches that f passes, the queue backlog for f is upper bounded by $Q_f = L_f + H_f L_f M/P_f$ (cell);*
Claim 3*) the end-to-end delay of f is upper bounded by $\Delta_f = H_f M + L_f M/C_f \leq H_f M + P_f$ (cell-time).*

Proof: First we define two functions of time t: *rate-latency* function $\mathsf{F}^{\mathsf{rate\_latency}}_{\gamma,\delta}(t)$ and *affine* function $\mathsf{F}^{\mathsf{affine}}_{\rho,\sigma}(t)$ as follows

$$\mathsf{F}^{\mathsf{rate\_latency}}_{\gamma,\delta}(t) \stackrel{\mathsf{def}}{=} \gamma \cdot [t-\delta]^+,$$
$$\mathsf{F}^{\mathsf{affine}}_{\rho,\sigma}(t) \stackrel{\mathsf{def}}{=} \begin{cases} 0 & \text{for } t = 0 \\ \rho t + \sigma & \text{for } t > 0 \end{cases}.$$

where $[x]^+ \stackrel{\mathsf{def}}{=} \mathsf{max}\{0, x\}$. We call γ and δ the rate and latency of $\mathsf{F}^{\mathsf{rate\_latency}}_{\gamma,\delta}(t)$ respectively; and ρ and σ the rate and burst of $\mathsf{F}^{\mathsf{affine}}_{\rho,\sigma}(t)$ respectively.

Now we can start our analysis. Without loss of generality, Fig. 3 shows a flow f that leaves source-end and travels H_f hops of TDMA crossbar real-time switches (denoted as $v_0, v_1, \ldots, v_{H_f-1}$ in the figure) to reach its destination end v_{H_f}.

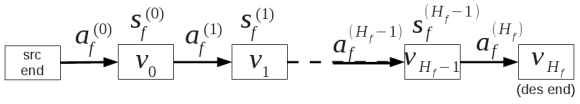


Fig. 3. An example flow f in a TDMA crossbar real-time switched network.

To analyze f, we carry out network calculus [18]. Let $a_f^{(k)}$ and $s_f^{(k)}$ be f's network calculus arrival curve and service curve respectively at switch v_k ($k = 0, 1, \ldots, H_f - 1$). In the following, we first try to derive the closed form expressions for $a_f^{(k)}$ and $s_f^{(k)}$.

We notice that the source-end packet size is $\frac{L_f^{\mathsf{src}}}{\ell}$ (cell) $\leq L_f$ (cell), and the source-end data rate is $\frac{L_f^{\mathsf{src}}/\ell}{P_f^{\mathsf{src}}/\tau}$ (cell/cell-time) $\leq \frac{L_f}{P_f}$ (cell/cell-time) (for consistency, throughout the proof, we convert all data size units to "cell", and time units to "cell-time"). These, combined with Assumption 1, imply a network calculus arrival curve [18] of

$$a_f^{(0)}(t) = \mathsf{F}^{\mathsf{affine}}_{L_f/P_f, L_f}(t). \tag{4}$$

On the other hand, because of the C_f TDMA slots allocated to serve f every M-cell-time frame, we can write

$$s_f^{(0)}(t) = \mathsf{F}^{\mathsf{rate\_latency}}_{C_f/M, M}(t).$$

Because $s_f^{(0)}(t)$ has a rate C_f/M no less than $a_f^{(0)}(t)$'s rate L_f/P_f (as $C_f \geq \theta_f \geq L_f M/P_f$), according to Theorem 1.4.1 of [18], the queue backlog of f at v_0 is upper bounded by

$$Q_f^{(0)} = L_f + \frac{L_f}{P_f} M;$$

according to Theorem 1.4.2 of [18], the packet delay of f at v_0 is upper bounded by

$$\Delta_f^{(0)} = M + \frac{L_f}{C_f/M};$$

according to Theorem 1.4.3 of [18], when f exits v_0, f complies with arrival curve

$$a_f^{(1)}(t) = a_f^{(0)}(t) \oslash s_f^{(0)}(t) = \mathsf{F}^{\mathsf{affine}}_{L_f/P_f, L_f + \frac{L_f}{P_f} M}(t).$$

We can apply the above analysis iteratively to v_1, v_2, …, v_{H_f-1} to derive per hop arrival curve, service curve, queue backlog upper bound, and delay upper bound respectively as

$$a_f^{(k)}(t) = \mathsf{F}^{\mathsf{affine}}_{L_f/P_f, L_f + \frac{kL_f M}{P_f}}(t) \quad (k = 0 \sim H_f),$$
$$s_f^{(k)}(t) = \mathsf{F}^{\mathsf{rate\_latency}}_{C_f/M, M}(t) \quad (k = 0 \sim H_f - 1), \tag{5}$$
$$Q_f^{(k)} = L_f + \frac{(k+1)L_f M}{P_f} \quad (k = 0 \sim H_f - 1), \tag{6}$$
$$\Delta_f^{(k)} = M + \frac{L_f + kL_f M/P_f}{C_f/M} \quad (k = 0 \sim H_f - 1). \tag{7}$$

Because $\forall k \in \{0, \ldots, H_f\}$, flow f's arrival curve $a_f^{(k)}(t)$ is an affine function with burst $L_f + kL_f M/P_f \leq L_f(1 + H_f M/P_f)$, flow f hence also complies with arrival curve

$$a_f(t) = \mathsf{F}^{\mathsf{affine}}_{L_f/P_f, L_f(1+\frac{H_f M}{P_f})}(t) \tag{8}$$

throughout its route. Due to the equivalence of affine function arrival curve and token bucket policing (see [18]), Formula (8) means f complies with a token bucket of bucket size $\sigma_f = L_f(1 + \frac{H_f M}{P_f})$ (cell) and token refilling rate $\rho_f = L_f/P_f$ (cell/cell-time). Hence Claim 1 sustains.

Because $\forall k \in \{0, \ldots, H_f - 1\}$, $L_f + \frac{(k+1)L_f M}{P_f} \leq L_f + \frac{H_f L_f M}{P_f}$, Formula (6) implies that Claim 2 sustains.

Though Formula (7) implies an end-to-end upper bound for f, a tighter bound exists. According to Formula (5), the concatenation of the $v_0 \sim v_{H_f-1}$ together (as a black box) provide a service curve of

$$\begin{aligned} s_f(t) &= s_f^{(0)}(t) \otimes s_f^{(1)}(t) \otimes \ldots \otimes s_f^{(H_f-1)}(t) \\ &= \mathsf{F}^{\mathsf{rate\_latency}}_{C_f/M, H_f M}(t) \end{aligned} \tag{9}$$

for flow f. Combining Formula (9) and (4), we get end-to-end delay upper bound for f is

$$\begin{aligned} \Delta_f &= H_f M + \frac{L_f}{C_f/M} = H_f M + \frac{L_f}{C_f} M \\ &\leq H_f M + \frac{L_f}{L_f/P_f} = H_f M + P_f. \end{aligned}$$

Hence Claim 3 sustains. ■

Combining Theorem 2 and the description of AFDX in Section I, we see a network built upon TDMA crossbar real-time switches is AFDX compliant.

IV. Resource Planning Problem

Section III proves the legitimacy of using TDMA crossbar real-time switches to build AFDX networks. Next, we are interested in how to optimally plan such AFDX networks.

In practice, AFDX networks mainly serve the persistent, real-time, and periodical sensing/actuating/video flows. Such a flow f has a real-time end-to-end deadline D_f (cell-time), i.e., every packet sent by its source-end must reach the destination-end within D_f cell-time. Also, f typically has a set of *discrete* alternatives on source-end maximum packet size L_f^{src} and source-end packet period P_f^{src} (hence sampling rate). For example, when f's source-end is a sensor, its sensor reading can be 500 bits or 1000bits, and sampling rate can be $1, 2, \ldots,$ 100Hz; when f's source-end is a camera, its video resolution can be 800×600, 1024×768, or 1920×1080 pixels, and video sampling rate can be 40Hz, 50Hz, or 100Hz etc.

With the above understanding of individual flow, we now model the network. Suppose we have an AFDX network $G(V, E)$ of TDMA crossbar real-time switches, where V is the set of all switches (vertices) of the network, and E is the set of links between the switches. Let F be the set of flows in the AFDX network. We assume

Assumption 3 (Fixed Routes)**.** *Unless explicitly denoted, for each flow $f \in F$, its route is given (i.e., fixed).*

For each flow $f \in F$, its set of alternatives for source-end maximum packet length and source-end period is

$$A_f^{\mathsf{src}} = \{(L_{f,k}^{\mathsf{src}}, P_{f,k}^{\mathsf{src}})\},$$

where $k = 0, 1, \ldots, |A_f^{\mathsf{src}}| - 1$ is the index of alternatives. In the following, we call A_f^{src} flow f's *source-end alternatives set*; we denote the kth alternative in A_f^{src} as $A_{f,k}^{\mathsf{src}}$ ($k = 0, 1, \ldots, |A_f^{\mathsf{src}}| - 1$); what is more, $\forall f \in F$, A_f^{src} always has its 0th alternative $A_{f,0}^{\mathsf{src}} = (0, M\tau)$ ($\forall f \in F$), which corresponds to the alternative that flow f is *inactive*.

A_f^{src} further implies another form to write the set of alternatives:

$$A_f = \{(L_{f,k}, P_{f,k}, \theta_{f,k}, u_{f,k}(L_{f,k}, P_{f,k})\},$$
$$(k = 0, 1, \ldots, |A_f| - 1 \text{ is the index of alternatives}),$$

where L_f is flow f's in-network maximum packet length (see Formula (1)), P_f is f's in-network period (see Formula (2)), θ_f is f's per-frame throughput (see Formula (3)), and $u_f(L_f, P_f)$ is f's utility. Utility $u_f(L_f, P_f) \in \mathbb{Z}^{\geq 0}$ is a function of L_f and P_f. It quantifies the quality of service that f provides to application layer. In the following, we call A_f flow f's *in-network alternatives set*; we denote the kth alternative in A_f as $A_{f,k}$ ($k = 0, 1, \ldots, |A_f| - 1$); what is more, $\forall f \in F$, A_f always has its 0th alternative $A_{f,0} = (0, M, 0, 0)$, which corresponds to the alternative that f is *inactive*.

For each switch $v \in V$, let $I^{(v)}$ be its set of inputs, and $O^{(v)}$ be its set of outputs. For input $i \in I^{(v)}$, let $\check{F}_i^{(v)}$ be the set of flows entering switch v via i. For output $j \in O^{(v)}$, let $\hat{F}_j^{(v)}$ be the set of flows leaving switch v via j. As all flows' routes are given (see Assumption 3), all $\check{F}_i^{(v)}$ and $\hat{F}_j^{(v)}$ are also given.

According to Theorem 1, switch v is schedulable if and only if Formulae (10) and (11) both hold:

$$\sum_{f \in \check{F}_i^{(v)}} C_f \leq M, \forall i \in I^{(v)}, \forall v \in V; \tag{10}$$

$$\sum_{f \in \hat{F}_j^{(v)}} C_f \leq M, \forall j \in O^{(v)}, \forall v \in V; \tag{11}$$

$$\text{where } C_f \geq \theta_f = \lceil \frac{L_f}{P_f/M} \rceil, \text{ and } C_f \in \mathbb{Z}^{\geq 0}.$$

is the per-frame allocated slots to forward f.

Therefore, we can model the TDMA crossbar real-time switch AFDX network *resource planning problem* $\mathcal{P}(G(V, E), F)$ as follows.

$$\begin{aligned}
&\max \sum_{f \in F} u_f(L_f, P_f) \\
&\text{s.t.} \quad \sum_{f \in \check{F}_i^{(v)}} C_f \leq M, && \forall i \in I^{(v)}, \forall v \in V; \\
&\qquad \sum_{f \in \hat{F}_j^{(v)}} C_f \leq M, && \forall j \in O^{(v)}, \forall v \in V; \\
&C_f \geq \theta_f = \left\lceil \frac{L_f}{P_f/M} \right\rceil, && \forall f \in F; && (12) \\
&H_f M + \frac{L_f}{C_f} M \leq D_f, && \forall f \in F \text{ and } \theta_f > 0; && (13) \\
&(L_f, P_f, \theta_f, u_f) \in A_f, && \forall f \in F; \\
&C_f \in \mathbb{Z}^{\geq 0}, && \forall f \in F.
\end{aligned}$$

where H_f is the route hop number for f. As routes are given (see Assumption 3), H_f is a constant. The variables of $\mathcal{P}(G(V, E), F)$ are tuple $(L_f, P_f, \theta_f, u_f) \in A_f$ ($\forall f \in F$) and the per-frame allocated slots C_f ($\forall f \in F$).

Based on the above definition, we have

Theorem 3. Claim 1) *$\mathcal{P}(G(V, E), F)$ is NP-Hard.* **Claim 2)** *$\mathcal{P}(G(V, E), F)$ is NP-Hard even if $\forall f \in F$, its utility u_f is a non-decreasing function of L_f/P_f.*

Proof: To prove Claim 1, we can show the well-known NP-Hard knapsack problem [19] can be reduced to $\mathcal{P}(G(V, E), F)$.

An instance of knapsack problem $\mathcal{K}(\Xi, \mathsf{size}, \mathsf{value}, \Theta_s, \Theta_v)$ is as follows.

Ξ is a finite set, each item $\xi \in \Xi$ has a size $\mathsf{size}(\xi) \in \mathbb{Z}^{>0}$ and a value $\mathsf{value}(\xi) \in \mathbb{Z}^{>0}$; $\Theta_s \in \mathbb{Z}^{>0}$; and $\Theta_v \in$

$\mathbb{Z}^{>0}$. The problem asks if there is a subset $\Xi' \subseteq \Xi$, such that $\sum_{\xi\in\Xi'} \mathsf{size}(\xi) \le \Theta_s$ and $\sum_{\xi\in\Xi'} \mathsf{value}(\xi) \ge \Theta_v$.

Given knapsack problem instance $\mathcal{K}(\Xi, \mathsf{size}, \mathsf{value}, \Theta_s, \Theta_v)$, we can construct an instance of $\mathcal{P}(G(V,E), F)$ as follows.

Initially, $F = \varnothing$. For each $\xi \in \Xi$, we construct a flow f and add it into F. Flow f has the following configurations. Its source-end alternatives set A_f^{src} has only two elements

$$A_f^{\mathsf{src}} = \{(L_{f,0}^{\mathsf{src}}, P_{f,0}^{\mathsf{src}}), (L_{f,1}^{\mathsf{src}}, P_{f,1}^{\mathsf{src}})\},$$

where $P_{f,0}^{\mathsf{src}} = P_{f,1}^{\mathsf{src}} = M\tau$ seconds (τ (second) is the duration of a cell-time), $L_{f,0}^{\mathsf{src}} = 0$, and $L_{f,1}^{\mathsf{src}} = \mathsf{size}(\xi)\ell$ bits (ℓ (bit) is the size of a cell).

Correspondingly, f's in-network alternatives set A_f also has only two elements

$$A_f = \{(L_{f,0}, P_{f,0}, \theta_{f,0}, u_{f,0}), (L_{f,1}, P_{f,1}, \theta_{f,1}, u_{f,1})\},$$

where per Formula (1), $L_{f,0} = L_{f,0}^{\mathsf{src}}/\ell = 0$, $L_{f,1} = \lceil L_{f,1}^{\mathsf{src}}/\ell \rceil = \mathsf{size}(\xi)$ (cell); per Formula (2), $P_{f,1} = P_{f,0} = \lfloor P_{f,1}^{\mathsf{src}}/\tau \rfloor = M$ (cell-time); per Formula (3), $\theta_{f,0} = L_{f,0} = 0$, $\theta_{f,1} = L_{f,1} = \mathsf{size}(\xi)$ (cell); and we set $u_{f,0} = 0$, $u_{f,1} = \mathsf{value}(\xi)$.

Since the above constructed $f \in F$ is one-to-one mapped with $\xi \in \Xi$, we use function $\mathsf{map} : F \mapsto \Xi$ to denote this mapping: $\xi = \mathsf{map}(f)$, and $f = \mathsf{map}^{-1}(\xi)$.

We further construct an AFDX network $G(V,E)$ of three nodes: one source-end, connected by one TDMA crossbar switch v, to one destination end. In other words, $G(V,E)$ has the same topology of the network in Fig. 3, where $H_f = 1$. The only switch v in the network has just one input and one output, i.e., $|I^{(v)}| = |O^{(v)}| = 1$. All flows $f \in F$ originates from the only source-end, enters v via its only input and exits v via its only output to the only destination-end. Therefore, $\forall f \in F$, f's route is fixed, which has only one hop, i.e., $H_f = 1$. Besides, we set $D_f = 2M$ (cell-time), and $M = \Theta_s$.

For our constructed $G(V,E)$ and F, the corresponding problem $\mathcal{P}(G(V,E), F)$ (after simplification) becomes following.

$$\begin{aligned}
&\max \sum_{f\in F} u_f \\
\text{s.t.} \quad &\sum_{f\in F} C_f \le \Theta_s; \\
&C_f \ge L_f, && \forall f \in F; \\
&(L_f, u_f) \in \{(0,0), \\
&(\mathsf{size}(\mathsf{map}(f)), \mathsf{value}(\mathsf{map}(f)))\}, && \forall f \in F; \\
&C_f \in \mathbb{Z}^{\ge 0}, && \forall f \in F.
\end{aligned}$$

Asking "yes/no" question to the original knapsack problem $\mathcal{K}(\Xi, \mathsf{size}, \mathsf{value}, \Theta_s, \Theta_v)$ is equivalent to ask "is the constructed $\mathcal{P}(G(V,E), F)$ results in a maximum $\ge \Theta_v$".

From all above, Claim 1 holds.

Meanwhile, $\forall f \in F$ and $\xi = \mathsf{map}(f)$, we have $u_{f,0} = 0$, $L_{f,0}/P_{f,0} = 0$, $u_{f,1} = \mathsf{value}(\xi) > 0 = u_{f,0}$, and $L_{f,1}/P_{f,1} = \mathsf{size}(\xi)/M > 0 = L_{f,0}/P_{f,0}$. Therefore, u_f is a non-decreasing function of L_f/P_f. So the same proof for Claim 1 also applies to Claim 2. ■

Corollary 1. *Without Assumption 3, Theorem 3 still holds.*

Proof: Without Assumption 3, the resource planning problem becomes more general (harder). ■

Therefore, to be focused, we leave the routing problem to future work, and concentrate on cases where Assumption 3 holds in the rest of this paper.

V. Approximation Algorithm

In this section, we try to address the challenge that resource planning problem $\mathcal{P}(G(V,E), F)$ is NP-Hard (see Section IV). Specifically, we propose a re-modeling approach, upon which, we propose an approximation algorithm for $\mathcal{P}(G(V,E), F)$.

A. Notations and Re-Modeling

To ease the narration of the approximation algorithm, we need to first clarify the following notations.

Let $X \subseteq \mathbb{R}^{K\times 1}$ be a set of K-dimension vectors. X's cardinality is denoted as $|X|$. An $\vec{x} \in X$ is denoted as $\vec{x} = (x_0, x_1, \ldots, x_{K-1})^{\mathsf{T}}$, whose modulus is defined as $\|\vec{x}\| \stackrel{\mathsf{def}}{=} \mathsf{max}_{k=0,1,\ldots,K-1}\{x_k\}$. Therefore for a $\vec{x} \in X$ and a scalar $y \in \mathbb{R}$, we define $\vec{x} \le y$ if and only if $\|\vec{x}\| \le y$. We define the minimum of X to be $\min X \stackrel{\mathsf{def}}{=} \vec{x}^*$, where $\vec{x}^* \in X$ and $\|\vec{x}^*\| = \mathsf{min}_{\forall \vec{x}\in X}\{\|\vec{x}\|\}$. For $\forall \vec{x}, \vec{y} \in X$, the vector $+$ and $-$ operations are defined as $\vec{x} \pm \vec{y} \stackrel{\mathsf{def}}{=} (x_0 \pm y_0, x_1 \pm y_1, \ldots, x_{K-1} \pm y_{K-1})^{\mathsf{T}}$; while scalar-vector product is defined as $c\vec{x} \stackrel{\mathsf{def}}{=} (cx_0, cx_1, \ldots, cx_{K-1})^{\mathsf{T}}$, $\forall c \in \mathbb{R}$.

Now we introduce some additional notations to re-model the aforementioned resource planning problem $\mathcal{P}(G(V,E), F)$.

First, since the TDMA crossbar real-time switch based AFDX network $G(V,E)$ is given, let us use Π to denote the set of all ports (no matter input port or output port, no matter which switch the port belongs to) involved in G. The kth ($k = 0, 1, \ldots, |\Pi| - 1$) element of Π is denoted as Π_k.

Next, we focus on the flow set $F = \{f_0, f_1, \ldots, f_{|F|-1}\}$. For each $f \in F$, since f's route is given, we define its *routing vector* $\vec{R}_f \in \{0,1\}^{|\Pi|\times 1}$ as follows:

$$\begin{aligned}
\vec{R}_f &= (R_{f,0}, R_{f,1}, \ldots, R_{f,|\Pi|-1})^{\mathsf{T}}, \text{ where} \\
R_{f,k} &= \begin{cases} 1, & \text{if } f \text{ routes through port } \Pi_k \\ 0, & \text{otherwise} \end{cases}, \\
k &= 0, 1, \ldots, |\Pi| - 1.
\end{aligned} \tag{14}$$

Thirdly, with

$$\Lambda \stackrel{\mathsf{def}}{=} \mathsf{max}_{f\in F}\{|A_f|\},$$

we rewrite the in-network alternatives set A_f into A'_f for each flow $f \in F$ as follows.

A'_f has Λ alternatives, the kth ($k = 0, 1, \ldots, \Lambda - 1$) alternative is denoted as tuple

$$A'_{f,k} = (L_{f,k}, P_{f,k}, \theta_{f,k}, u_{f,k}, C^{\mathsf{min}}_{f,k}, \vec{C}^{\mathsf{min}}_{f,k}),$$

where as before, L_f is the in-network maximum packet length, P_f is the in-network period, θ_f is the per-frame throughput, u_f is utility and is a function of L_f and P_f, and k is the index of the alternative. The added elements for the in-network alternative tuple are $C^{\mathsf{min}}_{f,k}$ and $\vec{C}^{\mathsf{min}}_{f,k}$ defined as follows:

$$C^{\mathsf{min}}_{f,k} \stackrel{\mathsf{def}}{=} \mathsf{min}\{c |\, c \in \mathbb{Z}^{\geq 0};\ \text{and } c \geq \theta_{f,k} = \left\lceil \frac{L_{f,k}}{P_{f,k}/M} \right\rceil;$$
$$\text{and in case } \theta_{f,k} > 0, H_f M + \frac{L_{f,k}}{c} M \leq D_f\}, \quad (15)$$

where H_f is the hop number of f's given route, and D_f is f's given end-to-end real-time deadline. $C^{\mathsf{min}}_{f,k}$ represents f's the *minimum slot-per-frame demand* for each port f passes. In other words, for each port (no matter input or output) that f passes, the port has to allocate no less than C^{min}_f slots per frame to forward f, so as to meet f's throughput demand (see Formula (12)) and end-to-end real-time demand (see Formula (13)). For convenience, we further define

$$\vec{C}^{\mathsf{min}}_f \stackrel{\mathsf{def}}{=} C^{\mathsf{min}}_f \vec{R}_f, \quad (16)$$

where $\vec{R}_f$ is f's routing vector (see Formula (14)).

When $k \in \{0, 1, \ldots, |A_f| - 1\}$, the first four elements ($L_{f,k}$, $P_{f,k}$, $\theta_{f,k}$, and $u_{f,k}$) of tuple $A'_{f,k}$ are just copied from the corresponding elements in tuple $A_{f,k}$; the next two elements $C^{\mathsf{min}}_{f,k}$ and $\vec{C}^{\mathsf{min}}_{f,k}$ are derived from Formulae (15) and (16) respectively. Note this implies

$$A'_{f,0} = (0, M, 0, 0, 0, 0)$$

for all $f \in F$, which corresponds to the alternative that f is *inactive*.

When $k \in \{|A_f|, |A_f| + 1, \ldots, \Lambda - 1\}$, we define $A'_{f,k} \stackrel{\mathsf{def}}{=} (0, M, 0, 0, 0, 0)$.

For narrative convenience, $\forall f \in F$, we say "f *selects* alternative $A'_{f,k}$" or "f is *configured* (*set*) to alternative $A'_{f,k}$" if and only if f is configured such that $(L_f, P_f, \theta_f, u_f, C^{\mathsf{min}}_f, \vec{C}^{\mathsf{min}}_f) = A'_{f,k}$. We say f is *active* (*inactive*) when f is configured such that $L_f \neq 0$ ($L_f = 0$), which also implies $\theta_f \neq 0$ ($\theta_f = 0$), $C^{\mathsf{min}}_f \neq 0$ ($C^{\mathsf{min}}_f = 0$), and $\vec{C}^{\mathsf{min}}_f \neq 0$ ($\vec{C}^{\mathsf{min}}_f = 0$).

Fourthly, we introduce the notion of configuration function and the set of configuration functions.

A *configuration function* cfg is a function of $F \mapsto \{0, 1, \ldots, \Lambda - 1\}$. We denote the set of all possible configuration functions as

$$\mathcal{C} \stackrel{\mathsf{def}}{=} \{\mathsf{cfg} \,|\, \mathsf{cfg} : F \mapsto \{0, 1, \ldots, \Lambda - 1\}\}.$$

We say "we *choose configuration* $\mathsf{cfg} \in \mathcal{C}$" or "*under configuration* $\mathsf{cfg} \in \mathcal{C}$" if and only if $\forall f \in F$, f is set to alternative $A'_{f,\mathsf{cfg}(f)}$.

In addition, given integer $\bar{\varphi} \in \{0, 1, \ldots, |F| - 1\}$, we define a subset of $\mathcal{C}$:

$$\mathcal{C}_{\bar{\varphi}} \stackrel{\mathsf{def}}{=} \{\mathsf{cfg} \,|\, \mathsf{cfg} \in \mathcal{C} \text{ and}$$
$$\forall \varphi \in \{\bar{\varphi} + 1, \bar{\varphi} + 2, \ldots, |F| - 1\}, \mathsf{cfg}(f_\varphi) = 0\};$$

i.e., under any configuration $\mathsf{cfg} \in \mathcal{C}_{\bar{\varphi}}$, only flow $f_0, f_1, \ldots, f_{\bar{\varphi}}$ can be active.

Further more, given $U \in \mathbb{Z}^{\geq 0}$, we define a subset of $\mathcal{C}_{\bar{\varphi}}$:

$$\mathcal{C}_{\bar{\varphi},U} \stackrel{\mathsf{def}}{=} \{\mathsf{cfg} \,|\, \mathsf{cfg} \in \mathcal{C}_{\bar{\varphi}} \text{ and } \sum_{f \in F} u_{f,\mathsf{cfg}(f)} = U\},$$

i.e., under any configuration $\mathsf{cfg} \in \mathcal{C}_{\bar{\varphi},U}$, only flow $f_0, f_1, \ldots, f_{\bar{\varphi}}$ can be active, and the total utility is U.

We define the *minimum total slot-per-frame demand vector under constraint* $(\bar{\varphi}, U)$, simplified as *min-dmd-*$(\bar{\varphi}, U)$*-vector*, as

$$\vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi},U} \stackrel{\mathsf{def}}{=} \begin{cases} \mathsf{min}_{\forall\, \mathsf{cfg} \in \mathcal{C}_{\bar{\varphi},U}} \{\sum_{\forall f \in F} \vec{C}^{\mathsf{min}}_{f,\mathsf{cfg}(f)}\}, \\ \qquad\qquad \text{when } \mathcal{C}_{\bar{\varphi},U} \neq \varnothing; \\ \infty, \qquad\qquad \text{otherwise;} \end{cases}$$

and the corresponding *minimum total slot-per-frame demand configuration under constraint* $(\bar{\varphi}, U)$, simplified as *min-dmd-*$(\bar{\varphi}, U)$*-configuration*, as

$$\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U} \stackrel{\mathsf{def}}{=} \begin{cases} \mathsf{argmin}_{\forall\, \mathsf{cfg} \in \mathcal{C}_{\bar{\varphi},U}} \{\sum_{\forall f \in F} \vec{C}^{\mathsf{min}}_{f,\mathsf{cfg}(f)}\}, \\ \qquad\qquad \text{when } \mathcal{C}_{\bar{\varphi},U} \neq \varnothing; \\ \mathsf{nil}, \qquad\qquad \text{otherwise.} \end{cases}$$

Let

$$u_{\mathsf{max}} \stackrel{\mathsf{def}}{=} \mathsf{max}_{\forall f \in F, \forall k \in \{0,1,\ldots,\Lambda-1\}} \{u_{f,k} | \vec{C}^{\mathsf{min}}_{f,k} \leq M\}. \quad (17)$$

Then the feasible values of the network's total utility is bounded in the set $\{0, 1, \ldots, |F| \cdot u_{\mathsf{max}}\}$.

We design a dynamic programming subroutine to calculate $\vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi},U}$ and the corresponding $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U}$ for $\bar{\varphi} = 0, 1, \ldots, |F| - 1$ and $U = 0, 1, \ldots, |F| \cdot u_{\mathsf{max}}$. Then the solution to resource planning problem $\mathcal{P}(G(V, E), F)$ is

$$U^* = \mathsf{max}\{U |\, \vec{\Phi}^{\mathsf{min\_dmd}}_{|F|-1,U} \leq M\},$$
$$\text{and } \mathsf{cfg}^* = \mathsf{cfg}^{\mathsf{min\_dmd}}_{|F|-1,U^*}.$$

B. Dynamic Programming Subroutine

The dynamic programming subroutine to calculate $\vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi},U}$ and $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U}$ ($\bar{\varphi} = 0, 1, \ldots, |F| - 1$ and $U = 0, 1, \ldots, |F| \cdot u_{\mathsf{max}}$) is as follows.

When $\bar{\varphi} = 0$, we can calculate $\vec{\Phi}^{\mathsf{min\_dmd}}_{0,U}$ and $\mathsf{cfg}^{\mathsf{min\_dmd}}_{0,U}$ ($U = 0, 1, \ldots, |F| \cdot u_{\mathsf{max}}$) within $\mathcal{O}(|F| \cdot u_{\mathsf{max}} \cdot \Lambda |\Pi|)$ time: only $f_0 \in F$ can be active, and we just try every alternative of f_0 to see if its utility is U.

Let us use

$$\mathsf{cfg}_2 = \mathsf{cfg}_1 + (f', k'), \text{ where } f' \in F, k' \in \{0, 1, \ldots, \Lambda - 1\},$$

to express the modification of an old configuration function cfg_1 into a new one cfg_2, where $\forall f \in F$,

$$\mathsf{cfg}_2(f) = \begin{cases} k' & \text{(when } f = f' \text{ and } \mathsf{cfg}_1 \neq \mathsf{nil}\text{)}, \\ \mathsf{cfg}_1(f) & \text{(when } f \neq f' \text{ and } \mathsf{cfg}_1 \neq \mathsf{nil}\text{)}, \\ \mathsf{nil} & \text{(when } \mathsf{cfg}_1 = \mathsf{nil}\text{)}. \end{cases}$$

Then we can calculate $\vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi},U}$ and $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U}$ ($\bar{\varphi} = 1, 2, \ldots, |F| - 1$ and $U = 0, 1, \ldots, |F| \cdot u_{\mathsf{max}}$) using the following dynamic programming. For $U = 0, 1, \ldots, |F| \cdot u_{\mathsf{max}}$:

$$\begin{aligned} \vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi}+1,U} \quad = \quad & \mathsf{min}\{\vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi},U-u_{f_{\bar{\varphi}+1},0}} + \vec{C}^{\mathsf{min}}_{f_{\bar{\varphi}+1},0}, \\ & \vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi},U-u_{f_{\bar{\varphi}+1},1}} + \vec{C}^{\mathsf{min}}_{f_{\bar{\varphi}+1},1}, \ldots, \\ & \vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi},U-u_{f_{\bar{\varphi}+1},\Lambda-1}} + \vec{C}^{\mathsf{min}}_{f_{\bar{\varphi}+1},\Lambda-1}\}, \end{aligned} \quad (18)$$

where each element in Formula (18)'s right hand side corresponds to a configuration function, respectively $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U-u_{f_{\bar{\varphi}+1},0}} + (f_{\bar{\varphi}+1}, 0)$ (note this configuration function equals $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U}$ as $(f_{\bar{\varphi}+1}, 0)$ means $f_{\bar{\varphi}+1}$ is set to inactive), $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U-u_{f_{\bar{\varphi}+1},1}} + (f_{\bar{\varphi}+1}, 1)$, ..., and $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U-u_{f_{\bar{\varphi}+1},\Lambda-1}} + (f_{\bar{\varphi}+1}, \Lambda - 1)$. Among these configuration functions, the one that corresponds to the minimum element (i.e., $\vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi}+1,U}$) is then $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi}+1,U}$.

C. Main Body of Approximation Algorithm

With the above dynamic programming subroutine, we propose the following main body of approximation algorithm:

1) Given $\varepsilon > 0$, let $\mu = \varepsilon u_{\mathsf{max}}/|F|$;
2) For each $f \in F$ and $k \in \{0, 1, \ldots, \Lambda - 1\}$, re-define utility $u'_{f,k} = \lfloor u_{f,k}/\mu \rfloor$;
3) With the re-defined utilities $\{u'_{f,k}\}$, use the dynamic programming subroutine of Section V-B to get set $S' = \{\mathsf{cfg}^{\mathsf{min\_dmd}}_{|F|-1,U'} \mid \vec{\Phi}^{\mathsf{min\_dmd}}_{|F|-1,U'} \leq M\}$. Find the element in S' with the largest U', denote it as $\mathsf{cfg}^{\sim}$;
4) Output $\mathsf{cfg}^{\sim}$ as the configuration for the original resource planning problem $\mathcal{P}(G(V, E), F)$.

Let $U^{\sim}$ and U^* be the total utility (using the original utilities $\{u_{f,k}\}$) corresponding to $\mathsf{cfg}^{\sim}$ and the actual optimal cfg^* respectively. We can prove the following:

Theorem 4. *$U^{\sim} \geq (1 - \varepsilon)U^*$; and the approximation algorithm's time complexity is $\mathcal{O}(\frac{|F|^3}{\varepsilon}(\Lambda|\Pi| + |F|))$.*

Proof: Denote $U^{*\prime} \stackrel{\mathsf{def}}{=} \sum_{f \in F} u'_{f,\mathsf{cfg}^*(f)}$, then

$$\begin{aligned} & U^{*\prime} = \sum_{f \in F} u'_{f,\mathsf{cfg}^*(f)} \geq \sum_{f \in F} \left(\frac{u_{f,\mathsf{cfg}^*(f)}}{\mu} - 1 \right) \\ \Rightarrow \quad & \mu U^{*\prime} \geq U^* - \mu|F|. \end{aligned} \quad (19)$$

On the other hand, denote $U'^* \stackrel{\mathsf{def}}{=} \sum_{f \in F} u'_{f,\mathsf{cfg}^{\sim}(f)} = \sum_{f \in F} \lfloor u_{f,\mathsf{cfg}^{\sim}(f)}/\mu \rfloor$, we have

$$\begin{aligned} U^{\sim} &= \sum_{f \in F} u_{f,\mathsf{cfg}^{\sim}(f)} \\ &\geq \sum_{f \in F} \mu \lfloor u_{f,\mathsf{cfg}^{\sim}(f)}/\mu \rfloor = \mu U'^* && (20) \\ &\geq \mu U^{*\prime} \geq U^* - \mu|F| \ (\because \text{Formula (19)}) && (21) \\ &= U^* - \frac{\varepsilon u_{\mathsf{max}}}{|F|}|F| = U^* - \varepsilon u_{\mathsf{max}} && (22) \\ &\geq U^* - \varepsilon U^* = (1 - \varepsilon)U^*. && (23) \end{aligned}$$

Note from Formula (20) to (21) is because U'^* is the optimal solution under re-defined utilities $\{u'_{f,k}\}$. From Formula (22) to (23) is because u_{max} corresponds to the total utility of a feasible configuration with a single active flow (see Formula (17)), while U^* is the optimum, so $u_{\mathsf{max}} \leq U^*$.

As for time complexity, the approximation algorithm Step 1, 2 and 4 take $\mathcal{O}(|F|\Lambda)$ time. For Step 3, the dynamic programming calculates $\mathcal{O}(|F| \cdot |F|u'_{\mathsf{max}})$ items of $\vec{\Phi}^{\mathsf{min\_dmd}}_{\bar{\varphi},U'}$ and $\mathsf{cfg}^{\mathsf{min\_dmd}}_{\bar{\varphi},U'}$, and each item takes $\mathcal{O}(\Lambda|\Pi| + |F|)$ time. So the total time complexity is $\mathcal{O}(|F|^2 u'_{\mathsf{max}} \cdot (\Lambda|\Pi| + |F|)) = \mathcal{O}(|F|^2 \lfloor \frac{u_{\mathsf{max}}}{\mu} \rfloor (\Lambda|\Pi| + |F|)) = \mathcal{O}(\frac{|F|^3}{\varepsilon}(\Lambda|\Pi| + |F|))$. ■

VI. Related Work

Since its publication, the AFDX standard is well-received by industry. This is followed by growing volume of academic literature that intends to carry out rigorous analysis/optimizations.

Scharbarg et al. [20] and Charara et al. [21] analyzed the real-time behavior of AFDX networks upon switches that run *first-in-first-out* (FIFO) queueing. Such switch architecture let multiple flows share queues, which is fundamentally different from the per-flow queueing switch architecture that this paper is about. Hua et al. [22] also analyzed the real-time behavior of AFDX networks upon switches that runs deficit round robin scheduling. Such queueing mechanism is work-conserving [23], which is also fundamentally different from our TDMA scheduling, which is non-work-conserving.

There are also many industrial fieldbus designs, such as Foundation Fieldbus [24], CAN bus [25], TTEthernet [26], and other designs in the academic literature [27]. However, these designs are mainly for shared medium local area networking, instead of multi-hop switched networking. More importantly, unlike AFDX, they are not specialized for future avionics, and are not widely adopted by the avionics industry as the future de facto standard.

As our resource planning problem is a generalization of knapsack problem (see proof for Theorem 3), it is not surprising that one source of inspirations for our approximation algorithm is the knapsack problem approximation algorithm [28]. One of our major contributions is that through the proposal of notions such as min-dmd-$(\bar{\varphi}, U)$-vector, min-dmd-$(\bar{\varphi}, U)$-configuration etc., we discovered a way to re-model the TDMA crossbar real-time switched AFDX network resource

planning problem, so that the new model can readily borrow knapsack approximation algorithm's heuristics.

VII. Conclusion

In conclusion, we proved that TDMA crossbar real-time switched networks comply with the AFDX network standard; and derived closed-form formulae on the corresponding AFDX network's traffic characteristics and real-time end-to-end delay bound. We also proved the resource planning on TDMA crossbar real-time switched AFDX networks is NP-Hard. To address this NP-Hardness challenge, we proposed a re-modeling approach, through which, we further proposed an approximation algorithm.

Acknowledgement

The project related to this paper is supported in part by NSERC Discovery Grant 341823-07, FQRNT grant 2010-NC-131844, CRIAQ AVIO402-INTL, The Hong Kong Polytechnic University (HK PolyU) Internal Competitive Research Grant (DA) A-PJ68, HK PolyU Newly Recruited Junior Academic Staff Grant A-PJ80, HK PolyU Fund for CERG Project Rated 3.5 (DA) grant A-PK46, HK RGC General Research Fund (GRF) PolyU 5245/09E. Dr. Qixin Wang, Dr. Xue Liu, and Mr. Yufei Wang are also supported by HK PolyU Dept. of Computing start up fund, departmental visitor program, and FTE fund respectively.

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of sponsors. The authors thank anonymous reviewers for their advice on improving this paper.

References

[1] *Head-up display*. http://en.wikipedia.org/wiki/Head-up_display.
[2] *Fly-by-wire*. http://en.wikipedia.org/wiki/Fly-by-wire.
[3] *ARINC Specification 664P7: Aircraft Data Network Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network*. http://www.arinc.com.
[4] *Avionics Databus Solutions*. http://www.afdx.com.
[5] *Avionics Full-Duplex Switched Ethernet*. http://www.wikipedia.org.
[6] L. L. Peterson and B. S. Davie, *Computer Networks: A System Approach*, 4th ed. Margan Kaufmann, 2007.
[7] Q. Wang and S. Gopalakrishnan, "Adapting a main-stream internet switch architecture for multi-hop real-time industrial networks," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, May 2010.
[8] Q. Wang *et al.*, "A switch design for real-time industrial networks," in *Proc. of IEEE RTAS 2008*, Apr. 2008, pp. 367–376.
[9] F. Dopatka and R. Wismuller, "Design of a realtime industrial ethernet network including hot-pluggable asynchronous devices," *IEEE International Symposium on Industrial Electronics (ISIE 2007)*, 2007.
[10] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "On service guarantees for input buffered crossbar switches: a capacity decomposition approach by birkhoff and von neumann," *IEEE IWQoS'99*, pp. 79–86, 1999.
[11] Y.-W. Leung and T.-S. Yum, "A TDM-based multibus packet switch," *IEEE Trans. on Communications*, vol. 45, no. 7, pp. 859–866, Jul. 1997.
[12] N. McKeown, "The *i*SLIP scheduling algorithm for input-queued switches," *IEEE/ACM TON*, vol. 7, no. 2, Apr. 1999.
[13] S. Gopalakrishnan *et al.*, "Switch scheduling and network design for real-time systems," in *Proc. of IEEE RTAS 2006*, Apr. 2006.
[14] L. Chen *et al.*, "A real-time multicast routing scheme for multi-hop switched fieldbuses," *Proc. of INFOCOM'11*, pp. 3209–3217, 2011.
[15] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucuman Rev. Ser. A*, vol. 39, pp. 147–151, 1946.
[16] J. von Neumann, "A certain zero-sum two-pwerson game equivalent to the optimal assignment problem," *Contributions to the Theory of Games*, vol. 2, pp. 5–12, 1953.
[17] *List of device bit rates*. http://en.wikipedia.org.
[18] J.-Y. L. Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.
[19] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.
[20] J.-L. Scharbarg, F. Ridouard, and C. Fraboul, "A probabilistic analysis of end-to-end delays on an afdx avionic network," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 1, pp. 38–49, 2009.
[21] H. Charara, J. luc Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an afdx network," *Euromicro Conference on Real-Time Systems*, 2006.
[22] Y. Hua and X. Liu, "Scheduling design and analysis for end-to-end heterogeneous flows in an avionics network," *Proc. of INFOCOM'11*, pp. 2417–2425, 2011.
[23] J. W. S. Liu, *Real-Time Systems*. Prentice Hall, 2000.
[24] *Fieldbus Foundation*. http://www.fieldbus.org.
[25] *CAN in Automation*. http://www.can-cia.org.
[26] *TTEthernet Specification*. TTTech Computertechnik AG, 2008.
[27] R. Santos, R. Marau, A. Vieira, P. Pedreiras, A. Oliveira, and L. Almeida, "A synthesizable ethernet switch with enhanced real-time features," *Proc. of IECON'09*, pp. 2817–2824, 2009.
[28] V. V. Vazirani, *Approximation Algorithms*. Springer, 2003.