用户公平的活动队列管理

徐 建, 李善平
(浙江大学计算机学院, 浙江杭州 310027)

摘 要: 用户公平活动队列管理算法 UFQ(User Fair Queuing) 的目标是在各种网络环境中都能为所有的用户提供满意度一致的服务. UFQ 采用在网络边缘标记用户所属数据报的期望服务满意度 u, 在网络核心根据数据报的满意度高低, 结合当前数据报流经节点的拥塞程度, 来决定数据报的丢弃或标记(使用 ECN), 从而获得不同用户一致满意的服务. UFQ 不要求接纳控制和信令. 它仅在网络边缘保持数据流的状态信息; 只维护一个先进先出队列, 通过拥塞时丢弃或标记较高满意度的数据报, 在不同的用户之间公平地分配网络带宽, 从而有效地控制、减轻拥塞. 通过 TCP/IP 网络的模拟, 证实了算法能够按照用户期望满意度公平地分配网络带宽, 提高网络的服务质量.

关键词: 满意度; 公平性; 队列管理
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2004) 03-0435-06

User Fair Active Queue Management

XU Jian, LI Shan-ping
(College of Computer Science, Zhejiang University, Hangzhou, Zhejiang 310027, China)

Abstract: A user fair active queue management algorithm named UFQ is introduced. UFQ aims at fairly distributing the network resources among users in various situations. Fairness here means different users who have different profiles will get network resources proportionally. User's utility is defined. User's Packets will be inserted a label with its utility in network edge. During network congestion, UFQ will drop or mark packets according to utility inserted in packets. So it distributes network resources among users with average utility. UFQ requires neither admission control nor signaling. Edge router maintains per flow state and core router maintains no flow state. Only one FIFO queue is kept in each node. With simulation and experiments of TCP and UDP traffic, UFQ has shown good performance on quality of service.

Key words: utility; fairness; queue management

1 引言

Internet 上新出现的应用(如网络电话、视频会议等), 要求对每个用户都能够保证其吞吐率. 保证网络服务质量的一个重要原则就是对用户不同需求应提供相对应的不同服务.

在商业性的 Internet 环境中, 数据流量的接入特性取决于 ISP 和用户之间的合同. 然而, 对接入后用户数据在网络上传输过程的研究, 很多都仅集中在数据流的公平性(如最大最小公平性和比例公平性[1])上, 而对不同合同用户之间网络资源分配的公平性的研究却较少. 本文的工作就是研究用户数据流在网络上传输时, 如何实现带宽资源在用户之间的公平分配.

传统的 Internet 提供的是尽最大努力的传输服务. 不同用户数据以不同的接入速率进入网络后, 它们的公平性主要是由具适应性的 TCP 保证的[2]. 但不是所有的 Internet 数据传输都使用 TCP, 结果非适应性数据流反而容易获得更大比例的网络带宽, 导致其他适应性数据流服务质量降低. 另外, 由于这种带宽资源分配使用端到端策略, 用户能够使用的网络资源与它发送的数据流数量有很大关系, 这也会导致网络资源的不公平分配.

由于端到端策略如 TCP 等的局限性, 研究者提出了数据流的公平排队机制. 典型的方法如: WFQ(Weighted Fair Queuing)[3]和 CBQ(Class-Based Queuing)[4]及许多其他的变种, 或者单个队列的丢弃机制如 FRED[5], MBLUE. 显然这些机制比被广泛使用的用 Drop Tail 管理的 FIFO 队列更难于应用. 所以尽管人们采用了许多技术减少排队的复杂性, 但在高速网络的每个核心路由上要对每个数据报都进行分类并管理数据流状态, 因而其开销是昂贵的.

CSFQ[6]是一种核心无状态的公平排队机制, 算法的目标是在网络边缘标记每个数据流的速率, 在网络核心不必维护每个流的状态, 而通过在拥塞节点计算公平的传输速率来分配网络带宽资源. CSFQ 及其它相关的算法[7,8]总体上简化了公平排队机制. 但算法仍采取平均分配网络带宽资源的办法, 而没有考虑合同用户的差异, 显然这是不公平的.

过去的几年里, 研究者还提出了两个完全不同的策略: IntServ[9]和 DiffServ[10]. 这两个策略的公平性概念主要是资源的预约分配. IntServ 用户预约一定数量的网络资源, 网络进行接纳控制决定一个资源是否能够满足请求. 但由于是基于数据流的控制, 其扩展性较差. DiffServ 提出了一个对于不同数据流聚集进行处理的可扩展的方法. 它将复杂的功能如数据

收稿日期: 2003-01-27; 修回日期: 2003-06-06
基金项目: 浙江省自然科学基金(No. 602032)

流的分类标记放到网络的边缘, 而在网络核心部署相对简单的如针对不同分类进行处理的功能. 虽然 IETF 已经有建议的 DiffServ 标准, 但是对相同服务级别内的不同数据流之间的公平性还是没有解决.

也有文章从用户的角度研究如何最大化用户的效用函数[11], 从而达到优化网络资源分配的目的. 还有部分学者从网络资费出发结合用户效用函数讨论数据流的速率控制来优化资源分配[12]. 但是没有网络中相关公平机制的保障, 要实现公平性还是有一定困难.

综合以上观点, 本文提出用户公平性的概念. 使用在网络边缘标记用户所属数据报的满意度, 在网络核心根据数据报的期望满意度高低进行数据报调度的 UFQ 算法以达到用户公平的目标.

2 用户公平的活动队列管理模型

在这一部分, 首先假设每个用户在某一时刻只发送一个数据流沿一定的路径到目的节点, 以简化讨论. 接着分析了这种模型的公平性. 最后把它扩展到一个用户同时拥有多个数据流的情况.

2.1 模型简介

假设网络有连接资源集合 J , j 是其中的一个连接, 使用 C_j 表示 j 的容量, C= (C_j , j I J) . 网络用户集合 I, i 是其中的一个用户. 用户预约的服务 m_i, 用户 i 数据流进入网络的速率 x_i . 路径 r 是 J 的一个非空子集, R 是所有路径 r 的集合, r_i 是用户 i 数据流经过的路径. 设 A_{jr}= 1 如果 j I r 即连接 j 在路径 r 上, 否则 A_{jr}= 0, 这样就定义了一个 0- 1 矩阵 A= (A_{jr}, j I J, r I R) . 集合数据流 y= (y_r, r I R) 表示路径 r 上数据流的聚合. 这个定义类似 Kelly[12] .

从网络提供商的角度定义 m_i / x_i 为用户数据流接受某次传输服务的出价(bid) K_i; 从用户的角度定义 x_i / m_i 为用户期望得到的服务满意度 u_i. 显然, u_i= 1/ K_i.

用户是按预约的服务付费的, 也即按接入的速率收费, 发送数据流量的多少与所付费用并无直接联系, 因此对于用户的一个优化问题是

$$USER_i(U_i; K_i): \text{Maximize } U_i(x_i) - K_i * x_i = U_i(x_i) - m_i$$
$$\text{Over } x_i \geq 0$$

其中 U 表示用户的效用函数.

服务提供商从用户那里收取的是固定费用, 那么在网络拥塞时, 在每个拥塞连接处网络的一个服务优化问题就是

$$NETWORK(A, C; K_i): \text{Maximize } \sum_{I_j} K_i^{out}, \sum_{I_j} x_i^{out}$$
$$\text{Subject to } Ay \leq C$$
$$\text{Over } x_i \geq x_i^{out} \geq 0, K_i^{out} \geq K_i \geq 0, y \geq 0$$

服务优化就是首先满足 K 值较大的用户, 又考虑公平性, 对所有的用户提供服务, 同时最大程度利用连接资源. x_i^{out} 是用户数据流在拥塞连接处分配到的传送速率, $x_i \geq x_i^{out}$ 表示数据流的某些数据报可能由于拥塞而被丢弃了. K_i^{out} 表示数据流经过调度后 x 发生变化, 相对于 m, 变化了的 K 值. 限制条件 $Ay \leq C$ 表示在每个连接处集合数据流的大小不能超过连接的容量, y \ 0 表示有数据流.

如果在网络中传输的每个数据报都携带有 u_i, 那么可以设计这样一种数据报的活动队列管理算法: 在拥塞时优先传输 u 值较小的数据报, 也就是出价 K 较高的数据报; 而对于 u 值较大的数据流按一定比例丢弃或标记, 使这些数据流能够得到与其预约成比例的传输服务. 本文使用以下的数据报丢弃概率计算方法达到这个目标:

$$drop\_p_i = \begin{cases} 0, & non\text{-}congestion \\ \max(0, 1 - u_{avg}/u_i), & congestion,\ u_i^{new} = u_{avg} \end{cases} \quad (1)$$

$drop\_p_i$ 是用户 i 所属数据报的丢弃概率, u_i 是用户 i 数据报的满意度, u_{avg} 是拥塞连接的服务水平. 服务水平表示某一段时间内拥塞连接对外提供服务的能力, 它是综合连接内所有用户数据流的 u 值及连接的拥塞程度得到的. 式(1) 表明在拥塞节点处, 如果网络处于正常状态, 用户数据报将顺利通过(丢弃或标记概率为 0) ; 反之如处于拥塞状态, 数据报将以一定比例的概率被丢弃(如果 u_i 小于 u_{avg}, 计算所得概率小于 0 那么 drop_ p 以 0 计) , 并且在 u_i 大于 u_{avg} 而数据报没有被丢弃的情况下, 重新标记用户的数据报满意度为当前连接的服务水平 u_{avg}. 节点的拥塞指示将在第 3 部分讨论.

显然, 这种数据报的调度策略对于网络来说是合理的, 因为它以最大限度对所有用户提供服务为目的、按出价高低次序又兼顾公平的满足了用户的传输需要.

如果 L 表示所有 u 值小于或等于 u_{avg} 的用户集合, H 表示所有 u 值大于 u_{avg} 的用户集合, 那么在连接 j 处有:

$$\sum_{i \in L} x_i + \sum_{i \in H} x_i \# u_{avg}/u_i = C_j$$

由于 $u_i = x_i / m_i$, 所以

$$\sum_{i \in L} x_i + u_{avg} \# \sum_{i \in H} m_i = C_j,\ u_{avg} = (C_j - \sum_{i \in L} x_i) \setminus \sum_{i \in H} m_i$$

因此服务水平 u_{avg} 的计算需要了解连接 j 处等待传输的所有用户的数据流速率 x 或者预约 m, 这样就必须保持每个用户在 j 处的状态, 而且 u_{avg} 的计算也很复杂. 如何简单地确定 u_{avg} 是算法实现的难点. 在第 3 部分将介绍一种不需维护用户状态的服务水平计算方法, 减小了复杂度, 使算法的设计实现成为可能.

用 u 而不是 K 为主要指标进行数据报调度是出于简化应用的考虑, 可减小数据报标记空间和方便服务水平的确定.

2.2 公平性

由式(1) 可知, 用户 i(如果他的 u 值大于 u_{avg}) 的数据流从拥塞连接 j 处出来后, 有

$$x_i^{out} = x_i^{in}(1 - drop\_p_i) = x_i^{in} u_{avg}/u_i$$

通过数据报标记的替换 $u_i^{new} = u_{avg}$, 上式可以导出连接 j 处 x_i^{out}、x_i^{in} 与 u_i^{new}、u_i 的关系

$$x_i^{out}/u_i^{new} = x_i^{in}/u_i$$

根据上式可知数据流的速率 x 和标记 u 的比值在经过一个拥塞连接的前后是保持不变的. $x_i^{out}/u_i^{new} = m_i$, 可以推导出 $x_i^{out} = m_i u_i^{new} = m_i u_{avg}$

因此, 在拥塞连接使用式(1) 进行数据报的管理, 两个具有竞争关系的用户数据流 i, k(用户 i, k 期望得到的满意度

都大于服务水平)之间是按其预约的带宽比例分配的

$$x_i^{out}/m_i = x_k^{out}/m_k = u_{avg}$$

如果增加经过瓶颈连接 j 的某个用户 i 带宽 x_i, 就会减少经过 j 的其他用户数据流的带宽. 最终, 如不减少某个用户 k 的 x_k, 且 $x_k/m_k F x_i/m_i$, 那么就不可能增加用户 i 的 x_i, 结果最大化了 $\min(x_i/m_i)$, 故在连接 j 处用户接受的服务是公平的. Alkert[7] 在其技术报告中有类似的分析.

213　一个用户同时发送经过不同路径多个数据流的情况

用户 i 总发送速率是 x_i, 该用户预约带宽是 m_i, 显然, 该用户期望得到满意度为 x_i/m_i. 若它同时有 l 个数据流, 各个数据流的速率分别是 $x_{i1}, x_{i2}, x_{i3}, , , x_{il}$, 由用户与网络提供商的约定可知, 用户接受的是总计为 m_i 速率的服务, 那么可以合理地假设 i 用户第 l 个数据流的预约是 $(x_{il}/x_i) * m_i$. 这样 i 用户第 l 个数据流期望满意度也是 $x_{il}/[(x_{il}/x_i) * m_i] = x_i/m_i$. 由此, 用户 i 第 l 个数据流应该接受的是预约为 $(x_{il}/x_i) * m_i$ 的服务, 用户 i 总共接受的是 m_i 的预约服务, 各个数据流的期望满意度是一样的. 一个数据流的数据报通过不同路径到达目的地的情况也适用这种分析. 所以在以后分析及模拟试验中本文都设想一个用户仅有一个数据流, 以简化讨论.

3　算法的设计

UFQ 算法首先在网络边缘通过观察数据流的速率, 结合用户的服务预约进行用户期望服务满意度的标记, 这个值将作为网络核心调度的依据. 其次, 在网络核心处 UFQ 将根据队列的拥塞程度, 结合到达队列的数据报的满意度对数据报进行调度))) 超过队列服务水平的数据报按照一定比例的概率丢弃, 对低于服务水平的数据报继续提供服务, 进而引导数据源调整发送速率以避免拥塞. 第三就是允许和保护短时间的数据流猝发.

网络核心不保留各个用户数据流的状态, 把复杂数据流分类标记放在网络边缘, 是符合 Internet 把复杂的控制机制留在端节点而网络核心保持相对简单通用的端到端设计原则的[13, 14]. 而网络核心依照数据报满意度来决定丢弃或标记的做法, 就是让不同用户都能够得到相同的服务. 公平性是依据用户在网络提供商处的预约服务来保证的, 不是简单地平均, 预约服务多的用户将等比例地分配到更多的网络带宽资源.

311　网络边缘数据报的标记

由随机服务系统理论[15], 可假设在一个很短的时间片段内数据报到达时间间隔分布的概率分布函数为 $A(t) = 1 - e^{-vt}$ (v 表示数据报的平均到达速率. 文中对所有用户使用同样的假设, 因此不影响整个算法的公平性). 根据数据报到达时间间隔的分布特点, 构造了下面的数据流速率估计等式

$$x_{new} = (1 - e^{-vt}) Pkt.\ size/t + e^{-vt} x_{old} \qquad (2)$$

等式代表的含义是: 一个数据流的当前速率是当前与上一个数据报的到达时间间隔的平均速率和上一个数据报到达时的速率的指数平均, 权值是数据报到达时间间隔分布的概率密度. 这样速率的计算能够适应数据流瞬时的猝发, 由于采用指数平均, 同时保持了对速率即时变化的敏感性.

式中 x 表示数据流速率, Pkt. size 表示当前到达数据报的长度, t 表示当前到达数据报与上一到达数据报的时间间隔, v 表示数据报的平均到达速率.

对于 v, 本文使用基于数据报到达的相临时间间隔数据 t 的学习估计, 其式如下

$$v_{new} = (v_{old} t_{interval} + 1)/(t_{interval} + t)$$

式中 $t_{interval}$ 表示估计 v_{new} 时向前学习取样的时间长度. 这样, v 的变化就能够动态地学习反映数据报到达的概率分布密度, 增强了算法对网络变化的适应能力.

最后, 用户期望的服务满意度为 $u = x/m$,　　(3)

其中 u 表示用户期望的服务满意度, m 表示用户预约的服务.

312　网络核心数据报的管理

核心节点连接拥塞的指示: 采用 RED 算法的活动队列长度计算方法[16]来指示连接早期的拥塞.

$$Avg2queue = (12W_q) Avg2queue + W_q q \qquad (4)$$

等式中参数含义同 RED. 当队列长度 Avg2queue 大于 minth 时, 算法认为该节点连接发生拥塞; 当 Avg2queue 大于 maxth 时, 认为发生严重拥塞.

核心节点服务水平的确定及其收敛性: 因为核心节点不保留每个用户的数据流状态, 所以服务水平的确定是使用如等式(5)动态平均方法估算的, 它近似地模拟某一时刻该连接的对外服务水平.

$$u_{avg} = (u_{avg} * q + u_{pkt})/(q + 1) \qquad (5)$$

式中 u_{avg} 是连接的服务水平, q 是队列实际长度, u_{pkt} 是入列数据报的编码换算的期望得到的服务满意度.

由式(5)可知, 用户可以通过发送大量的数据报提高 u_{avg} 来获得较多的服务, 这对保持算法的收敛性是不利的. 正如不保持数据流状态的活动队列管理算法都会面临的问题一样, 算法 UFQ 也有这样的问题: 如何保证算法的收敛性; 如何防止一个用户通过发送远大于其预约的数据流来提高节点服务水平而获取额外的服务. 本文的方法是:

首先, 当个别用户或多个用户造成的拥塞情况十分严重, 即队列长度大于 maxth 时, 就以 weight 权重积式减少队列的平均服务水平, 加大数据报的丢弃力度. 考虑式(5)计算的 u_{avg}, 这样服务水平值就以已经证明的/ 和式增加2积式减少(AIMD)0[17]方式动态地向真实值收敛. 从而保持队列长度在 minth 和 maxth 之间, 按照一般拥塞的处理方法管理队列, 使连接工作于合理地负载.

其次, 可以在网络边缘数据流分类标记的同时, 对数据进行过滤, 对于超过预约上限的数据进行限制, 防止恶意的数据流进入网络.

再次, 当连接的队列发生拥塞状态时即开始有选择的丢弃数据报或者使用 ECN, 使部分对传输有适应能力的数据流, 如 TCP 数据流自我调整, 降低数据的发送速率, 减轻网络的拥塞程度.

试验中获得的 u_{avg} 数据显示以上控制方法是可行的.

313　UFQ 算法伪代码

Upon each arriving packet P:

```
if ( ingress router)
  packet maring;
update average queue size avg_ queue;
if ( avg_ queue= > minth) {
  p_d calculation;
  if drop the packet with probability p_d;
    return;
  if ( p_d > 0)
    packet remarking;
  if avg_ queue= > maxth
    multiple decrease u_avg with weight;
  else
    update the average utility u_avg;
  return;
}
```

4 模拟分析

模拟的硬件平台是浙江大学至腾 2000 集群计算机. 使用面向对象模拟器 ns22[18]. 网络拓扑结构如图 1、2.

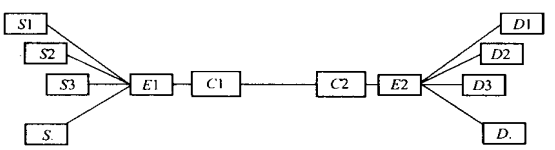

图 1 试验网络拓扑结构

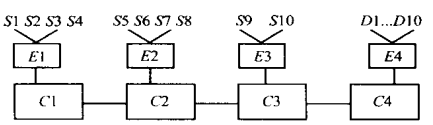

图 2 试验网络拓扑结构 2

在图 1、2 中, C 系列节点之间双向连接的带宽是 8Mbps, 延迟 2ms, 其余连接为双向 100Mbps, 2ms 的延迟. 显然, 网络中的瓶颈是 C 系列节点之间的连接. S、D 系列到 E 系列节点连接队列管理算法是 Drop Tail.

在各个试验中, 将使用 DiffServ 作为对照, 核心节点采用 RIO2C(Red with In and Out Coupled) 调度, 其参数如表 1.

通过一对源、目的 IP 地址来定义一个数据流, 数据流的定义和用户是相对应的. 数据报的长度都设置为 1Kb, 假设数据源都有无限的数据可供发送.

试验中各个用户的理论速率是依据第 2 部分的服务优化原则计算的. 以上条件在下面的模拟试验中都一致.

表 1 RIO2C 的参数

AF 类型	green	yellow	red
队列长度	L= 50	L= 50	L= 50
Max_ q	0. 8L	0. 4L	0. 2L
Min_ q	0. 4L	0. 2L	0. 1L
Max_ p	0. 02	0. 1	0. 2
W_q	0. 002	0. 002	0. 002

411 相同预约速率用户之间的公平性

使用图 1 的网络拓扑. S 系列十个 CBR (Constant Bits Rate) 数据源, 发送数据至 D 系列节点. 形成 C12C2 连接 8711 5%、11211 5%、150% 的负载水平. 十个用户的预约速率都是 800Kbps. 在 E 12C1、C12C2、C22E2 三对连接链路处分别使用 UFQ 和 DiffServ 进行试验, 在三个水平各重复 10 次. 在试验开始后, 数据源全部在第一秒内随机的启动发送数据. 每次的模拟时间都为 101s, 然后统计 12101s 内的平均速率, 取 10 次试验的平均值. 实验结果如图 3~ 5.

在 8711 5% 负载水平下, UFQ 和 DiffServ 表现性能差别不大.

在 112. 5% 负载水平下, S12S4 用户发送的数据流速率小于它们的预约速率, 不管是 UFQ 或是 DiffServ, 这几个用户都得到了 100% 的服务. S 52S 10 用户发送的数据流速率大于它们的预约速率, 在使用 UFQ 的情况下, S52S 10 的速率与理论速率吻合的非常好; 但是使用 DiffServ 时, 各个用户实际获得的网络带宽基本上伴随着发送速率的增大而不断增大, S5, S6 没有得到相应于预约速率的服务.

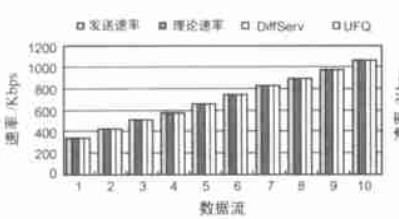

图 3 87% 负载相同预约时 C12C2 连接的带宽分配

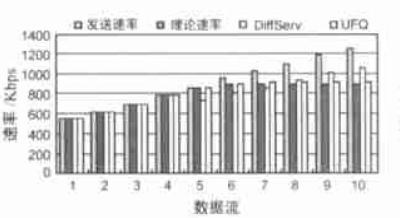

图 4 11211 5% 负载相同预约时 C12C2 连接的带宽分配

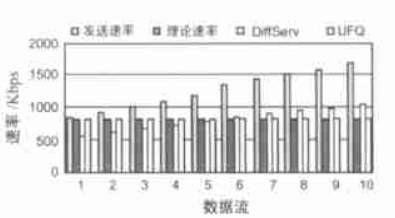

图 5 150% 负载相同预约时 C12C2 连接的带宽分配

在 150% 的负载水平下, 使用 UFQ 时十个用户获得的网络带宽就是其预约速率. 但是使用 DiffServ 时, S12S5 用户没有得到其预约的服务, 带宽分配没有表现出公平性. 其原因就是 DiffServ 采用的是分类标记分类调度的策略, 同一服务等级的调度使用 RED 算法, 没有考虑用户之间的公平性因素.

图 6 是试验中 112. 5% 负载时 UFQ 算法中服务水平的变化. 服务水平基本上围绕着 1. 10 值上下波动. 图中向上波动的尖峰是由于大量满意度大于平均服务水平的数据报的到达而产

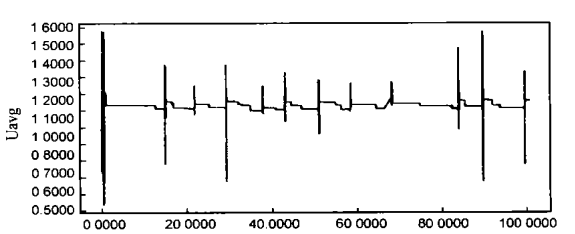

图 6 11211 5% 负载时连接服务水平的变化

生的, 而向下波动的尖峰是由于队列长度大于 maxth 时算法自动向下调整平均服务水平造成的, 曲线的波动显示算法的收敛速度非常快.

412　不同预约速率用户之间的公平性

第二个试验改变了十个用户的预约速率, 其他条件同一. S12S5 五个用户的预约速率是 600Kbps, S62S10 五个用户的预约速率是 1000Kbps. 在不同条件的重复试验完毕后, 试验结果如图 7~ 9.

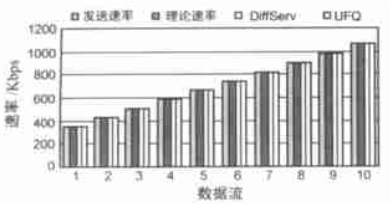

图 7　87% 负载不同预约时 C12C2 连接的带宽分配

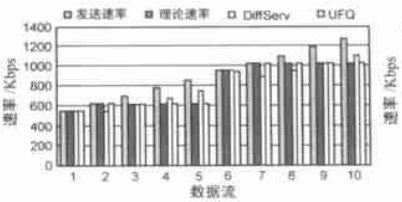

图 8　11215% 负载不同预约时 C12C2 连接的带宽分配

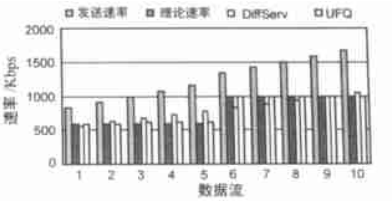

图 9　150% 负载不同预约时 C12C2 连接的带宽分配

413　适应性和非适应性数据流

使用 S12S5 五个 CBR 用户数据源, 同时发送(i* 160) Kbps 速率的数据至 D12D5. 取不同 i 值, 在 C12C2 连接上获得不同的 CBR 负载水平. S62S10 五个用户使用 TCP/ Reno 发送 FTP 数据. 预约速率都是 800Kbps. 各个负载水平各重复 10 次, 统计 FTP 的平均速率, 取每次试验中五个 TCP 连接的平均值. 得到如图 10 结果.

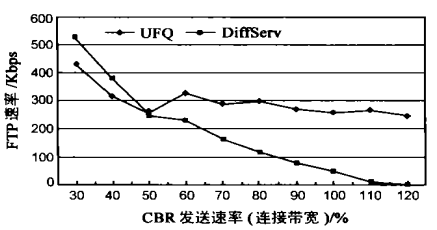

图 10　对适应性数据流的公平性

因 TCP 使用 AIMD 拥塞控制机制[16], 因此在拥塞时, 传输中数据报的每一次丢失, 都会引起传输窗口的积式减少, 不管是 UFQ 或是 DiffServ, FTP 的传输速率始终低于预约的速率. 从图中可看到, 当 CBR 数据流的速率达到连接带宽的 120% 时, 使用 DiffServ 时 FTP 的平均速率几乎降为零. 但使用 UFQ 算法管理时, 从 40%2150% 的 CBR 负载开始, FTP 传输速率就稳定在 2502450Kbps 之间. 在 120% 时两条折线的距离已经很大了, 这显示 UFQ 对非适应性数据流的控制能力强于 DiffServ, 适应性数据流在 UFQ 的调度下能够得到较好的服务.

414　多个拥塞连接、相同的预约速率

使用图 2 的网络拓扑. S12S10 十个用户发送 CBR 数据至 D12D10, 模拟时间 100s, 重复 10 次试验, 得到如表 2 结果.

使用 UFQ 时, 所有的用户都得到了接近理论的网络带宽. 但是使用 DiffServ 时, 在 C22C3 处, 连接的带宽并没有在 S12S8 用户之间公平的分配, 相反 S52S8 获得了较多的网络带宽, 总体上随着数据发送速率的增加获得的网络带宽也增加. 在 C32C4 处的情况类似.

在 87. 5% 负载水平下, 所得结果与试验一类似.

在 112. 5% 负载水平下, 各个用户带宽分配应如图中理论速率数据柱所示. 可以看到使用 UFQ 时网络带宽分配是与理论的数值非常接近的, 但是使用 DiffServ 却不是如此, S2, S7, S8 没有完全得到预约的服务.

在 150% 的负载水平下, 即使是不同水平的预约, UFQ 也能公平地在不同用户之间按比例合理地分配网络带宽. 但是使用 DiffServ, 网络带宽的分配对不同的用户是不公平的.

表 2　多个拥塞连接中的公平性　　(单位: Kbps)

用户	预约带宽	发送速率	C22C3			C32C4		
			理论带宽	Diffserv	UFQ	理论带宽	Diffserv	UFQ
S1	800	1000	1000	780. 2	977. 5	800	647. 0	782. 8
S2	800	1080	1000	841. 0	996. 0	800	698. 2	793. 6
S3	800	1160	1000	905. 5	998. 9	800	751. 8	795. 0
S4	800	1240	1000	966. 5	1000. 0	800	802. 9	794. 0
S5	800	1320	1000	1030. 7	998. 5	800	856. 1	794. 9
S6	800	1400	1000	1094. 9	998. 3	800	909. 6	792. 9
S7	800	1480	1000	1155. 5	1001. 7	800	958. 7	793. 8
S8	800	1560	1000	1220. 1	1002. 8	800	1012. 2	796. 6
S9	800	840	-	-	-	800	647. 8	790. 5
S10	800	920	-	-	-	800	709. 8	797. 0

5　算法的应用

UFQ 的应用需在边缘路由保持每个用户的状态, 在网络的核心需改变路由的调度算法, 这是应用的代价. 但在第四部分看到, 它能够对用户公平的分配瓶颈带宽, 总体上提高数据网络传输服务中对于用户预约的公平性, 同时控制了拥塞.

对于数据报的标记, 这里使用的是[6]介绍的 DPS(Dynamic Packet State) 技术. DPS 技术使用 IP 首部的 DS 字段的 4 位和实际网络传输中不经常使用的 13 位并结合一种动态编码技术对数据报进行标记. 另外, 由 RFC2474 可知, IP 首部的 DS 字段 DSCP(Differentiated Services Code Point) 中的组 2、3 是试验/ 本地/ 扩展使用的. 因此也可以使用组 2、3 中的 5 位对数据报的用户满意度编码标记, 其中假设用户进入网络的数据流速率都小于 2 倍的预约速率(实际应用中用户速率一般小于预约速率). 这样的编码在满足性能要求的情况下大大的降低了复杂性, 但是相对的也增大了数据报调度的粒度, 并且降低了同一编码数据报之间服务的公平性, 这其实是在复杂性/ 性能之间的一个折衷. 初步的试验显示其效果十分接近本试验所得数据, 作者正在做进一步的研究. 其方法如下:

2/3	XXXXX1	EXP/LU			
0< U[0105	0105< U[0110	0.10< U[0115	,	0.90< U[0195	0.95< U[1100
00001(1)	00010(2)	00011(3)	,	10011(19)	10100(20)
1100< U[1110	1110< U[1120	1120< U[1130	,	1180< U[1190	1190< U[2100
10101(21)	10110(22)	10111(23)	,	11101(29)	11110(30)
2100< U					
11111(31)					

算法的复杂性是应用的关键问题,但是随着技术的进步,ASIC 与 FPGA 内嵌的高性能网络服务处理器及大规模高性价比的缓存将为实施提供技术上的保障.

6 结论

本文提出了一种用户公平的活动队列管理算法 UFQ,它在网络边缘标记用户期望满意度,在网络核心结合网络的拥塞程度和数据报的满意度,进行队列的管理调度. 其目标在于不同的预约用户能够更加公平地获得网络资源. 相对于一般的公平排队机制,它不需要在每个网络传输节点都维护数据流的状态信息. 与 DiffServ 比较,UFQ 能更公平地分配网络带宽资源. 本算法具有一定的应用价值,同时它所使用的资源分配策略对网络资源的分配方法研究也极具借鉴意义.

下一步工作可以研究适应性数据流与 UFQ 算法相互作用的问题,基于游戏理论的算法 Nash 平衡性分析,合理的标记策略,以及如何结合网络流量工程对用户提供有保证的服务等等.

参考文献:

[1] Ravi Mazumdar, Lorne G Mason, Christos Douligeris. Fairness in network optimal flow control: optimality of product forms [J]. IEEE Transactions on Communications, 1991, 39(5): 775-782.

[2] V Jacobson. Congestion avoidance and control [J]. Computer Communication Review, 1988, 18(4): 314-329.

[3] A Demers, S Keshav, S Shenker. Analysis and simulation of a fair queuing algorithm [DB/OL]. http://www.acm.org/sigcomm, Proceedings of ACM SIGCOMM1989, 1989-09.

[4] S Floyd, V Jacobson. Link sharing and resource management models for packet networks [J]. IEEE/ACM Transactions on Networking, 1995, 3(4): 365-386.

[5] Dong Lin, Robert Morris. Dynamic of random early detection [DB/OL]. http://www.acm.org/sigcomm, Proceedings of ACM SIGCOMM1997, 1997-09.

[6] Ion Stoica. Stateless core: a scalable approach for quality of service in the Internet [D]. USA: CMU, 2000.

[7] Albert Banchs. User Fair Queing: Fair allocation of bandwidth for users [DB/OL]. http://www.ieee2infocom.org, Proceedings of INFOCOM 2002, 2002-06.

[8] Zhi2ruo CAO, Zheng WANG, Ellen Zegura. Rainbow fair queueing: fair bandwidth sharing without per2flow state [DB/OL]. http://www.ieee2infocom.org, Proceedings of INFOCOM 2000, 2000-03.

[9] R Braden, D Clark, S Shenker. Integrated services in the Internet architecture: an overview [DB/OL]. http://www.ietf.org, RFC 1633, 1994-06.

[10] S Blake, D Black, M Carlson, E Davies, Z Wang, W Weiss. An architecture for differentiated services [DB/OL]. http://www.ietf.org, RFC 2475, 1998-12.

[11] Koushik Kar, Saswati Sarkar, Leandros Tassiulas. A simple rate control algorithm for maximizing total user utility [DB/OL]. http://www.ieee2infocom.org, Proceedings of INFOCOM 2001, 2001-04.

[12] F Kelly, A Maulloo, D Tan. Rate control for communication networks: shadow prices, proportional fairness and stability [J]. Journal of Operations Research Society, 1998, 49(3): 237-252.

[13] J Saltzer, D Reed, D Clark. End2to2end arguments in system design [J]. ACM Transactions on Computer Systems, 1984, 2(4): 277-288.

[14] David D Clark, John Wroclawski, Karen R Sollins, Robert Braden. Tussle in cyberspace: defining tomorrow. s Internet [DB/OL]. http://www.acm.org/sigcomm, Proceedings of ACM SIGCOMM 2002, 2002-08.

[15] 刘锦德, 刘后铭, 周明天, 等. 计算机网络大全[M]. 北京: 电子工业出版社, 1996.

[16] S Floyd, V Jacobson. Random early detection gateways for congestion avoidance [J]. IEEE/ACM Transactions on Networking, 1993, 1(4): 397-413.

[17] D M Chiu, R Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks [J]. Computer networks and ISDN systems, 1989, 17(1): 1-14.

[18] ns22 [CP/OL]. http://www.isi.edu/nsnam/ns/.

作者简介:

徐 建 男, 1975 年 2 月出生于浙江缙云, 计算机科学技术专业博士研究生, 主要研究方向为操作系统, 计算机网络等.

李善平 男, 1963 年 8 月出生于浙江宁波, 计算机应用专业博士, 教授, 博士研究生导师, 主要研究领域为本体论, 语义 Web, 嵌入式操作系统开发应用, 信息集成技术等.