第 34 卷 第 23 期 计 算 机 工 程 2008 年 12 月
Vol.34 No.23 Computer Engineering December 2008
·网络与通信· 文章编号：1000—3428(2008)23—0119—04 文献标识码：A 中图分类号：TP393

WSN 中基于层次结构的可靠传输算法

耿晓义，柴乔林，张 擎
(山东大学计算机科学与技术学院，济南 250061)

摘 要：提出针对无线传感器网络分层结构的一种数据可靠传输算法。采用综合节点剩余能量、缓冲区可用率、信道错误率 3 因素的可靠路径与备选可靠路径策略，依据缓冲区空闲情况进行预拥塞节点检测，进行路径前后疏导缓解拥塞。对簇头节点和非簇头节点的失效进行分类处理。仿真结果证明，该算法可提高数据传输的可靠性。

关键词：可靠性；信道错误率；簇头；梯度；拥塞

Reliable Transport Algorithm Based on Hiberarchy in WSN

GENG Xiao-yi, CHAI Qiao-lin, ZHANG Qing
(Department of Computer Science and Technology, Shandong University, Jinan 250061)

【Abstract】 This paper proposes a reliable data transport algorithm for cluster structure in Wireless Sensor Network(WSN). It includes: the choice of reliable routing and candidate routing considering residual node energy, buffer usability and channel error ratio, the detection of pre-congestion according to the spare space of a node buffer and the following catabatic methods, category handling the failure of cluster node or others. Simulation result shows that the algorithm improves the reliability of data transport.

【Key words】 reliability; channel error ratio; cluster head; grads; congestion

1 概述

无线传感器网络(Wireless Sensor Network, WSN)的设计目标之一是通过合理的算法，有效延长网络的寿命。在文献[1]中，对于 *MTTF*(Minimum Time To-Fail)有如下定义：

$$MTTF = \int_0^{\infty} t \frac{\mathrm{d}Q(t)}{\mathrm{d}t}\mathrm{d}t = \int_0^{\infty} R(t)\mathrm{d}t$$

其中，$R(t)$是组件在时刻 t 的可靠性。由此可见，提高网络传输的可靠性对于网络实际的生存起至关重要的作用。

影响数据传输可靠性的因素有很多。传感网络拥有大规模的节点且数据的业务流量大，节点通信带宽窄，通信覆盖范围小，断续频繁，经常导致通信失败，因此，信道质量与网络拥塞成为影响可靠性的 2 大主要因素。另外传感器节点损坏或能量耗尽引起的节点失效影响到数据收集，造成数据丢失，应在路由选择上考虑到能耗均衡以降低节点失效的可能性。

目前已提出多种基于多跳通信的提高WSN中数据传输可靠性的算法。ESRT[2]着力解决可靠性中的拥塞问题，采用基于节点的本地缓冲监测的拥塞检测机制，根据网络的当前状态调节节点速率。但算法需要调节所有的源节点，不适用于大规模的网络。文献[3]在定向扩散协议的基础上提出在主路径之外建立若干备选路径的方案。该方法较好地处理了拓扑变化，但没有考虑信道质量的问题，且只适用于源节点数量较小的情况。

从网络拓扑角度分析，无线传感器网络协议可分为 2 类：平面路由协议与分簇的层次路由协议[4]。本文的算法基于层次路由协议的簇结构，从可靠路径选择、拥塞避免机制、节点失效处理 3 个概念层次，展开对无线传感器网络可靠性的研究。

2 算法设计思想

现有的WSN分簇协议的一大类是簇内单跳通信[4]，基本特点都是监测区域分为多个簇，簇内成员与簇头之间直接通信，每个簇内节点的监测信息由簇头收集并完成转发，信息在融合或未融合后传输到基站*sink*处。在路由方面，本文将影响可靠性的因素——信道错误率、节点当前剩余能量和缓冲区可用率的情况这 3 个方面作为选择可靠路径的依据。为了在路径失败时能最快最优找到可靠路径，在当前路径建立的同时也建立了可靠的备选路径。在拥塞控制方面着重对拥塞避免的处理，分为本地的分散控制和*sink*节点的集中控制两方面。对判断已处于预拥塞状态的节点，采用“给予优先处理权”和“等待再发送”的方式在节点发送和接收 2 个方向上减小预拥塞节点的负担。对任意 1 簇来说，簇头节点就是当前区域的软肋节点，它的失效等于这片区域的概率为 1 的破坏，如不采取措施，数据丢失严重。根据分层架构的这一特点，分为簇头和非簇头节点失效，其中，非簇头节点的处理包括路径的更新及失效量统计与处理；除此之外，簇头节点的处理还包括簇内节点“另找门户”的操作、减少数据丢失、维护正常数据收集转发。

3 算法详细设计

3.1 可靠路由问题

在 WSN 中，节点能量、信道质量对传输可靠性有极其重要的作用。因此，在路径建立上，加入有关信道错误率 *Err*、节点当前剩余能量 *EnC* 和缓冲区可用率 *Buf* 的考虑。可定义

作者简介：耿晓义(1982－)，女，硕士，主研方向：无线传感器网络；柴乔林，教授；张 擎，硕士

收稿日期：2008-04-18 **E-mail：**gengxiaoyi@mail.sdu.edu.cn

路径每一跳的可靠性指标 *ReliaIndex* 为

$$ReliaIndex=-\alpha Err+\beta EnC+\gamma Buf \quad (1)$$

相对于节点能量和缓冲区可用率来说，信道错误率会降低路径的可靠性，因此，对其采用负系数。路径建立时，就按式(1)选择可靠性高也就相当于权值高的路径作路由。示例如图 1 所示。

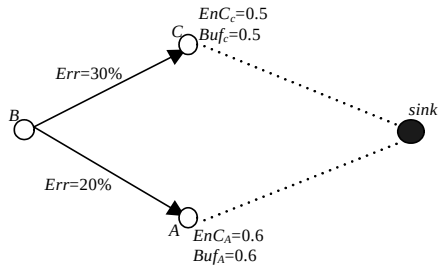

图 1　可靠路径示例

节点 *B* 在通往 *sink* 的过程中有 2 条路可选择，过节点 *C* 或 *A*，可通过计算当前的路径可靠性来决定，取α, β, γ参数均为 1，

ReliaIndex(*B*-->*A*)=-20%+0.6+0.6=1.0
ReliaIndex(*B*-->*C*)=-30%+0.5+0.5=0.7
ReliaIndex(*B*-->*A*)> *ReliaIndex*(*B*-->*C*)

因此，选择节点 *A* 做为下跳节点。

建立可靠路由的过程主要依赖对路径可靠性的计算。本算法中除当前使用路径外，还保存一条备用路径信息，它与当前使用的路径的建立过程同步进行，即相对于当前要使用的路径，它是一条次优选择。

一般情况下各簇的数据由簇头完成向 *sink* 节点的转发。本算法建立路由过程分为：(1)建立梯度场；(2)选择路径短且可靠性高的路径。先由 *sink* 节点发起 Dgree_Route_Form 包，其中，梯度值初始为 0。

选择路径和备选路径的过程如下：

(1)节点 *Node*(*i*)在收到 Dgree_Route_Form 包后，等待一随机时间；在收到的包中找到梯度最小(*Dgree*_Min)的一批包。若 *Node*(*i*)为簇头节点，转(2)；否则转(10)；

(2)若这些包中已有 2 个或 2 个以上节点表明自己已经建立了上一跳的路由，转(5)；

(3)如果簇头节点在收到的 Dgree_Route_Form 包中所有梯度最小的节点都没有建立自己的上一跳路由，则转(5)；

(4)如果簇头节点在收到的 Dgree_Route_Form 包中仅有 1 个已建立上一跳路由，则选其为上一跳节点；在剩余节点中计算一个可靠性最高的节点做为备选节点，转(6)；

(5)利用式(1)选择可靠性最高和次高的节点分别做为上一跳节点和备选节点；

(6)通知 2 个节点，发 Route_Use_U 包，建立起当前路径，保存备选路径；

(7)若 2 个节点存在没有自己的上一跳的节点，且自己度不为 1，则节点依据自身的梯度信息，向自己梯度小 1 的节点广播建立路由，转(5)；

(8)如果 2 个节点都有自己的上一跳且不为 *sink* 节点，则转(10)；

(9)如果 2 个节点发现自己上一跳为 *sink* 节点，则直接建立与 *sink* 的路由；

(10)*Node*(*i*)将 *Dgree*_Min 加 1 后作为自己的梯度值 *Dgree*_Myself，然后发送 Dgree_Route_Form 包，梯度值处填入 *Dgree*_Myself，标明自己是否为簇头，并附上当前的剩余能量，缓冲区利用率，信道错误率信息继续广播，逐步建立路由。

每个簇头都建立与 *sink* 节点可以传输数据的可靠路径，并沿路计算出整个路径的可靠性，如下：

$$Route(Head\_ID)\_ReliaIndex=\prod_{everyhop} ReliaIndex$$

沿路每跳的路径可靠性相乘得到某簇头节点*Head_ID*到*sink*处的路径总体可靠性，*sink*处作记录。路径建立后，随数据传输原来可靠的路径可能已变得不可靠。因此，所有簇头节点每隔两轮数据收集后，就沿路统计当前路径可靠性现状$ReliaIndex_{\text{current}}$与备选路径可靠性现状$ReliaIndex_{\text{candidate}}$，记录下路径可靠性出现负值的次数*Minus_NO*(*ReliaIndex*可能会为负值)，报告给*sink*节点。若满足：

$$\begin{cases} ReliaIndex_{\text{current}} < ReliaIndex_{\min} < ReliaIndex_{\text{candidate}} \\ ReliaIndex_{\text{current}} < ReliaIndex_{\text{candidate}} \\ Minus\_NO_{\text{current}} > Minus\_NO_{\text{candidate}} \end{cases}$$

3 个限定条件(其中，$ReliaIndex_{\min}$指在此之前*sink*节点记录的各路径可靠性最小的值)表明当前路径可靠性非常低，而备用路径状况较理想，因此，启用备选路径。

3.2　拥塞问题

3.2.1　分散控制

假设分簇后，每个簇的成员总数均值为 n。数据每轮收集时，若节点工作正常，则每个簇内节点至少发送 1 个包，假定每个包大小为 k，此处设定一个拥塞检测的阈值δ，

$$\delta=\frac{nk}{B} \quad (2)$$

其中，B 为簇头节点缓冲区大小；δ可作为簇头节点判断缓冲区拥塞情况的阈值。当缓冲区空闲率 $\frac{B_{\text{未占用}}}{B}\le\delta$ 时，则可判断认为在下一轮数据传输时就很可能发生拥塞。为尽量避免拥塞，此簇头节点即广播拥塞预通知 Pre_Congest 包，收到此包的不同节点处理方式不同：

(1)在此簇头节点通往 *sink* 节点的路径上的中间转发节点收到后，会提升该簇头所发出包的处理优先级，即优先处理和转发该簇头发出的包。若同时这些转发节点也收到别的簇头节点发送的 Pre_Congest 包，则发送此包的节点按请求先后顺序，依次处理。

(2)本簇内的非簇头节点收到自己簇头的拥塞预通知包后，各自等待一个随机时间后，再进行数据发送，给簇头一个处理已有数据的时间。这样在收集与转发的 2 个方向上对有可能发生拥塞的簇头节点给予协调，为簇头“减负”，给缓冲区尽可能的机会提升空闲率，避免拥塞的发生。

(3)当已发送过 Pre_Congest 包的簇头发现自己的当前的缓冲区空闲率 $\frac{B_{\text{未占用}}}{B}>\delta$ 时，则发送 Cancel_Pre_Congest 包，即取消拥塞预警包。在簇头节点通往 *sink* 路径上的节点取消相关优先级设置，恢复正常处理顺序。同时，当前簇的簇内节点也取消随机时间的等待机制，正常发送数据。

3.2.2　集中控制

sink 节点作为数据的最终接收者，应对观测区域有一个总体的了解并进行相应的调整。上文提到簇头节点在判断即将发生拥塞前会发送拥塞预通知包，这个包会沿路径转发，直到 *sink* 节点。*sink* 节点须维护 1 个预拥塞簇头数据统计量，*Head_PreCon_No* 收到 1 个节点的 Pre_Congest 包时，统计量加 1，并记录发送此包的簇头节点 *ID*；同样，当收到 1 个

Cancel_Pre_Congest 包时，就将统计量减去 1。此处依据节点 *ID* 信息，来保证加减的非重复性。可根据预拥塞节点增加与减少的情况对整个区域进行调整控制。对于 *Head_PreCon_No* 变量，每 5 s 统计对其进行了多少次加操作 *Add_PreCon_No*，多少次减操作 *Decre_PreCon_No*，当

$$\frac{Head\_PreCon\_No}{Head\_Total} \geq \lambda \tag{3}$$

且

$$\frac{Add\_PreCon\_No}{Decre\_PreCon\_No} > \eta \tag{4}$$

表明预拥塞簇头数过多，且拥塞缓解情况很不理想，这时 *sink* 节点有 2 种选择，对于本来就可以重新分簇的算法，可发送控制命令，重新分簇。另外，如果分簇协议没有重新分簇的情况或着当前算法再分簇也是一样的内部结构如 MAXId 算法，则 *sink* 节点可以暂停对兴趣数据的收集，缓解整体观测统计区域的工作量。

本文针对无线传感器网络的拥塞问题，从分散控制和集中控制相结合的角度，采取相应的措施，使拥塞这一可靠传输的关键影响因素得到有效缓解。

3.3 节点失效

3.3.1 簇头节点失效问题

在多跳传感器网络中，*sink*节点对其感兴趣的数据，进行了定期或不定期的询问查询，查询消息经过转发，到达各簇头节点，簇头节点负责本簇内查询消息的广播。若某簇内非簇头节点*A*收到一个查询消息*Mes*1，但此消息并非由本簇的簇头节点$Head_i$发送的，且在此之前*A*也未收到*Mes*1，则有可能簇头节点$Head_i$已失效。为了确定失效与否，节点*A*向$Head_i$发送询问包Ask_Head_Live，同时启动定时等待时间*T*。簇头节点收到自己成员发送的Ask_Head_Live包，若仍处于活的状态，就检查自己的当前的剩余能量$Energy_Current_i$，若满足$Energy_Current_i > \varepsilon$，则发送表明自己仍有效的Answer_Head_Live包给节点*A*。在*A*定时等待的时间*T*内，若收到来自簇头的有效包Answer_Head_Live包，则表明簇头工作状态正常，若在时间*T*内，未收到来自簇头的任何包，则表明此时簇头已经失效，或着处于因为拥塞等原因引起的“力不从心”状态。此时，为了进一步确认当前簇头的状态，节点*A*采用 2 次询问的方式，同样等待时间*T*，若仍未收到任何的来自簇头的包，则认定自己的簇头已失效。

在簇结构的算法中，簇头的失效会造成整个簇的数据丢失，若节点 *A* 确认簇头已经失效，应尽快通知原簇内所有成员“另找门户”，减少数据丢失。节点 *A* 广播发送自己当前簇头失效的报告包 Head_Fail 包，包含失效簇头节点的信息。当原簇内节点收到自己簇头已失效的报告后，第 1 时间广播 Join_In 加入请求包，这个请求包中除自身信息外还要加入原来簇头节点的信息，按偏好连接原则加入邻近的簇。

簇头节点的失效很容易引起路径的中断等问题。簇内节点也分 2 类：网关节点和普通节点(非网关非簇头节点)。对于网关节点，须检查其上下跳节点是否与失效簇头节点相关。若相关，则首先删除关于失效簇头节点的信息，并启用备选路径。若此时备选路径也已经失效就需要进行新路径的寻找建立过程。具体示例如图 2 所示。其中，H_{i+1}, H_i, H_{i-1}均为簇头节点；其他为非簇头节点。节点*A*到节点*E*是以H_i为簇头的成员，H_{i+1}→H_i→*B*→H_{i-1}是初始使用路径，如果H_i失效，则簇内节点就通过Join_In请求包，加入邻近簇。同时这条路径也中断了，依赖H_i的上下跳节点就需要更换路径。对于网关节点*B*，当其收到H_i失效的包时需更改路由表，广播Join_In，加入新簇。对于簇头节点H_{i+1}当其收到关于H_i失效的报告包时，就删除路由表中关于H_i的路径记录，并查看是否仍有备用路径，假设其备用路径为H_{i+1}→*C*→*E*→H_{i-1}，则H_{i+1}发送启用备选路径的消息给节点*C*，*C*收到包后如果自身正常，则回复节点H_{i+1}，并通知节点*E*，*E*也作回复并通知节点H_{i-1}，当H_{i-1}收到包后如果正常则发回复，这样启用备选路径，同时*C*与*E*也成为网关节点。如果此时备用路径不能正常工作如节点*C*并没有回复节点H_{i+1}，则认定备选路径失效，H_{i+1}删除掉备选路径的信息，再进行新的寻路操作。H_{i+1}发送路由建立包，找到自己可以通往 *sink* 节点的新路径，可以建立H_{i+1}→*D*→*C*→*B*→H_{i-1}的新路径，维持正常收集转发的任务。

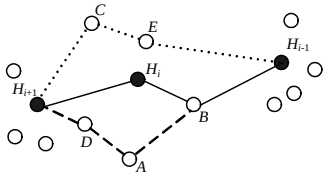

图 2　簇头节点失效问题

3.3.2 非簇头节点失效问题

在簇头节点每轮数据的收集时，簇内节点都应该向其发送数据，若某轮，除了节点 *A* 之外其他节点均已发送数据，那么簇头节点 *H* 就向 *A* 发送 Ask_aLive 包，询问节点 *A* 是否正常工作，簇头等待一个随机时间后，如果未收到 *A* 节点发送的任何消息，就认定节点 *A* 已经失效。簇头节点维护一个失效节点簇内统计 *inCluster_Fail*，记录簇内失效节点个数。

如果节点 *A* 是网关节点，则簇头节点同样要修改自己的路由表，启用备选路径，并广播 Useless_Tell 包，将网关节点 *A* 失效的消息进行通告并包含当前簇头 *H* 的信息，原来使用节点 *A* 与当前簇头相连的节点收到 Useless_Tell 包后，就发送广播包 Through_Ask 包，建立与 *H* 之间的新路径，尽可能避免网关节点失效带来的损失。如果节点 *A* 是普通节点，则只须维护簇头节点的失效节点统计量。

inCluster_Fail 的统计是为了簇头能够根据局部可靠性的现状与 *sink* 节点集中控制相配合。当有一片区域内监测节点大面积失效后，会导致信息的不准确不完全性。应当采取添补传感器或着其它的救助措施。层次结构的这种分簇方式，使得一个小区域的数据需要“托付”给簇头融合，那么簇头节点也需对当前小区域的节点工作情况有所统计，若当前区域状况使得数据区域性丢失，就应当报告基站 *sink* 处，以采取补救措施。因此，作一个条件限定如下：

$$\frac{S}{Nd\_\mathrm{Total}} \times inCluster\_Fail \geq \frac{S}{Cluster\_\mathrm{Total}}$$

其中，*S* 是指监测区域的面积；*Nd*_Toal 是节点总数；*Cluster*_Total 表示监测区域内簇的个数。本文假定所有的传感器节点为同构。如果当前簇内节点失效情况呈现上式的状况时，也就是簇内失效的节点平均所覆盖区域已经大于 1 个簇所占的平均 1 个监测区域时，就需要当前簇头节点向 *sink* 发 Mem_Fail 报告，基站可根据实际需要采取相应解决方法。

4　仿真实验

在 100×100 区域内，随机布置 200 个节点，随机分配信道的错误率。丢包率较直接地反映出网络的可靠性，丢包率定义如下：

$$丢包率 = 1 - \frac{sink收包量}{簇头发包总量}$$

APTEEN[5]和LEACH[4]是分簇协议中较为典型的算法。MESH[6]是较为成熟的可靠传输算法，采用多路策略达到较高的可靠性，将本文算法(RCNF)与MESH分别用在 2 个分簇算法架构之上，比较两者的丢包率。在APTEEN中 2 种算法丢包率的比较如图 3 所示，在LEACH中 2 种算法丢包率的比较如图 4 所示。

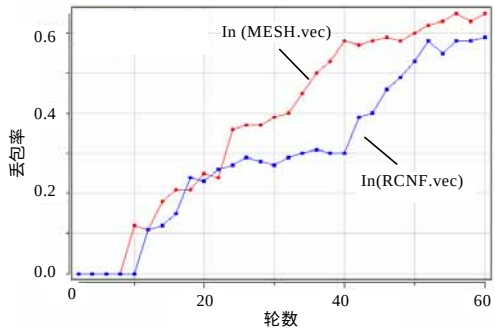

图 3　APTEEN 中 2 种算法丢包率的比较

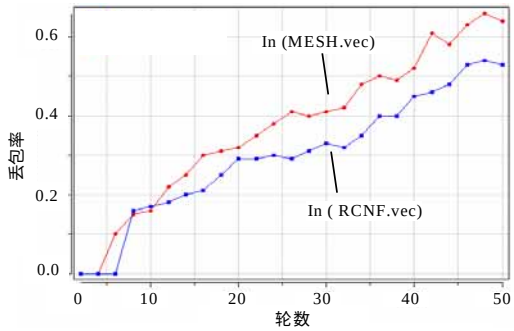

图 4　LEACH 中 2 种算法丢包率的比较

从图 3、图 4 中可以看出，本文算法(RCNF)在 2 种较为成熟的分簇协议架构下，比 MESH 算法表现出更高的可靠性，丢包率低于 MESH 算法。随数据收发轮数的增加，数据包可靠传输到基站的机会也更高。

本文的算法 RCNF 在数据传输的可靠性方面有较好的性能体现，但是算法在可靠路径选择上较为繁锁，还有待进一步改进。

参考文献

[1] Bein D, Jolly V. Reliability Modeling in Wireless Sensor Networks[J]. International Journal of Information Technology, 2005, 11(2): 2-5.

[2] Sankarasubramaniam Y, Akan O B, Akyildiz I F. ESRT: Event-to-sink Reliable Transport in WSN[C]//Proceedings of MobiHoc'03. Annapolis, Maryland, USA: [s. n.], 2003.

[3] Ganesan D, Govindan R, Shenker S, et al. Highly-resilient, Energy-efficient Multipath Routing in Wireless Sensor Networks[J]. ACM Mobile Computing and Communications Review, 2002, 1(2): 295-298.

[4] 沈　波, 张世永, 钟亦平. 无线传感器网络分簇路由协议[J]. 软件学报, 2006, 17(7): 1588-1600.

[5] Manjeshwar A, Agrawal D P. APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks[C]//Proc. of the 2nd Int'l Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing. [S. l.]: IEEE Computer Society, 2002.

[6] Ye Fan, Lu Songwu, Zhang Lixia. Gradient Broadcast: A Robust, Long-lived Large Sensor Network[Z]. (2001-01-01). http://irl.cs.ucla.edu/papers/grab-tech-report.ps.

（上接第 83 页）

(3)NGN 业务对内存数据库的访问目前只限于查询数据，对内存数据库的修改工作统一交给 SMS 来完成。可能会被业务修改的数据表暂时不考虑进行内存数据库处理。

目前已定义的操作内存数据库的基本接口如表 4 所示。

表 4　基本接口定义

序号	名称	输入参数	输出参数	说明
1	GetMem TablePtr	ServiceID 为业务类型，TableID 为该业务对应的表号	UINT8*：指向表节点的指针，pRet:函数执行返回状态	返回内存数据库中指定表节点指针
2	LoadMem Table	ServiceID 为业务类型，TableID 为该业务对应的表号	OUTPUT：布尔值，用来表示是否成功	加载内存数据库表节点
3	FreeMem DBData	无	OUTPUT：布尔值，用来表示是否成功	释放所有内存数据库资源
4	Delete MemDB	无	OUTPUT：布尔值，用来表示是否成功	删除内存数据库表
5	Refresh MemDB	UINT8 far * param :通知消息数据	无	接收到 SMS 修改数据表记录的通知消息后，向数据库发送同步数据消息
…	…	…		…

3　结束语

目前，在已开通的电信 NGN 业务中，如一号通业务、WebCall 业务等，该内存数据库系统已经正式上线使用，其稳定性、实时性得到了有力的验证。使用该内存数据库，在做性能测试的时候，访问数据库的时间基本可以忽略不计，这对于电信业务应用来说是个很大的优势。但是该系统还有一些需要改进的地方，如其对数据类型的支持有限，目前只支持数字和短字符串(小于 255 Byte)，其他的数据类型暂时不支持，不支持存储过程等，这在以后的工作中将持续改进。

参考文献

[1] 乔秀全, 李晓峰, 徐惠民. 基于 Parlay 思想的智能业务平台[J]. 北京邮电大学学报, 2004, 27(1): 61-62.

[2] ETSI. ES 201 915-1 V1.2.1-2002 Open Service Access; Application Programming Interface; Part 1: Overview(Parlay)[S]. 2002.

[3] ETSI. ES 201 915-10 V1.1.1-2001 Open Service Access; Application Programming Interface; Part 10: Connectivity Manager SCF[S]. 2002.

[4] 刘云生, 李国徽, 陆　舟. 实时内存数据库的装入[J]. 软件学报, 2000, 11(6): 829-835.

[5] 刘云生, 廖国琼, 付　蔚. 一个支持实时内存数据库的恢复系统[J]. 小型微型计算机系统, 2003, 24(3): 460-463.