面向变长分组的多优先级动态域值缓存管理算法

李锁钢, 徐 恪, 吴建平
(清华大学计算机系, 北京 100084)

摘 要: 缓存管理是高性能路由器需要解决的技术难题之一, 一个好的缓存管理算法可以提高路由器的缓存资源利用率并降低分组丢失率. 本文首先介绍了面向定长信元的几种典型动态域值缓存管理算法——DT 算法与最佳 DT 算法以及多优先级 DT 算法. 然后提出了一种适用于高性能路由器的多优先级最佳 DT 算法, 并面向变长分组进行了仿真模拟. 该算法继承了最佳 DT 算法缓存资源利用率高的优点, 在多优先级情况下分组丢失率很低, 模拟分析结果表明该算法的综合性能相当出色.

关键词: 缓存管理; 动态域值; 多丢失优先级
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2002) 08-1188-04

Buffer Management Algorithm Oriented to Length-Varied Packets with Dynamic Thresholds for Multiple Priorities

LI Suo-gang, XU Ke, WU Jiarr-ping
(Department of Computer Science Tsinghua University, Beijing 100084, China)

Abstract: Buffer management is one of the important and difficult problems in high performance routers and switches. One excellent buffer management scheme can improve the resource utilization and decrease the number of discard packets. We first discuss some buffer management schemes and algorithms oriented to fixed length cells with dynamic thresholds-DT and Optimal DT, and DT for multiple loss priorities. Then we put forward the optimal DT for multiple loss priorities that can be applied in high performance routers and simulate it orienting to varying-length packets. It receives the advantage of full use of buffer, owned by the Optimal DT, with fewer number of discard packets in the case of multiple priorities, and the simulation analysis indicates that its performance is quite excellent.

Key words: buffer management; dynamic thresholds; multiple loss priorities

1 引言

网络应用和通信技术的飞速发展, 将互联网上的核心设备——路由器推到了网络技术的焦点位置, 路由器的性能制约着互联网的发展. 如何使路由器的转发速率跟上底层传输链路的速率, 从而满足 Internet 的发展需求, 是路由器需要解决的主要技术问题. 近年来在路由器体系结构和内部的交换结构方面的研究都已经取得了较大的突破[1], 随着多媒体数据流需求迅速增长, 缓存管理成为制约路由器尤其是高性能路由器进一步发展的瓶颈.

本文首先介绍当前用于交换机缓存管理的一些优秀算法, 进而提出"多优先级最佳动态域值缓存管理算法", 然后实验模拟算法在路由器中面向变长分组的性能, 并根据实验结果做了对比和讨论.

2 面向定长信元的缓存管理算法

已有的理论研究结果将交换机的缓存管理策略分为三类, 分别是静态域值策略(ST), PUSH-OUT 策略(PO) 和动态域值策略(DT)[2]. ST 策略实现起来非常简单的, 将 ST 策略扩展到多优先级情况也比较容易, 还可以设定不同优先级的域值大小. 但是它的域值大小不能根据负载随时调节, 所以性能很差. 相反, PO 策略在性能上有很多的优点: 公平、高效和能够自适应. 但是在实际中实现起来很困难, 尤其在高速的交换机中. 与这两种策略相比, 动态域值策略 DT 能够兼有简单性和适应性. 下面介绍几种典型的动态域值缓存管理算法.

2.1 DT 算法[3]

这个算法的基本思想是: 在任意时刻, 输出队列的域值跟当前未用缓存的数量成比例. 输出端口队列按照未用空间的某个函数作为其长度限制, 长度在这个限制值(就是域值)以下的队列都能得到可用的空间. 假设: 在时刻 t, $T(t)$ 表示控制域值; $Q^i(t)$ 表示等待队列的长度; $Q(t)=\sum_i Q^i(t)$ 表示所有队列长度总和, 也就是整个缓存空间已占用的部分; 以 B

收稿日期: 2001-09-10; 修回日期: 2002-04-22
基金项目: 国家自然科学基金(No. 69725003, 90104002)

表示整个缓存的大小, 那么

$$T(t)=f(B-Q(t))=\alpha\cdot(B-Q(t)) \quad (1)$$

公式(1)表明, 如果 $Q^i(t)\geq T(t)$ 则新到来的进入队列的信元将会被阻塞, 直到这个队列的长度降到控制域值以下, 或者域值升高到这个队列长度以上. 这里, α 是未用缓存的系数, 它一般取 2 的幂, 这可以简化域值的计算, 也便于实际的应用.

为了在瞬变过程中减少信元的丢失, DT 算法要求预留出一小部分缓存空间. 它们同时起到监视的作用: 当某个等待队列负载增加占用大量缓存时通知缓存分配进程调整域值.

DT 算法综合了 PO 算法和 ST 算法的长处, 但是它在理论上需要预留空间, 使得缓存资源不能被充分利用.

2.2 最佳 DT 算法[4]

最佳的动态缓存管理算法的目标是: 最大限度的提高缓存利用率, 并保证缓存分配公平.

设有 K 个端口, 它们共享的缓存大小是 B 个信元大小; $Q_k(t)$ 表示 t 时刻输出端口 k($k=1, 2, \cdots, K$)的队列长度, $Q(t)=\sum_{k=1}^{K}Q_k(t)$ 表示所有队列长度总和. 当端口 k 的信元到达比率和输出带宽服务速率变化时 $Q_k(t)$ 会随之而变化. 那么这个算法的公用域值为 $T(t)$, 它可以依据下面的公式动态的调整大小:

$$T_{new}(t)=\begin{cases}\max(Q_k(t)+1, T_{old}), & \text{if } Q(t)<\alpha\cdot B;\\ \max(T_{old}-1, T_m), & \text{if } Q(t)\geq\alpha\cdot B.\end{cases} \quad (2)$$

其中, T_m 表示最小的缓存域值, 初始时设置, 默认可为 0. $Q_0=\alpha\cdot B$ 作为本算法对于超负荷时各个端口队列长度的的管理目标. 参数 α 决定共享的缓存数量.

公式(2)说明, 当所有队列的长度 $Q(t)$ 等于或者大于 Q_0 时, T 将以信元到达的速率减少; 当 $Q(t)$ 小于 Q_0 时, 到达的信元总能被接收, 域值 T 更新为当前的最大队列长度值.

与第一种 DT 算法比较, 本算法没有预留缓存, 所以本算法的资源利用率要高, 在低负载和中等负载情况下尤其明显.

2.3 多优先级 DT 算法[5]

下面介绍的是多丢失优先级情况下的动态域值缓存管理算法. 假设有 P 个丢失优先级: 0(级别最高)、…、$P-1$(级别最低). 定义 $Q_p^i(t)$ 为时刻 t 端口 i 的优先级为 p 的信元数量, 继而定义 $Q_p(t)=\sum_i Q_p^i(t)$, $Q^i(t)=\sum_p Q_p^i(t)$ 和 $Q(t)=\sum_p\sum_i Q_p^i(t)$.

研究在各不同优先级的竞争端口之间分配缓存策略的各种模式, 可分为三种[5]:

(1) OWA: 根据某个控制参数按比例分配缓存和输出负载, 以保证每个优先级端口都得到不同的资源.

(2) OWS: 将大部分资源分配给高优先级队列.

(3) OEA: 在极度拥塞情况下只给高优先级队列分配缓存资源.

将动态域值因子 α 和缓存资源 B 按照优先级作区分, α_P 和 B_P 分别对应的优先级为 P. 本文后面的模拟实验假设有四个队列, 两个优先级 $P=2$, $\alpha_0=2$, $\alpha_1=1$, $B_0=B$, $B_1=0.5B$, $B_2=0$; 并且假设到达第一、二输出端口队列的分组都是高优先级, 到第三、四输出端口队列的分组都是低优先级, 只要设定各个端口的分组到达的概率就可以控制高低优先级的分组分布情况. 这样可以简化三个模式的域值控制公式, OWA 模式是$\begin{cases}Q_0(t)<\alpha_0\cdot(B-Q(t))\\ Q_1(t)<\alpha_1\cdot(B-Q(t))\end{cases}$, OWS 模式是$\begin{cases}Q_0(t)<\alpha_0\cdot(B-Q_0(t))\\ Q_1(t)<\alpha_1\cdot(B-Q(t))\end{cases}$, OEA 模式是$\begin{cases}Q_0(t)<\alpha_0\cdot(B-Q(t))\\ Q_1(t)<\alpha_1\cdot(0.5B-Q(t))\end{cases}$.

3 多优先级最佳 DT 算法

现在提出"多优先级最佳 DT 算法", 能够在多优先级情况下提高缓存管理策略的综合效率, 其管理目标是充分利用缓存资源, 并且保证按照优先级的高低分配缓存资源. 前文介绍的最佳 DT 算法比 DT 算法的性能优越, 它的缓存利用率高, 域值抖动小, 分组丢失率小, 将它的优点引入到多优先级的情况, 得到的算法性能将会提高.

假设, 系统有 K 个输出端口队列, I 个丢失优先级, $i=0, \cdots, I-1$, 优先级依次降低. 定义 $Q_k^i(t)$ 为时刻 t 端口 k 的优先级为 i 的分组数量. 定义: $Q_k(t)=\sum_i Q_k^i(t)$, $Q^i(t)=\sum_k Q_k^i(t)$ 和 $Q(t)=\sum_k\sum_i Q_k^i(t)$. 那么

$$T_{new}^i(t)=\begin{cases}\max_{k\in K}\left\{\sum_{j\geq i}Q_k^j(t)\right\}, & \text{if } Q(t)<\alpha\cdot B;\\ \max\{T_{old}^i-c, T_m\}, & \text{if } Q(t)\geq\alpha\cdot B.\end{cases} \quad (3)$$

这就是每个优先级分组队列长度的控制域值公式. 其中, α 是算法可使用的缓存资源, c 是队列的变化值, 如果队列长度变化量为一个分组的大小, 那么 $c=1$(如公式 2). T_m 表示最小的缓存域值, 初始时设置, 默认可为 0.

公式(3)指出当所有队列的长度 $Q(t)$ 小于可用缓存资源 $\alpha\cdot B$ 时, 优先级为 i 的队列的域值 T^i 更新为 $\max_{k\in K}\{\sum_{j\geq i}Q_k^j(t)\}$, 也就是说, 此时的优先级为 i 的队列域值是各个端口中所有优先级低于或等于 i(i 越大, 优先级越低)的队列的和的最大值. 这符合多优先级情况下缓存管理的要求, 即高优先级队列的域值大于低优先级队列的域值.

综上, 令 $Q_0=\alpha\cdot B$, 当所有队列的长度 $Q(t)$ 等于或者大于 Q_0 时, 优先级为 i 的队列域值 T^i 将以分组到达的速率减少; 当 $Q(t)$ 小于 Q_0 时, 到达的分组总能被接收, 并且域值 T^i 更新为当前的各个端口中所有优先级低于或等于 i 的队列的和的最大值. 由此得到公式(3)等价的形式

$$T_{new}^i(t)=\begin{cases}\max\left\{\sum_{j\geq i}Q_k^j(t)+c, T_{old}^i\right\}, & \text{if } Q(t)<Q_0;\\ \max\{T_{old}^i-c, T_m\}, & \text{if } Q(t)\geq Q_0.\end{cases} \quad (4)$$

4 模拟实验环境和实验设计

4.1 分组长度的分布

统计结果表明[6], 广域网上的分组长度分布具有相对稳定的特点. 在实验中认为: 64Byte、512Byte 和 1500Byte 的分组大约各占总分组个数的 40%、20% 和 40%, 其余长度的分组以小概率出现.

4.2 分组源模型

分组的产生采用自相似过程, 用基于 Pareto 调制的 Pois-

son 过程模拟网络数据流[7]. Burst 以参数为 λ 的 Poisson 过程到达. Burst 流产生以后, 要对分组做进一步处理, 即将所有的分组按照时间先后排序.

此外, 虽然从路由器外部来看, 分组的到达是以 Burst 流的形式进入的, 但是路由器工作时的对象单元仍然是分组, 而且本文主要涉及缓存管理算法的性能, 所以文中进行实验结果分析时, 仍以分组为对象, 不考虑某个时刻的分组究竟属于哪个 Burst 流.

5 算法的模拟结果分析

由于前三个算法是面向定长信元的交换机使用的缓存管理算法, 为了便于分析比较, 在模拟实现时都是面向变长分组的. 下面比较本文介绍的两种多优先级下的动态域值管理算法. 多优先级最佳 DT 算法实验中简称为 myMO 算法.

5.1 输入负载平衡情况

缓存资源为 1M, 实验的分组为一万三千多个, 四个端口的分组到达概率相等都是 0.25. 图 1 是四种多优先级管理算法的缓存利用率比较曲线. 第一组曲线是高优先级分组队列的缓存利用率, 第二组曲线是低优先级的.

从低优先级队列的缓存利用率图中容易看出, OWA 的缓存利用率较高, OWS 次之, OEA 最差. 这是符合这三个模式的性质的: OWA 按比例配给资源, OWS 在拥塞时将大部分资源配给高优先级队列, 而 OEA 则由高优先级队列完全占有资源.

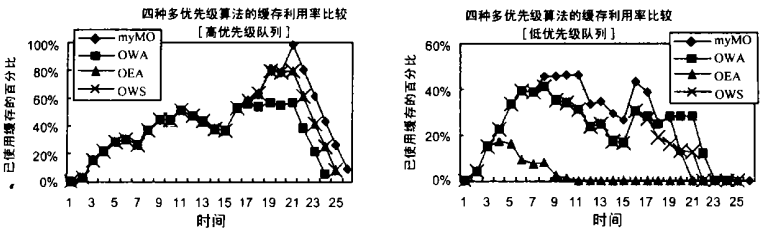

图 1 输入负载平衡时缓存利用率的比较

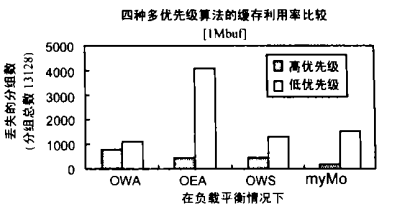

图 2 输入负载平衡时分组丢失情况比较

从图中可见高优先级分组队列的缓存利用率明显高于低优先级队列的, 而 myMO 算法的缓存资源利用率是最高的.

再比较四个算法的分组丢失情况, 如图 2 所示. 可以看出对于低优先级分组 OEA 的丢失率最高, 这体现出它“在极度拥塞情况下只给高优先级分配缓存资源”的特点; 对于高优先级分组 OEA 和 OWS 的丢失率较低, 而 myMO 最低, 这是它充分利用缓存资源的结果.

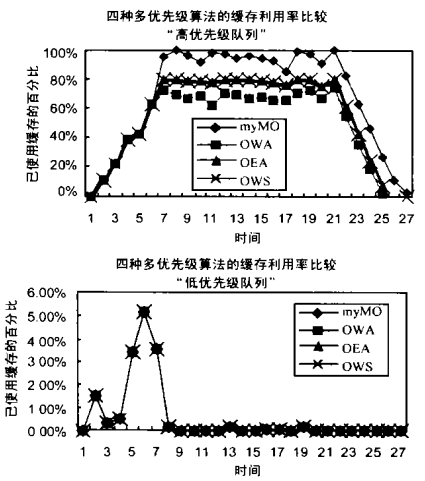

图 3 输入负载不平衡时缓存利用率的比较

5.2 输入负载不平衡的情况

缓存资源为 1M, 实验的分组为一万三千多个, 四个端口的分组到达概率不等, 比例是 4:3:1:2, 即高低优先级分组之比为 7:3. 缓存利用率曲线如图 3 所示.

通过对比这两组曲线和上面负载平衡情况的实验结果图 1 的曲线可见, 由于高优先级的分组很多, 使得高优先级队列的高的缓存使用率持续时间延长, 而低优先级队列的缓存利用率较低, 时间较短.

图 4 是分组丢失情况的比较. 图中曲线反映了高优先级分组的丢失多于低优先级分组, 这是由于高优先级分组数量占分组总数量的绝大部分. 在这种情况下, 只有 OEA 模式的高优先级分组丢失比低优先级分组丢失少, 但是分组丢失的总数量也相当多; 综合比较高低优先级的结果, myMO 算法的分组丢失仍然是最少的.

图 5 是当缓存增加时, 四种多优先级算法的分组丢失情况曲线, 分析其结果发现与前面的实验结论是一致的.

6 结束语

本文提出的多优先级最佳 DT 算法, 并且针对面向变长分组进行了模拟实验, 得到了较好的性能结果. 该算法继承了

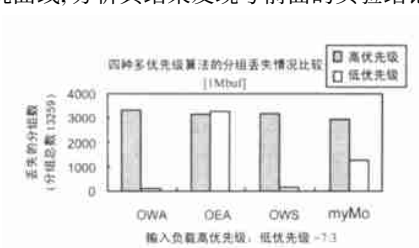

图 4 输入负载不平衡时分组丢失情况比较

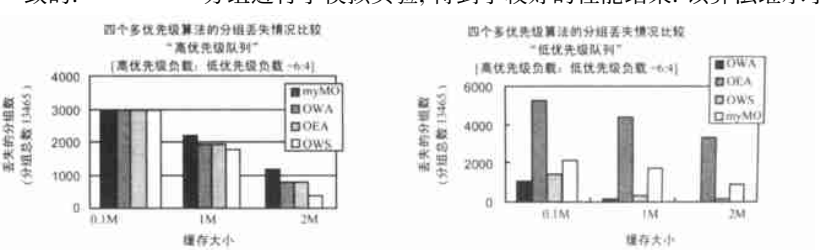

图 5 缓存增加时分组丢失情况比较

最佳 DT 算法缓存资源利用率高的优点, 使各个优先级队列的分组丢失数量都不同程度的减少, 因而能够保证高优先级队列优先获得资源. 该算法已经成功地用于国家“八六三”重点攻关项目“高性能安全路由器”[8] 的研制. 它的性能完全达到高性能路由器处理速度的要求.

对于多优先级最佳 DT 算法, 下一步的工作将结合高性能路由器的实际工作情况调整改进; 算法中的优先级可以从分组丢失率情况扩展到其他情况, 并增加对区分服务的支持.

参考文献:

[1] 徐恪, 熊勇强, 吴建平. 宽带 IP 路由器的体系结构分析 [J]. 软件学报, 2000, 11(2): 179- 186.

[2] Mutlu Arpaci, John A Copeland. Buffer management for shared-memory ATM switches [J]. IEEE Communication Surveys & Tutorials, First Quarter, 2000, 3(1).

[3] A K Choudhury, E L Hahne. Dynamic queue length thresholds for shared-memory packet switches [J]. IEEE/ACM Transactions On Networking, 1998, 6(2).

[4] R Fan, A Ishii, B Mark, G Ramamurthy, Q Ren. An optimal buffer management scheme with dynamic threshold [A]. Proc. IEEE Globecom' 99 [C]. IEEE, 1999.

[5] A K Choudhury, E L Hahne. Dynamic thresholds for multiple loss priorities [A]. Proc IEEE ATM' 97 Workshop [C]. Lisbon, Portugal: 1997. 272- 281.

[6] Planning the technology program [DB/OL]. http://www.quasar.ualberta.ca/edit 574/.

[7] Ronald G Addie, Timothy D. Neame, Moshe Zukerman. Modeling superposition of many sources generating self similar traffic [A]. in Proc IEEE ICC' 99 [C]. IEEE, 1999.

[8] 徐明伟, 徐恪, 吴建平, 等. 高性能安全路由器关键技术研究 [A]. 863 计划智能计算机主题学术会议论文集——智能计算机研究进展 [C]. 北京: 清华大学出版社, 2001.

作者简介:

李锁钢　男, 1978 年 7 月生于河北承德, 博士研究生, 研究方向是分布式路由器操作系统和缓存管理算法.

徐　恪　男, 1974 年 12 月生于江苏洪泽, 博士, 讲师, 主要研究领域为新一代互联网络体系结构, 高性能路由器体系结构, 实时分布式操作系统, 已经在国内主要学术刊物和国际学术会议上发表四十多篇学术论文.