EHiQ:一种基于增强型HiQ的RFID读写器MAC协议

杨 健[1] 王永华[1] 蔡庆玲[2] 詹宜巨[2] 万 频[1]
(广东工业大学自动化学院 广州 510006)[1] (中山大学工学院 广州 510006)[2]

摘 要 针对HiQ算法存在的缺点,提出一种增强型HiQ读写器MAC协议EHiQ。EHiQ对HiQ算法的结构、即时费用函数等方面进行改进,并引入模拟退火思想,用于平衡学习各个阶段的扩张和探索,缩短学习时间。EHiQ占用的读写器频率-时隙对资源以及频率干扰率略高于HiQ,但收敛时间仅为HiQ的1/3左右。此外,由于系统结构的改进,EHiQ的阻塞率始终保持为零。

关键词 射频识别,防冲突,多路接入控制协议,增强型HiQ
中图法分类号 TN911 **文献标识码** A

EHiQ:A RFID Reader MAC Protocol Based on Enhanced HiQ

YANG Jian[1] WANG Yong-hua[1] CAI Qing-ling[2] ZHAN Yi-ju[2] WAN Pin[1]
(Faculty of Automation,Guangdong University of Technology,Guangzhou 510006,China)[1]
(School of Engineering,Sun Yat-sen University,Guangzhou 510006,China)[2]

Abstract Based on HiQ reader anti-collision algorithm,an enhanced algorithm called EHiQ was proposed. EHiQ not only imporves HiQ's system structure and cost function,but also imports the concept of simulated anneaning which is applied to balance exploitation and exploration of every phase of learning process and shorten learning time. Simulation shows that resource usage and frequency interference rate of EHiQ are slightly higher than those of HiQ. However, EHiQ's convergence time is just one third of HiQ's. Moreover,due to the improvement of system structure,EHiQ's blocking rate remains zero all the time.

Keywords RFID,Anti-collision,MAC protocol,Enhanced HiQ

1 引言

Q学习是强化学习的一种。它是一种非监督型学习,对环境的先验知识要求较低,通过对环境到动作映射的学习感知环境状态,使动作从环境中得到的积累奖赏值最大,从而确定最优行为策略。Q学习模型主要由智能体、行为评价规则和环境3部分组成[1,2],如图1所示。Q学习在智能体中进行,通过对各个状态下采取不同动作后获得的环境反馈(即奖赏值)的多少,来推导出一个行为函数。该行为函数给出了不同状态时为获得最大奖赏值应采取的最优动作。

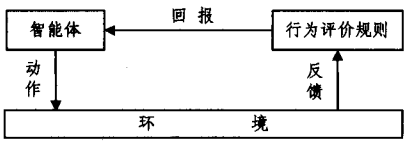

图1 Q学习过程

可以看出,Q学习的过程就是一个智能体逐步认识环境的过程。环境是一个含有状态变量的不确定系统,而行为评价规则规定了各个状态下采取不同动作后由环境反馈得到回报的类型,即是奖赏还是惩罚。

HiQ继承了Q学习的特点,使得可以在事先未知读写器网络拓扑结构的条件下,通过自主学习的方式优化分配的有限资源,从而达到降低读写器冲突的目的。HiQ将整个系统分为Q-Server,R-Server和读写器的上中下3层结构,每个中层结构体R-Server只与一个下层结构体读写器相连,上层结构体Q-Server可以与多个R-Server或Q-Server相连,但整个系统中只能有唯一的根Q-Server负责管理所有资源。HiQ学习在每个Q-Server中进行,通过试探不同动作后读写器的冲突数据计算出积累最优费用,指导选择合适的动作,优化、分配资源到相应的Q-Server和R-Server,并由R-Server将资源分配给读写器[3]。

可见,HiQ学习的值函数的迭代是通过策略迭代实现的,且迭代次数自始至终相对固定,只与状态动作对数有关。在学习初期,多次的策略迭代是有益的,因为这时各个读写器均不了解周围读写器占用资源的情况,无法得到最优动作,多

到稿日期:2010-08-18 返修日期:2010-11-12 本文受国家自然科学基金项目(60673132),广东省科技计划项目(2008B010200037),广州市科技计划项目(2008Z1-D141),广东工业大学博士基金项目资助。

杨 健(1982—),男,博士,讲师,主要研究方向为RFID组网及防冲突、认知无线电频谱检测,E-mail:jimmyyoungyj@163.com;王永华(1979—),男,博士,讲师,主要研究方向为认知无线电频谱共享技术;蔡庆玲(1966—),女,博士,讲师,主要研究方向为通信与信息安全;詹宜巨(1955—),男,博士,教授,主要研究方向为计算机测控与网络技术、RFID组网及安全;万 频(1963—),男,博士生,副教授,主要研究方向为计算机测控与数学智能化技术。

次的策略迭代可以帮助读写器增加对未知环境的探索；但是随着学习过程的进行，读写器当前策略逐步逼近最优策略时，过多的探索不仅没有意义，而且会影响奖赏值，降低算法的性能。

基于以上考虑，作者提出了一种增强型的 HiQ 读写器 MAC 协议(Enhanced HiQ，EHiQ)。EHiQ 不仅对 HiQ 系统结构进行了改进，给每个读写器均设立独立的 Q-Server，通过学习逐步逼近最优费用下的频率和时隙的分配；而且引入模拟退火思想，根据学习所处的阶段动态调整温度参数，选择合适的探索次数。仿真表明，EHiQ 的收敛速度要明显高于 HiQ 算法，对读写器拓扑变化的适应性较 HiQ 强；占用读写器资源和频率干扰率略高于 HiQ，但收敛时间仅为 HiQ 的 1/3 左右；EHiQ 的阻塞率始终保持为零。

2 EHiQ 算法的提出

2.1 系统结构

HiQ 的系统中存在多个 R-Server 共用一个 Q-Server 以及多个读写器共用一个 R-Server 的可能性，而 R-Server 本身不具有学习能力，只能从上层的 Q-Server 接收已分配好的频率和时隙资源，此时单个 R-Server 及单个读写器将无法实现最优的资源分配。鉴于此，我们对 HiQ 的系统结构进行如下改进：取消 R-Server，每个读写器与 Q-Server 直接相连，并接受 Q-Server 的资源分配；Q-Server 可以层层嵌套，但只能有唯一的根 Q-Server；各 Q-Server 拥有相同的状态空间和动作空间，按自身的最优策略分配频率和时隙资源，根 Q-Server 则对全局资源进行调配，如图 2 所示。

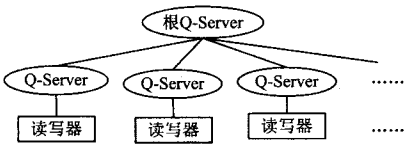

图 2 EHiQ 系统结构图

2.2 系统模型

定义系统中的资源为读写器可用的频率-时隙对，如 2 个频率和 2 个时隙对应的频率-时隙对资源为(f_1 t_1)，(f_1 t_2)，(f_2 t_1)，(f_2 t_2)。因此定义 Q-Server 的状态为所有可使用的频率-时隙对，可能的动作有 3 种，分别为增加一个频率-时隙对，减少一个频率-时隙对以及维持当前资源不变。

对于 Q-Server 来讲，其学习的目标是寻求最优策略(动作序列)π^*，使得到的积累费用最小，而环境则被看成是一个有限状态的离散马尔科夫过程(MDP)。若用一个五元组$\langle S, A, C, P, V\rangle$[4,5]来表示该 MDP，其中 S 为状态空间，A 为动作空间，C 为费用函数，P 为状态转移概率，V 为值函数，可得到状态 s、动作 a 及其后续策略为 π 下的折扣积累费用的数学期望[6,7]为：

$$Q^{\pi}(s,a)=C(s,a)+\gamma\sum_{s'\in S}P_{ss'}(a)V^{\pi}(s') \tag{1}$$

式中，$C(s,a)$为当前状态 s 下采取动作 a 的平均即时费用，即 $C(s,a)=E[c(s,a)]$；γ 为折扣系数，满足 $0\leqslant\gamma<1$，反映总费用受即时费用和未来费用的影响比例；$P_{ss'}(a)$表示当前状态 s 下采用动作 a 后转移到状态 s' 的概率(在当前定义下为 1)；$V^{\pi}(s')$为状态 s' 的值函数。根据 Bellman 最优性原则，可得到最优积累费用，表示为：

$$Q^*(s,a)=C(s,a)+\gamma\min_{b\in A}Q^*(s',b) \tag{2}$$

式中，$Q^*(s,a)$即为 $V^*(s,a)$，满足该式的动作 a^* 即为最优策略在状态 s 时的最优动作 $\pi^*(s)$。

在寻找 $Q^*(s,a)$的过程中，并不是直接计算式(2)。由于 Q 学习的收敛性已被证明，因此可采用迭代法去无限逼近。更新下一时刻的积累费用值通常需要当前时刻状态 s 和下一时刻状态 s'的积累费用值、当前时刻采取的动作 a_t、当前时刻的即时费用 $c_t(s,a)$。如果用查找表来存储积累费用函数值，其初始值为 0，则可用以下公式更新表中各个状态动作对对应的 Q 值：

$$Q_{t+1}(s,a)=\begin{cases}Q_t(s,a)+\alpha_t[c_t(s,a)+\gamma\min\limits_{b\in A}Q_t(s',b)-\\ \quad Q_t(s,a)], & s=s_t\ \text{且}\ a=a_t\\ Q_t(s,a), & s,a\ \text{取其他值}\end{cases} \tag{3}$$

式中，括号内表示下一时刻 Q 值与当前时刻 Q 值的差值，其中 α_t 为当前步的学习率，等于 t 个时刻中出现(s,a)次数的倒数，以此调整 Q 值更新的修改量，达到学习过程逐渐平稳的目的。随着步数的增加，Q 值将逐渐逼近最优解，此时所处状态即对应着分配给该读写器的所有频率-时隙对资源。

2.3 模拟退火思想的引入

对于式(3)中的 $Q_t(s,a)$，HiQ 算法中采用了贪婪策略，即每一时刻均选取最小值，这样很容易陷入局部最优。我们对文献[8-10]中的模拟退火思想加以改进：通过在一定范围内接受非最优解，跳出局部最优。此外，设置合适的温度参数还有利于合理调整学习各阶段的探索次数，缩短分配资源所需的时间。

引入模拟退火思想后，每个状态下 $Q_t(s,a)$的选择原则可表述为：若在时刻 t 状态 s 时从动作空间 A 中随机选择一动作 a'，则对应 Q 值为 $Q_t(s,a')$；而此刻最优动作 a^* 对应的 Q 值为 $Q_t*(s,a^*)$。如果满足：

$$\varepsilon<\exp\left[\frac{Q_t(s,a')-Q^*(s,a^*)}{T_k}\right] \tag{4}$$

则接受 a'为最优动作，并用其对应的 Q 值来计算下一时刻的 Q 值。其中 ε 为[0，1]间的随机数，T_k 为当前学习周期的温度参数。温度参数采用等比降温策略，即 $T_{k+1}=\lambda T_{k+1}$，降温系数 λ 为接近 1 的小数。模拟退火在一定范围内接受非最优解，扩大了探索空间，有利于跳出局部最优解；并由温度参数控制探索次数，有利于缩短后期的学习时间。

2.4 即时费用函数的改进

式(3)中的 $c_t(s,a)$为即时费用函数，是当前状态下动作空间向费用空间的映射，反映了动作对当前状态的直接影响。当动作产生积极效果时(如成功识别标签)，则通过减小即时费用来体现，反之亦然。我们的目的是通过最优策略得到最优状态，使读写器冲突的可能性最小。而读写器冲突又可分为多读写器之间的频率干扰以及多读写器对单标签的信号干扰，因此我们对文献[3]中的即时费用函数进行如下改进：

$$c_t(s,a)=x_1R_{suc}+x_2R_{freq}+x_3R_{tag}+x_4R_{sf} \tag{5}$$

式中，x_1-x_4 为加权系数，其值可根据相应参数的影响程度来决定。R_{suc}为成功通信率，其定义为成功通信次数与总通信次数之比；R_{freq}为频率干扰率，定义为频率干扰次数与总通信次数之比；R_{tag}为读写器-标签干扰率，定义为标签干扰次数与总通信次数之比。由于每个读写器均有 Q-Server，每次资源请求都可以被接受，因此取消文献[3]中的阻塞率(读写器请

求拒绝次数与总请求次数之比)。同时考虑到系统中同一时刻拥有相同状态动作对的其他读写器可能会因使用相同频率而对该读写器产生影响,因此增加同频干扰率 R_{sf},定义为系统中当前时刻其他拥有相同状态动作对的读写器的最优积累费用之和与所有读写器最优积累费用之和的比值,如下式所示:

$$R_{sf}=\frac{\sum_{i=1}^{m}Q_i(s,a)}{\sum_{i=1}^{n}Q_i} \tag{6}$$

式中,n 为所有读写器的数量,m 为当前时刻拥有相同状态动作对的读写器的数量。n,m 和 Q 值均可由访问根 Q-Server 得到。

3 EHiQ 算法的执行步骤

由以上分析,可以得到 EHiQ 算法的执行步骤如下:

步骤 1 根据可用的频率-时隙对初始化状态空间 S 和动作空间 A,初始化查找表中所有的 $Q(s,a)=0$;初始时刻 $t=1$、初始退火温度 $T_0=50$、学习时间 N 为 12 万次;

步骤 2 在当前时刻 t 的状态 s 下随机选择一个动作 a_r(增加一个频率-时隙对、减少一个频率-时隙对或维持当前频率-时隙对不变),按照式(3)计算出动作 a_r 对应的下一时刻的 $Q_{t+1}(s,a_r)$,并将其与状态 s 下最优动作 a^* 产生的 $Q_{t+1}(s,a^*)$ 比较。若满足 $\varepsilon<\exp\left[\frac{Q_{t+1}(s,a_r)-Q_{t+1}(s,a^*)}{T_k}\right]$,其中 $\varepsilon=\mathrm{rand}(0,1)$,则令 $a=a_r$,否则令 $a=a^*$;

步骤 3 执行动作 a,记录下即时奖赏值 $c(s,a)$ 和下一状态 s';

步骤 4 按照式(3)更新 $Q_{t+1}(s,a)$ 值;

步骤 5 判断 s' 是否为目标状态,若是则按等比降温策略更新退火温度 T_{t+1};

步骤 6 若 $t<N$,令 $t=t+1$,则转步骤 2,否则终止学习过程。

4 仿真结果

4.1 参数设置

仿真基于 matlab 实现。为了将所有指标均和 HiQ 进行比较,采用了 HiQ 中的方格形读写器拓扑结构,如图 3 所示。图中的圆形代表读写器,两读写器之间的连线表示可能会发生频率冲突[3]。

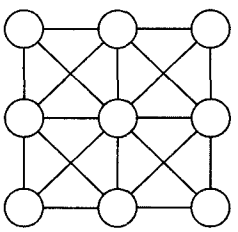

图 3 方格形读写器拓扑结构

仿真内容包括:1)系统实际所需的频率-时隙对随读写器数量变化的情况;2)频率冲突率随时间变化的情况;3)阻塞率随时间变化的情况。仿真参数的设置如下:折扣系数 $\gamma=0.9$;学习率 $\alpha=0.8$;模拟退火初始温度 $T_0=50$,并采用等比降温策略,$\lambda=0.95$。加权系数的设置为:$x_1=-4$,$x_2=4$,$x_3=1$,$x_4=1$。仿真持续时间为 12 万次。

4.2 仿真及分析

图 4 为 EHiQ 与 HiQ 占用的资源的比较,EHiQ 占用的频率-时隙对资源略高于 HiQ。这是由于 EHiQ 降低阻塞率的做法会导致读写器之间频率干扰几率上升,从而所需的频率-时隙对资源也有相应的增加。

图 5 为 49 个读写器、17 个可用频率-时隙对时 EHiQ 与 HiQ 的频率干扰率比较。可见,EHiQ 频率干扰率的收敛速度要明显快于 HiQ,达到稳定时所需运行次数仅为 HiQ 的 1/3 左右。但 EHiQ 稳定后的频率干扰率要高于 HiQ,这是由于 EHiQ 对于任何读写器的资源请求均接受,阻塞率的减低提高了读写器的反应速度,但是以增加读写器之间的干扰几率为代价的。

图 6 为 49 个读写器、17 个可用频率-时隙对时,EHiQ 与 HiQ 的阻塞率比较。由于 EHiQ 为每个读写器均配备 Q-Server,因此阻塞率始终保持为零。

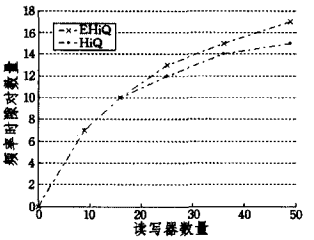

图 4 EHiQ 与 HiQ 占用资源对比

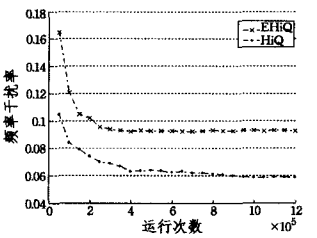

图 5 EHiQ 与 HiQ 频率干扰率比较

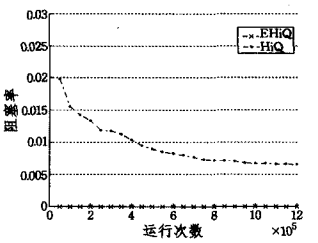

图 6 EHiQ 与 HiQ 阻塞率比较

结束语 仿真结果表明,EHiQ 频率干扰率的收敛速度要明显快于 HiQ,仅为 HiQ 的 1/3 左右,并且 EHiQ 阻塞率始终保持为零。但这是以占用读写器频率-时隙对资源以及频率干扰率要略高于 HiQ 为代价的,由此导致在学习初期的频率干扰率较高。下一步工作应考虑读写器相对较多而可用资源相对较少时阻塞率和频率干扰率的平衡问题。

参 考 文 献

[1] Mitchell T M. Machine Learning [M]. Columbus, USA: McGraw-Hill Companies, Inc., 1997: 154-199

[2] Weissensteiner A. A Q-lear-ning Approach to Derive Optimal Consumption and Investment Strategies [J]. IEEE Transactions on Neural Networks, 2009, 20(8): 1234-43

[3] Junius H, Engels D W, Sarma S E. HiQ: A Hierarchical Q-learning Algorithm to Solve the Reader Collision Problem[C]// Proceedings of the International Symposium on Applications and the Internet Workshop. USA: IEEE, 2006: 88-91

[4] 胡奇英,刘建庸. 马尔可夫决策过程引论[M]. 西安:西安电子科技大学出版社,2000: 132-138

(下转第 112 页)

图 8 中，先将 slice 制导变换为 sections 制导，之后在每个 section 内切片。切片之后的程序满足 slice 制导语义，也满足 OpenMP 标准指定，因此可以用标准的 OpenMP 编译器实现代码生成线程。

在实现中，为了与标准的 OpenMP 实现更好地结合在一起，首先将 slice 制导转换为 sections 制导。转换的方法是：如果有 n 个 slice 标准，则将 slice 制导控制的 structured-block 复制 n 份，作为 sections 控制的 structured-block，而每份作为 section 制导的控制代码区域。图 8 演示了这个变换过程，它是图 5 中含有 slice 制导的代码进行线程化的过程。

本文方法的主要优点在于和通用的 OpenMP 实现结合得很好，同时我们的方法有利于将含有 slice 制导的 OpenMP 程序源变换为对应的标准 OpenMP 程序。

4 实验结果与分析

我们选取了 SPEC CPU2000 中的 3 个程序进行实验，它们分别是 swim，applu，equake。实验是 HP ProLiant DL385 平台，CPU 是 AMD Opteron 285 双核(core)，运行 RHEL4AS 操作系统。原始串行测试用-O3 选项，并行测试用-O3 选项同时打开切片并行选项。表 1 列出了测试数据。

表 1 切片并行性测试数据

程序	串行(s)	并行(s)	性能提升(%)
swim	229.90	179.94	27.77%
applu	149.41	144.34	3.51%
equake	145.99	125.61	16.22%

首先利用切片分析工具分析 swim 和 applu，搜索能够获得切片并行的代码。分析结果中有些是不希望并行的，例如有些切片的并行粒度很小，因此从中选取合适粒度的切片用 slice 制导来并行化。循环与数组读写是判断计算粒度的主要依据，如果程序中没有循环和数组读写，或者循环次数是小常数，将不再分析这样的代码片段。在手工分析中，可以使用轮廓(profiling)信息寻找热代码，进行切片并行性分析。

在 swim 中选取 SHALOW(主函数)、CALC2 和 CALC3 3 个函数并行化。每个并行化代码片段都是指定 3 个切片变量，因此串行代码被分配到 3 个线程上执行。并行执行有 27.77%的性能提升。

在 applu 中选取 jacld 和 jacu 两个函数并行化。jacld 中并行化代码片段指定 4 个切片变量，jacu 中的代码片段指定 3 个切片变量，因此最多需要 4 个线程执行 applu。并行版本有 3.51%的性能提升。

在 equake 中，并行化核心热函数 smvp 指定 3 个切片变量，因此最多需要 3 个线程执行 equake。并行版本获得 16.22%的性能提升。

结束语 我们发现程序中含有一定数量的无关计算，这些计算可以并行执行并且能够获得一定的加速比。尤其是在程序的局部经常会有一些无关的计算被放在一起。本文提出的切片并行能够发掘程序中的无关计算带来的并行性，可以作为对数据并行的有益补充。基于 OpenMP 扩展的编程模型可以有效地表达这样的并行性，为程序员编程模型提供了方便。本文的切片分析可以帮助程序员更快捷、准确地寻找程序中的切片并行性，避免手工分析速度慢、易出错的问题。

目前的实验例子是手工写的，所以实验的充分性受到局限。将来的工作将考虑编译器全自动化地识别程序中的切片并行性。全自动的切片并行需要编译器自动识别并行区域、切片标准，并且进行切片，控制切片的大小和相互关系，评价切片并行的收益。这些问题对编译器都是极大的挑战，但问题的解决也将为切片并行带来更为广阔的应用前景。

参 考 文 献

[1] Weiser M. Program Slicing [J]. IEEE Transaction on Software Engineering, 1984, 10(4): 352-357

[2] Weiser M. Reconstructing Sequential Behavior from Parallel Behavior Projections[J]. Information Processing Letters, 1983, 17(3): 129-135

[3] Ferrante J, Ottenstein K J, Warren J D. The Program Dependence Graph and Its Use in Optimization[J]. TOPLAS, 1987, 9(3): 319-349

[4] Horwitz S, Reps T, Binkley D. Interprocedural slicing using dependence graphs[J]. ACM Transactions on Programming Languages and Systems, 1990, 12(1): 35-46

[5] Tip F. A survey of program slicing techniques[J]. J. Programming Languages, 1995, 3(3): 121-189

[6] Asanovic K, et al. The Landscape of Parallel Computing Research: A View from Berkeley[R]. UCB/EECS-2006-183. http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

[7] OpenMP Specifications [EB/OL]. http://www.openmp.org/drupal/

[8] Binkley D, Gold N, Harman M. An Empirical Study of Static Program Slice Size[J]. ACM Transactions on Software Engineering and Methodology(TOSEM), 2007, 16(2)

[9] Saleem M, Hussain R, Ismail V, et al. Cost effective software engineering using program slicing techniques[C]// Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human. Seoul, Korea, 2009: 768-772

(上接第 87 页)

[5] 朱江，徐斌阳，李少谦. 一种基于马尔可夫决策过程的认知无线电网络传输调度方案[J]. 电子与信息学报，2009，31(8)：2019-2023

[6] Watkins C J C H. Learning from Delayed Rewards [D]. Cambridge: Kings College, University of Cambridge, 1989

[7] Watkins C J C H. Q-Learning [J]. Machine Learning, 1992, 8(3): 279-292

[8] Lu Shou-feng, Liu Xi-min, Dai Shi-qiang. Incremental multistep Q-learning for adaptive traffic signal control based on delay minimization strategy[C]// Proceedings of the 7th World Congress on Intelligent Control and Automation. USA: IEEE, 2008: 2854-2858

[9] 陈圣磊，吴慧中，肖亮，等. 基于 Metropolis 准则的多步 Q 学习算法与性能仿真[J]. 系统仿真学报，2007，19(6)：1284-1287

[10] 周浦城，洪炳镕，韩学东，等. 基于多 Agent 的并行 Q-学习算法[J]. 小型微型计算机系统，2006，27(9)：1704-1707