基于流映射的负载均衡调度算法研究

戴 艺 苏金树 孙志刚
(国防科学技术大学计算机学院 长沙 410073)

摘 要 网络管理者需要能够提供可扩展性、吞吐率保证及报文顺序的高性能路由器体系结构.目前基于 Crossbar 的集中式路由器体系结构难以实现性能和规模的可扩展,基于两级 Mesh 网络的负载均衡交换结构成为扩展 Internet 路由器容量的有效的途径.负载均衡路由器存在严重的报文乱序现象,输出端报文重定序复杂度为 $O(N^2)$.文中提出一种区域均等的负载均衡交换结构,每 k 个连续的中间级输入端口划分为一个区域,输入端采用基于流映射的负载分配算法 UFFS-k(Uniform Fine-grain Frame Spreading,k 为聚合粒度,简称 UFFS-k),在 k 个连续的外部时间槽,以细粒度的方式将同一条流的 k 个信元分派到固定的映射区域,通过理论证明,该调度策略可获得 100%吞吐率并能够保证报文的顺序.为避免流量区域集中现象,采用双循环(dual-rotation)方式构建不同输入端口的流到区域的映射关系;为实现负载在中间级输入端口的均衡分布,每个输入端口维护全局统一视图的流量分布矩阵,UFFS-k 调度算法根据流量分布矩阵调度单位帧,可以证明,对任意输出端口 j,同一区域 OQ_j 队列长度相同且不同区域 OQ_j 队列长度至多差 1,从而实现了 100%负载均衡度.UFFS-k 调度算法分布于每个输入端口独立执行,根据流到区域的映射关系及负载分布状态分派信元,模拟结果显示,当聚合粒度 $k=2$ 时,UFFS-k 算法在同类维序算法中表现出最优延迟性能.

关键词 路由器体系结构;负载均衡路由器;报文乱序;双循环映射;可扩展
中图法分类号 TP306 **DOI 号**: 10.3724/SP.J.1016.2012.00218

Research of a Load-Balanced Algorithm Based on Flow Mapping

DAI Yi SU Jin-Shu SUN Zhi-Gang
(*School of Computer, National University of Defense Technology, Changsha* 410073)

Abstract Network operators would like high capacity router architectures that can offer guaranteed throughput, scalability and maintain packet ordering. However, current centralized crossbar-based architectures cannot scale to higher performance and port counts. The load-balanced switch architectures rely on two identical stages of fixed configuration meshes appear to be an effective way to scale Internet routers to very high capacities, they have the critical problem of packet mis-sequence with reordering complexity of $O(N^2)$. In this paper, we propose a region equalization architecture, in which every k consecutive intermediate inputs are organized as a region and each input port adopts a Uniform Fine-grain Frame Spreading (UFFS-k, where k is the aggregate factor) algorithm assigning k cells of the same flow to the fixed mapping region in a fine-grain manner during k successive external time slots. We give theoretical proofs that 100% throughput and packet ordering can be achieved by using UFFS-k algorithm at each input. To avoid traffic concentration in regions, a dual-rotation mapping algorithm is used to construct a mapping relationship between flows from different inputs and regions. Furthermore, in order to distribute input traffic equally among the intermediate inputs, the UFFS-k algorithm dispatches

收稿日期:2010-07-02;最终修改稿收到日期:2011-10-26.本课题得到国家自然科学基金(61003301)、国家"八六三"高技术研究发展计划项目基金(2008AA01A325)资助.**戴 艺**,女,1980 年生,博士,讲师,主要研究方向为路由器体系结构、高性能报文交换及交换调度算法.E-mail: y_dai@163.com. **苏金树**,男,1962 年生,博士,教授,博士生导师,主要研究领域为计算机网络体系结构、信息安全. **孙志刚**,男,1973 年生,博士,研究员,主要研究领域为计算机网络体系结构和通信技术、高性能报文交换体系结构.

each unit frame according to a global traffic distribution matrix maintained at each input. It is proofed that for each output port j, the lengths of OQ_j are equal for the same region and the lengths OQ_j differ by at most 1 for any two different regions, thus achieving 100% load balancing degree. The UFFS-k algorithm is distributed and can operate independently in each input. It spreads each flow to intermediate inputs according to the mapping relationship between flows to regions as well as traffic distribution state. As the simulation results demonstrate, when $k=2$, UFFS-k offers improved delay performance among existing scheduling algorithms that guarantee packet ordering.

Keywords　router architecture; load-balanced routers; packet mis-sequence; dual-rotation mapping; scalability

1　引　言

新一代互联网在规模、功能、性能和服务等多个维度上的可扩展特性需要核心路由器实现交换容量和端口密度的可扩展、提供延迟和吞吐率保证并维持报文的顺序.针对基于 Crossbar 的路由器体系结构受限于 LVDS 传输及封装技术,难以支持高速链路及高密度端口的问题[1],近年来,基于两级 Mesh 网络的负载均衡交换结构成为扩展路由器容量与规模的有效途径.负载均衡交换结构(load-balanced switch)最初由 Chang 等人[2]在他们设计的 Birkhoff-von Neuman 输入排队交换机中提出,它由第一级不带任何输入/输出缓冲区的 Crossbar 和第二级输入端带有 *VOQ* 队列的 Crossbar 构成.每一级 Crossbar 按照单循环置换矩阵周期性地建立连接模式,第二级 Crossbar 也就是 Birkhoff-von Neuman 输入排队交换机能够获得 100%的吞吐率(对于这一结论,Chang 等人[2]进行了严格的证明);模拟结果显示,在重负载的情况下,信元平均延迟接近 *OQ* 交换结构[2].针对 Crossbar 结构需要动态变化连接关系,难以扩展到更高速度的问题,近几年提出了一种采用两级全相连 Mesh 网络替代两级 Crossbar 的负载均衡交换结构[3-5].这种采用固定连接方式的负载均衡交换结构被认为是一种能够将路由器扩展到高容量和高速度的切实可行的方法.该交换结构的可扩展性得益于以下两个关键特性:(1)不需要集中式的调度器,在 $O(1)$时间内,所有的缓冲和转发操作在每个线卡本地完成.(2)采用两级固定配置的 Mesh 网络,其确定的连接模式独立于报文到达过程.因此不需要为每个报文动态配置交换网络,这种固定的连接方式也易于用光互连技术实现.斯坦福大学基于负载均衡交换结构,实现了一个 100 Tb/s 的光路由器[4],包含 640 个线卡,每个运行在 160 Gb/s 的速率.采用了一种层次化的 Mesh 体系结构,640 个线卡被组织成 40 个机柜,每个机柜包含 16 个线卡.机柜间通过静态配置的 40×40 光交换器件 MEMS(等同于光 Crossbar)进行互连.MEMS 交换速率可配置,当加入或移除线卡时,需要调节 MEMS 速率以保证流量畅通[4].

2　相关研究

网络流量特性决定路由器处理的数据模型.对于并行路由器而言,最重要的是报文乱序属性,研究者采用主动测量和被动测量方式对报文乱序行为做了大量研究.Bennett 在 MAE-East ISP 的交换中心测量了报文乱序情况,发现在重负载、网络设备并行度高的测量环境下,报文乱序情况非常严重,90%以上的连接发生乱序[6].Bennett 分析这种高乱序率主要来自网络内部的局部并行处理机制,包括并行交换设备和并行传输链路,并指出报文乱序不是网络的病态行为.负载均衡交换结构存在报文乱序现象,从最初基于两级 Crossbar 的负载均衡交换结构[2]到目前广泛研究的基于两级 Mesh 网络的负载均衡交换结构[3-4,7-8]均受到报文乱序问题的困扰.乱序报文会损害 Internet 网况,因为 Internet 广泛使用的 TCP 传输协议会错误地将乱序报文看作是报文丢失拥塞发生的标志,从而引发不必要的重传及 TCP 超时.这些重传和超时将降低 TCP 吞吐率,提高报文延迟,因此在路由器中保证报文的顺序是极其必要的.

防止报文乱序的方法可以分为两类:(1)限定乱序报文的数量,在输出端设置容量有限的重定序缓冲区,用于重定序乱序报文;(2)保证报文按照到

达顺序离开输出端口,从而避免了报文乱序.在负载均衡交换结构中,文献[4,7]采用的是基于限定报文重定序数量的第1种方法.由于缓冲区的容量有限,这些方法只能处理一定范围内的乱序报文,如果将重定序缓冲区尺寸增加至 $O(N^2)$,虽然能够完全解决报文乱序的问题但会相应地以二次方的时间尺度增加报文延迟,其中 N 为端口数目.因此,这些方法并不能有效解决报文乱序问题,并且难以适应路由器高端口密度的需求.文献[3]所提出的满帧优先(Full Ordered Frame First,FOFF)算法是第1种方法的代表算法,它允许路由器中存在一定数量的乱序报文,在输出端设置了容量为 N^2 个信元的 N 个的缓冲队列用于重定序乱序报文.文献[3]证明了,为了保证信元按序离开,重定序缓冲区容量至多为 N^2 个信元.为了限定乱序信元的数量,FOFF 算法每 N 个外部时间槽只能服务一个队列,当被服务队列的信元不能占用全部的内部链路时(所含信元小于 N 的情况),由于其它队列不能使用剩余的空闲链路,将造成 Mesh 网络带宽浪费,增加信元平均延迟.

近年来,更多的研究者趋向于采用第2种方法来保证报文的顺序,消除了输出端的重定序操作及缓冲区开销,有利于提高延迟性能[9-11].文献[5]提出一种邮箱交换(mailbox switch)的思想.采用对称型连接模式为通告报文离开时间创造反馈通路,调度器根据报文的离开时间调度报文.该策略能够保证每条流的报文按照其到达顺序离开交换系统但不能提供100%的吞吐率.文献[11]提出一种能够保证报文顺序的交替匹配交换结构(interleaved matching switch),采用集中式的调度模式,假设流量特征是预知的且固定不变,采用矩阵分解的方法[12]离线解决集中式调度问题,分布式实现在线调度并提供服务保证.然而,当流量变得不可预知并动态改变时,难以在大的交换尺寸下满足集中式的调度需求.文献[9]提出一种并发匹配交换结构 CMS(Concurrent Matching Switch),其基本思想为:通过在中间级线卡设置协同槽(coordination slots)即长度为1的缓冲区,保证了所有匹配报文从输入线卡到达目的输出线卡经历了相同的延迟,从而避免了信元乱序.CMS 结构首先为每个到达报文发送请求令牌到中间级输入线卡而不是报文本身,每一个请求令牌相当于一个占位符.中间级输入线卡为每一条流的请求令牌计数,因此需要维护 N^2 个虚拟令牌计数器,构成 $N\times N$ 的虚拟令牌计数矩阵.每一个中间级输入线卡根据本地虚拟令牌计数矩阵计算匹配,不需要任何全局的状态信息或从其它中间级输入线卡获取虚拟令牌计数信息,并根据匹配结果发送许可令牌到每一个输入线卡.任何双向匹配算法都可以用于计算 CMS 结构中的匹配问题,文献[11]证明,只要中间级输入线卡使用稳定的匹配算法,带流分离操作的 CMS 结构是稳定的,从而能够提供100%的吞吐率.然而 CMS 结构采用双向匹配算法,算法复杂度至少为 $O(N^2)$;另一方面,CMS 结构假设输入线卡和中间级输入线卡之间的令牌通信开销可以忽略不计,而在路由器硬件实现中令牌在线卡间的频繁传递将成倍增加调度周期[13].在负载均衡交换结构中,信元乱序问题源自不同的中间级输入线卡中目的端口相同的缓冲队列具有不同的长度.鉴于此,文献[3]提出一种 UFS(Uniform Frame Spreading)算法,通过将同一条流的 N 个信元依次分派到 N 个中间级输入线卡,使得中间级输入线卡相同目的端口的缓冲队列具有相同的长度,从而保证了信元的顺序.当流所含信元数目小于 N 时则需等待,因此在高端口密度下,UFS 算法所引发的信元延迟是不可容忍的.

为了降低 UFS 算法信元聚合延迟,我们在早期论文[14]中提出了固定映射的思想,由此设想了一种基于流映射的细粒度负载分配算法 UFFS-k(Uniform Fine-grain Frame Spreading,k 为聚合粒度,简称 UFFS-k),通过构建流到区域的固定映射,实现报文的顺序并通过降低聚合粒度以提高延迟性能.论文[14]提出的循环映射算法本身并不能保证负载均衡,本文首先通过理论证明,UFFS-k 算法能够保证报文的顺序;原创性地提出最小长度分派以及流量分布矩阵互斥写策略,证明了最小长度分派结合流量分布矩阵互斥写策略能够获得100%负载均衡度;最后在贝努利一致流量模型和突发流量模型下分析评测 UFFS-k 算法在不同聚合粒度 k 下的延迟性能,并与目前主流的负载均衡调度算法进行比较,模拟结果显示,当聚合粒度 $k=2$ 时,UFFS-k 算法在同类维序算法中表现出最优延迟性能.

3 UFFS-k 算法基本思想

3.1 体系结构

UFFS-k 算法采用的负载均衡交换结构如图1所示,输入端口将到达信元按照目的端口缓冲在 VOQ 队列,VOQ 队列来自同一条流的 k 个信元,称之为一个单位帧;N 个中间级输入端口被依次划分

为 N/k 组，每组含 k 个连续的中间级输入端口构成一个区域；每个输入端口独立执行 UFFS-k 信元分派算法，在 k 个连续的外部时间槽，UFFS-k 将同一条流的单位帧通过第一级 Mesh 网络发送该流固定的映射区域；各区域按照目的端口将信元缓冲在 OQ 队列，由于流到区域的映射关系固定，来自同一单位帧的 k 个信元将依次到达 OQ 队列头位置，等待第二级 Mesh 网络空闲时依次离开输出端口.

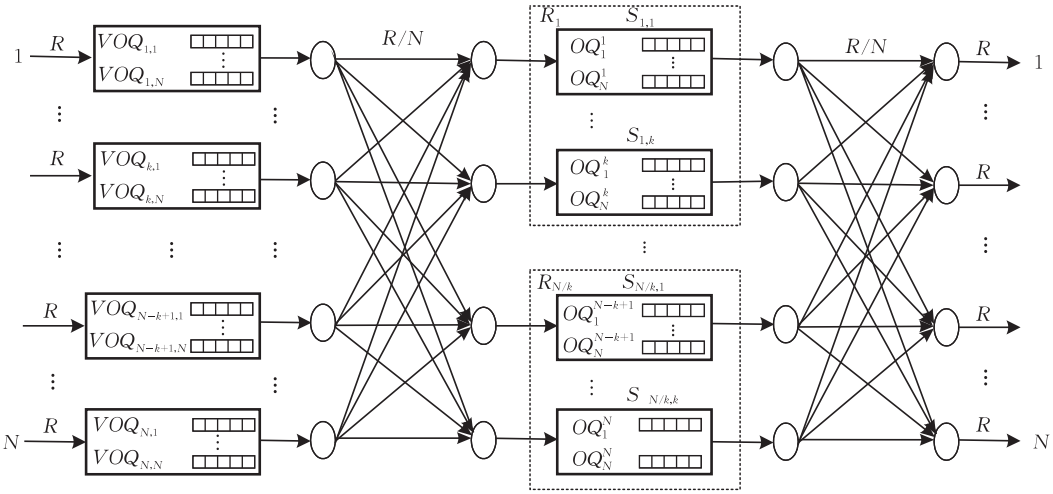

图 1　负载均衡交换结构

参照文献[14]给出的符号与定义，我们用 $VOQ_{i,j}$ 表示输入端口 i 的虚拟输出队列 j；OQ_j^l 表示中间级输入端口 l 的输出队列 j；$f(i,j)$ 表示从输入端口 i 到输出端口 j 的流；k 为聚合粒度，表示 UFFS-k 算法连续调度同一条流信元的数目（k 为端口数 N 的因数）；$VOQ_{i,j}$ 队列的每 k 个信元构成一个单位帧（unit frame），输入端 i，N/k 个不同 VOQ 队列的单位帧（共计 N 个信元）构成一个聚合帧（aggregate frame），$VOQ_{i,j}$ 队列的 N 个信元，构成一个满帧（full frame）；中间级输入端口 $1,2,\cdots,N$ 被依次分成 N/k 组，每一组含 k 个连续的中间级输入端口，第 g 组的 k 个中间级输入端口构成一个区域，记作 R_g，$S_{r,z}$ 表示区域 r 的第 z 个中间级输入端口；每个输入端口的 N 条流被划分为 N/k 组与 N/k 个区域相对应，每组含 k 条流，输入端口 i 第 f 组的 k 条流构成一个流分支，记作 Q_f^i.

为降低报文缓冲存储器带宽需求，Mesh 网络通常运行在速率 R/N（内部链路加速比为 1），由此得到以下定义.

定义 1.　在速率为 R 的链路发送或接收一个信元所耗费的时间为外部时间槽（external time slot）.

定义 2.　在速率为 R/N 的链路发送或接收一个单位帧所耗费的时间为时间槽（time slot），时间槽为外部时间槽的 N 倍.

定义 3.　受输入链路约束影响[15]，输入端口每 N 个外部时间槽恰能发送一个信元到同一中间级输入端口，可用输入链路集合（Available Input Link Set）是指在外部时间槽 e，能够从输入端口 i 接收信元的中间级输入端口的集合，记作 $AIL(i,e)$.

定义 4.　受输出链路约束影响[15]，中间级输入端口每 N 个外部时间槽恰能发送一个信元到同一输出端口，可用输出链路集合（Available Output Link Set）是指在外部时间槽 e，能够将信元发送到输出端口 j 的中间级输入端口的集合，记作 $AOL(j,e)$.

在负载均衡交换结构中，当同一条流的信元所在中间级输入线卡 OQ 队列长度不同时就会引发信元乱序，并且乱序信元数目随着 OQ 队列长度差异的增大而增大. UFS 算法通过将同一条流的 N 个信元分派到中间级输入线卡所有的 OQ 队列，保证了 OQ 队列长度的一致性从而实现了信元保序. UFS 的缺点是需要在输入端 VOQ 队列聚合 N 个信元，聚合粒度太大，增加了信元延迟，并存在“饿死”现象. 针对以上问题，UFFS-k 算法以细粒度的方式将同一条流的 k 个信元分派到预先设定的映射区域（k 个连续的中间级输入端口），由于流到区域的映射关系固定，对任意流，其信元所在映射区域的 OQ 队列长度相同，从而实现了信元的按序发送，我们在 3.3 节证明了该结论.

3.2　建立流到区域的映射

流到区域的映射算法用于构建流到区域固定的映射关系，它关系到中间级输入端存储资源及两级

Mesh 网络的利用率，为避免吞吐量损失(loss of throughput)流到区域的映射算法应实现输入负载在各区域的均衡分布.本节提出一种兼顾负载均衡和报文保序的双循环(dual-rotation)映射算法，其主要思想与我们之前提出的循环映射算法[14]类似：既然对于任意给定的区域，只能接收同一输入端口固定的 k 条流，那么为实现负载在各区域的均衡分布，以循环方式调整流分支到区域的映射，从而得到 N/k 种映射方式.如图 2 所示，第 1 层循环保证了每个区域恰能涵盖所有的流；进一步以循环方式调整不同输入端口与 N/k 种映射方式的关联，第 2 层循环保证了每个区域按照输入端口的均衡分布涵盖所有的流.从 3.3 节可以看到，UFFS-k 算法根据中间级输入端口流量分布矩阵调度单位帧，进一步保证了每条流在区域之间的均衡性.双循环映射算法通过简单的取模运算建立流到区域的映射关系，可描述如下：

```
//Define the following constants:
  //N, Number of ports
  //k, Aggregate factor
//Define the following variables:
  //i, Integer, input port number
  //f, Integer, flow branch number
  //g, Integer, region number
for(i=0, i<N, i++)
  for(g=0, g<N/k, g++)
    FlowMapping (Q_f^i, R_g) /*Map flow branch Q_f^i to R_g*/
Function FlowMapping (Q_f^i, R_g) R_g
    if (k==1)
      f=(g+i)%(N/k);
    else{
      if (g<(i%(N/k)))
        f=(g+(N/k))-(i%(N/k));
      else
        f=g-(i%(N/k));
       }
EndFunction FlowMapping
```

图 2 显示了当端口数目 $N=32$，聚合粒度 $k=8$ 时，循环映射算法得到的流到区域的映射结果(VOQ_{ij} 代表从输入端口 i 到输出端口 j 的流，→表示映射关系).

3.3 UFFS-k 调度算法

UFFS-k 算法分布于每个输入端口独立执行，根据流到区域的映射关系分派信元.UFFS-k 算法以轮询(round-robin)方式服务于 N/k 个流分支，以单位帧为最小调度单元，在连续的 k 个外部时间槽，发送流分支中固定 VOQ 队列的单位帧，输入端口

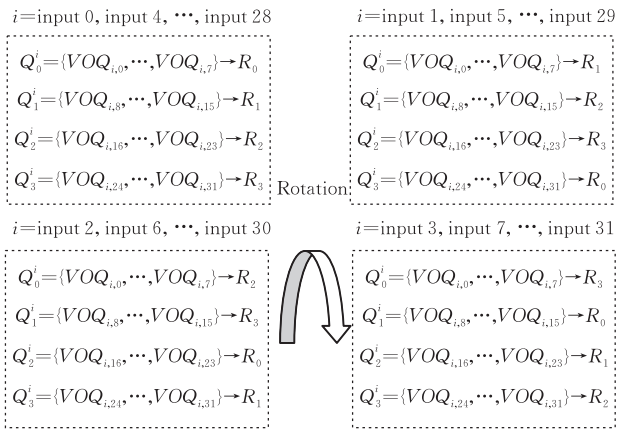

图 2 循环映射算法结果($N=32$，$k=8$)

i UFFS-k 调度算法每个外部时间槽的操作可描述如下：

步 1. 对区域 R_g(g 初始化为 0)，若流分支 Q_f^i(f 满足等式$(f+i\%(N/k))\%(N/k)=g$)最小均衡系数 VOQ 队列 $VOQ_{i,j}$($kf\le j<kf+k$)存在单位帧且输入端口 i 到中间级输入端口 $S_{g,1}$ 的内部链路空闲，则将 $VOQ_{i,j}$ 队列头信元发送到区域 R_g 中间级输入端口 $S_{g,1}$；若 Q_f^i 最小均衡系数 VOQ 队列不存在单位帧或者输入端口 i 到中间级输入端口 $S_{f,1}$ 的内部链路忙，则 $g=(g+1) \bmod N/k$，返回步 1；

步 2. 发送 $VOQ_{i,j}$ 队列头信元到区域 R_g 中间级输入端口 $S_{g,2}$；…

步 k. 发送 $VOQ_{i,j}$ 队列头信元到区域 R_g 中间级输入端口 $S_{g,k}$，$g=(g+1) \bmod N/k$，返回步 1.

通常情况下，在每个时间槽即每 N 个外部时间槽，UFFS-k 调度算法可从输入端 N/k 个流分支中聚合 N/k 个单位帧构成一个聚合帧分派到中间级输入端口.VOQ 队列均衡系数反映了流量在中间级输入端口 OQ 队列分布的均衡性，UFFS-k 调度算法根据各区域 OQ 队列长度调度单位帧，实现了区域间的负载均衡.下面将通过证明详细阐述均衡系数工作原理.

引理 1. 对任意输入端口 i，$e\ge 0$，输入链路集合个数满足 $|AIL(i,e)|\ge N-\lceil N/S\rceil+1$，其中 S 为内部链路加速比.

证明. 对引理 1 的证明见参考文献[16].

引理 2. UFFS-k 调度算法在没有考虑内部链路加速比的情况下($s=1$)，对任何区域 r，若 $S_{r,1}\in AIL(i,E)$ 则有 $S_{r,z}\in AIL(i,E+z-1)$，$z=2,\cdots,k$，且 UFFS-k 调度算法在任意时间槽即 N 个连续的外部时间槽，可用输入链路集合满足 $\bigcup_{v=E}^{E+N-1} AIL(i,v)=\bigcup_{0\le r<N/k}\{\bigcup_{z=1}^{k} S_{r,z}\}$.

证明. $S_{r,1}\in AIL(i,E)$ 表明中间级输入端口

$S_{r,1}$在最近的 $N-1$ 个外部时间槽内没有从输入端口 i 接收信元(若 $S_{r,1}$在外部时间槽 $E-(N-1)$接收过信元,那么 $N-1$ 个外部时间槽以后,$S_{r,1}$仍不可用,即 $S_{r,1}\notin AIL(i,E)$).不失一般性,假设 $S_{r,1}$在外部时间槽 $E-N$ 从输入端口 i 接收过信元,那么根据输入链路约束,N 个外部时间槽以后,$S_{r,1}$重新被释放即 $S_{r,1}\in AIL(i,E)$.由于 UFFS-k 调度算法总是在连续的 k 个外部时间槽依次发送信元至 $S_{r,1},\cdots,S_{r,k}$,那么 $S_{r,2},\cdots,S_{r,k}$分别在外部时间槽 $E-N+1,\cdots,E-N+k-1$ 接收信元,根据输入链路约束,$S_{r,z}\in AIL(i,E+z-1)$,$z=2,\cdots,k$ 成立.

接下来证明引理 2 的第二部分.根据输入链路约束不难推断,在任意 N 个连续的外部时间槽内不存在同一个中间级输入端口两次加入到可用输入链路集合中,该特性可表示为若 $S_{r,z}\in AIL(i,e)$,$S_{r',z'}\in AIL(i,e')$则 $S_{r,z}\neq S_{r',z'}$其中 $1\leq|e-e'|\leq N-1$.又由引理 1 知,$|AIL(i,e)|\geq N-\lceil N/S\rceil+1=1$,即在任意外部时间槽,至少有 1 个中间级输入端口可用,不失一般性,假设在每个外部时间槽 e,存在 $S_{r,z}\in AIL(i,e)$从输入端口 i 接收信元,那么根据引理 1 从 E 到 $E+N-1$ 连续的 N 个外部时间槽,一定有 $N-|AIL(i,E)|$个不同的中间级输入端口加入可用输入链路集合中,从而有 $\bigcup_{v=E}^{E+N-1}AIL(i,v)=\bigcup_{0\leq r<N/k}\bigcup_{1\leq z\leq k}S_{r,z}$ 成立,之前已经证明对任何区域 r,$\bigcup_{z=1}^{k}S_{r,z}$总是在 k 个连续的外部时间槽,以 round-robin 的方式依次加入可用输入链路集合,将 $S_{r,1}\in AIL(i,E)$,$S_{r,z}\in AIL(i,E+z-1)$,$z=2,\cdots,k$ 等价表示为 $\bigcup_{v=E}^{E+k-1}AIL(i,v)=\bigcup_{1\leq z\leq k}S_{r,z}$后,在任意 N 个连续的外部时间槽,$\bigcup_{v=E}^{E+N-1}AIL(i,v)=\bigcup_{0\leq r<N/k}\{\bigcup_{z=1}^{k}S_{r,z}\}$成立.

证毕.

引理 3.　UFFS-k 调度算法保证对任意区域 R_g,$0\leq j<N$,k 个中间级输入端口对应的 k 个 OQ_j^l 队列长度相同(忽略单位帧的传输延迟),其中 $l\in R_g$.

证明.　采用对时间槽的归纳法证明.

对第 1 个时间槽,引理 3 显然成立.

假设在时间槽 T 结束后,引理 3 成立,那么只需证明在时间槽 $T+1$ 结束后,引理 3 依然成立.由引理 2 知,在任意时间槽,可用输入链路集合满足 $\bigcup_{v=e}^{e+N-1}AIL(i,v)=\bigcup_{0\leq r<N/k}\{\bigcup_{z=1}^{k}S_{r,z}\}$.因此对每个输入端口 i,在同一时间槽至多发送一个单位帧到特定区域 R_g.假设在时间槽 $T+1$,区域 R_g从 P 个输入端口接收到了 P 个单位帧,其中 $0\leq P\leq N$.不失一般性,假设该 P 个单位帧目的端口集合为$\{j_1,j_2,\cdots,j_{P'}\}$,其中 $1\leq P'\leq P$,那么区域 R_g目的端口 j($j=j_1,j_2,\cdots,j_{P'}$)对应的 k 个队列 $OQ_j^{S_{g,1}}$,$OQ_j^{S_{g,2}}\cdots$,$OQ_j^{S_{g,k}}$分别接收到 d($d=d_1,d_2,\cdots,d_{P'}$)个信元,d_1表示区域 R_g接收到的目的端口为 j_1的单位帧的数目,以此类推.在时间槽 T 结束后,由引理 3 知对输出端口 j,OQ_j^l队列长度相同,其中 $l\in R_g$,那么在时间槽 $T+1$ 结束后 $OQ_j^{S_{g,1}}$,$OQ_j^{S_{g,2}}$,$\cdots$,$OQ_j^{S_{g,k}}$($j=j_1,j_2\cdots j_{P'}$)在队列长度增加 d($d=d_1,d_2,\cdots,d_{P'}$)后仍然相等,其它 OQ_j^l队列长度保持不变,引理 3 在时间槽 $T+1$ 结束后依然成立.　　证毕.

定义 5.　既然对任意区域 R_g,OQ_j^l队列长度相同($l\in R_g$),那么队列 $VOQ_{i,j}$均衡系数等于其映射区域 R_g输出队列 j 的长度,记作 $L_{g,j}$.

定理 1.　UFFS-k 调度算法能够保证每条流按序离开输出端口.

证明.　该命题等价于证明:假设流 $f(i,j)$任意两个信元 C_1,C_2,若 C_1先于 C_2到达输入端口 i,那么 C_1先于 C_2离开输出端口 j.考虑 C_1,C_2属于同一单位帧和不同单位帧两种情况:

若 C_1,C_2属于同一单位帧,那么根据 UFFS-k 调度算法,在连续的 k 个外部时间槽,依次将单位帧的 k 个信元以循环(round robin)方式分派至其映射区域 R_g;假设 C_1缓冲在队列 $OQ_j^{l_1}$,C_2缓冲在队列 $OQ_j^{l_2}$,则 $l_1,l_2\in R_g$,按照 C_1,C_2到达顺序 $l_1<l_2$.假设 $OQ_j^{l_1}$,$OQ_j^{l_2}$队列长度分别为 L_1,L_2,由引理 3 知,$L_1=L_2$,根据输出链路约束,在每个时间槽,输出线卡 j 恰能从 OQ_j^l,$l=\{1,2,\cdots,N\}$依次读出 N 个信元,那么 L_1个时间槽后,C_1,C_2将按序离开输出端口 j.

若 C_1,C_2属于不同单位帧,按照到达顺序,UFFS-k 算法首先调度 C_1所在单位帧,其次调度 C_2所在单位帧,C_1先于 C_2到达流 $f(i,j)$的映射区域 R_g,若 C_2到达区域 R_g时,C_1已经离开输出端口 j,则得证.否则,假设以 C_1为界,$L_{g,j}=L_1$,以 C_2为界,$L_{g,j}=L_2$,则 $L_2>L_1$.C_1首先离开输出端口 j,L_2-L_1个时间槽后,C_2离开输出端口 j.　　证毕.

定义 6.　若队列 $VOQ_{i,j}$存在单位帧且均衡系数满足 $L_{g,j}=\min_{0\leq g<N/k}(L_{g,j})$,则在连续的 k 个外部时间槽将 $VOQ_{i,j}$队列单位帧发送到区域 R_g,这就是最

小长度分派.

引理 4. 采用最小长度分派的 UFFS-k 调度算法保证在时间槽 T 结束后,对任意两个区域 R_{g_1}, R_{g_2},其 OQ 队列长度 $L_{g_1,j}$ 与 $L_{g_2,j}$ 最多差 1.

证明. 采用对时间槽的归纳法证明.

对第 1 个时间槽,引理 4 显然成立.

假设在时间槽 T 结束后,引理 4 成立,那么只需证明在时间槽 $T+1$ 结束后,引理 4 依然成立.

在时间槽 T 结束后,由引理 4 知,假设 $\min\limits_{0\le g<N/k}(L_{g,j})=L$,则 $\max\limits_{0\le g<N/k}(L_{g,j})=L+1$,其中 $L\ge 0$. 在时间槽 $T+1$ 内,对任意输出端口 j,可能存在一个或多个输入端口将单位帧发送到满足 $L_{g,j}=\min\limits_{0\le g<N/k}(L_{g,j})$ 的区域 R_g,不失一般性,假设被调度的区域集合为 $\{R_{g_1},R_{g_2},\cdots,R_{g_p}\}$,其中 $p\le num(L_{g,j}=\min\limits_{0\le g<N/k}(L_{g,j}))$,那么其对应的 OQ 队列长度 $\{L_{g_1,j},L_{g_2,j},\cdots,L_{g_p,j}\}$ 由 L 变为 $L+1$. 若 $p=num(L_{g,j}=\min\limits_{0\le g<N/k}(L_{g,j}))$,则有 $\min\limits_{0\le g<N/k}(L_{g,j})=\max\limits_{0\le g<N/k}(L_{g,j})=L+1$;若 $p<num(L_{g,j}=\min\limits_{0\le g<N/k}(L_{g,j}))$,则有 $\min\limits_{0\le g<N/k}(L_{g,j})=L$, $\max\limits_{0\le g<N/k}(L_{g,j})=L+1$;与此同时,受输出链路约束影响,输出线卡 j 将依次从 OQ_j^l 读出 N 个信元,当时间槽 T 结束后,所有 $OQ_j s$ 队列长度全部减 1,即 $L_{g,j}(0\le g<N/k)$ 全部减 1,综上所述,引理 4 在时间槽 $T+1$ 结束后依然成立. 证毕.

UFFS-k 调度算法需要维护全局统一视图的流量分布矩阵 $\boldsymbol{L}=[L_{g,j}]$,为了保证流量分布矩阵在每个输入端口视图的一致性,必须实现对流量分布矩阵写操作的互斥性. 我们采用锁机制实现对互斥量 $L_{g,j}$ 的互斥写:若 g,j 满足 $L_{g,j}=\min\limits_{0\le g<N/k}(L_{g,j})$ 且 $L_{g,j}$ 处于解锁(unlock)状态,那么输入端口 i 对 $L_{g,j}$ 加锁后将 $VOQ_{i,j}$ 队列单位帧发送至其映射区域 R_g, $L_{g,j}$ 加 1 后被解锁. 流分支中均衡系数 $L_{g,j}$ 处于加锁状态的 $VOQ_{i,j}$ 队列直接被跳过. 不难推断,只有那些流到区域的映射关系相同的输入端口同时调度目的端口相同 $VOQ_{i,j}$ 队列时才可能引发对同一 $L_{g,j}$ 的写冲突,对均衡系数 $L_{g,j}$ 的互斥写可以避免这些输入端口同时将多个单位帧发送到同一区域的输出队列 j,造成流量分布不均衡. 注:本文对引理 4 的证明也是以实现流量分布矩阵 $\boldsymbol{L}$ 互斥写为前提的.

由于流分支到区域的映射关系固定,存在某些区域因负载过重而发生缓冲区溢出而其它区域相对空闲的情况. 例如,输入端某个流分支的 VOQ 队列含 N 个信元,并且队列长度仍在不断增长,而其它流分支没有单位帧可以调度. 这样重负载流分支到其映射区域的内部链路将成为性能瓶颈,另一方面重负载流分支因无法使用输入线卡其它空闲链路而造成内部带宽浪费. 为了解决以上问题,UFFS-k 算法允许重负载流分支抢占链路资源,通过赋予满帧最高优先级将突发报文流均匀分布于每个区域. UFFS-k 算法调度满帧时可能引发信元乱序,图 3 显示了将满帧分派到中间级输入端口时,出现了信元乱序现象. 图 3 中 4 个区域 $\{R_1,R_2,R_3,R_4\}$ 输出队列 j 的长度 $\{L_{1,j},L_{2,j},L_{3,j},L_{4,j}\}$ 分别为 3,2,2,3(灰色部分所示),当输入端将流 $f(i,j)$ 的 N 个信元分派到这 4 个区域时(黑色部分所示),区域 2、3 的信元先于区域 1 的信元到达目的端口 j,而正确的离开顺序应该是区域 1 的信元最先离开,然后是区域 2,区域 3 和区域 4. 为了解决调度满帧引发的信元乱序问题,我们将从 $VOQ_{i,j}$ 队列读出的 N/k 个单位帧(即一个满帧)依次分派到目前 $L_{g,j}$ 最小的区域 R_g. 采用该策略得到的分派顺序是:从 $VOQ_{i,j}$ 队列读出的第 1 个及第 2 个单位帧依次分派在区域 2 和区域 3,第 3 个单位帧及第 4 个单位帧依次分派在区域 1 或区域 4,这 4 个单位帧将按读出顺序依次到达输出端口 j. 由引理 4 知,不同区域输出队列 j 的长度至多差 1,因此将满帧中 N/k 个单位帧依次发送到目前 $L_{g,j}$ 最小的区域不会引发信元乱序. 本质上,调度单位帧与调度满帧都采用了最小长度分派策略,两者的区别在于单位帧只能固定分派到其映射区域,而满帧将被拆分为 N/k 个单位帧均衡分布于 N/k 个区域.

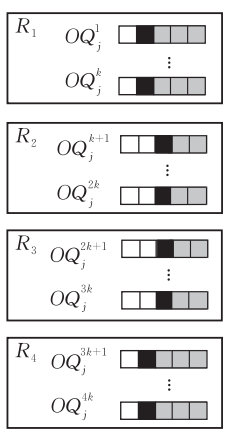

图 3 调度满帧所产生的信元乱序现象

3.4 UFFS-k 算法负载均衡度分析

我们采用负载均衡度来衡量负载在中间级输入线卡分布的均衡程度,负载均衡度可定义如下.

定义 7. 负载均衡度. 假设在时间段 $[t_r,t_v]$ 内的第 l 个交换模块转发了 $S_l[t_r,t_v]$ 个信元. 负载均

衡度为在该时间段内，不同交换模块转发的最小信元个数与最大信元个数的比值，即

$$E[t_r,t_v]=\frac{\min\limits_{l=0,\cdots,K-1}S_l[t_r,t_v]}{\max\limits_{l=0,\cdots,K-1}S_l[t_r,t_v]},$$

其中，K 为中间级交换模块的数目.

显然负载均衡度 $E[t_r,t_v]\leq 1$，$E[t_r,t_v]$ 趋近于 1，表示各交换模块处理的信元数基本相同，负载在交换模块的分布比较均衡. $E[t_r,t_v]$ 越小，交换模块负载均衡性越差.

定理 2.　UFFS-k 调度算法可获得 100%负载均衡度.

证明.　由引理 4 知，对任意两个区域 R_{g_1}，R_{g_2}，目的端口相同的 OQ 队列长度 $L_{g_1,j}$ 与 $L_{g_2,j}$ 最多差 1. 那么含 N 个 OQ 队列的交换模块，对任意时间段 $[t_r,t_v]$ 满足

$$\max_{l=0,\cdots,K-1}S_l[t_r,t_v]-\min_{l=0,\cdots,K-1}S_l[t_r,t_v]\leq N.$$

负载均衡度可计算为

$$E[t_r,t_v]=\frac{\min\limits_{l=0,\cdots,K-1}S_l[t_r,t_v]}{\max\limits_{l=0,\cdots,K-1}S_l[t_r,t_v]}\geq\frac{\max\limits_{l=0,\cdots,K-1}S_l[t_r,t_v]-N}{\max\limits_{l=0,\cdots,K-1}S_l[t_r,t_v]}$$
$$\geq 1-\frac{N}{\max\limits_{l=0,\cdots,K-1}S_l[t_r,t_v]}.$$

当 $\max\limits_{l=0,\cdots,K-1}S_l[t_r,t_v]\gg N$ 时，负载均衡度 $E[t_r,t_v]$ 趋近于 1，UFFS-k 调度算法可获得 100%负载均衡度.　证毕.

3.5　UFFS-k 算法吞吐率分析

3.5 节证明了 UFFS-k 算法能够获得 100%的吞吐率. 引理 3 摘自文献[3]，描述了工作保持(work-conserving)的服务特性.

引理 5.　考虑一种工作保持(work-conserving)的服务，令 $A(t)$ 和 $B(t)$ 分别表示在时间 t 以前到达和服务的信元数目. 假设服务容量是每个外部时间槽服务一个信元，那么对于所有的时间 $t\geq 0$，$B(t)=\min\limits_{0\leq s\leq t}[A(s)+t-s]$.

接下来，我们证明 UFFS-k 算法能够保证 100%的吞吐率. 考虑给定的输出端口 j，令 $A^j(t)$，$B^j(t)$，$C^j(t)$ 分别表示目的端口为 j 的到达输入端口、中间级输入端口及输出端口 j 的信元数目.

定理 3.　UFFS-k 算法和理想的输出排队交换结构具有相同的吞吐率，与输入流量到达过程无关.

证明.　首先，每个输入端口任意流分支至多包含 $k(k-1)$ 个信元，此时，流分支中没有一个单位帧可以调度，当存在单位帧可以调度时，输入端是工作保持的(在 $k-1$ 个外部时间槽的延迟范围内). 由于每个输入端在每个外部时间槽最多到达一个信元，因此当输入端工作保持时，流分支最多包含 $k(k-1)+k-1=k^2-1$ 个信元(输入端开始服务流分支后)，自此流分支中的信元数目不会进一步增加. 当不存在单位帧、输入端不再工作保持时，流分支信元数目不会超过 $k(k-1)$. 因此对任何流分支在任何时刻最多包含 k^2-1 个信元(不包括正在服务的信元). 那么 N 个输入端口最多包含 Nk^2 个信元，特别地，当所有到达信元目的端口相同时，最多有 Nk^2 个信元去往输出端口 j. 因此，在任意时刻 $t\geq 0$，$B^j(t)\geq A^j(t)-Nk^2$(在这里，假设信元传输是瞬时的. 如果报文传输延迟 $\tau>0$，那么 $B^j(t)\geq A^j(t-\tau)-Nk^2$，接下来的等式都照此简化).

令 $B^j_{FF}(t)$ 为已到达中间级输入端口的单位帧所含信元个数. 对任意流，当单位帧还没有全部到达中间级输入端时，最多有 $k-1$ 个信元在中间级 OQ 队列. 由于 UFFS-k 算法对流量分布矩阵写操作的互斥性，对于给定输出端口 j，至多有 N/k 个输入端口正在发送且未发送完毕目的端口为 j 的单位帧，则有

$$B^j_{FF}(t)\geq B^j(t)-N/k(k-1)$$
$$\geq A^j(t)-Nk^2-N/k(k-1).$$

只要存在单位帧完全到达中间级输入端口，则交换阶段是工作保持的. 因此，交换阶段至少与信元到达过程服从 $B^j_{FF}(t)$ 的工作保持服务(work-conserving server)一样多的信元，因此由引理 5 得到

$$C^j(t)\geq\min_{0\leq s\leq t}[B^j_{FF}(s)+t-s]$$
$$\geq\min_{0\leq s\leq t}[A^j(s)+t-s-Nk^2-N/k(k-1)]\quad(1)$$

我们现在比较在相同的到达模式下负载均衡交换结构与 OQ 交换结构的性能. OQ 交换结构中目的端口为 j 的信元行为可以建模为到达过程为 $A^j(t)$，服务时间间隔为常量 1 的工作保持服务. 假设 $C^j_{OQ}(t)$ 为这种服务的服务次数，那么根据引理 5，我们得到

$$C^j_{OQ}(t)=\min_{0\leq s\leq t}[A^j(s)+t-s]\quad(2)$$

比较式(1)与式(2)得到

$$C^j(t)\geq\min_{0\leq s\leq t}[A^j(s)+t-s]-Nk^2-N/k(k-1)$$
$$=C^j_{OQ}(t)-Nk^2-N/k(k-1).$$

因此，UFFS-k 算法服务信元的数目和 OQ 交换结构服务信元的数目仅相差常量 $Nk^2+N/k(k-1)$. 这意味着 UFFS-k 算法总是和 OQ 交换结构具有相同的吞吐率，并与输入流量的到达过程无关.　证毕.

4 UFFS-k 算法性能评测

目前一般采用软件模拟的方法对交换结构及其调度算法进行性能评估.我们对 Stanford 大学开发的 SIM 模拟器①,进行了修改和扩充,构建了一个输入端采用 *VOQ* 队列模型,中间级采用 *OQ* 队列模型的负载均衡交换结构(load-balanced switch).我们在贝努利一致流量模型和突发流量模型下,比较了 FOFF[3]、UFS[3]、UFFS-k 和基本的负载均衡算法 BLA[2](Basic Load-balanced Algorithm)的延迟性能,观察了聚合粒度 $k=8,4,2,1$ 时 UFFS-k 算法延迟性能的变化.

4.1 模拟环境

SIM 模拟器提供多种流量模型,我们采用了两种具有代表性的流量模型:(1)贝努利一致流.服从贝努利到达过程,独立同分布,目的端口均匀分布于所有的输出端口;(2)突发流.突发长度为 10,在忙-闲周期(busy-idle periods)突发信元,目的端口以连续突发或者一个信元接一个信元的方式分布于所有的输出端口.由于 Internet 流量具有突发特性,突发流量模型更接近真实的网络流量.

在所有的模拟实验中,模拟时间为 200 000 个外部时间槽;实验参数为:端口数目 $N=16$,内部链路加速比 $S=1$.所有队列长度为无穷大,即不限制队列长度,不丢弃信元.4.3 节在贝努利一致流量模型和突发流量模型下,模拟比较了 FOFF、UFS、UFFS-k 和 BLA 4 种算法的延迟性能,分析了 UFFS-k 算法在不同的聚合粒度下的延迟特性.

4.2 算法建模

我们在负载均衡交换结构中实现了 FOFF、UFS、UFFS-k 和 BLA 4 种调度算法.对于 FOFF 算法,输入线卡采用 *VOQ* 队列缓冲到达信元;中间级输入线卡采用 *OQ* 队列模型;每个输出线卡维护 N 个深度为 N 个信元的缓冲队列,用于重定序乱序报文.在输入端,满帧被赋予最高优先级,只要存在满帧则调度满帧,若不存在满帧则以 round-robin 的方式调度其它非空 *VOQ* 队列.为了限定负载均衡路由器内乱序信元的数量,FOFF 算法严格按照 round-robin 的顺序发送 *VOQ* 队列的信元到中间级输入线卡.因此,每 N 个外部时间槽 FOFF 只能服务一个 *VOQ* 队列,当被服务 *VOQ* 队列的信元不能占用全部的内部链路时(所含信元小于 N 的情况),将造成内部链路带宽的浪费.FOFF 算法在输入端保证了每条流的信元以 round-robin 的方式分布于中间级输入线卡.在输出端,FOFF 维护 N 个 round-robin 指针指示每条流的下一个信元所在的中间级输入线卡,来自该中间级输入线卡的信元就是下一个信元.因此,FOFF 算法能够在没有任通信开销的情况下实现信元的按序发送,但输出端 $O(N^2)$ 的重定序缓冲区开销加大了信元延迟.建模 UFS 算法为聚合粒度等于端口数目 N 的 UFFS-k 算法,在这种极端情况下已经不存在流到区域的映射关系.建模 UFFS-k 算法为:输入端采用 *VOQ* 队列模型,按照 3.3 节所描述的算法建模 UFFS-k 算法,UFFS-k 算法分布在每个输入端独立执行;中间级输入线卡采用 *OQ* 队列模型.在没有特殊声明的情况下,所有队列长度均为无穷大.对于基本的负载均衡调度算法 BLA,我们按照文献[2]所执行的调度算法建模 BLA 算法.输入端以固定轮转的方式将到达信元通过第一级 Mesh 网络发送到中间级输入线卡.由于连接模式的周期性,信元按照其到达时间均匀分布在中间级输入线卡,因此到达中间级输入线卡的流量是负载均衡的.中间级输入线卡采用 *VOQ* 队列模型,按照信元的目的端口缓冲信元,同样以固定轮转的方式将 *VOQ* 队列中的信元通过第二级 Mesh 网络发送到输出端口.文献[2]证明了,BLA 算法能够获得 100% 的吞吐率,在重负载的情况下,信元平均延迟接近 *OQ* 交换结构.BLA 算法不能保证报文的顺序,存在严重的报文乱序现象.

4.3 延迟实验

我们分别在贝努利一致流量模型和突发流量模型下分析比较了 FOFF、UFS、UFFS-k 和 BLA 4 种算法的信元平均延迟.图 4 显示了在贝努利一致流量模型下 UFS 算法的系统平均延迟远远大于其它算法,这是因为 UFS 需要在输入端聚合 N 个信元才能实施一次调度.通过不断降低聚合粒度 k,UFFS-k 算法的延迟性能得到了显著提升.在极端情况下,当聚合粒度为 1 时,UFFS-1 算法在低负载情况下具有与 BLA 算法相当的延迟性能,但在高负载情况下变得不稳定,当负载大于 0.7 时,UFFS-1 算法系统平均延迟高于 UFFS-2.这可能是因为聚合粒度为 1 时,UFFS-1 算法总是将流发送到固定的中间级输入线卡,这种每条流专享一条链路的调度方式可能会造成内部链路带宽的严重浪费更不利于负载均衡.对于 FOFF 算法,正如第 2 节所指出

① http://klamath.stanford.edu/tools/SIM/

的，内部链路带宽浪费在某种程度上降低了系统吞吐率，此外大部分的信元延迟发生在输出端的重定序操作上（重定序缓冲区开销为 $O(N^2)$）. 当输入负载大于 0.2 时，UFFS-4 具有比 FOFF 更低的信元平均延迟；而 UFFS-2 的延迟性能明显优于 FOFF. BLA 算法延迟性能略优于 UFFS-2 算法，但 BLA 算法存在严重的信元乱序问题，从而损害 TCP 协议的性能.

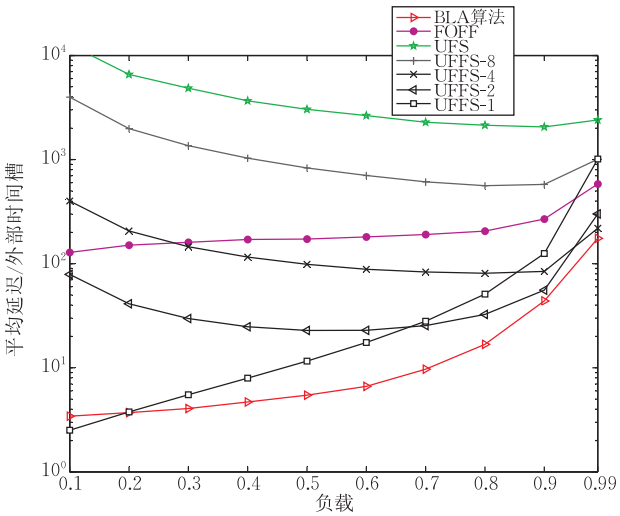

图 4　FOFF、UFS、UFFS-k 及 BLA 算法平均延迟（贝努利一致流量模型）

图 5 显示了在突发流量模型下的模拟结果. 随着聚合粒度的不断降低，UFFS-k 算法延迟性能获得的提升越少. 在极端情况下，当聚合粒度 $k=1$ 时，UFFS-1 系统平均延迟大于 UFFS-2；UFFS-1 算法在突发流量模型下变得更不稳定；当输入负载达到 0.99 时，UFFS-1 算法的平均延迟最大，甚至超过了 UFS 算法. UFFS-1 算法发送每条流到固定的中间级输入线卡，在突发情况下，UFFS-1 算法为实现负载均衡必将导致信元被延迟调度. 当输入负载大于 0.3 时，UFFS-4 算法延迟性能优于 FOFF 算法. UFFS-2 算法在所有能够保证报文顺序的算法中具有最优延迟性能，并与不具备报文保序特性的 BLA 算法性能相当.

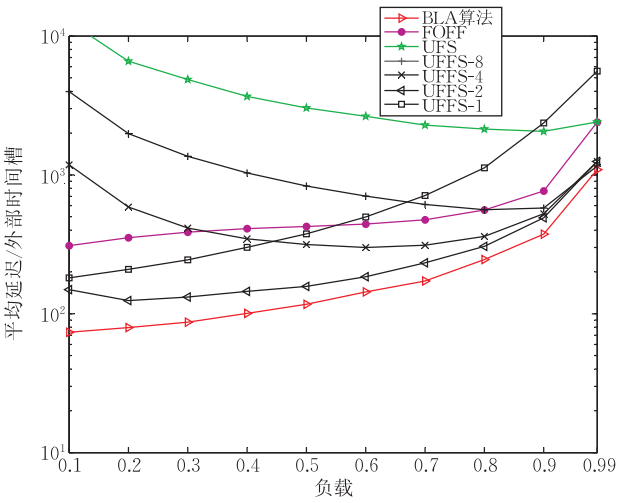

图 5　FOFF、UFS、UFFS-k 及 BLA 算法平均延迟（突发流量模型）

5　结　论

采用固定连接模式的负载均衡交换结构成为提高路由器性能、规模、可扩展性的有效途径. 负载均衡交换结构存在严重的报文乱序现象，乱序报文会降低 TCP 连接吞吐率增加报文延迟，在路由器中保证报文的顺序是极其必要的. 另一方面，为了保证报文的顺序，路由器需要完成大量额外的工作，这会影响路由器的性能. 本文提出一种基于流映射的负载调度算法 UFFS-k，采用双循环方式构建流到区域的映射，以细粒度的方式将同一条流的 k 个信元分派到固定的映射区域，通过理论证明，该调度策略可获得 100% 吞吐率并能够保证报文的顺序. 本文通过模拟验证 UFFS-k 算法在不同聚合粒度 k 下的延迟性能，并与目前主流的负载均衡调度算法进行比较. 模拟结果显示，当聚合粒度 $k=2$ 时，UFFS-2 算法在所有能够保证报文顺序的调度算法中具有最优延迟性能；并在突发流量模型下，表现出与不具备报文保序特性的 BLA 算法相当的性能；因此 UFFS-2 算法能够在保证报文顺序的同时提供延迟和吞吐率保证.

参　考　文　献

[1] Minkenberg C et al. Current issues in packet switch design. ACM SIGCOMM Computer Communications Review, 2003, 33(1):

[2] Chang C S, Lee D S, Jou Y S. Load balanced Birkhoff-Von Neumann switches part I: One-stage buffering. Computer Communications, 2002, 25(6): 611-622

[3] Keslassy I. The load-balanced router [Ph. D. dissertation]. Stanford University, Stanford, 2004

[4] Keslassy I, Chuang S T, Yu K, Miller D, Horowitz M, Solgaard O, McKeown N. Scaling Internet routers using optics//Proceedings of the ACM SIGCOMM. Karlsruhe, Germany, 2003

[5] Chang C S, Lee D S, Shih Y J. Mailbox switch: A scalable two-stage switch architecture for conflict resolution of ordered packets//Proceedings of the IEEE INFOCOM. Miami, FL, 2004

[6] Bennett J, Partridge C, Shectman N. Packet reordering is not pathological network behavior. IEEE/ACM Transactions on Networking, 1999, 7(6): 789-798

[7] Chang C S, Lee D S, Lien C M. Load balanced Birkhoff-Von Neumann switches part Ⅱ: Multi-stage buffering. Computer Communications, 2002, 25(6): 623-634

[8] Chang C S, Lee D S, Shih Y J. Mailbox switch: A scalable two-stage switch architecture for conflict resolution of ordered packets//Proceedings of the IEEE INFOCOM. Miami, FL, 2004

[9] Lin B, Keslassy I. The concurrent matching switch architecture//Proceedings of the 25th IEEE INFOCOM. Barcelona, 2006

[10] Keslassy I, McKeown N. Maintaining packet order in two-stage switches//Proceedings of the IEEE Infocom'02, New York, NY, 2002

[11] Lin B, Keslassy I. A scalable switch for service guarantees//Proceedings of the 13th IEEE Symposium on High-Performance Interconnects, Stanford, CA, USA, 2005

[12] Chang C S, Chen W J, Huang H Y. On service guarantees for input buffered crossbar switches: A capacity decomposition approach by birkhoff and von neumann//Proceedings of the IEEE IWQoS'99. London, UK, 1999: 79-86

[13] Kanizo Y. The crosspoint-queued switch//Proceedings of the 28th IEEE INFOCOM. Rio de Janeiro, Brazil, 2009

[14] Dai Yi, Su Jin-Shu, Sun Zhi-Gang, Guan Jian-Bo. A uniform fine-grain frame spreading algorithm for avoiding packet reordering in load-balanced switches//Proceedings of the IEEE APSCC. Tsukuba, Japan, 2007

[15] Iyer S, McKeown N. Making parallel packet switches practical//Proceedings of the IEEE INFOCOM. Anchorage, Alaska, USA, 2001: 1680-1687

[16] Iyer S, Awadallah A, McKeown N. Analysis of a packet switch with memories running slower than the line rate//Proceedings of the IEEE INFOCOM. Tel Aviv, Israel, 2000: 529-537

DAI Yi, born in 1980, Ph. D., lecturer. Her main research interests include router architectures, high performance packet switches and switch scheduling algorithms.

SU Jin-Shu, born in 1962, Ph. D., professor, Ph. D. supervisor. His main research interests include computer network architectures, information security.

SUN Zhi-Gang, born in 1973, Ph. D., professor. His main research interests include computer network architectures and communications technology, high performance packet switch architectures.

Background

Continuing growth in traffic, the size of Internet as well as the Internet applications puts forward great challenge to backbone router design. The load-balanced switch architectures without dynamic switch configurations for cell forwarding appear to be an effective way to scale Internet routers to very high capacities. However, the load-balanced router architecture has the critical problem that packet departures can be badly mis-sequenced. The approaches enforcing packet reordering either rely on complex, centralized schedulers, or require reordering buffers of size $O(N^2)$, and the corresponding quadratic increase in packet delays, where N is the switch size. In this paper, we introduce a novel Uniform Fine-grain Frame Spreading (UFFS-k) algorithm for the load-balanced router that can guarantees packet ordering by spreading cells of the same flow to the fixed region in a fine-grained way, thus eliminating cell reassembly overhead at output. The UFFS-k algorithm is distributed and can operate independently in each input. To avoid traffic concentration in regions, a dual-rotation mapping algorithm is used to construct a mapping relationship between flows from different inputs and regions. It is theoretically proofed that 100% throughput and packet ordering can be achieved by using UFFS-k algorithm at each input. In addition, by maintaining a global traffic distribution matrix at each input 100% load balancing degree can be achieved. As the simulation results demonstrate, when $k=2$, UFFS-k offers improved delay performance among existing scheduling algorithms that guarantee packet ordering.

This work is supported by the National Natural Science Foundation of China project "Research on Key Technologies of Router Architecture Based on FIS" under grant No. 61003301 and in part by the National High-Tech Research and Development Plan of China under grant No. 2008AA01A325. Today, the capacity and port density of the switch can hardly keep up with the growth of traffic resulting from increasing link speeds and the number of hosts on the Internet. On the other hand, IP-lookup performance in core routers can hardly keep up with the growth of FIB size resulting from widely used multi-homed technology and increased Internet size. A router logically consists of two consecutive stages namely forwarding stage and switching stage. These two processing stages are implemented in different hardware components and performed in order, which hinders parallelism development inside routers. We originally propose a distinct packet processing mechanism FIS (Forwarding In Switch) and make comprehensive research on the key technologies of FIS mechanism including the router architecture, partition and logical mapping of routing table, IPv6 forwarding mechanism, as well as the self-sufficient switch mechanism. This work focuses on distributed switch and load-balanced scheduling algorithm. By avoiding packet reordering in load-balanced switches this zero-communication scheduling algorithm is obviously feasible to implement in high-speed routers.