Arab Gulf Journal of Scientific Research, 25 (4): 171- 178 (2007)

Research Paper Ref. No 2453

Fault-tolerance of Cluster Management in (MANETs) by Assistant and Mobile Agents

تحسين مُديَات التوقفات التي تحدث في إدارة مجاميع شبكات المحمول (MANETs) بواسطة المُسَاعد ووكلاء المحمول

Hatem Hamad

حاتم حمد

Department of Electrical and Computer Engineering

The Islamic University of Gaza, Palestine,

E-Mail: hhamad@mail.iugaza.edu

ABSTRACT: Most researches today trend to clustering in ad hoc networks as a solution to the management problem in flat ad hoc networks. Clustering aims to choose suitable nodes to lead the network. i.e., cluster heads. When a cluster head fails, re-clustering is needed. However, this will be costly due to the characteristics of the ad hoc networks such as mobility and power exhaustion. In this research, we developed a fault tolerance mechanism to avoid re-clustering and other drawbacks that result from losing the Cluster Head (CH). A Mobile Agent (MA) will be responsible of carrying out the jobs of the CH. i.e., the CH will host the MA which will be the real manager of the cluster. Furthermore, the closest node to the CH will be chosen as an assistant and the MA will update the assistant. Each of the CH and the assistant will have Remote Objects (RO) to communicate with each other. The main jobs of the assistant are to temporarily replace the CH when fails; then it will choose the next CH based on the clustering algorithm. Simulation results show an enhancement of the performance of using an assistant, MA and RO.

Keywords: *Cluster, ad hoc networks, Fault-tolerance, assistant, mobile object, remote object.*

المستخلص: تميل معظم توجهات الباحثين في الوقت الحاضر إلى استخدام طريقة المجاميع (العنقدة) في شبكات (ad hoc) الموزعة أفقيا، وذلك لحل مشاكل إدارة هذه الشبكات، وتعتمد هذه الطريقة (المجاميع أو العنقدة)، على اختيار مجموعة مناسبة لكي تكون رأس المجموعة (Cluster Head)، مما يعني أنه وعند حدوث أي توقف أو عطل برأس المجموعة تكون هنالك حاجة ماسة إلى إعادة تشكيل المجاميع (العنقدة). وبما أن هذا الأسلوب مُكلف نسبة لخصائص شبكات (ad hoc)، تم في هذا البحث تطوير طريقة تسمح بمدى معين للتوقفات أو الأعطال حتى يتم تجنب عملية أعادة تشكيل المجاميع ، بالإضافة الي تفادى المساؤى التي قد تحدث نتيجة لفقدان رأس المجموعة (العنقدة). يشير البحث إلى أن وكالة المحمول (Mobile Agent) سوف تكون المسئولة عن وظيفة رأس المجموعة، كما سوف يكون لرأس المجموعة ومساعده (Assistant) كائنات البعيدة (Remote Objects) تقوم بالاتصال مع بعضها لاسيما وأنه من أهم واجبات المساعد (Assistant) استبدال رأس المجموعة عند حدوث أي عطل أو توقف. وعليه يتم اختيار المساعد (Assistant) لرأس مجموعة آخر وبديل، وذلك اعتمادا على خوارزمية المجاميع. تشير نتائج النمذجة إلى تحسن الأداء باستخدام المساعد (Assistant) ووكلاء المحمول (Mobile Agents) وكذلك الكائنات البعيدة (Remote Objects).

كلمات مدخلية : *المجاميع ،العنقده، شبكات (ad hoc)، مُديات الأخطاء، المساعد، كائنات المحمول، الكائنات البعيدة.*

INTRODUCTION

Hierarchal architecture has been proved an efficient solution for management and communication problems in (ad hoc) networks because it minimizes information update overhead, reduces the use of network bandwidth and resources throughout the network (Bassagni,

1999). (MANETs) are collections of wireless mobile nodes that dynamically form a network without the need for any pre-existing network. i.e., they are self-organizing and self-configuring (Shang and Cheng, 2005). MANETs can be employed in situations that require a dynamic network topology where the communication may be supported by intermediate nodes (Perkins, 2001). Example of these situations are law enforcement operations, battle field and disaster recovery operations.

Flat architectures face scalability problems especially in MANETs with the increased network size and mobility. By grouping the nodes into clusters, we create hierarchies. i.e., abstracting topology; this is called clustering (Chen, *et al.* 2004). In clustering, some nodes are elected as cluster heads; these nodes act as the managers of the clusters. A node that belongs to more than one cluster is called gateways and remaining members are called ordinary members (Figure 1).

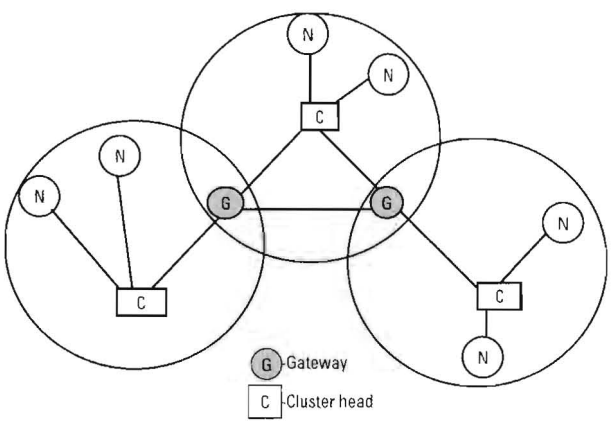

Fig. 1. Ad hoc Network Clustering.

CHs hold information and software needed to manage the cluster such as topology and routing information, so they are critical nodes in the cluster and loosing a CH suddenly due to mobility or power exhausting is a serious problem. In this paper we can enhance the clustering algorithm to improve MANETs efficiency.

The mobile agent technology and Remote Method Invocation (RMI) can play a vital role in the management of mobile ad hoc networks. Mobile agents are programs that travel from one node to another. After creating a mobile agent, it can carry its state and code to another node to continue its execution there or restart it (Satoh, 2003). RMI invokes object methods over a network. The server defines objects that client can use remotely and clients can now invoke methods of remote object as if it were a local object running in the same virtual machine as client (Aderounmu, *et al.* 2006). So, mobile agents may be employed as a manger to MANETs to gain benefits of both mobile agent technology and remote method invocation.

STATE OF THE ART

Currently, there is a growing interest in using mobile agents as part of the solution to implement more flexible and decentralized network architecture. Most research examples of the mobile agent paradigm as reported in the current literatures have two general goals: reduction of network traffic and asynchronous interaction. Some authors have suggested that agents can be used to implement network management and to deliver network services (Roy Choudhuril, *et al.* 2000).

Others introduced a state description for mobile agent systems on ad hoc networks, allowing agents to reason about how they communicate over the agent system's underlying network. The ability to make intelligent decisions about communications allows agents to achieve their goals more efficiently, and increase the security and survivability of an agent system. One application of this model includes a formal representation of information assurance for agent messaging on dynamic networks with possibly redundant routes. They used this framework to study the "compromised host" problem—specifically, how agents should react after an intruder has been detected. This is done by first quantifying the effect of the compromised host on the integrity of messages among agents; and then selecting a routing policy with respect to message integrity and the compromised host.

Lowest-ID with adaptive ID reassignment proposed that CHs are initially elected based on the time and cost-efficient lowest-ID method (Gavalas, *et al.* 2006). During clustering maintenance phase though, node IDs are re-assigned according to nodes mobility and energy status, ensuring that nodes with low mobility and sufficient energy supply are assigned low IDs and hence, are elected as CHs.

Cluster-based replication for large-scale mobile Ad-hoc networks proposed a replication scheme (Yu, *et al.* 2005). Distributed Hash Table Replication (DHTR) organizes all mobile nodes into non-overlapping clusters and builds a two-level distributed replica information directory on cluster heads to facilitate the propagation of query and update messages.

In previous works the researchers introduced ideas to avoid cases of losing the CH but they do not consider the long time to discover the failure of the CH which will affect the stability of the network due to problems in routing, out-of-date management code and data and the time consumed in reselecting new CH.

THE PROBLEM

The head node, CH, is the node that holds code and data needed to manage the cluster. When this node fails suddenly, member nodes will not be able to complete their jobs because the corresponding messages will not reach other nodes. The stability of the system will be in risk and the system will be damaged (Gupta and Younis, 2003). Therefore, it is very important to find a way to keep the information held at the CH and temporarily do the work till choosing the next CH. This can be achieved by saving the information and can be used when needed. That is, the code and data should still be valid for use when another node becomes a cluster head. In the next section we introduce our proposed solution.

PROPOSED SOLUTION

Mobility is a main factor affecting topology and route invalidation in MANETs. In addition, because mobile nodes depend on limited power supply, energy saving is another challenge. In addition to that, CHs are more important than ordinary nodes and loosing them due to mobility or power failure causes frequent re-clustering, therefore increasing control overhead. The proposed solution is considered as an enhancement for any clustering algorithm, i.e. any clustering algorithm can be chosen to formulate the structure of the network and then our solution is applied to get more efficient cluster management.

Traditionally, when a node needs a service, it has to broadcast a request over the cluster. Each node will receive the packet and process it. When the CH receives the Request packet, it responds to the node. These frequent broadcasts cause exhausting the network bandwidth and increasing the CPU time (Ahmed and Homayoun, 2005). Mobile agent technology provides important advantages due to their mobility of code, artificial intelligence, and improved data and network management possibilities (Ahmed and Homayoun, 2005; Beiszczad and Pagurek, 1998). By implementing the jobs in remote mobile agent software, nodes don't need to process each packet. When a node receives the packet, it only checks the destination address. If the destination address is its own, the node responds, else it will discard the packet, i.e. we address specific jobs. We use RMI to declare these methods and invoke them remotely over the network. This will save the network bandwidth and decreasing the CPU time.

In addition, the CH is still exposed to sudden failure or loss due to mobility or energy exhaustion. This node holds code and important data needed to manage the cluster and hence the network. Hence, failure of this node will cause re-clustering and drawbacks (Gavalas, *et al.* 2006). To improve the network performance, we must avoid the sudden failure of the CH and save management code and data. We can do that through assigning a node as a vice-CH or *assistant* to replace the CH when fails.

Based in the above ideas, we propose a solution by two steps:

a. Remote Object (RO): The CH will host two objectsthat carry out its jobs to manage the cluster. One Mobile Agent (MA), *manager*, will be responsible to carry out traditional jobs of the CH such as building routing tables, joining nodes to the cluster and maintaining the cluster formation (Bhaumik and Banyopadhyay, 2005). The second object, *CH-helper*, will be *Remote Object*. It implements new methods proposed in our solution to achieve fault tolerance that are defined below. Some of these methods will be declared as *remote* to enable Remote Method Invocation to them;

hence gaining advantages of both mobile agent technology and Remote Method Invocation. Moreover, since the MA can migrate between nodes, we gain another advantage which is to overcome the mobility of the nodes. i.e., it seems as we have static structure of nodes and remote mobile agents move between them.

b. The CH assistant: the CH will choose the closest node to it as an assistant. The main jobs of the assistant are to discover the failure of the CH; declare itself as a temporal CH and then will choose the next CH based on the clustering criteria. This node must be the closest to the CH to ensure that the assistant is able to deal with the existing structure and that the management information is still valid. The CH will send data and a copy of the *manager* MA to the assistant which will create a RO, *assistant-helper*. Now, the assistant has two objects, the first one, *manager*, will be sleeping as long as the CH is alive. This agent implements methods needed to execute traditional jobs of the CH i.e. they perform heavy work, but these methods are needed only when the CH fails, so by deactivating this agent we save energy and reduce processing overhead. The other object, *assistant-helper*, will be *remote* and will be active till canceling the *assistant* role of the assistant. This RO will deal with the *CH-helper* to receive updates from the CH (Figure 2). Once an update occurs the CH invokes a *remote* method *update()* implemented at the *assistant-helper* to send updates to the assistant. If no updates occur during some time t_u, the CH will invoke a *remote* method *resetTime()*, implemented at the *assistant-helper*, to reset the waiting time t_w. If the *assistant-helper* does not receive any message from the *CH-helper* for some time t_w, it will activate the assistant *manager* MA to manage the cluster. Then the *assistant-helper* will check if the CH is reachable. If so it will kill the old CH objects, choose the next CH, send a copy of the *manager* MA to the new CH and destroy the assistant *manager* and itself.

Now, we have three different objects:

1- *manager:* this MA declares methods that execute traditional jobs of the CH. Once the CH chooses its assistant, the CH will send a copy of the *manager* MA to this assistant i.e. each of the CH and the assistant will host a copy of this agent. At the assistant, the *manager* object will be sleeping as long as the CH is alive. Besides the traditional jobs, the *manager* object declares new method *createHelper* that creates the helper RO according to the role of the node. i.e., at the CH, the *manager* will create *CH-helper* and at the assistant *assistant-helper*. Figure 3 shows the *manager* class diagram.

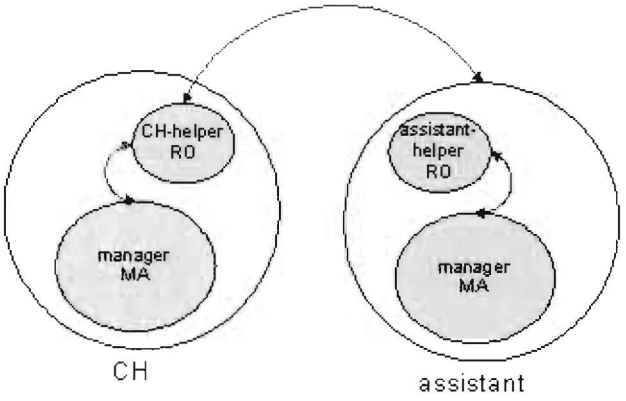

Fig. 2. Communication between the objects.

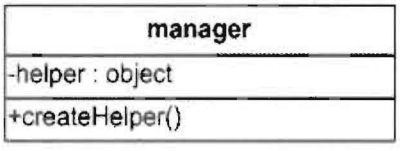

Fig. 3. Manager MA Class diagram.

2- *CH-helper:* the CH *manager* will create this RO which invokes methods executing the proposed jobs:

a. *chooseAssistant():* this method is implemented at the *CH-helper* which will invoke this method to choose the closest node as an assistant. Also, before sending data to its assistant, the *CH-helper* will check the availability of the assistant and if it is failed the *CH-helper* will invoke this method to choose a new assistant.

b. *update():* this *remote* method is implemented at the *assistant-helper*. The CH will be responsible of updating the assistant because the assistant will replace the CH when fails. Once the CH is updated, it invokes this *remote* method to update the assistant. If no updates occur during some time t_u the *CH-helper* will invoke

a *remote* method *resetTime()*, implemented at the *assistant-helper*, to reset the waiting time t_w. Figure 4 shows the *CH-helper* class diagram.

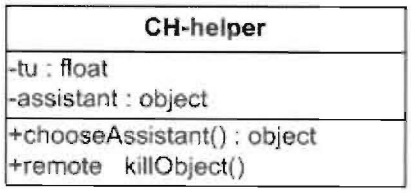

Fig. 4. CH-helper Class diagram.

3- *assistant-helper:* the assistant *manager* will create this RO which declares methods executing jobs of the assistant:

a *deactivate():* once the assistant *manager* creates the *assistant-helper*, the *assistant-helper* will deactivate the assistant *manager* object till the CH fails.

b. *activate():* once the *assistant-helper* discovers the failure of the CH, it will activate the assistant *manager* to manage the cluster.

c. *update():* when the *assistant-helper* receives the updates from the *CH-helper*, it stores these updates to pass them to the assistant *manager* when *assistant-helper* activates it.

d. *discover CH():* the *assistant-helper* expects a message from the *CH-helper* during t_w period. This message will contain updates or resetting t_w. When this message is received the assistant will reset t_w. If t_w expires, i.e., no messages received for t_w, the assistant will consider the CH to be failed. Then the *assistant-helper* will invoke the method *activate()* to activate the sleeping assistant *manager* MA, send updates to it and then declare the assistant as the temporal CH.

e. *chooseNextCH():* after the failure of the CH, the *assistant-helper* will choose the next CH depending on the considered criteria in the clustering algorithm. The method *chooseNextCH()* is declared to execute this job. Figure 5 shows the *assistant-helper* class diagram.

Heavy-loaded CH condition

Consider a case when the CH is heavy-loaded and does not send any updates to the assistant and it will respond slowly to requests. After t_w, the assistant will be the CH i.e. their will be two CHs in the cluster. We solve this problem by the method *kill Object()*.

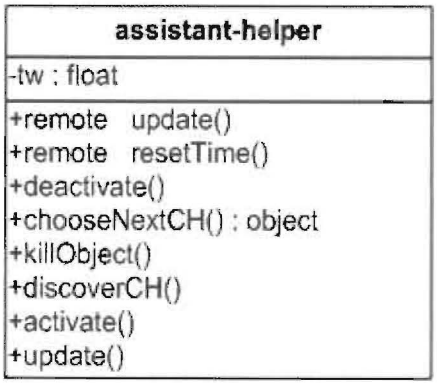

Fig. 5. Assistant-helper Class diagram.

Kill Object (): This method is to ensure that there is no more than one CH in the cluster. After choosing the next CH, the *assistant-helper* will call this method in two cases: *Remotely*: the *assistant-helper* will call this remote method to kill the assistant *manager* and so the *CH-helper* objects at the slow-responding CH., and locally: the *assistant-helper* will choose the Next CH and send it a copy of the *manager*. Then the *assistant-helper* will call *kill Object ()* locally to kill the *manager* object at the assistant and so the *assistant-helper*.

The sequence diagram in Figure 6 demonstrates the behavior and the messages passed between the CH and the assistant objects.

SIMULATION AND RESULTS

The JiST-SWANS simulator has been used to simulate the proposed solution. Our simulation compared the performance in a clustered ad hoc network with the assistant approach and without it (Barr, 2004; Barr, 2005).

Our evaluation is based on the simulation of 50 mobile nodes to test the undecided time of the nodes at discrete time. The radio transmission range is assumed to be 635m and the *two-ray ground propagation channel* is assumed with a data rate of 1 Mbps. The data traffic simulated is *constant bit rate* (CBR) traffic. 60% of nodes, CBR sources, generate ten 128-byte data packets every (20-25) second. *Random waypoint mobility model* is used in our experiments with node speed 2-10 m/s. With this approach, a node travels towards a randomly selected destination in the network. After the node arrives at the destination, it pauses for the predetermined period of time

and travels towards another randomly selected destination.

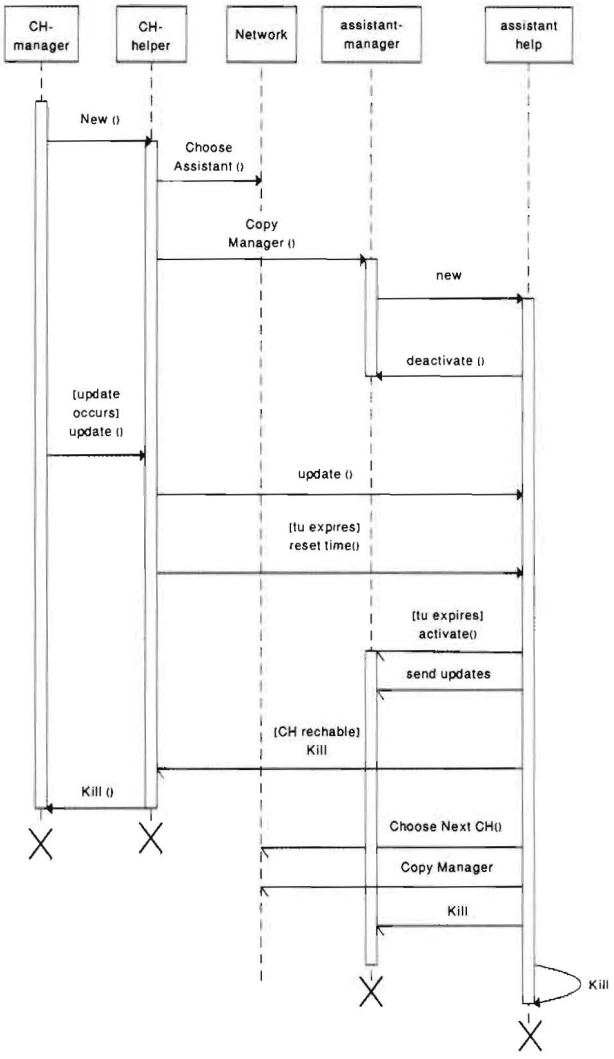

Fig. 6. Sequence Diagram.

Performance Metrics

A. Packet Delivery Ratio: Figure 7 illustrates the variance of Packet Delivery ratio in applying our solution and without it. Applying assistant approach increases Packet Delivery ratio significantly. Without applying the assistant approach many packets may be lost when loosing the CH; this will reduce the packet delivery ratio.

B. Average Undecided Time: Undecided time may be considered as an indicator about the stability of the nodes. As the undecided time increases, the stability of the MANET decreases because undecided nodes does not route any packets until its find a CH. As we see in the figure 8, the average undecided time of nodes in assistant approach minimized because as we discussed in the assistant approach, assistant cover the absence of CH robustly. In both cases, with and without assistant, undecided time is high because the network is not clustered yet. After that, with assistant approach, the undecided time decreases sharply and tends to be linear. This means that the nodes do not loose the CH because the assistant replaces it.

C. Average Undecided Time vs. Mobility: Figure 9 shows that undecided time increases with high motilities, but with assistant approach the undecided time will be reduced which indicates more stable network.

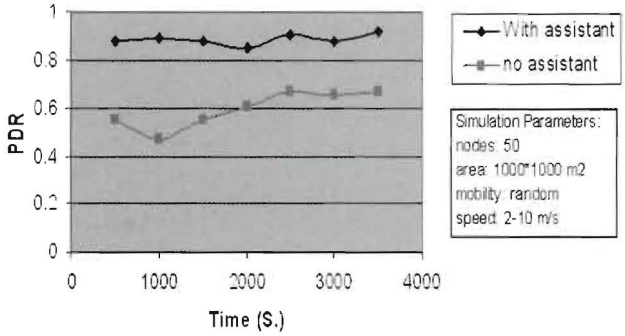

Fig. 7. Packet Delivery Ratio vs Time.

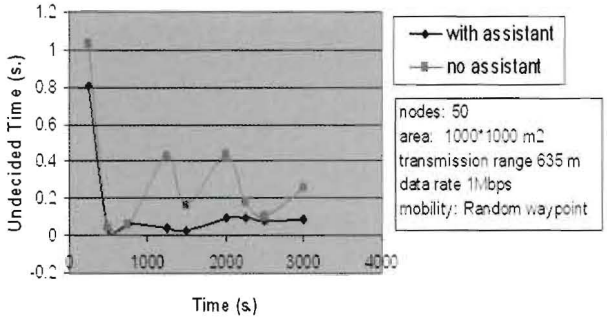

Fig. 8. Undecided Time vs. Simulation Time.

CONCLUSION

In this paper, we presented a solution to avoid the sudden failure of the CH and resulting faults in clustered MANETs. The solution involves three ideas, (a) the CH assigns a node as a vice-CH, or assistant, to temporarily replace the failed CH and so keep the management code and data. (b) A Mobile Object will be defined to carry out the traditional jobs of the CH to overcome the mobility of the nodes, and the assistant will have a copy of this mobile object. (c) A Remote Object defines new jobs of the CH and the assistant and its main job is to keep the communication between the CH and the assistant. Our solution achieves

fault-tolerance and improves the performance in MANETs.

Future research direction will be on developing a solution to distribute the CH jobs among some member nodes, and so we may relinquish the assistant.

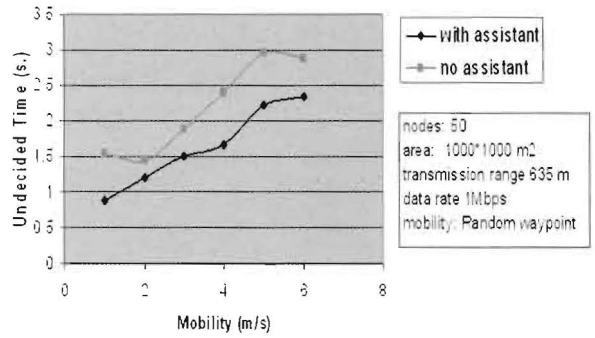

Fig. 9. Undecided Time vs. Mobility.

ACKNOWLEDGEMENT

The author would like to thank the research assistant Eng. Abdessalam Elhabbash for his assist in doing this work.

REFERENCES

Aderounmu, GA, Oyatokun, BO, and Adigun, MO (2006) Remote Method Invocation and Mobile Agent: A Comparative Analysis, *The Journal of Issues in Informing Science and Information Technology,* **3**: 1-11.

Ahmed, A, and **Far, BH** (2005) Topology Discovery for Network Fault Management using Mobile Agents in Ad-Hoc Networks. *In*: Institute of Electrical and Electronics Engineers *IEEE Canadian Conference on Electrical and Computer Engineering,* Institute of Electrical and Electronics Engineers, *IEEE,* pp 2041-2044.

Barr, R (2004) *JiST– Java in Simulation Time User Guide,* Department of Computer Science, Cornell University, Ithaca NY, USA.

Barr, R (2005) *JiST SWANS– Scalable Wireless Ad hoc Network Simulator User Guide,* Department of Computer Science, Cornell University, Ithaca, NY 14853, USA.

Basagni, S (1999) Distributed Clustering for Ad Hoc Networks. *In: Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN),* Perth, Western Australia, pp310-315.

Beiszczad, A, and **Pagurek, B** (1998) Mobile Agents for Network Management. *IEEE Communications Surveys*, **1** (1): 2-9.

Bhaumik, P, and **Banyopadhyay, S** (2005) Mobile Agent Based Message Communication in Large Ad hoc Networks through Co-operative Routing using Inter-Agent Negotiation at Rendezvous Points, *In: Lecture Notes in Computer Science,* Springer, Berline Heideilberg, Germany, pp554-559.

Chen, YP, Liestman, AL, and **Liu, J** (2004) Clustering Algorithms for Ad Hoc Wireless Networks., *In:* **Yi Pan** and **Yang Xiao (eds).** *Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing, vol. 2,* Nova Science Publisher, Hauppauge NY, USA, pp 145- 164.

Gavalas, D, Pantziou, G, Konstantopoulos, C, and **Mamalis, B** (2006) Lowest-ID with Adaptive ID Reassignment: A Novel Mobile Ad-Hoc Networks Clustering Algorithm. *In: International Symposium on Wireless Pervasive Computing, Institute of Electronics Engineers, IEEE,* Los Alamitos, USA, pp 5-.

Gupta, G, and **Younis, M** (2003) Fault-Tolerant Clustering of Wireless Sensor Networks. *In: Proceedings of International Conference on Wireless Communications and Networking, and Mobile Computing, WCNC, vol.3. Institute of Electrical and Electronics Engineers,* IEEE, Los Alamitos, USA, pp1579 – 1584.

Perkins, C (2001) *Ad Hoc Networking*, Addison-Wesley Professional, Boston, UK, pp1- 384.

Roy Choudhuri1, R, Bandyopadhyay, S, and **Paul, K** (2000) *Topology Discovery in Ad Hoc Wireless Networks Using Mobile Agents. In: Proceedings of the Second International Workshop on Mobile Agents for Telecommunication Applications*, Springer-Verlag, London, UK, pp1- 16.

Satoh, I (2003) Reusable Mobile Agents for Cluster Computing. *In:* Proceedings of the International Conference on Cluster Computing (CLUSTER'03), *Institute of Electronics Engineers,* IEEE, , Los Alamitos, USA. pp270-279.

Shang, Y, and **Cheng, S** (2005) A Stable Clustering Formation in Mobile Ad hoc Network. *In: Proceedings of International Conference on Wireless Communications and Networking and Mobile Computing, WCNC.,* Institute of Electrical and Electronics Engineers, IEEE, Los Alamitos, USA. pp714-718.

Yu, H, Martin, P, and **Assanein, H** (2005) Cluster-based Replication for Large-scale Mobile Ad-hoc Networks. *In: International Conference on Wireless Communications and Networking and Mobile Computing, WCNC.,* Institute of Electrical and Electronics Engineers, IEEE, Los Alamitos, USA. pp552-557.

Ref. No. (2453)
Rec. 13/ 09/ 2007
In- revised form 10/ 12/ 2007