A Dynamic Round-robin Packet Scheduling Algorithm

Yong He, Lei Gao
School of Computer Science and Engineering
Hunan University of Science and Technology
Xiangtan, Hunan, 411201, P. R. China
E-mail: ynghe@263.net, gaolei769123@126.com

Guikai Liu, Yuzhen Liu
School of Computer Science and Engineering
Hunan University of Science and Technology
Xiangtan, Hunan, 411201, P. R. China
E-mail: gkliu@hnust.edu.cn, 21182240@qq.com

***Abstract*—This paper puts forward a new dynamic round-robin (DYRR) packet scheduling algorithm with high efficiency and good fairness. DYRR algorithm introduces dynamic round-robin concept, that is, the allowance given to each of the flows in a given round is not fixed, but is related with the number of bytes sent of this and other flows of the last round scheduling. The time complexity of the DYRR algorithm is O(1). Results from performance simulation analysis shows that DYRR algorithm can effectively smooth output burst, realize fair scheduling, and have a good time delay characteristic.**

Keywords- scheduling; dynamic; round-robin; fairness; burs

I. INTRODUCTION

A variety of multimedia applications such as IP telephony, video conference call for packet network devices (ATM switches, IP router, etc.) to provide QoS for business, namely ask for specific requirements to the parameters such as delay, delay jitter, bandwidth, packet loss. A lot of relevant algorithms and methods have been proposed on how to provide QoS support. Packet scheduling is one of the most important methods, and is also important components in communication network QoS system structure [1].

The task of Packet scheduling is to transfer the data packets arriving at the network nodes out in a certain order. The switching nodes of network rely on the packet scheduling methods to cache, schedule and output for the entire packet arriving at them, achieving that provide different services for different types of service flow, thereby provide the corresponding QoS guarantees. The adoption of an appropriate scheduling algorithm is the key to provide QoS. Usually, a good scheduling algorithm should have the following features: (1). Reasonable distribution of link bandwidth. We should consider not only the high priority packet scheduling, but also to prevent low priority packets from being discard due to lack of scheduling for a long time[2]. (2). Predictability of Service. Good scheduler should serve the queue in a predictable way, and constantly adjust the laws of their services in order to ensure that the packets of each queue can be serviced in the scope of the configuration of the average bandwidth or delay within constantly adjust their service patterns to ensure that each queue packet can be provided with services within the configuration range of average bandwidth or delay [3]. The relation-ships between the data flows, queues, packet scheduling modules and output links is shown in Fig. 1.

The time complexity of round-robin scheduling algorithms is *O*(1). It is easy to realize and is suitable to use in high-speed networks. One of the most classical is DRR scheduling algorithm. But DRR algorithm has some shortcomings that its delay characteristic is not very ideal and its output burst is big. According to the deficiency of the DRR algorithm, we put forward a new dynamic round-robin packet scheduling algorithm with high efficiency and good fairness.

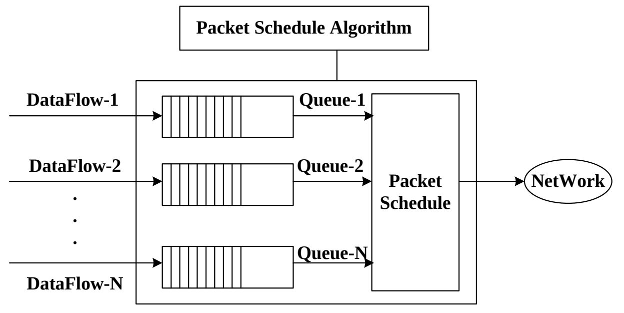

Figure 1. the relation-ships between data flow, queue, group scheduling module and the output link

II. THE EXITING PACKET SCHEDULING ALGORITHM OVERVIEW AND DRR ALGORITHM ANALYSIS

The main exiting packet scheduling algorithm can be divided into two classes: Time-stamp based and Round-robin based. The thought of Time-stamp basic is that which can keep two Time-stamp for every packet and mark the start time and the end time, and then sort the time as well as select the packet that which have the least time of service to send. This method such as WFQ (Weighted Fair Queuing), WF2Q (Worst-case Fair Weighted Fair Queuing), VC (Virtual Clock), SCFQ (Self-Clocked Fair Queuing) as so on, they all have a good time delay and fairness. Since it related to sequence calculate the time of every packet in system, the time-complexity of these methods is 0(*logN*) at least. There will be a higher time-complexity when the number of Activities data flow is a bit larger; therefore, this kind of method is greatly limited in practical application so it is unsuited for high-speed network equipment application [4].

The thought of the method of Round-robin is that the scheduler gives service to each queue of every flow in order to send the packet in the queue. It needn't to keep a time-stamp for every data flow in system, most of these have the time complexity of *O(1)* and realized easily. So It is more suitable for using in high-speed networks, such as DRR (Deficit Round Robin)—a typical Round-robin algorithm. Flowing is a brief description to the DRR algorithm.

Every service queue has a quantum and a deficit counter that associated with each flow in the DRR algorithm. The former is average bytes in every round service and the latter is initialized to the values of the former. The number of send bytes per round is equal to the quantum value theoretically: as long as the deficit counters greater than zero, the queue will get the services; after the packet gets the service, the deficit counter will minus the number of bytes of the packet; when the deficit counter is less than zero, the queue can't get service. When a new round begin, the deficit counter of all queue = the primary deficit counter + the quantum value. So through with given different quantum value can support different booking bandwidth [5].

From the description of DRR algorithm above we can know, applying the DRR scheduling algorithm can avoid the unfair that caused by different queue using different length of packet But DRR has the following deficiencies [6]:

- It can only provide fairness of long time scale, by the short time scale perspective, it will lead to the output of the data flow has a lot of burst.
- Time delay characteristic is also not ideal. Some queue can't get services for a long time.
- DRR give each queue a rating (Quantum), and through the DC (Deficit Counter) to measure the past inequality. Only when the first packet length of a queue less than the sum of rating and the value of the DC, will the scheduler send the packet, which requires that we must to know the length of the packet before sending the packet, but this increases the cost of the realization.

Therefore, the problems which need to solve at this stage in the kind of packet Round-robin method are to reduce the output burst and improve delay characteristics [7].

III. DYRR ALGORITHM DESCRIPTION

Because the DRR algorithm has some shortcomings above, according to the deficiency of DRR, this paper puts forward a new dynamic round-robin packet scheduling algorithm of high efficiency and good fairness. Its basic principle are described below: in order to show the number of data flows that need to access in a round, this paper introduce a counter to record the number of data flows. the counter called "AccessDataFlowCounter" that represents the number of data flows in “ActiveDataFlowList” at beginning of every round. DYRR algorithm introduce a byte counter "Sendi" used to record the number of data bytes sent out of each round from data flow i. We definite P-value as the byte count that a data flow permitted to send at every round. The p-value is not fixed in addition to the P-value of the first round of each data flow is 0, the rest round the P-value all need calculated again. The Calculation method is: P-value of this data flow in next round = P-value of this data flow in this round +average byte count that other data flow send at this round - byte count that this data flow send at this round. As long as P-value>0, the service queue and get services; after the packet sent, P-value minus "Sendi" of this packet, when P-value<0, this queue does not get service.

As we can see from the above algorithm, The data flow byte number P-value that each service queue is permitted to send each round is not fixed, and it must be recalculated again each round, and it is related to the number of bytes sent by itself and the last round, Obviously, For a service queue, if the number of bytes send in the last round is less than other queues, it will be compensated. In addition, we maintain an "ActiveDataFlowList" to store all activity data flow. In the implementation process of a round, the new arrival or back in the activities of the data flow can be added to the tail of "ActiveDataFlowList", from the next round began to be accessed. In order to show the number of data flow to be accessed in a round, we use the "AccessDataFlowCounter" to record the number of data flow. When the packet scheduling module access to a data flow, the "AccessDataFlowCounter" is reduced by 1, and when the counter is equal to zero it means the end of a round. The DYRR algorithm includes the initialization process, into the queue queuing process and out the queue scheduling process.

Initialization process is refers to the initialization operation on packet scheduling module when the network node has packet scheduling demand, including the initialization operation on "ActiveDataFlowList" and "AccessDataFlowCounter".

Each data flow corresponding to a queue, the process of into the queue is refers to that when a new packet arrived, it enter into the corresponding queue, waiting to send. If the data flow that the packet belongs to is not in "ActiveDataFlowList", add this data flow to the tail of "ActiveDataFlowList"; then Set the P-value of this data flow with initial value 0; set the value of "Sendi" of this data flow with the initial value 0.

The scheduling process of out queue is the core of the algorithm, describes that how to selection packet form the different queue that corresponding data flow, and sent the packet to the network through the output link. The specific process includes:

```
//the data structure of data flow
Class DataFlow
{
    //store the data of data flow
    Queue queue = new Queue();
    int send=0;
    int p = 0;
}
//define a list to store data flow object
List<DataFlow> activeDataFlowList= new List< DataFlow > ();
// define a int variable to store the count of active data stream
int dataFlowCount = 0;
Packege packge;
DataFlow activeDataFlow = null;
While (true) do
{
    if ((dataFlowCount = activeDataFlowList.length()) != 0)
        activeDataFlow = activeDataFlowtreamList [0];
        packge = activeDataFlow. queue.Pop();
        activeDataFlow.send = activeDataFlow.send + the length of packge;
        send packge;
        while (activeDataFlow. queue.length() != 0) do
            if (activeDataFlow.p - activeDataFlow.send>0) do
                packge = activeDataFlow. queue.Pop();
                activeDataFlow.send = activeDataFlow.send + the length of packge;
                send packge;
            end if;
            else do
            {
                move activeDataFlow to the end of activeDataFlowList;
                if (--dataFlowCount == 0) do
                    for (int i = 0; i < activeDataFlowList.length(); i++) do
                    {
                        Calculate the value of activeDataFlowList [i].p;
                        activeDataFlowList[i].send = 0;
                    }
                end if;
            }
            end if;
            if (activeDataFlow. queue.length() == 0) do
            {
                activeDataFlowList.delete(activeDataFlow);
                --dataFlowCount;
                break;
            }
    }
    end if
    // activeDataFlowList.length()) == 0
}
```

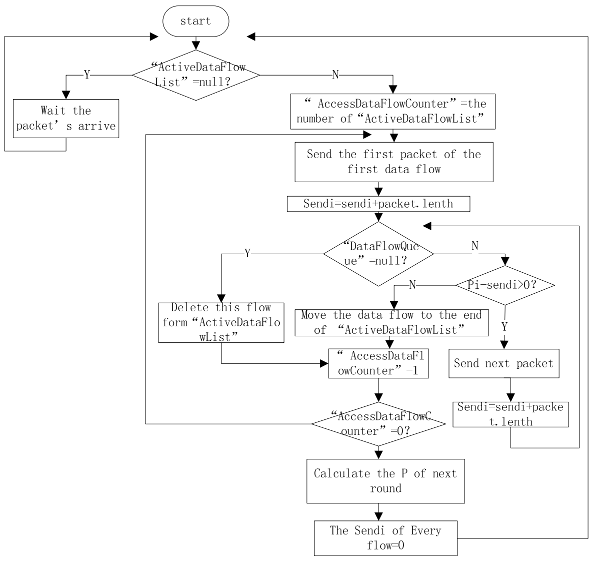
start
"ActiveDataFlowList" =null?
Y
N
Wait the packet' s arrive
" AccessDataFlowCounter" =the number of "ActiveDataFlowList"
Send the first packet of the first data flow
Sendi=sendi+packet.lenth
"DataFlowQueue" =null?
Y
N
Pi-sendi>0?
N
Y
Delete this flow form "ActiveDataFlowList"
Move the data flow to the end of "ActiveDataFlowList"
" AccessDataFlowCounter" -1
Send next packet
Sendi=sendi+packet.lenth
"AccessDataFlowCounter" =0?
Calculate the P of next round
The Sendi of Every flow=0

Figure 2. the flowchart of outputting queue

IV. SIMULATION AND PERFORMANCE ANALYSIS

This paper makes use of OMNET++ in the computer simulation for the performance of this paper's algorithm. The service is divided into ten classes which input the system in simulation. The numbers of the ten data flows were source 0-9 while the corresponding number of the buffer queue were 0-9, the packet length submit the random distribution from 30 to 100, the system would send a packet every 0.2s. We compared the performance in delay, fairness and burst between DYRR algorithm and DRR algorithm through the simulation.

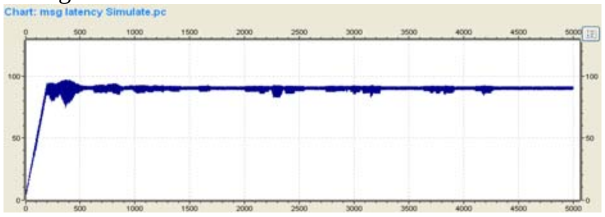

Figure 3. Every packet's delay of DYRR algorithm

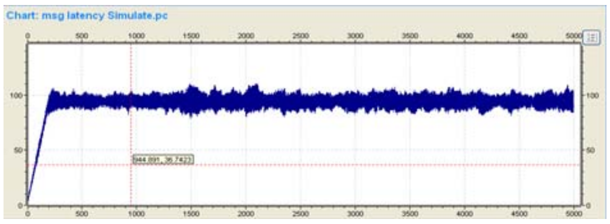

Figure 4. Every packet's delay of DRR algorithm

Fig. 3 and Fig. 4 respectively represent the simulation results of DYRR algorithm and DRR algorithm in the packet delay. From Figure 1, Figure 2 and the simulation results we know, the average delay of DYRR algorithm is 88.9s, and the average delay of DRR algorithm is 94.8 s.

TABLE I. THE EXITING PACKET SCHEDULING ALGORITHM OVERVIEW AND DRR ALGORITHM ANALYSIS

DYRR	1	2	3	4	5	6	7	8	9	10	11	12
Flow-0	125	133	161	268	222	210	149	147	216	170	165	179
Flow-1	158	204	161	220	200	230	141	154	194	150	231	199

TABLE II. THE NUMBER OF BYTES OF DATA QUEUE - 0 AND DATA QUEUE-1 IN DRR ALGORITHM SENT

DRR	1	2	3	4	5	6	7	8	9	10	11	12
Flow-0	207	95	225	261	91	185	168	113	143	230	231	159
Flow-1	137	162	316	167	94	190	127	138	149	335	130	190

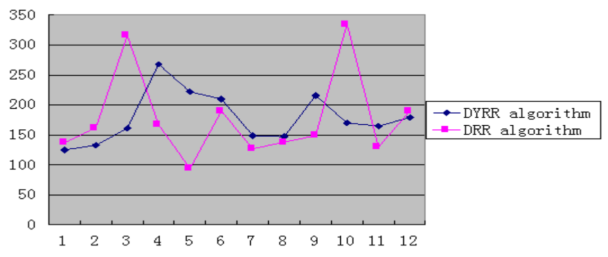

Figure 5. The number of bytes of data queue - 0 in DYRR algorithm and DRR algorithm sent every 5s

Fig. 5 represents the number of bytes of data queue - 0 in DYRR algorithm and DRR algorithm sent every 5s, the abscissa represents the time interval of 5s, the ordinate represents the number of bytes of data flow-0 sent every 5s. From Fig. 1, Table I and Table II, we can see that DYRR algorithm have smaller output burst than DRR algorithm.

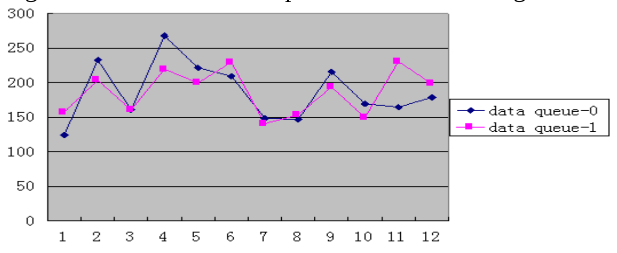

Figure 6. The number of send bytes that data queue - 0 and data queue-1 in DYRR algorithm

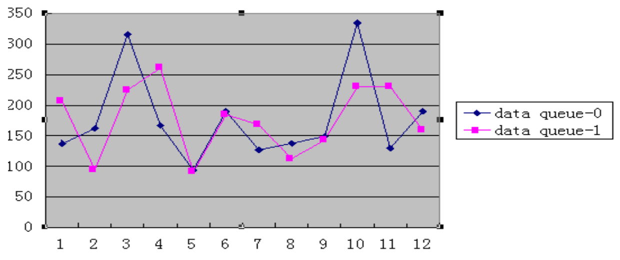

Figure 7. The number of bytes that data queue - 0 and data queue-1 in DRR algorithm sent

Fig. 6 and Fig. 7 respectively represent the number of bytes that data queue-0 and data queue-1 in the DYRR algorithm and the DRR algorithm sent every 5s. Form the Fig. 6, Fig. 7, Table I and Table II, it is known that compared with the DRR algorithm, the difference byte count that sended between data flow 0 and data flow 1 are smaller, and can improve Short-term fairness at the DYRR algorithm.

Through the computer simulation above, we can see the DYRR algorithm can realization in constant time as well as the DRR algorithm. The time complexity of the DYRR algorithm is still $O(1)$, achieving high efficiency. DYRR algorithm can effectively smooth output burst, realize fair scheduling, and has the good time delay characteristic.

V. CONCLUSIONS AND OUTLOOK

According to the deficiency of DRR this paper puts forward a new dynamic round-robin packet scheduling algorithm of high efficiency and good fairness (DYRR).it has the good characteristics that it's complexity is $O(1)$,and it's simple to realization. The DYRR algorithm introduces dynamic round-robin concept, that is, the number of bytes of allowed to send of each round scheduling is not fixed, but is related with the number of bytes sent of this and other flows of the last round scheduling. So the DYRR algorithm can effectively smooth output burst, realize fair scheduling, and have a good time delay characteristic. But DYRR algorithm could not allocate different bandwidth, the next work is to weight the queue scheduling of our algorithm, so that for different business it can allocated different flows bandwidth.

ACKNOWLEDGMENT

This work is supported by National Nature Science Foundation of China (No: 61272196), Scientific Research Fund of Hunan Provincial Education Department (No: 10B034), and Scientific Research Fund of Hunan Provincial Education Department (No: 11C0534).

REFERENCES

[1] Hongchao Hu, Peng Yi, Yunfei Guo, Yufeng Li. A Fair Service and Dynamic Rond Robin Scheduling Algorithm [J]. Journal of Software, 2008, 19(7): 1856-1864.

[2] Xiaodong Li, Lemin Li. LL-DRR: An Efficient Scheduling Algorithm for Packet Networks [J]. Journal of Electronics and Technology, 2002, 24(3).

[3] Xiang Wu, Hongwei Kong, Weizhang Wang, Ning Ge. An Improvement of DRR Packet Scheduling Algorithm [J]. Journal of Electronics and Technology, 2003, 25(5).

[4] Fei Gao, Yuan Zhang. An Improved Algorithm of DRR Combining with traffic shaper [A]. 2010 Third Pacific-Asia Conference on Web Mining and Web-Based Application [C]. 2010.

[5] Xiancheng Xu. Survey of Qos Scheduling Algorithms [J]. Journal of Henan University of Science and Technology, 2003, 24(4).

[6] SHREEDHA M, VARGHESE G. Efficient fair queuing using deficit round robin [J]. IEEE/ACM Transaction on Networking, 1996, 4(3):375- 385.

[7] Linhuan Zhong, Zhimei Wu, Zhong Zheng, Xianlei Wang, He Jiang. Research of Novel Packet Fair Scheduling Algorithm [J]. Application Research of Computers, 2008,25(4).