服务描述和服务匹配研究*)

臧 志 金蓓弘 李玉明
(中国科学院软件研究所软件工程技术中心 北京 100080)

摘 要 服务发现是分布式环境下进行资源共享、数据集成、流程协作的前提,而服务描述和服务匹配策略是其中的两个关键问题,服务的多样性、异构性和其上下文的动态变化使这些问题更加复杂。为此,提出了一种能支持多样性、屏蔽异构性、适应变化上下文的服务描述模型SDMDHD,并针对此模型设计实现了包括精确匹配、语义匹配在内的服务匹配引擎,还给出了对该匹配策略的理论评估和实验结果。

关键词 服务发现,服务描述模型,服务匹配策略

Research on Service Description and Service Matching

ZANG Zhi JIN Bei-hong LI Yu-ming
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)

Abstract Service discovery is the prerequisite of resource sharing, data integration and process collaboration in distributed environments. Service description and matching are key issues which become more complex due to diversity, heterogeneity and changing contexts of services. This paper proposes a service description model SDMDHD which can describe various services, hide their heterogeneities and adapt to changing contexts. Based on this model, service matching engine is designed and implemented to provide not only accurate matching but also semantic matching. This paper also gives the theovetilal evaluations and experiment results of the service matching mechanism.

Keywords Service discovery, Service description model, Service matching mechanism

1 引言

分布式环境是当前实现资源共享和用户协作的基础环境。服务提供者可在网络上共享服务资源,服务请求者可通过网络发现和访问自己所需的服务。网络中的服务资源通常具有多样性、异构性及动态变化的特点,体现在:(1)服务资源种类繁多,表现形式多种多样;(2)服务资源的形态、调用方式及底层通信机制互不相同;(3)服务的状态随时间动态变化。因此,建立一个支持多样性、屏蔽异构性、适应动态性的服务描述模型,同时构建一个高效的服务匹配算法,对于一个完备的服务发现系统而言至关重要。

本文通过服务发现原型系统 Service CatelogNet 的开发,提出了一种支持服务多样性、屏蔽异构性、适应动态性并且提供语义描述的服务描述模型 SDMDH,该模型通过动态上下文信息描述服务的非功能性特性。针对此模型,本文还设计和实现了包括精确匹配、语义匹配在内的服务匹配引擎,同时给出了对该匹配策略的理论评估和实际实验结果。

目前,对于服务发现,学术界和工业界进行了许多探索和研究,已经提出了一些较成熟的协议,开发了一些框架或系统。在这些已有工作中,服务描述模型及其相应的存储策略和匹配算法都是重要的关注点。服务描述主要关注其描述能力。早期的一些服务发现协议如 SLP[1], Bluetooth SDP[2],使用预定义的模板来描述和查询不同的服务。模板固定的服务描述方式可以简化服务的匹配,但难以表达服务属性之间纷繁的层次性关系,限制了服务描述的能力,同时难以适应服务的多样性及异构性。另一方面,其他一些服务发现协议,如 INS[3], Ninja SDS[4], GSD[5], Service rings[6]和 Superstring[7],采用了 XML,类 XML(如嵌套的属性-值对)或基于 XML 的描述语言(如 OWL)。例如,通过引入 DAML + OIL,GSD 不仅增强了服务描述的灵活性,同时实现了基于简单分类推理的语义级服务发现,然而这种灵活性也带来了服务匹配上的复杂性。主要用于 Web 服务应用场景中的 UDDI,则是根据用户的功能约束发现服务,没有考虑用户的语义需求。与已有工作相比,Service CatalogNet 提出的服务描述模型的特点在于提供了轻量级的语义描述,可以通过动态上下文描述服务的非功能性特性,同时通过引入匹配度的概念提供了语义级的服务查询。模型还支持用户自定义的标签、属性及数据类型,从而提高了模型的可扩展性。

2 服务模型和服务存储

通过权衡模板类服务模型和 XML 系列服务模型的利弊,我们设计了一种支持服务多样性、屏蔽异构性、适应上下文动态变化的服务描述模型 SDMDHD(Service Description Model supporting Diversity, Heterogeneity, Dynamic contexts of services)。SDMDHD 基于 XML 语言的一个子集,引入语义支持,并采用模板与自定义标签、自定义数据类型相结合的策略提供可扩展性。本节针对此模型给出了索引存储机制。

2.1 服务描述模型 SDMDHD

*)本文研究得到国家高技术研究发展计划 863 资助项目(编号 2006AA01Z231)和国家自然科学基金(编号 60673123)的资助。臧 志 硕士研究生,研究方向为分布式计算、软件工程;金蓓弘 博士,研究员,研究方向为分布式计算、软件工程;李玉明 硕士研究生,研究方向为分布式计算、软件工程。

服务信息是通过 XML 的一个子集来描述的。首先给出基本概念的定义。

定义 1(标签) 服务描述的主要组成部分,对应于 XML 中的元素。

定义 2(属性) 描述标签的元信息,对应于 XML 元素的属性。每个属性隶属于某个数据类型。

定义 3(数据类型) 属性取值所允许的数据类型及该类型所支持的操作符如图 1 所示。我们引入 RangeOf[T]类型来表示属于类型 T 的数据的取值区间,其中 T 必须是可以比较的。比如 RangeOf[Integer]表示整型区间,RangeOf[Time]表示时间区间,RangeOf[Time]数据取值的一个例子是[9:00:00,18:00:00]。

Integer	:=, >, <, >=, <=	Float	:=, >, <, >=, <=
Boolean	:=	Char	:=, >, <, >=, <=
String	:=, subString, >,<	Date	:=, >, <, >=, <=
Time	:=, >, <, >=, <=	Enumeration	:=
RangeOf[T]	: IsContain, IsIntersect, <, =, >	Set[T]	: sub, IsIntersect

图 1 SDMDHD 中的数据类型和适用于该类型的操作符

服务信息中的所有标签及属性都属于某个名字空间,在服务描述及查询时可指定其名字空间。若不指定,则默认为缺省的名字空间。

SDMDHD 对服务信息预定义了一些标签模板,其中第一级标签为 Service 标签,它包含了一些服务的通用属性,如服务提供商的标识 sp-uid、用于服务软状态管理的 expiredTime 等;每个 Service 标签必包含三个子标签:ServiceSpecifier,ServiceContext 和 ServiceRecord。三个子标签的描述内容分别是:(1)ServiceSpecifier,用于描述所提供的服务功能以及服务的特征。(2)ServiceContext,用于描述服务的一些上下文信息,该标签下属的子标签可以通过 expiredTime 属性和 updateEntry 属性来实现服务的动态上下文的更新,其中 expiredTime 属性表示服务的过期时间,updateEntry 属性表示获取当前上下文的访问入口点。例如,在查询到图 2 所示的打印机服务时,若发现 expiredTime 早于当前时间,那么系统将调用 updateEntry 的访问入口,取得当前该项的值。另外,还可以为标签设定历史平均值系列属性包括 HistoricalAverageLoad, HistoricalAverageProcessingTime, HistoricalMTTF,HistoricalMTTR,通过这些值,用户可以对服务的质量,包括平均负载、平均处理时间、平均出故障时间间隔、平均故障修复时间等进行评估。(3)ServiceRecord,用于描述服务的访问路径、服务访问权限以及服务访问方式等相关信息。

上述三个标签(包括其所包含的子标签)中,前两者可出现在服务查询条件中,最后一个不能作为服务查询条件,仅作为查询结果返回,例如将服务访问方式返回给用户。

依据上述定义,一个服务描述的示例如图 2 所示。

每个服务描述文档受对应的模式(Schema)文件的约束,该文件类似于 XML Schema 和 RDF Schema,包括如下内容:定义数据类型及其操作符;定义所有属性的数据类型、属性所属的标签元素以及属性之间的语义关系;定义标签、标签所含的属性及标签与标签的语义关系。图 2 所示的服务示例对应的 Schema 片段如图 3 所示。

由于网络环境中服务资源的多样性,难以事先定义出所有的服务描述所需的标签、属性和数据类型,因此我们在模式文件的基础之上提供对用户自定义的属性、标签、数据类型的支持,如图 3 中的 myunit 属性。由以上规则定义的服务描述模型,称之为 SDMDHD。

```
<Service name = "MyPrinter" sp-uid = "12345" uid = "KB2536" classification =
"printer" >
        <ServiceSpecifier>
                < NS1:Model NS1:brand = "HP" NS1:type = Laser NS1:modelNo =
"HKG1234"/>
             <NS1:PrintPrice  NS1:unit="RMB/page"  NS1:time="[9:00,  18:00]"
NS1:value="0.3"/>
                < NS1:PaperSizes NS1:paperSize="{Letter, legal, A4, A5 }"/>
                <  NS1:Standardinterfaces  NS1:interfaces="{USB,  Direct  Print
Port}"/>
        </ServiceSpecifier>
        <ServiceContext sensitive = true>
         <Location name="Angel Print Shop">
            <City vaule = "Beijing" />
          </Location>
                <Device value="{PC,PDA,Phone}"/>
                        <AccessTime value = "233ms" updateEntry =
                          "124.16.136.223:8080/ updateContext"   expiredTime =
                          "2007-12-12 06:32:23"/>
        </ServiceContext >
        <ServiceRecord URL = "124.16.136.223:8080/MyPrint" username = "iscas"
                psd = "otc">
    </ServiceRecord >
</Service>
```

图 2 打印服务的服务描述示例

```
<?xml version="1.0" encoding="GB2312"?>
<otc:namespace name="NS1">
    ……
<!--定义一个标签 PrintPrice -->
        <otc:tag name="PrintPrice">
                <otc:attribute ref="unit"/>
                <otc:attribute ref="time"/>
<otc:attribute ref="value"/>
        </otc:tag>
    <!--定义一个属性 unit，类型为枚举-->
        <otc:attribute name="unit" type="otc:enum">
                <otc:restriction base="otc:enum">
               <otc:emum Value value="RMB/page">
                    <otc:enum Value value="km/h" />
                    <otc:enum Value value="m/s" />
          </otc:restriction>
        </otc:attribute>
        <!--定义一个属性 value，类型为整数-->
        <otc:attribute name="value" type="otc:integer"/>
        ……
</otc:namespace>
<otc:namespace name="NS2">
        <!--定义一个属性 myunit，类型为枚举，为 NS1:unit 的近义词，相似度为
0.95-->
        <otc:attribute name="unit" type="otc:enum">
                <otc:restriction base="otc:enumeration">
               <otc:emum Value value="RMB/page">
                    <otc:enum Value value="km/h" />
          </otc:restriction>
        <otc:relationship ref="NS1:sex" type= " synonym " similarity="0.95" />
        </otc:attribute>
</otc:namespace>
```

图 3 与图 2 服务描述对应的模式文件

针对 SDMDHD,我们提供了语义匹配。语义匹配依赖模式文件中定义的标签及标签、属性与属性之间的语义关系(如同义词、近义词、继承等)及其相似程度,通过综合这些信息可以得到任意两个标签/属性之间的匹配程度。例如,表示地理位置可以用 Location,District,Place 或 Region 标签,它们名称不同,但如果模式文件中定义了它们的语义关系,那么它们仍是可匹配的。通过语义匹配可以提高服务发现系统的智能性。

2.2 服务描述的存储

基于可能存在大规模增删改等并发写操作的情况,我们目前采用按服务提供商进行分类并以多个文件的方式对服务描述的 XML 文件进行存储。在此过程中,若某个文件大小超过阈值,则进行拆分,拆分的原则是尽可能在单个文件之中存放同一服务提供商提供的服务。这样,最终形成的结果是提供较多服务的服务提供者独用一个或多个服务描述文件,

而提供较少服务的服务提供商共用一个服务描述文件，从而在文件数目与存取效率之间获得一个折衷方案。

我们为上面的存储方式建立服务提供商 sp-uid 与其使用的文件的索引映射 $Index_{sp\text{-}doc}$。同时，为了匹配算法的需要，我们对服务描述的内容进行位置编码，建立位置编码索引 $Index_{location\text{-}encode}$，即将服务描述中的每个元素表示为如下的四元组：(元素名称，元素所属的 Service 元素对应的记录 id，元素位置信息，元素所具有的属性)，其中元素位置信息由一个四元组表示：〈元素所在的文档 id，元素的起始位置，元素的结束位置，元素所在的层次〉。为了方便快速定位，我们采用相对于服务的 Service 元素的相对偏移量作为元素起止位置的表示方式。可以看出，该位置编码表示方式可以很容易确定元素之间的父子关系以及祖先后代关系。通过位置编码得到的记录集合可以存放在数据库中，也可以采用等长记录的方式存于文件之中。在系统实现中，我们采用了后者。

3 服务查询

我们采用类似于 XQuery 的描述方式表示查询。可以指定两类限制条件，一类是路径信息约束，另一类是属性的取值约束。查询的结果是符合要求的服务描述的某些片段。图 4 给出了一个服务查询请求实例。

```
Select "Service" Where Service [classification = "printer"] AND
Service/ServiceSpecifier/model[type = "Laser"] AND ../Standardinterfaces[value =
"USB"] AND Service/ServiceContext[sensitive = true]/Location/city[value =
"Beijing"] AND ../ Bandwidth [value ≥ 120 AND unit equals "MHz"] || Mode =
"MultiC" || TransmissionDelay < 100ms AND ServerLoad < 10job
```

图 4 服务查询请求实例

我们首先将查询语句解析成一多叉树结构 QueryTree，树中的叶节点为路径表达式(如 Service/ServiceContext[sensitive="true"] / Location [name="LA"])，非叶节点为 AND，OR 等逻辑操作符。针对 QueryTree 叶子节点的路径表达式，我们构建了三种数据结构：AD-Pair 用来存储路径表达式中标签之间的父-子或祖先-后代关系，TagCondition 用来存储路径表达式中每个标签的限定条件，AttributeCondition 存储某标签的所有属性的限定条件。图 5 给出了多叉树 QueryTree 的示意图。

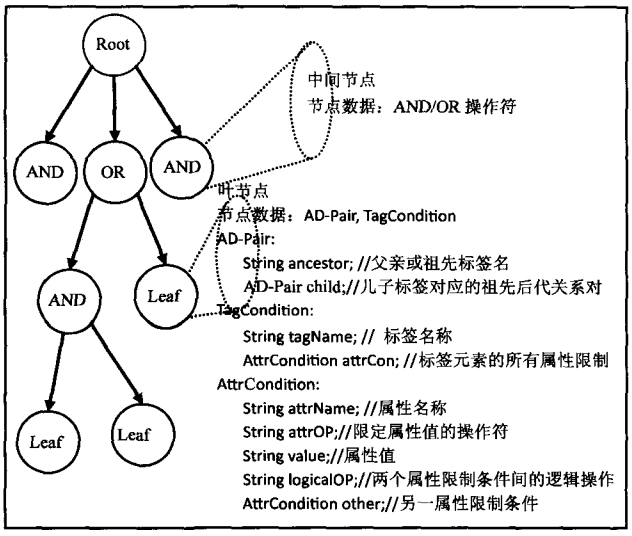


图 5 Query Tree 示意图

我们提供三种服务查询方式：精确查询、部分查询和语义查询。精确查询和部分查询要求查询语句包含的信息与服务描述完全匹配，二者不同之处在于查询语句的详尽程度；而语义查询主要涉及标签元素的语义匹配和属性的语义匹配。

精确查询和部分查询在算法上非常相似，其过程为：首先将查询路径表达式语句解析为 *AD-Pair* 集合 R 和单个标签限定条件集合 S；之后针对 S 中的每个标签 Tag_i 在 $Index_{location\text{-}encode}$ 中查找到对应 Tag_i 的记录集合 T_i，然后用 Tag_i 的属性限定去匹配 T_i 的每一条记录，将不匹配 Tag_i 的从 T_i 中删除。由于属性限定条件内可以包含多个操作符，因此若该属性的数据类型是用户自定义的，那么用反射技术来获得用户定义的操作符计算规则。

语义匹配的执行过程与上述过程的不同之处在于引入语义支持，具体体现为：(1)在根据标签名取出相应的集合 T_i 的时候，将根据标签之间的语义关系图，添加名字不同但语义相似度在一定范围之内的标签，并记录下相似度(即匹配度)。(2)在匹配标签包含的属性过程中，根据属性之间的语义关系图，将相似度在一定范围之内的属性也纳入为比较对象(前提是其数据类型相同)，同时记录下其相似度。(3)在得到叶节点的满足要求的服务集合的时候，按照相似度对查询结果进行排序，并按照用户要求(可能要求返回 n 个服务)返回满足条件的服务集合。

4 性能分析与测试

为了对服务匹配算法进行性能分析，首先设置如下参数：系统中服务总数为 n、标签总数为 t、每个服务描述平均使用 D 个标签、服务描述对应的 DOM 树的高度为 h、服务描述对应的 DOM 树的任意节点的子节点个数平均为 M，每个标签的相似标签数平均为 r_t，每个属性的相似属性数平均为 r_a。

根据上述参数可得：路径表达式的最大长度为 h，该路径表达式中标签的个数为 $h+1$。服务描述对应的 DOM 树中叶子节点的个数为 M^h，则查询表达式中出现的路径表达式最多为 M^h 个，即为从根节点到叶子节点的路径的数目。进一步可推出，查询时间 $T=M^h\times T_1$，其中 T_1 为查询一个路径表达式所用的时间。下面将计算 T_1：设每个标签对应的记录集合中服务的平均个数为 $n/B(B\geqslant 1)$，经过筛选之后剩余标签记录个数为原先的 $1/C(C\geqslant 1)$，则筛选前每个集合中每个服务对应的标签记录个数为 $(nD/t)/(n/B)=DB/t$，筛选后每个服务对应的标签记录个数为 DB/tC，从而有 $T_1=(h+1)\times(nD/t)+(n/B)\times(DB/tC)^h$，由此查询时间 $T=M^h\times T_1=M^h\times((h+1)\times(nD/t)+(n/B)\times(DB/tC)^h)=M^h\times(n/B)\times((h+1)\times(DB/t)+(DB/tC)^h)$。

在系统稳定运行后，上式中 M,B,t,C 均为常数，则可以看出 T 仅与查询语句的复杂度 h 及系统中服务个数 n 有关。若查询语句长度固定的话，则查询时间正比于系统中服务的个数 n。在语义查询情况下，由于算法的框架并没发生变化，只是在查询过程中查到每个标签对应的服务集合的个数受语义的影响最大，可变为 $r_t\times(n/B)$，而 r_a 对查询并不造成显著影响，所以相应的查询时间为：$T=M^h\times(r_tn/B)\times((h+1)\times(DB/t)+(DB/tC)^h)$，$T$ 依然正比于 n。下面的实验也验证了这种关系。

我们对语义查询匹配算法的性能评估实验是针对两种情况进行的：(1)服务规模(10000 个服务)固定，并发查询数量

(下转第 287 页)

Before R：当 R 成立时，表达的范围是 R 第一次成立之前的所有时刻，且不包括第一次成立的时刻；当 R 不成立时，表达的范围不存在。

After R：当 R 成立时，表达的范围是 R 第一次成立之后的所有时刻，且包括第一次成立的时刻；当 R 不成立时，表达的范围不存在。

Between R and Q：当 Q 在 R 之后成立时，每个 R 成立的时刻都与紧随其后的、第一次 Q 成立的时刻，构成 R 成立之后到 Q 成立之前的 R-Q 区间，包括左边界但不包括右边界；当 R 或 Q 不成立、Q 在 R 之前或与 R 同时成立时，表达的范围不存在。

After R until Q：当 Q 在 R 之后成立时，每个 R 成立的时刻都与紧随其后的、第一次 Q 成立的时刻，构成 R 成立之后到 Q 成立之前的 R-Q 区间，包括左边界但不包括右边界；当 R 成立，而 Q 不成立时，表达的范围是从 R 成立开始直到程序结束的区间，包括 R 成立的时刻；当 Q 成立而 R 不成立、Q 在 R 之前或与 R 同时成立时，表达的范围不存在。

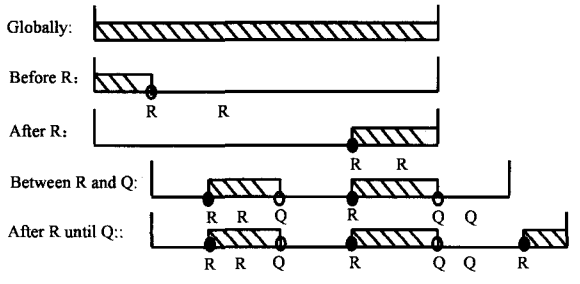


图 4　范围及其精确的语义描述

“After R until Q”与“Between R and Q”的区别仅在于：当 R 成立而 Q 不成立时，“Between R and Q”不能表达合理的范围，而“After R until Q”表达的范围存在。用图形描述 5 种范围的语义如图 4 所示。

结束语　本文采用 SPIN 模型检测工具，对规约模式系统对应的 LTL 表达式进行了分析验证。通过对比验证及现有的语义，给出了精确的语义描述。这种思想，文献[5]已经做了尝试，不过该文献只提出了错误，没有给出解决方法。本文在它的基础上，完成了对剩下的模式的验证，并给出了规约模式系统精确的语义描述。

参 考 文 献

[1] Dwyer M B, Avrunin G S, Corbett J S. A System of Specification Patterns[EB]. http://www.cis.ksu.edu/santos/spec_patterns. 1998

[2] Corbett J C, Dwyer M B, Hatcliff J, et al. A language framework for expressing checkable properties of dynamic software [C]// Havelund K, Panix J, Visser W, eds. LNCS 1885. Berlin: Springer-Verlag, 2000: 205-223

[3] Hatcliff J , Dwyer M . Using the Bandera Tool Set to Model - Check Properties of Concurrent Java Software//Larsen K. G, Nielsen M, eds. LNCS 2154. Berlin: Springer-Verlag, 2001. 39-58

[4] 戎玫，张广泉. 模型检测新技术研究. 计算机科学，2003，30(5)：101-104

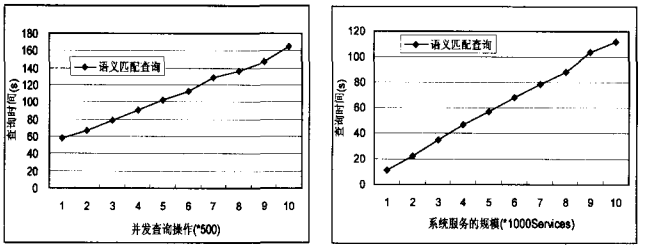
[5] Salamah S, Gates A, Roach S, et al. Verifying Pattern-Generated LTL Formulas: A Case Study // Godefroid P, ed. LNCS 3639. Berlin: Springer-Verlag, 2005: 200-220

[6] Yu Jian, Phan manh Tan, Han Jun, et al. Pattern based property specification and verification for service composition[C]//Aberer K, ed. LNCS 4255. Berlin: Springer-Verlag, 2006: 156-168

(上接第 268 页)

从 500 变化到 5000；(2)服务规模变化(1000～10000 个服务)，并发查询数量固定(1000 个)。实验所采用的查询语句类似于图 4，其中的约束条件是随机生成的。实验所报告的语义匹配消耗的时间如图 6 所示。

从图 6 可以看到，支持语义的服务匹配策略其查询所需时间随并发查询操作数量或系统服务的规模增大均近似为线性增长。在系统规模为 10000 个服务时，平均每增加 500 个并发查询操作，所需时间增加 11.221s。当固定并发查询数量为 1000 时，查询时间以 11.218s/1000Services 为起点，以 11.198s/1000Services 的速率增长。



(a)系统服务的规模：10000Service　(b)固定并发查询操作所数量：1000

图 6　语义匹配策略性能

语义匹配策略扩展了服务描述与查询条件的灵活性，从而在很大程度上提高了服务发现的表达能力和智能性。但与系统的精确/部分匹配操作的性能相比，语义匹配策略在提高了匹配能力的同时也带来了性能下降、匹配度规则难以制定的问题。

结束语　本文提出了一种支持服务多样性、异构性及上下文动态变化的服务描述模型 SDMDHD，同时设计实现了高效的存储结构和支持精确匹配/部分匹配、语义匹配的服务匹配策略。下一步，我们将进一步研究服务模型的语义支持，提高语义匹配策略的性能和智能性，并将工作拓展到 MANET 环境下，同时将研究多种服务发现协议的互操作问题。

参 考 文 献

[1] Guttman E. Service Location Protocol: Automatic Discovery of IP Network Services. IEEE Internet Computing, 1999, 3(4)

[2] Bluetooth SIG. Specification of the Bluetooth System, 2004

[3] Adjie-Winoto W, Schwartz E, Balakrishnan H, et al. The Design and Implementation of an Intentional Naming System // 17th ACM Symposium on Operating System Principles. 1999

[4] Czerwinski S E, Zhao B Y, Hodes T D, et al. An Architecture for a Secure Service Discovery Service // 5th Annual International Conference on Mobile Computing and Networks. 1999

[5] Chakraborty D, Joshi A, Yesha Y, et al. GSD: A novel group-based service discovery protocol for MANETS // IEEE International Conference on Mobile and Wireless Communications Networks. 2002

[6] Klein M, Konig-Ries B, Obreiter P. Service rings - a semantic overlay for service discovery in ad hoc networks // DEXA Workshops. 2003: 180-185

[7] Robinson R, Indulska J. Superstring: A Scalable Service Discovery Protocol for the Wide Area Pervasive Environment // Proc. of the 11th IEEE International Conference on Networks. Sydney, September 2003