普适环境下一种基于图的可靠服务组合机制

张 健 饶若楠
(上海交通大学软件学院 上海 200240)

摘 要 在普适环境下,由于设备、网络环境等的动态性,一些服务的提供者容易脱离当前的计算环境,从而造成服务组合方案失效。针对普适环境的特点,以及可靠服务组合的要求,提出了一种普适环境下基于图的可靠服务组合机制RScMPG,该机制从"预防"和"检测-补偿"的角度出发,在"生成可靠服务组合方案"和"监护服务组合正确执行"两个层面上,提高了服务组合的可靠性。

关键词 普适环境,图,可靠服务组合,RScMPG
中图法分类号 TP311.5 **文献标识码** A

Reliable Service Composition Mechanism in Pervasive Environments Based on Graph

ZHANG Jian RAO Ruo-nan
(School of Software, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract In pervasive environments, devices and network environments are extremely heterogeneous and highly dynamic. So, service providers are more possible to disconnect the current computing environments, leading to the failure of service composition. After considering pervasive computing features, and the requirements of reliable service composition, this paper gave a reliable service composition mechanism——RScMPG. RScMPG ensures service composition's reliability by "Generating More Reliable Service Composition Plan" and "Monitoring the Implementation of Service Composition".

Keywords Pervasive computing, Graph, Reliable service composition, RScMPG

1 引言

普适计算(Pervasive Computing)[8,9]是在分布计算、移动计算的基础上发展起来的新一代的计算技术,"无所不在"和"上下文感知"是普适计算的两大特征,而"以人为本"是普适计算的核心所在。一方面,随着通信技术以及硬件水平的发展,普适计算相关领域的研究越来越受到人们的重视。另一方面,随着服务计算、SOA 等技术的发展,软件即服务(SaaS)、基础设施即服务(IaaS)、平台即服务(PaaS)的理念正逐渐改变着人们对传统"服务"概念的理解,在此之上发展起来的"服务组合"技术也成为了计算机领域的研究热点。

在普适环境下,设备具有极端的异构性和很强的移动性,而且设备之间的通信协议多种多样。同时,与传统的 Internet 环境相比,普适环境下,设备的续航能力有限,而且出现故障的概率更高,因而,更容易造成某些设备脱离当前的计算环境,成为孤立的节点。这些因素共同决定了普适环境具有极大的动态性和不稳定性,服务边界也更难界定,与一般意义的服务组合相比,普适环境下的服务组合受环境的影响更大。面对普适计算这种相对脆弱的软硬件环境,有必要建立一种可靠的服务组合机制,以应对服务组合过程中出现的组合失效的问题,尽可能保证服务组合方案的顺利执行。这不仅满足了普适计算所倡导的"以人为本"的理念,也保证了人们对服务品质的追求,因而,建立这样一种可靠的服务组合机制意义重大。

目前,普适环境下的服务组合在方法和实现两方面都开展了广泛的研究[1,6,7]。这些研究从普适环境自身的特点出发,在"上下文感知"、"设备管理"、"偶发事件的处理"以及"用户体验"等方面增强了普适服务组合的功能。然而,在普适服务组合的可靠性方面,大部分研究都是针对服务组合可靠性的分析和度量进行的,相比之下,在方法层面上,提出一种构建或提高服务组合可靠性的策略却比较少。

本文针对普适环境的特点,以及可靠服务组合的要求,提出了一种普适环境下基于图的可靠服务组合机制 RScMPG (Reliable Service Composition Mechanism in Pervasive Computing Based On Graph)。RScMPG 从"预防"和"检测-补偿"的角度出发,在"生成可靠服务组合方案"和"监护服务组合正确执行"这两个层面上,提高了服务组合的可靠性。

本文第 2 节给出了普适环境下基于图的可靠服务组合机制模型,以及相关定义;第 3 节给出了实现该机制的具体算法,并用一个示例简单说明了该机制的工作过程;第 4 节对该机制所包含的 3 个算法在性能上做了一些分析;最后是结束语。

到稿日期:2010-06-22 返修日期:2010-09-24 本文受国防预研基金项目(513150302),上海市科委科技发展基金项目(08DZ2292902)资助。
张 健 硕士生,主要研究方向为分布式计算,E-mail:zhangjianvv@hotmail.com;饶若楠 副教授,主要研究方向为软件工程、分布式计算、网格计算。

2 普适可靠服务组合模型

服务组合，顾名思义，是将细粒度的服务组合成更大粒度的服务的一种方式。服务组合的方法有很多，由于普适环境的特殊性，并不是所有的组合方法都适合于普适环境。

基于图的服务组合方法，将服务组合问题看作是服务区域内的路径搜索问题。该方法能很好地描述服务之间"语义"的相容性，并且易于刻画全局服务区域的动态性，与普适计算的特点相一致，所以，很适合于普适环境。使用基于图的方法来构建一种普适环境下的可靠服务组合机制模型，下面给出该模型的相关定义。

定义 1(普适服务) 普适环境下能提供某种功能的执行单位。和一般的 Web Service 一样，普适服务也分为原子服务和组合服务。在本文中，一个原子服务抽象表示为：Service={name, location, input, output, function, reliability}。其中，name 是该服务的名字，是服务的唯一标识；location 是该服务所处的位置；input 和 output 分别代表服务的输入及输出；function 代表服务的功能，在本文中没有用到 function 属性；reliability 是一个概率值，表示服务的可靠性度量。

定义 2(任务) 普适环境下，一个服务组合请求称为一个任务(Task)[1]，任务表示为：Task={input,output}，一个任务可能由几个独立的子任务：SubTask(1)，SubTask(2)…SubTask(n)组成，其中，SubTask(i)={input(i),output(i)}。在当前的普适环境下，如果存在满足任务请求的服务组合方案，则称该任务是可解的。

基于图的服务组合中，利用图论中的基本图[1]或 Petri 网[5,10]来构建服务之间的依赖图[2]，是使用该服务组合方法的前提。服务组合引擎利用构建好的服务依赖图，查找匹配的原子服务，来完成任务的求解。本文使用基本图来构建普适服务依赖图，构建过程可以描述为：

1)利用图的一条有向边来表示一个普适服务，该边的起始结点代表服务的输入，终止结点代表服务的输出；

2)建立该领域内的领域本体，根据两个不同服务的输出(output)与输入(input)之间的匹配度，来决定这两个服务之间是否存在依赖关系；

3)对于存在依赖关系的两个普适服务 A 和 B，将 A 和 B 按照依赖次序连接起来，添加到服务依赖图中；

4)重复上述步骤，直到构建完该领域内的服务依赖图。

通过构建一张全局的普适服务网络，可以将当前普适环境下对任务的求解转化成图论中的路径选择问题。

定义 3(普适服务组合路径) 对于每一个 Task={input,output}，服务组合引擎将在构建好的普适服务依赖图上，搜索一条从 input 到 output 的最优路径。如果存在这样的路径，则称该路径为服务组合路径，该路径上的所有普适服务构成了该 Task 的服务组合方案。

定义 4(服务可靠性) 普适服务(原子服务或组合服务)能正确执行，并且不可能发生失效性。

一个原子服务的可靠性表示为：Reliability(Service)，可以用定义 1 中该 Service 的 reliability 属性描述。

对于组合服务 $CS=\{ES(1),ES(2)\cdots,ES(n)\}$，其可靠性为：

$$\text{Reliability}(CS)=\prod_{i=1}^{n}\text{Reliability}(ES(i)) \quad (1)$$

在普适计算的不同应用场景中，人们对可靠性的要求往往不同。根据当前的应用，设定可靠性阈值 RELIABILITY(0)，若 Service 的 Reliability 值大于该阈值，则认为该 Service 是可靠的，否则，认为该 Service 是易失效的。

定义 5(服务组合的可靠性) 对于一个 Task 请求，服务组合引擎会生成一个服务组合方案(定义 3)。服务组合的可靠性表示该组合方案能正确执行的可能性，以及抗失效的能力。文献[3,4]对服务组合的可靠性分析和预测等方面做了研究。

由于普适环境的动态性，容易造成服务组合方案失效，因而，可以用失效率(Failure Rate)来描述服务组合的可靠性。对于组合方案 ServComp={Service(1),Service(2),…,Service(n)}，

$$\text{FailureRate}(\text{ServComp})=1-\prod_{i=1}^{n}\text{Reliability}(\text{Service}(i)) \quad (2)$$

普适环境下，设备在计算能力和移动能力等方面存在较大的差异，因此，有必要对设备级别进行管理。

定义 6(设备层次结构) Swaroop Kalasapur 在文献[1]中，将普适设备分为 4 个级别，并提出了一种分层的体系架构。在普适服务组合中，设备存在 3 种角色：服务的提供者、服务执行代理和服务组合引擎。本文在文献[1]的基础上，提出了一种三层的体系结构，如表 1 所列。

表 1 普适环境下设备分级表

级别	属性描述	设备职能
0	无移动性、计算能力强	可作为服务组合引擎
1	无移动性、计算能力弱	可作为服务执行代理
2	较强移动性、计算能力弱	仅作为服务提供者

文献[7]认为，与传统的 Web 服务组合相比，普适环境下的服务组合面临以下几个方面的挑战：1)普适服务组合应该能够感知上下文，并根据上下文的改变对组合方案进行调整；2)普适服务组合应该能够应对各种偶发事件，并提供一定的容错机制；3)普适环境下设备存在很大的差异性，普适服务组合应该能够应对设备的异构性；4)普适环境强调"以人为本"，普适服务组合应该增强用户体验。

此外，在普适环境下，服务区域内的服务数量往往不是很大，而且由于设备和网络等因素的差异，设备之间的通信会占用大量的时间。因而，执行服务组合方案的代价往往会远大于生成组合方案的代价。

为了应对上述挑战，尤其是第 2)条所提出的要求，在充分考虑了普适环境自身特点的基础上，提出了一种可靠的服务组合机制模型 RScMPG。该模型从"预防"和"检测-补偿"两个角度出发，一方面，增强服务组合引擎生成组合方案的能力，另一方面，监控服务执行代理的工作过程。为了保证这些操作的完成，该模型还需要具备支持事务的能力，在服务组合执行的过程中，若发生服务失效，则要能正确执行回滚操作。RScMPG 可以分为以下两个阶段：

1)组合方案生成阶段：服务组合引擎首先生成一套服务组合方案，然后分析该方案中每个服务的 reliability 属性，对组合路径上容易失效的服务，生成一个针对该服务的备选组

合方案。利用这种冗余，在服务组合执行之前，预先做出防御措施，来“预防”可能发生的服务组合失效；

2)服务组合执行阶段：在服务执行代理执行组合方案的过程中，若发生阶段1中预期的服务失效(可能性大)，则用对应的备选组合方案替代发生失效的服务，继续服务组合的执行；若发生其它服务失效(可能性小)，则触发服务组合引擎，针对该服务失效生成一个最小代价的补偿措施，来保证服务组合透明、无缝地执行。

3 RScMPG实现

接下来，从服务组合方案的生成阶段，以及服务组合的执行阶段出发，详细描述RScMPG模型的实现。

3.1 组合方案生成阶段

服务组合引擎在生成服务组合方案之前，要获取当前普适环境中的所有服务提供者的信息，并由此构建服务区域内的服务依赖网络Graph_Service。针对一个Task请求，服务组合引擎将分两步来生成一个完整的组合方案。

第一步 在Graph_Service中，使用图论的BFS(广度优先搜索)算法，寻找服务组合路径，若该路径存在，则将该路径上包含的服务保存在组合方案栈SC_Stack中，否则，将返回NO_PATH_FOR_REQUEST，表示不存在满足该Task请求的服务组合方案。接下来，若组合方案存在，则分析SC_Satck中的每一个原子服务，如果该服务的Reliability值小于预设的阈值RELIABILITY(0)，则称该普适服务为“易失效”服务，并将该服务放入“易失效服务”栈Stack中。这个过程用伪代码表示如下：

算法1

```
1:Initialize(SC_Stack);//初始化组合方案栈
2:Initialize(Stack);//初始化组合方案中的易失效服务节点栈
3:for all SubTask(i) in Task:
4:  inRequest=Input(SubTask(i));//SubTask(i)的输入
5:  outRequest=Output(SubTask(i));//SubTask(i)的输出
6:      path=SearchShortestPath(Graph_Service,inRequest,out-
        Request);//搜索Graph_Service,寻找最优服务组合路径
7:  if path exists:
8:    push(SC_Stack,path);
9:      for all Service(j) in path://Service(j)表示SC_Stack中的第
        j个服务
10:       if Reliability(Service(j)) < RELIABILITY(0):
11:       push(Stack,Service(j));
12:       endif
13:     endfor
14:  endif
15:  else if path not exists:
16:    return NO_PATH_FOR_REQUEST;
17:  endelse
18:endfor
```

第二步 对于Stack中的每一个易失效服务，服务组合引擎将生成一个针对该服务的备选组合方案，并保存在栈数组Bak_Stack[1…Number]中该服务对应的位置，如果对该“易失效”服务，在当前的普适服务区域中，服务组合引擎没有找到相应的备选组合方案，则Bak_Stack[1…Number]中该服务对应的位置为空。这个过程用伪码表示为：

算法2

```
1:if Empty(Stack)://不存在易失效服务节点
2:  return;
3:endif
4:Number=Length(Stack);//计算易失效服务的个数
5:Initialize(Bak_Stack[1…Number]);
6:for i=1 to Number in Stack:
7:  position=Find(SC_Stack,Stack(i)) //查找Stack中的第i个元
    素在SC_Stack中的位置,Stack(i)代表Stack栈中的第i个元素,
    下同
8:  for j=position to 1 in SC_Stack:
9:    path=SearchShortestPath(Graph_Service,Input(SC_Stack
      (j)),outRequest);//outRequest为该“易失效”服务所对应
      SubTask的output
10:   if path exists:
11:     push(Bak_Stack[i],path);
12:     break;
13:   endif
14:   else if path not exists:
15:     continue;
16:   endelse
17:  endfor
18:endfor
```

在当前普适环境中，组合方案生成阶段在级别为0的设备上完成。对于一个可解的Task请求来说，该阶段将最终生成一个组合方案栈SC_Stack、一个易失效节点栈Stack以及备选组合方案栈的数组Bak_Stack[1…Number]。

3.2 组合方案执行阶段

在服务组合方案生成以后，将交给服务执行代理来监控组合方案的执行。由于普适环境的特殊性，执行组合方案的代价要高于生成组合方案的代价，在发生服务失效时，应尽可能利用已经执行过的组合方案的结果。

在执行的过程中，若某一个SubTask中的Service失效，首先检查该Service是否在Stack中。如果Service在Stack的位置i处，且该服务对应的Bak_Stack[i]不为空，则回滚到Bak_Stack[i]的入口处，并用Bak_Stack[i]来替换SubTask所对应的组合方案的剩余部分；否则，若Service不在Stack中，或者Bak_Stack[i]为空，则触发服务组合引擎，对该失效节点生成补偿组合方案。该过程用伪代码表示为：

算法3

```
1:position=Find(Stack,Service);
2:if position >= 1:
3:  if notEmpty(Bak_Stack[Position(Service)]):
4:    inPos=Entry(Bak_Stack[Position(Service)]);
5:    RollBack to inPos;
6:    Start Service Composition Bak_Stack[position];
7:  endif
8:endif
9:else://此处触发服务组合引擎
10:  Initialize(Comp_Stack);//初始化服务组合补偿栈
11:  pos=Find(SC_Stack,Service);
12:  for j=position to 0 in SC_Stack:
13:    path=SearchShortestPath(Graph_Service,SC_Stack(j),out-
       Request);
14:    if path exists:
```

```
15:    push(Comp_Stack,path);
16:    RollBack to position(j);//回滚到 SC_Stack(j)的服务入口处
17:    Start Service Composition Comp_Stack;
18:    break;
19:  endif
20:  else if path not exists;
21:    continue;
22:  endelse;
23:  endfor
24:  if Empty(Comp_Stack):
25:    return NO_PATH_FOR_COMPENSATE;
26:  endif
27:endelse
```

服务组合执行阶段在级别为 1 的设备上完成。如果一个 Task 由多个 SubTask 组成，且当前普适环境中存在多个服务执行代理，则这些 SubTask 可以在不同的服务执行代理上并行执行。

3.3 示例

本节以一个例子来说明 RScMPG 的工作过程。假设在当前的服务区域中存在如表 2 所列的普适服务。

表 2 普适服务描述

name	location	input	output	function	reliability
S(a)	M(1)	a	e	F(5)	0.6
S(b)	M(5)	a	b	F(7)	0.5
S(c)	M(10)	a	d	F(10)	0.8
S(d)	M(5)	e	g	F(2)	0.5
S(e)	M(8)	e	b	F(3)	0.7
S(f)	M(15)	b	c	F(6)	0.4
S(g)	M(2)	c	f	F(4)	0.7
S(h)	M(6)	b	f	F(8)	0.2
S(i)	M(9)	b	g	F(9)	0.4
S(j)	M(1)	g	h	F(15)	0.6
S(k)	M(5)	h	i	F(11)	0.7
S(l)	M(11)	j	g	F(13)	0.2
S(m)	M(7)	i	j	F(12)	0.5
S(n)	M(2)	j	k	F(20)	0.5
S(o)	M(6)	k	i	F(1)	0.8
S(p)	M(1)	f	i	F(21)	0.7

根据表 2，可以构建出该区域的服务依赖图，为了简便，在图中仅标注了服务的 name，input 和 output 属性，如图 1 所示。

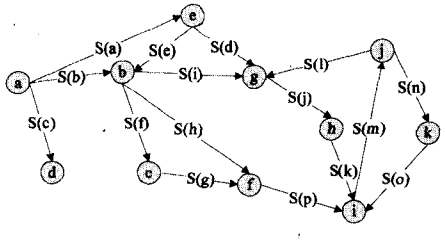

图 1 服务依赖图

如果有一个 Task＝{a,k}，且该 Task 包含两个 SubTask，其中，SubTask(1)＝{a,i}，SubTask(2)＝{i,k}，那么对于该 Task，预设的可靠性阈值为 0.3。

在服务组合的生成阶段，服务组合引擎，首先利用算法 1，可以求出 SC_Stack＝{S(b),S(h),S(p),S(m),S(n)}，其中，{S(b),S(h),S(p)}是 SubTask(1)的解；{S(m),S(n)}是 SubTask(2)的解。由于 Reliability(S(h))＝0.2 ＜ 0.3，因此 S(h)会被压入栈 Stack。接下来，利用算法 2，可以求出 Bak_Stack[Position(S(h))]＝{S(i),S(j),S(k)}（也可能是{S(f),S(g),S(p)}）作为 SubTask(1)的解中{S(h),S(p)}的备选方案。

在服务组合执行阶段，服务执行代理将按照 SC_Stack 中的组合方案以及服务的 location 属性（表示服务的位置），依次调用 S(b),S(h),S(p),S(m),S(n)。

如果服务执行代理在调用 S(h)的过程中，S(h)发生失效，服务执行代理将回滚到 Bak_Stack[Position(S(h))]的入口位置，并调用 Bak_Stack[Position(S(h))]替换 SC_Stack 中的{S(h),S(p)}，以保证服务组合的正确、顺利进行。

如果发生失效的不是 S(h)，而是 S(b)，则服务执行代理将触发服务组合引擎生成补偿方案。在本例中，会生成{S(a),S(d),S(j),S(k)} 或者{S(a),S(e),S(h),S(p)}，作为 SC_Stack 中{S(b),S(h),S(p)}的候选补偿方案。服务执行代理将回滚至该补偿方案的入口，并启动补偿组合方案的执行。

4 性能分析

本节将就 RScMPG 的性能做如下分析。

4.1 算法 1，算法 2 分析

对于普适环境下的服务组合来说，不可预期的服务失效在所难免，RScMPG 的目的在于尽可能地降低服务组合的失效率，以提高服务组合的可靠性。利用算法 1 和算法 2，可以很好地解决这个问题。

表 3 列出了，10 次实验中，当前服务区域内，“易失效节点数”占“全部服务节点数”的比例（易失效率）分别在 10%，20%，25%和 50%的情况下，服务组合路径长度以及该路径所包含的易失效节点的个数。

表 3 服务组合路径长度及包含的易失效节点数

组合路径长度	易失效率 10%	易失效率 20%	易失效率 25%	易失效率 50%
8	2	2	2	5
17	1	3	1	5
25	1	3	4	10
32	2	5	9	17
39	5	5	12	20
46	4	10	14	21
55	6	15	9	21
63	3	11	15	24
70	7	14	15	30
77	7	14	28	47

根据表 3，计算在 10 次实验中，4 种不同情况下，服务组合路径上，“易失效的服务节点个数”占“服务组合路径上的服务节点数”的比例，如图 2 所示。

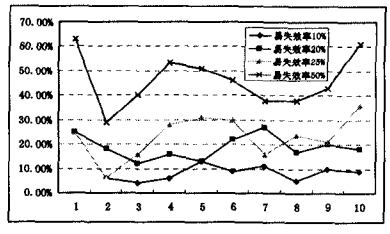

图 2 服务组合路径上易失效服务比例

由图 2 可以看出，如果当前服务区域内“易失效服务节点数/总服务节点数”的值为 m，那么，对于该区域内的一个

Task 请求，生成的服务组合方案中的“易失效服务节点数/服务组合路径长度”的值约为($m-10\%$，$m+10\%$)之间。可以认为，当前服务区域内“易失效节点”的所占比例，与一个组合方案中包含的“易失效节点”的所占比例大致一致。

下面是算法 1，算法 2 在提高服务组合可靠性方面的性能。

根据图 2 的分析，可以假设一个服务组合方案中的“易失效服务节点数/服务组合路径长度”的值，它与当前服务区域内“易失效服务节点数/总服务节点数”的值相等。假设服务组合路径上 Reliability 小于阈值的服务(“易失效”服务)发生失效的平均概率是 x，其他服务发生失效的概率是 y，备选服务组合方案中的服务发生失效的概率是 z。

对于一个服务组合路径长度为 10 的服务组合方案，图 3(其中，$x=15\%$，$y=5\%$，$z=10\%$)和图 4(其中，$x=15\%$，$y=2\%$，$z=5\%$)反映了当前服务区域内易失效服务所占比例由 10%到 100%的不同情况下，一般的基于图的服务组合策略[1]与 RScMPG 在组合方案的失效率上的差异。

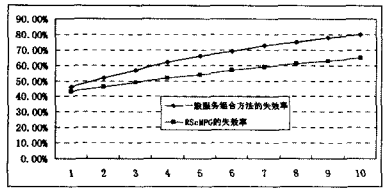

图 3 RScMPG 与一般服务组合的失效率比较(I)

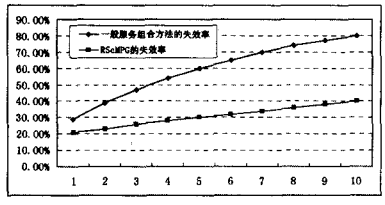

图 4 RScMPG 与一般服务组合的失效率比较(II)

其中，横坐标代表服务组合路径上包含的“易失效”服务节点的个数，纵坐标代表服务组合的失效率(失效率的计算见式(2))。由图 3 和图 4 可以看出，相比一般的基于图的服务组合策略，RScMPG 在组合方案的失效率上有明显的降低，随着易失效服务所占比例的增加，这种降低程度更加突出。这是因为，算法 2 中生成的备选组合方案在应对大多数情况下的服务节点失效时起到了很关键的作用。由此可见，算法 1 和算法 2 对改善服务组合的失效率有显著的效果。

4.2 算法 3 分析

与其他的服务组合方法[1]相比，在发生服务组合失效时，RScMPG 会尽可能利用已有的组合方案，因而，需要执行的回滚操作更少，生成的补偿组合方案的长度也更小。

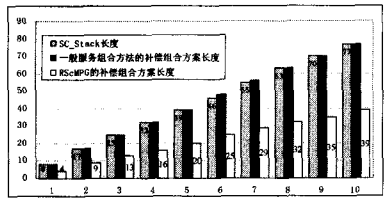

图 5 RScMPG 与一般服务组合的补偿服务长度比较

在图 5 中，一组柱状图自左向右依次为：针对某一 Task 生成的组合方案 SC_Stack 的长度，发生服务失效时，其他服务组合方法[1]生成的补偿组合方案的长度，以及 RScMPG 生成的补偿组合方案的长度。

从这 10 组实验数据中可以看出，RScMPG 的补偿组合方案的长度仅仅约为一般服务组合的一半，这是因为，在发生服务组合失效时，RScMPG 会尽可能利用已经执行的服务组合方案的结果，并在此基础上，生成补偿服务组合方案，而不是直接针对原 Task 生成新的组合方案。由此可见，在服务组合的执行过程中，若发生服务组合方案失效，利用算法 3，RScMPG 可以将补偿代价降低一半左右。

结束语 普适计算环境下的服务组合，面临着可靠性方面的更大挑战。本文从普适环境的特点出发，提出了一种普适环境下基于图的可靠服务组合机制模型 RScMPG，该模型包含两个阶段，分别从“预防”和“检测-补偿”两个角度提出了各自的算法，提高了服务组合的可靠性。

RScMPG 模型适用于当前服务区域内的服务个数不是很多，并且执行服务组合方案的代价大于生成服务组合方案的代价的普适场景。

本文使用的是一种集中式的服务组合模式，考虑到算法 1 中各个 SubTask 之间潜在的并行性，以及普适计算环境下，设备在拓扑结构上的分布特性，建立一种分布式的可靠服务组合模型将是下一步的工作目标。

参 考 文 献

[1] Kalasapur S, Kumar M, Shirazi B A. Dynamic Service Composition in Pervasive Computing[J]. IEEE Transaction on Parallel and Distributed System, 2007, 18(7): 907-918

[2] Hashemian S V, Mavaddat F. A Graph-based Approach to Web Services Composition[C]// Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'05). Trento, Italy, 2005: 183-189

[3] 钟读杭. Web 服务组合的可靠性预测研究[D]. 长沙：国防科技大学，2007

[4] 曹科强，顾庆，任颖新，等. 服务组合中基于 DTMC 的可靠性和性能分析[J]. 计算机科学，2009，36(10)：179-182

[5] Hamadi R, Benatallah B. A Petri net-based model for Web Service Composition[C]// Proceedings of the 14th Australasian Database Conference. Adelaide, Australian, 2003: 191-200

[6] Chakraborty D, et al. A Distributed Service Composition Protocol for Pervasive Environments[C]// Wireless Communications and Networking Conference(WCNC). March 2004: 2575-2580

[7] Brønsted J, Hansen K M, Ingstrup M. Service Composition Issues in Pervasive Computing[J]. IEEE Pervasive Computing, 2010, 9(1): 62-70

[8] Weiser M. The computer for the twenty-first century[J]. Scientific American, 1991, 265(3): 94-104

[9] Satyanarayanan M. Pervasive Computing: Vision and Challenges [J]. Personal Communications, IEEE, 2001, 8(4): 10-17

[10] 钱柱中，陆桑璐，谢立. 基于 Petri 网的 Web 服务自动组合研究[J]. 计算机学报，2006，29(7)：1057-1066