软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(8):2073−2083 [doi: 10.3724/SP.J.1001.2012.04109]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

朝向全局式的声明式传感器网络开发模式*

汪 芳+, 张云勇

(中国联合网络通信集团有限公司研究院,北京 100048)

Towards a Global and Declarative Development Model for Sensor Networking

WANG Fang+, ZHANG Yun-Yong

(China Unicom Research Institute, Beijing 100048, China)

+ Corresponding author: E-mail: wangfang313@chinaunicom.cn

Wang F, Zhang YY. Towards a global and declarative development model for sensor networking. *Journal of Software*, 2012,23(8):2073−2083 (in Chinese). http://www.jos.org.cn/1000-9825/4109.htm

Abstract: This paper proposes a global and declarative development model for sensor networking, within which the global and declarative programs are compiled into distributed programs that can be executed on virtual machines equipped on nodes, so that the developers declare the global functionality of a network.
Key words: sensor network; language; query; declarative; global

摘 要: 探讨一种全局式的声明式传感器网络开发模式,将全局式的声明式网络程序编译为分布式程序,再交给网络节点的虚拟机执行.因此,开发者只需声明网络全局的功能,而无须处理网络的分布式计算过程以及底层细节.
关键词: 传感器网络;语言;查询;声明式;全局式
中图法分类号: TP393 **文献标识码**: A

近年来,随着无线通信技术的发展渗透,以及微电子嵌入式计算设备种类的快速增加,泛在网络得到了一定程度的发展.然而,设备的多样性和不稳定性、网络的动态性以及数据的密集性等特点给泛在网络的发展带来各种挑战.泛在网络开发的一个瓶颈是缺乏编程抽象,这引起了学术界和工业界的广泛重视.

以发展相对比较成熟的传感器网络为例,根据欧盟“传感器网络开发和控制协同嵌入式系统”项目(embedded WiSeNts)研究路线图报告,“现在亟需一种新的高层编程模型,使抽象的层次从以系统为中心的编程提升到以应用为中心的编程.用户应该指定预期的行为,而不是必须去处理系统的细节.特别地,此类模型应支持通过对需要实现的功能的声明式描述来对传感器网络的整体进行编程,而不是针对单个节点的行为进行描述”[1],从而使用户从底层的细节中脱离出来,将注意力集中在高层的应用逻辑上.

在计算机技术领域,声明式编程语言被认为是编程语言未来的发展趋势,可以提供更高层抽象的编程模型[2].回顾数据库领域的发展,逻辑层相对物理层的独立是数据库管理系统的基本原则.利用关系演算提供以数据为中心的应用的抽象模型带来关系数据库管理系统(relational database management system,简称 RDBMS)在技术和商业上的巨大成功[3].另外,作为霍恩子句逻辑的不动点扩展,Datalog 语言被用来描述包含递归运算的数据计算功能[4].这些查询语言为数据查询提供了声明式的方法,使数据的查询独立于数据的物理表示,而让数据库管理系统负责优化和执行.

* 收稿时间: 2011-06-22; 定稿时间: 2011-11-10

近年来,研究人员尝试利用数据库管理系统(database management system,简称DBMS)对传感器网络数据进行管理和查询,以简化传感器数据密集型应用的开发[5,6].递归的 Datalog 语言可以描述图的连通性、可达性,如生成树和传递闭包等经典的图论问题[7,8].因此,研究人员尝试利用 Datalog 语言的扩展来描述生成树和路由等网络计算问题,以提供声明式的网络开发模式[9−11].然而,这些网络计算的声明式程序是分布式的.本文提出一种全局式的声明式网络开发模式,即将全局式的递归的声明式程序编译为分布式程序交给节点上的虚拟机处理.这样,用户只须提供对网络计算问题的全局描述,而将编译后的分布式程序交由网络本身去计算,从而可以更进一步地简化网络计算应用的开发.

本文第1节综述声明式传感器网络开发模式的发展和现状.第2节介绍分布式的声明式网络开发模式.第3节介绍全局式的声明式网络开发模式,描述从全局式程序到分布式程序的编译算法.第4节是本文的结束语.

1 声明式传感器网络开发模式

1.1 传感器网络声明式数据查询

传感器网络通过传感器节点的协同工作,对地理范围内的对象进行感知,采集并处理数据,并将处理结果发布给用户.对用户来说,传感器网络的核心是感知的数据,而不是网络硬件,如何对感知的数据进行管理和处理成为决定传感器网络是否可用的关键[12].近几年来,研究人员尝试将数据库查询语言应用于传感器网络的数据密集型应用.Fung 等人提出,可以将整个网络看成一个数据库,通过声明式查询来实现对传感器网络的数据查询[13].康奈尔大学的 Cougar 系统[5]和加州大学伯克利分校的 TinyDB 系统[6]等均支持用 SQL 编写的传感器数据查询,提供了节能的数据传播和查询的解决方案.在对网络的拓扑结构和节点设备容载的全局认识的基础上,Strivastava 等人设计了分布式查询处理计划[14].基于传感器数据的空间的和时态的特征,声明式方法还被 Jeffery 等人运用于清理网络上的不安全数据[15].

上述的传感器网络数据查询处理系统分为两个层次,即运行在查询节点(例如基站)上的DBMS层和运行在智能终端设备上的 DB 层,如图 1 所示.

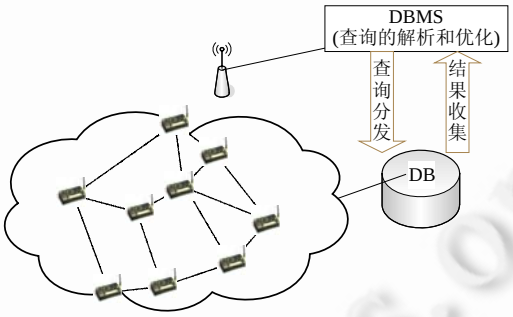

Fig.1 Sensor network data query process system

图 1 传感器网络数据查询处理系统

查询节点主要负责查询的解析和优化,生成分布式处理计划,并将查询以“多跳”路由方式分发到传感器节点上,传感器节点上的数据查询结果同样以“多跳”路由方式返回到查询节点.

传感器智能终端的软件构建在微系统,如 TinyOS[16]上.然而,其面向设备的开发模式需要用户深入地了解设备的硬件配置、操作系统和技术标准,使用户耗费大量的时间和精力处理如存储、并发以及通信协议等底层细节,是一件极其复杂的工作.

对底层细节的定义给应用的开发与维护都带来很大困难,软件模块的复用性也很差,不能很好地满足传感器网络快速部署所需的可扩展性、易开发性和易维护性等要求.

1.2 传感器网络声明式路由查询

在处理数据查询时,传感器网络分发查询和收集结果的过程需要传感器网络根据路由策略建立路由,以保证节点的协同工作.因此,路由发现和维护是传感器网络处理数据查询的基础.传感器网络是无基础设施、多跳结构的,甚至是动态的,其部署环境和条件差异性大.因此,需要采用不同的路由策略,如生成树、Ad Hoc 网络路由策略,或者针对传感器网络特点设计的以数据为中心的、基于位置的、节能的路由策略[17−20]等等.

面向系统的开发模式同样成为传感器路由协议迅速部署的瓶颈.研究人员认为,传感器网络的各种实现技术必须与传感器网络数据管理和处理技术密切结合,融为一体,而不是像目前其他网络设计那样分而为之,这样才能设计实现高效的以数据为中心的传感器网络系统[12].比如,可以将路由发现和维护也看作网络数据查询问题,利用传感器网络数据查询技术来处理路由等网络计算问题[21].

因为 SQL 是非递归语言,其本身不能定义图的连通性、可达性等问题,并不适合定义网络路由协议.因此,以递归的 Datalog 语言为基础,Loo 等人以及 Grumbach 等人提出用递归的网络查询语言来描述网络通信协议[9,11],并称其为“声明式网络”,用来定义路由发现协议[22]、覆盖网络协议[23]、Ad Hoc 网络路由[24,25]等.类似的工作还包括提供异步系统诊断[26]、网络监控[27]、节点探索[28]、QoS 路径维护[29]、拓扑探索[30]、网络安全[31]等.这些工作提供了新的面向应用的网络开发模式.

与传统的网络编程方式相比,声明式语言具有以下显著的优点:

(i) 它具有非常简洁的程序(与命令式语言相比尺寸小 1~2 个数量级)[32];

(ii) 与底层实现无关;

(iii) 可以形式化地定义网络语言的语义[11],使得协议的形式验证更加便利[33,34].

在以上的工作中,用户用声明式语言描述分布式的网络算法,节点利用本地执行计划[9]或本地虚拟机[35]执行分布式的声明式程序,完成协同工作.如上节所述,传感器网络数据查询处理系统中,查询的分布式处理过程对用户是透明的.因此,对于路由查询等网络计算,我们仍希望能够达到“用户只需声明网络整体性质,而将分布式计算交予网络完成”的目标.

基于这个目标,本文提出一种全局式的声明式网络开发模型,即用户提供对路由、生成树等网络功能的全局性描述,系统将全局性的声明式程序编译为节点可执行的分布式程序.

2 分布式的声明式网络开发模式

分布式的声明式网络开发模型是全局式的声明式网络开发模型的基础.该模型支持基于规则的递归的网络查询语言 Netlog[11],建立在网络节点配置的 Netquest 虚拟机(如图 2 所示)上[21,32,35,36].

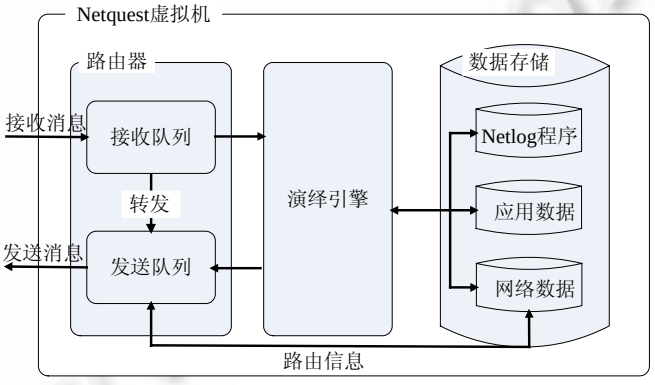

Fig.2 Netquest virtual machine

图 2 Netquest 虚拟机

用户用 Netlog 语言声明式地描述节点本地的功能,而将底层物理细节的实现交与系统处理.Netlog 程序为规则的集合,在网络的每个节点上并行地执行,利用规则递归地从已知的满足规则体的本地存储的事实和其他节点发送过来的事实以前推式的方法演绎新的事实,将新的事实存储在本地或者发送给其他节点,直到达到网络的分布式不动点.

2.1 网络计算模型

网络模型以分布式计算的消息传递系统[37]为基础.网络的拓扑由图 $\boldsymbol{G}=(V,Link)$定义,其中,V 是网络中节点的集合,$Link$ 是节点间的双向通信联接的集合.节点有唯一的身份标识,即 ID,取自 $1,2,\ldots,|V|$.数据以关系表的形式分布在各个节点上,节点之间不共享数据,也不控制其他节点.节点之间依靠消息进行通信,消息中含有内容和目标地址,其中,内容是由规则演绎的关系的实例,即事实;目标地址为节点的 ID 或广播地址 *all*.我们将发生在节点内的计算事件和节点间传递消息的通信事件区分开来.节点上的一个计算事件连同一个通信事件构成一个回合,而一系列回合构成一个执行序列.

节点有 3 个模块:路由器、演绎引擎和数据存储.节点有完全相同的行为.在回合 l,节点α的行为如下:

(1) 路由器将接收到的消息放入接收队列 $R^{\alpha}(l+1)$,并根据目标地址分成两组:

(i) 如果目标地址是α或 *all*,那么消息的内容组成 $L^{\alpha}(l+1)$,在下一回合开始时送给演绎引擎.

(ii) 如果目标地址是其他节点的 ID 或 *all*,那么这些消息组成 $F^{\alpha}(l+1)$被加入到发送队列 $P^{\alpha}(l+1)$中.同时,路由器将发送队列 $P^{\alpha}(l)$中的消息发送给其他节点:如果消息的目标地址是 *all*,那么发送给所有邻居节点;否则,在本地路由表中查找发往目标地址的路径信息,将消息发给该路径的下一跳.如果查找失败,则采取寻路失败策略,如丢弃该消息.

(2) 演绎引擎从本地数据存储中载入被 $L^{\alpha}(l)$中的事实触发的程序,循环地演绎程序的规则,直到不再演绎新的事实或者不再删除事实,同时更新本地数据库,并将部分演绎的事实作为消息送给路由器的发送队列 $P^{\alpha}(l+1)$.

(3) 数据存储保存两类数据:

(i) 节点相关的数据,即网络信息(例如拓扑、路由、通信代价等)或者与应用相关的数据.

(ii) 程序.

2.2 分布式的声明式程序

Netlog 语言以演绎规则语言 Datalog 为基础,为了支持分布式计算和网络通信,Netlog 语言在 Datalog 的基础上扩充了地址和通信等指令,具有分布式不动点语义[11].Netlog 程序是分布式的,节点不能读或者写其他节点存储的数据,这使得确定否定和删除谓词的值变得简单(否定和删除原子对于定义很多网络计算是必不可少的),同时也使得 Netlog 语言符合网络安全的需要.

Netlog 程序是形式为“规则头 :– 规则体”的规则的有限集合.规则具有前推的语义,即如果规则体满足,那么可以演绎规则头.所有规则并行地演绎.节点最初只知晓邻居节点,保存 *Link* 关系表的一个片段.节点循环地对程序的规则进行演绎,在每一次循环中,由本地数据存储中的事实演绎新的事实或者删除本地数据存储中的事实,直到本地数据存储的事实的集合达到不动点,即完成一次计算事件.同时,将上次回合中演绎的部分事实发送给其他节点,并接收其他节点发送来的事实,即完成一次通信事件.节点的一个回合包括一次计算事件和一次通信事件,一系列的回合组成节点的执行序列.当达到分布式不动点(网络上所有节点的数据存储不再改变)时,程序的执行终止.下文以 Ad Hoc 网络的先验式路由协议(如 DSDV 协议[38]、OLSR 协议[39]等)和反应式路由协议(如 AODV 协议[40]等)为例来介绍 Netlog 语言.

以下是简化的先验式路由协议,它维护完全路由表(关系 *Route*(*Source*,*NextHop*,*Destination*)):

程序 1. 先验式路由协议(分布式).

↓*Route*(@*x*,*d*,*d*) :– *Link*(@*x*,*d*).

↑*askRoute*(@*x*,*h*,*d*) :– *Link*(@*h*,*x*);*Route*(@*h*,*z*,*d*);$x\neq z$.

↓*Route*(@*x*,*h*,*d*) :– *Link*(@*x*,*h*);¬*Route*(@*x*,-,*d*);*askRoute*(@*x*,*h*,*d*).

规则头的存储/发送指令规定了由该规则演绎的事实的处理方式,其中,“↓”表示将事实存储在本地,“↑”表示将事实发送给别的节点.规则中含有地址指令“@”.规则体中的“@*x*”表示变量 *x* 只能被赋值为本地节点的 ID,而规则头中的“@*x*”表示该规则演绎产生的事实将被存储在/发送给为变量 *x* 赋值的 ID 的节点.规则体中的否定谓词“¬*Route*(@*x*,-,*d*)”被解释为被全称量词限定.比如,假设变量 *x* 和 *d* 分别被赋值为α和β,那么如果本地(节点α上)没有事实 *Route*(α,*C*,β),其中,*C* 可以代表任何值,那么¬*Route*(α,-,β)为真.

反应式路由协议并不维护完全路由表,而在有路由请求时才生成路由.以下是简化的反应式路由协议:

程序 2. 反应式路由协议(分布式).

↑*RouteReqMsg*(*x*,@*y*,*d*) :– *Link*(@*x*,*y*);*ReqNode*(@*x*);*Dest*(*d*).

↑*RouteReqMsg*(*x*,@*y*,*d*) :– *Link*(@*x*,*y*);*RouteReq*(*w*,@*x*,*d*);*x*≠*d*;*w*≠*y*.

↓*RouteReq*(*x*,@*y*,*d*) :– *RouteReqMsg*(*x*,@*y*,*d*);¬*RouteReq*(-,@*y*,*d*).

↑*RouteInf*(@*x*,*d*,*d*) :– *Link*(@*d*,*x*);*RouteReq*(*x*,@*d*,*d*).

↑*RouteInf*(@*x*,*y*,*d*) :– *Link*(@*y*,*x*);*RouteReq*(*x*,@*y*,*d*);*Route*(@*y*,*z*,*d*).

↓*Route*(@*x*,*y*,*d*) :– *RouteInf*(@*x*,*y*,*d*).

其中,路由请求节点被 *ReqNode*(*RequstingNode*)触发,它寻找到终点(由 *Dest* 事实规定)的路由.它将寻路消息 *RouteReqMsg*(*RequstingNode*,*RequstedNode*,*Destination*)发送给邻居节点.初次接收到寻路消息的节点保存请求记录 *RouteReq*(*RequstingNode*,*RequstedNode*,*Destination*)以备将来形成反向路由.同时,如果它是终点,则通过消息 *RouteInf*(*Source*,*NextHop*,*Destination*)将路由信息沿反向路由送回请求节点;否则,转播寻路消息给邻居节点.注意,节点接收到的事实只参与规则的演绎而不直接存储在本地.

另外,规则体中可以含有删除指令“!”,规则头可以为空.例如,规则“:– *Link*(@*d*,*x*);!*RouteReq*(*x*,@*d*,*d*).”的演绎结果是将满足规则体条件的本地的 *RouteReq* 事实删除.

3 全局式的声明式网络开发模式

用逻辑公式表达图上的查询是一个经典的论题.霍恩子句逻辑的不动点扩展,即 Datalog 语言,可以表达图的连通性、可达性等问题[7,8].这使我们想到,用户可以用 Datalog 语言定义网络的全局功能,再利用编译算法生成分布式的 Netlog 程序,交予网络节点的虚拟机执行.这如同将整个网络看成一个数据库,向其提出 SQL 查询,再利用分布式查询处理计划将执行分布到网络上.利用这种模式,用户可以在全局层次上定义网络功能,而不用定义每个节点的本地行为.

3.1 全局式的声明式程序

全局式的声明式程序是用Datalog语言定义的,只需定义网络的全局功能,不需要考虑全局功能是如何在网络上实现的.Datalog 程序也是形式为“规则头 :– 规则体”、前推式演绎的规则的有限集合.与 Netlog 语言不同,Datalog 规则不使用地址指令“@”规定哪些特定的变量应被赋值为节点的 ID,也不使用存储/发送指令“↓/↑”规定事实存储的节点以及节点之间的通信.Datalog 语言具有不动点语义,即循环地对程序的规则进行演绎,从已知的事实演绎新的事实或者删除事实,直到不再演绎新的事实或者删除事实.此时,称为程序的执行终止.为了便于与Netlog程序相区分,以下称Datalog程序执行时对所有规则的一次演绎为一个阶段.以下是全局式的先验式路由协议和反应式路由协议:

程序 3. 先验式路由协议(全局式).

Route(*x*,*d*,*d*) :– *Link*(*x*,*d*).

Route(*x*,*h*,*d*) :– *Link*(*h*,*x*);*Route*(*h*,*z*,*d*);¬*Route*(*x*,-,*d*);*x*≠*z*.

程序 4. 反应式路由协议(全局式)

RouteReq(*x*,*y*,*d*) :– *Link*(*x*,*y*);*ReqNode*(*x*);*ReqNode*(*d*).

RouteReq(*x*,*y*,*d*) :– *Link*(*x*,*y*);*RouteReq*(*w*,*x*,*d*);¬*RouteReq*(-,*y*,*d*);*x*≠*d*;*w*≠*y*.

Route(*x*,*d*,*d*) :– *Link*(*d*,*x*);*RouteReq*(*x*,*d*,*d*).

Route(*x*,*y*,*d*) :– *Link*(*y*,*x*);*RouteReq*(*x*,*y*,*d*);*Route*(*y*,*z*,*d*).

3.2 全局式程序的编译

本节描述将全局式的 Datalog 程序编译为分布式的 Netlog 程序的普适性算法.因为 Datalog 程序包含否定谓词和删除操作,为保持语义的一致,需假定程序在同步的网络上执行,对于异步的网络需要时钟同步机制.编译的目标主要是将 Datalog 规则的演绎过程分布到网络的各个节点上,并添加节点之间的通信,使节点协同工作.编译过程分为 4 个步骤:

1) 将数据分布到各节点:调用算法 Localize 为每个关系指定某一个变量为持有变量,为其添加地址指令,即规定事实如何分布式地存储在网络的各个节点上.例如,对 *Route*(*Sourse*,*NextHop*,*Destination*)关系,如果规定最左边的变量为持有变量,那么各个节点只存储全网路由表中与自己相关的片断,即以自己为起点的路由.

2) 将计算分布到各节点:如果规则体中的关系具有不同的持有变量,那么这条规则的演绎需要由多个节点协同完成.调用递归算法 Rewrite 将这条规则改写成分别在多个节点上演绎的多条规则.例如,对于图 3 中的规则,规则体中的 4 个关系具有不同的持有变量(分别为 x_1,x_2,y_1 和 z).算法 Rewrite 选定其中一个关系作为“本地部分”,如“*Link*(@x_1,x_2)”,在节点上演绎规则时,将对变量 x_1 赋予该节点的 ID 值;而其他部分,即“R_1(@x_2); *Link*(@y_1,y_2);R_2(@z).”为 3 个“互不连接的”“非本地部分”.在原规则中,将“非本地部分”用新的关系代替,对应每一个“非本地部分”添加一个新的规则,该规则以新的关系为规则头,以“非本地部分”为规则体.如果新规则的规则体中的关系仍旧具有不同持有变量,那么调用算法 Rewrite 对新规则进行同样方式的改写.原规则的演绎将被分布在不同的节点上,“非本地部分”在其他节点上演绎并将演绎的事实发送回本地节点,连同“本地部分”一起演绎原规则的结果.

3) 建立节点间通信:对经步骤 2)改写得到的规则,如果规则头的持有变量和规则体中的不同,那么需要在规则头前添加“↑”指令,即规定将该规则演绎的事实发送到其他节点;否则,在规则头前添加“↓”指令,即规定将该规则演绎的事实存储在本地.

4) 添加同步机制:如果一条 Datalog 规则被改写成几条在不同节点上演绎的 Netlog 规则,那么原 Datalog 规则的一次演绎需要几个节点协同完成,即原 Datalog 程序执行的一个阶段对应于 Netlog 程序执行的多个回合(以下称这几个回合为一个循环).因为程序含有否定谓词和删除操作,为保持语义的一致,需要添加一些规则来同步节点上的分布式计算.

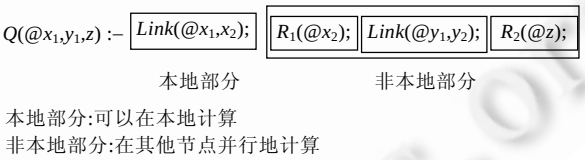

Fig.3 Local part and non-local parts of a rule

图 3 规则的“本地部分”和“非本地部分”

在给出编译算法之前,首先规定经过编译得到的规则需满足以下本地化限制条件,以保证其满足 Netlog 语言的语法规定[11]:

a) 规则体中所有的关系具有相同的持有变量;

b) 如果规则为存储规则,那么规则头中的持有变量和规则体中的相同;

c) 如果规则为外推规则,那么规则头中的持有变量(假设为@x)和规则体中的持有变量(假设为@y)不同,并且 *Link*(@y,x)出现在规则体中的非否定谓词中.

算法 Compile 将 Datalog 程序编译为分布式的 Netlog 程序.为简单起见,这里假设 Localize 算法选取每个关系最左边的变量为持有变量,且算法 Compile 和 Rewrite 皆为朴质算法,未考虑优化.另外,假设已知图 $\boldsymbol{G}$ 的直径.

算法 1. *Compile*(P_C).

输入程序 $\boldsymbol{P}_C$;输出程序 $\boldsymbol{P}_D$.

步骤 1. 将数据分布式化

输入程序 $\boldsymbol{P}_C$;输出程序 $\boldsymbol{P}_1$.

调用算法 *Localize*($\boldsymbol{P}_C$)为 $\boldsymbol{P}_C$ 中每个关系选择一个持有变量,添加标记“@”,得到 $\boldsymbol{P}_1$.

步骤 2. 将计算分布式化

输入程序 $\boldsymbol{P}_1$;输出程序 $\boldsymbol{P}_2$、整数κ.

对每个规则 $r\in\boldsymbol{P}_1$,假设 Add_r 是 r 的规则头的持有变量;令变量的集合 $Cn_r=\{Add_r\}$;

调用算法 *Rewrite*(r,Add_r,Cn_r),得到一个规则的集合 T_r 和整数κ_r.

令 $\boldsymbol{P}_2=\bigcup_{r\in P_1}T_r$, $\kappa=\max\{\Delta,\max\{\kappa_r|r\in\boldsymbol{P}_1\}\}$,其中,$\Delta$为图 $\boldsymbol{G}$ 的直径.

步骤 3. 建立通信

输入程序 $\boldsymbol{P}_2$;输出程序 $\boldsymbol{P}_3$.

对 $\boldsymbol{P}_2$ 进行如下修改得到 $\boldsymbol{P}_3$:对 $\boldsymbol{P}_2$ 中每个规则头和规则体的持有变量不同的规则,在规则头前添加标记“↑”;对 $\boldsymbol{P}_2$ 中每个规则头和规则体的持有变量相同的规则,在规则头前添加标记“↓”.

步骤 4. 协同计算

输入程序 $\boldsymbol{P}_3$、整数κ;输出程序 $\boldsymbol{P}_D$.

对 $\boldsymbol{P}_3$ 中的规则进行如下修改得到 $\boldsymbol{P}_D$:

1) 在每个规则的规则体前端添加字段“*Clock*(@x,q);$q\neq0$;”,其中,x 为规则体中关系的持有变量;
2) 对每个在规则头中含有 $\boldsymbol{P}_C$的一个外延关系 R 的规则(程序的外延关系为只出现在程序的规则的规则头中的关系),将规则头中的关系名“*R*”替换为一个新的关系名,例如“*TempR*”,并添加以下规则(假设关系 R 和 *TempR* 有 i+1 个变量,x 为关系 R 和 *TempR* 的持有变量):

   ↓*R*(@x,x_1,…,x_i) :– !*TempR*(@x,x_1,…,x_i);!*Clock*(@x,0).

   ↓*Continue*(@x) :– *TempR*(@x,x_1,…,x_i);!*Clock*(@x,0);¬*R*(@x,x_1,…,x_i).

   ↑*Inform*(@y,x) :– *TempR*(@x,x_1,…,x_i);!*Clock*(@x,0);¬*R*(@x,x_1,…,x_i);*Link*(@x,y).
3) 添加以下规则:

   ↓*Continue*(@x) :– !*Start*(@x).

   ↑*Inform*(@y,x) :– !*Start*(@x);*Link*(@x,y).

   ↓*Clock*(@x,κ) :– !*Start*(@x).

   ↓*Clock*(@x,κ) :– !*Clock*(@x,0);!*Continue*(@x).

   ↓*Clock*(@x,p) :– !*Clock*(@x,q);$q\geqslant1$;$p=q-1$.

   :– !*Clock*(@x,0);¬*Continue*(@x).

   ↑*Inform*(@z,y) :– *Inform*(@y,x);!*Clock*(@y,q);*Link*(@y,z);$x\neq z$;$q>\kappa-\Delta+1$.

   ↓*Continue*(@x) :– *Inform*(@x,y);!*Clock*(@x,q);$q\neq0$.

算法 2. *Rewrite*(r,Add_r,Cn_r).

输入规则 r、变量 Add_r、变量的集合 Cn_r;输出规则的集合 T_r、整数κ_r.

假设规则 r 为γ :– γ_1;…;γ_l,其中,l>0.以下用 Add_γ表示字段γ中关系的持有变量.

令字段的集合 $S=\{\gamma_1,\ldots,\gamma_l\}$.令字段的集合 $S'=\{\gamma_i|\gamma_i\in S$ 并且 $Add_{\gamma_i}=Add_r\}$.

(S'包含所有规则 r 的规则体中持有变量和 Add_r 相同的字段)

1. 如果 $S'=S$,那么令 $T_r=\{r\}$,令κ_r=1.
2. 如果 $S'\neq S$,那么,

   Begin

   令字段的集合 $S''=S-S'$.

(S''包含所有规则 r 的规则体中持有变量和 Add_r 不同的字段)

对任何$\gamma_j,\gamma_k\in S''$,令变量的集合 $V_{jk}=\{v|v$ 是γ_j 和γ_k 的共同变量$\}$.如果 $V_{jk}-Cn_r\neq\varnothing$,那么令$\gamma_j\approx\gamma_k$.

假设 S''在关系$\approx$下闭包的最小子集的划分为 $\{S_1'',...,S_n''\}$.对每个 S_i'',$i\in[1,n]$,令

$$T_i=\{Add_{\gamma_{i,w}}\mid \gamma_{iw}\in S_i'',\text{并且}Link(@Add_r,Add_{\gamma_{i,w}})\in S'\}.$$

(T_i 中包含所有的既是 S_i'' 中关系的持有变量又是 Add_r“邻居”的变量)

2.1. 如果 $T_i\neq\varnothing$(即 S_i'' 中的“非本地部分”与 S'中的“本地部分”单跳“连接”),那么从 T_i中选择一个变量,例如 $Add_{\gamma_{i,u}}$.

令 $S_i''=S_i''\cup\{Link(@Add_{\gamma_{i,u}},Add_r)\}$, 令 $Cn_{r_i}=Cn_r\cup\{Add_{\gamma_{i,u}}\}$, 令 $d_{r_i}=1$. 假设 $S_i''=\{\gamma_{i,1},...,\gamma_{i,m_i}\}$.

令规则 r_i 为

$$Q_i(@Add_r,y_{i,1},...,y_{i,l_i}):-\gamma_{i,1};...;\gamma_{i,m_i}.$$

其中,Q_i 是一个新的关系名,集合 $\{Add_r,y_{i,1},...,y_{i,l_i}\}$ 包含所有同时出现在 S_i'' 和 S'或者 r 的规则头中的变量.

2.2. 如果 $T_i=\varnothing$(即 S_i'' 中的“非本地部分”与 S'中的“本地部分”单跳不“连接”),那么从 S_i'' 中选择一个字段,例如$\gamma_{i,u}$.假设 y 是一个没有出现在 r 中的变量,令 $Cn_{r_i}=Cn_r\cup\{Add_{\gamma_{i,u}}\}$.令 $d_{r_i}=1+\Delta$.假设 $S_i''=\{\gamma_{i,1},...,\gamma_{i,m_i}\}$.令规则 r_i 为

$$Q_i(@Add_{\gamma_{i,t}},y_{i,1},...,y_{i,l_i}):-\gamma_{i,1};...;\gamma_{i,m_i}.$$

其中,Q_i 是一个新的关系名,$\{Add_{\gamma_{i,t}},y_{i,1},...,y_{i,l_i}\}$ 为所有同时出现在 S_i'' 和 S'或者 r 的规则头中的变量的集合.

令规则 r_i' 为

$$Q_i(@x,y_{i,1},...,y_{i,l_i}):-Q_i(@y,y_{i,1},...,y_{i,l_i});Link(@y,x).$$

End

假设 $S'=\{\gamma_1',...,\gamma_k'\}$,令规则 r'为

$$\gamma:-\gamma_1';...;\gamma_k';Q_1(@Add_r,y_{1,1},...,y_{1,l_1});...;Q_n(@Add_r,y_{n,1},...,y_{n,l_n}).$$

其中,$\gamma,\gamma_1',...,\gamma_k'$ 的持有变量都是 Add_r.

对每一个 $i\in[1,n]$,调用 $Rewrite(r_i,Add_{\gamma_{i,u}},Cn_{r_i})$,得到$\langle T_{r_i},\kappa_{r_i}\rangle$.

令 $T_r=\{r'\}\cup\bigcup_{i\in[1,n]}(\{r_i'\}\cup T_{r_i}),\kappa_r=\max\{\kappa_{r_i}+d_{r_i}\mid i\in[1,n]\}$.

通过编译算法 Compile 改写得到的程序中的规则符合本地化限制条件.程序 3 和程序 4 可以分别由程序 1 和程序 2 通过该算法改写并删减不必要的控制关系后得到.需要补充的是,在应用中,分布式的 Netlog 语言的程序前端还包含一段声明,用于定义数据结构(即关系)和元数据.当演绎引擎载入程序时,本地数据库载入声明中定义的元数据.因此,编译算法 Compile 还应为全局式的程序添加一段声明,包括数据结构和元数据“$Start(@x)$”,其中,x 在节点载入程序时被赋值为节点的 ID.以上算法描述中,我们主要关心规则的改写而忽略了声明的添加.

定理 1. 对于拓扑结构由图 $\boldsymbol{G}=\{V,Link\}$定义的同步网络 G,全局式 Datalog 程序 $\boldsymbol{P}_C$ 以及对它实行算法 Compile 得到的分布式 Netlog 程序 $\boldsymbol{P}_D$,$\boldsymbol{P}_D$ 在 G 上的执行终止当且仅当 $\boldsymbol{P}_C$ 对 G 的执行终止,并且具有相等的不动点.

证明(概要):$\boldsymbol{P}_D$ 延缓了 $\boldsymbol{P}_C$ 的执行过程,$\boldsymbol{P}_C$ 执行的一个阶段对应于 $\boldsymbol{P}_D$ 执行的 $\kappa+1$ 个回合(以下称为一个循环).在 $\boldsymbol{P}_D$ 执行的一个循环中,定时器(由关系 *Clock* 定义)从 κ 递减到 0,非本地节点执行“非本地部分”.根据算法 Compile 中 κ 的定义,κ 个回合足够使“非本地部分”演绎的事实通过消息发送回本地节点,用以生成 $\boldsymbol{P}_C$ 的外延关系 R 的附属关系 *TempR* 的事实.在一个循环结束之前,本次循环演绎的关系 R 的事实暂存在 *TempR* 中.在循环结束时(即定时器为 0 时),用 *TempR* 的事实更新关系 R.所以,$\boldsymbol{P}_C$ 执行的一个阶段演绎的关系 R 的事实等于 $\boldsymbol{P}_D$ 在各个节点并行执行的一个循环演绎的关系 R 的事实的并集.

$\boldsymbol{P}_D$ 在 G 上的执行的终止由关系 *Continue* 来控制.在各个节点,例如节点 α 上,元数据 $Start(\alpha)$ 触发 $\boldsymbol{P}_D$ 在节点

α上的执行,并且演绎事实 *Clock*(α,κ)、*Continue*(α)和发送给节点β的事实 *Inform*(β,α),其中,节点β是节点α的邻居.在定时器从κ递减到 1 的过程中,“非本地部分”执行完成,并且将演绎的事实传递回本地节点用以执行“本地部分”.$\boldsymbol{P}_C$ 的外延关系 *R* 的事实暂存在 *TempR* 中.当定时器为 0 时,更新 *R* 关系表并清空 *TempR* 关系表.如果本轮循环演绎了新的事实,那么生成 *Continue*(α),开始下一轮的循环.同时,发送事实 *Inform*(γ,α)给节点γ.在下一轮循环中,*Inform* 事实被转发到所有节点,例如α'上,生成 *Continue*(α').并且 *Continue*(α')将一直保存到该轮循环的结束.所以,当定时器再一次变为 0 时,对任何节点,例如α,α持有 *Continue*(α)事实,当且仅当其他所有节点,例如α',持有 *Continue*(α')事实.此时,每个节点,例如α检查本地是否有 *Continue*(α)事实.如果有,那么清除 *Continue*(α)事实,重新设定定时器,即生成 *Clock*(α,κ),继续下一个循环;如果没有,那么清除定时器,即清空 *Clock* 关系表,并且此时所有其他节点都因没有 *Continue* 事实而清除定时器.因为定时器是所有规则演绎的条件,所以所有节点上 $\boldsymbol{P}_D$ 的执行终止. □

4 结束语

面向系统的开发模式与传感器网络灵活性、可扩展性、易开发性和易维护性等要求相违背,缺乏编程抽象成为传感器网络发展的一个障碍.因此,声明式的网络开发模式被认为是传感器网络发展的一个重要问题.查询语言适合传感器网络的数据密集型应用,而递归的查询语言适合定义路由协议等网络计算问题.本文探讨一种更高层次抽象的网络应用开发模式,即全局式的声明式传感器网络开发模式.用户提供全局式的声明式网络程序,系统将其编译为分布式程序,再交给分布式的虚拟机执行.本文中的编译算法是朴质算法,下一步的工作包括基于全局式程序规则的形式对编译算法进行优化.

致谢 作者感谢林惠民院士和 Stéphane Grumbach 教授参与全局式的声明式网络开发模式的工作,并给予作者指导.

References:

[1] Marron J, Minder D. Embedded WiSeNts Research Roadmap. Berlin: Logos Verlag, 2006.

[2] Hejlsberg A. Trends and future directions in programming languages. Technical Report, TechDays 2010 Keynote, 2010.

[3] Ramakrishnan R, Gehrke J. Database Management Systems. New York: McGrawHill, 2003.

[4] Ramakrishnan R, Ullman JD. A survey of deductive database systems. Journal of Logic Program, 1995,23(2):125−149. [doi: 10.1016/0743-1066(94)00039-9]

[5] Demers AJ, Gehrke J, Rajaraman R, Trigoni A, Yao Y. The cougar project: A work-in-progress report. SIGMOD Record, 2003, 32(4):53−59. [doi: 10.1145/959060.959070]

[6] Madden S, Franklin MJ, Hellerstein JM, Hong W. Tinydb: An acquisitional query processing system for sensor networks. ACM Trans. on Database System, 2005,30(1):122−173. [doi: 10.1145/1061318.1061322]

[7] Abiteboul S, Hull R, Vianu V. Foundations of Databases. Boston: Addison-Wesley, 1995.

[8] Ebbinghaus HD, Flum J. Finite Model Theory. Berlin: Springer-Verlag, 1999.

[9] Loo BT, Condie T, Garofalakis MN, Gay DE, Hellerstein JM, Maniatis P, Ramakrishnan R, Roscoe T, Stoica I. Declarative networking: Language, execution and optimization. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Chicago, 2006. 97−108. [doi: 10.1145/1142473.1142485]

[10] Loo BT, Condie T, Garofalaks M, Gay DE, Hellerstein JM, Maninatis P, Ramakrishnan R, Stoica I. Declarative networking. Communications of the ACM, 2009,52(11):87−95. [doi: 10.1145/1142473.1142485]

[11] Grumbach S, Wang F. Netlog, a rule-based language for distributed programming. In: Proc. of the 12th Int'l Symp. on Practical Aspects of Declarative Languages (PADL 2010). Madrid: Springer-Verlag, 2010. 88−103. [doi: 10.1007/978-3-642-11503-5_9]

[12] Li JZ, Li JB, Shi SF. Concepts, issues and advance of sensor networks and data management of sensor networks. Journal of Software, 2003,14(10):1717−1727 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1717.htm

[13] Fung WF, Sun D, Gehrke J. Cougar: The network is the database. In: Proc. of the SIGMOD Conf. 2002. 621. [doi: 10.1145/564691.564775]

[14] Srivastava U, Munagala K, Widom J. Operator placement for in-network stream query processing. In: Proc. of the 24th ACM Symp. on Principles of Database Systems. 2005. 250−258. [doi: 10.1145/1065167.1065199]

[15] Jeffery SR, Alonso G, Franklin MJ, Hong W, Widom J. Declarative support for sensor data cleaning. In: Proc. of the 4th Int'l Conf. on Pervasive Computing. 2006. 83−100. [doi: 10.1007/11748625_6]

[16] TinyOS. http://www.tinyos.net

[17] Al-Karaki JM, Kamal AE. Routing techniques in wireless sensor networks: A survey. In: Proc. of the IEEE Personal Communications. 2004. 6−28.

[18] Chalermek I, Ramesh G, Deborah E. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proc. of the MobiCom. 2000. 56−67. [doi: 10.1145/345910.345920]

[19] Stojmenovic I. Position based routing in ad hoc networks. Communications Magazine, 2002,40(7):128−134.

[20] Alexandru C, Mario AN, Jorg S. A framework for spatio-temporal query processing over wireless sensor networks. In: Proc. of the 1st Int'l Workshop on Data Management for Sensor Network in Conjunction with VLDB. 2004. 104−110. [doi: 10.1145/1052199.1052217]

[21] Bauderon M, Grumbach S, Gu D, Qi X, Qu W, Suo K, Zhang Y. Programming iMote networks made easy. In: Proc. of the 4th Int'l Conf. on Sensor Technologies and Applications (SENSORCOMM 2010). Venice, Mestre: CPS Press, 2010. 539−544. [doi: 10.1109/SENSORCOMM.2010.87]

[22] Loo BT, Hellerstein JM, Stoica I, Ramakrishnan R. Declarative routing: Extensible routing with declarative queries. In: Proc. of the ACM SIGCOMM 2005 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications. Philadelphia, 2005. 289−300. [doi: 10.1145/1090191.1080126]

[23] Loo BT, Condi T, Hellerstein JM, Maniatis P, Roscoe T, Stoica I. Implementing declarative overlays. In: Proc. of the 20th ACM Symp. on Operating Systems Principles. Brighton, 2005. 75−90. [doi: 10.1145/1095809.1095818]

[24] Grumbach S, Lu JL, Qu WW. Self-Organization of wireless networks through declarative local communication. In: Proc. of the OTM, on the Move Conf., MONET Workshop. LNCS 4805, 2007. 497−506. [doi: 10.1007/978-3-540-76888-3_72]

[25] Liu C, Mao Y, Oprea M, Basu P, Loo BT. A declarative perspective on adaptive MANET routing. In: Proc. of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO 2008). New York: ACM Press, 2008. 63−68. [doi: 10.1145/1397718.1397733]

[26] Abiteboul S, Abrams Z, Haar S, Milo T. Diagnosis of asynchronous discrete event systems: Datalog to the rescue! In: Proc. of the 24th ACM SIGACT-SIGMOD—SIGART Symp. on Principles of Database Systems. Baltimore, 2005. 358−367. [doi: 10.1145/1065167. 1065214]

[27] Reiss F, Hellerstein JM. Declarative network monitoring with an underprovisioned query processor. In: Proc. of the ICDE. 2006. 56. [doi: 10.1109/ICDE.2006.46]

[28] Alonso G, Kranakis E, Sawchuk C, Wattenhofer R, Widmayer P. Probabilistic protocols for node discovery in ad hoc multi-channel broadcast networks. In: Proc. of the 2nd Int'l Conf. on Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW). 2003. 104−115.

[29] Bejerano Y, Breitbart Y, Orda Y, Rastogi R, Sprintson A. Algorithms for computing QoS paths with restoration. IEEE/ACM Trans. on Network, 2005,13(3):648−661. [doi: 10.1109/TNET.2005.850217]

[30] Bejerano Y, Breitbart Y, Garofalakis MN, Rastogi R. Physical topology discovery for large multi-subnet networks. In: Proc. of the INFOCOM. 2003. 342−352. [doi: 10.1109/INFCOM.2003.1208686]

[31] Abadi M, Loo BThau. Towards a declarative language and system for secure networking. In: Proc. of the 3rd USENIX Int'l Workshop on Networking Meets Databases (NETB 2007). Berkeley, 2007. 1−6.

[32] Suo K, Qu WW, Iriondo AB. Declarative programming of network protocols. In: Proc. of the Int'l Conf. on Communication Technology (ICCT 2010). Nanjing, 2010. 346−349. [doi: 10.1109/ICCT.2010.5689211]

[33] Wang A, Basu P, Loo BT, Sokolsky O. Declarative network verification. In: Proc. of the 11th Int'l Symp. on Practical Aspects of Declarative Languages (PADL 2009). Springer-Verlag, 2009. 61−75. [doi: 10.1007/978-3-540-92995-6_5]

[34] Deng YX, Grumbach S, Monin JF. A framework for verifying data-centric protocols. In: Proc. of the 31th IFIP Int'l Conf. on Formal Techniques for Networked and Distributed Systems (FORTE 2011). Reykjavik, 2011. 105−120.

[35] Bauderon M, Bobineau C, Grumbach S, Henry A, Qi X, Qu WW, Suo K, Wang F, Wu ZL. Netquest: An abstract model for pervasive applications. In: Proc. of the 7th Int'l Conf. on Pervasive Computing (Pervasive 2009). 2009. 467−481.

[36] Bellemon E, Dubosclard V, Grumbach S, Suo K. QuestMonitor: A visualization platform for declarative network protocols. In: Proc. of the 8th Int'l Conf. on Modeling, Simulation and Visualization Methods (MSV 2011). Las Vegas, 2011. 29−35.

[37] Attiya H, Welch J. Distributed Computing: Fundamentals, Simulations and Advanced Topics. Chichester: Wiley, 2004.

[38] Perkins CE, Bhagwat P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proc. of the ACM Conf. on Communications Architectures, Protocols and Applications (SIGCOMM'94). London, 1994. 234−244. [doi: 10.1145/190809.190336]

[39] Thomas C, Philippe P. Optimized Link State Routing Protocol (OLSR). Research Report, inria-00471712, Institut National de Recherché en Informatique et en Automatique, 2010.

[40] Perkins CE, Royer EM. Ad-Hoc on-demand distance vector routing. In: Proc. of the IEEE Workshop on Mobile Computing Systems and Applications. Los Alamitos, 1999. 90−100.

附中文参考文献:

[12] 李建中,李金宝,石胜飞.传感器网络与感知数据管理的概念、问题与研究进展.软件学报,2003,14(10):1717−1727. http://www.jos.org.cn/1000-9825/14/1717.htm

汪芳(1981－),女,安徽安庆人,博士,工程师,CCF 会员,主要研究领域为数据库理论,查询语言,并行计算.

张云勇(1976－),男,博士,高级工程师,CCF高级会员,主要研究领域为下一代开放网络,固定移动融合核心网,泛在网,云计算.