语义聚集的P2P服务组织模型

兰明敬
(解放军信息工程大学信息工程学院 郑州 450002)

摘 要 针对集中式和传统分布式服务注册与发现机制中存在的问题,提出一种新的服务组织模型。该模型归纳服务系统中各服务功能来建立语义树,依据此语义树产生的语义串对服务进行标识,采用改进的Kademlia算法将服务组织起来,形成按语义树聚集的、使用语义串进行结点发现的P2P覆盖网络,从而解决了单点失效、性能瓶颈问题,实现了不依赖注册中心和注册操作的、自发现的服务调用。它具有高可扩展性,能够支撑动态调度、模糊搜索等应用形式,已在某服务计算平台中成功应用,该平台已通过验收并连续运行近一年。

关键词 P2P,服务发现,语义树,模糊搜索
中图法分类号 TP393 **文献标识码** A

P2P Organization Model for Service Clustering Based on Semantic Tree

LAN Ming-jing
(College of Information Engineering, Information Engineering University, Zhengzhou 450002, China)

Abstract Aiming at solving problems existing in centralized and traditional distributed service discovery mechanism, a new service organization model was proposed. It identifies service by semantic string that comes from a semantic tree based on services functions in the service system. With an improved algorithm based on Kademlia, the model organizes all the services to form a P2P overlay network in which nodes are gathered together according to the semantic tree and can be found using semantic strings. This model has solved the single point failure and bottleneck problem, can find and invoke service without service registry. It is highly scalable, and supports more high-level applications such as dynamic scheduling, fuzzy search. The approach has been successfully applied in a service computing platform, which has already been verified and well operating nearly a year.

Keywords Peer to peer, Service discovery, Semantic tree, Fuzzy search

Web服务、云计算的迅猛发展及其在商业、市民服务、军事领域的广泛应用,对服务计算系统的实时性和可靠性提出了更高的要求。传统的服务组织模型包含提供者、请求者、注册者3个角色,其集中式的服务注册、发现机制必然存在单点失效和性能瓶颈等问题。

常见的解决方案是双机热备份技术,其通过共享数据的两台主机交替运行来保证业务的连续性。这种技术可以有效应对服务器宕机所引发的单点失效,但对于服务器所在网络发生故障所引发的服务中断却无能为力。同时,双机热备份技术的高昂造价和管理费用也使很多中小型企业望而却步。

2004年10月推出的UDDI3.0.2[1]规范中提出了附属中心的概念,将管理任务分解到多个本地注册中心上;随后人们提出了基于P2P覆盖网络的服务注册机制,将服务信息分散地存储在大量的、廉价的覆盖网结点上,利用覆盖网络本身的路由机制进行信息的注册与发现。这些分布式的服务注册与管理机制能够有效降低服务中断风险,降低系统造价。然而,结点数量的增加不可避免地带来结点扰动、数据一致性、冗余备份等一系列问题。

本文在前人研究的基础上提出了一种新的P2P服务组织模型,其特征为:1. 采用改进的Kademlia[2]覆盖网络组织服务;2. 结点标识并非传统的二进制码,而是一个能够描述结点对应服务功能的语义串;3. 该语义串来源于应用系统预先建立的、描述系统中各服务功能的语义树。

第2点特征是本文区别于传统P2P服务发现的关键。基于该特征,覆盖网络中功能相近的服务具有更近的距离,进而,网络中的各服务按语义树对应的功能域形成层次状的聚集关系。这种特点使得覆盖网路由过程与服务的功能以及上层的应用具有天然的关联,能够基于这种特征实现多种灵活有用的应用形式,包括无注册者角色和发布、查找操作的服务组织模型、异步服务调用、模糊调用、动态调度、并行计算、模糊搜索等。

本文第1节对模型及其关键点进行介绍;第2节对基于该模型的应用系统以及部分扩展应用形式进行了讨论;第3节对全文进行总结,讨论了模型的优缺点和适用领域。

1 模型

本文讨论的服务系统由多个主机结点组成,结点上部署若干服务,每个服务包含若干服务操作,同样的服务实现可以

到稿日期:2012-06-20 返修日期:2012-11-15 本文受国家863计划项目(2009AA012417)资助。
兰明敬(1982—),男,博士生,讲师,主要研究方向为P2P系统、服务计算,E-mail:lanmingjing@126.com。

在一个或多个主机结点上多次部署形成多个“服务副本”。

在此基础上，提出了一种新的、P2P 的、仅有请求者和提供者的服务组织模型，其基本思路是：

1)针对应用系统，对系统中各服务的功能进行归纳建立语义树 S-Tree；

2)修改 Kademlia 算法，以服务为覆盖网结点，依据服务在语义树中的路径而非传统的二进制形式对服务进行标识，相应地改造路由表和路由算法、路由表更新算法，建立支持语义路由的覆盖网络，称之为 SKad；

3)依托语义树和 SKad 网络来实现服务请求到服务操作的路由；

4)扩展 SKad 的功能，实现并行计算、模糊搜索等应用。

1.1 语义树

语义树描述了给定服务系统中各服务名称及其功能类别，其结构及例子如图 1、图 2 所示。语义树共有 $M=n+2$ 层，第 1 层为树根，用于区分其他服务系统；第 2 至 $M-2$ 层枚举了应用系统中所有服务操作的功能类别，其中各结点间的父子关系等价于对应功能类别的包含关系；M 层枚举了系统中的所有服务。

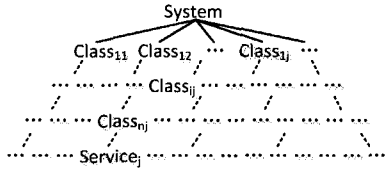

图 1 语义树结构

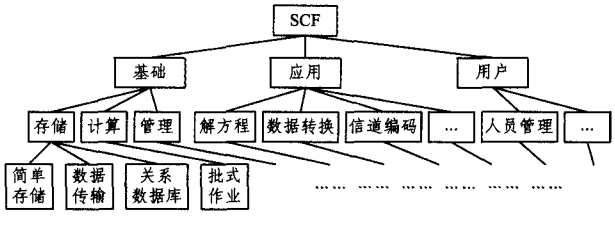

图 2 语义树例子

设树中任一结点的子结点数目为 W_i，且 $\max(W_i)=W$，即 W 为树中的最大兄弟个数。W 决定路由表 K 桶的大小(见第 1.5 节)，因此在实际应用中 W 的数值不宜过大，如果过大，则应对树进行调整，分裂直接子树过多的结点，减少 W 值。

1.2 结点

结点及结点标识是 SKad 区别于 Kademlia 的重要内容。

SKad 以服务为覆盖网结点，依据服务在语义树中的路径而非二进制编码的形式对结点进行标识：

$id=(s_1,s_2,\cdots,s_M)$

$s_1,s_2,\cdots,s_M$ 为语义树根结点到服务所在叶子结点路径上各结点的值，如图 2 中简单存储服务的 id 为：“SCF/基础/存储/简单存储”。显然，同一服务实现的不同服务副本具有完全相同的 id。

结点标识决定着结点在 SKad 网络拓扑中的位置。按照上述结点标识形式，在下文路由算法的作用下，结点(即服务)将呈现出按语义树所定义的功能进行聚集的形态，如图 3 所示。

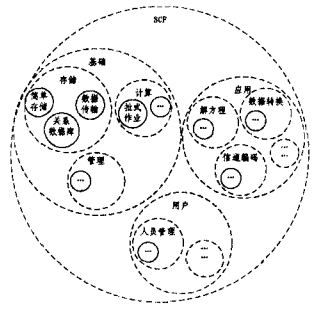

图 3 按语义聚集的服务组织结构

1.3 路由表

利用结点加入和更新算法(参见 Kademlia)，每个 SKad 结点都维持着一张路由表(Routing Table，T)。

路由表中存放着其他结点(即服务)信息。这些信息被称为实体(Item)，表示为：

$I=\{id,operation,endpoint,properties\}$

式中，id 为结点标识；$operations$ 为操作列表，描述结点对应的服务中包含的各操作及其输入、输出消息类型；$endpoint$ 为端点，包括该服务所在主机地址、端口号、URL 等信息；$properties$ 是属性列表，存放服务的 QoS、访问控制等信息，可以利用此项实现灵活多样的服务访问形式(见第 2 节)。

表 1 是路由表内容示例，为书写方便，表 1 中的一行表示路由表中存储的一项数据。

表 1 路由表内容示例

id	SCF/基础/存储/简单存储	…
Operations	operations={put,get};put_in_message={String,Object}; put_out_message={ };get_in_message={String}; get_out_message={Object};	…
endpoint	endpoint=“http://www.scf.com:8080/services/S3”	…
properties	qos={pe=32,mem=2GB,disk=30TB} restrict={user_type=AuthorizedUser,rganization=SCF}	…

路由表中的实体被划分到 $M+1$ 个 K 桶(KBucket)中。按 $1,2,\cdots,j,\cdots,M,M+1$ 的顺序对 K 桶进行编号，分别表示为 K_j。

设当前结点(即路由表所在结点)对应的实体为 i_c，其 id 为：$id_c=(c_1,c_2,\cdots,c_M)$，路由表中的某个实体 i_x 的 id 为：$id_x=(x_1,x_2,\cdots,x_M)$，则实体划分遵循以下规则：

$$K_j=\begin{cases}\{i_x\mid x_j\neq c_j\}, & j=1\\ \{i_x\mid x_j\neq c_j\wedge(\forall m)(1\leqslant m<j\rightarrow x_m=c_m)\}, & 1<j\leqslant M\\ \{i_x\mid x_j=c_j\}, & j=M+1\end{cases}$$

即，比较 id_x 与 id_n：

a)若第 1 元(即 x_1 和 c_1)不同，则 i_x 隶属于 1 号 K 桶(K_1)；

b)若第 1 元相同而第 2 元不同，则 i_x 属于 K_2；

c)若第 1、2 元相同而第 3 元不同，则 i_x 属于 K_3；

d)以此类推，对于 $j<=M$，若第 1 至第 $j-1$ 元全部相同，而第 j 元不同，则 i_x 隶属于 K_j。

K 桶中容纳的元素个数不小于 W，我们的取值为 $3W$(见第 1.1 节)。

1.4 结点发现

SKad 的结点发现即是服务发现，其过程为：

1)请求者查询语义树,确定目标服务 *id*。

2)请求者生成服务请求 $R=\{id, operation, args, cons, requester\}$。其中 *operation* 声明了所需要的服务操作;*args* 是调用目标操作所需的参数,用于服务调用;*cons* 是条件列表,包括对目标服务的服务质量、安全特征等方面的需求;*requester* 为调用者信息,包括调用者的 *ip*、端口等,用于回传结果以及更新路由表。

3)请求者将此请求交给任一 SKad 结点。

4)以此结点为出发点,请求在路由算法的作用下被路由到一个或多个目标结点上。

SKad 中上述过程并不孤立存在,而是伴随着服务调用等具体应用而进行的,第 2 节将结合应用对过程中涉及到的请求参数等问题进行详细阐述。

1.5 路由算法

路由算法根据服务请求查询路由表,将请求转发至与请求相匹配的目标结点(服务)或"下一跳结点",伪码如下:

```
ΔINPUT:Request R,Current Item i_c,Routing Table T;
ΔRETURN:Void;
ROUTING_ALGORITHM (R,i_c,T)
1. IF d(i_c,R) == 0   //若请求路径匹配当前结点 id
  1.1. IF isSatisfied(i_c,R->cons) THEN
        /* 判断 i_c 的 operations、properties 是否分别满足 R 中声明
           operation、cons
    1.1.1. invoke(R)  //满足则调用目标结点(服务)
  1.2. Return;
2. KB=getKBucket(R->id);//获取目标结点对应 K 桶
3. Foreach item in KB//检查桶中所有的实体
  3.1. IF isSuited (item->properties,R->cons) THEN
       //判断候选实体是否满足 R 中声明的条件
    3.1.1. NodeList+= item;//满足则加入结果列表
4. Result=getOptimal(NodeList,N);
     //依据匹配度,从结果列表中选取 N 个最优结点
5. Foreach item in Result
  5.1. forword(item,R) //转发请求至下一跳结点
```

路由算法首先使用 isSatisfied 方法来判断当前结点对应实体是否与请求相匹配,即:

1)当前实体 *id* 与目标服务 *id* 完全匹配(即匹配度为 0,见下文);

2)结点 *operations* 中包含请求中的 *operation*;

3)结点 *properties* 满足请求中声明的 *cons*。

若完全匹配则进行服务调用,否则从目标服务 *id* 所在 K 桶中获得实体列表,从列表中选取符合请求条件 *cons* 且匹配度较高的实体,将请求转发到实体对应的结点上(将这些结点称为"下一跳结点")。

设有两服务或结点 *id* 分别为:$id_x=(x_1, x_2, \cdots, x_M)$,$id_y=(y_1, y_2, \cdots, y_M)$,则 id_x、id_y 的匹配度定义为:

$$d(id_x, id_y)=\sum_{i=0}^{M}\left\lceil\left|\frac{\eta(x_i)-\eta(y_i)}{\lambda}\right|\right\rceil\times 10^i$$

$d(id_x, id_y)$越小,则 id_x 与 id_y 的匹配度越高,当 $d(id_x, id_y)=0$ 时,意味着 id_x 与 id_y 完全匹配。

上式中,η 为一个数值转换函数,能将字符串转换为一个整数。η 如何转换并不重要,只要满足条件:同样的字符串转换为同样的值,不同的字符串转换为不同的值。有很多这样的函数可以选择,比如将组成 x_i、y_i 字符串的字母按顺序取其 ASCII 值连接在一起。λ 为 η 取值范围中的最大值。

采用 η 函数将 id_x、id_y 中的每个单元串 x_i、y_i 分别转换为数字并计算其差值,而不是仅仅比较 x_i、y_i 是否相等,这种策略将使得 SKad 路由过程具有更好的收敛特性。

用 isSuited 方法来判断某实体 item 是否可以作为下一跳结点的判断依据为:

1)实体是否在线;

2)检查实体的 *properties*,确定其是否满足算法预定义的若干选择策略(比如带宽限制等);

3)检查实体的 *properties*,确定其是否满足服务请求中声明的 *cons*。与 isSatisfied 中类似判断不同,此项判断的意义在于:请求者可以在 *cons* 项中添加特定的信息(比如 ip 地址限制等),让路由避开某些结点,即请求者不仅仅可以有条件地选择目标,也可以选择到达目标的路径。

1.6 路由表更新

路由表更新算法与 Kademlia 类似,值得注意的是,当某个 K 桶已满,需要对待加入的实体进行选择时,应采用均匀分配的原则:

设当前 K 桶编号为 j,桶中实体按其 *id* 的第 j 元相等进行聚类形成若干集合,则这些集合的大小值应该尽量分布均匀。即,当实体需要加入而 K 桶已满时,从元素数目最多的集合中选取一实体,删除该实体。

与匹配度计算公式类似,该原则将使得路由过程具有更好的平均收敛特性。

2 应用

按语义聚集并依赖语义树进行结点(服务)发现的特征使得 SKad 具有高度的灵活性和可扩展性,在其基础之上能够实现多种方便、实用的应用形式。下面以某服务计算系统为例,对部分应用形式进行叙述。

2.1 基于 SKad 的服务计算系统

我们使用 SKad 搭建了某服务计算系统(SZ-CC),该系统当前具有 300 个主机结点,面向高性能计算领域提供基础服务 12 项、应用服务 43 项、用户服务 29 项。其中基础服务和应用服务被部署到所有主机结点上,其副本数(服务数 * 主机结点数)为 16500;用户可根据需要自行开发用户服务并部署到权限范围内的某些服务结点上,当前用户服务的操作副本数为 3470。系统已经通过验收并连续运转一年。

图 4、图 5 分别是 SZ-CC 服务结点的软件结构和服务语义树。

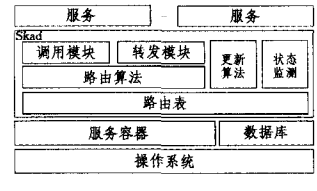

图 4 SZ-CC 软件结构

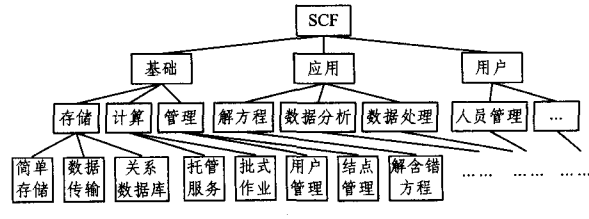

图 5 SZ-CC 语义树

2.2 服务调用

传统的服务系统中,服务的调用涉及到注册、发现和绑定

3个过程；而在SZ-CC中，调用者访问服务只需要一个步骤，即生成服务请求并交给SKad网络，SKad会自动将请求路由到调用目标产生调用。具体过程如下：

1)调用者查询语义树，根据目标服务的名称、功能或其他描述信息确定目标服务 *id*。

2)生成服务请求 $R=\{id, operation, args, cons, requester\}$。*operation* 是目标服务操作，包含操作名和输入、输出消息类型；*args* 是调用目标操作所需的参数；*cons* 是条件列表，包括对目标服务的服务质量、安全特征等方面的需求，用于在模糊调用的情况下对目标服务进行过滤(见下文)；*requester* 为调用者信息。

3)调用者将服务请求交给SKad的任一覆盖网结点(服务)。

4)以此结点为出发点，调用请求在路由算法的作用下被路由到一个或多个目标服务(结点)上。

5)完成服务调用并返回调用结果。

服务调用过程集中反映了SKad的特点。在SKad中，服务发现、调用所依赖的注册信息事实上被分为两类，一类是语义信息，包括服务所属功能类别及其相关描述信息，这些信息属于"元信息"、静态信息，预置在语义树中；除第一类信息外，服务操作、安全限制、服务质量信息等其他信息均属于第二类，称之为属性信息，这些信息存储在与服务相匹配的覆盖网结点上。由于覆盖网结点即是服务，因此这些信息实际是本地存储的。

因此，我们可以认为SZ-CC是没有服务注册和发现过程的。尽管在调用服务时需要查询语义树，但查询的是预置的、描述服务功能的语义信息，这些信息基本不发生变化，维护代价极低。

没有注册和发现过程、本地化的服务信息存储是本模型区别于传统集中式、分布式、P2P注册发现机制的核心特点，避免了集中式注册发现的单点失效、负载均衡问题，也避免了分布式、P2P注册发现系统中结点扰动带来的服务注册信息备份、同步问题，精简了系统结构，大大降低了网络负载。

同时，如上文所述，这种结构还是非常灵活的。下面探讨基于此衍生出的多种灵活、实用的调用方式。

2.2.1 同步调用和异步调用

SZ-CC中存在同步调用和异步调用两种服务调用形式。

上文描述的服务调用实际上是一种异步调用，即调用者将服务请求交给覆盖网某结点(称为接入点)后，无需等待调用结果即可离开；SKad将请求从接入点路由到一个或多个目标服务并进行调用，调用完毕后，SKad持有调用结果并以某种方式(方法回调、邮件通知)通知调用者。显然，SKad的路由算法天然支持这种调用模式。

同步调用是一种传统的调用模式，在SZ-CC中其过程为：结点(包括接入点)查得目标服务或是下一跳结点列表后，并不进行调用或转发请求，而是将目标服务或下一跳结点列表返回给调用者，由调用者实施调用或发起查询请求。要想实现这种模式，需要对SKad的路由算法作简单改造，将1.a.i和5.a改成返回相应的返回语句。在SZ-CC中，对请求的 *cons* 项进行扩展，加入了调用类型选项，路由算法依据此选项自动选择何种调用模式。

2.2.2 定点调用和模糊调用

定点调用是传统的调用模式，指的是调用者无二义性地指定单一目标进行调用。而模糊调用指的是调用请求中没有明确声明调用目标，覆盖网根据请求中的其他信息动态地选择调用目标。

在SKad中，模糊调用是天然的，因为调用所依赖的目标服务 *id* 描述的是服务操作的功能类别、所在服务名，而具有同样 *id* 的服务可以有多个(同名服务或服务的不同副本)。

为使调用更加"模糊"，对目标服务 *id* 的表示形式和匹配度的计算方法进行改进，采用正则表达式的形式定义请求路径 $id_r=(r_1, r_2, \cdots, r_M)$ 中的各组成元素 r_i，改造匹配度计算方法如下：

$$d(id_x, id_y)=\sum_{i=0}^{M} d_i \times 10^i$$

$$d_i=\begin{cases} 10, & match(x_i, y_i)=\text{true} \\ \left\lceil \left| \dfrac{\eta(x_i)-\eta(y_i)}{\lambda} \right| \right\rceil, & match(x_i, y_i)=\text{false} \end{cases}$$

式中，$match(x_i, y_i)$ 是按正则表达式规则来计算 x_i 和 y_i 是否相等。

按照这种方法，可以很方便地声明一个范围，在这个范围内的所有服务都是可被调用的服务。以SZ-CC的语义树为例，类似于 *id*={SCF，应用，*，*含错*}的声明将使得SKad将请求转发到所有隶属于"SCF/应用"类别下且名字包含"含错"字眼的服务，如果其中的某些服务满足请求的 *operation*、*cons* 项需求，则SKad将对这些服务进行调用。

模糊调用方式很可能得到若干不合适的目标服务，为使得这种调用具有更高的准确性，可在请求 *cons* 项中添加条件，对目标服务加以限定。

而要想实现定点调用，只需在请求 *cons* 项中声明一个确定条件即可。此条件可以是服务地址，如 *endpoint*=http://202.196.3.99/scf/services/ABCService，也可以声明一个在服务 *properties* 中预先定义的一个唯一的标识串，如 *id*=102-jhj-dksk-352-a。

2.3 动态调度

利用第2.2节所述的模糊调用的思想，SKad可以很方便地实现动态调用，即在路由过程中完成对目标服务的选择，其过程如下：

1)调用者发出请求，请求 *id* 项采用正则表达式的形式声明了候选服务集合，*cons* 项中描述了调度条件(如结点数、CPU类型、内存大小等)；利用第2.2节所述的模糊调用方法，SKad将此请求路由到多个满足需求的服务操作上。在这个过程中，不满足 *cons* 的服务将被过滤掉。

2)服务接收到请求后，与调用者或者某个托管服务通信请求调度，请求中携带服务的 *properties* 信息。

3)调用者或托管服务接收到若干调度请求后，根据调用目标的 *properties* 信息选择一个或多个最优的服务完成调用。

相对于传统的集中式调度模型，SKad有以下优点：

1)在路由过程中过滤掉不满足条件的服务，精简了调度对象集合。

2)满足条件的目标服务主动请求调用者调度，请求中携带目标服务的 *properties* 信息，调用者或托管服务对这些信息评估选取若干最优服务。显然，SKad中用于调度决策的信息是触发式的(只在调用前获取)、实时的，与传统调度模型周期采集、存储服务和资源信息以进行调度决策的机制相比，大

大降低了系统负载，提高了调度的准确性。

2.4 并行计算

对于非通信密集型且可以利用参数或数据进行任务分割的课题，可以利用 SKad 进行任务分解，从而实现并行计算，这在高性能计算、多播通信和云存储等领域具有广泛的应用需求。

其基本思路为：

1)调用者发出计算请求，请求 *id* 项采用正则表达式定义了参与计算的服务范围，*cons* 项中声明了计算任务的特征和需求。利用第 2.2 节所述的模糊调用方法，SKad 将此请求路由到多个满足需求的计算服务上。

2)计算服务接收到请求后，与调用者或者某个托管服务通信请求分配任务。

3)调用者或托管服务接收到若干分配任务请求后，对任务进行分割，将分割后的任务交给计算服务。

4)计算服务开始计算，将中间结果或最终结果回传给调用者或托管服务。

5)调用者或托管服务接受结果并合并，完成计算。

上述过程中，持有任务、分解任务、收集结果等操作可以由调用者(客户端)来完成，也可以交由某个能够完成上述任务的托管服务来完成。

2.5 模糊搜索

传统的结构化覆盖网络无法处理模糊搜索问题。实际应用中，一般的解决方法有两种：一是设置足够多的 key，多到足以涵盖用户搜索所用到的各种关键字；二是架设服务器集中存储 key 及其说明信息，用户首先访问此服务器，利用 key 的说明信息搜索满足需要的 key，得到 key 后再交由 P2P 网络查询目标结点。第一种方法显然不太现实，且势必大幅增加路由负担；第二种方法中，集中存储 key 的服务器势必成为性能瓶颈，而且某些应用场景中还不可避免地涉及到版权保护问题。

而 SKad 则可以方便、便宜地实现模糊搜索，且不涉及版权问题。其思路是：

1)删除语义树中的叶子结点，保留其非叶子结点(即描述类别的结点)来建立搜索树，为树中的结点增加描述信息。

2)以某种方式将此语义树提供给调用者(用户)。可采用集中式服务器存储此搜索树，也可将此搜索树保存在一个文件中，为此文件赋予一个明确、固定的键值投放到 P2P 网络中。

3)调用者可以首先获取搜索树，利用类型描述信息查得一个或多个满足需求的搜索树路径，并按第 2.2 节所述的方法形成若干正则表达式形式的 *id*，然后使用这些 *id* 向 SKad 发起调用请求，利用 SKad 的模糊调用机制获取若干满足需求的服务。为使得查询结果更加精确，调用者可在请求 *cons* 项中声明若干关键字，SKad 路由算法在判断是否匹配目标服务(isSatisfied 方法)以及选择下一跳结点(isSuited 方法)时，根据这些关键字过滤不符合要求的服务，这种过滤可以是模糊的(如利用字符串的包含关系判断关键字是否匹配服务 *properties* 的某项属性)。

可以将 SKad 应用到音乐、视频、文件分享等应用中，避免版权问题。以音乐分享应用为例，可采用集中式服务器存放搜索树，由于树中存放的是音乐类型及其描述，并不涉及具体的音乐文件，因此不违反版权要求。甚至可以将此搜索树以实体的形式存放在 SKad 网络中，由于音乐类型及其描述并不经常变动，因此在 P2P 中维护此搜索树文件的代价极低。

结束语 本文提出了一种新的 P2P 服务组织模型，其关键特征在于：决定网络拓扑和路由的结点标识并非传统的二进制码，而是一个能够描述结点对应服务功能的语义串，而该语义串来源于一个预先建立的、针对应用中各服务功能归纳而成的语义树。这种特征导致功能相近的服务在覆盖网络拓扑中具有更近的距离，网络中的各服务按语义树对应的功能域形成层次状的聚集关系。这使得位于底层的覆盖网路由与上层的应用具有天然的关联，能够基于这种特征实现多种灵活有用的应用形式。本文也对部分已经实现的应用形式进行了描述。

本文提出的模型具有很高的通用性，不仅仅可以用在服务组织方面，还可以用到其他需要结构化覆盖网的领域，如文件存储、分享网络等。

今后，我们将其搭建在 SZ-CC 上进行测试，重点研究语义树和匹配算法对网络拓扑和网络负载的的影响。

参 考 文 献

[1] Advancing open standards for the information societ(OASIS). UDDI Spec Technical Committee Draft Version 3.0.2[S]. http://uddi.org/pubs/uddi_v3.htm

[2] Maymounkov P, Mazieres D. Kademlia: A peer-to-peer information system based on the XOR metric[C]// Proc. of the 1st Int'l Workshop on Peer-to-Peer Systems (IPTPS 2002). Berlin: Springer-Verlag, 2002: 53-65

[3] Wang CZ, Yang N, Chen HW. Improving Lookup Performance Based on Kademlia[C]// Proc. of the Second International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC 2010). Hubei, 2010: 446-449

[4] Stevens T, Wauters T, Develder C, et al. Analysis of an anycast based overlay system for scalable service discovery and execution[J]. Computer Networks, 2010, 54(1): 97-111

[5] Pirrò G, Trunfio P, Talia D, et al. ERGOT: A Semantic-Based System for Service Discovery in Distributed Infrastructures[C]// Proc. of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID 2010). Melbourne. 2010: 263-272

[6] Banaei-Kashani F, Chen C-C. WSPDS: Web Services Peer-to-peer Discovery Service[C]// Proc. of the International Conference on Internet Computing (ICOMP 2004). Las Vegas, 2004: 733-743

[7] Guo D K, Ren Y, Chen H H, et al. A QoS-guaranteed and distributed model for Web service discovery[J]. Journal of Software, 2006, 17(11): 2324-2334

[8] Liu Z Z, Wang H M, Zhou B. A two layered P2P model for semantic service discovery[J]. Journal of Software, 2007, 18(8): 1922-1932

[9] Wu W M, Wu Y J, Zhao W Y. Chord-based Semantic Web Service Discovery[J]. Acta Electronica Sinica, 2007, 35(B12): 152-155

(下转第 118 页)

均值进行度量。网络系统节点数为 50，分布区域为 $1000m^2$，迭代次数为 30，权值值域为[0，6，1.0]，粒子数在每 10 次仿真中递减，分别取 30、25、20。实验数据如表 2、表 3 所列。

表 2　3 种算法的计算效果对比表

节点最大能量	MIP 算法		MDPSO 算法		本文算法	
	平均能耗	平均时耗/ms	平均能耗	平均时耗/s	平均能耗	平均时耗/s
250	392095.16	0.45	357327.15	0.21	324875.03	0.18
300	400769.08	0.3	362011.04	0.22	339823.71	0.19
350	393845.72	0.37	359141.87	0.22	328172.23	0.19
400	398793.62	0.42	361928.57	0.22	338232.87	0.19
450	401212.08	0.42	364782.39	0.22	339723.25	0.19
500	409938.02	0.53	372381.89	0.22	349233.27	0.19
600	412003.21	0.32	367321.72	0.22	329874.34	0.19
700	405344.73	0.43	367327.78	0.22	338764.83	0.19
800	396901.75	0.41	360517.97	0.22	338794.23	0.19
1000	403447.58	0.48	367786.46	0.22	339973.82	0.19

表 3　3 种算法的计算效果对比表

节点最大能量	MIP 算法		MDPSO 算法		本文算法	
	平均能耗	平均时耗/ms	平均能耗	平均时耗/s	平均能耗	平均时耗/s
200	108684.62	4.01	100558.43	1.60	99776.82	1.21
250	110694.67	3.48	101867.10	1.60	99983.74	1.22
300	108565.93	3.72	99793.46	1.61	98293.22	1.22
350	108959.22	3.82	101256.82	1.61	99939.29	1.22
400	110649.82	3.77	102124.34	1.61	99873.98	1.22
450	109466.72	3.92	100520.07	1.60	99923.87	1.21
550	108923.65	3.76	100882.76	1.61	99843.45	1.22
650	108466.23	3.46	99903.29	1.61	98329.32	1.22
750	107687.62	3.68	99035.85	1.61	98231.28	1.21
950	105827.73	3.89	102360.27	1.62	100217.62	1.22

表 2 为全向天线模式下 3 种算法在不同能耗门限时的算法执行效率；表 3 为在有向天线模式下，最小波束宽度为 90 时 3 种算法在不同能耗门限下的算法执行效率。实验证明，在不同能耗门限下，本文算法具有较好的最优能耗多播树构造效率。

结束语　Ad hoc 网络中最优节点路由路径构造是节点能耗优化的重要课题。本文通过对全向天线和有向天线模式下节点能耗模型进行分析，针对特定网络系统中最优能耗多播树的动态生成问题进行研究，提出了基于最优能耗多播树构造的 Ad hoc 网络节点路由算法。其通过对粒子群算法基于粒距聚类度和粒子信息熵计算的改进，提升了最优能耗多播树生成的搜索能力。实验仿真表明，该算法能够有效地降低中继节点选取对最优能耗多播树构造的影响，提升了算法的鲁棒性，降低了算法的执行时间和能耗。

参 考 文 献

[1] Wieselthier J E, Nguyen G D, Ephremides A. On the construction of energy-efficient broadcast and multicast trees in wireless networks [C]// Proceedings of IEEE INFOCOM'. Tel Aviv, Israel, 2000: 585-594

[2] Montemanni R, Gambardella L M, Das A K. The minimum power broadcast problem in wireless networks: a simulated annealing approach [C] // Proceedings of the 2005 IEEE Wireless Communications and Networking Conference. New Orleans, LA, USA, 2005: 2057-2062

[3] Hernandez H, Blum C. Energy-efficient multicasting in wireless ad-hoc networks: an ant colony optimization approach[C]// Proceedings of the 2008 IEEE International Symposium on Wireless Communication Systems. Reykjavik, Iceland, 2008: 667-671

[4] Das A K, Marks R J, El-Sharkawi M, et al. R-shrink: a heuristic for improving minimum power broadcast trees in wireless networks[C]// Proceedings of IEEE GLOBECOM'03. San Francisco, A, USA, 2003: 523-527

[5] Zhong W L, Huang J, Zhang J. A novel particle swarm optimization for the Steiner tree problem in graphs[C]// Proceedings of the 2008 IEEE Congress on Evolutionary Computation. Hong Kong, China, 2008: 2460-2467

[6] 黄泽霞，俞攸红，黄德才. 惯性权自适应调整的量子粒子群优化算法[J]. 上海交通大学学报，2012，2：228-232

[7] EBERHART R. Fuzzy adaptive particle swarm optimization[C]// Proceedings of 2001 IEEE International Conference on Evolutionary Computation. San Francisco: IEEE Press, 2001: 101-106

[8] 阳春华，谷丽姗，桂卫华. 自适应变异的粒子群优化算法[J]. 计算机工程，2008，16：188-190

[9] Wieselthier J E, Nguyen G D, Ephremides A. Energy-aware wireless networking with directional antennas: the case of session-based broadcasting and multicasting[J]. IEEE Transactions on Mobile Computing, 2002, 1(3): 176-191

[10] 朱晓建，沈军. 基于粒子群优化的 ad hoc 网络最小能耗多播路由算法[J]. 通信学报，2012，33(3)：52-58

[11] Kennedy J, Eberhart R. A discrete binary version of the particle swarm algorithm[C]// Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Orlando, FL, USA, 1997: 4104-4108

[12] 袁辉勇，阙清贤，羊四清. 传感器网络中基于能耗均衡的节点优化部署[J]. 计算机仿真，2010，27(8)：100-102

[13] 杨春德，邓超. 一种动态的时延约束费用优化多播路由算法[J]. 重庆邮电大学学报：自然科学版，2011，23(1)：96-100

(上接第 82 页)

[10] He Q, Yan J, Yang Y, et al. Chord4S: a P2P-based decentralised service discovery approach[C]// Proc. of the 2008 IEEE International Conference on Services Computing (SCC 2008). Salt Lake City, 2008: 221-228

[11] Sioutas S, Sakkopoulos E, Makris C, et al. Dynamic Web Service discovery architecture based on a novel peer based overlay network[J]. Journal of Systems and Software, 2009, 82(5): 809-824

[12] Skoutas D, Sacharidis D, Kantere V, et al. Efficient Semantic Web Service Discovery in Centralized and P2P Environments[C]// Proc. of the 7th International Semantic Web Conference (ISWC 2008). Karlsruhe: Springer Berlin, 2008: 583-598

[13] Zhang Y, Huang H, Yang D, et al. Bring QoS to P2P-based semantic service discovery for the Universal Network[J]. Personal and Ubiquitous Computing, 2009, 13(7): 471-477

[14] Di Stefano A, Morana G, Zito D. A P2P strategy for QoS discovery and SLA negotiation in Grid environment[J]. Future Generation Computer Systems, 2009, 25(8): 862-875

[15] Zhou J, Dou W. A QoS-Aware Service Selection Approach on P2P Network for Dynamic Cross-Organizational Workflow Development[C]// Proc. of the International Conference on Web Information Systems and Mining (WISM 2009). Shanghai: Springer-Verlag Berlin, 2009: 289-298

[16] 郭得科，任彦，陈洪辉，等. 一种 QoS 有保障的 Web 服务分布式发现模型[J]. 软件学报，2006，17(11)：2324-2334

[17] 刘志忠，王怀民，周斌. 一种双层 P2P 结构的语义服务发现模型[J]. 软件学报，2007，18(8)：1922-1932

[18] 吴万明，吴毅坚，赵文耘. 基于 Chord 网的语义 Web Service 发现[J]. 电子学报，2007，35(B12)：152-155