Lightweight Acknowledgement-Based Method to Detect Misbehavior in MANETs

Vahid Heydari and Seong-Moo Yoo
Department of Electrical and Computer Engineering, The University of Alabama in Huntsville
Huntsville, Alabama 35899, USA
[e-mail: {vh0008, yoos}@uah.edu]
* Corresponding author : Seong-Moo Yoo

Received July 13, 2015; revised September 9, 2015; accepted October 3, 2015; published December 31, 2015

Abstract

Mobile Ad hoc NETworks (MANETs) are the best choice when mobility, scalability, and decentralized network infrastructure are needed. Because of critical mission applications of MANETs, network security is the vital requirement. Most routing protocols in MANETs assume that every node in the network is trustworthy. However, due to the open medium, the wide distribution, and the lack of nodes' physical protection, attackers can easily compromise MANETs by inserting misbehaving nodes into the network that make blackhole attacks. Previous research to detect the misbehaving nodes in MANETs used the overhearing methods, or additional ACKnowledgement (ACK) packets to confirm the reception of data packets. In this paper a special lightweight acknowledgement-based method is developed that, contrary to existing methods, it uses ACK packets of MAC layer instead of adding new ACK packets to the network layer for confirmations. In fact, this novel method, named PIGACK, uses ACK packets of MAC 802.11 to piggyback confirmations from a receiver to a sender in the same transmission duration that the sender sends a data packet to the receiver. Analytical and simulation results show that the proposed method considerably decreases the network overhead and increases the packet delivery ratio compared to the well-known method (2ACK).

Keywords: Mobile ad hoc networks (MANETs), acknowledgment, random access MAC, misbehaving node, blackhole attack.

http://dx.doi.org/10.3837/tiis.2015.12.023 **ISSN : 1976-7277**

1. Introduction

When mobility and scalability are of vital importance, wireless networks are always preferred. In critical mission applications like military conflict or emergency recovery, Mobile Ad hoc NETworks (MANETs) are the best choice because of having a decentralized network infrastructure. MANETs rely on the benevolence of nodes within the network to forward packets from a source node to a destination node (multi-hop forwarding). Because of critical mission applications of MANETs, network security is one of the most important requirements. Unfortunately, due to the open medium and the lack of nodes' physical protection, attackers can easily capture and compromise nodes to achieve their goals. In particular, most routing protocols in MANETs assume that every node in the network is trustworthy and presumably not malicious. So, attackers can easily compromise MANETs by inserting misbehaving nodes into the network. A misbehaving node is a node that cooperates in route finding phase and forwards all control packets (such as route request, route error, and route reply) correctly, but silently drops all data packets. Many research efforts have been devoted to detect the misbehaving nodes in MANETs that can be classified into two categories: monitoring-based methods and acknowledgment-based methods. The main problem with monitoring-based methods is that the overhearing method is unable to determine accurately the misbehaving nodes because of transmission instability in wireless networking environments that have been explained in [3]. The acknowledgment-based method solves the problems of monitoring-based methods by using additional ACK packets to confirm the reception of data packets. However, this performance improvement has an additional cost which is the overhead made by these ACK packets.

When a method adds one extra packet to the network layer, the method is actually adding at least four extra packets (RTS, CTS, Data, ACK), according to IEEE 802.11 protocol, to the MAC layer. Note that in MAC layer, IEEE 802.11 protocol uses mandatory ACK packets to confirm the reception of transmitted data packets. Therefore, in this paper a special lightweight acknowledgement-based method, named PIGACK, is developed that on the contrary to existing methods, it uses ACK packets of MAC layer instead of adding new ACK packets to the network layer for confirmations. In fact, this novel method uses ACK packets of MAC 802.11 to piggyback confirmations from a receiver to a sender in the same transmission duration that the sender sends a data packet to the receiver. In this method, a message authentication code is used to authenticate the sender of the packet. Analytical and simulation results show that the proposed method considerably decreases the network overhead and increases the packet delivery ratio compared to the well-known method (2ACK [10]).

Our Contributions–We develop the PIGACK method for detecting and isolating misbehaving nodes. Compared to state-of-the-art, PIGACK provides the following additional features:

- Misbehaving node detection instead of misbehaving link detection: PIGACK can detect misbehaving nodes and prevent using any links that include them (increase in packet delivery ratio).
- Informing other source nodes by informing the neighbors (at least half of them) of the newly misbehaving node detected to prevent using this misbehaving node by other source nodes (increase in packet delivery ratio).

- Decrease in routing overhead: PIGACK uses ACK packets of MAC 802.11 to piggyback confirmations from a receiver to a sender in the same transmission duration that the sender sends a data packet to the receiver. Therefore, PIGACK does not use any extra ACK packets.

Paper Organization– The rest of this paper is organized as follows. Related work is covered in Section 2. Proposed PIGACK method is introduced in Section 3 including attack types. Section 4 and 5 cover analytical model and simulation. Finally, conclusion is explained in Section 6.

2. Related Work

In this section, previous methods, to detect and mitigate malicious nodes in MANETs, are explained. As above-mentioned, these methods can be classified into two main categories; monitoring-based methods and acknowledgment-based methods. We introduce some important methods of each category and explain their advantages and disadvantages.

2.1 Monitoring-Based Methods

The main idea of monitoring-based methods is to exploit wireless medium nature that each transmission can be heard by one-hop neighbors. In these methods a sender places itself in a promiscuous mode after transmitting a packet to overhear the retransmission by the forwarding node. Using this method, a node can know whether the packet, which has been sent to its neighbor, is indeed forwarded or not. Marti et al. [11] proposed two extensions to the DSR protocol, Watchdog and Pathrater. The Watchdog is based on promiscuous mode operation of the nodes to detect misbehaving nodes. Pathrater uses the knowledge from Watchdog to choose a path without any misbehaving nodes. Buchegger and Boudec [4] proposed CONFIDANT, an extension to the DSR which adds the reputation system and the trust manager to the Watchdog and Pathrater mechanisms. When misbehaving nodes are detected via Watchdog, an alarm is sent to the other nodes, defined as friends, to isolate the misbehaving nodes from the network. Winjum et al. [15] described a mechanism that uses a trust metric for routing. This method is based on detection of routing information sent by trusted nodes or sent by other nodes in the network. Each node should store two routing tables, one for trusted routes and another for unreliable routes. Pirzada et al. [12] used trust and reputation in the DSR protocol. The trust value calculated in the promiscuous mode is used to select a trusted route without misbehaving nodes. But in this method a wrong recommendation may occur and be used by other nodes. Chang et al. [5] used distributed authentication authorities which employ distributed trusted groups for the authentication process. However, this method works in the dynamic MANET with high delay. Chen et al. [6] employed a distributed trusted routing framework that authenticates messages, nodes and routes. But this method needs the certificate authority. Xia et al. [16] presented a dynamic trust prediction model to evaluate the trustworthiness of nodes, which is based on the nodes' historical behaviors, as well as the future behaviors via the prediction of extended fuzzy logic rules prediction.

The base of all above-mentioned methods is the trust degree of each node resulted from the promiscuous mode operation of the nodes. In other words, each node promiscuously listens to its wireless medium, catches all packets in its antenna range and sends these packets to the network layer, which saves all of them in the table and uses a watchdog timer to detect the misbehavior of the one-hop neighbors. However, the overhearing technique is not likely to be

accurate for MANETs due to varying noise levels, varying signal propagation characteristics in different directions, and interference from competing transmissions [3].

2.2 Acknowledgment-Based Methods

These methods are based on acknowledgments for packets delivery confirmation. Balakrishnan et al. [2] proposed a network-layer acknowledgment-based scheme, called TWOACK, to detect misbehaving nodes. The TWOACK is based on the Watchdog method; however, it does not use the overhearing technique (the promiscuous mode). It sends two-hop acknowledgment in the reverse direction to confirm that the intermediate nodes cooperated in the packet forwarding. The TWOACK resolved some problems of the promiscuous mode such as ambiguous collisions, receiver collisions and the limited transmission power. Liu et al. [10] proposed 2ACK scheme that uses acknowledgments like the original TWOACK scheme. Two major differences between them are as follows:

1) The 2ACK scheme uses an authentication mechanism to avoid tampering of the 2ACK packets, whereas the TWOACK scheme does not use authentication.
2) The intermediate nodes in the 2ACK scheme may only send a fraction of 2ACK packets with the tradeoff of increased delay in misbehavior detection (more packet dropping). Therefore, the 2ACK scheme may have less overhead than the TWOACK method because in the TWOACK scheme the intermediate nodes send the whole TWOACK packets for every data packet. Note that we will use 2ACK packets for all received data packets to achieve the best performance of 2ACK (minimum delay in misbehavior detection) in our analysis and simulation.

The shortcomings with the mentioned acknowledgement-based methods are as follows:

1) Message overhead: both methods have overhead messages (2ACK and TWOACK packets).
2) The 2ACK scheme cannot detect “false alarms”. The 2ACK waits a certain time to receive ACK packet; however, sometimes, because of congestion or link fail detection time, a link (a pair of nodes) may be reported as a misbehaving link. In this situation the 2ACK does not have any method to exonerate this link. In the worst case, “intentional false alarms” (slander attacks) may occur. Assume that we have n1, n2, n3, n4, n5, n6, n7 on the path. If n3 receives the ACK packet of n5, but sends a false alarm to n1, the link n4-n5 will be saved as a misbehaving link. However, neither n4 nor n5 is a misbehaving node. By this attack n3 can accuse all links between its neighbors and their neighbors.

Some other methods, mentioned below, claimed that not only can detect blackhole attacks but also are able to detect collusion blackhole attacks due to collusive misbehaving nodes which cooperate in the route discovery, but refuse to forward data packets and do not disclose the misbehavior of each other. Note that we assume no collusion among misbehaving nodes in our method.

Awerbuch et al. [1] proposed an on-demand secure routing protocol (ODSBR) to identify misbehaving links and collusion blackhole attacks. In ODSBR, audited nodes acknowledged audited packets back to the source. Then the source performs a binary search to identify the misbehaving link. The main problem of the ODSBR is explained in this example: Suppose in ODSBR a misbehaving node drops all data packets except the packets that contain probes for one of the next nodes, only the data packets contain probes will reach the destination node. Therefore, the packet delivery ratio cannot exceed 50%.

Sun et al. [14] proposed a scheme, called NACK, which adopts an acknowledgment-based approach and compares timestamps to resist collusion attacks. The NACK scheme, similar to

the TWOACK and the 2ACK schemes, sends two-hop acknowledgments in the reverse direction (NACK packets) to confirm that the intermediate nodes cooperated in the packet forwarding. For this goal, the NACK scheme uses a timestamp comparison and an additional route between source nodes and destination nodes to detect malicious nodes. The NACK method uses time synchronization techniques and assumed that the second route does not have any misbehaving node.

Shakshuki et al. [13] proposed a hybrid scheme, called EAACK, to detect collusion blackhole attacks which includes three phase. First of all, the destination must send an end-to-end acknowledgement packet (ACK) to the source node. If the source node does not receive the ACK packet, it starts the second phase, where EAACK uses the TWOACK to detect two neighbor nodes as misbehaving nodes. However, the source node does not immediately trust the misbehavior report. When the second phase is finished, the source node starts the third phase. On this phase, the source node uses a second route and asks the destination whether it received the removed packet or not. This phase helps the source node to detect false misbehaving reports. The EAACK, like the NACK, assumed that the second route is available and does not have any malicious nodes. The main problem of the EAACK is similar to the ODSBR that a malicious node can do partial dropping without being recognized.

Two main common problems of above-mentioned acknowledgment-based methods are:

1) Misbehaving link detection instead of misbehaving node detection: These methods cannot detect a misbehaving node. They can only determine two neighbor nodes and cannot detect which of them is misbehaving. By the error reports from one of the intermediate nodes, the source node detects the link between two nodes as a misbehaving link, but a link cannot be misbehaving, in fact, anyone of the two end nodes of a link may be misbehaving. The source node will not use this link in the future, but due to lack of accurate detection of the misbehaving node, the node may be used in another path by the source node. Therefore, a misbehaving node can be used for more than one time without being recognized. In fact, a misbehaving node can be placed on different paths equal to the number of its neighbors.
2) When a misbehaving link is detected, an alarm will be sent to the source node. All intermediate nodes between the reporter and the source node as well as the source node will be informed of this link. However, a new source node which was not informed must detect this misbehaving link by itself, which consumes time and decreases the packet delivery rate. In fact, the 2ACK does not have any method to inform other source nodes of misbehaving links detected.

In this paper we try to solve the above-mentioned problems. Therefore, our goals are: decreasing the network overhead, detecting false alarms, detecting misbehaving nodes instead of misbehaving links, and informing other source nodes by informing the neighbors (at least half of them) of misbehaving node detected.

3. Proposed PIGACK Method

3.1 Assumptions

- DSR is the protocol used for network layer.
- There is no collusion among misbehaving nodes.
- Each node has a unique ID and cannot be spoofed.

- A misbehaving node is a node that cooperates in route finding phase and forwards all control packets (such as route request, route error, and route reply) correctly, but silently drops all data packets. Each node in a path may be malicious except source and destination nodes which are assumed to be trusted.
- A suspicious node is a node that was reported as a misbehaving node by another node. However, it already sent the PIGACK packet of its next node to the neighbors (and source in new route) to exonerate itself. If a node is detected two times as a suspicious node (reported by different nodes), it will be marked as a misbehaving node.
- By default, each node has a table and for every node in the network it has an entry with four fields: its ID, its public key (or symmetric key), malicious flag, and reporter. The malicious flag is zero by default and if it is detected that the node is misbehaving, this flag will be two. This flag will be one if the node is detected as a suspicious node. The reporter field is the node ID of the node that reports this misbehavior.
- Each node has a data structure. Each entry include: *PreNodeID*, *NextHopNodeID*, *SecondHopNodeID*, C_{pkts}, C_{mis}, and a list that contained *PID*, *Send*, *State*, *Timeout*, *Confirmation*. For node *B*, as an example, on the route {*S, A, B, C, D*}, *PreNodeID* is *A*'s ID, *NextHopNodeID* is *C*'s ID and *SecondHopNodeID* is *D*'s ID. *PreNodeID* for *S* is zero (because *S* is the source). *NextHopNodeID* for *D* and *SecondHopNodeID* for *C* are zero (because *D* is the destination). C_{pkts} is a counter of forwarded data packets and C_{mis} is a counter of missed data packets. *PID* is the packet ID of received data packet. Received confirmation of next node in a PIGACK packet is saved on *Confirmation* field and *Send* field will be set when saved confirmation is forwarded to the previous node. *State* is zero when sending a new packet and *Timeout* (timeout period) is set. When next hop node's confirmation is received, *State* will be set to one and after receiving confirmation of the second-hop node, it will be set to two and *Timeout* will be reset.
- Each PIGACK packet contains confirmations that the number of them is determined by $C_{confirm}$ ($1 \leq C_{confirm} \leq$ maximum number of packets waited for confirmation). Each confirmation includes three fields: *NodeID*, *PID*, and *MAC*. *MAC* is the message authentication code created by the node that its ID is *NodeID* and *PID* demonstrates the ID of the received data packet by this node.

We add one field to the route error packet of DSR named *Malicious*. If this field is set, the next node of this error producer is a misbehaving node. *Malicious* is zero for a normal route error packet.

3.2 Proposed Method

According to IEEE 802.11, an ACK packet must be sent by the receiver to confirm the reception of the packet. Our goal is piggybacking some information to these ACK packets of MAC layer instead of using ACK packets of network layer. Hereinafter, we use "PIGACK" as the ACK packet of MAC layer which contains confirmations. Each node when sends a data packet and receives PIGACK packet must save the next node's confirmation (included in the received PIGACK) and send back with its confirmation for the next packet sent from the previous node inside a new PIGACK packet. In fact, each node when receives a data packet must send back a PIGACK packet that includes:

1) The confirmation of reception of this packet created by this node, and
2) The confirmation of the previous packet created by the next node to prove its honesty.

This idea works well in a network without any congestion so only one packet may be waited for confirmation in each node. Therefore, sometimes a node may not have any ready confirmation of its next node to send back to the previous node or may have more than one

confirmation ready to send back. Therefore, we add $C_{confirm}$ to PIGACK packet to specify the number of confirmation included in this packet. PIGACK method can be summarized in the pseudocode as follows:

A. Pseudocode of PIGACK Method

For all algorithms:
Require: *Path* includes [*PreNodeID*, *NextHopNodeID*, *SecondHopNodeID*] obtained from packet header.
Require: *Entry* of *Path* in the data structure of this node.
Require: *List* of (*PID*, *Send*, *State*, *Timeout*, *Confirmation*) for *Entry*

A.1 Sending a data packet:

if *Entry* is empty **then**
  Entry = new entry for this *Path* in data structure
  List ← Ø, C_{pkts} ← 0, C_{mis} ← 0
end if
C_{pkts} ++
add new entry to *List* of this *Entry*
PID ← packet ID, Send ← 0, State ← 0
Timeout ← maximum delay per 2-hops + current time

A.2 Receiving a data packet and sending a PIGACK packet:

if *Entry* is empty **then**
  Entry = new entry for this *Path* in data structure
  List ← Ø, C_{pkts} ← 0, C_{mis} ← 0
end if
prepare PIGACK packet and insert ACK fields to it
prepare new *confirmation*
NodeID ← ID of this node, *PID* ← received packet ID, *MAC* ← sign (*NodeID*, *PID*) by private key
insert confirmation into PIGACK
$C_{confirm}$ ++
for each stored *confirmation* in *List* **do**
  if *send* equals to zero **then**
    add stored *confirmation* to PIGACK
    $C_{confirm}$ ++, *Send* ← 1
  end if
end for

A.3 Receiving a PIGACK packet:

if *MAC* is not valid or $C_{confirm}$ equals to zero **then**
  prepare route broken alarm for upper layer
  Malicious ← 0
else
  find entry of *List* for this *Path* where *PID* = pachet ID
  store received confirmation to the entry of *List*
  state ← 1
  if *SecondHopNodeID* equals to zero **then**
    state ← 2 // the next node is destination so this node should not wait for the second hop node
  end if
  if *PreNodeID* equals zero **then**
    send ← 1 //this node is a source so should not send back confirmations
  end if
  for each extra confirmation in PIGACK **do**
    if *MAC* is verified by *SecondNodeID's key* **then**
      find entry of *List* where its *PID* equals to confirmation's *PID*
      State ← 2
    Timeout ← 0
    end if
  end for
end if
for each *PID* in *List* with *State* < 2 **do**
  if current time > *Timeout* and *Timeout* does not equal to zero **then**
    C_{mis} ++
    Remove this entry from *List*
  end if
end for
if C_{mis}/C_{pkts} > *threshold* for *Entry* **then**
  prepare route broken alarm for upper layer
  Malicious ← 1

31: remove *Entry* from data structure
32: **end if**

Nodes have a table for each *Path* (specified by the previous node, the next hop node, and the second hop node). When a node receives a packet, the node adds its ID as a new entry to this table and calculates a timeout period for it. This timeout period is equal to the maximum waiting time to prepare the confirmation of the second hop node in the next hop node for this packet ID (so at the next sending packet, this confirmation will be received). Therefore, this timeout is equal to the maximum delay per two hops. If we use the 95-percentile of response time of each node as the delay per hop, we must specify a node as a misbehaving node if the ratio of C_{mis}/C_{pkts} is more than five percent (*Threshold*). If the timeout is reached and this node did not receive specific confirmation (the second hop confirmation) after sending a new packet to the next node, this node removes the table entry for that packet ID and increases C_{mis}. If the ratio of C_{pkts} to C_{mis} is more than specific *Threshold*, the next node ID will be stored as a misbehaving node and an error signal will be sent up from MAC layer to network layer that mentions this misbehavior and DSR will set *Malicious* field of the route error packet to inform other nodes of this misbehaving node. This route error packet will be sent to the source node IP address as the destination IP address (similar to normal DSR). However, the MAC layer will send it as a broadcast message. By this way, neighbors of this node will see this misbehaving node report, but only one of them (according to the header of DSR) will forward this packet to the source node. These neighbors and the source node will remove all routes in their caches that include the misbehaving node and set the malicious flag of this node to two and store the ID of its reporter in the table. Malicious flag in the table will be set to one if the misbehaving node proves its honesty so it will be change from a misbehaving node to a suspicious node.

Using this technique that neighbors are informed of this misbehavior can increase the packet delivery ratio because they abstain to forward any packet from other sources to this misbehaving node.

We put some change to DSR to behave correctly with its route request and route reply packet. When a node receives a route request or reply packet, it must check the route in the packet header (list of intermediate nodes saved on the header) and drop this packet if the route contains any pre-reported misbehaving node (or two neighbor suspicious nodes).

We change the ACK packet format of MAC layer to include confirmations to it. As mentioned in assumptions (Section 3.1), the new packet (PIGACK packet) format includes: (see **Fig. 1**)

- Standard ACK fields
- $C_{confirm}$ (the number of confirmations included in this PIGACK)
- One or more confirmations (*NodeID*, *PID*, and *MAC*)

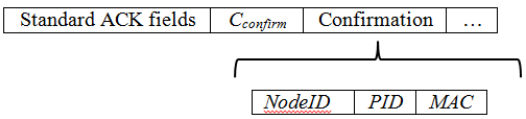

Fig. 1. PIGACK packet format (number of confirmation is equal to $C_{confirm}$).

3.3 Message Authentication Code

We have two choices for message authentication code:

1) Digital signature (default method for PIGACK): In this method we use elliptic curve online/offline digital signature [18] that needs 160 bits key size for security level 80 and creates signatures with the size of 320 bits. We assume offline key distribution method and

sequential packet ID started from a number mentioned in the first data packet (this packet does not need confirmations). Therefore, nodes will have enough time to prepare the next signature before receiving the next data packet.

2) UMAC: This is the fastest reported message authentication code in the cryptographic literature used symmetric technique and a fast universal hash function [9]. The key size is 128 bits (security level 80) and its output is 64 bits. We assume offline distribution shared keys so each node has shared keys of other nodes. The advantages of this method are: (1) smaller key size and output size and (2) fast enough to prepare the code during a short interframe space (SIFS) time interval (the time between receiving a packet and sending an ACK according to IEEE 802.11), so we do not need the sequential packet ID. On the other hand, its disadvantage is: having problem (similar to 2ACK) to detect attack type 2 and 3 explained in Section 3.4. Hereinafter, we use PIGACK_U to refer to PIGACK that uses UMAC for message authentication codes.

3.4 Attack Types

In this section we explain different types of attacks related to our method. Assume a route from *S* to *D* includes *A*, *B*, and *C* ({*S*, *A*, *B*, *C*, *D*}). In these attack types, a node may only drop data packets (blackhole attack), create false alarms by sending wrong 2ACK (or PIGACK) to slander other nodes (slander attack), or participate with other misbehaving nodes (collusion attack). Note that we assume no collusion attacks in the network so we will not discuss collusion blackhole, slander, or framing attack. The attack types that we consider are as follows:

Type 1. *C* does not send the 2ACK (or PIGACK) packet to *B*.

2ACK: *A* does not receive 2ACK of *C* so it detects link *B*-*C* as a misbehaving link and reports this link to the source. Other links that include *C* may be used by this source and all the links may be used by other sources.

PIGACK: A PIGACK packet contains the standard ACK packet and missing a PIGACK means missing the ACK, so *B* will send a route error packet according to IEEE 802.11 and DSR. On the other hand, if *C* sends a PIGACK without received *D*'s confirmation, *B* will detect *C* as a misbehaving node after the timeout period.

Type 2. *C* sends its 2ACK (or PIGACK) packet to *B* with a wrong message authentication code (or wrong *MAC* of its confirmation inside PIGACK).

2ACK: *B* cannot detect any problem because the packet is signed for *A* and according to the message authentication code used by 2ACK, only *C* and *A* can read this packet. Therefore, *A* will detect link *B*-*C* as a misbehaving link.

PIGACK: *B* reads the digital signature of *C* and detects this fault and behaves similar to the situation that *B* did not receive the PIGACK of *C* (creates the route error packet).

Type 3. *A* creates a false alarm and mentions *B* as a misbehaving node.

2ACK: This problem cannot be solved in 2ACK because the misbehavior of A will not be detected by the source so the source will mention link *B*-*C* as a misbehaving link.

PIGACK: *B* is one of the neighbors of *A* so *B* will see this false alarm and will send the PIGACK packet containing the confirmation of *C* as a broadcast message with Time To Live (TTL) equal to two to inform the neighbors of *A* and will send it in a new route to the source. In this situation, the neighbors and the source will mention both *A* and *B* as misbehaving nodes. Although *B* is detected wrongly as a misbehaving node, *A* isolates itself by doing this attack. Therefore, creating a false alarm is so costly for a misbehaving node.

4. Analytical Model for Overhead Ratio

In this section we use an analytical model using queuing theory to obtain the network overhead ratio. For calculating the overhead in this paper, we focus on the overhead of both MAC and network layers instead of considering only the overhead of network layer because of two reasons:

1) Some information is piggybacking to the ACK packet of MAC layer and 2ACK method adds new packets to the network layer, so we have to obtain the overhead from both layers.
2) By focusing on the overhead of both MAC and network layers we can calculate the real overhead that includes all DSR and MAC headers, routing packets of DSR, and control packets of MAC layer. For example, when we add 2ACK packets to the network layer, we will have some additional overhead because of MAC control packets such as Request to Send (RTS), Clear to Send (CTS) and ACK packets. The number of RTS packets is not equal to the number of transmissions because of collisions. Therefore, adding a new packet or changing a packet size will change the network load and will have effect on the number of collisions and RTS packets so our real network overhead must be obtained by considering both layers instead of only the network layer.

First of all, we will briefly introduce IEEE 802.11 MAC protocol and explain the assumptions and queuing network model. Then, we explain our overhead analysis.

4.1 MAC Protocol

IEEE 802.11 protocol defines a distributed coordination function (DCF), for sharing access to the medium, based on the CSMA/CA protocol. A node listens to the channel before the transmission to determine whether it is free or not. If the medium is sensed to be free for a DCF interframe space (DIFS) time interval, the transmission will proceed by sending a RTS packet to announce the upcoming transmission. When the destination node receives the RTS, it will send a CTS packet after a SIFS time interval. The source node is allowed to transmit the data packet only if it receives the CTS packet correctly. The destination node must send an ACK packet after a SIFS time on receiving the data packet.

If the medium is busy or the node does not receive the CTS packet, the node must defer its transmission until the end of the current transmission and then must wait an additional DIFS interval and generate a random backoff time. This process minimizes collisions during contention between multiple nodes.

Definition 1: *Backoff time:* backoff time = Random() × SlotTime where Random() is a pseudorandom integer among [0, CW-1] where $CW_{min} \leq CW \leq CW_{max}$ and the SlotTime is one of the physical layer characteristics.

The contention window (CW) parameter must take the CW_{min} as the initial value. Every node must maintain a Station Short Retry Count (SSRC) which must take an initial value of zero and its range is [0, 7]. After each unsuccessful transmission retry, the SSRC will be incremented and the CW will take the next value that is the result of multiplying the previous value by two. If the CW reaches the CW_{max}, it will not continue to increase. If the SSRC reaches the last number in its range, the packet will be dropped. After each successful transmission, the CW and SSRC will be reset to the initial value.

Definition 2*: The mean backoff time for the first try:* $E[T_B] = (CW_{min} \times SlotTime) / 2$.

Because we want to consider the medium as a server and nodes as its customer (will be explained in Section 4.3), our server always will have at least one customer that is in the first round (CW_{min}) of backoff time. Therefore, we use CW_{min} in Definition 2.

4.2 Assumptions

- The total number of nodes in the network and the size of network must satisfy the network connectivity according to the node density of the network.
- The network consists of N nodes that are distributed uniformly and independently over the area of $X \times Y$.
- We assumed, for simplicity, an equal transmission range denoted by $r(n)$ and equal carrier sensing range denoted by $c(n)$ for each node.
- Let r_{ij} denote the distance between node i and j. They can communicate with each other as the neighbor nodes if $r_{ij} \leq r(n)$.
- A free-space model in uncontested environments is used.
- DSR control packets are not included in our model. For this reason, we assume stationary nodes (avoiding broken routes and route error packets) and do not use the first 50 sec of simulation results (avoiding route setup duration with route request and reply packets).
- According to IEEE 802.11, we have seven retry limits for a packet transmission. We assume no packet dropping in our model. Therefore, at most after six transmission collisions, a packet will be transmitted.
- Each interfering neighbor node of a source or destination freezes its backoff counter as soon as detecting RTS or CTS packets. We do not consider noises. Because of these two assumptions, after receiving CTS by the source node, transmission will be done without any collision or problem. Therefore, the number of CTS and ACK packets are equal to the number of transmitted data packet per hop. Collisions may effect only on the number of RTS packets.

4.3 Queuing Network Model

An M/M/1 queue is used in our model. It is assumed that the arrivals occur at rate λ according to a Poisson process and the service times have an exponential distribution with parameter μ. There are no buffer or population size limitations and the service discipline is First Come, First Served (FCFS). To analyze the M/M/1 queue, we need to know only the mean arrival rate λ and the mean service rate μ in jobs per unit time. The mean service rate can be obtained by $\mu = 1/E[S]$ (S is the service time of the server).

Definition 3*: Utilization factor of the server:* $\rho = \lambda / \mu$ [8]

Definition 4: *Probability of zero job in the system:* $P_0 = 1 - \rho$ [8]

Each node and its neighbors within the interference range must compete for access to the medium. When a node gets the right to use the medium, its neighbors that have a packet to send, must freeze their backoff counter and wait for medium to be free. We need to obtain this freezing time to calculate the service time of each node. For this reason, we consider the medium as the server and the interfering range's nodes as the customers in the queue. In the service discipline of the server (the medium) each node with the backoff counter equal to zero will have access to the server.

In the traffic model, each node could be a source, a destination or a relay node. The packet generation process at each node is an independently and identically distributed Poisson process and the number of connections (packet flows) is constant and equal to nn. The arrival rate in jobs per unit time for each connection is λ_j ($1 \leq j \leq nn$). The mean data packet length (L_{data}) is the sum of data packet size of DSR, DSR header, the mean number of hops multiplied by four bytes (DSR packets include the route in their headers), and MAC header. The size of each 2ACK packet is equal to L_{2ACK} bits and the headers of network and MAC layers are included in L_{2ACK}, RTS, CTS, and ACK. The channel bit rate is constant and equal to W. The

parameters of the queuing network model summarized in **Table 1**.

Before calculating the new parameters of the model, we need to obtain some basic parameters. To obtain the mean number of hops (E[HOP]), the method of [7] is used in this paper. Other basic parameters are explained in definitions below.

Definition 5*: The mean number of neighbors* ($E[\zeta_r]$) *in range r:* $E[\xi_r] = \frac{N.r^2(-4r(X+Y)+8r(\sqrt{2}-1)+3\pi(XY))}{3(XY)^2}$ [7].

Definition 6: *The mean random distance per hop*: $E[r_i] = \frac{7.r(n)}{9}$ [7]

Table 1. Network model parameters

N	Number of nodes
nn	Number of connections
S	Service time of the server
μ	Service rate of the server
ρ	Utilization factor of the server
λ	Arrival rate of the server
λ_j	Arrival rate of each connection
λ_i	Arrival rate of each node
r(n)	Transmission range
c(n)	Detection range
i(n)	Interfering range
ξ_r	Number of neighbors in range r
r_i	Distance per hop
HOP	Number of hops between a source and a destination
H_i	Number of hidden neighbors of node i
T_D	Time duration to send a full data packet
T_B	Backoff time
W	Channel bit rate
P_C	Probability of occurrence of collision between a node and its hidden neighbors
P_0	Probability of zero job in the system
H_{data}	Header size of data packet
L_{data}	Data packet size (headers included)
L_{2ACK}	2ACK packet size (headers included)
L_{ACK}	ACK packet size (headers included)
$L_{confirm}$	Confirmation size
R_{2ACK}	Ratio of the number of 2ACK packets to the number of total data and 2ACK packets
C_{RTS}	Number of RTS packets per each data or 2ACK packet
O_{HOP}	Overhead per hop for each data packet
C_{pkt}	Number of data and 2ACK packets per hop
O_{Ratio}	Overhead ratio

4.4 Mean Number of Hidden Neighbors

In wireless networks, each node has three different ranges:

- The transmission range ($r(n)$) is mainly determined by transmission power and radio propagation properties.
- The carrier sensing range ($c(n)$) is usually determined by the antenna sensitivity.

- The interfering range ($i(n)$) is the range within which a node in receiving mode can be interfered by another transmission, leading to a collision at receiver.

The first two above-mentioned ranges are constant. However, the third one depends on the distance between the sender and the receiver. Note that the transmission power level assumed to be constant. Nodes within the interference range of a receiver are called hidden neighbors. Here, a receiver sends a CTS packet to the sender (as the response to the received RTS packet), if no any of its hidden neighbors are transmitting a packet.

We need to obtain the mean number of hidden neighbors to calculate the collision probability. For this purpose, we must obtain the interfering range and use this range for calculating the area of hidden neighbors.

Lemma 1: H_i is the number of hidden neighbors of node i. Then

$$E[H_i] = E\left[\xi_{1.78\times E[r_i]}\right] - 1 \qquad (1)$$

Proof: Using this usual assumption that the threshold of signal to noise ratio (SNR_THRESHOLD) equals to 10 and the mean distance between senders and receivers equals to $E[r_i]$ (Definition 6), The interfering range will be equal to $1.78 \times E[r_i]$ [17]. Therefore, using Definition 5, the mean number of nodes in this range equals to $E\left[\xi_{1.78\times E[r_i]}\right]$. Note that one of these nodes is the sender and other nodes are hidden neighbors.

4.5 Overhead Analysis

Lemma 2: R_{2ACK} is the ratio of the number of 2ACK packets to the number of total data and 2ACK packets. Then

$$R_{2ACK} = \begin{cases} \dfrac{2(E[HOP]-1)}{3(E[HOP])-2} & 2ACK \\ 0 & DSR\ or\ PIGACK \end{cases} \qquad (2)$$

Proof: Each data packet traverses E[HOP] to reach a destination and according to 2ACK method, each node on the route, except the first two nodes, must send back a 2ACK packet and each 2ACK packet traverses two hops. The number of nodes on the route is equal to the number of hops plus one. Therefore, (2) is proved. If we use DSR or PIGACK, we will not have 2ACK packets.

Lemma 3: L_{ACK} is the length (bytes) of ACK packet in MAC layer. Then

$$L_{ACK} = \begin{cases} ACK & DSR\ or\ 2ACK \\ ACK + 1 + \left(\left(\dfrac{2E[HOP]-1}{E[HOP]}\right) L_{confirm}\right) & PIGACK \end{cases} \qquad (3)$$

where $L_{confirm} = \begin{cases} 16\ B & PIGACK\_U \\ 48\ B & PIGACK \end{cases}$

Proof: DSR and 2ACK use standard ACK packets, but PIGACK has standard ACK fields plus some new fields. We assume no packet dropping in the network, so each node (except destinations) must forward the received packet and send back PIGACK packet. Therefore, the number of confirmations in each PIGACK is equal to two (one for received packet and the other for previously transmitted packet). However, destinations only must send back one PIGACK. Therefore, the mean number of confirmations is equal to $\left(\frac{2E[HOP]-1}{E[HOP]}\right) L_{confirm}$ and for each confirmation we have *NodeID* (four bytes), *PID* (four bytes), and *MAC* (40 bytes for PIGACK and eight bytes for PIGACK_U). Besides, we have a new field $C_{Confirm}$ in each PIGACK packet that needs one byte.

Lemma 4: λ_i is arrival rate in jobs per unit time for each node. Then

$$\lambda_i = \begin{cases} \dfrac{(\sum_{j=1}^{nn} \lambda_j) \times E[HOP]}{N} & DSR\ or\ PIGACK \\ \dfrac{(\sum_{j=1}^{nn} \lambda_j) \times (3(E[HOP]) - 2)}{N} & 2ACK \end{cases} \quad (4)$$

Proof: The number of sending (transmitting by the source node) and forwarding (transmitting by the intermediate nodes) per each packet of each connection is equal to the mean number of hops. Therefore, the total arrival rate of the network is $(\sum_{j=1}^{nn} \lambda_j) \times E[HOP]$. However, according to 2ACK method and proof of Lemma 3, we must add $(\sum_{j=1}^{nn} \lambda_j) \times 2(E[HOP] - 1)$ for 2ACK method. Therefore, the total arrival rate for 2ACK is equal to $(\sum_{j=1}^{nn} \lambda_j) \times (3(E[HOP]) - 2)$ and the mean of arrival rate for each node is (4).

Lemma 5: T_D is the time duration to send a full data packet. Then

$$T_D = \left[\left(\left((R_{2ACK} \times L_{2ACK}) + \left((1 - R_{2ACK}) \times L_{data}\right)\right) + RTS + CTS + L_{ACK}\right) / (W/8)\right] + 3SIFS \quad (5)$$

Proof: According to IEEE 802.11 and the fact that the packet is a 2ACK packet with the probability of R_{2ACK}, and is a data packet with the probability of (1- R_{2ACK}) and by using Lemma 2 and 3, we will obtain (5).

Lemma 6: S is the service time of the server. Then

$$E[S] = (E[T_B] + T_D + DIFS) + P_C \left(E[T_B] + \frac{RTS}{(W/8)} + DIFS\right) \quad (6)$$

Proof: According to the IEEE 802.11 protocol, the medium is given to a node for transferring a packet (T_D from Lemma 5) after a backoff time (E[T_B] from definition 2). The next packet will be sent after DIFS from the last sending packet. Therefore, each packet needs E[T_B] + T_D + DIFS to be served. (See **Fig. 2**)

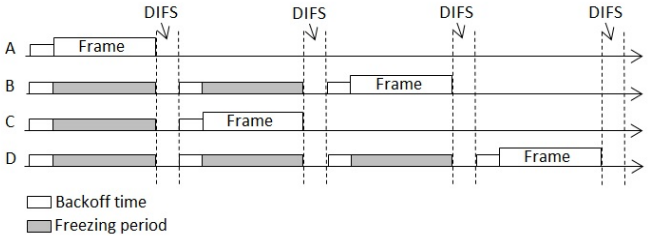

Fig. 2. Backoff mechanism of 802.11 MAC.

Theorem 1: P_C is the probability of collision between the source node's packets and the hidden neighbors' packets. Then

$$P_C = \frac{E[H_i] \times \lambda_i (E[T_B] + T_D + DIFS)}{1 - \left(E[H_i] \times \lambda_i \left(E[T_B] + \frac{RTS}{(W/8)} + DIFS\right)\right)} \quad (7)$$

Proof: we would like to know the probability of sending packet by hidden neighbors. P_0 is the probability of zero job in the system. According to Definition 4, $1 - P_0 = \rho$ is the probability of at least one job in the system. The simultaneous transmission by a hidden node and the source node makes a collision at the destination node. Therefore, P_C is equal to the probability of at least one hidden neighbor's job in the system (ρ). Using this fact that $\mu = \frac{1}{E[S]}$, Definition 3,Lemma 1, and Lemma 4 we have:

$$P_C = \frac{E[H_i] \times \lambda_i}{\mu} = E[H_i] \times \lambda_i E[S] \quad (8)$$

using Lemma 6:

$$P_C = E[H_i] \times \lambda_i \big((E[T_B] + T_D + DIFS) + P_C(E[T_B] + RTS + DIFS)\big) \quad (9)$$

Theorem 2: C_{RTS} is the number of RTS packets per each data packet (or 2ACK packet) transmission over one hop. Then,

$F(0) = 2, F(1) = 1 + (1 - P_C) + P_C(F(0)), \ldots, F(n) = 1 + (1 - P_C) + P_C(F(n - 1))$ $(2 \le n \le 5)$, and $E[C_{RTS}] = (1 - P_C) + P_C(F(5))$.

Proof: according to IEEE 802.11 protocol, a packet will be dropped after seven unsuccessful retries. We assume no packet dropping in our model. Therefore, we must include six collision possibilities and in the last retry the packet must be transmitted. In each retry step, the packet is sent by the probability of (1-P_C) (one RTS is sent), obtained by Theorem 1, and a collision may occur by the probability of P_C (previous RTS is wasted and new one must be sent). $F(0)$ is the last retry that includes one wasting RTS because of the last collision and one new RTS that the packet will be sent after this RTS.

Lemma 7: O_{HOP} is the overhead per hop for each data packet. Then

$$O_{HOP} = E[C_{RTS}] \times RTS + \big((1 - R_{2ACK}) \times H_{data}\big) + (R_{2ACK} \times L_{2ACK}) + CTS + L_{ACK} \quad (10)$$

where H_{data} is the header size of data packets (all headers included in H_{data}).

Proof: For transmitting each data packet (or 2ACK packet), a node must send $E[C_{RTS}] \times RTS$, receive a CTS, send the packet (if the packet is a data packet so the overhead is its header, but if it is a 2ACK packet, the whole packet is overhead), and receive an ACK packet (normal ACK packet size or our new ACK packet used in our new method). Equation (10) uses Theorem 2, Lemma 2, and Lemma 3.

Lemma 8: C_{pkt} is the number of data and 2ACK packets per hop. Then

$$C_{pkt} = \begin{cases} \sum_{j=1}^{nn} \lambda_j \times E[HOP] & DSR \text{ or } PIGACK \\ \sum_{j=1}^{nn} \lambda_j \times (3(E[HOP]) - 2) & 2ACK \end{cases} \quad (11)$$

Proof: According to the proof of Lemma 2 and 4, (11) will be proved.

Corollary: O_{Ratio} is the overhead ratio. Then

$$O_{Ratio} = \frac{C_{pkt} \times O_{HOP}}{\sum_{j=1}^{nn} \lambda_j \times E[HOP] \times (L_{data} - H_{data})} \quad (12)$$

Proof: According to Lemma 7 and 8, the overhead per hop is equal to $C_{pkt} \times O_{HOP}$. The denominator of equation is equal to the sum of data packet (without headers) per hop.

In the next Section we will compare our analytical results with those obtained from simulations.

5. Simulations

In the simulations, a version of network simulator (NS-2) is used. DSR and MAC 802.11 modules are modified to simulate PIGACK and 2ACK methods. Note that DSR and 2ACK methods use standard MAC 802.11, but PIGACK uses a modified version of that. The traffic and scenario patterns are randomly made using "cbrgen" and "setdest" tools of NS-2. Note that "setdest" uses uniform distribution for node places and random waypoint model for the

movement of nodes. We use five traffic and five scenario patterns (25 simulation runs) for each simulation result. The simulation parameters are summarized in **Table 2**.

First, the comparison between analytical and simulation results according to the assumption of analytical model are explained and then, the simulation results for a real network by considering the mobility, DSR routing packets, and assumptions in Section 3.1, are given.

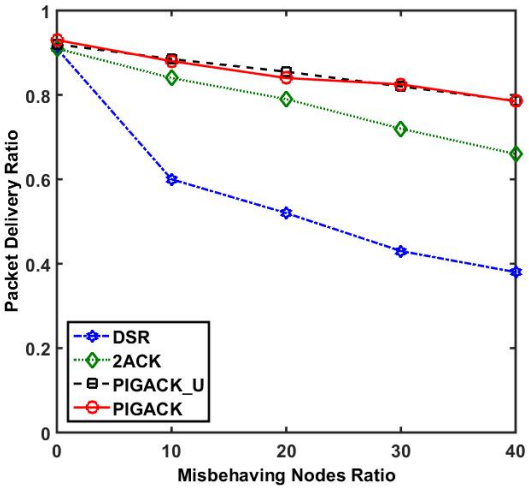

Fig. 3. Packet delivery ratio as a function of misbehaving nodes ratio.

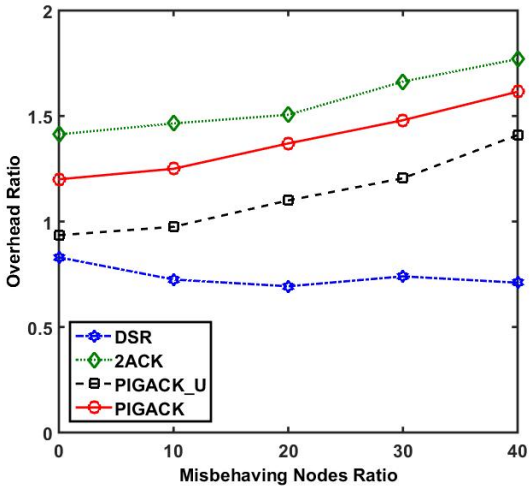

Fig. 4. Overhead ratio as a function of misbehaving nodes ratio.

Table 2. Simulation parameters

N	80
Simulation area	1000 m × 1000 m
Simulation time	100 sec
r(n)	250 m
c(n)	550 m
nn	20
λ_j	1
W	11 Mbps
Data packet size	512 B
L_{data}	607.32 B
L_{2ACK}	134 B
RTS (all headers are included)	44 B
CTS (all headers are included)	38 B
ACK (all headers are included)	38 B
SlotTime	20 μs
SIFS	10 μs
DIFS	50 μs
CW_{min}	32

Table 3. Comparing the overhead ratio obtained from analytical results with those obtained from simulations

	Simulation results	Analytical results
DSR	0.43	0.42
PIGACK_U	0.5	0.49
PIGACK	0.62	0.61
2ACK	1.16	1.13

5.1 Comparing Analytical Results with Simulation Results

Table 3 shows the comparison between analytical and simulation results for calculating overhead ratio. We used the assumptions introduced in Section 4.2 in the simulations. The values of parameters in Table 3 were used in analytical results for this comparison.

5.2 Simulation Results

In this part we use the simulation parameters summarized in **Table 2** with these differences:

- Node mobility is considered with the maximum speed and pause time equal to 20 m/sec and 20 sec, respectively.
- Misbehaving nodes are considered 0, 10, 20, 30, and 40 percent of the whole nodes.

The following metrics are used to evaluate the performance of the DSR, PIGACK, PIGACK_U, and 2ACK methods with regard to UDP traffic:

- Packet delivery ratio: The ratio of the number of data packets received by destination nodes to the number of data packets sent by source nodes.
- Overhead ratio: The ratio of the amount of routing packets (route request, route reply, route error, and 2ACK) and MAC control packets (RTS, CTS, ACK, and PIGACK) per hop (headers are included) to the amount of data packets per hop. Note that in the overhead ratio used in analytical model in Section 4, we assumed no routing packet and the network was stationary. However, in this section we assume that we have a MANET with DSR routing packets.

5.2.1 Packet Delivery Ratio

In **Fig. 3** we compare the packet delivery ratio as a function of misbehaving node ratio. The misbehaving node ratio ranges from 0 (all nodes are trustworthy) to 40 (40% of nodes are misbehaving). Here, we observe that increasing the number of misbehaving nodes reduces the packet delivery ratio. As expected, PIGACK and PIGACK_U methods have shown a much better performance in packet delivery ratio compared to 2ACK and DSR methods.

There are two reasons of low packet delivery ratio of 2ACK. (1) Lack of detecting misbehaving node (it detects misbehaving link) lets a misbehaving node exist still in different routes used by the source node. (2) Lack of informing other nodes to prevent the presence of detected misbehaving node on the routes. Because of this reason, each source node must detect each misbehaving link by itself.

5.2.2 Overhead Ratio

In **Fig. 4** we compare the overhead ratio of the methods as a function of misbehaving node ratio. The higher overhead in the all methods except the DSR is due to the transmission of extra acknowledgment packets (2ACK) or extra information in ACK packets of MAC layer (PIGACK and PIGACK_U). When a misbehaving node is detected, the source node must use other routes for packet transmission and this process may need new routing packets to find new routes. Because of this reason, the overhead of 2ACK and our methods are increased with the misbehaving nodes ratio. However, DSR does not detect the misbehaving nodes so it does not need to find new routes. Besides, increasing the misbehaving nodes will decrease the probability of broken route because they truncate the route and packet route errors may be created only by nodes between sources and misbehaving nodes not between sources and destinations and this is the reason of decline in the overhead of DSR as shown in **Fig. 4**.

5.3 One More Comparison Using Real Circumstance Example

2ACK had the minimum overhead between acknowledgment-based methods and we found that PIGACK has less overhead in comparison with 2ACK. So we can say that PIGACK has the minimum overhead between all acknowledgment-based methods.

To the best of our knowledge, previous acknowledgment-based methods used misbehaving link detection technique like [1], [2], [10], [13], [14]. Therefore, we can also claim that PIGACK has the best packet delivery ratio between acknowledgment-based methods when we do not have collusion among misbehaving nodes. We have two reasons for this claim:

1) Misbehaving node detection instead of misbehaving link detection.
2) Informing neighbors of misbehaving nodes.

We use a simple network example, shown in Fig. 5, for better understanding. The square-shaped nodes are the sources and the circular shaped nodes are the destinations. Node 13 is a malicious node that drops all data packets. Misbehaving link detection methods may try all three routes containing node 13 for sending packets from node 11 to node 15. Note that node 1 also may use node 13 to send packets to node 25. Therefore, each source node should detect the misbehaving node by itself and this detection may need to test all links containing the misbehaving node.

On the other hand, in PIGACK method when node 12 detects node 13 as a malicious node, node 12 informs its neighbors (nodes 6, 7, 8, 13, 18, 17, 16, and 11) so these neighbors will not use node 13 on a path to a destination. In fact, one time misbehavior detection is enough for at least half of the sources.

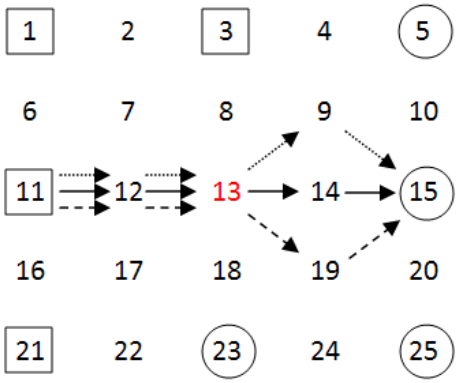

Fig. 5. Simple network example.

Therefore, PIGACK method has the minimum overhead and maximum packet delivery ratio in comparison with other acknowledgment-based methods. Also we know that acknowledgment-based methods are more accurate than monitoring-based methods because the overhearing technique is not likely to be accurate as explain in **[3]**.

6. Conclusion

In this paper a special lightweight acknowledgement-based method, named PIGACK, is developed that uses ACK packets of MAC 802.11, instead of adding new ACK packets to the network layer, to piggyback confirmations from a receiver to a sender in the same transmission duration that the sender sends a data packet to the receiver. According to the analytical and simulations results, the proposed method considerably decreases the routing overhead and increases the packet delivery ratio compared to the well-known method (2ACK).

In the future work, we will investigate how to arm PIGACK to resist collusion misbehaving nodes that make collusion blackhole and slander attacks in MANETs.

References

[1] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "ODSBR: An on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 4,pp. 1–35 , Jan. 2008. Article (CrossRef Link)

[2] K. Balakrishnan, J. Deng, and P.K. Varshney, "TWOACK: Preventing Selfishness in Mobile Ad Hoc Networks," in *Proc. of IEEE WCNC*, pp. 2137-2142, 2005. Article (CrossRef Link)

[3] R.V. Boppana and X. Su, "On the Effectiveness of Monitoring for Intrusion Detection in Mobile Ad Hoc Networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 8, pp. 1162–1174, 2011. Article (CrossRef Link)

[4] S. Buchegger and J. Y. Boudec, "Performance analysis of the CONFIDANT protocol: cooperation of nodes: Fairness in dynamic ad-hoc networks," in *Proc. of MobiHOC*, pp. 226–236, 2002. Article (CrossRef Link)

[5] C. Chang, J. Lin, and F. Lai, "Trust-group-based authentication services for mobile ad hoc networks," in *Proc. of ISWPC '06*, pp. 37–40, 2006. Article (CrossRef Link)

[6] T. Chen, O. Mehani, and R. Boreli, "Trusted routing for VANET," in *Proc. of ITST '09*, pp. 647–652, 2009. Article (CrossRef Link)

[7] V. Heydari, and S.M. Yoo, "Mean Number of Hops in Flooding-Based Ad Hoc Networks," *Submitted to Inf. Process. Lett.*,2015

[8] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling," *Wiley- Interscience*, New York, 1991. Article (CrossRef Link)

[9] T. Krovetz, "RFC 4418: UMAC: Message Authentication Code using Universal Hashing," 2006. Article (CrossRef Link)

[10] K. Liu, J. Deng, P. K. Varshney, and K. Balakrishnan, "An acknowledgment-based approach for the detection of routing misbehaviour in MANETs," *IEEE Trans. Mobile Comput.*, vol. 6, no. 5, pp. 536–550, 2007. Article (CrossRef Link)

[11] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," in *Proc. of MobiCom*, pp. 255–265, 2000. Article (CrossRef Link)

[12] A. A. Pirzada, A. Datta, and C. McDonald, "Incorporating trust and reputation in the DSR protocol for dependable routing," *Computer Communications*, vol. 29, no.15, pp. 2806–2821, 2006. Article (CrossRef Link)

[13] M. Shakshuki, N. Kang, and T. R. Sheltami, "EAACK—A Secure Intrusion-Detection System for MANETs," *IEEE Trans. Ind. Electron.*, vol. 60, no. 3, pp. 1089–1098, 2013. Article (CrossRef Link)

[14] H. M. Sun, C. H. Chen, and Y. F. Ku, "A novel acknowledgment-based approach against collude attacks in MANET," *Expert Systems with Applications*, vol. 39, no. 9, pp. 7968–7975, 2012. Article (CrossRef Link)

[15] E. Winjum, P. Spilling, and O. Kure, "Trust Metric Routing to Regulate Routing Cooperation in Mobile Wireless Ad Hoc Networks," in *Proc. of Wireless Conference 2005-European wireless*, pp. 1–8, 2005. Article (CrossRef Link)

[16] H.Xia, Z. Jia, X. Li, L. Ju, and E. H.-M. Sha, "Trust prediction and trust-based source routing in mobile ad hoc networks," *Ad Hoc Networks*, vol. 11, no. 7, pp. 2096–2114, 2013. Article (CrossRef Link)

[17] K. Xu, M. Gerla, and S. Bae, "How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks," in *Proc. 2002 IEEE GLOBECOM*, pp. 72–76, Nov. 2002. Article (CrossRef Link)

[18] A.C.-C. Yao and Y. Zhao, "Online/Offline Signatures for Low-Power Devices," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 2, pp. 283–294, 2013. Article (CrossRef Link)

Vahid Heydari received a Bachelor of Science from the University of Science and Culture and a Master of Science from Payame Noor University in computer engineering in Tehran, Iran. He is a PhD student in electrical and computer engineering and MS student in cybersecurity simultaneously at the University of Alabama in Huntsville. His research interests include mobile ad-hoc, sensor, and vehicular network security and moving target defense. He is a student member of ACM, IEEE Computer Society and Communications Society.

Seong-Moo Yoo is an Associate Professor of Electrical and Computer Engineering at the University of Alabama in Huntsville (UAH). Before joining UAH, he was an Assistant Professor at Columbus State University, Columbus, Georgia –USA. He earned MS and PhD degrees in Computer Science at the University of Texas at Arlington. His research interests include computer network security and wireless network routing. He has co-authored over 90 scientific articles in refereed journals and international conferences. He is a senior member of IEEE and a member of ACM.