TWOACK: Preventing Selfishness in Mobile Ad Hoc Networks

Kashyap Balakrishnan
Security Services Group
Deloitte & Touche LLP
McLean, VA 22102
kbalakrishnan@deloitte.com

Jing Deng
Department of CS
University of New Orleans
New Orleans, LA 70148
jing@cs.uno.edu

Pramod K. Varshney
Department of EECS
Syracuse University
Syracuse, NY 13244
varshney@ecs.syr.edu

***Abstract*— Mobile Ad hoc Networks (MANETs) operate on the basic underlying assumption that all participating nodes fully collaborate in self-organizing functions. However, performing network functions consumes energy and other resources. Therefore, some network nodes may decide against cooperating with others. Providing these selfish nodes, also termed misbehaving nodes, with an incentive to cooperate has been an active research area recently. In this paper, we propose two network-layer acknowledgment-based schemes, termed the TWOACK and the S-TWOACK schemes, which can be simply added-on to any source routing protocol. The TWOACK scheme detects such misbehaving nodes, and then seeks to alleviate the problem by notifying the routing protocol to avoid them in future routes. Details of the two schemes and our evaluation results based on simulations are presented in this paper. We have found that, in a network where up to 40% of the nodes may be misbehaving, the TWOACK scheme results in 20% improvement in packet delivery ratio, with a reasonable additional routing overhead.**

I. Introduction

Mobile Ad Hoc Network (MANET) can be described as an autonomous collection of mobile nodes (users) that communicate over relatively low capacity wireless links, without a centralized infrastructure. In these networks, nodal mobility and the wireless communication links may lead to dynamically changing and highly unpredictable topologies. All network functions such as routing, multi-hop packet delivery, and mobility management have to be performed by the member nodes themselves, either individually or collectively. So, network performance becomes highly dependent on collaboration of all member nodes. MANETs find applications in diverse fields ranging from low-power military wireless sensor networks to large-scale civilian applications, and emergency search/rescue operations.

There are two types of MANETs: open and closed [12]. An open MANET comprises of different users, having different goals, sharing their resources to achieve global connectivity, as in civilian applications. This is different from closed MANETs where the nodes are all controlled by a common authority, have the same goals, and work toward the benefit of the group as a whole. Open environment of a MANET may lead to misbehaving nodes. *Misbehaving nodes* come into existence in a network due to several reasons: (a) Mobile hosts lack adequate physical protection (due to the open communication medium), making them prone to be captured and compromised; (b) Usually mobile hosts are resource-constrained computing devices. Performing network functions consumes significant energy of participating nodes, as communication is relatively costly. Selfish nodes are unwilling to spend their precious resources for operations that do not directly benefit them. MANETs lack a centralized monitoring and management point, making it a challenging task to detect such misbehaving nodes effectively.

Non-cooperative actions of misbehavior are usually termed as *selfishness*, which is notably different from malicious behavior. Selfish nodes use the network for their own communication, but simply refuse to cooperate in forwarding packets for other nodes in order to save battery power. A selfish node would thus utilize the benefits provided by the resources of other nodes, but will not make available its own resources to help others. They have no intention of damaging the network. Malicious nodes injected by adversaries, on the other hand, will actively spend battery power to cause harm to the entire network. Providing nodes with an incentive to cooperate (by either rewarding them for active cooperation or punishing them for a lack of such cooperation) becomes an interesting research issue.

In [10], three types of selfish nodes related to routing such as Dynamic Source Routing (DSR) [7] are defined:

- **Selfish Nodes Type 1 (SN1)** – These nodes participate in the DSR Route Discovery and Route Maintenance phases, but refuse to forward data packets (which are usually much larger than the routing control packets);
- **Selfish Nodes Type 2 (SN2)** – These nodes participate in neither the Route Discovery phase, nor forwarding data packets. They only use their energy for transmissions of their own packets;
- **Selfish Nodes Type 3 (SN3)** – These nodes behave (or misbehave) differently based on their energy levels. When the energy lies between full energy E and a threshold T_1, the node behaves properly. For an energy level between T_1 and another lower threshold T_2, it behaves like a node of type SN1. Finally, for an energy level lower than T_2, it behaves like a node of type SN2. The relationship between T_1, T_2, and E is $T_2 < T_1 < E$.

The existence of the SN2 type nodes is simply ignored by

0-7803-8966-2/05/$20.00 (C) 2005 IEEE

the routing protocol. Thus, these nodes do not pose a significant threat to the normal operation of the routing protocol, even though they may degrade network connectivity. The SN1 and SN3 categories of nodes, on the other hand, are more dangerous to routing protocols. These nodes support the flow of route discovery traffic but interrupt the data flow, causing the routing protocol to restart the route-discovery process or to select an alternative route if one is available. The newly-selected routes may still include some of these SN1 type nodes, and hence the new route will also fail. This process will continue until the source of traffic concludes that data cannot be transferred. In this work, we focus only on the detection and mitigation of SN1 type misbehavior. SN3 type nodes will be detected when they behave similar to the SN1 type nodes.

In order to detect misbehaving nodes, we propose a network-layer scheme called TWOACK, which can be implemented as a simple add-on to any source routing protocol such as DSR [7] . When a node forwards a packet, the nodes routing agent verifies that the packet is received successfully by the node that is two hops away on the source route. This is done through the use of a special type of acknowledgment packets, termed TWOACK packets. TWOACK packets have a very similar functionality as the ACK packets on the Medium Access Control (MAC) layer or the TCP layer. A node acknowledges the receipt of a data packet by sending back a two-hop TWOACK packet along the active source route. If the sender/forwarder of a data packet does not receive a TWOACK packet corresponding to a particular data packet that was sent out, the next-hop's forwarding link is claimed to be misbehaving and the forwarding route broken. Based on this claim, the routing protocol avoids the accused link in all future routes, resulting in an improved overall throughput performance for the network. The S-TWOACK (Selective-TWOACK) scheme is a derivative of the basic TWOACK scheme, aimed at reducing the routing overhead caused by excessive number of TWOACK packets. We discuss our schemes in the framework of the DSR protocol as an example of source routing schemes. The operation of the TWOACK schemes when used with other source routing schemes is similar. Technical details of the DSR protocol, such as Route Discovery and Route Maintenance, and its optimizations, can be found in [7].

The rest of the paper is organized as follows: In Section II, we summarize the various approaches that have been proposed and studied in the technical literature to mitigate routing misbehavior. We then describe the details of the TWOACK and the S-TWOACK schemes and their technical merits in Section III. Section IV presents our performance evaluation setup and preliminary simulation results. We conclude our work and discuss our plan for future work in Section V.

II. Related Work

Various techniques have been proposed to prevent selfishness in MANETs. As described in [15], these schemes can be broadly classified into *reputation-based* schemes [2], [9] and *credit-based* schemes [3], [5], [6], the basic idea being to provide incentives to nodes to faithfully perform networking functions. In a reputation-based approach, nodes (either individually or collectively) detect, and then declare another node to be misbehaving. This declaration is then propagated throughout the network, leading to the misbehaving node being avoided in all future routes. A credit-based approach, on the other hand, uses the concept of virtual currency. Nodes pay virtual money for services (networking resources) that they get from other nodes, and similarly, get paid for providing services to other nodes. Since our TWOACK scheme is reputation-based, we only describe previous work of that category in this section.

In [9], Marti et al. proposed a reputation-based scheme. Two modules called watchdog and pathrater are implemented at each node, to detect and mitigate, respectively, routing misbehaviors in MANETs. Nodes operate in a promiscuous mode wherein, the watchdog module overhears the medium to check whether the next-hop node faithfully forwards the packet or not. At the same time, it maintains a buffer of recently sent packets. A data packet is cleared from the buffer when the watchdog overhears the same packet being forwarded by the next hop node over the medium. If a data packet remains in the buffer too long, the watchdog module accuses the next-hop neighbor to be misbehaving. Thus, the watchdog enables misbehavior detection at the forwarding level as well as the link level. Based on watchdog's accusations, the pathrater rates every path in its cache and subsequently chooses the path that best avoids misbehaving nodes. However, the watchdog technique may fail to detect misbehavior in the presence of ambiguous collisions, receiver collisions, limited transmission power, false misbehavior and partial dropping [9].

The CONFIDANT protocol proposed by Buchegger et al. in [2] is another example of a reputation-based scheme. The protocol is based on selective altruism and utilitarianism, thus making misbehavior unattractive. CONFIDANT consists of four important components - the Monitor, the Reputation System, the Path Manager, and the Trust Manager. They perform the vital functions of neighborhood watching, node rating, path rating, and sending and receiving alarm messages, respectively. Each node continuously monitors the behavior of its first-hop neighbors. If a suspicious event is detected, details of the event are passed to the Reputation System. Depending on how significant and how frequent the event is, the Reputation System modifies the rating of the suspected node. Once the rating of a node becomes intolerable, control is passed to the Path Manager, which accordingly controls the route cache. Warning messages are propagated to other nodes in the form of an *Alarm* message sent out by the Trust Manager. Of course, trust relationships and routing decisions depend on experienced, observed or reported behavior of other nodes, i.e. a node would obviously trust a first hand experience of misbehavior much more than if the misbehavior were reported by a third party.

In [12], Miranda and Rodrigues adopted a hybrid of reputation-based and credit-based schemes. Each node X maintains a data structure $Status_X[Y]$ about every other

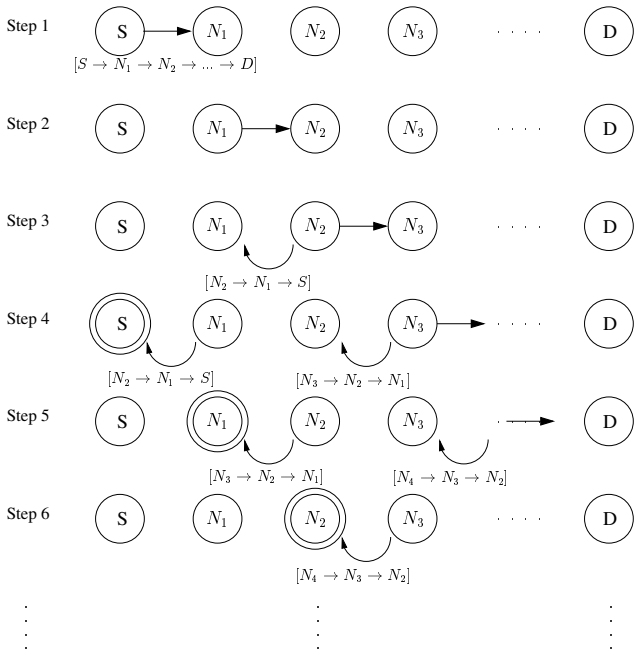

Fig. 1. The TWOACK Scheme

node Y, which is an indication of what impression node X has about node Y. Along with a credit counter, node X also maintains lists of nodes that Y will and will not provide service to. Every node periodically broadcasts relevant information in the form of a self-state message. Other nodes update their own lists based on the information contained in these self-state messages.

III. Scheme Descriptions

In this section, we describe our reputation-based schemes (TWOACK and S-TWOACK) that detect and mitigate the effects of node selfishness. The schemes aim to overcome some of the most prominent deficiencies of other reputation-based schemes.

A. The TWOACK Scheme

The TWOACK scheme can be implemented on top of any source routing protocol such as DSR. This follows from the fact that a TWOACK packet derives its route from the source route established for the corresponding data packet. The TWOACK scheme uses a special type of acknowledgment packets called TWOACK packets, which are assigned a fixed route of two hops (or three nodes) in the direction opposite to that of data packets.

Figure 1 illustrates the operational details of the TWOACK scheme. Suppose that the process of Route Discovery has already yielded a source route $[S \rightarrow N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow \cdots \rightarrow D]$ from a source node S to destination node D. For instance, when N_1 forwards a data packet to N_2, to be forwarded on to N_3, N_1 has no way of knowing if the packet reached N_3 successfully or not. Listening on the medium, as suggested in [9], would only tell N_1 whether N_2 is sending

N_2	N_3	C_{MIS}	$LIST$
Next Hop Receiver	Second Hop Receiver	Misbehavior Counter	List of Data Packet IDs Awaiting TWOACK

Fig. 2. Data Structure maintained for misbehavior detection

out the packet or not.[1] However, the reception status at N_3 is unclear to node N_1. The possibility of collisions at both N_1 and N_3 makes the overhearing technique vulnerable to medium access problems and false detections [9].

The TWOACK scheme is designed to solve these problems: when N_3 receives a data packet, it sends out a TWOACK packet over two hops back to N_1, carrying the packet ID of the corresponding received data packet. The route $[N_3 \rightarrow N_2 \rightarrow N_1]$ for the TWOACK packet is extracted from the source route of the original data packet. The aim of the TWOACK packet is to notify N_1 that the data packet has successfully reached a node that is two-hop away, namely N_3. Such a procedure will be carried out by every set of three consecutive nodes, termed *triplet*, along the source route.

Note that the ACK packets at the TCP layer have a similar effect as our TWOACK packets do. The main differences are the following: First, ACK packets in TCP are used for the purpose of flow-control and reliable end-to-end communication, while selfishness is more a problem that should be solved by the underlying IP layer. In the absence of a lower layer acknowledgment scheme, the source and other intermediate nodes have no way of finding out which of the downstream nodes is misbehaving. It will be inefficient to conclude that the entire route is misbehaving when indeed there is only one misbehaving node. To correctly detect and isolate such a misbehaving node, additional techniques such as the TWOACK scheme need to be employed. Second, ACK packets in TCP have to travel all the way from the final destination back to the source. Therefore, depending on the length of the path used for data packets, it is likely that ACK packets will arrive after significant delays. In contrast, TWOACK packets travel exactly two hops, making the timeout period shorter and more predictable.

To detect misbehavior, the sender or router of a data packet maintains a list of data packet IDs that have yet to receive a TWOACK acknowledgment packet from a node two hops away. Each node maintains a unique list for each forwarding link that it is using. Each item on the list has the following data members (cf. Fig. 2):

- N_2 and N_3: the receivers of the next two hops after this node, along the source route being used;
- C_{MIS}: counter for number of instances of misbehavior by forwarding link $N_2 \rightarrow N_3$;
- $LIST$: list of data packet IDs that are awaiting the TWOACK packets.

When a node, say, N_1, sends or forwards a data packet

[1]Due to potential packet collisions, N_1 should not conclude that N_2 has not sent out the data packet even if it has not overheard the transmission.

along a particular route, say, $N_1 \rightarrow N_2 \rightarrow N_3$, it adds the ID of the packet to $LIST$ on its list corresponding to $N_2 \rightarrow N_3$. When it receives a TWOACK packet, it checks for the $N_2 \rightarrow N_3$ combination, and then removes the packet ID from the corresponding $LIST$. If a data packet ID stays on $LIST$ longer than a certain period of time, termed *timeout*, misbehavior of link $N_2 \rightarrow N_3$ is suspected. Every time misbehavior is suspected, a non-negative misbehavior counter C_{MIS} is increased by one. When C_{MIS} exceeds a certain level, termed *thresh*, a node declares the corresponding link, $N_2 \rightarrow N_3$, misbehaving and sends out an RERR packet informing the source about the same.[2] Every node receiving or overhearing such an RERR packet should identify link $N_2 \rightarrow N_3$ as misbehaving. Every node maintains a list of misbehaving links that it has learned. Such links will not be chosen when it selects routes for data transmission later on.[3]

It might be unclear how the TWOACK scheme distinguishes *genuine route failures* from misbehaving nodes (links). Indeed, genuine route failures may take place due to mobility or excessive traffic in the vicinity of a forwarding node, e.g., N_2. When such failures appear, N_2 will voluntarily send an RERR packet to notify the source, as described in the routing protocol. Such an RERR packet is different from the RERR packet sent out by N_1 reporting a misbehaving link $N_2 \rightarrow N_3$.

The values assigned to *thresh* and *timeout* play an important role in determining the effectiveness of the TWOACK scheme. These parameters should be large enough so that intermittent failures or excessive transmission delays (due to collisions) of TWOACK packets are not interpreted as misbehavior. On the other hand, they should not be so large that a significant number of data packets are lost before a misbehaving node (link) is detected. Our studies on this are presented in Section IV.

B. The S-TWOACK Scheme

The TWOACK scheme described above gives rise to two hops of TWOACK packets for every hop of data packet being forwarded. Considering that each TWOACK packet is a unique entity and has to contend for the medium just like any other packet, the TWOACK packets may contribute to the traffic congestion on the routing path. Therefore, we further propose the S-TWOACK (Selective-TWOACK) scheme, a derivative of the TWOACK scheme, to reduce this extra traffic due to TWOACK packets. In the S-TWOACK scheme, instead of sending back a TWOACK packet every time when a data packet is received, a node waits until a certain number of data packets (through the same triplet) arrive. The node then sends back one TWOACK packet acknowledging multiple data packets that have been received so far.

[2]When link $N_2 \rightarrow N_3$ is misbehaving, it is because either N_2 refuses to forward the data packets or N_3 refuses to send back the TWOACK packets. It is difficult to identify which of these two nodes misbehaves. Since the link $N_2 \rightarrow N_3$ is unusable, we mark it as misbehaving.

[3]Ideally, bad routes will be avoided completely by storing both the node IDs and the link itself. However, such an approach blacklists a significant number of well-behaved nodes, consequently losing well-behaved routes in the network.

The S-TWOACK scheme has three parameters: *timeout*, *timeout_Last_Sent* and *maximum_IDs_Carried*. While *timeout* has the same usage as that in the TWOACK scheme, the use of the other two parameters can be explained as below: when the number of data packets received at N_3 reaches *maximum_IDs_Carried* or the duration since sending the TWOACK packet last time is larger than *timeout_Last_Sent*, a TWOACK packet will be sent. Note that, although the S-TWOACK scheme is expected to provide a significant reduction of routing overhead, it comes with a cost: the problem of false-alarms due to genuine TWOACK packets lost is more noticeable.

C. The TWOACK Scheme Revisited

The TWOACK technique has the following salient features (we use an example triplet of $N_1 \rightarrow N_2 \rightarrow N_3$ in our discussions):

Ambiguous Collisions: The TWOACK scheme will not be affected by ambiguous collisions at the sender or the receiver [9]. The reason is that a specific acknowledgment packet, the TWOACK packet, will be sent back from N_3 to the sender, N_1. The underlying MAC layer ensures the reliable transmission of such TWOACK packets.

Limited Transmission Power: Similarly, a selfish node, e.g., N_2, trying to use limited transmission power [9] to mislead the sender will be detected as well.

Missed Detection: A missed detection is the event of failure to detect a misbehaving node (link). Unless collusions between two neighboring misbehaving nodes, e.g., N_2 and N_3, exist, there will not be any missed detections in the TWOACK scheme.

Refusal of Sending TWOACK Packets: In the TWOACK scheme, a misbehaving N_3 may refuse to send TWOACK packets even if it receives the forwarded data packets successfully. While the TWOACK scheme cannot distinguish a misbehaving N_2 from a misbehaving N_3, such a scenario justifies our philosophy of marking the link $N_2 \rightarrow N_3$ as misbehaving.

Fabrication of TWOACK Packets: A misbehaving N_2 may fabricate a TWOACK packet and claim that it was generated by N_3. The current version of the TWOACK scheme cannot detect such fabricated TWOACK packets. However, authentication mechanisms similar to those used in SEAD [4] or SPINS [14] may be employed to detect such fabrications.

Re-Introduction of Misbehaving Nodes: Misbehaving nodes may be re-introduced to the network after a certain period of time. This is especially important for those networks that are supposed to run for a long period of time and some false alarms are possible. Some misbehaving nodes may be recharged and become well-behaved later on as well.

IV. Performance Evaluation

In this section, we present evaluation of the TWOACK and the S-TWOACK schemes through network simulations. We use a version of Network Simulator (*ns-2*) [16] that incorporates the CMU Monarch project's wireless extensions. We have modified the DSR module in *ns-2* to simulate selfish nodes of type SN1 described in Section I.

A. Simulation Methodology

Our simulations were carried out with 40 mobile nodes moving in a 670×670 m^2 flat area. Each node's transmission range is 250 m. The IEEE 802.11 MAC layer was used. A random waypoint mobility model was assumed with maximum speed of 20 m/sec and pause time of 0 second (high mobility).

For the communication pattern, we implement CBR transfers between pairs of nodes. The source and destination for each CBR pair are randomly chosen such that there is no limit on the number of sources or destinations that a node can host. The TWOACK scheme is analyzed under varying traffic conditions by running simulations for networks with 10 (low traffic), 20, and 30 (high traffic) CBR pairs. Each CBR source generates packets of size 512 Bytes, and transmits 4 packets per second. Each simulation lasts 900 seconds. We used a *thresh* value of 5 (allowable failures per link) for both TWOACK and S-TWOACK schemes. An experimental *timeout* value of 0.15 second was used in the TWOACK scheme. In the S-TWOACK scheme, this value was 1.15 seconds. The parameter *maximum_IDs_Carried* was set to 5 and *timeout_Last_Sent* to 1.10 seconds. 10 simulation runs (using different seeds) were used to obtain each data point.

We measured the following evaluation metrics for different percentage of misbehaving nodes in the network. *Packet Delivery Ratio:* defined as the ratio of the number of packets received at the destination node to the number of packets sent by the source node; *Routing Overhead:* defined as the ratio of the amount of routing-related transmissions in bytes (RREQ, RREP, RERR [7] and TWOACK) to the amount of data transmissions in bytes in a network; *End-to-End Delay:* defined as the time taken by a data packet to travel from the source to the destination through the source route. This metric is needed to examine whether the extra traffic due to the induction of new TWOACK packets affects packet latency.

Note that in the following discussions, when we mention the TWOACK scheme or the S-TWOACK scheme, we actually mean the TWOACK or the S-TWOACK scheme with the original DSR protocol.

B. Simulation Results and Discussions

Figure 3 compares the packet delivery ratio of the TWOACK scheme and the original DSR scheme as a function of different percentage of network nodes that are misbehaving. The percentage of misbehaving nodes in the network was varied from 0 (all nodes are well-behaved) to 40%. Different numbers of CBR pairs were simulated. From Fig. 3, we can observe that the packet delivery ratio of both schemes is close to 1 when no one is misbehaving. The packet delivery ratio decreases as more nodes in the network misbehave. This is due to the problem of missing routes and the overhead of searching for alternative routes. Compared with the original DSR scheme, our TWOACK scheme maintains a relatively high packet delivery ratio. For example, when there are 40% nodes that are misbehaving, the TWOACK scheme delivers about 85-90% of data traffic, while the original DSR scheme can only deliver 70-75%.

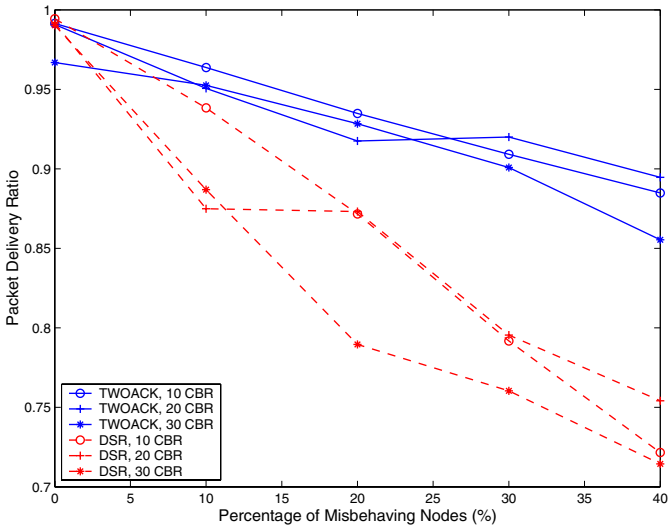

Fig. 3. Packet Delivery Ratio of TWOACK & DSR

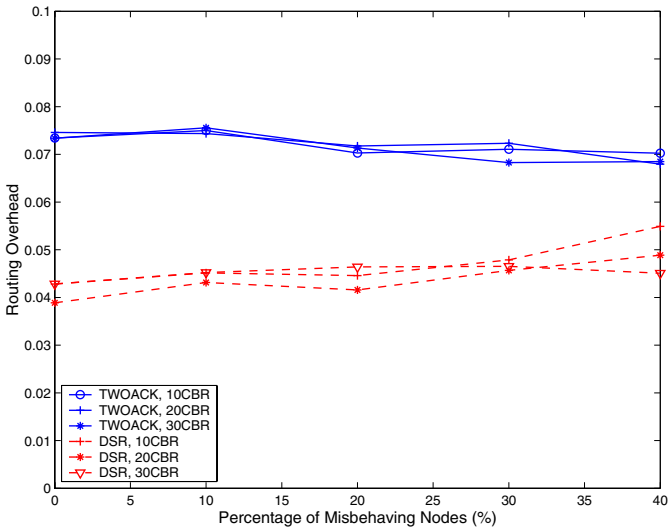

Fig. 4. Routing Overhead of TWOACK & DSR

In Fig. 4, we show the routing overhead of the TWOACK scheme and the DSR scheme. The network parameters are the same as those used to obtain Fig. 3. It is evident from the curves that the routing overhead increases from just over 4% for the original DSR scheme to around 7% for the TWOACK scheme. While such a 75% increase of routing overhead may seem large, it is only a net increase of 3 percentage points. Such an increase is mainly due to, the transmissions of the TWOACK packet for each data packet processed by each of the triplets and the transmissions of RERR packets to report misbehaving nodes.

In Fig. 5, we compare the routing overhead of the TWOACK, the S-TWOACK, and the DSR schemes for different percentages of misbehaving nodes. It can be observed that the routing overhead of the S-TWOACK scheme is about 3%, which is lower than the 4% of the original DSR scheme. This

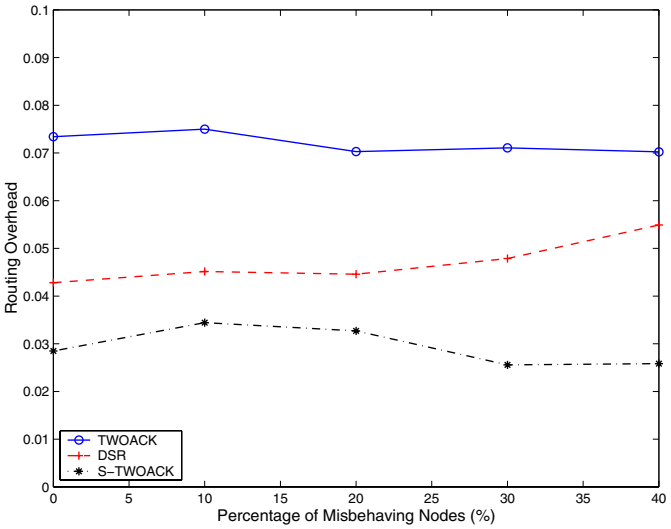

Fig. 5. Routing Overhead of TWOACK, S-TWOACK, and DSR

is due to an increase of data traffic being delivered successfully in the S-TWOACK scheme (the increase in control packets is negligibly small.)

Our further simulation results show that the S-TWOACK scheme has a slightly lower packet delivery ratio as compared to the TWOACK scheme. Our results also show that the TWOACK and the S-TWOACK schemes have a similar end-to-end delay of packet delivery as compared to the original DSR scheme, even under high traffic. These results can be found in [1].

V. Conclusions

Mobile Ad Hoc Networks (MANETs) have been an active area of research over the past few years, due to their potentially widespread application in military and civilian communications. Such a network is highly dependent on the cooperation of all its members to perform networking functions. This makes it highly vulnerable to selfish nodes.

In this paper, we have proposed and evaluated two network-layer acknowledgment-based schemes called TWOACK and S-TWOACK, which can be easily added-on to source routing protocols such as the DSR protocol. The schemes detect selfish nodes (links) so that other nodes may avoid them in future route selections, with the aim of overall improvement in end-to-end packet delivery ratio. Through simulations we showed that, in a network where up to 40% of the nodes are misbehaving, the TWOACK scheme improves the end-to-end packet delivery ratio from around 70% to almost 90% while increasing the overhead from 4% to 7%. The S-TWOACK scheme which is a derivative of the TWOACK scheme, achieves almost the same performance improvement without any routing overhead but with some expected increase of false alarms.

In our future research, we will work on the optimization of the parameters and extensive comparisons of the TWOACK and the S-TWOACK schemes with other related schemes.

Acknowledgment

The first two authors' work was performed when they were with the Department of Electrical Engineering and Computer Science at Syracuse University. This work was supported in part by the SUPRIA program of the CASE Center at Syracuse University.

References

[1] K. Balakrishnan, "Prevention of Node Selfishness in Mobile Ad Hoc Networks", *M.S. Thesis*, Department of EECS, Syracuse University, Syracuse, NY, USA, August 2004.

[2] S. Buchegger and J-Y. Le Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes, Fairness In Dynamic Ad-hoc Networks", *Proc. of the IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, June 2002.

[3] L. Buttyan and J-P. Hubaux, "Enforcing Service Availability in Mobile Ad-Hoc WANs", *Proc. of First IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, August 2000.

[4] Y-C. Hu, D. Johnson and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks", *Proc. of Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, June 2002.

[5] J-P. Hubaux, T. Gross, J-Y. Le Boudec, and M. Vetterli, "Toward Self-Organized Mobile Ad Hoc Networks: The Terminodes Project", *IEEE Communications Magazine*, January 2001.

[6] M. Jakobsson, J-P. Hubaux and L. Buttyan, "A Micropayment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks", *Proc. of Financial Crypto 2003*, January 2003.

[7] D. Johnson, D. Maltz, Y-C. Hu, J. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", Internet-Draft, February 2002.

[8] Y. Liu and Y. Yang, "Reputation Propagation and Agreement in Mobile Ad-Hoc Networks", *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, March 2003.

[9] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks", *Proc. of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, August 2000.

[10] P. Michiardi and R. Molva, "Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks", *Proc. of European Wireless Conference*, February 2002.

[11] P. Michiardi and R. Molva, "Game Theoretic Analysis of Security in Mobile Ad Hoc Networks", Research Report, April 2002.

[12] H. Miranda and L. Rodrigues, "Preventing Selfishness in Open Mobile Ad Hoc Networks", *Proc. of the Seventh CaberNet Radicals Workshop*, October 2002.

[13] C. Perkins, E. Belding-Royer and I. Chakeres, "Ad Hoc On Demand Distance Vector (AODV) Routing", IETF Internet-Draft, October 2003.

[14] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J. Tygar, "SPINS: Security Protocols for Sensor Networks", *Proc. of ACM SIGMOBILE Seventh Annual International Conference on Mobile Computing and Networking (MobiCom)*, July 2001.

[15] S. Zhong, J. Chen, and Y. Yang, "Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks", Technical Report, Yale University, July 2002.

[16] The Network Simulator (ns-2), URL: http://www.isi.edu/nsnam/ns/.