

减少拥塞的 AODV 路由协议

高圣国¹, 王汉兴^{1,2}

(1. 上海大学理学院, 上海 200444; 2. 上海立信会计学院数学与统计系, 上海 201620)

摘 要: 为了减少建立路由所产生的延时, AODV 路由协议允许拥有积极路由的中间节点响应路由请求, 而不考虑该节点当时的负荷, 这样虽然快速地建立了路由, 但并不一定能有效地传输数据。针对这一问题, 提出一个改进, 在改进的 AODV 路由协议中, 只有负荷较轻的中间节点才能响应路由请求或者进行局部路由维修, 使数据流不在某些路段上过于集中。仿真结果表明改进的协议和原协议相比, 降低了端到端的延时和丢包率, 提高了传输效率。

关键词: Ad hoc 网络; AODV 路由协议; 路由; 生存时间

Congestion-decreased AODV Routing Protocol

GAO Shengguo¹, WANG Hanxing^{1,2}

(1. College of Science, Shanghai University, Shanghai 200444;

2. Dept. of Mathematics and Statistics, Shanghai Lixin University of Commerce, Shanghai 201620)

【Abstract】 In AODV routing protocol, an intermediate node can reply a RREQ when it has a route with a sequence number that is greater than or equal to that contained in the RREQ, no matter how many packets are waiting to send in it. These routes' performance may degrade because local congestions may incur on those nodes with the increase of offered load. This paper proposes an improved-AODV routing protocol, which only allow the intermediate nodes whose sending queues are not too long can reply a RREQ or locally repair a route to decrease local congestions. The simulation results show that improved-AODV performs better than AODV.

【Key words】 Ad hoc network; AODV routing protocol; Routing; TTL

Mobile Ad hoc network(MANET)是由多个移动节点组成的一种无基站的无线网络。网络中距离小于无线传输半径的两个节点可以直接通信, 距离大于传输半径的两节点只能通过其它中间节点对数据分组的转发来通信。由于节点的移动, 网络拓扑会不断变化, 传输数据的路由会由于节点的移动而需要不断地重建或更新。因此, 路由协议的设计是ad hoc网络中最有挑战性的问题之一^[1]。

目前, Ad hoc网络路由协议主要可以分为表驱动路由协议、按需路由协议和混合式路由协议 3 类^[3]。表驱动路由协议与有线网络中的路由协议类似, 如DSDV、FRS、OLSR等, 每个节点维护一张完整的路由表, 并通过相邻节点周期地交换路由信息来维护和更新这张路由表。如果一个节点有数据要发送到其它节点, 只需在自己的路由表中查找目标节点的下一跳, 然后把数据传给下一跳, 下一跳收到数据包后, 再转发给自己路由表中的下一跳, 直到目标节点。由于并不是每对节点之间都始终有数据需要传输, 因此路由表中的路由条目大多不是当前所需要的, 维护和更新这样的路由表会浪费大量宝贵的无线带宽。与表驱动不同, 按需路由协议只建立和维护传输数据所需要的路由, 不参与传输数据的节点不需建立相应的路由表。如果一个节点在路由表中找不到需要的条目, 就通过泛洪法来新建一条。虽然临时创建路由过程会带来一定的首次传输延时, 但只建立和维护需要的路由表项可以节省大量的无线网络资源, 因此目前大多数路由协议都是按需路由协议, 如DSR、AODV等。混合式路由协议是按需路由和表驱动路由由两种方式的结合, 它把整个网络划分为若干小区, 小区内采用表驱动路由, 小区之间则采

用按需路由方式连接, 如ZRP、DST、DDR都是混合式路由协议。

1 AODV 路由协议及其不足

AODV路由协议是基于距离向量的按需路由协议^[2,4], 它的主要特点是使用系列号来标识一条路由的新旧程度, 同时也利用系列号来避免路由环的产生。为了减少传输数据的首次延时, AODV允许具有积极路由的中间节点直接响应路由请求。当节点探测到路由失败后, 该节点可以先进行局部修复, 修复不成功, 再发送错误报告给上游节点, 以便源节点重新建立路由。

(1)路由发现过程

当源节点有数据要发送给目标节点而路由表中又没有相应的有效路由时, 源节点就广播一个路由请求包(RREQ), 为了避免RREQ不必要的大范围广播, AODV采用扩展环搜索技术, 设置路由请求的生存时间(time to live, TTL)值, 一次请求没有响应, 即没有收到相应的回答(RREP), 则再次广播一个路由请求, 并增加TTL值和广播号, 这一过程持续到发现路由或者TTL值达到允许的最大值为止。广播号是为了减少对广播分组的重复转发和处理, 节点直接丢弃收到的重复广播。

当中间节点收到路由请求后, 如果它没有到目标节点的

基金项目: 国家自然科学基金资助项目(10471088, 60572126)

作者简介: 高圣国(1969-), 男, 博士生, 主研方向: 最优化理论, Ad hoc 网络; 王汉兴, 博士后、教授、博导

收稿日期: 2006-02-28 **E-mail:** gsg1688@163.com

积极路由,就广播不重复的路由请求并建立到源节点的反向路径,反向路径可以用来发送数据;如果中间节点确定自己有到目标节点的积极(有效的)路由,并且路由中的目标节点系列号大于或等于路由请求中的系列号,它就直接沿反向路径向源节点单波路由回答分组(RREP)并通知目标节点以保证路由是双向的。

目标节点收到路由请求后,不再广播路由请求,它先建立反向路径,产生一个 RREP, RREP 中含有最新的系列号等信息,沿反向路径单波给源节点。中间节点和源节点在收到 RREP 后会建立到目标节点的路由,并更新系列号等有关的信息。源节点收到 RREP 后即建立路由并开始传输数据。

图 1 中,节点 1 要发送数据给节点 5,如果没有可用的积极路由,就发起一个路由发现过程,节点 1 广播一个 RREQ 分组,分组中含有已知的节点 5 的最大的系列号,如果没有节点 5 的系列号,系列号就等于 0。广播可以沿不同的路径转发到节点 5,节点 5 收到第 1 个 RREQ 后,便建立反向路径并沿反向路径发送一个路由回答包(RREP),当 RREP 回传到节点 1 后,节点 1 到节点 5 的路由便建立起来并开始传输数据。

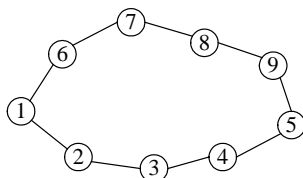


图 1 路由发现过程

如果节点 1 在启动路由发现过程之前,节点 3 已经在向节点 5 发送数据,并且这条路由的系列号是很新的,即大于或等于 RREQ 中的系列号,那么当节点 3 收到节点 1 的 RREQ 后,就不会再转发 RREQ,而直接向节点 1 发送 RREP,节点 1 收到 RREP 后建立路由并发送数据,数据传输的实际路线是 1-2-3-4-5。中间节点响应路由请求,显然减少了首次传输数据的延时,但显著地增加了路段 3-4-5 上的负荷,而另一条可用的道路 1-6-7-8-9-5 却空闲。

如果节点 3 不是正在对节点 5 发送数据,而是对别的节点发送数据,那么节点 3 会转发 RREQ,经过节点 4 到节点 5,节点 5 收到 RREQ 后,向节点 1 单波一个 RREP 分组,节点 1 收到 RREP 后,就可以建立到节点 5 的路由,即 1-2-3-4-5,并发送数据。如果节点 3 原有的发送任务就比较重,新建路由上传来的数据必然会加重节点 3 的负荷,导致丢包和延时的增加,降低网络的效率。因此,在这种情况下,节点 3 已不适合再参与新路由的建立。

(2)路由维护

为了维护路由,及时发现因节点移动或其它原因而中断的路由,每个节点周期地广播 HELLO 消息,HELLO 消息的生成时间即 TTL 值为 1,因此只能在相邻节点间传播。一个节点收到一个 HELLO 就可以新建一个邻居条目或者知道一个邻居与自己依然保持连接。如果在一定时间内收不到一个邻居的 HELLO 消息,则认为该邻居与自己不再连接,以这个节点为下一跳的路由都不能再用来传送数据,因此将这些路由设置为无效状态。AODV 路由协议允许局部维修,这个节点将启动路由发现过程,广播 RREQ 以便建立新路由,如果在给定时间里能重新建立起有效路由,就接着发送数据,如果建立路由不成功,则向上游发送 RRER。路由失败后先

进行局部维修可以减少数据传送的延时,可以减少上层控制和网络负荷。然而如果发起路由维护的节点的负荷很大,即使路由由局部维修成功,也会由于该节点的局部负荷过大而使数据传输的效率依然不高甚至可能下降。

AODV 路由协议中,不管是在路由发现还是在局部维修时,源节点在收到多条来自不同路径的 RREP 后,都是通过目标节点的系列号来确定路由的,在系列号相同时,选择跳数较小的路由。这样的选择方式有导致网络负荷在某些路段上集中的趋势,即使存在一些路段负荷较轻也没有去利用,也使整个网络的负荷分布不平衡,局部路段的忙碌导致传输效率下降。

2 改进方法

对 AODV 路由协议的分析表明,在创建新路由和路由的局部维修时,可以利用已有的路由,而这样的路由可能正在传输数据,这就可能加重这些路段的负荷,使其出现拥塞,而空闲道路可能存在而又没有利用。为了克服 AODV 路由协议的上述不足,本文作如下两点改进:

(1)在路由发现过程,中间节点在收到路由请求后,根据自己的负荷状态来决定是否广播分组或直接响应。在 AODV 中,发送分组缓存队列的最大长度是 64,如果一个中间节点的缓存队列长度达到 40($64 \times 62.5\%$),则不再适合传输新的数据流,因此,它丢弃所收到的目的节点不是它本身的 RREQ,即不作为中间节点参与建立新路由,它即使有到目标节点的积极路由,也不直接响应路由请求。如果它的缓存队列长度小于 40,就按 AODV 协议的方式处理路由请求。在两种情况下,反向路由都建立并可以用来发送数据。

(2)当一个正在传送数据的节点探测到路由失败后,不是直接进行局部维修,而是根据自己当前的负荷大小来确定是局部维修还是向上游报错。如果负荷较轻,即发送队列长度小于 40,就进行局部维修;反之,如果发送队列长度达到 40,则直接报错,使源节点可以新建一条负荷较轻的新路由。

图 1 中,节点 3 正在向节点 5 沿路径 3-4-5 发送数据,现在节点 1 也要向节点 5 发送数据,但节点 1 没有可用的路由,于是就广播 RREQ,当节点 3 收到 RREQ 分组后,按 AODV 协议,节点 3 直接向节点 1 发送 RREP,节点 1 收到后建立起路由 1-2-3-4-5 并发送数据,这时路段 3-4-5 上的负荷明显加重。按改进后的协议,节点 3 的负荷如果较大,就不会响应路由请求,也不转发 RREQ,这样 RREQ 就会沿 1-6-7-8-9-5 这条路径到达节点 5,这样新建的路由就可以分散网络负荷,减少拥塞发生。

如果节点 3 不是正在对节点 5 发送数据,而是对别的节点发送数据,按 AODV 协议,从节点 1 到节点 5 的路由 1-2-3-4-5 依然可以建立,按改进后的协议,如果节点 3 原有的发送任务达到一定程度,就不响应路由请求,节点 3 在收到从节点 1 传来的 RREQ 后,不再广播而直接丢弃,这样不仅节约了网络带宽,更有利于选择负荷较轻的路由。

对 AODV 的改进主要是通过增加简单的路由选择机制来避免网络负荷在部分路段上集中,使新建路由避开负荷较重的节点和路段而选择相对空闲的节点和路段,从而均匀网络负荷,提高整个网络的传输效率。

3 仿真环境和结果

仿真是在 NS-2 下进行的,网络拓扑结构是一个包含 30 个移动节点的网络模型,各节点随机分布在 $1000\text{m} \times 500\text{m}$ 的平面矩阵区域内,并随机地以均匀分布在 $[0, 20\text{m/s}]$ 之间的速度向任意的目的地移动,到达目的地后,停留一个暂停时间,然后再随机地选择一个目的地和速度移动,如此反复,

直到模拟结束。每个节点的无线带宽是 2Mbps，无线发射半径是 250m。MAC 层协议采用的是 IEEE 802.11。模拟时间为 600s，暂停时间为 0，则节点到达目的地后不停留，暂停时间为 600s，则节点静止不动。信源采用随机生成的 CBR(恒定比特流)，每个包的长度是 512B，不同次实验中 CBR 每秒产生的包的个数从 4 个到 8 个不等。

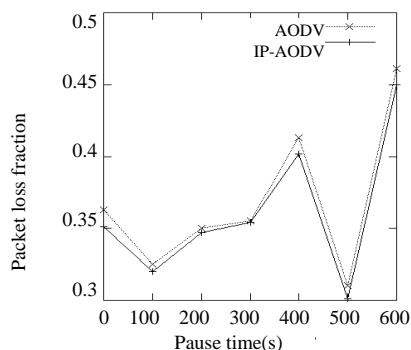


图 2 丢包率与节点暂停时间

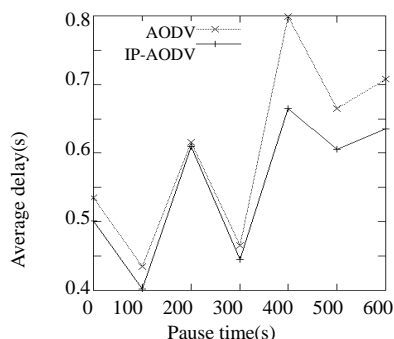


图 3 平均延时与节点暂停时间

本文主要是对暂停时间不同的节点运动场景进行模拟，

对丢包率和端到端的延时进行了数据分析。图 2 是节点在不同的暂停时间下 AODV 和改进的 AODV 数据丢包率的仿真结果，这表明改进后的协议的丢包率在各种情况下都有一定的减少，即数据传输成功率有所提高。图 3 是两个协议端到端的平均延时在不同的暂停时间下的比较，改进后的协议的平均延时有较明显的减少，总体平均减少达到 10%，并且模拟的结果还表明在网络负荷越大时，改进的效果越明显，因为在负荷很轻时，路由选择的机会和作用都会减少，因此改进的协议和原协议的路由选择方式几乎相同。

4 结论

本文对 AODV 路由协议进行了分析，针对 AODV 的路由发现和维修过程中路由选择时不考虑节点的负荷而可能导致局部路段上传输任务过重，在网络负荷较大时传输效率下降进行了简单的改进。改进后的协议根据自己当时的发送任务的大小可以决定是否参与建立一条新的路由，使新建路由尽可能选择负荷较轻的路段。仿真结果表明，改进后的协议能减少丢包率和端到端的延时，在网络负荷较大时效果更加明显。

参考文献

- 1 Chlamtac I, Conti M, Liu J. Mobile Ad hoc Networking: Imperatives and Challenges[J]. Ad hoc Networks, 2003, 1(1): 13-14.
- 2 Perkins C, Belding-Royer E, Das S. Ad hoc On-demand Distance Vector Routing[Z]. <http://www.ietf.org/rfc/rfc3561.txt>.
- 3 Abolhasan M, Wysocki T, Dutkiewicz E. A Review of Routing Protocols for Mobile Ad hoc Networks[J]. Ad hoc Networks, 2004, 2(1): 1-22.
- 4 Perkins C, Belding-Royer E, Das S. Ad hoc On-demand Distance Vector (AODV) Routing, Internet Draft[Z]. 2002. <http://www.ietf.org/internet-draft/draft-ietf-manet-aodv-11.txt>.

(上接第 107 页)

采集服务是实现无线子网管理的基础，通过被动侦听的方法，收集无线子网的拓扑信息、节点工作状态信息、网络可达信息，传输服务中的报文排队服务信息，响应其它服务向它申请的采集要求。

传输服务扩展了传统的路由管理功能，除了单纯地依据目的地址和路由表进行报文投送之外，能根据业务类型选择传输路径、传输协议（TCP、UDP 还是 RTP 等）传输的优先级、重传策略；根据链路的状态和传输队列长度调整传输策略，在一定程度上解决流量工程问题。

配置服务接收经过安全审查后的网管主动包，依据主动包中的配置文件、脚本或代码在运行环境内创建新的服务，修改原有服务的配置，改变原服务的服务能力和策略，通知 SNMP 服务创建和修改扩展 MIB，定制网络业务管理功能。

3 实验结论与未来工作

本文提出的利用主动网络技术管理战术互联网无线子网的方法，特别是主动网管节点服务器的 SNMP 服务、传输服务和被动侦听无线子网相关信息的采集服务的原型，都已在基于 Windows 和 VxWorks 平台的原型系统和工程项目中得到一定程度上的验证。该方法突破了传统网管系统不适合管理窄带无线子网的限制，克服了网络拓扑和设备状态感知和配置的困难，同时为流量工程和业务管理提供了新思路。

在实验和实际工程项目中也暴露了一些问题，今后还需要进一步研究和完善。简化各管理服务的功能和依赖关系的描述，特别需要简化配置服务的实现；加强安全认证与审计服务；完善运行环境，引入面向服务体系结构的理念，通过服务总线来改进数据总线和资源动态申请与管理；研究主动网管节点服务器的部署与协作问题，进一步提高战术互联网及其无线子网的管理效率。

参考文献

- 1 Rayes A, Sage K. Integrated Management Architecture for IP-based Network[J]. IEEE Communications Magazine, 2000, 38(4): 48-53.
- 2 Raz D, Shavitt Y. Active Network for Efficient Distributed Network Management[J]. IEEE Communications Magazine, 2000, 38(3): 138-143.
- 3 Raz D, Shavitt Y. Toward Efficient Distributed Network Management[J]. Journal of Network and Systems Management, 2001, 9(3): 357-361.
- 4 崔 灿, 常义林. 主动网络管理技术的研究[J]. 南京理工大学学报, 2004, 28(1): 85-89.
- 5 Kawamura R, Stadler R. Active Distributed Management for IP Networks[J]. IEEE Communications Magazine, 2000, 38(4): 114-120.
- 6 刘福杰, 常义林, 沈 中等. 一种自组织网络管理实现方法的研究[J]. 西安电子科技大学学报, 2004, 31(2): 182-185.

