一种 k 跳分簇 Ad Hoc 网络协作框架

李 勇 王 平 潘 勇
(重庆邮电大学网络化控制与智能仪器仪表教育部重点实验室 重庆 400065)
(重庆邮电大学自动化学院 重庆 400065)

摘 要 基于 k 跳簇的特殊结构,提出了一种 k 跳分簇 Ad Hoc 网络协作框架。簇首根据启发式规则选择合适的簇间协作网关节点,自适应地管理簇间协作关系。相邻的协作"网关节点"与簇首协作,管理簇内节点、网关节点的移动。当簇首发生移动时,选择合适的节点完成簇首信息的交接。移动代理在簇首节点间漫游,实现全网络协作,扩大 k 跳簇首的知识范围。仿真结果表明,协作框架能够以较低的控制负载高效地管理 k 跳分簇 Ad Hoc 网络中的移动节点,并提供网络级的协作。

关键词 k 跳簇,Ad Hoc 网络,协作框架,移动代理,控制负载
中图法分类号 TP393 **文献标识码** B

Cooperation Framework for k-hop Clustered Ad Hoc Networks

LI Yong WANG Ping PAN Yong
(Key Laboratory of Network Control & Intelligent Instrument, Ministry of Education, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)
(College of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

Abstract Based on the special structure of k-hop cluster, presented a kind of cooperation framework for k-hop clustered Ad Hoc networks. Cluster heads utilize heuristic rules to select cooperative gateway nodes and set up adaptive inter-cluster cooperation. Neighboring gateway nodes cooperate with cluster-heads to manage the mobile cluster members and gateway nodes. When cluster head is moving, it transmits information to an optimal replacing cluster head selected by cluster head itself. Mobil agents roam between cluster heads of network and establish a cooperation of network scale. By this way, knowledge ranges of cluster head is enlarged efficiently. Simulating results indicate that this cooperation framework can manage the mobile nodes of k-hop clustered Ad Hoc networks and provide network scale cooperation efficiently with moderate control overhead.

Keywords k-hop cluster, Ad Hoc networks, Cooperation framework, Mobile agent, Control overhead

1 引言

Ad Hoc 网络是由大量同构或者异构的资源和能量受限的可移动无线节点构成的网络。随着物联网时代的到来,Ad Hoc 网络将获得宽广的应用领域。由于节点的通信范围有限,节点间的通信大多采用多跳的方式,因此 Ad Hoc 网络中节点的协作十分重要。目前对 Ad Hoc 网络专门的协作研究主要集中在针对自私节点的协作激励机制上面,如基于代价的方法[1-3]和基于信誉的方法[4]。文献[5]研究了寻路阶段自私节点的协作激励机制。而在 Ad Hoc 网络未来的主要应用领域,如救灾、工业自动化、移动机器人、战场等,网络中的节点都是为了实现相同的分布式应用而部署,因此几乎不存在自私节点。对这样的专用 Ad Hoc 网络进行协作主要是为了降低能量消耗,实现拓扑控制、睡眠调度、负载均衡、攻击检测、分布式系统管理等复杂的功能。

一定网络范围内节点的协作,目前采用的方式主要有按需(洪泛)和分簇两种类型。文献[6]采用了各个节点向 k 跳邻居发送信息的形式实现节点间的协作,研究了对网络节点休眠调度、故障易发网络路由、移动 sink 路由等问题进行协作的代价与性能折中问题。由于缺乏有效的组织方式,产生的控制负载过大,作者指出这样的 Ad Hoc 网络协作会产生大量的控制负载,必须仔细进行权衡。

网络分簇实际上是簇首与簇成员间的一种协作关系。簇首节点负责管理簇成员节点,为成员节点提供分级路由、数据融合;簇成员节点轮流担当簇首。1 跳分簇网络,无论是成簇算法还是簇的维护都比较简单。1 跳分簇的最大缺点是扩展性差,当网络规模扩大时,簇首数量大。此外 1 跳分簇也限制了节点间协作的灵活性。

k 跳分簇是指网络中所有节点离簇首的跳数最大为 k 跳,当 k 大于 1 时,为多跳分簇。文献[7]提出了一种快速收

到稿日期:2010-05-01 返修日期:2010-07-26 本文受重庆市自然科学基金项目(CSTC,2009BB2418),重庆邮电大学博士启动基金(2007-13A)资助。

李 勇(1976—),男,博士,副教授,主要研究方向为工业无线网络、自适应网络,E-mail:023liyong@163.com;王 平(1963—),男,博士,教授,主要研究方向为网络控制。

致的 Ad Hoc 网络 Max-Min 启发式 k 跳分簇算法。文献[8]在此基础上，针对节点移动可能造成的簇重构，提出了新的簇首评价指标，使得簇稳定时间变长。

k 跳分簇与 1 跳分簇相比，网络的扩展性更好，簇首有 k 跳范围内节点的信息，因此簇首可以发挥更好的协作，为实现更高性能的路由、管理、拓扑控制等算法提供基础。文献[9]研究了基于分簇网络的拓扑控制框架，采用了簇内集中、簇间分布的混合拓扑控制算法对 Ad Hoc 的网络进行拓扑控制，保证簇内和簇间的连通性。

这些算法的研究与实现都涉及到了节点间的协作，但是对 Ad Hoc 网络的协作进行专门的研究，为上述功能提供统一的协作框架尚未有文献。

进行管理和控制的基础是能够准确获取、及时反馈网络的状态信息，因此必须对分簇 Ad Hoc 网络进行高效的维护。

Ad Hoc 网络中的节点移动会使得网络的拓扑发生变化，簇成员信息也会发生变化。对于 1 跳簇的维护相对简单，节点周期性地向簇首发送报文，报告自己的状态，簇首在一段时间内没有收到节点的消息就将节点从成员列表中删除。当节点与簇首失去联系时，节点加入新的簇。

采用周期性的通信的方法进行簇维护，虽然简单，但是会消耗大量的网络资源，产生大量的控制负载，并且很难实现自适应。对 k 跳分簇 *Ad Hoc* 网络采用网关节点周期性地向簇首发送报文的形式进行簇的维护。中间节点在收到网关节点的报文时，将自己的信息添加到报文里。簇首根据接收到的信息更新簇成员。除了能耗高外，这种方法的另一缺点是当网关节点离开簇时，簇首节点会误以为中间节点也离开了簇。

对 k 跳簇的维护，文献[8]并未进行专门的研究，仍然采用与 1 跳簇类似的方法，虽然提出了簇首节点切换的方法，但是没有提出具体的方法。

k 跳簇的维护与 1 跳分簇相比具有特殊性。本文为了在 k 跳分簇 Ad Hoc 网络上实现节点间的协作，为更高性能的算法研究和实现提供基础框架，针对 k 跳簇维护的特点，将节点间的协作引入到簇维护中。首先针对节点的各种运动情况提出了相应的低控制负载移动节点管理算法；在保证分簇结构稳定的前提下，采用移动代理扩大簇间协调的范围，实现高效的 k 跳分簇 Ad Hoc 网络协作。簇维护和簇间协作结合，构成较为完善的基于 k 跳分簇的 Ad Hoc 网络协作框架。

2 k 跳分簇网络维护

2.1 k 跳分簇网络模型

k 跳分簇 Ad Hoc 网络中，所有成员距离簇首的最大跳数为 k 跳，图 1 所示为 2 跳分簇网络。文献[8]中将簇边界上与其他簇边界节点间存在链接的节点称为网关节点(Gateway)。然而节点移动可能会导致普通边界节点成为网关节点，因此本文将簇边界节点统称为网关节点。

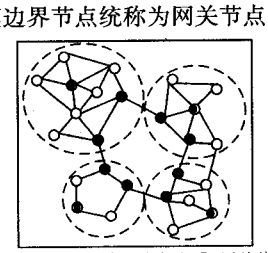

图 1　2 跳分簇 Ad Hoc 网络

Ad Hoc 网络可以被表示为无向图 $G=(N,E)$，其中 N 是无线节点集合，$E\subseteq V\times V$ 是节点间链路的集合。在彼此通信范围内的节点建立相互间的无线链路。C_i 表示以节点 $n_i\in N$ 为簇首的 k 跳簇。网络中所有节点的通讯范围同为 R 单位长度，网络的范围为 D 单位长度。

作为协作的基础，本文假设 k 跳分簇 Ad Hoc 网络中的节点，除了节点 ID、簇首 ID 之外还具有距离簇首的跳数信息。通过向邻居节点广播 Hello 包，节点可以被动获得 1 跳范围内邻居节点的信息。

2.2 移动节点管理

节点随机移动使得簇成员与簇首间的多跳链接中断，改变网络拓扑。Ad Hoc 网络的协作效率与网络中节点的移动方式密切相关。如果网络节点在大范围内快速运动，那么在这样的节点间建立由短距离无线通信节点构成的 Ad Hoc 网络优势并不明显，而对这样的网络进行分簇、协作和管理效率极低。为了方便，本文对所研究的 Ad Hoc 网络节点运动做如下合理假设或限制。

(1)移动节点数目有限

网络中同时移动的节点数目有限，比如厂区内的移动生产装置、仓库内的手持 RFID 读卡器、井下作业的工人或者救援人员等。

(2)运动区域受限

节点在一个大的区域内运动，但是不会运动出区域边界。这样网络在物理上撕裂的概率大大降低，减少了簇重构的能量消耗。

(3)节点运动速度受限

节点的运动速度与节点通信范围相互制约。为了将链路维护的负载限制在可承受的范围内，有必要根据节点的运动速度选择合适的通信半径。

文献[8]中，移动节点周期性地广播 Hello 消息，报告自己的坐标。邻居节点通过一段时间间隔内节点坐标的变化来判断节点的相对移动速度和方向，并作为选择簇首和簇维护的依据。如果采用文献中带有 GPS 模块的移动节点，那么节点与 GPS 系统的通信开销就不能够忽略。本文假设网络中的移动节点只能与 1 跳邻居节点进行通信。

节点 $n_i\in N$ 在 t 时刻的 1 跳邻居节点集合为 $N(n_i,t)$，在 $t+T$ 时刻为 $N(n_i,t+T)$，其中 T 为节点发送 Hello 消息的时间间隔。

节点 n_i 的新邻居节点集合和消失的邻居节点集合分别为 $N_n(n_i,t+T)$ 和 $N_d(n_i,t+T)$。邻居更新率为

$$U(n_i,t+T)=(\|N_n(n_i,t+T)\|+\|N_d(n_i,t+T)\|)/\|N(n_i,t)\|\times 100\% \quad (1)$$

为了节省邻居节点的能耗，根据 $U(n_i,t+T)$ 可以自适应地调整 T。

$$T'=\begin{cases}T-t, & \text{if}(U(n_i,t+T)>\beta_1 \text{ and} T>T_{\min}+t)\\ T+t, & \text{if}(U(n_i,t+T)<\beta_2 \text{ and } T<T_{\max}-t)\\ T, & \text{else}\end{cases} \quad (2)$$

式中，$0<\beta_2<\beta_1$，$T_{\min}$，$T_{\max}$ 分别为发送 Hello 消息的最大时间间隔和最小时间间隔。上述参数可以根据经验设定，比如 $\beta_1=60\%$，$\beta_2=20\%$，$T_{\min}=0.05$s，$T_{\max}=5$s 等。在簇成员数比较小时，可以直接用邻居节点的变化数来表示。

由于网络中节点发送 Hello 消息的周期不同，起协作管理作用的网关节点需要主动获得 1 跳邻居节点的信息。此时

节点可以在 Hello 消息中带上邻居节点信息请求。邻居节点接收到这样的 Hello 信息后，根据情况决定是否返回自己的节点信息。

2.3 k 跳簇的维护

k 跳簇中的节点移动会改变簇的拓扑结构，但并不一定直接改变簇的成员。中间节点 n_i 的移动并没有使簇内的其他节点与簇首的连接中断，那么簇首的成员列表没有发生改变，但是簇成员距离簇首的跳数会发生改变。在移动节点 n_i 使静止节点 n_j 失去中间节点时，n_j 必须加入邻居簇。单纯的网关节点的移动可能会改变簇与簇之间的边界。因此 k 跳分簇 Ad Hoc 网络的维护需要根据邻居节点的变化来判断簇成员和网络拓扑是否发生变化。

由于 k 跳簇只有在节点离开簇、簇的边界或者网关节点发生变化的情况下才有必要通知簇首，因此网关节点与簇首协作共同完成簇的维护。

相邻簇间的网关节点可以通过 1 跳邻居表中节点具有不同的簇 ID 来判断；而网络边界上网关节点的邻居节点全部是本簇的节点且距簇首的跳数最大。

2.3.1 普通节点随机运动

在 t 时刻 G_i 是簇 C_i 的网关节点。节点 G_j 是簇 C_j 的网关节点。n_i 是簇 G_i 的簇成员。如果在 t' 时刻，n_i 进入本簇边界 G_i 或者相邻簇边界 G_j 的通信范围，n_i 随即开始自适应地扫描邻居节点（见图 2 轨迹 1）。如果移动速度放慢，则扫描周期放长，反之缩短。

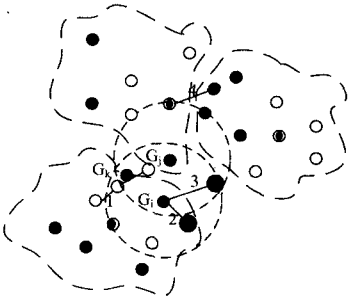

图 2 节点运动方式示意图

n_i 发现邻居表中原簇成员最小跳数为 k 时，将自己的簇号更新为邻居表中跳数最小节点的簇号，完成加入新簇的过程。G_i，G_j 通知各自的簇首。

簇成员的移动可能会导致网关节点与簇首间原有的路径中断，此时 G_i 从邻居表中选择跳数最小的节点发送消息，直到到达簇首。成为孤儿节点的情况见下文。

2.3.2 网关节点移动

如果节点 G_j 首先发现 G_i 从邻居表中消失（见图 2 轨迹 2），则 G_j 立即从邻居表中选择新的网关节点如 G_k，作为新的协作网关节点。新协作网关节点向簇首发送声明及 G_i 的移动情况。如果不存在这样的节点，那么 G_j 向自己的簇首报告 C_j 不再与簇 G_i 相邻。随后 G_i 也会做相同的操作。

当 G_i 向 G_j 所在的簇移动时（见图 2 轨迹 3），G_j 不会发现 G_i 的移动，G_i 进行如下操作：

步骤 1 当 G_i 邻居表 N_i 中原簇的 $k-1$ 跳邻居全部消失时，G_i 将自己的簇号更新为 N_i 中跳数最小节点的簇号。

步骤 2 G_j 发现 G_i 的簇号变更后，G_j 的操作同发现 G_i 消失。

2.3.3 簇首节点移动

簇首节点移动通常会使得边界上的节点与簇首失去联系。在有些分簇算法中，簇首可能本身就靠近簇边界，很容易离开簇范围。在这种情况下，必须对簇进行全面的簇重构维护。

簇重构是指重新发起成簇过程。通常当节点加入簇失败后就会发起簇重构。为了避免频繁切换簇首带来的额外开销，综合考虑节点的移动性、电池容量、计算能力等因素选择簇首。目前分簇网络的研究，大多集中在分级路由上面，簇节点的信息很容易获得。但是如果在簇内和簇间进行调度、优化或者拓扑控制，那么簇首的信息交接就显得尤为重要。

为了管理簇首移动，要求簇首节点监听自身的 1 跳邻居节点。如果发现自身邻居表变化剧烈，就说明簇首正在离开簇的范围，或者是簇成员开始疏远簇首，如图 2 中轨迹 4。此时簇首立即发起簇首交接过程，步骤如下：

步骤 1 簇首将自己的 1 跳邻居表向 1 跳邻居广播，1 跳邻居节点收到后计算自己的邻居表与收到的邻居表的相似度。相似度越高，簇成员变化越小，交接产生的孤儿节点越少。

步骤 2 1 跳邻居节点根据自身的条件判断是否成为新的簇首。

步骤 3 簇首根据接收到的回复，选择替代簇首，并向其交接信息。

步骤 4 新簇首接收到簇首交接的簇信息后，将自己的跳数更改为 0，同时向簇成员发送簇首更换消息。原簇成员节点接收到簇首更换消息后，更新节点的簇号和跳数。

2.3.4 孤儿节点管理

k 跳分簇 Ad Hoc 网络产生孤儿节点的原因有两个：一是移动到所有相邻网关节点的跳数都为 k；二是由于到簇首的路径发生断裂而被动离开簇。

对于前一种情况，通常的做法是重新发起成簇过程，这样做的缺点是要么产生的簇过小，要么会干扰相邻簇的稳定，折中的方法是成为 k 跳分簇的临时簇成员。这样虽然破坏了 k 跳分簇的前提，但是代价小。在被跳数小于 k 的网关节点探测到后，更换簇首。

后一种情况出现在节点比较稀疏时，断裂路径上的点都要选择重新加入簇，从而导致跳数增加以及簇首改变。

2.3.5 重新成簇

在网络运行超过一定时间后，特别是节点移动频繁的情况下，分簇情况可能变得不合理，簇维护已经无济于事，此时触发重新分簇。重新分簇由簇首根据移动代理信息（见 3.3 节）触发。

3 k 跳分簇 Ad Hoc 网络协作框架

3.1 k 跳簇间协作

根据文献[6]，当节点具有的邻居知识越多，越能够实现服务的优化。簇间协作的基础是相邻簇间信息的交换。簇间协作过程可以灵活设计。本文采用如下启发式规则进行簇间协作。

规则 1 簇间协作在相对稳定的邻居簇间进行。

规则 2 簇间协作在相对稳定的邻居网关节点间进行。

规则 3 相邻簇间为了降低开销，只建立 1 组协作网关

节点。

簇间协作的能量效率取决于簇的稳定。在不稳定的簇间进行协作，以及利用不稳定的网关节点进行簇间协作，能量效率都很低。规则1和规则2赋予了簇间协作的自适应性，避免了高能耗的簇间协作维护。规则3限制了在簇间存在多个相邻网关节点的情况下，简化协作关系，节省能量。

很多分簇算法尤其是针对移动 Ad Hoc 网络的算法，都强调所有的节点属于特定的簇。为了维护这种关系，往往要求整个网络都被动地运行在高负载状态。采用上述启发式规则进行簇间协作，可能会出现某些簇没有协作簇的情况。类似于无线网络的通信不确定性，这样在利用簇间协作进行管理、控制和优化时必须考虑 k 跳分簇 Ad Hoc 网络的簇间协作的可信度。

假设不同簇的网关节点 G_j 和 G_i 互为1跳邻居节点，那么可以在它们之间建立协作关系。

根据上述启发式规则，设计簇间协作算法如下：

步骤1　G_i，G_j 分别启动定时器 t_i 和 t_j。如果在定时器时间间隔内，簇首发生变化，或者利用式(1)计算 G_i，G_j 的变化率大于设定值 α，结束过程，否则转步骤2。

步骤2　G_i，G_j 分别向各自的簇首发送包含 ID 号、邻居簇号的协作请求报文。

步骤3　簇首接收到协作请求报文后，延迟一段时间。如果在这段时间内收到邻居簇号相同的协作请求，簇首选择合适的协作节点进行簇间协作。

步骤4　簇首向选定的簇间协作网关节点发送簇成员信息。

步骤5　网关节点接收到本簇成员表后，立即向相邻的邻居簇网关节点发送本簇的成员列表。

步骤6　网关节点将接收到的邻居簇成员列表信息向自己的簇首报告。

簇间协作建立后，簇首除了本簇的成员列表外，还具有了所有邻居簇的成员列表。进行协作的网关节点也获得了本簇和相邻簇的成员信息。这种簇间协作方式，实际上是通过簇间的协作扩大了簇首的知识范围，而簇首的管理范围没有改变。协作关系的维护也采用同样的启发式规则，判断是否维持协作关系。

3.2　基于移动代理的全网络协作

移动代理是一种获取网络信息的技术。通过移动代理在网络中漫游获取各个节点的动态信息，节点上的固定代理与移动代理交换信息。移动代理从当前的邻居表中选择尚未访问或者访问次数最少的节点作为下一个访问的节点。在网络规模较大时，为了保证信息的实时性，需要选择合适的移动代理数目及其漫游路径。虽然可以采用启发式的规则进行漫游，但是代理的行为仍然具有很强的随机性，使得代理间的协作变得十分的困难。

本文中 k 跳簇间协作可以扩大簇的知识范围，但是要获得更大范围的知识仍然需要在簇间进行洪泛，即使在分簇网络中仍然会占用大量的资源。

在 k 跳分簇的 Ad Hoc 网络上，移动代理只需要访问各个簇的簇首，大大提高了移动速度。而各个簇都有明确的范围，因此便于移动代理间进行协作。同时，簇首还起到静态代理的作用。

移动代理在 k 跳分簇网络中的漫游路径由簇首、网关节点构成。

移动代理由簇首发起。当簇首检测到被访问的频率过低，那么立即从邻居簇中选择一个簇发送一个移动代理。被访问次数最少的邻居簇具有高的优先级。移动代理实际上是一种特殊的服务报文。在本文中，移动代理的包结构如图3所示。

类型：移动代理	历史表：{hl_1,hl_2,hl_3,...,hl_n}	
簇号：hl_1	簇成员列表1	版本号
簇号：hl_2	簇成员列表2	版本号
……	……	……
簇号：hl_n	簇成员列表n	版本号

图3　移动代理包结构图

移动代理的历史表用于记录移动代理访问过的簇首。移动代理到达簇首节点后，簇首首先查找历史表，看自己是否在历史表中。如果不在，首先将自己的簇号添加到历史表中，然后将自己的簇号、簇成员列表、版本号添加到移动代理的信息表中。其中版本号用于记录邻居表的更新次数。簇维护过程每更新簇成员表一次，版本号增加1。如果簇首已经在移动代理的历史表中，并且版本号大于移动代理中的版本号，或者远远小于移动代理中的版本号，则簇首更新移动代理的信息表。

簇首节点分析历史列表中的簇首列表序列，根据需要获取2跳以上邻居簇的信息并保留簇间路由。

在完成信息的上载和下载后，簇首选择不在历史列表中的邻居簇，或者是移动代理具有的版本号与簇首具有的版本号相比最为陈旧的邻居簇，作为移动代理访问的下一个簇。

在移动代理的历史列表长度超过门限值，或者簇首节点所具有的邻居簇信息版本号都较新时，簇首直接丢弃接收到的移动代理。

3.3　重分簇触发

移动代理到达簇首后，簇首计算信息列表中各个簇的版本号之和。如果版本号大于门限值，那么开始全网络的重分簇。

4　实验仿真

高效的簇维护是 k 跳分簇 Ad Hoc 网络协作的基础。为了验证本文提出的 k 跳分簇 Ad Hoc 网络协作框架，主要对簇维护算法进行仿真。整个仿真网络的区域为 200×200 个单位长度，节点的通信范围长度固定为20。网络节点的规模为100，200，400，节点随机分布。节点移动速度为0.1，0.2，0.4，0.6，0.8，1倍节点通信半径，节点移动模型为随机 waypoint 模型。k 跳分簇算法，本文采用 Max-Min 算法。图4为节点数目为200的 Ad Hoc 网络的3跳分簇结果。

图5为节点移动速度与变更簇首次数的关系。节点移动速度越快，所需的簇维护量越大。但是由于采用了 k 跳分簇 Ad Hoc 网络的结构，大部分节点移动都是无需维护的簇内运动，大大降低了簇维护的需求。而且随着节点密度的提高，所需维护节点数目进一步下降，这主要是由于节点密度的增高，使得移动节点与簇首间的联系更加紧密。本文提出的 k 跳 Ad Hoc 网络簇维护的开销主要由网关节点协作进行移动节点管理产生。随着节点密度增大，网关节点数目急剧增大，

（下转第158页）

dings of 3rd International Semantic Web Conference(ISWC). 2004:683-697

[23] Miller G A. WordNet: A lexical database for english[J]. Communications of the ACM, 1995, 38(11): 39-41

[24] Giunchiglia F, Yatskevich M, Shvaiko P. Semantic matching: Algorithms and implementation[J]. Journal on Data Semantics IX, 2007(4601): 1-38

[25] Do H H, Rahm E. COMA-A System for Flexible Combination of Schema Matching Approaches[C]// Proceedings of VLDB. Morgan Kaufmann, 2002: 610-621

[26] Aumueller D, Do H H, Massmann S, et al. Schema and ontology matching with COMA++[C]// SIGMOD. 2005: 14-16

[27] Zhong Q, Li H, Li J, et al. A Gauss Function Based Approach for Unbalanced Ontology Matching[C]// SIGMOD 2009

[28] Li Y, Li J, Zhang D, et al. Result of ontology alignment with RiMOM at OAEI'06[C]// Proceedings of the 1st ISWC International Workshop on Ontology Matching(OM). 2006: 181-190

[29] Li Y, Zhong Q, Li J, et al. Result of Ontology Alignment with RiMOM at OAEI'07[C]// Proceedings of the ISWC International Workshop on Ontology Matching(OM). 2007: 227-235

[30] Zhang X, Zhong Q, Li J, et al. RiMOM Results for OAEI 2008 [C]// Proceedings of the ISWC International Workshop on Ontology Matching(OM). 2008: 182-189

[31] Zhang X, Zhong Q, Shi F, et al. RiMOM Results for OAEI 2009 [C]// Proceedings of the ISWC International Workshop on Ontology Matching(OM). 2009: 208-215

(上接第 136 页)

由此产生大量的开销，因此簇首必须控制参与簇维护的网关节点数量。图 6 所示为 3 跳簇维护能耗的估算。由于节点发射功率相等，以发送 1 个 hello 报文的能量为 y 轴单位。图 6 中虚线为网关节点自发参与簇维护产生的能耗；实线为经过簇首协调、优化选择参与维护的网关节点后的能耗。仿真结果表明通过簇首的协调，可以大大降低网关节点进行移动节点管理的能耗。

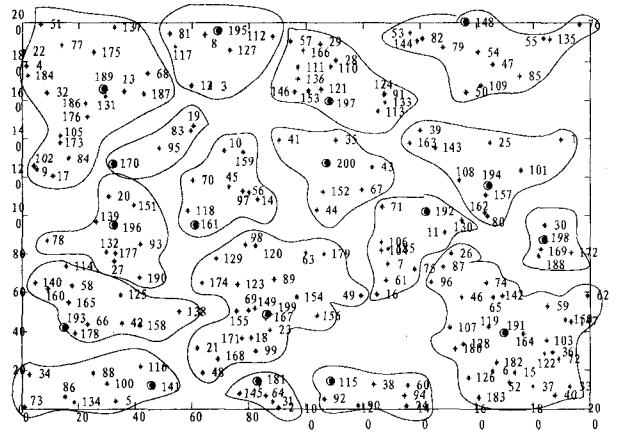

图 4 Max-Min 3 跳分簇的 Ad Hoc 网络

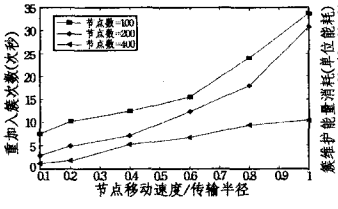

图 5 变更簇首的节点平均数目

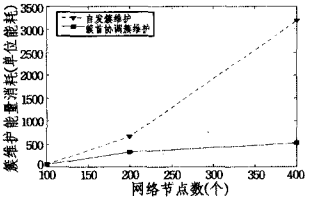

图 6 3 跳簇维护代价估计

结束语 本文提出的 k 跳分簇 Ad Hoc 网络协作框架，首先研究了移动节点的管理，在此基础上进行簇间和基于移动代理的全网络协作。宏观上讲，全网络的节点都参与到基本的簇维护协作中，但是不同类型的节点参与协作的方式不同。网关节点扫描邻居节点信息，管理节点的进出；节点在移动到边界附近开始扫描邻居，准备更换簇首；簇首扫描邻居节点进行簇首交接。

由于 k 跳簇的簇首管理的区域较大，簇首失去大部分簇成员而导致簇重构的概率较小，因此簇首的交接通常是由于簇首自身的原因。同样，成员即使移动速度较快，移出簇区域的过程也相对较长。这样 k 跳簇的簇维护需求，随着簇跳数的增大而减小。

本文提出的协作框架适用于不同的 k 跳分簇算法，但是 k 跳分簇网络的协作开销与分簇的情况密切相关。Max-Min 算法得出的分簇，会出现簇首位于簇边缘的情况，网关节点与簇首通信的开销偏大，因此有必要研究簇内通信开销最小的 k 跳分簇算法。

参 考 文 献

[1] Soltanali S, Pirahesh S, Niksefat S. An efficient scheme to motivate cooperation in mobile ad hoc networks[C]// Networking and Services. Athens, Greece: IEEE Computer Society Press, 2007: 98-103

[2] Buttyan L, Hubaux J. Stimulating cooperation in self-organizing mobile ad hoc networks[C]// Proc. ACM/Kluwer Mobile Networks and Applications (MONET). New York: IEEE Press, 2003: 579-592

[3] Buchegger S, Boudec J Y L. Performance analysis of the confidant protocol: cooperation of nodes fairness in dynamic ad-hoc networks[C]// Proc. of Mobihoc. Lausanne, Switzerland: ACM, 2002: 226-236

[4] Zhong S, Yang Y R, Chen J. Sprite: A simple, cheat-proof, credit-based system for mobile Ad Hoc networks[C]// Proc. of INFOCOM. San Francisco, USA: IEEE Press, 2003: 187-197

[5] 黄蕾，刘立祥. Ad hoc 网络寻路阶段的合作激励机制研究[J]. 计算机学报，2008，31(2)：262-269

[6] Bulut E, Zheng J, Wang Z, et al. Balancing cost-quality tradeoff in cooperative ad hoc sensor networks[C]// IEEE Military Communications Conference, MILCOM. San Diego: IEEE Press, 2008: 1-7

[7] Amis A D, Prakash R, Tai H P. Max-Min D-cluster formation in wireless ad hoc networks[C]// Proc. IEEE INFOCOM. New York: IEEE Press, 1999: 1-10

[8] Leng Su-peng, Zhang Li-ren, Chen Yi-fan. K-hop compound metric based clustering scheme for ad hoc networks[C]// Proc. IEEE International Conference on Communications. Seoul: IEEE Press, 2005: 3396-3400

[9] Shen C C, Srisathapornphat C, Liu R, et al. CLTC: A cluster-based topology control frame-work for ad hoc networks[J]. IEEE Transactions on Mobile Computing, 2004, 3(1): 18-32