基于输入端无冲突算法的面向输出排队的交换结构

夏 羽 高志江 曾华燊
(西南交通大学信息科学与技术学院 成都 610031)

摘 要 该文提出了一种全新的面向输出排队的交换结构,该结构将信元存储于输入端,而信元的调度请求面向输出端排队.利用输入端无冲突调度算法,可以使结构对存储器带宽的需求和输入排队交换机一致.该文设计的调度矩阵使算法复杂度和端口规模呈线性关系,且每一步只需要一次按位"与"操作.文章同时证明了结构要达到稳定的充分条件是使用 2 倍传输加速比.仿真实验表明,对于均匀流量,该结构时延性能和主流交换结构相似,且抖动性能远优于主流交换结构,而在 2 倍传输加速比时,其时延性能和 OQ 完全相同;对于非均匀流量,该结构吞吐率性能优于主流交换结构,且在使用 1.14 倍传输加速比时,其吞吐率性能和 OQ 相同.

关键词 交换机架构;吞吐率;稳定性;输入排队;面向输出排队
中图法分类号 TP393 **DOI 号**: 10.3724/SP.J.1016.2010.01213

Output-Oriented Queued Switch Architecture with Input Conflict-Free Algorithm

XIA Yu GAO Zhi-Jiang ZENG Hua-Xin
(*School of Information Science and Technology, Southwest Jiaotong University, Chengdu* 610031)

Abstract This paper proposes a new packet switching architecture, called output-oriented queued (OOQ) switch. Unlike any existing switching architectures, OOQ stores cells at input units, while the requests for scheduling is queued at output units. With a novel input conflict-free (ICF) algorithm proposed by the authors, the bandwidth requirement of memories in the switch remains the same as in traditional input-queued switches. To support ICF algorithm, a hardware-based scheduling matrix is also proposed to make the complexity of the algorithm linear as the switch size. And each step of the scheduling consists of only one parallel bitwise "and" operation. This paper also prove that the sufficient condition for OOQ/ICF to be stable under any admissible traffic is to use the transfer speedup of 2. Finally, the simulation results show that under uniform traffic, delays of OOQ/ICF without speedup are comparable to the mainstream switches but with much better delay jitters. Furthermore, with the speedup of 2, the delay performance of OOQ/ICF is exactly the same as the output-queued switch. While under non-uniform traffic, the throughput of OOQ/ICF is better than current mainstream switches, and achieves 100% when using the speedup of 1.14.

Keywords switch architecture; throughput; stability; input-queued; output-oriented queued

1 引 言

交换机/路由器已经成为影响网络性能的关键设备,对于未来单波长速率高达 100Gbps 以上的高速骨干通信子网,对交换结构和调度算法的挑战更为严峻.早期交换机使用的共享存储(Shared Memory,SM)和输出排队(Output-Queued,OQ)结构虽

收稿日期:2009-12-04;最终修改稿收到日期:2010-06-22.本课题得到国家自然科学基金(60773102)、"中国工程科技中长期发展战略研究"联合基金(U0970122)和四川大学基金(下一代 Internet 体系结构)资助.**夏 羽**,男,1983 年生,博士研究生,主要研究方向为高速交换结构、交换调度算法.E-mail: rainsia@gmail.com. **高志江**,男,1985 年生,博士研究生,主要研究方向为高速交换结构的设计及实现. **曾华燊**,男,1945 年生,博士,教授,博士生导师,主要研究领域为网络体系结构、高速交换结构、路由器测试技术.

然被证明能够提供最好的性能，但却不能用于现代高速网络环境，原因在于它们需要存储器速率分别为 $2NR$（N 次写入和 N 次读出）和 $(N+1)R$（N 次写入 1 次读出），其中 N 为交换结构的端口数，R 为端口速率. 目前的商用存储器发展远远落后于交换机对存储器的需求，存储器已经成为交换机性能的瓶颈，而 OQ 交换机需要交换矩阵（switch fabric）工作速率为线速率的 N 倍，这也使其难以满足大规模高速交换的需要.

输入排队（Input-Queued，IQ）的交换结构只需要在每个时槽内对存储器读、写各一次，在同样的端口速率下，其对存储器速率的要求（$2R$）远远低于 OQ 交换机，且与交换规模无关，因此成为目前主流的交换结构. 但是 IQ 交换机受“队头阻塞”（Head-of-Line Blocking，HoL blocking）的制约，对均匀流量也只能提供 58%左右的吞吐率[1]. 采用虚拟输出队列（Virtual Output Queue，VOQ）可以解决“队头阻塞”问题，但加剧了输出竞争，需要额外的匹配调度算法. 基于 VOQ 的交换结构的仲裁配调度问题被抽象成图论中的二分图匹配（bipartite graph matching）问题，并有如下结论：（1）使用最大匹配（Maximize Size Matching，MSM）算法，其复杂度为 $O(N^{2.5})$，且不能达到 100%吞吐率；（2）使用最大权值匹配（Maximize Weight Matching，MWM）算法，如 LQF（Longest Queue First）、OCF（Oldest Cell First）或 LPF（Longest Port First），可以达到 100%的吞吐率[2]. 但是 MWM 算法的复杂度为 $O(N^3)$，这使得它难以在高速（high-speed）或大规模（large-scale）的交换机中使用. 随后的文献提出了大量时间复杂度较低的极大匹配（maximal Size Matching，mSM）算法，如著名的 PIM[3]、iSLIP[4] 和 DRRM[5] 等算法以及它们的大量改进算法都是对极大匹配算法的趋近. 然而，这些启发式算法（heuristic algorithms），最多也只能对均匀流量提供 100%的吞吐率，而对非均匀流量不能保证 100%的吞吐率. 文献[6]提出的 MQWS 算法在保持和 iSLIP 算法相似复杂度的条件下，可以对非均匀流量提供很高的吞吐率，但还是不保证在任意许可流量下均达到 100%吞吐率. 因此如何设计高效、稳定且高性能的匹配调度算法便成为基于 VOQ 的交换机设计中的核心问题. 另一种可以有效提高 IQ 交换结构的吞吐率，而不需要高复杂度调度算法的途径是使用加速比（speedup），输入端口接收一个信元的时间称为一个（外部）时槽（external time slot），在加速比为 s 的条件下，一个外部时槽内有 s 个信元可以从输入端通过交换矩阵被发送到输出端（其中传输一个信元的时间称为一个传输时槽，有的文献也称其为内部时槽（internal time slot）），而一个时槽内，只有一个信元可以从输出端口离开，因此输出端需要缓存，从而形成输入输出结合排队的（Combined Input & Output Queued，CIOQ）交换结构. 当加速比为 2 时，采用极大匹配算法（Maximal Size Matching）的 CIOQ 结构可以提供 100%的吞吐率.

上述交换结构的后续研究文献中还提出过多种新型交换结构，如基于 Buffered Crossbar 的交换结构[7]、Clos 的网络结构[8] 和负载均衡的 BvN（Load Balanced Birkhoff-von Newmen）交换结构[9]. 第 1 种结构需要使用交叉节点带缓存的 crossbar，而后两者属于多级交换结构. Buffered Crossbar 交换结构的规模受限于超大规模集成电路的工艺而性能受限于大往返时间（Long Round-Trip Time，Long RTT）问题；Clos 结构的内部冲突问题，需要额外的算法来解决，且其性能的好坏还要取决于其所使用的单级交换结构；负载均衡交换结构虽然无需调度算法但时延性能不佳，且存在分组失序问题，解决分组失序需要额外的算法，且会进一步降低时延性能甚至吞吐率. 这些问题都阻碍了这些新结构在工业界的大规模使用，因此目前主流交换机的设计和生产主要还是使用或者基于 crossbar 单级交换结构.

IQ 交换结构调度算法复杂度高且可达性能受限，而 OQ 交换结构性能较好却依赖于高速存储器以及具有 N 倍加速比的交换矩阵. 本文试图在交换性能、调度及实现复杂度之间找一个平衡点，提出了一种能够发挥两种典型存储排队方式的优势，同时调度与实现复杂度相对较低的面向输出排队（Output-Oriented Queued）的交换结构，它属于输入存储的单级 crossbar 交换结构. 本文设计的输入端无冲突（Input Conflict-Free，ICF）算法，可以保证在 2 倍传输加速比（无调度加速比）的条件下，即每块存储器只需要 $3R$ 的带宽（输入缓存为一次写入和两次读取，输出缓存为两次写入一次读取），使 OOQ 交换结构对任意许可流量提供 100%的吞吐率. 配合本文设计的调度矩阵，每个分组到达时，只需要一次并行按位“与”（bitwise and）操作即可完成调度. 仿真实验表明，在许可流量下，不需要加速比，该交换结构即可对均匀流量提供 100%的吞吐率，且时延和现有主流结构或算法相当，而由于 OOQ 结构能最大程度地保持信元先进先出的顺序，因此时延抖

动性能明显高于现有交换结构而和 OQ 相当；对非均匀流量，OOQ 也可以提供很高的吞吐率，当使用 1.14 倍传输加速比而无需调度加速比，就可以对各种非均匀流量提供 100％的吞吐率.

本文第 2 节介绍面向输出排队的交换结构和输入端无冲突算法；第 3 节证明交换结构传输加速比为 2 是 OOQ/ICF 结构对任意许可流量提供 100％吞吐率的充分条件；第 4 节探讨 ICF 调度矩阵的硬件实现；第 5 节通过仿真比较 OOQ/ICF 和其它主流交换结构和算法的性能；第 6 节将讨论 OOQ/ICF 交换结构为何能更好地支持 SUPANET 网络环境；第 7 节总结全文并展望下一步的工作.

2　面向输出排队的(OOQ)交换结构和输入端无冲突(ICF)算法

2.1　OOQ 交换结构

设一个交换机有 N 个输入端口和 N 个输出端口(记为 $N\times N$ 的交换机)，通常交换机内部只处理定长分组，称为信元(cell)，若输入端口接收的是变长分组，则将其切分(segment)为定长信元，而在输出端口再将属于同一个分组的信元重组(reassemble).

在传统的交换结构中，存储与排队同时进行. 例如传统的 IQ 结构，信元存储在输入端，排队也在输入端，而 OQ 结构信元存储于输出端，而排队也在输出端. 在我们的新交换结构中，如图 1 所示，实际的信元存储于输入端的缓存中，而排队则在调度模块中进行，且队列被组织成面向输出端的方式，因此我们将其称为面向输出排队(Output-Oriented Queued，OOQ)的交换结构.

当一个信元从某输入端口到达，该输入单元在将信元存储于输入缓存的同时向调度组件发送一个排队请求. 调度组件接收到输入单元发送的请求后，使用相应的调度算法，在对应的输出端请求队列中找一个合适的位置，将该请求入队. 每个传输时槽开始时，输出单元取出队头请求，并配置交换矩阵从输入单元取得对应的信元并发送，此过程称为“拉”. 在传统的匹配算法(如 iSLIP)中，一个信元可能要请求多次才能匹配成功，而在 OOQ 结构中，一个信元一旦请求即可成功入队，从而避免了在输入单元和调度组件之间多次交换信息.

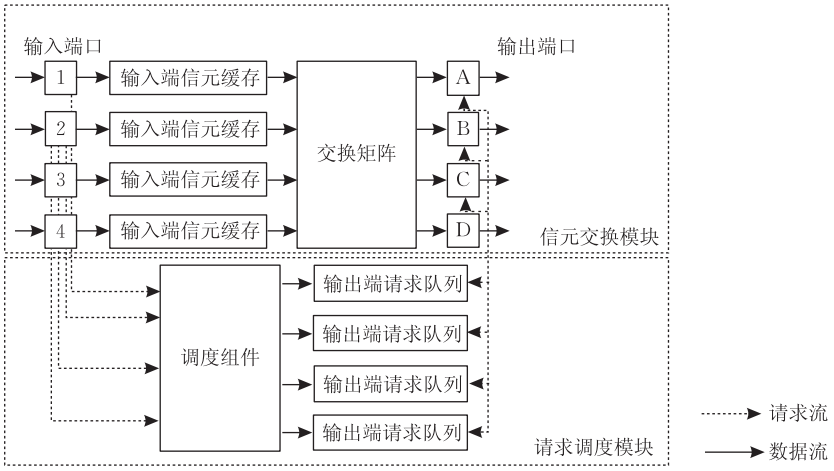


图 1　面向输出排队的交换结构

由于受到输入端口速率的制约，一个时槽内，只会有一个信元到达，因此对于输入端缓存只有一次写入. 但是在同一个时刻，一个输入单元可能同时被多个输出单元“拉”，然而，一个输入单元只能同时响应一个输出单元的拉请求. 因此需要额外的调度算法来解决这个冲突. 2.2 节将介绍的输入端不冲突(ICF)算法可以保证，在同一时刻，最多只有一个输出单元从一个输入端“拉”信元.

2.2　ICF 算法

和传统的使用 FIFO 队列(First In First Out 队列，只能从队尾写入，从队头读出)的交换结构不同，OOQ/ICF 使用 RIFO 队列(Random In First Out 队列，可以在队列中任意为空的位置写入，但只能从队头读出)作为 OOQ 交换结构的输出端请求队列. 由于存储器可以随机访问(random access)，对于定长信元，只要给定入队位置，则 RIFO 的实现和 FIFO 的实现复杂度相同.

在 OOQ 交换机中，一个输入单元可能收到多个拉请求，为了解决这一冲突，我们希望算法可以将请求安排到输出端请求队列中一个合适的位置，该

位置在输出时不会和其它输出单元发生“拉”冲突.为此,我们定义输入端无冲突位置.

定义 1. 输入端无冲突(Input Conflict-Free,ICF)位置:对于某个输入单元 i 发送到输出单元 j 的请求来说,输出端 j 的请求队列中的某一空闲位置,如果在其它输出端 $k\neq j$ 的请求队列中的同一位置上,没有来自输入单元 i 的请求,则该位置被称为是关于输入端 i 无冲突的位置.

对于特定的请求来说,输入端无冲突位置也称为可用位置(available location),如果将每个请求都入队到对应输出端请求队列中的可用位置,则在所有输出端请求队列中的同一列上,不会存在来自同一输入单元的请求.从而在任一时刻,不同的输出单元不会同时向同一个输入单元“拉”信元.因此每次输出时,输入缓存只需读取一次,最终需要的存储器速率仅为 $2R$,和 IQ 交换机一致.

在信元到达的时刻,对应的请求队列中可能存在多个可用位置,如果在其中随机地选择一个位置放置请求,则信元离开的顺序可能会与信元到达的顺序不同,即可能产生错序.为防止错序,我们始终将输入端发送的排队请求安排在首个可用位置(最接近队头的位置),对于从同一个输入端口进入而指向同一个输出端口的所有信元,先到达的信元的请求一定排在后到达信元的请求之前,而请求只能从队头取出,因此可以确保信元不会错序.对于一个信元,只要它的请求在 OOQ/ICF 中被入队,则其离开时间(排队时延)就已经确定.

ICF 的执行过程可以分为输入调度和输出调度两个独立的过程,其具体过程可分别描述如下.

输入端.

在每个时槽开始时

1. 对于每一个输入端口,如果该输入端有分组到达(各个输入端口之间并行执行):
   1.1. 将该分组缓存于对应的输入端缓存中;
   1.2. 向调度组件发送一个请求,该请求包含了对应的输入、输出端口号以及分组在输入端缓存中的地址;
2. 对于每一个输入端发送的请求:
   2.1. 调度组件在对应的输出端请求队列中查找首个输入端无冲突位置;
   2.2. 将请求入队到首个可用位置;

步 2.1 的查找过程是整个算法的核心,我们将在第 4 节中介绍专门设计的硬件(ICF 调度矩阵)来实现快速查找.使用 ICF 调度矩阵,每次查找只需要做一次寄存器的按位“与”操作.

输出端.

在每个时槽开始时

1. 对于每一个输出端请求队列(各个输出端口之间并行执行):
   1.1. 如果队首是空请求:
      1.1.1. 移除队首,队列中所有元素向输出端方向移动一个单位;
   1.2. 如果队首不是空请求:
      1.2.1. 取出队首请求(请求中包含输入端口号和分组在缓存中的位置),队列中所有元素向输出端方向移动一个单位;
      1.2.2. 配置交换矩阵;
      1.2.3. 向对应输入端发送拉信号,拉信号中包含输出端口号和分组在缓存中的位置;
      1.2.4. 对应输入端口接收到拉信号后,将对应分组发送至输出端口输出;

由于输入调度过程已经解决了拉冲突,输出时已经不存在冲突,而仅仅只是取出请求,然后拉对应的分组,随后输出,该过程将非常简单.

为了更好地理解 ICF 调度算法,在图 2 中,我们列举了 ICF 的调度过程的实例.如图 2(a)所示,输入端口 1 有一个指向输出端口 A 的分组到达,则输入单元 1 发送一个请求给调度模块,对于该请求来说,位置 4、5 是输入端无冲突位置.虽然位置 2 未被任何请求占用,但是输出端 B 的请求队列中,位置 2 有一个来自输入单元 1 的请求,因此位置 2 是输入冲突位置,不能使用.根据选择首个可用位置的原则,该请求应该被放置在位置 4.如图 2(b)所示,输入端口 1 到达一个指向输出端口 C 的分组,则输入端口 1 向调度模块发送请求,对于该请求来说,位置 3、5 是输入端无冲突位置,因此选择首个可用位置 3 放置该请求.从同一列(图中的虚线框)上看,所有请求都来自不同的输入单元,所以不会产生“拉”冲突.每一个传输时槽开始时,所有输出端口都同时取出队头请求,而不管该队头请求是否为空,例如输出端口 A 的请求队列中的位置 2 为一个空请求,当该位置到达队头时还依然为空请求,则输出端口 A 将该空请求移除,输出端口 A 在该传输时槽内将处于空闲状态.

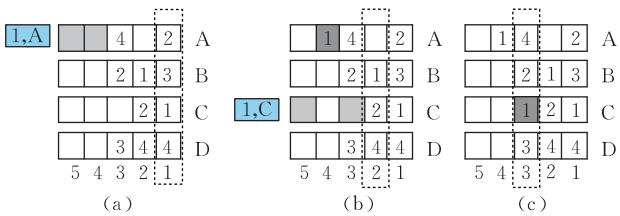


图 2 ICF 实例

3　OOQ/ICF 结构稳定性分析

在 IQ 交换机的设计中，可以通过提高加速比来提高交换吞吐率. 本节将证明 2 倍传输加速比足够让 OOQ/ICF 结构在任意许可流量下达到 100% 的吞吐率.

3.1　符号和定义

对于一个 $N\times N$ 的基于 VOQ 的交换机，其调度问题可转化为图论中的二分图匹配问题. 用随机过程 $A_{i,j}(n)\in\{0,1\}$ 表示时槽 n 从输入端口 i 到达并指向输出端口 j 的信元数，称方阵 $\boldsymbol{A}(n)=[A_{i,j}(n)]_{N\times N}$ 为到达过程矩阵. 其均值 $E[A_{i,j}]=\lambda_{i,j}\leq 1$ 称为从输入端口 i 到达并指向输出端口 j 的信元到达速率，而方阵 $\boldsymbol{\Lambda}=[\lambda_{i,j}]_{N\times N}$ 称为到达速率矩阵.

定义 2.　许可流量(admissible traffic). 如果到达过程 $\boldsymbol{A}(n)$ 满足如下条件：

$$\sum_{i=1}^{N}\lambda_{i,j}\leq 1,\ \sum_{j=1}^{N}\lambda_{i,j}\leq 1,$$

则称其为许可流量.

用方阵 $\boldsymbol{M}=[m_{i,j}]_{N\times N}$ 表示匹配矩阵，其中 $m_{i,j}=1$ 表示输入端口 i 和输出端口 j 匹配，而 $m_{i,j}=0$ 表示输入端口 i 和输出端口 j 无匹配. 该匹配矩阵其实就是 crossbar 的配置矩阵，因此 $\boldsymbol{M}$ 是一个置换矩阵(permutation matrix). 用 $t_{i,j}(n)$ 表示在时槽 n 时，$VOQ_{i,j}$ 的队头信元的等待时间，则方阵 $\boldsymbol{T}(n)=[t_{i,j}(n)]_{N\times N}$ 称为在时槽 n 的(队头)等待时间矩阵. 用 $d_{i,j}(n)$ 表示在时槽 n 时，队列 $VOQ_{i,j}$ 中的队头信元和其下一个信元的到达时间之差，则方阵 $\boldsymbol{D}(n)=[d_{i,j}(n)]_{N\times N}$ 称为时槽 n 时的到达时间间隔矩阵.

对于任意 N 阶方阵 $\boldsymbol{X}=[x_{i,j}]_{N\times N}$，可以定义其对应的向量形式 $\boldsymbol{X}=(x_{1,1},\cdots,x_{1,N},\cdots,x_{N,N})$. 而对于两个向量 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$ 和 $\boldsymbol{Y}=(y_1,y_2,\cdots,y_n)$，有 $\boldsymbol{X}\boldsymbol{Y}^{\mathrm{T}}=\sum_{i=1}^{n}x_iy_i$，其结果是一个数值.

3.2　稳定性证明

文献[10]证明了强稳定(strongly stable，简称稳定)的排队系统可以保证 100% 的吞吐率，并介绍了如何用 Lyapunov 方法来证明稳定性. 文献[2]用该方法证明了 MWM/OCF(Maximum Weight Matching/Oldest Cell First)算法对任意许可流量保证稳定. MWM 算法属于线性规划算法，它从全局的角度来优化匹配，从而可以保证得到的最终权值最优. 对于规划问题，另一种复杂度相对较低的算法是贪婪算法(greedy algorithm)，它只在每一步寻求最优解，但是最终结果不一定全局最优. 贪婪极大匹配算法(Greedy Maximal Weight Matching/Oldest Cell First，GMWM/OCF)过程如下.

1. 将所有 VOQ 队头信元的等待时间写入矩阵 $\boldsymbol{T}(n)$，如果 VOQ 为空，则相应元素为空，矩阵 $\boldsymbol{M}(n)$ 元素全置为 0.

2. 如果矩阵 $\boldsymbol{T}(n)$ 中所有的元素都为空，则结束并返回矩阵 $\boldsymbol{M}(n)$，否则继续步 3.

3. 从矩阵 $\boldsymbol{T}(n)$ 的所有元素中选择一个最大的 $t_{i,j}(n)$.

4. 将矩阵 $\boldsymbol{M}(n)$ 中对应的元素 $m_{i,j}(n)$ 设为 1.

5. 将矩阵 $\boldsymbol{T}(n)$ 中第 i 行的所有元素置为空，第 j 列的所有元素也置为空. 然后返回步 2.

本小节我们先证明 ICF 算法调度的结果和 GMWM/OCF 调度的结果一致. 随后再证明 GMWM/OCF 算法达到稳定的充分条件是使用 2 倍加速比.

定理 1.　ICF 调度算法的调度结果和 GMWM/OCF 算法的调度结果一致.

证明.　如果所有请求都按其到达顺序被插入到队列中，那么很容易可以证明 ICF 和 GMWM/OCF 算法具有相同的调度结果. 因此我们主要证明在请求插入队列时错序的情况下，两种算法仍然具有同样的调度结果.

若有两个请求 A 和 B，如果 B 比 A 晚到达，但是 B 却插入到了 A 的前面. 如图 3 所示，这有两种可能情况.

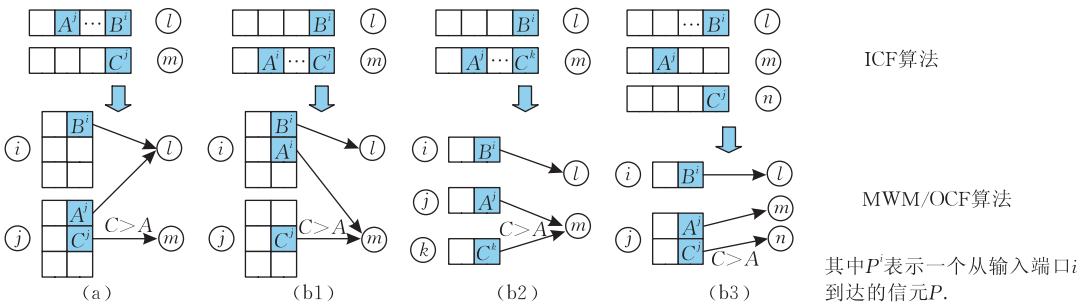


图 3　OOQ/ICF 和 GMWM/OCF 逻辑关系(相同)示意

(a) A 和 B 在同一个请求队列 l 中，若请求 A 和 B 是从同一个输入端口到达的，则它们一定是按到达顺序插入队列的. 因此我们只需证明 A 和 B 从不同的输入端口到达的情况，则此时在另外一个队列 m 中一定存在一个与 A 到达端口相同的请求 C，C 的到达时间一定早于 A，否则，当 A 到达时 B 的位置还不存在冲突，因此 A 应该插入到 B 的位置上. 而对于 GMWM/OCF 算法，由于 A 与 C 关于输入端口冲突，它也将选择 C 来匹配，而不是 A，因为 C 的等待时间大于 A.

(b) A 和 B 在不同队列中，假设 B 在队列 l 中，而 A 在队列 m 中. 这时存在 3 种情况：

(b1) A 和 B 从同一个输入单元到达，则在队列 m 中 B 的位置上一定存在一个和 A 从不同端口到达的请求 C，且 C 的到达时间一定早于 A，否则当 A 到达时 B 的位置还没有被占用，则 A 将被插入到队列 m 中 B 的位置. 同样对于 GMWM/OCF 算法，由于 A 与 C 关于输出端口冲突，它也将选择 C 而不是 A 来匹配，因为 C 具有更长的等待时间.

(b2)和(b3)都是 A 和 B 从不同输入单元到达的情况，无论在哪一种情况中，A 一定是被一个更早到达的信元 C 所阻挡，这和在 GMWM/OCF 中的情况是一致的.

在所有情况下，ICF 和 GMWM/OCF 都具有同样的调度结果，因此我们说 ICF 和 GMWM/OCF 在逻辑上相同. 证毕.

文献[11]中的定理 4 提供了一种比较不同调度算法稳定性的工具. 我们将利用此工具比较 GMWM/OCF 算法和 MWM/OCF 算法的稳定性，从而最终证明 GMWM/OCF 算法稳定的充分(非必要)条件是使用两倍加速比. 为此我们先证明如下引理.

引理 1. 如果一个调度算法 $\mathcal{P}$ 满足如下条件.

$$\boldsymbol{M}_i^{\mathcal{P}}(n)\boldsymbol{W}_i^{\mathrm{T}}(n)\geqslant\frac{1}{2}[\boldsymbol{M}_i^{\mathrm{MWM}}(n)\boldsymbol{W}_i^{\mathrm{T}}(n)+K_i(n)],\quad i=1,2 \tag{1}$$

其中 $\boldsymbol{W}_i(n)$ 是第 n 个时槽，第 i 个内部时槽的权值向量，而 $K_i(n)\geqslant 0$ 是同时被算法 $\mathcal{P}$ 和 MWM/OCF 选中的各队列在队头信元离开之后的权值变化量之和，则一个采用算法 $\mathcal{P}$ 的交换机稳定的充分条件是使用两倍加速比.

证明. 为简短起见，我们这里只基于 GMWM/OCF 算法证明. 对于 GMWM/OCF 算法，权值变化就是队列中队头信元到达时间和次信元到达时间的差值，但如果队列中没有次信元，则队头信元离开后，队列的权值将变为 0，此时权值变化即等于队头信元的等待时间. 假设 GMWM/OCF 算法符合式(1)，则可分解得到如下两个公式：

$$\boldsymbol{M}_1^{\mathrm{GMWM}}(n)\boldsymbol{T}_1^{\mathrm{T}}(n)\geqslant\frac{1}{2}[\boldsymbol{M}_1^{\mathrm{MWM}}(n)\boldsymbol{T}_1^{\mathrm{T}}(n)+K_1(n)] \tag{2}$$

$$\boldsymbol{M}_2^{\mathrm{GMWM}}(n)\boldsymbol{T}_2^{\mathrm{T}}(n)\geqslant\frac{1}{2}[\boldsymbol{M}_2^{\mathrm{MWM}}(n)\boldsymbol{T}_2^{\mathrm{T}}(n)+K_2(n)] \tag{3}$$

根据 MWM 的定义，即 $\boldsymbol{M}_2^{\mathrm{MWM}}(n)\boldsymbol{T}_2^{\mathrm{T}}(n)=\max_{\boldsymbol{M}}\{\boldsymbol{M}\boldsymbol{T}_2^{\mathrm{T}}(n)\}$，其中 $\boldsymbol{M}$ 表示所有可能的 $N!$ 种匹配；并且第 1 个内部时槽中可能有信元离开队列，从而到第 2 个内部时槽时权值可能减小，因此有

$$\begin{aligned}\boldsymbol{M}_2^{\mathrm{MWM}}(n)\boldsymbol{T}_2^{\mathrm{T}}(n)&\geqslant\boldsymbol{M}_1^{\mathrm{MWM}}(n)\boldsymbol{T}_2^{\mathrm{T}}(n)\\&=\boldsymbol{M}_1^{\mathrm{MWM}}(n)\boldsymbol{T}_1^{\mathrm{T}}(n)-K_1(n)\end{aligned} \tag{4}$$

同样对于 GMWM 算法，第一个内部时槽也会有信元离开，因此有

$$\boldsymbol{M}_2^{\mathrm{GMWM}}(n)\boldsymbol{T}_1^{\mathrm{T}}(n)\geqslant\boldsymbol{M}_2^{\mathrm{GMWM}}(n)\boldsymbol{T}_2^{\mathrm{T}}(n) \tag{5}$$

则由式(3)～(5)可以得到

$$\begin{aligned}&\boldsymbol{M}_2^{\mathrm{GMWM}}(n)\boldsymbol{T}_1^{\mathrm{T}}(n)\\&\geqslant\frac{1}{2}[\boldsymbol{M}_1^{\mathrm{MWM}}(n)\boldsymbol{T}_1^{\mathrm{T}}(n)-K_1(n)+K_2(n)]\end{aligned} \tag{6}$$

最后合并式(2)和(6)可以得到

$$\begin{aligned}&[\boldsymbol{M}_1^{\mathrm{GMWM}}(n)+\boldsymbol{M}_2^{\mathrm{GMWM}}(n)]\boldsymbol{T}_1^{\mathrm{T}}(n)\\&\geqslant\boldsymbol{M}_1^{\mathrm{MWM}}(n)\boldsymbol{T}_1^{\mathrm{T}}(n)+\frac{1}{2}K_2(n)\\&\geqslant\boldsymbol{M}_1^{\mathrm{MWM}}(n)\boldsymbol{T}_1^{\mathrm{T}}(n)\end{aligned} \tag{7}$$

应用文献[11]中的定理 4 和式(7)可以证明在两倍加速比下算法 $\mathcal{P}$ 稳定. 证毕.

我们也可以从一般意义上来解释式(7)：以两倍加速比运行 $\mathcal{P}$ 算法(即一个时槽运行两次)得到的权值大于等于 MWM 算法得到的权值，因为 MWM 算法稳定，则 $\mathcal{P}$ 算法必然也稳定.

引理 2. 对于任意一个非空的 $VOQ_{i,j}$，用 $w_{i,j}$ 表示在队头分组离开后该队列权值的变化量，那么一定有如下关系成立：

$$t_{i,j}\geqslant w_{i,j} \tag{8}$$

其中 $t_{i,j}$ 表示在队头信元离开前的等待时间.

证明. 存在两种情况.

(a) 该队列中队头分组之后没有分组. 在这种情况下，队头分组离开之后队列的权值将变为 0，那么队列的权值变化量直接等于队头分组的等待时间 $t_{i,j}$.

(b) 如果队头分组之后有其它分组，为了不失一般性，我们假设当前时槽为 n，队头分组的到达时

间为 k，而队头分组后的分组的到达时间为 $k+m$，其中 m 是这两个分组的到达时间间隔. 证明的关键是注意到 $m=w_{i,j}$ 以及 $t_{i,j}=n-k$. 由于当前队头分组后面的分组已经存在，也就是说 $k+m\leq n$，因此可以得到 $t_{i,j}=n-k\geq m=w_{i,j}$.

综合两种情况，最终可以证明式(8). 证毕.

由于在引理 1 中，我们假设 GMWM/OCF 满足式(1)，下面我们将证明其确实满足式(1)，从而最终证明 GMWM/OCF 算法在两倍加速比条件下稳定.

定理 2. 一个使用 GMWM/OCF 调度算法的交换机稳定的充分(非必要)条件是使用两倍加速比.

证明. 根据引理 1 只需要证明 GMWM/OCF 算法在每个内部时槽满足如下条件：

$$\boldsymbol{M}^{\mathrm{GMWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)>\frac{1}{2}[\boldsymbol{M}^{\mathrm{MWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)+K(n)] \tag{9}$$

其中 $K(n)$ 是同时被算法 $\mathcal{P}$ 和 MWM 算法选中的队列在队头信元离开之后的权值变化之和，即可证明其稳定性.

分别让 $I_{(1)}^{\mathrm{MWM}}(n)$ 和 $I_{(1)}^{\mathrm{GMWM}}(n)$ 表示被 MWM 算法和 GMWM/OCF 算法选中的队列的集合. 那么该集合之间只有如下两种关系.

情况 1. $I_{(1)}^{\mathrm{MWM}}(n)=I_{(1)}^{\mathrm{GMWM}}(n)$，则表明 MWM 算法和 GMWM/OCF 算法选择了相同的队列，因此它们选中队列的权值之和也相等：$\boldsymbol{M}^{\mathrm{GMWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)=\boldsymbol{M}^{\mathrm{MWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)$，因此 GMWM/OCF 算法稳定.

情况 2. $I_{(1)}^{\mathrm{MWM}}(n)\neq I_{(1)}^{\mathrm{GMWM}}(n)$，则令 $g_{i,j}^{(1)}(n)$ 表示集合 $I_{(1)}^{\mathrm{GMWM}}(n)$ 中具有最大队头等待时间的队列，则又有两种情况：

情况 2.1. $g_{i,j}^{(1)}(n)\in I_{(1)}^{\mathrm{MWM}}(n)$，即集合 $I_{(1)}^{\mathrm{GMWM}}(n)$ 与集合 $I_{(1)}^{\mathrm{MWM}}(n)$ 中都包含队列 $g_{i,j}^{(1)}(n)$. 因此

$$2W(g_{i,j}^{(1)})=W(g_{i,j}^{(1)})+W(g_{i,j}^{(1)}) \tag{10}$$

其中 $W(c)$ 表示队列 c 的队头信元的等待时间，然后令

$$I_{(2)}^{\mathrm{MWM}}(n)=I_{(1)}^{\mathrm{MWM}}(n)-\{g_{i,j}^{(1)}(n)\},$$
$$I_{(2)}^{\mathrm{GMWM}}(n)=I_{(1)}^{\mathrm{GMWM}}(n)-\{g_{i,j}^{(1)}(n)\}.$$

情况 2.2. $g_{i,j}^{(1)}(n)\notin I_{(1)}^{\mathrm{MWM}}(n)$，则集合 $I_{(1)}^{\mathrm{MWM}}(n)$ 中最多可能包含两个在输入或者输出端口上与 $g_{i,j}^{(1)}(n)$ 冲突的队列，设它们分别是 $c_{i,j^*}^{(1)}(n)$ 与 $c_{i^*,j}^{(1)}(n)$. 又由于 $g_{i,j}^{(1)}(n)$ 具有最大的队头等待时间，所以有

$$2W(g_{i,j}^{(1)}(n))\geq W(c_{i,j^*}^{(1)}(n))+W(c_{i^*,j}^{(1)}(n)) \tag{11}$$

然后令

$I_{(2)}^{\mathrm{GMWM}}(n)=I_{(1)}^{\mathrm{GMWM}}(n)-\{g_{i,j}^{(1)}(n)\}$ 以及

$$I_{(2)}^{\mathrm{MWM}}(n)=I_{(1)}^{\mathrm{MWM}}(n)-\{c_{i,j^*}^{(1)}(n),c_{i^*,j}^{(1)}(n)\}.$$

则此时 $I_{(2)}^{\mathrm{MWM}}(n)$ 中不再含有与 $g_{i,j}^{(1)}(n)$ 冲突的队列.

接下来再考虑 $I_{(2)}^{\mathrm{GWMW}}(n)$ 中队头等待时间最长的队列 $g_{i,j}^{(2)}(n)$，使用同样的处理方法处理集合 $I_{(i)}^{\mathrm{GMWM}}(n)$ 和 $I_{(i)}^{\mathrm{MWM}}(n)$，直到集合 $I_{(i)}^{\mathrm{GMWM}}(n)$ 为空为止. 假设集合 $I_{(k)}^{\mathrm{GMWM}}(n)$ 为空，则集合 $I_{(k)}^{\mathrm{MWM}}(n)$ 只可能包含空队列. 如果 $I_{(k)}^{\mathrm{MWM}}(n)$ 包含非空队列 $c_{i,j}(n)$，则 $c_{i,j}(n)$ 一定与 $I_{(1)}^{\mathrm{GMWM}}(n)$ 中的任何队列都不冲突. 而 GMWM/OCF 是极大匹配算法，因此这样的 $c_{i,j}(n)$ 是不存在的.

注意式(10)表示集合 $I_{(i)}^{\mathrm{MWM}}(n)$ 和 $I_{(i)}^{\mathrm{GMWM}}(n)$ 包含相同队列的情况，而式(11)表示集合 $I_{(i)}^{\mathrm{GMWM}}(n)$ 与 $I_{(i)}^{\mathrm{MWM}}(n)$ 包含不同队列的情况. 这两种情况之间互斥，且在每一步中，两者有一个成立. 因此将式(10)与(11)合并可以得到

$$\boldsymbol{M}^{\mathrm{GMWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)>\frac{1}{2}[\boldsymbol{M}^{\mathrm{MWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)+V(n)].$$

注意此时的 $V(n)$ 是同时被 GMWM/OCF 算法和 MWM/OCF 算法选中的队列的队头等待时间之和. 而由引理 2 可以推导出：

$$\begin{aligned}\boldsymbol{M}^{\mathrm{GMWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)&>\frac{1}{2}[\boldsymbol{M}^{\mathrm{MWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)+V(n)]\\&\geq\frac{1}{2}[\boldsymbol{M}^{\mathrm{MWM}}(n)\boldsymbol{T}^{\mathrm{T}}(n)+K(n)].\end{aligned}$$

因此最终可以证明 GMWM/OCF 算法稳定的充分条件是使用两倍的加速比. 证毕.

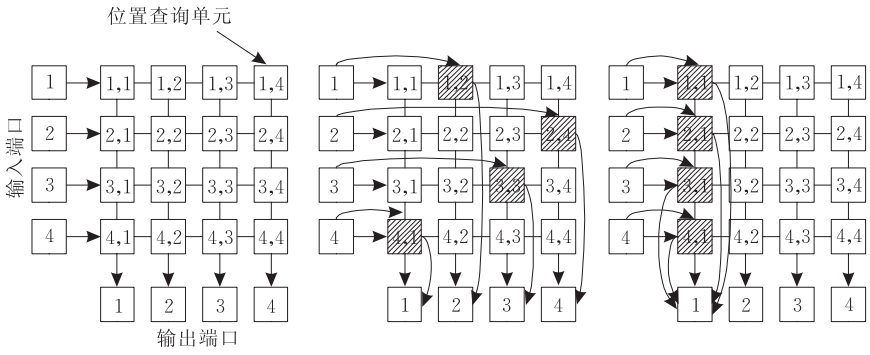
由于 ICF 调度算法最终从逻辑上等效于 GMWM/OCF 算法，所以 ICF 算法稳定的充分条件也是使用两倍加速比. 但是 GMWM/OCF 算法最多需要 N 次迭代才能得出一个极大匹配，且第 i 次迭代最多需要从 $(N-i)^2$ 个数中查找一个最大值. 如果还要使用两倍加速比，则每个时槽需要运行算法两次. 在高速交换机中，GMWM/OCF 算法太过于复杂. 而对于 ICF 算法，由于每个时槽最多有 N 个信元到达，因此每个时槽只需要运行算法一次，即可将所有请求调度完成，而对于输出过程，只需从队头取出请求即可. 下一节介绍的调度矩阵可以快速地查找出请求的入队位置.

4　ICF 调度矩阵的实现

ICF 调度器的工作方式不同于任何其它基于 crossbar 的调度器，传统的 crossbar 调度器在信元需要输出时执行调度，而 ICF 调度模块在信元到达时执行调度，且调度的结果即为 crossbar 的配置矩

阵.它们的区别在于:如果交换结构使用 s 倍加速比,则传统的 crossbar 调度器一个时槽必须执行 s 次,而 ICF 只需在每个时槽开始时执行一次,因为一个时槽内每个输入端口最多到达一个信元,而 ICF 调度矩阵一次可处理 N 个排队请求.

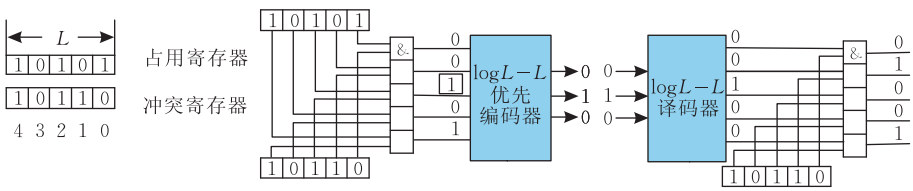
下面介绍 ICF 调度模块的核心组件:ICF 调度矩阵,该矩阵的主要功能是在接收到输入单元发送的请求后,在相应的输出端请求队列中快速查找到首个可用位置,然后将该请求入队.如图 4(a)所示,位置查询单元(Location Searcher,LS)维护相应的输出请求队列的占用和冲突状态.其中 $LS_{i,j}$ 用于为从输入单元 i 到达的请求在输出端 j 的请求队列中查找首个可用位置,一个 LS 查找到相应位置后,会将该位置广播给同行同列的 LS,以便它们及时更新占用和冲突的状态.图 4(b)展示了一种理想情况,各个输入端口发送的信元是指向不同的输出端口,则各个 LS 可以同时调度.图 4(c)显示了一种极端情况,各个输入端到达的信元都指向同一输出端口,由于各个 LS 独立工作,因此它们选择的位置可能存在冲突,因此每一步只能有一个 LS 进行查找位置,等完成后,将其结果广播给同行同列的 LS 更新它们的状态,然后下一个 LS 再执行查找,直到所有请求处理完成.



(a) 调度矩阵示意图 (b) 理想情况 (c) 输出冲突情况

图 4 调度矩阵示意图

LS 的核心功能就是在对应输出端请求队列中找首个可用位置,在查找时,关键要排除两种位置:(1)队列中已经被占用的位置;(2)队列中没有被占用,但是和其它队列存在输入端冲突的位置.假设输出端请求队列最大队长为 L,则我们在每个 LS 中使用两个容量为 L 位的寄存器分别表示队列中各个位置的占用情况和冲突情况,如图 5(a)所示,这两个寄存器分别称为占用指示寄存器(Occupation Indication Register,OIR)和冲突指示寄存器(Conflict Indication Register,CIR),而占用寄存器中的各个位称为占用指示位(Occupation Indicator,OI),1 表示该位置未被占用,而 0 则表示该位置被占用;类似地,冲突寄存器中的各个位称为冲突指示位(Conflict Indicator,CI),为 1 表示该位置没有和其它队列相同位置发生冲突,为 0 则表示发生冲突.图 5(a)中位置 0 虽然未被占用,但是该位与其它队列冲突;位置 1 虽然未与其它队列冲突,但已经被占用;位置 3 既被占用也和其它队列中的相同位置冲突;位置 2 和位置 4 都未被占用和冲突,都属于可用位置.因此 OIR 和 CIR 按位与的结果中所有为 1 的位置都是可用位置.由于需要找到首个可用位置,如图 5(b)所示,我们引入了优先级编码器,优先编码器可以处理优先级最高的输入,并编码输出.编码的另一个目的是减少结果的位宽,便于在各个 LS 之间广播数据.同行的 LS 接收到广播的结果之后将更新其 CIR,而同列的 LS 接收到广播结果之后将更新其 OIR.CIR 和 OIR 更新过程相同,如图 5(c)所示,一个 LS 接收到广播数据之后,先通过二进制

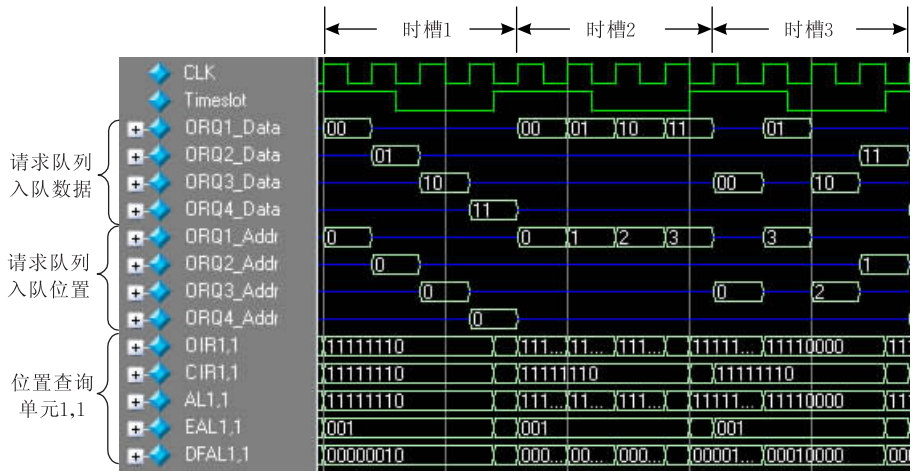


(a)占用和冲突寄存器 (b)首个无冲突位置查找、编码过程逻辑图 (c)寄存器更新过程逻辑图

图 5 位置查询单元工作原理逻辑图

译码器将位置信息恢复，其中为 1 的位置是需要更新的位置，将其取反之后再与寄存器中原来的数据按位与，最后得到更新结果. 对于调度矩阵，每个时槽最多只要做 N 次查找即可将所有请求都入队，属于线性时间复杂度，而每次查询只需一次寄存器的并行“与”操作，由于“与”和优先编码都属于组合逻辑，因此该过程只存在门时延.

我们使用 Verilog 语言对一个 4×4 的 OOQ 交换机的调度矩阵的调度情况进行了验证，图 6 为运行过程中 3 个时槽的波形图，该调度矩阵共有 16 个位置查询单元，图中只显示了位置查询单元 1,1 的状态. 时槽 1 为 4 个输入端口到达的信元分别指向不同输出端口的调度结果；时槽 2 为所有输入的信元都指向同一个输出端口的调度结果；时槽 3 是部分输入端口到达的信元指向同一个输出端口的调度结果. 从图中可以看出，调度结果与预期的一致. 该调度矩阵已经在 Altera 的 FPGA 上验证通过.



ORQ: 输出端请求队列, Data: 向队列插入的值, Addr: 向队列插入时的位置
OIR: 占用寄存器, CIR: 冲突寄存器, AL: 可用位置, CLK: 系统时钟, Timeslot: 时槽
EAL: 可用位置优先编码结果, DFAL: 首个可用位置二进制解码结果

图 6　4×4 交换机调度矩阵硬件仿真波形图

5　仿真实验

我们在为 NS-3(Network Simulator 3)建立的交换机仿真平台[12]上对 OOQ/ICF 和主流交换结构进行了仿真实验. 所有仿真中我们都设置端口数为 32(其它端口数情况有类似结果). 每一个仿真都运行 10^6 个时槽，并从第 10^3 个时槽开始收集数据. 平均时延为所有收集到的时延的平均值，而时延的变化为收集到的时延偏离均值的情况. 吞吐率的统计方法为在数据采集时段内，所有离开交换机的分组数和所有到达交换机分组数的比值. 限于篇幅，本文只列举两组主要的仿真结果.

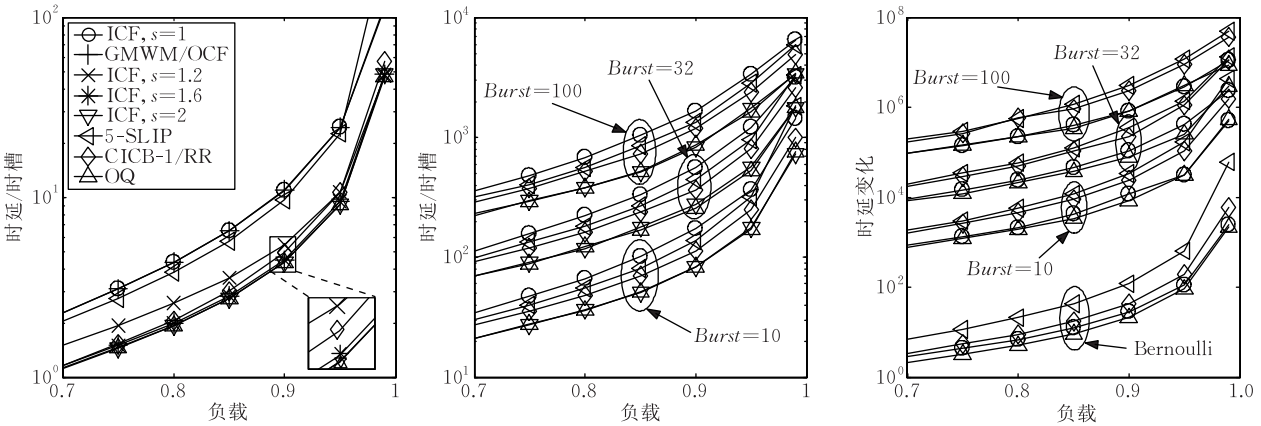
第 1 组仿真是对均匀分布流量下时延和抖动的仿真，由于在低负载下各算法性能比较接近，从而曲线不易区分，因此在图中，所有负载均从 70%开始，以便可以区别各条曲线. s 为传输加速比，$Burst$ 为突发流量长度.

图 7(a)为 Bernoulli 流量下的仿真结果，从中可见，ICF 算法和 GMWM/OCF 算法性能完全一致，这和分析是一致的. ICF 算法和 5-SLIP 算法时延性能相似但略差于后者(但在实际中，iSLIP 算法一般都只会实现 3 次迭代而不是 5 次). 而使用 1.2 倍传输加速比的 ICF 算法性能相对 5-SLIP 有大幅度提高，值得注意的是，ICF 算法提高传输加速比时，并不需要提高调度加速比，而这个特性对于其它算法并不成立. 使用 1.6 倍传输加速比时，OOQ/ICF 结构的时延性能将优于 CICB-1/RR 结构而和 OQ 非常接近，其中 CICB-1/RR 表示交叉点带一个缓存的 Buffered Crossbar 交换机，且输入端和输出端均使用轮询(Round-Robin)调度机制. 使用 2 倍传输加速比时，OOQ/ICF 时延性能和 OQ 完全一致.

图 7(b)所示为 ON/OFF 突发流量下的时延性能仿真，其中处于 ON 和 OFF 状态的时间均服从几何分布，处于 ON 状态时，每一个时槽都发送一个信元. 处于 ON 状态的时间的均值称为突发长度，仿真中我们取 10、32 和 100 分别表示小突发、中等突发和大突发. 同一突发内到达的信元都指向同一输出端口，而突发之间的指向在各个输出端口间均匀分布. 从图中可见，在这 3 种突发长度下，各结构时延

性能之间的关系与 Bernoulli 流量下类似.但随着突发长度的增大,ICF、iSLIP 和 CICB-1/RR 结构的时延性能越来越接近.当使用 2 倍传输加速比时,ICF 算法将获得和 OQ 相同的时延性能.

图 7(c)所示为各种结构时延抖动的仿真.从图中可以看出在 Bernoulli 流量或者突发流量下,5-SLIP(SLIP 最好迭代结果)都有较大的时延抖动.而随着突发长度的增大,CICB-1/RR 结构的抖动逐渐增大,最终和 5-SLIP 的抖动重合.而对于 OOQ/ICF 结构,其抖动始终和 OQ 结构相当,且当突发长度增大时,其时延抖动逐渐和 OQ 重合.这个特性主要是得益于 ICF 算法最大程度地保持了信元先进先出的顺序.对于目前多媒体和实时交互数据比重越来越大的网络环境,保持时延抖动稳定无疑有重要的实际意义.



(a) Bernoulli流量下平均时延 (b) 突发流量下平均时延 (c) 时延变化

图 7 均匀分布流量下时延和抖动的仿真结果

第 2 组仿真主要考查在非均匀流量下各交换机的吞吐率.交换机仿真中常用的非均匀流量主要有非平衡流量(unbalanced traffic)、非对称流量(asymmetric traffic)和对角流量(diagonal traffic)(包括强对角和弱对角).若设端口负载率为 λ,而分别用 $\lambda_{i,j}^{U}$,$\lambda_{i,j}^{A}$ 和 $\lambda_{i,j}^{D}$ 表示从输入端口 i 到输出端口 j 的数据流在非平衡流量、非对称流量和强对角流量(弱对角流量和非平衡流量类似)情况下的负载,则有如下表达式:

$$\lambda_{i,j}^{U}=\begin{cases}\lambda\left[w+\dfrac{1-w}{N}\right], & i=j\\ \lambda\dfrac{1-w}{N}, & \text{其它}\end{cases},$$

$$\lambda_{i,j}^{A}=a_k\lambda,\ j=(i+k)\bmod N,$$

$$a_k=\begin{cases}0, & k=0\\ \dfrac{(r-1)}{(r^N-1)}, & k=1,\ r=f^{-\frac{1}{N-2}},\\ a_1 r^{k-1}, & \text{其它}\end{cases}$$

$$\lambda_{i,j}^{D}=\begin{cases}d\lambda, & i=j\\ (1-d)\lambda, & j=(i+1)\bmod N,\\ 0, & \text{其它}\end{cases}$$

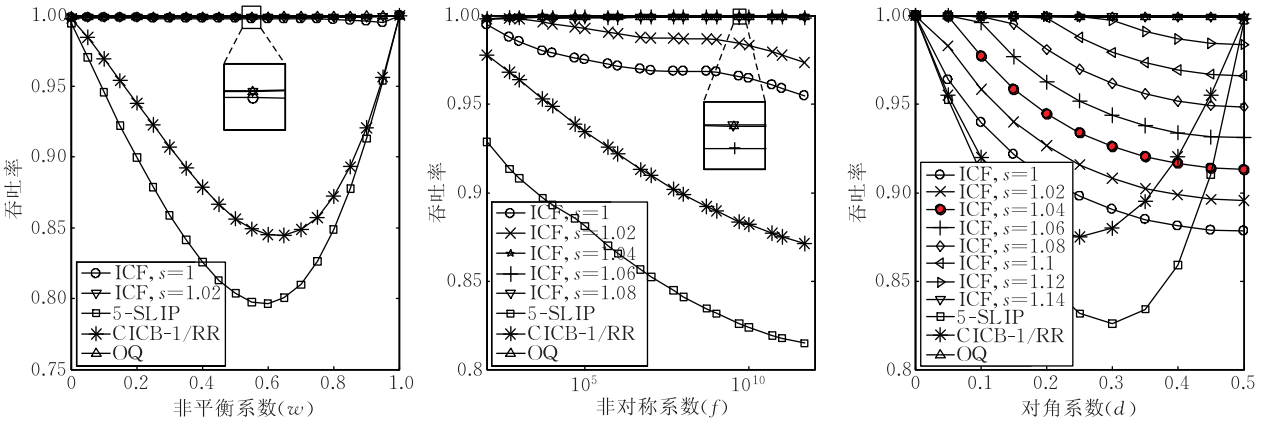
其中参数 $w\in[0,1]$ 称为非平衡系数;参数 $f\in(1,+\infty]$ 称为非对称系数;参数 $d\in[0,0.5]$ 称为对角系数(与取值[0.5,1]时对称).

图 8(a)显示了在非平衡流量下的吞吐率仿真结果,从中可以看出,在最坏情况下,iSLIP 只能达到 80%左右的吞吐率,而 CICB-1/RR 结构可以达到 85%左右的吞吐率,而 ICF 算法在无加速比的情况下就可以达到 98%以上的吞吐率,而仅需要 1.02 倍传输加速比,其吞吐率即可达到 100%.

图 8(b)显示了在非对称流量下的吞吐率仿真结果,该结果表明在最坏情况下,ICF 在无传输加速比时的吞吐率可以达到 95%以上,远远高于 iSLIP 和 CICB-1/RR.当使用 1.08 倍传输加速比时,ICF 算法可达到 100%吞吐率.

图 8(c)为在强对角流量下的吞吐率仿真结果,强对角流量是一种非常扭曲的许可输入流量.当对角系数 d 取值为 0.5 时,表示一个输入端口只会有指向两个相邻的输出端口的信元等概率地到达,这种情况对 ICF 算法是最不利的.此时,ICF 算法的吞吐率只能达到 88%左右,略高于 CICB-1/RR 在最坏情况下的吞吐率(87%).只有当加速比增加到 1.14 时,才能达到 100%吞吐率.

从仿真结果来看,ICF 算法只要使用 1.2 倍传输加速比就可以在时延和吞吐率上均达到比较理想的效果,而不需要使用前文证明的 2 倍传输加速比,1.2 倍传输加速比从实现的观点看也比较合理.



(a) 非平衡流量下的吞吐率　(b) 非对称流量下的吞吐率　(c) 对角流量下的吞吐率

图 8　非均匀流量下吞吐率的仿真结果

6　OOQ/ICF 交换机对 EPFTS 环境的支持

“单层用户数据交换平台体系结构”(Single-layer User-data switching Platform Architecture, SUPA)是四川省西南交通大学网络通信技术重点实验室针对目前 Internet 面临的高速交换、服务质量保障、安全等方面的挑战而提出的下一代高速网络的体系结构框架[13]. 考虑到“未来骨干通信子网必然以光通信技术(别是 DWDM)支撑,光通信技术的误码率已经降低到 10^{-12}”,OSI/RM 设置独立的数据链路层完成检错后重传的必要性已经大大降低. 加上许多物理层传输技术都引入了物理层帧的概念,而层帧间的不兼容又需要额外的“帧间适配开销”,因此将部分功能融合,并对网络层次进行简化,就是我们为骨干通信子网提出的单层用户数据交换平台体系结构. 通过对现有的基于光纤的通信技术的分析,我们认为这些技术都不适应单物理层用户数据传输平台的需要. ITU 的工作以光纤交换(fiber switching)和波长交换(lambda switching)为主,缺少多粒度能够适应不同速率的应用数据流向下复用和交换的需要;在多波长环境中直接利用 SDH 技术实现单波长内的向下复用又存在成本较高和对计算机网络典型分组提供服务时需要切分与重组等不利因素;而学界对 DWDM 交换技术的研究,如:光突发数据交换(Optical Burst Switching, OBS),为了避免在电域内进行交换,采用全光域内数据传输方式,由于光域内缺少缓存和处理能力,难以解决好服务质量保障与信道利用率之间的严重矛盾[13]. 由此可见,需要提出新的交换技术来构建单物理层用户数据传输平台,这正是面向以太网的物理帧时槽交换技术(Ethernet-oriented Physical Timeslot Switching, EPFTS)[14]提出的背景.

我们的经验表明,SUPA 及其 EPFTS 技术为 OOQ/ICF 的实现提供了有利的条件,而 OOQ/ICF 技术又反过来进一步支持了 SUPA 目标的实现. 与现有的网络体系结构相比,SUPA 对路由和交换平台的支持具有如下优势:

(1) 长帧. EPFTS 帧长 2024 字节,以实现对最大“以太网 MAC 帧”的封装而不需要进行分割与重组;对于传统的 64 字节的信元交换时槽,EPF 时槽可为交换节点提供更充足调度时间.

(2) 定长帧交换. 变长分组在交换前需切分为定长信元进行交换,交换后再重组;而 SUPANET 的以 EPF 定长帧为基础交换,交换前后不需拆分和重组,不存在帧内信元错序问题;定长帧的存储管理简单,队列的实现也更加简单.

(3) 面向连接的用户传输与交换服务. SUPA 通过运行服务质量协商协议(QoSNP)为一对用户间的数据流的传输建立基于服务质量的端到端的虚通路(virtual path),因此,同一对用户间的数据流可保证无错序帧;Internet 在网络层提供无连接的转发服务,每个 IP 报文都必须查找路径表后再进行转发,而基于 IP 的查找复杂度较高. 此外,路径更新可能造成同一对用户间的报文流内报文错序或因形成环路而造成 TTL 超限被丢失;SUPANET 与其它面向连接的交换技术,如 ATM(Asynchronous Transfer Mode)和 MPLS(Multi-Protocol Label Switching)一样,仅需在建立连接阶段查找路径表,而连接一旦建立,就只需要通过每个帧所携带的路径信息,使用固定长度的路径标示查找出口信息,从

而简化查表操作,降低线卡功耗和发热量.

(4)超前交换(step forward switching)技术.充分利用建立连接过程中获得的下游节点的输出端口信息作为连接标识符,因而当帧到达输出端口时,不经查表即可以直接发送排队请求.而下一个节点的出口信息的查找过程可以在帧交换时并行查找,并在出口处再进行更新.

综上所述,OOQ/ICF 结构在 SUPA 以 EPFTS 技术为基础的平台上实现,可以充分利用 SUPA 和 EPFTS 的优势,并且可以为 SUPA 提供更好的服务质量保证.相对于目前以 IP 为基础的网络环境,我们认为 OOQ/ICF 在 SUPA 环境下实现具有不可比拟的优势.

7 总结与展望

本文提出了一种面向输出端排队的交换结构(OOQ),该结构将实际的信元存储于输入单元,但是其请求在输出单元排队.输出单元在一个传输时槽开始时从请求队列中取出队头请求,然后从输入单元"拉"信元.此时存在多个输出单元"拉"同一个输入单元的冲突.本文随后提出了一种输入端不冲突(ICF)的算法来解决这一冲突,为了配合这一算法,我们设计了一种调度矩阵来快速选择出队列中的可用位置,并将请求放置到该位置.使用这种调度矩阵,ICF 算法的复杂度和端口数呈线性关系.我们同时也证明了 OOQ/ICF 结构调度的结果等价于贪婪算法 GMWM/OCF 的调度结果,在此基础上进一步证明了 ICF 算法要达到 100%吞吐率的充分条件是使用 2 倍的传输加速比.

在研究过程中我们发现,ICF 算法的思路和 FPCF(Forwarding Planning Conflict-Free)[15] 策略类似,但区别在于:(1)后者没有给出相应的理论分析并证明算法达到稳定性的条件;(2)FPCF 给出的位置查找算法性能低,不实用,本文设计了一种实用的快速查找不冲突位置的调度矩阵;(3)也是最重要的区别,ICF 算法排队是在输出端进行,而 FPCF 算法排队还是在输入端进行,从而输出端口不知道要到达该端口的具体信元数量.实验表明面向输出的排队机制对于拥塞控制和主动队列管理的实现都有很大优势.

文章最后通过仿真发现,无论在 Bernoulli 还是突发的均匀流量下,ICF 算法和 5-SLIP 算法的时延性能都比较接近,特别当突发长度较大时,ICF 和其它结构或算法的时延趋于相同.同时,ICF 算法的抖动性能都优于 iSLIP 算法和 CICB-1/RR 结构,且当突发增大时,ICF 算法的抖动性能越来越接近 OQ.这对于当前多媒体数据和实时数据传输非常有利.在非均匀流量下,仿真结果表明,只要使用 1.14 倍左右的加速比即可达到 100%吞吐率.综合时延和吞吐率的仿真结果来看只要使用 1.2 倍的传输加速比,即可达到比较好的时延、抖动和吞吐率性能.

由于本文中证明的 2 倍加速比不是必要条件,我们在下一步的工作中将试图找到比 2 更低的条件,使得结构达到 100%的吞吐率;其次我们将着重研究基于 OOQ/ICF 结构的优先级策略、拥塞控制、准入控制以及主动队列管理策略.

参 考 文 献

[1] Karol M, Hluchyj M, Morgan S. Input versus output queueing on a space-division packet switch. IEEE Transactions on Communications, 1987, 35(12): 1347-1356

[2] McKeown N, Mekkittikul A, Anantharam V, Walrand J. Achieving 100% throughput in an input-queued switch. IEEE Transactions on Communications, 1999, 47(8): 1260-1267

[3] Anderson T E, Owicki S S, Saxe J B, Thacker C P. High-speed switch scheduling for local-area networks. ACM Transactions on Computer Systems, 1993, 11(4): 319-352

[4] McKeown N. The iSLIP scheduling algorithm for input-queued switches. IEEE/ACM Transactions on Networking, 1999, 7(2): 188-201

[5] Li Yihan, Panwar S, Chao H. On the performance of a dual round-robin switch//Proceedings of the IEEE, INFOCOM 2001. 20th Annual Joint Conference of the IEEE Computer and Communications Societies. Location: Anchorage, Alaska, US, 2001, 3: 1688-1697

[6] Sun Shu-Tao, He Si-Min, Zheng Yan-Feng, Gao Wen. Matching algorithm with queue length weighted service for input queued switches. Chinese Journal of Computers, 2006, 29(6): 875-883(in Chinese)
(孙书韬, 贺思敏, 郑燕峰, 高文. 队列长度加权服务的输入排队匹配算法. 计算机学报, 2006, 29(6): 875-883)

[7] Rojas-Cessa R, Oki E, Chao H. On the combined input-crosspoint buffered switch with round-robin arbitration. IEEE Transactions on Communications, 2005, 53(11): 1945-1951

[8] Chiussi F, Kneuer J, Kumar V. Low-cost scalable switching solutions for broadband networking: The ATLANTA architecture and chipset. IEEE Communications Magazine, 1997, 35(12): 44-53

[9] Shen Yan-Ming, Panwar S, Chao H. Design and performance analysis of a practical load-balanced switch. IEEE Transactions on Communications, 2009, 57(8): 2420-2429

[10] Bianco A, Giaccone P, Leonardi E, Mellia M, Neri F. Theoretical performance of input-queued switches using Lyapunov methodology//Itamar Elhanany, Mounir Hamdi eds. High-Performance Packet Switching Architectures. London: Springer, 2007: 39-63

[11] Leonardi, Mellia M, Neri F, Marsan M. On the stability of input-queued switches with speed-up. IEEE/ACM Transactions on Networking, 2001, 9(1): 104-118

[12] Xia Yu, Zeng Hua-Xin, Shen Zhi-Jun. Design and implementation of switch module for NS-3//Proceedings of the 4th International ICST Conference on Performance Evaluation Methodologies and Tools (ICST ValueTools 2009). Pisa, Italy, 2009

[13] Zeng Hua-Xin, Dou Jun, Xu Deng-Yuan. Single physical layer U-Plane Architecture (SUPA) for next generation Internet//Comprehensive Report on VoIP and Enhanced IP Communications Services. IEC Publications, 2004: 197-227

[14] Zeng Hua-Xin, Dou Jun, Xu Deng-Yuan. Replace MPLS with EPFTS to build a SUPANET//Proceedings of the IEEE Workshop on High Performance Switches, Routers (HPSR 2005). Hong Kong, China, 2005: 39-43

[15] Yau Victor, Pawlikowski K. An algorithm that uses forward planning to expedite conflict-free traffic assignment in time-multiplex switching systems. IEEE Transactions on Communications, 1999, 47(11): 1757-1765

XIA Yu, born in 1983, Ph. D. candidate. His research interests include computer network architectures, high-performance packet switches and switch scheduling algorithms.

GAO Zhi-Jiang, born in 1985, Ph. D.. His main research interests include design and implementation of high-performance packet switches.

ZENG Hua-Xin, born in 1945, Ph.D., professor, Ph. D. supervisor. His research interests include computer network architectures and communications technology, high-performance packet switch architectures and router test system.

Background

The main background of this research is the Single User-data transfer Platform Architecture Network (SUPANET) platform proposed by the Sichuan Network & Communications Technology Key Lab. The research of this platform is supported in part by National Natural Science Foundation project "Next generation Internet architecture and its key technologies" under grant 60773102 and "Long-term strategy of engineering and technology development in China" united foundation project "Future network architecture and network technology strategies" under grant U090122, and in part by Sichuan University Foundation (SCUF) project "Next generation Internet Architecture".

As early as in 2003, the research group has deeply analyzed the problems in current Internet architecture, and found that the 30 years old Internet architecture, which provides the best effort service, has been the root of the inability of current Internet to provide QoS guarantee, thus an "clean-slate" architecture must be used in order to provide QoS guarantee in future Internet. This idea is the same as what GENI (Global Environment for Network Innovations) announced in 2006. SUPANET is an answer to this "clean-slate" architecture. By using physical frame, the authors put the QoS guarantee directly on physical layer, and this leads to the concept of Ethernet-oriented Physical Frame Timeslot Switching (EPFTS). Mean while, SUPANET is a connection-oriented network, i. e. a connection must be established by using QoSNP (QoS Negotiation Protocol) before the data can be transferred on the network. By doing so, the QoS of each connection can be easily reserved and then conserved. We have a lot research results on SUPANET platform, and many of our papers have been indexed by EI and SCI.

The authors found that EPFTS has been the most important component of SUPANET when we were building SUPANET, however, EPFTS urgently need a switching platform which can take full advantage of its characteristics, and this is the work presented in this paper.