

文章编号:1001-9081(2005)07-1488-03

在 CICQ 交换结构下实现分布式的输入排队 DRR 分组公平调度

王 荣¹, 陈 越²

(1. 装甲兵技术研究所, 北京 100072; 2. 信息工程大学 国家数字交换系统工程技术研究中心, 河南 郑州 450002)
(wang6802@sohu.com)

摘 要:传统的基于 crossbar 的输入排队交换结构在提供良好的 QoS 方面存在很大的不足, 而 CICQ(combined input and crosspoint buffered queuing)交换结构与传统的交换结构比, 不但能在各种输入流下提供接近输出排队的吞吐率, 而且能提供良好的 QoS 支持。基于 CICQ 结构, 提出了在输入排队条件下实现基于流的分布式 DRR 分组公平调度算法的方案, 并通过仿真验证了这一方案的有效性。

关键词:CICQ 交换结构; 分布式 DRR 调度算法; 输入排队交换结构

中图分类号: TP393.07 **文献标识码:** A

Implementing distributed DRR scheduling algorithm in CICQ switches

WANG Rong¹, CHEN Yue²

(1. Technology Institute of Armoured Force, Beijing 100072, China;
2. National Digital Switching System Engineering & Technology R&D Center, Information Engineering University, Zhengzhou Henan 450002, China)

Abstract: Traditional input-queued switches based on crossbar are insufficient in providing good QoS performance. As a contrast, the CICQ(combined input and crosspoint queuing) switches can provide almost 100% throughput under different input traffics. Its performance is very close to the OQ(output-queued) switch, and has the potential to support good QoS. Based on the CICQ switch, a new scheme was put forward, which could realize distributed DRR schedule for the packets of variable length. It had both the scalability of input-queued switches and QoS performance of output-queued switches. Simulation results show the scheme is very effective and have good performance.

Key words: CICQ switch; distributed DRR schedule; input-queued switch

0 引言

目前 Internet 的主干链路的带宽已经达到 10Gbps, 网络规模早已遍布世界各地。但是随着带宽的高速增长, IP 网络的关键问题——QoS 问题却一直没有得到很好的解决。在未来的 Internet 网络中, 如果要对视频、语音和数据业务提供良好的支持, 则必须解决 QoS 问题。

在高速 IP 网络中, 带宽和 QoS 似乎是一对矛盾。在 IETF 提出的综合服务(Int-serv)框架中, 保证服务可为单个流提供有严格端到端时延和低分组丢失率的电路型服务, 这种服务需要在路由器中实现基于流的加权公平调度服务。WFQ(PGPS)^[1], WF²Q, DRR^[2]等调度算法可实现基于流的加权公平调度, 但这类算法都是在输出排队的环境下实现的。输出排队虽然可以提供良好的 QoS 支持, 但难以在高速网络环境中应用。因为在输出排队的交换结构中, 对于 $N \times N$ 的交换结构, 需要输出缓存和交换结构的加速比为 N , 受商用随机存储器访问速率的限制, 在高速交换结构中一般采用输入排队的交换结构。输入排队的交换结构对存储器的带宽要求仅为输入链路速率的 2 倍, 但在输入排队的交换结构下难以提供良好的 QoS 支持。在现有的基于输入排队的 crossbar 交换结构中, 一般采用定长的分组交换方式, IP 包在交换前先切割(segment)成定长的信元, 交换结构的仲裁单元通过匹配算法找到输入和输出端的极大匹配, 然后配置 crossbar 同步完成各端口间信元的交换。在这种交换方式下, 为了实现

100% 的吞吐率, 需要实现输入和输出端口的最大匹配或加权最大匹配^[3], 实现 QoS 的要求和最大匹配的要求难以兼顾。现有的求极大匹配的输入排队调度算法如 iSLIP 等, 其功能是通过迭代找到输入输出间的极大匹配, 但难以提供相应的 QoS 支持, 更无法提供基于流的带宽保证。

本文的讨论基于一种新的交换结构 CICQ(combined input and crosspoint queuing)^[4], 这种交换结构在输入排队的情况, 不但能够对 i. i. d Bernoulli 均匀流提供百分之百的吞吐率, 还能对各种突发的、非均匀的流提供接近百分之百的吞吐率^[5], 在吞吐率方面与输出排队(OQ)结构很接近。基于这种 CICQ 交换结构本文提出了实现分布式的 DRR 加权公平调度算法的方案。这里所谓的分布式是指各个输入和输出端口的调度器之间不需要复杂的协作和通信, 而是相对独立地工作, 从而具有实现的简单性和可扩展性。

1 CICQ 结构及其实现

1.1 CICQ 结构的特点

传统的 crossbar 交换结构, 其缓存位于交换结构的输入或输出端口, crossbar 交换结构中是没有缓存的。CICQ 交换结构如图 1 所示, 除了在输入端有缓存外, 在每个交叉点上(crosspoint)都有一个小的缓存, 称为交叉点缓存。对于 $N \times N$ 的交换结构而言, 一共有 N^2 个交叉点从而有 N^2 个缓存, 这种 crossbar 又称 buffered crossbar^[6]。在 CICQ 交换结构的输入端和传统的输入排队 crossbar 一样采用虚拟输出排队结构

收稿日期: 2005-01-11; 修订日期: 2005-03-04 基金项目: 国家 863 计划项目(2003AA103510)

作者简介: 王荣(1968-), 男, 山东荣成人, 博士研究生, 主要研究方向: IP 网络交换结构及调度算法; 陈越(1965-), 男, 河南郑州人, 博士研究生, 主要研究方向: 网络路由协议。

(VOQ),以消除队头阻塞。

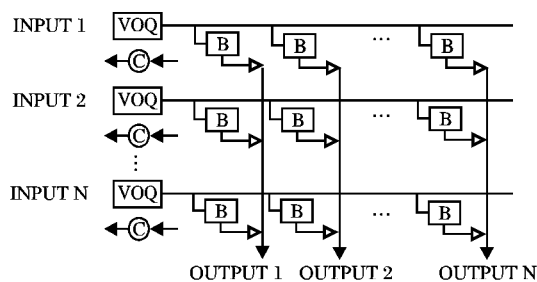


图 1 buffered crossbar 交换结构

CICQ 结构和传统 crossbar 结构的不同导致两者有显著的区别:

1) 传统的 crossbar 由于没有交叉点缓存,所有输入和输出端的连接按照事先确定好的配置同步进行操作。每个输入端只能连接到一个输出端,每个输出端也只能连接到一个输入端。而 buffered crossbar 由于在每个交叉点上有缓存,所有输入和输出端不必同步进行交换,每个输入端和每个输出端可以相互独立地和交叉点缓存进行交换。这样减少了同步所带来的问题,例如在同步情况下,每个输入端都属于不同的时钟域,各个时钟域间的信号需要进行同步。

2) 传统 crossbar 由于同步地进行交换, 需要一个专门的调度算法, 实现输入和输出端的最大匹配, 这是一个双向图匹配问题, 其算法复杂度为 $O(n^{2.5})$ 。在实际使用中为了在一个信元的时间内求出匹配矩阵, 一般采用求极大匹配的迭代算法。当链路速率提高、信元时间变短时, 性能良好的匹配算法就难以实现, 匹配算法的运算时间成为制约因素。而在 buffered crossbar 结构中, 由于取消了输入和输出的同步限制, 每个输入和输出端独立地操作, 从而不需要实现输入和输出端的匹配算法, 使交换结构在更高的速率下运行并支持良好的 QoS 成为可能。

3) 传统的 crossbar 需要同步完成各端口的交换, 输入的变长 IP 分组需要切成定长的信元, 这种定长的信元会浪费一部分带宽, 从而要求交换结构有一定的加速比以补偿这部分带宽。buffered crossbar 结构就不存在这种问题, CICQ 结构可以直接实现变长的分组交换, 交换结构的带宽可以得到更充分的利用。

4) 传统的基于输入排队结构的 crossbar 难以提供良好的 QoS 支持,像 WFQ,DRR 这样的基于流的加权公平调度算法难以实现。而在 buffered crossbar 交换结构中可以实现更好的 QoS 性能,本文就是讨论在输入排队的 CICQ 结构下,实现分布式的 WFQ 类加权公平调度方案。

1.2 CICQ 结构的实现

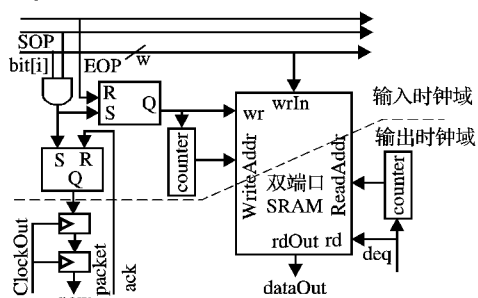


图2 crosspoint 的逻辑结构

如果把输入缓存全部设置在 crosspoint 中,由于节点数是 N^2 ,这样的结构显然难以实现。可以考虑只在 crosspoint 上设置小的缓存,把大量缓存设置在输入端的 VOQ 队列中,通过适当的流控机制,使 crosspoint 中的缓存不会溢出(图 1)。

图2为每个 crosspoint 的参考逻辑结构,存储器由一个双端口的 SRAM 组成,因为要同时完成读写操作,所以使用双端口 SRAM。SOP 表示分组到来,EOP 表示分组结束,每个分组通过 w 位宽的总线接入。在每个分组的头一个字节中包含一个多播的位图,规定接收分组的交叉点缓存。由于写入和读出的时钟域不同,因此通过由两个 D 触发器组成的同步电路完成 new packet 信号到输出时钟域的同步。

2 分布式 DRR 分组公平调度

2.1 分布式的 DRR 分组公平调度结构

DRR (Deficit Round Robin)^[2] 调度算法是由 M. Shreedhar 等人提出的基于流的分组公平排队调度算法。该算法和 WFQ 类公平调度算法一样, 可对变长分组提供基于流的公平调度, 但 DRR 调度算法的计算复杂度仅为 $O(1)$, 远低于 WFO 类调度算法。

在输出排队结构中由于只有一个资源竞争点,在输出端口,因此只需要在每个输出端口中实现一个分组调度器即可。在 CICQ 结构中,分组队列缓存在输入端口,在每个 crosspoint 缓存中也有分组,因此要想模仿输出排队结构,需要在多个竞争点进行资源调度^[4]:包括每个输入端的 VOQ 队列中,交换结构的每个输入端口和输出端口。如图 3 所示,图中实现的是一个 $N \times N$ 的交换结构。

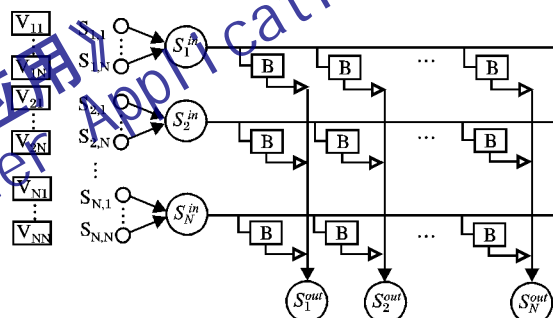


图3 在CICO结构下实现分布式DRR调度算法构架

具体实现如下:在每个 VOQ 队列 $V_{i,j}$ 中设置一个 DRR 调度器 $S_{i,j}$, 负责调度每个 VOQ 队列中的所有流, 同一输入端的所有调度器输出通过调度器 S_i^{in} 输出到 crossbar 的缓存中, crossbar 中所有到同一输出端的缓存, 由调度器 S_j^{out} 调度输出。在上述各个调度器中为了实现分布式的调度, 即各个 VOQ 队列中的调度器不需要互相通信和协作, 各个输入端口和输出端口的调度器也不需要互相通信和协作, 我们采用如下的调度策略: 每个 VOQ 中的调度器 $S_{i,j}$ 只对相应 VOQ 中的所有流进行调度, 输入端口的 S_i^{in} 调度器, 对输入端口 i 中的 N 个 VOQ 队列作为流进行调度, 各个 VOQ 队列的配额 (quota) 与相应 VOQ 队列中各个预约流的配额之和成比例, 如果一个 VOQ 队列作为整体它的带宽得到保证, 则其中每个流的带宽也相应能得到保证。输出端口调度器 S_j^{out} 对到输出端口 j 的 N 个 crosspoint 缓存作为流进行调度, 各个 crosspoint 缓存队列的配额与相应输入端中到输出端 j 的 VOQ 队列中各预约流的配额之和成比例, 同样如果 VOQ 队列整体的带宽得到保证, VOQ 队列中各流的带宽也得到保证。上述的各调度器 $S_{i,j}$, S_i^{in} 和 S_j^{out} 均采用 DRR 调度器。

这种调度结构除了有分布式的特点外,在硬件实现上有明显的优势。各个 VOQ 队列的调度器可以分布在各个线卡上实现,这样把数目庞大的流用分布式的方法来实现。可以设想,如果在输出排队结构实现 DRR 调度算法,所有的流都将在输出端口实现调度,这样每个输出端口调度的流的数

目将是以上 VOQ 队列中流的数目的 N 倍,显然分布式的调度结构比输出排队有更好的可扩展性;在交换结构中,每个输入端口以 N 个固定的 VOQ 队列作为调度流,每个输出端口以 N 个 crosspoint 缓存作为固定的调度流,在这样的结构中,流的数目是固定不变的,不仅可以简化调度算法,也方便硬件的实现,事实上,所有的 S_i^{in} 调度器和 S_j^{out} 调度器可以在一块交换芯片上实现,而各个 $S_{i,j}$ 调度器则在各个线卡上实现。

2.2 分布式 DRR 调度算法的实现

2.2.1 在 VOQ 队列中 DRR 调度的实现

对 VOQ 队列中的每个流 $flow_k$ 我们维持两个计数器, $quota_k$ 和 $counter_k$, $quota_k$ 与每个流的预留带宽成比例,假定 $quota_k \geq MTU$, $counter_k$ 代表这个流的当前可用配额。同时维持一个活动链表 $activeList$, 记录这个 VOQ 队列中的活动流。当有分组到达一个空流时,这个流被添加到链表的末尾;一个队列被服务完后,也添加到链表的末尾。调度器从链表头取出队列 k , 把 $counter_k$ 加上 $quota_k$, 如果 $counter_k$ 大于队列 k 的队头分组的长度,则调度输出这个分组;直到 $counter_k$ 小于队列 k 的队头分组的长度,将这个队列放到链表的末尾。VOQ 队列中 DRR 调度算法的伪码如下:

```

/初始化/
for (i=0; i<n; i++)
    counteri = 0;
/入队操作/
if (一个新分组 p 到达)
{
    i = extractflow(p);
    enqueue(i, p);
    if (ExistsInactiveList(i) = false)
    { InsertactiveList(i);
      Counteri = 0; }
}
/离队操作/
if (activeList 非空)
{
    i = accessactiveList;
    counteri = counteri + quotai;
    while ((counteri > 0) and (queuei not empty))
    {
        packetSize = size(head(queue(i)));
        if (packetSize ≤ counteri)
        { send(dequeue(queue(i)));
          counteri = counteri - quotai; }
        else break;
    }
    if (empty(queue(i))
        removeactiveList(i);
}

```

2.2.2 输入端口和输出端口调度器的实现

在每个输入端口,调度器 S_i^{in} 对 N 个 VOQ 作为固定的流进行调度。在输入端口 i , 维持一个有 N 个固定表项的调度列表 $inputlist_i$, 每一项对应一个 VOQ 队列, 其中每一个表项的内容为:

```

struct inputListItem
{
    empty;
    backpressure signal;
}

```

当调度器服务到该队列时,检查链表中的表项,如果该队列为空,则跳过该队列,继续寻找下一项;如果该队列非空,则检查反压信号项,如果反压信号项为真,则说明该队列处于反

压状态,继续检查下一个表项。对每个队列的调度方式同上面的 VOQ 队列调度。每个队列的配额 $quota_{i,j} = \frac{1}{c} \sum_{l \in VOQ_{i,j} \text{ 中所有流}} quota_l$, $j = 1, 2, \dots, N$, c 是一个常数,并且假定 $quota_{i,j} \geq MTU$ 。

在每个输出端口 j , 同样维持一个有 N 个表项的调度列表 $outputList_j$, 与输入端不同,每个表项只有一个 empty 指示,指示相应的 crosspoint 缓存队列是否为空。输出端也采用轮询的服务方式,当遇到表项中 empty 为空时,继续检查下一个表项。输出端每个队列的配额为 $quota_{i,j} = \frac{1}{c} \sum_{l \in VOQ_{i,j} \text{ 中所有流}} quota_l$, $i = 1, 2, \dots, N$, c 是常数,假定 $quota_{i,j} \geq MTU$, 每个队列的调度方式也与 VOQ 队列相同。

3 仿真环境和仿真结果

我们模仿一个 8×8 的交换结构,假定每个输入端口的链路速率为 1Gbps。每个输入端的流按照到不同的输出端排在不同的 VOQ 队列中,令 $f(i, j)$ 代表从输入端 i 到输出端 j 的流。

仿真情形 1:

假定流 $f(1,1)$, $f(2,1)$, $f(3,1)$, $f(4,1)$, $f(5,1)$, $f(6,1)$, $f(7,1)$, $f(8,1)$ 的预约带宽分别为 20%, 20%, 15%, 15%, 10%, 10%, 5%, 5%, 且这 8 个流的到达速率都相同,其他流的到达速率为 5%。改变这 8 个流的到达速率,考察在输出端口的带宽分配情况,见图 4。

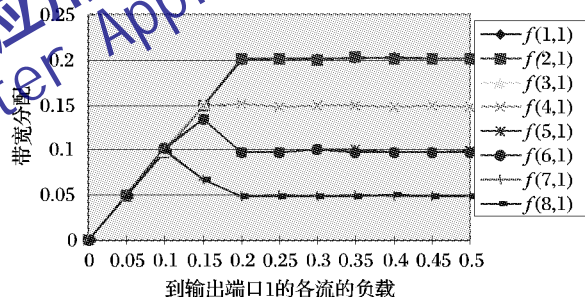


图 4 到输出端口 1 的各流的带宽分配

当输入负载在 12.5% 以下时,各个流得到的带宽相同;在输入负载在 12.5% 以上时,各个流得到的带宽开始不同。当输入负载大于 20% 时,各个流按照预约的带宽比例得到服务。当输入负载在 12.5% 和 20% 之间时,例如当负载在 15% 时,流 $f(1,1)$, $f(2,1)$ 得到的带宽分别为 15%, 两个流的预约带宽为 20%, 共剩下 10% 的带宽。流 $f(5,1)$, $f(6,1)$ 得到的带宽分别为 13.4%, 流 $f(7,1)$, $f(8,1)$ 得到的带宽分别为 6.7%, 多得到的带宽分别为 3.4% 和 1.7%, 可见剩余带宽按照预约的比例在各流中分配,符合加权公平分配的原则。

仿真情形 2:

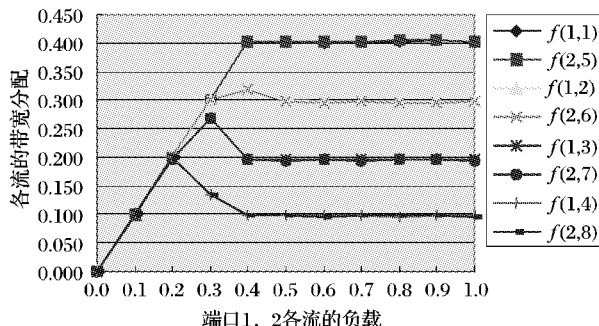


图 5 到输出端口 1,2 的各流的带宽分配

由输入端调度算法可知,只要可用的空闲输出缓存超过阈值 T ,输入 VOQ 的分组就能得到服务输出到输出缓存。而在输出队列中,一个到达分组只要在前面的分组服务结束后就能得到服务输出,同时由于输出队列长度受控于阈值 T 而不会出现无限长的情况,因此新到达的分组在一段有限时间后一定能得到服务。在一段有限时间后输出队列的空闲缓存空间一定会大于 T ,同时只要输入端的缓存足够大而不至于引起输入队列溢出而丢弃分组,输入的分组就一定能到达输出队列。因此,在输入缓存足够大的情况下到达的完整分组不会在系统中拥塞丢弃,整个交换系统可以达到 100% 的吞吐率。而在均匀流量模式下,这种 CIOQ 模型在负载 50% ~ 90% 时,平均时延比理想的 OQ 交换仅有稍微的增加,如图 3 所示。

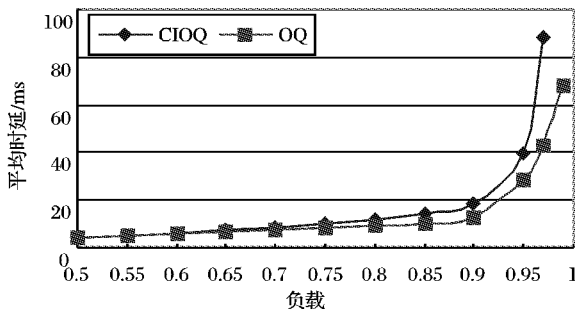


图3 CIOQ 和理想 OQ 的负载和平均时延比较

对于每个输出端的平行队列,设队列容量为 M 。由于我们对变长分组采用整包调度,每次以一个分组为单元,因此每次调度输出时队列中至少要有有一个完整的分组存在,这样队列容量 M 最小为一个最大分组长度。若队列容量 $M = MTU$,则输出缓存的阈值 T 应该满足:当正在输出的分组完全离开队列的同时队列中应该至少还有一个完整的分组可以被再次调度输出。因此,理想的情况下阈值 T 应该等于 $M/2$,即一个最大分组长度的一半。但是在实际操作中取 M 大于 MTU ,即 $M = 3k$ 字节, $T = MTU$ 。当输出队列中空闲空间小于 T 时输出队列输出一个分组,若这时空闲队列仍小于 T ,则继续从输出队列

中输出分组;若队列空闲空间大于 T 时,则在继续输出该队列中分组的同时再向输入端的调度器发送允许输出标识,那么在该队列为空时输入的分组已经存储到了队列中,可以继续连续输出,避免了工作不守恒状态的出现。若初始时输出队列空闲空间大于 T ,那么输入队列的分组就直接写到输出队列中输出。该结构相当于采用了两级流水线操作,实现了系统的守恒工作。

4 结语

文章在分析传统 CIOQ 交换结构的基础上对输出队列和内部交换互连部分进行扩展,实现了交换调度的分布式操作,降低了调度实现的复杂度。同时在扩展 CIOQ 结构中通过反馈控制和基于包调度的 WRR 算法,在保证 100% 吞吐率的情况下实现了对变长分组的交换。另外,可以适当修改输入端的队列分布规则使得这种 CIOQ 交换结构实现了对区分服务的 QoS 支持。

参考文献:

- [1] KAROL M, HLUCHYJ M, MORGAN S. Input versus output queuing on a space division switch[J]. IEEE Transactions on Communication, 1988, 35(12): 1347-1356.
- [2] PRABHAKAR B, MCKEOWN N, STAN-CSL-TR-97-738, On the Speedup Required for Combined Input and Output Queued Switching, Stanford University, 1997.
- [3] MARSAN M, BIANCO A, GIACCONE P, et al. Packet Scheduling in Input-Queued Cell-Based Switches [A]. INFOCOM 2001 [C], 2001. 1085-1094.
- [4] CHRYSOS N. TR 325, Design Issues of Variable-Packet-Size, Multiple-Priority Buffered Crossbars [R]. Heraklion, Crete, Greece, 2003.
- [5] SHIMONISHI H, SUZUKI H. Analysis of Weighted Round Robin Cell Scheduling and Its Improvement in ATM Networks [J]. IEICE Transactions on Communications, 1998, E81-B(5): 910-927.

(上接第 1490 页)

假定输入端口 1 中有 4 个流分别输出到端口 1 到 4,输入端口 2 中有 4 个流分别输出到端口 5 到 8,流 $f(1,1)$, $f(1,2)$, $f(1,3)$, $f(1,4)$ 的预约带宽分别为 40%, 30%, 20%, 10%, 流 $f(2,5)$, $f(2,6)$, $f(2,7)$, $f(2,8)$ 的预约带宽分别为 40%, 30%, 20%, 10%, 改变各流的输入负载,我们看各流实际得到的带宽,见图 5。

可见,当输入负载小于 25% 时,各流得到的带宽相同,当输入负载大于 40% 时,各流按照预约的比例得到带宽。当输入负载在 25% 和 40% 之间时,例如当负载为 30% 时,流 $f(1,3)$, $f(2,7)$ 得到的带宽分别为 26.8%, 流 $f(1,4)$, $f(2,8)$ 得到的带宽分别为 13.4%, 分别多得到 6.8% 和 3.4%, 因此剩余的带宽按照预约的比例在各流间公平的分配。

4 结语

传统的 crossbar 交换结构广泛用于各种现代路由器中,但传统的 crossbar 交换结构在提供良好的 QoS 方面存在着很大不足,本文在传统的交换结构基础上提出并讨论了一种新的交换结构——CICQ (combined input and crosspoint buffered queuing) 交换结构,这种交换结构相比传统的交换结构不但在各种输入流下能提供更好的吞吐率、更高的效率,其吞吐率

接近输出排队交换结构,而且能够实现良好的 QoS 能力。本文提出了在 CICQ 交换结构下实现分布式的 DRR 加权公平调度算法的方案,这一方案具有良好的可扩展性,可以适应高速的网络环境。仿真结果显示,分布式的 DRR 调度方案能够实现各个流在输入排队情况下的公平调度

参考文献:

- [1] PAREKH A, GRILLER R. A generalized processor sharing approach to flow control - The single node case [A]. Proc IEEE InfoCom [C], 1992. 915-924.
- [2] SHREEDHAR M, VARGHESE G. Efficient fair queuing using deficit round robin [A]. SIGCOMM [C], Boston, 1995.
- [3] MCKEOWN N, ANANTHARAM V, WALRAND J. Achieving 100% throughput in an input-queued switch [A]. Proc InfoCom'96 [C], 1996. 296-302.
- [4] STEPHENS D, ZHANG H. Implementing Distributed Packet Fair Queuing in a Scalable Switch Architecture [A]. Proc INFOCOM [C], 1998.
- [5] KATEVENIS M, PASSAS G, SIMOS D, et al. Variable Packet Size Buffered Crossbar (CICQ) Switches [A]. Proceedings of IEEE International Conference on Communications (ICC 2004) [C]. Paris, France, 2004.