

## Fair, Efficient and Low-Latency Packet Scheduling Using Nested Deficit Round Robin

Salil S.Kanhere and Harish Sethu

Department of ECE

Drexel University

3141 Chestnut Street, Philadelphia, PA 19104-2875.

E-mail: {salil, sethu}@ece.drexel.edu

### Abstract—

In the emerging high-speed integrated-services packet-switched networks, packet scheduling algorithms in switches and routers play a critical role in providing the Quality-of-Service (QoS) guarantees required by many applications. In this paper, we present a new scheduling discipline called *Nested Deficit Round Robin (Nested-DRR)*, which is fair, efficient and in addition has a low latency bound. Nested-DRR splits each DRR round into one or more smaller rounds, within each of which we run a modified version of the DRR scheduling discipline. In this paper, we analytically prove that Nested-DRR results in a significant improvement in the latency bound in comparison to DRR, and in addition preserves the good properties of DRR such as the per-packet work complexity of  $O(1)$ . Nested DRR also has the same relative fairness bound as DRR.

### I. INTRODUCTION

A variety of new applications such as video on-demand, distance learning and multimedia teleconferencing rely on network traffic scheduling algorithms in switches and routers to guarantee performance bounds and thus meet the Quality of Service (QoS) requirements. Given multiple packets awaiting transmission through an output link, the function of the scheduler is to determine the exact sequence of packets for transmission. Scheduling algorithms can be broadly classified into two categories—sorted priority schedulers and frame-based schedulers. The sorted priority schedulers need to maintain a global variable called the virtual time or system potential function. The priority of each packet, called the timestamp is calculated based on this variable. The packets are then scheduled in the increasing order of their timestamps. Note that this requires sorting of the packets based on their timestamps. Examples of sorted-priority schedulers are Weighted Fair Queueing (WFQ) [1], Self-clocked Fair Queueing (SCFQ) [2], Start-time Fair Queueing (SFQ) [3] and Worst-case Fair Weighted Queueing (WF<sup>2</sup>Q) [4]. They differ in the manner in which they calculate the global virtual time function. They generally provide good fairness and a low latency but are not very efficient due to the complexity involved in computing

the system virtual time and sorting the packets based on the timestamps. On the other hand, in frame-based schedulers like Deficit Round Robin (DRR) [5] and Elastic Round Robin (ERR) [6], the scheduler visits all the non-empty queues in a round robin order. The service received by a flow during its round robin service opportunity is proportional to its fair share of the bandwidth. The frame-based schedulers do not maintain the global virtual time function and also do not require any sorting among the packets. The per-packet work complexity of the frame-based scheduling disciplines, therefore, is  $O(1)$ . However, frame-based schedulers typically are not as good in fairness and latency. In particular, in the case of DRR, it can be shown that the latency bound of a flow  $i$  can be very large in DRR if there is another active flow  $j$  contending for the same output link with a much higher reserved rate.

The Nested DRR scheduler, proposed in this paper, modifies the frame-based DRR scheduler, by creating a nested set of multiple frames inside each DRR frame. This results in a significantly lower latency bound, while preserving the  $O(1)$  work complexity and the fairness characteristics of DRR. Under any combination of reserved rates, the latency bound of Nested DRR is almost always better than with DRR, and is the same as DRR only when the reserved rates are all identical.

Section 2 of this paper presents a brief overview of the DRR scheduling algorithm [5]. Section 3 presents the Nested DRR algorithm along with a pseudo-code implementation. Section 4 presents analytical results which prove the fairness, latency and efficiency properties of Nested DRR. Section 5 concludes the paper. While we present our work in this paper in the context of the DRR scheduler, the technique can actually be used on any frame-based scheduler to improve its latency bounds.

### II. DEFICIT ROUND ROBIN

Consider an output link of transmission rate  $r$ , access to which is controlled by the DRR scheduler. Let  $n$  be the total number of flows and let  $\rho_i$  be the reserved rate for flow  $i$ . Let  $\rho_{min}$  be the lowest of these reserved rates. Note that since all the flows share the same output link, a necessary constraint is that the sum of the reserved rates be no more than the transmission rate of the output link. In order that each flow receives

service proportional to its guaranteed service rate, the DRR scheduler assigns a weight to each flow. The weight assigned to flow  $i$ ,  $w_i$  is given by,  $w_i = \frac{\rho_i}{\rho_{min}}$ . Note that  $\forall i \in n, w_i \geq 1$ .

A flow is said to be *active* during a certain time interval, if it always has packets awaiting service during this interval. The DRR scheduler maintains a linked list of the active flows, called the *ActiveList*. At the start of an active period of a flow, the flow is added to the tail of the *ActiveList*. A *round* is defined as one round robin iteration during which the DRR scheduler serves all the flows that are present in the *ActiveList* at the outset of the round. Each active flow is assigned a *quantum* by the DRR scheduler. The quantum allocated to a flow is defined as the service that the flow should receive during each round robin service opportunity. Let  $Q_i$  represent the quantum assigned to flow  $i$  and let  $Q_{min}$  be the quantum assigned to the flow with the lowest reserved rate. The quantum assigned to flow  $i$ ,  $Q_i$  is given by  $w_i Q_{min}$ . Thus, the quanta assigned to the flows are in proportion of their reserved rates. In order that the work complexity of the DRR scheduler is  $O(1)$ , it is necessary that  $Q_{min}$  should be greater than or equal to the size of the largest packet that may potentially arrive during the execution of the scheduler. Note that during some service opportunity, a flow may not be able to transmit a packet because doing so would cause the flow to exceed its allocated quantum. The scheduler maintains a per-flow state, the *deficit count*, which records the difference between the amount of data actually sent thus far, and the amount that should have been sent. This deficit is added to the value of the quantum in the next round, as the amount of data the scheduler should try to schedule in the next round. Thus, a flow that received very little service in a certain round is given an opportunity to receive more service in the next round.

DRR can be easily adapted for guaranteed-rate service scheduling, and can be shown to belong to the general class of schedulers called *Latency-Rate (LR) Servers* [7]. In our analysis, we use the definition of latency used in [7]. Let  $m$  be the size in bits of the largest packet that is actually served and  $M$  be the size in bits of the largest packet that may potentially arrive during the execution of the scheduling algorithm. Note that,  $M \geq m$ . Also  $Q_{min}$  is set equal to  $M$ . The latency bound of a flow  $i$  served by the DRR scheduler is given by,

$$\Theta_i \leq \frac{(W - w_i)M + (n - 1)(m - 1)}{r} + (m - 1)\left(\frac{1}{\rho_i} - \frac{1}{r}\right) \quad (1)$$

where  $n$  is the total number of active flows and  $W$  is the sum of the weights of all the active flows. It is seen from Equation (1) that if  $W \gg w_i$ , then the latency bound for flow  $i$  can be very large. One may note that the above latency bound stated here is tighter than the one derived in [8].

### III. NESTED DEFICIT ROUND ROBIN

As mentioned earlier, a major drawback of DRR is that the latency bound for a flow with a small share of the bandwidth can be very large if the other flows contending for the same output link have much larger reserved rates. In other words the latency bound of a flow depends heavily on the reserved rates of the other flows which are multiplexed on the same output link. In this section we propose *Nested DRR*, which significantly improves the latency bound of low-rate flows, without hurting the latency bound of the higher rate flows.

The basic principle of the Nested-DRR algorithm is to split each round in DRR, henceforth called an *outer round*, into one or more smaller *inner rounds*, and then execute a version of the DRR algorithm over these inner rounds. The assignments of the weights and the quanta to the flows sharing the same output link is identical to those in DRR. The Nested-DRR scheduler uses the *quantum* value,  $Q_i$ , associated with flow  $i$  in DRR, to indicate the service that should be allocated to flow  $i$  during each outer round. Thus, during each outer round, the Nested-DRR scheduler tries to serve  $Q_i$  worth of data from flow  $i$ . During each inner round, the Nested-DRR scheduler tries to serve  $Q_{min}$  worth of data from flow  $i$  during the service opportunity of flow  $i$ . Since  $Q_{min}$  is greater than or equal to  $M$ , the scheduler is guaranteed to serve atleast one packet from flow  $i$  during each service opportunity. Just as in DRR, when the amount served in a certain inner round is less than  $Q_{min}$ , the deficit is carried over to the next inner round. The deficit from the last inner round is carried over to the next outer round. During each outer round, an active flow  $i$  is considered *eligible* for service in as many inner rounds as are necessary in the scheduler's attempt to serve  $Q_i$  worth of data in each outer round. Thus, the primary difference between Nested-DRR and DRR is that, in DRR a flow will receive its entire service during a round in one service opportunity, whereas in Nested-DRR, this service may possibly be split across several inner rounds.

A pseudo-code implementation of the Nested-DRR scheduling algorithm is shown in Figure 1, consisting of *Initialize*, *Enqueue* and *Dequeue* routines. The *Enqueue* routine is called whenever a new packet arrives at a flow. The *Dequeue* routine is the heart of the algorithm which schedules packets from the queues corresponding to different flows.

The Nested-DRR scheduler maintains a linked list of flows which are active and eligible to receive service during the current inner round. Note that the scheduler also needs to maintain another linked list of the flows which are active but have finished receiving their allocated service during the current outer round, i.e., flows that are not eligible to receive further service during the remaining inner rounds. These two linked lists of active flows are called *ActiveList(0)* and *ActiveList(1)* respectively. The variable *ActiveListId* is used to identify the linked list of the eligible flows, whereas the variable *NextActiveListId* identifies the list of flows which have finished receiving

ing their fair share of the bandwidth during the current outer round but will be eligible for receiving service in the subsequent outer round. The outer round in progress ends when  $ActiveList(ActiveListId)$  becomes empty. This marks the start of the next outer round during which the roles of the two linked lists are reversed. In other words, the linked list referenced by  $NextActiveListId$  during the previous outer round will now be pointed to by  $ActiveListId$  during the present outer round and vice versa.

The Nested-DRR scheduler maintains a per-flow state known as the  $UnServedQuantum$  which keeps an account of the portion of its allocated quantum that has not yet been used up by the flow in the current round. A flow whose queue was previously empty and therefore not present in either of the two linked lists is added to the tail of  $ActiveList(ActiveListId)$  whenever a new packet belonging to that flow arrives and its  $UnServedQuantum$  is initialized to its quantum. The scheduler always selects the flow at the head of the  $ActiveList(ActiveListId)$  for service. Let  $UQ_i$  represent the  $UnServedQuantum$  for flow  $i$ . During the service opportunity of flow  $i$ , the amount of data that the scheduler tries to serve from the flow is equal to the sum of  $\min(UQ_i, Q_{min})$  and the deficit count from the pervious inner round. The scheduler also decrements  $UQ_i$  by the amount of flow  $i$ 's quantum that has already been served in the current service opportunity, i.e.  $\min(UQ_i, Q_{min})$ . After the service opportunity, if flow  $i$  is still active and if the available bandwidth is sufficient to schedule the next packet from flow  $i$  in the next inner round, then flow  $i$  is added back to the tail of  $ActiveList(ActiveListId)$ . Otherwise, if flow  $i$  cannot receive any further service in the current round and is still active, it is added to the tail of the  $ActiveList(NextActiveListId)$  and any unused  $UQ_i$  is added to the deficit count for use in the subsequent outer round. In this case,  $UQ_i$  is also initialized to  $Q_i$  in preparation for the next outer round. However, if flow  $i$  does not have any packets awaiting service, its deficit count is reset to zero and it is not added to either of the two lists. In order that the scheduler knows the number of flows it has to visit in any given inner round, we introduce the quantity  $RoundRobinVisitCount$  which denotes the number of flows that are in the  $ActiveList(ActiveListId)$  at the start of the round.  $RoundRobinVisitCount$  is decremented by one after each flow is served, and when it eventually equals zero it implies the end of an inner round.

Figure 2(a) illustrates the first round in an execution of the DRR scheduler. At the beginning of this round the deficit counts for all the flows are initialized to zero. The total bandwidth available to any flow  $i$ , during this round is equal to its quantum,  $w_i Q_{min}$ . Note that  $Q_{min}$  is set equal to 20 bits which is the size of the largest packet that may potentially arrive during the execution of the algorithm. The sizes of the packets transmitted by the flows are shown by the horizontal bars. Figure 2(b) illustrates the execution of the first outer round of the Nested-DRR scheduler, assuming that the same

*Initialize:* (Invoked when the scheduler is initialized)

```
RoundRobinVisitCount = 0;
ActiveListId = 0;
NextActiveListId = 1;
for ( $i = 0; i < n; i = i + 1$ )
     $DC_i = 0;$ 
```

*Enqueue:* (Invoked when a packet arrives)

```
 $i = QueueInWhichPacketArrives;$ 
if ( $ExistsInBothLists(i) == FALSE$ ) then
    AddQueueToActiveList( $ActiveListId$ )( $i$ );
    Increment  $SizeOfActiveList(ActiveListId)$ ;
     $UQ_i = Q_i;$ 
     $DC_i = 0;$ 
end if;
```

*Dequeue:*

```
while (TRUE) do
    if ( $RoundRobinVisitCount == 0$ ) then
        if ( $SizeOfActiveList(ActiveListId) == 0$ ) then
            Increment  $ActiveListId$  modulo 2;
            Increment  $NextActiveListId$  modulo 2;
        end if;
         $RoundRobinVisitCount = SizeOfActiveList(ActiveListId);$ 
    end if;
     $i = HeadOfActiveList(ActiveListId);$ 
    RemoveHeadOfActiveList( $ActiveListId$ );
     $DC_i = \min(UQ_i, Q_{min}) + DC_i;$ 
     $UQ_i = UQ_i - \min(UQ_i, Q_{min});$ 
    do
        TransmitPacket( $Queue(i)$ );
         $DC_i = DC_i - LengthInBitsOfTransmittedPacket;$ 
    while ( ( $IsEmpty(Queue_i) == FALSE$ ) and
            ( $LengthInBitsOfPacketAtHead(Queue_i) \leq DC_i$ ) )
    if ( $IsEmpty(Queue_i) == TRUE$ ) then
         $DC_i = 0;$ 
        Decrement  $SizeOfActiveList(ActiveListId)$ ;
    else
        if ( ( $UQ_i + DC_i$ ) <  $LengthInBitsOfPacketAtHead(Queue_i)$  ) then
            AddQueueToActiveList( $NextActiveListId$ );
            Increment  $SizeOfActiveList(NextActiveListId)$ ;
            Decrement  $SizeOfActiveList(ActiveListId)$ ;
             $DC_i = DC_i + UQ_i;$ 
             $UQ_i = Q_i;$ 
        else
            AddQueueToActiveList( $ActiveListId$ );
        end if
    end if
    Decrement  $RoundRobinVisitCount$ ;
end while;
```

Fig. 1. Pseudo-Code for Nested DRR

flows shown in Figure 2(a) are active with an identical traffic distribution. Note that the total service received by each flow in both the cases is equal.

#### IV. ANALYTICAL RESULTS

In this section we present analytical results on the work complexity, latency properties and fairness of the Nested-DRR

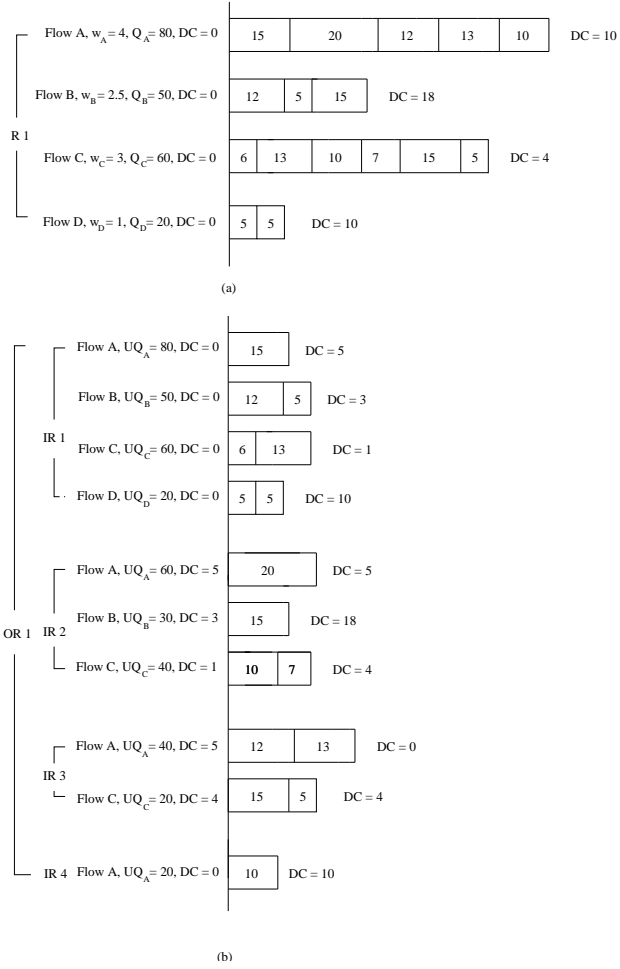


Fig. 2. (a) An execution of DRR (b) An execution of Nested-DRR

scheduler. In this paper, for reasons of brevity, we omit the proofs of the theorems. The following theorem shows that Nested-DRR is efficient and scalable with respect to the number of flows.

**Theorem 1.** The work complexity of a Nested-DRR scheduler is  $O(1)$  with respect to the number of flows.

Let  $(k, s)$  represent the  $s$ -th service opportunity received by a flow during its  $k$ -th outer round of service. Let  $\tau_i^{(k,s)}$  be the time instant marking the start of the  $(k, s)$ -th service opportunity of flow  $i$ . Assume that flow  $i$  becomes active at time instant  $\tau_i$  and that there are a total of  $n$  flows. In order that the latency bound for flow  $i$  is maximum we consider a time interval  $(\tau_i, \tau_i^{(k, \lceil w_i \rceil)})$  for any  $k$ , such that the time instant  $\tau_i$  must coincide with the start of some outer round  $k_0$  and flow  $i$  is the last to receive service during the inner round in which it receives its  $(k, \lceil w_i \rceil)$ -th service opportunity. The set  $G_{over}$  includes those flows that have completed receiving their allocated service in the duration of the  $(k_0 + k - 1)$ -th outer round

that is included in the time interval under consideration. Also, just as in DRR,  $Q_{min}$  is set equal to  $M$ .

**Theorem 2.** The Nested-DRR scheduler belongs to the class of  $\mathcal{LR}$  servers, with an upper bound on the latency  $\Theta_i$  for flow  $i$  given by,

$$\begin{aligned} \Theta_i \leq & \frac{(n-1)(m-1)}{r} + \frac{1}{r} \sum_{j \in G_{over}} w_j \cdot M \\ & + \frac{1}{r} \sum_{\substack{j \in (n-G_{over}) \\ j \neq i}} \lceil w_j \rceil \cdot M \\ & + (\lceil w_i \rceil - 1) \cdot M \left( \frac{1}{r} - \frac{1}{\rho_i} \right) \\ & + (m-1) \left( \frac{1}{\rho_i} - \frac{1}{r} \right) \end{aligned} \quad (2)$$

where  $n$  is the total number of active flows.

In this paper, we do not present a complete comparison between the latency expressions for DRR and Nested-DRR. Instead, in order to gain a quick understanding of the difference in the latency bounds of Nested-DRR and DRR, we consider two boundary conditions.

*Case 1:*  $\rho_i \ll \rho_j \forall j \in n, j \neq i$ .

In this case,  $w_i = 1$  and  $w_i \ll w_j, \forall j \in n, j \neq i$ . Also  $W \gg w_i$  and the set  $G_{over}$  will be empty. Hence from Equations (1) and (2), one can readily verify that the latency bound of flow  $i$ , is much lower for the Nested-DRR scheduler than that for the DRR scheduler.

*Case 2:*  $\rho_i \gg \rho_j \forall j \in n, j \neq i$ .

Now,  $w_i \gg w_j, \forall j \in n, j \neq i$ . Also the set  $G_{over}$  will include all  $(n-1)$  flows excluding flow  $i$ . From Equations (1) and (2), we can conclude that the latency bound for the Nested-DRR scheduler will still be marginally lower than that for the DRR scheduler.

In general, Nested-DRR has a significantly improved latency bound as compared to DRR. In addition, the latency bound of a flow  $i$  depends more on the reserved rate,  $\rho_i$  for flow  $i$  than on the reserved rates of the other flows contending for the same output link. The only exception is when the reserved rates for all the flows sharing an output link are the same, in which case the latency bound in DRR and Nested-DRR are the same.

We measure the fairness of Nested-DRR using the *Relative Fairness Bound (RFB)*, a popular metric first proposed in [2]. The RFB is defined as the maximum difference in the normalized service received by any two active flows over all possible intervals of time.

**Theorem 3.** For any execution of the Nested-DRR scheduler,  $RFB < M + 2m$ .

Note that the above bound on relative fairness is equal to that for DRR [5]. The above analytical results show that Nested-DRR is as fair and efficient as DRR and, in addition, results in an improved latency bound.

## V. CONCLUSION

In this paper we have presented a modified version of the DRR scheduler known as the Nested-DRR scheduler which creates a nested set of multiple frames inside each DRR frame. We have proved that Nested-DRR achieves a work complexity of  $O(1)$  with the same bound on the relative fairness measure as DRR. In addition, we have also proved that the latency bound of Nested-DRR is almost always better than DRR. We believe that our technique can also be applied to other frame-based schedulers such as Elastic Round Robin [6] resulting in similar improvements in latency bounds.

## REFERENCES

- [1] A. Demers, S. Keshav and S. Shenker, "Design and analysis of a fair queuing algorithm," *Proceedings of ACM SIGCOMM*, pp. 1-12, Austin, TX, September 1989.
- [2] S.J.Golestani, "A self-clocked fair queueing scheme for broadband applications," *Proceedings of IEEE INFOCOM*, pp. 634-646, Toronto, June 1994.
- [3] P. Goyal, H.M.Vin and H.Cheng, "Start-time fair queueing: a scheduling algorithm for integrated services packet switched networks," *IEEE Transactions on Networking*, Vol. 5, No. , pp. 690-704, October 1997.
- [4] J.C.R.Bennett and H.Zhang, "WF<sup>2</sup>Q: worst-case fair weighted fair queueing" *Proceedings of IEEE INFOCOM*, pp. 120-128, San Francisco,CA, March 1996.
- [5] M. Shreedhar and George Varghese, "Efficient fair queueing Using deficit round-robin," *IEEE Transactions on Networking*, Vol. 4, No. 3, pp. 375-385, June 1996.
- [6] S.Kanhere, A.Parekh and H.Sethu, "Fair and efficient packet scheduling in wormhole networks," *Proceedings of IPDPS 2000*, pp. 623-631, Cancun, Mexico, May 2000.
- [7] D. Stiliadis and A. Verma, "Latency-Rate servers: a general model for analysis of traffic scheduling algorithms," *IEEE Transactions on Networking*, Vol. 6, No. 3, pp. 611-624, October 1996.
- [8] D. Stiliadis and A. Verma, "Traffic scheduling in packet-switched networks: analysis, design, and implementation," *Ph.D dissertation, University of Santa Cruz, California*, June 1996.