一种具有低时延的分组公平循环调度

杨　帆　　刘增基　　邱智亮　　刘焕峰
(西安电子科技大学综合业务网国家重点实验室　西安　710071)

摘　要：本文提出了一种新的分组循环调度算法 LFRR(Large weight First Round Robin)。为了具有良好的时延特性和较低的实现复杂度，LFRR 采取了以下方法：(1)在调度表中为流分配时隙时，LFRR 以时隙完全均匀分布为参照，确保分配给一个流的时隙不会过早或过晚地出现在调度表中。(2)LFRR 算法中采用了等权值流合并的技术，把权值大于 1 且权值相等的流合并成一个虚流，以虚流为处理对象，使算法需要处理的对象数目大为减小。(3)当一个时隙适合分配给多个虚流时，LFRR 采用了简单的权值大的虚流优先占用时隙的原则。本文对 LFRR 进行了理论分析和计算机仿真，结果表明 LFRR 算法的时延性能比 WRR(Weighted Round Robin)有了很大提高，同时 LFRR 算法的公平性也有保证。

关键词：分组调度；时延；实现复杂度；公平性
中图分类号：TP393　　**文献标识码**：A　　**文章编号**：1009-5896(2007)04-0785-04

A Fair Round Robin Scheduling Algorithm with Low Latency

Yang Fan　Liu Zeng-ji　Qiu Zhi-liang　Liu Huan-feng
(*The National Key Lab of ISN*, *Xidian University*, *Xi'an* 710071, *China*)

Abstract: A new round robin scheduling algorithm named LFRR(Large weight First Round Robin) is proposed in this paper. In order to achieve good delay property and low implementation complexity, several methods are taken in LFRR. Firstly, when slots are allocated for flows in the schedule table, LFRR refers to the ideal situation where the slots of each flow are uniformly distributed in the schedule table to prevent flows' slots from appearing too early or too late. Secondly, in LFRR system flows which weights are the same and larger than 1 are aggregated into a virtual flow. LFRR deals with virtual flows instead of actual flows. This deceases the flow number that scheduling algorithm deals with. Thirdly, when a slot can be allocated to several virtual flows, flows with larger weight have high priority to posses this slot. Theoretical and simulation results show the delay property of LFRR is much better than that of WRR(Weighted Round Robin) and the fairness of LFRR can be guaranteed.

Key words: Packet scheduling; Delay; Complexity of implementation; Fairness

1　引言

随着网络技术的发展，保证业务的QoS越来越受到人们的重视。由于调度算法对于确保QoS起着非常重要的作用，因此被广泛地应用于无线网络的无线链路以及交换网络的路由器与交换机中。调度算法应该能够确保实时业务的时延要求，应该能够公平地为各个业务流服务，同时也必须简单易行，能在高速的无线链路以及高速大容量的交换机或路由器中实现。调度算法包含有重要的两大类：(1)WFQ[1]类公平调度算法(2)加权循环调度WRR[2](Weighted Round Robin)类调度算法。WFQ类算法有较好的时延特性，能够保障各个业务流的公平性，但需要实时地为各个流计算优先级，从而可扩展性受到影响，在流数目很多的高速大容量系统中，例如T bit级路由器中，实现起来比较困难。WRR类算法虽然性能不如WFQ类算法，但是算法简单，能够快速实现，适合在高速系统中应用。WRR算法中，系统中建立有一循环调度表。在每个时隙，WRR周而复始地依照调度表中的次序为各个流服务，当到达调度表的尾部时，再从调度表的头部开始一轮新的循环。WRR系统中每个流 i 分配有一个权重 w_i，代表在调度表中流 i 占有的时隙数目。

图 1 中给出了 WRR 算法在一个循环周期中时隙分配的例子。其中 $w_1=4$，$w_2=3$，$w_3=3$，$w_4=2$。在 WRR 的循环调度表中，分配给每个流的时隙是紧挨在一起的。设调度表的长度为 T，对于流 i 而言，如果一个分组到达时刚好错过了分配的时隙，那么只有等到下一个循环周期开始后才有可能被发送，时延将可能达到 $T-w_i$。当系统中流的数目较多时，T 往往很大，WRR 算法为一个流引入的服务时延将比较大。

流 1	流 1	流 1	流 1	流 2	流 2	流 2	流 3	流 3	流 3	流 4	流 4
时隙 0	1	2	3	4	5	6	7	8	9	10	11

图 1　WRR 算法中的时隙分配

2006-02-16 收到，2006-08-14 改回
国家“863”计划项目(2002AA103062)和重点实验室基金项目(51473070204DZ0103)资助课题

虽然 WRR 算法的时延特性不理想，但由于它只需构造循环调度表，依照循环调度表的次序为各个流服务，不需要动态地决定各个流的服务次序，因而实现简单。在文献[3–5]中给出了几种改进的 WRR 算法，这些算法的迟延性能较 WRR 有一定的改善，但是为了确定调度表中的每一项，这些算法需要从每个流的参量中寻找一个最小值。设系统中的流数量为 N，构造一个调度表的复杂度将达到 $T\log N$。当系统中复用了大量流时，T 和 N 的数目都很大，这些算法的复杂度很大。我们知道，每当系统中有流创建或结束时，调度表都需要重建，重新安排每个流在调度表中的时隙位置。当网络中的用户变化频繁时，调度表也需要频繁构造。因此如果调度表构造复杂，也将难以在高速大容量的系统中实现。本文提出一种新的循环调度算法 LFRR(Large weight First Round Robin)，该算法改善了 WRR 的迟延性能，算法的公平性有保证，而且不太复杂，能够做到快速实现。

2 LFRR 调度算法

从 WRR 算法的调度表中我们可以看出，造成 WRR 算法迟延性能不佳的主要原因是 WRR 对每个流的时隙分配是连续的。事实上，一个流在调度表中时隙分布越均匀，当该流的分组错过了一个时隙后，等待下一个时隙出现的时间就越短，该流分组的迟延也就越小。在理想情况下，权重为 w_i 的流时隙间的间隔为 T/w_i。本文提出的 LFRR 算法就是尽量使分配给每个流的时隙均匀分布。首先，引入下列符号和标识。

c_i^n 为在调度表中分配给流 i 第 n 时隙的时隙标号，如在图 1 中 $c_3^2=8$。

$b_{i,k}$ 为在调度表第 k 个时隙前已经分配给流 i 的时隙数量，在图 1 中 $b_{3,8}=1$。

$r_i=w_i/T$，r_i 系统为流 i 服务的归一化平均服务速率。显然 $r_i\le 1$。若 $w_i\ge w_j$ 则 $r_i\ge r_j$。

f_i^n 为流 i 第 n 个时隙的时间标签。$f_i^n=n/r_i$，f_i^n 随 n 的增长而递增。

E_k 为可以占有调度表中第 k 个时隙的流集合。流 $i\in E_k$ 当且仅当 $b_{i,k}<w_i$ 且 $b_{i,k}/r_i\le k$。

F_k 为应该优先占有调度表中第 k 个时隙的流集合。流 $i\in F_k$ 当且仅当 $b_{i,k}<w_i$ 且 $(b_{i,k}+1)/r_i\le k$。可以看出，如果 $i\in F_k$，则 $i\in E_k$，即 $F_k\subseteq E_k$。

S 为系统中所有流的集合。S 集合中的流按照权值降序排列。

L 为一个时隙所包含的 bit 数。

流 i 处于积压期(backlogged period)是指系统中存在流 i 的分组。p_i^k 为流 i 的一个积压期中第 k 个离开系统的分组。a_i^k, d_i^k 分别为 p_i^k 到达和离开系统的时刻。

LFRR 调度算法：对于调度表中的某个时隙 k，如果集合 F_k 不为空，则将时隙 k 分配给 F_k 集合中权值最大的流，如果集合 F_k 为空，则将时隙 k 分配给 E_k 中权值最大的流。LFRR 系统按照调度表中的次序输出各流的分组，循环往复。如果在某个时隙调度表中的某个流在系统中没有分组，则跳转到调度表中的下一项，为下一个流服务。

在理想情况下，流 i 的第 n 个时隙应该在调度表中 n/r_i 位置出现。LFRR 算法的基本思想是：流 i 的第 n 个时隙不能早于 $(n-1)/r_i$ 出现，但如果到 n/r_i 还没有出现，则将流 i 置于集合 F_k 中，在调度表中优先为流 i 安排时隙。

LFRR 算法在构造调度表时，首先要对各流的权值进行排队，得到集合 S。根据 S，可以很容易得到调度表第 k 个时隙的 E_k 或 F_k 中权值最大的流。例如可以按照 S 中流的次序判断流是否属于 E_k 或 F_k，如果一个流属于 F_k，就不需要判断它后面的流是否属于 F_k。这一规则对于 E_k 也适用。而集合 S 在每次有流创建或结束时，也只需要对 S 中个别元素进行插入或删除操作，也很容易得到。因此 LFRR 算法简单易行，能够快速实现。

图 2 中给出了各流的权值与图 1 相同的情况下，LFRR 算法中调度表中时隙的分配。可以看出，在 LFRR 算法中，分配给各个流的时隙均匀地散布在调度表中。

流 1	流 2	流 3	流 1	流 2	流 3	流 4	流 1	流 2	流 3	流 1	流 4
时隙 0	1	2	3	4	5	6	7	8	9	10	11

图 2 LFRR 算法中的时隙分配

3 LFRR 算法的时延及公平性分析

引理 1 对于调度表中的任一个时隙 k，E_k 集合都不为空集。

证明 假设存在某一时隙 n，集合 E_n 为空。$\therefore \forall$ 流 i，$b_{i,n}/r_i>n$，$b_{i,n}>(w_i/T)\cdot n$

$$\therefore \quad \sum_{i\in S} b_{i,n} > \sum_{i\in S}(w_i/T)\,n \tag{1}$$

$\because \sum_{i\in S} b_{i,n}=n$，而 $\sum_{i\in S} w_i=T$，式(1)两端出现矛盾。所以假设不成立，引理 1 得证。

在 LFRR 算法中，时隙 k 要分配给流 i，则 $i\in F_k$ 或 $i\in E_k$。而 $F_k\subseteq E_k$，因此 i 一定属于 E_k。从引理 1 可以看出，调度表中的每一个时隙都分配给了各个流，不存在空闲时隙。

引理 2 对于流 i，$c_i^{w_i}\le f_i^{w_i}$。

证明 由于调度表中不存在空闲时隙，而 $c_i^{w_i}$ 最晚出现在 $T-1$ 时隙(因为调度表的起始时隙标号为 0)。$f_i^{w_i}=w_i/r_i=w_i\cdot T/w_i=T$，得证。

引理 3 对于流 i，在 LFRR 算法中，如果 $c_i^n\ge f_i^n+1/r_i$，则 $\forall j, w_j\ge w_i$，$b_{j,c_i^n}\le r_j\cdot c_i^n$。其中 $n<w_i$。

证明 分两种情况，(1)调度表中从 $\lfloor f_i^n\rfloor$($\lfloor\ \rfloor$ 表示下取整)时隙到 c_i^n 时隙系统都没有给流 j 分配时隙。设 γ 时隙是 $\lfloor f_i^n\rfloor$ 时隙前最后一个分配给流 j 的时隙(如果 $\lfloor f_i^n\rfloor$ 时隙前系统没有为流 j 分配时隙，则记 $\gamma=0$)。γ 时隙分配给流 j，则 $j\in E_\gamma$，即 $b_{j,\gamma}/r_j\le\gamma$。由于 $b_{j,c_i^n}=b_{j,\gamma}+1$，而 $c_i^n\ge f_i^n+1/r_i\ge f_i^n+1/r_j\ge\gamma+1/r_j$。因此引理成立。

(2)调度表中从$\lfloor f_i^n \rfloor$时隙到c_i^n时隙系统给流 j 分配了时隙。设 β 为 c_i^n 时隙前系统分配给流 j 的最后一个时隙。如果 $\beta = \lfloor f_i^n \rfloor$，仿照(1)可以证明引理成立。如果 $\beta > \lfloor f_i^n \rfloor$，因为从 $\lceil f_i^n \rceil$ ($\lceil\ \rceil$ 表示上取整)时隙到 c_i^n 时隙间的任意时隙 k，$b_{i,k} \le n-1 = r_i f_i^n - 1 \le r_i \lceil f_i^n \rceil - 1 \le r_i k - 1$，因此 $i \in F_k$。由 LFRR 算法，$j \in F_\beta$ 。因此 $b_{j,\beta} + 1 = b_{j,c_i^n} \le r_j \cdot \beta \le r_j \cdot c_i^n$。引理成立。

定理 1　$\forall$ 流 $i, w_i > 1$，

$$(n-1)/r_i \le c_i^n \le f_i^n + 1/r_m \tag{2}$$

其中 $n < w_i, r_m = \min\{r_k \mid w_k > 1\}$。限于篇幅，定理 1 的证明略。

定理 1 中给出了调度表中分配给流 i 第 n 个时隙的时隙号 c_i^n 与 r_i 及 n 的关系。对于某些权值大的流，c_i^n 的出现时间要比式(2)右端的界限靠前。例如，如果 $\forall j (j \in S, j \ne i\}$，$w_j < w_i$，则由于流 i 的权值最大，一旦对于某个时隙 k，$i \in F_k$，时隙 k 就分配给流 i，因此 $c_i^n \le f_i^n + 1$。对于这类情况，我们用定理 2 给出 c_i^n 的上界。令 $B_i = \{j \mid w_j \ge w_i$且$j \ne i\}$

定理 2　如果 $\sum_{w_j \ge w_i} (r_j / r_i + 1) < 1/r_i + 1$，则 $c_i^n \le f_i^n + 1/r_i$

证明　采用反证法。假设 $c_i^n > f_i^n + 1/r_i$。采用与引理 3 相同的方法可证：$\forall j \in B_i$，$b_{j,\lfloor f_i^n + 1/r_i \rfloor} \le r_j (f_i^n + 1/r_i)$。设 c_l 为 $\lfloor f_i^n + 1/r_i \rfloor$ 时隙前最后一个分配给权值小于 w_i 的流的时隙(如果没有这样的时隙，则令 $c_l = -1$)。因为在 $\lceil f_i^n \rceil$ 时隙到 c_i^n 时隙间的任意时隙 k，$i \in F_k$，因此 $c_l \le f_i^n$，否则 c_l 时隙将分配给流 i。在 c_l 时隙，$\forall j \in B_i$，$j \notin F_{c_l}$，因此 $b_{j,c_l} + 1 > r_j \cdot c_l$，该式对流 i 同样也成立。因此从 $c_l + 1$ 到 $\lfloor f_i^n + 1/r_i \rfloor - 1$ 时隙之间，流 j 分配到的时隙数目为 $b_{j,\lfloor f_i^n + 1/r_i \rfloor} - b_{j,c_l} \le r_j (f_i^n + 1/r_i - c_l) + 1$。对于流 i，因为 $b_{i,\lfloor f_i^n + 1/r_i \rfloor} \le n - 1 = r_i f_i^n - 1$，因此在 $c_l + 1$ 到 $\lfloor f_i^n + 1/r_i \rfloor - 1$ 时隙之间，流 i 分配到的时隙数目为 $b_{i,\lfloor f_i^n + 1/r_i \rfloor} - b_{i,c_l} \le r_i (f_i^n - c_l)$。所以

$$\begin{aligned}
&\lfloor f_i^n + 1/r_i \rfloor - 1 - c_l = \sum_{w_j \ge w_i} (b_{j,\lfloor f_i^n + 1/r_i \rfloor} - b_{j,c_l}) \\
&= \sum_{j \in B_i} (b_{j,\lfloor f_i^n + 1/r_i \rfloor} - b_{j,c_l}) + b_{i,\lfloor f_i^n + 1/r_i \rfloor} - b_{i,c_l} \\
&\le \sum_{w_j \ge w_i} [r_j (f_i^n + 1/r_i - c_l) + 1] - 2 \\
&\le f_i^n - c_l + \sum_{w_j \ge w_i} (r_j / r_i + 1) - 2 \\
&< f_i^n - c_l - 1 + 1/r_i \le \lfloor f_i^n + 1/r_i \rfloor - 1 - c_l
\end{aligned} \tag{3}$$

式(3)两端出现了矛盾。因此假设不成立。　证毕

如果某个时隙该由某一流发送分组，而系统中却没有该流的分组，则称该时隙为轮空时隙。令 $e(t_1, t_2)$ 为 $[t_1, t_2]$ 期间系统中轮空时隙的个数。令 $D_i = 1/r_m$，如果 r_i 满足定理 2 的条件，则令 $D_i = 1/r_i$。即 $D_i = 1/r_m$或$1/r_i$。令 C 为系统输出分组的速率，令 $R_i = r_i C = w_i \cdot C / T$，$R_i$ 表示系统为流 i 提供的平均服务速率，设流 i 业务的平均到达率为 ρ_i，为了防止系统中出现拥塞，要求 $R_i \ge \rho_i$。采用文献[3]中的分析方法，可得到引理 4、引理 5、定理 3 3 条结论：

引理 4　$d_i^n - a_i^1 \le [(n+1)/r_i + D_i - e(a_i^1, d_i^n)] \cdot L/C$

$$d_i^n - d_i^l \le [(n+1-l)/r_i + D_i - e(d_i^l, d_i^n)] \cdot L/C$$

$$d_i^n - d_i^l \ge [(n-1-l)/r_i - D_i - e(d_i^l, d_i^n)] \cdot L/C$$

引理 5　LFRR 算法属于 latency rate server，算法的固有时延(latency)为 $2L/\rho_i + D_i L/C$

定理 3　如果在 $[t_1, t_2]$ 期间，流 i，流 j 都处于积压期，则 LFRR 算法的公平性由下式保证：$\left| W_i(t_1, t_2)/R_i - W_j(t_1, t_2)/R_j \right| \le (D_i/C + D_j/C)L + 2(L/R_i + L/R_j)$。其中 $W_i(t_1, t_2)$ 表示在 $[t_1, t_2]$ 期间流 i 获得的服务量。

定理 3 给出了 LFRR 算法的公平性保证。

定理 4　如果流 i 进入系统的业务量满足 (σ_i, ρ_i) 的漏桶约束，则 LFRR 系统中流 i 分组时延满足：

$$d_i^n - a_i^n \le \sigma_i/\rho_i + (n+1)L/R_i - (n-1)L/\rho_i + D_i L/C \tag{4}$$

如果 $R_i = \rho_i$，式(4)变为

$$d_i^n - a_i^n \le \sigma_i/\rho_i + 2L/\rho_i + D_i L/C \tag{5}$$

证明　由 a_i^k 的定义可知，$[a_i^1, a_i^n]$ 期间，流 i 进入系统的业务量为 $n-1$ 个分组

$\therefore (n-1)L \le \sigma_i + \rho_i (a_i^n - a_i^1)$，$\therefore a_i^n \ge a_i^1 + [(n-1)L - \sigma_i]/\rho_i$。由引理 4 及 R_i 的定义可得 $d_i^n - a_i^1 \le (n+1)L/R_i + D_i L/C$，

$\therefore d_i^n - a_i^n \le \sigma_i/\rho_i + (n+1)L/R_i - (n-1)L/\rho_i + D_i L/C$。当 $R_i = \rho_i$ 时，式(5)显然成立。　证毕

从定理 4 可以看出，要减小分组的时延，就要减小 D_i，即增大 r_m。为了做到这一点，可以对 LFRR 算法作出如下修正：将系统中权值大于 1 且权值相等的流合并成一个虚流，虚流的权值等于这些流的权值相加，分配给虚流的时隙再依次轮流分配给这些流。这样不仅可以减小分组的时延，而且由于虚流的数目可能远小于实际流的数目，因此算法的运算量也将大为减小。记 $\overline{r}_i$ 为虚流 $\overline{i}$ 的服务速率，$\overline{w}_i$ 为虚流 $\overline{i}$ 的权值。

定理 5　在 LFRR 算法中，设虚流 $\overline{i}$ 由流 $i_1, i_2, \cdots, i_k$ 合并而成，$\forall$ 流 $i_j (1 \le j \le k)$，$(n-1)/r_{i_j} \le c_{i_j}^n \le f_{i_j}^n + 1/\overline{r}_m$　其中 $n < w_{i_j}, \overline{r}_m = \min\{\overline{r}_k \mid \overline{w}_k > 1\}$。

证明　由定理 1 可得：$(n-1)/\overline{r}_i \le c_{\overline{i}}^n \le f_{\overline{i}}^n + 1/\overline{r}_m$。分配给虚流 $\overline{i}$ 的时隙再以固定的次序轮流分配给 $i_1, i_2, \cdots, i_k$。分配给流 i_j 的第 n 个时隙，最早出现在虚流 $\overline{i}$ 的第 $(n-1)k+1$ 个时隙上，最晚出现在虚流 $\overline{i}$ 的第 nk 个时隙上，即 $c_{\overline{i}}^{(n-1)k+1} \le c_{i_j}^n \le c_{\overline{i}}^{nk}$，因为 $\overline{r}_i = k r_{i_j}$，所以 $c_{i_j}^n \ge c_{\overline{i}}^{(n-1)k+1} \ge [(n-1) \cdot k + 1 - 1]/\overline{r}_i = (n-1)/r_{i_j}$，$c_{i_j}^n \le c_i^{nk} \le nk/\overline{r}_i + 1/\overline{r}_m = n/r_{i_j} + 1/\overline{r}_m$，得证。

定理 6　在 LFRR 算法中，如果虚流 $\overline{i}$ 由流 $i_1, i_2, \cdots, i_k$ 合并而成且 $\sum_{\overline{w}_j \ge \overline{w}_i} (\overline{r}_j / \overline{r}_i + 1) < 1/\overline{r}_i + 1$，则 $\forall$ 流 $i_j (1 \le j \le k)$，

$c_{i_j}^n \le f_{i_j}^n + 1/\overline{r}_i$。

定理 6 证明与定理 5 类似，这里就不再重复。从定理 5 及定理 6 可以看出，如果流 i 属于虚流 $\overline{i}$，则 $D_i = 1/\overline{r}_i$或$1/\overline{r}_m$。有关 LFRR 算法公平性和时延特性的定理 3 和定理 4 均不需要改动。

文献[6]中给出 WRR 算法中进入系统的业务量满足 (σ_i, ρ_i) 的漏桶约束，权值为 w_i 的流的分组时延界限为 $(T - w_i + 1)L/C + \sigma_i/\rho_i$。从文献[6]中 latency rate server 的分析中可知，上式及式(5)中 σ_i/ρ_i 是由流业务的突发性造成的(凡是符合漏桶约束的流在属于 latency rate server 的调度算法中在时延上限里都有这一项，它表示以速率 ρ_i 服务完该流的一个最大突发所需要的时间)，而 $(T - w_i + 1)L/C$ 是由调度算法特性决定的固有时延，不同的调度算法固有时延不同。因此，我们比较 LFRR 和 WRR 的时延特性时主要比较 LFRR 中的 $2L/\rho_i + D_iL/C$ 和 WRR 中的 $(T - w_i + 1)L/C$ 这两个由调度算法决定的固有时延。令 $\theta_{\mathrm{LFRR}}^i = 2L/\rho_i + D_iL/C$，$\theta_{\mathrm{WRR}} = (T - w_i + 1)L/C$。当 T 较大时，$\theta_{\mathrm{WRR}} \approx T \cdot L/C$。可以看出，在时隙连续分配的 WRR 算法中，不论流的权值大小，所有流的分组的都具有相同的固有时延，而且该固有时延与调度表的大小成正比。在 LFRR 算法中，令 $\overline{w}_m$ 为权值合并后，大于 1 的最小权值。由 D_i 的定义可知，T 至少是 D_i 的 $\overline{w}_m$ 倍。如果 $\overline{w}_i$ 满足定理 6，那么 D_i 更小，T 是 D_i 的 $\overline{w}_i$ 倍。$\overline{w}_m$ 或 $\overline{w}_i$ 越大，T 与 D_i 的差距越大，θ_{LFRR}^i 与 θ_{WRR} 的差距就越大。举个简单的例子，假设系统中采用等权值合并后，权值最小的虚流的权值是 40，则 $\overline{r}_m = T/40$，假设 $\rho_i = C/80$，那么 θ_{WRR} 将是 θ_{LFRR}^i 的 20 倍；如果 $\overline{r}_m = T/120$，那么 θ_{WRR} 将是 θ_{LFRR}^i 的 30 倍。可以看出 LFRR 算法的时延特性要比采用时隙连续分配的 WRR 好出很多。当系统中存在大量的流时，采用等权值流合并后，通常情况下 D_i 会远小于 T，因而 LFRR 算法的时延特性比 WRR 算法有较大改善。

4 计算机仿真

本文使用 OPNET 对 LFRR 算法的性能进行了仿真。在仿真中把进入系统的业务分为 12 类，每类业务又包含 5 个流，各个流的业务都服从一定的漏桶约束。第 1 类到第 6 类业务为具有 on-off 特性的突发业务，第 7 到第 9 类为到达服从泊松分布的业务，第 10 到第 11 类业务为恒定比特率业务。各类业务的权值依次为 72，68，65，59，53，48，42，36，29，20，16，11。调度表的长度为 2595。在确定每个流 i 的权值 w_i 时，我们使该流平均服务速率 R_i 等于该流业务的平均到达速率，也就是漏桶中令牌的平均到达速率 ρ_i。系统的服务能力为 $C = 10$ Mbit/s，系统中分组的长度为 53byte。第 1 到第 6 类业务的突发强度 σ_i 为 15 个分组，第 7 到第 9 类业务的突发强度为 2 个分组，第 10 到第 11 类业务的突发强度为 1 个分组。图 3、图 4、图 5 中分别给出了第 1、第 7、第 10 类业务的分组在 WRR 算法下和 LFRR 算法下的时延，可以看出 LFRR 算法的时延特性要比 WRR 好很多。

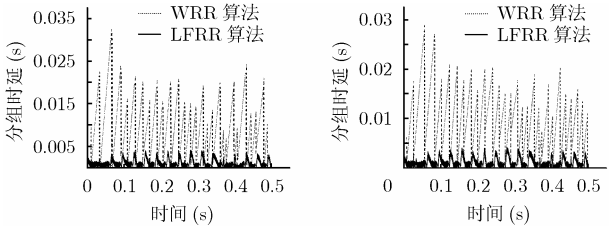

图 3 第 1 类业务在 WRR 与 LFRR 算法下的时延

图 4 第 7 类业务在 WRR 与 LFRR 算法下的时延

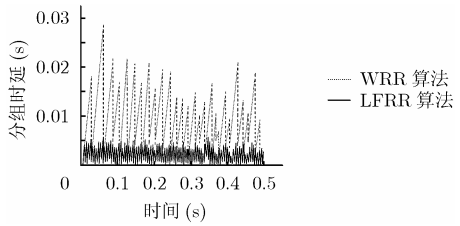

图 5 第 10 类业务在 WRR 与 LFRR 算法下的时延

5 结束语

本文提出了一种新的循环调度算法 LFRR，该算法的时延性能比 WRR 算法有较大提高，而算法的复杂性不高，可以在高速大容量的系统中实现。另外，不同流的公平性在 LFRR 算法中也能被保证。因此 LFRR 算法是一种能够确保网络中业务 QoS 的实用调度算法。

参 考 文 献

[1] Parekh A and Gallager R. A generalized processor sharing approach to flow control: The single node case. *IEEE/ACM Trans. on Networking*, 1993, 1(3): 344–357.

[2] Katevenis M and Sidiropoulos S. Weighted round-robin cell multiplexing in a general purpose ATM switch chip. *IEEE Journal on Selected Areas on Communication*, 1991, 9(8): 1265–1279.

[3] Matsufuru N and Aibara R. Efficient fair queueing for ATM networks using uniform round robin. In Proc. IEEE INFOCOM 1999, New York: 389–397.

[4] Saha D and Mukherjee S. Carry-over round robin : A simple cell scheduling mechanism for ATM networks. *IEEE/ACM Trans. on Networking*, 1998, 6(6): 779–795.

[5] Onur, Yukio, and Teruaki. Urgency-based round robin: A new scheduling discipline for packet switching networks. in Proc. IEEE INFOCOM 1998, San Francisco: 1179–1184.

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

杨 帆： 男，1973 年生，讲师，博士，研究方向为宽带交换网络.
刘增基： 男，1937 年生，教授，研究方向为宽带通信网.
邱智亮： 男，1965 年生，教授，研究方向为宽带通信网.
刘焕峰： 男，1972 年生，讲师，研究方向为宽带通信网.