一种输入输出排队交换机中分布式分组调度方法的研究 [1]

涂晓东 李乐民

(电子科技大学通信与信息工程学院 光纤传输与通信系统技术国家重点实验室 成都 610054)

摘 要 针对采用共享缓存 (shared memory) 做为交换机构 (switching fabric) 的输入输出排队交换机，该文给出了一个分布式分组调度方法 DHIOS(Distributed Hierarchical Ingress and Output Scheduling) 并做了详细的仿真。表明 DHIOS 可以支持变长分组，能够确保业务流的 QoS，性能优良。

关键词 共享缓存，交换机构，输入输出排队交换机，分组调度

中图号 TN913.24

1 前 言

在输出排队 (output queuing) 交换机中，到达输入端口的分组被立即送往输出端口，各个分组在输出端口排队。在输出端口采用 GPS(General Processor Sharing) 类型的分组公平调度算法就能够保证各个业务流的带宽、时延等 QoS 参数。输出排队交换机在文献中也被称为共享缓存交换机，去往各个输出端口的分组共享缓存。由于这种交换机中的共享缓存的读写速率是端口速率的 N 倍 (N 为端口数)，且为了支持突发业务，缓存数量应该足够大，当端口速率和端口数增加时，所需共享缓存的读写速率和数量都迅速上升，实现复杂性和成本直线上升。一种解决方法是在线路卡的入端口处设置缓存，采用“反压” (backpressure) 机制使得线路卡入端口处的缓存就像是交换机构中共享缓存的扩展。即当交换机构中某一队列的缓存占用达到一定门限时，产生反压信号，将相应输入队列的分组缓存在线路卡入端口处。当反压解除时，分组又可进入交换机构。由于线路卡入端口处的缓存的读写速率只取决于端口速率，将大量的缓存放置在线路卡入端口而将少量的缓存放置在交换机构中就可降低成本和实现难度。IBM 的 PRIZMA 芯片系列 [1] 和 Juniper 的路由器 M40[2] 就采用了这种结构。这种结构的交换机是一种输入输出排队交换机，为保证业务流的 QoS，必须采用分布式的调度方法，即在线路卡入端口和交换机构等多个排队点协同调度。许多文献习惯上将线路卡入端口和出端口称为 Ingress port 和 Egress port，将交换机构的输入端口和输出端口称为 Input port 和 Output port，本文采用这些称谓。第 2 节给出 DHIOS(Distributed Hierarchical Ingress and Output Scheduling：分布式层次化 (线路卡) 入端口 (交换机构) 输出端口调度)，第 3 节给出反压队列的势能更新方法，第 4、5 节通过仿真研究 DHIOS 的性能，第 5 节对全文进行总结。

2 DHIOS 调度算法

图 1 是 $(N+1)\times(N+1)$ 的交换机的 DHIOS 调度算法。规定 $\mathrm{VO}_{i,j,k}$ 表示从 Ingress port i 进入去往 Egress port j 的第 k 个流在 Ingress port i 中的队列，VO 表示虚拟输出排队。$\mathrm{VO}_{i,j}$ 表示从 Ingress port i 进入去往 Egress port j 的所有流在 Ingress port i 中的队列的集合。$\mathrm{FQ}_{i,j}$ 表示从 Ingress port i 进入交换机构并去往 Egress port j 的分组在交换机构中的队列。FQ 代

[1] 2001-11-30 收到，2002-11-08 改回
信息产业部电子科学技术研究院预研项目及深圳华为科技基金资助

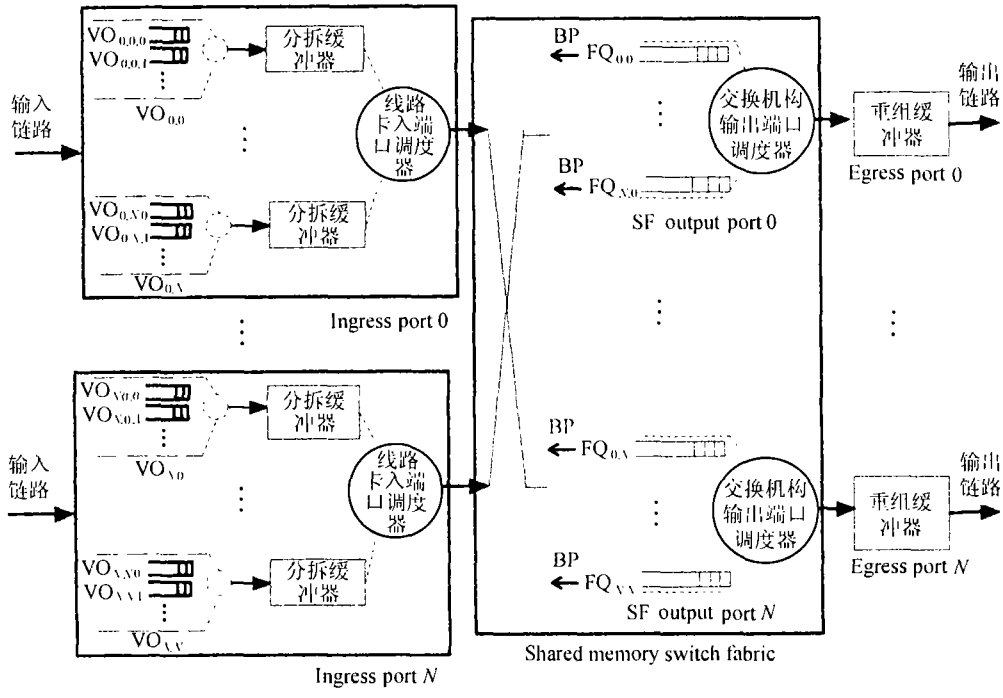

图 1 DHIOS 调度算法

表 Fabric Queue(交换机构队列)，总共有 $(N+1)^2$ 个 FQ． BP 代表反压 (Back pressure) 信号。

交换机中有 3 种调度器，分别是 VO 调度器、Ingress port 调度器、Output port 调度器。VO 调度器位于 Ingress port 中，每一个 $VO_{i,j}$ 对应于一个 VO 调度器。VO 调度器对 $VO_{i,j}$ 中的队列进行调度，输出分组到分拆缓冲器，分组在其中被分拆成长度相等的片段分组。总共有 $(N+1)^2$ 个 VO 调度器。当分拆缓冲器为空，才允许 VO 调度器输出一个分组。Ingress port 调度器对各个分拆缓冲器中的片段分组进行调度，在每一个时隙输出一个片段分组到交换机构。总共有 $N+1$ 个 Ingress port 调度器。交换机构根据片段分组要去往的线路卡出端口号和其进入的线路卡入端口号将其加入到相应的 FQ 队列。根据出端口号确定对应的 Output port 调度器，根据入端口号确定是调度器中的哪一个队列。在交换机构的输出端，在每一时隙，每一个 Output port 调度器对其所包含的 FQ 队列进行调度，输出片段分组到 Egress port ．总共有 $N+1$ 个 Output port 调度器。片断分组在 Egress port 的 Reassembly buffer 中等待组装，当一个分组的所有片段分组到齐后，该分组被恢复，并被发送到输出链路。

当队列 $FQ_{i,j}$ 的长度到达设定的门限时，产生反压信号，指示线路卡入端口调度器 i 停止输出去往线路卡出端口 j 的片段分组。所有从线路卡入端口 i 进入并去往线路卡出端口 j 的分组将滞留在队列中。当队列长度小于设定的门限时，解除反压信号。

VO 调度器可以采用各种支持变长分组的调度算法，例如 WF^2Q+[3] 和 Deficit round robin(欠亏轮循)[4]．线路卡入端口调度器和交换机构输出端口调度器只针对定长的片段分组进行调度，可以采用 WF^2Q+ 、WRR 等算法。VO 调度器中，队列的权重等于业务流的预约带宽。线路卡入端口调度器中队列的权重等于对应的 $VO_{i,j}$ 中各个队列的权重之和。交换机构输出端口调度器中队列 $FQ_{i,j}$ 的权重等于对应的 $VO_{i,j}$ 中各个队列的权重之和。

以上提出的这个分布式调度方法可以保证每一个业务流的 QoS，不仅可以支持 ATM 信元还可以支持变长的 IP 分组. 由于这一方法是分布式、层次化、线路卡入端口和交换机构输出端口联合调度，因此我们将其取名为 DHIOS . 由于反压的存在，调度器中队列更新势能时必须考虑反压的影响，而相关文献 [1, 5] 未对此进行研究，下面我们将对此问题进行分析.

3 反压队列的势能更新办法

3 种调度器中，Ingress port 调度器要处理反压信号. 一个队列在被反压期间不参与调度，其势能维持不变，调度器的系统势能的更新将不考虑该队列的影响. 当该队列被解除反压后，其势能如何更新？

一种方法是维持队列在反压前的势能. 该方法将产生“反压队列优先”现象，导致未被反压的队列的分组时延增加. 这是因为在反压期间，其它队列的势能和系统势能都在增加，反压的队列在解除反压后的势能可能相对较小，得到相对优先的被调度器选择的机会. 例如：Ingress port 调度器、Output port 调度器均采用 WF^2Q+ 算法，Ingress port 0 调度器有三个队列，队列 0 的权重为 r_0，队列 1 为 r_1，队列 2 为 r_2(权重均归一化). 队列 0 的输出端口为 0，在交换机构的输出端口 0 还有其他队列与队列 0 竞争. 用 $V(t)$ 表示调度器的系统势能，$S_i(t)$ 表示队列 i 的开始势能，$F_i(t)$ 表示队列 i 的结束势能. 假设在时刻 t_1 队列 0 被反压，有 $S_0(t_1)-V(t_1) \le 1/r_0$，在时刻 t_2 反压被解除，t_2-t_1 约等于 $1/r_0$. 假设在时间段 $[t_1,t_2]$ 内，队列 0 和队列 1 有分组在等待，队列 1 没有被反压，队列 2 处于空闲状态. 则此时段内，队列 1 一直被服务，其开始势能的增量约为 $1/r_0 \times 1/r_1$，系统势能的增量也约为 $1/r_0 \times 1/r_1$，即 $V(t_2)-V(t_1) \cong 1/r_0 \times 1/r_1$. 设在时刻 t_2，队列 2 到达一个分组 P，有 $S_2(t_2) \ge V(t_2)$，队列 1 变为空闲，而队列 0 仍然有分组在等待. 在此时刻，如果维持队列 0 的势能，则有 $S_0(t_2) = S_0(t_1)$，则 $S_2(t_2) - S_0(t_2)$ 约为 $1/r_0 \times (1/r_1 - 1)$，设 $r_0 = r_2$，则 $F_2(t_2) - F_0(t_2)$ 约为 $1/r_0 \times (1/r_1 - 1)$，队列 2 的分组 P 的时延约为 $1/r_1 - 1$，即不是与自身的预约带宽有关而是与其他队列的预约带宽有关. 当 r_1 很小时，时延很大，显然这不是我们所希望的.

解决这一问题的方法是：当反压队列被解除反压时，将队列的结束势能和系统势能作比较，如队列的结束势能小于系统势能，则将队列的开始势能设定等于系统势能，并相应改变队列的结束势能，如不小于则不做改变. 将这一方法用于上例，可使 $S_0(t_2) \ge V_0(t_2) - 1/r_0$，不会使队列 2 的分组 P 的时延与其他队列的预约带宽有关. 在我们的调度方案中正是采用这一方法来更新反压队列的势能.

4 仿真环境设置

我们利用 OPNET 构造了如图 2 的仿真环境，其中有 5 种模块，IP ingress 模块、Shared buffer switch fabric 模块和 IP reassembly 模块构成交换机，IP GEN 模块产生业务流，Flow sink 模块终结各个流并统计其参数，业务源模块 IP GEN 产生分组长度在 64byte 至 1500byte 之间变化的业务流，分组头含有 Source 和 Destination，Source 表示由哪一个 IP GEN 模块产生，Destination 表示目的端口，分组的 Payload 长度可变. 图 1 中，VO 调度器和 Ingress port 调度器是层次关系，VO 调度器可以采用各种支持变长分组的调度算法，如果 $VO_{i,j}$ 作为一个整体其 QoS 能够得到保证，则其中每一个流的 QoS 能够得到保证，因此在仿真中设置一个 $VO_{i,j}$ 只包含一个流，用 Fx_y 表示从业务源 x 发出目的端口为 y 的流. IP ingress 模块将到达的分组分拆成 64byte 的片段分组，经调度选出的片段分组在送往 Switch fabric 模块前被加上一个 Header，包含 1byte 的 Source 和 1byte 的 Destination，成为 66byte 的交换机构

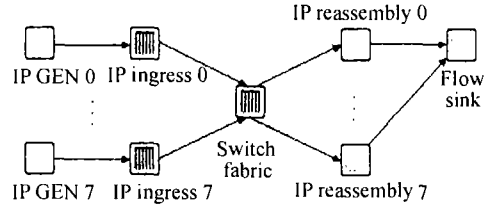

图 2 仿真环境

分组. 交换机构根据分组的 Destination 确定对应的 Output port 调度器, 根据 Source 确定队列, 并将分组加入到该队列. 交换机构分组经 Output port 调度器输出, 离开交换机构模块. IP reassembly 模块去掉交换机构分组的 2bytes 头, 将 64byte 的片段分组存储在 Reassembly buffer 中, 等一个分组的所有片段分组到齐后, 重装该分组并发送到输出链路. 所有的输出链路终结在 Flow sink 模块, 该模块统计每一个流的时延、通过速率等参数.

输入输出链路速率为 160Mb/s , 交换机构时隙为 (64byte)/(160Mb/s)=3.2μs . 每 3.2μs IP ingress 模块可以送一个 66byte 的分组到达交换机构模块, 交换机构模块可以送一个 66byte 的分组到 IP egress 模块, 其中有效负载为 64byte , 因此 IP ingress 模块和交换机构模块之间、交换机构模块和 IP egress 模块之间的连接速率略高于 160Mb/s , 为 $160 \times 66/64 = 165$Mb/s .

IP ingress 模块对分组的分拆可能放大业务流的流量. 一个长度不是 64 倍数的分组将被附加一个填充 (PAD) 使其长度等于 64 的倍数, 这样会使业务流的流量大于其预约带宽, 增加分组时延, 避免这一因素的影响有两个方法. 一是在 IP GEN 模块中考虑这一因素, 使得 IP GEN 产生的业务流流量略小于预约带宽, 而经过分拆作用其流量仍等于预约带宽; 二是规定产生的分组的长度是 64 的倍数. 采用第二种方法, 分拆过程对流量透明, 不增加或减少流量, 便于我们研究调度算法对业务流的作用, 因此我们选用了第二种方法.

IP ingress 模块中的 Ingress port 调度器和交换机构模块中的 Output port 调度器均采用 WF^2Q+ 算法.

5 仿真实验及结果

实验 1 IP GEN 0 产生 8 个流, F0_i, ($i = 0 \sim 7$) , 预约带宽分别为 40,30,20,15,15,20,10,10 , 单位是 Mb/s(以下同) . IP GEN 1~7 分别产生一个流, 目的端口都为 0 , 预约带宽分别为 30, 20, 15, 15, 20, 10, 10 . 除 F0_0 到达交换机的速率符合其预约带宽外, 其余流的到达速率均超过其预约带宽, 为 40Mb/s . 对于每一个流, 分组大小为 j×64byte , j 在 1~23 之间均匀分布. 交换机构中每个队列的反压门限设定为 4 个分组, 仿真 1s .

表 1 是对各业务流的时延及通过速率统计的结果. 在统计时延中忽略了输入、输出链路对分组的传输时延, 只考虑交换机对分组的时延. 业务流的通过速率等于仿真时间内业务流到达 Flow sink 的分组总长度除以仿真时间.

表 1　各业务流的时延及通过速率

业务流	最大时延 (ms)	平均时延 (ms)	通过速率 (Mb/s)
F0_0	0.307	0.160	39.998
F0_1	250	125	29.996
F0_2	500	249	20.000
F0_3	625	310	14.997
F0_4	624	315	14.993
F0_5	500	249	19.993
F0_6	750	379	9.999
F0_7	749	373	10.003
F1_0	250	124	30.002
F2_0	499	251	19.993
F3_0	625	314	14.995
F4_0	624	312	14.994
F5_0	500	249	19.996
F6_0	749	372	9.997
F7_0	750	375	9.998

在这种配置下， F0_0 既要在 Ingress port 和违规业务流竞争也要在交换机构的 Output port 和违规业务流竞争，可以检验我们这一套分布式调度算法对 QoS 的保证性能。从表 1 上看出其通过速率几乎等于预约带宽，最大时延几乎等于最大分组长度除以预约带宽，可见即使在最拥挤的情况下，调度算法仍然能够确保对守约业务流的服务质量。其它业务流的通过速率也与其预约带宽相差无几，但时延很大，这是由于它们违规，到达速率超过了被服务的速率。

图 3- 图 5 是对 F0_0 、 F1_0 在交换机构中的队列和在 Ingress port 中的队列的长度每 0.1s 记录一次的结果。在图 3 中是以交换机构分组 (66byte) 为单位的，图 4, 图 5 是以片段分组 (64byte) 为单位的。 F0_0 在 Ingress port 中的队列最长为 23 ，等于一个最大分组的长度，而在交换机构中为 1 。 F1_0 在交换机构中的队列长度几乎一直等于反压门限 4 ，而在 Ingress port 中的队列长度线性增加。流 F2_0 、 F3_0 、 F4_0 、 F5_0 、 F6_0 和 F7_0 的情况与 F1_0 相似。我们还观察到，流 F0_i ($i = 1 \sim 7$) 在交换机构中的队列的长度最大为 1 ，而在 Ingress port 中的队列的长度线性增加，这是由于对这些流来说，交换机构的输出端口不是瓶颈，瓶颈在 Ingress port 。

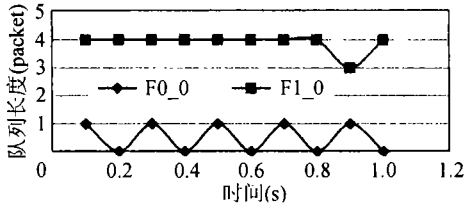

图 3　F0_0 和 F1_0 在交换机构中的队列长度

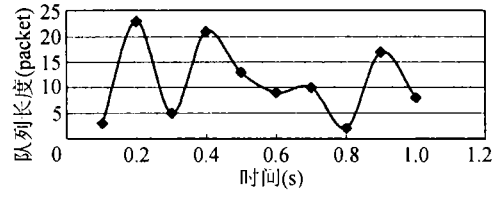

图 4　F0_0 在 Ingress port 中的队列长度

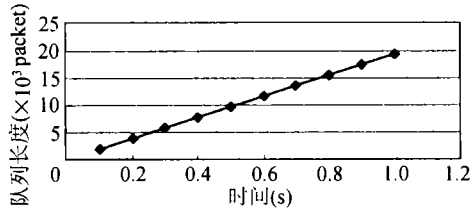

图 5　F1_0 在 Ingress port 中的队列长度

实验 2 保持实验 1 的其它条件不变，只将 F0_0 的产生速率从 40Mb/s 降为 30Mb/s，观察 F0_0 不用的带宽能否公平地分配给其余的流。从表 2 看出，各个流获得的额外带宽几乎与其预约带宽成正比。

表 2 各业务流获得的额外带宽

业务流	预约带宽 (Mb/s)	吞吐率 (Mb/s)	获得的额外带宽 (Mb/s)
F0_1	30	32.463	2.463
F0_2	20	21.665	1.665
F0_3	15	16.212	1.212
F0_4	15	16.248	1.248
F0_5	20	21.637	1.637
F0_6	10	10.821	0.821
F0_7	10	10.829	0.829
F1_0	30	32.479	2.479
F2_0	20	21.640	1.640
F3_0	15	16.238	1.238
F4_0	15	16.215	1.215
F5_0	20	21.645	1.645
F6_0	10	10.813	0.813
F7_0	10	10.798	0.798

6 结 论

在以共享缓存为交换机构的输入输出排队交换机中，本文提出的 DHIOS 调度方法能够在存在违规业务流时确保守约业务流的 QoS，守约业务流的最大时延几乎等于最大分组长度除以预约带宽，这与输出排队交换机所能达到的最大时延几乎一致。交换机构中队列的反压门限设为 2 个信元是合适的。如果反压信号传递到 Ingress port 的时延为 D 个时隙，则一个队列的最大长度可设为 $2+D$ 个信元。对于 $N \times N$ 的交换机，交换机构的缓存设为 $(2+D) \times N^2$ 个信元是合适的。DHIOS 调度算法还能使一个业务流不用的带宽被公平地分配给其它与之竞争的流，不论这种竞争是在交换机的线卡入端口还是在交换机构的输出端口。所以，DHIOS 算法性能优良，可与输出排队交换机媲美。

相比其它相关文献 [1,5]，本文的贡献在于：针对采用共享缓存做为交换机构的输入输出排队交换机，给出了一套完整的分组调度方法，提出并解决反压队列的势能更新问题。该方法将分布在线路卡入端口和交换机构输出端口的调度构成层次化结构，采用一定的反压队列势能更新方法。根据仿真结果本文还给出了反压门限、缓存量等参数的参考设置，这对于相关产品的研制有直接的参考价值。

参 考 文 献

[1] C. Minkenberg, T. Engbersen, A combined input and output queued packet-switched system based on PRIZMA switch-on-a-chip technology, IEEE Communications Magazine, 2000, 38(12), 70–77.

[2] http://www.juniper.net, M40.

[3] J. C. R. Benett, Hui Zhang, Hierarchical packet fair queuing algorithms, IEEE Trans. on Networking, 1997, 5(5), 675–689.

[4] M. Shreedhar, G. Varghese, Efficient fair queuing using deficit round-robin, IEEE Trans. on Networking, 1996, 4(3), 375–385.

[5] F. M. Chiussi, A. Francini, Providing QoS Guarantees in Packet Switches, IEEE GlOBECOM 1999, vol.2, 1582–1590.

THE RESEARCH ON DISTRIBUTED PACKET SCHEDULING METHOD FOR AN INPUT AND OUTPUT QUEUING SWITCH

Tu Xiaodong Li Lemin

(*National Key Lab. of Broadband Optical Fiber Transmission and Communication Networks, UEST of China, Chengdu* 610054, *China*)

Abstract In this paper, a distributed packet scheduling algorithm called DHIOS(Distributed Hierarchical Ingress and Output Scheduling)is presented and simulated for an input and output queuing switch whose switch fabric is a shared memory. The distributed packet scheduling algorithm can support variable length packets and guarantee the QoS (Quality of Service) of flows. The simulation on OPNET shows that the distributed packet scheduling algorithm has good performance.

Key words Shared memory, Switch fabric, Input and output queuing switch, Packet scheduling

涂晓东： 男， 1970 年生，博士，副教授，研究方向为宽带网络中的交换，调度及流量工程等.
李乐民： 男， 1933 年生，教授，博士生导师，中国工程院院士，从事宽带网络等方面的研究.