第 36 卷 第 6 期 计 算 机 工 程 2010 年 3 月
Vol.36 No.6 Computer Engineering March 2010
·网络与通信· 文章编号：1000—3428(2010)06—0126—02 文献标识码：A 中图分类号：TP393

AODV 路由协议的本地修复算法

丁绪星，吴 青，谢方方
(安徽师范大学物理与电子信息学院，芜湖 241000)

摘 要：针对传统按需平面距离矢量(AODV)路由修复算法路由开销大和端到端时延长的问题，提出一种改进的 AODV 本地路由修复算法。通过路由修复阶段 2hop_RREQ 和 NOTICE 报文的传递，将修复限制在断链的 2 跳范围内。在 NS2 平台下的仿真结果表明，与传统算法相比，改进算法的路由开销减少约 50%，包投递率增加约 5%。

关键词：Ad Hoc 网；按需平面距离矢量；本地修复

Local Repair Algorithm for AODV Routing Protocol

DING Xu-xing, WU Qing, XIE Fang-fang
(College of Physics and Electronic Information, Anhui Normal University, Wuhu 241000)

【Abstract】 Aiming at high control overhead and long packet delay of the traditional Ad Hoc On Demand Distance Vector(AODV) routing repair algorithm, and improved algorithm is proposed. By delivering 2hop_RREQ and NOTICE packets during route repair phase, the repair near the broken links of the improved algorithm is limited at the range of 2 hops. Compared with the traditional algorithm, the results obtained by NS2 show that the routing cost is decreased by 50% and the packet delivery ratio is increased by 5%.

【Key words】 Ad Hoc network; Ad Hoc On Demand Distance Vector(AODV); local repair

1 概述

移动 Ad Hoc 网(Mobile Ad Hoc Networks, MANET)[1]是由一组带有无线通信收发装置的移动终端组成的一种多跳的临时自治性系统，可以随时随地快速构建移动通信网络，并且不需要现有信息基础网络设施的支持，主要用于应急通信和军用移动通信。由于网络的终端具有移动性，必然导致网络拓扑的动态变化，从而要求路由算法能自适应这种变化并及时修复断开的链路，因此路由技术一直是自组网的研究重点与热点之一。研究表明[2]，在负载较重的情况下，文献[3]提出的按需平面距离矢量(Ad Hoc On Demand Distance Vector, AODV)协议的性能最为理想，更能适应 Ad Hoc 网络的动态拓扑、带宽受限和能源约束等要求。但该算法需要较大范围的泛洪，导致了较大的控制开销和较长的时延。为了解决该问题，近年来提出了许多改进算法，如文献[4]提出的改进算法避免了修复时大范围的泛洪，但在路由发现过程中增加了更多的路由开销；文献[5]提出了一种路由局部修复算法，用较少的开销实现了对断链的快速修复，但这种算法只能用于源路由协议。

本文对 AODV 算法的路由修复部分进行改进，将路由修复限制在 2 跳范围内，用断链附近的节点替换失效节点，避免出现大范围泛洪，且无须保存下 2 跳节点信息，避免增加额外的网络负担。

2 AODV 修复算法

AODV 是 MANET 主要采用的一种路由协议，其算法主要包括 2 个阶段：路由发现和路由维护。由于自组网终端的移动性，已经建立的路由会随着终端的移动发生改变，因此路由维护尤为重要。文献[3]提出的 AODV 修复算法如图 1 所示，当节点 *A* 发送或者转发数据分组到目的节点 *D* 时，节点 *A* 查找路由表。如果路由表项中目的节点为 *D* 的下一跳节点 *B* 失效或者移出节点 *A* 的通信范围，节点 *A* 到节点 *D* 的路径失效。此时如果 *A* 到 *D* 的距离比较近，则发起本地修复，广播目的为 *D* 的 RREQ(路由请求)消息，同时缓存数据分组，启动定时器等待路由发现周期获得响应的 RREP(路由应答)消息。RREQ 消息途经 *E*, *F* 等节点，最后到达 *D*，*D* 回复 RREP 消息，沿反向路由传送到 *A*。这样，就重新建立了从节点 *A* 途经 *E*, *F* 到 *D* 的路由。如果路由发现周期结束 *A* 仍然没有收到关于目的节点的 RREP，则向路由上游节点发送该目的节点的 RERR(错误消息)，直到该路由的源节点之后源节点标记路由失效，进行新一轮的路由发现。

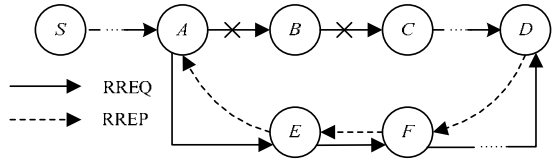

图 1 文献[3]的 AODV 修复算法

实验发现，路由失效通常发生在断链附近。AODV 修复协议在路由修复时，直接面向目的节点查找新路由将导致一些无谓的路由发现，引发大范围泛洪，增加控制开销和时延。

3 AODV 局部修复的改进

本文在上述 AODV 算法的基础上提出一种局部修复改进算法，其示意图如图 2 所示，当节点 *A* 到目的节点 *D* 路径的下一跳节点 *B* 失效或运动出节点 *A* 的通信范围时，节点 *A* 到节点 *D* 的路径失效。在断链的附近可能存在 *A*, *B* 的邻居节

基金项目：芜湖市科技计划基金资助重点项目(芜科计 2008320)
作者简介：丁绪星(1971－)，男，副教授、博士，主研方向：无线传感器网络，图像传输与处理；吴 青、谢方方，硕士研究生
收稿日期：2009-08-20 **E-mail：**dxx200@163.com

点，如果断链的上游节点发送 2 跳路由请求分组寻找下一跳节点 *B* 或者原路由上离目的节点更近的节点(如图 2 中的 *C*)以及其他拥有到达目的节点有效路径的节点，因修复限制在 2 跳范围内，网络上传播的控制分组数量将会减少，寻路时间将会减短。

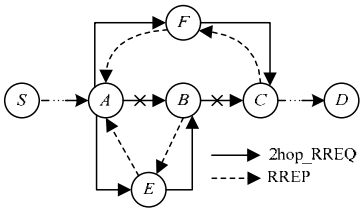

图 2　改进的 2 跳路由修复算法

算法主要步骤如下：

(1)当路由中节点 *A* 发现与下一跳节点 *B* 链路断开，先将该路由状态标记为 in_repair，缓存相应的数据分组。设置定时器，等待路由修复，等待时长设为 0.1 s。

(2)上游节点 *A* 发送 TTL=2 的 2 跳修复请求分组 2hop_RREQ 包。2hop_RREQ 消息包含下一跳节点和目的节点信息。收到 2hop_RREQ 消息的节点检测自己是否是要找的下一跳节点或者是否有“足够新”的到达目的节点的路由。如果是，确认到目的节点 *D* 的路由有效后发送路由应答 RREP 经反向链路到达节点 *A*，同时沿着去目的节点的路径发送 NOTICE 分组，通知节点更新目的节点序列号。

(3)节点 *A* 收到修复应答分组 RREP 后，将缓存的数据分组沿修复的路由发送。

(4)如果在修复周期内节点 *A* 未收到相应的修复应答分组 RREP，将丢弃缓存的数据分组，启动原来的本地修复算法，直接向目的节点发送路由请求。

在具体的实现过程中，为了进行路由的本地修复，新增加了 2 跳修复的路由请求分组 2hop_RREQ 和 NOTICE 分组。NOTICE 分组格式见图 3，其中，TYPE 为分组类型；DEST 为需要通知的目的节点；DEST_SEQNO 为需要通知的目的节点的新序列号；NEXTHOP 为下一跳，接收方用于判断该分组是否正确发送。

TYPE	DEST	DEST_SEQNO	NEXTHOP
(8 bits)	(32 bit)	(8 bit)	(32 bit)

图 3　NOTICE 分组格式

为了查找下一跳节点或者拥有“足够新”到目的节点路由的节点，路由修复请求信息要求携带修复路由的真实目的节点(图 2 中的节点 *D*)的信息和下一跳节点(图 2 中的节点 *B*)。于是，2hop_RREQ 分组在 RREQ 分组基础上添加了以下字段：真实目的节点的 IP 地址和序列号。

在本地修复情况下，当下一跳节点 *B* 响应时，为了使修复的路径在节点 *A* 处得到响应，需将节点 *B* 的目的节点序列号加 2，从而使 *A* 和 *B* 的真实目的节点的序列号更新，但 *B* 与 *D* 之间的节点没有得到更新，在以后的路由中可能形成环路，或者 *D* 发出的路由分组将因序列号低而无法更新其他节点。因此，协议增加了 NOTICE 分组。当节点 *B* 收到 2hop_RREQ 分组时，产生一个 NOTICE 分组并存入 *D* 的新序列号，并且将这个分组沿着 *B* 到 *D* 的路径发送，路径上的节点收到此分组后更新自己关于 *D* 的序列号，最后到达 *D*，*D* 更新自己的序列号，从而避免了以上问题。由于 NOTICE 分组很小，因此增加的协议开销很少。

当下 2 跳节点 *C* 收到节点 *A* 发送的 2hop_RREQ 分组后，节点 *C* 确认到节点 *D* 的路由有效，沿反向路径向节点 *A* 发送 RREP。如果节点 *C* 拥有的目的节点 *D* 的序列号与节点 *A* 拥有的相同，节点 *A* 接收到 RREP 后，却因跳数相同而导致路由无法更新。为了解决这一问题，在本改进算法中，节点 *A* 收到 RREP，检测自己是否处在 in_repair 状态，若是，将更新路由的条件改为：

```
if (路由处在修复状态)
    {if (RREP 中目的节点序列号新 ||(序列号相同 &&跳数≤原来的跳数) )
        更新路由;}
else if (RREP 中目的节点序列号新 || (序列号相同 &&跳数 < 原来的跳数) )
    更新路由;
else 不予处理;
```

当失效节点附近存在到达目的节点的足够新的其他路由，收到 2hop_RREQ 分组后，可将该路由完全替代原路由。

4　改进算法的仿真结果

在仿真场景中，30 个节点随机分布在 1 500 m× 500 m 的矩形区域内，每次仿真随机选择 20 对源-目的节点传输数据，通信源是 CBR 流，数据包大小为 512 Byte，每秒发送 4 个包。采用 Random Way Point 运动模型，节点最大移动速度 V_{max} 分别为 40 m/s, 30 m/s, 20m/s, 10 m/s，暂停时间为 30 s，每次仿真运行 400 s。

本文从数据包投递率、端到端的平均延迟、路由开销 3 个性能指标对改进的算法和原算法[3]进行了验证比较。仿真结果如图 4~图 6 所示(图中 AODV 表示文献[3]的算法，AODV-2hop 表示本文改进的算法)。从图中可看出，与传统算法[3]相比，改进算法在速度为 40 m/s 时包投递率从 82%增加到 87%，提高了 5 个百分点；路由开销减少了近一半；端到端的平均延迟在速度低于 30 m/s 时，性能有所优化，当速度提高到大于 30 m/s 后，延迟反而增加，这是因为本地修复虽然可以快速地修复断链，但同时也会造成路由跳数的增加，导致平均延迟没有相应的提高。

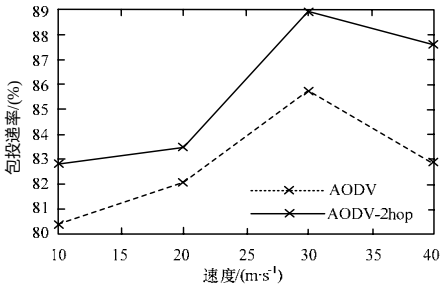

图 4　包投递率

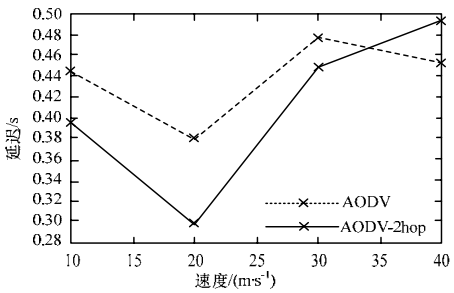

图 5　端到端的平均延迟　(下转第 130 页)