一种移动 Ad hoc 网络分簇算法及性能分析

高 邇[1] 慕德俊[2] 张 力[2] 张国庆[2]
(西北工业大学计算机学院 西安 710072)[1] (西北工业大学自动化学院 西安 710072)[2]

摘 要 提出了一种新的 Ad hoc 网络分簇路由算法。该算法引入异常度的概念,根据异常度判断何时启动簇结构调整,如何使原簇中保留较多的节点,提高分簇结构稳定性。分析及实验表明,该分簇算法具有簇首的更换频率低和高稳定性的特点。

关键词 Ad hoc 网络,分簇,路由算法,稳定性
中图法分类号 TP393 **文献标识码** A

Novel Clustering Algorithm in Ad hoc Network and Performance Evaluation

GAO Li[1] MU De-jun[2] ZHANG Li[2] ZHANG Guo-qing[2]
(College of Computer, Northwestern Polytechnical University, Xi'an 710072, China)[1]
(College of Automation, Northwestern Polytechnical University, Xi'an 710072, China)[2]

Abstract We proposed a new Clustering Routing algorithm. It introduced the conception of "Exception Degree" to define when to start cluster reconfiguration and how to retain nodes as many as possible in the original cluster to improve stability. Analysis and simulation show that the algorithm is effective with low replacement frequency of cluster head and high stability of cluster structure.

Keywords Ad hoc network, Clustering, Routing algorithm, Stability

1 引言

Ad hoc 网络是一种不依赖任何固定设施的移动无线自组织网络,具有分布性、动态性、自治性、移动性和异构性等特点,应用范围广泛,特别适合于战术通信、抢险救灾、临时集会等突发性、临时性场合。Ad hoc 网络具有两种体系结构:平面结构和分级结构。平面结构又称对等结构,所有节点的功能相似、地位相同,网络相对鲁棒性较强,在节点大量增加的情况下,特别是在移动的情况下,存在处理能力弱、控制开销大、路由经常中断等缺点,使得网络性能急剧下降,因此它主要用于中小型网络。分级结构采取分簇的方法,将整个网络划分为若干个簇,每个簇由一个簇首和多个成员节点组成,簇头作为簇的中心点负责簇结构的稳定和重组,通过簇头组成骨干网,可以实现跨簇通信。分级结构的优点是网络的可扩充性好、网络的规模不受限制、可以通过增加簇的个数或网络的级数来提高网络的容量;在相同网络规模的条件下,分级结构的路由和控制开销要比平面结构小,并且该结构容易实现移动性管理和网络的局部同步。但建立和维护簇结构时引入一定的开销,当网络拓扑变化时,特别是节点移动较强时,簇结构更新频繁,引入大量的维护开销,网络性能下降。

本文提出了一种新的分簇算法,该算法是基于最小 ID 的改进算法,兼顾最小 ID 算法成簇简单的优势,引入异常度的概念,在网络拓扑变化时根据异常度做簇结构调整,使网络尽量保持原有结构,原簇中保留较多的节点,簇节点更换簇首的频率较低,提高分簇结构稳定性,大大减少维护簇结构的控制开销,有效地提高了网络性能。

2 最小 ID 算法及最高节点度算法分析

2.1 最小 ID 算法

最小 ID 算法是一种简单的分簇算法。为每个节点分配一个唯一的 ID,选取 ID 最小的节点作为簇头。其邻居节点成为该簇头所在簇的成员,不再参与簇头的选举。这种分簇算法计算量小,无需进行复杂的簇结构调整判断,实现方便,算法收敛较快。但当在节点移动较频繁、需要频繁地执行分簇算法来维护网络拓扑结构,其带来大的开销,特别是两个簇首由于移动进入彼此通信范围内时,即出现所谓的簇合并问题。最小 ID 的算法性质,将导致整个簇的节点放弃其当前簇首,重新运行成簇的检测和算法过程。作为路由协议的基础算法,这种簇结构重建的开销是不能允许的。

2.2 最高节点度算法

针对最小 ID 算法在节点移动时可能产生的簇结构变化频繁问题,Lin 和 Gerla 等人提出了一种改进簇结构调整机制的最小 ID 分簇思路。当簇结构发生变化时,不再按照最小 ID 规则重新分簇,而是将连接度最大节点及其一跳邻节点留

到稿日期:2008-06-24 本文受航空科学基金项目(05F53029)资助。

高 邇(1969-),女,讲师,博士研究生,主要研究领域为移动自组网、网络安全,E-mail:gaoli@nwpu.edu.cn;慕德俊(1963-),男,教授,博士生导师,主要研究领域为网络安全、系统控制理论;张 力(1982-),女,硕士研究生,主要研究领域为移动自组网;张国庆(1982-),男,博士研究生,主要研究领域为移动自组网、网络安全。

在原簇中，每个节点周期性地检测本机是否在簇内具有最大节点度节点的一跳范围内。若不在，则进行离簇操作。

该算法的优点在于网络中簇的数目较少，即源目的节点对之间的平均跳数较少，从而减少了分组往返时延。该算法从原理上能够提高分簇结构调整时的稳定性，但其所要求的前提条件在实践中却不具备必然可行性，表现在以下两个方面：

1）当簇结构中具有最大节点度的节点同时有两个或多个时，则节点对于离簇的检测和判断度必须引入辅助的判断条件，作为分布式算法，对节点间的信息同步及判断准确性提出了更高的要求；

2）当簇中存在节点脱离簇内其它节点通信范围，尤其是存在多个节点同时脱离时，节点将无法获知或准确推测其余节点的节点度，对于最大节点度的判断势必是失败的，因此也就无法进行正确的簇结构调整处理。

3 改进的分簇算法

3.1 算法设计目标

算法目标应为在满足最小 ID 分簇结构的前提下，从节点之间的链路状态和节点职能的角度出发来调节簇结构，将尽可能多的节点留在原簇内，簇首担任的平均时间较长，节点更换簇首的频率较低，提高分簇结构稳定性；算法要能够明确地判断出何时开始簇结构调整阶段，保证每个节点都获取必要的其它节点信息来进行簇结构调整处理；并且即使是在节点高速运动的状态下，这种分簇算法都易于实现。

3.2 算法相关定义和假设

定义 1（异常链路） 当同一分簇的两个节点间距离等于或超过 3 跳时（无法抵达时认为距离为 255 跳），则这两个节点间存在异常链路，即节点移动导致簇结构改变。同一簇的两个节点现在处于不同簇，需要进行簇结构调整。

定义 2（异常度） 对于某个节点来说，异常链路的条数就是异常度。若簇存在异常度大于 0 的节点时，则认为该簇结构发生变化，需要进行簇结构调整。整个簇不再有异常度大于 0 的节点，簇结构进入稳定状态，不需要进行簇结构调整。

算法还满足以下假设：

（1）在网络初始化时，每个节点可以通过交互控制消息知道其邻居节点的 ID 号。

（2）一个节点发出的消息能够被其所有的邻居节点在很短的时间内正确收到。

（3）在分簇算法执行期间，网络的拓扑不发生改变。

3.3 算法思想

改进算法分为成簇阶段算法和簇结构维护算法。由于最小 ID 分簇算法有成簇快速、简单、高效、不依赖任何外部辅助条件的优点，在成簇阶段仍然沿用最小 ID 分簇方法的思路，作为面向实现的算法。当簇结构发生变化时，改进算法基于簇结构稳定性、减少簇结构维护的开销出发，将尽可能多的节点留在原簇内，使簇首担任平均时间较长、簇首更换频率降低。在簇结构维护阶段采用基于异常度的维护算法，根据异常度来判断哪些节点出现异常链路，不在原簇中。仅对移出原簇的节点做离簇处理，簇首及其他节点留在原簇中，维持了簇结构的稳定，减少了簇头更换引入的维护开销。

3.3.1 异常度获取

分簇结构调整是否需要进行，要根据异常度来判断。要使节点能够准确地判断簇内各节点的异常度，必须使每个节点都具有簇内所有异常链路的信息。考虑到簇结构变化后，节点和节点间可能不再连通，算法设计了两种方法来获取异常链路信息。

1）异常链路泛洪：当节点发现邻节点表中存在本节点到其它节点的异常链路时，将立即广播该消息。其余节点接收到这个广播消息后，处理并立即再次广播这个消息。该泛洪消息的生命周期为发起节点的异常链路中除了 255（断路）以外的最大跳数，这样就可以保证异常链路泛洪消息可以到达簇内所有与发起节点仍然连通的节点。节点泛洪时应判断该异常链路是否已经被广播。若已被广播，则不再重新泛洪。

2）异常链路推测：当节点发现本机存储的异常链路信息中存在断路的链路时，可知该链路的一端必然与本机也无法连通，即该节点的异常链路消息无法泛洪到本节点。此时应对该节点的异常链路进行推测，对所有的已知以该节点为终点的异常链路进行统计，得到该节点的异常链路的一个子集或者真子集。通常对不连通节点推测而统计出的节点度小于或等于该节点真实的节点度。

3.3.2 簇结构调整处理方法

1）异常节点离簇处理原则

簇存在异常度大于 0 的节点时，需要对异常度大于 0 的节点做离开簇处理，处理原则如下：

① 若异常度表中具有唯一的最大值，则将对应的 ID 做离簇处理。若该 ID 为本节点，则更改本节点为无簇首状态，相应地清理相邻节点表、异常链路表和异常度表，并退出该异常处理过程。否则，除了将该节点从同簇节点表中清除以外，还需要相应地丢弃异常链路表中对应该节点的项，异常度表也随之更新。

② 若异常度表中的最大值不唯一且除了这几个最大值节点外，没有其它的节点，通常为簇内存在断路的情况，则对所有取最大值的不连通节点进行离簇处理，离簇处理同①。

③若异常度表中最大值不唯一且异常度表中还有其它节点，则对所有取最大值的节点进行离簇处理。若取最大值的节点中包括本节点，则首先处理本节点，处理的过程同①。

2）两簇合并的处理方法

若两个簇首彼此进入对方的一跳通信范围，由于两个簇首不能直接相邻，两个簇将进入合并处理。为了使簇合并后新簇结构中保留尽可能多的节点，簇中现有节点数目较多的一方将成为新的簇首，另一个簇首将在簇内泛洪簇合并消息。簇内所有节点在接收到簇合并消息时，更改簇首为新簇首，新簇进行异常度检测和相应处理。

3）簇中添加新节点处理方法

当一尚未确定簇首的节点进入某簇首一跳范围内时，进行入簇处理。由于结构调整算法遵守的规则为异常度等于 0 时簇结构稳定，节点欲加入该簇，先确认节点和簇内所有节点都在两跳范围内，保证节点加入该簇后不会引起簇结构的再次调整，再进行要入簇处理。

3.3.3 簇结构调整算法实现过程

1）周期性地检查同簇邻节点信息或收到邻节点更新消息时，节点启动异常链路检测。

2) 若存在以本机为起点的异常链路且尚未泛洪过，则将包含这些链路的异常链路泛洪消息泛洪到整个簇，更新当前异常链路表。

3) 若存在不连通节点，则对异常链路表进行异常链路推测操作。

4) 按照簇结构调整法则对节点进行离簇检测和处理。若本机被离簇，则清理相关数据结构。

5) 重新检测异常度表。若仍存在异常度大于 0 项，则跳至步骤 4)。

6) 若簇首被进行离簇操作，则按照最小 ID 原则在剩余节点中重新选择簇首；若节点接收到新的来自其它节点的异常链路泛洪消息，则将该泛洪数据中的异常链路信息添加到本机的异常链路表中，并开始运行步骤 2)—步骤 6)。

4 算法分析和仿真

4.1 算法分析

图 1 是最小 ID 算法与改进算法簇结构调整结果对比图。图 1(a)是根据最小 ID 算法生成初始簇结构。当簇首节点移动，尽管这几个节点仍然相互连通，但根据最小 ID 算法同一个分簇内的节点立刻被重新划分成图 1 (b)中的 3 个新簇。若节点 5 往返移动，则分簇结构变化将相当剧烈。

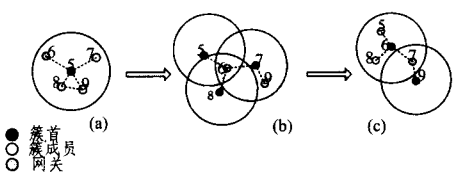

图 1 最小 ID 算法与改进算法簇结构调整结果

对于图 1(a)中分簇结构，各节点异常度均为 0。而对于图 1 (b)，由于节点移动出现异常链路，异常链路和异常度如表 1 和表 2 所列。对异常度最大的节点 9 做离簇处理，簇结构调整为两个分簇，如图 1 (c)所示，大部分节点都保留在原簇中，簇结构维持了一定的稳定性。

表 1 异常链路

Start Node	End Node	Hops
5	9	3
8	9	3
9	5	3
9	8	3

表 2 异常度

Node	Exception degree
5	1
8	1
9	2

图 2 的初始簇结构同图 1(a)。当簇内节点移动时，分簇结构更改而导致不同的新分簇结果，分析不同算法的稳定性能。图 2(a)，(b)，(c)和(d)各图左边为最小 ID 分簇算法调整结构后的结果，右边为改进算法调整结构的结果。

由图 2 可以看出，对于图 2(a)、图 2(b)场景，最小 ID 分簇法将簇结构重新分成 3 个分簇，改进后算法则重新分为两个分簇，且占绝对数目优势的节点仍保留在原簇中，避免了大量的重分簇开销。这种结果的优势不光表现在对簇结构的维持，更表现在对拓扑再次变化的适应性。以图 2(a)为例，若节点 5 重新移动到初始位置，那么最小 ID 分簇法将重新导致节点 7，8 放弃当前的簇首位置，加入节点 5 所在簇，节点 9 也要从节点 7 所在簇的簇成员变成无簇节点，再变成节点 5 所在簇的成员节点。而对于改进后的最小 ID 分簇法，节点 5 的移动只要还与节点 6 在一跳范围，则簇结构不需要发生任何变化；而对于图 2(b)，若节点 5 重新移回原位，改进算法只需判断当节点 5 重新同时可达节点 6，7 时，节点 5 就可以十分简单地加入原簇了，而最小 ID 分簇法带来的将是再一次的簇结构大规模调整。场景图 2(c)和图 2(d)中改进算法则和最小 ID 法产生同样的调整结果。经过前述算法分析可知，改进最小 ID 法的簇结构调整势必要么与最小 ID 等同，要么优于最小 ID。

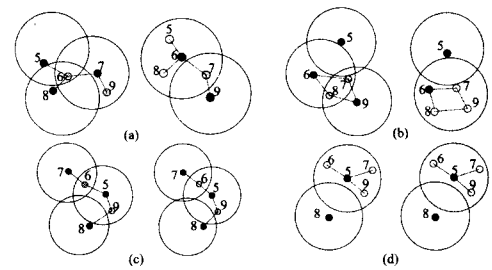

图 2 不同算法分簇结果对比图

4.2 实验仿真

在网络仿真工具 NS-2 中对最小 ID 法和改进分簇法分别进行仿真。选择的仿真场景为 1000m×1000m 的正方形区域，节点为 50 个。本仿真通过在测试脚本中设定物理层的 RXThreshhold 来调整节点的通信距离，并以此来更改网络中节点间的连通情况。节点的移动模型为“random waypoint”，运动速度范围设为 0～20m/s，仿真时间总长度为 400s。因为主要测试分簇算法的稳定性，场景内加入一个 CBR 数据业务，任意选在两个节点间进行。此外，协议的周期性广播节点信息间隔时间为 5s。

仿真考察的两项内容分别是节点在 100s 内平均更换簇首的次数和每个簇首节点的平均担任时间。仿真结果如图 3、图 4 所示。

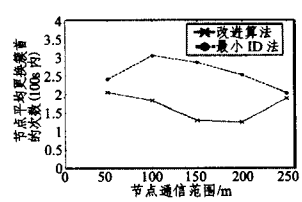

图 3 100s 内节点平均更换簇首的次数

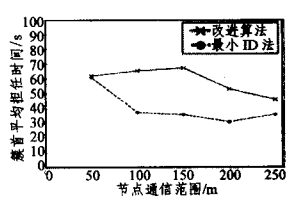

图 4 簇首平均担任时间

从这两个图可以看出，改进算法相比于最小 ID 分簇算法，平均更换簇首的频率要低，而簇首担任的平均时间则更长。节点更换簇首的频率低，说明节点加入、离开旧分簇、加入新分簇的次数少。簇首担任的时间长，说明簇结构稳定；从图中还可以看出，在节点通信范围较大或较小时，两种算法间的差距均小于在一般范围内的情况。考虑到当节点通信范围较大或较小时，节点本身移动对节点间连通情况影响较小，出现这种现象是合理的。实验数据表明，改进的分簇算法确实具有更好的稳定性能。

(下转第 128 页)

[16] Heuzeroth D, Holl T, Hogstrom G, et al. Automatic Design Pattern Detection [C] // Proceedings of the 11th International Workshop on Program Comprehension. Washington: IEEE Computer Society Press, 2003: 94-103

[17] Högström G. Detecting Communication Design Patterns by Both Static and Dynamic Analyses [D]. 03016. ISSN 1650-2647. School of Mathematics and Systems Engineering. Växjö University, April 2003

[18] Wendehals L. Improving Design Pattern Instance Recognition by Dynamic Analysis [C] // Proceedings of the International Conference on Software Engineering. Workshop on Dynamic Analysis. Washington: IEEE Computer Society, 2003: 29-32

[19] Smith J M C, Stotts D. SPQR: Flexible Automated Design Pattern Extraction from Source Code [C] // Proceedings of the 18th IEEE International Conference on Automated Software Engineering. Washington: IEEE Computer Society, 2003: 215-224

[20] Smith J M C, Stotts D. Case Studies in Automated Design Pattern Detection in C++ Code using SPQR[R]. TR05-013. Department of Computer Science, Univ. of North Carolina at Chapel Hill. June 2005

[21] Nickel U, Niere J, Zündorf A. The FUJABA Environment [C] // Proceedings of the 22nd International Conference on Software Engineering. New York: ACM, 2000: 742-745

[22] Arcelli F, Masiero S, Raibulet C, et al. A Comparison of Reverse Engineering Tools Based on Design Pattern Decomposition // Proceedings of the 2005 Australian Software Engineering Conference. Washington: IEEE Computer Society, 2005: 262-269

[23] Antoniol G, Casazza G, Penta M D, et al. Object-oriented Design Pattern Recovery[J]. The Journal of System & Software. 2001, 59(2): 181-196

[24] Niere J, Schafer W, Wadsack J P, et al. Towards Pattern-Based Design Recovery [C] // Proceedings of International Conference on Software Engineering. New York: ACM, 2002: 338-348

[25] Eide E, Reid A, Regehr J, et al. Static and Dynamic Structure in Design Patterns [C] // Proceedings of the International Conference on Software Engineering. New York: ACM, 2002: 208-218

[26] Derrick J, Boiten E. Refinement in Z and Object-Z: Foundations and Advanced Applications [M]. UK: Springer-Verlag London Limited, 2001

[27] Kerievsky J. Refactoring to Patterns [M]. US: Pearson Education Inc., 2005

[28] Dietrich J, Elgar C. A Formal Description of Design Patterns Using OWL [C] // Proceedings of the 2005 Australian Software Enginee-ring Conference. Washington: IEEE Computer Society, 2005: 243-250

[29] France R B, Kim Dae-Kyoo, Ghosh S, et al. A UML-Based Pattern Specification Technique [J]. IEEE Transactions on Software Engineering, 2004, 30(3): 193-206

[30] Kim Soon - Kyeong, Carrington D A. Using Integrated Meta - modeling to Define OO Design Patterns with Object-Z and UML [C] // Proceedings of the 11th Asia-Pacific Software Engineering Conference. Washington: IEEE Computer Society, 2004: 257-264

[31] Mitchell R, Mckim J. Design by Contract [M]. US: Addison - Wesley, 2002

[32] Gabbay D M, Hodkinson I M, Reynolds M A. Temporal Logic: Mathematical Foundations and Computational Aspects [M]. Volume 1. USA: Oxford University Press, 1994

[33] Pettersson N. Measuring Precision for Static and Dynamic Design Pattern Recognition as a Function of Coverage [C] // Proceedings of the Third International Workshop on Dynamic Analysis. New York: ACM, 2005: 1-7

[34] Heuzeroth D, Mandel S, Lowe W. Generating Design Pattern Detectors from Pattern Specifications [C] // Proceedings of the 18th IEEE International Conference on Automated Software Engineering. Washington: IEEE Computer Society, 2003: 245-248

[35] Baader F, Calvanese D, McGuinness D L, et al. The Description Logic Handbook: Theory, Implementation, and Applications [M]. UK: Cambridge University Press, 2003

[36] Collins A M, Quillian M R. Retrieval Time from Semantic Memory [J]. Journal of Verbal Learning and Verbal Behavior, 1969, 8 (2): 240-247. Reprinted in Computation and Intelligence. Edited by G. F. Luger. Menlo Park, CA: AAAI Press: 191-201

[37] 贾育，顾毓清. 基于领域特征空间的构件语义表示方法[J]. 软件学报，2002，13(2)：311-316

[38] Jezequel J M, Train M, Mingins C. Design patterns and contracts [M]. USA: Addison-Wesley, 1999

[39] Heuzeroth D, Holl T, Löwe W. Combining Static and Dynamic Aalyses to Detect Interaction Patterms // IDPT'02. Pasadena, CA

(上接第 84 页)

结束语 本文提出了一种新的基于最小 ID 的改进算法，算法兼顾最小 ID 算法成簇简单的优势，引入异常度的概念，在网络拓扑变化时根据异常度做簇结构调整，通过改进簇结构变化时调整机制来达到提升分簇结构稳定性，减少簇结构变化远距离簇间路由发现时所需时间及开销。算法性能分析和实验表明了改进算法簇头平均更换次数少，平均担任时间长，簇结构稳定性得到提升。

参 考 文 献

[1] 王金龙，等. Ad Hoc 移动无线网络[M]. 北京：国防工业出版社，2002

[2] Chiang C-C, Wu H-K, Liu W, et al. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel [A] // Proceedings of SICON. 1997

[3] Lin C R, Gerla M. Adaptive Clustering for Mobile Wireless Networks [J]. IEEE Journal on Selected Areas in Communications, 1997, 15(7)

[4] Yu J, Chong P. A survey of clustering schemes for mobile Ad hoc networks[J]. IEEE Communications Surveys, 2005, 7(1): 32-48

[5] Basagni S. Distributed Clustering for Ad Hoc Networks [A] // Proc. of Int'l Symp. on Parallel Architectures, Algorithms and Networks[C]. 1999: 310-315

[6] Basu P, Khan N, CLittle T D. A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks [A] // Proc. of IEEE ICDCS 2001 Workshop on Wireless Networks and Mobile Computing [C]. 2001: 413-418

[7] 程伟明，周新运. 一个用于 Ad Hoc 网络的分簇方法[J]. 计算机学报，2005，5(28)：864-868

[8] 王海涛，郑少仁，刘晓明. 移动 Ad hoc 网络中的分簇算法[J]. 解放军理工大学学报，2004，5(3)：28-32

[9] 徐浩，慕德俊，李立欣. 一种 Ad hoc 网络按需式分簇路由算法[J]. 计算机工程与应用，2007，43(14)：3-6