基于逆向传输机制的反馈型两级交换结构

申志军*[①] 高 静[①②] 乌日更[①]
[①](内蒙古农业大学计算机与信息工程学院 呼和浩特 010018)
[②](内蒙古农业大学信息与网络中心 呼和浩特 010018)

摘 要：为解决 FTSA-2-SS 结构中的信元冲突、信元失序以及交换流程复杂化等问题，该文提出一种基于逆向传输机制的反馈型两级交换结构(FRTM-TSA)。该结构通过 crossbar 逆向传输机制使得任意输入端口均可获得其相邻端口的调度结果并以此对目标端口所反馈的缓存信息进行修正，基于修正后的信息进行算法调度使得 FRTM-TSA 能够避免信元冲突和信元失序，也无需在输出端口设置重排序缓存。理论分析和仿真结果均表明 FRTM-TSA 能够以相对简洁的交换结构和交换流程获得更优的时延性能。

关键词：调度算法；交换；反馈；负载均衡
中图分类号： TP393 **文献标识码**： A **文章编号**：1009-5896(2018)03-0697-08
DOI: 10.11999/JEIT170531

Feedback and Reverse Transmission Mechanism Based Two-stage Switch Architecture

SHEN Zhijun[①] GAO Jing[①②] WU Rigeng[①]
[①](*College of Computer and Information Engineering*, *Inner Mongolia Agricultural University*, *Hohhot* 010018, *China*)
[②](*Center of Information and Network Technology*, *Inner Mongolia Agricultural University*, *Hohhot* 010018, *China*)

Abstract: In order to solve the problems arising from the 2-Staggered Symmetry connection pattern (2-SS) in Feedback mechanism based load balanced Two-stage Switch Architecture (FTSA), a Feedback and Reverse Transmission Mechanism based Two-stage Switch Architecture (FRTM-TSA) is proposed in this paper. A novel reverse transmission mechanism of crossbar is introduced so that any input port can obtain the scheduling results of its adjacent input port. Based on such scheduling results, the buffer status information of middle-ports that received one slot ahead can be corrected. The exact information obtained from preprocessing enables FRTM-TSA to avoid the cell-conflict and cell-disordering and thus make the re-sequencing buffers are no longer needed at the output ports. Theoretical analysis and simulation experiments show that FRTM-TSA can achieve a better delay performance with a simper switching fabric and process compared to existing schemes.

Key words: Scheduling algorithm; Switching; Feedback; Load balancing

1 引言

Internet 数据流具有明显不同于传统电信网络数据流量的特性，即自相似特性，其主要特征是数据包(信元)一簇一簇地以突发的形式到达网络设备输入端。这种突发到达的数据流量对网络设备的转发性能提出了新的挑战。传统的交换结构，如应用最为广泛的输入排队[1,2](Input Queuing, IQ)+iSLIP[3]等，在自相似数据流环境中，其时延性能随负载率上升会迅速恶化。为更好地适应 Internet 自相似数据流环境，文献[4–6]提出负载均衡交换结构(Load Balanced Birkhoff-von Neumann switch architecture, LB-BvN)。LB-BvN 具有两级 crossbar，其第 1 级 crossbar 能将到达输入端的突发数据流均匀散布到第 2 级 crossbar，因此能够较好地适应自相似数据流环境。但由于信元需要在交换结构内经过多级 crossbar 和多级缓存后才能完成传输和转发，LB-BvN 在时延性能方面一直不够理想，同时还存在信元可能会在输出端失序这一致命问题。业界针对这一问题做了大量研究工作：满帧优

收稿日期：2017-06-20；改回日期：2017-10-27；网络出版：2017-11-23
*通信作者：申志军 shensljx@sina.com
基金项目：内蒙古农业大学优秀青年科学基金(2014XYQ-17)，国家自然科学基金(61650204, 61462070)，内蒙古农业大学博士科研启动基金(BJ2013B-1)
Foundation Items: The Excellent Young Scientist Foundation of Inner Mongolia Agricultural University of China (2014XYQ-17), The National Natural Science Foundation of China (61650204, 61462070), The Doctoral Scientific Research Foundation of Inner Mongolia Agricultural University of China (BJ2013B-1)

先方案[7](Full Frame First, FFF)需要在线卡之间为寻找满帧而进行大量的通信。有序满帧优先方案[8](Full Ordered Frames First, FOFF)同样基于帧进行转发，但若算法选择非满帧将会造成 N 个时隙的带宽浪费，因此在负载率较低或 N 较大时时延性能较差。Mailbox[9]的吞吐率最高只能达到 75%，其改进方案最高也只能达到 95%。CR switch[10]在负载率较低时性能较好，在负载率高于 60%后时延性能明显恶化。Byte-Focal[11]利用信元的转发路径来完成信元的重排序，但其算法复杂度为 $O(N)$。更为关键的是以上方案的时延性能均不够理想。

反馈型两级交换结构[12](Feedback mechanism based load balanced Two-stage Switch Architecture, FTSA) 由两级 crossbar(分别记为 X_1 和 X_2)和两级缓存组成。其中位于 X_1 之前的缓存称为输入缓存，记为 VOQ1，位于 X_1 和 X_2 之间的缓存称为中间缓存，记为 VOQ2，VOQ1(i,k)用于缓存到达输入端口 i 且目标端口为输出端口 k 的信元，任意 VOQ2(j,k) 仅设置一个信元的缓存空间，用于缓存到达中间端口 j 且目标端口为输出端口 k 的信元。FTSA 采用错列对称连接模式，该连接模式的特征在于若某时隙中间端口 j 与输出端口 k 相连，则下一时隙必有输入端口 k 与中间端口 j 相连。这样中间端口 j 的缓存状态信息可方便地通过输出端口 k 反馈至位于同一线卡的输入端口 k，而下一时隙恰有输入端口 k 与中间端口 j 相连，故输入端口 k 可在已知目标缓存状态信息的前提下进行转发调度，从而在保证信元不失序的前提下获得了极为优异的时延性能。以此为基础的相关工作，如三级交换结构[13]、多播[14]、扩展性[15]和联合对称连接模式[16]等得到了迅速发展和充分研究。

虽然 FTSA 理论性能优异，但其对调度算法执行时间的限制是极为苛刻的，即要求复杂度为 $O(N)$ 的调度算法必须在 crossbar 重配置时间内完成，考虑到 FTSA 中两级 crossbar 均采用周期性连接的模式，即每时隙 crossbar 输入端与输出端的连接配置是确定的，故其重配置时间仅仅取决于电子元器件的开关速度。这使得 FTSA 优异的理论性能在当前微电子技术条件下是无法实现的。

2 解决算法执行时间问题的思路

目前解决这一问题的思路有两种：第 1 种是尽可能降低算法复杂度和算法耗时；第 2 种是尽可能为算法提供更长的执行时间。

作为第 1 种思路的解决方案，文献[12]提出使用 $O(1)$ 复杂度的准最长队列优先[17](Quasi-Longest Queue First, Quasi-LQF)算法替代复杂度为 $O(N)$ 的 LQF 算法，然而本文经分析发现，用于 FTSA 的 LQF 算法并非简单地从 N 个队列中选取最长的队列，而是每次调度均需事先根据中间端口的反馈信息构造一个集合，然后从集合中选择最长的队列，而 FTSA 中与某个输入端口相连的中间端口是不断变化的，即作为 LQF 算法的基础数据，该集合是每时隙都在变化的且和上一时隙的集合没有直接关联，故 Quasi-LQF 并不适用于 FTSA。

作为第 2 种思路的解决方案，使用 2-错列对称连接模式(如图 1 所示)的两级交换结构[18] (Feedback-based Two-stage Switch Architecture using 2-Staggered Symmetry connection pattern, FTSA-2-SS), FTSA-2-SS 采用两级 crossbar 和 3 级缓存，如图 2 所示。位于 X_1 之前的输入缓存 VOQ1 和位于 X_1 和 X_2 之间的中间缓存 VOQ2 均类似于 FTSA, FTSA-2-SS 的不同之处在于：

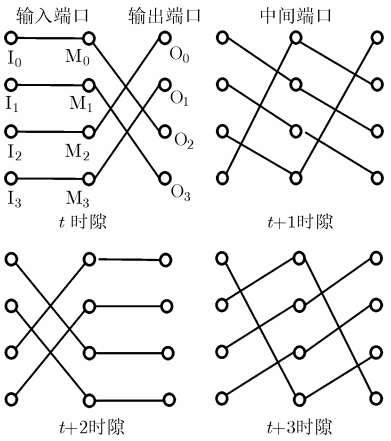

图 1 2-错列对称连接模式

(1)两级 crossbar 采用一种 2-错列对称的连接模式；

(2)任意 VOQ2(j,k) 需设置两个信元的缓存空间；

(3)在每个输出端口需设置重排序缓存纠正信元离开交换机的顺序。

2-错列对称连接模式的特征在于若 t 时隙中间端口 j 与输出端口 k 相连，则 $t+2$ 时隙必有输入端口 k 与中间端口 j 相连。这样中间端口 j 的缓存状态信息反馈至输入端口后可使其提前一个时隙进行算法调度，从而将算法的执行时间扩展到接近一个时隙的时长。然而算法真正需要的是一个时隙之后的目标缓存状态信息，基于提前反馈的信息进行调度会因信息失实而导致一个非空的目标缓存队列在调度时被误认为是空的，这种情况会使得调度算法所

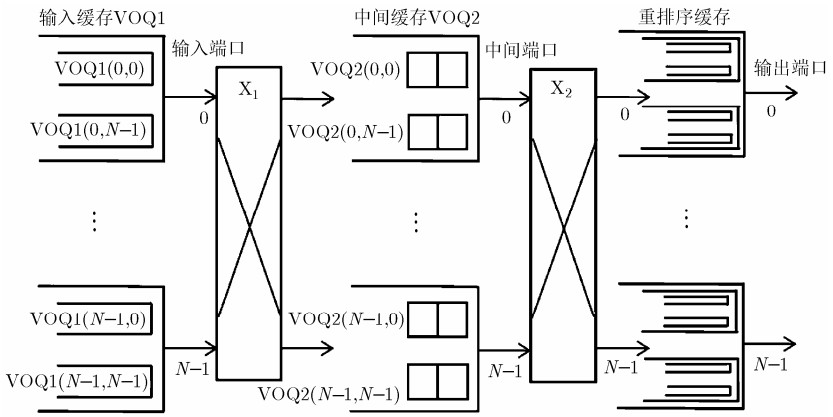

图 2　FTSA-2-SS 交换结构

选择的信元在被转发到中间端口后因目标缓存队列已被另一个信元占据而出现信元冲突问题。为此，FTSA-2-SS 不得不为任意 $\mathrm{VOQ2}(j,k)$ 设置两个信元的缓存空间来解决信元冲突问题，然而这一措施又造成同一个流的信元在中间缓存等待的时间因缓存位置的不同而不同，破坏了 FTSA 原本的保证信元不失序的保障机制。为避免信元在转发过程中失序，FTSA-2-SS 在输出端增加重排序缓存来纠正信元离开交换机的顺序。可以看出，FTSA-2-SS 虽然缓解了对调度算法的时间限制问题，但同时也付出了交换流程复杂化和时延性能两方面的代价。

针对这一现状，本文提出一种基于逆向传输机制的反馈型两级交换结构(Feedback and Reverse Transmission Mechanism based Two-stage Switch Architecture, FRTM-TSA)，FRTM-TSA 通过反馈和 crossbar 逆向传输机制，利用邻端口的调度结果修正提前反馈的中间端口缓存信息，以获得准确的目标缓存状态信息进行算法调度，避免信元冲突和失序，无需在中间端口处设置两个信元的缓存空间，也无需在输出设置重排序缓存，同时能够获得更高的时延性能。

为便于描述，本文做以下约定：

(1)交换结构的输入/输出端口数均为 N，两级 crossbar 分别记为 X_1 和 X_2；在不引起混淆的情况下，“输入端口”均指 X_1 的输入端口，“中间端口”均指 X_2 的输入端口，“输出端口”指 X_2 的输出端口；

(2)序号为 i 的输入端口记为 I_i，序号为 j 的中间端口记为 M_j，序号为 k 的输出端口记为 O_k；

(3)定义到达 I_i 且目的输出端口为 O_k 的信元集合为数据流 $\mathrm{F}_{i,k}$，记 $\mathrm{C}_{i,k}$ 泛指 $\mathrm{F}_{i,k}$ 的任意一个信元；

(4)位于 X_1 之前的缓存记为 $\mathrm{VOQ1}$，$\mathrm{VOQ1}(i,k)$ 用于缓存 $\mathrm{C}_{i,k}$；位于 X_1 和 X_2 之间的缓存记为 $\mathrm{VOQ2}$，$\mathrm{VOQ2}(j,k)$ 用于缓存到达 M_j 且目的输出端口为 O_k 的信元；

(5) I_i 与 M_j 相连记为 $\mathrm{I}_i \to \mathrm{M}_j$，$\mathrm{M}_j$ 与 O_k 相连记为 $\mathrm{M}_j \to \mathrm{O}_k$，$\mathrm{I}_i$ 通过 M_j 与 O_k 相连简记为 $\mathrm{I}_i \to \mathrm{M}_j \to \mathrm{O}_k$；

(6) M_j 在 t 时隙结束时刻的缓存队列信息记为 $\mathrm{Q}_j(t)$，$\mathrm{Q}_j(t)$ 仅有 N 个 bit，若其第 v 位为“1”，则表示 $\mathrm{VOQ2}(j,v)$ 非空，反之表示 $\mathrm{VOQ2}(j,v)$ 为空；

(7) I_i 在 t 时隙的调度结果信息记为 $\mathrm{S}_i(t)$，$\mathrm{S}_i(t)$ 仅有 N 个 bit，若其第 v 位为“1”，则表示将在下一时隙转发 $\mathrm{VOQ1}(i,v)$ 中的信元，若 $\mathrm{S}_i(t)==0$ 表示调度算法在本时隙未选择任何信元；

(8) I_i 在 t 时隙的本地缓存队列信息记为 $\mathrm{L}_i(t)$，$\mathrm{L}_i(t)[v]$ 表示 $\mathrm{VOQ1}(i,v)$ 的缓存队列长度；

(9)端口号的加减操作都要对 N 取模，即 $i-1$ 实质上是 $(i-1) \bmod N$。

3　基于逆向传输机制的反馈型两级交换结构

3.1 2-错列对称链接模式与提前反馈

FRTM-TSA 具有两级 crossbar 和两级缓存，其基本结构与 FTSA 相同。其两级 crossbar 所采用的连接模式和 FTSA-2-SS 相同，均为 2-错列对称连接模式，如图 1 所示。

FRTM-TSA 中采用 2-错列对称连接模式，其关键特征在于，若 t 时隙有 $\mathrm{M}_j \to \mathrm{O}_k$，则 $t+2$ 时隙必有 $\mathrm{I}_i \to \mathrm{M}_j$。这一特征可使输入端口提前一个时隙获得目标端口的缓存状态信息(虽然是不完备的)，从而为提前进行算法调度提供了可能。

由文献[18]可知，图 1 所示的 2-错列对称连接模式由式(1)-式(4)确定：

(1)与 I_i 相连的中间端口的序号 j 由式(1)确定：

$$j=(i+t) \bmod N,\quad i=0,1,\cdots,N-1 \tag{1}$$

(2)与 M_j 相连的输入端口的序号 i 由式(2)确

定：

$$i=(j-t)\bmod N,\quad j=0,1,\cdots,N-1 \tag{2}$$

(3)与 M_j 相连的输出端口的序号 k 由式(3)确定：

$$k=(j-2-t)\bmod N,\quad j=0,1,\cdots,N-1 \tag{3}$$

(4)2-错列对称连接模式具有如式(4)所确定的锚定特性：

$$k=(i-2)\bmod N,\quad i=0,1,\cdots,N-1 \tag{4}$$

式(4)表明对于一个确定的输入端口，无论在哪个时隙，该端口总通过不同的中间端口与固定的输出端口相连，如对于任意输入端口 I_i，与其相连的输出端口总是 O_{i-2}。

综上所述，若 t 时隙有 $\mathrm{I}_i\to\mathrm{M}_j\to\mathrm{O}_{i-2}$，则必有：

(1) t 时隙同时必有 $\mathrm{I}_{i-1}\to\mathrm{M}_{j-1}\to\mathrm{O}_{i-3}$；$\mathrm{I}_{i-2}\to\mathrm{M}_{j-2}\to\mathrm{O}_{i-4}$；

(2) $t+1$ 时隙必有 $\mathrm{I}_i\to\mathrm{M}_{j+1}\to\mathrm{O}_{i-2}$；$\mathrm{I}_{i-1}\to\mathrm{M}_j\to\mathrm{O}_{i-3}$；$\mathrm{I}_{i-2}\to\mathrm{M}_{j-1}\to\mathrm{O}_{i-4}$；

(3) $t+2$ 时隙必有 $\mathrm{I}_i\to\mathrm{M}_{j+1}\to\mathrm{O}_{i-2}$；$\mathrm{I}_{i-1}\to\mathrm{M}_{j+1}\to\mathrm{O}_{i-3}$；$\mathrm{I}_{i-2}\to\mathrm{M}_j\to\mathrm{O}_{i-4}$。

因此，引入 2-错列对称连接模式可使得 FRTM-TSA：

(1) M_j 可在 t 时隙结束时刻将其缓存队列状态信息 $\mathrm{Q}_j(t)$ 传输至 O_{i-2}。随后 O_{i-2} 将 $\mathrm{Q}_j(t)$ 传输至位于同一线卡的 I_{i-2}。

(2) $t+1$ 时隙，I_{i-2} 进行算法调度选择在下一时隙将传输至 M_j 的信元。

(3) $t+2$ 时隙有 $\mathrm{I}_{i-2}\to\mathrm{M}_j$，依据上个时隙的调度结果将算法所选择的信元转发至 M_j。

显然，$t+2$ 时隙时 I_{i-2} 将转发信元到 M_j，而 $\mathrm{Q}_j(t)$ 在 t 时隙结束时刻即可反馈至 I_{i-2}。这样就相当于 I_{i-2} 提前一个时隙得到了目标端口缓存队列信息(信息是不完备的)，使得提前进行算法调度成为可能。

3.2 $\mathrm{Q}_j(t)$ 的非完备特性和预处理

由图 1 和 3.1 节分析可知，I_{i-2} 在 $t+1$ 时隙之初收到的反馈信息 $\mathrm{Q}_j(t)$ 仅仅是 M_j 在 t 时隙结束时的缓存队列状态信息，然而 I_{i-2} 在 $t+1$ 时隙所进行的算法调度需要的是 $\mathrm{Q}_j(t+1)$。因此 FRTM-TSA 采取对 $\mathrm{Q}_j(t)$ 预处理的方法使其尽可能接近 $\mathrm{Q}_j(t+1)$。

考虑到一个时隙最多只能有一个信元到达和离开，故对于 $\mathrm{Q}_j(t)$，$\mathrm{Q}_j(t+1)$ 至多可能有两种情况的改变：

(1) $t+1$ 时隙，某个信元到达 M_j；

(2) $t+1$ 时隙，某个信元因被转发而离开 M_j。

对于第(1)种情况，其发生与否于 M_j 而言是无法预知的。

对于第(2)种情况，由于 X_2 的连接模式是周期性的，即在 $t+1$ 时隙信元因被转发而离开 M_j 的信息是可预知的，FRTM-TSA 对 $\mathrm{Q}_j(t)$ 预处理的方法如下：

$$v=(j+t-3)\bmod N$$
$$\mathrm{Q}_j(t)\,\&=\sim(1<<v)$$

v 表示 $t+1$ 时隙与 M_j 相连的输出端口的序号，即若 $\mathrm{VOQ2}(j,v)$ 非空，则在 $t+1$ 时隙，其队列中的信元必将被转发而离开 M_j。

预处理之后的 $\mathrm{Q}_j(t)$ 与 $\mathrm{Q}_j(t+1)$ 至多有 1 个 bit 不同。但若据此进行算法调度，则可能出现信元的目标缓存队列非空的情况，因 FRTM-TSA 仅为任意 $\mathrm{VOQ2}(j,k)$ 设置一个信元的缓存空间，这就意味着信元冲突，必须选择一个信元丢弃，其结果不仅会浪费传输带宽，还会造成信元在交换机内部转发过程中失序。为此 FRTM-TSA 还需利用邻端口的调度结果对 $\mathrm{Q}_j(t)$ 进一步进行修正。

3.3 基于 crossbar 的逆向传输机制

由 3.2 节中分析可知，$t+1$ 时隙必有 $\mathrm{I}_{i-1}\to\mathrm{M}_j$，即在 $t+1$ 时隙是否有信元到达 M_j(即 3.2 节所述之第 1 种情况)取决于 t 时隙结束时刻 I_{i-1} 的调度结果 $\mathrm{S}_{i-1}(t)$。

基于这一特性，FRTM-TSA 采用如图 3 所示的 crossbar 逆向传输机制。不失一般性，以 I_{i-2} 通过该机制接收 $\mathrm{S}_{i-1}(t)$ 为例说明其工作原理和过程。

如图 3(a)所示，t_0 时隙结束前时刻，I_{i-1} 在调度结束后将 $\mathrm{S}_{i-1}(t)$ 挂在信元之后传输至 M_{j-1}。注意到，与此同时，M_j 将 $\mathrm{Q}_j(t)$ 挂在信元之后传输至 O_{i-2}。随后 O_{i-2} 将 $\mathrm{Q}_j(t)$ 传输至位于同一线卡的 I_{i-2}。

如图 3(b)所示，t_0+1 时隙有 $\mathrm{I}_{i-2}\to\mathrm{M}_{j-1}$，故 FRTM-TSA 在时隙之初将 M_{j-1} 处的 $\mathrm{S}_{i-1}(t)$ 通过既有的 crossbar 连接逆向传输至 I_{i-2}。I_{i-2} 收到 $\mathrm{S}_{i-1}(t)$ 之后对 $\mathrm{Q}_j(t)$ 进行修正使之完全等同于 $\mathrm{Q}_j(t+1)$。随后开始进行算法调度。

如图 3(c)所示，t_0+2 时隙恰有 $\mathrm{I}_{i-2}\to\mathrm{M}_j$，$\mathrm{I}_{i-2}$ 基于上一时隙的调度结果将信元转发至 M_j。

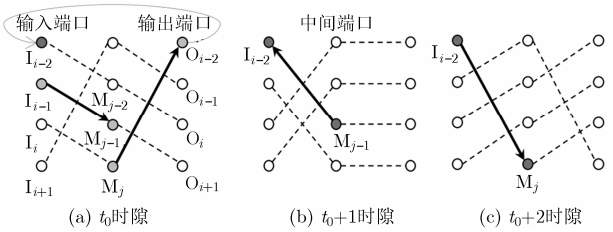

图 3 FRTM-TSA 的逆向传输机制

3.4 对反馈信息的再处理

若 I_{i-1} 在 t 时隙的调度算法选择在下一时隙转发 $\mathrm{VOQ1}(i-1,v)$ 的队首信元则：$\mathrm{S}_{i-1}(t)=1<<v$；否则 $\mathrm{S}_{i-1}(t)=0$；

I_{i-2} 在时隙 $t+1$ 之初收到 $\mathrm{Q}_j(t)$ 和 $\mathrm{S}_{i-1}(t)$ 后需要利用 $\mathrm{S}_{i-1}(t)$ 对 $\mathrm{Q}_j(t)$ 进行修正。修正后的信息记为 $\mathrm{MQ}_j(t)$ (Modified $\mathrm{Q}_j(t)$)，则

$$\mathrm{MQ}_j(t)=\mathrm{Q}_j(t)\big|\mathrm{S}_{i-1}(t)$$

$\mathrm{MQ}_j(t)$ 中第 v 个比特为 0 表示 $\mathrm{VOQ2}(j,v)$ 队列为空。

3.5 本地缓存状态信息的预处理

考虑到 $t+1$ 时隙 I_{i-2} 的调度算法的目的是选择下一时隙要转发的信元，因此算法所依赖的数据除 $\mathrm{MQ}_j(t)$ 之外还包括本地缓存的状态信息，特别地，应该基于 $\mathrm{MQ}_j(t)$ 和 $t+1$ 时隙结束时刻的本地缓存状态信息 $\mathrm{L}_{i-2}(t+1)$ 进行调度，然而算法在 $t+1$ 时隙之初就开始进行，因此 FRTM-TSA 采用对当前时刻的本地缓存状态信息 $\mathrm{L}_{i-2}(t)$ 进行预处理的方法使其尽可能接近 $\mathrm{L}_{i-2}(t+1)$，从而提高调度算法的效能，具体分析和预处理方法如下：

一个时隙最多只能有一个信元到达和离开 I_{i-2}，故对于 $\mathrm{L}_{i-2}(t),\mathrm{L}_{i-2}(t+1)$ 至多可能会有两种情况的改变：

(1) $t+1$ 时隙，某个信元到达 I_{i-2}。

(2) $t+1$ 时隙，某个信元因被转发而离开 I_{i-2}。

对于第(1)种情况，其发生与否于 I_{i-2} 而言同样是无法预知的。

对于第(2)种情况，即在 $t+1$ 时隙信元因被转发而离开 I_{i-2} 的信息取决于 I_{i-2} 在 t 时隙的调度结果 $\mathrm{S}_{i-2}(t)$，若 t 时隙的调度算法选择在 $t+1$ 时隙转发 $\mathrm{VOQ1}(i-2,v)$ 中的信元，则将 $\mathrm{L}_{i-2}(t)[v]$ 自减 1 即可。

3.6 算法调度

FRTM-TSA 可选择文献[12]提出的轮询(Round-Robin, RR)、最早离开者优先(Earliest Departure First, EDF)或 LQF 等算法进行调度。

3.6.1 基本约束条件　若 t 时隙有 $\mathrm{I}_i\to\mathrm{M}_j\to\mathrm{O}_{i-2}$，则据 3.1 节可知，$t+2$ 时隙有 $\mathrm{I}_{i-2}\to\mathrm{M}_j\to\mathrm{O}_{i-4}$，$t+3$ 时隙有 $\mathrm{M}_j\to\mathrm{O}_{i-5}$。不失一般性，考虑 I_{i-2} 在 $t+1$ 时隙的算法调度(其结果将决定 $t+2$ 时隙被转发至 M_j 的信元)。I_{i-2} 在 $t+1$ 时隙的算法调度受两个因素的制约：

首先，算法只能选择 I_{i-2} 本地的某个非空缓存队列，不妨将算法选择的缓存队列记为 $\mathrm{VOQ1}(i-2,v)$，则必有 $\mathrm{VOQ1}(i-2,v)$ 非空，即 $\mathrm{L}_{i-2}(t)[v]>0$。

其次，由于 $\mathrm{VOQ1}(i-2,v)$ 中的信元到达 M_j 后只能缓存于 $\mathrm{VOQ2}(j,v)$ 且 $\mathrm{VOQ2}(j,v)$ 只有一个信元的存储空间(如 3.2 节所述)，因此为保证信元在 $t+2$ 时隙到达 M_j 后不发生冲突，算法必须保证目标缓存队列 $\mathrm{VOQ2}(j,v)$ 为空，即满足条件 $\mathrm{MQ}_j(t)\&(1<<v)==0$。否则信元到达 M_j 后必将因 $\mathrm{VOQ2}(j,v)$ 非空而发生冲突。

综上所述，若 I_{i-2} 在 $t+1$ 时隙的调度算法选择缓存队列 $\mathrm{VOQ1}(i-2,v)$，则必须满足以下条件：

$$\mathrm{L}_{i-2}(t)[v]>0\ \ \&\&\ \mathrm{MQ}_j(t)\ \&\ (1<<v)==0$$

3.6.2 算法简述

(1)RR 算法：RR 算法设置轮转指针 p 指示调度的起始位置。每次调度都从 p 所指位置开始向后搜索第 1 个符合条件的缓存队列，若能够找到这样的缓存队列，则将其作为调度结果并更新指针 p 使其指向下一个缓存队列。

(2)EDF 算法：注意到在 $t+3$ 时隙有 $\mathrm{M}_j\to\mathrm{O}_{i-5}$，因此若 I_{i-2} 在 $t+1$ 时隙的算法调度选择的是 $\mathrm{VOQ1}(i-2,i-5)$，则其队首信元将在 $t+2$ 时隙到达 M_j，紧接着在 $t+3$ 时隙被转发至 O_{i-5}。这种情况下，信元在中间缓存等待的时延最短。

基于这一思想，I_{i-2} 在 $t+1$ 时隙的 EDF 算法按 $\mathrm{VOQ1}(i-2,i-5)$, $\mathrm{VOQ1}(i-2,i-6)$, …, $\mathrm{VOQ1}(i-2,\ 0)$, $\mathrm{VOQ1}(i-2,\ N-1)$, $\mathrm{VOQ1}(i-2,\ N-2)$, …, $\mathrm{VOQ1}(i-2,i-4)$ 的顺序找第 1 个符合条件的缓存队列并将其作为调度结果。

(3)LQF 算法：LQF 算法每次调度都选择符合条件的最长的缓存队列。

3.6.3 复杂度分析　为保证交换设备的可靠性，在工程实践中必须满足调度算法在最坏情况下的耗时需求，即调度算法在最坏情况下的复杂度对于交换设备的设计和性能具有实际的参考意义。在最坏情况，RR, EDF 和 LQF 算法均需搜索全部 N 个缓存队列，故最坏情况下，三者的复杂度均为 $O(N)$。

4 相关理论分析

4.1 保序性证明

定理 1　2-错列对称连接模式下属于同一个流的信元在中间缓存等待的时延相等且为定值。

证明　考虑 t 时隙，流 $\mathrm{F}_{i,k}$ 的信元 $\mathrm{C}_{i,k}$ 被转发至 M_τ，由锚定特性可知，t 时隙必有 $\mathrm{M}_\tau\to\mathrm{O}_{i-2}$。

由式(3)和式(4)可知，$t+1$ 时隙与 M_τ 相连的输出端口的序号 v 有

$$v=(i-3)\bmod N \tag{5}$$

即 M_τ 按照 $r=v,v+1,\cdots,N-1,0,1,\cdots,v-1$ 的顺序转发 $\mathrm{VOQ2}(\tau,r)$ 中的信元。故信元 $\mathrm{C}_{i,k}$ 在 $\mathrm{VOQ2}(\tau,$

k) 中等待的时延 $d_{i,k}$ 有

$$d_{i,k}=(k-v)\bmod N \tag{6}$$

由式(5)和式(6)可得

$$d_{i,k}=(k-i+3)\bmod N \tag{7}$$

式(7)表明，流 $\mathrm{F}_{i,k}$ 的任意信元在中间缓存等待的时延仅与 i 和 k 有关，即同一个流的信元在中间缓存等待时延相等且为定值。 证毕

定理 2 2-错列对称连接模式保证信元不会失序。

证明 因同一个流的信元离开 VOQ1 的顺序和到达 VOQ1 的顺序相同，且由定理 1 可知，同一个流的信元因在中间缓存处需等待相同的时间，故同一个流的信元到达目的输出端口的顺序与它们到达输入端口的顺序相同，即信元不会失序。 证毕

定理 3 $\mathrm{MQ}_j(t)$ 等同于 $\mathrm{Q}_j(t+1)$。

证明 由于一个时隙内至多只能有一个信元到达中间端口，也至多只能有一个信元离开中间端口，故考虑在 $t+1$ 时隙有一个信元离开和一个信元到达的情况，不妨设 $\mathrm{VOQ2}(j,v)$ 中的信元离开了 M_j，有一个新到达的信元缓存于 $\mathrm{VOQ2}(j,u)$。显然 $\mathrm{Q}_j(t)$ 和 $\mathrm{Q}_j(t+1)$ 有两个 bit 不同：$\mathrm{Q}_j(t)$ 的第 v 位为 1，第 u 位为 0，而 $\mathrm{Q}_j(t+1)$ 的第 v 位为 0，第 u 位为 1。

由 3.1 节中的预处理方案可知，$t+1$ 时隙有 $\mathrm{VOQ2}(j,v)$ 中的信元离开了 M_j，则在 $t+1$ 时隙必有 $\mathrm{M}_j\to\mathrm{O}_v$，则按照 $\mathrm{Q}_j(t)\&=\sim(1<<v)$ 的方法处理后必有 $\mathrm{Q}_j(t)$ 的第 v 位变为 0。

由 3.4 节中的再处理方案可知，在 $t+1$ 时隙有一个新到达的信元缓存于 $\mathrm{VOQ2}(j,u)$，这就意味着 $\mathrm{S}_{i-1}(t)$ 的第 u 位为 1，其余位均为 0，按照 $\mathrm{MQ}_j(t)=\mathrm{Q}_j(t)|\mathrm{S}_{i-1}(t)$ 的方法处理后必有 $\mathrm{MQ}_j(t)$ 的第 u 位为 1，其余位等同于预处理后的 $\mathrm{Q}_j(t)$。 证毕

综上所述，经过预处理后获得的 $\mathrm{MQ}_j(t)$ 与 $\mathrm{Q}_j(t+1)$ 是完全相同的。

基于 $\mathrm{MQ}_j(t)$ 进行调度，不会出现 FTSA-2-SS 中的信元冲突问题，仅需向 FTSA 一样为任意 $\mathrm{VOQ2}(j,k)$ 设置一个信元的缓存空间，定理 1 已证明这种机制可保证信元不会失序，自然无需在输出端设置重排序缓存来纠正信元离开交换机的顺序。

4.2 算法执行时间分析

若将一个时隙的时间记为 T_{Slot}，crossbar 的重配置时间记为 T_R，信元的发送时间记为 T_C，N 个 bit 信息的发送时间记为 T_N，N 个 bit 的反馈信息从输出端口反馈至位于同一线卡的输入端口所需时间记为 T_F，调度之前的信息处理时间记为 T_D，信元经过 crossbar 的传播时延记为 T_P，如图 4 所示，则 $T_{\mathrm{Slot}}=T_R+T_C+T_P+2T_N$。

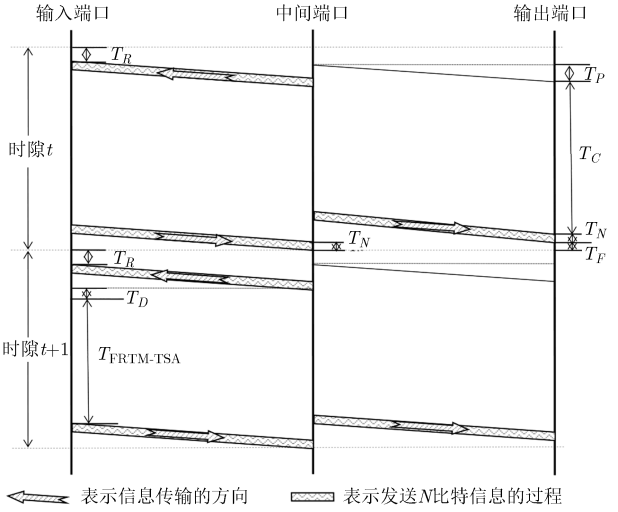

图 4 FRTM-TSA 中调度算法的执行时间分析

若将 FRTM-TSA 中算法的执行时间记为 $T_{\mathrm{FRTM\text{-}TSA}}$，则 $T_{\mathrm{FRTM\text{-}TSA}}=T_C-T_D-T_P$。

考虑相同的分析方法，FTSA, FTSA-2-SS 中算法的调度时间 T_{FTSA}，$T_{\mathrm{FTSA\text{-}2\text{-}SS}}$ 分别为

$$T_{\mathrm{FTSA}}=T_R-T_F,\ T_{\mathrm{FTSA\text{-}2\text{-}SS}}=T_{\mathrm{Slot}}$$

考虑到 T_R, T_F, T_P, T_N 均远小于 T_C，相对于 FTSA，本文所提出的 FRTM-TSA 能够为算法提供较长的执行时间，这一特性使得交换结构能够支持较大的交换规模和较高的交换速率。

4.3 时延性能分析

不失一般性，考虑 I_{i-2} 的算法调度：

(1)首先，FRTM-TSA 在调度之前基于将 $\mathrm{Q}_j(t)$ 和 $\mathrm{S}_{i-1}(t)$ 等信息进行预处理，由于在一个时隙内至多只能有一个信元到达 M_j，至多只能有一个信元被转发而离开 M_j，而这两种情况均已通过预处理而得到修正，因此修正后的 $\mathrm{MQ}_j(t)$ 和 $\mathrm{Q}_j(t+1)$ 是完全等同的。

(2)其次，FRTM-TSA 基于 $\mathrm{MQ}_j(t)$ 和 $\mathrm{L}_{i-2}(t)$ 进行调度。$\mathrm{L}_{i-2}(t)$ 在预处理过程中已经考虑到 $t+1$ 时隙因被转发而离开的信元，因此 $\mathrm{L}_{i-2}(t)$ 和 $\mathrm{L}_{i-2}(t+1)$ 至多只可能缺少一个信元的信息，即当前时隙到达 I_{i-2} 的信元。

综上所述，在相同的交换环境且采用相同调度算法时，FRTM-TSA 的调度算法仅比 FTSA 少考虑了当前时隙到达的信元。当且仅当 FRTM-TSA 的调度算法只能选择当前时隙到达的信元时，FRTM-TSA 才可能出现时延性能的损失。

5　仿真分析

5.1 实验方案

本文选择对输出排队结构(Output Queuing, OQ), IQ+iSLIP, Byte-Focal, FTSA, FTSA-2-SS 和本文提出的 FRTM-TSA 分别进行仿真实验。以上所选择的交换结构中，OQ 理论时延性能最优，但由于该结构需要 N 倍的加速比，故在高速交换环境中基本是无法实现的(除非 N 极小)，其时延性能常被视为交换结构性能的上限；IQ+iSLIP 应用最为广泛，但该结构需要复杂的集中式调度，在自相似数据流环境中其时延性能随负载率的上升迅速恶化。Byte-Focal 是一种优秀的两级交换结构，其所采用 VIQ(Virtual Input Queuing)模式的重排序缓存曾被许多交换结构方案借鉴；FTSA 是迄今为止理论性能最优的负载均衡类两级交换结构。FTSA-2-SS 和 FRTM-TSA 都是针对算法调度时间限制的解决方案，仿真中负载均衡类交换结构均在用 LQF 算法进行调度。

仿真实验利用 Opnet14.5 建立 32×32 的交换模型，分别利用均匀数据流模型、突发数据流模型和 Hot-Spot 流量模型分别模拟信元均匀到达、突发到达和不规则到达的交换环境。

5.2 结果分析

3 种环境中的仿真结果分别如图 5、图 6 和图 7 所示。图中，负载率表示单位时间(时隙)内到达输入端口的信元数量的均值。图 5 表明，在均匀数据流环境中，由于不需要对到达输入端的数据流进行负载均衡，两级交换结构 Byte-Focal, FTSA, FTSA-2-SS 和 FRTM-TSA 相对于传统的交换结构而言没有性能优势。图 6 表明，在突发数据流环境中，随着负载率的上升，传统交换结构的时延性能迅速恶化，两级交换结构 Byte-Focal, FTSA, FTSA-2-SS 和 FRTM-TSA 却因第 1 级 crossbar 的负载均衡作用而表现出较为优秀的时延性能。图 7 表明，在 Hot-Spot 数据流环境中，传统的交换结构最高只能达到 80%的吞吐率，而两级交换结构 Byte-Focal, FTSA, FTSA-2-SS 和 FRTM-TSA 仍保持较优的时延性能。

因反馈型两级交换结构是在已知目标缓存状态信息的前提下进行调度的，因此能够获得更为优异的时延性能。图 5、图 6 和图 7 的仿真结果表明 FTSA, FTSA-2-SS 和 FRTM-TSA 的时延性能均明显优于同环境下 Byte-Focal 的时延性能。

对于 3 种反馈型两级交换结构而言，工作机制的不同也会造成明显的时延性能差异。对于 FTSA 而言，FRTM-TSA 因提前一个时隙调度而无法转发当前时隙到达的信元，必然会造成一定的时延性能损失，但这种情况发生的概率较低，其所造成的时延性能损失较小，此外二者在中间缓存和输出端均具有相同的结构，因此 FRTM-TSA 的时延性能会略逊于 FTSA。对于 FTSA-2-SS 而言，FRTM-TSA 用精确的目标缓存状态信息进行调度，避免了中间缓存处的信元冲突问题，没有冲突就不会产生额外的等待时延；在输出端口，信元将直接离开交换机，避免产生重排序时延，因此从交换机制分析来看，FRTM-TSA 必然具有优于 FTSA-2-SS 的时延性能。图 5、图 6 和图 7 的仿真结果证实了这种分析。

6　结束语

FRTM-TSA 和 FTSA-2-SS 同属于解决算法执行时间限制问题的方案，相对于 FTSA-2-SS 而言，FRTM-TSA 结构简单，硬件开销小，时延性能更优，然而其代价分析如下：

(1)FRTM-TSA 所采用的逆向传输机制增加了 crossbar 控制的复杂度：　考虑到 FRTM-TSA 所增加的是常规的传输控制机制且其实现较为简单，相对于 FTSA-2-SS 中的双信元缓冲模式以及需要大量缓存空间的重排序缓存而言，实质上其硬件开销的代价是降低的。

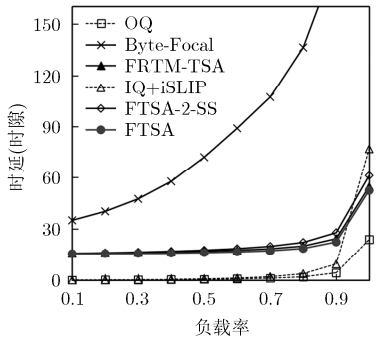

图 5 均匀数据流环境中的时延性能比较

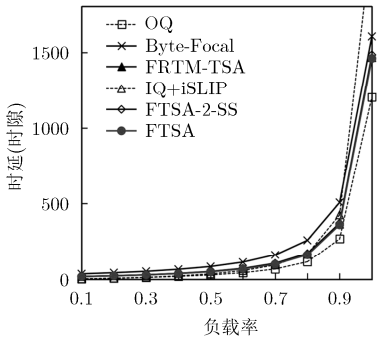

图 6 突发数据流环境中的时延性能比较

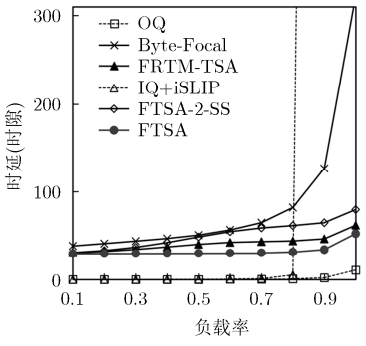

图 7　Hot-Spot 数据流环境中的时延性能比较

(2)FRTM-TSA 所允许的算法执行时间相对于 FTSA-2-SS 而言稍有下降： 一般而言，网络交换设备中调度算法的执行时间不会超过一个时隙。FRTM-TSA 所允许的算法执行时间基本接近一个时隙，在非极端的交换环境中 FRTM-TSA 能够满足大多数的交换需求。然而在极端的交换环境，如超高速、超大规模交换等，时隙将进一步缩短，允许调度算法执行的时间区间将更为狭窄，FRTM-TSA 可能再次面临调度算法执行时间不足的问题。因此为彻底解决两级交换结构对算法的时间限制问题，本项目组下一步将重点研究调度算法，从降低算法复杂度和算法耗时的角度进一步提高两级交换结构的高速交换能力和可扩展性。

参 考 文 献

[1] XIAO Jie, YEUNG K L, and JAMIN S. Pipelined scheduler for unicast and multicast traffic in input-queued switches[C]. IEEE Global Communications Conference, Washington, D.C., USA, 2016: 1–6. doi: 10.1109/GLOCOM.2016.7842148.

[2] HU Bing, YEUNG K L, ZHOU Qian, *et al*. On iterative scheduling for input-queued switches with a speedup of $2-1/N$[J]. *IEEE/ACM Transactions on Networking*, 2016, 24(6): 3565–3577. doi: 10.1109/TNET.2016.2541161.

[3] CERUTTI I, CORVERA J A, DUMLAO S M, *et al*. Simulation and FPGA-based implementation of iterative parallel schedulers for optical interconnection networks[J]. *IEEE/OSA Journal of Optical Communications and Networking*, 2017, 9(4): C76–C87. doi: 10.1364/JOCN.9.000C76.

[4] CHANG C S, LEE D S, and JOU Y S. Load balanced Birkhoff-von Neumann switches[C]. IEEE Workshop on High Performance Switching and Routing, Dallas, TX, USA, 2001: 276–280. doi: 10.1109/HPSR.2001.923646.

[5] YE T, ZHANG J, LEE T T, *et al*. Deflection-compensated Birkhoff-von-Neumann switches[J]. *IEEE/ACM Transactions on Networking*, 2017, 25(2): 879–895. doi: 10.1109/TNET.2016.2606766.

[6] DURKOVIC S and CICA Z. Birkhoff-von-Neumann switch with deflection based load balancing[C]. Telecommunications Forum, Belgrade, Republic of Serbia, 2016: 1–4. doi: 10.1109/TELFOR.2016.7818731.

[7] KESLASSY I and MCKEOWN N. Maintaining packet order in two-stage switches[C]. IEEE International Conference on Computer Communications, New York, USA, 2002, 2: 1032–104. doi: 10.1109/INFCOM.2002.1019351.

[8] KESLASSY I, CHUANG S, YU K, *et al*. Scaling Internet routers using optics[C]. Proceedings of ACM SIGCOMM, Karlsruhe, Germany, 2003: 189–200. doi: 10.1145/863955.863978.

[9] CHANG C S, LEE D S, SHIH Y J, *et al*. Mailbox switch: A scalable two-stage switch architecture for conflict resolution of ordered packets[J]. *IEEE Transactions on Communications*, 2008, 56(1): 136–149. doi: 10.1109/TCOMM.2008.050427.

[10] YU C L, CHANG C S, and LEE D S. CR switch: A load-balanced switch with contention and reservation[J]. *IEEE/ACM Transactions on Networking*, 2009, 17(5): 1659–1671. doi: 10.1109/TNET.2008.2010624.

[11] SHEN Y , PANWAR S S, and CHAO H J. Design and performance analysis of a practical load-balanced switch[J]. *IEEE Transactions on Communications*, 2009, 57(8): 2420–2429. doi: 10.1109/TCOMM.2009.08.070477.

[12] HU Bing and YEUNG K L. Feedback-based scheduling for load-balanced two-stage switches[J]. *IEEE/ACM Transactions on Networking*, 2010, 18(4): 1077–1090. doi: 10.1109/TNET.2009.2037318.

[13] CAI Yan, WANG Xiaolin, GONG Weibo, *et al*. A study on the performance of a three-stage load-balancing switch[J]. *IEEE/ACM Transactions on Networking*, 2014, 22(1): 52–65. doi: 10.1109/TNET.2013.2244906.

[14] HE Chunzhi, HU Bing, and YEUNG K L. FTMS: An efficient multicast scheduling algorithm for feedback-based two-stage switch[C]. Global Communications Conference, Anaheim, California, USA, 2012: 2541–2546. doi: 10.1109/GLOCOM.2012.6503499.

[15] DURKOVIC S and CICA Z. Birkhoff-von Neumann switch based on greedy scheduling[J]. *IEEE Computer Architecture Letters*, 2017, (99): 1–1. doi: 10.1109/LCA.2017.2707082.

[16] HUANG An and HU Bing. The optimal joint sequence design in the feedback-based two-stage switch[J]. *Journal of Network & Computer Applications*, 2014, 45: 27–34. doi: 10.1016/j.jnca.2014.06.011.

[17] LIN Y S and SHUNG C B. Quasi-pushout cell discarding[J]. *IEEE Communication Letters*, 1997, 1(5): 146–148. doi: 10.1109/4234.625041.

[18] 申志军, 曾华燊, 高志江. 一种改进的反馈制两级交换结构 FTSA-2-SS[J]. 电子与信息学报, 2011, 33(6): 1319–1325. doi: 10.3724/SP.J.1146.2010.01207.

SHEN Zhijun, ZENG Huasheng, and GAO Zhijiang. An improved feedback-based two-stage switch architecture using 2-staggered symmetry connection pattern[J]. *Journal of Electronics & Information Technology*, 2011, 33(6): 1319–1325. doi: 10.3724/SP.J.1146.2010.01207.

申志军： 男，1976 年生，副教授，研究方向为网络与通信技术、云计算、大数据处理等.

高 静： 女，1970 年生，教授， 研究方向为软件工程、云计算、大数据处理等.

乌日更： 男，1980 年生，讲师， 研究方向为软件工程.