OMH - Suppressing Selfish Behavior in Ad hoc Networks with One More Hop*

Chengqi Song and Qian Zhang
Hong Kong University of Science and Technology
Email: {lars, qianzh}@cse.ust.hk

Abstract

In ad hoc networks, wireless nodes rely on each other to transmit data over multi-hops by forwarding packets. A selfish node may decide not to forward packets for other nodes to save its own resource but still use the network to send and receive data. Such a selfish behavior can degrade network performance significantly. Most existing work took observation, reputation and token based mechanisms. However observation based mechanism suffers from mobility and collusion; reputation and token based mechanisms suffer from system complexity and efficiency. In this paper, we propose One More Hop (OMH) protocol which suppresses selfish behavior from a totally new angle. Basing on the fact that the selfish but rational nodes still want to receive and send packets, if a node can not determine whether a packet is destined for it or not, it can not drop the packet. With modified routing protocol and cryptographic techniques, OMH achieves this design target. It is robust and efficient. The simulation shows that OMH works well under different network situations.

1. Introduction

An ad hoc network consists of a group of wireless nodes, which cooperate with each other by forwarding packets to enable multi-hop communications. It requires no centralized or fixed network infrastructure. Ad hoc networks can be deployed in crisis applications such as in battlefield and also in civilian applications such as vehicular systems. In the former applications, all the nodes of the network belong to a single authority and have a common goal. But in civilian applications, the nodes may belong to different authorities such as different persons. In such situation, some nodes may attempt to be selfish; the selfish nodes are unwilling to share their resources like CPU cycles, battery power, or available network bandwidth to forward packets that are not of direct interest to them. The problem of selfish behaviors can be a serious problem to the overall system performance.

Before addressing this problem we need to point out that selfish behavior is different from malicious behaviors. The target of the node with selfish behavior is to save its own resource like battery life and network bandwidth, but not to disturb other nodes. More importantly, the selfish nodes are rational in that they want to leverage the other nodes in the network to send and receive data. In this paper we only focus on selfish behaviors.

Many works have been conducted on eliminating selfish behaviors in the literature. We can classify them into several categories: observation-based schemes, reputation-based schemes, and token-based schemes.

The observation-based solution is to detect selfish nodes and isolate them [1]. If a node behaves selfishly, it can be overheard by its neighbors and be isolated from the transmission. Although have been well discussed, those solutions have some common disadvantages: they require nodes to observe each other, thus, if a node colludes with some other nodes, it will have less chance to be detected. Moreover, when node mobility is taken into consideration, the efficiency of observation and isolation will become lower.

The reputation-based schemes make use of reputation to discourage selfish behaviors [2] [3]. Nodes evaluate the reputation of its neighbors basing on the completion of the requested packet forwarding. The nodes with bad reputation will not be trusted in further route selections. This type of solution has to make sure the reputation information is highly secure, which causes high complexity of the system design. Moreover, this type of solution suffers the same problem with observation-based solutions: i.e., the selfish nodes will be isolated and can not be used to help forward data packets and consequently the network performance is degraded.

In token-based solutions [4] [5], token is paid for helping forwarding packets. Token can be virtual money or of other forms. If a node is selfish, it can not earn enough token to send out its own packets. Such solutions need to maintain

Qshine08 July 28-31, 2008, Hong Kong, China

DOI 10.4108/ICST.QSHINE2008.4256

a trustable virtual token market, which is usually complex and not always practical.

To solve the problem of selfish behaviors, an elegant solution should be able to handle the following issues efficiently: it needs to work effectively under various network conditions, such as different traffic patterns, network sizes and mobility; meanwhile it should not bring too much computation complexity; if possible, it should not target at isolating some nodes, instead it should try to leverage all the nodes to help packet transmission. In this paper, designing such an elegant scheme to suppress selfish behaviors is our target. Since the existing schemes have different built-in problems as we mentioned above, we need to think in a totally new way.

As we mention above, a selfish node is rational. It needs to transmit and receive its desired data and meanwhile save its own resource as much as possible. Basing on this basic and fundamental observation, we have a totally new viewpoint of how to suppress selfish behavior, that is, if a node can not know whether a packet is destined to it or not, it can not drop the packet otherwise it may potentially lose data. However, there are some challenges to make a packet's real destination undeterminable. Firstly, in classical routing protocols, the destination of a packet is the last hop of the route path therefore easy to be detected. Secondly, destination can also be estimated by checking a packet's content. Thirdly, although we want to hide the real destination of a packet, the packet should still be able to be transmitted correctly.

In this paper, considering our basic viewpoint and the corresponding challenges, we proposed a totally new protocol named One More Hop (OMH) to suppress selfish behavior. At first, a longer path mechanism is used so that the destination is no longer fixed as the last hop but can be any intermediate hop. Secondly, encryption is used in a novel way so that only the upstream node can find out whether a packet is destined to the node or not, and encryption also makes the content of a packet can not be observed without a special key, which is known only to destination node and its upstream node. As a result intermediate nodes cannot drop packets because they cannot determine if the packet is destined for them or not. The transmission ends at the upstream node of destination node, so transmission in OMH always has one more hop, and that's why the protocol is named as One More Hop.

In summary, we have made the following major contributions in this paper:

1. Instead of trying to detect selfish behaviors and reduce its impact, OMH directly prevents selfish behaviors because a node's selfish behaviors will potentially cause its own data loss in OMH.

2. Instead of trying to isolate and remove selfish nodes from route path, OMH forces selfish nodes to work normally and makes use of them to forward packets.

3. Performance of OMH is analyzed with compare to Watchdog protocol.

The remainder of this paper is organized as follows. In Section 2 we summarize related works. Our system model and the detailed design of OMH protocol are presented in Section 3. Then some further discussions about OMH are presented in Section 4. In Section 5 we evaluate OMH protocol in multiple aspects and finally short conclusion is given in Section 6.

2. Related Work

Mitigating selfish behavior is a challenging topic in ad hoc networks; many researchers have proposed different types of schemes to address this issue. Basically their works can be classified into three different categories.

The first category is based on observation [1] [6] [7] related mechanism. Watchdog and Pathrater [1] uses observation-based techniques to detect misbehaving nodes and report them to the source of the traffic, then allows nodes to choose better path by avoiding the misbehaving nodes. This observation and reporting mechanism has also been used in sensor networks recently [6] [7]. However, this type of schemes has common drawbacks. At first, because of nodes mobility, the selfish nodes can move to other places if its selfish behavior has been reported by its neighbors. Moreover, such scheme may have problem with collusion behavior, as a misbehaving node may not report its collusion nodes selfish behavior. The weakness on collusion problem is mentioned in [1].

There are some other schemes adopting trust and reputation mechanism [2] [8] [3] [9] to address the selfish behavior issue. In CONFIDANT [8], each node in the network hosts a monitor for observation and reputation records, and a path manager is used by nodes to adapt their behavior according to reputation information. The first problem of such trust and reputation based mechanisms is that they take up considerable resources due to the constant transmission of observation data, which serves no purpose other than to monitor node behavior. The second problem is that such systems suffer from vulnerabilities due to exchanging second hand reputation information. Another problem is that most reputation-based solutions can not handle collusion. A recent work DARWIN [9] works out a new reputation mechanism to resist collusion, however our OMH can handle this problem in a more direct way.

Virtual token is another way to mitigate selfish behavior and encourage cooperation in ad hoc networks [4] [10] [5]. Nuglets [4] is a per-hop payment scheme; the payment units are called nuglets and reside in a secure tamper-proof

module in each node. They find that given such a module, increased cooperation is beneficial not only for the entire network but also for individual nodes. Virtual payment method is also used in [10] [5] and recent research work [11] to help enforcing packet forwarding. In [12] the authors pointed out that under the general token mechanism, a user needs to forward more than it sends and also limits the amount of information that a user can send at a given time. Moreover, such solutions generally need to maintain a complex token system and make it secure.

We design OMH in a totally different way comparing with the existing works. It leverages the fact that selfish nodes are rational and still need to receive their own desired packets, and works in a very efficient and elegant way. The details are introduced in the next section.

3. OMH Protocol

In this Section we introduce our assumption about the network model, then introduce the motivation of OMH, and at last introduce the design of OMH.

3.1. System Model and Assumptions

Before talking about the design of OMH protocol, we need to specify the system model and our discussion assumptions.

Firstly, since selfish behavior is a common problem in ad hoc networks, we consider networks with some selfish nodes and also some unselfish nodes, but not 100% nodes are selfish, which is a very extreme situation and we dont consider it in this work.

Secondly, we focus on selfish behaviors but not malicious behaviors. Selfish nodes drop packets to save their own resource but are still rational and do want to receive data transmitted to them. However malicious nodes may suffer the threat of losing their own data to disturb other nodes.

Lastly, in this paper we use IEEE 802.11 as MAC protocol and DSR as route protocol to describe the sample implementation of OMH.

3.2. Design of OMH

In this subsection we present the details of OMH protocol. The OMH protocol encrypts packets, makes the real destination of a packet not equal to the last hop of route path, and gives nodes acknowledgements from its next hop if the node is the real destination. To achieve the design target, we need to follow the 5 important steps.

Step *i*. Encryption

In this step, we encrypt a packet before sending it. If a node wants to open a packet, it has to get the key at first.

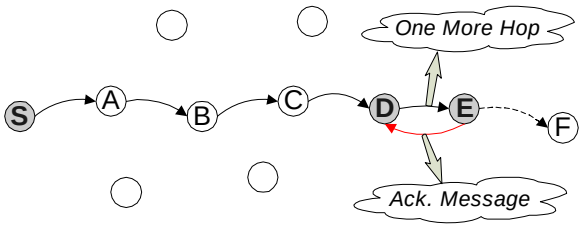


Figure 1: One More Hop and Longer Path (*D* is the destination, *D*-*F* is the one more hop, *S* to *F* is the longer path)

When sender *S* has a packet *M* to send to destination *D*, at first, *S* generates a random key K. K is used to encrypt packet M with symmetric encryption algorithms such as Data Encryption Standard (DES). In the following steps the packet body will be the cipher $DES_K(M)$, in which DES_K means DES encryption with key K. The cost of DES key generation and encryption is very low and can be performed very fast.

Step *ii*. Route Discovery - Longer Path

In OMH, the last hop of a packet' route path is not necessarily the same as the destination, because the longer path mechanism is used.

In OMH we not only need to find the path from source to destination, but also try to find a longer path. For example, in Fig. 1, *S*-*A*-*B*-*C*-*D* is a path from source node *S* to destination node *D*, while *S*-*A*-*B*-*C*-*D*-*E*-*F* is a *longer path*. Such path can be found basing on route protocols. Take DSR as an example.

In DSR [13], every node has a local route table to some other nodes. If no existing path can be found in local route table, then route discovery is performed. At first route request (RREQ) message is flooded from the source node to nearby nodes; then when the destination node or some other nodes who know the path to destination receive the RREQ, they send route reply (RREP) messages back to source node along the reversed path. As a result the source node finds a new path according to the RREP messages.

With some modification on DSR, longer path can be found. When the destination node replies route request, it can find a path in its local route table and attach it in RREP message. In the sample shown in Fig. 1, when node *D* replies *S*'s route request, it finds a path *D*-*E*-*F* in its route table and attaches in RREP message. Then S can combine the new discovered path *S*-*A*-*B*-*C*-*D* with it and get a longer path *S*-*A*-*B*-*C*-*D*-*E*-*F*.

When an intermediate node replies route request, it can also find longer path in its local route table. For example, if node *C* knows a path *C*-*D*-*E*-*F*, then when it receives RREQ, it can attach this path in RREP and send back to *S*. In this way a longer path is also found.

If we can find such longer path, go to step *iii*. Otherwise, OMH can still work, and the algorithm goes to step *v*.

Step *iii*. Sending Packet with Longer Path

In step *i* we get the message body $DES_K(M)$; in this step we are going to send out the packet through the longer path found in step *ii*. Before sending the packet, we need to add some extra information to the packet header. This extra information tells every node whether a packet is destined to its previous node. With such information, a node can only know a packet is destined to it or not from the node at next hop, therefore it has to forward received packets to next hop.

In DSR, the route path is stored in the packet header. In OMH, instead of the original route path, the longer path is stored in packet header. In the example, it's *S-A-B-C-D-E-F* stored in the packet header but not *S-A-B-C-D*, even though the real destination is *D* but not *F*. As a result, when a node receives a packet, it can not read the packet, because the packet body is encrypted; it also can not know the real destination, because of the longer path mechanism.

Here comes a problem, since the real destination D can neither know what the key *K* is nor the real destination is itself, how can it receive the packet successfully? The answer is the special information stored in packet header. In OMH, the source node prepares information for each node in the route path, and encrypts it using asymmetric encryption algorithm RSA with each node's public key. The information for each node includes 1) whether the packet's real destination is its previous node and 2) if it is, then the key *K* to open the packet. The header cipher is shown in Fig. 3:

For *A*	: $RSA_{PK(A)}$(Real Destination is *S*?	No	)
For *B*	: $RSA_{PK(B)}$(Real Destination is *A*?	No	)
For *C*	: $RSA_{PK(C)}$(Real Destination is *B*?	No	)
For *D*	: $RSA_{PK(D)}$(Real Destination is *C*?	No	)
For *E*	: $RSA_{PK(E)}$(Real Destination is *D*?	Yes, K	)
For *F*	: $RSA_{PK(F)}$(Real Destination is *E*?	No	)

Figure 2: OMH Packet Head 1 (*D* is destination, K is key)

Where RSA is a widely used asymmetric encryption algorithm; $PK(A)$ means public key of node *A*; $RSA_{PK(A)}()$ means cipher of RSA encryption with A's public key. We can see that node *E* is told the real destination is its previous node *D*, and also told the key K to open the packet.

Step *iv*. Receiving Packet

In this step we describe what happens when a node receives a packet. At first it extracts the part of information encrypted with its own public key, then knows whether the packet is destined to its previous node; if yes, then send back an acknowledgement, otherwise the packet may belongs to itself or following nodes, to get the answer, it has to forward the packet to next hop.

In the example, when node *B* receives this packet, it will do:

$DeRSA_{PK(B)}(RSA_{PK(A)}$(*Real Destination is A? No*))

In which $DeRSA_{PK(B)}$ means RSA decryption with private key of node *B*. The decryption result is *"Real Destination is A? No"*. Now we can see that node *B* can only know this packet's real destination is not *A*, but whether the real destination is *B* itself, and how to open the packet? *B* has no way to know now. So node *B* can only forward the packet to the next hop *C*.

The same thing happens to *C*, and also to *D*. Only when *E* receives this packet, it can read that *"Real Destination is* D*? Yes;* K*"*. Now node *E* knows the real destination is *D*, then it sends an acknowledgement back to *D* together with the key *K*. By now *D* receives the packet successfully and *E* does not need to forward the packet to *F* anymore. Now the transmission finishes successfully.

We can see that in OMH, when an intermediate node receives a packet, it has no choice but to forward it, otherwise it may lose its own data. This is guaranteed by both longer path and cryptographic mechanism. And this is also why we call our protocol "One More Hop": the transmission always reaches the next hop of the real destination. On the other hands, though the longer path can be much longer than the path from source to destination, the final transmission can only be one more hop longer, because when the next hop node of real destination (node *E*) needs not to forward the packet anymore.

The transmission is done by now, and step *v* is for special cases.

Step *v*. Sending and Receiving Packet without Longer Path

If a longer path can not be found, the source node also prepares information for each node; the content also includes two parts, and the second part has a little different with in step *iv*: 1) whether this packet's real destination is the node's previous node and 2) if the destination is the node itself, then the key K to open the packet; the second part is different with in step *iii*, now OMH tells the real destination the key K, but not its next hop node, because we can not find such node in this situation. Now the cipher is shown in Fig. 3:

For *A*	: $RSA_{PK(A)}$(Real Destination is *S*?	No	)
For *B*	: $RSA_{PK(B)}$(Real Destination is *A*?	No	)
For *C*	: $RSA_{PK(C)}$(Real Destination is *B*?	No	)
For *D*	: $RSA_{PK(D)}$(Real Destination is *C*?	No, K	)
For *E*	: $RSA_{PK(E)}$(Real Destination is *D*?	No	)
For *F*	: $RSA_{PK(F)}$(Real Destination is *E*?	No	)

Figure 3: OMH Packet Head 2 (*D* is destination, K is key)

When the intermediate nodes *A*, *B*, *C* receive the packet, the situation is the same as in step *iv*, they still have no choice but to forward the packet. When node *D* receives the packet, it can directly use its private key to do decryption and get the key K. OMH protocol still works.

4. Further Discussions

After introducing the details of the OMH protocol, in this section we discuss why it can eliminate selfish behavior and how it can work better.

4.1. Behavior of Last Node

In the OMH protocol, we require the last node to send an acknowledgement and a key back to its previous node. But the last node has no incentive to cooperate. It can selfishly keep quiet to save energy and time. However, from the following discussion we can see that both the motivation and chance of last node's selfish behaviors are reduced a lot.

At first, this message is very short. This message is an acknowledgement together with a key. The energy and time spent on such a short message is relatively little. From the incentive point of view, the incentive to act selfishly is relatively low.

Secondly, if the last node is a selfish node; its chance to drop packets is reduced a lot. Without OMH, the selfish node can selfishly drop packets at anytime. But with OMH, only when it is the last node of a transmission, can it drop the acknowledgement packet selfishly. Suppose average path length is n, then without OMH, the chance for the selfish node to act selfishly can be as much as 1; while with OMH, the chance is reduced to $1/(n+1)$, because a node has only a chance of $1/(n+1)$ at average to be the last node of a path.

Moreover, this problem can be solved by improving the strategy of construction of longer path. This method is described in next subsection.

4.2. Improved Longer Path

Because the last node can drop acknowledgement as it wish, when the fraction of selfish nodes is very high, the last nodes become a bottleneck of throughput. So it's very useful if we can select a better last node when constructing a longer path.

In Watchdog and Pathrater protocol [1], when a node in a route path is observed to be dropping packets, the current path breaks and a new route discovery is performed, in which the nodes marked as selfish will be avoided. This process is performed iteratively until a stable route path is found.

Inspired by Watchdog, the longer path can also be improved by observation, but only for the last node and not for all nodes. In OMH, after a node sends a packet, the receiving node is supposed to either send back an acknowledgement message or transfer the packet to next hop. Both actions can be overheard by the sending node. As a result, when a node sends a packet but does not overhear any reactions from its next hop node, it's very possible that the next hop node is a selfish last node. When enough evidence is observed, a route error message is sent back to source node and current traffic flow breaks. In the new route discovery, the destination node will choose a path through another neighbor node which is not marked as selfish yet.

By such improvement, OMH fails only if all neighbors of a destination node are selfish, probability of which is very low comparing to random selection on longer path.

A problem of this improvement is that more route discoveries are performed. However, because only the last node in a flow can drop packets, so the enumeration is only among destination's neighbors. Comparing to Watchdog and Pathrater in which every node in a flow may cause new route discovery, the iterations needed in improved OMH is much less.

4.3. Collusion Resistance

Collusive selfish behavior means that several nodes collude with each other to benefit from saving resource. Some observation-based and reputation-based mechanisms suffer from such more intelligent selfish behavior, because collusive nodes may report fake observation result or reputation record to help their partners. But it's not a problem in OMH any more.

We can assume the nodes C_1, C_2, C_3, ..., C_n along a transmission path collude to try to drop the packets that do not belong to anyone of them; and the direction of transmission is from C_1 to C_n; then C_1 will know whether the packet belongs to its previous node; C_2 will know whether it belongs to C_1 etc; at last C_n will know whether it belongs to C_{n-1}. But can any one of them know whether the packet belongs to C_n? In OMH only the next hop of C_n knows, so C_n has to forward the packet to its next node, which is not anyone of the collusion group. That's to say in OMH collusion can not help selfish nodes at all. This sample can be extended to more general cases and we can find that OMH is very robust again collusive selfish behavior.

Recent works [9] and [14] resist collusion in reputation and game theory ways, however, OMH works much more directly and efficiently than them.

5. Performance Evaluation

In this section we talk about the performance of OMH protocol from three aspects: the actual path length, the network throughput, and cryptograph.

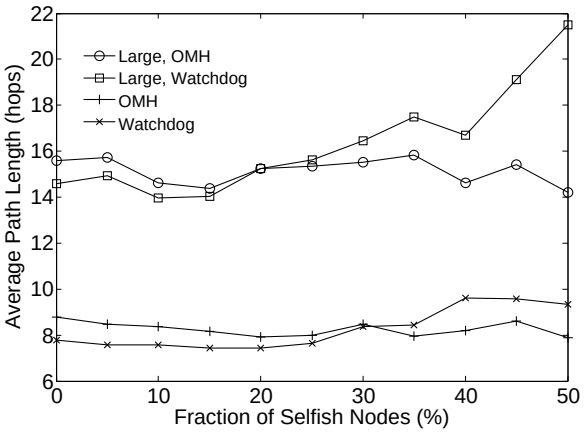


Figure 4: Path Length

5.1. The One More Hop and the Actual Path Length

In OMH, transmissions path is one hop longer than original DSR. As a result, the extra one hop causes built-in extra workload. However, on the other hand OMH makes use of selfish nodes to transmit while other solutions such as Watchdog and Pathrater try to avoid them. As a result, the actual paths in Watchdog and Pathrater can be even longer than in OMH when the fraction of selfish nodes is considerable. Simulation is made to compare the actual path length in DSR with Watchdog and DSR with OMH. In this simulation, average path length is tested in two scenes. In the first scene, 350 nodes are randomly placed in an area of $2500m \times 2500m$, and in the second scene, a large network is used, in which 1400 nodes are placed in an area of $5000m \times 5000m$. In both scenes, 20 pairs of nodes are randomly selected and route discoveries are performed between them. The simulation is repeated for 20 times and the average path length is calculated. The result is shown in Fig. 4. The results show that when the fraction of selfish nodes is low, OMH causes longer transmission because of the one more hop mechanism. However, when the fraction increases, OMH advances Watchdog and Pathrater in shorter path length.

5.2. Network Performance

In this section, at first we analyze the network performance of OMH and its related factors, then show our simulation results.

5.2.1. Theoretical Analysis

Since OMH aims on preventing dropping packets, we focus on the percent of packets receiving (PPR). In a network using DSR protocol, if there are p percent of selfish nodes, the average chance for a packet to be transmitted through a path of n hops is

$$PPR_{DSR} = (1-p)^{n-1} \tag{1}$$

in which $n-1$ means there are $n-1$ intermediate nodes and all of them may be selfish. With OMH implemented, all nodes can not drop packets except the last one, so the probability becomes

$$PPR_{OMH} = 1-p \tag{2}$$

With the improved longer path method, thus the transmission fails only if all neighbors are selfish. So if the destination node has m neighbors, the chance of successful transmission is

$$PPR_{OMHi} = 1-p^m \tag{3}$$

in which $PPROMHi$ denotes the PPR when OMH with improved longer path is used. It is much better than the original PPR_{OMH} when the destination node has some neighbors. However, because of mobility, actual throughput can not be as good as PPR_{OMHi}, because the neighbor nodes are moving and changing and therefore it takes more iterations to find an unselfish neighbor. The actual PPR should be between PPR_{OMH} and PPR_{OMHi} when mobility exists.

5.2.2. Simulation Results

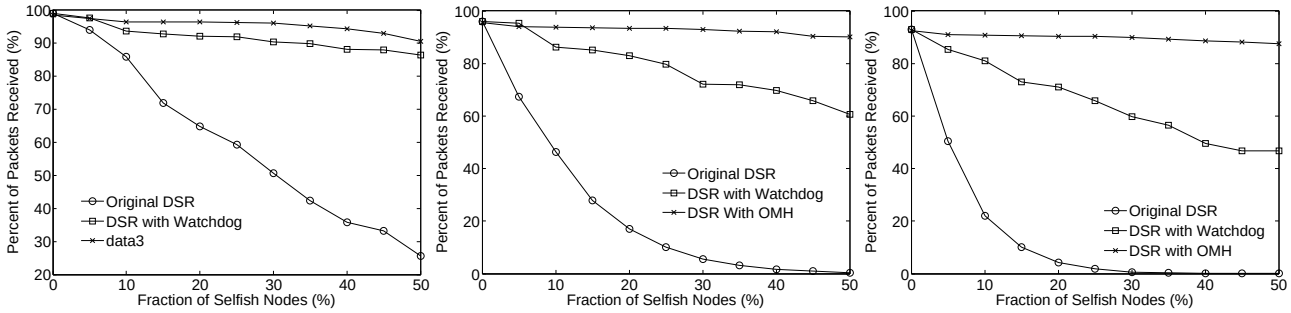
We observe transmissions through 3 hops, 9 hops and 15 hops and the results are shown in Fig. 5.

At first, the longer the path is, the more it suffers from selfish behaviors. When there are 10% selfish nodes, flows through 3 hops have a PPR of 86%, which is still useable; flows through 9 hops have a PPR of 46%, which is very poor; flows through 15 hops have a PPR of 21%, which is almost no useable.

Secondly, both Watchdog and OMH can improve PPR a lot, and OMH works much better. With 50% selfish nodes, when path length is 3, PPR with Watchdog is 86% and PPR with OMH is 91%; when path length is 9, PPR with Watchdog is 60% and PPR with OMH is 90%; when path length is 15, PPR with Watchdog is 47% and PPR with OMH is still 86%. Obviously OMH very well when path is long and percentage of selfish nodes is high. In such critical situation OMH is still very reliable when Watchdog is not suitable. Thats because Watchdog can mark only one node as selfish and avoid it in the future, therefore when there is a lot of mobile intermediate nodes, Watchdog needs many iterations to find a useable path.

6. Conclusion

In this paper we propose the OMH protocol to suppress selfish behavior in ad hoc networks. Our aim is to solve



(a) 3 hops (b) 9 hops (c) 15 hops

Figure 5: Path Length

the problem of selfish behaviors elegantly. Most previous efforts on this topic focus on observation, reputation and token based mechanisms. However they suffer from different weakness. The observation based mechanisms have problem with mobility and collusion; the reputation and token based mechanisms have problem with system complexity and efficiency. We design OMH protocol from a totally new angle. We notice the fact that the selfish nodes are rational and also want to receive and send packets. Therefore if one node can not know whether the packet it received is destined to itself or not, it can not drop the packet. We achieve this target by an intelligent way of routing and encryption, and finally suppress selfish behavior effectively. Simulations are conducted to evaluate OMH from multiple aspects. The result shows that OMH can work well in various network conditions.

References

[1] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 255–265, 2000.

[2] J. Hu and M. Burmester. Lars: a locally aware reputation system for mobile ad hoc networks. In *ACM-SE 44: Proceedings of the 44th annual Southeast regional conference*, pages 119–123, 2006.

[3] S. Buchegger and J.-Y. Le Boudee. Self-policing mobile ad hoc networks by reputation systems. *Communications Magazine, IEEE*, 43(7):101–107, July 2005.

[4] L. Buttyan and J.-P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks. Technical report, Swiss Federal Institute of Technology, 2001.

[5] M. Jakobsson, J. Hubaux, and L. Buttyan. A micropayment scheme encouraging collaboration in multi-hop cellular networks. pages 15–33, 2003.

[6] D. Djenouri and N. Badache. New approach for selfish nodes detection in mobile ad hoc networks. *Security and Privacy for Emerging Areas in Communication Networks, 2005. Workshop of the 1st International Conference on*, pages 288–294, 5-9 Sept. 2005.

[7] B. Yu and B. Xiao. Detecting selective forwarding attacks in wireless sensor networks. *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8 pp.–, 25-29 April 2006.

[8] S. Buchegger and J.-Y. L. Boudec. Performance analysis of the confidant protocol. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 226–236, 2002.

[9] J. J. Jaramillo and R. Srikant. Darwin: distributed and adaptive reputation mechanism for wireless ad-hoc networks. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 87–98, 2007.

[10] H. Tewari and D. O'Mahony. Multiparty micropayments for ad hoc networks. *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 3:2033–2040 vol.3, 16-20 March 2003.

[11] S. Zhong, L. E. Li, Y. G. Liu, and Y. R. Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: an integrated approach using game theoretical and cryptographic techniques. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 117–131, 2005.

[12] E. Huang, J. Crowcroft, and I. Wassell. Rethinking incentives for mobile ad hoc networks. In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 191–196, 2004.

[13] D. Johnson, D. Maltz, and J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*.

[14] S. Zhong and F. Wu. On designing collusion-resistant routing schemes for non-cooperative wireless ad hoc networks. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 278–289, 2007.