可编程路由器中基于缓冲队列长度阈值的处理器调度

徐 恪,林 闯,吴建平
(清华大学计算机系,北京 100084)

摘 要: 可编程路由器除了转发 IP 分组之外,还需要执行计算任务.如何调度可编程路由器中 CPU 的处理能力是一个需要解决的重要问题.本文首先建立了一种通用的可编程路由器软件体系结构,在此基础上,提出了一种基于缓冲队列长度阈值的 CPU 调度算法,采用随机 Petri 网对算法进行了模型分析和计算.结果表明,该调度算法可以同时保证可编程路由器中的尽力发送流和 QoS 流的计算需求.

关键词: 可编程路由器; 长度阈值; 随机 Petri 网
中图分类号: TP393 **文献标识码**: A **文章编号**: 0372-2112 (2001) 11-1449-05

Processor Scheduling in Programmable Router Based on Queue Length Thresholds

XU Ke, LIN Chuang, WU Jian-ping
(Department of Computer Science, Tsinghua University, Beijing 100084, China)

Abstract: Programmable router not only forwards IP packets but also executes computing tasks. CPU scheduling in the programmable router is an important open problem till now. In this paper, the authors establish general software architecture of programmable router firstly. Based on the architecture, authors present a novel CPU scheduling algorithm based on queue length thresholds. We model this algorithm using stochastic Petri nets. The analysis results show that the scheduling algorithm can fulfill the requirements of computing of both best-effort flows and QoS flows in programmable router at the same time.

Key words: programmable router; queue length thresholds; stochastic Petri nets

1 引言

路由器是 Internet 的核心,随网络技术的发展,路由器的功能已经从传统的尽力转发分组发展到支持服务质量.近来在研究界又出现了一个新的研究趋势,路由器除了转发分组之外,还执行某些计算任务,这种路由器通常被称为主动路由器或者可编程路由器[1,2](下文中统称为可编程路由器).

本文将研究和路由器功能扩展相关的一个重要问题:如何调度可编程路由器中处理器的计算能力.和其他的计算机系统类似,路由器也必须按照有意义的方式调度 CPU 的计算能力.例如,它必须决定什么时候转发分组,什么时候处理控制信息,什么时候运行 Proxy 应用.路由器对服务质量的支持使问题更加复杂,在这种情况下,路由器或者要支持集成服务或者支持区分服务,无论是哪一种服务模型都需要路由器统一调度 CPU 计算能力,网络带宽和缓冲资源,而多个资源的集成调度是一个困难的问题,目前还没有很好的解决方案.本文提出了一种基于可编程路由器中计算任务缓冲队列阈值的 CPU 调度算法,该调度算法既可以保证区分 QoS 流的服务质量又可以尽量保证尽力转发流占用的资源,是一种灵活简便的调度算法.本文给出了算法的随机 Petri 网(SPN)模型,并进行了分析计算.这种复杂系统的模型如何求解是一个困难的问题,文中对模型进行了分解和精化设计,大大减少了计算复杂度,得到了满意的结果.这种分解和精化设计方法可以用于其它类似的复杂系统的分析和求解.

2 路由器中的进程体系结构

本节介绍所提出的面向软件的路由器进程体系结构(如图 1 所示).

分组首先由输入进程(简称为 *I*)进行处理,输入进程执行下面的循环:

从输入端口读取一个分组
对分组进行分类
把分组加入合适的队列

分组处理的第三个阶段是输出进程(简称为 *O*),输出进程每个端口一个,它们执行下列循环:

选择下一个要发送分组的队列
把分组从队列中取出
把分组写入输出端口

收稿日期:2000-12-22;修回日期:2001-05-25
基金项目:国家自然科学基金(No. 69873012);国家重点基础研究发展规划项目(No. G1999032707);国家 863 计划重项目(No. 863-306-ZD-07-01)

在这个过程中,输出链路调度算法嵌入在输出进程的选择发送分组的过程中.

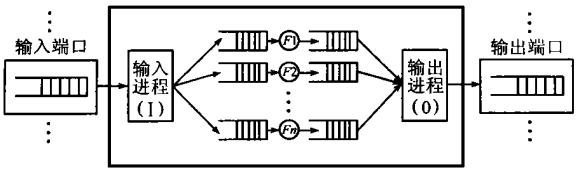

图 1 支持服务区分和可变的转发函数处理的路由器进程体系结构

转发路径中间的一个阶段,也就是图 1 中的计算进程 $F1$ 到 Fn,执行分组需要的处理. 每个进程都实现了某种转发函数 F. 在最简单的情况下,转发函数只简单地处理 IP 头部的 TTL 校验和域并修改链路层的头. 而在一般情况下,任意数量的不同的函数都可以应用于一个分组,从通常的 IP 转发,到 IP 选项处理,控制处理,代理代码,路由器扩展,主动代码等等. 每个这样的转发函数都有不同的处理需求.

3 相关工作

目前,关于可编程路由器方面的研究工作主要有华盛顿大学的 Router Plugin 体系结构[1]. 该体系结构提出了插件(plugin)的概念,在路由器的转发路径上插入称为 plugin 的扩展功能. 普林斯顿大学提出了可扩展路由器体系结构[3],该体系结构基于同一研究小组开发的 scout 操作系统[4],该操作系统中使用了 path 的概念对整个转发过程进行了优化. 纽约大学石溪分校的研究人员提出了基于机群系统的可扩展路由器体系结构[2]. 清华大学计算机系提出了一种基于可扩展组件的扩展服务路由器体系结构[5]. 但是这些研究工作都没有考虑可编程路由器中的 CPU 调度问题.

文[6]中,作者提出了一种面向可编程路由器的调度框架,该框架的基本思想是为尽力发送流和有计算需求的 QoS 流划分不同的 CPU 利用比例,该比例是静态配置的,CPU 调度器根据比例对各个计算流进行调度,调度的同时结合了判定流是否可以调度的机制,具体实现时使用的调度算法是 WF^2Q+ 算法[7]. 该算法的缺点是适应性比较差,而且该算法也很难真正实现按比例划分 CPU 的计算能力.

4 基于缓冲队列动态阈值的调度算法

4.1 调度算法的设计目标

(1) 保证具有不同服务质量要求的计算流的 QoS 要求(包括吞吐率,延迟和丢失率). 不同 QoS 要求的流应该得到不同级别的服务.

(2) 在保证了 QoS 流的计算要求之后,应该尽量保证尽力发送分组的带宽资源. 在网络中需要路由器参与计算的分组毕竟是少数,大量的分组仍然是尽力发送型的分组,因此,尽力发送分组的带宽必须尽量得到保证.

(3) 在系统过载(比如遭受恶意攻击)时,仍然能够正常转发分组.

4.2 算法的基本思想

根据图 1 中的进程体系结构,在可编程路由器中,主要的计算需求有:

(1) 物理接口的输入和输出处理,这部分由接口的驱动程序完成.

(2) 输入分组的分类处理,包括将分类后的分组加入合适的输入队列.

(3) 各个输入队列对应的处理,最简单的就是 IP 转发,复杂的可能是面向主动网络的 proxy 应用.

4.3 算法的详细描述

如果系统中一共有 n 个活动的流,则 CPU 调度器就需要监视 $2n+1$ 条队列的情况. 其中包括 n 个流的输入和输出队列和系统分类器的输入队列. 它们分别对应着系统中的 $n+2$ 个不同的任务,包括 n 个计算任务,一个分类任务和一个输出分组调度任务. 由 CPU 调度器根据这些队列中分组的数量决定调度哪个任务运行. 按照顺序把这些队列分别编号为 1 到 $2n+1$.

根据任务的急迫程度和待处理分组的实时性要求,可以给这些队列设定不同优先级 q_i. 优先级高的队列对应的任务首先得到服务,队列 1 的优先级最高. 除了优先级之外,每个队列还有一个相关的阈值,记为 Th_i. 阈值的含义如下:

当某条输入队列 i 中的分组超过了阈值 Th_i 时,如果优先级比它高的输入队列 j 中的分组数量还没有达到相应的阈值 Th_j,则队列 i 对应的任务首先得到调度. 一般来说,实时优先级越高的流的输入队列对应的输入队列阈值越小.

当输出队列 i 中的分组数量超过了阈值 Th_i 时,即使根据该任务的输入队列判断出应该运行该任务,CPU 也不选择该任务运行,而去寻找下一个能够运行的任务. 这是为了保证不出现输出队列的拥塞. 一般来说,带宽要求越大的流对应的输出队列阈值就越大.

综合以上两点,基于缓冲队列长度阈值的调度算法如图 2 所示.

```
CPU_Scheduler(int i);
/*
返回下一个要执行的任务的任务号
*/
{
  int j=0;/*记录输入队列非空的优先级最小的任务*/
  int k=0;/*记录系统输出队列是否已经发生拥塞*/
  for(i=1;i<=n;i++)
  /*任务号就用任务的优先级表示,并且任务的优先级两两不同*/
  {
    if 任务 i 的输入队列非空并且输出队列小于对应的阈值
      j=i;
    if 任务 i 的输入队列大于其相应的阈值
    {
      if 任务 i 的输出队列长度小于相应的阈值
        return i;
      else k=1;
    }
  }
  if(k==1) return n+1;/*任务 n+1 表示调度任务*/
  return j;
}
```

图 2 基于缓冲队列长度阈值的 CPU 调度算法

算法 CPU_Scheduler 返回下一个应该执行的任务的任务号. 如果某个任务的输入队列大于阈值,则判断其相应的输出队列,如果输出队列阈值小于对应的阈值,则调度该任务运行. 否则找下一个任务. 如果没有任何一个任务的输入队列超过阈值而且有任务的输出队列的长度大于阈值,则调度分组调度任务运行. 如果没有任何一个任务的输入队列超过阈值而且没有输出队列的长度超过阈值,则返回输入队列非空的优先级最高的任务. 如果返回 0,表示当前没有可以执行的任务,CPU 处于空闲状态.

算法 CPU_Scheduler 是一个线性复杂性的算法,算法复杂度为 $O(n)$, n 为系统中的任务数量,其平均性能为 $n/2$.

下面讨论如何选取阈值,不同的阈值选取方法决定了不同的调度策略.

(1) 静态优先级调度:最简单的方法是将所有输入队列的阈值设为 1,所有输出队列的阈值设为队列长度,在这种情况下,基于队列阈值长度的调度就变成了静态的优先级调度,*CPU* 调度器总是选择有分组需要处理的优先级最高的任务运行.

(2) 队列长度阈值调度:为每个输入队列和输出队列都分配一个静态的阈值,这种调度是一种动态优先级调度,但是队列长度阈值是静态的.

(3) 输入队列最小缓冲优先调度:空闲缓冲区最少的输入队列的阈值为 1,其余输入队列的阈值为队列长度,输出队列的阈值仍然静态分配. 这种调度方案也是一种动态优先级调度,而且其调度优先级和输入队列阈值都是可变的.

(4) 输入输出队列最小缓冲优先调度:输入队列空闲缓冲区加上对应的输出队列的占用缓冲区最小的输入队列的阈值为 1,其余输入队列的阈值为队列长度,这种方案同时考虑输入队列和输出队列的长度,也是一种动态的优先级调度.

需要说明一点,为了使调度算法简明,上述算法中没有讨论分组分类器的输入队列对调度决策的影响,我们会在下面讨论算法的 SPN 模型时加上分组分类器的输入队列. 实际上,增加分组分类器的输入队列对算法的原理没有影响.

5　系统的 SPN 模型及其近似求解

本节将使用随机 Petri 网 SPN 对上述调度算法建立模型并进行分析求解. 假定读者对 SPN 有一定的了解,关于 SPN 的详细讨论可见参考文献[8]. 在 SPN 模型中,分组的产生和发送可以用时间变迁表示,执行调度决策由瞬时变迁表示,它们不占用处理时间;缓冲队列可以用位置来表示,缓冲占有程度可以由位置的标识(marking)表示. 在模型中,允许变迁的可实施条件可用变迁的谓词规定,当变迁谓词条件不能满足时,变迁就不能实施. 对变迁的实施阻止概率或者服务竞争条件概率可以由变迁实施概率来规定.

5.1　系统的 SPN 模型

根据图 1 的进程模型,可以建立系统的 SPN 模型如图 3 所示. 下面简单说明图 3 中各个符号的含义.

c_1 到 c_m 表示路由器的 m 个输入端口,假定分组到达过程是泊松过程,其分组到达速率分别为 r_1 到 r_m.

q_c 表示分类器的输入队列, S_{c1} 到 S_{cn} 表示将分类后的分组放入计算函数的输入队列的分类计算过程.

q_{I1} 到 q_{In} 表示 n 个计算任务对应的输入队列, S_{I1} 到 S_{In} 表示 n 个计算任务相应的计算过程, q_{o1} 到 q_{on} 表示 n 个计算任务对应的输出队列, S_1 到 S_n 表示输出端口分组调度算法的执行过程.

C 表示系统中的共享资源 —CPU.

变迁 S_{c1} 到 S_{cn} 都有相应的随机开关,表示对应的流的分组在总的到达分组中的比例.

d_c, d_{I1} 到 d_{In}, d_{s1} 到 d_{sn} 构成了系统的调度框架. 这些变迁相联系的实施谓词和随机开关就决定了系统如何调度各个任务执行.

前面已经提到,给每个队列都设置了一个阈值,为了便于描述,我们把分类器输入队列的阈值记为 TH_C,调度器输出队列的阈值记为 TH_S,计算进程 i 的输入队列阈值记为 TH_I_i,计算进程 i 的输出队列的阈值记为 TH_O_i.

下面描述模型中各个变迁的实施谓词. 下文描述中的函数 $M(i)$ 表示队列 i 的长度.

(1) 静态优先级和队列长度阈值调度.

d_c 的谓词为:

$$(M(q_c) \geq TH\_C) \vee ((\forall i, 1 < i \leq n, (M(q_{Ii}) < TH\_I_i) \vee [(M(q_{Ii}) > TH\_I_i) \wedge (M(q_{oi}) > TH\_O_i)]) \wedge [\forall i, 1 < i \leq n, (M(q_{oi}) < TH\_O_i)]$$

d_{Ii} 的谓词为:

$$\left[\begin{array}{l}(M(qIi) > TH\_I_i) \wedge (M(q_{oi}) < TH\_O_i) \wedge (M(q_c) < TH\_C) \wedge \\ (\forall j, 1 \leq j < i, (M(q_{Ij}) < TH\_I_i) \vee [(M(q_{Ij}) > TH\_I_j) \wedge \\ (M(q_{oj}) > TH\_O_j)])\end{array}\right] \vee$$

$$\left[\begin{array}{l}(M(q_{Ii}) < TH\_I_i) \wedge (M(q_{oi}) < TH\_O_i) \wedge (M(q_c) = 0) \wedge \\ ((\forall j, i < j \leq n, (M(q_{Ij}) < TH\_I_j) \vee (M(q_{Ij}) > TH\_I_j) \wedge \\ (M(q_{oj}) > TH\_O_j))) \wedge \\ (\forall j, 1 \leq j < i, (M(q_{Ij}) = 0) \vee (M(q_{oj}) > TH\_O_j)) \\ \wedge [\forall i, 1 < i \leq n, (M(q_{oi}) < TH\_O_i)]\end{array}\right]$$

d_{si} 的谓词为:

$$(M(q_c) < TH\_C) \wedge (\forall j, 1 \leq j \leq n, (M(q_{Ij}) < TH\_I_j) \vee (M(q_{Ij}) > TH\_I_j) \wedge (M(q_{oi}) > TH\_O_j)) \vee (\forall j, 1 \leq j \leq n, (M(q_{ij}) = 0)) \wedge (M(q_c) = 0) \wedge (M(q_{Oi}) > 0)$$

(2) 输入队列最小缓冲优先调度

d_c 的谓词为:

$$(\forall i, 1 \leq i \leq n, (B_{Ii} - M(q_{Ii})) > (B_c - M(q_c)) \wedge ((B_{oi} - M(q_{oi})) > (B_c - M(q_c)))) \vee [(\forall i, 1 \leq i \leq n, (B_{Ii} - M(q_{Ii})) < (B_c - M(q_c)) \wedge (M(q_{oi}) > TH\_O_i))]$$

上式中, B_{Ii} 表示计算进程 i 的输入队列的容量, B_c 表示分类器缓冲队列的容量, B_{oi} 表示计算进程 i 的输出队列的容量.

d_{Ii} 的谓词为:

$$((\forall j, 1 \leq j < i, (B_{Ii} - M(q_{Ii})) < (B_{Ij} - M(q_{Ij})) \wedge (B_c - M(q_c) > (B_{Ij} - M(q_{Ij}))))$$

$\wedge(\forall j,1\le j\le n,(B_{oj}-M(q_{oj}))>(B_{Ii}-M(q_{Ii})))$

$\vee[(\forall i,1<i\le n,(B_{Ii}-M(q_{Ii}))<(B_c-M(q_c))\wedge(M(q_{oi})>TH\_O_i))]$

d_{si}的谓词为:

$((\forall j,1\le j<i,((B_{Oi}-M(q_{Oi}))<(B_{Oj}-M(q_{Oj})))\wedge(B_c-M(q_c)>(B_{Ii}-M(q_{Ii}))))$

$\wedge(\forall j,1\le j\le n,(B_{Ij}-M(q_{Ij}))>(B_{Oi}-M(q_{Oi})))$

$\vee[(\forall i,1<i\le n,(B_{Ii}-M(q_{Ii}))<(B_c-M(q_c))\wedge(M(q_{oi})>TH\_O_i))]$

(3) 输入输出队列最小缓冲优先调度

d_c的谓词为:

$(\forall i,1\le i\le n,(B_{Ii}-M(q_{Ii})+M(q_{Oi}))>(B_c-M(q_c)))$

$\wedge((B_{oi}-M(q_{oi}))>(B_c-M(q_c)))$

d_{Ii}的谓词为:

$$\left\{\begin{array}{l}(\forall j,1\le j<i,(B_{Ii}-M(q_{Ii})+M(q_{oi}))<(B_{Ij}-M(q_{Ij})+M(q_{oj}))\\ \wedge(B_c-M(q_c)>(B_{Ii}-M(q_{Ii})+M(q_{oi})))\end{array}\right\}$$

$\wedge(\forall j,1\le j\le n,(B_{oj}-M(q_{oj}))>(B_{Ii}-M(q_{Ii}))+M(q_{oi}))$

d_{si}的谓词为:

$((\forall j,1\le j<i,(B_{Oi}-M(q_{Oi}))<(B_{Oj}-M(q_{Oj}))\wedge(B_c-M(q_c)>(B_{Oj}-M(q_{Oj}))))$

$\wedge(\forall j,1\le j\le n,(B_{Ij}-M(q_{Ij})+M(q_{oj}))>(B_{Oi}-M(q_{oi})))$

上述列出了各个变迁的实施谓词,实施谓词保证了,如果有变迁可以实施,那么最多只有一个变迁可以实施,因此不需要给实施谓词联系随机开关.

5.2 模型的分析和求解

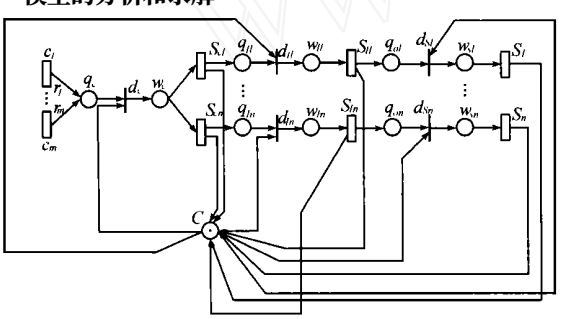

图 3 调度算法的 SPN 模型

图 3 中的模型比较复杂,直接求解将面临状态爆炸. 对这种多维的、复杂的马尔可夫链进行求解是一个还没有完全解决的困难问题. 下面采用基于模型分解和精化设计[9]的近似求解方法. 首先可以把图 3 中的共享资源 CPU 的位置和相关联的弧都去掉,因为所有的信息都包含在了瞬时变迁的实施谓词中. 然后再去掉所有的瞬时变迁,和瞬时变迁相关联的实施谓词描述在之前的时间变迁中. 这样就得到了如图 4 所示的经过精化设计的 n 个子模型. 其中 c_i 关联了 d_c 的谓词,S_{ci} 关联了 d_{Ii} 的谓词,S_{Ii} 关联了 d_{si} 的谓词.

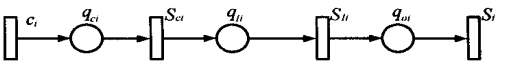

图 4 精化设计后的模型

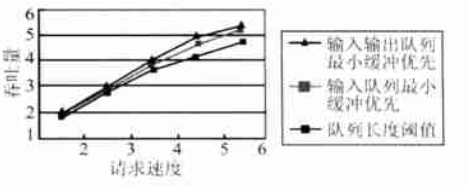

图 5 三种阈值选取方法的吞吐量比较

下面列出算法是在一个典型情况下经过计算得到的数值结果以表明算法有效性. 限于篇幅,略去了实际负载条件下的例子,只列出了可以说明算法有效性的较为简单的例子.

例 设路由器中有三条转发路径,其中两条是有 QoS 要求的转发路径,另一条是尽力发送路径. 将这三条路径分别记为 $q1$, $q2$ 和 $b1$,其中 $q1$ 的优先级最高, $b1$ 的优先级最低. 分组分类器的输入队列长度为 5, $q1$ 和 $q2$ 的输入队列长度为 5,输出队列长度也为 5. $b1$ 的输入队列和输出队列长度均为 8. 分组分类器和分组调度器的平均服务速率均为 10, $q1$ 的转发函数的服务速率为 2, $q2$ 的转发函数的服务速率为 3, $b1$ 的转发函数的服务速率为 6. 三条路径的分组到达速率之比为 1∶1.5∶2.

图 5 是采用三种不同的阈值选取方法(不包括静态优先级调度)时系统吞吐率的比较. 其中队列长度阈值调度时分组分类器的输入队列阈值为 5, $q1$ 的输入队列的阈值为 2,输出队列阈值为 3, $q2$ 的输入队列的阈值为 3,输出队列阈值为 3, $b1$ 的输入队列阈值为 4,输出队列阈值为 6.

从图中可以看出,输入队列最小缓冲优先和输入输出队列最小缓冲优先阈值方案的吞吐率要高于队列长度阈值方案,但是总的差别并不十分明显. 进一步的计算表明,采用队列长度阈值方案时,通过合理地选择队列阈值,可以较好地保证 QoS 流的服务质量要求. 采用输入输出队列最小缓冲优先可以较好的保证尽力发送流和整个系统的吞吐率. 限于篇幅,只列出队列长度阈值情况下的分组转发延迟,吞吐率和丢失率的曲线(图 6). 图 6 中的请求速率是系统总的请求速率,在三条转发路径之间的比例为 1∶1.5∶2. 从图 6 中可以看出,采用队列长度阈值调度方案时,可以充分保证有 QoS 要求的计算路径的吞吐率和转发延迟,但是尽力发送流的转发延迟和

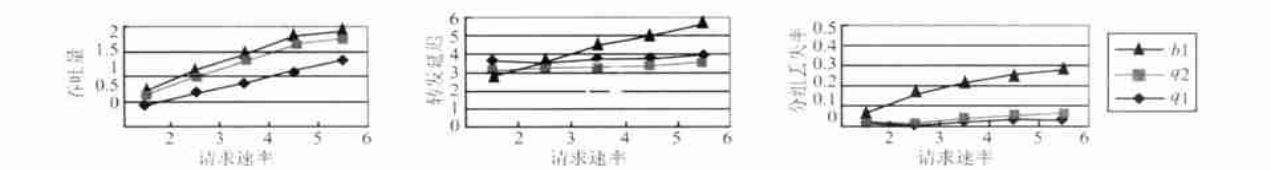

图 6 队列长度阈值下三条计算路径的吞吐量,延迟时间和丢失率的比较

丢失率都受到了较大的影响.如果采用输入输出队列最小缓冲优先,则可以同时保证 QoS 流和尽力发送流的吞吐量,但是随着输入分组速率的增加,QoS 流延迟和吞吐量将受到一定的影响(限于篇幅,相应的图表未列出).同时,基于阈值的调度算法可以保证在系统过载时仍然能够处理到达的分组.

由此,可以得到结论,提出的基于缓冲队列长度阈值的调度算法可以通过采用不同的阈值选择方案来满足 4.1 中提出的可编程路由器中调度算法的基本要求,是一种灵活简便的调度算法.

6　结论和进一步的研究工作

本文提出了一种基于路由器中计算任务缓冲队列阈值的 CPU 调度算法,该调度算法既可以保证区分 QoS 流的服务质量又可以尽量保证尽力转发流占用的资源,是一种灵活简便的调度算法.在此之前,国际上对可编程路由器中 CPU 调度的研究还不多见,本文是首次比较完整地提出并初步解决了这个问题.文中给出了算法的随机 Petri 网(SPN)模型,并进行了分析计算.为了解决复杂模型分析求解时的状态爆炸问题,采用了模型分解和精化设计方法.这种方法可以应用于其他类似的复杂模型的分析和求解.

本文提出的调度算法只考虑了单处理机的情况,而目前基于分布式处理结构的路由器是研究的热点,如何将本模型扩展到分布式结构的可编程路由器中是下一个研究目标.

参考文献:

[1] D Decasper, Z Dittia, G Parulkar, et al. Router plugins: A software architecture for next generation routers [A]. Proceedings of the ACM SIGCOMM'98 Conference [C], San Francisco, 1998: 229 - 240.

[2] Prashant Pradhan, Tzi-Cker Chiueh. A cluster-based scalable and extensible edge router architecture [R]. TR63-2000, Department of Computer Science, State University of New York at Stony Brook, 2000.

[3] L L Peterson, S C Karlin, K Li. OS support for general-purpose routers [A]. Proceedings of the 7th Workshop on Hot Topics in Operating Systems (HotOS-VII) [C], New York, 1999: 60 - 67.

[4] D Mosberger, L L Peterson. Making paths explicit in the scout operating system [A]. Proceedings of the Second USENIX Symposium on Operating System Design and Implementation (OSDI'96) [C], San Francisco, 1996: 153 - 167.

[5] 徐恪,吴建平,江勇,喻中超. 扩展服务路由器软件体系结构研究 [J]. 计算机研究与发展,2001,增刊:47 - 54.

[6] Andy Bavier, Scott C Karlin, Larry Peterson, Xiaohu Qie. Scheduling computations on a programmable router [R]. TR603, Department of Computer Science, Princeton University, 2000.

[7] J C R. Bennett, H Zhang. Hierarchical packer fair queueing algorithms [A]. Proceedings of the ACM SIGCOMM'96 Conference [C], San Francisco, 1996: 143 - 156.

[8] 林闯. 随机 Petri 网和系统性能评价 [M]. 清华大学出版社, 2000.

[9] 林闯. 随机 Petri 网模型的精化设计 [J]. 软件学报,2000,11(1): 104 - 109.

作者简介:

徐　恪　男. 1974 年 12 月出生于于江苏省洪泽. 清华大学计算机系博士,讲师,主要研究领域为计算机网络体系结构,高性能路由器体系结构,分布式实时操作系统,系统性能评价,算法分析与设计.

林　闯　男. 1948 年 7 月出生于辽宁省沈阳市. 博士,清华大学计算机系教授,博士生导师,网络研究所所长,计算机学报编委,中科院网络中心和北京科技大学兼职教授. 主要研究领域为计算机网络,系统性能评价,随机 Petri 网和逻辑推理模型等. 在国内外核心学术期刊和 IEEE 的重要学术年会上发表了六十多篇有影响的论文.

吴建平　男. 1953 年 10 月生于山西太原. 博士,清华大学计算机系教授,博士生导师,中国教育科研计算机网专家委员会主任,网络中心主任,教育部"长江学者奖励计划"特聘教授. 主要研究领域为计算机网络体系结构,计算机网络协议测试,形式化技术,已经在国内外核心学术期刊和重要国际学术会议上发表了一百多篇论文.