基于时延的动态优先级调度算法

张登银，许扬扬，蒋 娟
（南京邮电大学 计算机学院，江苏 南京 210003）

摘 要：队列管理是提高网络 QoS的一种有效方法。在基于时延的调度算法（BDS）基础上将时间片与优先级相结合，提出了一种基于时延的动态优先级调度算法（DDPQS）。为了实现该算法，针对进入缓冲区的每个子队列设置一个计数器，以调整的计数器值为基准来动态的改变队列的优先级，从而达到队列调度的效果；又从研究该算法的过程中，发现其局限性，即计数器值对时间片过于敏感的问题，于是进一步采用设置阈值进行区分的方法来优化。优化前后的仿真结果表明，时延和吞吐率性能具有明显改善。

关键词：队列调度；时延；动态优先级
中图分类号：TN911.7 文献标识码：A 文章编号：1673-629X(2011)02-0162-04

Delay-Based Dynamic Priority Queue Scheduling

ZHANG Deng-yin, XU Yang-yang, JIANG Juan
(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Queue management is an effective method to improve the QoS of network. Investigating queue management based on the delay scheduling (BDS), combining the time chip and priority, advances Delay-based Dynamic Priority Queue Scheduling (DDPQS) algorithm. To achieve the algorithm, sets a counter for every sub-queue entering the buffer. It dynamically adjusts the sub-queue priorities by the value of counters to achieve the effect of queue scheduling. During the research, the limitation was found that the counters are sensitive to the time chip. The paper uses threshold values to optimize the algorithm. The simulation results before and after the optimizations demonstrate that the performance of the delay and the throughput rate has significantly improved.

Key words: queue scheduling; delay; dynamic priority

0 引 言

调度[1]是解决多个业务竞争共享资源问题的有效手段。现有的调度算法根据调度规则可分为以下几类：基于区分业务优先级服务[2,3]、基于动态双向优先级[4]、比例区分算法[5]和基于离散粒子群[6]等。先进的队列调度算法都是调整队列中存放的分组，计算出触发次序，并以此为基准进行调度的控制。

为提高网络 QoS[7]，提出了基于时延的调度算法（Based-Delay Scheduling BDS）[8]。该算法中每个队列中的分组都有服务时间的限制，当分组接近其限定时间时对其分配一个较高的优先级，即越接近限定时间优先级越高。但该算法易导致系统的处理时限变大，使得分组有断续现象发生。文中在 BDS算法的基础上将时间片与优先级相结合，提出了一种新的基于时延的动态优先级调度算法（Delay-based Dynamic Priority Queue Scheduling DDPQS）。

结合 IPV4的结构[9,10]分析得知：其 TOS字节中的三位可以用来表示队列的先后触发次序，则可以表示8种触发类型，设定最低的触发次序为 0，依次增加，7为最高的触发次序，并设定计数器最大增到 8时表示子队列为空。因此将一个缓冲区等分成 8个队列。因为缓冲区中的子队列来自优先级不同的数据包，所以在分组进入队列时记录相应的优先级状态。根据 BDS思想，原本优先级低的子队列随着时间的推移提高本身得到服务的优先级，即队列的优先级不是始终不变而是随着时延动态变化的。

1 基于时延的动态优先级调度算法

1.1 优先级的制定

为了实现动态优先级算法，分别设置计数器用以记录进入缓冲区的队列，以其所属数据包的优先级作为计数器的初始值；每经过一个时间片，计数器值加 1（若计数器值大于 7，说明队列为空，不增加）；选取计

收稿日期：2010-05-26；修回日期：2010-08-16
基金项目：瑞典国家基金：中瑞国际合作 Swedish Research Links Programme（348-2008-6212）；南京市留学回国人员基金项目（No.209002）
作者简介：张登银（1964-），男，江苏靖江人，博士，研究员，博士生导师，CCF会员，研究方向为信号处理、IP网络技术。

数器值最高的子队列队头分组进行转发。假定优先级为 i(i= 0, 1, …, 7)的分组根据优先级进入子队列 i(i =0, 1, …, 7)中, 例如一个缓冲区中有 5个队列等待服务。各队列的计数器值依次为 6、4、3、2、0, 经过一个时间片, 转发计数器值为 6的分组, 而剩余队列的计数器值增加为 5、4、3、1。若此时队列 5中有新的分组到达, 即队列 5与队列 4的优先级相同时, 如何转发, 这就引入了堆的实现问题。

1.2 优先级与堆

由于本文使用的调度算法总是从缓冲区中选出并清除优先级最大值的队列, 即将分组转发。所以, 经过比较, 本文选用 "堆 "作为优先级队列存储和实现的数据结构[11]。插入节点和删除堆顶元素的平均时间代价和最差时间代价都为 O(logN)。

首先, 堆中的各个节点保存动态调整的优先级值, 即各队列的计数器值。另外, 由于在选出优先级最大的节点后, 需要从其所在的缓冲区中将此优先级对应的队列转发, 所以每个堆节点还需要保存优先级分组所进入的子队列号。可以借鉴线索二叉树的思想, 加一个标记位, 成为带标记的堆。为编程方便, 将标志位放在后面。在堆中每个节点由两部分组成: 前部分内容为优先级的值; 后部分为标记位, 记录该优先级所对应的子队列号。如图 1所示, 最后一个节点即为上节中假设的一个时间片后, 恰好队列 5有新的分组到达的情况。

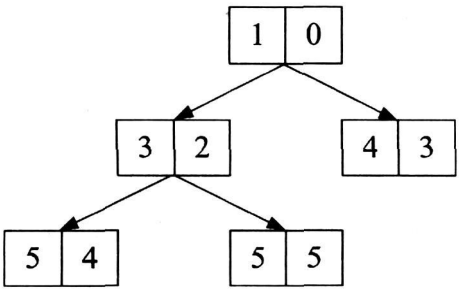


图 1　一个时间片后队列的堆实现

1.3 时间片内的调整

DDPQS的缓冲区可以使用双端口存储器实现[12]。同一个存储器里包含两组并行独立操作的相互独立的读写控制线路。本文设计新分组到来时只存储入空白区, 与已有分组在缓冲区中的地址不相同, 所以在这两个端口上进行读写操作, 一定不会发生冲突。一个时间片内, 新分组的到来与已有分组的转发可以并行处理, DDPQS算法流程如图 2所示。

如果既无新分组, 队列又为空 (即无已有分组), 则该时间片可出让给其它队列转发; 如果只有已有分组而无新分组到来, 则转发最大优先级的已有分组, 并对堆的节点进行一次调整; 当新分组到来而无已有分组时, 则该新分组可以不入缓冲区而直接转发; 而出现新分组入队列同时转发已有分组的情况时, 则需要调整两次堆结构, 因为此时并没有直接调整缓冲区的内容, 所以调整堆结构与分组的入队列 /转发可以并行的处理。因为调整堆的时间复杂度只有 O(logN), 只要堆的大小不大于分组区域, 则可以保证在分组插入或转发后就完成堆的调整。

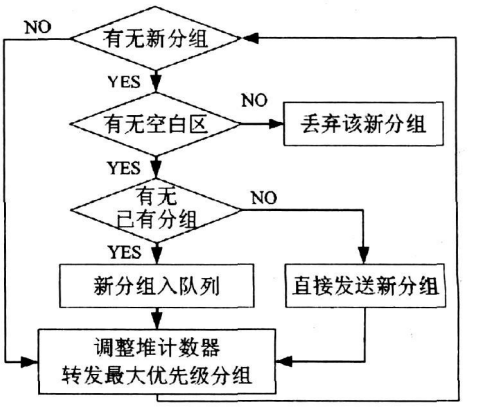


图 2　DDPQS算法流程

1.4 DDPQS算法优化

在基于优先级的调度算法中, 大量高优先级队列先后到来可能会造成其他低优先级队列长时间等待而得不到正常服务; 而在 DDPQS算法中, 随着时间片的增加, 低优先级的不停提高也可能抢占高优先级分组的服务时间。为保障不同优先级分组都能得到正常服务, 提高算法公平性, 可对 DDPQS算法进一步优化。

为了解决计数器值对时间片过于敏感的问题, 本文采用设置阈值进行区分的方法。子队列队头分组的阈值大小根据子队列分组的优先级高低以及各优先级分组的实际流量 (各子队列中存在的分组个数, 也即子队列长度)进行设置。通过这种方法, 结合了缓冲区中队列的实际使用情况, 在一个时间片后只有超过阈值的队列计数器才增加 1。设定优先级 0、1、2为低优先级, 阈值设为 30; 优先级 3、4、5为中优先级, 阈值为 20; 优先级 6、7为高优先级, 阈值为 10。若阈值、子队列优先级以及对应的分组数如图 3所示, 分组根据优先级进入对应的子队列 i(i= 0, 1, …, 7)中, 其中分组数超过预设阈值的子队列只有 1、3、6, 则经过一个时间片, 子队列 7中的队头分组由于计数器值最大而

子队例	优先级	阈值	子队列中分组数
0	0	30	32
1	1	30	28
2	2	30	29
3	3	20	21
4	4	20	19
5	5	20	17
6	6	10	11
7	7	10	9

图 3　阈值与各子队列中分组数

被转发,子队列 1、子队列 3、子队列 6的队头分组计数器值加 1,其余子队列中存在的分组个数没有超过阈值,因而其队头分组的计数器值保持不变。此过程缓冲区内各子队列计数器变化情况如图 4所示。

队列号	计数器
队列0	0
队列1	1
队列2	2
队列3	3
队列4	4
队列5	5
队列6	6
队列7	7

一个时间片后

队列号	计数器
队列0	0
队列1	2
队列2	2
队列3	4
队列4	4
队列5	5
队列6	7
队列7	

图 4 各子队列计数器变化情况

当各子队列队头分组的优先级经过一段时间的变化都改变为最高优先级(计数器值为 7)时,从高序号队列,即高优先级分组,开始调度,以保障高优先级分组的服务质量。

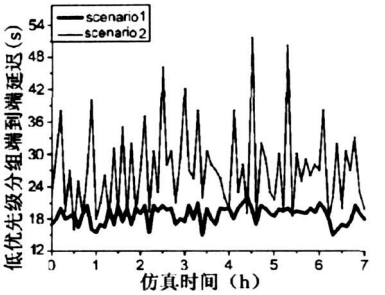
2 算法仿真与结果分析

采用 NS2仿真工具比较 DDPQS算法及其优化后的算法(简称 ADDPQS)的时延和吞吐率性能。仿真运行平台为 Pentium4 2.66 GHz CPU,512MB RAM。设置的仿真参数如下:链路速率设置为 1024 kbps,分组大小为 512 bit,缓冲区大小是 900个包,处理速率为 51200 bps。仿真结果中,scenario1(粗线)、scenario2(细线)分别对应改进前后的 DDPQS算法和 ADDPQS算法。

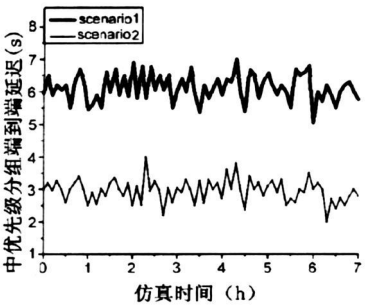
对应低、中、高三种不同优先级的分组端到端延迟仿真结果,分别如图 5所示。其中,x轴表示仿真时间,单位为小时;y轴表示延迟,单位为秒。由图可见,随着仿真时间的不断增加,DDPQS算法与 ADDPQS算法相比,低优先级的分组端到端延迟小,而中优先级和高优先级得分组端到端延迟大。

综合分析图 5中的仿真结果,scenario1中 DDPQS算法由于时间片的增加,低优先级不断提高,占用了其他中高优先级分组的正常服务时间,增加了这些分组的端到端延迟。因此 DDPQS算法对应的低优先级分组延迟小,而中优先级和高优先级分组的延迟比较大。为了保障各优先级分组都能得到公平服务,scenario2仿真了进一步完善的 ADDPQS算法,从图中可以看出,低优先级分组的延迟虽然有一定的增加,但是还是能够提供一个最小的服务速率。而相对于 DDPQS中、高优先级分组的延迟减少的程度就比较明显。

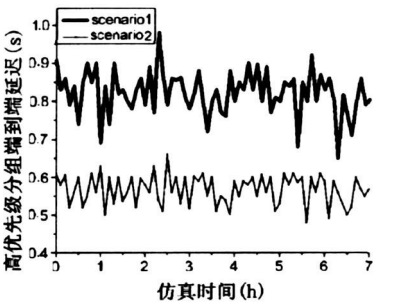
对应低、中、高三种不同优先级的分组吞吐率仿真结果,分别如图 6中所示。其中,x轴表示仿真时间,单位为小时;y轴表示分组吞吐率,单位为 kbps。由图可见,随着仿真时间的不断增加,DDPQS算法与 ADDPQS算法相比,低优先级的分组吞吐率大,中优先级的分组吞吐率小,高优先级的分组吞吐率相差不大。



(a) 低优先级



(b) 中优先级



(c) 高优先级

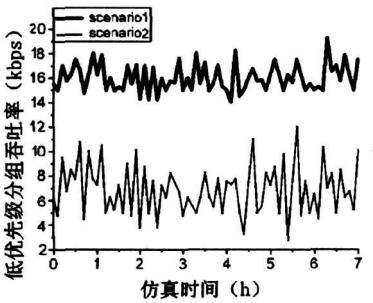
图 5 各优先级分组端到端延迟

综合分析图 6中的仿真结果,同样由于低优先级的动态变化过快,scenario1中 DDPQS算法对应的低优先级分组吞吐率大,而中优先级和高优先级分组的吞吐率要小。scenario2中 ADDPQS算法一定程度降低了优先级变化带来的对分组吞吐率的影响。由图中可以看出,虽然 ADDPQS算法降低了低优先级情况下的分组吞吐率,但是在一定程度上提高了高优先级分组吞吐率,而且很明显的提高了中优先级分组的吞吐率。

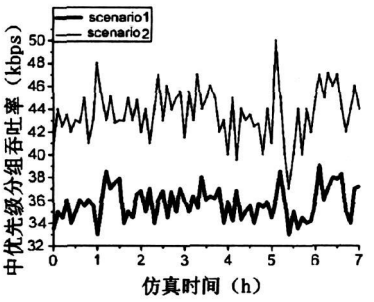
3 结束语

从技术实现和仿真的角度对多优先级队列管理机制进行了研究,提出了一种基于时延的动态优先级调度算法(DDPQS)。如果要将 DDPQS机制完全应用于网络,还需要进行进一步的研究,包括:(1) DDPQS算法优化虽然考虑了公平性,但对于稳定性、复杂性方等性能还有待进一步研究。(2) DDPQS算法优化只能通过增加缓冲区容量才能大幅提高吞吐率,而这就会

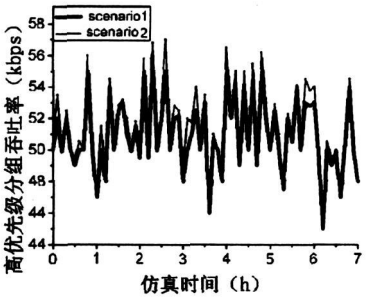
带来端到端延迟增加的问题。所以吞吐率和延迟两者之间如何折衷以获得更好的 QoS性能, 也是需要进一步研究的课题。



(a) 低优先级



(b) 中优先级



(c) 高优先级

图 6　高优先级分组吞吐率

参考文献:

[1] de Pablo D A L. On scheduling models: An overview[M] // Computers & Industrial Engineering. Troyes: [s n], 2009: 153－158.

[2] Liu Hongtao, Cheng Lianglun. Priority-Based Service Differentiation Scheme for Medium and High Rate Sensor Networks[M] // Communication Software and Networks. Singapore: [s n], 2010: 392－395.

[3] 杨志军, 赵东风, 丁洪伟, 等. 两级优先级控制轮询系统研究[J]. 电子学报, 2009, 37(7): 1452－1456.

[4] 龚　跃, 张真真, 黄小珂, 等. 基于动态双向优先级的任务分配与调度算法[J]. 计算机应用, 2009, 29(4): 1131－1134.

[5] Vukadinovic V, Karlsson G. Video streaming performance under proportional fair scheduling[J]. IEEE Journal on Selected Areas in Communications, 2010, 28(3): 399－408.

[6] 刘环宇, 侯秀萍. 基于离散粒子群算法的工作流任务调度研究[J]. 计算机技术与发展, 2010, 20(5): 88－91.

[7] 杨　勇, 王雪晶, 陈良臣. QoS在 IP中的研究和应用[J]. 计算机技术与发展, 2007, 17(5): 33－36.

[8] Dovrolis C, Stiliadis D, Ramanathan P. Proportial differentiated services: Delay differentiation and packet scheduling[J]. IEEE/ACM Trans on Networking, 2002, 10(1): 12－26.

[9] 沈庆伟, 张　霖. 基于隧道的 IPv4/IPv6过渡技术分析[J]. 计算机技术与发展, 2007, 17(5): 171－176.

[10] 庄正松, 吴家皋, 吴清亮, 陈国凤. 互联网基本服务 IPv4/IPv6过渡的研究与实现[J]. 计算机技术与发展, 2006, 16(8): 13－15.

[11] 滕　腾, 李龙澍. 基于大值堆的自调整粗粒度并行遗传算法模型[J]. 计算机技术与发展, 2007, 17(10): 105－108.

[12] 李　震, 周　玮. 一种基于双口 RAM的环形数据缓存系统[J]. 计算机技术与发展, 2010, 20(1): 201－204.

(上接第 161页)

产品质量、减少开发成本、提高软件可靠性和可维护性, 这些都是军用指控软件迫切需要改善的焦点, 指控软件复用技术必将使军用软件产业真正走上工程化的发展道路, 军工科研院所应加大软件构件复用技术的研究, 为实现软件产业跨越式发展奠定良好的基础。

参考文献:

[1] 杨芙清, 梅　宏, 李克勤. 软件复用与软件构件技术[J]. 电子学报, 1999, 27(2): 68－75.

[2] 杨芙清, 王千祥, 梅　宏, 等. 基于复用的软件[J]. 中国科学, 2001, 31(4): 367－371.

[3] 彭思鹏, 孔祥营. 构件技术与指控系统软件开发[J]. 舰船电子工程, 2001(5): 21－25.

[4] 王　刚, 李　锋. 基于可重用构件库的机载火控软件的开发与管理[J]. 电光与控制, 2003(1): 40－45.

[5] 黎　娅. 基于构件的软件复用技术应用研究[D]. 重庆: 重庆大学, 2008.

[6] 张海藩. 软件工程[M]. 北京: 人民邮电出版社, 2002.

[7] 郑　红, 李师贤. 可重用的分布式软件构件模型分析[J]. 计算机工程与应用, 2002(15): 68－71.

[8] 赵池龙, 杨　林, 陈　伟, 等. 实用软件工程[M]. 北京: 电子工业出版社, 2006.

[9] 麦克克劳埃. 软件复用技术: 在系统开发过程中考虑复用[M]. 北京: 机械工业出版社, 2003.

[10] OMG. OMG unified modeling language specification[EB/OL]. 2005－02. http://www.omg.org.

[11] OMG. The common object request broker: Architecture and specification[EB/OL]. 2005－05. http://www.omg.org.

[12] Microsoft DCOM technical overview[EB/OL]. 2006－07. http://www.microsoft.com/ntserver/library/dcomtec.exe.