Hop-by-Hop Traceback in Wireless Sensor Networks

Muhammad Shoaib Siddiqui, *Member, IEEE,* Syed Obaid Amin, and Choong Seon Hong, *Member, IEEE*

***Abstract*—We propose two advancements in the existing Bloom filter based traceback schemes for WSNs (Wireless Sensor Networks): the support of directed queries, and a way of reducing the false positive rate at the nodes nearer to the sink. Simulation results show that the proposed mechanism can efficiently traceback the packets with very few false positives.**

***Index Terms*—Traceback, WSN, sensor networks.**

I. Introduction

TRACEBACK in communication networks, is a promising solution to counter the spoofed attacks by determining the probable source of the malicious packet(s) or the attack path(s). Traceback schemes are very well addressed in the literature; however, most of the schemes are proposed for IP networks and very limited work can be seen in the field of WSN (Wireless Sensor Network) traceback. The traceback schemes proposed for IP networks are not straight forwardly applicable to WSN, as traditional IP traceback solutions require fair amount of resources to be implemented; however, private ownership of WSNs makes it relatively easier to add a traceback support as compared to traditional IP networks.

Traceback schemes can be divided into three categories: *messaging*, *packet marking*, and *logging*. Messaging has communication overhead due to extra message passing and packet marking requires extra bits in packet header, which increases the size of packets. As communication is the most costly function in a sensor node, our work falls under the category of logging based traceback; in which packet relaying nodes store the information of the forwarded packets in a suitable data structure. In case of an attack, the victims consult upstream nodes to reconstruct attack paths by broadcasting the information of the malicious packet(s) in the traceback request.

Very limited work has been done on logging based traceback schemes for WSNs. We can see two schemes namely CAPTRA [1] and SNTS [2] in this regard. Both of these schemes use Bloom filters [3] to store the traffic logs. A Bloom filter is a randomized data structure, which is represented in a form of a bit vector. It is used for membership queries, whether an entity (data packet in traceback) in question is a part of the dataset or not; however, with controllable false positives. In its basic form, a Bloom filter can only be used for membership queries. It does not have any information about the entity who added the element in question; therefore, a traceback request is usually broadcasted to the neighboring nodes. Consequently, due to the false positives of Bloom filter, the chances of generating false query also increase. To eliminate this shortcoming, we propose two advancements in current Bloom filter based traceback schemes for WSNs. First of all, we add the support of directed queries, which significantly reduces the number of messages generated by the traceback process. Secondly, we provide a mechanism of reducing the false positive rate at the nodes nearer to the sink by using multiple IDBFs. The proposed scheme can efficiently trace back to the single attacker in a hop-by-hop fashion.

Manuscript received November 7, 2011. The associate editor coordinating the review of this letter and approving it for publication was F. Granelli.

This research was supported by the MKE, Korea, under the ITRC support program supervised by the NIPA (NIPA-2011-(C1090-1121-0003)).

M. S. Siddiqui and C. S. Hong are with the Dept. of Computer Engineering, College of Electronics and Information, Kyung Hee University, Korea (e-mail: shoaib@networking.khu.ac.kr, cshong@khu.ac.kr). Dr. C. S. Hong is the corresponding author.

S. O. Amin is with the Dept. of Computer Science, Yale University, New Haven, Connecticut, USA (e-mail: obaid.amin@yale.edu).

Digital Object Identifier 10.1109/LCOMM.2011.121311.112265

II. Proposal

A. Directed Traceback Queries

For our traceback scheme, we extend the basic Bloom filter design, for identifying the entity which added the given element with some false positives. Our design is inspired by Tagged Bloom filters given in [4]; however, the definition of tags, their purpose, and way of usage are different. Due to its nature, we call our scheme IDBF (ID based Bloom Filters).

In IDBF, firstly we maintain a table called ID Table (IDT) at every traceback enabled node. IDT stores the addresses of the neighboring nodes and assigns them $i-$bits long IDs. In this way, a big source address (for example, 64 or 16 bit address in IEEE 802.15.4) of neighboring nodes can be represented by only a few bits. The value of i is calculated as: $i = \lceil \log_2 (l_{IDT} + 1) \rceil$; where, l_{IDT} is the maximum length of the IDT set at the deployment phase. As the value of l_{IDT} actually depends upon the maximum number of neighbors that a node can have, we believe that $l_{IDT} = 7$ is a reasonable upper bound for many networks [6]; however, $l_{IDT} = 15$ may be used for densely deployed networks.

Secondly, instead of taking a single bit array of length m, we take a two dimensional array of $m \times i$ dimensions so that the ID assigned by the IDT can be stored. For an incoming packet, if there is no entry of the immediate forwarder's address in the IDT, a new ID to that forwarder is assigned. Otherwise, the ID already assigned by the IDT would be used. The payload is then passed from k hash functions, which results into k indexes, similar to a normal Bloom filter operation. Instead of setting only one bit, the ID assigned by the ID table is inserted into those k indexes. Effectively, the payload of the packet helps in deciding the location of a given payload in IDBF; whereas, the source of the packet helps in deciding that what should be written on the given location.

Membership queries can be carried out by passing the payload to the k hash functions and checking the corresponding

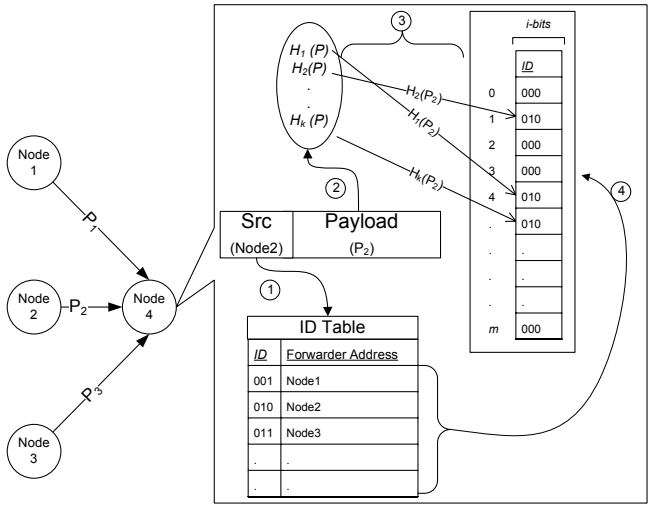

Fig. 1. Logging phase of IDBF.

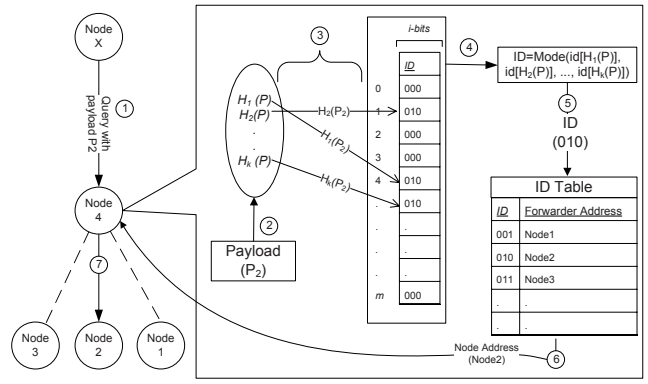

Fig. 2. Query phase of IDBF.

positions of the $m \times i$ array. If any of the cell positions is found to be zero then the payload in question had not been stored. If all of the values are nonzero then we can identify the node which forwarded the packet in question by: $\overrightarrow{ID} = Mode(id[H_1(P)], id[H_2(P)], ..., id[H_k(P)])$; where $id[H(P)]$ returns the ID at location $H(P)$ and $\overrightarrow{ID}$ is a vector of IDs pointing to the corresponding addresses of the neighbors in the IDT, while the *Mode* function returns the most occurring value. The traceback query is then forwarded to only those nodes which are returned by the IDT.

Figure 1 and 2 depict the logging and query phase of the scheme, respectively. In Figure 1, Node 2 sends data as a payload P_2 to Node 4. The source of the packet is first sent to the IDT of the Node 4, which assigns 010 as an ID. After that the payload P_2 is passed from k hash functions, and corresponding cell positions are set to 010. Figure 2 shows the traceback phase when Node X sends the query with payload P_2. The payload in the query is passed from k hash functions and corresponding cell positions are checked. In the given example all three cell positions will return 010; therefore, 010 is passed to the IDT, which as a result returns Node 2 as a possible forwarder of the packet.

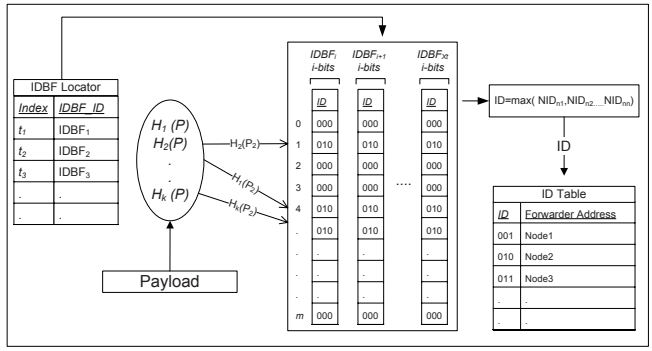

Fig. 3. Complete architecture of a traceback system with multiple IDBFs.

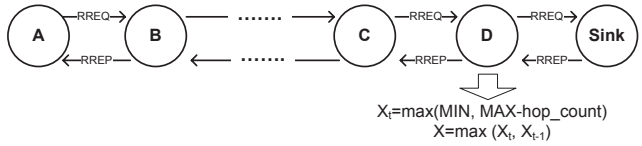

Fig. 4. Calculation of X_t with the hop-count value.

B. Multiple IDBFs (MIDBF)

In sensor networks, the traffic usually flows from the sensor nodes to the sink; therefore, nodes nearer to the sink observe higher traffic load than others. Intuitively, if nodes nearer to the sink have larger filter size as compared to the other nodes, the false positive rate can be reduced. Motivated by this observation, we further improve our scheme by allowing variable size IDBFs at different nodes. For dynamically adjusting the IDBF size, we first need to know the distance of a certain node from the sink in terms of hops. Secondly, we need to select the appropriate parameters among the length of IDBF (m), number of packets inserted (n), and the number of hash functions (k), which can be scaled according to the network size. For the distance issue, we utilize the hop-count value provided by the routing protocols. Any routing protocol which deals with the hop-count value can support $MIDBF$ scheme. For example, in AODV routing protocol, the RREP message is used to send back a route reply. It contains the hop-count value that reflects the distance of a node from the sink. We use hop-count available in RREP to estimate the IDBF size. If the routing protocol cannot provide the hop-count value then the sink sends a message with hop-count value 1. When a sensor node receives this message, it collects the hop-count value for itself, update the value by incrementing it and forwards the message to the upstream nodes.

The next question is that according to the position of a node which parameters of IDBF can be scaled up or down. In fact we are trying to increase the capacity of the IDBF (n) that is proportional to the length of the IDBF (m). The dynamic adjustment of the value of m also requires the set of hash functions with higher range so that the input entities can be mapped to a larger data set. Replacing the set of hash functions is complex and not a feasible solution. Even if we can replace the set of hash functions, the values already stored in the IDBF cannot be verified in the query phase as the original set of hash functions used to insert those values has

been replaced by the new set of hash functions. This is why, adjusting the size of m or k is not a feasible solution. To work around this problem, instead of altering the IDBF size we take multiple IDBFs, as shown in Figure 3. The number of IDBF is denoted by X_t. Figure 4 depicts that how hop-count value can be used for knowing the value of X_t. A brief algorithm is given as follows: (1) Node A sends an RREQ message to obtain the routing information of the sink. (2) The sink in response, replies to this RREQ with RREP message. (3) The RREP message is captured and processed by the intermediate node. (4) The intermediate node calculates the value of X_t, which is given by: $X_t = max(MAX - hop\_count, X_{t-1})$, where $X_o = MIN$.(5) Finally, the value of X_t is used for creating multiple IDBF.

Above mentioned algorithm ensures that the nodes nearer to the sink would have multiple IDBF ranges from $IDBF_1$ to $IDBF_{X_t}$. Once an IDBF, say $IDBF_i$ reaches to its optimal capacity it is archived and $IDBF_{i+1}$ takes the place of $IDBF_i$ (by optimal capacity we mean that the *fpr (false positive rate)* of IDBF is less than 0.05). The values MIN and MAX are used to avoid any forgery of hop-count value in RREP messages. For example, any colluding node between the sender and the receiver can modify the hop-count value, which may result into too many IDBFs or no IDBF at all. Therefore, in the form of MIN and MAX, upper and lower bounds (of hop-count) are provided which can be set according to the network requirements. To ensure that we are storing only unique packets, a simple lookup in archived IDBFs would be required as well. When all of the IDBFs are optimally filled, the first IDBF is set to zero again.

C. Hash Functions

Hash functions involved in IDBF are required to be lightweight as we do not require cryptographic hardness properties for hash functions. Therefore, simple integer hash functions can be used for this purpose. It is also required that the set of hash functions, S, at one relay node should not be same as the set of hash function at its neighboring relaying node, S'; or in other words $Pr[S = S']$ is very low. This way, the probability of false positives at one relaying node is ensured to be independent of the false positives at another relaying node and any traceback request generated due to false positives will die down as it progresses further.

III. Evaluation Results

For evaluation, we consider a static WSN in which a single or a group of adversaries can attack the sensor network, such as DoS attack etc. The adversaries can collude by spoofing or lying about the source address. In the case of spoofing or lying, the complete attack-path may not be found but attacker's one-hop region can be discovered. An adversary may also try to fill up the Bloom filters with false data, so that any information of the malicious packet can be overwritten. However, doing so in our scheme would eventually lead us to the attacker, because benign nodes do not flood the networks with unnecessary messages and we are taking $mode$ in the membership query phase.

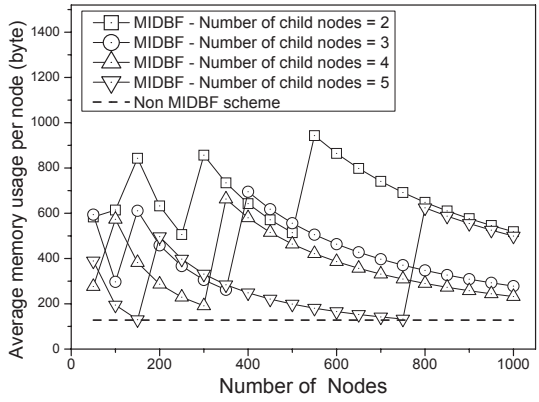

Fig. 5. Average memory required per node by the $MIDBF$ and the non-$MIDBF$ schmes as the number of nodes increases in the network.

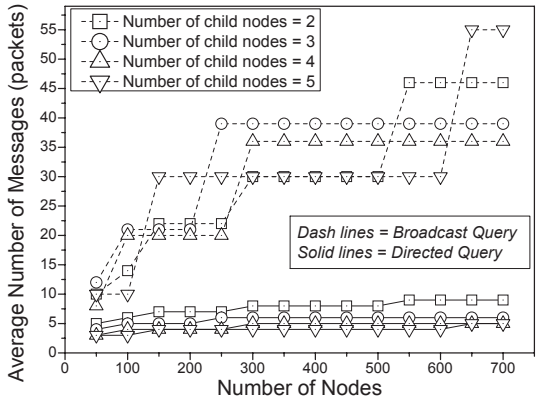

Fig. 6. Number of messages required by the boradcast and the directed query schemes, with respect to different network topologies, as the number of nodes increases in the network.

Similar to Bloom filters the performance of IDBF mainly depends upon the values of m, n, and k. We start our performance evaluation from the analysis given in [5]. According to [5] the *fpr (false positive rate)* of a Bloom filter can be given by $\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{\frac{-kn}{m}}\right)^k$. This analysis helped us to find out that when an $IDBF_t$ should be replaced by $IDBF_{t+1}$. For example, with $m = 1024$ and $k = 4$ we can store approximately 164 unique packets in an IDBF for $fpr = 0.05$. Therefore, after inserting 164 unique packets $IDBF_t$ would be replaced by $IDBF_{t+1}$.

The use of multiple IDBF would require more memory at the sensor nodes, as shown in Figure 5. This figure shows the memory requirement for both traditional and MIDBF traceback schemes. Each solid curve shows the memory requirement for a specific type of tree; for example, with a binary tree (number of child nodes = 2), the scheme uses more memory as compared to the tree with higher number of child nodes, as it would increase the number of hops from the leaf node to the sink. The dotted line shows the memory requirement with the traditional scheme. The memory in this case is static and set to 128 bytes (1024 bits) only; therefore, produces high false positive as compared to the MIDBF (as shown in Figure 8.

Figure 6 shows the number of messages required to find the correct path in the query phase. For broadcast query (dotted

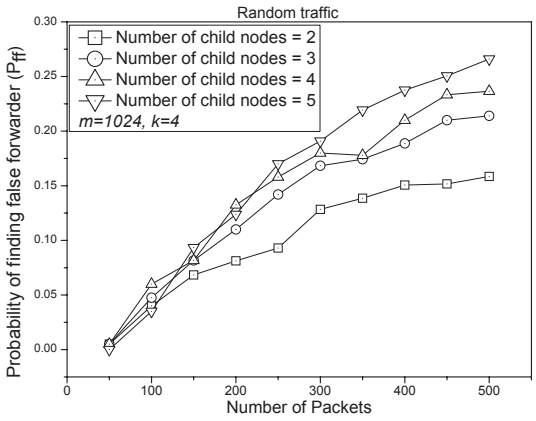

Fig. 7. Probability of finding the false forwarder as the number of packets, inserted in the Bloom filters, increases.

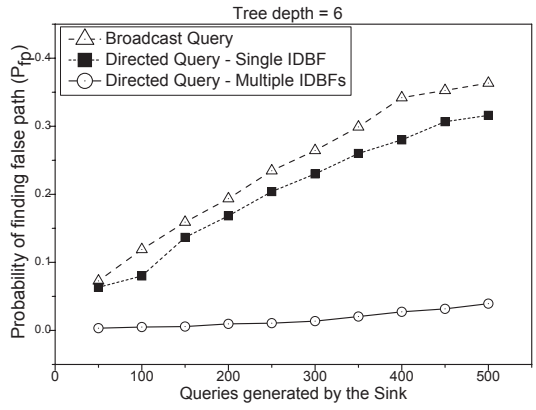

Fig. 8. Probability of finding a false path as the queries generated by the victims increases for simple boradcast query mechanism, directed query with single IDBF and directed query with multiple IDBFs schemes.

lines in Figure 6), the values are higher as compared to the directed query mechanism (solid lines in Figure 6). This is because of the fact that in a broadcast scheme, the possible path found can be more than one.

Figure 7 shows the P_{ff} of a packet as the number of packets inserted in an IDBF increases for different number of child nodes (N) with random packet generation. We can see that the number of child nodes slightly affects the P_{ff}, because P_{ff} mainly depends upon the number of packets inserted by a specific child node. Figure 8 shows the P_{fp} of the traceback schemes based on broadcast queries and directed queries for single and multiple IDBFs. These results are achieved by the set of simulations conducted on a simulator made by our lab in Java programming language. A tree of sensing nodes was constructed. The average number of child nodes in the tree was variable and only leaf nodes were allowed to randomly generate packets. As expected, in terms of P_{fp}, traceback scheme based on multiple IDBFs outperforms the traditional broadcast based traceback scheme.

IV. Conclusion

In this letter, we propose two advancements in the existing Bloom filter based traceback schemes for WSNs. Firstly, we add the support of directed queries. Secondly, we provide a way of reducing the false positive rate at the nodes nearer to the sink. Graphs in Figure 5-8 show that the proposed scheme reduces the number of traceback messages and the probability of finding a false path as compared to the traditional broadcast based traceback scheme; however, at the cost of higher memory usage. Furthermore, we can also note that with $n = 164$ ($fpr = 0.05$) both P_{ff} and P_{fp} are almost zero.

References

[1] D. Sy and L. Bao, "Captra: coordinated packet traceback," in *Proc. 2006 International Conference on Information Processing in Sensor Networks*, pp. 152–159.

[2] D. Smith, R. Mahon, S. Koundinya, and S. Panicker, "Project SNTS: sensor node traceback scheme," in *2004 ACM Workshop on Wireless Security*.

[3] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, pp. 422–426, 1970.

[4] Y.-a. Huang and W. Lee, "Hotspot-based traceback for mobile ad hoc networks," in *Proc. 2005 ACM Workshop on Wireless Security*, pp. 43–54.

[5] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: a survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[6] C. Zhang, Y. Zhang, and Y. Fang, "A coverage inference protocol for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 6, pp. 850–864, 2010.