Integrating Offset in Worst Case Delay Analysis of Switched Ethernet Network With Deficit Round Robbin

Aakash Soni, Xiaoting Li, Jean-Luc Scharbarg, Christian Fraboul

Integrating Offset in Worst Case Delay Analysis of Switched Ethernet Network With Deficit Round Robbin

Aakash SONI
ECE Paris - INPT/IRIT, Toulouse
France
aakash.soni@ece.fr

Xiaoting Li
ECE Paris
France
xiaoting.li@ece.fr

Jean-Luc Scharbarg
IRIT-ENSEEIHT, Toulouse
France
Jean-Luc.Scharbarg@enseeiht.fr

Christian Fraboul
IRIT-ENSEEIHT, Toulouse
France
Christian.Fraboul@enseeiht.fr

***Abstract*—In order to handle mixed criticality flows in a real-time embedded network, switched Ethernet with Quality of Service (QoS) facilities has become a popular solution. Deficit Round Robin (DRR) is such a QoS facility. Worst-Case Traversal Time (WCTT) analysis is mandatory for such systems, in order to ensure that end-to-end delay constraints are met. Network Calculus is a classical approach to achieve this WCTT analysis. A solution has been proposed for switched Ethernet with DRR. It computes pessimistic upper bounds on end-to-end latencies. This pessimism is partly due to the fact that the scheduling of flows by end systems is not considered in the analysis. This scheduling can be modeled by offsets between flows. This modeling has been integrated in WCTT analysis of switched Ethernet with First In First Out (FIFO) scheduling. It leads to a significant reduction of delay upper bounds.**

The contribution of this paper is to integrate the offsets in the WCTT analysis for switched Ethernet with DRR and to evaluate the reduction on delay upper bounds, considering a realistic case study.

***Index Terms*—Network Calculus, DRR, Offset, Switched Ethernet**

I. Introduction

Switched Ethernet has become a popular solution in the context of embedded systems. For example, the Avionics Full DupleX switched Ethernet network (AFDX) is the de facto standard for the transmission of critical avionics flows. It implements a First-in First-out (FIFO) service discipline in switch output ports. Actually, two priority levels are available, but they are rarely used. In this avionics context, a Worst-Case Traversal Time (WCTT) analysis is mandatory, in order to ensure that timing constraints are respected. Network Calculus (NC) is classically used for this WCTT analysis [1]. It considers FIFO scheduling. This approach gives pessimistic upper bounds on end-to-end latencies, due to over estimation of network traffic and under estimation of network service for the reason of mathematical feasibility. These upper bounds can be significantly reduced by taking into account the scheduling of flows by source end systems. Indeed NC approach in [1] makes no assumption on this scheduling. Thus it considers the worst-case scenario. Taking into account this scheduling comes to associate an offset to each flow. NC approach in [1] has been extended with offsets in [2]. Since end systems interconnected by an AFDX network are not synchronized, offsets are defined between flows generated by the same end system. The approach in [2] leads to significantly lower delay upper bounds (more than 40 % on a typical industrial configuration). This extended approach can be applied with any offset assignment algorithm. In literature the effect of offset integration in different networks is shown, for instance, [3] shows the effect of offset integration in FIFO and Priority Queue in the context of CAN network. A response time analysis is also done in [4], [5], [6] for CAN messages with offset.

The fact that worst-case scenarios have a very low probability to occur leads to a very lightly loaded network. Typically, less than 10 % of the available bandwidth is used for the transmission of avionics flows on an AFDX network embedded in an aircraft. One solution to improve the utilization of the network is to introduce Quality of Service (QoS) mechanisms. Deficit Round Robin (DRR) is such a mechanism and it is envisioned for future avionics networks.

DRR scheduling was proposed in [7] in order to achieve fair sharing of network resources among the flows and it is well-known for its low complexity $O(1)$, under specific constraints, but an undeniable latency. In literature [8], [9], [10], [11] a significant improvement in latency and fairness have been proposed along with some implementation techniques while still preserving $O(1)$ complexity.

A WCTT analysis for DRR has been proposed in [9]. It is based on network calculus and it doesn't make any assumption on the scheduling of flows by end systems.

The first goal of this paper is to integrate offsets in the WCTT analysis for DRR in [9]. This integration is done in the same way as in [2]. The second goal of the paper is to evaluate the reduction brought by offsets on delay upper bounds, considering a realistic case study.

The paper is organized as follows. Section II presents the context of the study. It includes a description of network and flow models (II-A), DRR scheduling policy (II-B) and its latency (II-C), recall of NC approach (II-D) and delay bound computation using NC (II-D3). In section III, we present a method to integrate offset in NC. Section IV shows results

on a case study based on a realistic industrial configuration. Section V concludes the paper and gives directions for future works.

II. Context

A. Network model

This paper considers a real-time switched Ethernet network. The switched Ethernet network interconnects a set of end systems by full duplex links, defined by IEEE 803.1e., with maximum transmission rate of R Mbps. A flow v_i from each end system is forwarded through output port h of switch S_j in its path based on a predefined forwarding table. A set of buffers in each output port is managed by a scheduler supporting a scheduling policy like First-In-First-Out (FIFO), Fixed Priority (FP) queuing or Round Robin (RR) etc. In this paper, we consider that the network uses Deficit Round Robin (DRR) scheduler at each output port. An example of such network is shown in Figure 1 which interconnects 5 end systems ($e_1 \dots e_5$) to transfer 13 flows ($v_1 \dots v_{13}$) via 3 switches ($S_1 \dots S_3$). Each link provides a bandwidth of $R = 100Mbps$. Table I shows the temporal characteristic of each flow. Each flow v_i from a source end system initiate a sequence of frames according to the minimum inter-arrival duration T_i imposed by a traffic shaping technique. The size of each frame of flow v_i is constrained by a maximum frame length (l_i^{max}) and a minimum frame length (l_i^{min}). Each flow v_i follows a predefined path $\mathcal{P}_i$ from its source end system till its last visited output port, and then arrives at its destination end system.

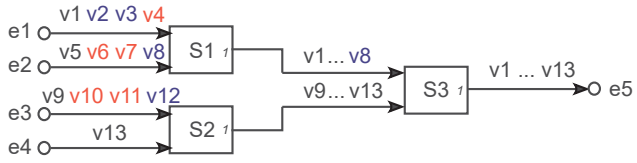


Fig. 1: Network Configuration

TABLE I: Network Flow Configuration

Flows v_i	$T_i(msec)$	$l_i^{max}(byte)$	$l_i^{min}(byte)$
v_7,	2	500	125
v_3,	2	500	250
v_{13},v_4	2	1000	500
v_2,v_{12},	4	750	125
v_5,	4	750	500
v_1,v_{11},	8	500	125
v_6,	8	750	125
v_9,	16	750	125
v_8,v_{10},	32	1000	500

Table II show the DRR scheduler configuration at each output port

B. Deficit round robin scheduler

Deficit Round Robin (DRR) was designed in [7] to achieve a better quality of service by fair sharing of available network bandwidth among the flows. DRR is basically a variation of

TABLE II: DRR scheduler configuration

Class C_x	Flows v_i	Q_x (byte)	l_i^{max} (byte)	l_i^{min} (byte)
C_1	v_1 v_5 v_9 v_{13}	1999	1000	125
C_2	v_4 v_6 v_7 v_{10} v_{11}	1999	1000	125
C_3	v_2 v_3 v_8 v_{12}	1999	1000	125

Weighted Round Robin (WRR) which allows sharing of server bandwidth among variable length flow packets. A DRR scheduler divides the flow traffic based on few predefined classes and serves each class sequentially based on the presence of a pending flow in a class buffer and the credit assigned to the class. The DRR service cycle is divided into rounds. In each round all the *active* classes are served. A class is said to be active when it has some flow packet waiting in output port buffer to be transmitted. The basic idea of DRR is to assign a credit quantum Q_x^h to each class C_x at each switch output port h. Q_x^h is the number of bytes allocated to class C_x in each round at port h. The quantum assigned to a class should be at least the maximum frame size l_x^{max} of class C_x flows at port h. At any time, an active class can receive service of Q_x^h bytes plus a deficit Δ_x^h from previous round. Each time when a class C_x is selected by the scheduler, Q_x^h is added to its deficit Δ_x^h. As long as a C_x queue is not empty and the remaining credit is larger than the size of the *head-of-line* packet, this packet is transmitted and the credit is reduced by this packer size. Thus the scheduler moves to the next active class when either C_x queue is empty or there is not enough credit to serve the next packet. In the former case, there is no deficit for the next round, i.e. $\Delta_x^h = 0$. In the latter one, the remaining credit is kept as a deficit Δ_x^h.

Algorithm 1 shows an implementation of DRR at a switch output port h with n traffic classes. First deficit is set to 0 for each class (lines 1-3). Then queues are selected in round robin order (lines 4-16). Empty queues are ignored in each round (line 6). Each non-empty queue is credited by Q_x^h added to its previous deficit Δ_x^h (line 7). The packets are sent as long as the queue is not empty and the deficit is larger than the head-of-line packet (line 8-12). If the queue becomes empty, the deficit is reset to 0 (lines 13-14).

C. Deficit round robin scheduler latency

A DRR scheduler serving n active classes at a given output port h defines a long-term service rate ρ_x^h to the class C_x, which can be computed by:

$$\rho_x^h = \frac{Q_x^h}{\sum\limits_{1 \le j \le n} Q_j^h} \times R \tag{1}$$

It is worth noting that this is a long-term service rate for class C_x, the actual service rate could be different on a smaller interval. The actual service rate can be given by the stair case curve shown in Figure 2. The DRR scheduler latency Θ_x^h experienced by a class C_x flows at output port h is defined as the delay before C_x packets are served at their long-term service rate. Before the class C_x starts receiving its service at its long term service rate, it could wait for a cumulative latency

Algorithm 1: DRR Algorithm

```
Input: Per flow quantum: Q_1^h ... Q_n^h (Integer)
Data: Per flow deficit: Δ_1^h ... Δ_n^h (Integer)
Data: Counter: i (Integer)
for i = 1 to n do
    Δ_i^h ← 0 ;
end
while true do
    for i = 1 to n do
        if notempty(i) then
            Δ_i^h ← Δ_i^h + Q_i^h;
            while (notempty(i)) and
             (size(head(i)) ≤ Δ_i^h) do
                send(head(i));
                Δ_i^h ← Δ_i^h − size(head(i));
                removeHead(head(i));
            end
            if empty(i) then
                Δ_i^h ← 0
    end
end
```

due to the nature of the DRR scheduling. This cumulative latency has been characterized in [8] and considered as two parts:

- The delay X_x^h before class C_x receives service for the first time.
- Another delay Y_x^h before class C_x receives service at long-term service rate, if it was served at reduced rate in the first round.

The delay X_x^h is due to the fact that if a class C_x flow arrives at the output port at an instant when it just missed its turn to be served in the present round, then it must wait for the next turn. It is shown in [8] that this delay is maximized when class C_x has to wait for all the other classes which are being served with the maximum transmission capacity. This maximum delay is computed as:

$$X_x^h = \frac{\sum\limits_{\substack{j=1,2,...n \\ j \neq x}} (Q_j^h + \Delta_j^{max,h})}{R} \tag{2}$$

Where $\Delta_j^{max,h}$ is the maximum deficit of class C_j at node h. Since, in any DRR schedule round, class C_j packets are served as long as the remaining credit is not smaller than the size of the head-of-line packet, thus the maximum deficit will always be smaller than the size of the largest packet $l_j^{max,h}$ of class C_j flows.

$$\Delta_j^{max,h} = l_j^{max,h} - 1 \tag{3}$$

The delay Y_x^h is based on the fact that class C_x might receive minimum service in the first round, i.e. it might be served at less than its guaranteed long-term service rate and it has to wait for the next turn to get its long-term service rate. According to [8], this delay is maximized when class C_x receives minimum service and all the other classes receives maximum service in first round. For a class C_x, the minimum used service is considered to happen when the maximum credit deficit $\Delta_x^{max,h}$ is left after its opportunity. Thus, minimum service received by any class C_x at port h is computed by:

$$minimum\_service = Q_x^h - \Delta_x^{max,h}$$

Whereas the maximum service is when all the credit is consumed by the given class.

Thus, the delay Y_x^h for the class C_x flows can be computed by:

$$Y_x^h = \frac{\Delta_x^{max,h}}{R} \left(\frac{\sum\limits_{j=1}^{n^h} Q_j^h - Q_x^h}{Q_x^h} \right) \tag{4}$$

For further explanation on derivation of equation(4), readers can refer to [8]. Finally, the DRR scheduler latency Θ_x^h experienced by a class C_x flows at output port h is given by:

$$\Theta_x^h = X_x^h + Y_x^h \tag{5}$$

D. Existing network calculus approach for end-to-end delay calculation

In this section we summarize the worst case traversal time (WCTT) analysis for DRR with Network Calculus (NC) modeled in [9].

The NC theory is based on the (min, +) algebra. It has been proposed for worst-case backlog and delay analysis in networks [12]. It models the traffic and network elements by piecewise linear curves called arrival curves and service curves respectively.

1) Arrival Curve: In NC, an arrival curve represents an over-estimation of the traffic of a flow v_i at an output port h. At any instant t, an arrival curve can be used to model a flow v_i at its source end system as:

$$\alpha_i^h(t) = (r_i \times t) + b_i, \ for\ t > 0\ and\ 0\ otherwise$$

with flow arrival rate $r_i = \frac{l_i^{max}}{T_i}$ and burst $b_i = l_i^{max}$, where T_i is the minimum inter-frame arrival time of flow v_i. A frame of flow v_i can experience jitter due to the fact that it can be delayed by other frames before it arrives at a port h. This jitter can be integrated into the arrival curve by left shifting the curve by jitter value, for more information on jitter integration readers can refer to [1].

In a DRR scheduler, the arrival traffic of a class C_x at an output port h is due to the queuing of different flows from class C_x, thus the class C_x traffic can be defined by an overall arrival curve which is the sum of individual arrival curves of each flow and is given by:

$$\alpha_{C_x}^h(t) = \sum_{i \in \mathcal{F}_{C_x}^h} \alpha_i^h(t) \tag{6}$$

where $\mathcal{F}_{C_x}^h$ is the set of C_x flows traversing port h.

2) Service Curve:: In NC, a switch output port h with maximum service rate R bits/sec and switching latency sl is modeled by a service curve:

$$\beta^h(t) = R[t - sl]^+$$

where $[a]^+$ means $max\{a, 0\}$.

In a DRR scheduler, the service is shared by all the DRR classes at the output port h and each class C_x receives a fraction of maximum service rate R based on the assigned quantum Q_x^h as shown in equation (1). Moreover, a class C_x experiences the DRR scheduler latency Θ_x^h given by equation (5). Therefore, the residual service to each class C_x is given by:

$$\beta_{C_x}^h(t) = \rho_x^h[t - \Theta_x^h - sl]^+ \tag{7}$$

In DRR scheduler, as the class C_x flows alternate between being served and waiting for DRR opportunity, the actual service curve is a staircase curve but, for computation reasons, the NC approach considers a convex curve given by equation (7) which is an under-estimation of this actual staircase curve This curve is also shown in Figure 2.

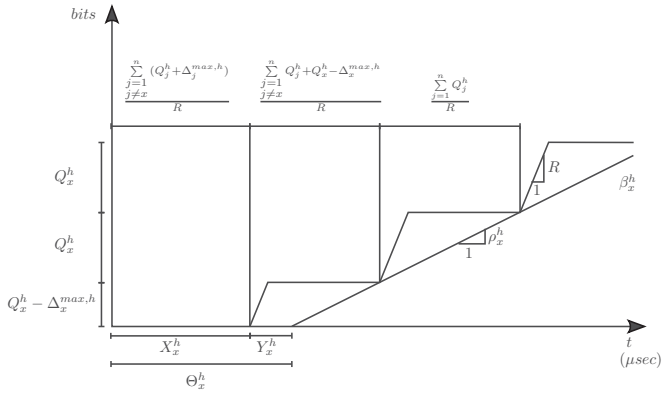


Fig. 2: NC DRR Service Curve

3) End-to-end delay bound: At a switch output port h, the delay experienced by a class C_x flow v_i is bounded by maximum horizontal difference between the arrival curve $\alpha_{C_x}^h(t)$ and the service curve $\beta_{C_x}^h$, and it is computed by:

$$D_i^h = \sup_{s \geq 0} \left(\inf\{\tau \geq 0 | \alpha_{C_x}^h(s) \leq \beta_{C_x}^h(s+\tau)\}\right) \tag{8}$$

A dataflow computation is implemented. At each output port, the output traffic curve for each flow is obtained by shifting to the left the input curve by the jitter in the port. This jitter is the maximum waiting time in the port buffer.

At the end of the process, the end-to-end delay upper bound of a class C_x flow v_i is computed by adding delays in switch output ports:

$$D_i^{ETE} = \sum_{h \in \mathcal{P}_i} D_i^h \tag{9}$$

III. Integrating Offset in NC for DRR scheduler

The computation summarized in previous section makes no assumption on the scheduling of flows by the end systems. Thus, it assumes, for any flow, the scheduling which maximizes the waiting delay in output buffers. This worst-case scheduling is modeled by Equation 6. Thus, it considers a burst of traffic where there is one frame from each flow at the same instant. This situation is most of the time impossible, since an end system distributes frame generations over time in order to produce temporal separation between transmission of frames. Such temporal separation is classically modeled by the assignment of an offset to each flow. In NC, the integration of offsets affect the computation of arrival curves. The offset integration in NC was first proposed in [2] for First-In-First-Out (FIFO) scheduler. In this paper we extend this approach for DRR schedulers.

A. DRR scheduling with offset at source end system

Scheduling of the flows emitted by a given end system is characterized by the assignment of offsets which constrain the release times of flows and, consequently, their arrivals at switch output ports.

In the context of FIFO, [2] defines two kinds of offsets:

- *definite offset* $O_{d,m}^{e_i}$ is the release time of the first frame of a flow v_m at its source end system e_i,
- *Relative offset* $O_{r,m,n}^h$ at an output port h is the minimum time interval between the arrival time of a frame f_m from a benchmark flow v_m in a port h and the arrival time of a following frame f_n from flow v_n in the same port h.

Definite offsets are fixed by scheduling of flows by end systems, while *Relative offsets* are computed from this scheduling.

The computation of a *Relative offset* $O_{r,m,n}^h$ depends on the considered port h.

In a source end system port, it is implemented by considering all frame generations within a time interval which includes all possible situations (e.g. twice the least common multiple of flow periods). $O_{r,m,n}^h$ is the smallest possible duration between one frame from v_m and one frame from v_n within this interval. For details about the computation of offset at source end system readers can refer to the algorithm given in [2].

In a switch output port, the computation of *Relative offset* $O_{r,m,n}^h$ has to take into account flow jitters. Typically, f_m delay between the source end system and the considered switch output port can be longer than f_n delay, leading to a smaller *Relative offset*. From [2] $O_{r,m,n}^h$ in node h is computed by considering that f_m experiences its maximum delay $D_{max,m}^{e_i,h}$ between its emission at source end system e_i till its arrival at output port h, while f_n experiences its minimum delay $D_{min,n}^{e_i,h}$. Thus, $O_{r,m,n}^h$ is given by:

$$O_{r,m,n}^h = O_{r,m,n}^{e_i} + D_{min,n}^{e_i,h} - D_{max,m}^{e_i,h} \tag{10}$$

Since f_m and f_n share the same input link of h, they are serialized and, hence, the $RelativeOffset$ between f_m and

f_n cannot be less than the transmission time of f_m, denoted by tr_m. Then:

$$O^h_{r,m,n} = \max\{O^h_{r,m,n}, tr_m\}$$

[2] implements offset computation on a per end system basis. Indeed, with FIFO, all the flows generated by a given end system share the same bandwidth. Considering DRR, each class is considered separately and it gets a dedicated bandwidth. Therefore, for each end system, the effect of offsets is applied on a class by class basis.

It has to be noted that offsets cannot be defined between flows from different source end systems, since there is no common clock between end systems.

Let's consider the network configuration in Figure 1. For the rest of the paper, we assume the definite offsets for flows at their respective end system as given in Table III.

TABLE III: Definite Offset computation results for source end systems in Figure 1

v_k	$O^{e_i}_{d,k}(\mu s)$	v_k	$O^{e_i}_{d,k}(\mu s)$	v_k	$O^{e_i}_{d,k}(\mu s)$	v_k	$O^{e_i}_{d,k}(\mu s)$
v_1	1500	v_5	1000	v_9	6000	v_{13}	0
v_2	500	v_6	3000	v_{10}	14000		
v_3	1000	v_7	0	v_{11}	2000		
v_4	0	v_8	7000	v_{12}	0		

TABLE IV: Offset computation results for port S^1_1 in Figure 1

v_k	$O^{S_1}_{r,k,m}(\mu s)$			
	v_1	v_2	v_3	v_4
v_1	-	2300.48	780.48	40
v_2	220.48	-	60	760.48
v_3	40	2740.48	-	260.48
v_4	660.48	80	160.48	-
	v_5	v_6	v_7	v_8
v_5	-	1240.48	220.48	1260.48
v_6	1240.48	-	220.48	3260.48
v_7	240.48	240.48	-	260.48
v_8	1180.48	3180.48	160.48	-

Figure 3 illustrates the temporal separation of class C_2 flows v_6 and v_7 frames due to relative offset at e_1, S^1_1 and S^1_3.

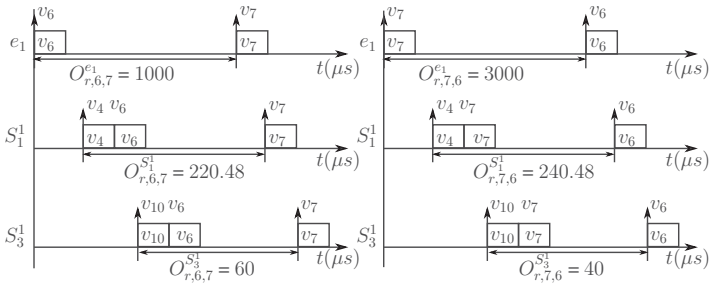


Fig. 3: Relative offset of v_6 and v_7

In order to compute Relative offsets at switch output ports, we need to compute flow delays till these ports. This computation takes into account offsets in previous ports. It is detailed in the following section.

B. Delay computation

In [2], an aggregation technique is used to integrate offset in NC. In DRR scheduler, flows of each class C_x, from same source end system, can be aggregated as a single flow. This is valid because the flows of a class C_x transmitted from same source end system are affected by temporal separation and share the same bandwidth. The aggregation technique takes into account the relative offset between the class C_x flows. Now, we show delay computation through an example given in Figure 1.

Let us calculate the node delay at output port S^1_1 for class C_2 flow v_6. At an output port h the overall arrival curve $\alpha^h_{C_x}$ of a class C_x flows can be computed as :

a) Step 1: Make i subsets SS_i of class C_x flows, based on the flows sharing same source end system.

Since the arrival traffic at S^1_1 from class C_2 is due to flows v_6, v_7 and v_4 and since v_6 and $v_7$share the same source node e_2, they belong to a subset $SS_1 = \{v_6, v_7\}$. Whereas, v_4 from source node e_1 belongs to another subset $SS_2 = \{v_4\}$.

b) Step 2: Aggregate the flows of each subset SS_i as one flow and characterize its arrival curve $\alpha^h_{SS_i}$.

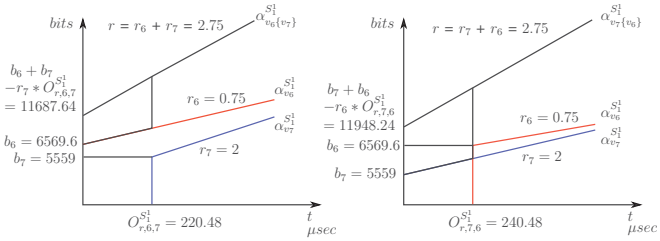
Based on the configuration given in Table I, definite offset given in Table III and the relative offset computed using equation (10) we have $O^{S^1_1}_{r,6,7} = 220.48\mu sec$, $O^{S^1_1}_{r,7,6} = 240.48\mu sec$. The arrival curve of subset SS_1 is

$$\alpha^{S^1_1}_{SS_1} = \max\{\alpha^{S^1_1}_{v_6\{v_7\}}, \alpha^{S^1_1}_{v_7\{v_6\}}\}$$

where, $\alpha^h_{m\{n\}}$ is the arrival curve obtained when flow v_m arrives before flow v_n at output port h, with temporal separation of $O^h_{r,m,n}$. This curve is shown in Figure 4 and Figure 5.

In subset SS_2, since there is only one flow v_4, the overall arrival curve is the same as the arrival curve of flow v_4 at port S^1_1, thus

$$\alpha^{S^1_1}_{SS_2} = \alpha^{S^1_1}_{v_4}$$



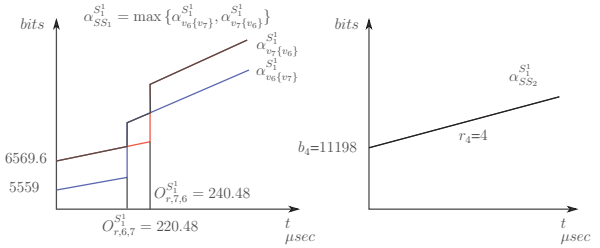
(a) arrival curve $\alpha^{S^1_1}_{v_6\{v_7\}}$ (b) arrival curve $\alpha^{S^1_1}_{v_7\{v_6\}}$

Fig. 4: Possible arrival curves for subset SS_1

c) Step 3: The overall arrival curve is the sum of the arrival curve of each subset, i.e. $\alpha^h_{C_x} = \sum_{j=0}^{i} \alpha^h_{SS_j}$.

Thus, we have $\alpha^{S^1_1}_{C_2} = \alpha^{S^1_1}_{SS_1} + \alpha^{S^1_1}_{SS_2}$ as shown in Figure 6.

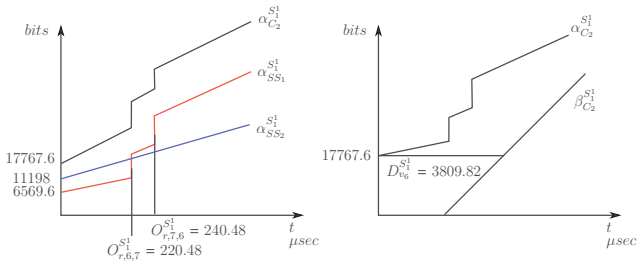
The service curve $\beta^{S^1_1}_{C_2}$ for class C_2 flows at output port S^1_1 can be computed using equation (7), which is also shown



(a) aggregated curve $\alpha^{S_1^1}_{SS_1}$ (b) aggregated curve $\alpha^{S_1^1}_{SS_2}$

Fig. 5: Aggregated arrival curves at S_1^1

in Figure 6. Now with this overall arrival curve $\alpha^{S_1^1}_{C_2}$ and the service curve $\beta^{S_1^1}_{C_2}$, we can compute the delay $D^{S_1^1}_{v6}$ using equation (8), which gives $D^{S_1^1}_{v6} = 3809.82\mu sec$.



(a) Overall arrival curve $\alpha^{S_1^1}_{C_2}$ (b) Delay at S_1^1

Fig. 6: Overall curve & Node delay

Figure 7 shows comparison of end to end delay computation results, for network given in Figure 1, using classical NC approach (black) and NC approach with offset (purple). For the given configuration, the integration of offset results in improvement of delay computation by 12.5%.

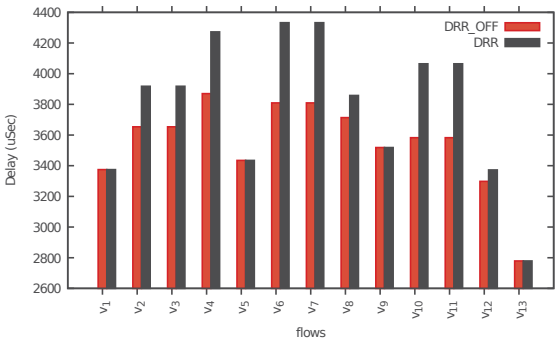


Fig. 7: E2E delay : Network in Figure 1

IV. Evaluation of an Industrial Configuration

Now we show the evaluation of proposed approach on an industrial-size configuration. It includes 96 end systems, 8 switches, 984 flows, and 6276 paths (due to VL multicast characteristics). The flows are divided into three classes namely critical flows, multimedia flows and best effort flows. Table V shows the DRR scheduler configuration at each output port. Definite offsets are generated, using the algorithm in [2].

TABLE V: DRR Scheduler Configuration for Industrial Network

Class	Number of Flows	Frame Length (byte)	Q_x (byte)	BAG (msec)	Category
C_1	128	84 - 147	3070	4 - 128	Critical
C_2	590	84 - 475	1535	2 - 128	Multimedia
C_3	266	84 - 1535	1535	2 - 128	Best-effort

Figure 8 shows a comparison between classical NC approach and NC approach with integrated offset, the average improvement of the E2E delay bound computed in the given industrial configuration is 26.9% and a maximum gain of 70.05%. This is a significant improvement. However, as shown in [2], a much higher average gain was obtained with FIFO scheduling on a similar configuration. This result is not surprising. Indeed, with DRR, only flows from the same class are offset dependent. It leads to smaller sets of flows.

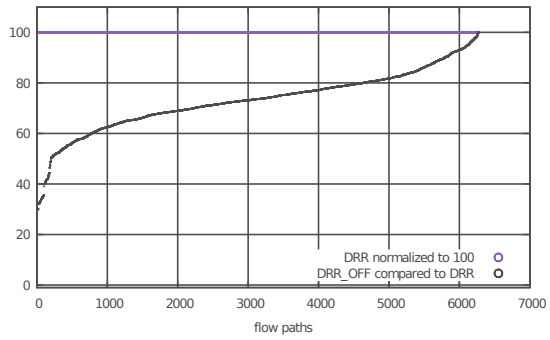


Fig. 8: E2E delay : NC DRR vs. NC DRR + Offset

It has been shown in [1] that, bursts in each output port can be limited, since flows arriving from the same input link are serialized and, consequently, they cannot arrive at the same time. This serialization effect can be directly integrated in arrival curves, in the same manner as in [1]. As shown in Figure 9, it leads to a further average reduction of 2.43%, with a maximum reduction of 14.75%. The reduction is small because, thanks to offsets, there are only few bursts.

In the figure 8 and 9, the paths are sorted by decreasing order of comparative gain in E2E Delay computation. For example, in Figure 8, there are at least 4000 flow paths for which the gain is more than 20%.

V. Conclusion

In this paper, we combine two existing contributions in the context of real-time switched Ethernet networks:

- worst-case traversal time analysis for deficit round robin service discipline, based on network calculus,

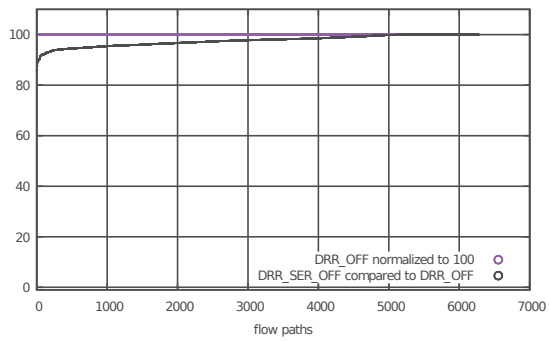


Fig. 9: E2E delay : NC DRR + Offset vs. NC DRR + Offset + Grouping

- integration of offsets in worst-case traversal time analysis in the context of FIFO.

First, we show how offsets can be integrated in WCTT analysis for DRR. Second we evaluate the benefit of this integration. On a realistic case study, the average reduction of worst-case end-to-end latencies is 26.9%. This result shows the significant impact of the scheduling of flows at their source nodes on worst-case latencies.

As future work, we plan to optimize WCTT analysis for DRR. Indeed, the existing approach builds service curves without considering effective traffic. Thus it considers that all the classes are always active, which might not be the case.

We also plan to extend our work to other service disciplines such as Weighted Round Robin, which leads to simpler implementations in switches.

REFERENCES

[1] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Improving the worst-case delay analysis of an afdx network using an optimized trajectory approach," *IEEE Trans. Industrial Informatics*, vol. 6, no. 4, Nov 2010.

[2] X. Li, J.-L. Scharbarg, and C. Fraboul, "Improving end-to-end delay upper bounds on an afdx network by integrating offsets in worst-case analysis," pp. 1–8, Sept 2010.

[3] Y. Chen, R. Kurachi, H. Takada, and G. Zeng, "Schedulability comparison for can message with offset: Priority queue versus fifo queue," in *RTNS*, 2011.

[4] P. M. Yomsi, D. Bertrand, N. Navet, and R. I. Davis, "Controller area network (can): Response time analysis with offsets," in *2012 9th IEEE International Workshop on Factory Communication Systems*, May 2012, pp. 43–52.

[5] S. Mubeen, J. Mki-Turja, and M. Sjdin, "Response time analysis with offsets for mixed messages in can supporting transmission abort requests," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Sept 2014, pp. 1–10.

[6] L. Du and G. Xu, "Worst case response time analysis for can messages with offsets," in *2009 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, Nov 2009, pp. 41–45.

[7] M. SHREEDHAR and G. VARGHESE, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on networking, 1996, vol. 4, no 3, p. 375-385*, p. 11, 1996.

[8] S. S. KANHERE and H. SETHU, "On the latency bound of deficit round robin," *Computer Communications and Networks, 2002. Proceedings. Eleventh International Conference on. IEEE, 2002. p. 548-553.*, p. 7, October 2002.

[9] M. BOYER, G. STEA, and W. M. SOFACK, "Deficit round robin with network calculus," *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th International Conference on (pp. 138-147). IEEE*, p. 10, October 2012.

[10] M. BOYER, N. NAVET, M. FUMEY, J. MIGGE, and L. HAVET, "Combining static priority and weighted round-robin like packet scheduling in afdx for incremental certification and mixed-criticality support," *5TH EUROPEAN CONFERENCE FOR AERONAUTICS AND SPACE SCIENCES (EUCASS)*, 2013.

[11] L. Lenzini, E. Mingozzi, and G. Stea, "Aliquem: a novel drr implementation to achieve better latency and fairness at o(1) complexity," *IEEE 2002 Tenth IEEE International Workshop on Quality of Service*, 2002.

[12] J.-Y. L. Boudec and P. Thiran, *Network Calculus: a theory of deterministic queuing systems for the internet*. LNCS, April 2012, vol. 2050.