Scheduling Heterogeneous Flows with Delay-Aware Deduplication for Avionics Applications

Yu Hua, *Member*, *IEEE*, and Xue Liu, *Member*, *IEEE*

Abstract—An avionics network demands determinism and predictability. This is especially challenging because of the relatively low bandwidth of the on-board network, and the emerging needs of heterogeneous flows due to the proliferation of avionics applications. Redundant transmission and hard real-time scheduling potentially generate many duplicate data, which makes deduplication become more difficult. Many avionic flows further exhibit dynamic workloads which may change abruptly online. Hence, besides the guarantee of transmission delay, modern avionic network design needs to flexibly handle burst flows and efficiently implement data deduplication for bandwidth saving. In order to address these challenges, we propose a DeDuplication-aware Deficit Round Robin (D2DRR)-based scheduling scheme for Avionics Full DupleX (AFDX) networks with the benefits of low complexity and easy implementation. The core idea is to judiciously offer proper "*division of labor*" between switches and end systems and transform the services for heterogeneous flows to a single representation of utilization, i.e., DRR quantum, which can be flexibly reconfigured. We further leverage Bloom filters to support fast deduplication in order to reduce the load on the AFDX network. D2DRR, hence, offers salient features, elastic scheduling and adept deduplication, to deliver substantial performance improvements. Through both simulations and real implementations, extensive experimental results in an AFDX testbed demonstrate the efficacy and efficiency of our proposed schemes.

Index Terms—Cyber physical systems, avionics networks, scheduling analysis, data deduplication.

1 Introduction

The standard of Avionics Full DupleX (AFDX) switched Ethernet supports real-time data transmission among avionics subsystems. This standard builds upon protocol specifications of IEEE 802.3 [1] and ARINC 664 [2] and aims to eliminate the potential indeterminism of conventional Ethernet transmission and alleviate frame losses. The indeterministic problem, unfortunately, is not completely solved but shifted to the switch level, where heterogenous flows compete for available resources at the switch, thus resulting in potential network congestion, larger delays, jitter, and unfairness [3]. Moreover, in order to obtain high reliability, these heterogeneous flows generally are transmitted on redundant links of the AFDX network, which further exacerbates the scheduling complexity. Heterogeneous flows refer to the transmitted multitype flows, such as avionics, multimedia and best effort data packets, which have different requirements in terms of bandwidth and delay.

AFDX has been developed to offer reliable and deterministic delivery of frames for avionics applications. In order to guarantee the transmission reliability in an AFDX network, one of the most important characteristics is the redundant management on Virtual Links (VLs). Specifically, an AFDX network constructs two independent paths between end systems and redundant switches to protect the network from a failure at the MAC level. The same frame is then transmitted independently. In order to further simplify the operations at the destination end system, a redundant copy of a frame is sent within a maximum interval of 0.5 ms at the source end system [2]. The destination end system only accepts the first valid frame and discards the redundant one. The parameter in implementing redundancy management is *SkewMax*, which represents the maximum time between the reception of the original frame and its redundant copy. The value of *SkewMax* depends upon network topology, i.e., the number of switches crossed by the transmitted frames, and is defined by system administrators. Since the frames transmitted within an AFDX network come from heterogeneous flows with different priorities and the end-to-end delays meanwhile depend upon the configurations of two independent networks, performing end-to-end delay analysis becomes more challenging. Even the implementation of a simulation platform, although not impossible, requires nontrivial work.

End-to-end delays are the sum of transmission delays on the links and processing delays in switches and end systems, usually depending upon network configurations, such as available bandwidth, predefined frame length, and scheduling policies. The dominant delays come from the longest waiting service time within frame queues, where FIFO policy is often used, which assumes that all VLs have the same priority to obtain the resource *utilization*. The utilization is interpreted as the resource consumption rate within a given time interval. Each VL flow can transmit frames when it

- *Y. Hua is with the School of Computer Science and Technology, Wuhan National Lab for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: csyhua@hust.edu.cn.*
- *X. Liu is with the School of Computer Science, McGill University, Montreal H3A 2A7, Canada. E-mail: xueliu@cs.mcgill.ca.*

Manuscript received 13 Sept. 2011; revised 10 Jan. 2012; accepted 21 Jan. 2012; published online 27 Jan. 2012.
Recommended for acceptance by S. Papavassiliou, N. Kato, Y. Liu, C.-Z. Xu, and X. Wang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-2011-09-0624.
Digital Object Identifier no. 10.1109/TPDS.2012.51.

obtains enough utilization. However, it is difficult to accurately obtain *quantitative* representation of multitype flows due to their heterogeneity.

The AFDX network in the envisioned future involves heterogeneous flows, such as avionics, multimedia (video and audio) and best effort data, to monitor real-time working status and manipulate avionics devices. For instance, a pilot needs to check navigation videos from electronic cameras, monitor real-time data from installed sensors and discuss with other flight crew through audio voice, thus requiring reliable and real-time services for heterogeneous flows that have differentiated requirements for transmission delays. Besides the FIFO policy in the AFDX standard [2], Static Priority (SP) [4], and probabilistic scheduling [5] have been also proposed in an AFDX network to provide suitable flow management, which unfortunately fail to efficiently schedule heterogeneous flows that are transmitted on redundant links due to the existence of two main challenges.

Challenge 1: Indeterminism of burst flows. Avionics systems generally support real-time data transmission for burst flows when unpredictable cases occur, such as weather changes and emergency malfunction. These burst flows often result in large amounts of variable-size data packets. Important flows, e.g., avionics data with hard real-time deadlines, must be timely handled with higher priority and more transmission bandwidth, to meet the needs of the high reliability and high security of avionics applications. Unfortunately, existing schemes, such as FIFO and SP, are inefficient to deal with the indeterminism of burst flows, since they heavily rely on static "one-size-fit-all" policies for heterogeneous flows. The inflexible scheduling further exacerbates the quality of data transmission due to hard real-time requirements from avionics flows, possibly resulting in severe network congestion. Network congestion occurs when virtual links and switches carry so many data that the transmission quality deteriorates, such as larger queuing delay and more packet losses. Although existing aggressive retransmission can partially compensate for the packet loss, the hard real-time deadlines can be satisfied with small probability. Therefore, in order to address the indeterminism of burst flows, a flexible and reconfigurable scheduling scheme will be helpful and become important.

Challenge 2: Existence of duplicate copies. There are large amounts of duplicate copies in AFDX networks. The reasons are twofold. First, as mentioned above, in order to alleviate packet loss, the networks have to retransmit data whose hard real-time deadlines cannot be satisfied. Second, due to the redundant transmission property of AFDX networks, original and duplicate data have to be routed in independent paths. The destination end systems hence receive two copies of the same data. The existence of duplicate copies incurs extra computation and space overheads in end systems, thus increasing the processing delay. The destination end system becomes the potential performance bottlenecks of entire AFDX network. We hence need to offer a fast and efficient deduplication scheme.

In order to address the above challenges, we propose a deduplication-aware Deficit Round Robin, called D2DRR, that offer flexible scheduling for heterogeneous flows and implement fast deduplication. D2DRR is a proper extension of conventional DRR scheduling policy [6], [7], [8] in an AFDX network. Note that the delay bounds of an AFDX network using D2DRR scheduler are highly versatile and can be tailored for delay analysis of other networked applications. Specifically, we make the following contributions.

First, (for Challenge 1), in order to efficiently handle burst flows and alleviate network congestion, D2DRR makes use of a quantitative approach to flexibly scheduling heterogeneous flows, and unifies their utilization requirements into the quanta of the D2DRR scheduler. The different transmission requirements are then represented as a single number of allocated quanta. We leverage this property in our analysis for studying the end-to-end delays in an AFDX network. We hence transform the problem of scheduling heterogeneous flows into simple quanta computation. D2DRR is a frame-based round robin schedulers and can reduce per-packet computation cost with worst case per packet $O(1)$ complexity. This indicates that the number of operations required for selecting the next transmitted packet is constant with respect to the number of flows. D2DRR works well since it is flexibly reconfigurable by simply changing the quanta values. Our work focuses on the delay analysis of redundant transmission on two independent paths and the management of duplicate data in the switches and end systems.

Second, (for Challenge 2), in order to implement fast and accurate deduplication, D2DRR leverages space-efficient Bloom filters [9] as an index to identify potential duplicate copies. Since Bloom filters suffer from false positives with some probability, we carry out the deduplication in a two-level structure that consists of Bloom filters and a hash table. After Bloom filters identify a duplicate packet, D2DRR needs to examine its original ID in the hash table to avoid the false positives. Moreover, in order to support deletion operations in Bloom filters, we use the form of Counting Bloom Filter (CBF) [10] that uses 4-bit counters, rather than one bit. D2DRR hence provides cost-effective deduplication for avionics applications.

Third, in order to examine the performance of our proposed scheduling schemes, we build a testbed of an AFDX network to evaluate the deduplication function and end-to-end delays for heterogeneous flows. The components and interfaces of the prototype resemble those in typical AFDX networks. Our experiments in Section 5 compare D2DRR with worst case fair Weighted Fair Queuing (WF2Q) [11]. WF2Q is a data packet scheduling technique that allows different scheduling priorities to statistically multiplex data flows. Each data flow has an independent FIFO queue. WF2Q incurs the $O(\log n)$ scheduling complexity, where n is the number of active flows. In contrast, D2DRR introduces only $O(1)$ complexity by handling constant-scale packets with variable sizes. Experimental results demonstrate the efficacy and efficiency of the proposed schemes.

The rest of the paper is organized as follows. Section 2 shows the backgrounds of scheduling the flows in an AFDX network. Section 3 presents the design of D2DRR. Section 4 studies the end-to-end delays of scheduling D2DRR. Section 5 shows the simulation-based analysis. Section 6 describes the real implementation and evaluation results. Section 7 shows the related work. Finally, Section 8 concludes our paper.

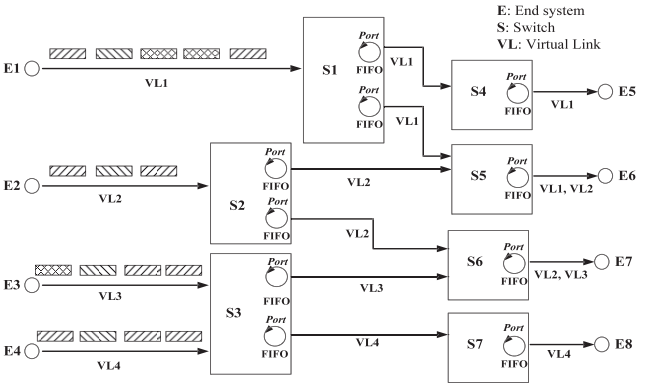


Fig. 1. The architecture of an AFDX network.

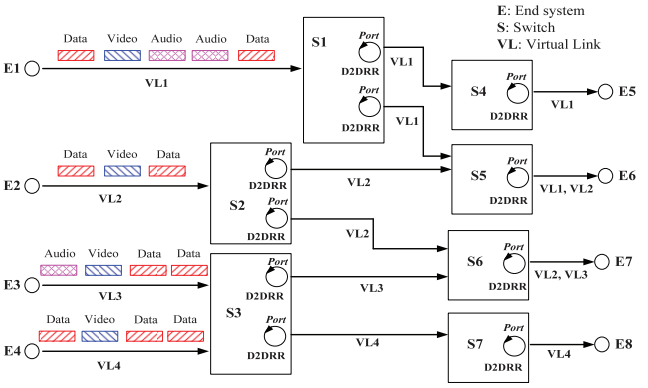


Fig. 2. AFDX network with D2DRR scheduler for scheduling heterogeneous flows.

2 Scheduling AFDX Flows

An AFDX network generally consists of avionics subsystems, interconnect networks and source/destination end systems. Specifically, the avionics subsystems include traditional on-board aircraft systems, such as Global Position System (GPS) and Flight Control System (FCS). The interconnect networks leverage a full-duplex switched Ethernet that consists of links and switches to transmit heterogeneous flows among avionics end systems. The end systems actually serve as an interface between the subsystems and the interconnect networks to guarantee real-time and reliable data transmission by using deterministic Virtual Links.

A virtual link establishes a virtual communication connection from one source end system to one or more destination end systems, forming a monosender multicast path. According to the AFDX specification [2], a VL has its own 16-bit ID, Bandwidth Allocation Gap (BAG) and the largest length of VL frames ($Lmax$). Specifically, BAG represents the minimum interval between two consecutive frames sent to a VL. An AFDX network specifies the BAG duration from 1 to 128 ms to implement bandwidth control. Moreover, $Lmax$ parameter is the largest length of VL frames. An AFDX network obtains deterministic end-to-end delays due to its static definition of VLs routing paths.

Fig. 1 shows the architecture of an AFDX network that consists of end systems, multiple interconnected switches and VLs. Each VL is characterized by its BAG and $Lmax$ parameters. The end systems serve for the input and output of entire network. A switch has a buffer at its input ports and incurs a constant delay proportional to the maximal packet length of the VLs. There is one FIFO buffer at each output port that unfortunately overlooks differentiated transmission requirements from heterogeneous flows. The FIFO buffer, thus, becomes potential transmission bottleneck due to longer queuing delay and possible network congestion when transmitting heterogeneous flows.

Heterogeneous flows are prevalent in an AFDX network. The AFDX network aggregates data flows from output ports into VLs to carry out deterministic routing. A VL beginning from a single end system transmits data to a fixed set of end systems. Moreover, an AFDX network improves upon transmission reliability by sending each frame to two independent switched networks. The destination end system then receives two copies of each frame. Conventional redundancy management mechanism within an AFDX network identifies duplicate copies by checking the predefined sequence number, which is not cost-effective in scheduling end-to-end heterogeneous flows.

The analysis of end-to-end delay relies on scheduling algorithms. The main function of a scheduling algorithm is to select the next transmitted packet and transmission time while taking into account performance metrics [12], [13]. Existing work-conserving scheduling algorithms can be classified into two categories, i.e., sorted-priority and time slot-based. The former first allocates a timestamp to each queued packet and then transmits the packets in the order of increasing timestamp. The latter divides time into slots and a packet is selected in a per-slot basis. Round-robin algorithms belong to the time slot-based method and can cyclically offer services for various flows. In order to efficiently schedule heterogeneous flows, we propose Deficit Round Robin (DRR) [6] based scheduling algorithm, while adding deduplication functionality, called deduplication-aware DRR (D2DRR), due to its ease of use and simplicity of implementation.

3 Scheduling Design in D2DRR

In this section, we first present the flexible scheduling in D2DRR. We then show the design for efficient deduplication.

3.1 Scheduling Design

We propose D2DRR scheduling policy in an AFDX network to efficiently support flexible scheduling for multiple heterogeneous flows with bounded end-to-end delays. Fig. 2 shows an example of the AFDX network using D2DRR schedulers, rather than FIFO schedulers, at output ports in switches. Each end system sends multitype flows, including avionics, multimedia (video and audio), and best effort data, which are further aggregated into VLs. VLs then transmit these multitype flows that have different transmission requirements, which are represented as different amounts of quanta.

VLs follow standard operations at switch as defined in ARINC 664 [2]. The main problem of efficiently transmitting multitype flows comes from their heterogeneity that introduces relatively high operation complexity for unified scheduling, i.e., by using a single utilization to carry out

quantitative evaluation for end-to-end delays. Recall that the utilization is the resource consumption rate within the given time interval. We handle this problem by using D2DRR scheduler for incoming flows at output ports to guarantee fairness among all transmitted flows and mitigate potential network congestion.

3.2 Homogeneous Utilization Representation for Scheduling Heterogeneous Flows

We use the DRR-based approach to scheduling heterogeneous flows that are represented by using homogeneous utilization, i.e., DRR quanta. DRR is a variation of Weighted Round-Robin (WRR) [14] through allowing flows with variable packet sizes to share available bandwidth, i.e., obtaining the utilization corresponding to their requirements. Specifically, D2DRR characterizes each flow by a quantum and a deficit variable. The quantum viewed as utilization shows the quantity of packets that a flow ideally transmits during a round. The deficit variable measures the available quantum at the current round, i.e., the balance of utilization. In each round, D2DRR allows a flow to transmit packets no more than the sum of the allocated quantum and deficit variable. After servicing for a backlogged flow, D2DRR decreases its deficit by the number of transmitted bits. When a flow cannot transmit a packet due to too large size in the current round, its allocated quantum will be added to the flow's deficit variable for the next round. D2DRR can further offer efficient scheduling upon heterogeneous flows over redundant virtual links in an AFDX network.

We use the D2DRR policy to represent utilization of heterogeneous flows in a homogeneous way to facilitate the analysis of end-to-end delays. Utilization is generally defined as the resource consumption rate within a measured time interval [15], which is also called as a flow's period. However, since not all tasks are periodic in real-world applications, existing work mainly makes use of relative deadlines to represent the measured intervals and has to introduce extra restrictive constraints, being unable to provide versatile utilization bounds for scheduling models. In order to handle this issue and further provide the efficient scheduling for heterogeneous flows, we present a unified utilization policy based on the D2DRR scheduler, in which we formulate the utilization of different flows into their allocated quanta. The quanta in fact demonstrate the amounts of resources to be used by a flow in each round. Since the utilization of all heterogeneous flows can be easily represented as a single quantum, we schedule these VL flows in a unified way by adjusting their associated quanta. The transmission requirements from heterogeneous flows are then transformed into the amounts of allocated quanta. For example, a multimedia flow demanding higher priority than best effort data can be represented as the description that the former obtains larger quanta than the latter.

3.3 Fast Deduplication by Using Bloom Filters

End-to-end delay can be decreased through fast deduplication upon duplicate packets at destination end systems with the aid of Bloom filters. A standard Bloom filter is a space-efficient data structure that consists of a bit array of M bits for representing a set $S = \{a_1, a_2, \ldots, a_N\}$ of N items. All bits in the array are initially set to 0. A Bloom filter then uses q independent hash functions $\{h_1, \ldots, h_q\}$ to map the set to the bit address space $[1, \ldots, M]$. For each item a, the bits of $h_i(a)$ are set to 1. To check whether an item a is a member of S, we need to check whether all $h_i(a)$ are set to 1. If not, item a is not in the set S. If so, a is regarded as a member of S with a false positive, which suggests that set S contains an item a although it in fact does not. In general, the false positive is acceptable if the false positive probability is sufficiently small. The false positive probability is

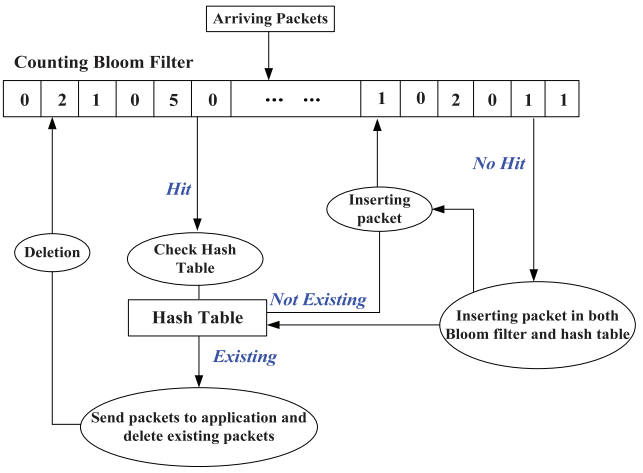


Fig. 3. Fast deduplication with the aid of counting Bloom filters.

$$Pr(False\_Positive) \approx \left(1 - e^{-\frac{qN}{M}}\right)^q = (1 - e^{-\frac{N}{m}})^q,$$

when the Bloom filter has totally M bits and q hash functions for storing N items. The probability becomes its minimum $(1/2)^q$ or $(0.6185)^{M/N}$ when $q = (M/N)\ln 2$. The detailed proof can refer to [16].

Since destination end systems need to identify and delete duplicate packets, conventional deduplication is becoming the performance bottleneck of entire end-to-end transmission. In order to decrease the delay at the end systems, we make use of Bloom filter-based deduplication upon data redundancy. Fig. 3 shows the deduplication architecture that consists of two components, i.e., a counting Bloom filter [10] and a hash table. The counting Bloom filter as a membership index is a variant from standard Bloom filters by replacing original bits with counters so as to support deletion operation. It has been well recognized that 4-bit counters can satisfy most real-world applications [16]. Moreover, the hash table is used for actually storing packets.

Deduplication operations occur on the two-level structures. First, when a packet arrives, we first hash its ID into the counting Bloom filter to check whether it is duplicate. If the counting Bloom filter says no hit, this packet is regarded as a new one. It is then inserted into both the Bloom filter and the hash table. If a hit occurs, it means that the packet is duplicate with some probability due to potential false positives. In order to avoid false positives, we carry out the indexing in the hash table. If we can find the same packet in the hash table, the packet will be deleted from both the Bloom filter and hash table, and sent to the higher level applications. If not, the packet is falsely regarded as duplicate copies by the Bloom filter, and we further insert it into both Bloom filter and hash table. Due to the very small

probability of false positives in the Bloom filter, the two-level hashing-based design can significantly reduce the delay of deduplication operations at destination end systems.

The use of Bloom filters can significantly reduce the delay of deduplication operations due to fast identification of duplicate data in a space-efficient way. The Bloom filters have been widely used or discussed in real-world deduplication systems, such as DDFS [17], Sparse Indexing [18], ChunkStash [19], SiLo [20], and cluster-based deduplication system [21].

4 End-to-End Delay Analysis

In this section, we study the end-to-end delay of transmitting heterogeneous flows through multiple intermediate switches where D2DRR scheduling is implemented at output ports. In our analysis, the end-to-end delays are first decomposed into several parts, including the delays from source end systems, transmission links, switches and destination end systems, which are then studied, respectively.

4.1 Redundant Frame Transmission

An AFDX network improves upon system reliability by transmitting each frame into two independent switched networks. The destination end system, thus, receives two copies of each frame. Conventional redundancy management mechanism relies on time-consuming brute-force approach, i.e., checking the predefined sequence number on a VL, to identifying redundant copies.

Moreover, in order to decrease the potential interference among multiple virtual links that use the same physical link, an AFDX network limits the transmission rate of frames on a virtual link, represented as BAG, to explicitly regulate the minimum transmission interval between successive frames. On the other hand, since the original frame and its redundant copy are sent within the maximum time interval of 0.5 ms, they hence produce an upper bound of 0.5 ms delay at the source end system. Note that the BAG interval is used to shape multiple frames and the 0.5 ms time interval is used for the original frame and its redundant copy.

4.2 End-to-End Delay Composition

The end-to-end delay of a VL, called $E(D_{VL})$, can be characterized by the devices that the VL passes through. Hence, the total delay is the sum of all the delays of individual devices including source end system, transmission links, switches and destination end system, i.e., $E(D_{VL}) = D_{source} + D_{transmit} + D_{switch} + D_{destination}$.

- D_{source} is the processing delay in the source end system while generating (b, r)-constrained flows as shown in Section 4.3. The source end system needs to first select a transmitted frame from a virtual link queue, then assign the per-VL sequence number, further replicate the frame in the redundant management and finally transmit the frame and its duplicate copy on the physical links.
- $D_{transmit}$ is the transmission delay over the links and mainly consists of the number of links n_l, available bandwidth b_{n_i} and frame size F_{VL}, i.e., $D_{transmit} = \sum_{i=1}^{n_l}(b_{n_i}) * F_{VL}$, which in practice depends upon environment settings and is a constant in specific network configurations.
- D_{switch} is the delay in the switches from source to destination end systems. Specifically, the delay can be divided into two parts, technological delay, d_t, which is bounded by a *constant* for specified hardware and switch capacity [22], and queuing delay, D_{D2DRR}, which depends upon the scheduling strategy. We make comprehensive analysis of delays using the D2DRR policy as shown in Section 4.4.
- $D_{destination}$ is the delay in the destination end system where each arriving frame needs to pass through integrity checking and redundancy management, since source end system generally sends each frame twice, i.e., frames F_1 and F_2, with the same sequence number to transmit within two independent networks and obtain high reliability. The end-to-end delays are, respectively, represented as D_{VL_1} and D_{VL_2}. In addition, the destination end system allows at most $SkewMax$ time to wait [2] for receiving redundant frames.

According to the above composition, we further study each part of the end-to-end delays when transmitting heterogeneous flows by using the D2DRR scheduler in an AFDX network. Moreover, we make use of leaky bucket function as arrival curve, i.e., $\alpha(t) = rt + b$, where b is the burst and r is the rate, to generate (b, r)-constrained flows. The service curve is rate delay function, i.e., $\beta(t) = R(t - T)$ with delay T and rate R.

4.3 Delay from Source End System

The delay in source end system depends upon arrival curves that produce (b, r)-constrained flows with different priorities.

Theorem 1 (Source End System Delay). *Consider n arrival nonpreemptive flows with arrival curves $\alpha_1, \alpha_2, \ldots, \alpha_n$ that are (b_i, r_i)-constrained $(i = 1, 2, \ldots, n)$ with static priorities. By using redundant transmission management in an AFDX network, the source end system delay D^i_{source} of flow i is bounded by*

$$\frac{\sum_{j:\omega_j\Delta \geq \omega_i\Delta} b_j + L_{i,max}}{C - \sum_{j:\omega_j\Delta > \omega_i\Delta} r_j} + 0.5\ ms, \tag{1}$$

where $0.5\ ms$ is the maximum time interval of original frame and its copy at the source end system, $L_{i,max}$ is the maximum length of packets in flow i, C is the output capacity, Δ is the standard allocated quantum and ω_i is a scaling factor.

Proof. When a flow i with static priorities arrives, we identify the priority by checking its allocated quantum $\omega_i\Delta$. The processed packets contain the bursts from higher priority flows, i.e., $\sum_{j:\omega_j\Delta \geq \omega_i\Delta} b_j$, and its own maximum packet, i.e., $L_{i,max}$. On the other hand, the allocated bandwidth for flow i is $C - \sum_{j:\omega_j\Delta > \omega_i\Delta} r_j$. Thus, for a single flow i, its delay can be bounded by

$$\frac{\sum_{j:\omega_j\Delta \geq \omega_i\Delta} b_j + L_{i,max}}{C - \sum_{j:\omega_j\Delta > \omega_i\Delta} r_j}.$$

AFDX network generally makes use of redundant management to transmit packets for high reliability.

Specifically, two VL frames need to be sent within the maximum interval, i.e., 0.5 ms. Thus, we can obtain the delay of flow i is bounded by

$$\left(\sum_{j:\omega_j\Delta\geq\omega_i\Delta} b_j + L_{i,max}\right) \Big/ \left(C - \sum_{j:\omega_j\Delta>\omega_i\Delta} r_j\right) + 0.5 \text{ ms}.$$

□

4.4 D2DRR-Based Switching Delay

An AFDX network utilizes the D2DRR scheduler to support the transmission of heterogeneous flows essentially by allowing the remaining quantum from previous rounds to be used for the next round. D2DRR exhibits $O(1)$ complexity if the allocated quantum of each flow is no smaller than its maximum packet size. Specifically, in this paper, the allocated quanta are scaled by the tunable factor ω according to their heterogeneous types, rather than active list number [7] and relative deadline [15].

D2DRR uses round-robin fashion to provide fair transmission service and guaranteed bounds of end-to-end delays. Initially, the deficit counter θ_i of flow i is set to 0 and the counter θ_i is increased by a quantum Δ that is further scaled by ω_i. The selected flow i thus obtains $\omega_i\Delta$ quantum for data transmission. The flow i totally has $\theta_i + \omega_i\Delta$ allowable transmitted length each round. When the current packet size in the flow i is larger than $\theta_i + \omega_i\Delta$, the packet cannot be sent in this round. However, the allocated quantum $\omega_i\Delta$ is saved and added into deficit counter for next round. When a packet is transmitted, the deficit counter is decreased by the packet length. D2DRR can also be considered as a delay-rate server [23], characterized by its worst case delay and guaranteed service rate, in order to provide affine services.

Theorem 2 (D2DRR Scheduling Delay). *Given time interval $[t_1, t_2]$, the upper bound of D2DRR scheduling delay D^i_{D2DRR} for flow i is*

$$\frac{1}{C}\left[\sum_{j=1,j\neq i}^{n}\left(\frac{L_i}{\omega_i\Delta}+1\right)\omega_j\Delta + \sum_{j=1,j\neq i}^{n} L_{j,max} + L_{i,max}\right], \quad (2)$$

where C is the output capacity, ω_i is scaling factor, Δ is the standard allocated quantum and $L_{i,max}$ is the maximum length of packets in flow i during time interval $[t_1, t_2]$.

Proof. According to the conclusion in [6], two flows i and j obtain one round-robin opportunity for data transmission between two opportunities if they are backlogged during time interval $[t_1, t_2]$, represented as

$$|opportunity(i) - opportunity(j)| \leq 1.$$

In the meantime, a packet in flow i with length L_i may obtain scheduling service if the opportunity numbers of all flows are $\lceil\frac{L_i}{\omega_i\Delta}\rceil + 1$ [6], [7], [8], which allows the flow i to have another opportunity. Thus, during interval $[t_1, t_2]$, flow i allows to send at most $\lceil\frac{L_i}{\omega_i\Delta}\rceil\omega_i\Delta + L_{i,max}$ packet length. Furthermore, a flow $j(j \neq i)$ is able to send at most $(\lceil\frac{L_i}{\omega_i\Delta}\rceil + 1)\omega_j\Delta + L_{j,max}$ packet length. Therefore, when a packet with length L_i is transmitted, the total transmitted packets for n flows during interval $[t_1, t_2]$ are

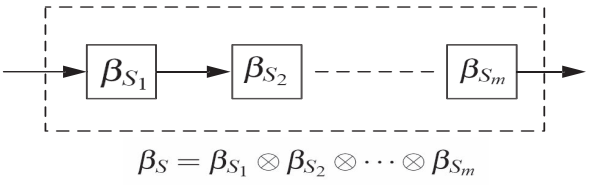


Fig. 4. A "*pay-bursts-only-once*" scenario.

$$\sum_{j=1,j\neq i}^{n}\left(\frac{L_i}{\omega_i\Delta}+1\right)\omega_j\Delta + \sum_{j=1,j\neq i}^{n} L_{j,max} + L_{i,max}.$$

When further taking into account the available output capacity C, we can obtain the final result. □

The above result generalizes previous work [6] and [7]. When setting $\omega_i = 1$ and $\omega_i\Delta = L_{i,max}$, we can, respectively, obtain delay results in [6] and [7], which are the specified cases of our analysis.

We further study the D2DRR-based service curve in a single switch and then extend it into multihop scenarios.

Theorem 3 (D2DRR-Based Service Curve). *Consider a switch serving for n arriving flows. Its service curve using D2DRR-based scheduling in a single node (i.e., no hop) during time interval $[t_1, t_2]$ is $\beta^0_{D2DRR} = R^0_{D2DRR}(t - T^0_{D2DRR})$, where $R^0_{D2DRR} = \sum_{i=1}^n \omega_i\Delta_{t_1,t_2}/(t_2 - t_1)$, $T^0_{D2DRR} = \sum_{i=1}^n D^i_{D2DRR}$, and Δ_{t_1,t_2} is the allocated quantum in the time interval $[t_1, t_2]$.*

Proof. A standard service curve is $\beta = R(t - T)$, where R and T are, respectively, service rate and delay. During time interval $[t_1, t_2]$, the ideal number of transmitted frames through D2DRR is $\sum_{i=1}^n \omega_i\Delta_{t_1,t_2}$, thus obtaining the R^0_{D2DRR}. Furthermore, the T^0_{D2DRR} is the sum of all flows through D2DRR-based scheduler-based on (2). □

We consider the multihop transformation where flows transmit through multiple switches by using "*pay-bursts-only-once*" property of network calculus [24]. We study the end-to-end delays across multiple switches as shown in Fig. 4. The concatenated switches function as a single one by taking into account the convolution operations since a burst cannot occur in all switches at the same time.

Theorem 4 (Pay-Bursts-Only-Once Concatenation). *Assume a flow traverses two servers with service curves $\beta_1(t)$ and $\beta_2(t)$, respectively. The concatenation of two servers is equivalent to a single server system with service curve $\beta(t) = (\beta_1 \otimes \beta_2)(t)$.*

The theorem can be easily proved by using the association of min-plus convolution, i.e., $((R \otimes \beta_1) \otimes \beta_2)(t) = (R \otimes (\beta_1 \otimes \beta_2))(t)$. The concatenation theorem in fact exhibits "*pay-bursts-only-once*" property that allows the end-to-end delays and backlog bounds to be scaled linearly in the number of multiple servers, rather than quadratic scaling in summing up single servers.

The upper bound of the end-to-end delay is tightly correlated with the transmission time and the aggregate waiting time in queues.

Corollary 5 (Switches Concatenation). *Consider a VL flow that routes through a switch set $S = \{S_1, S_2, \ldots, S_m\}$ represented as service curves $\beta_k, (k = 1, \ldots, m)$. The switch*

set can be concatenated into a single switch by using convolution among all switches $\beta_S = \beta_{S_1} \otimes \beta_{S_2} \otimes \cdots \otimes \beta_{S_m}$.

Corollary 6 (Multihop Switching Delay). *Consider a VL flow* i *routing through* m *switches. The total delay from intermediate switches,* D_{switch}*, is*

$$\sum_{k=1}^{m}\left(T^k_{D2DRR} + \frac{\max_{\omega^k_i>\omega^k_j}\{L^k_{j,max}\}}{R^k_{D2DRR}} + \frac{L^k_{i,max}}{\min_{\forall j\neq i}(R^k_{D2DRR} - r_j)}\right).$$

Therefore, we obtain the delays of a flow before it arrives at the destination, called "*Before-destination delay*," to facilitate further analysis of redundant management in the destination end system.

Definition 1 (Before-Destination Delay). *Before arriving at destination end system, the delay of each* VL *flow is* $D_{VL} = D_{source} + D_{transmit} + D_{switch}$.

4.5 Destination End Delay

Destination end system needs to spend processing time, called D_{dest}, on carrying out integrity check to guarantee accurate transmission and redundant management to remove duplicate copies. However, since two frames with the same sequence number transmit within independent networks and route in different paths, the time arriving at destination end system depends upon device configuration and link status, thus leading to some randomness and becoming difficult to accurately predict in advance. Parameter $SkewMax$, therefore, defines the maximum allowable time between valid frames according to the specification in an AFDX network [2]. When $SkewMax$ value for a VL is exceeded, the integrity check is reset to accept the next valid frame, regardless of its sequence number.

4.6 End-to-End Delay

The end-to-end delay of transmitting a frame in a VL, represented as $E(D_{VL})$, essentially depends upon the redundant transmission in an AFDX network. Specially, the end-to-end delay of a frame is the sum of delays experienced at all hops from the source to the destination end systems. The lower bound of delay is $D(h_1)$, which assumes that a frame in a VL transmits through totally h_1 hops and is processed at intermediate nodes without any queuing delays. On the other hand, the upper bound is represented as D_{VL} as shown in Definition 1. In practice, an AFDX network makes use of redundant transmission to guarantee the reliability and protect the network from a failure. Fig. 5 shows the end-to-end transmission delays of original flow VL_1 and its corresponding duplication VL_2. $F(VL_1)$ and $F(VL_2)$ represent their end-to-end delays with a maximum 0.5 ms time difference when sending them at the source end system [3], [4], [25]. At the destination end system, the membership of a frame that has first arrived can be maintained for a maximum time, i.e., $SkewMax$, to discard its redundant one that transmits through another path. Fig. 5 uses $Arrive(VL_1)$ and $Arrive(VL_2)$ to, respectively, represent the random arriving time of flow VL_1 and VL_2.

Therefore, we obtain the end-to-end delay using D2DRR scheduling in an AFDX network.

Corollary 7 (End-to-End Delay). *The upper bound of end-to-end delay of a flow transmitting through an AFDX network is*

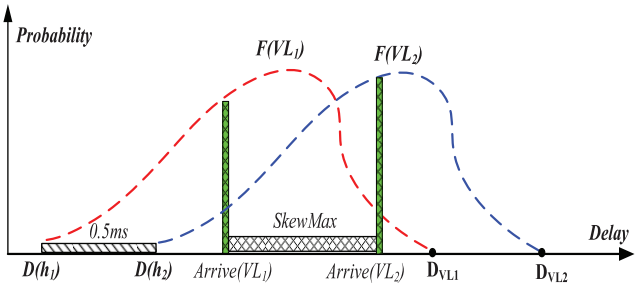


Fig. 5. Redundant end-to-end transmission delays for flows VL_1 and VL_2.

$$E(D_{VL}) = \begin{cases} D_{VL_1} + D_{dest} \\ \quad + SkewMax, & \text{if } 0.5 \text{ ms} > SkewMax, \\ D_{VL_2} + D_{dest}, & \text{if } 0.5 \text{ ms} \leq SkewMax. \end{cases}$$

where D_{VL_1} *and* D_{VL_2}*, respectively, represent the delays of original and duplicate VL flows,* $SkewMax$ *is maximum time between the valid frames.*

Proof. The upper bound of end-to-end transmission delays depends upon the parameter setting, i.e., 0.5 ms and $SkewMax$. When 0.5 ms is larger than $SkewMax$, flow VL_1, which first arrives, needs to spend a D_{dest} time to process the frame and then wait a $SkewMax$ time to complete the end-to-end transmission. On the other hand, when 0.5 ms is smaller than $SkewMax$, the maximum value of end-to-end delay comes from the flow VL_2 and the upper bound becomes D_{VL_2} and the processing time D_{dest}. □

4.7 Parameter Analysis for Quanta

One key issue in an AFDX is to accurately determine the quanta of heterogeneous flows for D2DRR-based scheduling. The quanta allocation is nontrivial since it depends upon the specified network configuration and unpredictable bursts of network traffic. If the quanta is too large, the current flow allowing to use will occupy the scheduler for a long time and other flows have to wait, thus resulting in long delays and potential packet loss. On the other hand, if the quanta is too small, the scheduler will run for many rounds, in which few packets can be transmitted, and most flows have to wait to obtain larger quanta, also leading to potential delays and packet loss.

In order to select the proper quanta for network implementations, we make use of a popular experiment-based methodology, i.e., sampling approach [26], [27], to accurately identify the optimal values to facilitate the analysis of the end-to-end delays. Specifically, we first construct a simulation platform according to the specifications in real-world avionics applications [2], [3] to find the quanta for multiple heterogeneous flows, represented as different ω values. The sampling-based approach examines end-to-end delays under different ratios of allocated quanta with standard setting $\Delta = 1{,}000$ and $\text{BAG} = 32$ ms, when taking into account a heavy-load star topology that runs at 100 Mbps.

Fig. 6 shows the end-to-end delays of heterogeneous flows in terms of increased ratios of $\omega(Avionics)$ to $\omega(Data)$ and $\omega(Multimedia)$ to $\omega(Data)$. We finally select the optimal ratio of three typical flows to be 6:3:1, respectively,

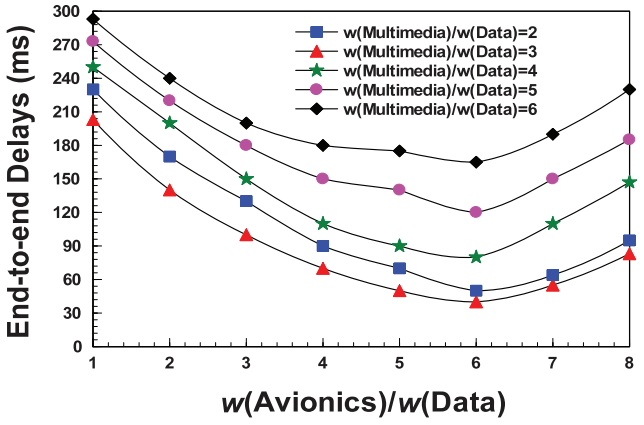


Fig. 6. End-to-end delays for different ratios of allocated ω values.

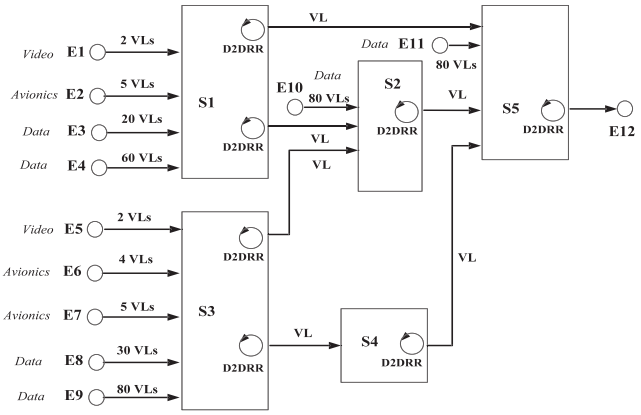


Fig. 7. An AFDX prototype configuration under one partial scenario.

for avionics, multimedia and best effort data to facilitate the following experiments.

4.8 Bloom Filter and Hash Table for Deduplication

We leverage a counting Bloom filter and a hash table to offer efficient deduplication for avionics applications as described in Section 3.3. We first study the false positive probability in the Bloom filter. In order to address the potential false positives, we make use of the hash table to verify the duplicate data.

A counting Bloom filter (CBF) consists of an array of M counters. Each counter contains 4-bits-based on the conclusion in [10], which has been accepted in the research community and widely used in real-world applications [16], [28], [29], [30], [31], [32]. All values are initialized to 0. We then represent a set $S = \{a_1, a_2, \ldots, a_n\}$ of n items with the aid of the computation of q independent hash functions $\{h_1, \ldots, h_q\}$. These results in fact map the set S to the counter address space $[1, \ldots, M]$.

We insert an item a into the space-efficient Bloom filter and in the meantime increase the hashed counters of $h_i(a)$ by 1. By checking whether all $h_i(a)$ are equal to or larger than 1, we determine the membership of item a in a set S. If there exists a counter 0, item a is not a member in the set S. Otherwise, item a is considered as a member of set S with certain false positive probability [10], [16]. A false positive means that an item a is considered in the set S although it is actually not.

Theorem 8. *The false positive probability is* $f_{CBF} \approx (1 - e^{-\frac{qn}{M}})^q = (1 - e^{-\frac{n}{m}})^q$ *when the Bloom filter contains* M *counters and* q *hash functions for* n *items. The probability achieves the minimum* $(1/2)^q$ *or* $(0.6185)^{M/n}$ *when* $q = (M/n)\ln 2$.

Proof. Assume that a Bloom filter has q random hash functions, each of which is associated with m ($m = M/q$ and $m > n$) counters. The probability that a particular counter in an array is equal to or larger than 1 by a hash function is $\frac{q}{M}$. Moreover, the probability that a counter is not increased by the hash function is $(1 - \frac{q}{M})$ (or $(1 - \frac{1}{m})$). After inserting n items into the Bloom filter, the probability that a counter is still 0 is $(1 - \frac{q}{M})^n \approx e^{-\frac{qn}{M}}$. Thus, the false positive probability in the counting Bloom filter is $f_{CBF} = (1 - (1 - \frac{q}{M})^n)^q \approx (1 - e^{-\frac{qn}{M}})^q = (1 - e^{-\frac{n}{m}})^q$ because indexed counters in q arrays must be equal to or larger than 1.

It is easy to check that the minimum of the false positive probability $min(f_{CBF}) = (1/2)^q \approx (0.6185)^{M/n}$ when $q = \ln 2(M/n)$. □

Obviously, the probability of false positives decreases as M increases, and increases as n increases. Therefore, a CBF can decrease its probability by increasing the storage space (M) for a fixed number of stored items.

5 Performance Evaluation

In this section, we present the experimental results from the simulation of D2DRR-based scheduler to schedule heterogeneous flows. Current AFDX standards support FIFO [33] scheduling. In order to efficiently examine the performance, besides FIFO and static priority scheduling policies (i.e., Rate Monotonic Analysis (RMA) [34][1] that generally exhibits pessimistic worst case estimate to determine whether a particular set of tasks can be scheduled in a given environment), we compare D2DRR with WF2Q [11] that can also schedule heterogeneous flows. The simulation results further guide the system implementation in practical devices. It is worth noting that the traditional DRR [6] mainly schedules data packets in the intermediate switches/routers, not including packet management in destination end systems. Instead, D2DRR provides a comprehensive end-to-end scheduling solution and traditional DRR is not comparable with the proposed D2DRR due to their different functionalities in networked systems.

5.1 Simulation Setup

Our simulations are configured according to the specifications of AFDX networks [2]. Our experiments evaluate the end-to-end delays of different scheduling policies. Fig. 7 shows the network configuration for our experiments, which include 11 source end systems, 1 destination end system, and 5 switches. In order to facilitate comparisons, we also implemented FIFO, SP, and WF2Q. We use 16-bit values as virtual link IDs to route Ethernet frames in the entire AFDX network. A VL transmission ends when it

1. RMA generally exhibits pessimistic worst case estimate to determine whether a particular set of tasks can be scheduled in a given environment.

TABLE 1
Parameter Settings for BAG and Frame Length

Avionics Flows			
BAG (ms)	Number of VLs	Frame Length (bytes)	Number of VLs
2	5	0-100	180
4	25	100-200	70
8	35	200-400	35
16	70	400-600	20
32	90	600-800	15
64	60	800-1000	10
128	50	> 1000	5
Multimedia Flows			
BAG (ms)	Number of VLs	Frame Length (bytes)	Number of VLs
2	2	0-100	0
4	2	100-200	1
8	5	200-400	5
16	10	400-600	8
32	24	600-800	12
64	16	800-1000	18
128	15	>1000	25
Data Flows			
BAG (ms)	Number of VLs	Frame Length (bytes)	Number of VLs
2	2	0-100	5
4	2	100-200	8
8	6	200-400	16
16	25	400-600	35
32	60	600-800	50
64	32	800-1000	20
128	20	>1000	18

passes through integrity and redundant checking as shown in Section 4.5.

In our simulation, we take into account three types of heterogeneous flows, i.e., avionics, multimedia (video and audio) and best effort data, to comprehensively evaluate the proposed scheme in terms of end-to-end delays. Table 1 demonstrates the BAG and frame sizes of avionics, multimedia and best effort data flows. These settings come from a synthetic scenario to simulate a real AFDX network. All links run at 100 Mbps. All BAGs are limited within 128 ms and most frames are smaller than 1,000 bytes according to the AFDX specification [2] and the experiences from real-world environments [2], [5], where avionics VLs often show heavier workloads than multimedia and best effort data flows.

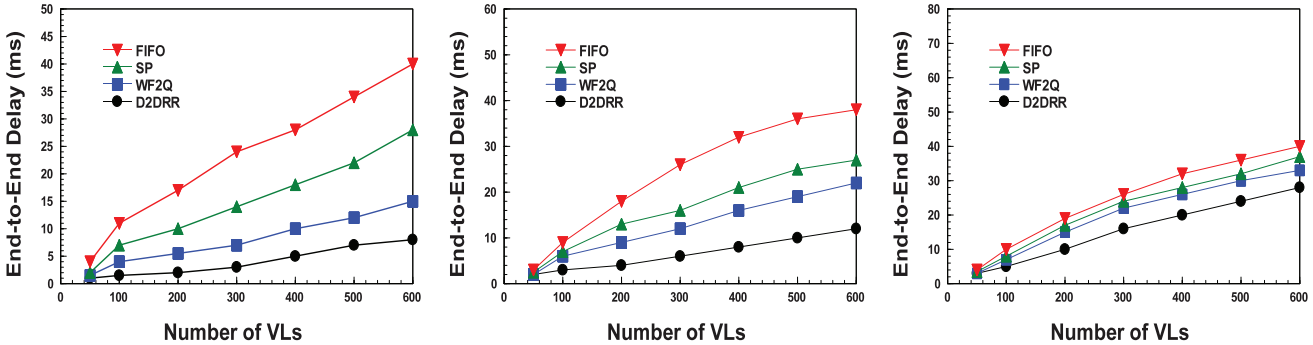
We evaluate end-to-end delays of heterogeneous flows that are periodically generated. We define the "periodicity" that is a period for transmitting a periodic message. The simulations set the periodicity to be 40 ms, which actually serves as traffic shaping function. Therefore, for periodic flows with BAGs smaller than 40 ms, we replay them every 40 ms and others are replayed in the time interval of their own BAG values. The simulations examine the end-to-end delays of variable loads that are represented as the number of transmitted VLs. We examine the end-to-end delays according to the constrained BAG values from 2 to 128 with exponential growth of 2. The experimental results are the average values of 50 runs.

5.2 End-to-end Delays

Fig. 8 shows experimental results when executing FIFO, SP, WF2Q, and D2DRR for scheduling avionics, multimedia (video and audio), and best effort data. The end-to-end delays are evaluated with the increased numbers of VLs. Four scheduling policies produce different end-to-end delays due to their different strategies for heterogeneous flows. D2DRR obtains the best performance since it can significantly alleviate the network congestion and decrease retransmission probability.

For different scheduling policies, we observe that the delay of FIFO is the longest while that of D2DRR is the shortest. FIFO produces, on average, 3.2 times longer delay than D2DRR. FIFO equally treats heterogeneous flow that has different deadlines of hard real-time transmission, which leads to the longest delays among the schedulers. WF2Q and SP take into account the flow priorities and a flow is thus allowed to transmit when the queues of all higher priority flows are empty. These two scheduling schemes essentially suffer from inflexible configuration and deteriorate end-to-end transmission quality. In practice, due to no real-time reconfiguration, they become inefficient to handle burst flows that are heterogeneous and meanwhile demand variable transmission. Instead, D2DRR offers flexible reconfiguration and supports fast deduplication. It significantly reduces the end-to-end latency. D2DRR can handle the packets of variable sizes without knowing their mean size.

For scheduling heterogeneous flows, the evaluation of end-to-end delays involves in avionics, multimedia (video and audio), and best effort data VLs. Their performance depends upon frame sizes and scheduling policy. First, as shown in Fig. 8a, we observe that the average latencies in avionics flows are, respectively, 4.6 ms in D2DRR, 12.1 ms in WF2Q, 17.6 ms in SP, and 29.8 ms in FIFO. D2DRR can guarantee the hard real-time requirements for avionics flows. Moreover, as shown in Fig. 8b, the latencies of multimedia flows are, respectively, 8.2 ms in D2DRR, 17.6 ms in WF2Q, 21.2 ms in SP, and 30.4 ms in FIFO. D2DRR delivers substantial performance improvements due to its flexible



(a) Avionics flows. (b) Multimedia flows. (c) Data flows.

Fig. 8. End-to-end delays of heterogeneous flows under FIFO, SP, WF2Q, and D2DRR schedulers.

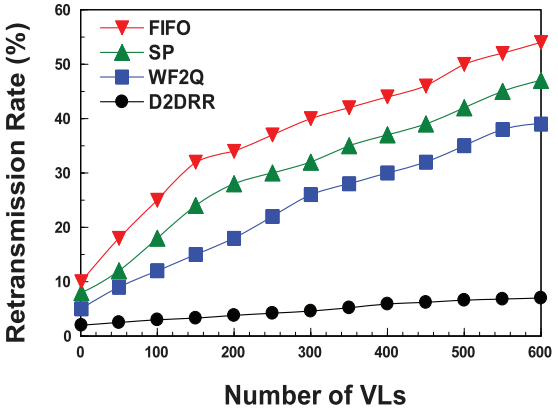


Fig. 9. Retransmission rates of heterogeneous flows.

scheduling. In addition, as shown in Fig. 8c, the average latencies of data flows are 12.5 ms in D2DRR, 22.7 ms in WF2Q, 26.3 ms in SP, and 30.9 ms in FIFO. Note that compared with FIFO, SP, and WF2Q, the end-to-end latency for the data flows in D2DRR is much shorter. The reason is that D2DRR alleviates the number of data to be retransmitted, and further mitigates network congestion.

5.3 Retransmission Rate

The number of retransmitted data is a key metric to evaluate the end-to-end transmission performance. In order to provide efficient reliability, data are generally retransmitted when packet loss or timeout occurs. Fig. 9 shows the retransmission rates of scheduling heterogeneous flows. We observe that compared with FIFO, SP, and WF2Q, D2DRR can significantly reduce retransmission rate. The reasons is that D2DRR reduces end-to-end delay and satisfies the deadlines of most transmitted data. D2DRR hence decreases the number of time-out packets. Moreover, D2DRR also reduces the number of packets waiting in the queues of intermediate switches, hence mitigating packet loss.

6 The Results from System Implementation

In this section, we present the end-to-end transmission performance by using a real system implementation. Based on our analysis from simulation results, one of main performance bottlenecks comes from data deduplication at end systems. We, hence, leverage a space-efficient Bloom filter [9] with $O(1)$ time complexity to carry out the fast deduplication, thus obtaining significant performance improvements in a real implementation of the AFDX testbed to examine the real-time transmission performance.

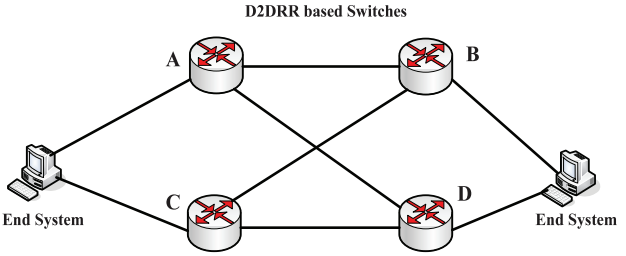


Fig. 10. System implementation topology.

6.1 Experimental Environments

We have built a real testbed to examine the actual performance of our proposed schemes. Fig. 10 shows the network topology that consists of four switches between source and destination end systems. Each switch uses the D2DRR-based scheduling to handle incoming data packets. At the end systems, we embed the Bloom filter-based structures for fast deduplication. The virtual links consist of either switch $(A, B)\&(C, D)$ or switch $(A, D)\&(C, B)$ to facilitate reliable data transmission.

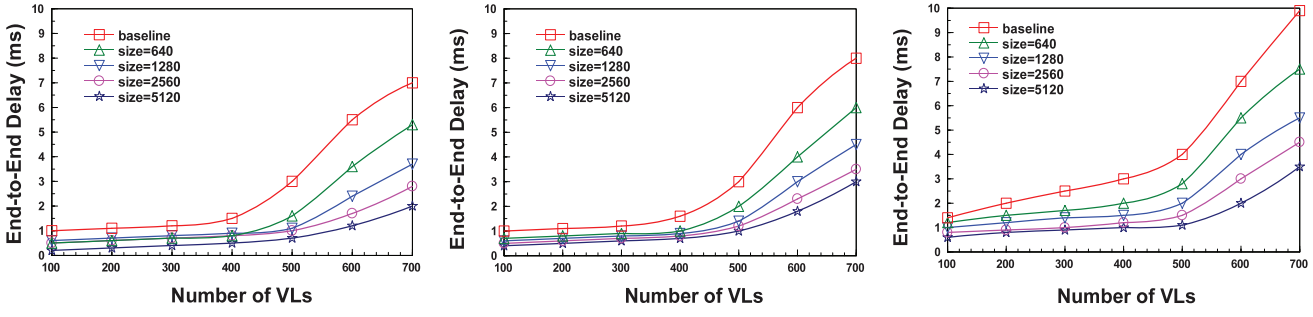
The end systems of the testbed run on Linux kernel 2.4.21 environments that use 3.2 GHz Dual Core processor with 4 GB RAM. The baseline quantum of D2DRR scheduling is set to $\Delta = 1{,}000$ and we keep adaptive $\sum_{i=1}^{n} \omega_i = 1$ for all i flows, in which $\omega^{avionics} : \omega^{multimedia} : \omega^{data} = 6 : 3 : 1$ according to the observation of sampling-based approach in Section 4.7. We make use of the same parameter settings as simulations for the network traffic (e.g., BAG and frame length). Furthermore, the Bloom filters used in our implementation demonstrate the different end-to-end delays with variable filter sizes.

6.2 Results and Analysis

We demonstrate the performance evaluation in terms of end-to-end transmission delays and per-switch throughput.

6.2.1 End-to-End Delays

We examine the end-to-end delays when scheduling heterogeneous flows. The end systems have employed the counting Bloom filters for fast data deduplication. We further adjust the filer sizes to examine the actual performance. Fig. 11 shows the end-to-end delays. Note



(a) Avionics flows. (b) Multimedia flows. (c) Data flows.

Fig. 11. End-to-end delays of scheduling heterogeneous flows using D2DRR with different Bloom filter sizes (No. of counters) in real implementations.

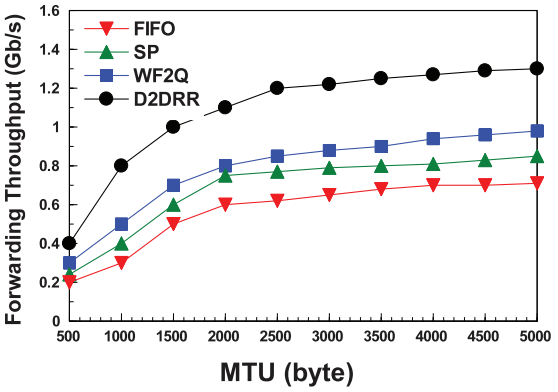


Fig. 12. Average per-switch throughput with different MTU sizes.

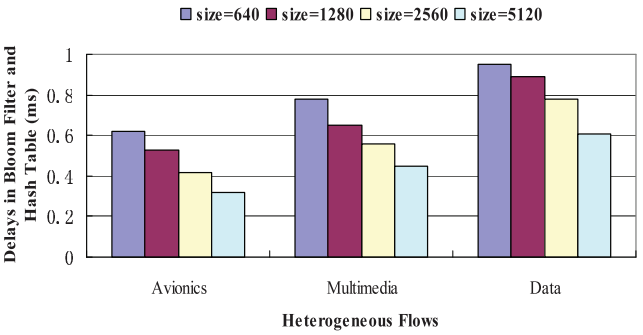


Fig. 13. The delays in Bloom filter and hash table for deduplication.

that the baseline demonstrates the transmission delay without using Bloom filters. We observe that compared with the baseline scheme, avionics flows, using 640, 1,280, 2,560 and 5,120 counters, on average decrease the end-to-end delays, respectively, by 24.6, 29.2, 35.7, and 43.5 percent. For multimedia flows, these decrements are respectively by 28.7, 35.2, 44.9, and 56.1 percent and for data flows, those are 31.6, 40.2, 51.3, and 62.6 percent. The main reasons come from the $O(1)$ complexity for fast deduplication in Bloom filters. The hash computation can significantly simplify the operation complexity at end systems.

The sizes of Bloom filters have the impact on the end-to-end delays as shown in Fig. 11. In general, the larger size the Bloom filter has, the smaller the false positive probability is. A false positive possibly results in an unnecessary checking operation on the hash table, thus incurring extra delays. Hence, we need to make this sampling over different filter sizes to obtain a suitable tradeoff between space overhead and end-to-end transmission delays.

6.2.2 Per-Switch Throughput

In order to examine per-switch throughputs under different scheduling policies, we vary the Maximum Transmission Unit (MTU) size of the NICs and evaluate the forwarding throughput in the AFDX switches. Fig. 12 illustrates the experimental result. Compared with the baseline FIFO scheduling, D2DRR obtains, on average, one time throughput improvement. Furthermore, the throughput of using D2DRR is also much larger than that of WF2Q by 41.8 percent and SP by 70.2 percent. The reason is that SP and WF2Q make use of static and "one-size-fit-all" mechanism to schedule data flows, which fails to efficiently handle the potential network congestion. Instead, D2DRR can flexibly schedule heterogeneous flows and improve the utilization of network resources, thus efficiently alleviating the effects of the congestion and meanwhile significantly enhancing the throughput.

6.2.3 Delays of Deduplication Structures

We examine the delays in deduplication structures, i.e., the Bloom filter and hash table, as shown in Fig. 13. We observe that the deduplication structures produce limited processing delays, on average, 0.42 ms, 0.56 ms, and 0.78 ms, respectively, for avionics, multimedia, and data flows. The delays thus become a small fraction of entire end-to-end delays, compared with the results in Fig. 11. We argue that the deduplication structures actually introduce slight processing overhead. Moreover, as described in Section 6.2.1, the sizes of Bloom filters produce some impacts on the delays since larger size can reduce the false positives and further alleviate the verification operations on the hash table.

7 Related Work

AFDX Ethernet technology offers reliable deterministic network service and guaranteed bandwidth by using redundant virtual links for end-to-end transmission. This technology has recently received the attention from both industry and academic fields [2], [3].

Existing theoretical analysis and simulation-based design mainly focus on the end-to-end delays when leveraging FIFO and static priority [5] scheduling. In order to examine the end-to-end delays of using FIFO scheduling in switch outputs, three methods, including network calculus, queuing networks simulation and model checking, are compared [3]. Deterministic network calculus often gives a guaranteed upper bound of end-to-end delay. Given an exceedable probability, Ridouard et al. [4] take into account the transmission of three heterogeneous flows, including avionic, multimedia and best effort data flows, and leverage FIFO and static priority scheduling. Moreover, stochastic network calculus [35], [36] evaluates the distribution of end-to-end delays that often demonstrate pessimistic results. In order to improve the computation of end-to-end delay bounds, Cruz [25] presents network calculus-based design and handle shaped leaky buckets flows traversing simple rate-latency network elements, while taking into account global aggregated flow and FIFO policy. Unlike D2DRR, they rarely take into account the redundant management upon dynamic and heterogeneous flows.

A formal model of the AFDX frame management proposed in [37] leverages a network of timed automata to reveal the vulnerability. In order to address the problems, the redundancy management and integrity checking are integrated by using priority queue and duplication of reset message. Moreover, traffic shapers in AFDX networks [38] use static priority [39], [40] and weighted fair queuing policies [41] in the switch to guarantee real-time behaviors while managing the traffic.

Utilization-based schedulability analysis becomes an efficient tool for the design and implementations of real-time systems by deriving utilization bounds [42]. Utilization-bound schedulability analysis using Weighted Round

Robin [15] maximizes utilization bounds by establishing a unified modeling framework to facilitate the derivation of utilization bounds. Recently, nonutilization-based schedulability analysis has been studied in [43], [44]. There still lacks the study of end-to-end delay analysis for heterogeneous flows, especially transmitted within independent networks for reliability concerns such as in the design of AFDX network. Instead, D2DRR is flexible and efficient to schedule heterogeneous flows and further satisfy the real-time requirements for AFDX networks.

Compared with our conference version [45], this paper has made significant improvements. We add detailed backgrounds and characteristics analysis of AFDX networks. In order to further accelerate the processing operations and decrease the waiting time at the destination end systems, we propose and implement the deduplication functionality that can fast identify redundant data.

8 Conclusion

Switched Ethernet technology provides a deterministic network with the guaranteed service to support real-time data transmission in real-world avionics applications. The determinism provides a worst case upper bound of end-to-end transmission delays of Virtual Links that are often assumed to be homogeneous and have similar transmission requirements. Due to potential network congestion and redundant transmission property in an AFDX network, it is still a challenging problem to perform real implementations and study the end-to-end transmission delays of heterogeneous flows.

An AFDX network offers reliable and deterministic delivery of frames for avionics applications by using redundant links. In order to address the problems of indeterminism of burst flows and wide existence of duplicate copies, this paper proposes DeDuplication-aware DRR, called D2DRR, to provide flexible reconfiguration and fast deduplication. D2DRR offers the salient features of simplicity and ease of use. D2DRR judiciously leverages conventional DRR with the functionality improvements upon deduplication. We hence meet the needs of easy implementations of D2DRR that delivers satisfactory performance. This paper also studies end-to-end delays of heterogeneous flows. Different from existing state-of-the-art work, this paper focuses on studying the end-to-end delays of heterogeneous flows with different transmission requirements. We also investigate the redundant management on end-to-end delays based on parameter analysis and deduplication at the destination end systems. D2DRR supports reliable transmission of heterogeneous flows, including avionics, multimedia and best effort data. Extensive experimental results demonstrate the effectiveness and efficiency of our proposed schemes.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (NSFC) under Grant 61173043 and 60703046, Fundamental Research Funds for the central universities, HUST, under grant 2012QN098, NSERC Discovery Grant 341823-07, CRIAQ AVIO402-INTL, and University of Nebraska-Lincoln. The authors greatly appreciate anonymous reviewers for constructive comments.

References

[1] H. Frazier and C. Inc, "The 802.3z Gigabit Ethernet Standard," *IEEE Network,* vol. 12, no. 3, pp. 6-7, May/June 1998.

[2] ARINC 664 "Aircraft Data Network, Part 7 Avionics Full Duplex Switched Ethernet (AFDX) Network,"ARINC 05-005/ADN-39, 2005.

[3] H. Charara, J. Scharbarg, J. Ermont, and C. Fraboul, "Methods for Bounding End-to-End Delays on an AFDX Network," *Proc. 18th Euromicro Conf. Real-Time Systems (ECRTS),* 2006.

[4] F. Ridouard, J. Scharbarg, and C. Fraboul, "Probabilistic Upper Bounds for Heterogeneous Flows Using a Static Priority Queueing on an AFDX Network," *Proc. IEEE Int'l Conf. Emerging Technologies and Factory Automation,* pp. 1220-1227, 2008.

[5] J.-L. Scharbarg, F. Ridouard, and C. Fraboul, "A Probabilistic Analysis of End-To-End Delays on an AFDX Avionic Network," *IEEE Trans. Industrial Informatics,* vol. 5, no. 1, pp. 38-49, Feb. 2009.

[6] M. Shreedhar and G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin," *IEEE/ACM Trans. Networking,* vol. 4, no. 3, pp. 375-385, June 1996.

[7] L. Lenzini, E. Mingozzi, and G. Stea, "Tradeoffs between Low Complexity, Low Latency, and Fairness with Deficit Round-Robin Schedulers," *IEEE/ACM Trans. Networking,* vol. 12, no. 4, pp. 681-693, Aug. 2004.

[8] H. Chaskar and U. Madhow, "Fair Scheduling with Tunable Latency: A Round-Robin Approach," *IEEE/ACM Trans. Networking,* vol. 11, no. 4, pp. 592-601, Aug. 2003.

[9] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *Comm. ACM,* vol. 13, no. 7, pp. 422-426, 1970.

[10] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *IEEE/ACM Trans. Networking,* vol. 8, no. 3, pp. 281-293, June 2000.

[11] J. Bennett and H. Zhang, "WF2Q: Worst-Case Fair Weighted Fair Queueing," *Proc. INFOCOM,* 1996.

[12] D. Stiliadis and A. Varma, "Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms," *IEEE/ACM Trans. Networking,* vol. 6, no. 5, pp. 611-624, Oct. 1998.

[13] P. Brucker, *Scheduling Algorithms.* Springer Verlag, 2007.

[14] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip," *IEEE J. Selected Areas in Comm.,* vol. 9, no. 8, pp. 1265-1279, Oct. 1991.

[15] J. Wu, J. Liu, and W. Zhao, "Utilization-Bound Based Schedulability Analysis of Weighted Round Robin Schedulers," *Proc. IEEE 28th Int'l Real-Time Systems Symp. (RTSS),* pp. 435-446, 2007.

[16] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Math.,* vol. 1, pp. 485-509, 2005.

[17] B. Zhu, K. Li, and H. Patterson, "Avoiding the Disk Bottleneck in the Data Domain Deduplication File System," *Proc. Sixth USENIX Conf. File and Storage Technologies (FAST),* 2008.

[18] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, "Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality," *Proc. Seventh Conf. File and Storage Technologies (FAST),* pp. 111-123, 2009.

[19] B. Debnath, S. Sengupta, and J. Li, "ChunkStash: Speeding Up Inline Storage Deduplication Using Flash Memory," *Proc. USENIX Ann. Technical Conf.,* 2010.

[20] W. Xia, H. Jiang, D. Feng, and Y. Hua, "SiLo: A Similarity-Locality Based Near-Exact Deduplication Scheme with Low RAM Overhead and High Throughput," *Proc. USENIX Ann. Technical Conf.,* 2011.

[21] W. Dong, F. Douglis, K. Li, H. Patterson, S. Reddy, and P. Shilane, "Tradeoffs in Scalable Data Routing for Deduplication Clusters," *Proc. Ninth USENIX Conf. File and Storage Technologies (FAST),* 2011.

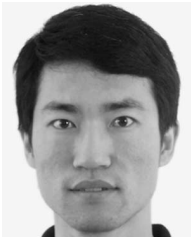
[22] Y. Jenq, "Performance Analysis of a Packet Switch Based on Single-Buffered Banyan Network," *IEEE J. Selected Areas in Comm.,* vol. 1, no. 6, pp. 1014-1021, Dec. 1983.

[23] D. Stiliadis and A. Varma, "Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms," *IEEE/ACM Trans. Networking,* vol. 6, no. 5, pp. 611-624, Oct. 1998.

[24] R. Cruz, "A Calculus for Network Delay. I. Network Elements in Isolation," *IEEE Trans. Information Theory,* vol. 37, no. 1, pp. 114-131, Jan. 1991.

[25] M. Boyer and C. Fraboul, "Tightening End to End Delay Upper Bound for AFDX Network Calculus with Rate Latency FIFO Servers Using Network Calculus," *Proc. IEEE Int'l Workshop Factory Comm. Systems,* pp. 11-20, 2008.

[26] L. Rizzo, "Dummynet: A Simple Approach to the Evaluation of Network Protocols," *ACM SIGCOMM*, vol. 27, no. 1, pp. 31-41, 1997.
[27] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker, "Scalability and Accuracy in a Large-Scale Network Emulator," *Proc. Fifth Symp. Operating Systems Design and Implementation (OSDI)*, pp. 271-284, 2002.
[28] S. Dharmapurikar, P. Krishnamurthy, and D. Taylor, "Longest Prefix Matching Using Bloom Filters," *Proc. SIGCOMM*, pp. 201-212, 2003.
[29] Y. Zhu, H. Jiang, and J. Wang, "Hierarchical Bloom Filter Arrays (HBA): A Novel, Scalable Metadata Management System for Large Cluster-Based Storage," *Proc. IEEE Int'l Conf. Cluster Computing*, pp. 165-174, 2004.
[30] Y. Hua, Y. Zhu, H. Jiang, D. Feng, and L. Tian, "Scalable and Adaptive Metadata Management in Ultra Large-Scale File Systems," *Proc. 28th Int'l Conf. Distributed Computing Systems (ICDCS)*, 2008.
[31] F. Deng and D. Rafiei, "Approximately Detecting Duplicates for Streaming Data Using Stable Bloom Filters," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 25-36, 2006.
[32] F. Bonomi, M. Mitzenmacher, R. Panigrah, S. Singh, and G. Varghese, "Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines," *Proc. SIGCOMM*, pp. 315-326, 2006.
[33] M. Bramson, "Instability of FIFO Queueing Networks with Quick Service Times," *The Annals of Applied Probability*, vol. 4, pp. 693-718, 1994.
[34] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *J. ACM*, vol. 20, no. 1, pp. 46-61, 1973.
[35] F. Ridouard, J. Scharbarg, and C. Fraboul, "Stochastic Network Calculus for End-to-End Delays Distribution Evaluation on an Avionics Switched Ethernet," *Proc. IEEE Fifth Int'l Conf. Industrial Informatics*, 2007.
[36] Y. Jiang, "A Basic Stochastic Network Calculus," *Proc. SIGCOMM*, 2006.
[37] M. Anand, S. Vestal, S. Dajani-Brown, and I. Lee, "Formal Modeling and Analysis of the AFDX Frame Management Design," *Proc. IEEE Int'l Symp. Object and Component-Oriented Real-Time Distributed Computing*, 2006.
[38] A. Mifdaoui, F. Frances, and C. Fraboul, "Full Duplex Switched Ethernet for Next Generation "1553B"-Based Applications," *Proc. IEEE 13th Real Time and Embedded Technology and Applications Symp. (RTAS)*, 2007.
[39] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings, "Applying New Scheduling Theory to Static Priority Pre-Emptive Scheduling," *J. Software Eng.*, vol. 8, pp. 284-292, 1993.
[40] H. Zhang and D. Ferrari, "Rate-Controlled Static-Priority Queueing," *Proc. INFOCOM*, pp. 227-236, 1993.
[41] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Proc. Applications, Technologies, Architectures, and Protocols for Computer Comm.*, pp. 1-12, 1989.
[42] J. Wu, J. Liu, and W. Zhao, "On Schedulability Bounds of Static Priority Schedulers," *Proc. IEEE 11th Real Time and Embedded Technology and Applications Symp. (RTAS)*, pp. 529-540, 2005.
[43] X. Liu and T. Abdelzaher, "On Non-Utilization Bounds for Arbitrary Fixed Priority Policies," *Proc. IEEE 12th Real-Time and Embedded Technology and Applications Symp. (RTAS)*, pp. 167-178, 2006.
[44] X. Liu and T. Abdelzaher, "Non-Utilization Bounds and Feasible Regions for Arbitrary Fixed-Priority Policies," *J. ACM Trans. Embedded Computing Systems*, vol. 10, 2010.
[45] Y. Hua and X. Liu, "Scheduling Design and Analysis for End-to-End Heterogeneous Flows in an Avionics Network," *Proc. INFOCOM*, 2011.



Yu Hua received the BE and PhD degrees in computer science from the Wuhan University, China, in 2001 and 2005, respectively. He was a research assistant at Hong Kong Polytechnic University and PostDoc at the University of Nebraska-Lincoln. He is an associate professor at the Huazhong University of Science and Technology, China. His research interests include cyber physical systems, computer architecture, cloud computing, and network storage. He has more than 30 papers to his credit in major journals and international conferences including *IEEE Transactions on Computers (TC)*, *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, USENIX ATC, INFOCOM, SC, ICDCS, ICPP, and HiPC. He is a member of the IEEE and USENIX.

Xue Liu received the BS degree in mathematics and the MS degree in automatic control both from Tsinghua University, China, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign, in 2006. Currently, he is working as an associate professor in the School of Computer Science at McGill University. His research interests include computer networks and communications, smart grid, real-time and embedded systems, cyber-physical systems, data centers, and software reliability. He has been granted one US patent and filed four other US patents, and published more than 120 research papers in major peer-reviewed international journals and conference proceedings, including the Year 2008 Best Paper Award from *IEEE Transactions on Industrial Informatics*, and the First Place Best Paper Award of the ACM Conference on Wireless Network Security (WiSec 2011). He is a member of the IEEE, ACM, and USENIX.