一种简单的平滑公平轮转调度算法

郭子荣[1,2] 曾华燊[1] 窦 军[1]
(西南交通大学信息科学与技术学院 成都 610031)[1] (包头铁道职业技术学院 包头 014060)[2]

摘 要 根据通用处理器共享的公平排队思想,针对数据包或信元交换,提出了一种将数据流的预订速率作为时隙分配的权值来构建动态调度树的公平轮转调度算法。其主要思路是:当有新数据流到达时,将各数据流按其权值均匀分布到完全二叉树的叶子节点上,在每个时隙开始时轮转调度算法负责从叶子节点中依次取出数据流号,发送该数据流的信元,调度复杂度为O(1)。与其他经典的公平调度算法引比,所提出的公平轮转调度算法实现简单。理论分析和仿真结果都表明,这种简单的平滑公平轮转调度算法(SSFRR)具有良好的公平性,对源端为漏桶控制的数据流能够提供端到端的有界时延,且能够提供基于数据流的QoS保证。

关键词 平滑公平轮转调度,QoS保证,端到端时延,时隙分配
中图法分类号 TN91 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.2.028

Simple and Smoothed Fair Round Robin Scheduling Algorithm

GUO Zi-rong[1,2] ZENG Hua-xin[1] DOU Jun[1]
(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China)[1]
(Baotou Railway Vocational and Technical Institute, Baotou 014060, China)[2]

Abstract A simple and smoothed fair round-robin(SSFRR) scheduling algorithm for packet or cell switching was proposed based on the idea of packet generalized processor sharing(GPS) service discipline. The algorithm uses the reserved rates of active flows as their weights to build scheduling tree for allocating slots. The basic idea of SSFRR is to distribute all flow IDs over the leaf nodes of a complete binary tree according to the weight of each flow by scanning the weight matrix when a new flow arrives. At each time slot, SSFRR visits next leaf node of the binary tree in sequence from left to right, takes out the flow ID from the current leaf node and schedules the cell of this flow. SSFRR scheduling algorithm has O(1) scheduling time complexity. Compared with other classical fairness round robin scheduler, SSFRR algorithm is simpler to implement. Analysis and simulation show that SSFRR scheduling algorithm has good fairness property, and can provide end-to-end delay bounds for those flows which are token-bucket regulated by sources. SSFRR can provide flow-based QoS guarantees.

Keywords Smoothed fairness round robin scheduling, QoS guarantees, End-to-end delay, Slots allocation

1 引言

目前,互联网已成为全球信息基础设施,通过互联网实现的语音、视频和数据的融合通信已得到广泛的应用。然而传统的Internet只提供"尽力而为"的服务,在网络负载较大的情况下不能保证如VoIP、视频会议等实时应用的服务质量。网络节点设备(如路由器或交换机)负责对通过该网络节点的数据包进行调度和转发,对提供服务质量(QoS)保证起着至关重要的作用。

通常,能够提供服务质量保证的包调度器应具有下列属性[10,11]。首先,调度器必须能够隔离不同的流,使得一个数据流在即使存在恶意流的情况下也能得到性能保证;第二,调度器需要以公平的方式在竞争的数据流之间分配可用的带宽;第三,提供最低的最坏情况时延和时延抖动,对源端由漏桶控制的数据流能够提供端到端的时延界限保证;第四,调度算法具有较低的复杂度,足够简单、易于硬件实现;最后,包调度器能够适应数据流数的变化,能在较宽范围的链路速率下执行。

基于通用处理器共享(GPS)[1,2]思想的公平排队包调度算法因为具有端到端的时延保证和公平性等属性,在过去的20余年中被广泛地研究,研究人员所提出的包调度算法主要分为3大类,一类是基于时间戳的调度算法[1-6],其特点是使用某种方法计算和维护一个虚拟时钟,以此来仿效理想的通用处理器共享(GPS),典型的算法有加权公平排队(WFQ、PGPS)[1,2]、最坏情况公平的加权公平排队(WF^2Q)[3]及其改进算法WF^2Q+[4]、虚拟时钟(VC)[5]算法等,这类调度算法通常都有较好的公平性和有界时延,但都有较高的时间复杂度(至少需要$O(\log N)$),这在高速链路中无法实现。第二类是基于轮转的公平调度算法,DRR[7]、WRR[8]、CORR[9]、RRR[10]、SRR[11]、改进的SRR[13]等都是典型的轮转调度算

到稿日期:2015-01-10 返修日期:2015-05-21

郭子荣(1963—),女,博士生,副教授,主要研究方向为高速交换技术、数据库技术,E-mail:zirongguo@163.com;曾华燊(1945—),男,教授,博士生导师,主要研究方向为网络体系结构、高速交换技术和网络测试技术;窦 军(1963—),男,博士,副教授,主要研究方向为网络体系结构、高速交换技术。

法，这类算法简单且易于实现，调度时间复杂度为 O(1)，但通常存在输出突发性和短期不公平的问题，不能提供有界的调度时延。第三类公平调度算法是结合了时间戳和轮转特性的分层或分类轮转调度器[14-17]，通常是将数据流按照其预订速率或权值分成一定数量的类，每一类内包含速率或权值相近的流，使用两级调度，通过基于时间戳的方法确定要服务的类，类内采用轮转方法调度不同数据流的包，这种方法虽然提供了较好的短期公平性，但一个类内具有稍大权值的数据流会产生较大的时延和不公平，同时调度复杂度为准 O(1)。另一种最近被提出的虚拟公平排队机制[18]实际上也属于这种分类的两级调度方法。

本文给出一个简单的公平轮转调度算法（Simple and Smoothed Fairness Round Robin Scheduling Algorithm，SSFRR），该算法使用一个完全二叉树的逻辑结构来构建调度树，用一个权值矩阵[11]来表示当前所有活动数据流在质量协商阶段所获得的带宽即权值的二进制编码。每当有新数据流到达时，重建调度树，将各数据流按其权值均匀分布到完全二叉树的叶子节点上。轮转调度算法负责在每个时隙开始时依次从叶子节点中取出数据流号，发送该数据流的信元，因此调度时间复杂度为 O(1)；同时理论分析和仿真实验表明，所提算法有很好的公平属性和确定的调度时延界限，能够提供数据流的性能保证。

本文第 2 节详细描述 SSFRR；第 3 节给出 SSFRR 算法的理论性能分析；第 4 节是 SSFRR 算法的仿真结果；最后总结并展望未来。

2 简单平滑公平轮转调度算法 SSFRR

包交换机或路由器的每个端口都有一个调度器，用来确定哪一个数据包被发送，假设有 N 个活动的数据流竞争输出带宽，取名为 $f_1, f_2, \cdots, f_N$，输出链路带宽为 R，每个数据流 f_i 有一个预定的速率 r_i，在建立连接分配速率阶段，应保证 $\sum_{i=1}^{N} r_i \leqslant R$。数据流 f_i 被分配一个权值 w_i，正比于它的速率 r_i。

首先说明：SSFRR 算法通过给每个数据流分配与其权值成正比的时隙数目来保证数据流的带宽。时隙是定长的时间片，即在每个时隙内为一个数据流发送固定的字节数 L。对于变长的数据包，可以采用分段传输的方式，即在每个时隙内只从当前数据包中按顺序取出 L 字节传输，在这 L 字节数据之前要加上数据流的 ID，在到达输出线的缓存区时去掉数据流 ID 即可恢复为原来的数据包。采用定长的时隙来调度数据包，主要有两点优势：1）可以均匀地分配调度时隙，使各数据流以更平滑、公平的方式来共享带宽；2）可以缩短实时语音包的等待时间，通常语音包较小（几十字节），如果在它之前传输长度为 1500 字节的数据包，就会造成语音包的过度延迟。将较长的数据包分段传输，可以使较小的语音包插在这些分段之间尽快传输。为描述方便，在下面的叙述中将在一个时隙内传输的分段称为信元。

SSFRR 算法通过构建一个完全二叉树即调度树（Scheduling Tree，ST）将活动的数据流按照其权值均匀地分布到这个完全二叉树的叶子节点上，之后轮转调度算法依次访问叶子节点，即可确定数据流的调度顺序。若完全二叉树的深度为 T_d（只有根结点时 $T_d=0$），则 ST 中叶子节点的数目 $S_r=2^{T_d}$，S_r 即一轮调度中所包含的时隙数目。T_d（或 S_r）可以作为交换机或路由器的参数由用户设置。一种计算 S_r 的方法是：根据端口速率和信元长度，计算出单位时间内（例如 1ms）包含的时隙数目 S_0，再找到大于等于 S_0 的最小的 2 的幂数，以此作为一轮调度中包含的时隙数目 S_r。例如，若端口速率为 100Gb/s，信元长度为 64 字节，则 1ms 中包含的时隙数目 S_0 为 195312.5，取 $S_r=2^{18}=262144$ 作为一轮调度中的时隙数目，完全二叉调度树的深度为 18。

在确定了 S_r 值之后，流 f_i 的权值 w_i 就可按下式计算：

$$w_i=\frac{r_i}{R}\cdot S_r=\rho_i\cdot S_r \tag{1}$$

其中，w_i 表示流 f_i 在一轮（S_r 个时隙）调度中所占用的时隙数目，ρ_i 为 f_i 的归一化预定速率。

2.1 权值矩阵

为了将每个流 f_i 的 w_i 个时隙均匀地分布到 S_r 个时隙中，将权值 w_i 转换为 T_d 位的二进制编码，即 $w_i=\sum_{j=0}^{T_d-1} a_{i,j}2^j$，$N$ 个数据流权值的二进制系数构成一个权值矩阵 WM（N 行 $\times T_d$ 列）。

例如，对于 $T_d=4$，一轮中的总时隙数 $S_r=16$，假设有 4 个活动流 f_1、f_2、f_3 和 f_4，权值分别为 $w_1=1$，$w_2=4$，$w_3=8$，$w_4=3$，则权值矩阵 WM 为

$$WM=\begin{bmatrix}0&0&0&1\\0&1&0&0\\1&0&0&0\\0&0&1&1\end{bmatrix}$$

权值矩阵的列从右至左依次编号为 $0,1,\cdots,T_d-1$，注意在列 j 中各项的权重为 2^j。

2.2 SSFRR 算法描述

算法分为 3 部分：1）时隙分配算法 SlotsAllot，每当有新数据流到达时调用，重新分配时隙；2）调度算法 Scheduler，在每一个时隙开始时调用，确定下一个时隙要发送哪个数据流的信元；3）时隙回收算法 SlotRecall，在有数据流离开时调用。

2.2.1 时隙分配算法 SlotsAllot

在确定了一轮调度中包含的时隙数目 S_r 后，二叉树的层次和大小也就确定了。二叉树的每个非叶子节点都保存一个数值，表示本节点所代表的子树已分配的时隙数目；叶子节点从左至右表示一轮中每个时隙的分配情况，初始为 0，表示未分配，若一个时隙已分配给某个数据流，则相应的叶子节点值为该数据流的 ID。

为了将数据流均匀地分布到二叉树的叶子节点中，按照权值矩阵从最左列（第 T_d-1 列）到最右列（第 0 列）的顺序进行时隙分配，在第 j 列中从上至下依次扫描值为 1 的元素 a_{ij}，计算它所对应的权重 2^j，将它作为流 f_i 的时隙数分量 $slotsNum$ 分配到从根结点开始的二叉树中。

从根节点出发，将根节点保存的值增加 $slotsNum$，如果未超过总时隙数，则向子树分配，分配方法是：将待分配的时隙数 $slotsNum$ 除以 2，若 $slotsNum$ 为奇数，则 $bigHalf=slotsNum/2+1$，$smallHalf=slotsNum/2$；否则 $bigHalf=smallHalf=slotsNum/2$。之后比较左、右子树已分配的时隙数，将 $bigHalf$ 分配给已分配时隙数较少的子树（这样，对每一个非叶子节点，其左右子树已占用的时隙数目之差不会超过 1），重复这个过程，直到将一个时隙分配一个空的叶子节点。这个过程可以通过一个递归程序 buildTree 实现（在以 node 为根结点的子树中为流 $flowNo$ 分配 $slotsNum$ 个时隙），如算法 1 所示。

算法 1 buildTree

```
buildTree(flowNo, slotsNum, node) {
1. if (时隙数 slotsNum 为 0) 返回;
2. if (node 是叶子节点)
     将叶子节点 node 分配给 flowNo,并返回;
3. else {
4.   节点 node 已分配的时隙数增加 slotsNum;
5. smallHalf= bigHalf=slotsNum/2;
6.   if (若 slotsNum 为奇数)
       bigHalf=bigHalf+1;
7.   left=node 的左子树;
8.   right=node 的右子树;
9. if (left 空闲隙数>right 空闲隙数) {
10.     buildTree(flowNo, bigHalf, left);
11.     buildTree(flowNo, smallHalf, right); }
12. else {
13.     buildTree(flowNo, smallHalf, left);
14.     buildTree(flowNo, bigHalf, right); }
15. }
16.}
```

最后叶子节点从左至右保存的就是一轮时隙中数据流的调度顺序。图 1 给出了对 2.1 节中的例子的权值矩阵的每一列的非 0 元素分配时隙后的结果,即在一轮中数据流的调度顺序为:f_3、f_2、f_3、f_4、f_3、f_2、f_3、f_1、f_3、f_2、f_3、f_4、f_3、f_2、f_3、f_4。

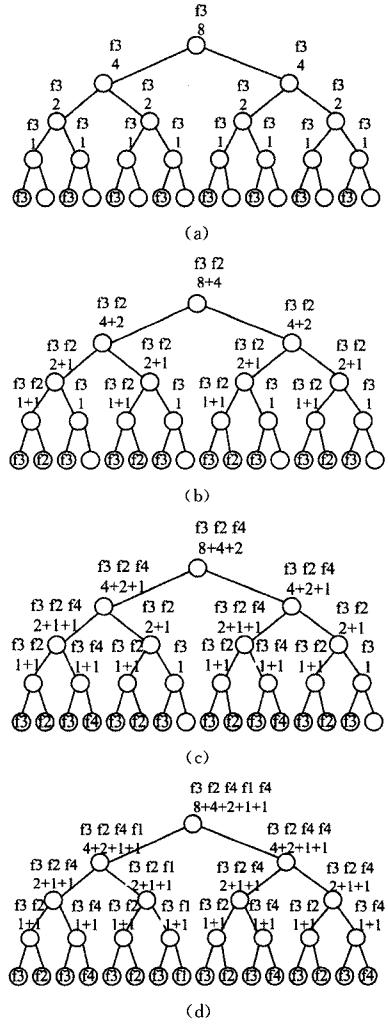


图 1 构建调度树分配时隙

需要注意的是,如果数据流的权值之和小于一轮中总的时隙数,则未分配的叶子节点的值为 0,可用于调度尽力而为业务的信元,或者跳过该节点,直接调度下一个叶子节点中的数据流。

2.2.2 轮转调度算法 Scheduler

轮转调度算法在每个时隙开始时检查下一个叶子节点中保存的活动流的 ID,若节点值大于 0,且对应的数据流有信元等待发送,则从该数据流的队列头部调度信元;若节点值为 0,表示对应的时隙未分配,则从尽力而为的数据流队列中调度信元。如果要调度的数据流队列中没有信元等待,则检查下一个叶子节点。因为每次调度只需按顺序从下一个叶子中取出流 ID,不需任何计算,所以调度时间复杂度为 O(1)。轮转调度算法如算法 2 所示。

算法 2 轮转调度算法

```
Scheduler () {
1. i=最左边的叶子节点;
2. while ( 处于一个忙期) {
3.   f=叶子节点 i 中保存的流 ID;
4. if ( f==0)
5.     if (有尽力而为数据包)
         发送尽力而为数据包;
6.     else if (f 有数据包排队)
         发送数据流 f 的队头数据包;
7.   i=下一个叶子节点;
8.   if (i 超过最右边叶子节点)
       i=最左边的叶子节点;
9.  } }
```

2.2.3 时隙回收算法 SlotsRecall

在 SSFRR 调度器中,每当有新数据流到达时重新进行分配时隙,即调度树的非叶子节点中保存的已分配的时隙数目只在构建调度树时有意义,因此当有数据流离开时,只须将该数据流占有的叶子结点清 0 即可,不必修改非叶子节点的值,因为再次分配时隙时,会重新从初值开始。时隙分配算法如算法 3 所示。

算法 3

```
// 时隙分配算法
SlotsAllot () {
1. int slots=Sr/2;
2. for (col=Kd-1;col>=0;col--){
3.   for (flow=1;flow<=N; flow++)
4.     if (wm[flow][col] == 1)
         buildTree(flow, slots, root);
5.   slots=slots/2;
6.  } }
```

3 SSFRR 算法的属性

由本节可知,SSFRR 调度器的所有属性都只与数据流本身的流量属性以及为本数据流分配的时隙数有关,不受其他数据流影响。

3.1 SSFRR 的时延界限

3.1.1 一个 SSFRR 调度器的排队时延上限

由前面的叙述可知,每当有新数据流到达时,时隙分配算法将当前活动的数据流按其预留的时隙数目均匀地分布到完

全二叉树的叶子节点中。对于数据流 f_i 的权值 $w_i=\sum_{j=0}^{T_d-1}a_{i,j}2^j$ 中每一个非 0 的位 j，每隔 2^{T_d-j} 个叶子节点就有一个叶子分配给数据流 f_i。在调度阶段，SSFRR 调度器依次轮转调度每个叶子节点对应的数据流，若该数据流队列中没有信元排队，则直接调度下一个叶子对应的数据流，保证 SSFRR 调度器是持续工作的。

对于一个数据流 f_i，如果在时间段[0，t]内不断有信元排队，则在该时间段内 SSFRR 调度器服务数据流 i 的信元数目 $S_i(t)$ 满足：

$$S_i(t)\geqslant\sum_{j=0}^{T_d-1}\left[\left\lfloor\frac{t-\theta(a_{ij})}{2^{T_d-j}}\right\rfloor+1\right]\cdot a_{ij} \tag{2}$$

其中，$\theta(a_{ij})$ 是数据流 f_i 的权值的第 j 位分量被分配的首个叶子节点的序号，$1\leqslant\theta(a_{ij})\leqslant 2^{T_d-j}$，此后序号为 $\theta(a_{ij})+2^{T_d-j}$、$\theta(a_{ij})+2*2^{T_d-j}$、$\theta(a_{ij})+3*2^{T_d-j}$…的叶子节点也分配给这个权值分量。

定理 1 如果 s 是一个 SSFRR 调度器，数据流 f_i 的流量是由漏桶(σ_i,ρ_i)限制的，则 f_i 的任何一个信元在 s 中的最大排队时延 D_i 为：

$$D_i\leqslant\frac{1}{\rho_i}(\sigma_i+C_i) \tag{3}$$

其中，σ_i 为漏桶限制的数据流 f_i 的最大突发长度(单位为信元数)，ρ_i 为 f_i 的归一化平均速率($\rho_i=r_i/R$)，C_i 为数据流 f_i 的二进制权值中非 0 系数的个数。

证明：设$[t_1,t_2]$是数据流 f_i 的一个忙期，不失一般性，假设在这个忙期内的时间 t^* 到达的一个信元经历了最大的排队时延 D_i，这意味着数据流 f_i 在时间段$[t_1,t^*]$内到达的信元在时间 t^* 之前都被调度，即

$$A_i(t_1,t^*)=S_i(t_1,t^*+D_i) \tag{4}$$

因为数据流 f_i 的流量是由漏桶限制的，所以有

$$A_i(t_1,t^*)=\sigma_i+\rho_i\cdot(t^*-t_1) \tag{5}$$

注意，这里的 t_1、t^* 和 D_i 均为时隙数。在$[t_1,t^*+D_i]$时间段内 SSFRR 调度器为数据流 f_i 服务的信元数为

$$\begin{aligned}
&S_i(t_1,t^*+D_i)\\
&\geqslant\sum_{j=0}^{T_d-1}\left[\left\lfloor\frac{(t^*+D_i-t_1)-\theta(a_{ij})}{2^{T_d-j}}\right\rfloor+1\right]\cdot a_{ij}\\
&\geqslant\sum_{j=0}^{T_d-1}\left[\frac{(t^*+D_i-t_1)-\theta(a_{ij})}{2^{T_d-j}}\right]\cdot a_{ij}\\
&=\frac{(t^*+D_i-t_1)}{2^{T_d}}\sum_{j=0}^{T_d-1}2^j a_{ij}-\sum_{j=0}^{T_d-1}a_{ij}\\
&=\frac{w_i}{2^{T_d}}(t*+D_i-t_1)-C_i\\
&=\rho_i(t*+D_i-t_1)-C_i
\end{aligned} \tag{6}$$

将式(6)和式(5)代入式(4)，得

$$\rho_i(t^*+D_i-t_1)-Nc_i\leqslant\sigma_i+\rho_i\cdot(t^*-t_1)$$

即

$$D_i\leqslant\frac{1}{\rho_i}(\sigma_i+C_i)$$

3.1.2 网络中多个调度器的排队时延上限

考虑一个数据流 f_i 穿越网络中的 M 个节点，假设每个节点都使用 SSFRR 调度器，在每个节点为数据流 f_i 分配相同的时隙数目 w_i。根据前面的分析可知，在任意的$[t_1,t_2]$时间间隔内，一个 SSFRR 节点对数据流 f_i 提供的服务为

$$S_i(t_1,t_2)=\sum_{j=0}^{T_d-1}\left[\left\lfloor\frac{(t_2-t_1)-\theta(a_{ij})}{2^{T_d-j}}\right\rfloor+1\right]\cdot a_{ij}$$

这个式子的最大值为(见后文中定理 2 的推导过程)

$$S_i(t_1,t_2)\leqslant\frac{w_i}{2^{T_d}}(t_2-t_1)+C_i-\frac{w_i}{2^{T_d}}$$

这个式子说明，一个 SSFRR 调度器的输出符合参数为$(C_i-\frac{w_i}{2^{T_d}},\rho_i)$的漏桶输出，由定理 1 可知，下一个节点排队时延为

$$D_i^k\leqslant\frac{1}{\rho_i}(2C_i-\frac{w_i}{2^{T_d}}),1<k\leqslant M$$

由此可得，数据流 f_i 的端到端的排队时延上限为

$$D_{Mi}\leqslant\frac{1}{\rho_i}[\sigma_i+(2M-1)C_i-(M-1)\frac{w_i}{2^{T_d}}]$$

其中，M 为数据流 f_i 经过的 SSFRR 节点的数目。

3.2 SSFRR 的公平属性

3.2.1 服务公平指标

在一个链路没有完全被数据流占用的情况下，剩余的链路容量应该由活动的流按它们的预定速率公平共享[2]。如果数据流 f_i 和 f_j 在时间段$[t_1,t_2]$内不断有信元排队，则在理想的情况下，在这个时间段内服务 f_i 和 f_j 的流量应为：

$$\frac{S_i(t_1,t_2)}{w_i}=\frac{S_j(t_1,t_2)}{w_j}$$

实际应用中以信元为单位进行服务或调度，因此不可能获得理想的公平性。为此定义一个公平性度量即“服务公平指标(Service Fairness Index，SFI)”来量化一个调度算法的公平性[10]。

$$SFI=\max_{t_1,t_2}\left|\frac{S_i(t_1,t_2)}{w_i}-\frac{S_j(t_1,t_2)}{w_j}\right|$$

一个调度算法具有的 SFI 值越小，意味着算法的比例公平性越好，$SFI=0$ 为理想公平。

定理 2 持续工作的 SSFRR 调度算法为每个数据流提供的服务只与数据流预定的权值(时隙数)有关，不受数据流的流量类型影响，其服务公平指标为

$$SFI_{SSFRR}=\frac{C_i}{w_i}+\frac{C_j}{w_j}-\frac{1}{2^{T_d}}$$

证明：由式(6)可知，SSFRR 调度器在$[t_1,t_2]$时间段内为数据流 f_i 调度的信元数目最少为

$$S_i(t_1,t_2)\geqslant\frac{w_i}{2^{T_d}}(t_2-t_1)-C_i \tag{7}$$

另一方面，$S_i(t_1,t_2)$可以获得的最大值为

$$\begin{aligned}
S_i(t_1,t_2)&=\sum_{j=0}^{T_d-1}\left[\left\lfloor\frac{(t_2-t_1)-\theta(a_{ij})}{2^{T_d-j}}\right\rfloor+1\right]\cdot a_{ij}\\
&\leqslant\sum_{j=0}^{T_d-1}\left[\frac{(t_2-t_1)-1}{2^{T_d-j}}+1\right]\cdot a_{ij}\\
&=\frac{(t_2-t_1)}{2^{T_d}}\sum_{j=0}^{T_d-1}2^j a_{ij}-\frac{1}{2^{T_d}}\sum_{j=0}^{T_d-1}2^j a_{ij}+\sum_{j=0}^{T_d-1}a_{ij}\\
&=\frac{w_i}{2^{T_d}}(t_2-t_1)-\frac{w_i}{2^{T_d}}+C_i
\end{aligned} \tag{8}$$

结合式(7)和式(8)，可得

$$SFI_{SSRR}=\max_{t_1,t_2}\left|\frac{S_i(t_1,t_2)}{w_i}-\frac{S_j(t_1,t_2)}{w_j}\right|=\frac{C_i}{w_i}+\frac{C_j}{w_j}-\frac{1}{2^{T_d}}$$

3.2.2 最坏情况公平指标

最坏情况公平指标(WFI)是通过一个队列在基于信元的系统和相应的理想化的流体系统上接受到的服务量的最大差

值来定义公平性，较大的 WFI 意味着调度输出业务有较大的突发性。这里引入 Benett 和 Zhang 关于常速率服务器的最坏情况公平指标的定义，以此来衡量 SSFRR 调度算法的最坏情况公平程度。

定义 1（最坏情况公平） 如果一个调度器 s 在任意的$[t_1, t_2]$时间间隔之内，对数据流 f_i 提供的服务满足 $S_i(t_1, t_2) \geqslant r_i(t_2 - t_1) - \gamma_i$，则称调度器 s 对数据流 f_i 保证 γ_i 位的最坏情况公平指标，其中 r_i 为数据流 f_i 的保证速率。

定理 3 调度器 SSFRR 对数据流 f_i 有 C_i 的最坏情况公平指标，即 $WFI = C_i$，其中 C_i 为数据流 f_i 的二进制权值中非 0 系数的个数。

证明：由前面的式(7)可知，SSFRR 调度器在$[t_1, t_2]$时间段内为数据流 f_i 调度的信元数目满足

$$S_i(t_1, t_2) \geqslant \frac{w_i}{2^{T_d}}(t_2 = t_1) - C_i = \frac{r_i}{R}(t_2 = t_1) - C_i$$

上式的含义为：在任意时间段$[t_1, t_2]$内，SSFRR 调度器为数据流 f_i 调度的信元数目与对应的理想的 GPS 调度器相比不少于 C_i 个信元，即 SSFRR 调度器为数据流 f_i 提供 C_i 个信元的最坏情况公平指标。

3.3 SSFRR 算法的平滑性

由前面的叙述可知，SSFRR 算法的时隙分配过程是按照权值矩阵从最左列（第 $T_d - 1$ 列）到最右列（第 0 列）的顺序进行，即按照权值分量由大到小的顺序将数据流分布到完全二叉树中，并且每一层都是均匀地分布到左右子树中，保证每个数据流被调度的时隙间隔最多为 2^{m_i}，其中 m_i 为数据流 f_i 权值的二进制编码的最高非 0 位，因此 SSFRR 算法对每个数据流的调度都是平滑均匀的。

4 仿真结果

我们利用 OPNET 设计了仿真结构，如图 2 所示，其中 N1－N3 为采用公平调度算法的 OQ 交换节点，S1－S7 是发送数据流的源节点，每个节点最多可以产生 4 个不同类型、不同参数的数据流，都发往 D1 节点。这里发送和传输最大长度为 1500 字节的数据包，分段（信元）长度为 64kB。所有的数据链路速率都是 2Mbps，传输实验 200s。我们在 OQ 交换节点上分别使用了 SSFRR 算法、STRR 算法、FRR 算法、WF^2Q 算法和 DRR 算法，对这些算法的平均端到端时延、最坏情况端到端时延、短期吞吐率和公平性进行了比较。

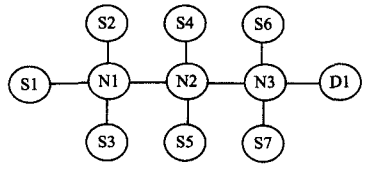


图 2 仿真网络拓扑

4.1 端到端时延

在实验中，从 N1 到 D1 有 10 个 CBR 流，速率分别为 10kbps，20kbps，40kbps，60kbps，80kbps，100kbps，120kbps，160kbps，200kbps，260kbps，另外从 N1、N2 和 N3 到 D1 有若干个指数分布的 ON/OFF 流作为背景流。首先设置这些背景流的速率，使总的数据流速率之和不超过链路容量，图 3 和图 4 分别为链路满载但不过载（即不存在恶意流）的情况下各种公平调度算法所测得的平均端到端时延和最大端到端时延。从图中可以看出，在链路流量不超载的情况下，各种调度算法都有较好的端到端时延属性，因为 SSFRR 调度算法的排队时延与时隙分配权值中包含的二进制 1 的位数 C_i 和预订速率 ρ_i 的比值(C_i/ρ_i)成正比，因此在低速率时有较大的最坏情况排队时延。

然而，SSFRR 算法有较强的隔离性能。在网络中存在恶意流的情况下，DRR 和 FRR 等调度算法的性能会明显降低，而 SSFRR 算法能保持较好的时延属性和公平属性。例如，将从源节点 S3 发出的两个背景流的预订速率设为 30kbps 和 40kbps，而它们的实际发送速率分别为 600kbps 和 800kbps，远远超过其预订速率，同时也使 N1 到 N2 节点的链路流量超载，图 5 和图 6 分别为在这种情况下各调度算法的平均端到端时延和最大端到端时延。可以看出，SSFRR 调度算法提供的时延性能几乎与基于时间戳的调度算法 WF^2Q 一致，而其调度复杂度仅为 O(1)。

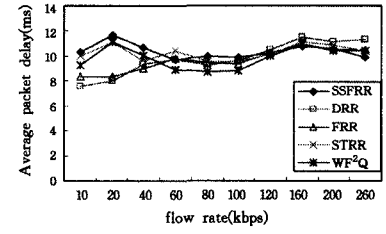


图 3 链路不超载情况下的平均端到端时延

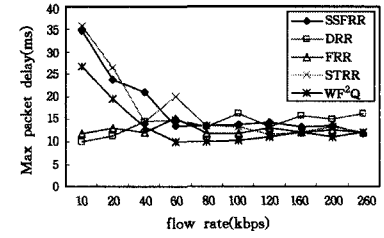


图 4 链路不超载情况下的最大端到端时延

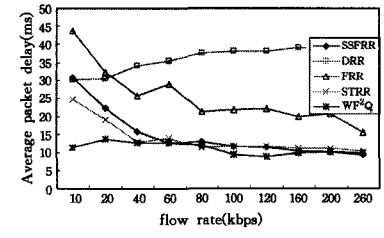


图 5 存在恶意流情况下的平均端到端时延

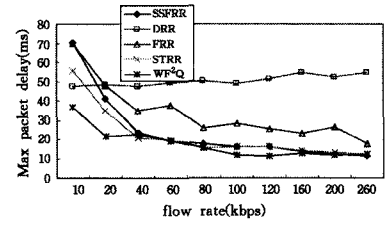
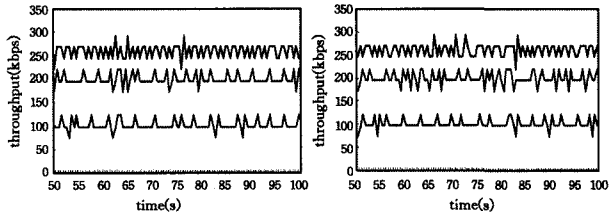


图 6 存在恶意流情况下的最大端到端时延

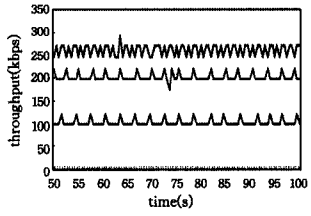
4.2 短期吞吐量和比例公平

图 7 给出了 SSFRR 和另外两种典型的公平调度算法 FRR 及 WF^2Q 的短期吞吐量。实验环境与前面的设置基本相同，只是将从源节点 S3 发出的一个背景流改为预定速率为

140kbps 的恶意流，我们跟踪了 3 个速率分别为 100kbps、200kbps 和 260kbps 的 CBR 流离开节点 N3 时的速率。在 FRR 调度算法中，按照其分类方法，200kbps 的流和 140kbps 的流属于同一类，在同一类内采用轮转调度，因此 140kbps 的恶意流严重影响了 200kps 流的性能。而 SSFRR 算法中每个数据流的性能不受其他数据流影响，只与数据流本身的参数有关，显示了较平滑的吞吐量，其虽然与理想的基于时间戳的 WF^2Q 算法还有一些差距，但优于分类轮转调度和其他轮转调度方法。



(a)SSFRR (b)FRR



(c)WF^2Q

图 7 短期吞吐量

图 8 给出了 SSFRR 算法的比例公平性。在这个实验中，4 个预定速率分别为 100kbps、200kbps、300kbps 和 400kbps 的流以两倍于它们预定的速率从源节点 S1 和 S2 发送至 D1。第 5 个数据流预定速率为 400kbps，在 S3 节点从第 10s 开始发送，在第 100s 时停止。第 6 个数据流预定速率为 600kbps，在 S3 节点从第 60s 开始发送，在第 140s 时停止。从图 8 可以看出，在 10～60s、60～100s、100～140s 和 140～190s 的时间段内，前 4 个数据流以 1:2:3:4 的比例分别共享了 1.6M、1M、1.4M 和 2M 的带宽，即它们获得的实际带宽与它们的预定带宽成比例。

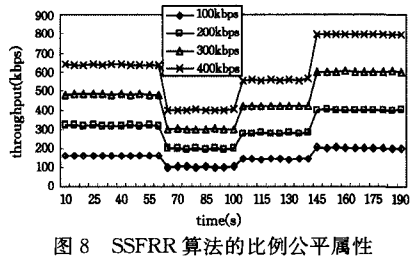


图 8 SSFRR 算法的比例公平属性

结束语 本文针对共享同一输出链路的数据流调度，提出一种能够保证数据流的 QoS 性能的公平调度算法 SSFRR，相对于众多的基于 OQ 的公平调度算法，SSFRR 具有简单易实现、调度复杂度为 O(1)的特点，这一特点使得 SSFRR 算法能被应用到带有交叉点缓存的 Crossbar(CICQ)交换结构的各个调度环节，使目前被广泛关注和使用的 CICQ 交换能够提供基于数据流的服务质量保证。另一方面，也可以将 SSFRR 应用于多资源共享的公平调度中，这些是我们下一步要做的工作。

参 考 文 献

[1] Demers A, Keshav S, Shenker S. Analysis and Simulation of a Fair Queuing Algorithm[C]// Proc. ACM SIGCOMM. 1989:1-13

[2] Parekh A, Gallager R. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case[J]. IEEE/ACM Trans. Networking, 1993, 1(3):344-357

[3] Bennett J, Zhang H. WF^2Q: Worst Case Fair Weighted Fair Queuing[C]// Proc. IEEE INFOCOM. 1996:120-128

[4] Bennett J, Zhang H. Hierarchical Packet Fair Queueing Algorithms[J]. ACM/IEEE Trans. Networking, 1997, 5(5):675-689

[5] Zhang L. Virtual Clock: A New Traffic Control Scheme for Packet Switching Networks[C]// Proc. ACM SIGCOMM. 1990:19-29

[6] Golestani S. A self-clocked fair queueing scheme for broadband applications[C]// Proc. IEEE INFOCOM'94. Toronto, ON, Canada, 1994:636-646

[7] Shreedhar M, Varghese G. Efficient Fair Queuing Using Deficit Round Robin[C]// Proc. ACM SIGCOMM. 1995:231-242

[8] Katevenis S S M, Courcoubeti C. Weighted round robin cell multiplexing in a general purpose ATM switch chip[J]. IEEE Journal on Selected Areas of Communications, 1991, 9:1265-1279

[9] Saha D, Mukherjee S, Tripathi S. Carry-over round robin: a simple cell scheduling mechanism for ATM networks[J]. IEEE/ACM Trans. Networking, 1998, 6:779-796

[10] Garg Ra-hul, Chen Xiao-qiang. RRR: Recursive round robin scheduler[J]. Computer Networks, 1999, 31(18):1951-1966

[11] Guo Chuan-xiong. SRR: An O(1) time complexity packet scheduler for flows in multi-service packet networks[J]. IEEE/ACM trans. Networking, 2004, 12(6):1144-1155

[12] Guo Chuan-xiong. G-3: An O(1) time complexity packet scheduler that provides bounded end-to-end delay[C]// Proc. Infocom. 2007:1109-1117

[13] Guo Chuan-xiong. Improved Smoothed Round Robin Schedulers for High-Speed Packet Networks[C]// IEEE INFOCOM 2008 Proceedings. 2008:1579-1587

[14] Yuan Xin, Duan Zhen-hai. Fair Round-Robin: A Low-Complexity Packet Scheduler with Proportional and Worst-Case Fairness [J]. IEEE Transaction on Computers, 2009, 58(3):365-379

[15] Ramabhadran S, Pasquale J. The Stratified Round Robin Scheduler: Design, Analysis, and Implementation [J]. IEEE/ACM Trans. Networking, 2006, 14(6):1362-1373

[16] Wittevrongel S, De Vuyst S, Sys C, et al. A reservation-based scheduling mechanism for fair QoS provisioning in packet-based networks[C]// 2014 26th International Teletraffic Congress (ITC). 2014, 1(8):9-11

[17] Checconi F, Rizzo L, Valente P. QFQ: Efficient Packet Scheduling With Tight Guarantees[J]. IEEE/ACM Transactions on Networking, 2013, 21(3):802-816

[18] Chang G, Lee C C. Fair queueing without per-flow queues: A Virtual Queueing Machine[C]// 2014 International Conference on Computing, Networking and Communications (ICNC). 2014:124-130