基于 Crossbar 的可重构网络输入排队分域调度研究

张博[1]，汪斌强[1]，王珊珊[1]，卫红权[1]，李挥[2]
(1. 国家数字交换系统工程技术研究中心，河南 郑州 450002;
2. 北京大学深圳研究生院 深圳市云计算关键技术和应用重点实验室，广东 深圳 518055)

摘 要：为解决传统网络技术体系中交换结构无法满足大量差异化业务规模化应用的问题，本文基于可重构网络技术体系，采用选择关闭部分 Crossbar 交叉节点的分域模型，提出了分域调度的思想，分析并推导了承载组内的 SDRR 调度算法和域内最长队列优先调度算法。最后采用交换性能仿真平台对该调度算法进行了复杂度和时延的仿真比较，结果表明：分域调度的最长队列优先算法比一般最长队列优先算法相对复杂度低，且随着调度域个数增加，相对复杂度降低。在相同业务源输入条件下，Crossbar 三分域调度算法的时延小于非分域调度算法的时延，接近公平输出排队调度算法的时延。

关键词：可重构网络；服务承载网；分域调度；最长队列优先
中图分类号：TP393 **文献标识码：**A **文章编号：**1000-436X(2012)09-0105-11

Research on input-queued slicing domain scheduling based on Crossbar in the reconfigurable network

ZHANG Bo[1], WANG Bin-qiang[1], WANG Shan-shan[1], WEI Hong-quan[1], LI Hui[2]
(1. National Digital Switching System Engineering Technological R&D Center（NDSC）, Zhengzhou 450002,China;
2. Shenzhen Key Lab of Cloud Computing Technology and Application, Peking University Shenzhen Graduate School, Shenzhen 518055, China)

Abstract: In order to solve the problem which was switching fabric could not meeting scale application of abundant different business in traditional technology system. The slicing domain scheduling viewpoint based on reconfigurable network technology system was proposed. It used selecting and closing part Crossbar switching points slicing model. It analyzed and deduced smoothed deficit round-robin (SDRR) scheduling algorithm in carrying group and longest queue first (LQF) scheduling algorithm in scheduling domain. Then it contrasted complexity and time delay in switching performance evaluation system (SPES). The results show that the complexity of slicing domain LQF is less than traditional LQF. The more number of domains is, the less complexity of slicing domain LQF is. The time delay of slicing domain scheduling with three slices is less than others without slicing domain, and is close to time delay of fair output-queued scheduling algorithm.

Key words: reconfigurable network; service carrying network; slic g domain scheduling; longest queue first

1 引言

互联网基于资源统计复用、“尽力而为”服务模式的特点，辅以 Overlay、CDN 等技术拓展了其所承载的业务范围，一定程度上满足了规模化的音视频业务、安全业务等承载要求，但并未从根本上

收稿日期：2011-11-28；**修回日期：**2012-02-24
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（2012CB315901, 2012CB315905**）；国家自然科学基金资助项目（**NSFC61179028**）**
Foundation Items: The National Basic Research Program of China (973 Program) (2012CB315901, 2012CB315905); The National Natural Science Foundation of China (NSFC6119028)

解决互联网面临的安全、多播、QoS 等问题。其根本原因在于一方面网络是刚性的，网络的设计与构建依据特定业务需求进行，改造只能依靠升级和扩展，无法实现功能重构；其二节点是封闭的，节点的升级和扩展只能由原提供商实施，无法实现开放。从而导致了互联网的僵化问题。针对上述问题，摆脱传统网络技术体系束缚，提出了一种面向服务提供的柔性网络技术体系。

面向服务提供的柔性网络技术体系对现有和未来可能出现的用户业务进行科学聚类[1]，业务的聚类方法有多种，因应用环境不同而异，当前主要是通过业务对服务质量的要求进行聚类。还可以按照功能特征进行聚类，例如互联网上的业务可以分为视频、语音、交互式命令等。还可以按照通信方式的数量进行分类，将业务分为点到点业务、点到多点业务和多点到多点业务等[2]。

在柔性网络技术体系中，基础设施提供商建设、管理和维护物理网络基础设施，为可重构服务承载网(RSCN, reconfigurable service carrying network)提供网络资源。服务提供商在网络级，根据业务需求考虑异质和同质、成本和收益[3]和负载流量均衡[4]等因素实现 RSCN 优化构建。在节点级，通过重构将节点资源分割为某个 RSCN 独享，支持网络级承载网之间的独立运行。当某一业务取消时，该 RSCN 可立即拆除，将网络资源应用到新的 RSCN 构建中。

网络基础设施包括可重构路由交换平台、智能光节点和相关链路等，其中可重构路由交换平台是网络基础设施的核心。可重构路由交换平台中的交换结构作为业务时延、时延抖动、分组丢失率等特性主要的影响组件，具有更加重要的作用，可以通过对交换资源分割，使各 RSCN 独享交换资源，来满足业务对 QoS 的需求。如图 1 所示，R 为可重构路由交换平台，S 为对应的交换资源(包括缓存、调度、交换结构等)，综合管理接收用户需求，生成 RSCN 构建命令，下发给各个可重构路由交换平台，其中包括对交换结构的分割参数，可重构路由交换平台接收命令，通过重构，将交换资源分割给各个承载网，完成不同承载网之间交换资源的隔离。其中，▨为承载网 1 对应交换结构，▤为承载网 2 对应交换结构，▬为承载网 3 对应交换结构，其他为未使用的交换资源。

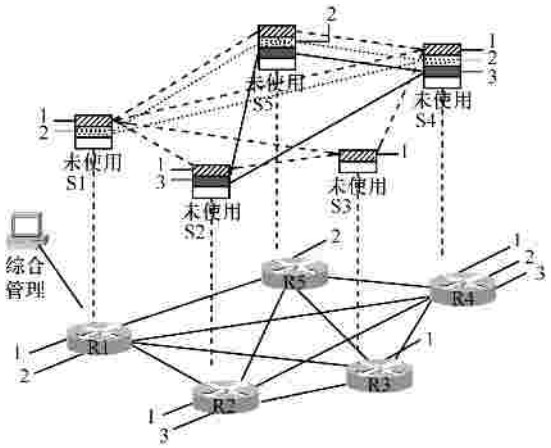

图 1 可重构路由交换平台交换结构空分示意

本文在面向服务提供的柔性网络技术体系下，基于 RSCN 的构建，针对可重构路由交换平台中 Crossbar 交换资源，以 8×8 Crossbar 的分域为例引出了输入排队 Crossbar 交换结构调度问题，提出了一种分域承载组调度算法，该算法在单个承载组内进行轮询调度，域内进行最长队列优先调度，推导了组内和域内的调度过程。并在交换性能仿真平台 SPES 中进行了复杂度和时延特性的仿真，仿真结果表明：分域调度算法的调度复杂度小于不分域调度算法的调度复杂度，相对于传统典型的调度算法，分域承载组调度算法具有更优的时延特性。

2 交换结构分域原理

2.1 面向 RSCN 的交换结构需求

RSCN 与虚拟网构建不同之处在于：网络环境和实现技术不同，虚拟网构建通过设备虚拟化的形式，选择和设置虚拟节点，连接虚拟节点间的虚拟链路，而 RSCN 构建基于可重构路由交换设备的柔性网络，通过设备的重构形式支持。构建拓扑不同，虚拟网拓扑和实际物理网络拓扑是分离的，而 RSCN 拓扑与实际物理网络拓扑是对应的。

在 RSCN 的构建中，网络资源可抽象表示为 $G(V,E)$，其中，V 表示物理节点的集合，E 表示物理链路的集合，$\forall e\in E$，$c(e)$ 表示链路 e 所能够承载的最大带宽。对于 RSCN 构建需求，可抽象表示为 $\{G^v(V^v,E^v),D\}$，其中，$G^v(V^v,E^v)$ 表示 RSCN 的拓扑，V^v 表示 RSCN 逻辑节点的集合，E^v 表示 RSCN 逻辑链路的集合，$D=\{d_{e^v}\mid e^v\in E^v\}$ 为逻辑链路 e^v 的带宽需求 d_{e^v} 的集合。RSCN 构建问题可描述为将逻辑链路映射到物理路径，记作 $path(s,t)$

$=M_{\text{link}}(e^v)$，其中，$s=M_{\text{node}}(s^v)$，$t=M_{\text{node}}(t^v)$，$path(s,t)$ 表示 s 至 t 所经过的物理链路，$M_{\text{link}}(\cdot)$ 表示逻辑链路到物理路径的映射关系，$M_{\text{node}}(\cdot)$ 表示逻辑节点到物理节点的映射关系。RSCN 构建是求解在 $G(V,E)$ 中构建一个子图 $G'(V',E')$，该子图满足：

$$\begin{cases} V'=\{v\mid \forall v^v\in V^v, v=M_{\text{node}}(v^v)\} \\ E'=\bigcup\limits_{\forall e^v\in E^v} M_{\text{link}}(e^v) \end{cases} \tag{1}$$

且 $\forall e\in E'$，链路 e 上的带宽为

$$x(e)=\sum_{\left\{e^v\mid e\in M_{\text{link}}(e^v),\forall e^v\in E^v\right\}} d_{e^v} \tag{2}$$

设 $f(e)$ 表示链路 e 单位流量的带宽所需的费用，对 RSCN 构建的设计目标是构建子图 $G'(V',E')$ 的费用最小化，如式(3)所示。

$$Minimize \quad cost=\sum_{e\in E'} f(e)x(e) \tag{3}$$

假设节点 $v'_i\in V'$ 的需求为 $\{x_{11}(e),x_{12}(e),\text{L},x_{1N}(e);\ x_{21}(e),\text{L},x_{2N}(e);\text{L};x_{i1}(e),\text{L},x_{iN}(e);\text{L}\}$，可转换为如图 2 所示的需求。

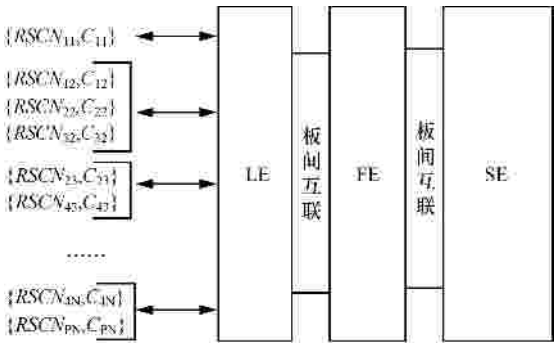

图 2　节点构建需求

其中，$\{RSCN_{ki},C_{ki}\}$ 表示 k 号服务承载在第 i 个端口的带宽需求为 C_{ij}。在可重构路由交换平台中，SE(switching element)为交换组件，FE(forwarding element)为转发组件，LE(link element)为接口组件，其具体的实现方式见文献[5]。在 SE 中，采用 Crossbar 交换结构的 IQ 调度。

2.2　Crossbar 结构分域

对于输入端口 i，考虑到双向传输的需求，则第 k 个承载网在输入端口 i 的带宽需求 $Input(C_{ki})$ 等于第 k 个承载网在输出端口 i 的带宽需求 $Output(C_{ki})$。定义承载网输入矩阵为 $\boldsymbol{R}=\left[r_{ij}\right]_{P\times N}$ 表示承载网在各输入端口的分布，其生成过程如算法 1。

算法 1　$\boldsymbol{R}$ 生成过程

1) $j=1$
2) for j do
3) for each $RSCN_{ki}$ do
4) if $i\le P$ then
5) $r_{ij}=1$
6) else $r_{(\cdot/i)j}=0$
7) end if
8) end for
9) $j++$
10) end for

如果函数 $S(x)$ 满足 $S[x]=\begin{cases}0, x=0\\ 1, x\neq 0\end{cases}\quad \forall x\in R$，则称函数 $S(x)$ 为定义在实数空间上的状态矩阵。

如果对于任意实矩阵 $\boldsymbol{X}=\left[x_{ij}\right]_{N\times N}$ 满足 $S[\boldsymbol{X}]=\left[S[x_{ij}]\right]_{N\times N}$，则称 $S[\boldsymbol{X}]$ 为定义在实矩阵上的矩阵状态函数。

由 $\boldsymbol{M}'=\boldsymbol{R}^{\text{T}}\boldsymbol{R}$ 得交叉开关匹配矩阵 $\boldsymbol{M}$ 为

$$\boldsymbol{M}=S[\boldsymbol{M}']=\begin{bmatrix} m_{11} & m_{12} & \text{L} & m_{1N} \\ m_{21} & m_{22} & \text{L} & m_{2N} \\ \text{L} & \text{L} & \text{L} & \text{L} \\ m_{N1} & m_{N2} & \text{L} & m_{NN} \end{bmatrix} \tag{4}$$

定义 1　第 n 个调度域用 SD_n 表示，满足 $SD_n(I_r)$ 数据分组不到达 $SD_{n'}(O_r)$，且 $SD_{n'}(I_r)$ 数据分组不到达 $SD_n(O_r)$，其中，$SD_n(I_r/O_r)$ 为第 n 个调度域的任意输入/输出端口。

对于一个 $N\times N$ 的 Crossbar 交换结构，为了更清晰的说明分域原理，选择 $N=8$，如图 3(a)所示。Crossbar 交换结构内部无阻塞，如果无出端口竞争，可实现输入 8 个端口到输出 8 个端口的数据交换。假设该结构被平均分割给 3 个调度域，即每个调度域需要该交换结构端口数量为 2、3 和 3。共有 $C_8^2\times C_8^3\times C_8^3$ 种分法。假设其中调度域 1 需求的输入端口为 $\{1,4\}$，输出端口为 $\{1,4\}$，调度域 2 需求的输入端口为 $\{3,5,8\}$，输出端口为 $\{3,5,8\}$，调度域 3 需求的输入端口为 $\{2,6,7\}$，输出端口为 $\{2,6,7\}$，如图 3(b)所示，将每个交叉开关当作交换资源，平均分割成 3 个承载网调度域，其交换资源利用率为 $l_3=\dfrac{2\times 2+3\times 3+3\times 3}{8\times 8}=\dfrac{22}{64}$，因为有 42 个交叉开关在该分割周期内不再被控制，对该 42 个交叉开关进行关闭来实现分域调度。

假设 $N\times N$ 的 Crossbar 交换结构分割成 n 个端口数不等的承载网调度域，即 n 个承载网调度域，可用 (N_1,N_2,L,N_n) 来表示其占有的输入总端口数，满足 $N_1+N_2+\text{L}+N_n=N$，则第 $i(1\le i\le n\le N)$ 个调度域占用输入端口数为 N_i。其 Crossbar 交换资源利用率为

$$l_n=\sum_{i=1}^{n}\left(\frac{N_i}{N}\right)^2 \tag{5}$$

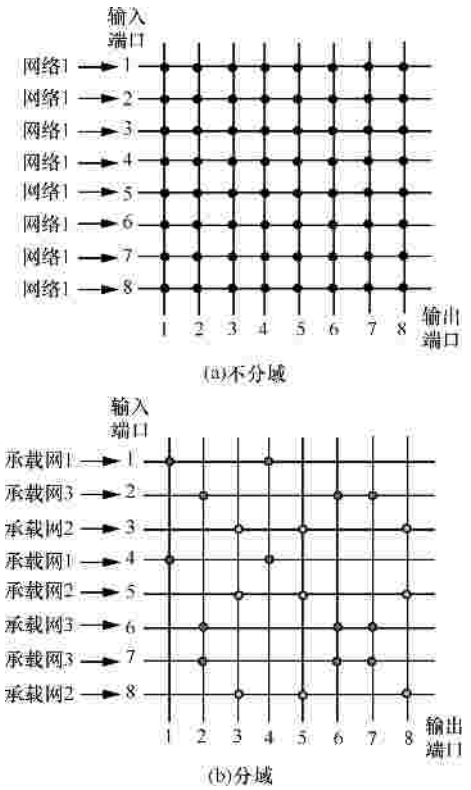

图 3 8×8 交换结构分域示意

对于 Crossbar 交换资源利用率 l_n 来说，当 $n=1$ 时，$l_{\max}=1$，为传统的无分域交换；当 $n=N$ 时，$l_{\min}=\frac{1}{N}$，为 N 个点到点交换；当 $1<n<N$ 时，$l_n=\sum_{i=1}^{n}\left(\frac{N_i}{N}\right)^2$，为分域交换。对于分域交换，当 $n=M_0$ 时，假设 $a_i=\frac{N_i}{N}$，共有 $\prod_{i=1}^{M_0}C_{N(1-\sum_1^{i-1}a_i)}^{Na_i}$ 种分域方式。通过分割成 n 个 $N_i\times N_i,i=1,2,\text{L},n$ 的调度域，减少单个调度域输入输出端口数，将集中式的 $N\times N$ 调度复杂度转换为 $N_i\times N_i$ 调度复杂度。

3 分域承载组调度算法

3.1 输入排队调度

在分组交换路由技术的发展上，交叉（crossbar）矩阵在 N 较小时是一类实现无阻塞的理想交换结构，它的调度过程是先由调度器对活跃输入端口进行无输出端口争用的配对，决定所有活跃输入端口下一个时隙的输出端口，而在下一个时隙进行传输。仅靠交换结构本身还无法实现无阻塞交换，必须与相应的缓存方式与调度算法相结合。输出排队（OQ）[6]调度，能够为业务提供 100%吞吐量、速率及时延方面的 QoS 保证，但需要交换结构的加速比达到 N，当 N 较大时是很难实现的。

比较而言，输入队列（IQ）调度，只需交换单元和存储单元工作在线速，采用虚拟输出排队（VOQ）机制解决队头阻塞问题。但输入排队交换结构的调度算法需要全局考虑交换结构所有输入端口和输出端口的带宽使用，因此必须采用集中式的调度机制，其本质是一个双向图的匹配求解问题。集中式的调度机制使得在输入排队交换结构实现服务质量保障十分复杂。已提出的如最大权重匹配调度算法已经证明可以提供 100%的吞吐量，然而其复杂度为 $O(N^3\log N)$ [7]，很难具有现实意义。文献[8]致力于对最大权重匹配算法进行简化，但是其复杂度降低有限，依然无法在高速环境下应用。另一种思路通过利用输入排队交换结构双向图匹配特征，采用随机化思想[9]求解最大匹配的逼近匹配关系。但它仅从吞吐量单个方面优化调度性能，缺乏其他相关的服务质量保障措施。

虽然研究人员提出种种其他解决方案试图降低 MWM（maximum weight matching）和 MSM（maximum size matching）的实现复杂度[10,11]，然而不加速条件下，这种复杂度降低是以牺牲 IQ 交换结构性能为代价的。通过对基于 Crossbar 输入缓存调度的研究不难发现，影响缓存调度的一个关键因素是交换结构的输入输出端口数，如表 1 所示。

表 1 输入排队调度算法复杂度与 N 的关系

调度算法	匹配方法	复杂度
MSM	双向图最大边数匹配	$O(N^{2.5})$
iSLIP	双向图极大边数匹配	$O(N^2)$
LPF	双向图最大权重匹配	$O(N^{2.5})$
iOCF	双向图极大权重匹配	$O(N^2\text{lb}N)$
LAURA	双向图权重随机选择	$O(N\text{lb}N)$
MUCFA	基于稳定婚姻匹配	$O(N^2)$

为在复杂度和性能之间进行折中，研究人员逐

步着眼于将基于时间戳和轮询的调度算法进行结合。GRR(group round robin)[12]引入一种流分组策略将大量流聚类为“流组”，并采用基于时间戳的 WF^2Q（worst-case fair weighted fair queuing)算法[13]作为组间调度算法，DRR(deficit round-robin)算法[14]作为组内调度算法，提高算法的公平性和降低复杂度。在组数较小的常量假设下，GRR 能基于现有算法获得 $O(1)$ 时间复杂度。当流速动态改变时，基于时间戳的调度策略无法提供恒定的 GPS(generalized processor sharing)[15]相对时延。

FRR(fair round-robin)[16]组间采用基于时间戳的调度策略，组内采用轮询的调度策略，能为流组 i 提供端到端时延上限。不足之处在于其“摊还”复杂度，在传输前 FRR 算法需要将组内分组组装较大的“帧”，所以分组会经历 $O(K)$ 的组装时延。

轮询调度算法 RR/GPFQ(round-robin/goup packet fair queuing)[17]将基于分组的 GPS 算法进行改进以支持流速率的动态调整，降低了算法复杂度，并获得严格的时延上限，且能够提供时延和公平性特性。RR/GPFQ 算法的时延和公平性上限仅取决于给定组内或组群的流状态，而不是分组数目 N。

针对以上输入调度和分层调度的不足之处，本文针对可重构服务承载网构建的特殊性，采用分层调度的思想，将调度过程分为 2 层，如图 4 所示，第 1 层将单个 RSCN 对应一个调度组，进行承载组内调度(CGS, carrying group scheduling)；第 2 层进行组间的域调度(DS, domain scheduling)。

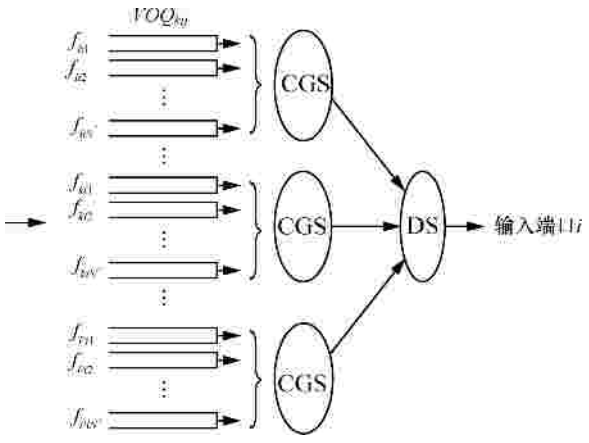

图 4　承载组调度示意

其中，队列 VOQ_{kij} 缓存第 k 个承载网到达输入端口 i、去往输出端口 j 的业务流 f_{kij}、N' 为本调度域所包含的输入端口数。具体而言，首先根据 P 个承载网划分为 P 个承载组 $\boldsymbol{G}_i=\{\boldsymbol{G}_i^1,\boldsymbol{G}_i^2,\cdots,\boldsymbol{G}_i^P\}$，在每个承载组内采用改进的轮询机制决定组内数据分组 $\{p_{i1}^k,p_{i2}^k,\cdots,p_{iN'}^k\}$ 的调度顺序。而后，DS 采用基于最长队列优先的调度机制决定各 $\boldsymbol{G}_i^k$ 调度顺序。

3.2　承载组内调度

针对本文承载组内调度借鉴文献[18]中 SRR 调度算法，对 LDRRWA 进一步改进，采用平滑的 DRR 调度策略(SDRR, smoothed DRR)。

在单个承载网内，用 r_{kij} 标识 f_{kij} 的带宽需求，用 $w_{kij}=\dfrac{r_{kij}}{R}$ 表示调度权重，如式(6)所示。

$$\boldsymbol{G}_{ij}^k=\left\{f_{kij}\in\boldsymbol{G}_{ij}^k:\frac{1}{B^h}\le w_{kij}<\frac{1}{B^{h-1}}\right\} \tag{6}$$

其中，B 为常量，令 $r_{\max}=\max_{k,j}\{r_{kij}\}$，$r_{\min}=\min_{k,j}\{r_{kij}\}$，则 $\dfrac{1}{B}\le\dfrac{r_{\max}}{R}$，取 $B=\dfrac{R}{r_{\max}}$。又 $\dfrac{1}{B^k}\le\dfrac{r_{\min}}{R}<\dfrac{1}{B^{k-1}}$，所以 $\log_B\dfrac{R}{r_{\min}}\le h<\log_B\dfrac{R}{r_{\min}}+1$。取 $h=\left\lceil\log_B\dfrac{R}{r_{\min}}\right\rceil$，得

$$h=\left\lceil\log_{\frac{R}{r_{\max}}}\frac{R}{r_{\min}}\right\rceil=\left\lceil\frac{\lg R-\lg r_{\min}}{\lg R-\lg r_{\max}}\right\rceil \tag{7}$$

考察组 $\boldsymbol{G}_i^k$，单个组中，SDRR 为每条流 $f_{kij},j=1,2,\cdots,N'$，按照其权重分配轮询调度份额 x_{kij}，即：$x_{kij}=B_k w_{kij}L_{\max}$，其中，$L_{\max}$ 为最大分组长度。

SDRR 算法以“帧”组织一次轮询调度过程，具体如算法 2。行 1)将承载组共享份额计数器 $CG\_deficit$ 清“0”。行 2)~16)对至少输出一个分组的流组成的链表 $acitve\_list$ 进行轮询输出，非空的流 f_{ijk} 将转移到链表 $acitve\_list'$ 中，以进行第 2 轮分组的共享份额轮询调度。如行 17)~23)所示，该共享份额过程借用 $CG\_deficit$ 中份额来满足部分不能支持本轮分组输出的调度份额需求，出现负份额，在下一轮调度过程中，上次没有借用到份额的流优先借用。

算法 2　SDRR 调度过程

1) $CG\_deficit=0$

2) for each flow f_{kij} in $acitve\_list$ do

3) $\quad s_{kij}+=x_{kij}$

4) if $f_{kij} \neq NULL$ then
5) $p_{kij}^{HOL} = HOL(f_{kij})$
6) if $l_{kij}^{HOL} <= s_{kij}$ then
7) $s_{kij} -= l_{kij}^{HOL}$
8) remove and add l_{kij}^{HOL} to $sent\_list$
9) add f_{kij} to the tail of $acitve\_list$
10) else $CG\_deficit += s_{ijk}$
11) add f_{kij} to the tail of $acitve\_list'$
12) break
13) end if
14) remove f_{kij} from $acitve\_list$
15) end if
16) end for
17) for each flow f_{kij} in $avtive\_list'$ do
18) if $CG\_deficit > 0$ then
19) $CG\_deficit -= l_{kij}^{HOL}$
20) $s_{kij} = s_{kij} - l_{kij}^{HOL}$
21) remove and add p_{kij}^{HOL} to the $sent\_list$
22) end if
23) $avtive\_list = avtive\_list'$

3.3 域调度

域调度采用最长队列优先(LQF, longest queue first)的调度方法，记为 DSLQF。如图 3(a)所示，对于一个 8×8 的无分割交换结构，二分图匹配下的 8×8 匹配矩阵表示为 $\boldsymbol{M}(n)$，其中，元素 $m_{ij}(n)$ 表示第 n 时隙内输入端口 $I_i(i=1,2,\text{L},8)$ 到输出端口 $O_i(i=1,2,\text{L},8)$ 的交叉开关的状态。

$$\boldsymbol{M}(n)=\left[m_{ij}(n)\right]_{8\times8} \tag{8}$$

$$m_{ij}(n)=\begin{cases}0, n\text{时隙交叉节点}cross\text{状态}\\ 1, n\text{时隙交叉节点 bar 状态}\end{cases},\ \forall i,j=1,2,\text{L},8$$

依图 3(b)所示，每个 SD 可以用 3 个参数表示为 $C_k=\{N_k,\vec{I}_k,\vec{O}_k\}$，其中，$N_k$ 为第 k 个 SD 占用的端口数，$\vec{I}_k$ 为第 k 个 SD 的占用输入端口向量，$\vec{O}_k$ 为第 k 个 SD 的占用输出端口向量。依图 3(b)所示，对于分割后的 8×8 交换结构。

DS_1 可以表示为

$$\begin{aligned}C_1&=\{N_1,\vec{I}_1,\vec{O}_1\}\\&=\{2,[1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0],[1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0]\}\end{aligned} \tag{9}$$

DS_2 可以表示为

$$\begin{aligned}C_2&=\{N_2,\vec{I}_2,\vec{O}_2\}\\&=\{3,[0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1],[0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1]\}\end{aligned} \tag{10}$$

DS_3 可以表示为

$$\begin{aligned}C_3&=\{N_3,\vec{I}_3,\vec{O}_3\}\\&=\{3,[0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0],[0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0]\}\end{aligned} \tag{11}$$

二分图匹配下的 8×8 匹配矩阵为

$$\boldsymbol{M}'(n)=\boldsymbol{M}(n)\odot\left(\vec{I}_1^{\mathrm{T}}\cdot\vec{O}_1\right)+M(n)\odot\left(\vec{I}_2^{\mathrm{T}}\cdot\vec{O}_2\right)+M(n)\odot\left(\vec{I}_3^{\mathrm{T}}\cdot\vec{O}_3\right) \tag{12}$$

其中，$\odot$ 为 Hadamard 积[19]。

$$\boldsymbol{M}'(n)=\begin{bmatrix}0 & m_{112}(n) & 0 & 0 & m_{115}(n) & 0 & 0 & 0\\ m_{321}(n) & 0 & m_{323}(n) & m_{324}(n) & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & m_{236}(n) & m_{237}(n) & m_{238}(n)\\ 0 & m_{142}(n) & 0 & 0 & m_{145}(n) & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & m_{256}(n) & m_{257}(n) & m_{258}(n)\\ m_{361}(n) & 0 & m_{363}(n) & m_{364}(n) & 0 & 0 & 0 & 0\\ m_{371}(n) & 0 & m_{373}(n) & m_{374}(n) & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & m_{286}(n) & m_{287}(n) & m_{288}(n)\end{bmatrix}_{8\times8} \tag{13}$$

$$m_{kij}(n)=\begin{cases}0, n\text{时隙交叉节点}cross\text{状态}\\ 1, n\text{时隙交叉节点 bar 状态}\end{cases},\ \forall i,j=1,2,\text{L},N_k$$

其中，$m_{kij}(n)$ 表示第 n 时隙内第 $k(k=1,2,3)$ 个 SD 输入端口 $I_i(i=1,2,\text{L},8)$ 到输出端口 $O_i(i=1,2,\text{L},8)$ 的交叉开关的状态。

对于基于队长的调度算法，相应变化的有队长矩阵、状态矩阵和到达矩阵。其匹配矩阵由 8×8 变为了 $N\times N$，假设其分割为 W 个 SD，分别为

$$C_k = [N_k, \vec{I}_k, \vec{O}_k], k = 1, 2, \cdots, W \tag{14}$$

则其匹配矩阵为

$$\boldsymbol{M}'(n) = \boldsymbol{M}(n) \odot \sum_{k=1}^{W} \left(\vec{I}_k^{\mathrm{T}} \cdot \vec{O}_k \right) \tag{15}$$

无分割队长矩阵 $L(n)$ 表示为

$$\boldsymbol{L}(n) = \begin{bmatrix} l_{11}(n) & l_{12}(n) & \cdots & l_{1N}(n) \\ l_{21}(n) & l_{22}(n) & \cdots & l_{2N}(n) \\ \cdots & \cdots & \cdots & \cdots \\ l_{N1}(n) & l_{N2}(n) & \cdots & l_{NN}(n) \end{bmatrix} = \begin{bmatrix} \vec{L}_1(n) \\ \vec{L}_2(n) \\ \cdots \\ \vec{L}_N(n) \end{bmatrix} = \left[\vec{L}_1(n) \quad \vec{L}_2(n) \quad \cdots \quad \vec{L}_N(n) \right] \tag{16}$$

其中，元素 $l_{ij}(n)$ 表示第 n 时隙内输入端口 $I_i (i = 1, 2, \cdots, N)$ 到输出端口 $O_j (j = 1, 2, \cdots, N)$ 的 VOQ_{ij} 队列队长，行向量 $\vec{L}_i(n)$ 是输入端口 I_i 的队长向量，列向量 $\vec{L}_j(n)$ 是输出端口 O_j 的队长向量。

有分割队长矩阵 $L'(n)$ 表示为

$$\boldsymbol{L}'(n) = \boldsymbol{L}(n) \odot \sum_{k=1}^{W} \left(\vec{I}_k^{\mathrm{T}} \cdot \vec{O}_k \right) = \begin{bmatrix} \vec{L}'_1(n) \\ \vec{L}'_2(n) \\ \cdots \\ \vec{L}'_N(n) \end{bmatrix} = \left[\vec{L}'_1(n) \quad \vec{L}'_2(n) \quad \cdots \quad \vec{L}'_N(n) \right] \tag{17}$$

无分割到达矩阵 $A(n)$ 表示为

$$\boldsymbol{A}(n) = \begin{bmatrix} a_{11}(n) & a_{12}(n) & \cdots & a_{1N}(n) \\ a_{21}(n) & a_{22}(n) & \cdots & a_{2N}(n) \\ \cdots & \cdots & \cdots & \cdots \\ a_{N1}(n) & a_{N2}(n) & \cdots & a_{NN}(n) \end{bmatrix} = \begin{bmatrix} \vec{A}_1(n) \\ \vec{A}_2(n) \\ \cdots \\ \vec{A}_N(n) \end{bmatrix} \tag{18}$$

$$a_{ij}(n) = \begin{cases} 0, n\text{时隙}\mathrm{VOQ}_{ij}\text{无信元到达} \\ 1, n\text{时隙}\mathrm{VOQ}_{ij}\text{有信元到达} \end{cases}, \quad \forall i, j = 1, 2, \cdots, N$$

其中，元素 $a_{ij}(n)$ 表示第 n 时隙内输入端口 $I_i (i = 1, 2, \cdots, N)$ 到输出端口 $O_i (i = 1, 2, \cdots, N)$ 的 VOQ_{ij} 队列到达过程，行向量 $\vec{A}_i(n)$ 是输入端口 I_i 的到达向量。在容许流量模型下，需满足条件

$$a_i(n) = \sum_{j=1}^{N} a_{ij} \le 1, \forall i \in \{1, 2, \cdots, N\} \tag{19}$$

$$a_j(n) = \sum_{i=1}^{N} a_{ij} \le 1, \forall j \in \{1, 2, \cdots, N\} \tag{20}$$

有分割到达矩阵 $\boldsymbol{A}'(n)$ 表示为

$$\boldsymbol{A}'(n) = \boldsymbol{A}(n) \odot \sum_{k=1}^{W} \left(\vec{I}_k^{\mathrm{T}} \cdot \vec{O}_k \right) = \begin{bmatrix} \vec{A}'_1(n) \\ \vec{A}'_2(n) \\ \cdots \\ \vec{A}'_N(n) \end{bmatrix} \tag{21}$$

在容许流量模型下，需满足条件

$$a_{ki}(n) = \sum_{j=1}^{N_k} a_{kij} \le 1, \forall i \in \{1, 2, \cdots, N_k\} \tag{22}$$

$$a_{kj}(n) = \sum_{i=1}^{N_k} a_{kij} \le 1, \forall j \in \{1, 2, \cdots, N_k\} \tag{23}$$

交换结构队长矩阵迭代过程[20]为

$$\boldsymbol{L}(n+1) = \boldsymbol{L}(n) - \boldsymbol{M}(n) + \boldsymbol{A}(n+1) \tag{24}$$

$$\boldsymbol{L}'(n+1) = \boldsymbol{L}'(n) - \boldsymbol{M}'(n) + \boldsymbol{A}'(n+1) \tag{25}$$

无分割状态矩阵 $\boldsymbol{Q}(n)$ 为

$$\boldsymbol{Q}(n) = \left[q_{ij}(n) \right]_{N \times N} \tag{26}$$

$$q_{ij}(n) = \begin{cases} 0, n\text{时隙无信元排队} \\ 1, n\text{时隙有信元排队} \end{cases}, \quad \forall i, j = 1, 2, \cdots, N$$

其中，元素 $m_{ij}(n)$ 表示第 n 时隙内输入端口 $I_i (i = 1, 2, \cdots, N)$ 到输出端口 $O_j (j = 1, 2, \cdots, N)$ 的 VOQ_{ij} 队列中信元的有无。

其状态矩阵 $\boldsymbol{Q}(n)$ 迭代过程为

$$\boldsymbol{Q}(n+1) = S[\boldsymbol{L}(n) - \boldsymbol{M}(n) + \boldsymbol{A}(n+1)] \tag{27}$$

有分割状态矩阵 $\boldsymbol{Q}'(n)$ 为

$$\boldsymbol{Q}'(n) = \boldsymbol{Q}(n) \odot \sum_{k=1}^{W} \left(\vec{I}_k^{\mathrm{T}} \cdot \vec{O}_k \right) = = \begin{bmatrix} \vec{S}'_1(n) \\ \vec{S}'_2(n) \\ \cdots \\ \vec{S}'_N(n) \end{bmatrix} \tag{28}$$

DS 调度过程描述为

步骤 1　调度器获得交换结构分割参数 $C_k = \{N_k, \vec{I}_k, \vec{O}_k\}, k = 1, 2, \cdots, W$，同时令 $n = 1, P = 1$，$\boldsymbol{L}'(0) = \boldsymbol{M}'(0) = \boldsymbol{A}'(0) = \boldsymbol{Q}'(0) = [0]$。

步骤 2　在时隙 n 中有信元到达，状态矩阵 $Q'(n)$ 发生变化，q_{kij} 向其调度器发出请求，输入端口 I_i 形成到达行向量 $\vec{A}'_i(n)$。

步骤 3　调度器将 N 个到达行向量 $\vec{A}'_i(n)$ 送入到达矩阵 $\boldsymbol{A}'(n)$。根据队长矩阵迭代公式 $\boldsymbol{L}'(n) = \boldsymbol{L}'(n-1) - \boldsymbol{M}'(n-1) + \boldsymbol{A}'(n)$ 计算当前时隙队长矩阵 $\boldsymbol{L}'(n)$。

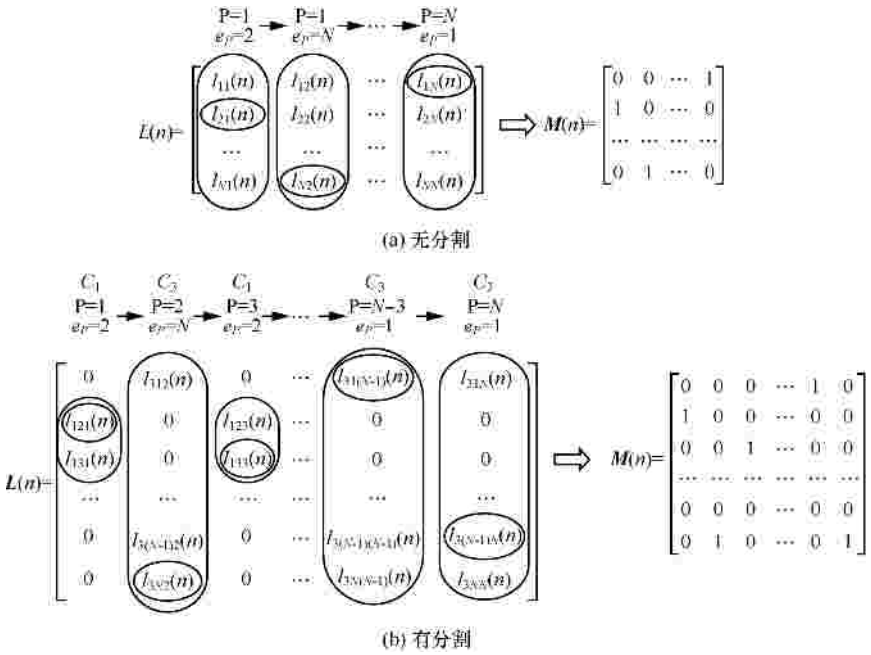

图 5 步骤 4 中无分割仲裁和有分割仲裁的比较

步骤 4 优先级指针 P 指向输出端口 O_p 所对应的队长矩阵列向量 $\vec{A}'_j(n)$ 中同一 SD 中权重最大（队列最长）的元素 e_p，e_p 决定了 $\boldsymbol{M}'(n)$ 中第 P 列非 0 元素的位置。

步骤 5 如果 $P = P + 1 \bmod N \neq 1$，重复步骤 4；如果 $P = P + 1 \bmod N = 1$，执行下一步。

步骤 6 将 $\{e_P, P = 1, 2, \mathrm{L}, N\}$ 送入 $\boldsymbol{M}'(n)$，$n = n + 1$，转向步骤 2。

步骤 4 中无分割仲裁和有分割仲裁的比较如图 5 所示，图 5(b)中以 3 个 SD 为例表示，可见步骤 4 中，对于无分割交换结构，其单次进行 N 个权值的仲裁，分割交换结构单次进行 N_k 个权值的仲裁，减小了仲裁复杂度。

4 仿真实验

本文采用基于 Crossbar 交换性能仿真平台 SPES(switching performance evaluation system)[21]设计并实现了基于交换结构分域的输入排队调度算法。

在原来仿真平台基础上，增加了网络承载控制系统，该系统配置或随机生成调度域分割参数，并将参数传递给输入端子系统、输出端子系统和调度子系统，以决定其调度过程所对应的 SD 和 Crossbar 匹配点，其仿真结构如图 6 所示。

1) 相对运算复杂度仿真

该仿真中假设平均分成 3 个 SD，满足不同 SD 端口数之差不大于 1。图 7 给出了传统矩阵模型的 LQF 与域调度的 DSLQF 算法的相对运算复杂度比较曲线图，相对运算复杂度是对调度算法中算术加法次数进行的相对比较结果。当交换结构端口数 N 依次从 4 增加到 20 时，相对运算复杂度曲线呈类指数迅速增加，传统的 LQF 算法的相对运算复杂度大于 DSLQF 算法的相对运算复杂度。

相对运算复杂度的减小主要由交换结构 SD 的个数决定，如果 SD 分割个数增加，则其运算效率还会提高，但倍数低于 SD 个数。对于同一个 Crossbar 交换结构，DSLQF 算法的相对运算复杂度随 SD 个数增加而减小，但非线性关系。图 8 为16×16 Crossbar 结构相对运算复杂度与 SD 个数关系图，其 SD 对应端口数分割原则为：不同 SD 端口数之差不大于 1。

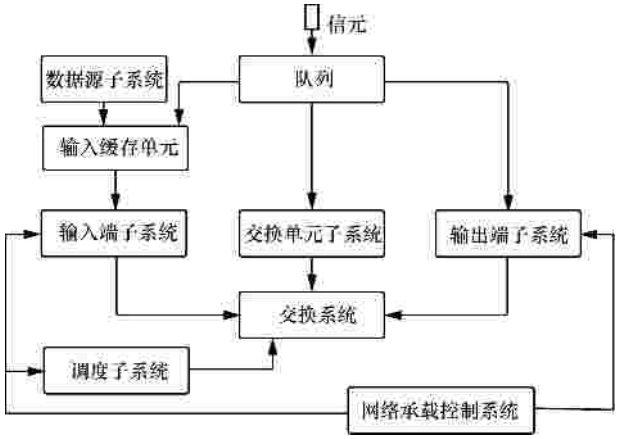

图 6 加入分割参数的输入排队仿真平台

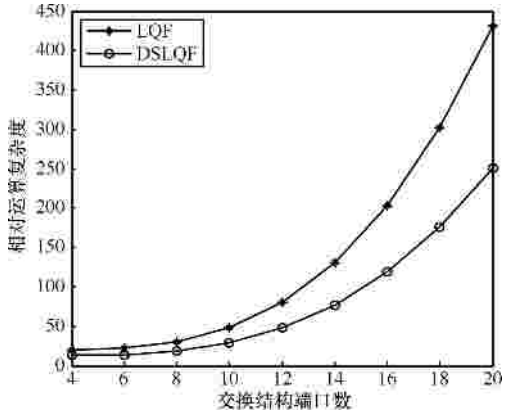

图 7　SM-LQF 与 LQF 相对运算复杂度比较

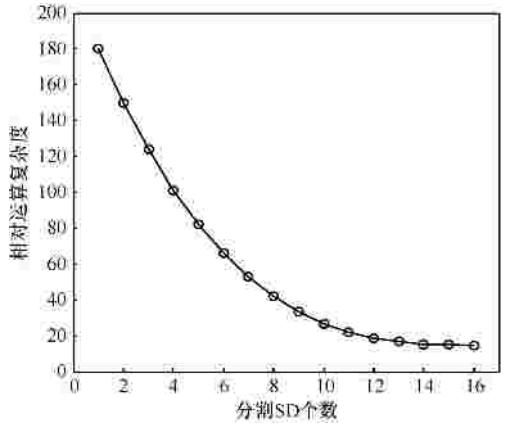

图 8　相对运算复杂度与 SD 个数关系

2) 不同负载时延仿真

时延仿真采用 16×16 的交换结构，cell 长度为 64byte，仿真业务选取了贝努利业务源和 ON-OFF 突发业务源，其中突发业务的突发长度为 20，目的端口分布分别采用均匀分布和 Diagonal 分布，Diagonal 分布的表达式为

$$\text{Diagonal}: l_{ij}=\begin{cases} l\ w, & j=i \\ l\ (1-w), & j=(i+1)\ \bmod\ N \\ 0, & \text{否则} \end{cases} \quad (29)$$

令 $w=2/3$，即 $l_{ii}=2r/3$，$l_{i|i+1|}=r/3$。为了便于和典型交换结构及其调度算法进行性能对比，下文还给出了相同仿真条件下多种算法的仿真结果，具体包括：基于输出排队交换结构的公平排队调度算法 FQ，基于输入排队交换结构的 LQF 调度算法，基于输入排队交换结构的 iSLIP（迭代次数为 4）调度算法，基于分层调度的 DRR-WF^{2}Q 算法和 DRR-LQF 算法，基于不分域的 SDRR-LQF 调度算法和基于分域的 SDRR-DSLQF 调度算法。其分域规则是将 16×16 的交换结构分为 3 个域，每个域的端口数分别为 5×5，5×5 和 6×6。分组丢失率定义为一定时间段内，交换系统丢弃分组数目 n_d 与到达交换系统分组总数目 n_a 之间的比值，即：n_d/n_a。本仿真在交换分组丢失率不大于 1%的情况下讨论时延。

图 9(a)给出在贝努利均匀业务源条件下，各调度算法在归一化负载强度区间 $[0.6,1]$ 的平均时延对比曲线，用 $T_{(*)}$ 表示*调度算法的时延。可见，输出排队 FQ 的 $T_{(FQ)}$ 最小，$T_{(SDRR\text{-}DSLQF)}$ 次之，输入排队 LQF 的 $T_{(LQF)}$ 最大，分层调度算法的时延均小于非分层调度。LQF、DRR-LQF、SDRR-LQF 和 SDRR-DSLQF 4 种调度算法的时延比较，$T_{(LQF)} > T_{(DRR\text{-}LQF)}$ 表明分层调度的时延特性优于非分层调度，因为分层调度将调度过程分为组内调度和组间调度 2 个部分，将输入调度中整体集中式的复杂度变为局部调度复杂度之和，减小仲裁端口数，减小了算法复杂度。$T_{(DRR\text{-}LQF)} > T_{(SDRR\text{-}LQF)}$ 表明 SDRR 比 DRR 调度算法时延小，原因是 *SDRR* 的轮询过程采用份额借用机制，使得单个时隙内更多信元被调度。

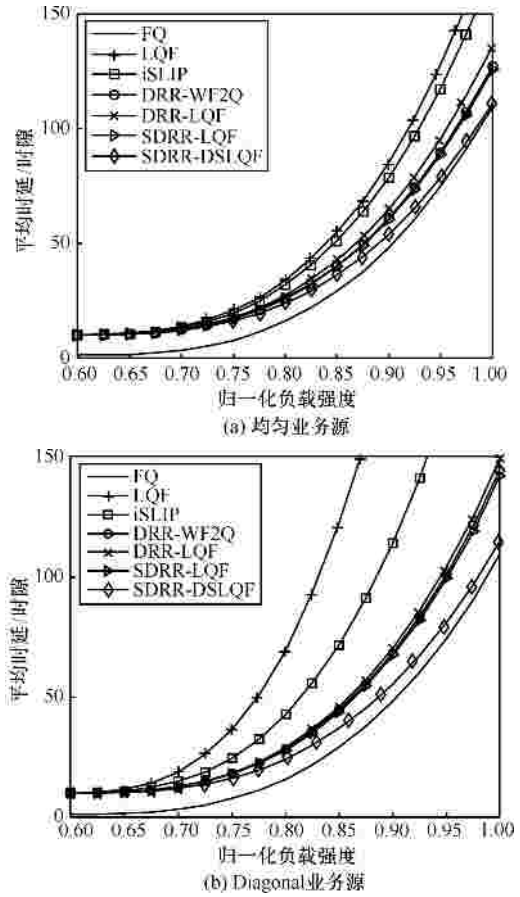

图 9　贝努利业务源时延对比

$T_{(\text{SDRR-LQF})} > T_{(\text{SDRR-DSLQF})}$ 表明分域的调度比不分域调度时延小，原因是分域调度仲裁复杂度小。

图 9(b)给出在贝努利 Diagonal 业务源条件下，各调度算法在归一化负载强度区间[0.6,1]的平均时延对比曲线。与图 9(a)比较可见，非均匀业务源条件下的时延比均匀业务源条件下的时延大，非均匀业务源对输出排队 FQ 的 $T_{(\text{FQ})}$影响最小，对输入排队 LQF 和 iSLIP 的时延影响最大，非均匀业务源输入与均匀业务源输入下的时延大小关系不变。

图 10(a)给出在 ON-OFF 突发均匀业务源条件下，各调度算法在归一化负载强度区间[0.6,1]的平均时延对比曲线。与图 9(a)比较可见，ON-OFF 突发业务源输入下的时延约为贝努利业务源输入下时延的 10 倍，ON-OFF 突发业务源对各算法时延的影响相同，ON-OFF 突发业务源输入与贝努利业务源输入下各算法时延大小关系不变。

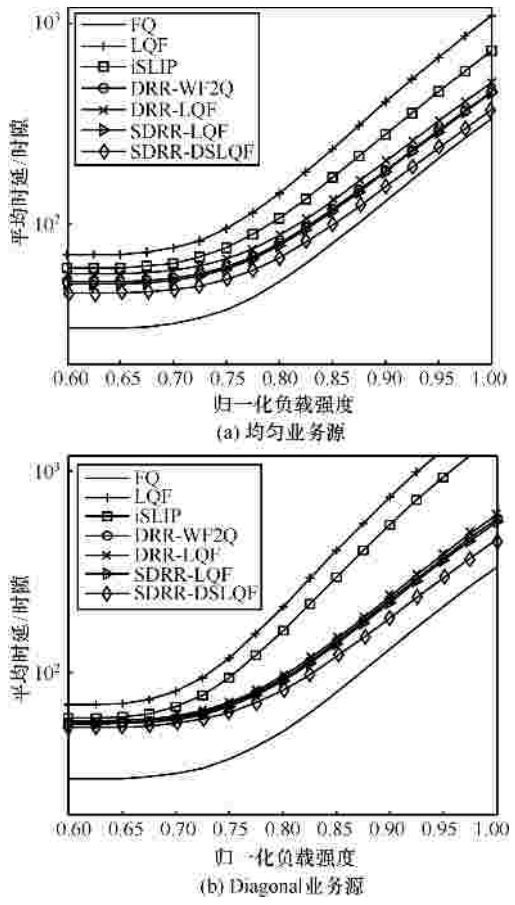

图 10 ON-OFF 突发业务源时延对比

图 10(b)给出在 ON-OFF 突发 Diagonal 业务源条件下，各调度算法在归一化负载强度区间[0.6,1]的平均时延对比曲线。与图 10(a)比较可见，非均匀突发业务源输入下的时延大于均匀突发业务源输入下的时延，在非均匀突发业务源条件下，输入排队 LQF 和 iSLIP 的时延增长迅速，非均匀突发业务源输入下的时延与均匀突发业务源输入下的时延大小关系不变。

对以上仿真结果分析得出如下结论：

1) 对于 $N \times N$ 的 Crossbar 交换结构，SD 个数越大，相对运算复杂度越小，当个数为 N 时，每个 SD 只有一个端口，无需仲裁就可调度，可实现调度延迟为 0。

2) 在 4 种业务源输入条件下，ON-OFF 突发 Diagonal 业务源输入下的时延最大。

3) 任何一种业务源输入下，输出排队 FQ 调度算法的时延都是最小，且在均匀和非均匀业务源条件下保持稳定，可见输出排队虽然需要内部加速，但能为业务提供好的时延 QoS 保证。

4) 任何一种业务源输入下，分层调度的时延总小于非分层调度的时延，主要原因是分层调度将调度过程分为组内调度和组间调度 2 个部分，将输入调度中整体集中式的复杂度变为局部调度复杂度之和，减小仲裁端口数，减小了算法复杂度。

当然从理论上分析，该算法也存在一些不足:

1) Crossbar 交换结构分域承载组调度的前提是在可重构网络技术体系下构建 RSCN 的需求，RSCN 对输入输出端口的约束性条件形成了自封闭的调度域，域内包括几个 RSCN 的业务，且每个 RSCN 业务不全部使用域内输入输出端口。如果没有这个前提，则该分域调度就会出现输入到输出交换路径的不可达问题。

2) 分域承载组调度中的域分割参数是由网络级可重构综合管理系统下达，当单个网络节点接收到该分割参数后，需要停机进行先分割再启动后调度的过程，增加了宕机时延。

3) 分域承载组调度算法的设计需具有可重构性，即需要根据分割参数进行调度的变化，当 N 很大时，重构过程需确定大量关闭的 Crossbar 交叉节点，会提高一定的复杂度。

5 结束语

在目前对通信节点功能和性能要求越来越高的发展模式下，可以通过构建 RSCN 实现网络级的优化，通过分割节点资源，尤其是节点内的交换资

源，来满足不同 RSCN 业务的区分服务。交换结构的分割带来了调度算法的分割，业务特性需求（时延、抖动和分组丢失率等）的不同，对调度算法也有不同的要求，可采用单个调度域中部署不同调度算法来满足要求，这样就可以为特定的业务实现独立的调度，采用交换资源独享将 QoS 等级划分。

参考文献：

[1] **张伟，吴春明，姜明. 网络业务聚类研究**[J]. **解放军信息工程大学学报**, 2009, 10(1):53-56.
ZHANG W, WU C M, JIANG M. Study of network services aggregation[J]. Journal of Information Engineering University, 2009, 10(1): 53-56.

[2] **龚双瑾，刘多，张雪丽等.下一代网关键技术及发展**[M]. **北京：国防工业出版社**, 2006.
GONG S J, LIU D, ZHANG X L, *et al*. Key Technology and Development of Next Generation Network[M]. Beijing: National Defense Industry Press, 2006.

[3] CAPONE A, ELIAS J, MARTIGNON F. Routing and resource optimization in service overlay networks[J]. Elsevier Computer Networks, 2009, 53(2):180-190.

[4] YU M, YI Y, REXFORD J. Rethinking virtual network embedding: substrate support for path splitting and migration[J]. ACM SIGCOMM Computer Communications Review, 2008, 38(2):17-29.

[5] ZHUGE B, YU C, LIU K P. Research on internal flow control mechanism of ForCES routers[J]. Information Technology Journal 2011 Asian Network for Scientific Information, 2011, 10(3):626-638.

[6] KESIDIS G, MCKEOWN N. Output-buffer ATM packet switching for integrated-services communicateon networks[A]. Proceedings of IEEE ICC'97[C]. Montreal, Canada, 1997. 342-350.

[7] MEKKITTIKUL A, MCKEOWN N. A practical scheduling algorithm for achieving 100% throughput in input-queued switches[A]. INFOCOM'98[C]. San Francisco, CA, 1998. 792-799.

[8] MEKKITTIKUL A. Scheduling Non-uniform Traffic in High Speed Packet Switches and Routers[D]. Stanford University, 1998. 17-20.

[9] GIACCONE P, PRABHAKAR B, SHAH D. Towards simple high-performance schedulers for high-aggregate bandwidth switches[A]. IEEE INFOCOM2002[C]. New York, USA, 2002. 120-127.

[10] LEE Y, LOU J Y, LUO J Z. An efficient packet scheduling algorithm with deadline guarantees for input-queued switches[J]. Journal IEEE/ACM Transitions on Networking, 2007, 15(1): 212-225.

[11] LEONARDI E, MELLIA M, NERI F. Design and implementation of a fast VOQ scheduler for a switch sabric[J]. IJCSNS International Journal of Computer Science and Network Security, 2008, 8(9):32-36.

[12] CAPRITA B, NIEH J, CHAN W C. Group round robin: improving the fairness and complexity of packet scheduling[A]. Proc of the ACM/IEEE ANCS[C]. Princeton, New Jersey, USA, 2005. 29-40.

[13] BENNETT J C R, ZHANG H. WF^2Q: worst-case fair weighted fair queuing[A]. Proc of the IEEE INFOCOM[C]. New York, USA, 1996. 120-128.

[14] SHREEDHAR M, VARGHESE G. Efficient fair queuing using deficit round robin[J]. IEEE/ACM Transactions on Networking, 1996, 4(3): 375-385.

[15] PAREKH A K, GALLAGER R G. A generalized processor sharing approach to flow control in integrated services networks: the single-node case[J]. IEEE/ACM Transactions on Networking, 1993, 1(3): 344-357.

[16] YUAN X, DUAN Z. FRR: a proportional and worst-case fair round robin scheduler[J]. IEEE Transactions on Computers, 2009, 58(3): 365-379.

[17] COMER D, MARTYNOV M. Design and analysis of hybrid packet schedulers[A]. Proc of the IEEE INFOCOM 2008[C]. Phoenix, AZ, 2008. 1570-1578.

[18] GUO C X. Improved smoothed round robin schedulers for high-speed packet networks[A]. Proc of the IEEE INFOCOM[C]. Phoenix, AZ, 2008. 1579-1587.

[19] **张贤达. 矩阵分析与应用**[M]. **北京：清华大学出版社**, 2004. 101-106.
ZHANG X D. Matrix Analysis and Application[M]. Beijing: Tsinghua University Press, 2004. 101-106.

[20] **马祥杰，毛军鹏，兰巨龙. 输入排队** Crossbar **架构下的矩阵模型及** MM-LQF **调度策略**[J]. **电子学报**, 2008, 36(1):9-16.
MA X J, MAO J P, LAN J L. Matrix model for input-queued crossbar fabric and MM-LQF scheduling scheme[J]. Journal of Electronics, 2008, 36(1):9-16.

[21] HU H C, YI P, GUO Y F. Design and implementation of high performance simulation platform for switching and scheduling[J]. Journal of Software, 2008, 19(4):1036-1050.

作者简介：

张博（1982-），男，河北张北人，国家数字交换系统工程技术研究中心博士生，主要研究方向为可重构路由理论与技术、可重构交换理论与技术。

汪斌强（1963-），男，安徽人，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。

王珊珊（1983-），女，河北邢台人，国家数字交换系统工程技术研究中心博士生，主要研究方向为移动通信、无线资源管理和无线网络。

卫红权（1971-），男，河南唐河人，国家数字交换系统工程技术研究中心副研究员，主要研究方向为电信网信息安全。

李挥（1964-），男，广东汕头人，北京大学教授、博士生导师，主要研究方向为三网合一流媒体云计算网络视频关键技术研发、下一代网络体系结构研究、网络路由和宽带交换结构。