*electronics*

MDPI

*Article*

# Regulating Scheduler (RSC): A Novel Solution for IEEE 802.1 Time Sensitive Network (TSN)

**Jinoo Joung**

Department of Human Intelligence Information Engineering, Sangmyung University, Seoul 03016, Korea; jjoung@smu.ac.kr; Tel.: +82-2-2287-5452

Received: 18 January 2019; Accepted: 2 February 2019; Published: 6 February 2019


**Abstract:** Emerging applications such as industrial automation, in-vehicle, professional audio-video, and wide area electrical utility networks require strict bounds on the end-to-end network delay. Solutions so far to such a requirement are either impractical or ineffective. Flow based schedulers suggested in a traditional integrated services (IntServ) framework are O($N$) or O(log $N$), where $N$ is the number of flows in the scheduler, which can grow to tens of thousands in a core router. Due to such complexity, class-based schedulers are adopted in real deployments. The class-based systems, however, cannot provide bounded delays in networks with cycle, since the maximum burst grows infinitely along the cycled path. Attaching a regulator in front of a scheduler to limit the maximum burst is considered as a viable solution. International standards, such as IEEE 802.1 time sensitive network (TSN) and IETF deterministic network (DetNet) are adopting this approach as a standard. The regulator in TSN and DetNet, however, requires flow state information, therefore contradicts to the simple class-based schedulers. This paper suggests non-work conserving fair schedulers, called 'regulating schedulers' (RSC), which function as a regulator and a scheduler at the same time. A deficit round-robin (DRR) based RSC, called nw-DRR, is devised and proved to be both a fair scheduler and a regulator. Despite the lower complexity, the input port-based nw-DRR is shown to perform better than the current TSN approach, and to bind the end-to-end delay within a few milliseconds in realistic network scenarios.

**Keywords:** time sensitive network (TSN); delay bound; scheduler; regulator; asynchronous approach

## 1. Introduction

Various emerging applications such as industrial automation, building automation, inter-vehicle network, in-vehicle network, professional audio video (AV) network, and wide-area electrical utility network require strict delay bounds, which range from a few milliseconds to a few seconds. IEEE 802.1 time sensitive network (TSN) and IETF deterministic network (DetNet) are representative standard groups for dealing such delay requirements on various networks. TSN aims guarantee end-to-end delay and zero packet loss on an Ethernet data plane [1]. IETF's DetNet has a similar scope with TSN, but is biased on network layer issues [2]. DetNet can be seen as an extension of TSN into routed networks, with IP and MPLS data plane. DetNet claims its scope is limited to a single administrative domain such as campus-area network or private WAN, but does not restrict itself from expanding into wider area networks. DetNet defines the electrical utility networks using time division multiplexing currently to be one of the key use cases [2]. For example, the utility network in Quebec, Canada, which includes 60 generating stations, 513 substations, and 10,500 km of optical fiber, carries instantaneous electrical information (current, voltage, active power, etc.) and real-time commands such as trip, open/close relay, etc. This network has a mixed Layer 2 and Layer 3 topology, and requires the deterministic behavior from the network such as bounded delay/jitter and precise timing. By applying DetNet technology the utility networks can increase the electrical grid's reliability and efficiency.

TSN and DetNet both cover a wide area of technological aspects, including packet forwarding, time synchronization, path selection, resource reservation, reliable transmission, etc. This paper focuses on the packet forwarding technology. The packet forwarding function includes switching, queueing, and scheduling of packets. The TSN task group (TG) was initiated as the residential Ethernet TG in the IEEE 802.1 working group (WG) around the year 2005 and expanded to the audio video bridge (AVB) TG. Both TGs considered the professional AV traffic transport among nodes in close proximity as the main target. In its early stage, the TGs set a clear performance goal to bind the delay within "2 ms in seven hops". This clear goal helped the TGs got the attention from the beginning. The AVB TG in 2012 evolved into the TSN TG with the extended scope for various industrial applications. The residential Ethernet and the AVB TG's target, the AV applications have a small number of large bandwidth flows. A flow is defined to be a set of packets of the same source-destination processes pair, generated within a relatively limited amount of time. Meanwhile in TSN, the target applications also generate numerous flows with relatively small bandwidth with nodes in a complex, larger area network, such as industrial sensor-actuator networks. For example, IEC/IEEE 60802 TSN Profile for Industrial Automation, a joint project of IEC SC65C/MT9 and IEEE 802 to define TSN profiles for industrial automation, has defined the traffic types of industrial automation and control systems, at the November 2018 meeting. According to this definition, isochronous traffic with 100 byte maximum packet length requires less than 2 ms end-to-end delay, while cyclic traffic with 1000 byte maximum packet length requires 2 ms to 20 ms end-to-end delay. These end-to-end delay requirements will be the primary target of this paper.

There has been a tremendous amount of research activities regarding the end-to-end delay bound in large scale networks, since early 1990s [3–11]. While various approaches for such requirements have their pros and cons, meeting delay requirement and implementation simplicity have been incompatible with one another so far. We propose a novel solution that can perform comparatively with the state-of-the-art solutions that are considered as the standard for the TSN and the DetNet, but with a far less complexity.

Our key contribution is schedulers that also perform the regulating function at the same time. This set of schedulers, called the 'regulating scheduler' (RSC), are devised by giving a non-work conserving characteristics to fair schedulers. Then a deficit round robin (DRR) based RSC, called nw-DRR, is devised and proved to be both a fair scheduler and a regulator. Despite the lower complexity, the input port-based nw-DRR is shown to perform better than the current TSN approach, and to bind the end-to-end delay within a few milliseconds in realistic network scenarios. In seven-hop networks with a few crossing flows, the maximum end-to-end delay can be bound within 2 ms, if the packet length is limited to be less than 400 bit.

This paper is comprised as follows. The related works to the problem of delay bound are reviewed in Section 2. In Section 3 the RSC and its realization with DRR, nw-DRR, are introduced and formally analyzed. The performance of the nw-DRR is investigated through numerical analysis with an exemplary network topologies in the Section 4. The last section discusses the result, the remaining issues, and possible follow up research activities.

## 2. Related Works

### *2.1. TSN Synchronous Approach*

As the target applications of the TSN standard have diversified, there emerged two different forwarding approaches in the standard. The first one is the synchronous approach, which includes the cyclic queuing and forwarding (CQF) [12] function. The CQF is rooted from the residential Ethernet TG era's small number of large bandwidth flows. In the synchronous approach, time is divided into slots, whose start and end times are synchronized across all the nodes in the network. In order to guarantee a bounded delay, the allocation of slots at all the nodes are precisely planned at packet level. A predefined slot has to be allocated to a predefined packet, at every node along the path of the packet. This allocation process, which is not defined in the standard, is quite burdensome in networks even

with moderate number of flows. This slot allocation problem is similar to the job-shop scheduling problem [13], which is NP complete. It is shown that with 200 flows the slot allocation takes around 200 s with a contemporary personal computer, even with a heuristic algorithm [13].

The second one is the asynchronous approach, in which the asynchronous traffic shaping (ATS) [14] function takes the central role. The ATS, also called the interleaved regulator, is basically a regulation module in front of a scheduler. In this approach there is no need for the synchronization across the network. The ATS can be seen as a revision of an existing solution stemmed from the IntServ framework, with a modification for a simpler implementation. In the following we will see how the ATS is derived.

### *2.2. Integrated Services (IntServ) and Latency-Rate (LR) Servers*

Binding end-to-end network delay is already achievable with the traditional IntServ framework [3]. Let us look at the theoretical background of IntServ that has been actively researched since the 1990s. The core benefit of the IntServ is that the maximum end-to-end delay of a packet is guaranteed if the following three conditions are met:

1. The arrival process of every incoming flow conforms to the flow's arrival curve.
2. The sum of the arrival rates of flows on every link in the network is less than or equal to the link's bandwidth.
3. Every link in the network provides service to each flow with a service process that conforms to a service curve.

More specifically, Condition 1 means that a flow arrival process meets the following inequality.

$$A_i(t_0, t) \leq \rho_i(t - t_0) + \sigma_i, \tag{1}$$

where $A_i(t_0, t)$ is the amount of the arrived traffic in an arbitrary period $(t_0, t]$ from the flow $i$, $\rho_i$ is the average arrival rate, and $\sigma_i$ is the maximum burst size of the flow $i$. A burst can be thought as a large number of packets go together with almost no time interval between them. The curve $A_i(t) = \rho_i(t - t_0) + \sigma_i$ is called the arrival curve.

Now let us introduce a set of schedulers that conform to a flow's service curve therefore meet Condition 3. This set of schedulers is called the latency-rate (LR) servers [10]. We will elaborate on the properties of the LR servers. The mathematical notations frequently appear in this paper are summarized in Table 1.

**Table 1.** Mathematical notations used in the paper.

| Notation | Meaning |
|---|---|
| $L_i$ | Maximum packet length of flow $i$ |
| $r$ | Link capacity |
| $\rho_i$ | Arrival rate of flow $i$ |
| $\sigma_i$ | Max burst size of flow $i$ |
| $\varphi_i$ | Quantum value assigned for flow $i$ |
| $\Theta_i^S$ | Latency of flow $i$atsever$S$ |
| $D_i$ | Delay experienced by packets of flow $i$ |

In the following the definitions and the properties are given according to [10].

**Definition 1.** *Flow i busy period is the maximum time period* $(t_1, t_2]$ *that for all* $t \subset (t_1, t_2]$ *the following inequality holds.* $A_i(t_1, t) \geq \rho_i(t - t_1)$*, where* $A_i(t_1, t)$ *is the amount of traffic arrived during* $(t_1, t]$ *from flow i.*

Note that a flow busy period is not the same with a flow backlogged period, during which a flow always has packet(s) to be served in the scheduler. In the following a start time and a finish time of a

period are the arrival time of the first packet's last bit and the departure time of the last packet's last bit, respectively.

**Definition 2.** *Let $t_0$ be the start time of a flow i busy period in server S and $t^*$ the finish time of the busy period. Then, server S is an LR server if and only if a nonnegative constant ${C_i}^S$ can be found such that, at every instant in the period $(t_0, t^*]$,*

$$W_i{}^S(t_0, t) \geq max\lceil 0, \rho_i(t - t_0 - {C_i}^S)\rceil, \tag{2}$$

*where $W_i(t_0, t)$ is the amount of service given during $(t_0, t]$ to flow i.*

*The minimum ${C_i}^S$ that satisfies (1) is the latency of the flow i at the server S and denoted by ${\Theta_j}^S$. The curve $W_i(t) = max\lceil 0, \rho_i(t - t_0 - {C_i}^S)\rceil$ derived from the inequality (2) is called the service curve. If a service given to a flow satisfies (2) then it conforms to the service curve. Condition 3 above means that a scheduler at an output port has to satisfy the inequality (2). Therefore, LR servers are the key functional elements of the IntServ framework. For an LR server, only two parameters, service curve $\rho_i$ and latency ${\Theta_j}^S$ define the service curve. LR servers have the following properties [10].*

**Property 1.** *If a flow traverses two adjacent LR servers, it is equivalent for the flow to traverse a single LR server with the latency equal to the sum of the latencies of the two adjacent LR servers.*

Property 1 means the series of LR servers can be seen as a single LR server with a latency equal to the sum of the latencies of the LR servers in series.

**Property 2.** *If a flow i traverses only LR servers $S_j$ in its path, then the end-to-end delay experienced by the packets in the flow is bounded by the following inequality.*

$$D_i \leq \frac{\sigma_i - L_i}{\rho_i} + \sum_{j=1}^{k} {\Theta_j}^{S_j} \tag{3}$$

Property 2 means the maximum end-to-end delay is the sum of latencies of the LR servers plus the single delay term caused by the initial max burst. As in Property 3, the maximum burst of a flow increases as it passes the LR servers, yet it does not affect the delay. This property is called "pay burst only once".

**Property 3.** *If a flow i conforms to the arrival curve with parameters $\{\rho_i, \sigma_i\}$, i.e. $A_i(t_0, t) \leq \rho_i(t - t_0) + \sigma_i$, then after traversing an LR server S it follows the curve $(\rho_i, \sigma_i + {\Theta_i}^S\rho_i)$, i.e. the maximum burst increases as much as $\rho_i{\Theta_i}^S$. In other word, if $t_0$ is the start time of a busy period, then $W_i(t_0, t) \leq \rho_i(t - t_0) + \sigma_i + \rho_i{\Theta_i}^S$.*

Note that a service process for a flow at a scheduler is identical to the arrival process to the next scheduler. Now we have defined the LR servers and its important properties. Fair schedulers that can guarantee sustainable service rate to every flow fall into this category. The ideal generalized processor sharing (GPS) [4], packetized-GPS [4], self-clocked fair queuing [5], and virtual clock [7], which are all based on the "virtual finish time", are LR servers. Round robin based deficit round robin (DRR) [8] and weighted round robin (WRR) are also LR servers. Virtual finish time-based LR servers generally have smaller latencies but are more complex to implement. DRR and WRR are easier to implement and therefore widely deployed. These LR servers' latencies are proportional to the maximum packet size and the maximum burst size of a flow.

The application service procedure in the IntServ is composed of the connection establishment phase and the data transfer phase. In connection establishment phase, a flow specifies its input parameters $\{\rho_i, \sigma_i\}$, then the network checks if the traversing nodes have enough bandwidth, reserve the bandwidth for the flow, then admit the flow and guarantee the service. The admitted flows are placed in its own queues and serviced fairly with schedulers at the output ports of relaying nodes in

the data transfer phase. The LR servers in an IntServ network work as a scheduler and provide a fair share of the resource. The series of LR servers form a single virtual LR server to a flow, according to Property 1. The maximum delay of the flow then limited by (3).

The flow-based scheduling function is a key component of the IntServ, yet its complexity prohibits practical implementation. Flow based schedulers suggested in the IntServ framework are O($N$) or O(log $N$), when $N$ is the number of flows in the scheduler. $N$ can be tens of thousands in core networks. Due to such complexity, flow-based schedulers have not been deployed.

### *2.3. Alternatives to the IntServ*

As a consequence, a simpler solution with the class based scheduling emerged. A class is a coarser set of packets, which is a set of packets of similar performance requirements. This solution is adopted in the differentiated services (DiffServ) [15] framework. The DiffServ categorizes the whole traffic into eight classes and schedules the queues according to the classes. In the DiffServ, however, the connection establishment and the admission to a network are still flow basis. A flow specifies its input parameters $\{\rho_i, \sigma_i\}$, then the network checks if the traversing nodes have enough bandwidth, reserve the bandwidth for the flow, then admit the flow and guarantee the service. After connection establishment, in data transfer phase, at the boundary of a network a flow is regulated. Regulation is a process to enforce the arrival process of a flow to conform to its arrival curve. Packets of a flow that does not conform to the negotiated arrival curve may be delayed or dropped if necessary. A properly regulated flow will conform to the arrival curve in (1) after the regulation. The leaky bucket [16] and the token bucket [17,18] are the commonly implemented regulators.

After the regulation process at the network boundary, a flow is put into the network relaying nodes. At the output of the relaying nodes a flow is put into a proper class. The packets in a class is placed into a single queue. Then the queues for different classes are scheduled accordingly. The scheduler used in the DiffServ network is the strict priority (SP) scheduler. In an SP scheduler, among the packets queued, the packet with the highest priority is selected to be served. It is shown that the strict priority scheduler is also an LR server [11] for a flow, but its latency is a function of the sum of all the other flows' maximum burst sizes as

$$\Theta_i{}^{SP} = \frac{\sigma^{SP} - \sigma_i^{SP} + 2L}{r^{SP}}, \tag{4}$$

where $\sigma^{SP}$ is the sum of maximum bursts of all the flows in the SP scheduler.

DiffServ is suitable for the applications with the performance metrics such as average delay and packet loss rate, but not for ones with the strict delay bounds. This is because the class-based SP scheduler in the DiffServ cannot provide guaranteed delay for networks with cycle. The burst increases infinitely along the cycle. Consider the following network topology depicted in Figure 1. Nodes 1-4 forms a cycle. The flows traversing node 1-4 are higher priority flows. They affect to each other as follows. Flow 1 shares the queue at the node 1 with flow 4. Flow 1's latency at the node 1, $\Theta_1{}^{SP1}$, is a function of the max burst of flow 4 arriving to the node 1, $\sigma_4^{SP1}$. Since the max burst of the flow 1 arriving to the node 2, $\sigma_1^{SP2}$, is a function of the latency $\Theta_1{}^{SP1}$, it is also a function of $\sigma_4^{SP1}$. In other words,

$$\sigma_1^{SP2} = f(\sigma_4^{SP1}). \tag{5}$$

Now, flow 1 affects flow 2 the same way as flow 4 does to flow 1, i.e., $\sigma_2^{SP3} = f(\sigma_1^{SP2}) = f(\sigma_4^{SP1})$. It eventually evolves to the point where $\sigma_4^{SP1} = f(\sigma_4^{SP1})$. The only value that satisfies this equation for $\sigma_4^{SP1}$ is infinity.

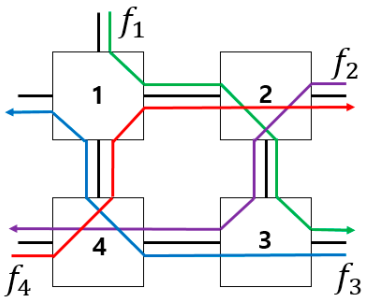


**Figure 1.** Network with a cycle.

One possible solution for the infinitely increasing burst is the regulation per flow at every node. Therefore, a regulator placed right before the class-based scheduler has been considered [6,9].

Figure 2 depicts such a system architecture [6,9]. In Figure 2, a connection is synonymous to a flow. The max burst size of a flow at the output of the system in Figure 2 is proportional to the sum of all the flows in a queue. The max burst size, however, is reduced drastically by the regulator at the next node. The packets in the flow under regulation may suffer from additional delays at the regulator, but it is shown that the maximum delay does not increase with the addition of the regulator [9,19]. It can be interpreted that the packets with large enough delays are not delayed further at the regulator.

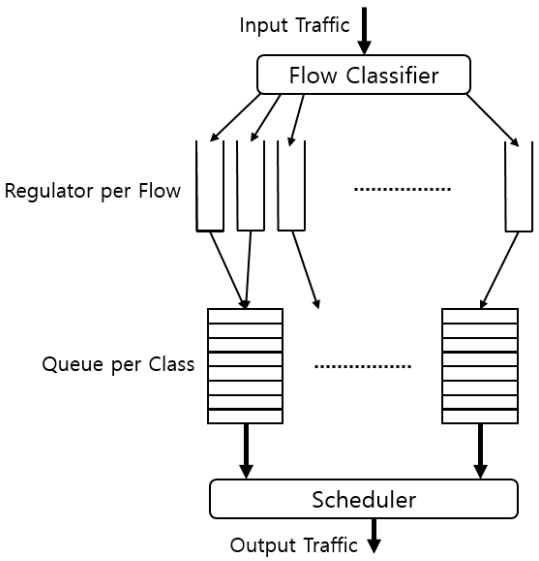


**Figure 2.** Architecture of the flow-based regulator plus class-based scheduler system proposed in [6].

In the system of Figure 2, the central scheduler is relieved from per-flow scheduling task. The complexity goes to the parallel distributed regulators, however. Since the distributed regulators have most of complexity, compare to the flow-based schedulers, the regulator-scheduler system is more plausible for implementation. This architecture has not been widely accepted though, due to the massive number of regulators.

### *2.4. TSN Asynchronous Approach*

TSN's asynchronous approach has also suggested a regulator-scheduler pair for each output port. The regulation function in the TSN is performed by the aforementioned ATS module, which was originally called the interleaved regulator [14]. ATS places queues per-input port for higher priority class. After the regulation with the ATS, the TSN schedules with strict priority scheduling with queues assigned to each classes. ATS, however, still needs to maintain flow states, find the flow to which the

packet at the head of the queue belongs, decides whether the packet is to be regulated based on the flow state. Therefore, it does not completely remove the per-flow complexity [20,21].

Figure 3 depicts the architecture of the TSN asynchronous approach. ATS maintains the queue per input port, for class A and B. A and B indicates the high priority traffic Class A and Class B, while BE0 stands for Best Effort Class 0 traffic. The CDT (control data traffic) represents the highest priority traffic for control and management. There are usually eight classes excluding CDT. Class A and B are further regulated by the credit based shaper (CBS). The CBS has been adopted in the standard at the early stage of standardization activity, before the asynchronous approach was introduced. The reason for the existence of CBS is now unclear. It was meant to prevent the exhaustion of link bandwidth by the high priority traffic, but a part of its function is now duplicated with ATS.

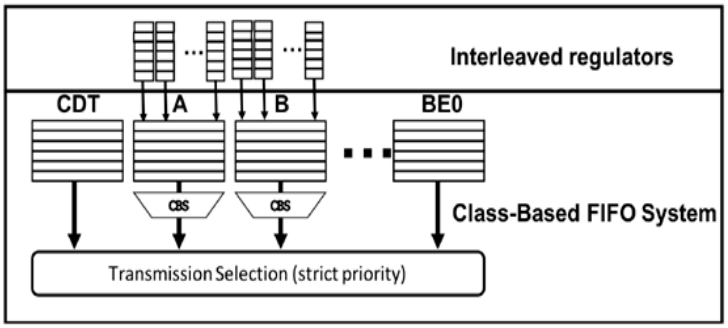


**Figure 3.** Architecture of the output port module in the TSN asynchronous approach. ATS (interleaved regulator) is placed prior to the class-based first in first out (FIFO) scheduler [14,19].

## 3. Proposed Scheduler and Its Analysis

### *3.1. Proposed Approach*

This paper proposes a simpler alternative for end-to-end delay guarantee, the regulating scheduler (RSC). The RSC can be thought as a non-work conserving version of fair schedulers. In order to build an RSC, a fair scheduler is modified to make their queues always filled with a virtual packet if there is no actual packet waits for the service. The virtual packet receives the service but it just disappear after the service rather than be transferred to the next node. When an actual packet arrives, the virtual packet in that queue, if existing, is removed even if it was in the middle of a service. By this modification all the queues look backlogged and the queue under observation will be served at a steady pace. Therefore, the traffic served out from the queue under observation is strictly regulated. Intuitively, a burst increases at a node as follows. At a certain moment, a single queue is full of traffic while all the other queues are empty. The incoming burst to the queue arrives right before the queued burst's service time. They merged together and served instantly, and becomes a larger burst. The RSC prevents the instant service of a large amount of traffic from a single queue by having the other backlogged queues.

We carefully devised a pure per input port RSC for the TSN environment. Its operation can be described as the following. The whole traffic is divided into high priority and low priority traffic. The high priority traffic includes CDT, Class A, and Class B. The low priority traffic includes the rest. At an output port of a relaying node such as switch and router the high priority traffic is divided based on their input ports and allocated to different queues. The low priority traffic is allocated to another queue at the output port. Therefore, if there are N input ports, (N+1) queues will be established. These queues are scheduled with a DRR scheduler [8] but based on the RSC principle. The algorithm detail will follow in Section 3.3. This scheduler is the main contribution of this paper, which is called nw-DRR (the non-work conserving DRR).

Figure 4 depicts the proposed input-port based nw-DRR's architecture. It replaces the whole system depicted in Figure 3. One can easily see the simplicity of the proposed system's architecture. We analyze and prove that the nw-DRR is a scheduler and a regulator at the same time in Section 3.3.

We also prove that nw-DRR is a fair scheduler with the same {latency, service rate} parameters with the original DRR before modification. It is also shown that the input-port based nw-DRR can guarantee end-to-end delay within the required bounds of a few milliseconds.

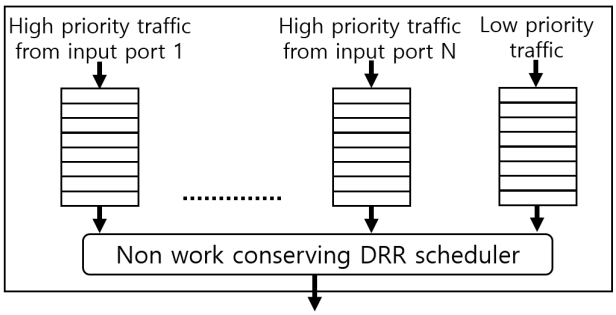


**Figure 4.** Architecture of the proposed scheduler at an output port. The scheduler works as a regulator as well.

### *3.2. DRR Scheduler*

DRR's complexity depends heavily on its parameter, deficit value. Larger the deficit easier to implement, but with larger latency. DRR works as the follows [8].

1. Each queue is assigned a deficit value ($\delta$) and a quantum value ($\varphi$).
2. Deficit value is variable but initialized to zero. Quantum value is fixed and determined to be proportional to the arrival rate of the traffic to the queue. For example, if arrival rates to two queues are 1 Mbps and 2 Mbps then quantum value of the latter queue should be twice to the one of the former. The quantum values of two queues can be set to be 10 byte and 20 byte, or 50 byte and 100 byte, respectively as well. It is shown that a smaller quantum gives a smaller latency [10]. A smaller quantum makes the operation of the scheduler slower, however.
3. Each backlogged queue takes a turn, and a complete cycle of turns of backlogged queues is called a frame ($F$).
4. At the start of each turn a queue increases the deficit value as much as the quantum value.
5. If the deficit value is larger or equal to the packet length at the head of the queue, then serve the packet. After service of the packet, the deficit is reduced as much as the packet length. Repeat the Step 5 as long as the condition holds.
6. If the queue becomes empty then reset deficit to zero.
7. Proceed to the next backlogged queue, and go to the Step 4.

The latency of a DRR server, in case the deficit value is smaller than the max packet length, is given as the follows [22].

$$\Theta_i^{DRR} = \frac{1}{r}\left[(F-\varphi_i)\left(1+\frac{L_i}{\varphi_i}\right)+\sum_{n=1}^{N} L_n\right] \tag{6}$$

where $F$ is the sum of all $\varphi_i$ of active flows in the server, and N is the number of active flows. The significant term in (6) is $F$ and $\sum_{n=1}^{N} L_n$. $F$ can be reduced by reducing deficit values locally, but sum of max packet lengths of each queue is not easily controlled.

### *3.3. DRR Scheduler as a Regulator*

From now on, we will show that a DRR scheduler can work as a regulator, with a proper modification in line with the RSC principle. The following theorems show that a general DRR scheduler bounds the work given to a flow within a certain limit.

**Theorem 1.** *Assume that flow i is continuously backlogged during time period (a,b]. Assume that the flow is served k turns during (a,b]. The work given to the flow during this period is bounded by the following inequality.*

$$\mathrm{k}\varphi_i - \delta_i^k \leq W_i(a,b) \leq \mathrm{k}\varphi_i + \delta_i^0, \tag{7}$$

*where $\delta_i^k$ is the deficit value at the finish time of $k_{th}$ round during (a,b].*

**Proof.** See the proof of Theorem 4.2 in [8]. □

Theorem 2 generalizes the result of Theorem 1 without using k and the deficit value terms.

**Theorem 2.** *During any time period (a, b] within a flow i backlog period, the work given to the flow is bounded by the following inequality.*

$$W_i(a,b) \leq \rho_i \frac{F_{max}}{f}(b-a) + \varphi_i + L_i, \tag{8}$$

*where $f$ is the sum of quantum values of the flows that have been always backlogged during (a,b], and $F_{max}$ is the sum of quantum values of all the flows in the scheduler, in an imaginary situation where the sum of arrival rates of all the flows equals to the link capacity.*

**Proof.** Let $t_k$ be the finish time of $\mathrm{k_{th}}$ round, counting from $a$. Let $t_0 = a$. Duration of a round $t_k - t_{k-1} = \frac{1}{r}\sum_{j\in B_k}(\varphi_j + \delta_j^{k-1} - \delta_j^k)$, where $B_k$ is the set of flows that are continuously backlogged during $(t_{k-1}, t_k]$. During $(t_{k-1}, t_k]$ the elements of $B_k$ are not changed. Let $B$ be the set of flows that are continuously backlogged during $(t_0, t_k]$. For any $k$, $B \subset B_k$. The following equation holds. $t_k - t_{k-1} = \frac{1}{r}\sum_{j\in B}(\varphi_j + \delta_j^{k-1} - \delta_j^k)$. By summing up the equation for all $k$, we get $t_k - t_0 \geq k\frac{f}{r} + \frac{1}{r}\sum_{j\in B}(\delta_j^0 - \delta_j^k) \geq \frac{f}{r}(k-1)$, since $\sum_{j\in B}\delta_j^0 \geq 0$ and $\sum_{j\in B}\delta_j^k \leq f$. Therefore $k \leq \frac{r}{f}(t_k - t_0) + 1$. From Theorem 1, during $(t_0, t_k]$.

$$W_i(t_0,t_k) \leq \mathrm{k}\varphi_i + \delta_i^0 \leq \frac{r}{f}(t_k - t_0)\varphi_i + \varphi_i + \delta_i^0 \leq \rho_i \frac{F_{max}}{f}(t_k - t_0) + \varphi_i + L_i, \tag{9}$$

because $L_i \leq \delta_i^0$ and $\rho_i/r = \varphi_i/F_{max}$. In other words, the deficit values cannot be larger than the maximum packet length $L_i$, at all the round finish time instants including $t_0$. Furthermore, the flow arrival rate is proportional to the quantum value of the flow, and $F_{max}$ is so to the link capacity $r$. Therefore, for arbitrary time instant $b$ between $t_k$ and $t_{k+1}$,

$$W_i(a,b) = W_i(a,t_k) \leq \rho_i \frac{F_{max}}{f}(t_k - a) + \varphi_i + L_i \leq \rho_i \frac{F_{max}}{f}(b-a) + \varphi_i + L_i, \tag{10}$$

and the theorem follows. □

From Theorem 2, if a DRR scheduler satisfies two conditions 1) the sum of arrival rates of flows is equal to the link capacity and 2) all the queues in the scheduler are always backlogged, then the work given to the scheduler is bounded as the following inequality.

$$W_i(a,b) \leq \rho_i(b-a) + \varphi_i + L_i, \tag{11}$$

which is the inequality for the amount of work of a regulator with maximum burst size $\varphi_i + L_i$, or an arrival curve of a flow $\{\rho_i, \varphi_i + L_i\}$ where $\varphi_i + L_i$ is the max burst size.

As the next step, we will introduce a modified DRR that satisfies the two conditions given above, and show that it indeed works as a regulator with inequality (11). Consider a DRR scheduler that is modified to work with the following algorithm.

1. Assign a queue for a virtual flow $v$, which has the arrival rate equals to the difference of the link capacity and the sum of the arrival rates of all the flows in the scheduler. ($\rho_v = r - \sum_{i \in H} \rho_i$.) Flow $v$ will serve the low priority traffic, whose arrival rates are not predefined.
2. If a queue, including the queue for $v$, is empty then place a virtual packet in the queue immediately, so that all the queues are never empty. The length of the virtual packet is set to the quantum value of the queue. In other words, the virtual packet is sized such that it can be served in a single turn.
3. The virtual packet is served the same way with the actual packets. After the service virtual packets do not leave the scheduler, though.
4. Consider an actual packet arrives to the queue with a virtual packet. If the virtual packet was not in a service then remove the virtual packet. If it was during a service then immediately stop serving the virtual packet, set the deficit value to 0, and proceed to the next queue.

The above scheduler may serve a virtual packet even when actual packets are waiting in the other queues, therefore is non-work conserving. This is called the nw-DRR scheduler.

### *3.4. Properties of the nw-DRR*

The following theorem proves the nw-DRR gives the same service curve, i.e.,

$$W_i^{nw-DRR}(t_0, t) \geq max\left[0, \rho_i(t - t_0 - \Theta_i{}^{DRR})\right], \tag{12}$$

with the original DRR scheduler that does not generate virtual packets.

**Theorem 3.** *The nw-DRR is an LR server with the same service rate and latency with the DRR server that does not generate any virtual packets.*

**Proof.** Assume $N$ flows arrive to an nw-DRR, thus there are $N$ queues. It is sufficient to prove that for all the packets in an arbitrary flow $i$, $i \in \{1, \ldots, N\}$, there exists a work conserving DRR scheduler that serves later than or equal to the nw-DRR would. □

Let all the other flows, except the flow under observation $i$, have the same arrival rates with those with the nw-DRR but sufficiently large maximum burst sizes to make their queues always be backlogged during the flow $i$ backlog period. Recall that the max burst size does not affect the latency of a flow with a DRR scheduler. Furthermore upon the start of the flow $i$ backlogged period, let the service of the $(i+1)_{th}$ queue start. Let us call this work conserving server DRR*. Intuitively, DRR* gives the worst case DRR service to $i$ in terms of packet service finish time.

Now we will come back to the nw-DRR scheduler. Let us call $p_k$, $k = 0, 1, \ldots, K$, the packets arrive during a flow $i$ backlog period. $p_0$ arrives to a queue with a virtual packet, but nw-DRR removes the virtual packet just before the $p_0$'s arrival. This is true even if that virtual packet was under service, in which case the $(i+1)_{th}$ queue's service starts upon the $p_0$'s arrival. Therefore, $p_0$ does not see that it is non-work conserving. The subsequent packets $p_k$, $k = 1, \ldots, K$, arrive to a backlogged queue and do not see that it is non-work conserving, as well. Therefore, the service times of $p_k$, $k = 0, 1, \ldots, K$, of the nw-DRR are always earlier than or equal to those of the DRR*. The theorem follows.

By Theorem 3, the nw-DRR is an LR server with the same {latency, service rate} parameters with the ordinary work-conserving DRR. Let $i$ be a set of packets assigned to the $i_{th}$ queue in the nw-DRR. We will call $i$ the flow aggregate. The maximum delay experienced by packets in a flow-aggregate $i$ in an nw-DRR's is given as

$$D_i \leq \frac{\sigma_i - L_i}{\rho_i} + \Theta_i^{nw-DRR}. \tag{13}$$

Here, $\Theta_i^{nw-DRR}$ is the latency of the nw-DRR server and equals to a work conserving DRR scheduler, given in (6).

Consider a node architecture with nw-DRR at the output ports, as depicted in Figure 5. Let us calculate the maximum burst of $i_q^{D+1}$. Let us denote it with $\sigma_q$. $i_q^{D+1}$ is the set of flows from the input port q to the output port under observation of the node (D+1). We argue that $\sigma_q \leq \sigma_{I^D}$, and $\sigma_{I^D} \leq \sum_p (\varphi_p + L_p)$, therefore

$$\sigma_q \leq \sum_p (\varphi_p + L_p). \tag{14}$$

Similarly, we can argue that $\sigma_p \leq \sum_o (\varphi_o + L_o)$, if the input port p is connected to an nw-DRR scheduler in the upstream node (D-1), with $o$ representing the input ports to the output port under observation in the node (D-1).

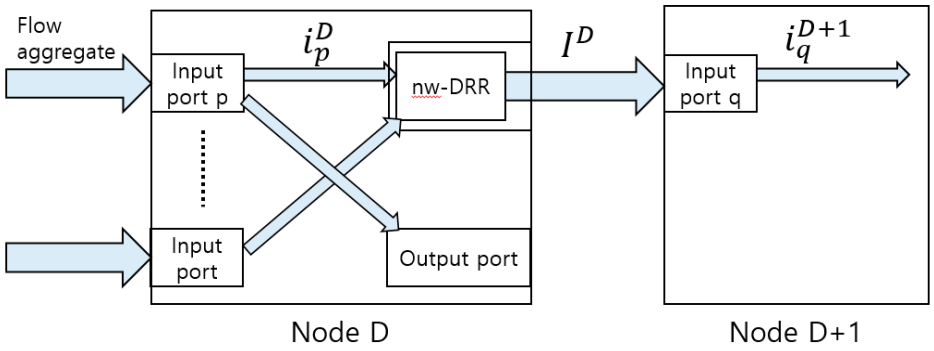


**Figure 5.** Architecture of a relaying node with the nw-DRR scheduler at the output ports.

The first argument above ($\sigma_q \leq \sigma_{I^D}$) holds. For any $t$ and $t_0$, $t > t_0$,

$$\begin{aligned} A_{I^D}(t_0, t) &= \sum_q A_{i_q^{D+1}}(t_0, t) = \sum_q (\sigma_{i_q^{D+1}} + \rho_{i_q^{D+1}}(t - t_0)) \\ &= \sum_q \sigma_{i_q^{D+1}} + \sum_q \rho_{i_q^{D+1}}(t - t_0) = \sigma_{I^D} + \rho_{I^D}(t - t_0), \text{ and } \sum_q \rho_{i_q^{D+1}}(t - t_0) = \rho_{I^D}(t - t_0). \end{aligned}$$

Therefore $\sum_q \sigma_{i_q^{D+1}} = \sigma_{I^D}$, and $\sigma_q \triangleq \sigma_{i_q^{D+1}} \leq \sigma_{I^D}$, for any $q$. The second argument ($\sigma_{I^D} \leq \sum_p (\varphi_p + L_p)$) also holds because of (11).

We can now calculate the delay of packets in $i_p^D$ at the node D. The delay of $i_p^D$ is bounded by (13), where $\rho_i$, $L_i$, and $\Theta_i^{nw-DRR}$ can be easily obtained by letting $i_p^D$ be the aggregate of all the flows traverse from the input port p to the output port under observation at the node D. The maximum burst of $i_p^D$, $\sigma_p$, can be obtained similarly with (14).

## 4. Case Study

### *4.1. Case of a Network with a Cycle*

In this subsection we investigate the performance of the nw-DRR with an exemplary network depicted in Figure 6, which was suggested in [19] for the study of the performance of the TSN asynchronous forwarding module in Figure 3. By adopting the already studied network topology, we can clearly compare the performance of the two solutions head to head. The flow under observation, $f_1$, starts from the host 1, traverses the node 1, 2, 3, 4, and then sinks to the host 4. All the other flows from $f_2$ to $f_5$ have the same parameters with $f_1$. Low priority traffic also exists and occupies the low priority traffic queue in each nw-DRR.

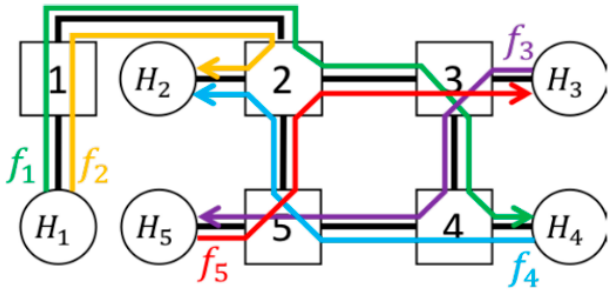


**Figure 6.** Network topology with a cycle, which is used for the case study in this paper and in [19].

We calculated the maximum end-to-end delay while varying the maximum packet length that applies the same to both high and low priority traffic, the max input burst size, the flow input data rate, and the quantum size, and fixing the link capacity, as in Table 2.

**Table 2.** Parameter values used for the case study.

| Parameter | Value |
|---|---|
| $L$ (Maximum packet length) | 400–3200 bit |
| $r$ (Link capacity) | 100 Mbps |
| $\rho_i$ (Arrival rate of a flow) | 10–40 Mbps |
| $\sigma_i$ (Max initial burst size of a flow) | 400–3200 bit |
| $\varphi_i$ (Quantum value assigned for a flow with $\rho_i$) | 80–400 bit |

The delay at a node is calculated with (13). At the first node $f_1$ and $f_2$ share a queue, since they enter the node with the same port. The maximum burst at the first node is equal to the max packet length. Therefore the delay of $f_1$ at the first node satisfies the inequality $D^1_{f1} \leq \Theta^1_1$, where $\Theta^1_1 = \frac{1}{r}\left[\left(\frac{r}{\rho_i} - 2\varphi_i\right)\left(1 + \frac{L}{2\varphi_i}\right) + 2L\right]$ from (6). The delay of $f_1$ at the nodes from 2 to 4 satisfies the inequality

$$D^j_{f1} \leq \frac{\sigma^j_1 - L}{\rho_i} + \Theta^j_1, \tag{15}$$

where $\Theta^j_1 = \frac{1}{r}\left[\left(\frac{r}{\rho_i} - \varphi_i\right)\left(1 + \frac{L}{\varphi_i}\right) + 3L\right]$ with $j$ is the node identifier that $j$ = 2, 3, or 4. $\sigma^j_1$ in (15) is the max burst into the queue of the output port of node $j$, to which $f_1$ belongs. $\sigma^j_1$ is calculated with (14), and in our case

$$\sigma^j_1 \leq 2(\varphi_i + L), \tag{16}$$

for all $j$ = 2, 3, or 4, since there are always two flows in the queue in the previous node $j$-1.

Figure 7 shows the maximum end-to-end delay of $f_1$ with varying max packet size from 400 bit to 3200 bit, flow arrival rate from 10 Mbps to 40 Mbps, and a fixed quantum value at 80 bit. It is shown that the max packet length and the input data rate are both significant parameters for the delay. The input data rate is inversely proportional to the max delay. One can consider to increase the input data rate during the connection setup phase, while actual amount of traffic remains unchanged, in order to reduce the max delay.

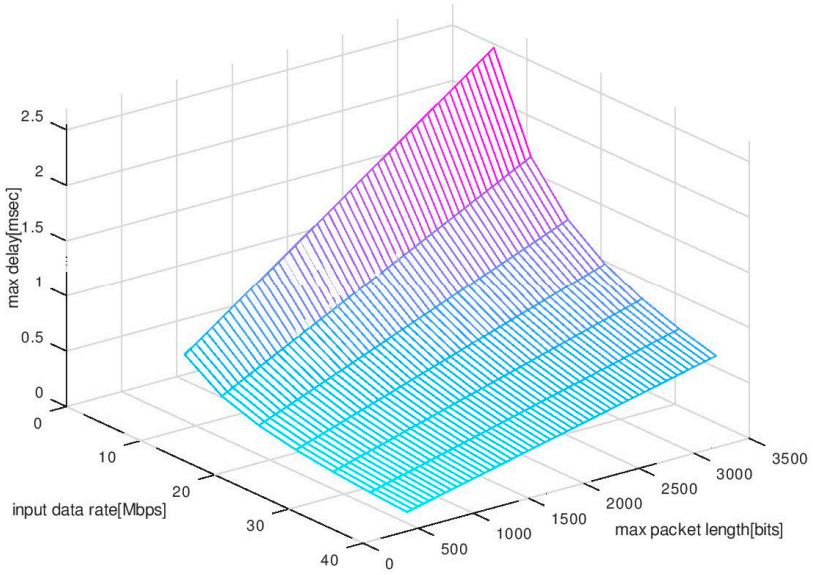


**Figure 7.** Max end-to-end delay with variable max packet size, variable flow arrival rate, and fixed quantum value 80 bit.

The Table 3 shows the exact values of the max end-to-end delay at sample parameter value sets.

**Table 3.** End-to-end delay bound at sample parameter sets (ms) with the fixed quantum value at 80 bit.

| $L \backslash \rho_i$ | 10 Mbps | 40 Mbps |
|---|---|---|
| 400 bit | 0.364 | 0.109 |
| 1000 bit | 0.796 | 0.249 |
| 3200 bit | 2.38 | 0.76 |

As in Table 3, with the quantum value 80 bit and flow arrival rate 10 Mbps, the maximum end-to-end delay is around 2 ms even when the max packet length is 400 Byte. If we can limit the max packet length to 50 Byte, the max delay reduces to 0.364 ms.

Next, we investigated how the max end-to-end delay changes when the quantum size varies. The smaller quantum size implies more computational burden to the scheduler.

Figure 8 shows that the quantum size does not significantly affects the delay. When the max packet length becomes larger, the effect of the quantum value becomes even less significant. One can choose to use a relatively large quantum value in case the complexity is one of key design issues, without sacrificing the delay performance too much.

Next, we compare our performance with the ATS system in [19]. It is revealed that the ATS system shows the max end-to-end delay of 0.70 ms when the max packet length 1000 bit, flow arrival rate 20 Mbps [19]. The same parameter set gives 0.431 ms with quantum value 80 bit and 0.575 ms with quantum value 400 bit in our result, which can be found at Table 4. This result is summarized in Table 5.

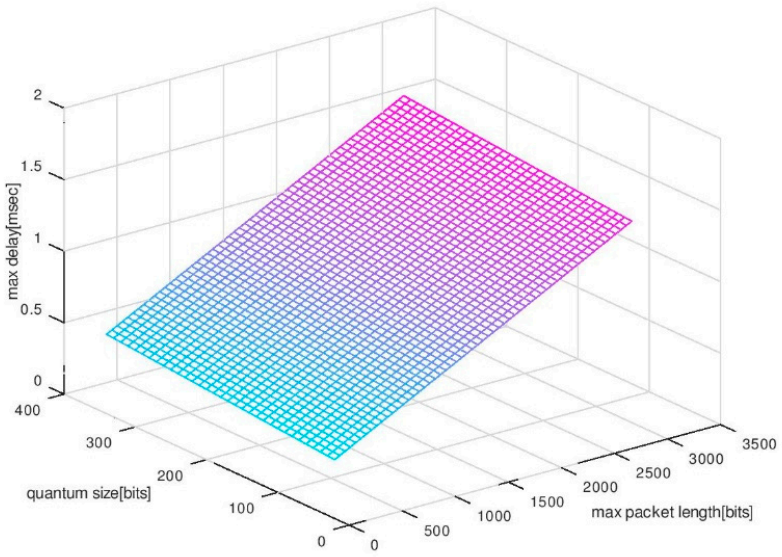


**Figure 8.** Max end-to-end delay with variable max packet size, variable quantum value, and fixed flow arrival rate at 20 Mbps.

**Table 4.** End-to-end delay bound at sample parameter sets (ms) with the flow arrival rate at 20 Mbps.

| $L \backslash \varphi_i$ | 80 bit | 400 bit |
|---|---|---|
| 400 bit | 0.194 | 0.338 |
| 1000 bit | 0.431 | 0.575 |
| 3200 bit | 1.30 | 1.444 |

**Table 5.** End-to-end delay bound comparison with [19] (ms) with the maximum packet size 1000bit and the flow arrival rate at 20Mbps.

| [19] | This Paper with $\varphi_i$ = 80 bit | This Paper with $\varphi_i$ = 400 bit |
|---|---|---|
| 0.70 | 0.431 | 0.575 |

The performances of two different approaches are remarkably similar, while ours is slightly better. This is encouraging considering the simplicity of the proposed architecture.

### *4.2. Case of a Seven-Hop Network*

In this subsection a network with a flow that traverses seven hops is considered. The early residential Ethernet TG defined the "2 ms delay bound in seven hops" as an ultimate performance goal for the strictest applications. We will investigate if this goal can be met with the proposed RSC. Consider the following topology depicted partly in Figure 9. There are 6 relaying nodes between the source and the destination of the flow, which makes up seven hops between the source and the destination. All the nodes have multiples input and output ports. The flow under observation enters the input port 1 and then departs from the output port 1, in every bridge node. A flow with the same specification with the flow under observation comes in at every other input port and then leaves at the output port 1. The set of such flows from all the input ports is called crossing flows. The crossing flows in a node shares the output ports with the flow under observation, but are separated to different output ports at the next node. This pattern continues at the subsequent nodes. This type of tandem networks is suitable for the investigation of the delay performance of schedulers with flow aggregation [23]. Although not specified in the figure, there exists low priority traffic at every output port. This setup of flows makes the network quite busy and large-scale.

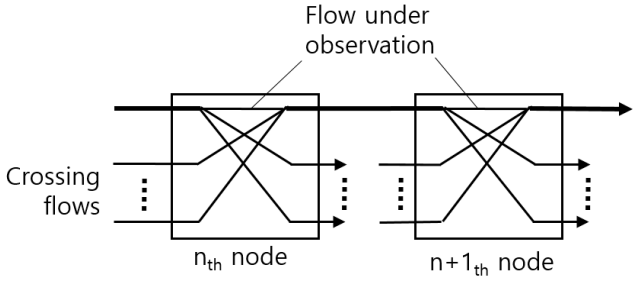


**Figure 9.** Network topology of a series of relaying nodes with the presence of crossing flows.

As in Section 4.1, we set the parameters within a predefined range specified in Table 6. We fixed the values of $\rho_i$, $\varphi_i$. The number of crossing flows, $N$-1, varies from 1 to 8, which makes the number of ports, $N$, from 2 to 9. The number of queues in an RSC scheduler is then $N$+1, because of the low priority traffic queue.

**Table 6.** Parameter values used for the case study.

| Parameter | Value |
|---|---|
| $L$ (Maximum packet length) | 400–1600 bit |
| $r$ (Link capacity) | 100 Mbps |
| $\rho_i$ (Arrival rate of a flow) | 10 Mbps |
| $\sigma_i$ (Max initial burst size of a flow) | 400–1600 bit |
| $\varphi_i$ (Quantum value assigned for a flow with $\rho_i$) | 80 bit |
| $N$-1 (Number of crossing flows in a node) | 1–8 |

Let us call $f_1$ the flow under observation. The delay of $f_1$ at the nodes from 1 to 6 satisfies the inequality

$$D_{f1}^{j} \leq \frac{\sigma_1^j - L}{\rho_i} + \Theta_1^j, \tag{17}$$

where $\Theta_1^j = \frac{1}{r}\left[\left(\frac{r}{\rho_i} - \varphi_i\right)\left(1 + \frac{L}{\varphi_i}\right) + (N+1)L\right]$ with $j$ is the node identifier that $j = 1, \ldots, 6$. $\sigma_1^j$ in (17) is the max burst into the queue of the output port of node $j$, to which $f_1$ belongs. $\sigma_1^j$ is calculated with (14), and in our case

$$\sigma_1^j \leq N(\varphi_i + L), \tag{18}$$

for $j = 2, \ldots, 6$, since there are always $N$ high priority queues in the scheduler in the previous node $j$-1. The maximum burst at the first node is equal to the max initial burst, which is same with the packet length in our case. Therefore $\sigma_1^1 = \sigma_1 = L$.

Figure 10 shows the maximum end-to-end delay of $f_1$, $\sum_{j=1}^{6} D_{f_1}^{j}$ with varying max packet size from 400 bit to 1600 bit, number of crossing flows from 1 to 8, fixed flow arrival rate at 10 Mbps, and a fixed quantum value at 80 bit.

As depicted in Figure 10, even when the number of crossing flows is eight, which represents quite a busy network, the end-to-end delay bound can be near 2 ms if we set the maximum packet length to be less than 400 bit.

Table 7 shows that with the maximum packet size 400 bit, or 50 byte, even if the number of crossing flows at every node is 8, the maximum delay is 2.175 ms, which is around the IEEE TSN's ultimate performance goal of the "2 ms with seven hops".

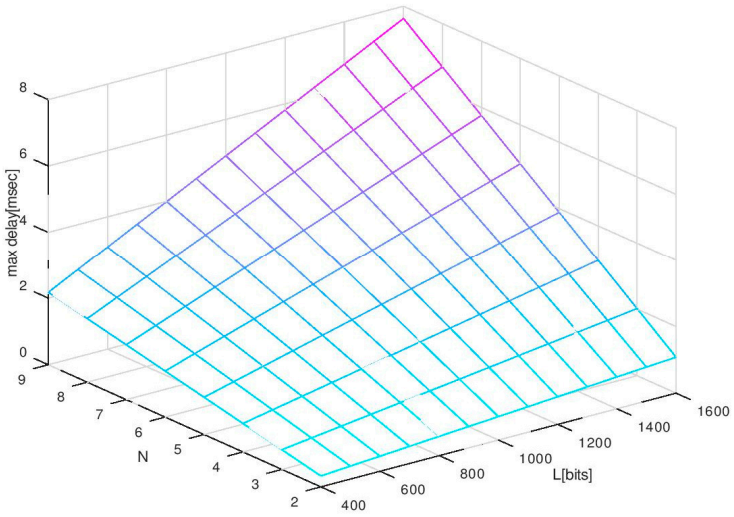


**Figure 10.** Max end-to-end delay of a seven-hop network with variable max packet size, variable number of crossing flows, and a fixed arrival rate at 10 Mbps.

**Table 7.** End-to-end delay bound (mc) at sample parameter sets with the fixed arrival rate at 10 Mbps.

| ***L**N*** | **2** | **9** |
|---|---|---|
| 400 bit | 0.327 | 2.175 |
| 1600 bit | 1.08 | 7.632 |

## 5. Discussion

A method of building a regulating scheduler (RSC) from a fair scheduler is proposed in this paper, which has non-work conserving characteristic. Possibility of the RSC as a delay bounding module is investigated. Especially, the DRR scheduler is modified to be an RSC, the nw-DRR. The nw-DRR is proved to be an LR server, with the same {latency, service rate} parameters with the original DRR server. The nw-DRR is then applied to work as an input-port based scheduler. At an output port of a relaying node the high priority traffic is divided based on their input ports and allocated to different queues. The low priority traffic is allocated to another queue at the output port. Therefore, if there are $N$ input ports, ($N$+1) queues will be established.

The input-port based scheduling prevents the ill-behaving flows from overwhelming the network, if applied at the boundary of a network. As such it can guarantee the fair resource sharing among all the flows in the high priority class, without per-flow scheduling. For example, if the first Ethernet switch from a flow's source host employs an input port-based RSC at the output ports, then automatically the flow is regulated, therefore shares only a fair amount of network resources.

Low priority traffic also benefits from an RSC since they do not suffer from the exhaustion, the bandwidth monopoly by high priority traffic. Low priority traffic is assigned a single queue in an RSC and guaranteed to receive the remaining available bandwidth without any resource reservation process. This property fits well to the "best-effort" traffic class, literally, and is in line with the available bit rate (ABR) service paradigm defined in the ATM networks [24].

The performance of the nw-DRR was investigated with realistic network topologies. The first network examined has a cycle, so the class-based scheduler would not meet any delay bound requirement. It was shown that in over various network parameters range, the nw-DRR bounds the delay within a few milliseconds. The network topology was adopted from [19] to compare the performance of the nw-DRR with that of the ATS approach of TSN. While the two perform similarly, ours was slightly better in the given parameter set. Moreover, our architecture is simpler in two ways. First, it is purely input port-based. Second, the regulator and the scheduler are in one piece. Despite the simplicity, the performance was acceptable. It is shown that the max packet length and the input

data rate are both significant parameters for the delay. The input data rate is inversely proportional to the max delay. One can consider to increase the input data rate during the connection setup phase, while actual amount of traffic remains unchanged, in order to reduce the max delay. The quantum size does not significantly affect the delay. When the max packet length becomes larger, the effect of the quantum value becomes even less significant. One can choose to use a relatively large quantum value in case the complexity is one of key design issues, without sacrificing the delay performance too much. In the second network examined, a flow under observation traverses seven-hops with a number of crossing flows. Even with eight crossing flows at every node, which represents quite a busy network, if we set the maximum packet length to be less than 400 bit, then the maximum end-to-end delay bound is around the IEEE TSN's ultimate performance goal of "2 ms with seven hops".

The non-work conserving nature of the proposed scheduler may increase the average delay. The guarantee on the delay bound, however, is crucial in emerging applications. Therefore, the RSC will play a key role in such application networks. Various fair schedulers can be modified to be an RSC. The DRR's latency depends on the sum of max packet lengths of each queues ($\sum_{n=1}^{N} L_n$), which makes it too large in situations with many input ports, and thus many queues in the nw-DRR. The virtual-finish time based LR servers' latency is usually dominated by the max packet length, not their sum, therefore will be more suitable to TSN. In future work, the possibility of modifying a virtual finish time based LR server to an RSC will be investigated. Although they are more complex, input port-based implementation will mitigate the complexity greatly.

**Funding:** This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B03932066).

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. IEEE 802.1 Time-Sensitive Networking Task Group Home Page. Available online: http://www.ieee802.org/1/pages/tsn.html (accessed on 13 December 2018).
2. IETF DetNet Working Group Home Page. Available online: https://datatracker.ietf.org/wg/detnet/documents/ (accessed on 13 December 2018).
3. Braden, R.; Clark, D.; Shenker, S. Integrated Services in the Internet Architecture: An Overview. Available online: https://tools.ietf.org/html/rfc1633 (accessed on 13 December 2018).
4. Parekh, A.; Gallagher, R.G. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Trans. Netw.* **1994**, *2*, 137–150. [CrossRef]
5. Golestani, S.J. A Self-Clocked Fair Queueing Scheme for High Speed Applications. In Proceedings of the IEEE Conference on Computer Communications and Networking (INFOCOM), Toronto, ON, Canada, 12–16 June 1994.
6. Zhang, H.; Ferrari, D. Rate-controlled service disciplines. *J. High Speed Netw.* **1994**, *3*, 389–412.
7. Xie, G.; Lam, S. Delay guarantee of virtual clock server. *IEEE/ACM Trans. Netw.* **1995**, *3*, 683–689. [CrossRef]
8. Shreedhar, M.; Varghese, G. Efficient fair queueing using deficit round-robin. *IEEE/ACM Trans. Netw.* **1996**, *4*, 375–385. [CrossRef]
9. Georgiadis, L.; Guerin, R.; Peris, V.; Sivarajan, K.N. The effect of traffic shaping in efficiently providing end-to-end performance guarantees. *Telecommun. Syst.* **1996**, *5*, 71–83. [CrossRef]
10. Stiliadis, D.; Varma, A. Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms. *IEEE/ACM Trans. Netw.* **1998**, *6*, 611–624. [CrossRef]
11. Joung, J.; Choe, B.-S.; Jeong, H.; Ryu, H. Effect of Flow Aggregation on the Maximum End-to-End Delay. In Proceedings of the High Performance Computing and Communications, Munich, Germany, 13–15 September 2006; Volume 4208, pp. 426–435.
12. IEEE 802.1 Time-Sensitive Networking Task Group P802.1Qch—Cyclic Queuing and Forwarding. Available online: https://1.ieee802.org/tsn/802-1qch/ (accessed on 13 December 2018).

13. Dürr, F.; Nayak, N.G. No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN). In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 203–212. [CrossRef]
14. Specht, J.; Samii, S. Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks. In Proceedings of the 28th Euromicro Conference on Real-Time Systems (ECRTS), Toulouse, France, 5–8 July 2016; pp. 75–85.
15. Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W. An Architecture for Differentiated Service, IETF RFC 2475. Available online: https://tools.ietf.org/html/rfc2475 (accessed on 13 December 2018).
16. Le Boudec, J.-Y. Some properties of variable length packet shapers. *IEEE/ACM Trans. Netw.* **2002**, *10*, 329–337. [CrossRef]
17. Gao, F.; Qian, H. Efficient, real-world token bucket configuration for residential gateways. *IEEE/ACM Trans. Netw.* **2016**, *24*, 462–475. [CrossRef]
18. Thangamuthu, S.; Concer, N.; Cuijpers, P.J.L.; Lukkien, J.J. Analysis of Ethernet-switch traffic shapers for in-vehicle networking applications. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2015; pp. 55–60.
19. Mohammadpour, E.; Stai, E.; Mohiuddin, M.; Le Boudec, J.-Y. Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping. In Proceedings of the 30th International Teletraffic Congress (ITC 30), Vienna, Austria, 4–7 September 2018.
20. Zhao, L.; Pop, P.; Zheng, Z.; Li, Q. Timing analysis of AVB traffic in TSN networks using network calculus. In Proceedings of the 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Porto, Portugal, 11–13 April 2018; pp. 25–36.
21. De Azua, J.A.R.; Boyer, M. Complete modelling of AVB in network calculus framework. In Proceedings of the 22nd International Conference on Real-Time Networks and Systems, Versaille, France, 8–10 October 2014; pp. 55–64.
22. Lenzini, L.; Mingozzi, E.; Stea, G. Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers. *IEEE/ACM Trans. Netw.* **2004**, *12*, 681–693. [CrossRef]
23. Bouillard, A.; Stea, G. Exact worst-case delay for FIFO-multiplexing tandems. In Proceedings of the 6th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS), Cargese, France, 9–12 October 2012; pp. 158–167.
24. Imer, O.C.; Compans, S.; Basar, T.; Srikant, R. Available bit rate congestion control in ATM networks. *IEEE Control Syst. Mag.* **2001**, *21*, 38–56.