软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(10):2644−2655 [doi: 10.13328/j.cnki.jos.004739]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 一种面向 MSM 型 Clos 交换结构的启发式并发调度算法*

刘晓锋[1], 赵有健[2], 陈 果[2]

[1](电子科技大学 计算机科学与工程学院,四川 成都 611731)

[2](清华大学 计算机科学与技术系,北京 100084)

通讯作者: 刘晓锋, E-mail: xhxfliu@163.com

**摘 要**: 调度算法一直是交换系统中不可或缺的研究内容.为满足新型高速路由及交换系统的研究需求,提出一种主动授权并发轮询调度算法——CRRD-AG 算法.多级交换结构 Clos 交换网络以其良好的可扩展性作为高速交换结构倍受关注,但与之相适应的调度算法却并不多.目前主流算法,如并发分派算法(CD)和基于轮询的并发分派算法(CRRD),不是吞吐率较低就是所处理的业务流单一.CRRD-AG 算法以 CRRD 为基础,将经典的"请求-授权-接受"的匹配计算模式改进为"主动授权-接受"的匹配模式,不仅能够降低 CRRD 算法在第 1 阶段的仲裁信息量,而且充分利用了中间级链路带宽,从而降低了整个系统的平均延迟,提高了吞吐率.进行充分的实验后,其结果表明,无论是在均匀业务,还是在突发业务环境中,CRRD-AG 算法都能保证 100%的吞吐率,更为重要的是,在不降低吞吐率的情况下能够显著改善分组的平均延迟.

**关键词**: Clos 网络;调度算法;交换结构;轮询迭代

**中图法分类号**: TP393

中文引用格式: 刘晓锋,赵有健,陈果.一种面向 MSM 型 Clos 交换结构的启发式并发调度算法.软件学报,2015,26(10):2644−2655. http://www.jos.org.cn/1000-9825/4739.htm

英文引用格式: Liu XF, Zhao YJ, Chen G. Heuristic concurrent dispatching algorithm for MSM Clos-network switches. Ruan Jian Xue Bao/Journal of Software, 2015,26(10):2644−2655 (in Chinese). http://www.jos.org.cn/1000-9825/4739.htm

## Heuristic Concurrent Dispatching Algorithm for MSM Clos-Network Switches

LIU Xiao-Feng[1], ZHAO You-Jian[2], CHEN Guo[2]

[1](School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

[2](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

**Abstract**: A dispatching algorithm is always indispensable research topic for a switching system. In studying the future-generation high-speed switching and routing system, a concurrent round-robin dispatching algorithm with active grants, called CRRD-AG, is proposed in this paper. As a multistage switching architecture, Clos-network has been receiving increasing attention due to its scalability and modularity, and yet it is not much that the corresponding dispatching algorithm can be applied to the multistage switching fabric. So far, some well-known dispatching algorithms, such as concurrent dispatching (CD) and concurrent round-robin dispatching (CRRD), either have low throughput or can not handle various traffic. CRRD-AG algorithm based on CRRD improves the typical request-grant-accept (R-G-A) matching model used by CRRD to the active grant-accept (AG-A) matching model. Consequently, not only the iterative messages of the request phase can be reduced significantly, but also the bandwidth of the central stage can be utilized sufficiently. Therefore, the average delay is decreased and the throughput is increased. Also, simulations show that CRRD-AG achieves 100% throughput under uniform traffic and bursty traffic. More importantly, the average delay performance of the whole switching system is improved significantly without reducing the throughput of the switching system.

* 基金项目: 国家高技术研究发展计划(863)(2013AA013302); 国家重点基础研究发展计划(973)(2013CB329105); 国家自然科学基金(61233007)

收稿时间: 2014-02-08; 修改时间: 2014-05-09; 定稿时间: 2014-10-04

**Key words**: Clos network; dispatching algorithm; switching architecture; round-robin iteration

Internet 已是人们生活、学习和工作必不可少的组成部分,各种网络应用对网络带宽的需求与日俱增.同时,传输系统的传输速度随着密集波分复用技术(DWDM)的成熟大幅提升,如英国南安普敦大学最近研究出一种空心光纤,其传输速度高达 73.7Tbps[1].另一方面,由于受存储器的容量及访问速率的约束,分组交换机(packet switch)/路由器(router)的发展远远落后于传输技术.因此,阻碍网络性能的主要瓶颈在于网络交换节点,即,分组交换机/路由器.

调度算法是交换系统中非常重要的研究对象,但交换结构是研究调度算法的物理基础,不同的交换结构会有不同的调度算法.目前,交换结构主要有以 Crossbar 为代表的单级结构(single-stage)和以 Clos 网络[2]为代表的多级结构(multi-stage).虽然有不少以单级结构为交换结构的高性能产品,但是随着端口数量和端口速率的提高,其缺陷也越发明显,如它的可扩展性较差、实现成本较高.这些决定了它不适宜用来构建高速大容量的交换系统.多级结构因端口阻塞点较多(每级都存在阻塞点),调度相对复杂,但可扩展性好,很容易构建大容量的交换系统.如:实现一个 256×256 的交换系统,可用 48 个 16×16 的 Crossbar 构建一个 Clos 网络来实现,其交叉点数相对单级结构可减少 80%以上,其实现难度(硬件成本)大为降低.因此,有不少知名厂商推出基于多级结构的高端产品,如 Cisco CRS 系列和 Juniper T1600/TX-Matrix Plus.

本质上,调度算法就是为到达交换结构的分组指派合理的路由路径,使其顺利到达各自的目的端口,这等价于寻找一个二分图的匹配问题.基于最大匹配或最大权重匹配的调度算法虽然能够达到 100%的吞吐率,但其时间复杂度较高,分别为 $O(N^{2.5})$和 $O(N^3\log N)$[3],在实际应用中很少被采用.实际上常采用一些迭代调度算法来逼近(或收敛)一个极大匹配,如并行迭代匹配(parallel iterative matching,简称 PIM)[4]、*i*-SLIP[5]、DRRM[6]等.这类算法为了提高吞吐率,通常需要多次迭代(平均需要 $O(\log_2 N)$次)才可能得到一个输入-输出的极大匹配;更重要的是,这些算法是为单级结构设计的,不能简单地应用于多级结构.目前,针对多级结构的调度算法并不多,主要有 CD[7], CRRD[8],Distro[9]和 MAC[10].这些算法在本质上与上述单级结构没有多大区别,都是利用多次迭代来逼近一个端口间的极大匹配.由于多级结构存在较多的内部阻塞点,要确保每级都没有阻塞,就需要逐级匹配.这种匹配模式给交换系统带来两方面的影响.

- 延迟增加.为了排除每级的阻塞,在迭代过程中就需要维护与存储大量的仲裁信息,这需要够快够大的缓存支持,而成本及实现技术等原因导致实际中很难实现大容量高速缓存;
- 吞吐率降低.仲裁信息的交换不仅增加了延迟,而且成功匹配的端口数会逐级减少,因为后一级的匹配操作可能阻塞掉前一级的部分成功匹配,导致每个时隙获得传输资格的分组数有所减少.

针对面向多级结构调度算法的上述现实问题,本文在并发轮询调度算法(concurrent round-robin dispatching,简称 CRRD)[8]的基础上提出一种新的多级结构调度算法——主动授权并发轮询调度算法 CRRD-AG(CRRD with active grants)调度算法.本文的研究内容主要是尽可能地减少在第 1 级匹配过程中产生的仲裁信息量,降低存储瓶颈的影响;其次,充分利用第 2 级的链路带宽,减少后一级对前一级阻塞掉的匹配数,从而降低整个交换系统的平均延迟,提高吞吐率.大量的实验结果表明:CRRD-AG 算法不仅能够保持 100%的吞吐率,而且能够很明显地改善交换系统的平均延迟性能.

本文第 1 节介绍 MSM(memory-space-memory)型 Clos 交换网络模型,这是调度算法的物理基础.第 2 节简单介绍多级结构调度算法的相关背景,主要讨论 CRRD 算法.第 3 节详细讨论 CRRD-AG 调度算法,并对 CRRD-AG 算法进行性能评估.最后给出相应结论.

## 1 MSM 型 Clos 交换网络模型

Clos 交换网络[2]是一种很著名且被广泛研究的多级结构.在 Clos 发表论文后的 60 余年里,交换技术不断发展,但 Clos 提出的体系结构在大容量交换结构的研发中继续发挥着重要作用.因为在设计高速大容量交换结构时,Clos 网络展现出无可比拟的优势,如,可大幅减少构建无阻塞交换结构所需要的交叉点;其次,Clos 结构是一

个多路径结构,故它对故障的抵御能力很强;第三,Clos 结构具有调度的可扩展性,支持构建大容量交换结构.因此,单级结构无法撼动 Clos 网络在高速大容量交换结构的统治地位,研究 Clos 网络的调度算法也具有重要的实际意义及理论价值.

一个 Clos 交换网络是以多个小规模的 Crossbar 为交换模块,通过链路互连成的一个 3 级交换网络,分别称为输入级(input stage,简称 IS)、中间级(central stage,简称 CS)和输出级(output stage,简称 OS),对应级上的交换模块称为输入模块(input module,简称 IM)、中间模块(central module,简称 CM)和输出模块(output module,简称 OM).假设 IS 级含有 $k$ 个 IMs,每个 IM 的维数为 $n\times m$;CS 级含有 $m$ 个 CMs,每个 CM 的维数为 $k\times k$;OS 级含有 $k$ 个 OMs,每个 OM 的维数为 $m\times n$;级与级之间用链路互连起来.这种 Clos 网络通常记为 $C(n,m,k)$,其交换容量 $N=n\times k$.

为了除去队头(head-of-line,简称 HoL)阻塞,在每个 IM 里为每个输出端口配置一个 VOQ,每个 VOQ 在一个时隙内最多可接收 $n$ 个信元(每个输入端口到达 1 个),但只能送走 1 个信元(队首信元)到下一级交换模块(CM 模块).在 OM 内,为自己的每个输出端口配置一个输出队列,该输出队列每个时隙可接收 $m$ 个信元(来自 $m$ 个 CM 模块),而且假定这个队列无穷大,不会出现信元丢失现象,这样可以消去 OS 级的出口阻塞,那么整个交换过程就只需关注 IS 和 CS 两级的出口阻塞情况.这种结构称为 MSM 结构的 Clos 交换网络,如图 1 所示.

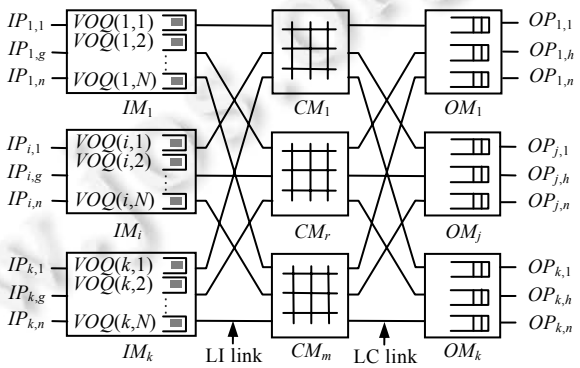



Fig.1 System model of an MSM Clos-network $C(n,m,k)$

图 1 MSM Clos 交换网络 $C(n,m,k)$系统模型

本文的研究工作也是基于 MSM 结构的 Clos 交换网络,下面给出 MSM 结构 Clos 网络的基本模型及本文所使用的相关术语(见表 1).

**Table 1** Notations used in this paper

**表 1** 本文所用的术语记号

| Notations | Meaning |
|---|---|
| IM | 输入模块 |
| CM | 中间模块 |
| OM | 输出模块 |
| $IM(i)$ | 第 $i$ 个输入模块,$1\leqslant i\leqslant k$ |
| $CM(r)$ | 第 $r$ 个中间模块,$1\leqslant r\leqslant m$ |
| $OM(j)$ | 第 $j$ 个输出模块,$1\leqslant j\leqslant k$ |
| $IP(i,g)$ | $IM(i)$的第 $g(1\leqslant g\leqslant n)$个输入端口 |
| $OP(j,h)$ | $OM(j)$的第 $h(1\leqslant h\leqslant n)$个输出端口 |
| $LI(i,r)$ | $IM(i)$输出链路,连接到 $CM(r)$ |
| $LC(r,j)$ | $CM(r)$的输出链路,连接到 $OM(j)$ |
| $VOQ(i,v)$ | $IM(i)$中第 $v(1\leqslant v\leqslant N)$个 VOQ,对应于 $OP(j,h)(1\leqslant j\leqslant k,1\leqslant h\leqslant n)$ |

## 2 多级结构调度算法相关研究

在多级交换结构中,每对输入-输出都有多条路由路径可供选择,但为输入-输出请求选择合适的路由路径实属不易,因为在实现无阻塞时还需考虑其他性能指标,如低加速比($m/n$)、低延迟和高吞吐量等.因此,多级结构的调度相对单级结构而言会更复杂,而且相关研究并不多,目前主要有并发分派(concurrent dispatching,简称 CD)算法[7]和基于轮询的并发分派算法 CRRD[8],Distro[9],MAC[10].

实际上,CD 算法是一种随机分派算法.各交换模块在路径寻找过程中没有信息共享,即,每个仲裁器可能收到多个请求或授权信号.在 CD 算法中,仲裁器为每个请求随机选择目标并送出相应请求信号,仲裁器收到请求信号后再随机选择一个给予授权.这种随机思想始于平衡中间级负载,但在高速环境中不容易实现这种随机性(因为需要随机函数来实现)[5].更重要的问题是:它的吞吐量不高,要取得 100%的吞吐率,加速比应在 1.6 左右.如果没有加速比或加速比为 1($m=n$),则其吞吐率仅为 63%[8].Distro 算法是针对无缓存 Clos 网络而设计的,而且是一款基于静态轮询的调度算法,因此对突发业务很难取得高吞吐率.MAC(matching algorithms for Clos)也是针对无缓存 Clos 网络而设计的,而且 MAC 算法将整个调度过程分成两个子过程:端口的匹配(port-to-port matching)及路径的分派(route assignment).MAC 算法首先根据分组的到达情况将 Clos 网络模拟成一个类似单级的调度器,并在其中完成输入-输出端口的匹配;然后,借助其他算法为成功匹配的输入/输出对指定路由路径,即,MAC 只能解决调度的一部分.CRRD 算法是一种完全分布式、动态轮询的调度算法,虽然是一款很成功的调度算法,但仍然存在很大的改进空间.下面首先对 CRRD 算法作一简单介绍.

### 2.1 CRRD算法[8]简介

CRRD 算法本质上可谓是 $i$-SLIP[5]算法和 CD[7]相结合的产物,CRRD 的基本思想是:基于“请求-授权-接受(request-grant-accept,简称 RGA)”的轮询迭代匹配模式,在输入-输出之间寻找一组无阻塞的路由路径.由于算法利用仲裁指针非同步化的更新方法,它可在均匀业务环境下达到 100%的吞吐率,在突发业务环境下也可取得较好吞吐率.为了能够更好地利用各仲裁指针更新的去同步化效应,CRRD 将 $IM(i)(1\leqslant i\leqslant k)$内的 $VOQ(i,v)(1\leqslant v\leqslant N)$与输出端口 $OP(j,h)(1\leqslant j\leqslant k,1\leqslant h\leqslant n)$按如下关系映射起来:$v=(h-1)\times k+j,1\leqslant h\leqslant n,1\leqslant j\leqslant k$.如:

$$VOQ(i,1)\leftrightarrow OP(1,1),VOQ(i,2)\leftrightarrow OP(2,1),\cdots$$

在 CRRD 算法中,$IM(i)(1\leqslant i\leqslant k)$为每个 $VOQ(i,v)(1\leqslant v\leqslant N)$及输出链路 $LI(i,r)$(此链路连接到 $CM(r)$)设立了一个仲裁器,仲裁指针以固定轮询的方式决定请求、授权及接受 3 个过程;$CM(r)(1\leqslant r\leqslant m)$为每个输出链路 $LC(r,j)$(此链路连接到 $OM(j)$)设立一个仲裁器,仲裁指针同样以固定的轮询顺序决定自己的授权.CRRD 算法用两个阶段分别解决 IS 级和 CS 级出口阻塞,具体描述如下:

- 第 1 级:在 IM 内部的匹配(消去 IS 的出口阻塞)
  - 第 1 次迭代:

    Step 1. Request:每个非空的 VOQ 向每个输出链路 LI 仲裁器发出输出请求;

    Step 2. Grant:每个输出链路 LI 仲裁器根据自己授权指针以固定的轮询顺序选择一个请求,并给予授权;

    Step 3. Accept:每个非空 VOQ 仲裁器根据自己的授受指针以固定的轮询顺序选择一个授权并授受;
  - 第 $i^{th}(i>1)$迭代:

    Step 1. Request:在上次迭代中匹配未成功的非空 VOQ 向未匹配的输出链路 LI 再次发出请求;

    Step 2 和 Step 3 与第 1 次迭代中的 Step 2 和 Step 3 一样;
- 第 2 级:IM 和 CM 之间的匹配(消去 CS 的出口阻塞)
  - Step 1. Request:在第 1 级成功匹配的输出链路 $LI(i,r)$向 $CM(r)$的输出链路 $LC(r,j)$发出输出请求;
  - Step 2. Grant:$CM(r)$的每个输出链路 $LC(r,j)$根据自己授权指针以固定轮询顺序选择一个请求,并给予授权.

如果 $LI(i,r)$收到 $LC(r,j)$的授权,则 $IM(i)$在下一个时隙传输对应 VOQ 中的信元;否则,$IM(i)$在下一个时隙不能传输信元,而必须在下一个时隙再次匹配.只有在两个阶段都获取了成功匹配,相应的仲裁指针 $g$ 才能以固定的轮询顺序更新自己的指向,即 $g=(g+1)$ modulo $N$.

当分组以 Bernoulli 过程到达,且均匀分布在各输出端口时,CRRD 算法可达 100%吞吐率,且只需较少的迭代次数就可收敛于一个极大匹配.这种多次迭代可以在每个时隙以分布式的方式产生一个近似的极大匹配,每次可以传输更多的分组,从而提高吞吐率,但是每次传输都需要通过多次迭代才能取得一个近似的最大匹配.从理论上看,这样会增加分组在 VOQ 中的等待时间,从而降低吞吐率.另外,我们从上述匹配过程可以看出,CRRD 算法存在如下两个特点:

(1) 它在第 1 级的每次迭代都需要完成请求-授权-接受这 3 步;

(2) 它在第 2 级被阻塞掉的请求可能要等到下一个时隙再匹配.

第(1)点会产生大量仲裁信息,这些信息的传输及存储随着端口数量及速度的提高会变得很困难.其实,在第 2 次、第 3 次等后续迭代中,未获得匹配的输入仲裁器没有必要再次向输出仲裁发送输出请求,因为在第 1 次迭代后,所有的输出仲裁器已完全掌握当前时隙的请求状态.第(2)点说明,在第 1 级的部分成功匹配在第 2 级可能被阻塞掉,这意味着中间级的输出链路带宽得不到充分利用,导致延迟增加,吞吐率降低.这些都说明 CRRD 在性能上存在不足,不能很好地适应高速环境.为了很好地解决这些问题,本文提出一种主动授权并发轮询调度算法——CRRD-AG 算法.

## 3 主动授权并发轮询调度算法(CRRD-AG)

### 3.1 多级结构的调度目标

在多级结构中,路由路径上存在较多阻塞点,当多个分组竞争同一资源(如输出端口、交换结构等)时,必须采用某种合理的调度策略来决定它们的去留.调度策略的合理性主要表现在:(1) 要尽可能地提高吞吐率,降低延迟,充分利用各级链路带宽;(2) 尽量减少实现分布式调度所需要的附加信息量,这更有利于实现分布式调度及增强调度算法的可扩展性[1].因此,一种针对多级结构的调度算法必须实现每一级无阻塞,而且每个时隙尽可能地交换更多的分组,充分利用各级链路带宽,提高系统吞吐率,降低系统的平均排队延迟.

### 3.2 CRRD-AG

针对 CRRD 存在的上述不足及多级结构的调度目标,我们提出主动授权并发轮询调度算法——CRRD-AG 算法.CRRD-AG 算法在第 1 级将 RGA 匹配模式改进为主动授权-接受(active-grant accept,简称 AG-A)匹配迭代模式,这样,CRRD-AG 算法在每次迭代时只需要 2 次数据交换,极大地降低了仲裁信息的交换量,结果算法实现更简单且易扩展;其次,在第 2 级,由单次握手改为多次询问,充分利用 CS 级链路带宽,提高了 CS 级匹配率,降低了交换的平均延迟,进而提高了吞吐率.

在第 1 级第 1 次迭代之后,每个输出仲裁器已经完全掌握所有 VOQ 在当前时隙的请求状态,因此在后续的迭代中,那些未匹配成功的 VOQ 无需再向未匹配的输出仲裁器发出请求,而由未匹配的输出仲裁器向下一个请求(相对于当前授权的请求)送出“主动授权(AG)”信号,收到此信号的 VOQ 仲裁器根据自己是否获得匹配来决定如何处理此信号.如果 VOQ 已经获得匹配,则直接丢弃该信号;否则,以固定轮询的方式接受其中一个主动授权.因此,CRRD-AG 算法采取主动授权模式,而 CRRD 却是被动授权,所以我们称此调度算法为 CRRD-AG.

在第 1 级获得匹配的 $LI(i,r)$链路向自己的目标输出链路 $LC(r,j)$发出传输请求,$LC(r,j)$链路按固定轮询的方式进行授权.在这个阶段,可能有多个 $LI(i,r)$链路申请同一个 $LC(r,j)$链路,但 $LC(r,j)$每次只能授权一个请求,因此可能存在部分请求得不到 $LC(r,j)$链路的授权.当这种情况发生时,CRRD 算法让被阻塞的请求在下一个时隙再进行匹配.CRRD-OG[12]算法让未收到请求的 $LC(r,j)$链路选择一个未被授权的请求并给予一个公开授权(open grant,简称 OG),以表明自己当前处于空闲状态.未被授权的 $LI(i,r)$链路收到 OG 信息后,再通知相应的输入模块 $IM(i)$,并检查是否存在需要送到 OG 所包含的目的端口的请求:如果存在,则无需匹配,在下一个时隙直接传送.

CRRD-OG 算法通过 OG 信息可以充分利用每条链路的带宽,从而提高性能.但在完全分布式的端口匹配模式下,各仲裁器没有任何信息共享或信息交换,那么未收到请求的 $LC(r,j)$链路无从得知当前时隙的请求状况,它不知道有哪些请求,更不知道哪些请求未被授权,因此,未收到请求的 $LC(r,j)$链路根本没有办法选择那些未被授权(或被阻塞)的请求.基于此,在 CRRD-AG 算法中,未收到请求的 $LC(r,j)$链路以固定轮询的方式向输入链路 $LI(i,r)$发送一个 AG 信号,以表明自己当前不仅处于空闲状态,而且还可以通过自己到达输入模块 $OM(j)$及相应的输出端口.收到 AG 信号的 $LI(i,r)$仲裁器根据自己是否被授权来决定如何处理此信号:如果 $LI(i,r)$的请求当前已被授权,则丢弃该信号;否则,以固定轮询的方式接受其中一个主动授权.通过这样的改进后,CS 级的输出带宽得到了充分利用,大幅提升了 CS 级的匹配率(如后文图 5 所示).CRRD-AG 算法具体描述如下:

- 第 1 级:在 IM 内部的匹配(消除 IS 的出口阻塞)
  - 第 1 次迭代:

    Step 1. Request:当前时隙每个非空的 VOQ 向所有输出链路仲裁器发送输出请求;

    Step 2. Grant:每个输出仲裁器按自己授权指针以固定的轮询顺序授权一个请求;

    Step 3. Accept:每个非空 VOQ 仲裁器按自己接受指针以固定的轮询顺序接受一个授权;
  - 第 $i^{th}(i>1)$迭代:

    Step 1. Active Grant:每个未被接受的输出仲裁器以固定轮询的顺序向自己收到的下一个请求(即自己刚授权的下一个请求)送出主动授权:如果此次主动授权被接受,则终止主动授权;否则,以固定轮询的方式进入下一轮迭代,向下一个请求再次送出主动授权;

    Step 2. Accept:收到主动授权的 VOQ 仲裁器根据自己当前时隙的匹配情况决定是否接受主动授权:如果该 VOQ 在当前时隙没有获得匹配,则以固定轮询的方式接受其中一个主动授权;否则,不理会这个主动授权;
- 第 2 级:IM 和 CM 之间的匹配(除去 CS 的出口阻塞)
  - 第 1 次迭代:

    Step 1. Request:在第 1 个阶段成功匹配的输出链路 $LI(i,r)$向自己的目标输出链路 $LC(r,j)$发出输出请求;

    Step 2. Grant:每个输出链路 $LC(r,j)$根据自己授权指针以固定轮询顺序授权其中一个请求;
  - 第 $i^{th}(i>1)$次迭代:

    Step 1. Active Grant:每个未收到请求的 $LC(r,j)$仲裁器向自己授权指针所指 $LI(i,r)$送出一个主动授权:如果 LC 仲裁器的主动授权被接受,则终止主动授权的发送;否则,以固定轮询的方式进入下一轮迭代,向下一个目标再次送出主动授权;

    Step 2. Accept:一个 LI 链路仲裁器根据自己的请求是否被授权来决定如何处理收到的主动授权:如果 LI 发出的请求已获得了授权,则丢弃该主动授权;否则,根据主动授权所包含的信息检查相应的 IM 是否存在分组交换需求,如果有,则以固定轮询的方式接受其中一个主动授权,否则,放弃主动授权.

如果 IM 与 CM 之间的匹配获得成功,则在下一个时隙传输相应的一个信元;如果收到 AG 信息且已成功匹配,也会在下一个时隙传输一个信元.只有那些有资格传输信元的仲裁器的指针才能更新自己的指向.

### 3.3 交换仲裁信息量的比较

为描述方便起见,后文中:“一次匹配”是指所有 VOQ 或所有的输出仲裁器均获匹配的整个过程;“一次迭代”是指一次 RGA 过程.显然,一次匹配包含多个一次迭代.另外,我们用信息的交换次数来定义信息量.假设 IM 中的 VOQs 在每次匹配的开始都处于饱和状态,即,每个 VOQ 都有传输请求.$R_i$ 表示第 $i$ 次迭代有传输请求的 VOQ 数,$x_i$ 表示第 $i$ 次迭代匹配成功的 VOQ 数,则第 $i$ 次迭代匹配失败的 VOQ 数为 $R_i-x_i$,这也是第 $i$+1 次迭代应发出的请求数.第 $i$ 次迭代中,$R$,$G$ 和 $A$ 三次握手所需交换的仲裁信息量为

$$\begin{cases} R: R_i \times \left( m - \sum_{j=1}^{i} x_{j-1} \right) \\ G: m - \sum_{j=1}^{i} x_{j-1} \\ A: x_i \end{cases},$$

其中, $\begin{cases} R_i = R_{i-1} - x_{i-1} \\ R_0 = N \\ x_0 = 0 \end{cases}, i \geqslant 1.$

$i$ 的取值最坏情况为 $m$,但平均情况为 $\log_2 N$,因为在平均情况下,迭代 $\log_2 N$ 次后可收敛到一个极大匹配.CRRD 算法在完成一次匹配所需要交换的仲裁信息量为 $\sum_{i=1}^{\log_2 N} \left( R_i \times \left( m - \sum_{j=1}^{i} x_{j-1} \right) + \left( m - \sum_{j=1}^{i} x_{j-1} \right) + x_i \right)$(上述 3 部分之和),而 CRRD-AG 算法在完成一次匹配所需交换的仲裁信息量为 $(N \times m + m + x_1) + \sum_{i=2}^{\log_2 N} \left( \left( m - \sum_{j=1}^{i} x_{j-1} \right) + x_i \right)$.虽然总的迭代次数没变,但每次迭代交换的仲裁信息量却减少了,共减少了 $\sum_{i=2}^{\log_2 N} \left( R_i \times \left( m - \sum_{j=1}^{i} x_{j-1} \right) \right)$ 的额外信息量.

### 3.4 主动授权的迭代次数分析

CRRD-AG 算法在第 1 阶段的主要目标是降低额外的交换信息量,减轻对这些信息的存储与管理负担,迭代次数并未减少,与 CRRD 算法是一致的.因此,我们在这里主要分析 CRRD-AG 算法第 2 阶段主动授权需要迭代的次数以及对交换性能的影响.

CRRD-AG 算法在完成 IM 与 CM 之间的匹配时若未收到来自 IM 请求的 CM 模块,会主动轮询地向 IM 模块发出 AG 信号.虽然 CRRD-AG 以这种匹配模式能够明显降低平均延迟时间、提高吞吐率,但其性能仍会受中间级迭代次数的影响.在最坏情况下,某个 CM 模块的每个输出仲裁器可能会轮询一周,即,此 CM 模块未收到任何传输请求.这种情况在负载较轻时容易出现,随着负载的增加,出现这种情况的可能性就很小了,这也是导致在负载较轻时 CS 级的匹配率可能大于 1 的原因.因为在负载较轻时,在第 1 级被阻塞掉的部分请求可能在第 2 级通过迭代成功传输,这部分请求在第 2 级统计请求时未被计算,但在统计被授权数据时却进行了计算,因此在这种情况下,CS 级的匹配率可能稍大于 1,如后文图 4 所示.

那么在平均情况下,CS 级需要迭代多少次呢?由于 Clos 交换网络的所有 CM 交换模块及其输出端口具有相同的统计特性,下面我们以任意交换模块 $CM(r)(1 \leqslant r \leqslant m)$的任一输出端口 $O_q(1 \leqslant q \leqslant k)$为例进行分析.假设整个交换网络已处于饱和状态,而且在某个时隙有 $x(0 \leqslant x \leqslant k)$个请求到达 $CM(r)$,其中有 $s(s \leqslant x)$个请求没有得到授权,则有 $x–s$ 个请求得到了授权,未收到请求的输出端口数为 $k–x+s$,而 $O_q$ 就是其中之一.$O_q$ 送出的主动授权能否被接受,由以两个条件决定:

(1) 被 $O_q$ 授权的对象之前是否未被授权;

(2) 被授权对象是否有分组需要传送到 $O_q$ 所指的目的端口.

只有这两个条件同时成立,$O_q$ 的主动授权才能被接受,而且这两个条件是相互独立的.第(1)个条件为真的概率为$(k–x+s)/k$;第(2)个条件为真的概率为$\rho/N$,其中,$\rho$表示分组到达输入端的强度.而且到达的分组在所有输出端口间是均匀分布的,$N=n\times k$ 表示交换容量,即,端口总数.当 $O_q$ 的授权目标同时收到 $h(1 \leqslant h \leqslant k–x+s)$个主动授权时,就随机接受一个.因此,$O_q$ 的主动授权被接受的概率为

$$\frac{\rho}{N}\frac{k-x+s}{k}\frac{1}{h} = \frac{\rho(k-x+s)}{nhk^2}.$$

假设随机变量 $X$ 表示 $O_q$ 的主动授权被接受时已授权次数(即迭代次数),当 $h$=1 时,$O_q$ 送出的主动授权信号要么被接受,要么不被接受,因此可以认为 $X$ 是服从几何分布的(稍后分析系统稳定时 $h$ 的取值),即:

$$\Pr[X=x]=\left(1-\frac{\rho(k-x+s)}{nhk^2}\right)^{x-1}\frac{\rho(k-x+s)}{nhk^2},x=1,2,...,k \tag{1}$$

由于 CRRD-AG 算法在每个阶段从第 2 次迭代开始的后续迭代中,每个仲裁器每次只送出一个信号,这与 DRRM[6]类似.CRRD-AG 与 DRRM 的区别在于:CRRD-AG 算法在匹配未成功时更新指针,而 DRRM 却是在匹配获得成功时更新指针.为了分析公式(1)中 $h$ 的取值情况,我们定义一个向量 $Y_i=(y_{1,i},\ldots,y_{r,i},\ldots,y_{k,i})$来表示 CM 的 $k$ 个输入在第 $i$ 时隙收到其输出的主动授权的状态,其中,$y_{r,i}$ 表示 CM 的第 $r$ 个输入在第 $i$ 时隙收到主动授权的数目,而且 $0\leqslant y_{r,i}\leqslant k,\sum_{r=1}^{k}y_{r,i}=k$. 根据 CRRD-AG 的迭代规则,我们有:

$$y_{(r+1)\,\mathrm{MOD}\,k,i+1}=\begin{cases}0, & y_{r,i}\leqslant 1\\ y_{r,i}-1, & y_{r,i}>1\end{cases} \tag{2}$$

根据公式(2),利用"球盒模型"可以证明:经过一定时隙 $T$ 之后,向量 $Y_T$ 的每个元素均为 1.这说明,无论 $Y_T$ 的初始情况如何,经过一定时隙 $T$ 之后,系统已达到稳定状态,CM 的每个输入只能收到一个 AG 信号,即,公式(1)的 $h$=1.这表明,变量 $X$ 在系统稳定时可认为服从几何分布,因此,AG 信号是否被接受就由上述两个条件完全决定,则变量 $X$ 的均值 $E(X)=\frac{nk^2}{p(k-x+s)}$. 这是在平均情况下 CM 级的迭代次数,但由仿真实验可知,实际情况比这个理论结果要好很多.如图 2 所示:当迭代次数为 $\log_2 N$ 时,就能取得较为理想的效果.

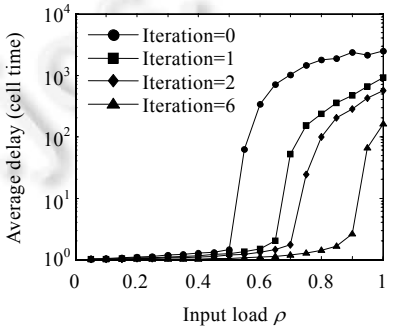



Fig.2 Simulated performance of iterations of CRRD-AG ($n$=$m$=$k$=8)

图 2 CRRD-AG 的 CM 级迭代次数的仿真性能($n$=$m$=$k$=8)

另外,如果根据文献[13]中关于调度器的设计思想,调度器的运行时钟频率设计为 175MHz(可能会更高),那么一个时钟周期约为 5.7ns.如果端口速度为 10Gbps,交换结构采用定长交换,分组长度为 64B,则交换一个分组所需时间(一个时隙)约为 51ns,包含约 9 个时钟周期.在调度器中,仲裁器间用高速串行链路(SerDes)连接,这样,在一个时钟周期内足以完成一个仲裁信息的传输.另外,算法中的多次迭代可采用流水线(pipeline)技术实现,而且各输出仲裁器是并行工作.因此,在一个时隙内完成端口的匹配是可行的,而且我们假定每次传输都在一个时隙的开始进行,所以我们在仿真实验中未考虑两种算法因端口匹配造成的一个时隙的延迟,只考虑了分组在 VOQ 中的等待时间.

## 4 仿真实验及性能分析

在仿真实验中,我们主要分析了 CRRD-AG 算法的吞吐率及平均延迟两大主要性能指标.仿真实验假设分组以贝努利(Bernoulli)和突发(bursty)两种到达过程到达交换结构的输入端,到达的分组在输出端口间有均匀分布(uniform)和非均匀分布(non-uniform).实验以 $C$(8,8,8)的 Clos 交换网络为仿真模型.

在实验中,业务负载假设从 0.05 增长到 1,步长为 0.05.平均延迟表示 VOQ 中一个信元在被传输之前的等待时间,忽略了仲裁信息的交互所带来的延迟(第 3.4 节分析了忽略的原因).实验的仿真时隙数尽可能地长,以确保

平均延迟的 95%的置信区间(按 $t$ 分布)长度小于 2%.

### 4.1 业务的到达及分布模型

任何调度算法的性能都会受业务的到达过程及分布情况的影响.对于独立同分布到达过程,假设分组以独立同分布逐时隙的到达方式到达每个输入端口,且相互独立地到达任一输入端口.每个输入端口是否有分组到达的概率设为$\rho$,同时表示该端口的负荷强度;对突发到达过程,本文采用 ON-OFF 两状态 Markov 模型来模拟分组的突发性.ON-OFF 模型表征为:分组只在 ON 状态下产生,而且可以连续产生 1 至多个分组(称为一个突发);在 OFF 状态下不产生分组.在 ON 或 OFF 状态上的持续时间是一个服从几何分布的随机变量[14].

假设 $p$ 和 $q$ 分别表示在 ON 和 OFF 状态上的转移概率(如图 3 所示),则在 ON 和 OFF 状态下的持续时间的概率分布如公式(3)和公式(4)所示.

$$\Pr[T_{\mathrm{ON}\to\mathrm{OFF}}=i|S_1=\mathrm{ON}]=(1-p)^{i-1}p,\ i\geq 1 \tag{3}$$

$$\Pr[T_{\mathrm{OFF}\to\mathrm{ON}}=j|S_1=\mathrm{OFF}]=(1-q)^{j-1}q,\ j\geq 0 \tag{4}$$

根据公式(3)和公式(4)的概率分布以及马尔可夫稳定方程,可以得出相应的参数:

- 突发的平均长度 $burst\_length=1/p$;
- ON 状态持续平均长度(单位为时隙)$average\_on=1-p$;
- OFF 状态的持续平均长度 $average\_off=1-q$.

当突发的平均长度(burst_length)和负荷强度($\rho$)已知时,这些参数就可计算出来.

流量分布是指到达交换系统输入端口的业务流在输出端口间的选择问题,通常,流量的分布可分为均匀分布模型及非均匀分布模型:均匀分布是指输入端口 $i$ 有信元到达,则其目的(输出)端口均匀分布在所有输出端口,即,每个输出端口 $j(j=1,2,\ldots,N)$被选中的概率均为 $1/N$,其中,$N$ 表示输出端口数量;非均匀分布模型有多种不同的形式,在本文实验中,采用如公式(5)[8,14]所示的非均匀分布模型.假设输入端口 $s$ 到输出端口 $d$ 的流量负载为 $p_{s,d}$,如公式(5)所定义.

$$p_{s,d}=\begin{cases}p\left(w+\dfrac{1-w}{N}\right), & s=d\\ p\dfrac{1-w}{N}, & \text{otherwise}\end{cases} \tag{5}$$

其中,$N$ 表示输出端口数量,$w(0\leq w\leq 1)$表示非均匀因子.此非均匀分布模型引用一个非平衡因子 $w$ 作为输入负载的一部分流量直接交换到预先确定的输出端口,而其余部分$(1-w)$流量以均匀分布的方式交换到其余所有输出端口.从公式(5)可以看出:当 $w=0$ 时,就变成了均匀模型;当 $w=1$ 时,就是完全不均匀.需要注意的是:在实验中,当分组以突发形式到达时,其非均匀分布未采用公式(5)所示的模型,而是同一个突发的所有分组到达相同的输出端口,不同突发的分组在输出端口间均匀分布,这和绝大多数的研究是一致的.

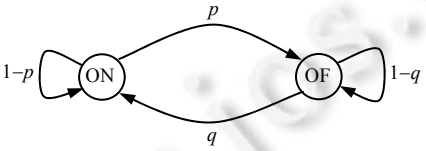



Fig.3 ON-OFF model

图 3 ON-OFF 模型

### 4.2 实验结果与分析

为了分析 CS 级链路带宽的使用情况,我们首先定义 CS 级的 IM 与 CM 之间的匹配率 $R_{\mathrm{IM\_CM}}$[8].

$$R_{\mathrm{IM\_CM}}=\sum_r\frac{\sum_i M_{IM(i)\_CM(r)}}{\sum_i\theta(i,r)} \tag{6}$$

如果 $IM(i)$有到 $CM(r)$的请求,则$\theta(i,r)$=1,否则为 0;如果 $CM(r)$授权了 $IM(i)$的请求,则 $M_{IM(i)\_CM(r)}$=1,否则为 0.文献[8]给出计算每个 CM 模块的匹配率,我们在仿真实验中则计算整个系统中 IM 与 CM 之间的匹配率.从图 4 可以看出:无论分组以 Bernoulli 还是以 Bursty 到达过程到达交换系统,CRRD-AG 算法在 CS 级的匹配率都明显高于 CRRD,即,CRRD-AG 算法对 CS 级的输出带宽的利用率要高于 CRRD 算法.这是 CRRD-AG 算法的平均延迟优于 CRRD 算法的根本原因.

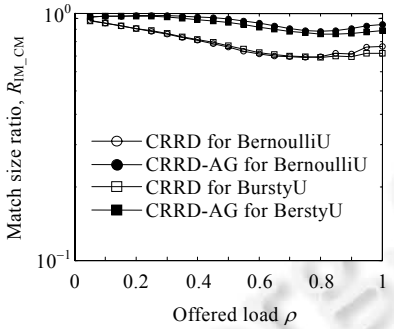
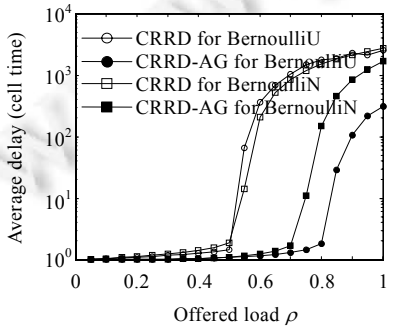


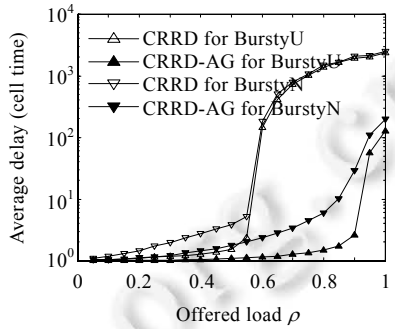

Fig.4 Match size ratio $R_{IM\_CM}$ in CM stage ($n$=$m$=$k$=8)

图 4 CM 级匹配率 $R_{IM\_CM}$ ($n$=$m$=$k$=8)

图 5 反映了 CRRD-AG 和 CRRD 算法在各种业务流下的平均延迟性能:图 5(a)反映出分组以独立同分布的 Bernoulli 过程到达交换系统的输入端口,而且以均匀分布(在图中以 BernoulliU 表示)和非均匀分布(图中以 BernoulliN 表示)两种形式分布于输出端口;图 5(b)反映出分组以突发过程到达输入端口时两种算法的延迟性能差异,分组在输出端口间仍以均匀分布(BurstyU)和非均匀分布(BurstyN)选择.




(a) Bernoulli 到达




(b) Bursty 到达

Fig.5 Average delay-performance comparison between CRRD-AG and CRRD algorithms

图 5 CRRD-AG 和 CRRD 算法的平均延迟性能比较

从图 5 的仿真结果可以看出:无论业务流以哪种到达过程到达交换网络,无论到达业务流在输出端口间是均匀分布还是非均匀分布,CRRD-AG 算法都能达到 100%的吞吐率,其平均延迟明显低于 CRRD 算法.这是因为 CRRD 算法在执行 IM 与 CM 的匹配时,那些被 CM 出口拒绝的分组会在下一个时隙再次匹配,导致 CM 的部分出口链路在当前时隙处于空闲状态,未被利用,在 VOQs 里留下的分组就增多,从而增加了延迟;而 CRRD-AG 算法充分利用 CM 的所有出口带宽,每个时隙尽可能地匹配 IM 中的请求,这样,CRRD-AG 算法在每个时隙成功传输的分组数量比 CRRD 算法要多,VOQ 中滞留的分组相对减少,从而降低了整个系统的平均延迟.虽然 CRRD-AG 算法的优势在负载较轻时并不明显,这是因为在低负载情形下,整个系统并未达到饱和状态,CRRD-AG 算法的中间级迭代并没有发挥出优势;随着负载的增加,整个系统逐渐达到饱和状态,CRRD-AG

算法的优势就凸显出来.

## 5 总 结

如今,高速的传输能力和层出不穷的网络业务对带宽的需求均给交换系统的交换能力带来了前所未有的压力.Clos 交换结构因具有良好的可扩展性倍受关注,但与之相适应且行之有效的调度算法并不多.目前的主流算法要么就是吞吐率不高或者需要一定的加速比才能取得较好的吞吐率,要么就是不能很好地处理各种业务流,比如突发业务流.为了适应新型高速路由与交换系统的研究需求,我们提出了一种主动授权的启发式迭代算法——CRRD-AG 算法.CRRD-AG 算法将经典的三阶段 RGA 匹配计算模式改进为 AG-A 匹配计算模式,不仅能够有效降低多级交换网络的第 1 级的额外信息交换量,降低对有限存储的需求,增强算法的可扩展性,而且 CRRD-AG 算法在中间级还增加了主动授权,充分利用了中间级的链路带宽,能够明显降低整个交换系统的平均延迟,提高整个系统的吞吐率.通过充分的实验,其结果表明:CRRD-AG 算法在没有内部加速比(或加速比为 1)的情况下可以取得 100%的吞吐率,而延迟明显低于 CRRD 算法,而且不仅在均匀业务环境中能有 100%的吞吐率,在突发业务环境中依然如此,较 CRRD 更适合高速大容量交换环境.

本文提出的"主动授权"的思想,在本质上就是最大化每级的匹配数量,降低延迟.在我们的研究中,虽然是将这种思想应用于 CRRD,而且取得了明显的效果,但更为重要的是,这种思想能够适用于其他多级结构的迭代算法,如 Distro 算法[9]等,因此,这不仅仅是对某种算法的修改,而是能够对多级结构调度性能产生一定影响的设计思想.

**References**:

[1] Poletti F, Wheeler NV, Petrovich MN, Baddela N, Fokoua EN, Hayes JR, Gray DR, Li Z, Slavı'k R, Richardson DJ. Towards high-capacity fibre-optic communications at the speed of light in vacuum. Nature Photonics, 2013,7(4):279−284. [doi: 10.1038/nphoton.2013.45]

[2] Clos C. A study of non-blocking switching networks. Bell Systems Technical Journal, 1953,32(3):406−424. [doi: 10.1002/j.1538-7305.1953.tb01433.x]

[3] McKeown N, Mekkittikul A, Anantharam V, Walrand J. Achieving 100% throughput in an input-queued switch. IEEE Trans. on Communications, 1999,47(8):1260−1267. [doi: 10.1109/26.780463]

[4] Anderson T, Owicki S, Saxe J, Thacker C. High speed switch scheduling for local area networks. ACM Trans. on Computer Systems, 1993,11(4):319−352. [doi: 10.1145/161541.161736]

[5] McKeown N. The *i*SLIP scheduling algorithm for input-queued switches. IEEE/ACM Trans. on Networking, 1999,7(2):188−201. [doi: 10.1109/90.769767]

[6] Chao HJ, Park JS. Centralized contention resolution schemes for a large-capacity optical ATM switch. In: Proc. of the IEEE ATM Workshop. Fairfax, 1998. 11−16. [doi: 10.1109/ATM.1998.675103]

[7] Chiussi FM, Kneuer JG, Kumar VP. Low-Cost scalable switching solutions for broadband networking: The ATLANTA architecture and chipset. IEEE Communication Magazine, 1997,35(12):44−53. [doi: 10.1109/MCOM.1997.642833]

[8] Oki E, Jing Z, Cessa RR, Chao HJ. Concurrent round-robin-based dispatching schemes for Clos-network switches. IEEE/ACM Trans. on Networking, 2002,10(6):830−844. [doi: 10.1109/TNET.2002.804823]

[9] Pun K, Hamdi M. Distro: A distributed static round-robin scheduling algorithm for bufferless clos-network switches. In: Proc. of the IEEE GLOBECOM 2002. 2002. 2298−2302. [doi: 10.1109/GLOCOM.2002.1189041]

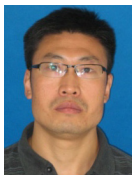
[10] Chao HJ, Jing Z, Liew SY. Matching algorithms for three-stage bufferless Clos network switches. IEEE Communications Magazine, 2003,41(10):46−54. [doi: 10.1109/MCOM.2003.1235594]

[11] Li Y, Panwar S, Chao HJ. On the performance of a dual round-robin switch. In: Proc. of the IEEE INFOCOM 2001. 2001. 1688−1697. [doi: 10.1109/INFCOM.2001.916666]

[12] Kleban J, Wieczorek A. CRRD-OG: A packet dispatching algorithm with open grants for three-stage buffered Clos-network switches. In: Proc. of the High Performance Switching and Routing (HPSR). 2006. 315−320. [doi: 10.1109/HPSR.2006.1709727]

[13] Gupta P, McKeown N. Designing and implementing a fast crossbar scheduler. IEEE Micro, 1999,1(1-2):20–28. [doi: 10.1109/40.748793]

[14] Tsaur DJ, Tang CF, Wu CC, Lin W. A threshold-based matching-algorithm for photonic clos network switches. In: Yang LT, Rana OF, Martino BD, Dongarra J, eds. Proc. of the HPCC 2005. LNCS 3726, Heidelberg: Springer-Verlag, 2005. 166–179. [doi: 10.1007/11557654_22]



**刘晓锋**(1972－),男,重庆人,博士生,CCF 会员,主要研究领域为计算机网络体系结构,路由与交换.

**赵有健**(1969－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机网络体系结构,路由与交换,计算机网络安全.

**陈果**(1989－),男,博士生,主要研究领域为数据中心网络,高性能交换.