应用于无线信道的混合型冲突分解算法

盛 敏, 李建东, 江 帆
(西安电子科技大学 ISN 国家重点实验室、信息科学研究所、宽带无线通信实验室, 陕西西安 710071)

摘 要: 冲突分解算法是一种能够有效改善无线信道多址接入协议性能的方式. 传统的冲突分解算法包括树形分解算法和先到先服务的冲突分解算法, 但是这两种分解算法都存在着一定的不足. 本文提出了一种新型的混合型冲突分解算法(Hybrid Splitting Algorithm-HSA), HSA 算法继承了树形分解算法和先到先服务冲突分解算法的优点, 不仅考虑了碰撞分组的产生时间, 使先产生的分组先得到服务, 同时当系统中存在产生间隔比较接近的分组时, 采用树形分解算法, 使整个分解过程不再仅仅局限于分组的产生时间, 从而有效地减少了分解所需的总时隙数, 提高了冲突分解算法的性能. 理论分析和仿真结果显示 HSA 算法是一种正确可行的算法.

关键词: 冲突分解算法; 树形分裂算法; 先到先服务; 分组产生间隔; 重尾分布
中图分类号: TN919 文献标识码: A 文章编号: 0372-2112 (2005) 04-0692-05

Hybrid Collision Splitting Algorithm for Wireless Channel

SHENG Min, LI Jian-dong, JIANG Fan
(*State Key Lab & Information Science Institute & Broadband Wireless Lab, Xidian University, Xi'an, Shaanxi 710071, China*)

Abstract: Collision resolution is an effective strategy to improve the performance of medium access control protocol. Both tree splitting algorithm and First Come First Service (FCFS) splitting algorithm are classical algorithms of Splitting algorithm. But both of them have some limitations. In this paper, a novel splitting algorithm- Hybrid Splitting Algorithm (HSA) has been presented. Based on the tree splitting algorithm and FCFS splitting algorithm, the HSA has the packet transmitted in the order of their arrival partially, while if the packet generation time is too near to split using by FCFS, the HSA adopts the tree splitting algorithm. So, the whole splitting procedure has relations with both the generation time and the generation interval of packets. Simulation results show that HSA outperforms tree splitting and FCFS splitting algorithm.

Key words: collision resolution; tree splitting algorithm; FCFS splitting algorithm

1 引言

在一些无线网络中, 由于网络的分布式特性和临时性, 以及业务的突发性, 使得固定多址接入协议(如 TDMA[1], FDMA[1]等)、基于预约的多址接入协议(MACAW[2]等)都不能有效的发挥作用. 因此, 在这类网络中, 通常都是采用随机的多址接入协议(如 ALOHA[3], CSMA[4]等). 在随机多址接入协议中, 由于信道是系统中多个用户共享的资源, 当多个用户同时使用信道时, 各用户之间的传输必然发生碰撞. 因此, 在随机多址接入协议中, 一旦发生碰撞, 如何有效的分解碰撞, 尽可能的减少冲突分解的时间是冲突分解策略的主要作用. 目前, 经典的冲突分解算法有树形冲突分解算法[5,6] (Tree Splitting Algorithm)和先到先服务冲突分解算法[6] (FCFS Splitting Algorithm). 在树形冲突分解算法中, 参加碰撞的分组以某一概率随机的选择进入左集或右集, 并保证左集的优先级比右集的高, 即只有当左集分解成功后, 才能进入右集的分解. 而在先到先服务冲突分解算法中, 参加分解的分组按产生时间先后进行分解, 保证先到达的分组先成功传输. 虽然这两种算法都可以提高 MAC 层协议的性能, 但两者都有一定的缺陷. 树形冲突分解由于完全基于随机的方式, 当发生冲突的分组数比较多时, 系统的时隙利用率不高; 而 FCFS 分解算法由于基于时间调度, 可以减少空闲时隙, 但是当碰撞分组中存在产生间隔比较近的分组时, 用时间条件分解将需要进行多次分解.

本文提出了一种混合型的冲突分解算法, 该算法不再只依赖于分组的产生时间, 还充分考虑了分组的产生间隔, 不仅尽可能的使先产生的分组先得到服务, 同时还使得产生间隔比较接近的分组能够迅速的分解, 从而使系统的整体性能得到改善.

2 混合型冲突分解算法

如果将系统中发生碰撞的分组按其产生时间先后进行分解(FCFS 分解算法)可以使系统的最大通过率达到 0.4871[6],

收稿日期: 2004-05-27; 修回日期: 2004-12-25
基金项目: 国家自然科学基金和微软亚洲研究院联合资助项目(No. 60372048); 国家自然科学基金重大项目(No. 60496316); 高等学校优秀青年教师教学科研奖励计划

这显然优于树形分解算法的最大系统通过率 0.346[6]. 然而在采用 FCFS 冲突分解算法时, 当发生碰撞的分组中存在产生间隔非常接近的分组时, 性能就会迅速地恶化, 因此, 本文提出了混合型冲突分解算法(Hybrid Splitting Algorithm-HSA), HSA 的基本思想是首先按照碰撞分组产生的时间先后顺序进行分解, 即先执行 FCFS 分解算法, 当按照 FCFS 分解到一定次数还没有分解完成时, 就意味着这个冲突分解期中存在着产生间隔非常接近的分组, 此时 HSA 算法将采用树形分解算法, 即随机的对当前分解区间的分组进行冲突分解.

2.1　系统模型描述

为了便于描述系统的行为, 本文假定系统中的所有用户都工作在时隙状态下, 用户产生的分组长度定长. 系统中碰撞的分组将参与一系列的分解, 而此时新产生的分组将在系统内等待而不参与此次冲突分解行为. 定义“冲突分解期”(Collision Resolution Period-CRP)为从发生碰撞的那个时隙开始直到这次碰撞的所有分组都成功传输结束的时隙之间的总时隙数.

2.2　HSA 算法描述

混合型分裂算法(HSA)的基本思想是首先根据分组到达的时间进行冲突分解, 并力图保证先到达的分组最先传输成功. 同时还根据分组产生的时间间隔, 适时地采用树形冲突分解, 从而有效减少邻近分组之间需要分解多次的现象.

设 $T(k)$ 以前到达的分组都已发送完毕, 现在需确定从 $T(k)$ 开始, 长度为 $\alpha(k)$ 区间内到达的分组在第 k 个时隙中传输. 称区间 $\alpha(k)$ 为指配区间(AI). 从 $T(k)+\alpha(k)$ 至当前传输时刻称为等待区间(WI). HSA 算法的主要功能是根据冲突分解的情况, 动态地调整指配区间的长度和起始时刻, 并根据分解的次数, 切换分解策略. HSA 分裂算法的一个示意图如图 1 所示.

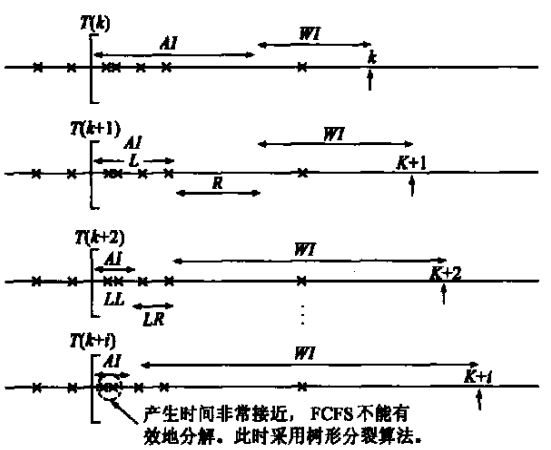

图 1 HSA 算法的分解过程

图 1 中, 在 $[T(k), T(k)+\alpha(k)]$ 内到达的分组在第 k 个时隙内传输. 如果第 k 个时隙发生了碰撞, 则将指配区间分为两个相等部分, 左集(L)和右集(R). 其中, $L=[T(k+1), T(k+1)+\alpha(k+1)]$, $T(k+1)=T(k)$, $\alpha(k+1)=\frac{1}{2}\alpha(k)$, 且 L 首先在第 $k+1$ 个时隙内传输. 如果第 $k+1$ 时隙发生碰撞, 这说明 L 区间至少有 2 个分组. (由于 L 区间的碰撞没有给出任何关于 R 区间的信息, 因此, 将 R 区间划入到等待区间内, 这次碰撞分解不再考虑 R 区间). 很显然, 图 1 给出的实例是在 $k+1$ 时隙发生碰撞的情况, 将 L 分为 LL 和 LR. 其中 $LL=[T(k+2), T(k+2)+\alpha(k+2)]$, $T(k+2)=T(k+1)$, $\alpha(k+2)=\frac{1}{2}\alpha(k+1)$. 在该例中 LL 中包含两个产生时刻非常接近的分组, 则 FCFS 无法将其分解, 分解将一直进行. (还有一种情况是第 $k+1$ 个时隙空闲, 则必然在第 $k+2$ 个时隙内碰撞. 由于 R 区间必定包括 2 个以上的分组, 则在第 $(k+1)$ 时隙结束时刻, 立即进行分解, 得 RL 和 RR 两个相等的区间. 其中, $RL=[T(k+2), T(k+2)+\alpha(k+2)]$, $T(k+2)=T(k+1)+\alpha(k+1)$, $\alpha(k+2)=\alpha(k+1)/2$. 其基本工作过程和图 1 类似, 这里就不再给出图形描述). 在 HSA 算法中, 当按时间顺序分解到一定次数后开始采用随机分解的算法(在后面有关 HSA 算法分析时, 可以看到最佳的次数是 4, 即若分解 4 次还未能用 FCFS 分解开, 则采用树形分解算法), 并且在左集分解结束后, 进入右集的分解. 右集分解过程和左集类似. 如果当前指配区间内所有碰撞分组分解成功后, 一个 CRP 结束.

2.3　HSA 算法分析

由前面的描述可知, HSA 算法首先采用和 FCFS 冲突分解算法一致的行为, 当分解达到一定次数后, 采用树形分解算法. 很显然, 对于 HSA 算法而言, 分解到什么程度可以采用树形分裂算法将是影响 HSA 算法性能的关键参数. 为了分析 HSA 算法, 本文作如下定义:

(1) 系统最佳切换状态: HSA 算法需要切换分解策略(从 FCFS 切换至树形分解算法)时系统所处的状态;

(2) 系统最佳切换分解次数(Optimal-State-Change-Splitting-Number, 我们将其简写为 OSN): HSA 算法处于系统最佳切换状态时已经分解的次数.

本文采用如下方法来分析 HSA 算法: 首先讨论执行 HSA 算法时, 系统怎样才能处于最佳系统切换状态; 然后再求解 HSA 算法处于最佳切换状态时的最佳切换分解次数.

2.3.1　HSA 算法的系统最佳切换状态

由前面的定义可知, 所谓 HSA 算法的最佳切换状态是指 HSA 在该状态切换至采用树形冲突分解算法是最佳的. 因此, 需要来分析一下此时 HSA 算法的状态. 由于此时 HSA 算法将采用树形冲突分解算法, 即在发生碰撞后, 各分组独立的以某一概率从其所属的集合中分别进入左集和右集. 很显然, 如果系统此时只有 1 个分组或没有分组, 则只需要一个时隙; 如果系统中的分组数大于等于 2, 则需要分解. 因此, 下面分析一下什么情况下树形冲突分解的效能比较好.

® n 个元素的集合中有 i 个进入同一集合的概率分析

在有 n 个分组的集合中, 各分组独立的以概率 p 随机的选择进入左集还是右集. 有 i 个分组进入左集的概率为

$$Q_i(n)=\binom{n}{i}p^i(1-p)^{n-1} \qquad (1)$$

如果有 i 个分组进入左集, 则有 $n-i$ 个分组将进入右集, 分

解所需的总时隙数为

$$K = 1 + k_i + k_{n-1} \qquad n \geqslant 2 \tag{2}$$

这里, 1 表示 n 个用户碰撞的那个时隙, k_i 表示进入左集的用户分解所需的时隙数, 而 k_{n-i} 表示进入右集的分组分解所需的时隙数.

显然: 对于 $n \geqslant 2$ 有

$$E\{K\} = 1 + \sum_{i=0}^{n} Q_i(n)(E\{k_i\} + E\{k_{n-i}\}) \tag{3}$$

化简得:

$$E\{K\} = \frac{1 + \sum_{i=0}^{n-1} Q_i(n) K_i + \sum_{i=1}^{n} Q_i(n) K_{n-i}}{1 - Q_0(n) - Q_n(n)} = \frac{1 + \sum_{I=0}^{N-1} [Q_i(n) + Q_{n-i}(n)] K_i}{1 - Q_0(n) - Q_n(n)} \tag{4}$$

通过计算, 可以得

$$E\{K\} = 1 + \sum_{i=2}^{n} \binom{n}{i} \frac{2(i-1)(-1)^i}{[1 - p^i - (1-p)^i]} \qquad n \geqslant 2 \tag{5}$$

并且有

$$\begin{aligned} n &= 0, \quad E\{K\} = 1; \\ n &= 1, \quad E\{K\} = 1; \end{aligned} \tag{6}$$

® 数值分析

通常, 在树形分裂算法中, 都采用完全随机的的方式决定某一分组是进入左集还是进入右集, 因此, 有 $p = \frac{1}{2}$. 由上面的分析, 可以得到表 1 的结果.

表 1 树形冲突分解算法的吞吐率统计表

n	1	2	3	4	5	6	7
$E\{K\}$	1.0000	5.000	7.6667	10.5238	13.4191	16.3131	19.2010
$S = n/E\{K\}$	1.0000	0.4000	0.3913	0.3801	0.3726	0.3646	0.3622

从表 1 中, 可以看出对于树形冲突分解算法, 当系统中的分组数小于等于 2 时, 系统的通过率高于或接近于 FCFS 的通过率(高于 FCFS 的最大平均通过率 0.4871) . 因此对于 HSA 算法而言, 可以认为当从 FCFS 分解算法切换到树形冲突分解算法时, 如果此时参与分解的分组个数小于等于 2, 则可以充分发挥树形冲突分解的优势, 并且使系统的分解行为不再受时间的约束.

综合上面的分析, 得出结论 1.

结论 1 对于 HSA 算法, 其最佳切换状态是当 HSA 算法开始切换分解算法时, 系统中剩余的参与碰撞分解的分组数小于等于 2.

2.3.2 HSA 算法系统最佳切换分解次数 从 2.3.1 节中, 得出了系统的最佳切换状态, 下面讨论 HSA 算法怎样才可以进入最佳切换状态, 即求解系统的最佳切换分解次数.

由于在 HSA 算法的前 OSN 次(系统最佳切换分解次数) 分解过程中采用的是 FCFS 分裂算法, 所以很显然, 如果左集碰撞则右集不可能获得任何有价值的信息. 如果是这种情况, 在分析的过程中指配区间将不再考虑右集, 而将右集归入到下一次新的冲突分解期里去考虑.

为了便于分析, 令分组的到达是服从参数为 λ 的 Poisson 过程, 开始时的指配区间长度为 α, 很显然, 按照 HSA 前 m($m \leqslant$ OSN) 次分解的原则, 每分解一次, 考察的区间的长度应该为 $2^{-i}\alpha$, 并且在该区间内的分组数为

$$G_i = \lambda \cdot 2^{-i}\alpha = 2^{-i}\lambda\alpha \tag{7}$$

并且系统在每一次的分解区间内(不论是左集还是右集) 都是服从参数为 G_i 的 Poisson 分布.

从前面的分析中, 可以知道最佳切换状态是网络中参与分解的分组数小于等于 2 个的情况, 下面来讨论一下在参数 $\lambda\alpha = 1.266$ 时(在此参数状态下, FCFS 性能达到最佳[6]) , 对于 HSA 分解算法, 系统最佳切换分解次数 OSN 的具体值.

假设在第 i 次分解后, 左集内的分组数为 x_L, 右集内的分组数为 x_R. 显然在该指配区间内系统中总的分组数为 $x_L + x_R$. 由于需要再次进行分解, 所以必然有 $x_L + x_R \geqslant 2$. 下面将讨论前 OSN 次分解时, 在当前区间发生碰撞而分解后进入左集的分组个数小于等于 2 的概率.

$$\begin{aligned} p_r &= p\{\text{当前的左集分组数小于等于 2} \mid \text{未分解前的区间里的分组数大于等于 2}\} \\ &= P\{x_L \leqslant 2 \mid x_L + x_R \geqslant 2\} = \frac{P\{x_L = 0\}P\{x_R \geqslant 2\} + P\{x_L = 1\}P\{x_R \geqslant 1\} + P\{x_L = 2\}P\{x_R \geqslant 0\}}{P\{x_L + x_R \geqslant 2\}} \\ &= \frac{e^{-G_m}[1 - (1 + G_m)e^{-G_m}] + G_m e^{-G_m}(1 - e^{-G_m}) + \frac{G_m^2 e^{-G_m}}{2}}{1 - (1 + G_{m-1})e^{-G_{m-1}}} \end{aligned} \tag{8}$$

如果由式(8) 计算得到的概率足够大, 则认为此时采用树形冲突分解算法可以有效改善系统的性能. 对式(8) 作如下计算 $\min_m \lim p_r \to 1$, 解得的 m 值为 4. 该值即为系统的最佳切换分解次数.

结论 2 对于 HSA 算法, 其最佳切换分解次数为 4.

2.3.3 HSA 算法饱和吞吐率分析 由前面的分析已经计算出 HSA 算法的最佳切换分解次数为 4, 即 HSA 在分解的前 4 次将采用 FCFS 算法, 而以后将采用树形分解算法. 下面, 简单分析一下 HSA 算法的最大吞吐率.

令 HSA 算法分解超过 4 次的概率为 p_{change}, 则 $p_{change} = 1 - (1 + G_3)e^{-G_3} \approx 0.113$; FCFS 系统的最大稳定吞吐率为 $S_{FCFS} = 0.4871$[6], HSA 算法中采用的树形冲突分解的最大稳定吞吐率为 S_{Tree}, 由于在 HSA 算法中采用树形冲突分解算法时, 系统中的分组数至多为 2 个, 根据表 1 有 0.7, 则 HSA 算法的最大稳定吞吐率为

$$S_{HSA} = (1 - p_{change})S_{FCFS} + p_{change}S_{Tree} = 0.4871 * (1 - 0.113) + 0.113 * 0.70 \approx 0.51 \tag{9}$$

很显然, HAS 算法的最大稳定吞吐率高于 FCFS 算法的最大吞吐率. 这是由于 HSA 算法及时地调整了分解策略, 在不影响总体的时间服务原则基础上, 适当地降低了算法对时间的依赖性, 从而有效地避免了分组产生间隔对算法造成的长时间影响, 提高了系统的吞吐率.

结论 3 HAS 算法的最大稳定吞吐率为 0.51.

3 HSA 算法仿真

通过仿真验证了 HSA 算法的性能. 主要从两个方面对其

进行了考虑: 冲突分解过程中的吞吐率和平均冲突分解期. 所谓冲突分解过程中的吞吐率是指在冲突分解期中单位时间内(这里是一个时隙) 成功传输的平均分组数; 所谓平均冲突分解期是指完成一次冲突分解所需的平均时隙数.

由于 FCFS 算法的性能是优于树形分裂算法的, 所以在仿真中, 仅将 FCFS 算法的性能和 HSA 算法的性能进行了比较. 这里讨论了两种场景: 即分组的到达过程是一个泊松过程和分组的到达过程是一个重尾分布[7]– Weibull 分布的过程. 下面将对仿真结果进行分析. 在这里为了简化书写, 令 m 等于系统的最佳切换分解次数(OSN) .

3.1　分组到达过程是泊松到达过程

由于分组的到达过程是一个泊松过程, 所以由泊松过程的特点可以知道, 在一个比较小的区间内有两个或两个以上到达的概率是一个高阶无穷小, 并且泊松到达的二阶矩是一个常量[8] . 因此在研究泊松到达过程时, 网络中不可能存在着到达间隔非常小的两个分组. 但显然存在分解了 m 次还没有分解开来的情况.

从图中可以看出随着到达率的增大, 算法的吞吐率在减少而平均冲突分解所需的时隙数增大. 并且从图 2 中, 可以看出随着到达率的增加, 系统的吞吐率性能开始恶化, 这和传统的理论分析是吻合的. 当到达率比较小时, 由于分组数比较少, 所以系统只需要很少的次数就可以将碰撞的分组分解开来, 此时 FCFS 算法的性能和 HSA 的性能比较相近. 当到达率小于等于 4 时, HSA(m = 4) 和 FCFS 算法的平均偏差仅为 1.072%, 当到达率小于等于 9 时, HSA(m = 4) 和 FCFS 算法的平均偏差仅为 2.0225%, 此时由于发生碰撞的分组数比较少, 分解在还没有达到 m 次时, 就已经结束, 所以两种算法性能比较接近. 但是, 一旦到达率超过了 9 以后, HSA 算法的性能明显优于 FCFS, 即 HSA 算法突破了 FCFS 的吞吐率上限. 图 3 中的结论也表明了这一点, 图中到达率接近 10 后, FCFS 算法的性能迅速恶化, 而且显然 HSA 算法分解冲突所用的平均时隙数要少于 FCFS 算法. 从图中还可以看到对于 HSA 算法, 在达到过程是泊松过程时, m 等于 4 和 m 等于 6 时, 在性能上存在差别, 显然 m 等于 4 时系统的性能要优于 m 等于 6 时的.

为了验证 HSA 算法在到达过程为泊松过程时, m 取值对系统性能的影响, 针对不同的 m 取值的吞吐率性能进行了深入的仿真, 发现 m 取值越大时, 系统的性能越紧接于普通的 FCFS 分裂算法; 而 m 的取值越小时, 系统的行为越趋近于树形冲突分解(但由于还是 HSA 算法, 所以性能比普通的树形和 FCFS 都要好) . 即对于分解了多次的分组, 在分解到一定次数后, 越早采用随机的方式越有利于改善系统性能. 从图 4 可以看出, 显然当 m 等于 4 时, 系统的性能是最好的. 当到达率比较小时, m 的取值对系统而言, 不是一个起决定性作用的参数. 随着到达率的增大, 发现离最佳值比较近的 m 值对应的曲线的性能相对来说性能要好于离最佳值偏离比较远的 m 值曲线的性能. 但总的来说, 各曲线的吞吐率特性均好于 FCFS 分裂算法.

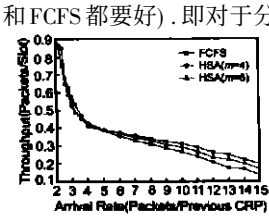

图 2 分组到达率与系统吞吐率曲线

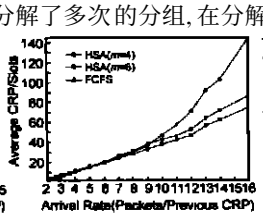

图 3 分组到达率与平均冲突分解期曲线

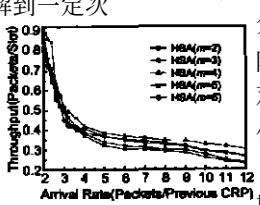

图 4 HSA 算法不同 m 值时性能曲线

3.2　分组到达过程是服从重尾分布—**Weibull** 分布的过程

为了更进一步的研究到达间隔对 HSA 算法性能的影响, 研究了分组的到达过程服从 Weibull 分布的场景. Weibull 分布是一个典型的具有重尾特性的分布, 其分布函数为

$$F(x) = 1 - e^{-(x/\beta)_\alpha} \qquad (10)$$

其中, 参数 α、β 都是实数, 且 α> 0, β> 0. β 被称为尺度参数; α 被称为形状参数, 它用来描述分组到达突发性的程度. 当 α = 1 时, Weibull 分布就“蜕化”为指数分布; 当 α< 1 时, 随着 α 的值越来越小, 分布的“重尾”特性就越来越明显, 即分组到达的突发性就越强.

因此, 在下面的研究中, 将重点研究 α 值对 HSA 算法性能的影响, 即研究分组到达的突发性对 HSA 算法性能的影响. 此时, 可以看到如果产生的分组突发性越强, 即产生的分组间隔越近, HSA 算法将具有不可比拟的优越性.

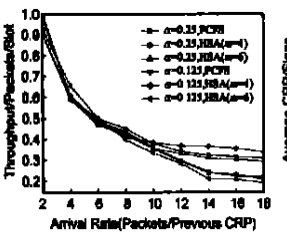

图 5 系统吞吐率曲线

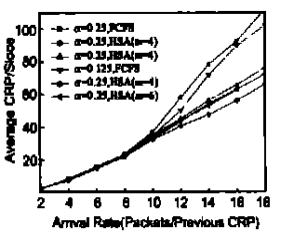

图 6 平均冲突分解期曲线

从图 5 中可以看出, 当系统到达率比较小时, 几种参数的情况都比较接近. 当到达率增加到一定程度后, 可以很明显的看出, FCFS 算法的通过率比较低, 而且随着分组数目的增多, 通过率也在逐渐下降, 这种性能的下降除了 FCFS 算法本身的吞吐率约束之外, 很重要的一个原因是由于碰撞的分组中存在着间隔非常近的分组, 从而使 FCFS 算法无法有效地将其分解开来, 而且从图 5 中还可以看出, FCFS 对于业务在强突发性情况下(即分组中存在产生间隔非常接近的情况), 性能比较差, 且突发性越强性能越差. 而 HSA 分裂算法恰好解决了这种分组到达时间很相近的情况, 使得通过率提高了很多. 从图中还可以看出对于 HSA 算法, m 取最佳值时, 系统的通过率相对较高, 因为这样做及时地用树形分裂算法来随机的划分碰撞分组中相距很近的分组, 大大避免了空闲时隙的出现. 尤其重要的是, 从图 5 中看到 *HSA* 算法对业务的突发性反映敏感, 突发性越强 *HSA* 算法的优越性越明显.

图 6 为分解同样分组数时, 各算法需要的分解期的长度. 很显然, 随着碰撞分组数的增加, 分解所需的时隙数随之增加. 由于碰撞的分组中存在产生间隔接近的分组, 所以如果采用 FCFS 分裂算法, 只

有将时间的间隔划分的非常小时才能够将碰撞的分组分解开,在分解的过程中出现了大量空闲的时隙.而如果采用 HSA 算法,则可以有效地避免上述情况的出现,此时分解过程不再受分组产生时间间隔的约束,而是完全随机的将碰撞分组分解,从而有效的减少了空闲的时隙数,从整体上减少了冲突分解期的长度.从图 6 中可以看出突发性越强 HSA 算法的性能越优越.

下面将讨论在分组的到达服从重尾分布时,在相同突发强度情况下,m 的取值对系统性能的影响.由于分组的到达服从重尾分布,所以,α 值越小突发性越强.在这种情况下,由于存在时间间隔近的分组,所以采用随机的分裂算法,可以使系统碰撞的分组容易分解开来.

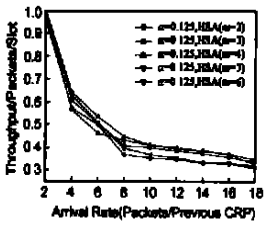

图 7 不同 m 值时吞吐率曲线

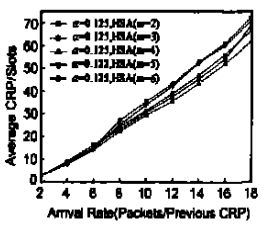

图 8 不同 m 值时 CRP 性能曲线

从图 7 和图 8 中可以看出,对于高突发强度,m 等于 4 时系统的吞吐率和分解所用时隙数的性能都是最好的,超过最佳切换分解次数后(如 m= 5、6)的系统性能都很差.而 m= 2、3 时系统的性能和 m= 4 时比较接近,这是因为此时分组突发性非常强,FCFS 分解在时间上的优势将不明显,采用树形分解比 FCFS 浪费的时隙数相对较少.但由表 1 可以看出分组数比较多时,过早采用随机树形分解,系统效率也不高.所以 m = 4 时性能仍是最好的,这和理论分析的结果吻合.

从图 5 到图 8 的结果可以看出,当分组的到达时间比较接近的时候,仅靠时间来划分指配区间,会造成许多时隙的浪费.在这种情况下,越早采用随机的方式,越能够有效的避免空闲时隙的产生.而且,在 HSA 算法的后续分解过程中,分组的产生时间间隔非常接近,所以采用随机的分解方式,基本上不影响整个网络的先到先服务原则.

4 结论

本文提出了一种混合型的冲突分解算法 HSA,该算法在传统的 FCFS 分裂算法和树形冲突分解算法之上,结合了二者的优点,当按先到先服务的原则分解了 4 次之后,HSA 算法将采用随机分解的树形分裂算法.在保证了系统的先到先服务原则,降低了系统分解行为对分组产生时间间隔的依赖性,从而有效地改进了系统的性能,使系统的最大吞吐率达到 0.51.HSA 算法不仅对 Poisson 到达过程适用,同时对于具有重尾特性的到达过程突发性越强采用 HSA 算法的优越性就越强.

参考文献:

[1] 郭梯云,杨家玮,李建东,数字移动通信[M].北京:人民邮电出版社,1996.365-366.

[2] V Bharghavan, A Demers, et al. MACAW: A media access protocol for wireless LANs[A]. Proc ACM SIGCOMM'94[C]. London, UK, Oct. 1994. 212-225.

[3] N Abramson. The ALOHA system-another alternative for computer communications[A]. Proc of the Fall Joint Computer Conference[C]. Montvale, NJ, 1970. 281-285.

[4] F A Tobagi, L Kleinrock. Packet switching in radio channels: part I-carrier sense multiple-access modes and their throughput-delay characteristics[J]. IEEE Trans Commun, 1975, 23(12): 1400-1416.

[5] J I Capetanakis. Tree algorithm for packet broadcast channels[J]. IEEE Tran Info, Theory, Sept. 1979, IT-25: 505-515.

[6] D Bertsekas, R Gallager. Data Network[M]. USA: Prentice-Hall, 1992.

[7] Walter Willinger, Murad S Taqqu, Robert Sherman, Daniel V Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level[J]. IEEE/ACM Transactions on Networking, 1997, 5(1): 71-86.

[8] 李建东,盛敏.通信网络基础[M].北京:高等教育出版社.

作者简介:

盛 敏 女,1975 年出生于湖南省长沙市,现为西安电子科技大学副教授、博士,主要研究方向包括移动 Ad Hoc 网络多址接入协议、QoS 路由、个人通信网络等.E-mail: msheng@mail.xidian.edu.cn

李建东 男,1962 年出生于江苏省阜宁县,西安电子科技大学博士导师,中国通信学会会士、IEEE 高级会员、中国电子学会高级会员,第一届和第四届 863 个人通信技术专业专家组成员,享受国家政府特殊津贴,从事移动通信、个人通信、软件无线电、分组无线网、自组织网络、宽带无线 IP 技术等方面的研究.

江 帆 女,1982 年出生于陕西省西安市,西安电子科技大学硕士生,主要研究方向是无线网络接入技术.