基于 Virtual Clock 调度算法的接入允许控制算法

杨 帆, 刘增基
(西安电子科技大学综合业务网国家重点实验室, 陕西西安 710071)

摘 要: 本文针对于 Virtual Clock 调度算法提出了一种基于生存期的接入允许控制算法. 原有 Virtual Clock 的接入允许控制算法没有考虑到系统中在有连接建立和拆除的情况下如何动态分配带宽, 致使分组的时延无法确保. 本文从 Virtual Clock 算法的参考模型出发, 对带宽释放和分配的时机作出规定. 理论分析和仿真结果表明, 本文的算法能够保证 Virtual Clock 算法的时延特性. 本文的思路对于其他调度算法的接入允许算法也具有参考意义.

关键词: 接入允许控制算法; 分组调度算法; 时延
中图分类号: TN913.24 文献标识码: A 文章编号: 0372-2112 (2002) 07-1092-04

The Admission Control Algorithm Based on the Virtual Clock Packet Scheduling Algorithm

YANG Fan, LIU Zeng-ji
(*National Key Lab of ISN in Xidian University, Xi'an, Shaanxi 710071, China*)

Abstract: A new admission control algorithm for the virtual clock packet scheduling algorithm is proposed in this paper. The existing admission control scheme of the virtual clock algorithm does not consider the circumstances when sessions are set up and torn down. This may worsen packet delay property. Based on the reference model of virtual clock algorithm, the new algorithm prescribe the condition when the bandwidth can be released and allocated. Theoretical and simulation results show this algorithm can fully guarantee the delay property of virtual clock algorithm. The idea of this algorithm can also be applied to other admission control scheme of packet scheduling algorithm.

Key words: admission control algorithm; packet scheduling algorithm; delay

1 引言

分组调度算法对于分组交换系统中服务质量(QoS)的保证至关重要. 分组公平排队调度算法(PFQ: packet fair queueing)是被广泛研究的一类调度算法. 在 PFQ 算法中 Virtual Clock[1,3], FFQ[5], SPFQ[5], Leap Forward Virtual Clock[2] 及 WF^2Q+ [4]具有较低的实现复杂度, 能够满足实时性业务的时延要求. 其中 Virtual Clock 算法计算最简单, 是 PFQ 算法中的一种基础算法. 由于分组调度算法往往要与接入允许控制算法相结合才能确保连接(尤其是实时性业务连接)的 QoS, 因此研究基于 Virtual Clock 调度算法的接入允许控制算法具有较强的理论和实用意义.

2 现有基于 Virtual Clock 的接入允许算法的缺陷

在 Virtual Clock 算法中, 系统为每个连接 i 分配了一个服务速率 r_i. 现有 Virtual Clock 算法的接入允许控制算法要求

$$\sum_{i\in V} r_i \le C \tag{1}$$

其中 C 为交换机中一条输出链路的容量, V 为该链路上复用的连接集合.

设连接 i 第 k 个到达系统的分组为 p_i^k, 设 p_i^k 的长度为 L_i^k, p_i^k 到达系统的时刻记为 a_i^k. Virtual Clock 算法为 p_i^k 计算一优先级(或称为虚拟完成时间) F_i^k, 在选送分组到输出链路上时, Virtual Clock 算法总是选择虚拟完成时间最小的分组. F_i^k 的计算方法为:

$$F_i^k = \max\{a_i^k, F_i^{k-1}\} + L_i^k / r_i,\ F_i^0 = 0 \tag{2}$$

设 p_i^k 从链路上完全被输出的实际时刻为 f_i^k, Virtual Clock 算法认为, 只要接入允许控制算法能够保证式(1)成立, 则 f_i^k 满足:

$$f_i^k \le F_i^k + L_{\max}/C \tag{3}$$

其中 $L_{\max}$ 是系统中最长的分组长度.

我们对 Virtual Clock 算法进一步研究发现, 仅凭式(1)这个条件并不足以在 Virtual Clock 系统中确保每个连接对时延的要求. 下面给出一个例子.

设 Virtual Clock 系统中链路容量 $C=1$, 设系统中的分组长度都为 1. 在时刻 0 系统中一共建立了 5 条连接, 其中 $r_1=0.5$, $r_2=r_3=r_4=r_5=0.105$. 在时刻 0, 式(1)显然满足. 为了叙述简单起见, 假设连接 1 建立后只有一个分组需要发送. 因为 $F_1^1=2$, $F_2^1=F_3^1=F_4^1=F_5^1=9.5$, 因此系统先服务 p_1^1. 假定在时刻 1 系统服务完 p_1^1 后, 连接 1 立即被拆除. 由式(1), 在时刻 1

收稿日期: 2000-12-18; 修回日期: 2001-11-23

系统可以接入新的服务速率为 0.5 的连接. 设连接 6 此时被接入, r_6= 0.5. 又设连接 6 也只有一个分组要发送, F_6^1= 3, 因此系统在时刻 1 选择 p_6^1 服务. 同样的情况在时刻 2、3、4、5、6、7、8 都发生, 连接 7、8、9、10、11、12、13 在时刻 2、3、4、5、6、7、8 被允许接入系统, 这些连接的服务速率都为 0.5, 且都只有一个分组需要发送. 在时刻 2、3、4、5、6、7、8, p_6^1、p_7^1、p_8^1、p_9^1、p_{10}^1、p_{11}^1、p_{12}^1分别离开系统. 因为 F_{13}^1= 10, 所以在时刻 9、10、11、12, 系统为连接 2、3、4、5 服务. f_3^1= 11, f_4^1= 12, f_5^1= 13, f_{13}^1= 14. 显然对于 p_3^1、p_4^1、p_5^1 及 p_{13}^1, 式(3)不成立. 因此在这个例子中, Virtual Clock 算法无法满足由式(3)确定的时延上界.

Virtual Clock 算法的时延特性之所以不能得到保证, 主要是因为现有的 Virtual Clock 接入允许控制算法有可能使得 Virtual Clock 系统的容量在一定时期内被超出. 从 Virtual Clock 算法中分组的虚拟完成时间的计算中可以看出, Virtual Clock 算法中假设了一个参考模型, 这一模型中每个连接 i 并行地接受固定速率为 r_i 的服务器的服务, F_i^k 可以看作是在该服务器的服务下, p_i^k 被服务完的时间. 但由于 Virtual Clock 系统只能串行为每个连接服务, 所以一个分组在 Virtual Clock 系统中的离去时刻可能远超前于其在固定速率服务器服务下的离去时刻. 从宏观上讲, 在某个时期内 Virtual Clock 系统的服务容量可以看成是该时期的每个连接的服务器的服务容量相加组成的. 我们在将系统容量在各个连接之间进行分配时, 应该以每个连接的固定速率服务器的工作情况为基准. 一个分组在 Virtual Clock 系统中服务被终结, 并不意味着其在固定速率服务器中也被服务完了, 如果将一个连接的最后一个分组在 Virtual Clock 系统中被服务完作为该连接的服务带宽可以被释放的标志, 将使得这一带宽在系统中过早被释放, 从而使得原本仍在固定速率服务器中被占用的带宽又被分配给新的连接使用, 导致系统总的服务容量被超出.

在上例中, 连接 1 的 p_1^1 在固定速率服务器中要等到时刻 2 才离去, 但由于它在 Virtual Clock 系统中在时刻 1 就离去了, 因此在时刻 1, 连接 1 的服务带宽又交给连接 6 使用, 这一现象在时刻 2 到时刻 8 都发生. 致使从时刻 1 到时刻 9, 各个连接实际使用的服务容量总和为 1.42, 超出了系统容量 1, 系统接纳的业务量超出了同期的服务能力, 因而系统的时延特性无法得到保证.

3　基于连接生存期的接入允许控制算法

由于 Virtual Clock 系统中现有接入允许控制算法存在上述缺陷, 因此本文提出基于生存期的接入允许控制算法. 首先给出以下相关定义.

一个连接的生存期是指该连接从建立到该连接的最后一个分组的虚拟完成时刻之间的时间. 如果该连接拆除时, 其生存期已经结束, 则分配给该连接的服务带宽可以被收回, 变为可分配带宽. 否则分配给该连接的带宽需要等到生存期结束后, 才能变为可分配带宽.

在 Virtual Clock 系统中, 如果有连接有分组等待发送, 则称系统处于忙期.

记 $t_{1,is}$ 为连接 i 在 t_1 时刻后生存期的起始时刻, 即如果连接 i 在 t_1 时刻之前建立, 则 $t_{1,is}$= t_1, 如果连接 i 在 t_1 时刻后建立, 则 $t_{1,is}$ 为连接 i 建立的时刻.

记 $t_{2,ie}$ 为连接 i 生存期在 t_2 时刻前的截止时刻, 即如果连接 i 在 t_2 时刻生存期还未结束, 则 $t_{2,ie}$= t_2, 否则 $t_{2,ie}$ 为连接 i 最后一个分组的虚拟完成时刻.

基于生存期的 Virtual Clock 接入允许控制算法为: 在某一时刻 t, 只有当该时刻系统中的可分配带宽大于等于 r_i 时, 服务速率为 r_i 的连接 i 才被允许接入系统.

引理 **1**　在 Virtual Clock 系统中, 采用基于生存期的接入允许控制算法能够使得在任意一段时间间隔$[t_1, t_2]$内处于生存期的连接 i 满足

$$\sum_i r_i \cdot (t_{2,ie} - t_{1,is}) \le C \cdot (t_2 - t_1) \tag{4}$$

证明　采用归纳法. 第一步: 记 t_{n1} 为系统中第一次有连接建立的时刻, 设该时刻系统中建立的连接为连接 1, 连接 2, …, 连接 m, 由接入允许控制算法有

$$\sum_{i=1}^{m} r_i \le C \tag{5}$$

记 t_{n2} 为系统中第二次有连接建立的时刻, 在 t_{n1} 与 t_{n2} 之间的任意时间段$[t_1, t_2]$, 对于任意的连接 $i \in \{1, 2, \cdots, m\}$, 由 $t_{1,is}, t_{2,ie}$ 的定义可知, $t_1 \le t_{1,is}, t_{2,ie} \le t_2$. 记 I_{t_1,t_2} 为$[t_1, t_2]$期间处于生存期的连接集合, 显然有 $I_{t_1,t_2} \subseteq \{1, 2, \cdots, m\}$. 由式(5)有

$$\sum_{i \in I_{t_1,t_2}} r_i \cdot (t_{2,ie} - t_{1,is}) \le \sum_{i=1}^{m} r_i \cdot (t_{2,ie} - t_{1,is}) \le C \cdot (t_2 - t_1) \tag{6}$$

第二步: 记 t_{nk} 为系统中第 k 次有连接建立的时刻, $t_{n(k+1)}$ 为系统中第$(k+1)$次有连接建立的时刻. 假设在 t_{nk} 之前, 引理 1 都成立. 设在 t_{nk} 时刻系统接纳的连接为 $k_1, k_2, \cdots, k_m$. 设在 t_{nk} 时刻系统内处于生存期的连接为 $a_1, a_2, \cdots, a_n$, 则由接入允许控制算法可得

$$\sum_{i=a_1}^{a_n} r_i + \sum_{i=k_1}^{k_m} r_i \le C \tag{7}$$

对于任意时间段$[t_1, t_2]$ $(t_2 < t_{n(k+1)})$, t_1、t_2 的取值分以下几种情况

①$t_2 < t_{nk}$, 则由归纳假设, 式(4)成立.

②$t_2 \ge t_{nk}$, 且 $t_1 \ge t_{nk}$, 由式(7)显然有

$$\sum_{i=a_1}^{a_n} r_i \cdot (t_{2,ie} - t_{1,is}) + \sum_{i=k_1}^{k_m} r_i \cdot (t_{2,ie} - t_{1,is}) \le C \cdot (t_2 - t_1) \tag{8}$$

$\because I_{t_1,t_2} \subseteq \{a_1, a_2, \cdots, a_n\} \cup \{k_1, k_2, \cdots, k_m\}$

因此 $\forall i \in I_{t_1,t_2}$, 在$[t_1, t_2]$期间引理 1 成立. 即

$$\sum_{i \in I_{t_1,t_2}} r_i \cdot (t_{2,ie} - t_{1,is}) \le C \cdot (t_2 - t_1)$$

③$t_2 \ge t_{nk}$, 且 $t_1 < t_{nk}$, 记 $I_{t_1,t_{nk}}$ 为在$[t_1, t_{nk}]$期间处于生存期的连接集合, 由归纳假设可知

$$\sum_{i \in I_{t_1,t_{nk}}} r_i \cdot (t_{nk,ie} - t_{1,is}) \le C \cdot (t_{nk} - t_1) \tag{9}$$

记 I_{t_{nk},t_2} 为在 $[t_{nk}, t_2]$ 期间处于生存期的连接集合, 则

$$I_{t_{nk},t_2} \subseteq \{a_1, a_2, \cdots, a_n\} \cup \{k_1, k_2, \cdots, k_m\}$$

由式(7)可知,

$$\sum_{i \in I_{t_{nk},t_2}} r_i \cdot (t_{2,ie} - t_{nk,is}) \leq C \cdot (t_2 - t_{nk}) \tag{10}$$

由于 $I_{t_1,t_2} \subseteq I_{t_1,t_{nk}} \cup I_{t_{nk},t_2}$, 所以将式(9)、(10)相加可得

$$\sum_{i \in I_{t_1,t_2}} r_i (t_{2,ie} - t_{1,is}) \leq \sum_{i \in I_{t_1,t_{nk}}} r_i \cdot (t_{nk,ie} - t_{1,is}) + \sum_{i \in I_{t_{nk},t_2}} r_i \cdot (t_{2,ie} - t_{nk,is}) \leq C \cdot (t_2 - t_1)$$

由①、②、③知, 对于 $t_2 < t_{n(k+1)}$ 的任意时间段 $[t_1, t_2]$, 引理 1 均成立,

根据归纳法原理, 对于任意时间段 $[t_1, t_2]$, 引理 1 成立. 证毕.

引理 **2** 在 Virtual Clock 系统中, 采用基于生存期的接入允许控制算法, 如果 p_i^k 被服务完时, 在其之前被服务的分组的虚拟完成时间都小于 F_i^k, 则 $f_i^k \leq F_i^k$

证明 设 t_0 为距离 f_i^k 时刻最近的系统忙期的开始时刻. 假设 p_i^k 的实际服务完时间 $f_i^k > F_i^k$. 设 I_s 为从 t_0 到 f_i^k 系统服务过的连接集合. $\forall$ 连接 $j \in I_s$, 设 $p_j^{k_1}, p_j^{k_1+1}, \cdots, p_j^{k_j}$ 为在 $[t_0, f_i^k]$ 期间被服务的连接 j 的分组. 由 $t_{0,js}$ 的定义知 $a_j^{k_1} \geqslant t_{0,js}$, 由式(2)递推可得

$$F_j^{k_j} = \max\{a_j^{k_j}, F_j^{k_j-1}\} + \frac{L_j^{k_j}}{r_j} \geq a_j^{k_1} + \sum_{n=k_1}^{k_j} \frac{L_j^n}{r_j} \geq t_{0,js} + \sum_{n=k_1}^{k_j} \frac{L_j^n}{r_j} \tag{11}$$

$$\therefore \sum_{n=k_1}^{k} L_j^n \leq r_j \cdot (F_j^{k_j} - t_{0,js}) \tag{12}$$

式(12)对于 $\forall j \in I_s$ 都成立, 由假设 $f_i^k > F_i^k$ 可得

$$\sum_{j \in I_s, j \neq i} r_j \cdot (F_j^{k_j} - t_{0,js}) + r_i \cdot (f_i^k - t_{0,is}) > \sum_{j \in I_s} r_j \cdot (F_j^{k_j} - t_{0,js}) \geq \sum_{j \in I_s} \sum_{n=k_1}^{k} L_j^n \tag{13}$$

$$\because \sum_{j \in I_s} \sum_{n=k_1}^{k} L_j^n = C \cdot (f_i^k - t_0)$$

$$\therefore \sum_{j \in I_s, j \neq i} r_j \cdot (F_j^{k_j} - t_{0,js}) + r_i \cdot (f_i^k - t_{0,is}) > C \cdot (f_i^k - t_0) \tag{14}$$

设 I_{t_0,f_i^k} 为 $[t_0, f_i^k]$ 期间处于生存期的连接集合, 则 $I_s \subseteq I_{t_0,f_i^k}$. 设 $t_{f_i^k,je}$ 为连接 j 生存期在 f_i^k 之前的截止时刻, 如果有连接 $m \in I_{t_0,f_i^k}$ 在 $[t_0, f_i^k]$ 期间生存期结束, 根据假设, $p_m^{k_m}$ 是连接 m 在 f_i^k 之前被服务的最后一个分组且 $F_m^{k_m} \leq F_i^k < f_i^k$. 得到 $t_{f_i^k,me} = F_m^{k_m}$. 反之, 如果连接 m 的生存期在 f_i^k 之前没有结束, 则 $t_{f_i^k,me} = f_i^k > F_m^{k_m}$

$$\sum_{j \in I_{t_0,f_i^k}} r_j (t_{f_i^k,je} - t_{0,js}) \geq \sum_{j \in I_s, j \neq i} r_j \cdot (F_j^{k_j} - t_{0,js}) + r_i \cdot (f_i^k - t_{0,is}) > C \cdot (f_i^k - t_0) \tag{15}$$

式(15)与引理 1 相矛盾, 因此假设 $f_i^k > F_i^k$ 不成立, 引理 2 得证.

引理 **3** 在 Virtual Clock 系统中, 采用基于生存期的接入允许控制算法, 如果有虚拟完成时间大于 F_i^k 的分组在 p_i^k 之前被服务完, 则 $f_i^k \leq F_i^k + L_{max}/C$.

证明 设 p_l 是最后一个虚拟完成时间大于 F_i^k 而在 p_i^k 之前被服务的分组, 设 F_l 为其虚拟完成时间, f_l 为其实际服务完时间, L_l 为其长度, t_s 为 p_l 开始被服务的时刻. 记 I_{sl} 为在 $[f_l, f_i^k]$ 期间被服务过的连接的集合. $\forall$ 连接 $j \in I_{sl}$, 记 $p_j^{l_1}, \cdots, p_j^{l}$ 为在 $[f_l, f_i^k]$ 期间服务的连接 j 的分组. 利用式(2)递推得,

$$F_j^l \geq \max\{a_j^{l_1}, F_j^{l_1-1}\} + \sum_{n=l_1}^{l} \frac{L_j^n}{r_j} \geq a_j^{l_1} + \sum_{n=l_1}^{l} \frac{L_j^n}{r_j} \tag{16}$$

$$\therefore \sum_{j \in I_{sl}} r_j \cdot (F_j^l - a_j^{l_1}) \geq \sum_{j \in I_{sl}} \sum_{n=l_1}^{l} L_j^n = C \cdot (f_i^k - f_l) \tag{17}$$

由于从 f_l 时刻到 f_i^k 时刻被服务的分组的虚拟完成时间都小于等于 F_i^k, 所以 $F_j^{l_1} \leq F_i^k < F_l$, 而 $p_j^{l_1}$ 在 p_l 后被服务, 只可能是因为 $p_j^{l_1}$ 到达系统时 p_l 已经开始服务或已经被服务完了,

$$\therefore a_j^{l_1} \geq t_s \quad \text{假设 } f_i^k > F_i^k + \frac{L_{max}}{C}$$

$$\therefore C \cdot (f_i^k - f_l) = C \cdot (f_i^k - \frac{L_l}{C} - t_s) > C \cdot (F_i^k + \frac{L_{max}}{C} - \frac{L_l}{C} - t_s) \geq C \cdot (F_i^k - t_s) \tag{18}$$

记 I_{t_s,F_i^k} 为在 $[t_s, F_i^k]$ 期间系统中处于生存期的连接集合, $\forall$ 连接 $j \in I_{sl}$, 连接 j 的分组 $p_j^{l_1}$ 在 p_i^k 之前被服务, 连接 j 显然应该 $[t_s, F_i^k]$ 期间中已处于生存期, 否则如果连接 j 在 F_i^k 时刻后才进入生存期, 则 $a_j^{l_1} > F_i^k$, 由式(2)可得 $F_j^{l_1} = a_j^{l_1} + L_j^{l_1}/r_j > F_i^k$. 因此 $p_j^{l_1}$ 是 p_l 之后又一个虚拟完成时间大于 F_i^k 且在 p_i^k 之前被服务的分组, 这与 p_l 是最后一个虚拟完成时间大于 F_i^k 而在 p_i^k 之前被服务的分组相矛盾. 所以 $\forall$ 连接 $j \in I_{sl}$, $j \in I_{t_s,F_i^k}$, 因此 $I_{sl} \subseteq I_{t_s,F_i^k}$.

$\forall$ 连接 $j \in I_{sl}$, 记 $t_{F_i^k,je}$ 为连接 j 生存期在 F_i^k 前的截止时刻, $\because F_j^l \leq F_i^k$, $\therefore F_j^l \leq t_{F_i^k,je}$. 由 $t_{s,js}$ 的定义可知, $t_{s,js} \leq t_s \leq a_j^{l_1}$. 由式(17)、(18)可得

$$\sum_{j \in I_{t_s,F_i^k}} r_j \cdot (t_{F_i^k,je} - t_{s,js}) \geq \sum_{j \in I_{sl}} r_j \cdot (t_{F_i^k,je} - t_{s,js}) \geq \sum_{j \in I_{sl}} r_j \cdot (F_j^l - a_j^{l_1}) > C \cdot (F_i^k - t_s) \tag{19}$$

式(19)与引理 1 相矛盾, 所以假设 $f_i^k > F_i^k + L_{max}/C$ 不成立, 因此引理 3 成立. 证毕.

定理 在 Virtual Clock 系统中, 采用基于生存期的接入允许控制算法, 对于任意分组 p_i^k, 有 $f_i^k \leq F_i^k + L_{max}/C$.

证明: 分组 p_i^k 在被服务之前, 被系统服务的分组或者虚拟完成时间都小于 F_i^k, 或者有虚拟完成时间大于 F_i^k 的分组在 p_i^k 之前被服务. 前一种情况引理 2 证明有 $f_i^k \leq F_i^k$, 后一种情况引理 3 证明 $f_i^k \leq F_i^k + L_{max}/C$. 因此定理成立.

4 计算机仿真

在仿真中我们设定 $C = 155$Mbit/s, 系统中共存在 6 类业务, 前 4 类业务为具有 on-off 特性的突发性业务, 服务速率分别为 $C/8$、$C/16$、$C/32$、$C/64$. 后两类业务为恒定比特率业务,

服务速率为 $C/128$、$C/256$, 系统中分组长度为 53byte. 我们在仿真中设定第 1 到第 6 类业务中容纳的连接数目分别为 1、3、8、4、16、64. 为了比较现有 Virtual Clock 算法的接入允许控制算法与基于生存期的接入允许控制算法的性能差别, 我们设定: 在基于生存期的接入允许控制算法中, 每当一个连接的生存期结束后, 一个相同服务速率的新连接被立即允许接入系统. 而在仿真现有算法时, 当一个连接拆除时, 以概率 p 产生一个相同服务速率的新连接接入系统, 如果新连接没有立即产生, 则随机延时一段时间后再以概率 1 产生新连接接入系统. 令 $r_i^k = F_i^k + L_{max}/C - f_i^k$, r_i^k 表示了 p_i^k 离开系统的实际时刻与理论上限的差值. 令 $j(\min r_i^k)$ 表示第 j 类业务中 r_i^k 的最小值. 在表 1 中给出了本文算法与现有算法在 p 取不同值的情况下各类业务 r_i^k 的最小值. 可以看出, 在本文算法中分组离开系统的时刻都没有超过理论上限, 而在现有算法中, 即使在 $p = 0.3$ 的情况下, 现有算法中仍有分组的离去时刻超过理论上限.

表 1　本文算法与现有算法在 p 取不同值时各类业务 r_i^k 最小值的比较(仿真时间 0.015s)

算法	1($\min r_i^k$)	2($\min r_i^k$)	3($\min r_i^k$)	4($\min r_i^k$)	5($\min r_i^k$)	6($\min r_i^k$)
本文算法	1.9×10^{-5}	3.3×10^{-5}	5.2×10^{-5}	7.7×10^{-5}	2.8×10^{-6}	2.8×10^{-6}
现有算法 $p=1$	-3.0×10^{-3}	-5.6×10^{-3}	-4.9×10^{-3}	-6.2×10^{-3}	-2.3×10^{-3}	-2.3×10^{-3}
现有算法 $p=0.7$	-7.9×10^{-4}	-2.7×10^{-3}	-3.1×10^{-3}	-4.5×10^{-3}	-7.0×10^{-4}	-6.6×10^{-4}
现有算法 $p=0.3$	1.9×10^{-5}	-2.0×10^{-4}	-3.5×10^{-3}	-2.2×10^{-3}	1.6×10^{-4}	8.9×10^{-5}

图 1 及图 2 给出了本文算法及现有算法在 $p = 0.7$ 的情况下第 2 类及第 5 类业务中的分组离开系统时的 r_i^k.

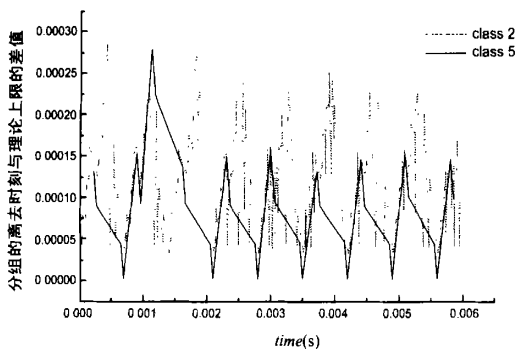

图 1　本文算法 r_i^k 曲线

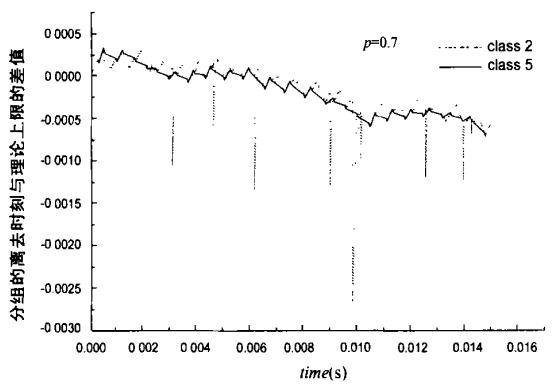

图 2　现有算法 r_i^k 曲线

从图 2 中可以看出, 在现有算法中在 $p = 0.7$ 的情况下, 有很多分组的实际离去时刻都超出了理论上限, 而图 1 中本文的算法能够确保每个分组的时延.

5　结论

现有 Virtual Clock 调度算法的接入允许控制算法仅要求系统中复用的各个连接的服务速率的总和不超过系统的容量, 而对一个新的连接的接入时机没有作出规定. 事实上, 即使一个连接已经被拆除, 但是由于该连接的分组被较早地服务完, 该连接占用的带宽并不是马上就变为可用. 本文提出基于生存期的接入允许控制算法, 在该算法中一个连接的带宽是否可再分配, 要等到该连接的生存期结束. 使用本文算法, 可以使 Virtual Clock 算法的时延特性得到满足. 由于目前很多调度算法都是以 Virtual Clock 算法为基础的, 因此本文提出的思路也适用于其它类似的调度算法.

参考文献:

[1] L Zhang. Virtual clock: a new traffic control algorithm for packet switching networks [A]. Proc. ACM SIGCOMM' 90 [C], 1990. 19-29.

[2] Subhash Suri, George Varghese, Girish Chandranmenon. Leap Forward Virtual Clock: A New Fair Queueing Scheme With Guaranteed Delays and Throughput Fairness [A]. Proc IEEE INFOCOM' 97 [C], 1997. 557-565.

[3] Norival R Figueira, Joseph Pasquale. An Upper Bound on Delay for the Virtual Clock service Discipline[J]. IEEE/ACM Trans Networking, 1995, 3(4): 399-408.

[4] Jon C R, Hui Zhang. Hierarchical packet fair queueing algorithms [J]. IEEE/ACM Trans Networking, 1997, 5(5): 675-689.

[5] D Stiliadis, A Varma. Efficient fair queueing algorithms for packet-switched networks [J]. IEEE/ACM Trans Networking, 1998, 6(2): 113-122.

作者简介:

杨　帆　男, 1973 年 3 月出生于陕西径阳, 分别于 1995 年和 1998 年获得西安电子科技大学通信与信息系统专业获学士、硕士学位, 现于西安电子科技大学 ISN 国家重点实验室攻读博士学位, 研究方向为现代通信网络中 QoS 保障机制及 ATM.

刘增基　男, 1937 年 12 月生于浙江丽水, 现为西安电子科技大学教授、博导、ISN 国家重点实验室主任、中国通信学会会士, 当前主要从事宽带网络技术的研究.