International Journal of Computer Theory and Engineering, Vol. 2, No. 1 February, 2010
1793-8201

Enchanced Self Umpiring System For Security Using Salvaging Route Reply

Ayyaswamy Kathirvel ,Member MISTE, IACSIT, Rengaramanujam Srinivasan, Member MISTE, FIE

***Abstract*—Mobile ad hoc networks are self-creating, self-administering and self-organizing entities. A self-motivated set of mobile wireless users dynamically exchange data among themselves in the absence of a predetermined infrastructure and controller. Malicious nodes adversely affect the performance of such networks. In an earlier paper we had suggested a self umpiring system for security as an enhancement to AODV protocol, in which mobile nodes in their role as umpire not only detect the functioning of malicious nodes, but also prevent them from further participation in the network activity. In this paper, we propose an enhancement to the self umpiring system, called USS-SRR, with the incorporation of salvaging routing reply (SRR). Exhaustive simulation studies using QualNet 4.5 bring out the strength of the method. We show that, in the presence of 40 % malicious nodes and with a mobility of 20 m/s, the increase in throughput is nearly 2 % and reductions in control overhead and end-to-end delay respectively are 1.7 % and 4 % as compared to basic self umpiring system.**

***Index Terms*— MANET, self-USS, USS-SRR, and malicious nodes**

I. Introduction

Mobile ad hoc networks (MANET) are self-creating, self-administering and self-organizing entities. Thus a set of self-motivated mobile wireless users is able to dynamically exchange data among themselves even in the absence of a predetermined infrastructure and controller. Each user of mobile ad hoc network also acts as a router allowing other users to communicate through their mobile communication device. The communication range of each device is limited; therefore, at any given time a user can exchange packets only with any one of the devices in its transmitting or receiving range.

Unlike the conventional cellular networks that rely on extensive infrastructure to support mobility, a MANET does not need expensive base stations and wired infrastructure. These features are important for potential use in a wide variety of disparate situations. Such situations include battlefield communications and disposable sensors, which are dropped from high altitudes and are dispersed on the ground for hazardous materials detection. Civilian applications include emergency situations such as responses to hurricane, tsunami, earthquake, and terrorism. Another interesting example is the case, where a set of mobile vehicles on the highway form an ad hoc network of their own in order to provide vehicular traffic management. Security provisioning in wireless ad hoc networks plays an integral part in determining the success of network centric warfare as envisioned for future military operations [8][9]. Thus, security is an important issue for these mission-critical applications [4-7].

This paper is based on the foundations of two systems already proposed self-USS[3] and SRR[1]. A brief look at them is in order.

Kathirvel and Srinivasan, have proposed a self umpiring system for security in mobile ad hoc network. In the self-umpiring system each node is issued with a token at the inception. The token consists of two fields: NodeID and status [3]. NodeID is assumed to be unique and deemed to be beyond manipulation; status is a single bit flag. Initially the status bit is preset to zero indicating a green flag. The token with green flag is a permit issued to each node, which confers it the freedom to participate in all network activities. Each node in order to participate in any network activity, say, Route Request RREQ, has to announce it's token. If it's status bit is "1" indicating "red flag" protocol does not allow the node to participate in any network activity. The working of the self-umpiring system is explained with reference to Fig. 1.

In the self-umpiring system all the nodes have dual roles – packet forwarding and umpiring. In the forward path during data forwarding, each node monitors the performance of immediate next node. That way, node A can tell correctly whether B is forwarding the packet sent by it, by promiscuously hearing B's transmissions. Similarly during reply process RREP, C can verify whether B is unicasting the route reply RREP and whether the hop count given by B is correct. Thus during forward path A is the umpire for B and C is the umpire for B during reverse path operations.

When a node is found to be misbehaving – say dropping data packets, the corresponding umpire immediately changes the status bit of guilty node to "1" indicating red flag.

The loss of route reply packets causes serious impairment of performance of routing protocol. This is because route reply packets are obtained after flooding the entire network with RREQs. Mekesh Singhal et al [1] have proposed and implemented the idea of salvaging route reply (SRR) for on

Manuscript received April 01, 2009. Manuscript Revised on July 22, 2009. Manuscript accepted August 05, 2009.

Ayyaswamy Kathirvel, Assistant Professor. He is now with the Department of Computer Science and Engineering, BS Abdur Rahman University, Chennai 600048, Tamilnadu, India
(E-mail: kathir@crescentcollege.org).

Rengaramanujam Srinivasan, Professor. He is now with the Department of Computer Science and Engineering, BS Abdur Rahman University, Chennai 600 048, Tamilnadu, India (e-mail: rsvasan@crescentcollege.org).

demand routing protocols. The basic idea is illustrate in Fig.2. Assume that, initially there exists no active path from source node S to destination node D. Node S is discovering a route to node D. Node D sends a RREP to node S, through intermediate nodes A, B, C and X. Node C cannot send the RREP to node B because B has moved away. Node C becomes the salvor, it saves the RREP message, and then it broadcasts a $RREQ_{SRR}$. Node V receives the $RREQ_{SRR}$ and finds a route to the source node S in its routing table, so V sends a $RREP_{SRR}$ to C. C receives the $RREP_{SRR}$ and successfully salvages the original RREP by sending it along the path discovered by SRR. It can use the new alternative route to send RREP packets to node S, through intermediate nodes A, U, V and C. Then the return path after SRR is D-X-C-V-U-A-S.

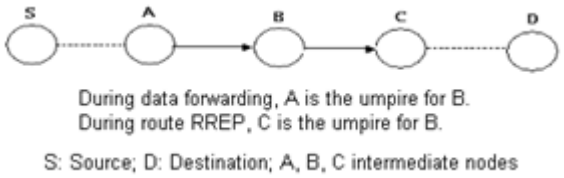

Fig.1 Self umpiring system model

Route maintenance deals with routing information at nodes, typically involving three possible operations: handling route errors, deleting stale route entries, and learning new routes from the traffic.

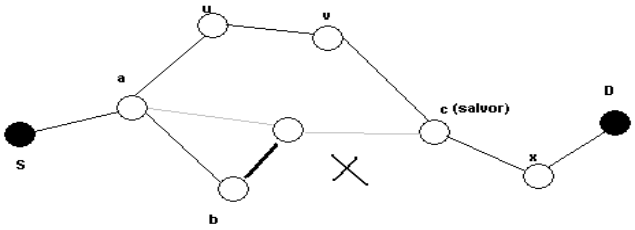

Fig.2 AODV-SRR mechanism: Link broken between B and C. Salvor node is C; intended RREP return path is D-X-C-B-A-S. Actual return path after SRR is D-X-C-V-U-A-S.

The present proposal is to enhance the performance of self-umpiring system with the incorporation of SRR. With this if the nodes behave maliciously during route reply phase,say,by giving a wrong hop count,such nodes will be flagged off from the network by the umpire and salvaging route reply packet commences immediately..

The rest of the paper is organized as follows: Section II describes the implementation of USS-SRR; Section III describes the simulation model using QualNet 4.5; Section IV gives an analysis of results; related work is reviewed in the Section V, while Section VI draws up conclusions.

II. Implementation of USS-SRR

The implementation of USS-SRR is based on three algorithms. Algorithm 1- USS route Request and Algorithm 3 USS Data packets are common to self umpiring system and USS _SRR. Algorithm 2 which involves RREP packets, is modified for SRR implementation. The modifications are given in bold letters.

Each node in order to participate in any network activity, says Route Request RREQ, has to announce it's token. as described in algorithm 1. If status bit is "1" indicating "red flag" protocol does not allow the node to participate in any network activity.

Algorithm 1: While sending an Umpire RREQ packet

1: **for** each umpire RREQ packet (P) sent do
2: **if** each node status is green flag then
3: broadcast RREQ
4: prevhop ← currenthop[node address]
5: repeat step 2 until it reaches the destination node
6: **else**
7: drop umpire RREQ packet (P) sent
8: **endif**
9: **endfor**

In the self-umpiring system all the nodes have dual roles – packet forwarding and umpiring. In the forward path during data forwarding, each node monitors the performance of immediate next node. That way, node A can tell correctly whether B is forwarding the packet sent by it, by promiscuously hearing B's transmissions. Similarly during reply process RREP as given in algorithm 2, C can verify whether B is unicasting the route reply RREP and whether the hop count given by B is correct. Thus during forward path A is the umpire for B and C is the umpire for B during reverse path operations.

Algorithm 2: While sending an Umpire RREP packet

1: **for** each umpire RREP packet (P) sent do
2: **if** node status is green flag then
3: unicast RREP to previous node
4: nexthop ← prevhop [node address]
5: repeat step 2 until it reaches the source node
6: **if** currenthopcount is equal to nexthopcount then
7: process this RREP as specified in the standard protocol
8: **else**
9: ***save current RREP message in the buffer***
10: it broadcast MERR packet to 1-hop or 2-hop node distance
11: nextnode status is marked as red flag
12: ***currentnode is the source node and the source node becomes a destination node, thus, start MRREQ procedure***
13: ***Process this MRREQ and MRREP as specified in the standard protocol***
14: ***it reaches the MRREP to the currentnode***
15: ***retrieve previous saved RREP message from the buffer***
16: send RREP message in newly identified path to the source node
17: **end if**
18: **endif**
19: **endfor**

When a node is found to be misbehaving – say dropping data packets, the corresponding umpire immediately sends a M-ERROR message to the source and the status bit of guilty node is set to "1" – red flag using M-Flag message as shown in algorithm 3. In order to correctly correlate the overheard

messages an additional field next_hop has been introduced in all routing messages as done in SCAN [2]. Though there are several kinds of misbehavior that could be captured by promiscuous hearing we are focusing only on two types of malicious actions: dropping packets and transmitting false hop count.

Algorithm 3: While sending an Umpire data packet

1: **for** each umpire DATA packet (P) sent do
2: **if** node status is green flag then
3: send a packet to the next forwarded node
4: it tampered with the payload or header of the currently sent packet
5 nexthop ← currentpacketheader
6: it keeps this header information until next packet is forwarded to the node
7: **else**
8: nextnode has dropped the packet, thus, the malicious node
9: prevnode is umpire node for next immediate forwarded node
10: **if** nexthop← currentpacketheader is not equal to prevhop ← currentpacketheader
11: it has marked as malicious node
12: it broadcast MERR packet to 1-hop or 2-hop node distance
13: Nextnode status is marked as red flag
14: Umpire node sent link error message to the source node
15: process this RERR message as specified in the standard protocol
16: **endif**
17: **endif**
18: **endfor**

III. Simulation Model

We use a simulation model based on QualNet 4.5 in our evaluation [10-11]. Our performance evaluations are based on the simulations of 100 wireless mobile nodes that form a wireless ad hoc network over a rectangular (1500 X 600 m) flat space. The MAC layer protocol used in the simulations was the Distributed Coordination Function (DCF) of IEEE 802.11[12]. The parameter settings are given in Table 1.

Before the simulation we randomly selected a certain fraction, ranging from 0 % to 40 % of the network population as malicious nodes. Though we had simulated two types of attacks – modifying the hop count and dropping packets in our earlier work, in order to focus attention on SRR, we are considering in the present paper only the attack of modification of hop count by the malicious nodes. Each flow did not change its source and destination for the lifetime of a simulation run. For all our studies we had kept the simulation time as 900s.

Packet delivery ratio is the ratio of the data packets successfully, delivered to the destinations to those generated by the CBR sources.

Average end-to-end delay, It is the average time taken for a packet to be transmitted across a network from source to destination. It includes transmission delay, propagation delay and processing delay.

Communication overhead is the total number of control packets sent by routing protocols in order to achieve its goal.

Simulation Time	900 seconds
Propagation model	Two-ray Ground Reflection
Transmission range	250 m
Bandwidth	2 Mbps
Movement model	Random way point
Maximum speed	0 – 20 m/s
Pause time	0 seconds
Traffic type	CBR (UDP)
Payload size	512 bytes
Number of flows	10 / 20

TABLE.1 Parameter Settings

IV. Analysis of Results

A. Packet Delivery Ratio

In the world of MANET, packet delivery ratio has been accepted as a standard measure of throughput. We present the packet delivery ratios of self-USS and USS-SRR, for malicious node percentages of 0, 10, 20, 30 and 40, with node mobility varying between 0 to 20 m/s. In general, in the absence of malicious nodes both routing protocols (self-USS and USS-SRR) have got good packet delivery ratios (Figs. 3 &. 4**).**

From the results presented in Fig.3 and Fig.4 the following conclusions can be drawn:

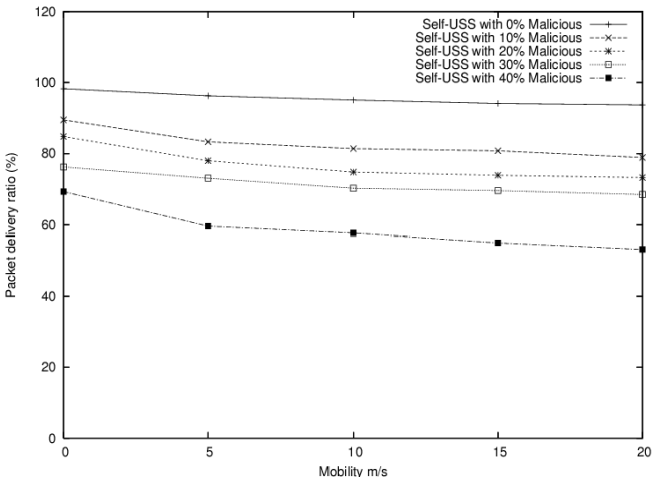

Fig.3 Packet delivery ratio versus mobility of self-USS

In the case of self-USS, with 10% malicious nodes, packet delivery ratio decreases from 84.48%, when the nodes are stationary to 78.89%, when the nodes are moving at 20 m/s. Corresponding figures for USS-SRR are 92.28 % and 82.03 %.

1) In general packet delivery ratio decreases as mobility and percentage of malicious nodes increase.

2) With self-USS, packet delivery ratio has a steep fall from 98.28 (0% malicious nodes, mobility=0m/s) to 53.18 (40% malicious node, mobility=20m/s). Corresponding figures for USS-SRR are 99.08%

and 54.35%. Thus throughput is increased nearly by 2 %.

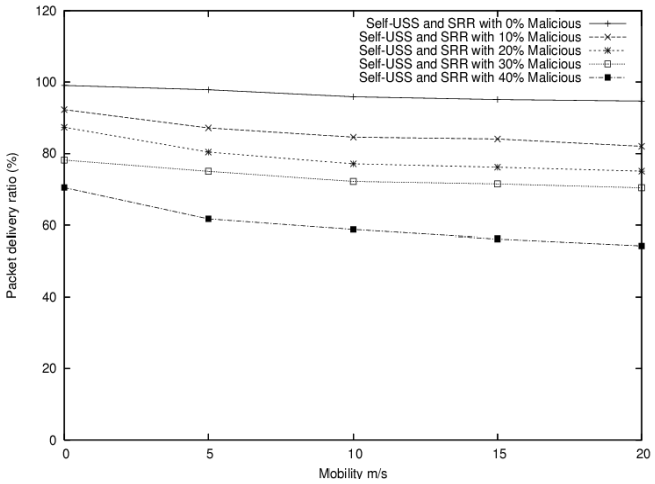

Fig.4 Packet delivery ratio versus mobility of USS-SRR

B. Communication overhead

Communication overhead can be evaluated based on the number of transmissions of control messages like RREQ, RREP, RERR in the case of plain AODV and in addition $RREQ_{SRR}$, $RREP_{SRR}$ in the AODV-SRR. RREQ are to be decimated to the entire network, where as RREP messages are unicasts.

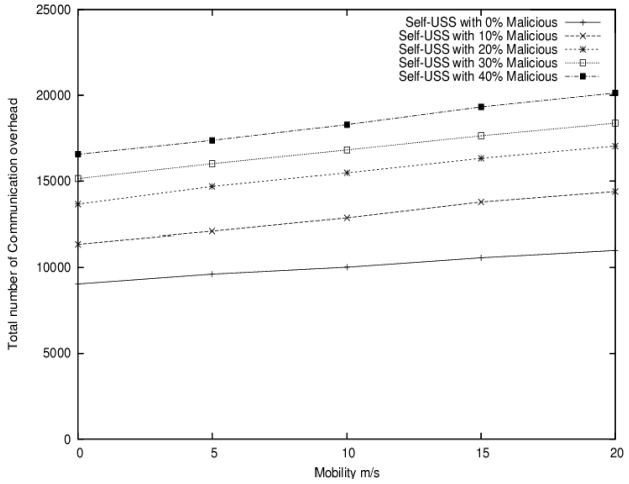

Fig.5 Communication overhead versus mobility of self-USS

From results of Self-USS (Fig. 5) and USS-SRR (Fig. 6.) following inferences can be drawn:

1) The communication overhead increases with increasing percentage of malicious nodes.
2) In the case of self-USS, with 10% malicious nodes, communication overhead increases from 11357, when the nodes are stationary to 14386, when the nodes are moving at 20 m/s as shown (Fig. 5), whereas USS-SRR (Fig. 6) with the same percentage of malicious nodes, communication overhead has reduced values of 11106 (0 m/s) and 14178 (20 m/s).
3) In USS-SRR with 40% malicious nodes, we find that the decrease in communication overhead is 1.7 % as compared with self-USS.

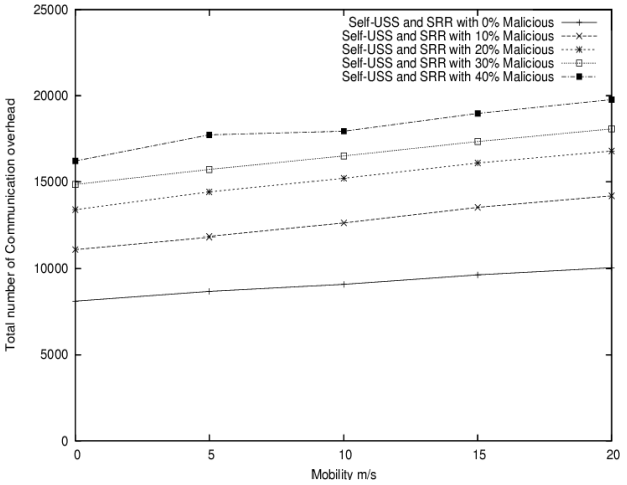

Fig.6 Total number of Communication overhead versus mobility of USS-SRR

C. End-to-end delay

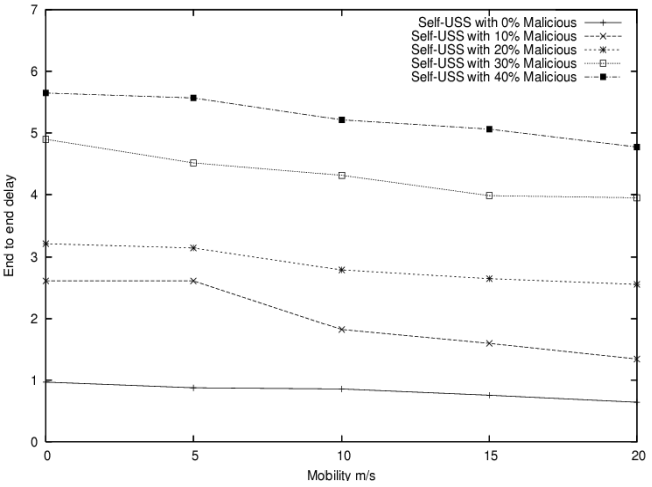

Fig.7 End-to-end delay verses mobility of self-USS

From results of Self-USS (Fig. 7) and USS-SRR (Fig. 8.) following inferences can be drawn:

1) In general end-to-end delay decreases as mobility and percentage of malicious node increases.
2) In the case of self-USS, with 10% malicious nodes, end to end delay decreases from 2.6, when the nodes are stationary to 1.3, when the nodes are moving at 20 m/s. Corresponding figures for AODV-SRR are 2.4 and 1.1 .
3) In USS-SRR with 40 % malicious nodes, we find that decrease in end-to-end delay is 4 % as compared with self-USS.

Table 2 presents a comparison of consolidated performances in the presence of 40 % malicious nodes moving at 20 m/s speed. It is seen that of the four protocols presented USS-SRR fairs the best. It is noted further that the proposed USS-SRR yields 93 % increased throughput, with a 6 % reduction in end-to-end delay and with an increase of 29.1 % in communication overhead, as compared to plain AODV.

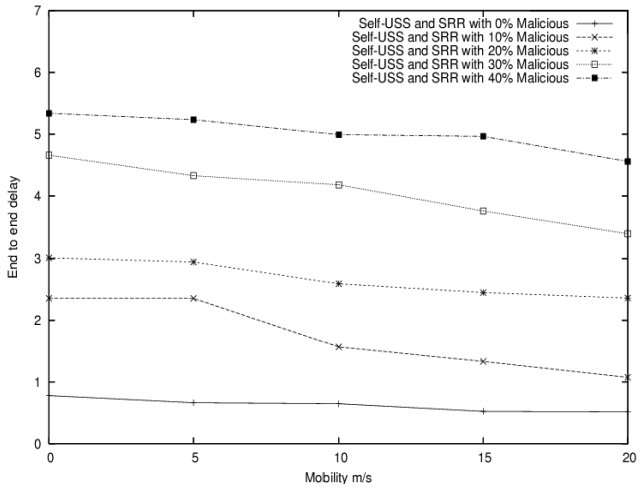

Fig.8 End-to-end delay versus mobility of USS-SRR

Performance Metrics	AODV	AODV with SRR	Self-USS	USS-SRR
Packet Delivery Ratio	28.1	30.5	53.2	54.4
End to end delay	4.9	3.9	4.8	4.6
Communication overhead	100	95	131	129

TABLE.2 Comparison of performances of various protocols (40% malicious nodes; mobility 20 m/s)

V. Related work

Many approaches have been proposed to improve the performance of reactive routing protocols. Some approaches have been beneficial to most of the reactive routing protocols [13-16]. The loss of route reply packets causes serious impairment of performance of AODV protocol [1]. This is because route reply packets are obtained after flooding the entire network with RREQs. Mekesh Singhal et al [2] have proposed and implemented the idea of salvaging route reply (SRR) for on demand routing protocols. In SCAN [2] two ideas are exploited to protect the mobile ad hoc network: (i) local collaboration where the neighboring nodes collectively monitor each other and (ii) information cross-validation by which each node monitors neighbors by cross-checking the overheard transmissions.

In SCAN, each node monitors the routing and packet-forwarding behavior of its neighbors and independently detects the existence of malicious nodes in its neighborhood. This is made possible because of wireless nature of the medium and all the involved nodes are within each other's transmission. In order to enable cross-checking they have modified AODV protocol and added a new field next_ hop in the routing messages so that each node can correlate the overheard packets accordingly.

VI. Conclusions

We have conducted simulation studies to evaluate the performance of USS-SRR in the presence of malicious nodes and have compared it with self-USS, AODV-SRR and plain AODV routing protocols. The results show that USS-SRR significantly improves the performance of self-USS in all metrics, packet delivery ratio, control overhead and end-to-end delay. In the presence of 40 % malicious node, USS-SRR yields a packet delivery ratio of 54.4, which is an improvement of 93 % over plain AODV protocol. Further, end to end delay is reduced by 6 %, with an increase in control overhead of 29 %, as compared to plain AODV. Our future work will focus on improving the umpire performance, by minimizing the innocent node booking.

Acknowledgment

We express our thanks to Dr. P. Kanniappan, the Vice Chancellor, Prof. V. M. Periasamy, the Register and Prof. K.M.Mehata, the Head, Department of CSE & Dean, School of Computer and Information Science B.S.A.Crescent Engineering College Chennai, Tamilnadu, India for the encouraging environment provided.

References

[1] Rendong Bai, and Mekesh Singhal. Salvaging Route Reply for On-Demand Routing Protocols in Mobile Ad-Hoc Networks, in proc. ACM MSWiM 2005.
[2] Hao Yang, James Shu, Xiaoqiao Meng and Songwu Lu, "SCAN: Self-Organized Network-Layer Security in Mobile ad hoc networks", IEEE Journals on selected areas in communications, vol. 24, No. 2, February 2006.
[3] Ayyaswamy kathirvel and Rengaramanujam Srinivasan, "Self _USS:A Self Umpiring System for Security in Mobile Ad hoc Network" , Journal of computer communication, Elsevier. (under review).
[4] Sergio Marti, T.J. Giuli, Kevin Lai and Mary Baker, "Mitigating routing misbehavior in mobile ad hoc networks", in proc. ACM MobiCom, 2000, pp- 255-265.
[5] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for MANET", in Proc. IEEE ICNP, 2001, pp. 251-260.
[6] J. Hubaux, L. Buttyan, and S. Capkun, "The quest for security in Mobile ad hoc networks", in Proc. ACM MobiHoc, 2001, pp. 146-155.
[7] S. Capkun, J.Hubaux, and L. Buttyan, "Mobility helps security in ad hoc networks", in Proc. ACM MobiCom, 2003, pp 46-56.
[8] C. Sivaram murthy and B.S. Manoj, " Ad hoc wireless networks architectures and protocols", Pearson Education, First edition, 2007.
[9] Jochen Schiller, "Mobile Communications", Pearson Education, Second edition, 2007.
[10] L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla, "Glomosim: A scalable network simulation environment. Technical Report 990027, 1999.
[11] Scalable Networks Technologies : QualNet simulator 4.5 http://www.scalable-networks.com/
[12] IEEE 802.11. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, August, 1999.
[13] C. Perkins, and E. Royer, "Ad hoc on-demand distance vector routing", in Proc. IEEE WMCSA, 1999, pp. 90-100.
[14] C. Perkins, E. Royer and S. Das, "Ad hoc on demand distance vector (AODV) routing", Internet Draft, draft-ietf-manet-aodv-10.txt, 2002.
[15] S. Das, C. Perkins, and E. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks", in Proc. IEE INFOCOM, 2003, pp. 3-12.
[16] C.E.Perkins, E.M. Belding-Royer, and I.D.Chakeres. Ad hoc on demand distance vector (aodv) routing. IETF Internet draft, oct. 2003.

Author Biographies:

Kathirvel - born in 1976 in Erode, Tamilnadu, India, received his B.E. degree from the University of Madras, Chennai, in 1998 and M.E. degree from the same University in 2002. He is currently with B.S.A. Crescent Engineering College in the Department of computer science and Engineering and pursing Ph.D. degree with the Anna University, Chennai, India. He is a member of the ISTE and IACSIT. Research interests are protocol development for wireless ad hoc networks, security in ad hoc networks.

Rengaramanujam Srinivasan -- born in 1940 in Alwartirunagari, Tamilnadu, India, received B.E. degree from the University of Madras, Chennai, India in 1962, M.E. degree from the Indian Institute of Science, Bangalore, India in 1964 and Ph.D. degree from the Indian Institute of Technology, Kharagpur, India in 1971. He is a member of the ISTE and a Fellow of Institution of Engineers, India. He has over 40 years of experience in teaching and research. He is presently working as a Professor of Computer Science and Engineering at BSA Crescent Engineering College, Chennai, India and is supervising doctoral projects in the areas of data mining, wireless networks, Grid Computing, Information Retrieval and Software Engineering.