

# 一种移动数据同步算法

王若莹<sup>1</sup>, 李 梁<sup>2</sup>, 张润洲<sup>1</sup>, 王 健<sup>3</sup>

(1. 西安通信学院, 陕西 西安 710106; 2. 西安邮电学院, 陕西 西安 710121;

3. 西安交通大学, 陕西 西安 710049)

**摘 要:** 由于无线网络的带宽窄、不可靠和频繁断接等限制, 传统的基于应答消息的数据同步算法不再适用。提出一种基于时间戳的多次传输算法来解决这个问题。它缓存移动基站发给移动客户机的消息, 对比版本信息时间戳识别移动客户机丢失的消息, 并重新发送这些消息。通过建立数学模型与已有的经典移动数据同步算法作比较, 证明其更能有效地降低数据访问出错率, 保持数据的一致性, 且移动客户机在接收到消息后无需发送应答消息, 更适合在通信不稳定的无线网络环境下使用。

**关键词:** 移动计算; 移动数据管理; 数据同步

**中图分类号:** TP391.72

**文献标识码:** A

**文章编号:** 1673-629X(2010)12-0137-04

## A Mobile Data Synchronization Algorithm

WANG Ruo-ying<sup>1</sup>, LI Liang<sup>2</sup>, ZHANG Run-zhou<sup>1</sup>, WANG Jian<sup>3</sup>

(1. Xi'an Communications Institute, Xi'an 710106, China;

2. Xi'an Institute of Post and Telecommunications, Xi'an 710121, China;

3. Xi'an Jiaotong University, Xi'an 710049, China)

**Abstract:** Because of wireless network's features such as narrow network width, bad network reliability, and frequent disconnection, the traditional data synchronization algorithm which depends on the responding message isn't suitable any more. Represent a new multi-transport algorithm based on the latest timestamp to resolve this problem. It caches the messages that mobile support station sends to mobile hosts, verify which messages are lost by comparing the messages' timestamp, and then resend the lost messages to the mobile host. During setup mathematical models and comparing with these classic mobile data synchronization algorithms, it is proved that it can reduce the data access mishit ratio, keep the data consistency better, and the mobile host doesn't need to send a response to the mobile support station which is more suitable for the instable wireless network environment.

**Key words:** mobile computing; mobile data management; data synchronization

## 0 引言

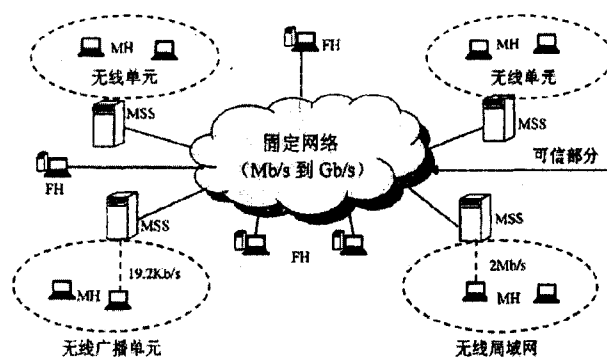
目前移动计算系统大都基于图1所示的移动计算环境的经典基础网络模型<sup>[1]</sup>。

模型由三类节点组成<sup>[2]</sup>:

1) 固定主机(Fixed Host): 每个FH维护一个本地数据库, 位于高速固定网络中。

2) 移动基站(Mobile Support Station, 简称MSS): MSS也位于高速固定网络中, 并具有无线联网能力, 它用于支持一个无线网络单元(Cell), 该单元内的移动客户机如果想和固定有线网络通信, 必须经过MSS的中转。MSS是无线网络实现与固定有线网络通信的

接口<sup>[2]</sup>。



FH: Fixed Host, 固定主机; MSS: Mobile Support Station, 移动基站; MH: Mobile Host, 移动主机, 移动客户机

图1 移动计算环境的经典基础网络模型

3) 移动客户机(Mobile Host, 简称MH): MH的处理能力与存储能力相对于服务器来说非常有限, 而且

收稿日期: 2010-04-12; 修回日期: 2010-07-24

作者简介: 王若莹(1978-), 女, 陕西西安人, 讲师, 硕士, 研究方向为智能设计理论。

受所处网络环境影响,经常与服务器断接<sup>[3]</sup>。

在无线网络环境中,网络通信质量差<sup>[4]</sup>,MSS 发给 MH 的消息很有可能会在传输过程中丢失<sup>[5]</sup>,此外,MH 因环境、节省电源、节约无线网络通信成本等原因会频繁处于断接状态<sup>[6]</sup>,这也会导致 MH 接收不到 MSS 发来的消息,如果采用 MH 每接收到一个消息就发送应答消息给 MSS 这种在固定网络中常用的方法是不可行的,因为:

1) 在通信质量较差的无线网络环境中,应答消息丢失的概率同样也比较大<sup>[7]</sup>。

2) 在通信质量不稳定的无线网络环境中,MH 发送应答消息给 MSS 消耗的时间不定,无法规定一个时间限制以判定消息是否丢失。

3) 在昂贵的无线网络环境中,MH 每接收到一个消息后都发给 MSS 一个应答消息将占用过多的无线网络资源,是一笔不小的费用,而移动环境下设计的目标之一是节约成本<sup>[8]</sup>,该方法违背了这个设计目标。

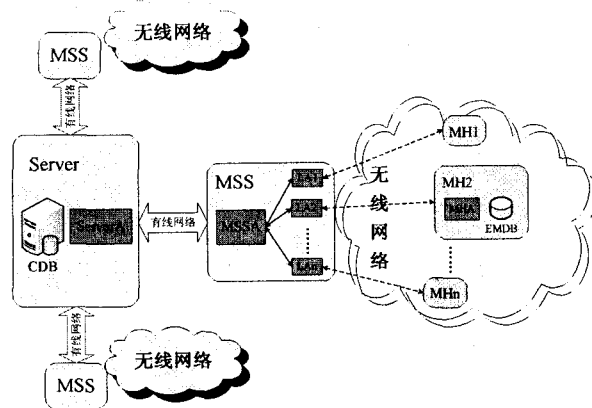
由于以上三个原因,传统的消息应答机制不适合在本模型无线网络环境下使用,需要一个新的算法来解决这个问题。

## 1 体系结构

文中提出了一种 MTLT(Multi-Transport Based Last Timestamp,基于时间戳的多次传输)算法来解决这个问题,它暂时缓存 MSS 发给 MH 的消息,通过对比 MH 版本信息时间戳识别 MH 丢失了哪些消息,重新发送这些消息给 MH,同时清除缓存中 MH 已经接收到了的消息,通过这种方式,MH 接收到消息后无需发送应答消息,并能够识别出哪些消息 MH 是已经接收到了的,哪些消息是 MH 丢失没有收到的,适合在通信不稳定的无线网络环境下使用。为更好地使用该算法,需要对经典基础网络模型做一个扩展,称新的体系结构为 MAMDM(Mobile Agent based Mobile Data Management and Synchronization Model,基于移动代理的移动数据管理与同步模型)。

如图 2 所示,MAMDM 模型是对经典基础网络模型的扩展。为了方便数据访问的管理和提高系统的可伸缩性,模型中各个部分的数据相关操作都由 Agent 来管理。MSS 上有两类 Agent:MSSA 和  $LA_i(i=1, 2, \dots, n)$ 。每一个  $LA$  都与无线网络中的一个 MH 相对应,它是 MH 在固定有线网络中的代理,它们之间是一一对应的关系,它作为无线网络环境中  $MH_i$  在固定有线网络中的代理,负责  $MH_i$  断接期间与固定网络交互的一切事务,比如缓存 MSSA 发给  $MH_i$  的数据信息,直到  $MH_i$  再次连接到网络中时,发送给  $MH_i$ ;缓

存  $MH_i$  发给固定网络的数据信息,在  $MH_i$  断接期间,固定网络与  $LA_i$  进行数据交互,通过这种方式,可以在无线网络不稳定, $MH_i$  频繁断接的情况下,实现  $MH_i$  与固定网络中 Server 的高质量数据通信。



CDB: Center Database, 中心数据库; ServerA: Server Agent, 服务器代理; MSSA: MSS Agent, 移动基站代理  
LA: Local Agent, 本地代理; MHA: MH Agent, 移动客户机代理;  
EMDB: Embedded Mobile Database, 嵌入式移动数据库

图 2 MAMDM 体系结构

## 2 算法描述

为了便于理解,把模型中的消息定义为两类:UR\_DOWN 和 UR\_UP。UR\_DOWN 表示模型自上而下发送的消息,每一个 UR\_DOWN 可以表示为: $URi\_DOWN = \{[M, tdc] | M \text{ 表示消息的主体, } tdc \text{ 表示消息的下载时间戳}\}$ ;UR\_UP 表示模型自上而下发送的消息,每一个 UR\_UP 可以表示为: $URi\_UP = \{[M, tuc] | M \text{ 表示消息的主体, } tuc \text{ 表示消息的上载时间戳}\}$ 。

MTLT 算法由 ServerA, MSSA, MHA 和 LA 四个 Agent 相互协调工作完成。ServerA 对于接收到的每个消息,首先给消息中的 tdc 赋值系统的当前时间,然后再发给 MSS。currentTime()表示 ServerA 的当前系统时间,在整个系统中,所有时间戳信息都是由 ServerA 生成的,在分布式系统中,不同机器的系统时间不是严格相同的,而如果时间戳生成的基准本来就不同,时间戳之间的比较就变得毫无意义了。其算法描述为:

```

对于每个消息 URi_DOWN(M, tdc){
    tdc = currentTime(); // currentTime()表示系统的当前时间
    sendToMSS(URi_DOWN);
}
MSSA 对于接收到的每个消息,除发给 MHA 以

```

外,还发给 MHA 在固定网络中的代理 LA,用于暂时缓存消息。其算法描述为:

```

对于每个接收到的消息 URi_DOWN(M, tdc) {
    sendToLA(URi_DOWN);
    sendToMHA(URi_DOWN);
}

```

MHA 对于接收到的每个消息,首先根据消息中的 tdc 修改当地数据库版本信息,再执行消息的主体;对于每个准备向上发送给 LA 的消息,首先赋值消息中 tuc 为当前数据库版本信息 T,再发给 LA。其算法描述为:

```

对于每个接收到的消息 URi_DOWN(M, tdc) {
    T = tdc;
    执行 M;
}

```

```

对于每个准备发给 LA 的消息 URj_UP(M, tuc)

```

```

{
    tuc = T;
    sendToLA(URj_UP);
}

```

LA 中有一个 la2mhURQ 队列,用于暂时缓存 MSS 发给 MH 的消息。LA 对于接收到的每个 URi\_DOWN 消息,都把它暂时缓存在队列 la2mhURQ 中;对于接收到的 URj\_UP 消息,根据 URj\_UP 中的 tuc,依次判断 la2mhURQ 中的每个 URk\_DOWN(M, tdc) 消息,如果  $tuc \geq tdc$ ,代表 URk\_DOWN 一定被 MHA 接收到了,这时可以清除 la2mhURQ 中的该消息,否则,如果  $tuc < tdc$ ,代表 MH 没有接收到 URk\_DOWN,需要重新发送 URk\_DOWN 给 MHA。然后,把接收到的消息 URj\_UP 向上发给 MSS 处理。其算法描述为:

```

对于每个接收到的 URi_DOWN(M, tdc) {
    la2mhURQ.put(URi_DOWN); //save(),缓存起来
}
对于每个接收到的 URj_UP(M, tuc) {
    对于 la2mhURQ 中的每个消息 URk_DOWN(M, tdc) {
        if (tuc >= tdc)
            la2mhURQ.remove(URk_DOWN); //清除 MH 一定收到的消息
        else if (tuc < tdc)
            sendToMHA(URk_DOWN);
    }
    sendToMSS(URj_UP);
}

```

### 3 性能仿真实验分析

对此方面的研究较多,如 Daniel Barbara, Tomasz Imielinski 提出的 TS (Broadcasting Timestamps)、AT (Amnesic Terminals) 算法<sup>[9]</sup>是 MSS 周期性以数据广播的形式向其管辖区内每个 MH 发送缓存失效报告,这种周期性发送数据广播的形式也称为同步 (Synchronous) 缓存一致性策略。与此对应的就是异步 (Asynchronous) 缓存一致性策略,例如由 Zhijun Wang 等人提出的 SACCS (Scalable Asynchronous Cache Consistency Scheme) 算法<sup>[10]</sup>,它也是以数据广播的形式发送缓存失效报告的,但是它不是周期性发送的,而是当 Server 端有数据更新或者有 MH 向 MSS 发出数据访问请求时才发送数据广播。

本章拿 MAMDM 与经典策略 AT 在访问出错率这个主要性能指标上做比较。参考文献[9,11]的数学建模方法,采用同样的数学建模方式,对 MAMDM 的访问出错率进行建模。

#### 3.1 模型环境建模

首先对模型环境做如下假设:

- 1) 整个系统中有  $M$  个 MSS;
- 2) 每个 MSS 管辖区内有  $N$  个 MH;
- 3) 每个 MH 对本地 EMDB 中任意一个数据项的更新服从系数为  $1/\mu$  的指数分布;
- 4) 中心服务器 Server 对数据库中任意一个数据项的更新也服从系数为  $1/\mu$  的指数分布;
- 5) MH 会频繁访问缓存在本地 EMDB 中的数据,这些数据也称为“热点数据”。每个 MH 对本地 EMDB 中任意一个数据项的访问服从系数为  $\lambda$  的泊松分布;
- 6) MH 频繁断接,有时处于断接状态,有时处于连接状态, MH 处于断接状态的概率为  $s$ ,处于连接状态的概率为  $1-s$ ,并且 MH 的行为之间互相独立,没有任何依赖性,这是一个为了使建模简单的假设,因为在实际情况下, MH 的行为之间经常具有依赖性,比如当 MH 发出查询或者更新数据请求时,它会尽量地在接下来的一段时间处于连接状态。

#### 3.2 MAMDM 访问出错率评估

$$P[\text{MH 处于断接状态下}] = st \quad (1)$$

$$P[\text{MH 断接状态下查询数据项 } x] = sl e^{-slx} \quad (2)$$

MH 断接状态下进行数据访问并且造成访问出错仅可能在如下事件发生时出现: MH 要访问的数据项在 MH 断接的这段时间内被其它 MH 或者 Server 修改过至少一次,如图 3 所示。

$$P[\text{某个 MH 没有更新数据项 } x] = e^{-\mu x} \quad (3)$$

所有其它 MH 的数量 =  $MN - 1$

$$P[\text{所有其它 MH 没有更新数据项 } x] = \prod_{i=1}^{MN-1} e^{-\mu t} = e^{-\mu(MN-1)t} \quad (4)$$

$$P[\text{Server 没有更新数据项 } x] = e^{-\mu t} \quad (5)$$

$$P[\text{所有外界没有更新数据项 } x] = e^{-\mu(MN-1)t} * e^{-\mu t} = e^{-\mu MN t} \quad (6)$$

$$P[\text{所有外界更新数据项 } x \text{ 至少一次}] = 1 - e^{-\mu MN t} \quad (7)$$

$$P[\text{外界在 MH 断接期间更新数据项 } x \text{ 至少一次}] = s(1 - e^{-\mu MN t}) \quad (8)$$

$$P[\text{MH 断接状态下查询数据项 } x, \text{ 并且外界在 MH 断接期间更新数据项 } x \text{ 至少一次}] = s\lambda e^{-s\lambda t} * s(1 - e^{-\mu MN t}) = s^2\lambda e^{-s\lambda t} * (1 - e^{-\mu MN t}) \quad (9)$$

MAMDM 的访问出错率  $P_{\text{stale}}$  如下所示:

$$P_{\text{stale}} = \int_0^{\infty} s^2\lambda e^{-s\lambda t} * (1 - e^{-\mu MN t}) dt = \frac{s\lambda MN}{s\lambda + \mu MN} \quad (10)$$

根据参考文献[5,12],经典模型 AT 的访问出错率  $P_{\text{staleAT}}$  为

$$P_{\text{staleAT}} = \frac{\mu MN}{(1-s)\lambda + \mu MN} \quad (11)$$

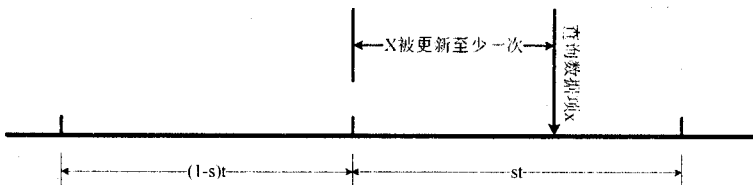


图3 MAMDM 中 MH 数据访问出错事件

### 3.3 与 AT 模型的比较

根据表1给出的模型参数数值,用MATLAB仿真绘出MAMDM的访问出错率  $P_{\text{stale}}$  和 AT 的访问出错率  $P_{\text{staleAT}}$  对比图如图4所示。

表1给出的四组参数值分别对应着图4中的(a)、(b)、(c)、(d)四幅图,从这四幅图中都可以得到如下结论:

- 1) MAMDM 的访问出错率明显低于 AT 的访问出错率;
- 2) 访问出错率随 MH 断接概率  $s$  的增大而变大;图4(b)与图4(a)的不同点在于 MSS 个数  $M$  的值由3变为了5;
- 3) 访问出错率随 MSS 个数  $M$  值的增大而变大;图4(c)与图4(a)的不同点在于每个 MSS 管辖区内的

平均 MH 个数  $N$  的值由10变为了15;

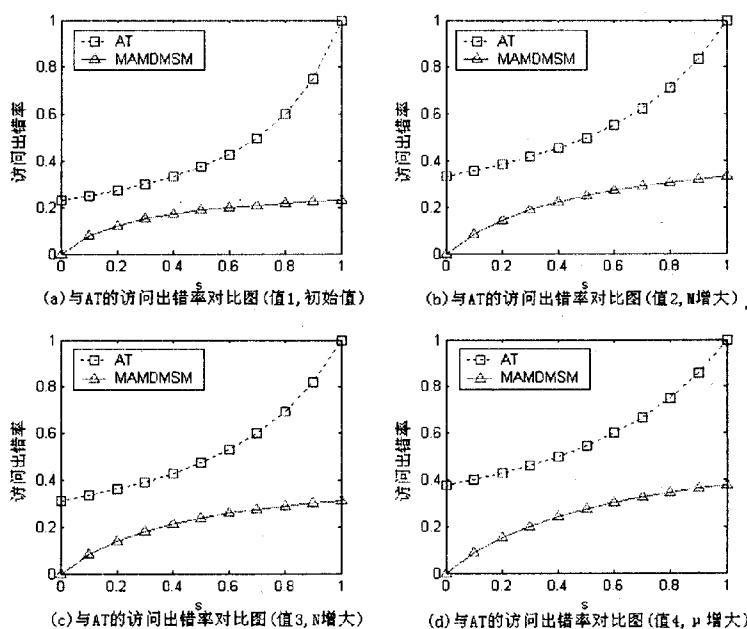


图4 MAMDM 与 AT 的访问出错率对比图

表1 模型参数数值表

参数	值1	值2	值3	值4
M	3	5	3	3
N	10	10	15	10
$\lambda$	1/100 次 查询/秒	1/100 次 查询/秒	1/100 次 查询/秒	1/100 次 查询/秒
$\mu$	10-4 次 更新/秒	10-4 次 更新/秒	10-4 次 更新/秒	2×10-4 次 更新/秒

4) 访问出错率随 MSS 管辖区内 MH 平均个数  $N$  值的增大而变大;图4(d)与图4(a)的不同点在于每个 MH 和 Server 对数据库中任意数据项的更新服从的指数分布系数  $1/\mu$  中  $\mu$  的值由  $10-4$  增大到了  $2 \times 10-4$ ;

5) 访问出错率随 MH 和 Server 对数据库的更新频率的增大而变大。

MAMDM 访问出错率优于 AT, 源于在 MSS 上消耗了额外的缓存空间, MAMDM 的 MSS 消耗磁盘空间为  $O(KN)$ , 而 AT 仅为  $O(N)$ 。但当前磁盘空间是比较便宜的, 所以付出消耗磁盘空间的代价换来系统性能的提高是十分值得的。

## 4 结束语

文中提出了一种 MTLT 算法缓存移动基站发给移动客户机的消息, 通过对比移动客户机版本信息时间戳识别丢失了哪些消息, 从而重新发送这些消息给移动客户机, 能有效地降低数据访问出错率, 而且移动客户机接收到消息后无需发送应答消息, 适用于通信不稳定的无线网络环境。

(下转第145页)

- (2) 使用 Matlab COM Builder 创建工程;
- (3) 创建、编译、调试 M 文件;
- (4) 生成组件;
- (5) 打包、分发组件。

在 Visual C++ 中调用 COM 组件的步骤:

- (1) 建立 Visual C++ 工程并设置必要的环境;
- (2) 初始化 COM 库;
- (3) 得到 COM 对象的 CLSID;
- (4) 创建一个 COM 对象的实例;
- (5) 使用 COM 对象;
- (6) 退出 COM 库。

以上描述及步骤从 Matlab 混合编程的需要,混合方法、选择与具体实现进行了分析与阐述,表明了该算法混合集成的可行性与其易于实现工程应用的特点。从而对进一步将 Matlab 中一系列专业用领域中的算法进行工程应用,创新与发展起到一定的意义。

#### 4 结束语

文中提出的张量积样条混合算法,将克里金估计与张量积样条插值方法以及 Matlab 与 VC++ 有效地结合起来,共同完成计算机在地质建模中三维地层曲面的构造与可视化。

从该方法所产生的图形 2,4,5 可见:该方法既保证了地层曲面的插值精度,同时又提高了地层曲面的光滑度,并且所用代码简洁有效。克服了克里金估计所产生曲面的粗糙性,又保证了最后生成的光滑曲面与原曲面基本重合。因此,文中所述方法不仅提高了三维地层曲面显示效果且保持精度,同时也满足了计

算机地质建模与油藏表征的要求。

#### 参考文献:

- [1] 李玉梅. P 元 Q 次超趋势面的算法研究及在油藏描述中的应用[J]. 西安石油学院学报, 2002, 17(4): 45-48.
- [2] Yamamoto J K. Correcting the Smoothing Effect of Ordinary Kriging Estimates[J]. Geoinformatics, 2001, 12(2): 148-151.
- [3] 闫存章, 李 阳, 王家华. 油藏地质建模与数值模拟技术文集[M]. 北京: 石油工业出版社, 2007: 83-85.
- [4] 邵才瑞, 关 丽, 张福明. 基于测井数据的地质曲面插值重构方法比较[J]. 测井技术, 2005, 29(4): 311-313.
- [5] 李 辉. 三次样条插值函数的研究[D]. 北京: 北京大学, 2008.
- [6] 李 涛, 贺勇军, 刘志俭. Matlab 工具箱应用指南—应用数学篇[M]. 北京: 电子工业出版社, 2000: 173-184.
- [7] 胡玉兰, 李沫沫. 样条曲面拟合及其 Matlab 实现[J]. 云南民族大学学报, 2005, 14(2): 170-171.
- [8] Min Qi. 3D Modeling and Visualization of Geology Volume Based on Geophysical Field[J]. Data Science Journal, 2007(6): 652-657.
- [9] 杨化斌, 王家华. 最小张力法在地层面光滑显示中的应用[J]. 西安石油学院学报, 2003, 18(2): 83-85.
- [10] 戴仕全, 罗钟铨. 自适应三次样条插值逼近算法研究[D]. 大连: 大连理工大学, 2008.
- [11] Nobuya, Maeshima. Tensor Product State Formulation for the Spin 1/2 Antiferromagnetic XXZ Model on the Checkerboard Lattice[J]. Journal of the Physical Society of Japan, 2004, 73(1): 60-67.
- [12] 杨高波, 元 波. 精通 Matlab 7.0 混合编程[M]. 北京: 电子工业出版社, 2006: 37-188.

(上接第 140 页)

#### 参考文献:

- [1] Imielinski T, Badrinath B R. Mobile wireless computing challenges in data management[J]. Communications of the ACM, 1994, 37(10): 18-28.
- [2] 邵雄凯, 郭卫华, 李 莉. 基于变周期数据广播的缓存一致性维护[J]. 计算机技术与发展, 2007, 17(3): 77-80.
- [3] 李智超, 何丕康, 雷 鸣. 移动计算中一种基于最小代价的语义缓存替换算法[J]. 小型微型计算机系统, 2006, 27(8): 1506-1508.
- [4] 段红亮, 雷向东. 移动环境下多信道广播的有效数据访问[J]. 计算机技术与发展, 2009, 19(4): 5-8.
- [5] 孟芝佳, 董逸生. 基于移动数据库 SwiftDB 的 XML 数据管理[J]. 计算机技术与发展, 2007, 17(2): 115-117.
- [6] 周丽芬, 杨春金. 嵌入式系统中移动数据库的研究[J]. 通讯和计算机, 2007, 4(2): 58-60.
- [7] 曾文龙, 刘跃龙, 宋 玮. 移动数据库关键技术研究[J]. 计

算机工程与应用, 2009, 45(14): 25-29.

- [8] 王 辉, 马维华. 基于 SQL Server CE 数据同步技术的研究与实现[J]. 计算机技术与发展, 2008, 18(4): 232-235.
- [9] Barbara D, Imielinski T. Sleepers and Workaholics: Caching Strategies in Mobile Environments[J]. Very Large Database Journal, 1995, 4(4): 567-602.
- [10] Wang Z, Sajal K, Che H, et al. A Scalable Asynchronous Cache Consistency Scheme (SACCS) for Mobile Environments[J]. IEEE Transactions on Parallel and Distributed Systems, 2004, 15(11): 983-995.
- [11] Khurana S, Kahol A, Gupta S, et al. An Efficient Cache Maintenance Scheme for Mobile Environment[C]//Proceedings of the 20th International Conference on Distributed Computing Systems (ICSCS 2000). [s. l.]: [s. n.], 2000: 530-537.
- [12] 王 红, 曾广周. 无线迁移 workflow 按需移动中停靠站缓存管理机制[J]. 计算机工程与应用, 2007, 43(29): 30-35.