基于 Intel 网络处理器的路由器队列管理：设计、实现与分析

林 闯 周文江 李 寅 郑 波 田立勤

（清华大学计算机科学与技术系 北京 100084）

摘 要 通过设计并实现的基于 Intel 网络处理器(IXP1200)和相对区分服务模型的队列管理服务模块和一种新的缓冲管理算法 DPBS (Dynamic Partial Buffer Sharing)，研究了系统同步、线程的任务分配、队列管理基本操作等几个关键问题.

关键词 网络处理器；缓冲管理；分组调度；队列管理；相对区分服务

中图法分类号 TP393

Building Queue Management Module Based on Intel Network Processor: Design, Implementation and Analysis

LIN Chuang ZHOU Wen-Jiang LI Yin ZHENG Bo TIAN Li-Qin

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract Buffer management and packet scheduling, integratedly called queue management, is one of the key mechanisms for QoS provision, and it is also an important subsystem of router. Network processor is a kind of programmable processor performing network computing with special design and optimization. It has the property of combination of the flexibility of software and the high performance of hardware, indicating one trend of technology development. The key characteristics of network processors are specialized instruction set optimized for network computing and the hardware-level parallel processing. The design and development of networking software using network processors is an emerging field that offers numerous challenges and opportunities. This paper provides the design and implementation of queue management module that uses Intel's network processor and follows the relative differentiated service model. The architecture, the manipulation process, especially some key design issues such as the system synchronization, the task assign to threads and basic manipulations of queue management are discussed in detail. And a new buffer management algorithm called DPBS (Dynamic Partial Buffer Sharing) is also designed and implemented.

Keywords network processor; buffer management; packet scheduling; queue management; relative differentiated services

收稿日期：2002-07-15；修改稿收到日期：2002-11-07. 本课题得到国家“九七三”重点基础研究发展规划项目(G1999032707)、国家“八六三”高技术研究发展计划项目(2001AA112080)、国家自然科学基金(90104002,60173012)和 Intel 公司 IXA 大学研究项目(9077)资助.
林 闯，男，1948 年生，教授，博士生导师，主要研究领域为计算机网络、服务质量控制、系统性能评价、随机 Petri 网、逻辑推理模型等. E-mail: chlin@tsinghua.edu.cn. **周文江**，男，1975 年生，硕士研究生，主要研究方向为计算机网络、分组调度算法和系统性能评价. **李 寅**，男，1979 年生，硕士研究生，主要研究方向为计算机网络、缓冲管理算法和系统性能评价. **郑 波**，男，1978 年生，博士研究生，研究方向为计算机网络、随机 Petri 网和系统性能评价. **田立勤**，男，1970 年生，硕士研究生，研究方向为计算机网络、随机 Petri 网和系统性能评价、工作流模型.

1 引　言

服务质量(Quality of Service,QoS)是网络发展的核心技术之一,也是当前的一个研究热点.服务质量控制,是指网络能够提供有保证的、可控制的、可预测的数据传输服务,满足不同用户的应用需求①.衡量服务质量的指标有带宽、端到端时延、时延抖动、分组丢失率等.现有的尽力而为(best-effort)的网络服务模型不能有效地提供服务质量支持,为此IETF(Internet Engineering Task Force)以及一些研究者先后提出了几种新的服务模型:集成服务(IntServ)[1]和区分服务(DiffServ)[2],后者又分为绝对区分服务(absolute differentiated services)和相对区分服务(relative differentiated services)[3].目前,区分服务,尤其是相对区分服务是一个重要的研究方向[3,4].这些服务模型分别有许多机制,比如:集成服务的资源预留(信令协议 RSVP)、接纳控制等,区分服务的流量整形/管制、分组标记等.其中不同服务模型所共有的部分,也是一个核心的部分是缓冲管理和分组调度,本文统称队列管理.

缓冲管理实现对缓冲区的管理,研究的主要内容是缓冲区如何分配和当缓冲区占用率到达一定程度时如何选择分组进行丢弃,所影响的性能参数主要是分组丢失率.而分组调度则是对链路带宽的管理,是指按照一定的规则来决定从多个等待队列中选择哪个分组进行发送,它影响的主要性能参数包括带宽分配、时延和时延抖动.目前已经有许多关于缓冲管理和分组调度的算法[3~5,6~14].

队列管理是路由器的一个重要的子系统,因此,研究队列管理的设计和实现技术是路由器开发的一个关键部分[15,16].

从传统意义来讲,网络设备一般基于 CPU 或 ASIC.也就是说,队列管理算法的实现一般是基于 CPU 的软件实现[17~19],或者是基于 ASIC 的硬件实现[20,21].然而,基于 CPU 的软件实现虽然灵活,但很难获得高性能,而基于 ASIC 的硬件实现虽然性能高,但灵活性很差,而且费用高②.这使得传统的实现机制不能很好地满足当前网络高速度和多样化业务的发展趋势.因此有必要采用更先进的实现平台,研究新的实现技术.

网络处理器(network processor)就是一种很好的选择.通过良好的体系结构设计,网络处理器将软件的灵活性和硬件的高性能特点结合在一起,是继 ASIC 之后一个新的技术发展方向.目前世界上有很多公司和大学在研究网络处理器技术③.网络处理器的一般特点为:专门针对网络数据处理而优化了的指令集和硬件级并行处理(多个片内处理器、多硬件线程、多组高速总线、多组存储器、多个 I/O 接口单元等).

基于网络处理器设计开发网络软件系统是一个新兴领域,面临许多技术挑战,比如多个线程之间的同步问题.因此,基于网络处理器的队列管理服务模块设计的关键是如何有效地管理系统多线程以提高任务处理的并行性,以及如何动态编程以进行灵活的系统配置[22,23].具体来讲,需要研究下列问题:如何解决系统同步问题;如何管理系统资源;如何对每个线程进行合理的任务分配;如何设计合理的模块化结构和清晰的接口等.

Intel 公司的 IXP 系列网络处理器是目前市场上的一种典型产品,是 Intel 公司 IXA(Internet eXchange Architecture)架构的核心,其它很多公司的产品都有与之相似的结构[24,25].本文以 Intel 的 IXP1200 为平台并参照相对区分服务模型进行了队列管理模块和一种新的缓冲管理算法 DPBS(Dynamic Partial Buffer Sharing)的设计,并重点研究了几项关键技术.

本文在讨论体系结构(第 2 节)、系统处理基本流程和 DPBS 算法(第 3 节)的基础上,重点分析了系统同步、线程的任务分配等一些相关的关键技术(第 4 节)和系统的实现性能(第 5 节),最后,对全文作了总结.

2 体系结构设计

Intel 网络处理器 IXP1200 的基本结构如图 1 所示.包括 1 个通用处理核心 StrongArm、6 个 RISC 结构的专用可编程微引擎、1 个 IX 总线接口单元 FBI(Fast Bus Interface,连接 MAC 等网络接口)、一个 SRAM 接口单元(外接 SRAM 存储器)、1 个 SDRAM 接口单元(外接 SDRAM 存储器)和一个 PCI 总线接口单元.每个微引擎具有独立的指令存储器、控制器以及寄存器,能支持 4 个硬件线程,并且线程之间可进行零开销切换.

① IP QoS FAQ. Technical Reports. http://www.cs.tut.fi/~yk/faq-qos.pdf

② http://user.gru.net/darusdp/cise/cda5155paper.html

③ http://www.npforum.org
Husak D. Network processors: A definition and comparison. C-Port Corporation, Technical Reports, 2000
http://www.linleygroup.com/npu/
http://www.networkprocessors.com

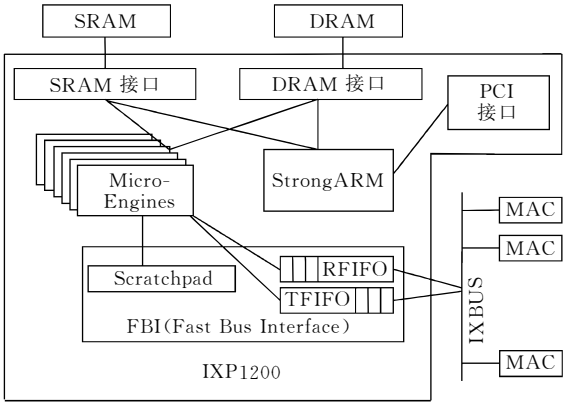


图 1 IXP1200的硬件结构

FBI 中含有两组缓冲寄存器 RFIFO 和 TFIFO，分别用于从/到网络接口的数据接收/发送. 每组含有 16 个 64 字节的寄存器单元(element)，每个寄存器单元可单独编址进行访问. 另外 FBI 中还有一组 4KB 的片内 ScratchPad 存储器.

2.1 软件体系结构

队列管理是软件路由器的一个重要的子系统. 整个路由器是一种模块化结构，划分为两个部分：控制平面(control plane)和数据平面(data plane)，两个平面通过一组通信协议进行通信，相互协作完成网络节点的完整功能[26,27]. 在 IXP1200 系统中，StrongArm 负责控制平面的处理，微引擎负责数据平面的处理.

控制平面运行操作系统，完成诸如路由协议处理、系统管理等计算性较强的“慢通道”功能. 有时也完成复杂分组(比如 IP 头部可选字段)的处理. 在数据平面，存在着一个或多个服务组件构成可扩展服务层来执行每个分组的“快通道”转发处理，比如分组头有效性检验、路由表查找、TTL 修改、分组分类、缓冲管理、分组调度等. 数据平面从网络接收分组，分组经过转发处理(或被重定向到控制平面)之后，再发送到网络中去.

2.2 模块接口

每一个服务组件构成一个软件模块，该模块的接口包含三部分：输入数据、输出数据、控制/配置信息. 其一般化描述如下：

模块输入：源数据的信息(存放位置、用以描述数据的标签、类型、值等)、操作者的信息(处理器位置、类型、编号等)、操作规则(功能配置、规则数据库的位置等)、同步信息.

模块输出：处理后的数据信息(数据处理后的存放位置、用以描述数据的标签、类型、值等)、操作者的信息(处理器位置、类型、编号等)、操作结果(错误信息等)、同步信息.

2.3 系统资源分配

为简化设计，本系统基本采用静态分配的资源管理模式. 将系统配置为支持 8 个 10/100Mbps 和 1 个 1Gbps 以太网端口，但本文讨论以实现 8 个 10/100Mbps 为主.

微引擎/线程/FBI 缓冲寄存器：用 2 个微引擎负责 8 个 10/100Mbps 端口的数据接收(每个端口静态对应一个 FBI 接收缓冲寄存器 RFIFO element，每一个物理端口都由一个专门的硬件线程进行接收服务)，用 1 个微引擎负责 8 个 10/100Mbps 端口的数据发送(每个端口静态对应一个 FBI 发送缓冲寄存器 TFIFO element，一个硬件线程循环服务两个端口). 如图 2 所示.

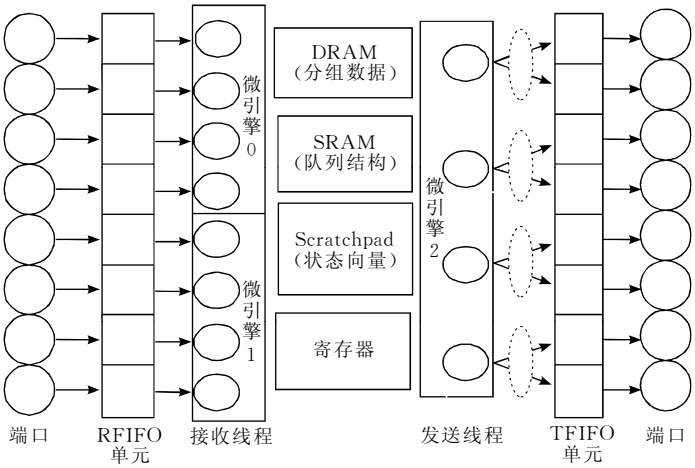


图 2 系统资源分配

IXP1200 系统中有多种存储部件，根据其不同的特点，分别用于不同的存储目的.

寄存器(包括通用寄存器和传输寄存器). 速度最快，数量最少. 用于同一微引擎内线程间的通信、

临时变量数据.

ScratchPad. 速度较快,数量较少. 用于微引擎间的通信、中间结果暂存、意外处理结果纪录.

SRAM. 速度一般,数量较大. 存放路由查找表格、分组和队列的描述信息、微引擎间的通信邮箱、微引擎和 StrongArm 间的通信管道等.

DRAM. 速度最慢,数量最多. 存放路由转发信息表,分组数据(Payload)等.

2.4　队列结构

系统采用输出排队策略,即每个输出端口对应多个逻辑队列,IP 分组按照其服务质量要求选择相应的队列. 各个逻辑队列对应了不同的时延优先级,在每一个逻辑队列内部,分组又按照丢弃优先级分成多个子类. 采用输出排队的原因如下:(1)输出排队最适合进行 QoS 控制[28,29];(2)硬件系统中不存在交换机构,所有端口共享 IXBUX 总线,不存在加速比问题.

如图 3 所示,该队列结构由下列部分组成:

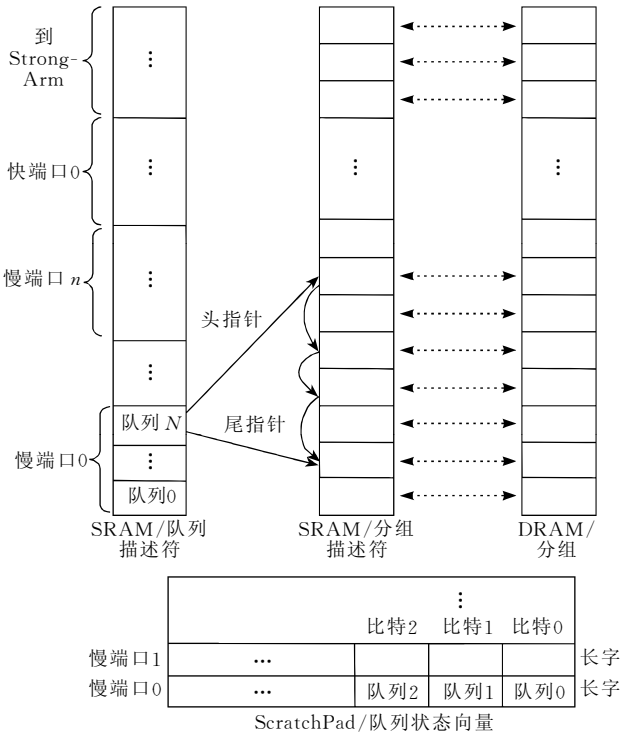


图 3　队列结构

SRAM 分组描述符. 存放分组的基本信息(比如分组长度)和分组调度算法参数(比如分组到达时间戳). 未分配(空闲)的分组描述符构成堆栈结构;已分配的分组描述符构成链表结构,形成逻辑队列.

DRAM 分组缓冲区. 把 DRAM 的某个地址连续区域分成固定大小(2KB)的页,每一页存放一个分组(整个分组,确切地说,是以太网帧),并静态对应 SRAM 中的一个分组描述符.

SRAM 队列描述符. 存放每个队列的基本信息(比如队列长度)、缓冲管理算法参数(比如分组丢失率)和分组调度算法参数(比如队列的权值 weight).

SRAM 队列结构. 一系列队列描述符组成数组,构成系统完整的队列结构.

Scratchpad 队列状态向量. 存放队列的状态(是否为空,每一比特位对应一个队列). 这一结构主要是为了提高性能.

每端口所对应的逻辑队列数量可以配置.

3　系统处理基本流程

系统把分组(确切地说是以太网帧)分为 64 字节长的单元(最后一个单元可能会小于 64 字节),称为 MAC-Packet,简称 MP. 第一个 MP(SOP)、最后一个 MP(EOP)和中间的 MP(MOP)分别需要不同的处理.

3.1　输入处理

在输入端,接收线程不停地检查 MAC 端口的状态. 当有分组/MP 到达时,(1)如果是 MOP,则简单地将其存入预先分配的 DRAM 缓冲区中. (2)如果是 SOP,则进行分组头校验以及其他相应处理,并进行路由表查找和分组分类. 通过路由表查找,确定分组转发的物理端口号. 需要本地处理的分组被重定向到 StrongArm 中进行处理;而需要转发到下一节点的分组通过分类,确定将要进入的队列编号和丢弃优先级. 以上数据保存在全局寄存器中. (3)如果是 EOP,则根据缓冲管理算法和上述分类结果进行分组丢弃判断. 如果分组不需要被丢弃,则被加入队列.

分组分类. 一般来讲,分组头中有五个字段域可供分类:IP 地址(源、目标地址)、协议类型和传输层端口号(源、目的端口)[30]. 可以根据需要从一维(一个字段域)到五维分类. 当然,维数越多,处理速度越慢.

缓冲管理. 缓冲管理算法有三种配置:Tail-drop,SPBS(Static Partial Buffer Sharing)[6]和 DPBS. Tail-drop 是最简单的算法,即当队列满的时候不加选择地丢掉所有新到达的分组. 而 SPBS 则对分组加以区分:给不同的数据流汇聚(类)分配不同的丢弃优先级,并为每个类维护一个丢弃阈值(队列长度阈值). 对于某个类,只有当当前的队列长度大于其丢弃阈值时,属于该类的新到达的分组才被丢弃. 丢弃阈值是静态配置的,在系统运行过程中保持不变. 而 DPBS 算法则通过动态调整丢弃阈值的大小,可以使得各个类的分组丢失率保持较为恒定的比率,实现更为精确的丢弃控制.

DPBS 算法. 为每一个服务类 Class-i 维护一个计数器 C_i,记录在某时间段内该服务类的分组丢弃个数. 当该计数器到达某个界值 K_i 时,则把该服务类的丢弃阈值 T_i 增加某个量值 TH_i,同时把次优先级服务类 Class-$(i-1)$ 的丢弃阈值 T_{i-1} 减小某个量值 TL_i. 当然对于最低优先级服务类 Class-1 和最高优先级服务类 Class-n,只需要分别调整 T_1 和 T_{n-1}.

3.2 输出处理

在输出端,发送线程首先计算将要服务的物理端口号(因为两个物理端口竞争一个发送线程),然后检查端口的队列状态. (1)如果该端口无数据要发送,则跳过(skip)该端口继续检查下一端口. (2)如果该端口有数据要发送,则检查上一个分组的最后一个 MP(EOP)是否已发送. (a)如果没有,则继续发送该分组的下一个 MP;(b)如果已发送,则根据分组调度算法从多个队列中选择一个分组,发送该分组的 SOP.

分组调度. 有多种分组调度算法可以配置:PQ (Priority Queuing), RR (Round Robin), WRR (Weighted Round Robin)[8], DRR (Deficit Round Robin)[9]. PQ 算法给不同的队列分配不同的调度优先级,并且每次调度具有最高优先级的队列直到该队列为空. 这样保证重要的分组得到最好的服务. RR 算法只是简单地循环调度每个队列,以获得某种程度的公平性. WRR 算法在 RR 的基础上给队列分配不同的权值,该权值对应每次调度发送的分组数. 这样,在保持某种程度公平性的基础上,使得权值比较大的队列能得到更好的服务. DRR 算法则考虑了 IP 网络中分组长度变化这一特点,以字节为单位为每个队列分配一个带宽配额,该配额的比例对应队列服务速率的比例. 这样,DRR 算法能获得比 WRR 更好的公平性和更精确的控制.

4 几个设计问题

基于网络处理器的队列管理模块设计的目标之一是在保持灵活性的同时能够获得较高的系统性能. 由于网络处理器本身硬件级并行处理的技术特点和本系统采用的输出排队的策略,使得系统同步、系统资源管理、线程的任务分配以及其它涉及分组发送的问题对系统性能具有非常大的影响. 另外,结合网络处理器的硬件特点,总结队列管理的基本操作,对于进行系统优化也有非常重要的意义. 因此,本节着重讨论了一些相关的关键技术问题.

4.1 系统同步

系统同步问题存在的原因:

(1)系统多线程. IXP1200 具有先进的硬件多线程系统,不同的线程可以负责不同的任务处理,通过硬件级并行化来提高系统吞吐率,提高系统性能. 然而,多个线程之间并不总是相互独立的,有时候具有某些相互制约关系. 主要体现在服务的时序关系和数据一致性两个方面.

(2)服务的时序关系. 有些服务的执行需要一定的条件,比如另外一个任务的处理结果作为本任务的输入. 有些服务的执行要求具有一定的先后关系,比如同一个分组的多个 MP 的发送必须保持严格的先后关系.

(3)数据一致性. 数据一致性主要是由于资源共享引起的. 当多个具有读取/修改权限的线程共享同一个数据结构时,必须保证多个线程对数据的修改不造成冲突. 这需要同步控制,典型的是通过加锁或信号量机制.

系统同步问题的影响.

多线程设计的目的是任务处理的并行化,从而获得很高的系统性能. 而同步控制恰恰相反,目的是通过限制某些任务的执行,保持服务的时序关系和数据一致性. 两者具有一定的矛盾性. 因此,解决同步问题,需要兼顾系统运行的正确性和性能. 即在保证正确的时序关系和数据一致性的基础上,提高系统性能.

解决方案.

(A)为保证系统处理正确性,设计严格的同步控制. 由于篇幅所限,本文主要讨论缓冲管理和分组调度模块之间的同步控制解决办法.

(B)应尽量避免不必要的同步关系,以提高系统吞吐率. 发送 MP 时对于 TFIFO 的软件管理模式就体现了这一点,在 4.3 节中将详细讨论.

缓冲管理和分组调度之间进行同步控制是为了解决由于资源共享而引起的数据一致性问题. 所涉及的共享资源有 3 个:分组描述符、队列描述符和队列状态向量.

(1)当分组入队列或出队列时,需要修改队列描述符和分组描述符中的某些信息. 因此必须保证当缓冲管理模块或分组调度模块对描述符进行修改时,该数据不能被其它模块读取. 队列描述符和分组描述符保存在 SRAM 中,而系统提供了 SRAM 的硬件加锁机制. 因此,可以利用硬件加锁来进行同步控制. 当某模块需要修改描述符时,在读取数据的同时将该描述符加锁(read_lock 指令);而当写完数据后,将该描述符解锁(write_unlock 或 unlock 指令). 在加锁和开锁期间,任何其它模块不能再读取该描

述符.如果有读请求到达,则被阻塞.

(2)队列状态向量用来指示某队列是否为空.当某空闲队列有分组到达时,将向量中该队列对应的比特位置 1;当某队列最后一个分组被调度时,将向量中该队列对应的比特位清 0.分组调度模块根据队列状态向量来判断队列是否为空.因此必须保证向量的某比特位为 1 时,对应队列中确实存在着有效分组.队列状态向量也是被缓冲管理和分组调度模块共享,并存在着比(1)更复杂的数据一致性问题.队列状态向量被保存在 ScratchPad 中(为了保证较快的访问速度),而 ScratchPad 没有提供硬件加锁机制.因此只能利用软件的方法进行控制.比如通过调整指令代码的执行位置,把对状态向量读写的操作安排在 SRAM 的加锁期间,当然这样会在某种程度上降低系统效率.

综上所述,确定缓冲管理模块和分组调度模块对分组描述符、队列描述符和队列状态向量的操作规则如下:

(1)在修改队列描述符之前(读取数据时),对该描述符加锁;修改结束后(写入数据时)解锁.为提高效率,上锁的时间尽量短,即加锁和解锁之间安排尽量少的指令.

(2)在缓冲管理模块中:当某空闲队列有分组到达时,分组描述符和队列描述符的更新(写)操作要早于队列状态向量的更新(写)操作,以保证状态向量中对应比特位为 1 时,队列描述符中确实存在有效的分组的相关数据.

(3)在分组调度模块中:当某个队列最后一个分组被调度之后,状态向量要尽快更新(相应比特位清零).

(4)根据以上三点,缓冲管理模块中,队列状态向量的更新应该在队列描述符解锁操作之后,分组调度模块中,状态向量的更新应该在队列描述符解锁操作之前.

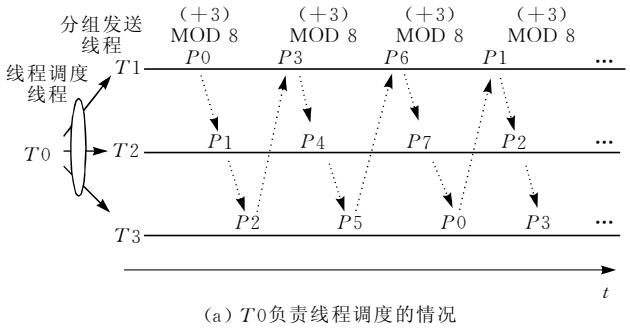
(5)由于分组描述符的读写总是在已经读取队列描述符并计算出分组描述符地址之后,因此不必专门对分组描述符做同步控制.

4.2　线程分配

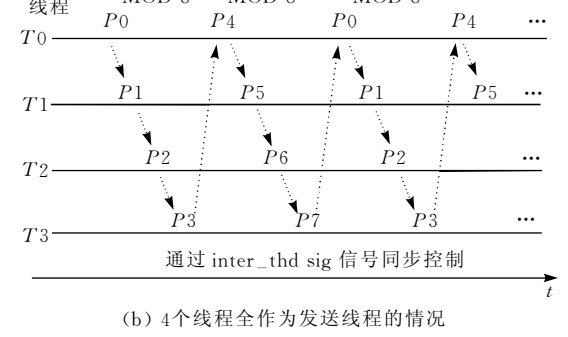
IXP1200 的每个微引擎支持 4 个硬件线程,通过线程之间零开销的切换可以在某种程度上隐藏内存访问的时间延迟,以并行技术提高系统的性能.

在发送端,由于用一个微引擎负责 8 个 10/100Mbps 端口的分组发送,因此一个线程需要服务多个端口.在传统的软件开发模式(IXP1200 参考设计)中,专门分配了一个线程作为调度线程,即决定另外 3 个发送线程分别负责哪一个端口的分组发送.

然而在本质上,调度线程的开销属于内部管理开销,对分组的转发没有直接意义,因此应使这种开销最小化.经过分析发现,专门用一个线程作为调度线程是不必要的,在某种程度上浪费了处理器资源.比如从调度线程的调度结果来看,是使得每个发送线程所服务的物理端口号按照加 3 再对 8 求模的规律进行变化(当然,也可以有其它的调度策略).实际上,这种调度策略的执行可以交给发送线程自己,即四个线程都作为发送线程,并且服务的物理端口号按照加 4 再对 8 求模的规律进行变化.这样提高了处理器资源的利用率,进而提高系统性能.图 4(a)表示线程 $T0$ 作为调度线程而其余 3 个线程作为发送线程模式下的端口服务序列,图 4(b)表示调整结构后的服务序列.可以看出,第二种情况下,各个端口得到更快的服务.



(a) $T0$负责线程调度的情况



(b) 4个线程全作为发送线程的情况

图 4　线程调度

4.3　发送缓冲 TFIFO 的管理

在发送 MP 时对于 TFIFO 的软件管理模式也涉及到同步问题.传统的设计模式中用线程间的同步控制来保证各个线程的时序关系,而经过分析发现这种同步控制是没有实际意义的.在我们的新系统中,通过解除这种同步控制,既保证了发送的正确性,又提高了发送速度.

IXP1200 通过 FBI 中的状态寄存器 XMIT_

PTR 来指示当前 TFIFO 中哪一个 element 的数据将被发送到 IX 总线,并且采用轮循的服务方式.一般来讲,一个 TFIFO element 必须满足 3 个条件才能被服务:控制/状态寄存器已写;有效标记位(Valid flag)已置位;XMIT_PTR 已指向该 element.如果有效标记位被置 1,则根据控制寄存器中的信息把 element 中的数据发送到 IX 总线上(控制寄存器指示该 element 中有数据要发送),同时 XMIT_PTR 加 1,去处理下一个 element;或者只是简单地跳过当前 element(控制寄存器指示该 element 中无数据要发送),同时 XMIT_PTR 加 1,去处理下一个 element.当 XMIT_PTR 等于 15 时,则返回到 0 继续下一次循环.

提高数据发送速度的关键在于提高 XMIT_PTR 的轮转速度.

在传统的软件设计模式中,系统通过线程间通信信号 inter_thd_sig 来进行同步控制,使得发送线程满足某种时序关系,如图 4(b)所示.这在很大程度上限制了 XMIT_PTR 的轮转速度.比如,当某个 element 准备好数据并且写入控制/状态寄存器中的信息后,服务该 element 的线程必须被阻塞(blocked)等待前一个线程的 inter_thd_sig 信号.只有当接收到这一信号并且 XMIT_PTR 到达时,有效标记位才能被置 1,从而该 element 被继续服务,同时发送一个 inter_thd_sig 信号给下一个线程.这样,当某个线程等待 inter_thd_sig 信号而被阻塞时,其它线程有可能由于访问存储器也被阻塞,从而使得处理器处于空闲状态,利用率下降.经过统计发现,处理器(负责发送的微引擎)存在 15%~20%的空闲状态.

实际上,对于本系统的 10/100Mbps 端口应用来讲,一个端口只对应一个 TFIFO element,各个物理端口相互独立,因而相应 TFIFO element 之间不存在服务时序关系.这样,发送线程之间的同步控制是没有必要的.这与千兆口不同,由于一个千兆口对应多个 TFIFO element,为保证 MP 发送的先后顺序,这些 element 需要一个服务的时序控制,因而相应的发送线程需要同步控制.

解除发送线程间不必要的同步关系之后,四个发送线程的服务序列如图 5 所示,可见,端口被服务的速度有很大提高.

然而,由于 XMIT_PTR 对 TFIFO 轮循服务的硬件管理模式的限制,实际上各个发送线程之间仍然隐藏着某种服务的时序问题.这是软件的方法所不能解决的.

当 XMIT_PTR 指向某个 element 时,由于该

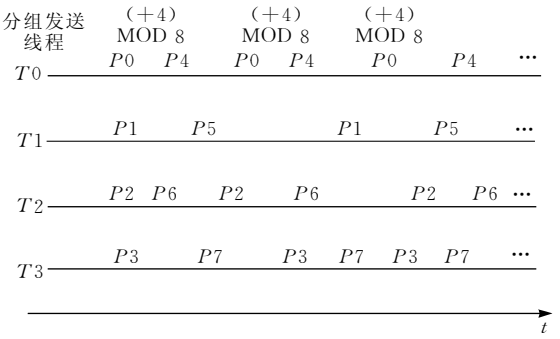


图 5 没有同步控制的线程分配

element 数据未准备好,使得 valid flag 未置 1,则 XMIT_PTR 停下来等待,直到 valid flag 被置 1.在这种情况下,即使后面的某个 element 已准备好数据并且 valid flag 已经置 1,则由于 XMIT_PTR 未指向该 element,数据也不能被发送,服务该 element 的发送线程被阻塞等待.

一种可行的硬件解决方案是用一个 FIFO 队列代替 XMIT_PTR 的轮循服务方式:当某个 element 准备好数据及控制信息后,其编号被写入 FIFO 队列.系统不断从该 FIFO 队列中读取 element(编号)进行服务.

4.4 队列管理的几个基本操作

乘/除法运算.由于网络处理器是针对网络信息处理而优化了的专用处理器,在设计上采用了一套专用的精简指令集,没有提供乘/除法指令.如果需要,乘/除法运算只能由软件来完成,即由多条整数加/减/移位指令来软件模拟,但一般需要花费几十个指令周期,其性能是非常低的.而实际上,服务质量控制本身是一种计算性较强的服务,乘/除法运算对于缓冲管理和分组调度而言,是非常重要的基本操作.比如 RED[12] 需要用乘法计算平均队列长度,用除法计算分组丢失概率.在一类分组调度算法 PFQ (Packet Fair Queuing)[13] 中,除法用来计算每个分组的服务时间.WTP(Waiting Time Priority)算法[3] 用除法计算调度优先级.因此,在网络处理器中提供硬件乘/除法的功能单元,对于提高队列管理子系统的性能,是非常重要的.

查找/排序.查找/排序,更确切地说,从数据列表中查找最大或最小值,是分组调度的另一个重要的基本操作.基于优先级的调度算法是优先级的排序,基于轮循的调度算法是逻辑队列编号的排序.这两者的排序关系一般是固定的,因此复杂度为 $O(1)$.而另外许多算法都具有 $O(\log N)$ 的复杂度,其中 N 代表队列数量.PFQ 算法是服务时间的排

序(SFF:最小虚拟完成时间;SSF:最小虚拟开始时间;或 SEFF:最小合法虚拟完成时间)[21]. EDF 及其变种[5,14]是排对时延的排序. 又如,URR[11]是关于紧急程度参数的排队,WTP[3]是关于分组等待时间优先级的排队. 一般来讲,数据列表存储在片外存储器(比如 SRAM)中,或片内存储器(比如 Scratch-pad,如果其容量足够大的话)中. 但不管怎样,都需要大量的内存访问操作. 对于大多数算法来讲,$O(\log N)$的复杂度会使得内存访问成为一个性能瓶颈. 因此,研究在网络处理器体系结构上的快速排序方法,或提供某种硬件支持,对于提高分组调度的执行速度是非常重要的.

5　性能评价

队列管理子系统的实现性能应该从两个层次来评价:局部性能和系统性能.

局部性能是指有关队列操作的性能,体现为缓冲管理和分组调度算法的控制结果,比如不同队列/服务类的分组丢失率、排队时延、带宽、队列长度等;系统性能是指整个路由器系统的性能,体现为系统中所有服务模块的综合处理性能、算法实现的复杂度等,可衡量指标为系统吞吐率等. 我们通过 IXP1200 SDK 1.2 的模拟器进行了性能分析

模拟实验基本配置:8 个全双工 100Mbps 速率端口,端口 MAC 接收和发送缓存均为 256 字节,每端口对应 8 个逻辑队列,队列容量为 32 个分组.

5.1　局部性能

局部性能测试配置:(1)数据流配置. 8 个端口接收到的数据只向 4 个端口轮流发送(本端口除外);(2)数据源分组长度为 64～1500 字节随机;(3)PQ 算法. 队列 1 优先级最低,队列 8 优先级最高,其余队列优先级随着编号的增大逐次提高;WRR 算法. 各个队列权值与队列编号相同,即队列 1 权值为 1,队列 2 权值为 2,依此类推;DRR 算法:队列 1 带宽配额为 1540 字节,其余队列随着编号的增加其带宽配额逐步增加 220,队列 8 配额为 3080;(4)分类结果使得队列 1 中没有分组,一直为空.

局部性能测试结果见图 6(纵坐标代表各个性能参数,图 6(a)～6(d)横坐标代表逻辑队列编号).

局部性能测试结果分析:

PQ 算法. 最重要的分组能得到最好的服务,然而在高优先级队列源源不断有分组到达时,低优先级的队列容易被"饿死",即很长时间内得不到服务. 因而公平性很差.

RR 算法. 循环地服务每个队列,但由于分组长度不固定,使得长分组队列可能比短分组队列得到更多的服务,因而其公平性特点受到很大限制.

WRR 算法. WRR 可以比 RR 更容易控制带宽在不同队列的分配,但仍然存在由于分组长度变化而带来的公平性差的弱点.

DRR 算法. 它很好地解决了上述公平性问题,而且能实现比 WRR 算法更精细的队列间的服务速率分配.

以上结果是在输出端口为重负载情况下测量得到的,因而分组丢失率很高. 而在轻负载情况下,某些队列的服务速率等于分组的到达速率,因而算法的控制结果几乎相同(丢失率为零,带宽相等). 由于篇幅所限,本文未列出其曲线图.

5.2　系统性能

系统性能测试配置:(1)数据流配置. 8 个端口接收到的数据向 8 个端口轮流发送(本端口除外);(2)数据源分组长度配置有 8 种:固定长度 64 字节、固定长度 65 字节、固定长度 128 字节、固定长度 129 字节、固定长度 256 字节、64～128 随机长度、64～256 随机长度和 64～1500 随机长度;(3)算法组合有 6 种:FIFO(没有 QoS 控制和第 4 节所讨论的优化)、Tail-drop/PQ、Tail-drop/RR、Tail-drop/WRR、Tail-drop/DRR 和 SPBS/RR.

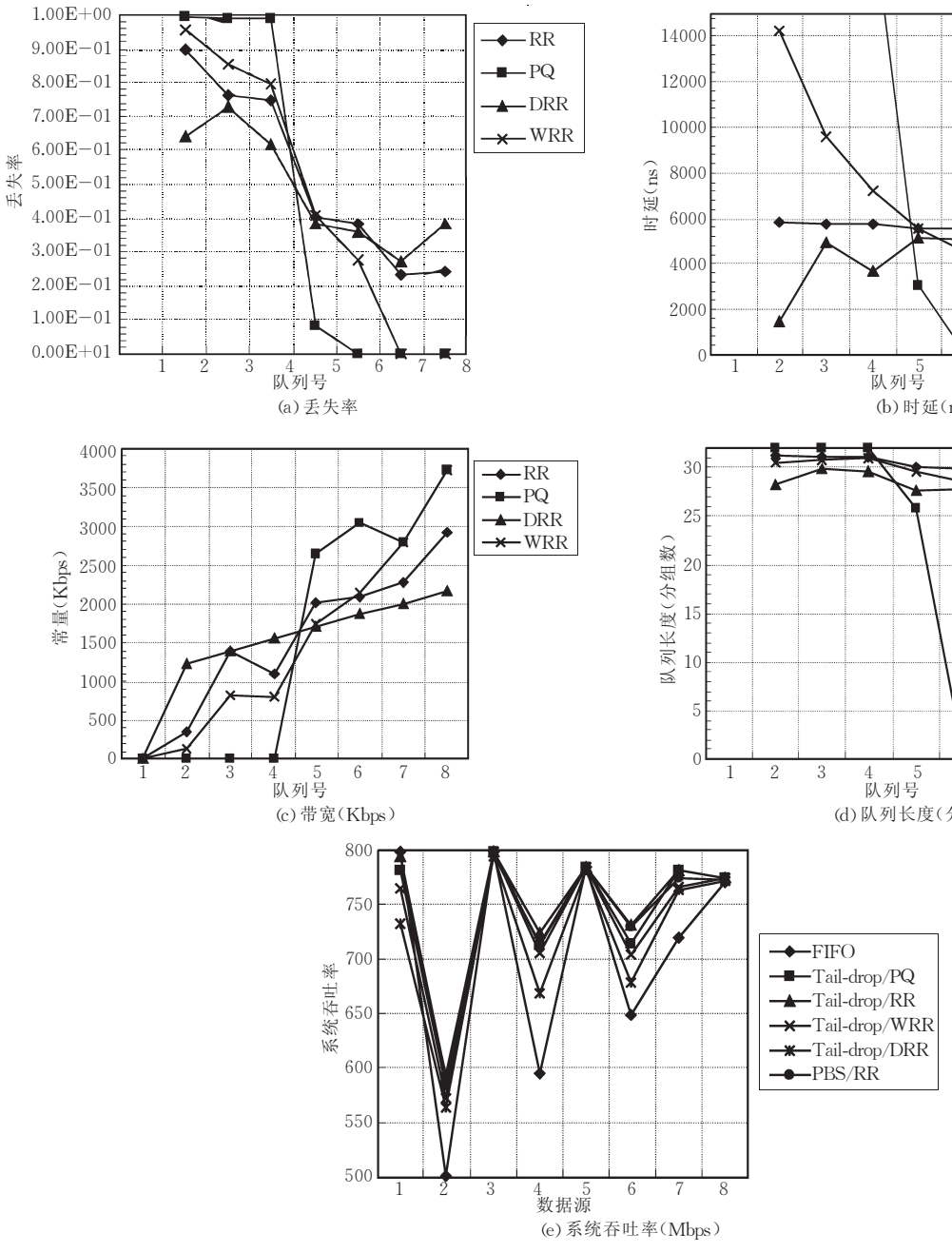
系统性能测试结果如图 6(e)(横坐标代表不同的数据源,纵坐标代表系统吞吐率,曲线为不同的算法配置).

系统性能测试结果分析:

(1) 由于系统采用 MAC-Packet(64 字节)的工作方式,即一次接收/发送一个 MAC-Packet,而分组最后一个 MAC-Packet 的有效数据可能会小于 64 字节,因此当数据源为 65,129,193,257 字节长度的分组时,系统吞吐率会有明显下降(图中只画出了 64/65 和 128/129 的对比).

(2) 经过第 4 节所述的优化后,系统吞吐率有很大提高. 从图中可以看出,在某些数据源的配置下,系统吞吐率甚至比优化前 FIFO 队列管理模式下的还要高.

(3) 分组调度算法的实现复杂度有如下关系:DRR/WRR 的复杂度高于 RR/PQ. 而从图中可以看出,DRR/WRR 配置下的系统吞吐率低于 RR/PQ. 因此,算法复杂度对系统性能的影响是不能被忽略的,这也是设计算法时需要考虑的一个重要问题.



(a) 丢失率 (b) 时延(ns) (c) 带宽(Kbps) (d) 队列长度(分组数) (e) 系统吞吐率(Mbps)

图 6

6 总 结

本文主要贡献如下:(1)详细研究了基于网络处理器(Intel IXP1200)队列管理子系统的体系结构和关键技术:通过对每个线程进行合理的任务分配和解决系统同步问题,有效地管理系统多线程以提高任务处理的并行性;通过设计合理的模块化结构和清晰的接口,灵活地进行系统配置和管理;(2)设计并实现了一种新的缓冲管理算法 DPBS;(3)针对网络处理器的硬件特点,总结了一些队列管理的重要的基本操作,并对网络处理器的硬件结构提出了一些改进的建议.这些改进将使得队列管理系统的开发更容易,性能更好.

然而,目前的队列管理子系统从结构和实现的算法上都只是单独方案,即缓冲管理和分组调度是分开的.实际上,缓冲管理和分组调度的综合方案能获得比单独方案更好的控制结果,也开始受到越来越多的研究者的重视[7,10].因此,有关队列管理系统的下一步工作将是设计队列管理的综合方案,并设计适合于综合方案的软件体系结构,在网络处理器上实现上述综合方案.

参 考 文 献

1 Braden R, Clark D, Shenker S. Integrated services in the Internet architecture: An overview. IETF RFC 1633, 1994

2 Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. An architecture for differentiated services. IETF RFC 2475, 1998

3 Dovrolis C, Ramanathan C. A case for relative differentiated services and the proportional differentiation model. IEEE Network, 1999, 13: 26～34

4 Striegel A, Manimaran G. Differentiated services: Packet scheduling with delay and loss differentiation. Computer Communications, 2002, 25(1): 21～31

5 Zhu K, Zhuang Y, Viniotis Y. Achieving end-to-end delay bounds by EDF scheduling without traffic shaping. In: Proceedings of IEEE Infocom'01, Anchorage, Alaska, 2001. 1493～1501

6 Causey J W, Kim H S. Comparison of buffer allocation schemes in ATM switches: Complete sharing, partial sharing, and dedicated allocation. In: Proceedings of International Conference on Communications, 1994, 2: 1164～1168

7 Lin C, Sheng L. Integration of scheduling real-time traffic and cell loss control for ATM networks. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2002, E85-B(4): 778～795

8 Shimonishi H, Yoshida M. An improvement of weighted round robin cell scheduling in ATM networks. In: Proceedings of IEEE Globecom'97, Phoenix, AZ, USA, 1997, 2: 1119～1123

9 Shreedhar M, Varghese G. Efficient fair queueing using deficit round-robin. IEEE Transactions on Networking, 1996, 4(3): 375～385

10 Liebeherr J, Christin N. JoBS: Joint buffer management and scheduling for differentiated services. In: Proceedings of IWQoS 2001, Karlsruhe, Germany, 2001. 404～418

11 Altintas O *et al*. Urgency-based round robin: A new scheduling discipline for multiservice packet switching networks. IEICE Transactions on Communication, 1998, E81-B(11): 2013～2021

12 Floyd S, Jacobson V. Random early detection gateways for congestion control. IEEE/ACM Transactions on Networking, 1993, 1(4): 397～413

13 Bennett J R, Zhang H. WF2Q: Worst-case fair weighted fair queueing. In: Proceedings of IEEE INFOCOM'96, San Francisco, CA, 1996, 1: 120～128

14 Liu, Layland J. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of ACM, 1973, 20(1): 46～61

15 Keshav S, Sharma R. Issues and trends in router design. IEEE Communications Magazine, 1998, 36(5): 144～151

16 Aweya J. On the design of IP routers- Part 1: Router architectures. Journal of Systems Architecture, 2000, 46(6): 483～511

17 Almesberger W. Linux network traffic control- Implementation overview. In: Proceedings of 5th Annual Linux Expo, Raleigh, NC, 1999. 153～164

18 Cho K. Managing traffic with ALTQ. In: Proceedings of USENIX 1999 Annual Technical Conference: FREENIX Track, Monterey CA, 1999. 121～128

19 Kohler E, Morris R, Chen B, Jannotti J, Kaashoek M F. The click modular router. ACM Transactions on Computer Systems, 2000, 18(3): 263～297

20 Cisco Systems Inc. Cisco 7500 series router data sheet. http://www.cisco.com

21 Stephens D. C, Bennett J. C. R, Zhang H. Implementing scheduling algorithms in high-speed networks. IEEE Journal on Selected Areas in Communications, 1999, 17(6): 1145～1158

22 Spalink T, Karlin S, Peterson L, Gottlieb Y. Building a robust software-based router using network processors. In: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP01), Chateau Lake Louise, Banff, Alberta, Canada, 2001. 216～229

23 http://www.cs.princeton.edu/nsg/router/papers/ixp.html

24 Vitesse Semiconductor Corporation. IQ2000 network processor product brief, 2000. http://www.vitesse.com

25 IBM Microelectronics Division. IBM powerNP NP4GS3 network processor solutions product overview, 2001. http://www.ibm.com

26 Yang L, Dantu R, Anderson T. ForCES architectural framework. IETF Internet Draft, 2002

27 Karlin S, Peterson L. VERA: An extensible router architecture. Computer Networks, 2002, 38(3): 277～293

28 Kam A. C, Siu K. Linear-complexity algorithms for QoS support in input-queued switches with no speedup. IEEE Journal on Selected Areas in Communications, 1999, 17(6): 1040～1056

29 Jiang Yong, Wu Jia-Ping, Xu Ke. Analysis of architecture and scheduling algorithms for fast switch core. Acta Electronica Sinia, 2000, 28(11A): 105～109(in Chinese)
(江　勇, 吴建平, 徐　恪. 高性能交换体系结构及其调度算法分析. 电子学报, 2000, 28(11A): 105～109)

30 Yu Zhong-Chao, Wu Jia-Ping, Xu Ke. Study of IP classification technology. Acta Electronica Sinia, 2001, 29(2): 260～262(in Chinese)
(喻中超, 吴建平, 徐　恪. IP 分类技术研究. 电子学报, 2001, 29(2): 260～262)

LIN Chuang, born in 1948, Ph. D., professor. In 1985～1986, he was a Visiting Scholar with the Department of Computer Sciences, Purdue University. In 1989～1990, he was a Visiting Research Fellow with the Department of Management Sciences and Information Systems, Graduate School of Business, University of Texas at Austin. In 1995～1996, he visited the Department of Computer Science, Hong Kong University of Science and Technology. His current research interests include computer networks, performance evaluation, logic reasoning, and Petri net theory and its applications.

ZHOU Wen-Jiang, born in 1975, master candidate. His research mainly focuses on quality of service, queue management schemes.

LI Yin, born in 1979, master candidate. His researches mainly focus on network control, QoS solving, etc.

ZHENG Bo, born in 1978, Ph. D. candidate. His research interests include computer networks, Petri net theory and performance evaluation.

TIAN Li-Qin, born in 1970, master candidate. His research interests include computer networks, Petri net, performance evaluation and workflow model.