宣告式网络程序设计语言比较研究

齐 欣[1] 曲文武[2]
(中国科学院自动化研究所中法联合实验室 北京 100190)[1]
(中国科学技术大学计算机科学与技术系 合肥 230027)[2]

摘 要 网络技术的发展和异构计算设备种类的增多给网络协议的设计开发带来诸多挑战。网络协议设计者面临的最基本问题是如何脱离繁琐的协议实现细节,而将主要精力放在协议的功能设计上。近年来,为了解决这个问题,宣告式网络程序设计语言,被提出。宣告式网络程序设计语言吸取数据库管理系统成功的经验,将网络划分为逻辑层和物理层。网络协议设计者只需利用其提供的高层编程抽象设计网络协议的功能,而不用关心繁杂的物理层实现。通过分析和比较不同宣告式网络程序设计语言,对其发展进行了总结,并指出了进行进一步的研究工作需要注意的问题。

关键词 宣告式,编程抽象,逻辑层,物理层
中图法分类号 TP393 **文献标识码** A

Comparison Study of Declarative Networking Program Language

QI Xin[1] QU Wen-wu[2]
(Institute of Automation, Chinese Academy of Science, Beijing 100190, China)[1]
(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)[2]

Abstract The development of network technology and the increasing type number of heterogeneous computational devices have brought lots of challenges to network protocol design. The fundamental problem in front of the network protocol designer is how to get rid of the tedious protocol implementation details and pay their attention mainly to the design of protocol function. Recent years, for sake of solving this problem, the declarative networking programming language has been proposed. The declarative networking programming language has absorbed the experience of database management system's success and divided the network into logical level and physical level. The network protocol designers only need to use the high-level programming abstraction provided by declarative networking programming language to design the function of network protocol and don't need to take care of the tedious physical implementation. This paper surveyed the development of declarative networking programming language through analyzing and comparing different declarative networking programming languages. Besides, we pointed out the problems that should be paid attention to by further research work.

Keywords Declarative, Programming abstraction, Logical level, Physical level

近年来,随着网络技术的发展与异构计算设备种类数目的增多,普适计算的趋势不可阻挡。异种计算设备的间歇可用性、网络的动态性等都给普适计算带来了巨大挑战。在这些异构的网络设备上开发应用,使其协同工作而产生的一个基本障碍是缺乏好的编程抽象。现在的设备不够"灵巧",各种设备在硬件和软件上存在着很大的差异,不能够满足未来普适计算应用的需要(灵活性、可测性、易于实现维护等)。例如无线传感器网络的配置工作非常枯燥,它需要既懂网络协议又懂操作系统和硬件的专家来操作。要解决此难题,正确的做法是:"编程模型应该可以使应用领域的专家更多地关注应用本身的各种行为,而不是埋头于处理为了实现这些行为而需要解决的细节性软硬件问题。特别地,在编程模型上应该能够将整个无线传感器网络作为一个整体进行编程,而不是对于每个网络节点进行编程"[1]。这句话体现的基本思想是,把系统的逻辑层从物理层独立出来,用户或具体应用直接访问逻辑层而无需考虑物理层的细节。这也是数据库管理系统的基本准则[2]。基于此准则提出了宣告式网络程序设计语言。

宣告式网络程序设计语言是一个崭新的研究领域,本文旨在总结已有的宣告式网络程序设计方法在网络环境中的应用,概括其已有的研究工作,分析和比较不同宣告式网络程序设计语言在语法、语义上的差别以及在语言支持系统实现上的异同点。最后,本文对宣告式网络程序设计语言的发展进行了总结,并指出了进行进一步的研究工作时需要注意的问题。

1 宣告式网络程序设计语言的概述

在宣告式网络程序设计语言出现之前,宣告式方法已在

到稿日期:2009-12-04 返修日期:2010-03-02 本文受法国国家研究机构(ANR)Ubiquest项目基金(Project No. NT09_537153)资助。
齐 欣(1984-),男,硕士生,主要研究方向为无线网络宣告式程序设计,E-mail:qixin1984@gmail.com;曲文武(1979-),男,博士生,主要研究方向为无线网络声明方法。

网络环境中存在广泛的应用。在数据查询操作较为集中的无线传感器网络中，宣告式数据查询语言被提出。美国加州大学伯克利分校在无线传感器网络中开发了数据查询处理系统TinyDB[3]。在TinyDB系统之上，用户程序不需要书写繁琐易出错的C程序从网络中提取数据，而是通过该系统提供的与SQL语言类似的编程接口，像利用SQL语言从数据库中进行数据查询一样在无线传感器网络中抽取数据信息。TinyDB系统需要装备到有TinyOS操作系统的无线传感器组成网络上。

此外，与TinyDB系统类似，由美国康奈尔大学开发的Cougar[4]也为无线传感器网络中的数据查询提供了一种宣告式方法。在为数据查询提供高层抽象的同时，该系统还利用跨层优化对无线传感器网络中的能源管理问题进行了研究。

TinyDB系统和Cougar系统说明了在无线网络中使用与SQL类似的语言进行查询是可行的。在考虑能量限制的前提下，这两个系统分别给出了有效的数据分布及查询处理策略。在已知网络全局信息和网络约束节点的计算能力的情况下，它们能集中式计算出数据子查询在网络中的最优策略[5]。

另外，根据无线传感器数据的空间和时序特征，宣告式方法还应用在传感器网络不可靠数据的删除上[6]。

宣告式方法也被应用在描述网上的一些其他的问题，如节点发现[7]、路由发现、Qos路经维护[8]、拓扑发现(包括物理拓扑发现[9])等。美国加州大学伯克利分校数据库研究组第一次将递归查询语言用于表达网络层问题，他们在推导式数据库查询Datalog[19]的基础之上提出了描述路由协议[10,18]宣告式网络程序设计语言NDLog和描述覆盖层网络协议[11]的OverLog。NDLog和OverLog的提出标志着宣告式网络程序设计语言概念的提出。

但是NDLog和OverLog只是一个由启发式方法定义的语言，它本身缺乏对语法、语意的探讨。而且，它们缺乏描述无线网络动态性和实时性的语言元素。针对此问题，中科院自动化所中法联合实验室Netquest研究组(本研究组)形式化地定义了一种新的宣告式网络程序设计语言NetLog[12]。与NDLog和OverLog相比，NetLog更适合描述无线网络的动态性和实时性。此外，该研究组语言对NetLog语言的语法、语意进行了深入探讨[13]。

本文第2节将对NDLog和OverLog的定义和支持系统的实现进行介绍；第3节对NetLog和支持系统的实现进行介绍；第4节比较两类宣告式网络程序设计语言在语法、语意和支持系统上的异同；最后给出总结并对今后的工作进行展望。

2 NDLog语言和OverLog语言介绍

在大范围、分布式的计算机网络环境中，网络协议通常要在网络的可扩展行和灵活性之间以及网络的鲁棒性和效率之间进行博弈[14]。另一方面，网络环境的不断变化以及网络应用的层出不穷都使得已经成熟的网络协议失去本身的优势。以无线传感器网络为例，与一般无线设备不同，传感器本身能量有限，而且传感器之间发送的数据包往往只有几十个字节。这些特点使得原本在无线Ad Hoc网络中适用的协议在无线传感器网络中不再适用。因此，无线传感器网络需要自己特有的网络协议来适应低能量、数据包小的特点。传统的网络协议的设计开发是通过低层的程序设计语言直接控制网络设备，网络协议的开发往往涉及很多技术细节，使协议的设计者不能完全将精力集中于网络协议本身的功能特性上。解决以上问题的一个有效途径是为网络协议设计者提供一个好的原型系统，在此系统中，协议设计者只需关注协议的行为，而无需过多关心这些行为实现的细节，因此他们可以根据需要快速地开发出协议，并在原型系统中进行试验，利用试验结果的反馈更新改进协议。NDlog，OverLog以及支持它们的系统正是为网络协议的设计者提供这样一种工具。

2.1 语言介绍

在NDLog中网络中的所有数据都以数据库关系表的形式存储。这沿袭了推导式数据库查询语言Datalog的风格，利用NDLog语言写出的程序由一系列规则组成，规则的形式如下：

$p:-q_1,q_2,\cdots,q_n.$

式中，p是需要查询的数据的关系名和模式，它构成了规则的头部。$q_1,\cdots,q_n$ n个元素组成了规则的主体，其中任意的q_i可以是在相应域上的函数或者是谓词。这里谓词理解为数据库中的关系名，谓词q_i为真当且仅当数据库中存在该谓词对应的关系的元组。这些函数或是谓词在规则中出现的顺序与它们的执行顺序无关，它们之间的逗号表示逻辑与操作。也就是只有在所有函数中的布尔函数和谓词都满足的情况下，这条规则才被执行并推导出结果数据。在规则的执行过程中，函数中非布尔函数被执行。

下面利用路由协议中求最短路由的实例来介绍NDLog语言[10]。

2.2 程序实例

以下是由NDLog语言表示的求最短路由算法[14]：

$SP1$: $path(@S,D,P,C):-link(@S,D,C),$
$\quad P=f\_init(S,D).$

$SP2$: $path(@S,D,P,C):-link(@S,N,C1),$
$\quad Path(@N,D,P2,C2),C=C1+C2,$
$\quad P=f\_concatPath(S,P2).$

$SP3$: $spCost(@S,D,\min\langle C\rangle)$
$\quad :-path(@S,D,P,C).$

$SP4$: $shortestPath(@S,D,P,C)$
$\quad :-spCost(@S,D,C),$
$\quad path(@S,D,P,C).$

$Query$: $shortestPath(@S,D,P,C).$

以上4条规则表示了路由协议中最短路由的算法。其中@为地址标示符，它一般位于一个关系的地址属性前。当这个关系在规则体中时，地址标示符后面的属性标明这个关系的元组的实际存储地址；当这个关系在规则头部时，地址标示符标明这个关系将要生成的元组的存储地址。地址标示符的出现，扩展了Datalog已有的语言成分，为NDLog递归查询语言添加了表示分布式计算的能力。

SP1规则使网络中的每个节点根据本地链接表link生成由本地到邻居的平凡的路径元组并存储在本地的Path表中。SP2规则使本地节点根据其他节点已有的Path元组递归地生成到网络中其他目标节点的路径并存储在本地Path元组中。此外，规则中Path关系的第一个属性表示路径的源点，在这个属性前有地址标示符，所以每个本地生成的Path元组

都要被广播出去，为网络中的其他节点计算新的 Path 提供信息。在这个网络中我们可以假想各个节点规则的执行都有一个全局的轮数[14]。在第 i 轮 SP2 的计算利用第 $i-1$ 轮邻居生成的 Path 元组与本地 link 元组生成新的 Path 元组。SP3 规则根据本地存储的 Path 元组选出每对源点到终点路由的最小代价并将节点对以及代价保存在关系 spCost 中。SP4 规则根据已有的最小代价节点对元组和所有路径元组生成最短路径元组。Query 是数据查询语句，它从本地数据库的 shortestPath 关系表中抽取所需的信息。

OverLog 相对 NDLog 增加了对网络数据状态的描述，我们称之为数据的软状态。在分布式的网络环境中，已经存在的数据在网络中不断流动，新的数据不断产生，然而网络的整体状态随着节点和链接的变化而不断变化，不同的网络状态下，同一数据的功效性可能是不同的。为了区别在某一时刻特殊网络状态下的无效数据和有效数据，很多网络协议都给数据定义了相关属性来甄别它的有效与无效，比如数据的生存周期 TTL 属性。

OverLog 就是在 NDLog 语言的基础之上增加了对网络中数据软状态（生存周期）的描述，利用它来删除过期数据。若一个数据的生存时间超过其生存周期，那么它将被删除；如果这个数据在超过其生存周期前被节点重新生成，那么它的生存周期将被重置。

目前，P2P 协议 Chord 和 Narada 网状网络维护算法已用 OverLog 表示[11]，并在其支持系统 P2 上成功运行。

2.3 程序的编译与执行

由于 NDLog 语言和 OverLog 语言的编译执行类似[9,10]，本文以 OverLog 语言为例介绍它们语言支持系统 P2 的实现。

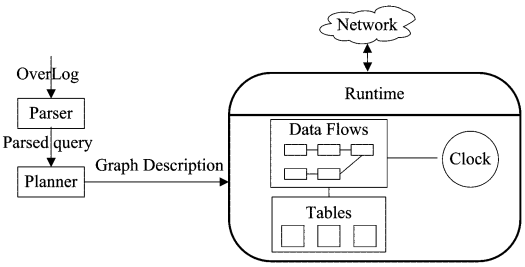

图 1　P2 系统结构[11]

图 1 是 P2 系统的体系结构。这里假设网络中的每个节点都装有 P2 系统，我们称之为 P2 节点。该系统的实现受数据库管理系统和分布式系统的启发，主要由 3 个主要部分组成：OverLog 语言的解析器（Parser）、数据查询策划器（Planner）和数据流执行系统（Runtime）。

P2 系统利用 C++ 语言实现，其中解析器负责将 OverLog 语言转换成一个正则表达式和一系列时间、规则和数据表。数据查询策划器负责将解析器的输出转化为无向数据流图并最终交给数据流执行器执行。

在 OverLog 程序转化成最终可执行的代码这个过程中，数据流图是最重要的结构。数据流图中的节点称之为数据流元素，它与数据库管理系统中的逻辑操作符类似，数据流元素表示对数据的操作，数据延数据流图中的边从一个节点流出（输出）进入另一个节点（输入），在这个过程中数据被数据流元素节点不断地改变直到成为最后的输出被系统存储或者发送。与传统的数据库数据流不同，为了处理递归规则，P2 系统中的数据流图往往是含有圈的[11]。P2 系统中数据流元素实现了数据库管理系统中的选择、投影、连接、分组以及集合操作。这些基本操作的实现依赖于 PEL 语言。PEL 是 P2 研究组在开发 P2 系统的过程中，为了简化开发过程而定义的基于堆栈的类逆波兰后缀语言。由此，每个数据流图中的数据流元素可以看作是一段 PEL 程序。

数据流图结构最终交给数据流执行系统执行。该子系统的实现基于 libsync 库，并遵循单线程、事件驱动模式。在这个模式下，在这个子系统中，所有事件组成一个队列，按照先进先出的规则依次执行。

2.4 小结

NDLog 和 OverLog 为网络协议设计者摆脱各种异构设备物理层的细节提供了可能的途径。但是，这两个语言的定义缺乏对语法、语义的形式化探讨。另外，在无线网络环境中，由于网络节点移动以及网络节点的失效，网络具有很强的动态性，然而这两个语言没有明确的语言成分用于对网络的动态性进行有效支持。

为了给无线网络协议设计者提供好的编程抽象，本研究组形式化定义了 Netlog 语言。

3 Netlog 语言介绍

本研究组已对 Netlog 语言语法、语义的进行了详细的讨论[13]，由于篇幅所限，本文不对此进行总结。

3.1 语言介绍

与 NDlog 类似，Netlog 沿袭了 Datalog 的风格，Netlog 程序由一系列规则组成。每个规则由规则头和规则体组成。下面通过路由协议中路由计算过程介绍 Netlog 语言。

$Route(x,y):-Link(x,y).$

$Route(x,z):-Route(x,y);Route(y,z).$

以上两规则通过本地邻居信息（Link(Self,Neighbor)表）和本地路由信息（Route(Src,Dest)表）递归地计算新的路由信息。在规则体中，关系名的出现顺序与执行顺序无关；分号表示逻辑与操作。另外，在 Netlog 的语言中，规则体只由关系名和布尔函数构成，一条规则被执行当且仅当本地节点收到或者产生符合规则体中某一关系模式的元组且规则体中的布尔函数被满足。

Netlog 程序在网络中的每个节点并发地执行，程序产生的数据存在本地数据库或者发送到其他节点。为了便于表示存储和发送数据操作，Netlog 引入了“存储”（↓）、“发送”（↑）和“存储与发送”（↕）操作符。此外，Netlog 也引入了地址标示符@。当地址标示符出现在规则头部的关系中时，它标明将要生成的元组的存储地址；当地址标示符出现在规则体的关系中时，在程序执行前，其后的属性被替换为节点本地地址。例如：网络中节点 A 的地址为 192.168.80.123，在节点 A 上运行以下 Netlog 程序：

$\updownarrow Route(x,z):-Route(@x,y);Route(y,z).$

在这条规则执行前，$@x$ 在节点 A 被替换成 192.168.80.123，新生成的 Route 元组存入本地数据库并广播出去。注意，在这条规则的头部没有地址标示符标明新元组的存储地址，此时默认动作是将新生成的元组向所有邻居广播。这条规则存在的一个问题是会重复计算一些路由元组，为了避

免重复计算，Netlog 引入否定操作符“¬”。

$$\updownarrow Route(x,z):-Route(@x,y);Route(y,z);\neg Route(x,z).$$

Netlog 对否定操作符的解释只限于本地，以 $\neg Route(x,z)$ 在节点 A 上为例，在这条规则中 $\neg Route(x,z)$ 为真当且仅当本地数据库中不存在由节点 A 出发带 z 代表的节点的路由元组。由此，这条规则避免了对本地数据库中已有路由元组的重复计算。

Netlog 支持赋值运算符(:=)、布尔运算符(<，>，<=，>=，==，!=)和集合操作符(min，max，avg，count)。此外，为了表示节点对网络无效数据的删除，Netlog 引入元组删除符号“!”。例如下面规则计算本地节点到其他节点间拥有最小跳数的路由。用到两个关系模式 SLength(Src，Dest，MinHops)和 WRoute(Src，NextHop，Dest，NumHops)。

$$\updownarrow SLength(x,z,\mathrm{Min}(n)):-WRoute(@x,y,z,n);!WRoute(@x,y',z,n');n'>n.$$

$!WRoute(@x,z,n');n'>n.$ 为本地节点 $@x$ 到属性 z 表示的每个非本地节点的路由选择唯一的具有若干极小跳数的下一跳 y。经过其他下一跳的路由在元组删除操作符的作用下被删除。在选定下一条 y 之后，Min(n)从通过 y 的具有极小跳数的若干路由中选出具有最小跳数的路由。

此外，在计算机网络中，同一个问题经常会面临几种不同的答案。比如从节点 A 到另外节点 X 可能存在若干条路由。这些路由可能经过相同的下一跳，也可能经过不同的下一跳。有时我们往往只需要一随机的路由，而不关心路由的质量。为此，Netlog 引入不确定性操作符“◇”。

$$\downarrow NextHop(\diamond y,z):-WRoute(@x,y,z).$$

这条规则利用不确定性操作符为本地节点随机选择到其他节点的唯一的下一跳。

目前，Netlog 已经成功表述了一系列网络问题：经典无线 Ad Hoc 网络协议(DSDV、OLSR 等)、分布式最大流算法、自定义 P2P 协议等。

3.2 支持系统 Netquest

Netlog 为网络协议设计者提供的编程抽象基于以下对网络的假设：网络中每个节点把除去本身的整个网络视为“数据库”，节点可以向“数据库”发送查询索要信息，并接受从“数据库”返回的答案；从而达到网络交换信息和数据的目的。图 2 显示了封装前后的网络。

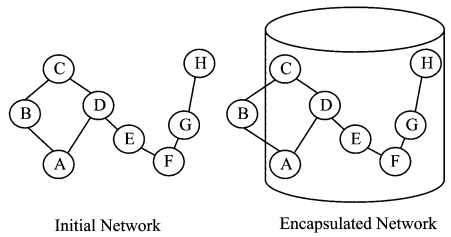

图 2 网络封装

为了处理数据的查询与接收，网络节点需要一个类似“数据库管理系统”的软件。为此，本研究组针对 Netlog 本身的特点开发了支持 Netlog 程序运行的 Netquest 系统。与 P2 系统不同，Netquest 系统构建于数据库管理系统之上，利用数据库存储 Netlog 程序和网络数据。在 Netquest 系统中，Netlog 程序首先被翻译成一系列 SQL 语句，而后存储在节点本地数据库中。

为了在网络中运行 Netlog 程序，每个节点都需要安装 Netquest 系统来参与其它节点发出的数据查询的计算。网络可以看作是关于数据计算的消息传递模型[15]。查询和数据都是以消息的形式在网络中传递。所有的节点都通过交换消息与其他节点实现数据的查询和交换。图 3 是 Netquest 系统的体系结构。

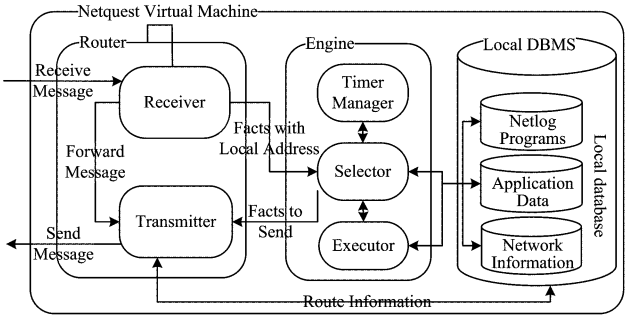

图 2 Netquest 系统体系结构

路由(Router)，主要负责本地节点和网络之间的通信：

- 它发送本地生成的消息；
- 转发目标节点非本地节点的消息；
- 将目标节点是本地节点的消息交给查询处理引擎(Engine)；
- 过滤掉冗余的消息。

查询处理引擎(Engine)，主要负责运行本地 Netlog 程序：

- 生成数据查询并将其交给路由(Router)发送到网络中去；
- 接受路由(Router)传递上来的数据，并做相应的计算。

本地数据库(Local DB)，主要负责存储 Netlog 程序和网络数据，为查询处理引擎提供数据查询处理的接口。

3.3 小结

目前，Netquest 系统已经成功地在基于事件驱动机制的无线网络模拟平台 WSNet[16] 上运行。其简化版本也已成功配置到智能手机以及无线传感器网络上去。

4 两类宣告式网络程序设计语言的比较分析

4.1 语法、语意

由于 NDLog 和 OverLog 在语法、语意上基本类似，OverLog 在 NDLog 基础之上添加了对数据软状态以及删除无效数据的支持，因此这里用 OverLog 作为其与 NDLog 的代表。

OverLog 和 Netlog 作为宣告式网络程序设计语言的代表，都是在已有的推导式数据库查询语言 Datalog 基础之上进行扩展，为网络协议设计者提供高层的编程抽象。这里，我们可以把这个高层编程抽象想象成用于网络协议开发的中间件。由 OverLog 和 Netlog 编写的网络程序只从高层描述其所要完成的网络中的各种行为，而不涉及程序的具体物理实现。

从语言角度，OverLog 和 Netlog 程序都是由一系列规则组成，每条规则由规则头和规则体组成。规则的执行均是在规则体中的关系或函数被满足的条件下，推导出规则头部关系元组。

不同的是，一方面，OverLog 的应用范围是表示传统的计

算机网络层和覆盖层协议。OverLog 语言本身是根据需要启发式定义的，没有经过形式化的语法、语意的讨论。另一方面，Netlog 的应用范围是无线网络，特别是无限 Ad Hoc 网络。Netlog 语言的定义是形式化的，是经过充分的语法语义讨论的。Netlog 语言中元组删除操作符和不确定性操作符为网络协议准确描述无线网络高度的动态性和不确定性提供了必要的支持。此外，否定操作符大大降低了节点的重复计算。

4.2 支持系统实现

OverLog 的支持系统 P2 和 Netlog 的支持系统 Netquest 在执行程序的时候都是改进已有的 Datalog 的执行技术 semi-naive 算法[17]。此算法在每轮计算的时候只利用上一轮新生成的元组作为输入，这样有效降低了不必要的重复计算。

在 P2 系统中，OverLog 首先被语言解析器翻译成系统内部定义的语言 PEL，而后 PEL 又被数据查询策划器翻译成数据流图的形式，最终数据流图被数据流执行器执行。综上，P2 系统是由 C++编写的包含 OverLog 语言的编译、优化和执行的系统软件。

在 Netquest 系统中，Netlog 程序首先被编译器翻译成等价的 SQL 语句，而后系统的查询处理引擎利用数据库管理系统控制执行这些 SQL 语句。因此，Netquest 系统可以看作是利用 C++实现的基于数据库管理系统的应用程序。

由于 Netquest 系统建立在数据库管理系统之上，在执行 Netlog 程序的过程中包含很多数据库的读写操作，因此其执行效率相对 P2 系统较低。然而，正因为 Netquest 系统建立在数据库管理系统之上，Netlog 的执行依赖数据库管理系统，所以只要网络节点上存在数据库管理系统，那么只要适当修改 Netquest 系统与数据库管理系统的接口部分，就可以将 Netquest 系统安装到节点上去，所以相对于 P2 系统，Netquest 系统具有较高的可移植性。

结束语 宣告式网络程序设计语言是一种利用演绎数据库技术描述并解决网络问题的方法。本文列举了具有代表性的宣告式网络程序设计语言，并对它们进行了比较分析。

此外与传统网络程序设计语言相比，宣告式网络程序设计语言具有以下优点：

简洁：声明算法往往只用几十条规则替代了传统算法的几百上千行的代码。

可扩展性：当网络环境改变时，升级传统的协议是一件很繁琐且容易出错的事情，而宣告式路由协议往往只需要更新规则就可以实现。

适应性：传统的路由协议只适用于某些定型的网络，随着无线网络的发展，越来越复杂的网路的出现，往往需要不停地更新协议。宣告式网络协议既可以处理网络信息的传输，又可以处理本地数据库的消息，它更能适应未来网络的发展。

可靠性：传统的协议，大多数是无法证明其可靠性的，而基于逻辑的宣告式路由协议则存在证明其可靠性的反例方法。

宣告式网络程序设计语言为网络的协议设计提供了高效、便利的前景。然而，目前宣告式网络程序设计语言只是一个崭新的方向，还有很多问题值得继续研究：

- 针对不同的网络环境，定义不同的宣告式网络程序设计语言，并在语法、语意上进行详细探讨。
- 语言支持系统的优化，比如利用数据流图的共享实现多层协议的优化以及设计有效的 Netlog 程序到 SQL 语句的翻译方案，充分利用数据库管理系统的优化器进行优化。
- 宣告式网络程序设计语言的支持系统在无线传感器上的配置，也对无线传感器上操作系统和数据库管理系统的设计与实现提出了挑战。

参 考 文 献

[1] Marron P J, Minder D. Embedded WiSeNts Research Roadmap [C]// Embedded WiSeNts Consortium. 2006:127

[2] Ramakrishnan R, Gehrke J. Database Management Systems[M]. Mc. Graw Hill, 2003:15

[3] Madden S, Franklin M J, Hellerstein J M, et al. Tinydb: An Acquisitional Query Processing System for Sensor Networks[J]. ACM Trans. Database Syst., 2005, 30(1):122-173

[4] Demers A, Gehrke J, Rajaraman R, et al. The Cougar Project: A work-in-progress report[J]. ACM SIGMOD Record, 2003, 32: 53-59

[5] Srivastava U, Munagala K, Widom J. Operator Placement for In-network Stream Query Processing[C]// PODS. 2005:250-258

[6] Jeffery S R, Alonso G, Franklin M J, et al. Declarative Support for Sensor Data Cleaning[C]// Pervasive. 2006:83-100

[7] Alonso G, Kranakis E, Sawchuk C, et al. Probabilistic Protocols for Node Discovery in Ad Hoc Multi-channel Broadcast Networks[C]// ADHOC-NOW. 2003:104-115

[8] Bejerano Y, Breitbart Y, Orda A, et al. Algorithms for Computing QoS Paths with Restoration[J]. IEEE/ACM Trans. Netw, 2005, 13(3):648-661

[9] Bejerano Y, Breitbart Y, Garofalakis M, et al. Physical Topology Discovery for Large Multi-subnet Networks[C]// Proc. IEEE Infocom. 2003:342-352

[10] Loo B T, Condie T, Garofalakis M, et al. Declarative Networking: Language, Execution and Optimization[C]// Proceedings of the ACM SIGMOD International. Conference on Management of Data. Chicago, Illinois, USA, June 2006:27-29

[11] Loo B T, Condie T, Hellerstein J M, et al. Implementing Declarative Overlays[C]// SOSP. 2005:75-90

[12] Bauderon M, Bobineau C, Grumbach S, et al. Netquest: An Abstract Model for Pervasive Applications[C]// Pervasive 2009, the 7th International Conference on Pervasive Computing

[13] Grumbach S, Wang Fang, Wu Zhilin. On the Distributed Evaluation of Recursive Queries Over Graph[C]// NOTERE 2009, the 9th Annual International Conference on New Technologies of Distributed System

[14] Loo B T, Condie T, Garofalakis M, et al. Declrative Networking [J]. Communication of the ACM, 2009, 52(11):97-108

[15] Attiya H, Welch J. Distributed Computing: Fundamentals, Simulations and Advanced Topics [J]. John Wiley Interscience, March 2004:9-25

[16] Chelius G. Worldsens: a fast and accurate development framework for sensor network applications[C]// Proceedings of the 2007 ACM symposium on applied computing. 2007:222-226

[17] Abiteboul S, Hull R, Viano V. Foundations of Databases[Z]. Addison-Wesley, 1995:312-316

[18] Thau B, Joseph L, Hellerstein M, et al. Declarative routing: Extensible Routing with Declarative Queries[C]// Proceedings of ACM SIGCOMM. 2005:289-300

[19] Ramakrishnan R, Ullman J D. A Survey of Deductive Database Systems[J]. J. Log. Program, 1995, 23(2):125-149