Reactive Identification of Misbehavior in Ad Hoc Networks Based on Random Audits

William Kozma Jr. and Loukas Lazos
Department of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona
{wkozma, llazos}@ece.arizona.edu

***Abstract*—We address the problem of identifying misbehaving nodes that (selectively) drop packets, in order to degrade the network performance. Such nodes may agree to forward packets by participating in the route discovery process, but refuse to do so once the packets have been received. We propose a reactive approach where the source initiates an audit process if a significant performance degradation is observed. We employ a compact representation of the behavioral proof of a node by adopting Bloom filter structures and show that the misbehaving node can be identified based on random audits. Our approach provides significant energy savings compared to previously proposed methods that rely on reputation systems, or intensive acknowledgment schemes.**

I. Introduction

In contrast to infrastructure-based networks, ad hoc networks do not rely on dedicated network components for implementing the basic network functionalities. In fact, the network operations are realized in a distributed and cooperative fashion, with network nodes acting as relays, gateways, coordinators, etc. Given that network devices may operate unattended or be in the possession of malicious users, an adversary can easily gain physical access to the hardware or software of a device and reconfigure it in any desirable manner.

Due to the relative ease of compromising devices, nodes are expected to deviate form the normal protocol execution in multiple ways. A compromised node may participate in the routing discovery process, but drop packets in order to conserve energy [1]–[4]. Such a node appears to collaborate in packet forwarding, but only benefits from other nodes. In a more severe type of attack known as the *blackhole attack,* a compromised node intentionally attempts to attract more traffic from neighboring nodes by, for example, advertising short routes to many destinations [5]. Once nearby nodes attempt to route through the blackhole, the malicious node (selectively) drops packets to degrade the overall throughput.

Current methods for detecting misbehaving nodes can be classified to reputation based systems [1], [3], credit based systems [2], or acknowledgment systems [4]. Reputation-based methods rely on the continuous monitoring of the behavior of neighboring nodes (watchdog model) and the association of a reputation index. However, to implement the monitoring module, nodes need to operate in promiscuous mode, thus incuring a very high resource overhead. Credit-based systems can only alleviate selfish behavior by providing incentives for forwarding packets. A potential credit incentive cannot motivate adversaries whose goal is to maximize damage. Finally, acknowledgment-based systems require a large amount of ACK messages to be transmitted, thus incurring a high communication overhead.

Our Contributions: We develop a mechanism for the unique identification of misbehaving nodes that refuse to forward packets to a destination. Our solution does not rely on energy expensive overhearing techniques, or communication intensive acknowledgment schemes. We adopt a reactive approach where a random audit process is initiated only after a performance degradation is observed, thus significantly reducing the associate resource overhead. Based on a storage and communication efficient representation of the packets forwarded by each intermediate node, we construct a publicly verifiable proof of node misbehavior.

II. Problem Statement and Assumptions

Misbehavior identification problem: Consider a path P_{SD} from the source S to the destination D, consisting of nodes $P_{SD} = \{n_1, \ldots n_k\}$ where for simplicity node n_j is "downstream" from node n_i if $i < j$. Let a node $n_m \in P_{SD}$ misbehave by (a) dropping all packets routed through n_m or, (b) selectively dropping packets. We investigate the unique identification of the misbehaving node n_m. In Figure 1, we show a source S routing data to a destination D via a path $P_{SD} = \{n_1 - n_5\}$. Node n_4 drops all packets it receives.

Assumptions: We assume that the source is aware of the intermediate nodes in the path P_{SD} to the destination [8]. We also assume that the destination can notify the source if a significant drop in throughput or delay is observed. Furthermore, we assume that a node can sign its messages using a resource-efficient signature scheme such as the Elliptic Curve Digital Signature Algorithm (ECDSA) that has been shown feasible even for sensors [7]. Finally, we assume that there are at least two independent paths to any destination, i.e., the network is two-connected. This assumption is essential for reaching eny node in the path P_{SD}, through an alternative route that does not contain the misbehaving node(s).

III. Constructing Evidence of Misbehavior

Though a hop-by-hop acknowledgment scheme guarantees the per hop link reliability, it is not sufficient to guarantee

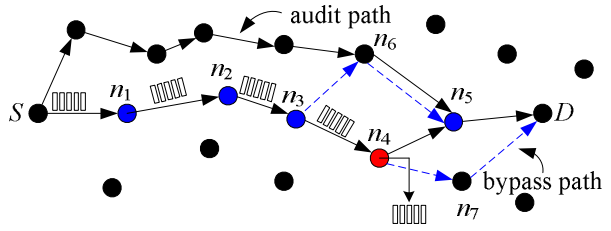

Fig. 1. A path from the source S to the destination D. Node n_4 is misbehaving by dropping packets. Node n_5 is audited by S using an independent path. Once the suspicious nodes are narrowed down to n_4, n_5 the source exclude each node in turn from P_{SD} by using bypass paths.

the reliability of the entire path. A malicious node can verify the receipt of messages by sending an ACK, but refuse to forward the received packets. In case of an abnormal drop in throughput or increased delay, the destination D may notify S for the poor path performance. This reactive approach saves a significant amount of resources since the destination need not acknowledge every packet in an end-to-end fashion. When S is notified by D, it initiates a process of random audits to identify the misbehaving node or the faulty link.

To evaluate the behavior of a node $n_i \in P_{SD}$, the source audits n_i to prove that it faithfully forwards packets to the next hop. Node $n_i \in P_{SD}$ stores a proof of the set of packets it forwards to the next hop. Buffering the packets themselves would require a high amount of storage and a significant overhead for the transmission of the proof back to the source. We note that S need only be able to perform a simple test: "*Did node n_i forward packets $X_i = \{x_1, \ldots, x_N\}$ to the next hop?*" Bloom filters provide a storage-efficient way of performing membership testing for the packets forwarded [6].

A. Membership Testing Using Bloom Filters

A Bloom filter provides a compact representation of membership for a set $X = \{x_1, x_2, \ldots, x_N\}$ of N elements in a vector v of m bits. All m bits of v are initialized to zero. To add a member x_i in the Bloom filter, x_i is passed through q independent hash functions h_ℓ, $1 \leq \ell \leq q$ with each h_ℓ having a range of $\{1, \ldots, m\}$. The corresponding bits $h_\ell(x_i)$, $1 \leq \ell \leq q$ of vector v are set to one. To check if y is a member of X, element y is hashed q times using the hash functions h_ℓ and the corresponding bits are checked against the vector v. If a zero is found at a corresponding location in v, the element y is *not* a member of the set X. Else $y \in X$ with a very high probability. This means that a Bloom filter may yield a false positive, i.e. the filter may indicate that an element y is in X even though it is not. For perfectly random hash functions, the false positive probability p_f can be computed as [9]:

$$p_f = \left(1 - \left(1 - \frac{1}{m}\right)^{qN}\right)^q \approx \left(1 - e^{-\frac{qN}{m}}\right)^q. \quad (1)$$

The number of hash functions q that minimize p_f, is known to be $q = ln2(\frac{m}{N})$, but any choice on q can be made to allow a graceful tradeoff between p_f and q. We can also compute the minimum storage required (size of vector v) so that $p_f \leq \epsilon$, to be equal to $m \geq N\frac{\log_2 \epsilon}{\ln 2}$. For a given storage m, we can compute the size N of the set X that can be represented by the Bloom filter when p_f must be kept under ϵ.

B. Misbehavior Detection Based on Random Audits

Auditing a single node–In the audit process, the source decides to audit a particular node $n_i \in P_{SD}$. Using a path independent from P_{SD}, the source sends an audit request to n_i, indicating the beginning and end time of the audit. An independent path is followed to ensure that the audit is not dropped by the misbehaving node. Fail to reply to an audit is perceived as a misbehavior. Once the audit request is received, node n_i initializes its Bloom filter v_i, and records all packets from S for time period defined in the audit. Then n_i sends the Bloom filter v_i back to the source signed, so that the authenticity and integrity of v_i can be verified. In Figure 1, the source uses a path independent of P_{SD} to audit node n_5.

The source computes the inner product $< v_i, v_S >$, with its own Bloom filter v_S. The inner product is a measure of the similarity between vectors v_i, v_S and can serve as an estimate of the cardinality of the set $X_i \bigcap X_S$, where X_i is the set of packets in the filter v_i and X_S is the set of packets in the filter v_S. The inner product can be approximated by [9]:

$$\begin{aligned} < v_i, v_S > \quad \approx \quad & m\left(1 - \left(1 - \frac{1}{m}\right)^{q|X_i|} - \left(1 - \frac{1}{m}\right)^{q|X_S|}\right. \\ & \left. + \left(1 - \frac{1}{m}\right)^{q(|X_i|+|X_S|-|X_i \bigcap X_S|)}\right). \quad (2) \end{aligned}$$

Given that the vector length m, the cardinalities of sets X_i, X_S and the number of hashes q are known, the source can compute the size of the intersection set,

$$\begin{aligned} |X_i \bigcap X_S| \quad \approx \quad & |X_i| + |X_S| - \frac{\log\left(\frac{< v_i, v_S >}{m}\right.}{} \\ & \frac{\left. + \left(1 - \frac{1}{m}\right)^{q|X_i|} + \left(1 - \frac{1}{m}\right)^{q|X_S|}\right)}{q \log\left(1 - \frac{1}{m}\right)}. \quad (3) \end{aligned}$$

Using $|X_i \bigcap X_S|$ as a metric, the source S can verify if packets in X_S are members of X_i. Furthermore, the source can maintain a sampling of X_S to perform membership tests on v_i for an additional verification of the packets in X_i. The sampling of X_S can be either random or contain packets of higher importance. As an example, if the source is sending MPEG encoded video, a misbehaving node can selectively drop only I-frames to degrade the video quality. The source S can specifically check if the I-frames are members of X_i.

Identifying the misbehaving node–Assume a single compromised node $n_m \in P_{SD}$. Let A denote a one-dimensional array of length k, such that $A[i] = |X_i \bigcap X_S|$. Array A is almost sorted in a descending order, since for any node upstream of the misbehaving one, $|X_i \bigcap X_S| \approx |X_S|$, and for any node downstream from the misbehaving one $|X_i \bigcap X_S| << |X_S|$. Hence, one of the following cases is possible: (a) $A[m-1] \approx |X_S|$, and $A[m] << |X_S|$, in which case n_m faithfully reports its Bloom filter or, (b) $A[m] \approx |X_S|$, and $A[m+1] << |X_S|$, in which case n_m lies about its Bloom filter. In both cases,

we can limit the set of suspicious nodes to only two one-hop neighbors; n_{m-1}, n_m or n_m, n_{m+1}. To further identify n_m, the source can request the slight modification of the path P_{SD} to exclude the two nodes in turn, and monitor the path performance. In Figure 1, half of the traffic bypasses node n_4 via node n_6 while the remaining traffic bypasses node n_5 via node n_7.

Note that a node may misbehave in such a way that $|X_i \bigcap X_S| \approx |X_S|$. Although such node will escape the misbehavior detection mechanism, this attack has minimal effect on the path performance. Also note that in the case of collisions, a benign node will still have a filter v_i similar to v_S if a retransmission mechanism is in place. For protocols without a retransmission mechanism, the adversary still needs to drop packets at a much higher rate than the normal collision rate to affect the path performance. Hence, its Bloom filter will be drastically different than the filters of benign nodes.

The task of identifying the misbehaving node is equivalent to identifying the transition from $A[i] \approx |X_S|$, to $A[i] << |X_S|$. We can identify such a transition via *binary search* using comparison questions, thus requiring only $O(\log_2(N))$ audits. However, a deterministic audit algorithm reveals the sequence of nodes that will be audited given a path P_{SD}. Hence, the adversary can target particular nodes, and stop its misbehavior for certain periods of time in order to avoid detection.

To alleviate the shortcomings of a deterministic audit sequence, we employ a random audit process as follows. Initially, all nodes $M = \{n_1, \ldots n_k\}$ belong to the suspicious set of nodes. The source audits any node $n_i \in M$ at random. If $A[i] \approx |X_S|$, the set of suspicious nodes is reduced to $M = \{n_i \ldots n_k\}$, else $M = \{n_1 \ldots n_i\}$. Note that the audited node always remains in the set of suspicious nodes, since it can always lie about its Bloom filter. The audit process is repeated until the set of suspicious nodes M contains only two nodes. The misbehaving node is further identified by in turn exclusion of each suspicious node from P_{SD}. The pseudocode for the random audit algorithm is given in Figure 2.

A node can try to avoid being accused of misbehavior by arbitrarily constructing its own Bloom filter. This can be achieved, for example, if all bits of the Bloom filter are set to one. In such a case, $|X_i \bigcap X_S| \approx |X_S|$ since $X_S \subseteq X_i$ and any membership test would come out positive. Note, however, that a source can easily verify if the Bloom filter X_i contains a packet not in X_S and hence, X_i is modified. The source S can pick any $x \notin X_S$ and test if it is a member of X_i. If the membership test is positive, the source can assume that $x \in X_i$ with a probability $(1 - p_f)$. The probability of false positive can be further reduced by repeating the experiment t number of times, yielding a successful identification of Bloom filter manipulation with a probability $1 - (p_f)^t$. Note that each Bloom filter is signed and hence, acts as a public commitment to the packets forwarded by each corresponding node. Misbehavior can be be publicly verified via the evidence provided from the source.

Detecting misbehavior of multiple nodes–It is possible that more than one nodes are compromised on the path P_{SD}. In such a case our audit process can identify one misbehaving node at a time. The problem of identifying a compromised node can be modeled after a noisy binary search. In the noisy binary search, a misbehaving node can be identified as long as a limited number of audited nodes lie about their Bloom filter. The mapping of the problem of misbehavior detection by multiple nodes to the problem of noisy binary search is not presented here, due to space limitations.

Algorithm Random Audit Algorithm

```
1:  Initialize: M_ℓ ← n_1, M_r ← n_k, M = {M_ℓ, ..., M_r}
2:  while |M| > 2 do
3:      audit n_i = M[rand]
4:      if |X_i ⋂ X_S| ≈ |X_S|
5:          M_ℓ ← n_i
6:      else M_r ← n_i
7:  return M
```

Fig. 2. The pseudocode of the random audit algorithm.

IV. Conclusion

We addressed the problem of identification of misbehaving nodes refusing to forward packets to a destination. We proposed a reactive identification mechanism that does not rely on continuous overhearing or intensive acknowledgment techniques, but is only activated in the event of performance degradation. Our solution employs Bloom filters to compactly represent the set of packets forwarded by an intermediate node, thus significantly reducing the storage required to construct a publicly verifiable proof of node behavior. Future work includes identification of other types of misbehavior such as time delay, and addressing node collusion.

References

[1] S. Buchegger, and J-Y. Le Boudec, *Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes, Fairness in Dynamic Ad-Hoc Networks,* In Proc. of MobiHoc'02, 2002.

[2] L. Buttyan, and J.-P. Hubaux, *Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks,* ACM/Kluwer Mobile Networks and Applications, vol. 8, no. 5, 2003.

[3] S. Marti, T. Giuli, K. Lai, and M. Baker, *Mitigating Routing Misbehavior in Mobile Ad Hoc Networks,* In Proc. of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom'00), 2000.

[4] K. Liu, *An Acknowledgment-Based Approach for the Detection of Routing Misbehavior in MANETs,* IEEE Transactions on Mobile Computing vol. 6, no. 5 536–550, 2006.

[5] C. Karlof, and D. Wagner, *Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures,* Elsevier's Ad Hoc Networks Journal, Special Issue on Sensor Network Applications and Protocols, vol. 1 no. 2–3, pp. 293–315, 2003.

[6] B. Bloom, *Space/Time Tradeoffs in Hash Coding with Allowable Errors,* Communications of the ACM vol. 13 no. 7 pp. 422-426, 1970.

[7] A. Liu, and P. Ning, *TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks,* to appear in Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track, 2008.

[8] D. Johnson, D. Maltz, and Y-C. Hu, *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR),* draft-ietf-manet-dsr-09.txt.

[9] A. Broder, and M. Mitzenmacher, *Network Applications of Bloom Filters: A Survey,* Internet Mathematics, vol. 1, no. 4, pp. 485–509, 2004.