

一种基于 Agent 的 Ad hoc 网络路由算法

邵 星,王汝传,徐小龙

(南京邮电大学 信息网络研究所,江苏 南京 210003)

摘 要:Ad hoc 网络因其具有分布式、无中心、自组织、节点可以移动等特点,在军事通信、灾后紧急救援、传感器网络、局域网、车辆通信等方面有着广阔的应用前景,成为研究领域的一个热点。但同时由于 Ad hoc 网络拓扑结构的动态变化,使得作为 Ad hoc 网络关键技术之一的路由算法的实现较为困难。提出了一种基于 Agent 的 Ad hoc 网络路由算法,设计并实现了 4 种 Agent。该算法通过在 Ad hoc 网络中加入一定数量的移动 Agent 来进行路由探寻,一方面降低了网络负载,另一方面降低了网络发送数据的时延。其实质是在现有的表驱动路由算法和按需驱动路由算法之间寻求一个折中。

关键词:Ad hoc 网络;路由算法;移动 Agent

中图分类号:TP301.6

文献标识码:A

文章编号:1673-629X(2010)04-0021-04

A Kind of Agent - Based Network Routing Algorithm for Ad hoc Networks

SHAO Xing, WANG Ru-chuan, XU Xiao-long

(Institute of Information Network Technology, Nanjing University
of Posts and Telecommunications, Nanjing 210003, China)

Abstract:Ad hoc network for its characters of distributed, no center, self-organization and mobile nodes, has a broad application prospects in military communications, post-disaster emergency relief, sensor networks, local area networks and communications of vehicles, and it has become a hot spot of research field. However, the dynamic changing of the topology of Ad hoc network, makes routing algorithm, a key technology of Ad hoc network, more difficult to achieve. Presents an agent-based Ad hoc network routing algorithm, besides it designs and realizes four kinds of Agents. By adding a certain number of mobile agents into the Ad hoc network to explore routing information, the algorithm, on the one hand, reduces the loads of Ad hoc network, on the other hand it reduces the delay time of data sending of the Ad hoc network. Its essence is to find a compromise between the existing table-driven routing algorithm and on-demand-driven routing algorithm for Ad hoc network.

Key words:Ad hoc network; routing algorithm; mobile Agent

0 引 言

目前,Ad hoc 网络技术已经成为一个研究热点。但同时,由于 Ad hoc 网络拓扑结构动态变化,使得作为 Ad hoc 网络关键技术之一的路由算法的实现较为困难^[1,2]。目前基于 Ad hoc 网络的路由协议主要有两

种:表驱动路由协议;按需驱动路由协议^[3]。但两者在 Ad hoc 网络中的应用均不是十分理想^[4,5]。近年来,移动 Agent 技术得到广泛的关注,移动 Agent 具有移动性、能动性、反应性、自主性、通信性等特性^[6],成为研究领域的另一个热点^[7~9]。

文中创新点在于,提出了一种基于 Agent 的 Ad hoc 网络路由算法,设计并实现了 4 种 Agent。该算法通过在 Ad hoc 网络中加入一定数量的移动 Agent 来进行路由探寻^[6],一方面降低了网络负载,另一方面降低了网络发送数据的时延。其实质是在现有的表驱动路由算法和按需驱动路由算法之间寻求一个折中。

1 基于 Agent 的 Ad hoc 网络路由算法

1.1 路由算法

文中提出的基于 Agent 的 Ad hoc 网络路由算法,

收稿日期:2009-08-04;修回日期:2009-11-09

基金项目:国家自然科学基金(60973139,60773041);江苏省自然科学基金(BK2008451);现代通信国家重点实验室基金(9140C1105040805);国家和江苏省博士后基金(0801019C,20090451240,20090451241);江苏高校科技创新计划项目(CX08B-085Z,CX08B-086Z);江苏省六大高峰人才项目(2008118)

作者简介:邵 星(1985-),男,江苏宿迁人,硕士研究生,研究方向为计算机网络、对等计算、卫星通信网络技术等;王汝传,教授,博士生导师,研究方向是计算机软件、计算机网络和网络、对等计算、信息安全、无线传感器网络、移动代理和虚拟现实技术等。

基于按需驱动路由算法,其实质是通过在网络中加入一定数量的移动 Agent,将路由算法封装到移动 Agent 中,利用移动 Agent 的自主性、反应性、能动性,让移动 Agent 自主地动态更新和维护 Agent 途径节点的路由表,提高节点对网络的认知度。从而可以提高发送数据时,路由选择的成功率。实现表驱动路由算法和按需驱动路由算法的折中。

为此,该算法设计了 4 种 Agent:

NodeAgent: 固定 Agent (stationary Agent)。每个节点创建一个 NodeAgent,用于存储节点的路由信息,并向移动 Agent 提供外部访问和更新节点路由信息的接口。

HelloAgent: 移动 Agent (mobile Agent)。用于节点实时探查相邻节点的链路通断。

FloodAgent: 移动 Agent (mobile Agent)。当需要向某个节点发送数据,而源节点没有到目的节点的路由信息时,在源节点创建 FloodAgent,用于泛洪探寻源节点到目的节点的路由信息。

RouteAgent: 移动 Agent。网络中的每个节点,周期性地创建 RouteAgent 用于探寻到网络中其他节点的路由信息。

1.2 路由节点的设计

路由节点中用于存储其信息的有路由表和相邻节点表。

路由表中包含三个表项:“目的节点”,用于存储该条路由信息的目的节点;“跳数”用于存储选择该条路由信息时,需要经过的跳数;“经过路径”用于存储选择该条路由信息时,应该依次经过哪些节点,即经过节点所连成的路径。路由表结构如表 1。

表 1 路由表结构

目的节点	跳数	经过路径
NodeId	Hopcounter	NodeId1 - NodeId2 - NodeId3 -
.....

由于 Ad hoc 网络的拓扑结构是动态变化的,路由信息也是经常变化的。为此,为每条路由信息设置了一个有效时间。在超过有效时间以后,该条路由信息即宣告无效。

相邻节点表中存储的是与本地节点存在直接链路的相邻节点的 ID 信息。相邻节点表结构如表 2。

表 2 相邻节点表结构

相邻节点 I	相邻节点 II	相邻节点 III	相邻节点 IV	相邻节点 V
NodeId	NodeId	NodeId	NodeId	NodeId

1.3 NodeAgent 的设计

NodeAgent 是一个固定 Agent (stationary Agent)。网络中每个节点在初始化的时候,都会创建一个

NodeAgent,用于存储节点的相关信息。NodeAgent 包含两张表:相邻节点表和路由信息表,这两张表存储节点的路由信息。另外 NodeAgent 存储有本地节点的 ID 标识。

相邻节点表存储与本地节点存在无线链路的相邻节点的 ID 标识。所有节点的相邻节点表实际上反应了整个网络的拓扑信息。路由信息表存储从本地节点到网络中其他节点的路由信息,包括到某个节点的代价、路径。

此外,NodeAgent 向外部移动 Agent,提供了访问和更新 NodeAgent 中相邻节点表和路由表的接口。

1.4 HelloAgent 的设计

Ad hoc 网络所有节点的相邻节点表反映了整个网络的拓扑结构,而 Ad hoc 网络节点的不停移动,使得 Ad hoc 网络的拓扑结构不断地变化。因此,每个节点必须实时地更新相邻节点表,以使得它们能够实时正确地反映网络的拓扑结构的变化,而这也是得到正确路由的前提。HelloAgent 就是基于上面的考虑而提出和设计的。HelloAgent 就是每个网络节点用于探查网络节点之间链路状况的移动 Agent。

HelloAgent 的迁移过程如下:

1) 每个节点周期性地创建 HelloAgent。HelloAgent 进行初始化时,在其自身空间中,存储创建它的本地节点的 ID 标识(sID)。

2) HelloAgent 通过无线信道,向节点信号覆盖的范围泛洪迁移。

3) 网络中的每个节点,当发现有 HelloAgent 到达时,读取 HelloAgent 存储的源节点的 ID 标识(sID),查看 sID 是否已经在其 NodeAgent 的相邻节点表中出现。如果没有出现过,表明源节点和该节点之间建立了链路,则将 sID 添加到其相邻节点表中。否则,不对相邻节点表操作。

4) HelloAgent 读取到达节点的 ID 标识(dID),存储到其自身空间中。

5) HelloAgent 迁移回到创建它的源节点。察看存储的到达节点的 ID 标识是否已经在该节点的 NodeAgent 的相邻节点表出现过。如果没有出现过,则将该 ID 标识添加到相邻节点表中。如果已经出现过,不对相邻节点表操作。至此,HelloAgent 的任务完成,立即自杀(remove)。

如果在一个规定的时间 T 内,没有从相邻节点表中的某个节点迁移返回得到 HelloAgent,则认为本地节点与该相邻节点之间的链路已经断开,将该节点的 ID 从相邻节点表中删除。

HelloAgent 虽然其迁移控制在一跳的范围内,但

是由于是在每个节点的广播发送,其给网络带来的负载还是有一定程度的。为此,必须根据网络的拓扑结构的变化速率,调整 HelloAgent 的生成速率。在网络负载和网络拓扑的实时性之间做出一个平衡。

1.5 FloodAgent 的设计

文中提出的算法基于按需驱动路由算法。FloodAgent 就是实现了按需驱动路由算法中,用于泛洪发现路由的移动 Agent。

当网络中某个节点向网络中另外一个节点发送数据时,发现在源节点的路由表中没有到目的节点的路由信息。于是,在源节点创建生成 FloodAgent, FloodAgent 向目的节点泛洪迁移进行路由信息的探寻过程。FloodAgent 在初始化时,在内部开辟一定空间用于存储迁移过程中得到的路由信息。初始时, FloodAgent 在内部空间创建了一个空堆栈,并将源节点(sID)和目的节点(dID)的 ID 存入内部存储空间中,另外设置一个整数值 MaxHop(最大跳数值)。如果目的节点刚好是源节点的相邻节点,则路由信息直接获得,不需要 FloodAgent 的探寻过程。

FloodAgent 在向目的节点方向泛洪迁移的过程中,每到达一个节点,首先将该节点的 ID,连同到达该节点的跳数,压入到自身堆栈中。然后 FloodAgent 的处理过程如下:

- 1) FloodAgent 检查目的节点是否是此节点的相邻节点,如果是,则直接向目的节点迁移。

- 2) FloodAgent 发现目的节点不是此节点的相邻节点,则 FloodAgent 查找该中间节点的相邻节点表,找出没有在其堆栈中出现的相邻节点(避免环路)。FloodAgent 向得到的这些相邻节点泛洪迁移。如果没有找到这样的相邻节点,则认为该 FloodAgent 此次路由探寻失败,自动 remove。

这里移动 Agent 所谓的泛洪迁移,实际实现方法是 FloodAgent 为每个相邻节点(可能形成环路的相邻节点除外)复制一个自身。然后让这些复件向各个相邻节点迁移,而父本则 remove,从而实现 FloodAgent 的泛洪迁移。

FloodAgent 在泛洪迁移的过程中,如果所经过的跳数超过了 MaxHop,则认为该 FloodAgent 失效,该 FloodAgent 自动 remove。从而可以有效控制 FloodAgent 的泛洪范围,控制和降低网络负载。

由于泛洪的存在,所以在网络中会产生很多具有相同目的节点和起点的 FloodAgent,这就引发了竞争的问题。竞争问题的解决办法是:第一个到达目的节点的 FloodAgent 所经过的路由必定是最优的,所以目的节点仅处理最先到达的 FloodAgent。第一个到达目

的节点的 FloodAgent 修改目的节点的状态标志,使得后续到达的 FloodAgent 根据状态标志自动 remove。

然后该 FloodAgent 将目的节点的 ID,连同到达该节点的跳数,压入到自身堆栈中。随后 FloodAgent 根据其堆栈中的信息,沿着原路径返回,最终到达源节点。在源节点根据其自身堆栈中的内容,更新源节点到目的节点的路由信息。然后节点根据得到的路由信息向目的节点发送用户数据。

1.6 RouteAgent 的设计

创建 FloodAgent 进行泛洪迁移来探寻路由信息,是在需要发送数据的源节点没有到目的节点的路由信息的情况下被迫所采取的方法。按需驱动路由算法与表驱动路由算法相比,可以较好地降低网络负载。但是一定程度上来说,泛洪方法的使用不仅增加了网络负载,而且也增加了数据发送的延时。

如果在网络中创建一定数量的移动 Agent,让这些移动 Agent 在网络运行的正常状态下,周期性地探寻到其他节点的路由信息,则可以增加用户数据发送时路由信息获取的成功率。从而降低数据发送延时的平均水平,避免泛洪导致网络一时的负载加重甚至拥塞的可能性,实现表驱动路由算法和按需驱动路由算法的折中。RouteAgent 就是基于上面的考虑而提出和设计的。

网络中的每个节点在网络的正常运行状态下,周期性地生成以网络中的其他节点作为目的节点的移动 Agent,即 RouteAgent。这个目的节点,是在网络的所有节点中等概率地随机选择得到的。

RouteAgent 的内部结构和 FloodAgent 相同。在初始化时,RouteAgent 在内部开辟一定空间用于存储迁移过程中得到的路由信息。初始时,RouteAgent 在内部空间创建了一个空堆栈,并将源节点(sID)和目的节点(dID)的 ID 存入内部存储空间中,另外设置一个整数值 MaxHop(最大跳数值)。如果目的节点刚好是源节点的相邻节点,则路由信息直接获得,不需要 RouteAgent 的探寻过程。另外在源节点为 RouteAgent 设置了有效时间 t 和生存周期 T 。

由于在生成 RouteAgent 的节点没有到目的节点的用户数据等待发送,基于节省网络带宽资源的考虑,RouteAgent 的迁移方式采用自主选择方式。即每到达一个中间节点,让 RouteAgent 自主选择下一跳节点。

RouteAgent 在向目的节点迁移的过程中,每到达一个节点,首先将该节点的 ID,连同到达该节点的跳数,压入到自身堆栈中。然后 RouteAgent 的处理过程如下:

- 1) RouteAgent 检查该中间节点是否有到目的节

点的路由信息,如有,则 RouteAgent 将把该中间节点到目的节点的路由信息读取到自身存储空间中,然后根据自身内部的堆栈信息,向源节点方向迁移返回。

2) RouteAgent 发现该中间节点没有到目的节点的路由信息,则 RouteAgent 查看目的节点是否是该节点的相邻节点。如果是,则 RouteAgent 直接向目的节点迁移。

3) RouteAgent 发现该中间节点没有到目的节点的路由信息,并且目的节点不是该中间节点的相邻节点,则 RouteAgent 查看该中间节点的相邻节点表,找出没有在其堆栈中出现的相邻节点(避免环路),然后在其中随机选取一个作为下一跳节点。

4) 如果没有 3) 中这样的相邻节点,则 RouteAgent 目前所在的节点选择失误。RouteAgent 需要根据堆栈中的信息向后面的节点回退,直到达到一个可以得到有效下一跳节点为止,以继续进行路由探寻过程。如果回退到了源节点,则认为 RouteAgent 此次路由探寻过程失败,RouteAgent 自动 remove。

RouteAgent 向目的节点迁移过程中,在到达目的节点以及每个中间节点时,都会根据自身堆栈中的信息更新该节点到 RouteAgent 堆栈中的各个节点(即 RouteAgent 经过的各个节点)的路由信息,称为反向路由更新。

RouteAgent 到达目的节点后,向源节点逆向迁移返回的过程中,每到达一个节点都更新该节点到目的节点的路由信息。最终到达源节点更新源节点到目的节点的路由信息。这个过程,称为正向路由更新。

RouteAgent 在向目的节点迁移的过程中,如果所经过的跳数超过了 MaxHop,则认为该 RouteAgent 失效,该 RouteAgent 自动 remove。

RouteAgent 的一次探寻过程中,如果源节点发现有效时间 t 之内,RouteAgent 不能返回,则认为此次 RouteAgent 探寻失效,重新生成 RouteAgent 进行新一轮的探寻。

如果在生存周期时间 T 内,RouteAgent 返回,则 RouteAgent 开始到原来那个目的节点的新一轮的路由探寻,即选择不同路径,进行新的探寻。所以最终源节点可以得到不同的路径,从中选择最优的路由信息。

生存周期时间 T 结束后,源节点宣布之前生成的 RouteAgent 无效,生成一个新的 RouteAgent,选取不同节点作为目的节点,开始新的探寻过程。

2 原型系统设计与实现

为验证该算法的正确性与可行性,文中设计了一个网络模拟环境,以及用于验证算法的原型系统。原

型系统可以控制和显示 Agent 的创建和迁移过程,并能够实时显示节点的路由信息。

如图 1 所示,文中设计了一个包含 7 个节点的小型网络,通过节点加入和移出来模拟 Ad hoc 网络。

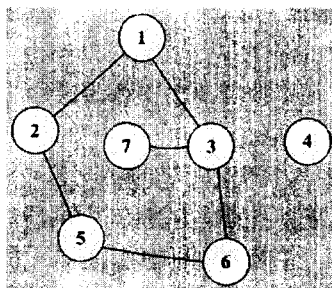


图 1 模拟网络初始拓扑

在此基础上,文中开发了用于验证算法的软件原型系统。网络中的每个节点在有移动 Agent 对其路由信息进行更新后,都自动显示更新后的节点路由信息,并可在原型系统界面中实时显示。网络拓扑结构的辩护与节点的加入和删除,反映在节点的相邻节点表中。

当 RouteAgent 在探寻过程中出现环路,系统会弹出对话框,如图 2 所示。从而提醒用户,对 RouteAgent 的回溯过程进行跟踪,验证算法运行的正确性。

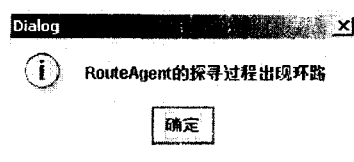


图 2 RouteAgent 环路提醒

原型系统对文中所提算法的模拟结果表明,该算法具有可靠的正确性与可行性。

3 结束语

文中介绍了 Ad hoc 网络路由算法的研究现状以及 Agent 技术的特点和优势,提出了一种基于 Agent 的 Ad hoc 网络路由算法,设计了 4 种 Agent 来实现算法的各种功能,并对各种 Agent 的详细实现方法进行了具体的描述。该算法实质上是通过对 FloodAgent 和 RouteAgent 的路由探寻来实现表驱动路由算法和按需驱动路由算法的折中。HelloAgent 的生成速率、FloodAgent 和 RouteAgent 的 MaxHop、RouteAgent 的有效时间 t 和生存周期 T , 这些具体的参数还需要根据网络的具体情况进行具体地调整,以达到最佳的效果。该算法理论上可以实现表驱动路由算法和按需驱动路由算法的折中,即同时具有较低的数据发送延时和相对较高的路由获取成功率。下一步的研究重点是,针对各种不同的 Ad hoc 网络的具体环境,发现和调整算法

(下转第 28 页)

针对不同的安全等级,可以采用不同的安全策略。

* 同一应用域中,可以包含多个线程。Elastos 应用的消息处理机制采用的就是这一机制。

* 域的错误或异常不会影响到其它的域或者整个进程。由于域是独立的,域之间的错误或异常会保持相互的独立。

* 每个域的配置信息是该域的一部分,而不属于该域所在的进程。

* 一个域中的程序终止的时候,不影响同一进程中的其它域的状态。

构件 domain 技术通过在同一进程中设置多个域,来减小应用切换带来的系统消耗,提高应用的安全性,保证应用程序能安全有效的运行。通过构件 domain 技术,可以将调试范围定位到某一个构件的内存中,方便调试。

5 基于伪驱动的内核窥探调试

一个普通的程序通常只能运行在用户态,但就调试而言,这是不够的。有时,需要获取更多的内核信息,比如:线程局部存储(TLS)的当前值,内核的一些信息。而这些信息只能通过处在内核态的线程才能取得。为了能做到这一点,需要向内核中派个亲信,称之为伪驱动。

Elastos 系统中驱动程序^[8]有较高的权限,已安装使用的驱动可以获取内核的所有信息。正因这样,可以编写一个伪驱动,让其运行于内核态,获取无法从用户态获取的信息,帮助更好的调试。调试流程如下:

- * 运行应用程序、调试程序和伪驱动;
- * 当应用程序需要获取内核态的信息时,则通过

调试程序调用伪驱动的相关接口,以获取相应的内核信息,用户可以根据这些信息,观察内核中的信息是否异常,以采取进一步的调试策略。

6 结束语

在 Elastos 内存管理体系结构中,利用栈、堆、构件 domain 和伪驱动的内核窥探技术,对软件的调试提供良好的支持。通过这些支持,用户可以在开发 Elastos 程序中,方便地定位程序在内存方面的错误。同时,用户亦可以在此基础上,结合 Elastos 平台的其它特性,增强对程序的调试能力,提高软件的可靠性。

参考文献:

- [1] Koretide. CAR's Manual [EB/OL]. 2009-07. <http://www.koretide.com.cn>.
- [2] 王 铮,李志军.一种适用嵌入式系统的自适应动态内存管理方案[J].计算机技术与发展,2007,17(3):48-54.
- [3] 王明路,王希敏,王 哲.嵌入式系统中池式内存分配方法的分析[J].计算机与数字工程,2008(2):57-61.
- [4] Fog A. Calling conventions for different C++ compilers and operating systems[M]. [s.l.]:[s.n.],2009:15-23.
- [5] 袁 宁.基于共享主存计算机的含错与动态检查点技术研究[D].长沙:国防科学技术大学,2006:42-50.
- [6] Chen Rong. The Application of Middleware Technology in Embedded OS[C]//Workshop on Embedded System, in Conjunction with the ICYCS(6th). Hangzhou:[s.n.],2001.
- [7] 白铁荣,陈 榕. Elastos 平台上可执行文件的三种入口规范[J].计算机技术与发展,2008,18(11):220-222.
- [8] 杨向科,陈 榕.基于 Elastos 的构件化驱动编程模型的研究[J].计算机技术与发展,2008,18(12):25-31.

(上接第 24 页)

的参数规律,进一步地优化算法,并且给出算法的仿真结果。

参考文献:

- [1] IETF. Mobile Ad hoc Network Chapter [EB/OL]. 2003. <http://www.ietf.org>.
- [2] 董传杰,王汝传. Ad hoc 网络路由问题的研究[J].南京邮电学院学报,2005,25(3):67-72.
- [3] Charles E, Perkins, Elizabeth M, et al. Ad hoc on Demand Distance Vector Routing [C]// Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications. Los Alamitos, CA, USA: [s. n.], 1999: 90-100.
- [4] Ehsan H, Uzmi Z A. Performance Comparison of Ad hoc Wireless Network Routing Protocols [J]. INMIC IEEE,

2004,9(4):450-465.

- [5] Corson S, Macker J. Mobile Ad hoc networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations[S]. RFC 2501, 1999.
- [6] Caro G D, Dorigo M. An adaptive multi-agent routing algorithm inspired by ants behavior [C]//In: Fifth Annual Australasian Conference on Parallel and Real-Time Systems. Adelaide, Australia: [s. n.], 1998: 28-29.
- [7] Foundation for Intelligent Physical Agents (FIPA) [EB/OL]. 2005. <http://www.fipa.org>.
- [8] 王汝传,徐小龙,黄海平.智能 Agent 及其在信息网络中的应用[M].北京:北京邮电大学出版社,2006.
- [9] Wooldridge M. An Introduction to Multi-agent Systems [M]. Chichester, England: John Wiley & Sons, 2002: 10-31.