无线移动环境中基于移动 Agent 的实时构件组装机制研究*)

胡海洋

(浙江工商大学计算机与信息工程学院 杭州 310018)

(南京大学计算机软件新技术国家重点实验室 南京 210093)

摘 要 由于无线网络较窄、昂贵的带宽,及移动设备自身有限的硬件能力,在无线移动环境中进行实时构件组装调用时面临着更多的技术挑战,需要一种高效、可靠的机制给予支持。本文提出一种基于移动 Agent 的无线网络环境构件组装调用机制。该机制利用移动 Agent 作为移动客户端与构件服务器之间交互的中介,提供了灵活、可配置的组装结构,从而有效避免移动客户端不必要的访问操作,节省了移动客户端的无线网络连接开销,与其他机制相比有着较高的执行效率;同时该机制可有效支持移动客户端发生故障后的状态恢复,从而确保运行过程的可靠性。最后本文通过实验分析了 MAWA 机制的可靠性与高效性。

关键词 移动 Agent,实时构件组装,无线移动网络

Research on Real-time Component Assembly Based on Mobile Agents in Mobile Wireless Environments

HU Hai-Yang

(College of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018)

(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

Abstract Because of the limited capability of mobile devices and lower bandwidth of wireless network, real-time component assembly in mobile wireless environments suffers more challenges and it needs an efficient and reliable mechanism to support. This paper presents a flexible and configurable mechanism, which uses mobile agents as brokers of mobile unit (MU) to communicate with component server, thus to reduce the unnecessary wireless communication costs of mobile client. It can also facilitate MU recovery back when suffering a failure to increase the reliability of the whole assembly process. Compared with other mechanisms through performance analysis, our mechanism shows its efficiency and reliability.

Keywords Mobile agents, Real-time component assembly, Mobile wireless environments

1 前言

通过无线接入设备进行无线通信已成为现代社会生活的一项规范,人们不仅通过移动设备相互通讯,也可用其访问网络中服务信息并进行相关商务应用,而不必关心自身的物理位置和相关的移动行为[1]。在无线移动环境中进行电子商务活动承受着更多的技术挑战,如由于自身硬件能力的限制,昂贵、较窄的无线网络连接及其不可靠性等,使得客户端应用不希望长久、持续地连接在 Internet 网络中,而需要有较为灵活的机制来协助其完成相关的商务活动。另一方面,从目前无线移动环境中所开展的 Web 电子商务应用来看,已从较早的简单数据浏览访问[2,3]转变为具有较为复杂读写操作的实时服务构件组合应用[4~7],此时移动客户端若能较好地进行实时电子商务活动,需要有一种高效且可靠的机制给予支持。

本文主要围绕服务构件状态信息实时改变的电子商务应用场景展开,给出一种基于移动 Agent[13,14] 的无线环境中实时服务构件组装访问机制 MAWA。该机制利用移动 Agent 作为移动客户端与服务构件之间交互的中介,有效避免移动客户端不必要的无线网络连接开销,与其他方式相比有着较高的执行效率;同时该机制能根据应用的要求,重配置对服务构件的组装协同结构,从而能满足移动应用动态变化的需求;该机制还可有效支持移动应用发生故障后的状态恢复,从而确保了整个构件组装应用过程的可靠性。

本文第 2 节给出基于移动 Agent 的实时服务构件组装应用系统的基本模型;第 3 节给出基于移动 Agent 的服务构件组装调用机制,包括构件组装配置结构、交互协议等;关注构件状态信息实时改变、移动终端可靠性的执行开销分析在第 4 节给出;第 5 节则是相关的性能分析;相关工作比较与本文小结在最后中给出。

2 基本模型

在无线移动环境中,移动应用进行实时电子商务活动时一般需经历储藏(hoarding)、断连(disconnection)与重集成(reintegration)三个阶段[8,9]。在第一阶段中,移动应用从提供服务的站点下载相关的数据信息到移动终端,然后断开网络连接,根据所下载的数据进行应用操作;在重集成阶段,移动客户根据应用操作的结果对服务构件进行再次访问,从而最终完成商务应用。由于在 MAWA 系统中,我们利用移动 Agent 作为移动客户端的代理来与构件服务器进行交互,因此在重集成阶段,移动客户端并不直接与服务构件连接交互,

*)本课题得到国家自然科学基金(60233010,60273034)、南京大学计算机软件新技术国家重点实验室开放基金资助。胡海洋 博士、讲师,主要研究领域为构件、中间件技术。

而是把相关构件组装调用信息递交给移动 Agent，并由 Agent 替代完成对服务构件的访问调用。

MAWA 的系统结构如图 1 所示，它利用移动 Agent 对移动客户端所进行的商务活动进行日志管理，并作为移动应用与服务构件间交互的中介，代替移动应用完成与构件服务器进行交互与调用。MAWA 系统中主要包含用户 Agent (UsAg)与组装移动 Agent(MoAg)两类 Agent：

• UsAg 由提供中介服务的节点生成，每一 UsAg 对应一特定的移动客户端，用于协助移动客户端完成相关的构件组装调用活动。UsAg 根据移动客户端传送的组装连接信息，负责对 MoAg 的创建、发送与接受，及对移动应用所进行的构件组装调用活动进行日志记录等。

• MoAg 是具体用于完成对服务构件访问调用的媒介，由 UsAg 根据移动客户端传送的组装调用信息创建而成，并代替移动客户端完成相关的构件组装调用活动。UsAg 可根据移动客户端新的组装应用需求对其进行重配置。

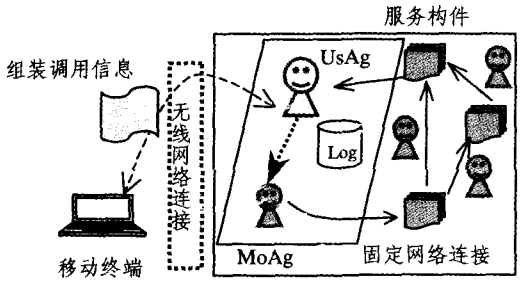

图 1　MAWA 框架的基本结构

3　可配置的组装调用过程

3.1　组装配置结构

在 MAWA 机制中，MoAg 由 UsAg 根据移动客户传送的构件组装调用表动态创建而成。这样的组装调用表 Ast (Assemble table)可形式化定义如下：

定义　$Ast=\langle AT, CR, CM, CL, AP\rangle$，其中：

• AT 为对外界构件的组装配置结构。MAWA 中的配置结构有基本(primary)结构和嵌套(nested)结构两种，嵌套结构由若干基本结构组合而成；

• $CR=\{ref_1, ref_2, \cdots, ref_n\}$ 为所需调用的构件指引集，用于内部标识构件；

• $CM=\{I_1.M_1, I_2.M_2, \cdots, I_n.M_n\}$ 为所需调用的构件方法集，这样的构件方法与特定构件实例相关联；

• $CL=\{\langle R_i, H_i, N_i, I_i\rangle; 1\leqslant i\leqslant n\}$ 为构件连接集，给出了所需连接调用的构件集。其中 R_i 表示构件 i 在表中对应的构件指引，H_i 为构件 i 所在的物理位置，N_i 为构件 i 名，I_i 为构件 i 的具体实例名；

• AP 为对构件组装调用后的后处理操作。

MAWA 中的配置结构 AT(Assembly Type)有基本结构与嵌套结构两种，其中基本结构有：SEQ、SLOOP、PAR 三种，嵌套结构由基本结构组合而成。本文现给出它们的 BNF 文法如下：

1. AT=SEQ 时
〈Assembly Type〉::=〈SEQ〉
〈SEQ〉::=〈Condition〉;〈Primary〉;〈Ref〉;〈MethodCall〉;〈AfterProcessing〉{;〈Ref〉;〈MethodCall〉;〈AfterProcessing〉;}
〈SEQ1〉::=〈Condition〉;〈Nested〉;〈AssemblyType〉
〈MethodCall〉::=〈InterfaceName〉〈MethodName〉{〈ParameterList〉}&〈MethodType〉
〈MethodType〉::=〈Mode〉〈Type〉{;〈Mode〉〈Type〉}
2. AT=PAR
〈AssemblyType〉::=〈PAR〉
〈PAR〉::=〈Condition〉;〈Primary〉;〈Ref〉;〈MethodCall〉{;〈Ref〉;〈MethodCall〉;}
〈PAR1〉::=〈Condition〉;〈Nested〉;〈AssemblyType〉
〈MethodCall〉::=〈InterfaceName〉〈MethodName〉{〈ParameterList〉}&〈MethodType〉
〈MethodType〉::=〈Mode〉〈Type〉{;〈Mode〉〈Type〉}。
3. AT=SLOOP 时
〈AssemblyType〉::=〈SLOOP〉
〈SLOOP〉::=〈Condition〉;〈Primary〉;〈Ref〉;〈MethodCall〉;〈AfterProcessing〉{;〈Ref〉;〈MethodCall〉;〈AfterProcessing〉;}
〈SLOOP1〉::=〈Condition〉;〈Nested〉;〈AssemblyType〉
〈MethodCall〉::=〈InterfaceName〉〈MethodName〉{〈ParameterList〉}&〈MethodType〉
〈MethodType〉::=〈Mode〉〈Type〉{;〈Mode〉〈Type〉}

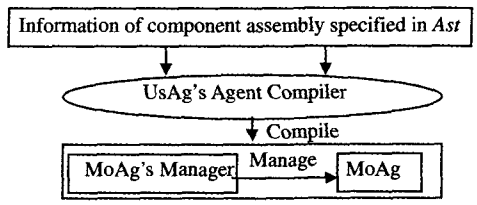

图 2　MoAg 及辅助管理者的生成

```
Class PAR-MoAgManager()
{   //Instance variables definitions for parameters;
    //Instance variables definitions for private data;
      public   PAR-MoAgManager(){//constructor}
      public   synchronized   void run(){
          // create the number of MoAgs that need sending out;
          //create the MoAgs with Itinerary type  ;
          //transfer values of instance variables from MoAg manager to MoAg;
          //start all of the MoAgs;
          //waiting for all the MoAgs returning back;
       }
       public synchronized void put_1()   {
         //  transfer the values of instance variables in MoAg 1 back to MoAg Manager;  }
       ..............
       public   synchronized put_n()   {
            // transfer the values of instance variables in MoAg N back to MoAg Manager; }
}
```

图 3　PAR 结构的 MoAg Manager 的代码模板

UsAg 接收这样的 Ast 表后将动态创建生成 MoAg 及辅助的 MoAg 管理者(如图 2 所示)。其中 MoAg Manager 是暂时类的 Agent，主要依据 Ast 表的信息完成构造 MoAg 迁移计划的工作；它固定在 UsAg 端，专司管理 MoAg 之责，如进

行参数传递、在 SLOOP 结构中判断循环是否继续及收集执行结果等。MoAg Manager 的实例变量包括在 Ast 表中定义的参数及在表中定义的私有数据，这些私有数据可包括新的类型定义、命名常量及用于中间处理的临时变量。图 3 给出了 PAR 结构的 MoAg Manager 的代码模板。

3.2 MAWA 的基本交互协议

UsAg 驻守于网络节点上，代替移动客户端完成与服务构件间的交互，移动客户端在重集成阶段所预取的服务构件数据信息在 UsAg 处存有备份。在移动客户端将进入重集成期间前，UsAg 将访问相关的服务构件，看其相关状态是否已实时改变。若某服务构件相关状态已被其他应用所更新，则 UsAg 把更新后的数据信息下载到本地并同时锁定(locking)[4] 这样的服务构件。移动客户端在重集成阶段重新连接上网络，发送讯问消息给 UsAg。在获取 UsAg 处的最新数据信息后，有如下的两种可能情形：

(1)对于某服务构件，若在移动客户端断连期间，该构件的状态信息已被其他应用所更新，则移动客户端将依据 UsAg 传送的数据信息重新制定相关的组装调用逻辑并将其传送给 UsAg。

(2)对于某服务构件，若在移动客户端断连期间，该构件的状态信息未被其他应用所更新，则移动客户端无需修改已制定相关的组装调用逻辑，而直接将组装调用逻辑发送给 UsAg 即可。

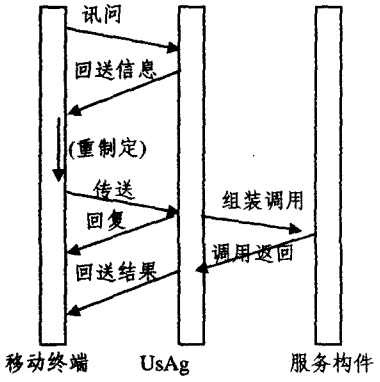

图 4 三方之间的交互协议

4 执行开销分析

4.1 实时改变的构件状态信息

对于任一被移动客户端预取相关信息的构件 i 而言，设在断连期间，移动客户根据其相关信息制定组装调用逻辑后对构件状态信息的更新率为 λ_i^0，外部其他应用对该构件进行调用的状态更新率为 λ_i^0。C_m 为在有线网络中相邻两节点间传输一个控制消息所需平均开销；α 为有线网络速率与无线网络速率的比值，则在无线网络中传输相同消息所需的开销为 αC_m。C_w 为在无线网络中传输组装调用信息和服务构件的状态信息所需平均开销。C_m 为移动客户端重新制定组装调用信息时所需的平均开销。C_{Ag} 为在有线网络中节点间传送 MoAg 所需的平均开销。C_s 为移动应用重集成阶段所需的平均时间开销。λ_f 为应用所在的移动设备的失败率，其服从期望值为 $1/\lambda_f$ 的指数分布。与文[10,11]一样，MAWA 机制中采用“一致时间间隔”(coherency interval)机制，则可设移动应用断连时间间隔为 L_x。

在移动应用断连期间，构件 i 的状态信息被其他应用更新的概率为

$$p_i = 1 - e^{-\lambda_i^0 L_x} \tag{1}$$

则对移动应用而言，该实时构件状态信息的可靠性为 $1-p_i$。

设移动应用所需组装调用的实时服务构件数为 n，则在重集成阶段其无需变动相关组装调用信息的概率为

$$P = \prod_{1 \leqslant i \leqslant n} (1 - p_i) \tag{2}$$

则依据图 4 中所示协议，移动应用在重集成阶段所需平均开销为

$$C_s = P(2\alpha C_m + 2\alpha C_w + C_{Ag}) + (1-P)(2\alpha C_m + 2\alpha C_w + C_{Ag} + C_m) \tag{3}$$

当 L_x 取得符合式(4)的值，C_s 可取得其极小值，即

$$\frac{\partial C_s}{\partial L_x} = 0 \text{ 且 } \frac{\partial^2 C_s}{\partial L_x^2} > 0 \tag{4}$$

4.2 移动设备的可靠性

由于移动设备自身硬件能力的脆弱性，移动客户端在运行时会有失败情形的发生[12]。具体说来可存在着如下的两种失败情形：1)MU 在断连阶段产生失败；2)MU 在重集成阶段发生失败。

本文在解决上述失败情形时，给出基于移动 Agent 的移动应用状态恢复算法如下：

(1)移动终端重新连接上网络并向 UsAg 发送询问消息，通过在消息中设置相关标识位表明自身目前处于“失败\待恢复”状态；

(2)UsAg 收到“失败\待恢复”消息后查询移动应用的操作日志：

a)若未发现移动应用已递交的组装调用信息，UsAg 将把被移动应用所预取的服务构件实时状态信息发送过去；

b)若发现移动应用已递交的组装调用信息，UsAg 将把移动应用所递交的组装调用信息发送给移动客户端即可；

(3)移动客户端接受到 MoAg 传送的数据后：

a)若移动客户端处于断连阶段，其恢复状态后可重新进入断连状态；

b)若移动客户端处于重集成阶段，其恢复状态后将继续进行组装信息的更新操作。

由于移动客户端在断连期间的失败并不影响其后续操作的执行开销，因此在这里我们仅考虑其在重集成阶段失败/恢复时所需的平均开销。

移动客户端在重集成阶段发生失败的概率 P_f 为：

$$P_f\{L_x < X < L_x + C_s \mid X > L_x\} = \int_{L_x}^{L_x + C_s} \lambda_f e^{-\lambda_f t}\, dt / (1 - F(L_x)) = 1 - e^{-\lambda_f C_s}$$

综合以上，移动应用在重集成期间遭受失败情形下，执行组装调用所需的平均开销为

$$C_{s_f} = (1 - P_f) C_s + P_f (C_s + \alpha C_m + \alpha C_w) \tag{5}$$

当 L_x 取值满足式(4)时，C_{s_f} 可取得极小值。

5 性能分析

本文模拟了(λ_i^0，λ_i^m，C_m)三种参数变化时对组装调用执行开销的不同影响，并和文[11]中提出的访问调用机制 SPU-

PA、MPUPA 进行了比较，所需各种参数如表 1。

表 1　相关参数设置

参数	参数值
失败率 λ_f	(0.005,0.2)
固定网络与无线网络速率比 α	10
服务构件的状态更新率 $i\lambda_i(=\lambda_i^o+\lambda_i^m)$	(0,15)updates/hour
固定网络中传输控制消息所需开销：C_m	0.01
固定网络中传输组装调用信息所需的平均开销：C_w	(0.5, 5)
重制定组装信息所需开销：C_{rm}	(30, 180)
在固定网络中两节点间传输移动 Agent 所需的开销：C_{Ag}	0.1

为分析参数$(\lambda_i^o,\lambda_i^m)$对单个实时构件组装执行开销的影响，本文设定 $C_{rm}=100$，$C_w=1.5$，$\lambda_i=10$ updates/h，$C_m=0.01$s，并分别取 $\lambda_i^m/\lambda_i=\{0.2,0.4,0.80\}$，相关实验结果如图 5～7 所示。从图中可以看出，当 L_x 位于(200,600)时，重集成阶段所需开销可取得其极低值，但这需移动客户端较为频繁地连接上网络，以缩短重集成阶段所需的开销，即移动客户端每隔约 400s 便连接上网络并进行约 110s 的集成处理；当曲线处于平缓期时，移动客户端可处于较长的断连期而仅需付出较小的额外开销，如在 $\lambda_i^m/\lambda_i=0.4$ 时，当 L_x 从 2400s 延续到 3200s 时，其额外开销仅增长了约 1.4s，而比起该曲线的极低值($L_x\approx$400s 时)，其开销也仅增长了约 25s。因此移动客户端可在曲线平缓期时选择合适的 L_x 值，以从总体上缩减网络连接时间。从图 5～7 中可见，MAWA 比起 SPUPA 平均执行开销可减低 14.7%。

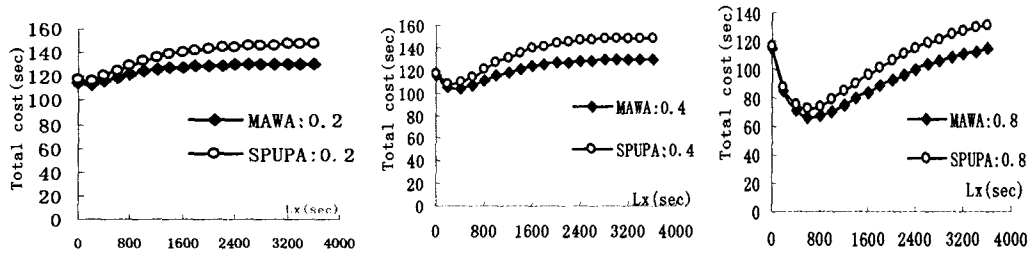

图 5～7　服务构件更新率对执行开销的影响

图 8～10 显示了参数 C_{rm} 对两种机制执行效率的影响。在此实验中，本文设定 $\lambda_i^m/\lambda_i=0.6$，$\lambda_i=10$ updates/h，$C_w=1.5$s，$C_m=0.01$ s，分别取 $C_{rm}=\{30,90,150\}$。从图中可见，当 C_{rm} 逐渐增加时，曲线峰谷值的差别也随着增大，这反映了移动客户端在重制定组装调用逻辑时需付出更多的开销。尽管随着 C_{rm} 值的增加，其 C_s 影响愈大，但从整个过程来看，其影响仍较有限，如图 9 中，当 $L_x=2400$ 时，其所需开销比起整个曲线在 $L_x\approx600$ 处所取的极低值仅高出约 52 s，与之相比，移动设备的断连期却可延长 1800 s。C_{rm} 对 MAWA 与 SPUPA 协议的执行影响随着其值的上升而不断增大。当 $C_{rm}=30$ 时，MAWA 的平均执行开销比 SPUPA 低 26.7%；而当 $C_{rm}=180$ 时，MAWA 的执行效率有所下降，但整体开销仍比 SPUPA 协议低 9.4%。

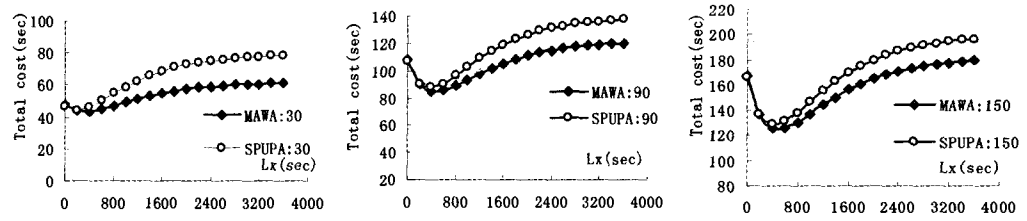

图 8～10　C_{rm} 对运行开销的影响

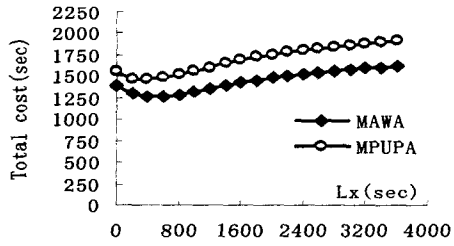

图 11　两种机制对多服务构件组装调用的执行开销比较

在测试对多服务构件组装访问时，本文取 $n=10$，对于每一构件的状态更新率 λ_i 为(0,15)之间的随机数，且对服务构件的访问调用采用“PAR”。则从实验中可以看出 MAWA 的性能开销略优于 MPUPA，这是因为在 MAWA 机制中，移动客户端向 UsAg 发送询问消息，讯问相关服务构件的状态信息并决定采取进一步的动作，避免了一些不必要的组装调用操作；同时 MAWA 中移动客户端所需的服务构件状态信息均由 UsAg 提供，而在 MPUPA 中此类信息需要移动客户端连接服务构件所在的服务器来获取。如图 11 所示，MAWA 比 MPUPA 平均可减低 15.7%的执行开销。

相关工作比较与小结　文[4]中给出构件信息的版本管理与加锁的概念，用以支持断连状态下的写操作访问。该机制的具体做法为：构件服务器中的状态信息均有相应的版本，当某状态信息被修改后，其版本信息将改变，则以后其他客户应用对该状态信息旧版本的更新请求将被构件服务器拒绝。文[5]中，作者设计了一个 HTTP 客户代理运行于移动客户端，在断连状态下用户对缓冲信息所做的修改将被该代理捕获。当移动设备重新连接上时，构件服务器将接受客户代理传送的更新请求。文[6]中，作者设计了缓存管理者 Venus 运行于移动客户端上，用于管理缓存对象。所有断连状态下

(下转第 282 页)

由 $halt(x=d)$ 的定义知，如果 $\eta_{[r_{4i},r_{4i+1}]} \models halt(y=10)$，则 $g_y(r_{4i+1})=10$。$\eta_{[r_{4i},r_{4i+1}]} \models keep(\dot{y}=1)$ 说明 y 是个区间 $[r_{4i}, r_{4i+1}]$ 上斜率始终为 1 的连续变量。由于 $g_y{}^+(r_{4i})=1$，因此，在区间 $[r_{4i}, r_{4i+1}]$ 上，恒有 $1 \leqslant y \leqslant 10$。即 $\eta_{[r_{4i},r_{4i+1}]} \models \Box(1 \leqslant y \leqslant 10)$。

类似地，如果 $\eta_{[r_{4i},r_{4i+2}]} \models halt(x=2)$，则 $g_x(r_{4i+2})=2$。$\eta_{[r_{4i},r_{4i+2}]} \models keep(\dot{x}=1) \wedge keep(\dot{y}=1)$ 说明 x 和 y 是区间 $[r_{4i+1}, r_{4i+2}]$ 上斜率都始终为 1 的连续变量。由于 $g_x{}^+(r_{4i+1})=0$，$g_y{}^+(r_{4i+1})=10$，因此，在区间 $[r_{4i+1}, r_{4i+2}]$ 上，恒有 $10 \leqslant y \leqslant 12$。即 $\eta_{[r_{4i+1},r_{4i+2}]} \models \Box(10 \leqslant y \leqslant 12)$。

同样的方法可以证明 $\eta_{[r_{4i+2},r_{4i+3}]} \models \Box(5 \leqslant y \leqslant 12)$，$\eta_{[r_{4i+3},r_{4i+4}]} \models \Box(1 \leqslant y \leqslant 5)$。因此，对任意的 $0 \leqslant j \leqslant 4k+1$，有 $\eta_{[r_j,r_{j+1}]} \models \Box(1 \leqslant y \leqslant 12)$。即(1)成立。

结论 本文定义了混合投影时序逻辑 HPTL，HPTL 不仅适用于刻画系统性质，也很适用于描述系统行为。这样，就可以在统一的模型框架下进行混合系统的验证。本文给出了 HPTL 的逻辑等价式系统，同时给出了一个用 HPTL 进行系统验证的实例。然而，本文仅仅对混合系统的验证进行了简单的尝试，验证实例采用的方法也只是逻辑推演。但是，本文定义了一种全新的逻辑，开辟了混合系统建模和验证的一个新的领域。作为今后的一项任务，我们将研究 HPTL 的模型检查问题，进行混合系统自动化验证的探索。

参 考 文 献

1 Duan Z. Modeling of hybrid systems: [Ph D thesis]. Shefield: Department of Computer Science. UK: University of Shefield, Department of Computer Science, 1997

2 Moskowski B. Executing Temporal Logic. UK: Cambridge University, Department of Computer Science, 1986

3 Kapur A, Henzinger T A, Manna Z, et al. Proving Safety Properties of Hybrid Systems. In: Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863, Springer-Verlag, 1994

4 Alur R, Courcoubetisz C, et al. The Algorithmic Analysis of Hybrid Systems. Theoretical Computer Science, 1995, 138(1): 3～34

5 Li G Y. LTLC: A Continuous-Time Temporal Logic for Real-Time and Hybrid Systems[D]: [Ph D thesis]. Beijing: Institute of Software, the Chinese Academy of Sciences, 2001

6 Henzinger T A, Kopke P W, Puri A, et al. What's Decidable About Hybrid Automata? J Comput Syst Sci, 1998, 57: 94～124

7 Alur R, Henzinger T A, Lafferriere G, et al. Discrete Abstractions of Hybrid Systems. Proceedings of the IEEE, 2000, 88(7): 971～984

8 Alur R, Courcoubetis C, Henzinger T A, et al. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In: Proceedings of Hybrid Systems' 93, LNCS 736, Springer-Verlag, 1993

(上接第 273 页)

的更新操作均记录在日志中。当移动客户端重新连接上服务器时，Venus 将缓存中的数据重新集成至服务器中。在此过程中，若 Venus 探测到有数据分歧时，将启动移动客户端上的应用协调器来解决这样的不一致性，甚至在必要时需要人工介入。文[11]中采用了文[4]中的版本管理与加锁机制、文[6]中的分歧协调机制，给出了三类具体的更新算法，用以解决何时移动客户端应连接上 Web 服务器，使得整个通信开销达到最优。上述现有工作较少考虑当移动客户端发生失败情形时如何继续维持构件组装访问过程的高效与可靠。此外，在这些工作中，移动客户端与服务器进行交互时需要长久、持续地连接在无线网络中，并增加了一些不必要的网络连接开销。

本文针对无线移动环境中服务构件组装应用场景，给出一种基于移动 Agent 的无线环境中实时服务构件组装访问机制 MAWA，可有效避免移动客户端不必要的无线网络连接开销；同时该机制能根据应用的要求，重配置对服务构件的组装协同逻辑结构，从而能满足移动应用动态变化的需求；本文还对应用执行过程中关注于服务构件状态信息实时变化的可靠性及相关的执行开销进行分析，并提供了支持移动客户端发生故障后的状态恢复机制，从而确保了整个应用执行过程的可靠性。

进一步的工作包括移动 Agent 自身安全性的考虑、移动 Agent 在基点(base station)之间自主性迁移以更好适应移动设备切换(handoff)的需求，及利用移动 Agent 对构件组装信息进行数据压缩/解压缩以提高传输效率等方面。

参 考 文 献

1 Dhawan C. Mobile Computing: A systems Integrator's Handbook. McGraw- Hill, USA, 1997

2 Joshi A, Weerawarana S, Houstis E, On Disconnected Browsing of Distributed Information. In: Proceeding of the 7th IEEE Work-shop on Research Issues in Data Engineering RIDE, 1997. 101～107

3 Floyd R, Housel R, Tait C. Mobile Web access using eNetwork Web Express. IEEE Personal Communications, 1998, 5(5): 47～52

4 Whitehead E J, Wiggins M. WEBDAV: IEIF Standard for collaborative Authoring on the Web. IEEE Internet Computing, Sept. 1998, 2(5): 34～40

5 Mazer M S, Brooks C L. Writing the Web while disconnected. IEEE Personal Communications, 1998, 5(5): 35～41

6 Kaashoek M F, Pinckney T, Tauber J A. Dynamic documents: mobile wireless access to the WWW. In: IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, Dec 1994. 179～184

7 Debra V M, Anindya D, Kaushik D. Mobile User Recovery in the Context of Internet Transactions. IEEE Transactions on Mobile Computing, 2003, 2(2)

8 Jing A S. Client-Server Computing in Mobile Environ-ments. ACM Computing Survey, June 1999, 31(2): 117～157

9 Pitoura E, Samaras G. Data Management for Mobile Computing. Kluwer Academic Publishers, 1998

10 Floyd R, Housel R, Tait C. Mobile Web access using eNetwork Web Express. IEEE Personal Communications, 1998, 5(5): 47～52

11 Chen I R, Phan N A, Yen I L. Algorithms for Supporting Disconnected Write Operations for wireless Web Access in Mobile Client-Server Environments. IEEE Transactions on Mobile Computing, 2002, 1(1)

12 Pedregal-Martin C, Ramamritham K. Support for recovery in mobile Systems. IEEE Transactions on Computers, October 2002, 51(10)

13 胡海洋，马晓星，陶先平，等. 反射中间件的研究与进展. 计算机学报，2005, 28(9): 1407～1420

14 胡海洋，杨玫，陶先平，等. Cogent 后组装机制的研究与实现. 电子学报，2002, 30(12): 1823～1827