# 一种改进的 DRR 调度算法 [1]

伍 翔 孔红伟 汪卫章 葛 宁 冯重熙

(清华大学电子工程系 北京 100084)

**摘 要** 为了简单有效地分配链路带宽，该文分析了 DRR(Deficit Round Robin) 算法在数据交换实现上的局限性，提出了一种基于令牌扣除并平滑输出突发性的调度算法。该算法能够减小调度开销，平滑输出突发性，硬件实现简单。证明了在多数情况下改进算法的公平性优于 DRR 算法。仿真结果表明改进算法能够有效地分配输出链路的带宽，缓解令牌数选取对输出突发性和抖动性的影响。

**关键词** 调度，交换，DRR，突发性，公平性

**中图号** TN919.3

## 1 引 言

传统的 Internet 网络采用的是先到先服务 (FCFS) 的策略，不能有效地保证服务质量 (QoS)。在数据交换网络上，由于不同的数据流在交换节点互相影响会降低网络的性能。所以，一个支持不同服务等级的网络应该能够对资源进行有效地调度以满足有不同应用的要求。设计合适的调度策略具有非常重要的意义。在具有不同性质的业务竞争带宽的情况下，理想的调度算法应该具有以下一些特性：保证各数据流得到所分配的带宽；保证数据流之间不互相影响；实现简单。

近年来提出了大量的调度算法。其中有基于时间戳的算法 Virtual Clock[1]，基于 GPS (Generalized Processor Sharing) 的算法 WFQ (Weighted Fair Queueing)[2]， SCFQ (Self-Clocked Fair Queueing)[3]， $WF^2Q$ (Worst-case Fair weighted Fair Queueing)[4]，也有基于轮询的算法 WRR (Weighted Round-Robin)[5] 和 DRR (Deficit Round Robin)[6]。D. Stiliadis 对它们的延时特性进行了分析 [7]。

基于 GPS 的算法具有很好的延时特性，但是为获得理想的延时特性增加的设计复杂性太大，所以更多地是采用实现复杂度较小的基于轮询的算法。DRR 在对变长数据包的支持方面性能较好，而且可以做到实现复杂度为 $O(1)$[6]。本文将对 DRR 算法的一些不足之处进行改进，以提高其可实现性并改善它的输出突发特性。

## 2 DRR 局限性及其改进

**2.1 DRR 算法实现上的局限性** DRR 算法将有数据包等待发送的队列序号放入一个链表中。每次访问链表头上的队列时，先将队列的令牌数加上一个预先分配的值 (每次轮询允许发送的字节数)，然后服务该队列。每次服务前判断队头上的数据包长度是否大于队列令牌数。如果队列头上的数据包长度小于该队列剩余的令牌数，服务后队列令牌数减去发送的数据包长度，否则将其从链表中取出并插入链表尾部，然后访问链表中的下一个队列。当队列空时将其移出链表 [6]。

从 DRR 算法描述可知，实现 DRR 需要在发送数据包之前知道数据包长度以决定是否可以发送，但是这个要求会增加实现的开销。同时，DRR 持续对满足条件的链表头上的队列服务，直到队列用尽它的令牌。

目前采用的交换结构一般可以分为两种：一种是共享存储器结构；另一种是 Crossbar 结构。

在共享存储器交换中，所有的输入和输出口都要访问存储器模块。在一个时隙中，最多有 $N$ 个数据块从输入口写入存储器并且最多有 $N$ 个数据块被读出存储器。在一个 $N \times N$ 的共享

[1] 2001-09-18 收到，2002-06-13 改回
国家自然科学基金资助项目 (69896240)

存储器型交换机中，存储器会有 $N$ 次写入和 $N$ 次读出。如果每个端口速率为 $C$bps，那么共享存储器的速率为 $2NC$bps。

为了提高交换容量，在一个端口向存储器输入输出数据时，另一个端口进行准备（如路由表查询，调度计算等）。由于包长信息通常都在包头携带，如果调度需要包长信息有两种方法获得：一是从共享存储器中读取包长信息，这样相当于增加了对存储器的访问频率，从而会降低交换容量；另一种方法是使用另外的存储器复制一份队列中数据包的长度信息，这样会浪费存储器资源。

在 Crossbar 交换机中，调度需要输入口和输出口交换信息。如果在调度之前需要知道包长信息，就增加了调度时交互的信息量，提高了实现的复杂性。

所以，如果调度算法在调度之前不需要数据包长度信息，而在调度之后从发送的数据包中得到包长信息在实现上可以带来很大的好处。

另外，DRR 服务队列如图 1(a) 所示方式：连续服务一个队列直到用完令牌再服务下一个队列。图 1(b) 中服务器交替访问每个客户，对客户的访问更均匀[5]。前者服务可能由于令牌未用完而滞留在某个队列，造成输出突发性增加。为了减小输出数据的突发性，我们将算法改进为如图 1(b) 所示的方式。交织破坏了这种滞留，从而减小了突发性。

2.2 *算法改进*　如图 2 所示，2 个环形链表：Active1 和 Active2。它们交替存放正在被服务的队列序号和下一个轮询周期将要被服务的队列序号。idle 中存放的是空队列的序号。算法描述如下：

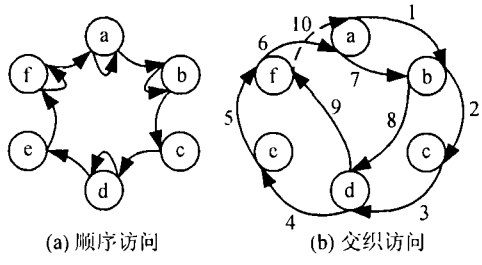


图 1　两种调度算法

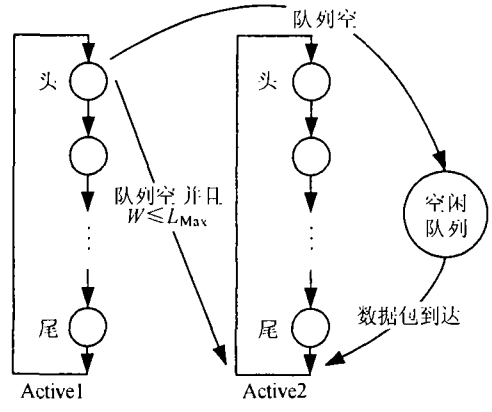


图 2　减小输出突发性的调度算法

令 $Q_i$ 表示第 $i$ 个队列，$\Phi_i$ 表示每次循环分配给队列 $i$ 的令牌数，$\mathrm{DC}_i$ 表示队列 $i$ 的令牌数，$L_{\mathrm{Max}}$ 表示最大数据包长度，$V$ 表示队列个数。

初始化
 ActiveNow=Active1
 ActiveNext=Active2
 For($i = 0; i < V; i = i + 1$)
  $\mathrm{DC}_i = L_{\mathrm{Max}}$;
入队模块：数据包 $p$ 到达
$i = p$ 所属的队列
If ($Q_i$ 由空变为非空) then
  将 $i$ 加入调度链表尾部;
  $\mathrm{DC}_i = \mathrm{DC}_i + \Phi_i$;
End if;

$p$ 加入 $Q_i$ .

出队模块:

```
While (TRUE) do
{
   While (ActiveNow 非空) do
   {
      将 ActiveNow 链头出队, 假设为 Q_i;
      发送 Q_i 队头数据包;
      DC_i =DC_i − Q_i 队头数据包长度;
      If (DC_i > L_Max) and (Q_i 非空) do
         将 i 重新加入 ActiveNow 尾部;
      Else if (Q_i 为空) then
        DC_i = L_Max ;
      Else
         将序号 i 插入 ActiveNext 尾部;
         DC_i =DC_i + Φ_i;
      End if;
   }
   ActiveNow<=>ActiveNext(交换服务链)
}
```

改进后算法判断是否允许发送一个数据包不需要知道数据包的长度. 调度之后发送数据包的同时可以得到数据包的长度, 然后从令牌数中扣除. 我们假设 $\Phi_i \geq L_{\mathrm{Max}}$ , 那么在离开 ActiveNow 的队列再次进入 ActiveNow 时令牌值一定会大于 $L_{\mathrm{Max}}$ , 即至少可以发送一个数据包, 调度器每次都能访问到可以发送的队列. 从而使算法的时间复杂度为 $O(1)$ .

## 3 性能分析

3.1 算法公平性 我们使用的公平性参数基于 Golestani 分析 SCFQ 时采用的服务公平指数 (SFI: Service Fairness Index)[3] . 假设 $r$ 为链路带宽, $W_i(\tau,t)$ 为在 $(\tau,t]$ 队列 $i$ 发送的业务量. 如果 $\rho_i$ 表示分配给队列 $i$ 的带宽, 我们称 $W_i(\tau,t)/\rho_i$ 为在 $(\tau,t]$ 内队列 $i$ 得到的服务. 如果任何两个队列 $i,j$ 在时间 $(t_1,t_2]$ 内持续有数据包等待发送, 并且

$$|W_i(t_1,t_2)/\rho_i - W_j(t_1,t_2)/\rho_j| \leq S \tag{1}$$

$S$ 为一个常数. 那么 $S$ 为服务的公平性.

令 $F$ 表示算法的最大帧长 $F=\sum_{i=1}^{V}\Phi_i$ , 其中 $V$ 为队列个数. 那么分配给一个队列的带宽为

$$\rho_i \geq \Phi_i r/F \tag{2}$$

考虑时间段 $(t_0,t_n]$ , 其中 $t_0$ 为第 1 个轮询周期的开始, $t_n$ 为 $t_0$ 后第 $n$ 个轮询周期的结束时刻. 如果 $k=1,2,\cdots,n$ 队列 $i$ 持续非空, 那么

$$W_i(t_{k-1},t_k) = \mathrm{DC}_i^{k-1} + \Phi_i - \mathrm{DC}_i^k \tag{3}$$

其中 $\mathrm{DC}_i^k$ 为连接 $i$ 在第 $k$ 个周期结束时的令牌数.

$$W_i(t_0,t_k) = k\Phi_i + \mathrm{DC}_i^0 - \mathrm{DC}_i^k \tag{4}$$

初始化时，$\mathrm{DC}_i = L_{\mathrm{Max}}$。当队列在 ActiveNow 链表中时，$\mathrm{DC}_i$ 最大为由空闲进入该链表的队列，为 $L_{\mathrm{Max}} + \Phi_i$；$\mathrm{DC}_i$ 最小为 $L_{\mathrm{Max}}$，否则它必须离开 ActiveNow 链表。当队列在 ActiveNext 链表中时，令牌数为队列离开 ActiveNow 时的令牌数加上 $\Phi_i$，由于离开 ActiveNow 链表的队列剩余令牌数大于 0 而且小于 $L_{\mathrm{Max}}$，所以在 ActiveNext 链表中的队列的令牌数满足 $\Phi_i < \mathrm{DC}_i < L_{\mathrm{Max}} + \Phi_i$。当队列空闲时，$\mathrm{DC}_i = L_{\mathrm{Max}}$，所以 $\Phi_i \le \mathrm{DC}_i^k \le L_{\mathrm{Max}} + \Phi_i$。再由 (4) 式可以得到 $W_i(t_0, t_k) \le k\Phi_i + L_{\mathrm{Max}}$。算法为提高效率，时间复杂性一般为 $O(1)$，即 $\Phi_i \ge L_{\mathrm{Max}}$，所以对每个 $k$

$$W_i(t_0, t_k) \le k\Phi_i + \Phi_i \tag{5}$$

同上，如果连接 $j$ 在相同的时间段内持续非空，那么

$$W_j(t_0, t_k) = k\Phi_i + \mathrm{DC}_j^0 - \mathrm{DC}_j^k \ge k\Phi_j - L_{\mathrm{Max}} \ge k\Phi_j - \Phi_j \tag{6}$$

如果时刻 $t$ 是任取的第 $k$ 个周期中的任意时刻，我们首先考虑算法改进前的情况。那么公平性最差的情况是一个队列消耗完所有的令牌，而另一个队列却还没有开始发送，如图 3(a) 所示。有

$$W_i(t_0, t) \le k\Phi_i + \Phi_i \tag{7}$$

$$W_j(t_0, t) \ge k\Phi_j - 2\Phi_j \tag{8}$$

由于队列得到的带宽和分配的令牌数成比例，即 $\Phi_i/\rho_i = \Phi_j/\rho_j$，再由 (2) 式可得改进前算法公平性参数

$$S_1 \le 3\Phi_i/\rho_i \le 3F/r \tag{9}$$

算法改进后，两个队列交织发送。如图 3(b) 所示。其中 $\varphi_i$ 与 $\varphi_j$ 分别为第 $k$ 周期队列 $i$ 和队列 $j$ 到时刻 $t$ 还可以发送的业务量。

$$W_i(t_0, t) \le k\Phi_i + \Phi_i - \phi_i \tag{10}$$

$$W_j(t_0, t) \ge k\Phi_j + \Phi_j - \phi_j \tag{11}$$

得到公平性参数

$$S_2 \le 2\Phi_i/\rho_i + \phi_j/\rho_j - \phi_i/\rho_i \tag{12}$$

其中 $\mathrm{Max}(\phi_j) = \Phi_j$, $\mathrm{Min}(\phi_i) = 0$。如果 $Q_i$ 只有一个数据包需要在本轮询周期发送，则图 3(b) 退化为图 3(a) 的形式。否则，它们不可能同时分别取得最大值和最小值。所以

$$S_2 < 3\Phi_i/\rho_i \tag{13}$$

由以上分析可知，算法改进后公平性能够得到改善。

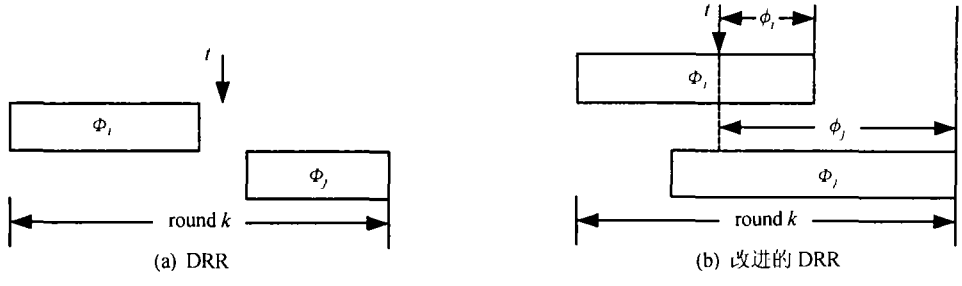


图 3　算法公平性

3.2 带宽分配 4 个数据流，$\Phi_i(i=0,1,2,3)$ 分别为 1600, 3200, 4800, 6400byte . 最大包长 $L_{\mathrm{Max}}=1500$byte . 包长为指数分布，平均包长为 1024byte . 数据包速率为 0.1Mpacket/s , 到达间隔为指数分布. 链路速率 155Mbps . 4 个数据流分别从 0s , 5s , 10s , 15s 开始发送.

从图 4 的仿真结果看到，在 5s 之前，只有 flow0 独占所有的输出带宽. 在 5s 之后 flow1 开始发送，由于 $\Phi_1=2\Phi_0$ , 所以 flow0 得到了 1/3 的输出带宽，flow1 得到了 2/3 的输出带宽. 在 15s 之后，4 个数据流都开始发送数据，得到的带宽分别为 1/10, 2/10, 3/10, 4/10, 与分配的令牌 $\Phi_i$ 比例相同.

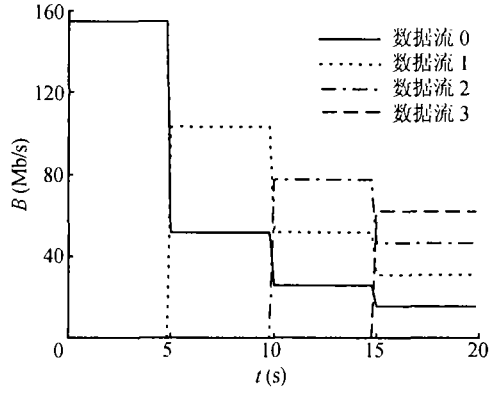


图 4 算法的带宽分配特性

3.3 令牌大小对性能的影响 由于数据网络上的数据包大小差别很大，为了满足 $O(1)$ 时间复杂度的要求，令牌大于最大的数据包长度. 根据 latency-rate server 理论 [7] 可以证明：当其它队列等待发送时，空队列到达的第一个数据包经历的最坏延时 $\Theta$ 与 DRR 算法相等，为

$$\Theta=(3F-2\Phi_i)/r \tag{14}$$

当队列的令牌数取得越大，最大帧长 $F$ 也越大，从而 $\Theta$ 也越大. 这样导致端到端延时上界也变大 [7] . 所以，令牌数应该在保证 $O(1)$ 复杂性的条件和带宽比例的情况下尽量取得小.

两个 ATM 数据流 $a$ 和 $b$ , 信元长度 53byte , 信元速率为 1Mcell/s . 信元到达间隔为负指数分布. 流 $a$ 的令牌为 16000 , 而流 $b$ 的令牌为 32000 . 输出链路 155Mb/s . 队列最大长度为 1Mbyte .

图 5 的结果显示了输出数据的突发性. 横坐标表示仿真时间，纵坐标表示是否有数据包离开. 算法改进前数据离开时间相对改进后更集中，说明改进后输出突发性减小. 图 6 显示了两者的延时抖动特性的比较，算法改进后延时抖动减小，性能得到了改善.

## 4 结 论

在当前比较得到关注的数据包调度算法中，DRR 算法以其实现上的简单性得到了广泛的应用. 本文指出 DRR 算法需要在调度前知道数据包长度增加了实现复杂度，同时 DRR 算法的输出突发性较大. 针对这两个问题，我们提出了一种改进算法，算法实现简单. 通过理论分析，证明了在多数情况下改进后的算法公平性得到改善. 仿真表明改进算法能够很好地分配输出链路的带宽. 最后，对令牌的选取对性能的影响进行了研究. 理论分析表明，当令牌数取得较大时，延时性能会受到影响. 仿真结果显示算法改进后，令牌数目选取在突发性和延时抖动上造成的影响减小了.

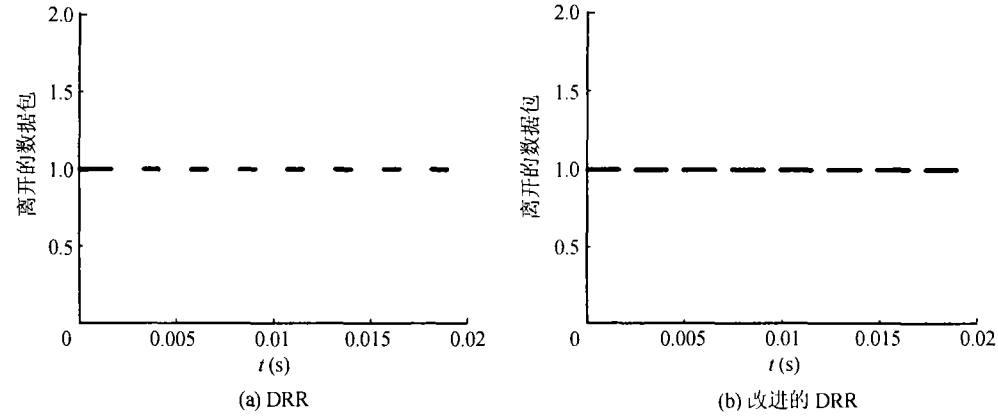


图 5　数据流 $a$ 的突发性

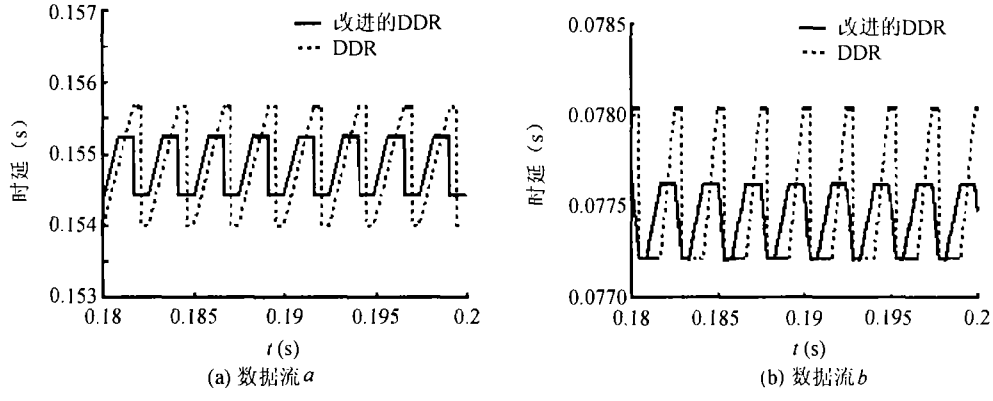


图 6　抖动特性

## 参　考　文　献

[1] L. Zhang, Virtual clock: A new traffic control algorithm for packet switching networks, Proc. of ACM SIGCOMM'90, NY: ACM Press, 1990, 19–29.

[2] A. K. Parekh, R. G. Gallager, A generalized processor sharing approach to flow control in integrated services networks: The single node case, IEEE/ACM Trans. on Networking, 1993, 1(3), 344–357.

[3] S. J. Golestani, A self-clocked fair queueing scheme for broadband applications, Proc. of IEEE INFOCOM'94[C], NJ: IEEE Press, 1994, 636–646.

[4] J. C. R. Bennett, H. Zhang, $WF^2Q$: Worst-case Fair weighted Fair Queueing, Proc. of IEEE INFOCOM'96, NJ: IEEE Press, 1996, 120–128.

[5] M. Katevenis, S. Sidiropoulos, C. Courcoubetis, Weighted round-robin cell multiplexing in a general-purpose ATM switch chip, IEEE J. on Selected Areas in Communications, 1991, 9(8), 1265–1279.

[6] M. Shreedhar, G. Varghese, Efficient fair queueing using deficit round robin, IEEE/ACM Trans. on Networking, 1996, 4(3), 375–385.

[7] D. Stiliadis, A. Varma, Latency-rate servers: A general model for analysis of traffic scheduling algorithms, IEEE/ACM Trans. on Networking, 1998, 6(5), 611–624.

# AN IMPROVEMENT OF DRR PACKET SCHEDULING ALGORITHM

Wu Xiang Kong Hongwei Wang Weizhang Ge Ning Feng Chongxi

(*Department of Electronic Engineering, Tsinghua University, Beijing* 100084, *China*)

**Abstract** In order to allocate the bandwidth more efficiently and simply, this paper analyzes the limitation of the DRR(Deficit Round Robin) algorithm in the packet switch, and proposes an algorithm based on deducting credit and interleaving the output packets. The improved algorithm lowers the scheduling overhead, smoothes the output burst and is very simple for hardware implementation. Analysis indicates the improved algorithm has better fairness over DRR in many cases. Results from analysis show that the scheme is able to maintain bandwidth and is affected by the number of credits less.

**Key words** Schedule, Switch, DRR(Deficit Round Robin), Burst, Fairness

伍 翔: 男, 1974 年生, 博士生, 研究方向为数据通信, 交换, 光通信.
孔红伟: 男, 1974 年生, 博士生, 研究方向为网络拥塞控制, 业务质量保证.
汪卫章: 男, 1976 年生, 博士生, 研究方向为数据通信, 交换.
葛 宁: 男, 1973 年生, 副教授, 研究方向为光通信, 网络同步, 数据通信.
冯重熙: 男, 1930 年生, 教授, 研究方向为光通信.