RFID 信息服务网络中支持复合订阅的路由算法研究

刘殿兴[1,3],赵 文[2,3],李信鹏[1,3],冯志明[4],张世琨[2,3],王立福[2,3]
(1.北京大学信息科学技术学院,北京 100871; 2.北京大学软件工程国家工程研究中心,北京 100871;
3.北京大学信息科学技术学院软件研究所高可信软件技术教育部重点实验室,北京 100871;
4.北京团市委信息中心,北京 100871)

摘 要: 在 RFID 信息服务网络中,设计了一种支持复合订阅的可靠的路由算法:在订阅转发阶段,我们将每一个复合订阅按照其语法结构进行分解,并将每个复合订阅成份分配给多个 RFID 信息服务(构成一个复合订阅存储单元)去维护,而事件则按照订阅分解的反方向和匹配的结果进行转发与合并.由于每个复合订阅存在多个副本,因而会显著减少因某个副本失效而产生的事件丢失.本文也给出了复合订阅存储单元内部多个复合订阅副本一致性的保持方法,以及核心信息服务的选举方法.实验结果表明,该路由算法有较高的容错性,性能可以满足实际应用的需要.

关键词: 无线射频识别 (RFID); 复合订阅; 发布/订阅; 路由; Kademlia
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2010) 2A-033-08

Research on Routing Algorithm Supporting Composite Subscriptions in RFID Information Service Network

LIU Dian-xing[1,3], ZHAO Wen[2,3], LI Xin-peng[1,3], FENG Zhi-ming[4], ZHANG Shi-kun[2,3], WANG Li-fu[2,3]
(1. *School of Electronics Engineering and Computer Science, Peking University, Beijing* 100871, *China*;
2. *National Engineering Research Center for Software Engineering, Peking University, Beijing* 100871, *China*;
3. *Key Laboratory of High Confidence Software Technologies (Ministry of Education), School of Electronics Engineeringand Computer Science, Peking University, Beijing* 100871, *China*;
4. *Information Center of China Communist Youth League Beijing Committee, Beijing* 100083, *China*)

Abstract: In RFID information service networks, an innovative routing algorithm supporting composite subscriptions and taking system reliability into consideration is designed. In subscription forwarding, we decompose each composite subscription and assign each of its composite elements to multiple RFID information services to maintain, while events are forwarded along the reverse path of the subscription factorizing direction based on the matching work. In order to keep the consistency of multiple copies of a composite subscription and to ensure the correctness of message forwarding, we import heartbeat messages. The simulation results show that the routing algorithm has strong fault-tolerance and its performance can satisfy the need of application.

Key words: radio frequency identification(RFID); composite subscription; publish/subscribe; routing; Kademlia

1 引言

RFID 是一种高效的非接触自动识别技术,到目前为止,已经有了很多 RFID 闭环应用,而开环应用却很少.其原因是由于 RFID 开环应用需要一套可靠高效的公共服务基础设施和信息共享机制,而这些目前还没有建立起来.RFID 信息服务[1]作为基础设施的重要部分负责捕获和保存本地企业产生的业务事件,并向上层应用提供标准的查询接口.在 RFID 开环应用中,每个企业维护一个或多个 RFID 信息服务,众多企业的 RFID 信息服务就构成了 RFID 信息服务网络(RFID Information Service Network,简称 RFID-ISN),它们协同工作,共同为用户提供决策信息.由于每个 RFID 信息服务可以按照发布/订阅的方式同用户进行交互,因而该网络就成为一个巨大的分布式发布/订阅系统,而每个 RFID 信息服务(简称信息服务)扮演事件代理的角色.

收稿日期:2009-11-13;修回日期:2010-01-10
基金项目:国家高技术研究发展计划(863)(No.2006AA04A119,No.2006AA04A121);国家重点基础研究发展计划(973)(No.2009CB320706);国家自然科学基金(No.60803014)

在 RFID 开环应用中,每个企业可能与多个企业之间有业务往来,企业通常需要向 RFID-ISN 发出复合订阅来表达自己的业务需求.以物流应用场景为例,企业 1,企业 2、企业 3 向装配商供应零件,其中企业 1 和企业 2 供应同一种零件,装配商的管理者可能有如下的业务需求:"如果企业 3 出货,并且企业 1 和企业 2 有一个出货就通知我".如果我们用 A、B、C 分别来表示企业 1、企业 2 和企业 3 出货的规约,那么装配商管理者的业务需求就可以用复合订阅 $(A|B)\&C$ 来表示.该复合订阅在 RFID-ISN 中按照一定的规则转发,并最终到达对应企业的信息服务(例如 A 到达企业 1 的信息服务).当出货事件产生时(例如企业 1 的信息服务捕获到匹配 A 的出货事件),这些事件就会按照复合订阅的路由信息进行反向转发与合并,并最终转发给装配商的管理者.如何保证事件沿着正确,可靠的路径转发给订阅者是 RFID-ISN 中路由算法所要解决的主要问题.

2 相关研究介绍

到目前为止,对多 RFID 信息服务协同工作和路由算法的研究还很少,本节我们从发布/订阅系统研究的角度对支持复合订阅的路由算法进行介绍.支持复合订阅的路由算法大致可以分为两类:一类是基于静态网络拓扑的路由,一种是基于 P2P 的路由.

SIENA[2]是基于静态网络拓扑路由的代表,在这种路由策略中,事件代理从应用中接收原子事件,或是接收其他事件代理传来的事件序列,并将这些事件按照复合订阅进行聚合和转发.在订阅转发的过程中,复合订阅尽可能不分解地向距离事件发布者最近的地方转发,从而在事件转发阶段,子事件可以尽可能早的结合为复合事件,从而避免不必要的事件转发.这种方法需要在全网内广播广告信息,因而限制了它的可扩展性.PADRES[3]的路由思想与 SIENA[2]相似.

为了提高系统的可扩展性,一些学者将支持复合订阅的发布/订阅系统构建在 P2P 网络之上,使系统具有分散控制和自组织的特点.文献[4]提出了一种基于 Chord[5]的路由算法,在该算法中,事件代理起到服务容器(service containers)的作用,它们既可以是接收外部事件或用户订阅的本地访问点,也可以是事件检测器(atomic event detector or composite event detector),每个复合订阅和原子订阅都被赋予一个唯一的编号,并被分配到相应的事件代理,分配的策略就是事件代理的 Chord-ID 距离订阅的编号距离最近.在 Hermes[6]支持复合订阅的扩展版本中,原子订阅按照 Pastry[7]算法分配到事件代理网络上,而每个复合订阅作为复合事件检测点被手动静态的部署于事件代理网络的周围,这些复合事件检测点产生的复合事件作为新的原子事件被重新发送回事件代理网络中做进一步的匹配.文献[4,6]两种算法都有一个共同的问题,就是一个事件代理的失效会导致部分订阅的失效,从而导致事件丢失.

借鉴了上述研究的路由思想,我们提出一种基于 Kademlia[8]的 RFID-ISN 上支持复合订阅的路由算法.在路由算法中,每个复合订阅和复合订阅成份被分配给多个信息服务去维护,而每个原子订阅则根据其地址信息分配给对应的信息服务.这种方法可以显著降低由于信息服务失效而引起的事件丢失.

3 算法的主要思路

本文提出的路由算法对网络拓扑没有特殊要求,只要信息服务之间能构成一个连通图即可.算法的基本思路如图 1 所示.图中每个圆圈代表一个信息服务,圆圈中的数字是唯一标识它的 ID.

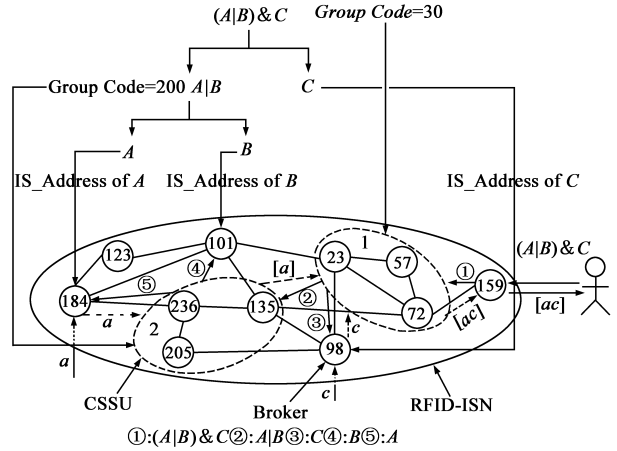


图1 算法主要思路.

当用户向 RFID-ISN 中发出复合订阅时,系统访问点会接收该订阅,并计算其分组码,然后根据分组码将该复合订阅分配给多个事件代理去维护(例如图 1 中 $(A|B)\&C$ 按照箭头①方向分配给虚线椭圆 1 中的三个信息服务),然后虚线椭圆 1 中的某个信息服务根据复合订阅的语法结构将其分解(例如 $(A|B)\&C$ 将被分解为"$A|B$"和"C"),其中的复合订阅成份(例如"$A|B$")将会按照同样的方式进行转发(例如"$A|B$"按照箭头②方向分配给虚线椭圆 2 中的信息服务),而原子订阅将会按照其信息服务地址分配给对应的信息服务(如箭头③,④,⑤所示).

当原子事件产生时,(如图中"a","c"所示),它们按照订阅分解的反方向以及匹配结果进行转发及合并(如图中虚线箭头所示),最终复合事件(例如"[ac]")将被转发给订阅者.

从网络层次的角度看,RFID-ISN 可以分为三个层次[9],如图 2 所示.P2P 层的功能是将信息服务组织成

一个 P2P 网络,事件通知层负责事件和订阅的转发,应用层是发布和接收事件的应用系统.本文的路由算法位于事件通知层.

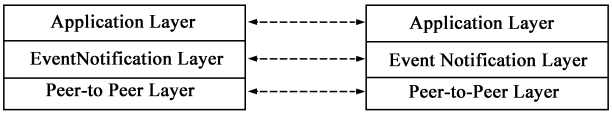


图2　RFID-ISN的层次结构

事件通知层通过"操作"与另外两个层次进行交互.一般来说,该层从应用层接收订阅和事件,并将事件通知推送给应用层的订阅者.

在 P2P 层,我们采用 Kademlia 算法.下面两个操作是 P2P 层提供的接口.

■ FIND_NODE(*GroupCode*):返回 ID 距离 *GroupCode* 最近的 α 个信息服务(ID 之间的距离可以通过 ID 之间的异或运算来得到).α 的值可以根据启发规则得到,例如 α 个信息服务同时不在线的概率趋于 0(图 1 中我们选择 α 为 3).

■ SEND(*dest*, *msg*):将消息"*msg*"发送给信息服务 *dest*,如果发送成功返回 1,否则返回 0.该操作与 Kademlia 中的 STORE 相似.

需要注意的是,一旦为信息服务分配 ID,ID 就不能再变,否则会产生事件无法转发给订阅者的错误.

4　基于 Kademlia 网络的路由算法

4.1　订阅的数据结构与存储

订阅的数据结构如图 3 所示,其中 *ASub* 和 *CSub* 分别表示原子订阅和复合订阅.每一个原子订阅都含有一个信息服务地址 IS_Address,该地址是原子订阅要发送到的信息服务地址;所有订阅都包含一个 *source*,用于保存信息服务的 ID,表示该订阅从哪个信息服务发来,用于后续事件的反向路由;*conditions* 存储了订阅的条件;*flag* 是订阅和退订的标志.

```
Class ASub {                    Class CSub {
    string conditions;              string conditions;
    IP IS_Address;                  ISID source;
    ISID source;                    bool flag;
    bool flag;                  }
}
```

图 3　订阅的数据结构

为了支持订阅和事件的转发,每一个信息服务需要维护从客户和邻居转发来的订阅信息,为了便于管理,我们将复合订阅分成若干组,并由一个复合订阅分组表(简称 CSGT)来维护,位于同一组的复合订阅拥有相同的分组码,每个 CSGT 由其分组码来唯一标识.由于每个复合订阅按照其分组码被分配给 α 个信息服务,因而每个 CSGT 在 α 个信息服务中各有一份拷贝.

■ 分组码(*GroupCode*)

复合订阅的转发和维护是根据其分组码进行的,我们定义 *GC* 为全体分组码的集合,如下所示:

定义 1　令 *CS* 为全体复合订阅的集合,$GC = \{GroupCode \mid \forall\ CSub \in CS, GroupCode = 2^n \times Hash(CSub)/groupNum\}$.我们用 *GC*(*CSub*)表示复合订阅 *CSub* 的分组码.

"n"表示信息服务 ID 的二进制码长度;"*Hash*(*CSub*)"是一个哈希表为[1, *groupNum*]的哈希函数,"*groupNum*"为系统中的分组数量.考虑到负载均衡的因素,*groupNum* 应该为系统中信息服务数量的倍数.*groupNum* 越大,负载平衡性越好,而维护 CSGT 多个副本的开销也会越大.定义 1 说明,只要给出一个复合订阅 *CSub*,我们就可以通过 $2^n \times Hash(CSub)/groupNum$ 计算出它的分组码 *GC*(*CSub*).

■ 复合订阅存储单元(CSSU)

为了降低因信息服务失效而产生的事件丢失,每个复合订阅根据其分组码被分配给 α 个信息服务去维护,这 α 个信息服务构成一个 CSSU,并由这个分组码唯一标识,CSSU 的定义如下:

定义 2　令 *B* 为系统中所有信息服务的集合.CSSU(*GroupCode*)称为分组码为 *GroupCode* 的复合订阅存储单元,当且仅当:

(ⅰ)CSSU(*GroupCode*) $\subset B$,

(ⅱ)|CSSU(*GroupCode*)| = α,

(ⅲ) $\forall\ n \in$ CSSU(*GroupCode*) $\forall\ m \notin$ CSSU(*GroupCode*),$(n.ID \oplus GoupCode) < (m.ID \oplus GroupCode)$.

FIND_NODE(*GroupCode*)就是用来返回 CSSU(*GroupCode*).

■ 复合订阅分组表(CSGT)

每个 CSGT 维护一组复合订阅,并且由 *GroupCode* 唯一标识,每个 CSGT 在对应的 CSSU(*GroupCode*)中有 α 份拷贝.为了保证事件和订阅转发的正确性,我们需要按照一定的规则选择出一个信息服务作为核心信息服务,由它来完成事件和订阅的转发以及匹配工作(例如我们可以选择在 CSSU 中 ID 最小的信息服务为核心信息服务).

我们在 CSGT 中加入一个 *InCoreBroker* 标识,来表示该份 CSGT 拷贝是否位于核心信息服务上(4.2,4.3,4.4 节会详细给出该标识的用途,在 4.5 节会给出核心信息服务的选举过程).*GroupCode* 和 *InCoreBroker* 构成了 CSGT 的表头部分,表体中的每一项是一个五元组:

〈*CSub*, *Subscribers*, *Elements*, *LogicTimer*, *TimeExpired*〉

CSub 保存接收到的复合订阅,*Subscribers* 保存订阅者地址,*Elements* 保存由复合订阅分解得到的订阅成份,这些订阅成份通过函数 decomposeSub(*CSub*)来得

到.为了维护复合订阅 α 个拷贝的一致性,用 *LogicTimer* 来标识复合订阅的更新状态(只要复合订阅更新,对应的 *LogicTimer* 就加 1).在退订操作中,如果只是简单的将对应的表项删除可能会引起不一致(例如 CSSU 的某些信息服务没有接收到退订),因而我们加入 *TimeExpired* 标识,当 *Subscribers* 为空时,该标识开始计数,当到达一个阈值后,系统才将对应的表项删除(因为这期间心跳消息会将 *TimeExpired* 信息传递给所有 CSSU 中的信息服务,阈值越大系统越稳定).

定义 3 令 *CS* 为全体复合订阅的集合,二元组〈*head*, *body*〉叫做 CSGT,当且仅当:

(ⅰ)head 是一个二元组〈*GroupCode*, *InCoreBroker*〉,满足 *GroupCode*∈*GC* ∧ *InCoreBroker*∈Boolean,

(ⅱ)body 是五元组〈*CSub*, *Subscribers*, *Elements*, *LogicTimer*, *TimeExpired*〉的集合,满足 *CSub* ∈ *CS* ∧ *GC*(*CSub*) = *GroupCode* ∧ *Elements* = decomposeSub(*CSub*) ∧ *LogicTimer*, *TimeExpired*∈unsignedInt

图 4 是一个 CSGT 的举例.需要说明的是,在某个 *TimeExpired* 开始计数的过程中,当其对应的复合订阅被重新订阅的时候,*LogicTimer* 将重新开始增长,而 *TimeExpired* 将被清 0.

head	GroupCode(10500)		InCoreBroker(true)		
	CSub	Subscribers	Elements	LogicTimer	TimeExpired
body	*B*|*C*	{tcp://192.168.100.59:3035 tcp://192.168.100.19:3035}	{*B*,*C*}	2	0
	D&*E*	{tcp://192.168.100.37:3035 tcp://192.168.100.106:3035}	{*D*,*E*}	2	0
	F&*H*		{*F*,*H*}	5	2

图4 CSGT举例

■ 原子订阅表(AST)

每个信息服务除了维护一个或多个 CSGT,其内部还维护一个原子订阅表,该表仅用于维护发送到本地的原子订阅,其每一个表项是一个二元组:

〈*ASub*, *Subscribers*〉

ASub 保存接收到的原子订阅.

4.2 订阅转发

订阅转发的基本思路如图 5 所示,虚线表项表示接收到订阅后的插入项.

当一个信息服务接收到一个订阅时,首先要判断是否为一个原子订阅.如果是一个原子订阅,那么该信息服务要判断其 IP 地址是否与订阅的 IS_Address 相同,如果相同则直接插入 AST,否则根据 IS_Address 进行转发.

如果是复合订阅,该信息服务将会计算出其分组码,并根据该分组码找到对应的 CSGT.如果对应的 CSGT 中存在该复合订阅,那么将订阅者的 ID 插入到 *Subscriber* 中,将 *LogicTimer* 加 1,同时将 *ExpiredTime* 清 0;否则创建一个新表项将订阅插入.如果 CSGT 中的 *InCoreBroker* 为 true(说明该 CSGT 位于核心信息服务上),则该信息服务还要负责将复合订阅进行分解和转发:分解得到的原子订阅成份按照其 IS_Address 进行转发,分解得到的复合订阅成份按照其分组码进行转发.算法如图 6 所示,一些算法中用到函数功能如下:

■ generateGroupCode(*CSub*):返回复合订阅 *CSub* 的分组码

■ findCSGT(*GroupCode*):根据分组码 *GroupCode* 找到对应的 CSGT

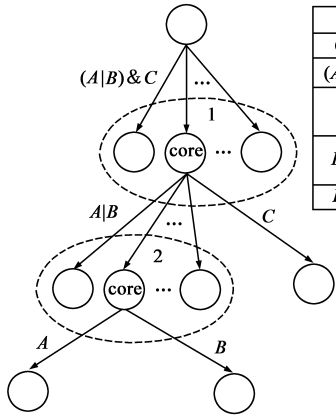
■ findItem(*CSub*):查找复合订阅 *CSub* 在表中对应的表项,如果不存在返回空值

■ decomposeSub(*CSub*):将复合订阅 *CSub* 按照其语法结构进行分解

需要说明的是,为了防止将新提交的订阅转发给已经失效的信息服务,我们按照信息服务 *ID* 由小到大的顺序将复合订阅进行转发,并判断 *SEND* 的返回值,如果对核心信息服务(*ID* 最小)的 *SEND* 操作失效,在后续的 *SEND* 操作中,需要指定对方暂时担任核心信息服务的工作(如订阅的分解和转发),并且这样的信息服务只能指定一个.由于篇幅所限,细节就不再展开.

4.3 退订转发

由于每一个复合订阅由对应 CSSU 中的 α 个信息服务去维护,当一个复合订阅需要退订时,这 α 个信息服务都需要执行退订操作.当一个信息服务接收到退订信息时,它首先判断退订的是否为原子订阅,如果退订的是原子订阅,那么该信息服务将从 AST 中找到对应的表项并删除;否则,计算待退订订阅的分组码,根据分组码找到对应的 CSGT,并从 CSGT 中对应表项的 *Subscribers* 中删除订阅者信息.当 *Subscribers* 为空时,*Ex-*



GroupCode(10500)		InCoreBroker(true)		
CSub	Subscribers	Elements	LogicTimer	TimeExpired
(*A*|*B*)&*C*	{tcp://192.168.100.22:3035}	{*A*|*B*,*C*}	1	0
B|*C*	{tcp://192.168.100.59:3035} tcp://192.168.100.19:3035	{*B*,*C*}	2	0
D&*E*	{tcp://192.168.100.37:3035} tcp://192.168.100.106:3035	{*D*,*E*}	2	0
F&*H*		{*F*,*H*}	5	2

ASub	Subscribers
K	{tcp://192.168.100.35:3035} tcp://192.168.100.104:3035
G	{tcp://192.168.100.20:3035} tcp://192.168.100.49:3035
C	{tcp://192.168.100.57:3035}

图5 订阅转发的基本思路

```
//Subscription 为原子订阅或复合订阅
class CTableItem {//CSGT 中的表项
    Subscription sub;
    HashSet < Subscriber > subscripers;
    HashSet < Subscription > Elecments;
    Integer logicTimer;
    Integer timeExpired;
}
class ATableItem {//AST 中的表项
    Subscription sub;
    HashSet < Subscriber > subscribers;
}
//接收订阅时调用该函数
void recv_sub(Subscription msg){
    if( msg == ATOMIC) {
        if( msg. IS_Address == this. IP) {
            //each broker maintains an atomic subscription table
            ATableItem ti = atomicTable. findItem( msg)
            if( ti < > null)
                ti. subscribers. add( msg. source);
            else
                atomicTable. addItem( < msg, { msg. source} > );
    }
    else forwardMsg( msg);
  }
  else {
    GC groupCode = generateGroupCode( msg);
    CSGT t = findCSGT( groupCode);
    CTableItem ti = t. findItem( msg)
    if( ti < > null) {
        ti. subscribers. add( msg. source);
        ti. timeExpired = 0;
    }
    else {
        HashSet < Subscription > S = decomposeSub( msg);
        t. addItem( < msg, { msg. source}, S, 0, 0 > );
        if( ti. InCoreBroker)
            for each subscription in S {
                subscription. souce = this. ID
                forwardMsg( < subscription, true > );
            }
    }
    ti. logicTimer ++ ;
  }
}
//将 msg 转发出去
void forwardMsg(Subscription msg) {
    if( msg == ATOMIC)
        SEND( msg. IS_Address, msg);
    else {
        GC groupCode = generateGroupCode( msg);
        for each Broker b in CSSU( groupCode)
            SEND( b, msg);
    }
}
```

图 6　订阅转发算法

piredTime 开始计时,当 *ExpiredTime* 大于设定的某个阈值时,就可以将该表项删除,不过在删除表项以前需要先退订 *Elements* 中的订阅成份.为了避免一个订阅成份被多次退订的错误,我们只允许核心信息服务对订阅成份进行退订.

需说明的是,在退订 *Elements* 订阅成份的过程中,需检查订阅成份是否存在于其他表项的 *Elements* 中,如不存在,说明该订阅成份确实不被该信息服务所需要,才可进行退订;否则不能进行退订.算法如图 7 所示.

```
void recv_unsub(Subscription msg) {
    if( msg == ATOMIC) {
        if( msg IS_Address == this. IP) {
            ATableItem ti = atomicTable. findItem( msg)
            if( ti < > null) {
                ti. subcribers. remove( msg. source);
                  if ( ti. subscribers. isEmpty( )) atomic Table. re-
                    moveItem( ti);
            }
        else forwardMsg( msg);
    }
    else {
        GC groupCode = generateGroupCode( msg);
        CSGT t = findCSG( groupCode);
        CTableItem ti = t. findItem( msg);
        ti. subscribers. remove( msg. source);
    if( ti. subscribers. isEmpty( )) {
        if( ti. timeExpired > threshold) {
            if( t. InCoreBroker) {
                for each Subscription ele in ti. Elements
                    if ( ele does not appear in the ti. Elements field
                      of other intems) {
                        ele. source = this. ID;
                        forwardMsg( < ele, false > ); delete( ele);
                    }
            }
            else if( ti. Elements. isEmpty( )) t. removeItem( ti);
        }
        else ti. timeExpired ++ ;
    }
    else ti. logicTimer ++ ;
  }
}
```

图 7　退订转发算法

4.4　事件转发

订阅的转发将会形成一个转发树,而事件则是根据订阅转发的相反方向及匹配结果进行转发.

由于每个复合订阅由对应 CSSU 中的 α 个信息服务来维护,为了保证事件转发的正确性,我只允许核心信息服务来完成匹配和转发工作(匹配工作不是本文的重点,有兴趣的读者可以参看文献[3,4,10]).假设复合订阅 *A&B* 由对应 CSSU 中的 α 个信息服务来维护,匹配 *A* 和 *B* 的原子事件 e_a 和 e_b 已经产生,如果这两个原子事件转发给同一个信息服务,那么该信息服务可以根据复合订阅信息得到正确的复合事件[$e_a e_b$],并进一步转发;如果这两个原子事件转发给不同的信息服务,那么就得不到正确的结果.

通过上述分析,我们给出事件转发的过程如下(原子事件的转发相对简单就不再详细描述):当一个信息服务接收到一个事件时,它会遍历所有 *InCoreBroker* 为 true 的 CSGT,如果该事件与某个表项的 *Elements* 中的订阅成份相匹配,说明该事件是匹配对应 *CSub* 的复合事

件的组成部分,如果能形成匹配 *CSub* 的复合事件,那么就将该复合事件转发给所有的订阅者(在订阅者不在线的情况下,系统需要保存该事件,并周期性的试探发送,直到订阅者在线,或新的核心信息服务选举完毕),否则将该事件记录到缓存中,为以后可能的匹配做准备.算法如图 8 所示.

```
class Event {
   string type;              //标识是原子事件还是复合事件
   Subscription belongTo;    //标识该事件匹配了哪个订阅
   DateTime begin;           //记录事件的开始时间
   DateTime end;             //记录事件的结束时间
}
//根据订阅信息和匹配结果合并及转发事件
void forwardNotification(Event event) {
  if(event.belongTo == mull) {
    for each TableItem ti in atomicTable {
      if(event matches ti.subscription) {
        for each Subscriber subscriber in ti.subscribers
          SEND(ti.source,
            [ATOMIC, ti.subscription, event.begin, event.end]);
      }
    }
  }
      for each CSGT t in current broker {
        if(t.InCoreBroker) {
          for each TableItem ti in t {
            if((event.belongTo[]ti.Elements)) {
              if(ti.subscription is matched) {
                for each subscriber in ti.subscribers {
                  newEvent = new ComEvent();
                  //set the attributes in newEvent. Is is the matching
                    work.
                  SEND(subscriber, newEvent);
                }
                delete component events of newEvent in the buffer
              else
                put event into the buffer corresponding with event.be-
                longTo;
        }
      }
    }
  }
}
```

图 8 事件转发算法

4.5 心跳消息

在 Kademlia 网络中,为了防止无效资源的散播,所有的资源都有一个失效时间.为了使信息服务能够长久的保存资源(CSGT),我们对 Kademlia 算法稍做改动,使信息服务能够定期的发出心跳消息.具体来说,每个信息服务都要定期的向存储同样 CSGT 拷贝的 $\alpha-1$ 个信息服务发出心跳消息,心跳消息的格式如下:

< *ISID*, *GroupCode*, abstract of CSGT >

"*ISID*"是发出心跳消息的信息服务的 ID, *GroupCode* 是对应 CSGT 的分组码,"abstract of CSGT"是一个 128 位的二进制数,表示 CSGT 的摘要,它是对 CSGT 中 *CSub* 和 *Subscribers* 内容进行哈希得到的.

除了上述功能,心跳消息还有以下两个功能:一是核心信息服务的选举,二是 CSSU 中多个 CSGT 副本一致性的保持.

核心信息服务的选举过程如下:当一个信息服务接收到心跳消息时,它会检查 *ISID* 是否小于自己的 ID,如果小于则说明自己不会成为核心信息服务,否则该信息服务向对方发出心跳消息来表明自己的 ID.通过上述过程,所有 CSSU 中的信息服务都将知道哪个信息服务的 ID 最小,也就知道了谁是核心信息服务.选举过程结束后,核心信息服务将会检查对应 *GroupCode* 的 CSGT 中的 *InCoreBroker* 标志是否为真,如果为真则返回;否则更改标志位,然后对表中内容重新进行订阅.篇幅所限就不再展开.

CSSU 内部多个 CSGT 拷贝一致性的维护是为了防止如链路和节点不稳定所造成的错误.过程如下:当信息服务 *A* 接收到 *B* 传来的心跳消息时,*A* 会查找与心跳消息有相同 *GroupCode* 的 CSGT,如果没有查找到,则 *A* 会向 *B* 发出请求,索要对方的 CSGT 并保存;如果找到,那么 *A* 计算该 CSGT 的摘要,并同心跳消息传来的摘要相比,如果相同则返回,否则向 *B* 发出请求索要对方的 CSGT,并根据 *LogicTimer* 同本地 CSGT 进行合并.例如对于一个给定的表项,如果 *B* 中的 *LogicTimer* 大,则说明它是最新的,那么用该表项替换 *A* 中的对应表项.

5 性能测试

本节将对路由算法的容错性和效率进行分析.基于开源项目 JKademlia 实现了一个原型系统,JKademlia 提供了一个网络模拟程序,可以在一台机器上模拟大量的 P2P 节点.测试所用的计算机配置是 Intel Pentium IV CPU,主频 2.0GHz,2GB 内存,操作系统采用 Windows XP Professional,JDK 采用 1.6.0_03 版本.在实验中,模拟节点的数量是 1024 个,每个节点的 ID 是 160 位二进制数.测试中采用的复合订阅具有 3 个复合订阅成份和 4 个原子订阅(例如 $(A|B)\&(C|D)$,3 个复合订阅成份分别是 $(A|B)\&(C|D)$、$A|B$ 和 $C|D$,4 个原子订阅分别是 A、B、C 和 D).由于每个节点每隔 t 时刻(t 通常为 1 小时以上)发出一次心跳消息,远小于订阅和事件的转发频率,加上心跳消息通常很短,因而模拟实验中没有考虑心跳消息的因素.

为了性能比较,也初步实现了另外两种路由:基于 SIENA 的路由(static-topology-based routing)和 Chord-based 路由(基于 OpenChord 来实现).并在相同实验环境和负载情况下对性能做了比较.

为了研究不同路由算法的容错性能,令部分节点失效(以 JKademlia 为例,可以修改节点线程的启动代码,不启动特定 ID 的节点线程,失效节点采用随机选择的方式),然后我们观察复合事件丢失率(CELR).对于

一个给定的复合事件 ce ,CELR 指标计算如下:

CELR(ce) = 对 ce 感兴趣却没有接收到 ce 的节点/所有对 ce 感兴趣的节点.

在讨论问题以前,先给出如下参数:

b 表示系统中节点的数量;

α 表示 CSSU 中节点的数量;

i 表示节点失效率;

m 表示每个复合订阅含有的复合订阅成份的平均数量;

n 表示每个复合订阅含有的原子订阅的平均数量.

在最坏的情况下,所有无效的节点集中在几个 CSSU 中,以至于这些 CSSU 无法正常工作.在这种情况下,CELR 近似为 $mbi/C_b^\alpha + n/b$,而另外两种路由算法的 CELR 近似为 $mi + n/b$.显然 mbi/C_b^α 小于 mi.

图 9 显示了当节点失效率从 0.5% 增加到 10% 时的 CELR.α 值取为 10,x 轴表示节点失效率,y 轴表示 CELR.从图中我们可以看出,当节点失效率较低时 CELR 变化较小,随着节点失效率的增加,变化趋势有所不同,当节点失效率达到 10% 时,本文路由算法的 CELR 仅为 1.2%,远低于其他两种路由算法,说明本文提出的路由算法有较好的容错性能.

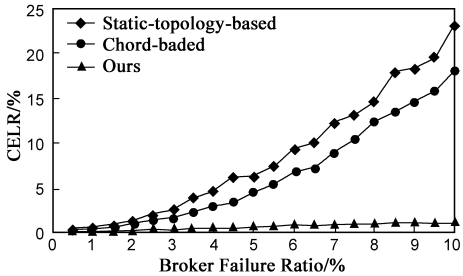


图9　容错性能比较

需要说明的是,Static-topology-based 和 Chord-based 的两条曲线并不重合,并且前者 CELR 的值大于后者.产生这种情况的原因是 Static-topology-based 路由中的节点只能将事件转发给特定的邻居节点,当某个邻居节点失效时,所有需要经过它转发的节点都将失效.而在 Chord-based 路由中,如果一个邻居节点失效,通过 Chord 算法仍然可以通过其他邻居节点转发事件.

在下面的实验中我们研究了不同负载情况下的空间占用情况.假设系统中一共维护 s 个复合订阅,在最坏的情况下,这些复合订阅没有共享的子结构,那么本文提出算法的空间占用是 $s(\alpha m + n)$,而另外两种算法的空间占用是 $s(m + n)$.从上面两式可以看出,本文提出的算法占用了较多的空间,但是在复合订阅之间具有某些共享子结构的情况下(例如($A|B$)&C 和($A|B$)&D 之间就共享 $A|B$),这种情况会有显著改观.

图 10 显示了复合订阅数量从 5000 增长到 5×10^4 时系统空间的占用情况.其中每个复合订阅包含 3 个复合订阅成份和 4 个原子订阅.原子订阅的种类有 100 个.α 值取为 10,x 轴表示复合订阅的数量,y 轴表示用 AST 和 CSGT 中存储订阅的表项来度量的空间占用.图中显示出本文提出算法的空间占用大于另外两者.但随着复合订阅数量的增加,空间消耗的趋势趋于平坦,这是由于后面发出的复合订阅开始共享先前发出复合订阅的子结构.对于本文的算法,这种存储增长变缓的趋势更加明显,这是由于一次子结构的共享可以节省 $\alpha - 1$ 个存储空间.从图中可以看出,当 x 为 5000 时,本文算法的空间消耗是另外两种算法的 5 倍,当 x 增长到 5×10^4 时,本文算法的空间消耗是其他两种算法的 2 倍左右.

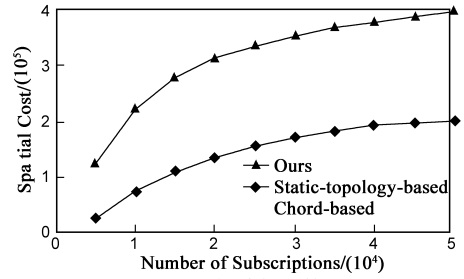


图10　间消耗比较

在实际应用中,订阅的数量相对稳定,而事件的数量是远大于订阅数量的.此外,本实验中我们选择的 α 值较大,当 α 为 3 时,系统的可靠性就比较令人满意.因而本算法的存储性能在实际中还是可以接受的.

在下面的实验中,我们通过一个复合订阅的发布所导致的消息数量来研究订阅转发的效率.假设每个复合订阅与系统中现存的复合订阅相等的概率为 p,每个节点平均维护 k 个复合订阅.对于本文的算法,一个复合订阅的发布所导致的消息数量的最小值可以表示为 $m(pk + \alpha(1 - pk)) + n$,而 Chord-based 路由算法可以表示为 $m(1 - pk) + n$,从上述两式可以看出,随着 k 的增长,消息数量将会减少.

图 11 显示了当 k 从 2 增加到 20 时一次复合订阅的发布所导致的消息数量.x 轴表示了 k 值的变化,y 轴表示一次复合订阅的发布所导致的消息数量.从图中可以看出,Static-topology-based路由将会产生更多的

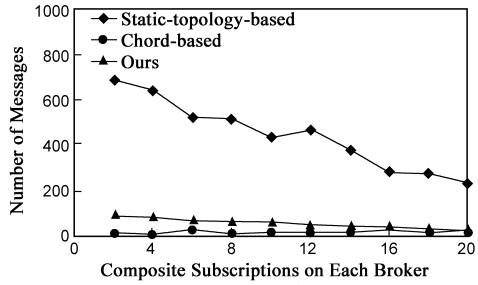


图11　一次复合订阅发布所导致的消息数量比较

消息.这是由于复合订阅分解得到的复合订阅成份和原子订阅成份在初始的时候将在系统中广播,随着每个节点中复合订阅数量的增加,消息的数量会逐渐减少,这是由于存在的原子订阅之间具有某种覆盖关系,可以避免一些消息的转发.显然 Chord-based 路由算法和本文的算法产生的消息数量较少.

在下面的实验中,我们研究了事件转发的效率.图 12 显示了对某个复合事件 *ce* 感兴趣的订阅者数量从 10 增加到 100 时,*ce* 的发布所导致的消息转发数量.*x* 轴表示对某个复合事件 *ce* 感兴趣的订阅者数量,*y* 轴表示导致的消息数量.从图中可以看出,在 Static-topology-based 路由中,一个复合事件的发布将导致较少的消息转发数量,这是由于复合订阅在转发过程中,会尽可能的靠近事件发布者,因而原子事件会较早的结合为复合事件,从而避免了一些不必要的消息转发.Chord-based 路由算法和本文的路由算法在这方面区别较小.

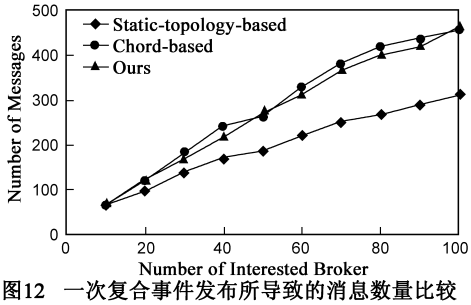


图12 一次复合事件发布所导致的消息数量比较

由于三种算法的订阅转发和事件转发的时间效率都与复合订阅平均拥有的复合订阅成分的数量成正比,因而对时间效率的分析就不再展开.

6 结论

本文提出了一种 RFID-ISN 中支持复合订阅的路由算法.该算法基于 Kademlia 网络,重点考虑了路由的可靠性.它可以显著降低由于信息服务失效而导致的事件丢失,这对于 RFID 开环应用环境是比较有意义的.本算法在 863 项目"RFID 公共服务体系架构设计及应用服务关键技术研究与开发"中得到了实践,显示了较好的效果.在以后的研究中,我们将重点研究如何利用复合订阅之间的覆盖关系来减少空间的消耗.

参考文献:

[1] EPCglobal. EPC Information Services (EPCIS) Version 1.0 Specification[S]. http://www.epcglobalinc.org/standards/epcis/epcis_1_0-standard-20070412.pdf,2007-04.

[2] Carzaniga A, Rosenblum DS, Wolf AL. Design and evaluation of a wide-area event notification service[J]. ACM Trans. on Computer Systems,2001,19(3):332 – 383.

[3] Li GL, Jacobsen A. Composite subscriptions in content-based publish/subscribe systems[A]. Proceedings of the 6th ACM/IFIP/USENIX International Middleware Conference[C]. Grenoble: Springer-Verlag,2005.249 – 269.

[4] Courtenage S, Williams S. The design and implementation of a P2P-based composite event notification system[A]. Proceedings of the 20th International Conference on Advanced Information Networking and Applications[C]. Vienna: IEEE Computer Society Press,2006.701 – 706.

[5] Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for Internet applications[A]. Proceedings of the 2001 Conference on Applications, Technologies, Architectures and Algorithms for Computer Communications[C]. New York: ACM Press, 2001. 149 – 160.

[6] P R Pietzuch, B Shand, J Bacon. Composite event detection as a generic middleware extension[J]. IEEE Network,2004,18(1): 44 – 55.

[7] Rowstron A, Druschel P. Pastry: Scalable, distributed object location, and routing for large-scale peer-to-peer systems[A]. Proeedings of the IFIP/ACM Int'l Middleware Conference [C]. London: Springer-Verlag,2001.329 – 350.

[8] Maymounkov P, Mazières D. Kademlia: A peer-to-peer information system based on the XOR metric[A]. Proceedings of the 1st Int'l Workshop on Peer-to-Peer Systems[C]. Cambridge: Springer-Verlag,2002.53 – 65.

[9] Jinling W, Beihong J, Jing L. Building reliable content-based routing protocol over structured P2P networks[J]. Journal of Software,2006,17(5):1107 – 1114.

[10] Dianxing L. Research on Key Technologies and Applications of Publish/Subscribe Systems[R]. Report of comprehensive examination, Beijing: Peking University, Dec 2008.

作者简介:

刘殿兴 男,1980 年出生,北京大学博士研究生,主要研究方向为软件工程,发布/订阅系统,RFID 相关技术.

赵 文 男,1967 年出生,博士,副研究员,主要研究领域为软件工程、工作流技术和 RFID 相关技术. E-mail: zhaowen@pku.edu.cn