New Approach for Selfish Nodes Detection in Mobile Ad hoc Networks

Djamel Djenouri
CERIST, Basic Software Laboratory
Algiers, Algeria
ddjenouri@mail.cerist.dz

Nadjib Badache
USTHB, Computer Science Department
Algiers, Algeria
badache@wissal.dz

Abstract

A mobile ad hoc network (MANET) is a temporary infrastructureless network, formed by a set of mobile hosts that dynamically establish their own network on the fly *without relying on any central administration. Mobile hosts used in MANET have to ensure the services that were ensured by the powerful fixed infrastructure in traditional networks, the packet forwarding is one of these services.*
The resource limitation of nodes used in MANET, particulary in energy supply, along with the multi-hop nature of this network may cause a new phenomena which does not exist in traditional networks. To save its energy a node may behave **selfishly** *and uses the forwarding service of other nodes without correctly forwarding packets for them. This deviation from the correct behavior represents a potential threat against the quality of service (QoS), as well as the service* **availability***, one of the most important security requirements. Some solutions have been recently proposed, but almost all these solutions rely on the watchdog [13] technique in their monitoring components, which suffers from many problems. In this paper we propose a new approach to mitigate some of these problems, and we assess its performance by simulation.*
Key words: *mobile ad hoc networks, security, selfishness, packet forwarding, power control, GloMoSim simulation.*

1. Introduction

In some MANETs applications, such as the battlefield or the rescue operations, all nodes have a common goal and their applications belong to a single authority, thus they are *cooperative by nature*. However, in many civilian applications, such as networks of cars and provision of communication facilities in remote areas, nodes typically do not belong to a single authority and they do not pursue a common goal. In such self-organized networks forwarding packets for other nodes is not in the direct interest of any one, so there is no good reason to trust nodes and assume that they always cooperate. Indeed, each node tries to save its resources, particularly its battery power which is a precious resource. Recent studies show that most of the nodes energy in MANETs is likely to be devoted to forward packets for other nodes. For instance, Buttyan and Hubaux simulation studies [5] show that; when the average number of hops from a source to a destination is around 5 then almost 80% of the transmission energy will be devoted to packet forwarding.

Therefore, to save energy, nodes may misbehave and tend to be *selfish*. A selfish node regarding the packet forwarding process is a node which takes advantage of the forwarding service and asks others to forward its own packets but does not actually participate in providing this service. Some solutions have been recently proposed, but almost all these solutions rely on the watchdog [13] technique which suffers from many problems that will be presented later. The purpose of this paper is to propose a novel solution to monitor and detect selfish nodes, that overcomes some of these problems.

The remainder of this paper is organized as follows: In the next section we present related work, and we briefly present the watchdog technique in section 3. Section 4 is devoted to the presentation and the analysis of our solution, followed by a simulation-based performance evaluation. Finally, section 6 concludes the paper and summarizes our perspectives.

0-7803-9469-0/05/$20.00

2. Related work

Nodes non-cooperation or selfishness is an emergent problem in MANETs that has recently received attention among researchers. In [9], we have surveyed the current proposed solutions and classified them into two main categories; reactive solutions that aim at detecting the misbehavior when it appears in the network, and preventive solutions which try to inhibit the misbehavior either by motivating nodes to cooperate or by taking measures to prevent packets from being dropped. To the best of our knowledge, Marti et al. are the first who dealt with the problem of selfishness on packet forwarding in MANETs. In [13] they define two techniques called *watchdog* and *pathrather*, the former is to identify misbehaving nodes whereas the latter helps the routing protocol to avoid these nodes. These techniques are used along with DSR [7] to build a misbehavior mitigating routing protocol. The watchdog is used by almost all the subsequent proposed reactive solutions. Nevertheless, it suffers from some problems, especially when using the power control technique, employed by some new power-aware routing protocols following the watchdog's proposal [11, 8, 10].
In [12] Yang et al. describe a unified network layer solution to protect both routing and data forwarding in the context of AODV. Michiardi and Molva [14] propose CORE, a generic reputation-based mechanism supposed to be easily integrated with any network function. Another reputation-based solution is proposed by Buchegger and Le-Boudec [2, 3]. They suggest a protocol called CONFIDANT, that relies on the DSR [7] used as benchmark in their GloMosim-based simulation [18].
Still, all these solutions rely on the watchdog technique for monitoring, and consequently inherit all the watchdog's problems.

Buttyan and Hubaux [4] propose a preventive economic-based approach which stimulates nodes to cooperate, this solution is modelized and analyzed in a further work [5]. They introduce what they call *virtual currency* or *nuglets*, a long with mechanisms for charging/rewarding service usage/provision. The main idea of this technique is that nodes which use a service must pay for it (in nuglets) to nodes that provide the service. Another preventive mechanism is the game theory approach, Vikram Srinivasan et al [16] propose a solution to stimulate cooperation based on this approach. In this solution, nodes are sometimes allowed to refuse the participation in the data forwarding process. We think this can presents a potential risk of the service unavailability. These preventive solutions just motivate nodes to cooperate, but do not aim at detecting the misbehaving nodes and do not inhibit nodes to behave selfishly.

In [15] Papadimitratos and J.Haas present the SMTP protocol, it prevents the selfishness effects (packets lost) by dispersing packets, and detects it by employing the end-to-end feedbacks. This kind of feedbacks allows detection of routes containing selfish nodes but fails to detect these nodes.

3. Overview of the watchdog

The watchdog method is a basic technique on which many further solutions rely, it aims to detect misbehaving nodes that do not forward packets by monitoring neighbors in the promiscuous mode.

When a node A transmits a packet to B to forward to C, A monitors B's forwarding by promiscuously listening to all the packets sent in its neighborhood. The watchdog is implemented at A by maintaining a buffer of the recently sent packets and comparing each overheard packet with the packets in the buffer to check whether there is a match. If so, the packet in the buffer is assumed forwarded, hence its is removed and forgotten by the watchdog. If a packet has remained in the buffer for longer than a certain timeout, the watchdog increments a *failure tally* of the node responsible for forwarding the packet, if this tally exceeds a certain threshold then the monitoring node (A) considers that B is misbehaving, and sends a message to the source notifying it of the misbehaving node. The watchdog technique supposes that each transmission can be overheard by all neighbors if no collision takes place. However, this assumption is not inevitably correct when the transmission power is not constant. For instance, the use of the power control technique [11], like in the recently proposed power-aware routing protocols [8, 11], renders some nodes unable to overhear transmissions even though they are within the sender's power range.

In this case we remark a serious problem when using the watchdog, namely the possibility of false detections (false positives). Assume that B uses controlled powers and the required power from B to C is less than the one needed to reach A from B, thereby the packets sent from B to C will not be received at A. Node A may accuse wrongly B as misbehaving even though it correctly forwards packets to C. This example shows how the watchdog fails when the power control technique is employed, the purpose of our proposal is mainly to overcome this problem.

Moreover, this technique cannot detect the misbehavior in many cases. In [13, 9] all these cases have been presented and analyzed, for space limitation we just cite them:

1. Partial dropping: node B can circumvent the watchdog by dropping packets at a lower rate than the watchdog's configured minimum misbehavior threshold

2. Receiver collision: after a collision at node C, B could skip retransmitting the packet without being detected by A

3. False misbehavior accusations: A node may falsely report other innocent nodes in its neighborhood as misbehaving to avoid getting packets to forward

4. Insufficient transmission power: B can control its transmission power to circumvent the watchdog. if A is closer to B than C, then B could attempt to save its energy by adjusting its transmission power and makes it strong enough to be overheard by the previous node (A) but less than the required power to reach the true recipient (C)

5. Cooperated misbehavior: B and C could collude to cause mischief. In this case, B forwards a packet to C but does not report to A when C drops the packet. C does the same thing when it is B's predecessor in some route.

4. The new approach

4.1. Solution overview

to mitigate the watchdog problem related to the power control usage we propose a new approach. Like the watchdog, we suggest that each node in the route monitors the forwarding of each packet it sends. To explain the concepts we suppose without lose of generality that A sends packets to B and monitors its forwarding to C. A source routing protocol is also assumed to be used.

We define a new kind of feedbacks we call *two-hop ACK*, it is an ACK that travels two hops. Node C acknowledges packets sent from A by sending this latter *via B* a special ACK. Node B could, however, escape from the monitoring without being detected by sending A a *falsified* two-hop ACK. Note that performing in this way is power economic for B, since sending a short packet like an ACK consumes too less energy than sending a data packet. To avoid this vulnerability we use an asymmetric cryptography based strategy as follows:
Node A generates a random number and encrypts it with C's public key (PK) then appends it in the packet's header as well as A's address. When C receives the packet it gets the number back, decrypts it using its secret key (SK), encrypts it using A's PK, and puts it in a two-hop ACK which is sent back to A via B. When A receives the ACK it decrypts the random number and checks if the number within the packet matches with the one it has generated, to validate B's forwarding regarding the appropriate packet. However, if B does not forward the packet A will not receive the two-hop ACK, and it will be able to detect this misbehavior after a time out. This strategy needs a security association between each pair of nodes to ensure that nodes share their PK with each other. This requires a key distribution mechanisms which is out of the scope of this paper, but a mechanism like [17] or [6] can be used. Another problem would take place when node C misbehave. If C does neither forward the packet nor send the two-hop ACK back to A, B could be supposed by A to not forward the packet even it actually does. To overcome this problem we propose that the sending of the two-hop ACKs is provided implicitly upon the reception of the packet at the *MAC layer*, and we assume that lower layers (the MAC and physical layers) are robust and tamper resistant. This can be ensured by the hardware and the operating system of each node, that is the operations of the lower layers cannot be modified by any node, and node C could not get rid of sending the two-hop ACK back to A upon the reception of the packet, thereby the B's monitoring is performed accurately. However, the upper layer including the network layer may be tampered by a selfish or a malicious, and falsified packets can be sent.

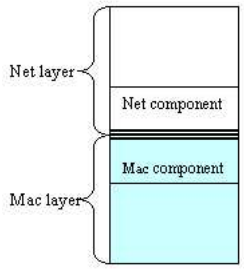

FIGURE 1. SOLUTION FRAMEWORK

Our solution is composed of two parts, the first one is located at the network layer and can be viewed as a sub layer at the bottom of this layer, whereas the second one is located over the MAC layer and is a sub layer at the top of this latter. Figure 1 illustrates this framework.

4.2. The protocol

Each node, except the destination, is monitored by its predecessor in the source route. To monitor its successor, each node i adds the random number it generates for each packet encrypted with the successor's successor PK along with i's address to each packet it receives from the routing protocol, and maintains the generated random number as well as the monitored node (i's successor) address in an entry within the Wait2HopsACK buffer. When a packet is received from another node X, i's MAC component automatically generates and sends X back a two-hop ACK after encrypting and decrypting again the random number as described previously. The Network layer component removes the appropriate entry upon the reception of the two-hop ACK, and as a timeout is associated to each entry, the lack of a two-hop ACK after the timeout results in the increasing of the rating regarding the appropriate forwarder node, thus a node is considered as selfish if its rating exceeds a given threshold. Like in the watchdog, we use this rating and we do not accuse directly the forwarder, this because lost of packets may be caused by channel conditions or nodes mobility and is not inevitably due to an intentional misbehave.
Algorithm 1 and 2 describes respectively the network and the MAC components. In these scripts we use the following notation
•R_{Key}: encrypting R with Key
•R^{key}: decrypting R with Key
•P_X: the public key of node X
•S_X: the secrete key of node X.
Note that we minimized operations and data structure of the MAC layer component to facilitate its implementation.

Algorithm 1 Network module of solution 1

When receive a packet D from the routing protocol to send to node X (X either the next hop or the destination and i is either the source or a forwarding node):
if (X ≠ D's destination) **then**
R = a generated random number
Y = X's successor in the source route
append (R_{P_Y}, i) to D's header
add(R,X) to the buffer Wait2HopsACK
end if
send D to X
When receive a packet D from the MAC protocol sent by X:
if X ≠ D's source **then**
remove the random number generated by X's predecessor from the header along with the corresponding node address
end if
send the packet to the network layer protocol
When receive a two-hop ACK packet TwoHopsACK from the MAC layer component
$R' = TwoHopsACK.Rand^{S_I}$
if $(R', TwoHopsACK.sender) \in Wait2HopsACK$ **then**
remove (R',TwoHopsACK.sender) from Wait2HopsACK
end if
When a timeout out of a Wait2HopsACK entry (R,X) is exceeded
increment the rating regarding node X
if X's rating > threshold **then**
consider X as a misbehavior
end if

Algorithm 2 MAC layer located component of solution 1

when receive a packet D sent by X from the MAC protocol
if $X \neq D'ssource$ **then**
Y = X's predecessor in the source route
Get the random number R generated by y
$R' = R^{S_I}$
$R'' = R'_{P_Y}$
construct a two-hop ACK packet TwoHopsACK
TwoHopsACK.Rand= R''
TwoHopsACK.sender=I
TwoHopsACK.dest=Y
send two-hop ACK to X
end if
pass the packet up to the network component
when receive a two-hop ACK packet TwoHopsACK
if TwoHopsACK.dest ≠ i **then**
TwoHopsACK.sender=i
forward TwoHopsACK to TwoHopsACK.dest
else
pass the packet up to the network layer component
end if

4.3. Discussions

In our solution we get rid of the promiscuous mode use employed by the watchdog, nodes are therefore not required to receive all packets sent in their neighborhood, which can reduce significantly the energy consumption. Instead, our solution relays on the new efficient technique we propose, namely the two-hop ACK. The monitoring node (A) validates the monitored node (B) forwarding when it receives an ACK from the successor of this latter (node C). This is independent of the power control usage, thereby our solution resolves the watchdog false detection problem related to the employment of this technique. Moreover, our solution resolves the problem 2 of the watchdog (section 3). When a collision appears at C, B should retransmit the packet, otherwise A will not validate its forward-

ing. This because B's forwarding will not be validated at A until C really receives the packet and send back the two-hop ACK, unlike the watchdog where the validation is only related to B's first transmission. The new solution also solves the problem 4 of the watchdog, we have seen that when A is closer to B than C and when A uses the watchdog to monitor B, this latter could save its energy and makes the transmission power strong enough to be overheard by A but less than the required one to reach C. This is avoided in our solution because B's forwarding validation at A is not just related to B's transmitting, but to C's reception. However, the other problems (1, 3, and 5) remaind unresolved.

5. Simulation results

To asses the proposed protocol performance we have driven a GloMoSim-based [18] simulation study, that we will present hereafter.
We have simulated a network of 50 nodes, located in an area of $1500 \times 1000m^2$, where they move following the random way-point [1] model with an average speed of 1m/s, for 900 seconds (the simulation time). To generate traffic, we used three CBR sessions between three pairs of remote nodes, each consists of continually sending a 512 bytes data packet each second. On each hop, each data packet is transmitted using a controlled power, according to the distance between the transmitter and the receiver. We will compare our technique to the standard promiscuous monitoring of the watchdog with regard to three metrics: false positive rate, end-to-end delay, and power consumption. We measured these metrics vs the selfish nodes rate, which represents the rate of selfishly vs nodes number. Note that selfish nodes misbehave selfishly during the whole simulation time and drop all data packets they are asked to relay. These nodes have been chosen in such a way to appear in routes used in our scenarios. Each point of the plots presented hereafter has been obtained by averaging three measurements with different seeds. Note that we implemented our protocol with DSR for this simulation, like the watchdog. However, it can be implemented with any source routing protocol.

5.1. False positive rate

This metric, we denote by FPR, will show how our protocol mitigates false detections of packet dropping due to the power control use. It is the average rate of false detections, giving by the following formula.

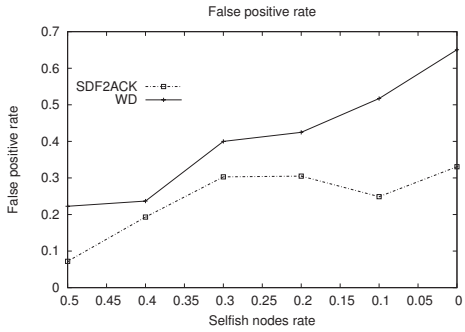

FIGURE 2. FALSE POSITIVE VS SELFISH NODES RATE

$$FPR = \sum_{i=0}^{k} \frac{fd_i/m_i}{k}$$

where:
fd_i: is the false detections of node i, i.e the number of packets node i falsely detect as dropped.We mean by false detection on a packet the fact that the detected monitored node is not selfish, and has not actually received and intentionally droped the packets.
m_i: the number of packets monitored by node i.
k: the number of nodes participating in the monitoring.
As illustrated in figure 2, our protocol (denoted by SDF2ACK) has low false detection rates compared with those of the watchdog (WD) which increase dramatically with the selfish rates decrease. The false detection in the mobile scenario is mainly due to packets lost caused by nodes mobility.
Note that in our scenarios, the true positive rate (rate of true dropping detection) has been always equal to one for both protocols.

5.2. End-to-end delay

End-to-end delay, or latency, is a big issue, especially for real time applications. All security protocols require communication and computation overhead, which might affect of the packet transfer latency. However, an efficient protocol should not affect a lot this transfer.
Hereafter, we will investigate whether our protocol influences the end-to-end delay. We define this metric as the average time separating the sending of a data packet from a source node and its arrival to the corresponding destination. Formally speaking:

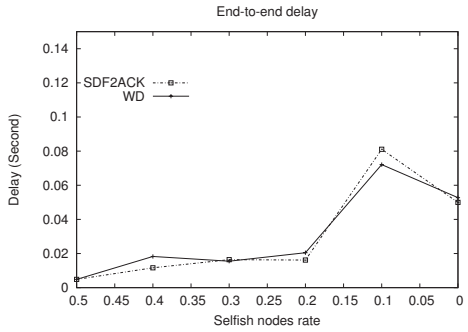

FIGURE 3. END TO END DELAY VS SELFISH NODES RATE

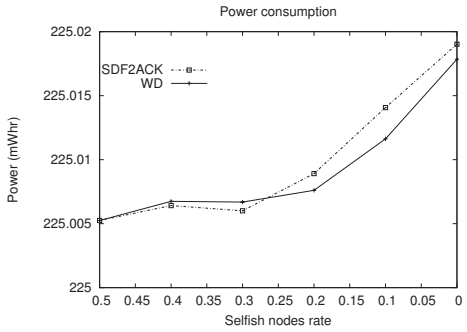

FIGURE 4. POWER CONSUMPTION VS SELFISH NODES RATE

$$delay = \sum_{i \in Rec} \frac{\sum_{j \in pr_i} \frac{delay_j}{nbpr_i}}{nbRec}$$

Rec: is the set of *destination* nodes that received data packets, nodes that did not receive any data packet are eliminated
nbRec: is the number of receiver nodes ($\| Rec \|$)
pr_i: is the set of packets received by node i as the final destination, packets that did not arrive to their destination are eliminated.
$nbpr_i$: is the received packets number ($\| pr_i \|$)
$delay_j$: is the transfer delay of packet j, such that:
$delay_j$ = packet j arrival time to its destination - packet j sending time by the source
As shown in figure 3, the delay of SDF2ACK is very close to that of the WD, they are almost identical. For both protocols we remark that the delay increases with the decreasing of selfish rate. We explain this by the fact that the diminution of selfish nodes rises the number of packets received at the final destination (counted in the metric computation), especial those transmitted on long paths and having important delays, that are more likely to be dropped before reaching their destination in scenarios with important selfish rates.

5.3. Power consumption

Because energy resources of devices used in MANETs are limited, energy consumption is an issue of high importance. And because it is the main reason that motivates nodes to behave selfishly, the selfish nodes detection protocol should not cause much power consumption when executed by the nodes. In the following we will investigate how much our overhead, particulary the two-hop ACKs, affect the total power consumption.
We define the average consumed power as:

$$average\_power = \sum_{i=1}^{n} \frac{PC_i}{n}$$

Such that PC_i is the power consumed by node i during the simulation, computed in GloMoSim using the NCR Wavelan radio model [18].
We can see in figure 4 that the difference in power consumption between the two protocols is minor in all situations.

6. Conclusion and future work

As we have seen, the watch dog technique, used by almost all the solutions currently proposed to detect nodes that misbehave on packets forwarding in MANETs, fails when employing the power control. In this paper, we have proposed a new approach that overcomes this problem. Unlike the end to end ACK, our approach allows to detect the misbehaving node. Moreover, it resolves the problems related to cases 2 and 4 of the watchdog (section 3). In the previous section we have assessed our protocol's performance by simulation. The results show that our protocol hugely decreases the false detection rate of packet dropping compared to the watchdog, while keeping the latency and the consumed energy too close to those of the watchdog.

Simulation results also show that there are always possibility of false detection. Consequently, one monitoring node cannot immediately accuse another as selfish when detecting that a packet has been dropped at

this latter. Instead, a threshold should be used like in the watchdog, and the monitored node will be considered selfish as soon as the number of packets dropped at this latter exceeds this threshold whose value should be well configured to overcome dropping caused by collisions and nodes mobility.

As perspective, we plan to improve the solution and decrease its overhead. We also plan in our further research to complete the proposal by: given a rigorous definition to the tolerance threshold, defining actions that have to be taken when a node is accused as a selfish, and proposing a mechanism allowing nodes to exchange their knowledge regarding nodes that behave selfishly.

References

[1] N. Badache, D. Djenouri, and A. Derhab. Mobility impact on mobile ad hoc networks. In *ACS/IEEE conference proceeding, Tunis, Tunisia*, July 2003.

[2] S. Buchegger and J.-Y. L. Boudec. Performance analysis of the confidant, protocol cooperation of nodes fairness in dynamic ad hoc networks. In *Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02), Lausanne, Switzerland*, pages 80–91, June 2002.

[3] S. Buchegger and J.-Y. Le-Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Second Workshop on the Economics of Peer-to-Peer Systems*, June 2004.

[4] L. Buttyan and J.-P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. *Technical report No. DSC/2001/001, Swiss Federal Institution of Technology, Lausanne, Switzerland*, January 2001.

[5] L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications, Vol 8, N 5*, October 2003.

[6] S. Capkun, L. Buttyan, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing, Vol.2, No.1*, pages 52–64, January 2003.

[7] B. David and A. David. Dynamic source routing in ad hoc wireless networks. *Mobile Computing, Chapter 5*, pages 153–181, 1996.

[8] D. Djenouri and N. Badache. New power-aware routing for mobile ad hoc networks. *The International Journal of Ad Hoc and Ubiquitous Computing (Inderscience)*, 1(2), 2005.

[9] D. Djenouri, L. Khalladi, and N. Badache. Security issues in mobile ad hoc and sensor networks. *IEEE Communications Surveys and Tutorials*, 7(4), 2005.

[10] D. Djenouri and N.Badache. Simulation performance evaluation of an energy efficient routing protocol for mobile ad hoc networks. In *IEEE International Conference on Pervasive Services (ICPS'04)*, American University of Beirut (AUB), Lebanon, July 2004.

[11] S. Doshi and T. Brown. Minimum energy routing schemes for a wireless ad hoc network. In *IEEE INFOCOM 2002*, 2002.

[12] X. M. H. Yang and S. Lu. Self-organized network layer security in mobile ad hoc networks. In *ACM MOBICOM Wireless Security Workshop (WiSe'02), Georgia, Atlanta, USA*, September 2002.

[13] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *ACM Mobile Computing and Networking, MOBICOM 2000*, pages 255–65, 2000.

[14] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Communication and Multimedia Security 2002 Conference, Portoroz, Slovenia*, September 26-27 2002.

[15] P. Papadimitratos and Z. J. Haas. Secure data transmission in mobile ad hoc networks. In *ACM MOBICOM Wireless Security Workshop (WiSe'03), San Diego, California, USA*, September 2003.

[16] V. Srinivasan, P. Nuggehalli, C. F.Chiasserini, and R. R.Rao. Cooperation in wireless ad hoc networks. In *IEEE INFOCOM'03, San Francisco, California, USA*, April 2003.

[17] S. Yi and R. Kravetso. Moca : Mobile certificate authority for wireless ad hoc networks. In *The second anunual PKI research workshop (PKI 03), Gaithersburg*, 2003.

[18] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for the parallel simulation of large-scale wireless networks. In *proceeding of the 12th Workshop on Parallel and distributed Simulation. PADS'98*, May 1998.