基于 CICQ 的新型并行交换结构研究

任涛，兰巨龙，扈红超，程东年
（国家数字交换系统工程技术研究中心，河南 郑州 450002）

摘 要：提出一种基于 CICQ 的支持区分服务的分布式并行分组交换结构（CDPPS, distributed CICQ based diffServ supporting parallel packet switch）。通过使用多个低速交换设备来构建支持区分服务的并行分组交换，CDPPS 解决了在高速环境下构建可行的区分服务调度问题。由于可以在不做任何改动的情况下将 CICQ 结构作为中间层平面，提高了现有资源的利用率。采用全分布式的调度算法，CDPPS 避免了系统的通信开销，降低了硬件实现复杂度。理论分析和实验仿真表明，CDPPS 能为各类业务提供满意的服务质量。

关键词：区分服务；分布式；并行分组交换；联合输入交叉点排队
中图分类号：TP393 **文献标识码**：A **文章编号**：1000-436X(2010)10-0098-10

Research on new kind of parallel packet switch based on CICQ

REN Tao, LAN Ju-long, HU Hong-chao, CHENG Dong-nian
(National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China)

Abstract: A distributed CICQ based DiffServ supporting parallel packet switch(CDPPS) was proposed. By using multiple low speed switches to construct parallel packet switch, CDPPS made it possible to construct DiffServ supporting scheduling scheme in high speed environment. Because CICQ could be used as central plane without any modification, CDPPS increased the efficiency of existing resources. As a result of using distributed scheduling algorithm, CDPPS eliminates the communication overhead, and reduces the hardware implementation complexity. Theoretic analysis and simulation results show that CDPPS can provide satisfying QoS for different traffic classes.

Key words: DiffServ; distributed; PPS; CICQ

1 引言

互联网业务的发展经历了若干个阶段：在 20 世纪，互联网上运行的主要是 HTTP、Telnet 和 E-mail 等简单的应用；在 21 世纪的前 10 年里，互联网上的业务经历了一次爆炸式的发展，出现了许多新型的应用，如即时通信、网络视频、网络购物和网上银行等；而从现在往后，互联网上业务的种类将更加丰富，正如时下的“三网融合”和“物联网”等所要实现的那样，手机、电话、家用电器和各种电子产品都将会被接入到网络中。而作为网络的一个重要组成部分，交换设备在这个过程中也发挥着重要的作用。交换设备的端口数已经由 16 端口发展到 64 端口、128 端口，甚至更高；交换设备的端口速率也由 1Gbit/s 发展到 10Gbit/s、40Gbit/s、160Gbit/s 等。长期以来，交换设备一直在朝着高速率、大容量的方向飞速发展。然而随着互联网的发展和各种新型应用的出现，一味地追求交换的高速率、大容量并不能保证各种业务获得满意的服务。

为此，IETF 先后提出综合服务(IntServ, integrated services)[1]和区分服务 (DiffServ, differentiated services)[2]2 种服务质量模型。最早被提出的

收稿日期：2010-06-29；**修回日期**：2010-09-27
基金项目：国家重点基础研究发展计划（“973”计划）基金资助项目（2007CB307102）
Foundation Item: The National Basic Research Program of China (973 Program) (2007CB307102)

IntServ 模型虽然能够提供满意的服务质量，但由于它是针对用户流的调度，需要为每条用户流维护大量的状态信息，可扩展性较差。因此，IETF 接着又提出了 DiffServ 模型。DiffServ 模型将 DiffServ 域中复杂的功能分离到边缘路由器中，从而使得核心路由器变得尽可能简单。边缘路由器为所有的用户流维护状态信息，并根据 SLA 协议将进入 DiffServ 域的数据划分为相应的业务类；不同的业务类具有不同的单跳行为(PHB, per hop behavior)，通过数据包头中的区分服务码点 (DSCP, differentiated service code point)[3]来表示；核心路由器只需要根据 PHB 对各类业务进行转发。目前 DiffServ 模型中定义了 3 类 PHB：快速转发(EF, expedited forwarding)[4]、确保转发(AF, assured forwarding)[5]和尽力而为(BE, best effort)。EF 主要用于在 DiffServ 域内提供低时延、低抖动、低丢包和具有带宽保证的端到端服务。AF 用于提供具有最低速率保障和低丢包率服务。其中，AF 内部又定义了 4 个子类，分别为 AF1、AF2、AF3 和 AF4；每个子类中按照丢包策略又可分为 3 个优先级。BE 提供的是尽力而为的服务，无法获得任何服务保障。

PHB 的实现主要依赖于路由器交换机上的调度算法和排队策略。现有的基于输出排队(OQ, output queuing)交换结构的 DiffServ 调度算法主要有优先级排队(PQ, priority queuing)[5]、加权轮询服务(WRR, weighted round-robin)[6]、联合优先级排队加权轮询(PQWRR, priority queuing weighted round-robin)[7]和带优先级的自适应加权公平排队(AWFQP, adaptive weighted fair queuing with priority)[8]等。相比于 PQ 和 WRR，PQWRR 通过在 EF 类业务和非 EF 类业务之间使用优先级调度，在 AF 类业务和 BE 类业务之间使用加权轮询调度，使得在为 EF 类业务提供低时延和低抖动的情况下，为 AF 类业务和 BE 类业务提供了更好的带宽分配。相比较于 PQWRR，AWFQP 能够在不损失 EF 业务性能的前提下，提高 AF 和 BE 业务的性能。虽然这些算法吞吐率高、复杂度低，易于实现，但由于算法本身基于 OQ 结构，受到 N 倍加速问题的限制，难以在高速环境下实现。

因此，基于输入排队(IQ, input queuing)交换结构的 DiffServ 调度算法受到了广泛的关注和研究。典型的算法包括动态区分服务调度(DDS, dynamic diffserv scheduling)[9]和等级区分服务调度(HDS, hierarchical diffserv scheduling)[10]等。虽然通过采用迭代方式逼近最大匹配可以在一定程度上降低算法复杂度，但它们仅能在均匀业务条件下获得较好的性能。对于非均匀业务，DDS 和 HDS 算法都会导致 IQ 交换结构吞吐量降低，当业务负载较重时，其时延性能将急剧恶化。因此基于 IQ 交换结构支持 DiffServ 模型的调度算法在实际应用中也存在着很大的局限性。由于芯片设计技术的进步和芯片工艺水平的提高，联合输入/交叉节点排队(CICQ, combined input-crosspoint-queued)交换结构获得了广泛的研究和应用，也出现了基于 CICQ 结构的 DiffServ 调度算法，如分布式区分服务调度(DDSS, distributed diffserv supporting scheduling)[11]。DDSS 算法采用全分布式的调度机制，可以并行工作于交换结构的各个输入输出端口，算法复杂度仅为 $O(\log N)$；且对比于 PQWRR 和 DDS，DDSS 具有良好的时延性能和公平特性，能够更好地支持 DiffServ 模型。

近年来，链路速率飞速发展，然而动态存储器 DRAM 的速率却发展缓慢。无论是 IQ 交换结构还是 CICQ 交换结构，其内部存储器至少工作于链路速率。然而，目前主流 DRAM 的随机访问频率为 100MHz 左右，巨大的速度差导致无法使用 DRAM 在一个时隙（链路传输一个信元所需要的时间）内完成一个信元的写入或读出操作。因此 DDS、HDS 和 DDSS 等 DiffServ 调度算法的应用也受到了巨大的限制。本文提出了一种基于 CICQ 的支持 DiffServ 的分布式并行分组交换结构(CDPPS, distributed CICQ based diffserv supporting parallel packet switch)，以解决在链路速率和存储器速率发展严重差异的情况下，支持各类业务的区分服务。CDPPS 的解复用器使用了负载均衡的分布式信元分派算法，保证了业务流在各平面间的均匀分布，从而提高了系统的整体吞吐率；CDPPS 的中间层平面基于已经获得广泛生产和应用的联合输入交叉点排队（CICQ, combined input crosspoint queued）交换结构；CDPPS 的复用器使用了支持 DiffServ 的分布式信元保序调度算法，既保证了业务流的按序输出，也为各类业务提供了区分服务。 而且整个交换结构采用全分布式的调度机制，各个算法独立工作于各个部件，消除了系统的通信开销。为了进一步验证 CDPPS 的性能，本文通过仿真对 PQWRR、DDS、DDSS 和 CDPPS 的公平性和时延性能进行了比较，

仿真结果表明，CDPPS 算法始终具有良好的时延性能和公平特性，能够更好地在高速环境下支持 DiffServ 模型。

2 交换结构

CDPPS 采用了基于 CICQ 的并行交换结构（如图 1 所示）主要由解复用器、中间层平面和复用器 3 个部分组成。这里 CDPPS 采用了分布式交换结构，不需要任何集中式调度器。解复用器和复用器工作于外部链路速率 R，中间层平面和内部链路工作于内部链路速率 r，且内部加速为 S 时，$r=SR/L$（L 为中间层平面的数目）。以速率 R 传输一个信元的时间为外部时隙（后面部分如不做特别说明，时隙均代表外部时隙）；以速率 r 传输一个信元的时间为内部时隙。每个解复用器分布独立运行，将到达的业务流（具有相同的输入端口、目的端口和业务类型的流）均匀分发到各个中间层平面。中间层平面采用了流行的 CICQ 交换结构。每个中间层平面也是分布独立运行，每隔 L 个时隙运行一次平面调度算法，并根据算法结果将平面输入端的信元交换到输出端。每个复用器分布独立运行，对来自各个平面输出端的信元进行重组，在满足保序的前提下，对各个业务流进行区分调度并输出。

交换结构的相关定义如下：连接第 i 个输入端口的解复用器记为 D_i；连接第 j 个输出端口的复用器记为 M_j；第 l 个中间层平面记为 S_l；从输入端口 i 发往输出端口 j 的第 k 类业务的流记为 TF_{ijk}。

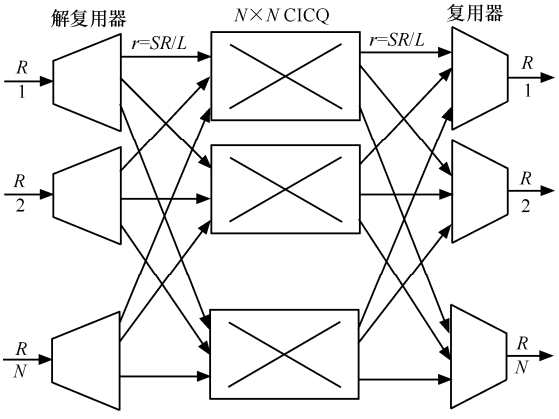


图 1 CDPPS 体系结构

3 调度算法

CDPPS 主要包括了运行于解复用器中的信元分派调度算法、运行于中间层平面的 SQF-LQF 调度算法和运行于复用器中的信元重组调度算法。各个算法分布独立运行，相互之间不需要通信。

3.1 负载均衡的分布式信元分派算法

解复用器的内部结构如图 2 所示，每个解复用器维护了 L 个 FIFO 队列，用于缓存因内部链路争用而无法被直接发送的信元。该队列的定义如下。

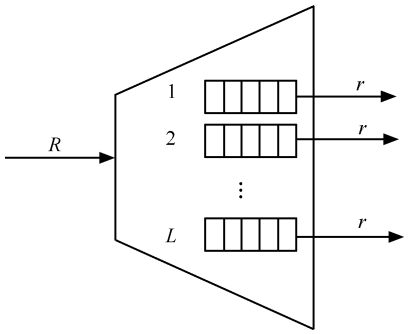


图 2 CDPPS 解复用器结构

PQ_{il}：位于 D_i 中的第 l 个队列，用于缓存发往 S_l 的信元，$1\leqslant i\leqslant N$，$1\leqslant l\leqslant L$。

每当解复用器 D_i 从输入端口接收到一个新到达的信元 C_{ijk} 时，通过该信元头部的 DSCP 值可以辨别出其所属的业务流 TF_{ijk}。解复用器内部维护了 K 组指针 P_{jk}^i，$1\leqslant j\leqslant N$，$1\leqslant k\leqslant K$，i 代表了当前的解复用器号。每一个指针 P_{jk}^i 代表了当业务流 TF_{ijk} 的下一个信元到达时，应当被分派到的中间层平面号。因此，当信元 C_{ijk} 到达时，假设 $P_{jk}^i=l$，它会被分派到平面 S_l 中（由于存在内部输入链路争用，会被暂时缓存于队列 PQ_l），同时 $P_{jk}^i=(l+1)\%L$。

该算法的功能是将每一个业务流 TF_{ijk} 的信元均匀分发到各个中间层平面，提高中间层平面交换结构和存储器的利用率，从而提高整个系统的吞吐率。算法的复杂度为 $O(1)$，易于高速环境下的硬件实现。

3.2 CICQ 平面调度算法

中间层平面的结构如图 3 所示，每个输入端口 i 维护了 N 个队列，用于缓存发往 N 个输出端口的信元，记为 VOQ_{ij}，$1\leqslant i,j\leqslant N$；在交换结构内部，每个输入端口 i 和输出端口 j 交汇处的队列用于缓存从输入端口 i 发往输出端口 j 的信元，记为 CB_{ij}，$1\leqslant i,j\leqslant N$；输出端口 j 处的队列，称为输出队列，记为 O_j，$1\leqslant j\leqslant N$。

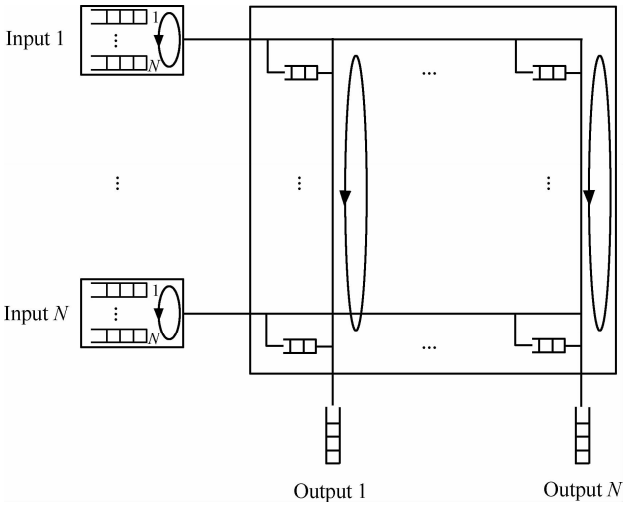


图 3 中间层平面交换结构

定义 1　长度为 N 的向量 V_{voq}^{i}，其第 j 个元素 $V_{voq}^{i}(j)$ 的值是一个布尔变量 b_j，若输入 i 的第 j 个 VOQ 队列为空，则 b_j 的值为 0，反之则为 1，则称 V_{voq}^{i} 为输入 i 的虚拟输出状态向量。

每一个平面并行独立地进行调度。这里每个平面使用了分布式负载均衡的交叉点调度算法 SQF-LQF。每个输入端口和每个输出端口都包含了一个独立的调度器。

定义 2　位于输入端口 i 处的调度器，负责从当前输入端口中 N 个 VOQ 队列的头部选择一个信元发送到相应的交叉点缓存中，称该调度器为 i 输入端口调度器，记为 IS_i。

定义 3　位于输出端口 j 处的调度器，负责从当前输出端口所在列的 N 个交叉点队列的头部选择一个信元发送到输出端口 j 队列中，称该调度器为 j 输出端口调度器，记为 OS_j。

每个输入调度器和每个输出调度器为了能够正常运行，分别维护了一个行状态向量和一个列状态向量，其定义分别如下。

定义 4　长度为 N 的向量 V_{row}^{i}，其第 j 个元素 $V_{row}^{i}(j)$ 的值为当前中间层平面位于输入 i 和输出 j 交叉点处的队列长度，则称向量 V_{row}^{i} 为调度器 IS_i 的行状态向量。

定义 5　长度为 N 的向量 V_{col}^{j}，其第 i 个元素 $V_{col}^{j}(i)$ 的值为当前中间层平面位于输入 i 和输出 j 交叉点处的队列长度，则称向量 V_{col}^{j} 为调度器 OS_j 的列状态向量。

SQF-LQF 算法步骤包括 2 个部分。

1) 输入调度

N 个输入调度器并行独立地执行输入调度。对于任意的调度器 IS_i，输入调度过程为：首先对输入 i 的虚拟输出状态向量和行状态向量进行点乘运算，得到输入 i 的输入调度向量 $V_{IS}^{i}=V_{voq}^{i}\ V_{row}^{i}$；其次遍历 V_{IS}^{i}，寻找值最小的非零成员 $V_{IS}^{i}(j)$（如果有多个相等的最小值，则取第一个最小值）；最后将 VOQ_{ij} 的队头信元输出到 CB_{ij} 中。

2) 输出调度

N 个输出调度器并行地执行输出调度。对于任意的调度器 OS_j，输出调度过程为：遍历列状态向量 V_{col}^{j}，寻找值最大的非零成员 $V_{col}^{j}(i)$（如果有多个相等的最小值，则取第一个最大值）；将 CB_{ij} 的队头信元输出到 O_j 中。

在上面的输入调度和输出调度中，都需要进行遍历寻找最大或最小值的过程。对于一个 $N\times N$ 的交换平面，该操作的复杂度为 $O(N)$。为了进一步降低平面调度算法的复杂度，从而提高整个交换系统在高速环境下的可行性，本文引入了基于堆的平面调度算法 HSQF-HLQF。

定义 6　对输入端口 i 的输入调度向量 $V_{IS}^{i}=V_{voq}^{i}\cdot V_{row}^{i}$ 中的所有成员使用堆排序算法进行排序，得到最小堆 H_{IS}^{i}，称为输入端口 i 的输入调度堆；该堆中代表了交叉点 CB_{ij} 队列状态的堆成员记为 $H_{IS}^{i}(j)$。

定义 7　对输出端口 j 的列状态向量 V_{col}^{j} 使用堆排序算法进行排序，得到最大堆 H_{col}^{j}，称为输出端口 j 的输出调度堆；该堆中代表了交叉点 CB_{ij} 队列状态的堆成员记为 $H_{col}^{j}(i)$。

虽然建立 H_{IS}^{i} 和 $H_{col}^{j}(i)$ 这 2 个堆所使用的排序操作的复杂度为 $O(N\log N)$，但每个平面建立 H_{IS}^{i} 和 $H_{col}^{j}(i)$ 只需要在整个交换系统初始时运行一次，所以建立 H_{IS}^{i} 和 $H_{col}^{j}(i)$ 的操作不影响整个交换系统的调度复杂度。

定义 8　从堆 H 的所有元素中查找最大值（最小值）的过程，称为堆 H 的查找最大值（最小值）操作，记为 Max(H)（Min(H)）。

定义 9　从堆 H 的所有元素中查找值为 V 的成员的过程，称为堆 H 的查找操作，记为 $F(V)$。

定义 10　对堆 H 中某个成员的值进行修改后，为了使 H 继续成为堆而进行的堆成员调整操作，称

为堆 H 的维护操作，记为 $M(H)$。

HSQF-HLQF 算法步骤包括 2 个部分。

1) 输入调度

N 个输入调度器并行独立地执行输入调度。对于任意的调度器 IS_i，输入调度过程为：首先获取 H_{IS}^i 的堆顶元素 $H_{IS}^i(j)$，将 VOQ_{ij} 的队头信元输出到 CB_{ij} 中；如果输出了队头信元后 VOQ_{ij} 的队列变为空，则将堆顶元素 $H_{IS}^i(j)$ 从 H_{IS}^i 中删除后执行堆维护操作 $M(H_{IS}^i)$；如果输出了队头信元后 VOQ_{ij} 的队列非空，则将堆顶元素 $H_{IS}^i(j)$ 的值加 1 后执行堆维护操作 $M(H_{IS}^i)$；对输出调度堆 H_{col}^j 执行 $F(H_{IS}^i(j))$操作，找到堆成员 $H_{col}^j(i)$；将 $H_{col}^j(i)$ 的值加 1 后执行堆维护操作 $M(H_{col}^j)$。

2) 输出调度

N 个输出调度器并行独立地执行输出调度。对于任意的调度器 OS_j，输出调度过程为：首先获取 H_{col}^j 的堆顶元素 $H_{col}^j(i)$，将 CB_{ij} 的队头信元输出到 O_j 中；将堆顶元素 $H_{col}^j(i)$ 的值减 1 后执行堆维护操作 $\mathrm{M}(H_{col}^j)$；对输入状态调度堆 H_{IS}^i 执行 $\mathrm{F}(H_{col}^j(i))$ 操作，找到堆成员 $H_{IS}^i(j)$；将 $H_{IS}^i(j)$ 的值减 1 后执行堆维护操作 $\mathrm{M}(H_{IS}^i)$。

由于 Max(H)和 Min(H)操作的复杂度为 $O(1)$，$F(V)$操作的复杂度为 $O(\log N)$，M(H)操作的复杂度为 $O(\log N)$，所以 HSQF-HLQF 算法输入调度的复杂度为 $O(\log N)$，输出调度的复杂度为 $O(\log N)$。由于中间层平面每隔 L 个时隙执行一次调度算法，所以 HSQF-HLQF 算法更有利于高速环境下的实现。

3.3 支持 DiffServ 的分布式信元保序重组算法

CDPPS 复用器的内部结构如图 4 所示，每个复用器维护了 L 块缓存，每块缓存对应于来自某个中间层平面的数据；每个缓存又被分为 N 个虚拟输入队列，每个虚拟输入队列缓存来自某个输入端口的数据；为了支持区分业务的调度，每个虚拟输入队列在逻辑上又被分为 K 个虚拟输入优先级队列（VIPQ, virtual input priority queue），用于缓存属于同一业务类型的信元。虚拟输入优先级队列的定义如下。

定义 11 $VIPQ_{ijlk}$：位于 M_j 中的虚拟输入优先级队列，用于缓存来自第 i 个输入端口，并经平面 S_l 交换过来的第 k 类业务的信元，$1\leqslant i,j\leqslant N$，$1\leqslant l\leqslant L$，$1\leqslant k\leqslant K$。

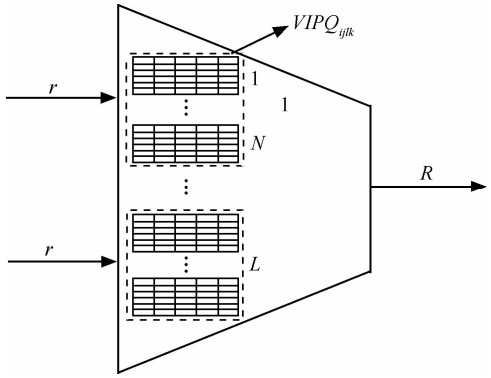


图 4 CDPPS 复用器结构

根据 DiffServ 模型的定义，峰值信息速率(PIR, peak information rate)代表了 EF 类业务所能获得的最高带宽，承诺信息速率(CIR, committed information rate)代表了各个 AF 类业务所能获得的最小带宽，分别用 CIR1、CIR2、CIR3 和 CIR4 表示。为了支持 DiffServ，CDPPS 的复用器调度算法采用了基于预约带宽的调度策略；为此，定义各类业务的预约带宽分别为 $RB_1=1.1\times PIR$，$RB_k=CIR(k-1)$，k=2,3,4,5。这里将 EF 业务的预约带宽设置为 PIR 的 1.1 倍是为了更好地保证 EF 业务的性能。

复用器 M_j 维护了 K 个计数器 C_{jk}，$1\leqslant k\leqslant K$，用于表示每类业务已经获得服务的信元数目。同时为了保证每条业务流的保序输出，M_j 还设置了 K 组平面指针 P_{ik}^j，$1\leqslant i\leqslant N$，$1\leqslant k\leqslant K$；每一个指针代表了为保证按序输出，业务流 TF_{ijk} 的下一个输出信元应当从哪一个平面中读取；由于平面转发过来的信元被缓存在队列 VIPQ 中，该指针也就代表了应当从哪一个队列中读取业务流 TF_{ijk} 的下一个信元。为了使得来自不同输入端口的业务流平等获得输出的机会，M_j 还为每类业务设置了一个流指针 F_k，用来表示该业务下次输出时首先尝试的输入端口号。

具体的复用器调度过程如下：在任意时刻 t，复用器读取 EF 和各个 AF 业务调度器内的计数器 C_{jk}（$1\leqslant k\leqslant K-1$），并按照优先级从高到低的顺序依次判断 C_{jk} 是否满足 $C_{jk}/t<RB_k$。如果满足，则从流指针 F_k 开始依次轮询遍历所有的输入端口号 i，并判断每一个平面指针 P_{ik}^j 所指向的 VIPQ 队列 $VIPQ_{ijlk}$（假设 $l=P_{ik}^j$）是否为空；如果不为空，则输出 $VIPQ_{ijlk}$ 的队头信元，并且 $P_{ik}^j=(P_{ik}^j+1)\%L$，$F_k=(F_k+1)\%N$，$C_{jk}=C_{jk}+1$，完成当前时刻的调度。如果遍历了所有的 k（$1\leqslant k\leqslant K-1$）都不满足 $C_{jk}/t<\mathrm{RB}_k$，则首先让 AF 调度器和 BE 调度并行地在各

自业务流对应的队列集合 $VIPQ_{ijlk}(l=P_{ik}^{j}, i=1,2,\cdots,N)$ 中选取出一个队头信元等待时间最长的队列，接着再从这 $K-1$ 个队列中选出一个队头信元等待时间最长的队列，输出该队列的队头信元，并更新该队列所对应的平面指针 P_{ik}^{j}、流指针 F_k 和计数器 C_{jk} 的值。如果出现多个队列的队头信元最有相同的最长等待时间，则按轮询方式选择一个。

通过上面对 CDPPS 的交换结构和调度算法的描述，可以发现解复用器和中间层平面没有为不同的业务流设置单独的逻辑队列，也没有对各类业务进行区分服务的调度，只是在复用器中为各类业务设置了单独的逻辑队列并执行了基于预约带宽的区分服务调度。下面就证明这样做并不会影响各类业务所获得的服务质量。下面首先给出一个定义。

定义 12　CDPPS'：将不支持区分服务的 CDPPS 定义为 CDPPS'，即在 CDPPS 输出端的每个复用器中，取消为不同业务设置的逻辑队列和区分服务调度。

引理 1　CDPPS'结构可以模仿一个 FCFS-OQ 交换结构。

证明　由于使用 SQF-LQF 作为调度算法的 CICQ 结构能够模仿一个 FCFS-OQ 结构。因此，使用 CICQ 作为中间层平面的 CDPPS'就可以看做是使用 FCFS-OQ 作为中间层平面的分布式并行分组交换结构。而根据文献[12]，使用 FCFS-OQ 结构作为中间层平面的分布式并行分组交换结构可以模仿一个 FCFS-OQ，因此，CDPPS'结构可以模仿一个 FCFS-OQ 交换结构。

定理 1　仅在 CDPPS 的复用器中对各类业务执行区分服务的调度，不会影响各类业务的性能。

证明　根据上面对 CDPPS'的定义，在 CDPPS 的复用器中执行对各类业务的区分服务调度，可以看作是在 CDPPS'的每个输出端口加入对业务的区分服务调度。根据引理 1，由于 FCFS-OQ 交换结构是无阻塞的交换，因此 CDPPS'也可以看做能够提供无阻塞的交换。在 CDPPS'的复用器中加入对各类业务的区分服务调度，也就相当于在一个无阻塞交换结构的输出端加入对业务的区分服务支持，不会影响各类业务的性能。

可见，由于中间层平面不需要为不同的业务流设置单独的逻辑队列，也不需要对各类业务执行区分服务的调度，因此原有的 CICQ 交换结构可以在不做任何改动的情况下直接被用作中间层平面；而 CICQ 结构已经在商业领域获得了广泛的生产和应用，因此使用流行的 CICQ 结构可以提高已有资源的利用率，降低整个系统的实现成本。

3.4　性能分析

根据最新的网络监测数据[13]，当间隔时间小于 100ms 时，每一个业务流可以看作泊松过程到达，并且不同业务流的到达过程相互独立。

定义 13　业务流 TF_{ijk} 建模为泊松过程 P_{ijk}，其到达速率为 α_{ijk}；输入端口 i 的业务到达过程建模为泊松过程 P_{ik}，其到达速率为 α_{ik}；输出端口 j 的业务到达过程建模为泊松过程 P_{jk}，其到达速率为 α_{jk}。因此，

$$\alpha_{ik}=\sum_{j=1}^{N}\alpha_{ijk}\ ,\alpha_{jk}=\sum_{i=1}^{N}\alpha_{ijk}\ \forall k\in[1,K]\ i,j\in[1,N]$$

引理 2　解复用器 D_i 中的队列 PQ_{il} 可以建模为 M/D/1 系统。

证明　按照前面的解复用器信元分派算法，每一个业务流 TF_{ijk} 被循环分派到各个平面，因此缓存到队列 PQ_{il} 中的 TF_{ijk} 业务流的到达过程可以近似为具有到达速率 α_{ijk}/L 的泊松过程；而队列 PQ_{il} 同时缓存了发往所有输出端口的业务流，因此队列 PQ_{il} 可以近似看作是具有到达速率 $\sum_{j=1}^{N}(\alpha_{ijk}/L)=\alpha_{ik}/L$ 的泊松过程。同时由于与 PQ_{il} 相连的内部输入链路的速率为 r，即内部输入链路以固定的速率 r 从 PQ_{il} 中读取数据，因此 PQ_{il} 可以看作是具有负载 $\frac{\alpha_{ik}}{Lr}$ 的 M/D/1 系统。

引理 3　中间层平面可以建模为 M/D/1 系统。

证明　由于 CICQ 结构可以模仿 FCFS-OQ 结构，这里将中间层平面 $CICQ_l$ 近似为 $OQ_l^{'}$ $(1\leqslant l\leqslant L)$。$OQ_l^{'}$ 的输入过程是所有输入端口的 PQ_{il} 队列输出的汇合；因此 $OQ_l^{'}$ 的输入可以看作是具有到达速率 $\sum_{i=1}^{N}(\alpha_{ijk}/L)=\alpha_{jk}/L$ 的泊松过程；由于与中间层平面输出相连的内部输出链路具有固定的速率 r，因此中间层平面可以建模为具有负载 $\frac{\alpha_{jk}}{Lr}$ 的 M/D/1 系统。

定理 2　EF 业务的平均时延为 $\frac{\alpha_{ik}^2}{2L^2r^2-2Lr\alpha_{ik}}+\frac{\alpha_{jk}^2}{2L^2r^2-2Lr\alpha_{jk}}+2L$。

证明 由于 EF 业务在复用器中具有最高的优先级，因此 EF 业务在复用器中时延可以忽略。EF 业务的时延主要由 3 部分组成：在解复用器队列 PQ 中的排队时延 D1、在中间层平面的排队时延 D2 和在内部链路中的传输时延 D3；根据引理 2 和引理 3，D1 为 $\left(\frac{\alpha_{ik}}{Lr}\right)^2 / 2\left(1-\frac{\alpha_{ik}}{Lr}\right)$，D2 为 $\left(\frac{\alpha_{jk}}{Lr}\right)^2 / 2\left(1-\frac{\alpha_{jk}}{Lr}\right)$，内部链路传输时延 D3 为为固定值 $2L$。因此 EF 业务的平均时延为 $\frac{\alpha_{ik}^2}{2L^2r^2-2Lr\alpha_{ik}}+\frac{\alpha_{jk}^2}{2L^2r^2-2Lr\alpha_{jk}}+2L$。

4 仿真分析

本节分别对基于 OQ 结构的 PQWRR、基于 IQ 结构的 DDS、基于 CICQ 结构的 DDSS 和基于并行交换结构的 CDPPS 的公平性和性能进行了实验仿真分析。仿真结果表明，CDPPS 的公平性优于 PQWRR，与 DDS 和 DDSS 相近；CDPPS 的 AF 业务性能优于 DDS，接近于 DDSS；尤其是在非均匀业务流条件下，相对于 DDS，使用基于并行交换结构的 CDPPS 对 AF 业务性能的提升更加明显。

4.1 实验环境及相关参数

使用 C++面向对象技术开发了模拟仿真交换环境。开发的模型包括基于 OQ 结构的 PQWRR、基于 IQ 结构的 DDS、基于 CICQ 结构的 DDSS 和基于并行交换结构的 CDPPS。突发流量模型的业务到达过程通过 ON-OFF 模型产生，突发长度服从几何分布，平均突发长度为 32。非均匀流的目的端口分布服从 $\lambda_{i,i}=0.67\rho$、$\lambda_{i,i+1}=0.33\rho$ 分布（ρ 为业务负载）。由于 Internet 流量具有突发特性，突发流量模型更接近于真实的网络流量，所以这里仅使用了突发流量模型进行实验。每个输入端口 EF、AF1、AF2、AF3、AF4 和 BE 业务的比例依次为 18%、24%、20%、16%、12%和 10%。预约带宽 $RB_k(k=1,2,\cdots,5)$ 分别为 19.8%、24%、20%、16%和 12%。

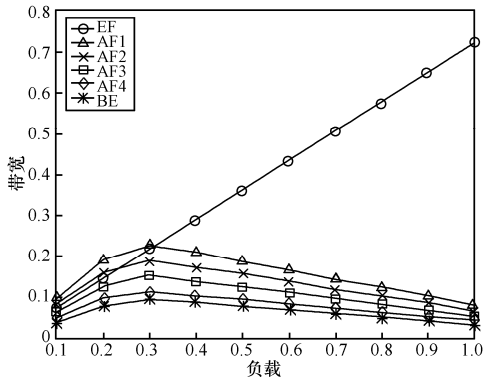
下面分别进行了 2 种实验。第 1 种是在业务过载的情况下测试各种结构和算法的公平性；与文献[9]和文献[11]相同，业务过载的设置为：采用了 4×4 的交换结构，并且所有输入端口的业务都以第 1 个输出端口为目的端口。第 2 种是在突发均匀和突发非均匀业务流下分别比较各种结构和算法的业务时延和时延抖动性能，交换结构为 16×16。

在所有的实验中，模拟时间均为 1 000 000 个时隙；CICQ 的交叉点队列大小为 8；CDPPS 的中间层平面的数目 L 为 10。

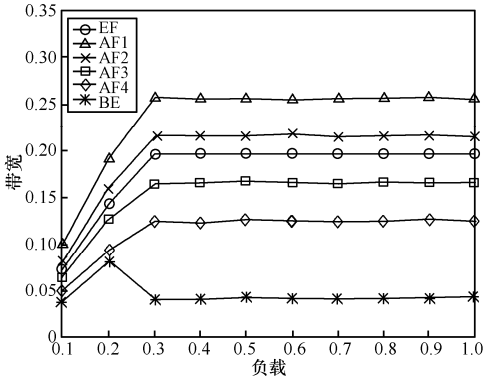
4.2 仿真结果分析

图 5 分别给出了 PQWRR、DDS、DDSS 和 CDPPS 在业务过载的情况下，各类业务的带宽占用情况。由图 5（a）可知，采用 PQWRR 时，过载的 EF 业务会抢占其他业务的带宽，从而严重影响了 AF 业务的服务质量；而 DDS、DDSS 和 CDPPS 保证了带宽在各类业务间的合理分配，具有良好的公平性，从而既使得 EF 和 AF 业务可以获得满意的服务质量，也避免了 BE 业务出现饿死的现象。

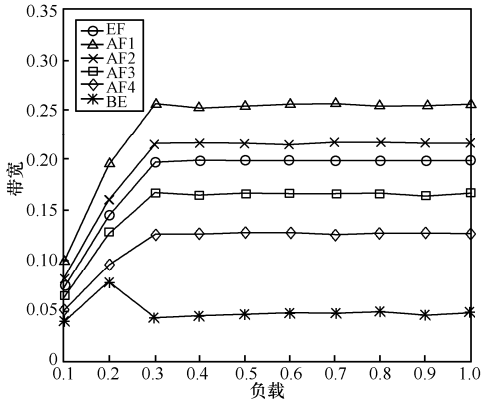
图 6 分别给出了 PQWRR、DDS、DDSS 和 CDPPS 在突发均匀业务流下，EF 业务、AF1 业务、AF3 业务、BE 业务的平均时延。由图 6(a)可见，CDPPS 的 EF 平均时延明显高于其他三者，这主要归结于 2 方面的原因：①CDPPS 的内部输入链路和内部输出链路工作于内部链路速率，这 2 条链路传输的时延为 $2L$ 个时隙；②CDPPS 的内外链路速率差所导致的解复用器和复用器排队时延，这部分时延随负载的增加会略微变大。所以 CDPPS 的 EF 平均时延比其他的大二十几个时隙，并且在负载较高时这个差值略有增大。在图 6(b)和图 6(c)中，PQWRR 的 AF 平均时延性能在负载接近于 1 时骤降，这是因为 PQWRR 在进行业务调度时使用的权值是队列长度，而不是队头信元的等待时延；这也就导致了图 6(d)中所示的 PQWRR 的 BE 时延在负载接近于 1 时优于其他三者。图 6(b)和图 6(c)表明，CDPPS 的 AF1、AF3 性能在负载较低时不如 DDS，而在负载分别高于 0.7 和 0.8 之后，CDPPS 的 AF1、AF3 性能逐渐优于 DDS。原因是低负载时 CDPPS 的内部链路传输时延在整个交换时延中占较大比重；而在高负载时，大量的信元在 IQ 结构的输入队列中排队，导致 DDS 的 AF 时延增加；而由引理 1 可知，CDPPS 可以模仿无阻塞 FCFS-OQ 交换结构，所以其 AF 性能并未受明显影响。图 7 给出了 EF 业务在突发均匀业务流负载为 0.7 时的时延抖动分布。可见，无论是平均时延还是时延抖动性能，CDPPS 总是低于 DDSS。因为，虽然 CICQ 和 PPS 都可以模仿 FCFS-OQ 交换结构，但 CICQ 的内部存储器和交换结构都工作于外部链路速率，而 CDPPS 的中间层交换平面和内部链路都工作于内部链路，因此 CDPPS 更适合于高速环境下的应用。



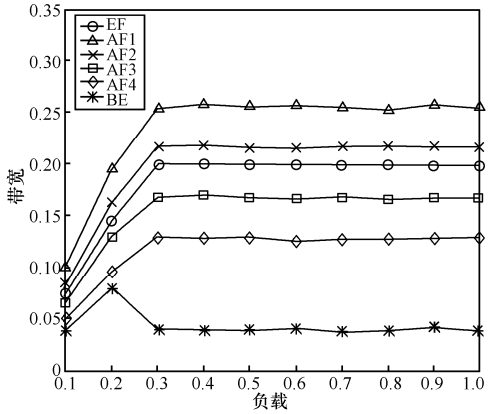
(a) PQWRR 各类业务带宽占用情况



(b) DDS 各类业务带宽占用情况

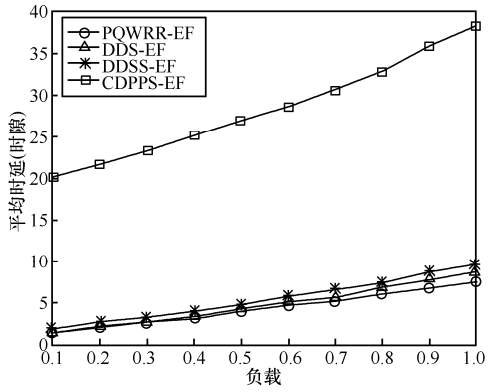


(c) DDSS 各类业务带宽占用情况

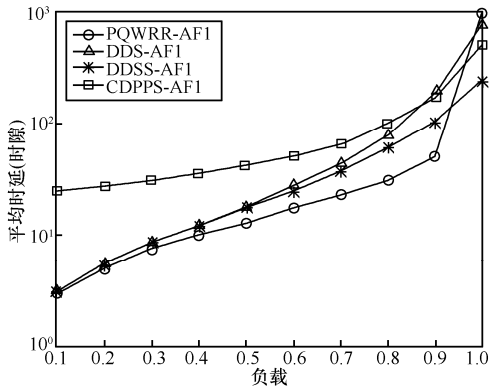


(d) CDPPS 各类业务带宽占用情况

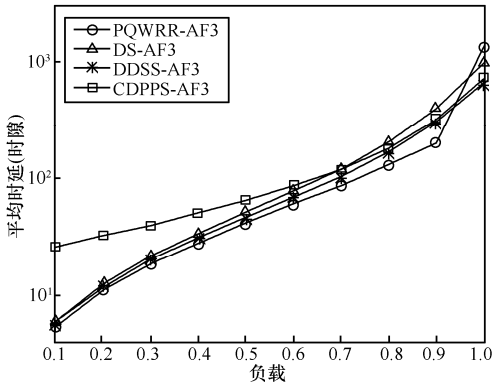
图 5　各类业务的带宽占用情况



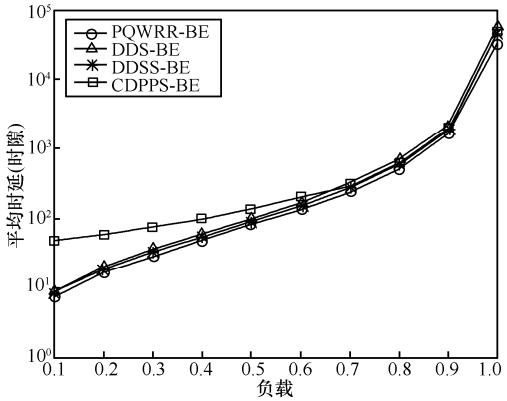
(a) 突发均匀业务流下 EF 的平均时延



(b) 突发均匀业务流下 AF1 的平均时延



(c) 突发均匀业务流下 AF3 的平均时延



(d) 突发均匀业务流下 BE 的平均时延

图 6　突发均匀业务流下各种业务的平均时延

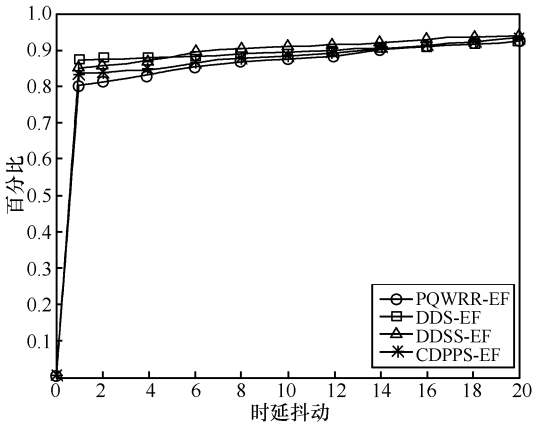
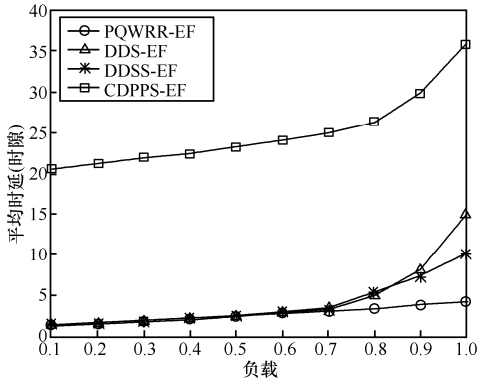


图 7 突发均匀业务流下 EF 的时延抖动

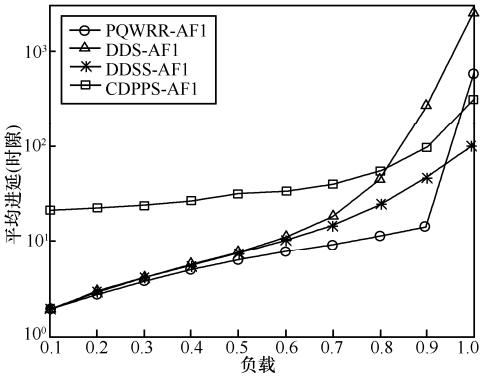
图 8 分别给出了 PQWRR、DDS、DDSS 和 CDPPS 在突发非均匀业务流下，EF 业务、AF1 业务、AF3 业务、BE 业务的平均时延。图 9 给出了 EF 业务在突发非均匀业务流负载为 0.7 时的时延抖动分布。PQWRR、DDS、DDSS 和 CDPPS 之间的时延性能关系与突发均匀业务流下类似，但是 DDS 的 AF1 业务性能在负载高于 0.9 时急剧下降，DDS 的 AF3 业务在负载接近于 1 时变得不稳定。这是因为非均匀业务流对 IQ 交换结构的性能影响较大，而对于能够提供无阻塞交换的 OQ 结构却没有任何影响；而 DDSS 所基于的 CICQ 结构和 CDPPS 所基于的并行交换结构由于能够模仿 FCFS-OQ 结构，所以在突发非均匀业务流下的 AF 业务性能并未受太大影响，平均时延大小与突发均匀流下相近。

5 结束语

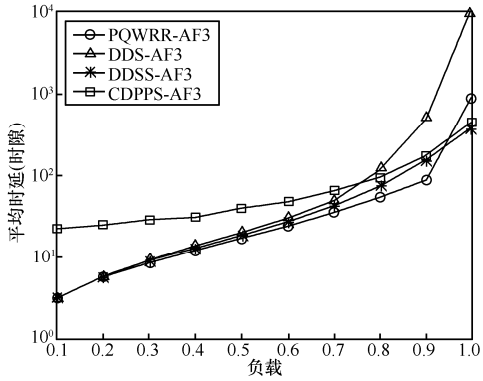
由于链路速率飞速发展，而存储器速率发展缓慢，因此基于单芯片 OQ、IQ 或 CICQ 来构建支持区分服务的交换系统已经变得越来越不可行。为此，本文基于 CICQ 提出了一种支持区分服务的分布式并行交换 CDPPS。它通过使用多个低速交换设备构建高速交换系统，解决了在链路速率和存储器速率发展差异情况下的高速数据交换问题。在不做任何改动的情况下将 CICQ 结构作为中间层平面，充分地利用了现有的交换结构资源。通过使用分布式调度算法，避免了系统各个部件之间的通信开销，降低了硬件复杂度。理论分析和实验仿真都表明，CDPPS 具有良好的公平性和时延性能，可以在高速环境下为各类业务提供满意的服务质量。笔者下一步工作主要包括降低复用器区分业务调度的复杂性，以及对下一代网络的各种业务提供更普适的数据交换支持。



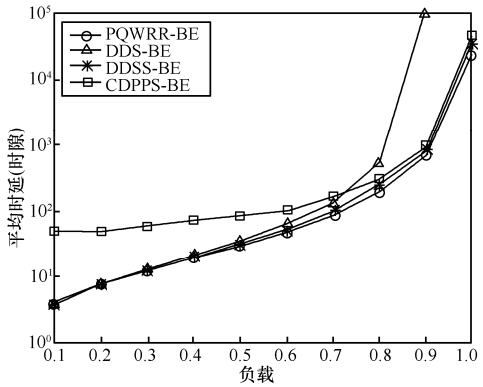
(a) 突发非均匀业务流下 EF 的平均时延



(b) 突发非均匀业务流下 AF1 的平均时延



(c) 突发非均匀业务流下 AF3 的平均时延



(d) 突发非均匀业务流下 BE 的平均时延

图 8 突发非均匀业务流下各种业务的平均时延

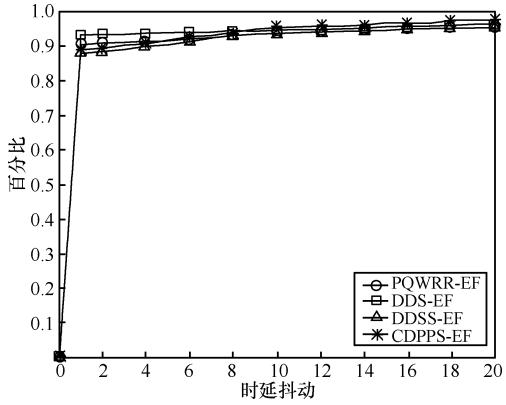


图 9　突发非均匀业务流下 EF 的时延抖动

参考文献：

[1] BRADEN R, CLARK D, SHENKER S. Integrated Services in the Internet Architecture: An Overview[S]. IETF RFC 1633, 1994.

[2] BLAKE S, BLACK D, CARLSON M. An Architecture for Differentiated Services[S]. IETF RFC 2475, 1998.

[3] NICHOLS K, BLAKE S, BAKER F. Definition of the Differentiated Service Field (DS Field) in the IPv4 and IPv6 Headers[S]. IETF RFC 2474, 1998.

[4] JACOBSON V, NICHOLS K, PODURI K. An Expedited Forwarding PHB[S]. IETF RFC 2598, 1999.

[5] HEINANEN J, BAKER F, WEISS W. Assured Forwarding PHB Group[S]. IETF RFC 2597, 1999.

[6] KWAK J Y, NAM J S, KIM D H. A modified dynamic weighted round robin cell scheduling algorithm[J]. ETRI Journal Trans on Communications, 2002,24(5): 360-372.

[7] MAO J, MOH W M, WEI B. PQWRR scheduling algorithm in supporting of DiffServ[A]. Proceedings of the IEEE ICC2001[C]. Helsinki, Finland, June 2001. 679-684.

[8] HWANG I S, HWANG B J, DING C S. Adaptive weighted fair queueing with priority (AWFQP) scheduler for DiffServ networks[J]. Journal of Informatics & Electronics, 2008, 2(2): 15-19.

[9] YANG M, LU E, ZHENG S Q. Scheduling with dynamic bandwidth share for DiffServ classes[A]. Proceedings of the ICCCN[C]. Dallas, USA, 2003. 319-324.

[10] YANG M, WANG J, LU E. Hierarchical scheduling for DiffServ classes[A]. Proceedings of the IEEE Globecom[C]. Dallas, TX, November 2004. 707-712.

[11] 伊鹏, 扈红超, 于婧. 一种支持 DiffServ 模型的全分布式调度算法[J]. 软件学报, 2008, 19(7): 1847-1855.
YI P, HU H C, YU J. A distributed DiffServ supporting scheduling algorithm[J]. Journal of Software, 2008, 19(7): 1847-1855.

[12] IYER S, MCKEOWN N. Making parallel packet switches practical[A]. Proceedings of the IEEE INFOCOM[C]. Anchorage, AK, April 2001. 1680-1687.

[13] ZHANG Z L, RIBEIRO V J, MOON S. Small-time scaling behaviors of Internet backbone traffic: an empirical study[A]. Proceedings of IEEE INFOCOM2003[C]. San Francisco, April 2003. 1826-1836.

作者简介：

任涛（1982-），男，江苏镇江人，国家数字交换系统工程技术研究中心博士生，主要研究方向为宽带信息网络交换技术、高速路由器关键技术。

兰巨龙（1962-），男，河北张北人，国家数字交换系统工程技术研究中心教授、博士生导师，主要研究方向为宽带信息网络、高速路由器核心技术。

扈红超（1982-），男，河南商丘人，国家数字交换系统工程技术研究中心博士生，主要研究方向为高速路由器关键技术、网络流量均衡技术和业务管控技术。

程东年（1957-），男，河南原阳人，国家数字交换系统工程技术研究中心教授，主要研究方向为宽带信息网络体系结构、网络安全协议和网络性能分析技术。