LL-DRR：一种有效的用于高速分组网络的调度算法[1]

涂晓东 李乐民

(电子科技大学光纤通信国家重点实验室 成都 610054)

摘 要 该文在亏空轮循 (Deficit Round Robin, DRR) 算法的基础上提出了一种新的适用于变长分组的调度算法——低时延亏空轮循 (Low Latency Deficit Round Robin, LL-DRR)。仿真和理论分析表明，在时延性能上 LL-DRR 比 DRR 有显著的改善，并具有连接的最大时延与连接数无关的特性，可以支持实时业务。LL-DRR 继承了 DRR 在平均吞吐率上的公平性。LL-DRR 易于实现且适用于高速网络。

关键词 分组公平排队，亏空轮循，低时延亏空轮循

中图号 TN919.3

1 前 言

各种多媒体应用如 IP Phone、视频会议等要求分组网络设备 (ATM 交换机、IP 路由器) 能够为业务提供服务质量 (QoS: Quality of Service)，在分组网络设备中采用适当的排队算法是提供服务质量的关键。从 20 世纪 80 年代末以来，国际上对分组公平排队算法 (Packet Fair Queuing, PFQ) 进行了大量的研究 [1−8]。在本文中，调度和排队是同义词。一个公平调度算法应具有如下的性质：(1) 隔离性。即能够确保连接 (session 或 connection，本文的术语“连接”并不特指 ATM 中的 VC，而是指排队系统中的一个排队队列) 的预约带宽，即使系统中有违约连接。违约是指连接的实际流量超过了其预定的流量。(2) 时延可限定性。即当连接的流量符合漏桶限制时，能够保证连接的分组最大时延只与连接本身的参数有关。(3) 较低的实现复杂性。即可用较低的代价在设备中实现公平调度。(4) 易扩展性。即当连接数增加或链路速率增加时，调度算法仍能有效工作。

以往文献中提出的公平调度算法，或是实现比较复杂，例如 WFQ[3] 需要做乘法运算以求得分组时标并且需对多个分组的时标进行比较以确定下一个发送的分组。或是实现简单但性能较差，例如亏空轮循 (DRR) [2] 实现简单但时延性能较差。本文在 DRR 基础上提出了低时延亏空轮循 (Low Latency Deficit Round Robin, LL-DRR) 算法，既显著地改善了时延性能又容易实现。本文第 2 节概述了路由器中调度分组的公平算法。第 3 节详细描述了 LL-DRR 算法。第 4 节通过仿真和理论分析比较了 LL-DRR 和 DRR。第 5 节讨论了有关 LL-DRR 实现的几个问题。第 6 节对全文进行总结。

2 路由器中分组公平排队算法概述

图 1 示出输出缓存型路由器的输出排队模型，从各个输入端口进入的分组只在输出端口排队等待传输。虽然本文讨论的是输出缓存型，但本文的调度算法也可以适用于输入 - 输出缓存结构。例如用于输入端，在各个相互竞争的分组流之间公平调度。

由 A.Demers 等提出的 WFQ(Weighted Fair Queuing) 算法 [3] 虽然具有很好的隔离性和时延可限定性，但其实现复杂性很高，复杂性主要来源于两个方面：(1) 系统虚拟时间 $V(t)$ 的计算。WFQ 算法需要跟踪相应的流体化排队系统 GPS (General Processor Sharing) 的状态 [4]，

[1] 2000-08-28 收到，2001-01-11 定稿

信息产业部电子科学技术研究院预研项目资金；深圳华为科技基金；国家自然科学基金 (No.69882003)；博士点专项科研基金 (No.98061409) 资助

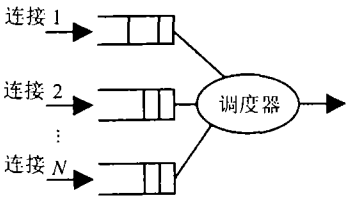

图 1 N 个连接在输出端排队

这种操作的复杂性为 $O(N)$, 即与系统中的连接数成正比。(2) 对时标的计算以及比较操作，最坏情况下在一个分组的传输时间内需要对 N 个时标进行比较，复杂性为 $O(\log N)$，当分组变长时，计算时标将做乘法运算，这将显著增加实现的复杂性。在 WFQ 以后提出的各种算法都试图从以上两个方面降低复杂性。Virtual clock[1] 、SCFQ(Self-Clocked Fair Queuing)[1] 等都降低了 $V(t)$ 的计算复杂性。文献 [5] 从减少所需比较的时标个数方面做了研究。

虽然以上这些方法改善了 PFQ 算法的可实现性，但有一点它们是共同的，即在一个分组的传输时间内都需要做时标的比较运算 (这类算法可被称为基于分组时标比较的公平调度算法)，计算时标还需要做乘法运算。这使得实现这些算法的硬件比较复杂，设备成本较高。当网络链路速率很高时，这类算法不易实现。

Round robin(轮循) 类型的调度算法，按一定顺序轮循发送各个连接的分组，不需进行有关时标的操作，实现简单，也试图达到公平的目的。

Nagle[6] 提出的 round robin 算法能在一定程度上体现公平性，避免违约连接过度地占用网络的带宽资源。但有两个问题没有解决。(1) 不能体现不同连接的不同的带宽需求。(2) 当不同连接使用不同长度的分组时，会产生不公平。Katevenis 等人 [7] 提出的 Weighted Round-Robin(WRR) 可以用于 ATM 交换机，根据连接的权重成比例地分配带宽，解决了第 (1) 个问题，却没有解决第 (2) 个问题。

Shreedhar 等人提出的 DRR[2] 解决了这两个问题。DRR 的基本思路如下：为每一个连接维护一个 deficit(亏空) 计数器和 quantum 寄存器 (分配给连接的服务量定额)，quantum 寄存器表示在对连接的一次服务中的服务量定额。服务量用服务的分组长度表示。所谓对连接的一次服务是指一次连续不断发送一个连接的分组的过程。Deficit 表示在前一次服务中，原本应该发送的分组总长度与实际发送的分组总长度的差别，deficit 计数器为非负整数，表示上一次未用完的服务量。当下一次又轮到连接发送分组时，将 quantum 加上 deficit 的值作为这一次允许对连接服务的分组总长度。通过给不同的连接设置不同的 quantum 值，可以支持不同的预约带宽。应用 deficit 计数器可以避免由不同的连接使用不同长度的分组引起的不公平，可以有效地保证各个连接在平均吞吐率上的公平。但是 DRR 有两个问题：(1) 预约带宽越大的连接其 quantum 越大，导致经过调度器输出后连接的突发性越大，不利于连接通过后续的节点。(2) 当连接的流量符合漏桶限制时，连接的分组的最大时延不仅取决于连接本身的流量特性，还与连接数等其它参数有关。DRR 的这些缺点使得它不能很好地支持实时业务。

针对 DRR 的不足，本文提出了一种新的适用于变长分组的公平调度算法 LL-DRR 。与 DRR 一样，LL-DRR 为每一个连接维护一个 deficit 计数器和 quantum 寄存器，同样也试图在一轮 (round) 内发送总长度为 quantum 的分组。与 DRR 不一样的地方在于，DRR 是试图一次性地发送总长度为 quantum 的分组，而 LL-DRR 是试图分几次间隔均匀地发送总长度为 quantum 的分组，减小了突发性并改善了时延性能。LL-DRR 根据一个预先安排好的调度表来发送各个连接的分组。调度表的安排借鉴了其它 PFQ 算法利用时标排序的方法，在调度表中各连接间隔均匀。

3 低时延亏空轮循 (LL-DRR) 算法

定义如下的一些符号和术语。

连接 i：表示第 i 个连接，$i = 0, 1, \cdots, N-1$。

N：连接个数。

C：调度器输出端口速率。

LMAX：最大分组长度；LMIN：最小分组长度。

一个分组到达的时刻：当一个分组的最后一个比特进入系统。

一个分组离开的时刻：当一个分组的最后一个比特离开系统。

调度表：一个在新的连接建立或旧连接拆除时进行更新的表。调度器从表头读到表尾，根据表的内容，对各个连接进行服务，例如表 1 所示一个长度为 6，连接数为 3 的调度表。

表 1　一个调度表

调度表中位置的编号：	0	1	2	3	4	5
连接号：	0	1	0	1	0	2

一轮 (one round)：调度器根据调度表轮循一周，称为一轮。例如表 1，在一轮中连接 0 被服务 3 次，连接 2 被服务 1 次。

SQ：Service Quantum, 服务粒度，表示调度器对一个连接服务一次试图服务的分组总长度，以字节为单位，如果分组长度固定，例如为 53byte 的 ATM 信元，则 SQ 可取值为 53byte。

n_i：连接 i 在一轮中应该被服务的次数。例如表 1，$n_0 = 3, n_1 = 2$

Quantum_i：对连接的一轮服务量定额，连接 i 在一轮中应该被服务完的分组长度。显然有 $\text{Quantum}_i = \text{SQ} \times n_i$。

Deficit_i：连接 i 的亏空计数器，表示对连接 i 服务一次后剩余的服务量。初始值为 0。

r_i：连接 i 的预约带宽，有 $r_i = (n_i/F) \times C$；F：调度表长度，如表 1，$F = 6$。有 $F = \sum_{j=0}^{N-1} n_j$。

下面分调度表建立和发送分组两个方面来描述 LL-DRR.

(1) 调度表建立　当连接建立或拆除时，需要重新建立调度表。建立调度表的算法伪码如下：

$S_i = 0$, $F_i = F/n_i$, k, m: 整数变量

for$(T = 0; T < F; T++)$

{

A 是所有开始时标小于或等于 T 的连接组成的集合

$k \in A$ 并且 $F_k \le F_m, \forall m \in A$

将编号为 T 的位置分配给连接 k

$S_k = S_k + F/n_k$, $F_k = F_k + F/n_k$

}

说明：为每一个连接 i 设置一个开始时标 S_i 和一个结束时标 F_i, S_i 的初值为 0, F_i 的初值为 F/n_i。给调度表的每一个位置按顺序编号为 $0, 1, \cdots, F-1$。设 T 是调度表里一个位置的编号，从所有开始时标小于或等于 T 的连接中选出结束时标最小的连接，将编号为 T 的位置分配给这个连接 (如果有几个连接符合条件，从中任选一个)，这个连接的开始时标和结束时标分别增加 F/n_i。接着分配下一个位置。表 1 就是按此算法建立的。总之，给出调度表长度和各个连接的 n_i 参数 (等价于给出预约带宽)，就可建立调度表。

(2) 发送分组 伪码如下：

k, m, temp: 整数变量， $L(k)$: 连接 k 队列的头一个分组的长度

Deficit_k: 连接 k 的 Deficit 寄存器

读取调度表中的一项，将其值赋予 k

if 连接 k 不为空，

then {

temp=Deficit_k+SQ;

while(temp> $L(k)$and 连接 k 不为空)

{

temp=temp−$L(k)$;

发送连接 k 头一个分组;

}

if(temp< $L(k)$) then Deficit_k=temp

if(连接 k 为空) then Deficit_k=0

读取调度表中的下一项;

}

else 读取调度表中的下一项;

说明：调度器从调度表中读取一项，根据其内容访问相应的连接的队列 (若连接的队列为空，则略过该连接读取调度表中下一项)，将该连接的 Deficit 寄存器和服务粒度 SQ 相加，即 temp=Deficit+SQ, temp 作为这一次对该连接服务的分组长度的上限 (注意：一次服务可能连续服务几个分组)。每服务完该连接的一个分组，就从 temp 中减去该分组的长度，当 temp 小于下一个该连接的分组长度或者连接为空，停止对该连接的这次服务。若 temp 小于下一个该连接的分组长度，则 Deficit=temp; 若该连接为空， Deficit=0 。调度器从调度表中读取下一项。当调度器读到调度表末尾时，一轮结束。下一轮又从调度表头开始。虽然我们分成两个阶段来描述 LL-DRR，实际上这两个阶段的操作可以同时进行。在系统中维护两个调度表，一个新调度表和一个旧调度表。当连接建立或拆除时，建立新的调度表，旧调度表供系统继续使用，当新调度表建立后，新表和旧表切换。

4 LL-DRR 与 DRR 的比较

4.1 一个例子

假设调度器输出端口速率为 C=2Mbit/s, 连接数为 3,SQ=500byte, Quamtum_0=1500byte, Quamtum_1=1000byte, Quamtum_2=500byte 。在一轮的开始时刻，连接 0 有 6 个分组在等待，头一个分组长度为 200byte，连接 1 和连接 2 有若干分组在等待，长度都为 500byte, 如表 2。LL-DRR 的调度表如表 1 。

表 2 连接的分组排列情况

分组	第 1 个	第 2 个	第 3 个	第 4 个	第 5 个	第 6 个
连接 0	200	200	300	400	400	500
连接 1	500	500	500	500		
连接 2	500	500	500	500		

采用 LL-DRR 和 DRR，连接 0 的分组经链路输出后的时间 (以毫秒为单位) 如图 2 。可见，LL-DRR 输出的分组间隔比 DRR 情况下均匀 (突发性小)。DRR 情况下，对连接 0 服务一次，就发送总长度为 1500byte 的 5 个分组。在 LL-DRR 情况下，是服务 3 次才发送完这 5 个分组的，每次服务有一定的时间间隔。当连接的分组到达较为均匀时 (符合预约带宽)，可以预料 LL-DRR 情况下分组的时延特性比 DRR 情况下要好。

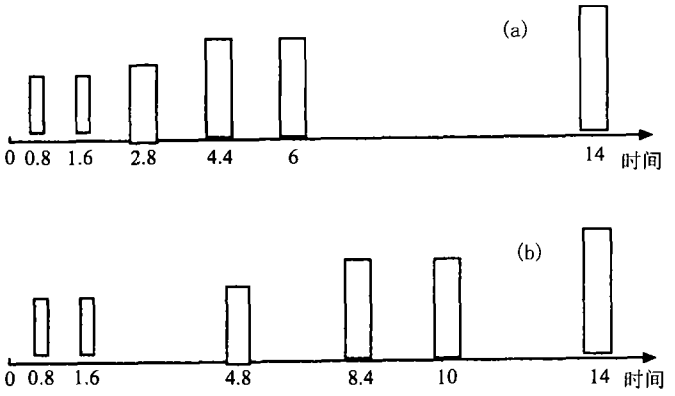

图 2　连接的输出分组时间间隔的比较 (a) DRR; (b) LL-DRR

4.2 仿真实验

通过计算机仿真，比较了 LL-DRR 和 DRR 在时延、吞吐率以及所需缓存方面的性能。

实验 1 链路速率为 2Mbit/s，SQ=LMAX=1518byte, LMIN=64byte, 共有 10 个连接，连接 i 的 $\text{Quamtum}_i = (i+1)\times\text{SQ}$, 除连接 9 外，其余连接的分组到达速率为其预约速率，即 $r_i = [(i+1)/55] \times 2\text{Mbit/s}$，其流量可用参数为 (LMAX×8, r_i) 的漏桶限制 (LMAX×8 表示突发度，以比特为单位)。连接 9 为恶意连接，到达速率为 2Mbit/s，分组长度在 [LMIN,LMAX] 内均匀随机分布，仿真时间为 30s。表 3 表 4 为统计结果。发送分组总长度是指连接在仿真时间内经过链路输出的分组长度的总和，吞吐率等于发送分组总长度除以仿真时间。

可以看出，LL-DRR 和 DRR 在连接发送分组总长度和吞吐率的指标上相当接近，并且各个连接发送的分组总长度和吞吐率几乎与它们的一轮服务量定额成正比，这表明在吞吐率上，LL-DRR 和 DRR 都具有良好的公平性。在最大时延和平均时延的指标上，对于连接 0，LL-DRR 和 DRR 相差不大，这是由于 n_0=1, 采用 LL-DRR 和 DRR 都是试图服务一次将总长度为 Quantum_0 的分组服务完。对于连接 1-8，LL-DRR 的时延明显比 DRR 小，这是由于 DRR 试图服务一次将总长度为 Quantum_i 的分组服务完，而 LL-DRR 试图均匀服务多次将总长度为 Quantum_i 的分组服务完。对于连接 9，LL-DRR 和 DRR 的时延都很大，这是由于连接 9 是恶意用户，其到达速率超过了其预约带宽。因此，对于流量守规的连接，LL-DRR 的时延性能比 DRR 好。

表 3　LL-DRR 的统计结果

连接号	0	1	2	3	4	5	6	7	8	9
最大时延 (ms)	643	323	231	183	145	143	106	91	78	24488
平均时延 (ms)	335	159	116	96	76	64	51	44	38	12144
发送分组总长度 (byte)	134442	271727	407806	544894	679802	817215	953334	1089773	1225194	1376144
吞吐率 (kbit/s)	35.85	72.46	108.75	145.31	181.28	217.92	254.22	290.61	326.72	366.97

表 4　DRR 的统计结果

连接号	0	1	2	3	4	5	6	7	8	9
最大时延 (ms)	613	471	398	389	361	346	344	319	317	24240
平均时延 (ms)	307	228	200	187	174	166	164	156	154	11856
发送分组总长度 (byte)	135414	271054	405976	542352	679802	815971	953334	1082481	1217249	1395923
吞吐率 (kbit/s)	36.11	72.28	108.26	144.63	181.28	217.59	254.22	288.66	324.60	372.25

实验 2 为了验证 LL-DRR 和 DRR 的性能是否与分组长度有关，我们将实验 1 中的连接 8 的分组长度设定为 500byte 和 1000byte, 其它条件不变，统计连接 8 的数据，得到表 5。可见，不论 LL-DRR 还是 DRR, 分组长度对吞吐率没有太大的影响。短分组比长分组的时延有略微的

减少。平均时延与平均分组长度有关，平均分组长度大，则平均时延大。例如采用均匀随机分布的分组长度 (平均长度为 791byte) 的平均时延 (见实验 1) 介于分组长度为 1000byte 和分组长度为 500byte 的平均时延之间。

表 5(a) 分组长度对性能的影响 LL-DRR

	最大时延 (ms)	平均时延 (ms)	发送分组总长 (byte)
1000byte	68	42	1225903
500byte	64	34	1225998

表 5(b) 分组长度对性能的影响 DRR

	最大时延 (ms)	平均时延 (ms)	发送分组总长 (byte)
1000byte	315	160	1216903
500byte	301	148	1216998

实验 3 为了研究分组的最大时延与连接数的关系，我们设定连接 9 的预约带宽为 $r_9 = 10/55 \times 2\text{Mbit/s}$，连接 9 的流量符合其预约带宽，其它连接平分剩余的预约带宽，其它连接都为恶意用户，到达速率为 2Mbit/s。图 3 表示了连接 9 的最大时延随连接数变化的情况。时间单位为毫秒。可见，DRR 的最大时延随连接数的增加而线性增大，而当连接数为 676 时，LL-DRR 的最大时延增加到 288ms，不再随连接数增加而增大。可见 LL-DRR 具有最大时延与连接数无关的特性。时延的改善导致所需缓存减少。在本实验中，当连接数为 676 时，LL-DRR 情况下连接 9 不丢失分组所需的缓存为 20 个分组，而 DRR 情况下为 122 个分组。

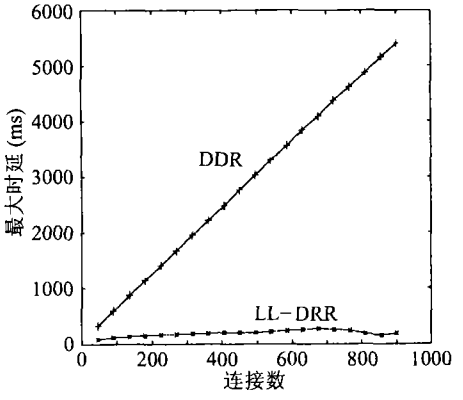

图 3 最大时延随连接数的变化

实验 4 IP 电话和视频会议等实时业务允许一定的分组丢失率。允许一定的分组丢失率可以减小 LL-DRR 提供给连接的最大时延同时减少所需缓存。保持实验 3 的条件不变，连接数设定为 676，限定连接 9 的缓存门限，可得到最大时延和丢失率。表 6 给出了仿真结果。缓存门限的单位为分组，最大时延的单位为 ms。

表 6 LL-DRR 的缓存门限和最大时延、丢失率的关系

缓存门限	11	12	13
最大时延 (ms)	165	192	219
丢失率	1.25×10^{-3}	5×10^{-4}	2.5×10^{-4}

4.3 理论分析

可以将 DRR 也看成是一种基于调度表的轮循方式，表中同一个连接的表项是紧挨着的，每一个表项表示一个 SQ 的服务定额。对于 4.1 节例 1，DRR 对应的调度表为表 7。

表 7　DRR 的一个调度表

调度表中位置的编号：	0	1	2	3	4	5
连接号：	0	0	0	1	1	2

定义一个连接的最大服务间隔为相邻两次对一个连接服务的间隔表项数目的最大值。例如表 7 中连接 0 的最大服务间隔为 4，表 1 中连接 0 的最大服务间隔为 2。

对于 DRR，连接 i 的最大服务间隔等于 $F-n_i+1=F\times(1-n_i/F)+1=F\times(1-r_i/C)+1$。当连接数增加时，$F$ 将相应增大，最大服务间隔增大导致最大时延上升。LL-DRR 在安排调度表时试图均匀地公平地安排各个连接的表项。定义 T_i^k 是连接 i 的第 $k(k=0,1,\cdots,n_i-1)$ 个表项在 LL-DRR 的调度表中的编号，则有如 (1) 式所示的不等式成立。例如对于表 1 的调度表有 $T_0^0=0, T_0^1=2, T_0^2=4, T_1^0=1, T_1^1=3, T_2^0=5$，

$$k(F/n_i)\le T_i^k\le(k+1)(F/n_i) \tag{1}$$

证明　由 LL-DRR 建立调度表的算法可知，只有当连接的开始时标小于或等于调度表中一个表项的编号，该表项才能分配给此连接，因此有 (2) 式成立

$$k(F/n_i)\le T_i^k \tag{2}$$

设一个调度器 Z 采用 WF^{2}Q[8] 排队算法。有 N 个连接，每个连接的分组长度固定，称为信元，时间单位为调度器发送一个信元的时间。调度器的服务速率为 1，连接 i 的预约带宽为 n_i/F，$\sum_{i=0}^{N-1}n_i=F$。从时刻 0 开始，连接 $i(i=0,1,\cdots,N-1)$ 快速到达 n_i 个信元。用 P_i^k 表示连接 i 的第 k 个信元 $(k=0,1,\cdots,n_i-1)$。由于快速到达，P_i^k 的开始时标 $S_i^k=k\times(F/n_i)$，结束时标 $F_i^k=(k+1)\times(F/n_i)$。由 WF^{2}Q 和 LL-DRR 算法的定义容易知道，调度器 Z 输出信元的顺序与 LL-DRR 调度表中连接的表项排列的顺序是一致的，即 P_i^k 是调度器 Z 第 T_i^k 个输出的信元，P_i^k 离开调度器的时间为 T_i^k+1。

F_i^k 表示 P_i^k 在调度器 Z 对应的 GPS 系统中离开时的系统虚拟时间[4]。由于 Z 对应的 GPS 系统虚拟时间的增长率为 1，则 F_i^k 就是 P_i^k 离开 GPS 的时间。由于 WF^{2}Q 中信元离去的时间最多比在相应的 GPS 中离去的时间晚 1 个信元时隙，则有

$$T_i^k+1\le F_i^k+1\Rightarrow T_i^k\le(k+1)(F/n_i) \tag{3}$$

由 (2), (3) 式知，(1) 式成立。　　证毕

由 (1) 式可知，LL-DRR 的调度表中一个连接的最大服务间隔为 $2F/n_i-1=2C/r_i-1$，与连接的预约带宽有关，与连接数无关。这就是 LL-DRR 的最大时延与连接的预约带宽有关而与连接数无关的原因。

5 有关 LL-DRR 实现的几点考虑

虽然 LL-DRR 选取下一个分组的复杂性为 $O(1)$，但是在连接建立和拆除阶段，为了更新调度表，LL-DRR 需要做额外的计算。当更新调度表的频率不高时 (对于路由器，公平调度是属于网络管理的一部分[9]，因此，调度表的更新频率不高)，现有的硬件水平可以很好地消化这些开销。如果用 ASIC(其复杂性与连接建立和拆除的频率以及调度表的长度有关) 来实现调

度表的建立，则可以大大缩短调度表的建立时间，LL-DRR 就可以适用于连接接纳控制频率较高的场合。

服务粒度 SQ 不能取得太小，太小会使很多情况下 SQ 和 Deficit 的和小于连接头一个分组的长度，从而使调度器不能输出分组，影响调度器输出分组的速率。SQ 也不能取得太大，太大会使连接经调度器输出后突发性很强，且最大时延增加。本文的仿真实验取 SQ 等于 LMAX. 这使得 SQ 和 Deficit 的和一定大于或等于连接头一个分组的长度。SQ 取值也可以小一些，这样可降低一些分组时延。

调度表的长度取决于连接的最小预约带宽和最小预约带宽间隔，长度越长，支持的最小预约带宽和预约带宽间隔越小. 例如对于本文的仿真实验，如果调度表长度为 2000，可以支持 1kbit/s 的预约带宽。当连接的预约带宽之和小于调度器速率时，连接实际可分配带宽等于 $r_i \times C'/\sum_{j=0}^{N-1} r_j$，大于连接的预约带宽，相应地 $n_i = r_i \times F/\sum_{j=0}^{N-1} r_j$。

可以构造层次化的 LL-DRR 使得用较短的调度表可以支持很多的连接。如图 4。将预约速率相同的连接 (例如 $S1, S2$) 组成一个组 (Group)，相当于一个宏连接，宏连接的预约带宽为组内各个连接的预约带宽之和，在宏连接之间采用 LL-DRR 进行调度，在一个组的各个连接之间采用 DRR 进行调度，这种层次化结构可以支持很多的连接，保证每个连接的吞吐率和时延。

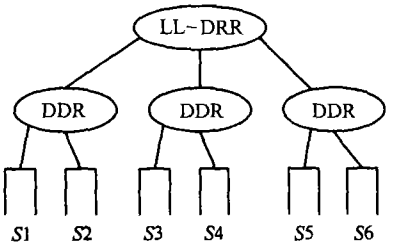

图 4 层次化的 LL-DRR

6 结 论

在时延性能上本文提出的 LL-DRR 比 DRR 有显著的改善，且具有连接的最大时延与连接数无关的特性。由于时延的改善，LL-DRR 所需缓存也比 DRR 少。LL-DRR 继承了 DRR 在平均吞吐率上的公平性。LL-DRR 既可调度实时业务也可调度非实时业务 (best-effort 业务)。由于只需读取调度表就可确定下一个发送的分组，LL-DRR 的实现复杂性为 $O(1)$，易于实现，适用于从低速 (2Mb/s) 到高速 (622Mb/s, 2.4Gb/s 等) 的各种网络。LL-DRR 是一种有效的分组调度算法。

参 考 文 献

[1] H. Zhang, Service disciplines for guaranteed performance service in packet-switching networks, Proc. IEEE, 1995, 83(10), 1374–1396.

[2] M. Shreedhar, G. Varghese, Efficient fair queuing using deficit round robin, ACM SIGCOMM., 1995, 25(4), 231–242.

[3] A. Demers, S. Keshav, S, Shenker, Analysis and simulation of a fair queuing algorithm, ACM SIGCOMM., 1989, 19(4),1–13.

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

[5] D. C. Stephens, J. C. R. Bennett, H. Zhang, Implementing scheduling algorithms in high-speed networks, IEEE J. on SAC, Special Issue on Next Generation IP Switches and Routers, 1999, 17(6), 1145–1158.

[6] J. B. Nagle, On packet switches with infinite storage, IEEE Trans. on Communications, 1987, COM-35(4), 435–438.
[7] M. Katevenis, S. Sidiropoulos, C. Courcoubetis, Weighted round-robin cell multiplexing in a general-purpose ATM switch chip, IEEE J. on SAC, 1991, 9(8), 1265–1279.
[8] J. C. R. Bennett, H. Zhang, WF^2Q: Worst-case fair weighted fair queuing, San Francisco, IEEE Infocom, California, 1996, 120–128.
[9] 北京希望电脑公司, Cisco IOS 12.0 参考库—服务质量优化技术, 北京希望电子出版社, 1999, 7, 59–86.

LL-DRR: AN EFFICIENT SCHEDULING ALGORITHM FOR PACKET NETWORKS

Tu Xiaodong　　Li Lemin

(*National Key Lab. of Optical Fiber Comm., UEST of China, Chengdu* 610054, *China*)

Abstract　A novel fair queuing algorithm LL-DRR(Low Latency Deficit Round Robin)is proposed in this paper, which is based on DRR(Deficit Round Robin) algorithm and suitable to schedule variable length packets. The simulation and theoretical analysis show that the delay performance of LL-DRR is much better than DRR, and the maximum packet delay of a session in LL-DRR is independent of the number of sessions. LL-DRR can support real time service. LL-DRR inherits the fairness of DRR on average throughput. LL-DRR is simple to implement and applicable to high speed networks.

Key words　Packet fair queuing, Deficit round robin, Low latency deficit round robin

涂晓东: 男, 1970 年生, 博士, 研究方向为宽带网络中的 QoS 技术、主动网等.
李乐民: 男, 1932 年生, 教授, 博士生导师, 中国工程院院士, 从事宽带网络等方面的研究.