改进 AntNet 算法在 Ad Hoc 网络路由中的应用①

肖军弼, 刘战军

(中国石油大学(华东) 计算机与通信工程学院, 青岛 266580)

摘 要: 当前大部分的 Ad Hoc 网络路由算法在选择路由的时候都没有很好地将节点的能量状态引入到评价系统中去. 针对这一问题, 本文对 AntNet 算法进行了适当改进, 使其能够记忆和衡量整个路由的能量状态变化. 文章详细描述了算法的数据结构, 以及节点选择规则和数据结构更新规则. 仿真实验和结果分析表明, 改进的 AntNet 算法能够找到平均能量较高且各节点能量较稳定的路径, 从而提高网络的生存时间和吞吐量.

关键词: AntNet; Ad Hoc 网络; 能量状态; 路由算法; OMNET

Improved AntNet Algorithm for Routing in Mobile Ad Hoc Networks

XIAO Jun-Bi, LIU Zhan-Jun

(College of Computer&Communication Engineering, China University of Petroleum, Qingdao 266580, China)

Abstract: Nowadays, the energy state has not been introduced completely into the routing evaluation system in most routing algorithm of Ad Hoc networks. In order to solve this problem, the AntNet algorithm in this paper is improved appropriately to memory and evaluates the changes of the energy state in entire path. Data structures, rules of selecting next node and rules of updating data structures are described in detail in this paper. By Simulation study and analysis, the lifetime and throughput of the Ad Hoc networks can be increased by the improved AntNet algorithm by using the route in which the average energy of each node is higher and more stable.

Key words: AntNet; Ad Hoc network; energy status; routing algorithm; OMNET

Ad Hoc 网络(又称无线移动自组网, Mobile Ad Hoc NetWorks, 简称 MANETs)是一种具有临时快速自动组网能力的新型网络, 这种网络不存在固定基础设施, 拓扑结构变化频繁, 并且所有节点地位平等, 可作为路由转发节点[1]. Ad Hoc 网络所具有的诸多特点使得很多在传统固定网络中运行的各种路由协议无法直接运用在上面, 所以设计一种较好的动态路由协议就成为了 Ad Hoc 网络研究的关键和难点之一. 20 世纪 90 年代以来, 学者们相继为 Ad Hoc 网络设计了很多新型的路由协议, 例如: AODV[2,3]和 DSR[4], 但是这些协议都没有专门评论整个路由的能量状态. 由于 Ad Hoc 网络中节点的能量通常有非常有限的, 所以设计一种能够找到一条平均能量较高且各节点能量较稳定的路径, 进而增加节点的生存时间和提高网络的吞吐量就成为一个值得研究的问题.

在 1991 年, 由意大利学者 Marco Dorigo 提出的、具有分布式计算能力的蚁群优化算法(Ant Colony Optimization, ACO)是一种用于解决复杂组合优化问题的启发式方法[5]. 后来, Gianni Di Caro 和 Marco Dorigo 在 ACO 在基础上发展了 AntNet 算法[6,7]来解决电子通信网络的路由问题, 并取得了非常好的计算性能. 因此本文在 AntNet 算法的基础上引入节点能量检测和评价机制, 并且对算法的数据结构进行了改进, 提出了在延迟较好的情况下尽量使用能量较高的节点转发数据分组, 进而延长节点的生存时间, 从而降低 Ad Hoc 网络拓扑结构变化频率, 最终提高网络吞吐量的一个解决方案. 由于 AntNet 算法中的 Agent 是一种能够独立完成简单任务的软件实体, 所以只要 Agent 在遍历网络的过程中收集到节点的能量信息就可以对由节点组成的整个路径的能量状态进行评价, 从而找

① 收稿时间:2013-04-27;收到修改稿时间:2013-05-21

到一条符合上面要求的路径, 达到增强网络拓扑的稳定性的目的.

为了统计网络中节点的能量状态, 本文对 AntNet 算法中的网络状态统计模型进行了改进和扩展. 改进后的网络状态统计模型可以同时记录与蚂蚁遍历时间和路由节点能量状态有关的统计信息. 借鉴 AntNet 算法中对信息素对更新规则, 本文根据改进后的网络状态统计模型对信息素的更新规则进行了适当改进, 把节点的能量状态加入到信息素更新规则当中. 为了获得改进网络状态统计模型中的统计信息, 本文对 AntNet 算法中 Agent 数据结构进行了相应扩展和改进, 增加了 Agent 对节点能量状态的收集功能. 下文对算法设计进行了详细说明.

1 问题描述

1.1 问题描述

本文将 Ad Hoc 网络抽象为一个图 $G=(C,L)$, 其中 C 表示节点集合, $|C|=n$ 表示节点集合中元素个数, 每个节点有唯一的标识符 $c_i, i\in C$, 同时节点的能量剩余用 $E_i, i\in C$, L 表示节点间的双向无线通信链路集合且两节点之间只有一条链路 $l(i,j), i,j\in C$, 如果两个节点能互相接收无线信号, 就认为这两个节点是相邻节点. 路由算法的目的就是在图中找出两节点间符合约束的节点序列.

1.2 算法主要思想

本文对标准 AntNet 算法的数据结构进行了适当的改进, 使其能够适应改进的算法. 这种改进最主要是在两个方面: 一方面是在蚂蚁和节点的网络状态参数统计模型这两个数据结构中增加记录节点能量状态的数据结构; 另一方面是对信息素的更新公式进行适当地变化, 使其能够合理利用蚂蚁新收集的路由能量状态信息.

AntNet 算法的主要思想可以概括为: 在每个中间节点上, 与数据包有相同队列优先级的正向蚂蚁 $agentF_{s\to d}$ 按照贪婪随机策略选择下一个要到达的节点, 同时收集与该节有关的各种状态信息. 当到达目的节点的正向蚂蚁 $agentF_{s\to d}$ 收集完该节点信息后, 它将会生成一个逆向蚂蚁 $agentB_{d\to s}$ 并把自己的记忆按文中的规则复制给该逆向蚂蚁, 然后正向蚂蚁被删除. 逆向蚂蚁 $agentB_{d\to s}$ 行进的路径和对应的正向蚂蚁完全相同, 但是方向正好相反. 在逆向蚂蚁使用高优先级队列快速返回源节点的过程中, 它将向节点存储的信息素表传递正向蚂蚁收集的信息, 完成信息素的更新. 更新后的节点信息素表就可以反映每条路由的最新延时和能量状态, 为数据分组选路提供可靠的依据.

2 算法设计

2.1 数据结构

本文中的 AntNet 算法要使用到的数据结构主要包括两部分:

一部分是节点 i 中存储的信息素表 $T_i=[\tau_{idj}]$、网络状态参数统计模型 $M_i=[S(t_{i\to d})]$, 邻接表 N_i 和路由表 $R_i=[d,P_{idj},c_i]$. 信息素表 T_i 的元素 τ_{idj} 表示的是目的节点为 d 的蚂蚁在结点 i 时选择节点 j 作为下一个结点的期望度; 网络状态参数统计模型 $M_i=[S(t_{i\to d})]=[\mu_{id},\sigma_{id}^2,W_{id}]$ 是自适应的, 其中 $S(t_{i\to d})$ 是存储正向蚂蚁遍历时间和节点能量状态的栈结构, μ_{id} 是目的节点为 d 的蚂蚁在遍历过程中收集的网络状态参数的样本均值, σ_{id}^2 是其样本方差, W_{id} 是一个移动观察窗口, 它用来记录蚂蚁遍历过程中收集的网络状态参数的最好结果 $W_{id}=[W_{best}^t,W_{best}^e]$, W_{best}^t 是 $t_{s\to d}$ 的最小值, W_{best}^e 是蚂蚁遍历路由的节点能量总和的最大值; 邻接表 $N_i=[c_j,v_j], j\in|N_i|$, c_j 表示相邻节点标识, v_j 用来标识该节点是否在一定时间内被正向蚂蚁访问过, 通过节点之间周期性广播的 Hello 包来建立和维护, 并且 $n_i=|N_i|$ 是节点 i 的邻接节点个数; 路由表 $R_i=[d,P_{idj},c_i]$ 存储了一定时间内 AntNet 算法所发现的最好路由, 其中, P_{idj} 表示的是目的节点为 d 的数据分组在节点 i 时选择节点 j 作为下一个节点的概率.

另一部分是蚂蚁数据结构 $agent$: AntNet 算法将蚁群分为正向蚂蚁 $agentF_{s\to d}$ 和逆向蚂蚁 $agentB_{d\to s}$, 并且两类蚂蚁使用相同的数据结构 $agent=[ID,S_{s\to d},s,d,TTL]$, 其中 ID 是蚂蚁标识, s 是源节点标识符, d 是目的节点标识符, $S_{s\to d}$ 是用来存储正向蚂蚁 $agentF_{s\to d}$ 从源节点 s 到目的节点 d 所经过的中间节点信息的栈结构, 例如, $S_{s\to d}$ 包括访问过的节点 i 的标识符、从源点 s 到达节点 i 所用的时间和节点 i 此时的能量状态, TTL 是蚂蚁在网络中最长的生存时间, 通常为跳数. 与正向蚂蚁不同, 逆向蚂蚁的 $S_{s\to d}$ 中存储的是从节点 i 到达目的节点 d 所有的时间, 该时间可以通过正向蚂蚁中的 $S_{s\to d}$ 计算得出.

2.2 信息素表初始化

在 Ad Hoc 网络中, 节点的加入和离开比较频繁. 对一个刚刚加入 Ad Hoc 网络的节点, 并且当其邻接表在一定时间内不发生变化的时候, 就对该节点的信息素表进行初始化工作. 由于 AntNet 算法没有具体的初始化方法, 所以本文使用文献[8]提出的一种能够反映网络拓扑的信息表初始化方法:

如果目的节点 d 正好是当前节点 i 的一个相邻节点, 即 $d \in N_i$, 那么使用下式初始化:

$$\tau_{idd} = \frac{1}{n_i} + \frac{3(n_i - 1)}{2n_i} \tag{1}$$

对于其他的节点 $j \notin N_i$, 则使用下式初始化:

$$\tau_{idj} = \begin{cases} 0, n_i = 1 \\ \dfrac{1}{n_i} + \dfrac{3(n_i - 1)}{2n_i}, n_i > 1 \end{cases} \tag{2}$$

并且公式(1)和(2)都满足 $\sum_{j \in N_i} \tau_{idj} = 1$.

2.3 节选择规则

当有新的路由请求发出的时候, 在源节点 S 每隔一定的时间间隔 Δt 就有一只目的节点为 d 的正向蚂蚁 $agentF_{s \to d}$ 出发. 在 AntNet 算法中, 正向蚂蚁 $agentF_{s \to d}$ 在当前节点 i 选择相邻节点 j 作为下一个访问节点的概率 P_{idj} 按照公式(3)[7]计算, 并按其最大值选择.

$$\begin{cases} P_{idj} = \dfrac{\tau_{idj} + \alpha \eta_{ij}}{1 + \alpha\left(\left|N_i - 1\right|\right)}, j \in N_i \\ \eta_{ij} = 1 - \dfrac{q_{ij}}{\sum_{l \in N_i} q_{il}} \end{cases} \tag{3}$$

其中, q_{ij} 是节点 i 与它的相邻节点 j 之间链路上的队列长度, α 是衡量在 q_{ij} 中启发值 η_{ij} 相对于信息素 τ_{idj} 的重要性. 启发值 η_{ij} 的值反映了节点 i 当前队列的瞬时状态, 可以被用来减少网络波动对算法的影响, 即链路上队列越小, 该链路被选择的概率就越大. 在计算 P_{idj} 的同时, 与最大 P_{idj} 相关的目的节点 d 和下一个节点 c_i 就会被放入到节点 i 的路由表 $R_i = [d, P_{idj}, c_i]$ 中.

2.4 数据结构更新规则

2.4.1 网络状态参数统计模型更新规则

在本文的 AntNet 算法中, 逆向蚂蚁并不是完全复制与其对应的正向蚂蚁所收集的信息. 逆向蚂蚁的 $S_{s \to d}$ 的时间项是在与其对应的正向蚂蚁的 $S_{s \to d}$ 中时间项的基础上按照公式(4)计算求得的:

$$t_{s \to d} = t_{s \to d} - t_{s \to i} \tag{4}$$

公式(4)中, $t_{s \to d}$ 和 $t_{s \to i}$ 都是正向蚂蚁的中时间项.

为了使节点能够及时了解网络状态的变化情况, 逆向蚂蚁使用公式(5)更新节点 i 的网络状态参数统计模型 $M_i = [S(t_{i \to d})]$, 进而使节点 i 的信息素表也能够反映网络状态的变化情况.

$$\begin{cases} \mu_{id} \leftarrow \mu_{id} + \varsigma(o_{i \to d} - \mu_{id}) \\ \sigma_{id}^2 \leftarrow \sigma_{id}^2 + \varsigma((o_{i \to d} - \mu_{id})^2 - \sigma_{id}^2) \end{cases}, 0 < \varsigma < 1 \tag{5}$$

公式(5)中 $o_{i \to d}$ 是最近观测到的正向蚂蚁从节点 i 到节点 d 的遍历时间或者是正向蚂蚁最近观测到的节点 i 的能量状态. 因子 ς 是衡量会真正影响平均值的最近样本数目的比重. 因为被 Ad Hoc 网络中节点的特点所限制, 所以设观测窗口的大小为 $w_{\max} = 5c/\varsigma, c < 1$.

2.4.2 节点信息表更新规则

当逆向蚂蚁从某个相邻节点 f 到达节点 i 后, 它就会对节点 i 上所有与节点 d 相关的数据结构进行更新, 其中公式(6)用来更新信息素 τ_{idj}:

$$\begin{cases} \tau_{idf} \leftarrow \tau_{idf} + r(1 - \tau_{idf}) \\ \tau_{idj} \leftarrow \tau_{idj} - r(1 - \tau_{idj}), j \in N_i 且 j \neq f \\ r \equiv (T, E, M_i) = \gamma_1 r^t + \gamma_2 r^e, 0 < r \leq 1 \end{cases} \tag{6}$$

公式(6)中 $r \equiv (T, E, M_i)$ 是有关时间、能量和网络状态参数统计模型的信息素更新系数, 用来评价遍历路径的“好坏”, γ_1 和 γ_2 分别是遍历时间和节点能量状态在 r 中的权重.

借鉴标准 AntNet 算法中计算 r 的公式, 本文使用公式(7)来得到较好的 r^t 和 r^e 值:

$$\begin{cases} r^t = c_1 \dfrac{W_{best}^t}{T} + c_2 \dfrac{I_{\sup}^t - I_{\inf}^t}{\left(I_{\sup}^t - I_{\inf}^t\right) + \left(T - I_{\inf}^t\right)} \\ r^e = c_3 \dfrac{W_{best}^e}{E} + c_4 \dfrac{I_{\sup}^e - I_{\inf}^e}{\left(I_{\sup}^e - I_{\inf}^e\right) + \left(E - I_{\inf}^e\right)} \end{cases} \tag{7}$$

公式(7)中, 系数 c_1、c_2、c_3 和 c_4 表示各项的权重, T 和 E 分别表示最近一次遍历的时间 $t_{s \to d}$ 和路由节点能量的和, $I_{\inf}^t$ 和 $I_{\inf}^t$ 分别设置为存储在网络状态参数统计模型中的 W_{best}^t 和 W_{best}^e, 且 $I_{\sup}^t$ 和 $I_{\sup}^e$ 分别使用网络状态参数统计模型对应的数据来根据公式(8)计算:

$$I_{\sup} = \mu_{id} + \frac{\sigma_{id}}{\sqrt{1-z_{id}}\sqrt{w_{id}}} \quad (8)$$

公式(8)中 z_{id} 是对相应 μ_{id} 的置信水平.

最后对公式(7)中的 r^t 和 r^e 的值使用挤压函数 $s(x)$ 来转换, 使得 r 的值在低值区域收缩, 在高值区域扩展, 就把重点放在好的结果上:

$$\begin{cases} r = \dfrac{s(r)}{s(1)} \\ s(x) = (1+\exp(\dfrac{a}{x\left|N_i\right|}))^{-1}, x \in (0,1], a \in R^+ \end{cases} \quad (9)$$

3 算法描述

算法的详细步骤如下:

1) 首先使用 Hello 包的形式建立并维护节点的邻接表, 然后按照公式(1)和(2)对节点进行信息素表的初始化工作.

2) 如果源节点 s 想和目的节点 d 进行通信, 它必须先检查路由表 R_i 是否有一个目的节点为 d 的路由. 若存在一条这样的路由, 节点 s 就可直接按照此路由与节点 d 进行通信, 否则, 源节点 s 就发起路由请求, 每隔一定的时间间隔 Δt 就向网络中放出正向蚂蚁来不断地去寻找更好的路由, 即开始 AntNet 算法的迭代过程.

3) 正向蚂蚁 $agentF_{s\to d}$ 完成路由构建过程的高层伪代码如下:

```
if (t mod Δt) == 0 then
  发出 agentF_{s→d}(源节点,目的节点);
end-if
while (当前节点不是目的节点) do
  next_hop = 选择下一个节点(当前节点, 目的节点, 链路状态, 信息素);
  agentF_{s→d} 前进(当前节点, next_hop);
  当前节点 = next_hop;
  agentF_{s→d} PushMemory(当前节点, 跳数加一, 延时, 当前节点的能量状态);
end-while
发出逆向蚂蚁 agentB_{d→s}(目的节点, 源节点);
```

4) 逆向蚂蚁 $agentB_{d\to s}$ 完成数据结构更新过程的高层伪代码如下:

```
while (当前节点不是源节点) do
  next_hop = PopMemory;
  agentB_{d→s} 前进(当前节点, next_hop);
  from = 当前节点;
  当前节点= next_hop;
  更新本地网络状态统计模型(M, 当前节点, from, 源节点);
  更新信息素(M, 当前节点, from, 源节点);
  更新路由表(T, 当前节点, 源节点, R);
end-while
```

4 模拟仿真与结果分析

为了验证改进的 AntNet 算法的可行性, 本文利用 OMNET++仿真软件对改进算法和标准 AntNet 算法进行了对比的模拟实验和结果分析.

4.1 仿真情景设计

本文的仿真场景为: 节点被随机放置在 1000*500m^2 的区域中,每次模拟都运行 900s. 所有源节点都是 20CBR, 且每秒发送一个 64byte 的数据包. 每个源节点在模拟开始后的 0 到 180s 内的随机时刻发送数据直到结束实验. 在无线电传播模型中使用一个双向射线路径损耗模型. 节点的无线电范围是 100 米, 数据传输速率是 2Mbit/s. 在 MAC 层使用 MANET 中常用的 802.11b DCF 协议. 节点的移动速度为 5m/s, 方向随机. 参数设置如表 1:

表 1　算法运行时的参数设置表

节点电池容量 E_{max}	$E_{max} = 2000mAH$
时间间隔 Δt	$\Delta t = 0.3s$
蚂蚁生存时间 TTL	$TTL = 14$
公式(3)中的 α	$\alpha = 0.45$
公式(5)中的 ς	$\varsigma = 0.005$
公式(5)中的 c	$c = 0.3$
公式(6)中的 γ_1,γ_2	$\gamma_1 = \gamma_2 = 0.5$
公式(7)中的 c_1,c_3	$c_1 = c_3 = 0.7$
公式(7)中的 c_2,c_4	$c_2 = c_4 = 0.3$
公式(8)中的 z_{id}	$z_{id} = 1.7$
公式(9)中的 x	$x = 0.3$
公式(9)中的 a	$a = 10$

4.2 实验结果分析

本文将模拟仿真实验分为 3 个规模不大的场景, 节点个数分别为: 15、25 和 50, 每情景都记录编号为 1

的节点到节点 15 的实验数据. 将每个情景都运行 20 次后的结果被整理成如下的三个表格并分析:

表 2 场景 1 的统计结果

统计项	AntNet	改进 AntNet
路由的最短延时	0.78ms	0.85ms
路由的平均延时	1.00ms	1.10ms
路由中节点的平均能量	95.7W	129.5W
路由的延时方差	105	106
路由的能量方差	158	98

表 3 场景 2 的统计结果

统计项	AntNet	改进 AntNet
路由的最短延时	0.82ms	0.96ms
路由的平均延时	1.24ms	1.53ms
路由中节点的平均能量	93.8W	118.6W
路由的延时方差	126	142
路由的能量方差	185	114

表 4 场景 3 的统计结果

统计项	AntNet	改进 AntNet
路由的最短延时	1.12ms	1.35ms
路由的平均延时	1.47ms	1.83ms
路由中节点的平均能量	84.3W	90.8W
路由的延时方差	137	159
路由的能量方差	215	106

① 从延时方面来看, 随着 Ad Hoc 网络中节点数目的增加, 两个算法的延时都有不同程度的增加, 这是由于网络中数据量增加和节点链路队列变长所引起的正常变化[9]. 将两个算法分开来看, 改进的 AntNet 算法的路由延时和 AntNet 算法相差很小, 并且路由的延时波动情况也差距不大. 在实际环境中这两者的差别对小规模的 Ad Hoc 网络的数据传输不会对数据传输产生明显的影响. 因此, 本文提出的算法对路由延时性能的影响很小.

② 从节点能量方面来看, 随着 Ad Hoc 网络中节点数目的增加, 两个算法的节点能量都有不同程度的下降, 这是由于网络中数据量增加和节点处理数据分组时能耗增加所引起的正常变化. 将两个算法分开来看, 改进的AntNet算法的节点平均能量比AntNet算法要多, 并且路由的能量方差更是比 AntNet 算法要小得多. 因此, 本文提出的算法在路由的能量状态评价方面是取得了一些进展, 可找到一条平均能量较高且各节点能量较稳定的路径.

5 结束语

为了能够找到平均能量较高且其各节点能量较稳定的路径, 从而使整个Ad Hoc网络中的节点可以运行更长的时间, 因此, 本文提出了基于路由能量评价机制的 AntNet 算法. 经过模拟仿真实验和结果分析, 改进 AntNet 算法能够在小规模的 Ad Hoc 网络中找到符合算法设计目的的路由, 并且不会对路由延时造成明显的影响. 所以, 改进的 AntNet 算法可以延长节点的生存时间, 维持网络拓扑的稳定, 从而达到提高网络的吞吐量的目的. 另外, 本文将密切关注人们对 Ad Hoc 网络在 QoS 方面的要求, 并且会对这方面的问题展开进一步的研究.

参考文献

1 郑相全.无线自组网技术实用教程.北京:清华大学出版社.2004:2–8.

2 Perkins C, Perkins C, Das S. Ad hoc On-Demand Distance Vector(AODV) Routing. RFC 3561, 2003 July.

3 吴国风,邵臣.Ad Hoc 网络中 AODV 路由协议的分析与改进.计算机系统应用,2010,19(10):221–224.

4 Johnson D, Johnson D, Maltz D. The Dynamic Source Routing Protocol(DSR) for Mobile Ad Hoc Networks for IPv4.RFC 4728, 2007 February.

5 Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating Agents. IEEE Transactions on Systems, Man, and Cybemetics Part B, 1996, 26(1): 29–41.

6 Caro GD, Dorigo M. AntNet: A Mobile Approach to Adaptive Routing[Technical Report]. IRIDIA/97-12. Universite Libre de Bruxelles, Belgium, 1997.

7 Dorigo M, Stutzle T,张军,胡晓敏,罗旭耀等译.蚁群优化.北京:清华大学出版社,2007:215–232.

8 Baran B, Sosa R. A new approach for AntNet routing. Proc. of Ninth International Conference on Computer Communications and Networks. Las Vegas, USA. 2000.

9 耿鹏,邹传云.Ad Hoc 网络中节点密度对路由协议性能的影响.计算机系统应用,2007,16(1):25–27.