A Probabilistic Framework for Distributed Localization of Attackers in MANETs

Alessandra De Benedictis[2], Behzad Koosha[1], Massimiliano Albanese[1], and Valentina Casola[2]

[1] Center for Secure Information Systems
George Mason University, Fairfax, VA, USA
{malbanes,bkoosha}@gmu.edu
[2] Department of Electrical Engineering and Information Technology
University of Naples "Federico II", Naples, Italy
{alessandra.debenedictis,casolav}@unina.it

Abstract. Mobile Ad-hoc Networks (MANETs) are frequently exposed to a wide range of cyber threats due to their unique characteristics. The lack of a centralized monitoring and management entity and the dynamic nature of their topology pose new and interesting challenges for the design of security mechanisms for MANETs. While conventional methods primarily focus on detecting attacks, in this work we focus on estimating the attackers' physical location in the network, and propose a probabilistic framework for aggregating information gathered from nodes reporting malicious activities in their vicinity. In order to be consistent with the decentralized nature of MANETs, we present a distributed approach to attacker localization based on dynamically partitioning the network into clusters. These self-organized clusters can (i) independently find the approximate location of the attackers in real time, and (ii) deploy trusted resources to capture attackers. We show through experiments in a simulated environment that our approach is effective and efficient.

Keywords: Attacker Localization, Mobile Ad-hoc Network, Distributed Framework, Cluster

1 Introduction

Mobile Ad-hoc Networks (MANETs) consist of mobile nodes able to transmit and receive data without the need for a fixed infrastructure. Due to their flexibility, they have been widely adopted in a variety of applications such as military battlefield monitoring and control, civilian sensor networks, file sharing and humanitarian disaster relief operations, etc.

MANETs are characterized by several features that uniquely differentiate them from wireless devices, such as the lack of a centralized management, the absence of rigid boundaries, power constraints, bandwidth limitations, dynamic topology, scalability and cooperativeness. These features expose MANETs to a wide range of cyber threats and pose new challenges for the design of security

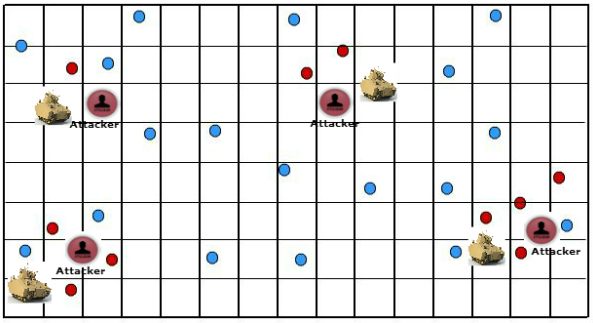


Fig. 1: a MANET scenario where attackers are captured by trusted resources.

mechanisms. As Fig.1 illustrates, network nodes could be threatened by several attackers physically located within their transmission range. It would be desirable for the nodes to cooperate locally and locate the attacker as soon as possible. Moreover, once localized, the attackers could be physically captured by dispatching trusted resources so that they cannot cause further harm.

Extensive research efforts have been dedicated to the issue of detecting various types of attacks, while the problem of physically localizing attackers with respect to specific types of attacks has been marginally studied. Existing approaches are mostly based on measuring and processing parameters related to node communication, such as connections with neighboring nodes, time of arrival (TOA), angle of arrival (AOA) and received signal strength (RSS), but they could be easily manipulated by attackers, thus reducing the effectiveness of such solutions.

In our previous work [1], we proposed a more general solution to the problem of attackers' localization, based on the processing of alerts generated in the network on a probabilistic basis. Estimation of the attacker's location was based on information collected from the nodes raising alerts, assuming that malicious nodes reside in the vicinity of those nodes. This assumption is reasonable and has been adopted by several intrusion detection systems, such as [7],[5].

The main issue with our previous solution is the way alerts are processed, as we assumed the existence of a centralized authority able to gather information about all alerts generated in the network and deploy proper resources to capture localized attackers. In this paper, we aim at overcoming such limitation and propose a distributed localization framework, in which information about raised alerts is processed locally within dynamically established clusters of nodes. Moreover, unlike its centralized version, the proposed distributed framework aggregates and processes alerts as soon as they emerge. This is vital to cease any network malfunction during further communication between nodes.

The reminder of this paper is organized as follows. Section 2 discusses related work about attacker localization and clustering strategies in mobile networks. Section 3 clearly states the goals of this paper with respect to the state of the

art. In Section 4, we present a detailed description of the framework. Section 5 provides our clustering scheme adopted to run the localization algorithm in a distributed fashion. Experimental results are presented in Section 6, and finally Section 7 presents some concluding remarks.

2 Related Work

Considerable research has been recently devoted to the problem of detecting various types of attacks against wireless networks, and there is an increasing interest in attacker localization, in both wireless sensor networks and ad-hoc mobile networks.

The work presented by Zeng et al. [9] discusses and categorizes current solutions in both secure localization and location verification fields for wireless sensor networks. Indeed, in case of deployment in hostile environments, the attackers might interfere with the localization process, so as to produce wrong estimated locations. In addition, since sensor nodes might be compromised, the base station may not rely on the locations disclosed by sensor nodes.

In this regard, the work by Zhan and Li [10] tackles the problem of locating a static malicious source that deliberately conceals or forges its position with the help of a directional antenna in sensor networks. The principal idea is to use coordination of multiple sensors to locate the adversary and optimize the process with a finite horizon discrete Markov decision process. In this work, a localization mechanism called Active Cross Layer Location Identification (ACLI) for sensor networks has been devised. Unlike existing localization schemes, this mechanism is not influenced by an attacker that falsifies its location by methods such as smart antennas and software defined radio equipments.

Yang et al. [8] proposed the use of spatial information to localize various adversaries performing a spoofing attack. They analyzed the spatial correlation of received signal strengths of the wireless nodes. The received signal strength-based spatial correlation, as a physical property related to every wireless node, is difficult to forge, and is independent of cryptography schemes. The proposed approach can detect the existence of attacks in addition to determining the number of adversaries.

Liu et al. [6] addressed the problem of localizing multiple jammers in wireless networks, by analyzing the network topology changes caused by jamming activities. The proposed framework groups the network nodes into clusters, and estimates the positions of jammers by analyzing situations where jamming areas have common intersections.

Most of the existing approaches that provide ad-hoc solutions to the problem of localizing attackers depend on the specific nature of attacks. Seeking for the definition of a more general approach to attacker localization in MANETs, we proposed in [1] a framework based on a probabilistic model of the attacker's location, and presented two polynomial time heuristic algorithms to estimate the position of attackers in a MANET on a probabilistic basis. The proposed framework relies on the capability of nodes to detect a malicious activity in

their vicinity. This can be accomplished by having proper IDSs running on each node. The localization algorithm is run by a centralized authority, that collects information about alerts that have been raised throughout the network during an observation interval.

The main goal of the localization algorithm is to find the minimum set of candidate locations that could explain, if containing attackers, all the alerts generated in the network. After modeling the field of observation as a grid, we assigned a probability value $Pr(attacker(p))$ to each point p of the grid, representing the probability that p is an attacker, based on its proximity to alerts. $Pr(attacker(p))$ is obtained as a combination (see [1]) of all the values $Pr(causes(p, a))$, representing the probabilities that p has caused for each alert a, that has been raised.

Unlike cellular networks where the nodes (users) can gather information about other nodes via a control unit (base station), the ad-hoc network lacks the mentioned feature due to its infrastructure-less architecture. It is observed that communications in cellular networks are mainly executed point to point, while, in ad-hoc networks, the communications are mostly between groups which are likely to harmonize their movements in the network. In order to increase the routes life cycle and reduce the routing control overhead, clustering of the nodes into groups is considered.

Using the concept of clusters in an ad-hoc network has several benefits. Current routing protocols can be immediately applied to the clusters. Also communication overhead can be reduced by lowering the sum of control and signalling data needed to achieve a consistent transmission of data in the network. This will have a substantial effect in reducing routing overhead particularly in large dense networks where finding a solution to the scalability problem is of great importance.

Several research works have been proposed to form clusters and elect clusterheads in ad-hoc networks. In the *Lowest ID cluster algorithm* (LIC) [3] every node is assigned a unique *id*. At regular intervals, each node broadcasts a list of nodes that it can hear in its vicinity. The node with the minimum *id* is selected as a clusterhead. The downside to this algorithm is the fact that some selected nodes are likely to operate as clusterheads for a longer interval and this causes them to loose their battery power.

In the *Highest Connectivity Clustering algorithm* (HCC) [2], the selection of the clusterheads is executed such that the node having the highest number of neighbors (maximum degree) in its transmission range is elected as a clusterhead. This network is very stable in terms of clusterhead change despite the low throughput.

The main idea in *Weighted Clustering algorithm* (WCA) [4], is to assign a weight to each node in the network according to its mobility, degree of freedom, cumulative time of acting and the remaining battery level. In our framework, we adopted an enhanced version of such algorithm, that we will present in section 5.

Although our previous solution is able to obtain good results in practice, it raises some drawbacks that we aim to overcome in this paper: first, it relies

upon the existence of a centralized authority, that is usually not practical in a MANET due to the lack of a fixed infrastructure and its typical self-organizing nature. Second, it only processes alerts at the end of an observation interval instead of trying to identify and capture attackers as soon as possible.

In the following, we introduce a distributed version of the probabilistic localization framework, able to overcome the discussed issues.

3 Problem Statement

In this paper, we propose a framework for attackers' localization, based on a completely distributed localization algorithm, directly deriving from the centralized one presented in our previous work [1]. In this revised version of the localization framework, we eliminate the need for a centralized authority and distribute the localization process among nodes. Moreover, we perform an early processing of alerts, in such a way that countermeasures can be taken to reduce harm for the system. The main contributions of this paper are the following:

1. we introduce a distributed strategy to smartly process alerts as they emerge,
2. we define a protocol to exchange messages related to the localization strategy, defining specific message types.
3. we evaluate the performance of the distributed framework compared to the centralized version.

Before going into details about our proposal, we present some definitions that will be used throughout this paper.

Definition 1 (Neighbors). *Two nodes i and j are considered neighbors if they are in the transmission range of each other. If considering a free space propagation model, it means that $\mathcal{D}$ $(i,j) \leq r$, where $\mathcal{D}$ denotes the Euclidean distance and r is the transmission range.*

Definition 2 (k-Neighbors). *Two nodes i and j are considered k-neighbors if there exists a path between them of at most k hops.*

Definition 3 (Cluster). *Given a space $\mathcal{S}$, and a node $n \in \mathcal{S}$, a cluster $C \in \mathcal{S}$ is composed of all nodes $i \in \mathcal{S}$ such that i and n are k-neighbors. The k parameter will be referred as "cluster depth" later on in the experimental results section.*

Definition 4 (Clusterheads). *Given a cluster $C \in \mathcal{S}$ composed of N nodes, the clusterhead is a node $CH \in C$ such that it maximizes a quality function f (e.g. battery level, degree, etc.). The clusterhead functions as a local coordinator and supervises the cluster's overall activity.*

Definition 5 (Compatible Alerts). *Alerts that are potentially raised by the same attacker are said to be compatible. Two alerts a_1 and a_2 are compatible if $\mathcal{D}(a_1,a_2) \leq 2 \cdot r$.*

The general idea behind the proposed framework is the following: when a node detects an attack and raises an alert, it starts a clustering procedure that involves its k-neighbors to elect a clusterhead based on significant parameters such as the current battery level, computational power, average speed or the number of neighbors. The clusterhead collects information about the new generated alert, along with information about other possible alerts known by nodes belonging to the cluster, and runs the localization algorithm.

In this approach, the process of deploying resources to capture attackers, introduced in [1] can still be applied, assuming that nodes can communicate to a headquarter to signal the high risk locations. Even in this case, to increase precision, deployment should be performed iteratively, by checking the suspected areas and updating, step by step, the attackers' probability distribution. As new alerts could be raised later on, caused by attackers localized in areas that have been checked in a previous run of the algorithm, the cleaning operation is limited within a certain run.

In the next section, a detailed description of the localization framework is presented, followed by a graphical overview of its behavior. In addition, we illustrate the clustering strategy adopted by our framework for the actual execution of the localization algorithm.

4 Distributed Framework For Attackers' Localization

The core of our approach is the strategy adopted to group and process alerts as soon as possible, to perform an early localization of attackers, in order to stop them before they can cause further problems in the network. The adopted strategy is characterized by:

- the events that activate the localization process,
- the information used by the localization process, and
- the actual executor of the localization algorithm.

The precision of the localization algorithm proposed in [1] depends on the distribution of alerts: the closer the alerts to process, the more the suspected areas related to different alerts will overlap; identifying high risk areas where the attacker probability is higher than elsewhere. These regions are candidates to contain attackers, and can be checked and possibly cleaned by proper deployed resources. This suggests that the localization algorithm should be run on a significant subset of alerts, while on the other side trying to process them as soon as possible. Therefore, the algorithm should not be run for each alert that is raised, but after when the information about a group of compatible alerts has been acquired, in order to obtain more precise results.

These considerations led us to define a trigger for the localization process, depending on the availability of other alerts previously raised in the same area. In this way, the set of candidate locations that could contain attackers, resulting from the localization algorithm, can be reduced. At the detection of an attack, the victim node advertises the local alert to its neighbors, that in turn sends

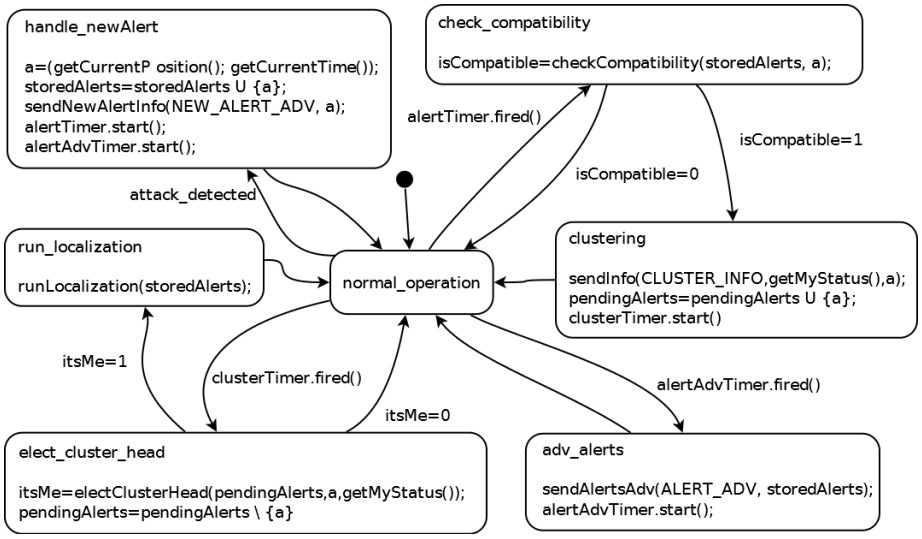


Fig. 2: FSM representation of the behavior of a node at the detection of an attack

information about stored alerts. If there exist two or more *compatible* alerts, the node will launch the localization process on this set of alerts.

In order to maximize the dissemination of information about alerts that are physically localized in different areas, nodes periodically broadcast information about known alerts while moving. Such information is stored by their neighbors and used later when needed. To limit the overhead, alert information is not disseminated over the whole network, but only sent to a subset of nodes that physically reside in the vicinity of the attacked node.

As stated, the localization process involves not only the node raising the alert, but also all the nodes residing in its vicinity at that moment, forming a cluster. The execution of the localization algorithm requires a certain computational capability and could influence the node's normal operation. For this reason, we devised the election of a clusterhead, that actually runs the algorithm on behalf of all the nodes belonging to the cluster.

By the described strategy, an alert can be processed many times by different clusters, as the related information is carried by different nodes moving through the network, helping increase the localization precision.

The behavior of the framework is illustrated by means of the state machine diagrams depicted in Fig.2 and Fig.3. More specifically, Fig.2 reports the behavior of a node when it detects a malicious activity: information about the physical location of the node at the time of detection and the timestamp itself is inserted into a packet, and sent to its k-neighbors.

The node then waits for a certain amount of time for possible reply packets, containing information about other alerts. Upon reception of those replies, the

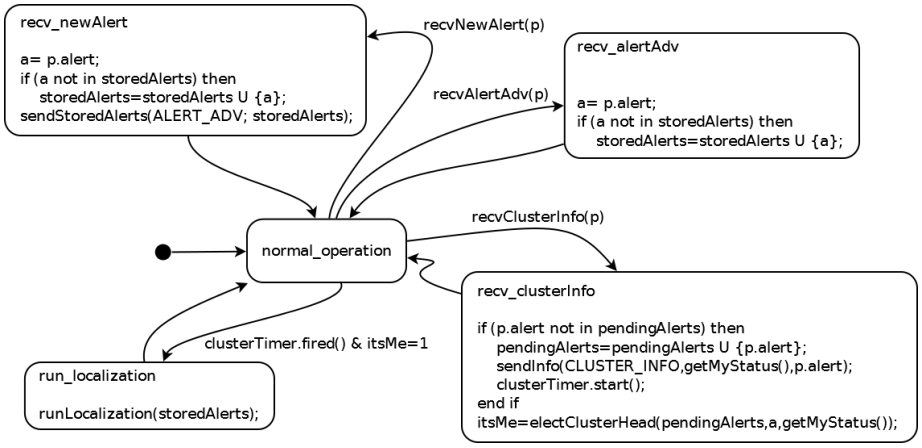


Fig. 3: FSM representation of the behavior of a node at reception of protocol messages

node will update its local list of stored alerts and check if the clustering procedure can be launched, based on the existence of a sufficient number of compatible alerts. In this case, the node sends a packet containing information about its current status to its k-neighbors, in order to participate in the clustering procedure.

Fig.3 shows the behavior of a node at the reception of a packet related to the localization protocol. The protocol adopts 3 different types of packets:

- NEW_ALERT_ADV: contains information about new alerts raised in the network. Nodes receiving such packet must respond with their *storedAlerts* list.
- $ALERT\_ADV$: contains information about stored old alerts that are being re-advertised by mobile nodes.
- $CLUSTER\_INFO$: contains information on the current status of a node, useful to elect the clusterhead that will compute the localization algorithm. Such information is related to a specific alert event, as clustering is launched by a node raising an alert. Actually, a single node could be involved in different clustering procedures at the same time, launched by different nodes. For this reason, as shown in Fig.3, a node receiving such kind of packet will first check whether it is a duplicate of a previous packet, and then will add this request to a local list of pending alerts. Afterwards, it will send information about its current status to its k-neighbors, in order to participate to the clustering procedure for the involved alert. For each received $CLUSTER\_INFO$ packet, the node will run a comparison routine to compare its own status to the received ones, in order to determine whether it should be elected as the clusterhead or not. After a predefined amount of time, if the variable $itsMe$ for that alert is set to 1, the node will assume it is the clusterhead and will run the localization algorithm on alerts in its *storedAlert* list.

The behavior of the proposed framework can be graphically visualized by means of Fig.4. Fig.4(a) illustrates the initial arrangement of 8 nodes in space $\mathcal{S}$. These nodes with identical transmission ranges are capable of broadcasting signals to their neighbors and transmit or receive data within their pre-defined transmission radius.

In the network configuration depicted in Fig.4, P_3 raises the alert a_1 at time t_1, and sends a NEW_ALERT_ADV packet to its neighbor, setting a Time-To-Live (k) equal to 2. Node P_1 first receives the packet and after updating its *storedAlerts* list, broadcasts it to node P_4, without sending anything to P_3 as its initial list was empty. The packet reaches P_4 with a TTL=0; node P_4's *storedAlert* list is empty too, therefore, it simply updates it by adding alert a_1. From this moment on, nodes P_1, P_4 and P_3 will have the information about alert a_1 stored in their local lists; this condition is denoted with $P_1(a_1)$, $P_4(a_1)$ and $P_3(a_1)$ respectively in the figure. No clustering is launched, as there exists no compatible alert related to a_1.

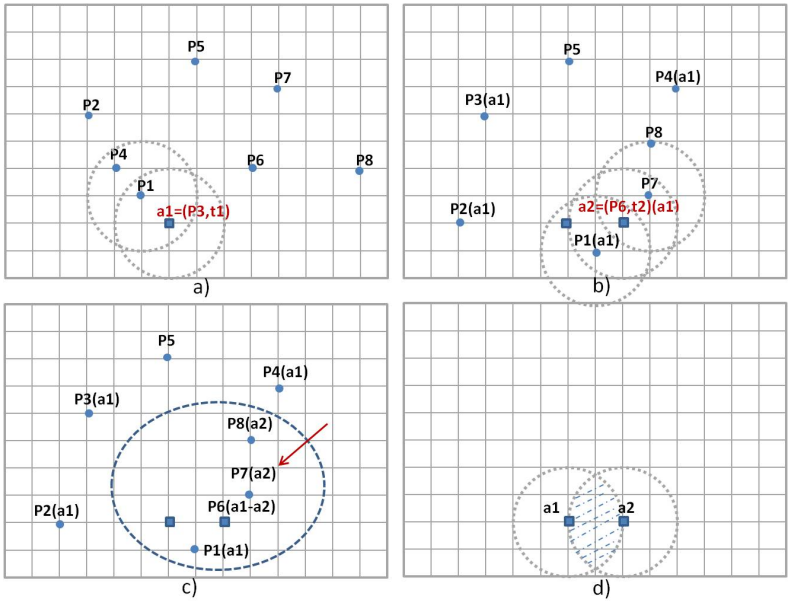


Fig. 4: A simple example of execution of the distributed localization algorithm: a)alert a_1 is raised in P_3, no clustering launched; b) alert a_2 is raised in P_6, that starts clustering; c)the cluster is formed and P_7 is elected as the cluster-head; d)P_6 runs the localization algorithm on the set of alerts $\{a_1,a_2\}$

Fig.4(b) shows a new alert a_2 raised by P_6 at time $t2$. Assume that in the meantime, node P_3 moved in the neighborhood of node P_2 and re-advertised

information about alert a_1: this condition is depicted by denoting $P_2(a_1)$ in Fig.4(b). At time $t2$, node P_6 sends a NEW_ALERT_ADV packet with TTL=2 to its neighbors; the packet is first received by P_1, P_7 and P_8 : node P_7 only updates its *storedAlerts* list and then broadcasts the packet. The packet reaches P_8, that acts similarly. As for node P_1, it knows about alert a_1, therefore, it sends an $ALERT\_ADV$ packet to P_6 to inform the node. At this point, node P_6 will run the *checkCompatibility*() routine on the set $\{a_1,a_2\}$, to find that they are compatible.

As the following step, node P_6 launches the clustering procedure by broadcasting the $CLUSTER\_INFO$ packet for alert a_2; nodes P_1, P_6, P_7 and P_8 participate to the clustering as shown in Fig.4(c), and node P_7 is elected as the cluster head. Finally, node P_7 runs the localization algorithm on the set $\{a_1,a_2\}$, delivering the shaded area in Fig.4(d) as its output.

5 Proposed Clustering Scheme

In this section, we present the clustering scheme adopted by the proposed distributed framework. In order to efficiently assemble the nodes which have raised a security alert, we have proposed to form the clusters based on the node's geographical location, its neighbors and the pre-defined transmission radius.

Each node can hear activities from its neighbors in distance r. If there is any malicious activity in distance r from a node, this information is taken into consideration and will be processed when forming a cluster. The main focus for clustering is to group the nodes which have detected some sort of malicious activity in the same area. As previously pointed out, in order to make the best use of alert information, a cluster is built, starting from a raised alert and including all the k-neighbors of such alert.

Thus, the nodes which have formed a cluster might not be all neighbors with one another straightforwardly, but there are nodes in between which can form a chain (through direct neighbors). In this established cluster there exists a chain of neighbors such that at least two nodes are neighbors pairwise. In other words, the following property holds (for clusters with more than two nodes):

$$\begin{aligned} &(\forall C \in \mathcal{C}) \quad (\forall\, n_i, n_j \in C) \\ \mathcal{D}(n_i, n_j) \leq 2 \cdot r \;\; \vee \;\; &\exists\, n_1, ..., n_k : \quad \mathcal{D}(n_i, n_1) \leq 2 \cdot r \;\wedge \\ \forall l \in (2, k-1) \quad \mathcal{D}(n_l, n_{l+1}) \leq 2 \cdot r \;\; &\wedge \;\; \mathcal{D}(n_k, n_j) \leq 2 \cdot r \end{aligned} \tag{1}$$

Intuitively, for clusters with one or two nodes there exists no constrain as the actual definition of the neighboring nodes clarifies the description.

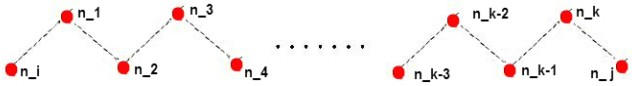


Fig. 5: Chain of nodes forming a cluster

The established clusters can individually and independently perform and execute the localization algorithms proposed in [1] without any exchange of information between clusters. The reason behind this is that we consider the transmission range of the attackers to be the same as the nodes and they are stationary. In other words, the approximate location of an attacker could be determined by processing alert information belonging to a particular cluster. The nodes which have raised an alert gather the information listed below to exchange them with their k-neighbors:

-Node Degree ($Deg(n)$): in graph theory, the order (degree) of a node is the number of attached nodes. In our scenario, nodes within the transmission range are counted as a node's degree. Intuitively, hub nodes have a higher order compared to ordinary nodes. The difference between in-degree and out-degree in a directed graph could be calculated at unique depths: adjacent nodes (depth 1), adjacent nodes of adjacent nodes (depth 2), etc. The following equation holds for adjacent nodes (depth 1):

$$Deg(n_i) = \mid Neighbors(n_i) \mid = \sum_{n_i, n_j \in N} (\mathcal{D}(n_i, n_j) \leq r \ ) \tag{2}$$

This parameter could determine the number of inter-connected nodes which are in the transmission range (depth 1) and it could be used later on as an weighting factor to determine the priority for the node to become a clusterhead.

-Mobility(M_n) : for each node, the average speed of the node until time T is calculated. The formula below determines the relative mobility:

$$M = \frac{1}{T} \cdot \sum_{t=1}^{T} \sqrt{(X_t - X_{t-1})^2 + (Y_t - Y_{t-1})^2} \tag{3}$$

In the equation above, (X_t, Y_t) and (X_{t-1}, Y_{t-1}) are the Cartesian coordinates for the node at time t and *t-1*, respectively. The nodes with less mobility are more likely to be selected as clusterheads as they will be potentially more immune to sudden changes, assuring more stability.

-Residual power (P_n): each transmitted packet includes a value that stores the residual power of each transmitting node. This estimate might not be precise as nodes consume power while receiving packets. Nevertheless, it can be used as an acceptable evaluation standard for the purpose of clusterhead election. This vital information aids in determining if a node has enough power to perform the

tasks related to a clusterhead. Nodes with superior energy have a better chance to be selected as clusterheads, as they have the required resources to operate for a sufficient amount of time.

In summary, an ideal clusterhead should maintain high node degree and residual power in addition to low mobility, compared to other candidates. This ensures best performance as a dominant node which supervises the cluster activities during the network operation.

The above mentioned parameters represent quality factors assigned to each node over time. In order to select the clusterhead, they must be combined according to a quality function to achieve a final weight. As shown in [4], the weight to be assigned to each node n (which has raised an alert) can be computed using the following formula:

$$W_n = k_1 \cdot Deg(n) + k_2 \cdot M_n + k_3 \cdot P_n \tag{4}$$

Parameters k_1, k_2 and k_3 are selected as corresponding weighting factors which all add up to a constant value. When a node receives a $CLUSTER\_INFO$ message, the included weight is compared with its own weight. The node which has the smallest weight factor of all the neighboring nodes is selected as the clusterhead.

In addition, when the process of selecting a clusterhead begins, depending on the geographical location of legitimate nodes and the nodes raising an alert, it might be prudent to elect a legitimate node in the close vicinity as a clusterhead to avoid any future possible failure of the clusterhead. Nevertheless, the proposed attributes need to be evaluated for this election.

Once the clusters have been established and the clusterheads are elected, we can execute the heuristic algorithms to search for the attackers approximate location as it has been proposed in [1]. Using the local information obtained by the nodes (which have raised an alert), the MIN-K and MULT-UPD algorithms could be run locally to find the approximate location of the attackers.

6 Experimental Results

We implemented a prototype of the proposed framework in the NS-2 network simulator, and developed a Java application for the processing of alerts in the clusterheads. As previously illustrated, nodes exchange information on alerts raised in the network, and nodes that have detected an attack autonomously decide whether or not to launch the clustering procedure in order to process known alerts. We used NS-2 to simulate different scenarios in which nodes move according to a Random Way Point model[3] and attackers randomly choose their neighbor(s) as targets. We recorded the time of each attack, along with the position of the victim at the time of the attack, and run the localization algorithm

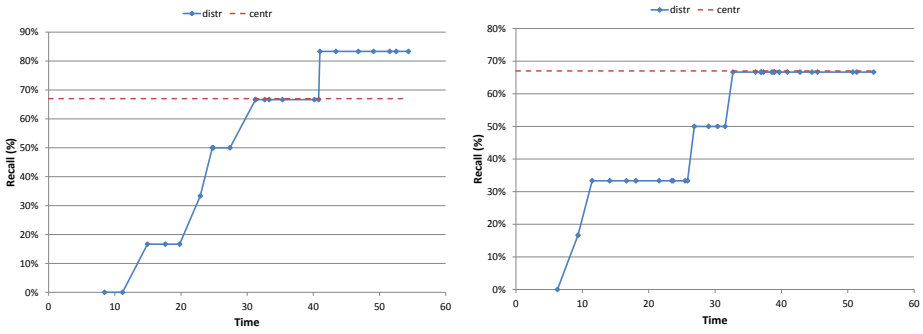
[3] However, our approach allows us to use any mobility model as well as any radio propagation model in the simulation.

on the set of alerts known by each cluster at the time when localization was launched.

For our experiments, we adopted the MIN-K deployment algorithm presented in [1], and analyzed the behavior of the distributed localization framework in different scenarios and operative conditions.

In the first set of experiments, we considered a $1km \times 1km$ field, and deployed 40 network nodes and 6 attackers, both uniformly distributed. We considered an observation interval of 60 ms, enabling attacks in the first 50 ms of simulation, and set the cluster depth to 2. All nodes are assumed to be compatible with the free space radio propagation model and to have a transmission range of 100 meters. To calculate the attacker probability distribution, we assumed that, given an alert a, the attacker probability distribution for a is uniformly distributed in the circle having its center in a and radius equal to the transmission range.

We considered several random scenarios and run both framework versions on each scenario to compare the number of attackers that they were able to capture within a single deployment cycle. Contrarily to what might be believed, the centralized version of the localization framework does not always perform better than the distributed one, even if the knowledge about existing alerts is more complete. This is due to the way alerts are combined by the localization algorithm, that aims at minimizing the number of expected attackers in the network, trying to combine as many compatible alerts as possible. Fig. 6 shows the fraction of attackers "captured" within the first and only deployment cycle (also referred to as a *recall* parameter) in two different cases: in the case shown in (a), the distributed framework is able to capture more attackers than its counterpart before the end of the observation interval, while in (b) the centralized framework works better.



(a) (b)

Fig. 6: Centralized Vs. Distributed Localization in different scenarios

In order to analyze the impact of the number of alerts on the localization accuracy, we considered a particular attack scenario, composed of a single attacker that launches an attack against all the nodes in its transmission range (e.g. a jammer). In this scenario, the above discussed influence of the alert distribution on localization accuracy is reduced, as the goal of minimizing the number of attackers responsible for all alerts is consistent with the existence of a single attacker. In [1], we already showed that our approach is able to localize jammers with an higher precision than other existing approaches based on geometrical considerations, and is less dependent on the network density. With the introduction of the distributed version of the framework, we are able to obtain even better results, as the attacker can be localized earlier, by processing a limited number of alerts. Fig. 7 illustrates the average localization error as the number of alerts increases, showing that it sensibly reduces even with small increments in the alerts' number.

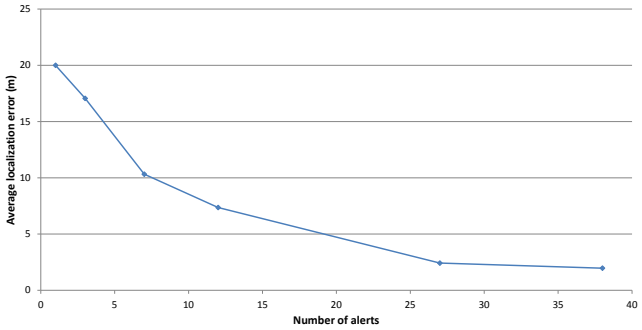


Fig. 7: Impact of the number of alerts on the localization accuracy

As discussed in Section 4, in order to cope with mobility and temporary network partitioning, nodes advertise both locally generated alerts and old stored alerts to their k-hop neighbors. The choice of the k parameter, also called *cluster depth*, impacts both on the protocol overhead and on localization precision: as the size of the set of alerts to process in a localization step increases, the probability that such set contains overlapping alerts which are useful for a successful localization, grows. Fig. 8 shows the recall parameter when choosing two different values of cluster depth, namely 1 and 3, for the same simulation scenario. As shown, the framework achieves better results when the depth is set to 3, even if in this case the total number of alert advertisement packets sent by nodes during the simulation time is much higher compared to the other case (139 to 57).

Clearly, as the introduced alert advertisement protocol adopts a controlled-flooding strategy, the localization framework is subject to a communication overhead due to the forwarding of packets containing information about generated alerts. Nevertheless, such overhead is limited and has a linear trend, as shown

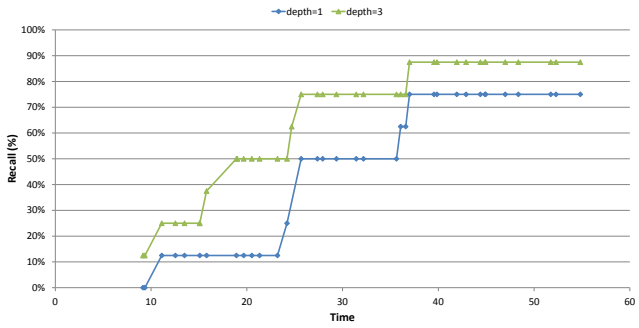


Fig. 8: Recall values for different cluster depths

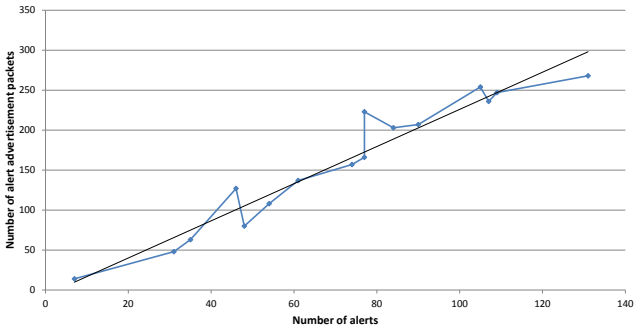


Fig. 9: Communication overhead

in Fig. 9, which depicts the number of NEW_ALERT_ADV packets generated and forwarded to k-hop neighbors (with k=2).

7 Conclusions

In this paper, we addressed the problem of localizing attackers present in MANETs. In particular, we developed a distributed framework based on the dynamic partitioning of the network. The proposed distributed framework assembles the alerts as soon as they are reported. The protocol can independently localize the approximate position of attackers in a distributive mode through cooperation of neighboring nodes.

In order to improve the efficiency of the computation, we implemented two tasks. First we grouped the alerts to form clusters for local processing and secondly we introduced a startegy to elect a clusterhead for the actual execution of the localization algorithm. Polynomial heuristic algorithms have been iteratively implemented to precisely detain all the attackers in the network.

We evaluated the performance of our distributed framework in NS-2 network simulator and experiments indicated that our scheme achieves better results

compared to the centralized localization algorithm. Further, based on the amount of attackers defined by our mechanism, our distributed localization framework can dispatch trusted resources to capture the attackers in the effected regions of the network. In our future work, we will extend our distributed localization scheme with the assumption that attackers are dynamic so they can be chased and captured.

References

1. Massimiliano Albanese, Alessandra De Benedictis, Sushil Jajodia, and Paulo Shakarian. A probabilistic framework for localization of attackers in manets. In *Computer Security–ESORICS 2012*, pages 145–162. Springer, 2012.
2. Dennis Baker and Anthony Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *Communications, IEEE Transactions on*, 29(11):1694–1701, 1981.
3. Dennis J Baker and Anthony Ephremides. A distributed algorithm for organizing mobile radio telecommunication networks. In *Proceedings of the 2nd International Conference on Distributed Computer Systems*, pages 476–483, 1981.
4. Mainak Chatterjee, Sajal K Das, and Damla Turgut. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5(2):193–204, 2002.
5. Yih-Chun Hu, Adrian Perrig, and David B Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11(1-2):21–38, 2005.
6. Hongbo Liu, Zhenhua Liu, Yingying Chen, and Wenyuan Xu. Localizing multiple jamming attackers in wireless networks. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 517–528, 2011.
7. Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 255–265, New York, NY, USA, 2000. ACM.
8. Jie Yang, Yingying Chen, Wade Trappe, and Jerry Cheng. Detection and localization of multiple spoofing attackers in wireless networks. 2013.
9. Yingpei Zeng, Jiannong Cao, Jue Hong, and Li Xie. Secure localization and location verification inwireless sensor networks. In *Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on*, pages 864–869, 2009.
10. Siyu Zhan and Jianping Li. Active cross-layer location identification of attackers in wireless sensor networks. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 3, pages V3–240–V3–244, 2010.