When the Network of a Smart City Is Not So Smart

Ali Tabaja Reuven Cohen*
Technion - Israel Institute of Technology
Haifa, Israel

***Abstract*—In this paper we show that a simple repeated (ON-OFF) jamming attack on a wireless distance vector protocol can have a global effect on the network and be as harmful as a sophisticated Network layer attack. This attack takes advantage of the nodes' limited awareness of topological changes when distance vector routing is used. To present the attack in a specific context, we consider the new RPL protocol, which plays an important role in the context of Smart City and Smart Utility networks. We describe the RPL approach to recover from link failures and show that this protocol is susceptible to the proposed attack. We also present a remedy technique and show that it can significantly alleviate the impact of the proposed attack.**

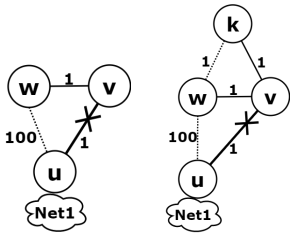
I. Introduction

Intra-autonomous system routing protocols have traditionally been classified into two categories: those that use distance vectors and those that use link state. The main advantage of distance vector protocols is their simplicity and efficient bandwidth utilization. Link state protocols, on the other hand, are known to enable fast convergence after topological changes due to each node being fully aware of the network topology.

In today's wireline networks, link state protocols, such as OSPF and IS-IS, are much more popular than distance vector protocols because the efficient bandwidth utilization provided by the latter is no longer important. However, in wireless networks, where bandwidth is always a scarce resource, distance vector routing protocols are very popular, especially for proactive routing and for networks where the nodes are stationary and constrained. For example, RPL [1] is a relatively new IETF protocol for low power and lossy networks (LLNs) [2], and it is based solely on distance vector routing.

Wireless mesh networks are known to be vulnerable to numerous attacks that may be launched on any layer [3], [4]. Most of the attacks on wireless networks are conducted either on the Phy/MAC layer or on the Network layer. These attacks differ mainly in that the former is much easier to conduct. A Phy/MAC layer attack can be conducted simply by transmitting radio signals over the channel, a process known as jamming [5], whereas conducting an effective Network layer attack usually requires the attacker to gain a full control of at least one legitimate node in the network and change the code executed by this node. However, the impact of Phy/MAC layer attacks is usually limited to a certain area, whereas Network layer attacks have the potential to sabotage the entire network.

*The present work has been partially funded by NATO Science for Peace and Security (SPS) Programme, in the framework of the project SPS G5428 "Dynamic Architecture based on UAVs Monitoring for Border Security and Safety"



(a) (b)

Fig. 1: (a) A 2-node loop may be formed; (b) A 3-node loop may be formed when "split horizon" is used.

In this paper we show that a simple repeated (ON-OFF) jamming attack on a wireless distance vector protocol can have a global effect on the network and be as harmful as a sophisticated Network layer attack. This attack takes advantage of the nodes' limited awareness of topological changes when distance vector routing is used. As an example, consider Figure 1(a), which shows routing from three nodes to Net1. The figure also shows the cost/weight associated with every wireless link. Initially, nodes w and v route through $w - v - u$ and $v - u$ respectively. Now suppose that a jamming attack takes link $v - u$ down. One would expect the nodes to then use the costly link $w - u$ instead of $v - u$. However, with distance vector routing, a routing loop might be formed, and it might take a long time before the network recovers from the loop and converges to the correct routing. The reason is that node w is not aware of the failure of $v - u$ and it thus informs v that it can route to Net1 with cost 2. Node v chooses w as its next hop towards Net1 with cost 3. Then, it informs w that it has a route with this cost, and node w believes that it can still route through v, but with cost 4 rather than 2. This process is repeated until the cost of the path from w through v is 102, in which case w prefers to route through u, with cost 100. Only then, does the network stabilize and packets reach Net1.

Most distance vector routing protocols use "split horizon" to prevent such 2-hop loops. With "split horizon" a node w never informs its neighbor v about its route to a given destination if this neighbor v is its next hop towards this destination. But "split horizon" does not work when more nodes are involved in the loop, as in Figure 1(b). In this example, in the worst case, about 100 exchanges of distance vectors between nodes v, k and w are needed before all nodes understand that the only option to route to Net1 is the expensive link $w - u$. This problem of oscillating between non-existing routes until stabilizing on the correct-but-expensive route is also known as

"count-to-infinity".

Consider again Figure 1(b). When the attacker stops jamming, node v realizes that it is again connected to u, and it informs w and k. Then, w and k change their route to Net1 such that it goes via v rather than via w. This implies that when the jamming stops, the network will rapidly converge to the correct state. But if the attacker starts jamming again, it takes again a very long time for v, w and k to converge on the correct route via the link $w-u$. The attack is summarized in Figure 2.

"Split horizon" is not the only way to mitigate the "count-to-infinity" problem. However, *past works considered the problem only in the context of legitimate topological changes and thus do not provide a solution for the proposed attack*. In this paper we show that a repeated (ON-OFF) jamming attack on a distance vector routing protocol, which repeatedly takes down one or several links temporarily, can take down a significant part of the network for as long as the attack continues.

The proposed jamming attack is actually a Network layer attack not because it (obviously) affects routing, but because it is so difficult for the routing protocol to converge to the new (inefficient) routes. In particular, packets are routed in loops for a long time, wasting a lot of bandwidth but not reaching their destinations.

In other words, any jamming attack that takes one or more links down is likely to make routing less efficient. The difference between the cost of efficient routing (when all the links are up) and the cost of less efficient routing (when some links are down due to the jamming) is a straightforward penalty of the jamming. We call it a *routing penalty*. This is, however, not a penalty of a Network layer attack, because there is nothing the Network layer can do except than routing over the best routes available after the topological changes. To be considered as a Network layer attack, the penalty of the attack must exceed the penalty of the less efficient routing, and this extra penalty must be attributed to bad design of the Network layer. As shown in this paper, this is the case when the network layer is based on the concept of distance vector routing. We call this Network layer penalty a *convergence penalty*.

As an example of this convergence penalty, consider the proposed repeated (ON-OFF) jamming attack, whose ON period is 60 seconds and whose OFF period is 10 seconds. We now estimate the convergence penalty of link state routing compared to the convergence penalty of distance vector routing. With link state wireless routing protocols, such as OLSR [6], the network will stabilize within a short time, about 5 seconds, after the ON period starts. Thus, the convergence penalty is $5/60 = 8\%$ of the ON period during which routing is unstable or impossible. With distance vector routing, the convergence time during the jamming ON period is very long, e.g., 50 seconds. Thus, the convergence penalty is $50/60 = 83\%$ of the ON period, which is 10 times larger than the previous case.

To present the attack in a specific context, we consider the new RPL protocol, which plays an important role in the context of Smart City and Smart Utility networks. We describe the RPL approach to recover from link failures and show that this protocol is susceptible to the proposed attack.

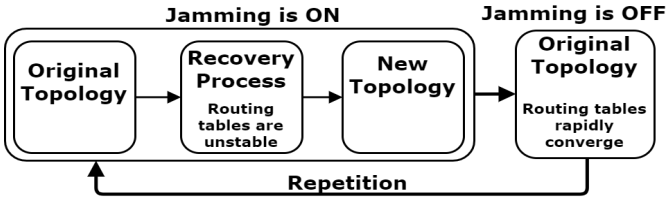


Fig. 2: High level description of the proposed attack.

The contribution of this paper is three-fold.

- We show that a simple Phy layer jamming attack can sabotage the functionality of the network layer, and it is therefore as harmful as sophisticated Network layer attacks that usually require the attacker to gain control of a legitimate node.
- We show that RPL is susceptible to the proposed attack despite its built-in mechanism (called Local Repair) for coping with the "count-to-infinity" problem of distance vector routing.
- We present a remedy technique, and show that it can significantly alleviate the impact of the proposed attack.

The rest of the paper is organized as follows. Section II discusses related work. Section III summarizes the major RPL features that are relevant to the presented attack and the protocol's repair mechanisms. Section IV introduces the new attack in the specific context of RPL and proposes a mitigation technique. Section V discusses how to plan the most effective attack. Section VI presents simulation results, and Section VII concludes the paper.

II. Related work

Many works discuss jamming attacks in wireless networks and possible detection and mitigation techniques. In [5], the authors distinguish between four different jamming attacks: constant, deceptive, random and reactive. They discuss a detection system that uses only one measurement, such as calculating the packet sent ratio and using a signal strength spectral discrimination technique. They show that such a system can only distinguish between some normal and abnormal scenarios. Then, they show that only a combination of different techniques is useful for detecting a jamming attack. An example of such a combination is measuring the packet delivery ratio and performing a signal strength consistency check. The takeaway is that jamming detection is not a simple task, and it requires significant network resources.

In [7], the authors evaluate the effectiveness of jamming according to the attacker's energy, the probability to be detected, the extent of the denial of service and more. In addition, detection techniques are introduced and discussed. One of them is a technique proposed in [5], which offers consistency checks. The authors state that when the distance, the effectiveness of the technique is harmed. The authors also propose a punishment-based approach against internal nodes

that use the channel too heavily. However, this technique is problematic since it requires a modification to the MAC layer protocol.

The authors of [8] introduce selective jamming and selective dropping attacks. Selective jamming targets either a specific channel or specific data packets. Replication of control packets over all control channels, or changing channels dynamically are proactive approaches to alleviate the damage of this attack. In data selective jamming, the attacker spoofs and analyzes packets to determine if it is beneficial to trigger a MAC layer collision.

In [9], several RPL attacks are discussed. These attacks are classified according to the CIAA model (Confidentiality, Integrity, Authentication and Availability). The authors first define routing assets and points of access that an attacker may want to sabotage. Then, they provide different attack schemes for each category. For example, under integrity, they mention routing information manipulation and node identity misappropriation attacks.

In [3], attacks on RPL are classified according to the attacker's goals and means. This is the first work that discusses well-known attacks in the specific context of RPL. The various attacks are classified to those aimed at resources, at network topology and at traffic. For each attack, it is indicated whether the attack does or does not require a node, whether the attack is passive or active, the possible mitigation, and so on. Two main observations from the analysis in this paper are that many attacks require a node, and that further research is required in order to cope with many RPL attacks.

Other RPL security studies discuss attacks that require control of a node, or present high level security issues. In [10], several attacks are simulated to explore the protocol's self-healing ability. One such attack is selective-forwarding [11], in which a malicious node selectively filters the network messages. The authors show that RPL alone cannot recover from this attack.

III. RPL and Local Repair

RPL builds a collection of Destination Oriented Directed Acyclic Graphs (DODAGs), through which every node is connected to one gateway. Every node in a DODAG has one or more parents through which it can reach the gateway. A tree is a special case of a DODAG, where every node has at most one parent.

RPL is a quite complicated protocol whose RFC contains more than 150 pages. In what follows we summarize the main rules according which a node chooses a parent, a gateway and a default route:

1) Each gateway is configured to be a root of a tree. When it wakes up, it periodically broadcasts DODAG Information Object messages (DIOs) to invite nearby nodes to choose it as a parent.
2) When a node wants to join the network, it waits to hear DIO messages from nearby nodes that have already joined one of the trees, or from a gateway. The best path is selected according to an Objective Function (OF).
3) A node that already belongs to a certain tree advertises DIOs, to invite its neighbors to join its tree as its children.
4) After a node chooses a parent and joins the tree of its parent, it continues listening to DIO messages. If it receives a more attractive DIO (with respect to the OF), it can switch to a new parent, regardless of whether this parent is in the same or in a different tree.

Each RPL network associates a metric (cost) with every link and path. The metric is supposed to reflect the property of the link/path according which each node decides which of its neighbors to choose as a parent. A metric of a link can represent the link quality, the link throughput, the link robustness, etc. It can also represent any combination of several properties. A RPL network is also associated with an Objective Function (OF), which dictates how the metric is processed. Each node v that receives a DIO message from a neighbor, say u, finds in this message the metric (cost) of the path from the root to u. Using this metric, as well as the metric of the link $v-u$ and the OF, node v determines the cost of the path to the root through node u. Examples of common OFs are:

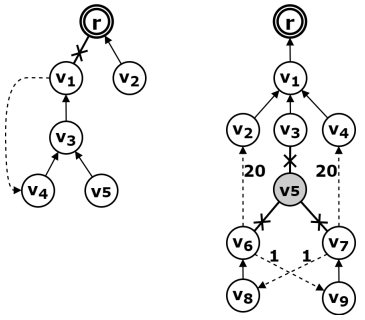
1) Minimize the hop distance from the gateway.
2) Maximize the minimum link quality on the path to the gateway.

An important difference between the two OFs described above is that the first is additive and the second is not. As a distance vector protocol, RPL works much better with additive OFs, mainly because they make it easier to prevent routing loops. When the chosen OF is additive, the metric computed by a parent and announced in its DIOs is always smaller than the metric computed by its descendants, whereas with a non-additive OF this is not necessarily the case.

RPL uses an additive metric, called *rank*, whose main role is preventing routing loops. When the OF is additive, it can also be used as a rank. A node that loses its parent is allowed to switch to a new parent while keeping its entire subtree only if this will not increase its rank. This guarantees that selecting the new parent will not create loops. If such a new parent is not found, node v must first dismiss its children, as part of the Local Repair procedure described later, and then choose a new parent regardless of the rank.

To simplify the discussion in this paper, we assume that the network uses an additive OF. Thus, the metric of a path from a gateway to any node also represents the rank of this node. The additive OF we choose is a generalization of shortest path: each node assigns to each outgoing wireless link an integer between 1 and 5, which represents the quality of the link (1=high, 5=low); the metric of a path is then the sum of the metrics of the links that it uses. This concrete assumption does not affect the generality of our results.

RPL proposes two different techniques to cope with topology changes: Global Repair and Local Repair. Global Repair is simple but very costly. The idea is that from time to time the root nodes (gateways) initiate the creation of a "new version" of the trees. A decision to initiate a new version can follow a



(a) (b)

Fig. 3: Routing loops created when nodes choose new parents after a failure.

constant schedule (e.g, once an hour), or can be made when the network operator believes that the trees are not stable. A new sequence number is assigned to each new version, and it is attached to the DIOs broadcast by the gateways.

To address topological changes without invoking Global Repair too often, RPL also uses Local Repair. In Local Repair, when a node v loses its parent, it first searches for an alternative parent, namely, a neighbor that can replace the old parent without increasing the rank of v before the failure. If such a neighbor is found, node v can switch to the new parent without risking the creation of a routing loop. If such neighbor is not found, there are 3 possible cases:

(a) Node v has no connectivity to any gateway.
(b) The only way for node v to reconnect to the network is by choosing as a new parent a node in its subtree, thereby increasing its rank.
(c) The only way for node v to reconnect to the network is by choosing as a new parent a node that is not in its subtree, thereby increasing its rank.

In case (b), a loop would be created as depicted in Figure 3(a). In case (c), a loop might be created as shown in Figure 3(b). In this Figure, node v_5 goes down, leaving v_6 and v_7 without a parent. When a DIO from v_8 arrives to v_7 with cost = 5 (the cost of the path from v_8 before it knows that its path is down), v_7 chooses v_8 as its new parent since this offer is better than the one advertised by v_4 (6 vs. 22). For the same reason, v_6 chooses v_9 as a parent, and a loop is created.

To prevent such loops, Local Repair dictates that when a node v loses its parent and has no alternative parent that does not increase node v's rank, it dismisses its children, waits for a while, and only then tries to rejoin the network even if this would increase its previous rank. In Figure 3(a), after v_1 decides that it cannot choose a new parent without increasing its rank, it broadcasts DIOs that dismiss v_3 from its subtree. Then, v_3 broadcasts similar DIOs and dismisses v_4 and v_5. Algorithm 1 summarizes the Local Repair procedure.

IV. Attacking Local Repair

As explained in Section III, when a node v discovers that the connectivity to its parent u is lost, it invokes Local Repair.

Algorithm 1 Local Repair

```
1: if there is a node that can be chosen as a parent without
   increasing your rank then
2:     choose it as your new parent
3: else
4:     dismiss your children by advertising an infinite metric
5:     wait for a predefined period
6:     choose as a parent the neighbor with the best offer
```

If node v has an alternative parent, namely a parent through which its rank does not increase, it immediately switches to this parent. If not, it dismisses its children. Consequently, many nodes in its subtree might have to invoke Local Repair as well, and it may take a long time until the tree is re-stabilized. Suppose that when the whole tree is stabilized, node v has a new parent w. It is very likely that its path cost in the new tree is higher than in the old tree; if not, node v would have chosen w as a parent before the failure.

Now suppose that the wireless link between v and its previous parent u is up again. If the quality of this link is similar to what it was earlier, it is better for v to choose u as a parent once again. Switching to a better parent is relatively fast and easy because it requires node v to leave w and to associate with u without invoking Local Repair. But if the connectivity between node v and u is lost again, Local Repair needs to be reinvoked by v and by most of the nodes in its subtree.

The attack is described in Figure 4. The various steps are as follows:

- Step 1(a): A jamming attack takes place close to the root r. As a result, both v_1 and v_9 lose their connectivity to their parent r.
- Step 1(b): Node v_1 invokes Local Repair. Since it does not have an alternative parent, namely, a neighbor that can be selected as a new parent without increasing node v_1's current rank, v_1 must dismiss its children. This process is likely to be repeated recursively, because it is invoked by each descendant that is dismissed by its parent and does not have an alternative parent. Node v_9 also invokes Local Repair, but it neither has an alternative parent nor children to dismiss.
- Step 1(c): Node v_1 and all the nodes in its previous subtree try to choose new parents, even if this requires increasing their previous ranks. In this example, some of the compromised nodes (v_1, v_3, v_4, v_7 and v_8) are able to reconnect to the tree, while others (v_9) remain disconnected.
- Step 2: The attacker stops jamming. Consequently, the wireless links that were affected by the jamming are up again. Node v_1 chooses r as its parent. This decreases the rank of v_1, and Local Repair is not invoked. Node v_9, which was not able to find a new parent while the jamming was on, can now reconnect to the tree. After a while, the original tree is reconstructed.

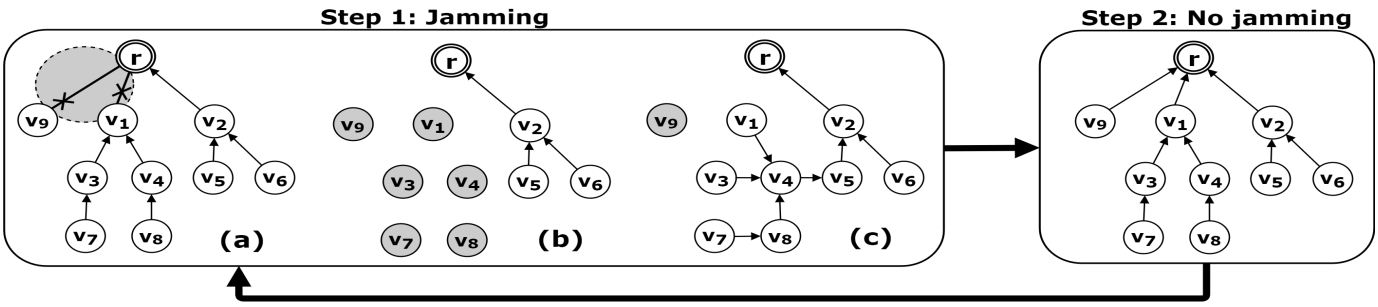


Fig. 4: The proposed jamming attack in RPL.

- The attacker starts jamming again.

The attack has a significant negative impact on the network for the following reasons:

(a) Both upstream traffic (from the nodes to a gateway) and downstream traffic (from a gateway to the nodes) are lost. Downstream traffic is lost because it is misrouted. Upstream traffic is lost because a node that changes its parent must reset the Layer 2 buffers associated with its previous parent.
(b) The process of finding a new parent in Local Repair requires significant time and bandwidth. Each node that invokes Local Repair must wait until all the nodes in its subtree either dismiss their children or find alternative parents. Then, such a node must scan several channels and search for the most attractive parent. Then, it must associate with the new parent.
(c) In many systems, if a node connects to a new parent, it must inform the gateway or even a remote central office. This is required for authentication, registration and for managing the node application. Registration requires the exchange of several end-to-end messages, which are very costly in a multi-hop wireless network.

As shown in Section VI, the most effective location for the attack is very close to a gateway. When a node close to the gateway loses its parent, it is likely that this node has a subtree with many descendants. Thus, many nodes would invoke Local Repair, and the tree reconstruction is likely to be costly and long.

In terms of the jamming power, the attacker has two contradictory goals: using maximum power in order to force as many nodes as possible to lose their parents and dismiss their children vs. minimizing the power so as not to be easily discovered. **In the proposed attack the transmission power of the attacker can be significantly lower than that of a legitimate user, since it is sufficient to destroy one wireless link in order to affect tens or even hundreds of nodes.**

The proposed attack is very effective and there is no easy way to handle it. If the attacker chooses a random location to jam, and then – during the OFF period – it moves to a new location before it hits again, we see no remedy technique except to find and stop the attacker, which is hard to do in a city (outdoor) network, especially if the attacker uses normal transmission power.

We now introduce a back-off scheme to mitigate the negative impact caused by parent nodes that become disconnected very often. We call these parents "non-persistent". In the proposed scheme, a node v ignores a specific non-persistent parent u for a time period $B(u)$, whose length is determined by the following parameters:

1) The length of the last alive period of this parent, denoted Δ_1. A parent is considered alive if its BEACONS are heard.
2) The length of the last non-alive period of this parent, denoted Δ_2. A parent is considered non-alive if, after it became alive, β consecutive BEACONS it was supposed to send were not received.
3) A moving average of the recent alive and non-alive periods of this parent, denoted $\overline{\Delta_1}$ and $\overline{\Delta_2}$ respectively. A moving average of Δ_1 is defined as $\overline{\Delta_1} = \alpha * \overline{\Delta_1} + (1-\alpha) * \Delta_1$, where $0 < \alpha < 1$. $\overline{\Delta_2}$ is computed in the same way. These averages are updated every time a parent moves from being alive to non-alive or vice versa.

Non-persistent parents differ on how often they fluctuate between alive and non-alive periods and in the percentage of time during which they are non-alive. The two parameters are approximated as follows:

1) Fluctuation frequency is $p_1 = \frac{1}{\overline{\Delta_1}+\overline{\Delta_2}}$.
2) Non-alive period is $p_2 = \frac{\overline{\Delta_2}}{\overline{\Delta_1}+\overline{\Delta_2}}$.

The value of p_1 is then normalized by dividing it by the sum of p_1 of all neighbors. From here on, we use p_1 to refer to the normalized version of this property.

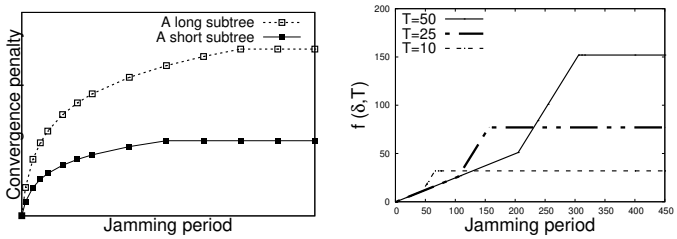
The Non-Persistence (*NP*) rank of each parent is defined as:

$$NP = \gamma * p_1 + (1-\gamma) * p_2, \tag{1}$$

where $0 \leq \gamma \leq 1$. *NP* is used for comparing the non-persistence level of potential parents. A more persistent parent has a lower *NP* value.

if *NP(u)* is the *NP* rank of u, the amount of time during which u is banned, denoted $B(u)$, is:

$$B(u) = A * \overline{\Delta_2} * NP(u), \tag{2}$$



(a) (b)

Fig. 5: (a) The convergence penalty as a function of the jamming period length; (b) The convergence penalty as a function of the jamming period length δ in the formal model ($K = 5$).

where $1 \leq A$ is a multiplication factor that reflects the degree of amplification of the banning. When A is smaller, the algorithm is more tolerant toward non-persistent parents.

This technique serves to maintain stability in the network. In Section VI, we show that the remedy technique can significantly alleviate the impact of the proposed attack, even in a special case of this algorithm where $B(u)$ is a constant function.

V. Planning the most effective attack

In Section I we distinguished between the following two impacts of the attack:

- **Convergence penalty:** incurred from the beginning of the jamming period till the routing tables converge.
- **Routing penalty:** incurred during the entire jamming period due to routing on a network that cannot use the links in the jammed area.

We also said that the routing penalty is not unique to the proposed attack, because it is a direct consequence of jamming under any Network layer design. What makes the proposed attack very effective is the convergence penalty, which is unique to networks that use distance vector routing, such as RPL.

In order to maximize the impact of the attack, the attacker's goal is to maximize the fraction of jamming period during which the convergence penalty is incurred. For example, if the convergence time is 10 seconds, the attacker strategy should be to stop the jamming after 10 seconds, let the network to stabilize again on a topology that includes all the links, and then restarts jamming for 10 seconds. If the attacker does not stop jamming after 10 seconds, but after 20, then the network incurs the convergence penalty during only 50%, rather than 100%, of the jamming time.

Therefore, to maximize the effect of the attack, the attacker should estimate how long time it would take for the network to converge after the jamming starts. In RPL, this time depends on the depth of the subtree rooted in the area affected by jamming. For example, an attack near v_3 in Figure 3(b) should last longer than an attack near v_6, because the convergence time in the first case is longer than in the second.

This idea is depicted in Figure 5(a), which illustrates the relationship we expect to find between the length of the jamming period and the total convergence penalty for the case where the affected subtree is long and for the case where it is short (this graph is used for illustration only, and it was not obtained through simulations). We expect the jamming period for the short subtree to be smaller. We also expect the total convergence penalty to increase till it reaches a maximum (when the routing tables are converged), and that this maximum will be bigger when the affected subtree is longer.

To summarize, the attacker needs to jam as close to the gateway as possible, and it needs to estimate the convergence time in order to decide when to stop jamming. We now present a mathematical model to determine the length of the jamming period for maximizing the convergence penalty. We start with a few definitions:

- A **time unit** is the time to deliver one message.
- A **convergence round** is a time unit during which at least one node either receives a release message from its parent or receives an offer from a better parent and switches parents during the ON period. If the node is disconnected, the new offer is always considered a better one for this definition.
- The **subtree release threshold**, denoted by K, is the number of time units a node waits to determine that its parent does no longer offer connectivity. A node understands that this is the case either by receiving a release message from its parent and then waiting for K time units, or by waiting K time units after receiving the last DIO message from its parent.

We can now define **the convergence penalty of an attack** more formally. For a subtree whose depth is T, and for an attack whose jamming period is δ time units, the convergence penalty is the number of convergence rounds that take place due to the attack, and it is denoted $f(\delta, T)$.

Claim 1: It takes $(i+1) * K$ time units for a node in depth i to disconnect from its parent after an attack starts. Recall that K is the subtree release threshold.

Proof. Denote by $h(i)$ the number of time units it takes for a node in depth i to disconnect from its parent. $h(0) = K$ by definition. Node v understands that its parent u does no longer offer connectivity if the following two events occur consecutively: 1) u loses its parent and sends v a release message m that indicates that u does no longer provide connectivity to the root; 2) v waits K time units after m arrives and then decides that u is no longer a suitable parent. Thus $h(i) = h(i-1)+K$, which yields that $h(i) = i * K + K = (i + 1) * K$. □

Claim 2: The jamming period length required to release the first i layers of the attacked subtree is at least $K * (i + 1) - (i + 1) + 1 = (K - 1) * (i + 1) + 1$ time units.

Proof. Following Claim 1, node v_i at depth i needs $(i + 1) * K$ time units to determine that its parent is lost. When the jamming period stops, it takes $(i+1)$ time units for v_i to hear

from its parent v_{i-1} again, since the information propagates from v_0 (the "root" of the attacked subtree) to v_i through the path $v_0, ..., v_i$ (v_0 also needs one time unit to hear from its original parent). If the jamming period lasts $K*(i+1)-(i+1)$ time units, v_i receives a message from its original parent v_{i-1} exactly when v_i is about to determine that v_{i-1} is lost. Thus, an extra time unit is used to avoid this race. □

Denote $(K-1)*(i+1)+1$ by r_i, the time units required to release the subtree until the ith layer. Let T be the depth of the affected subtree. The shortest time needed to release the whole subtree is therefore r_T. As a result, if the jamming period length, denoted by δ, fulfills that $r_i \leq \delta < r_{i+1}$ holds for some $0 \leq i \leq (T-1)$, the subtree is released exactly until layer i. Therefore, the convergence penalty is $(i+1)$, which is the number of convergence rounds for releasing the subtree until the ith layer. We now discuss what happens after the release of the subtree.

Claim 3: In the worst case, it takes $2T+1$ time units after the subtree is released until it converges to the new topology (the original topology without the nodes/links taken down by the jamming).

Proof. If the depth of the attacked subtree is T, the maximum distance between two leaves is $2T$. If one of these leaves finds a new parent, it takes $2T+1$ time units before the other leaf is informed. □

From Claim 2 and Claim 3 follows that, if the jamming period δ is longer than r_T but is shorter than $r_T + (2T+1)$ (i.e. long enough to release the entire subtree, but not long enough to enable the subtree to complete its convergence to the new topology), the convergence penalty is $(T+1)+(\delta - r_T)$, where $(T+1)$ rounds are needed for releasing all the nodes, and $(\delta - r_T)$ rounds are needed for re-connecting the nodes (the convergence is not necessarily completed).

To maximize the convergence penalty, the attacker must stop jamming after $t' = r_T + (2T+1)$ time units. The convergence penalty in this case is $(T+1)+(2T+1) = 3T+2$ convergence rounds: $(T+1)$ rounds are required for releasing the entire subtree, and $(2T+1)$ rounds are required to enable the subtree to fully converge to the new topology.

To summarize, for a release threshold K, an attack whose jamming period is δ on a subtree whose depth is T, yields the following convergence penalty:

$$f(\delta, T) = \begin{cases} i+1 & \exists i \in \{0, ..., (T-1)\}: \\ & r_i \leq \delta < r_{i+1} \\ (T+1)+(\delta - t_T) & r_T \leq \delta < t' \\ (T+1)+(2T+1) & t' \leq \delta \\ 0 & \text{Otherwise} \end{cases}$$

Figure 5(b) depicts this function for three values of T (depth of the affected subtree), when the release threshold $K = 5$. We use the formal model to determine the jamming period length of the attacks simulated in Section VI.

If the attacker knows the depth T of the affected subtree, it can use the formal model to determine the optimal jamming period length. However, it is very likely that the attacker does not know the value of T. A possible way to estimate T is to listen to the channel and count the number of unique IP source addresses heard. In this way, the attacker can determine how many nodes send packets upward, and can estimate the depth of the subtree.

We now discuss the length of the OFF period. When the OFF period starts, the network *rapidly* converges to the new topology, because "count-to-infinity" plays a role only when links/nodes fail, not when they recover. Thus, the attacker may use one of the following two strategies. The first strategy is to choose the length of the OFF period randomly. This will make the detection of the attack very difficult, but it will allow the network to function normally immediately after the convergence to the new topology (with all the links active). The second strategy is to minimize the OFF period, namely, to wait until the network converges and then to start jamming again. In RPL, this waiting time depends on how fast a node broadcasts a DIO message after it is informed that a new link is up. As a heuristic, the attacker can simply wait a few seconds after the OFF period starts, and then to start jamming again.

VI. Simulation Study

In this section we present simulation results for the proposed attack. Our node placement model is based on the Node Placement Algorithm for Realistic Topologies (NPART) [12], which creates high quality wireless multi-hop networks that share many graph features with real networks. The authors of [12] considered almost 1,500 topological samples from real networks for designing their algorithm. The algorithm also ensures that each created topology is connected.

For the sake of our simulation study, we classify the nodes affected by the attack into three classes:

- Class-A nodes are those that remain disconnected during the entire ON period. This may happen if a node loses its parent but does not find a new parent during step 2 and step 6 of Algorithm 1. A node may lose its parent either because the link to the parent is directly affected by the jamming, or because its parent dismisses all its children during step 4 of Algorithm 1.
- Class-B nodes are those that lose their previous parent, but are able to find a new parent during the ON period, either in step 2 or step 6 of Algorithm 1. These nodes lose upstream and downstream messages during the process of switching their parents. Moreover, they impose a lot of management burden due to the need to associate with a new parent, and maybe to register again at the gateway.
- Class-C nodes are those that do not leave their parent, but one of their ancestors changes its parent in step 2 of Algorithm 1 without dismissing its children. Consequently, the path between such nodes and their gateways is modified, and these nodes are likely to lose most of their upstream and downstream messages.

Each point in our graphs is calculated as the average of 5,000 independent simulation instances conducted in the following way:

- Draw 20 different topologies using NPART, each with 250 nodes
- For each of the 20 topologies do
  - Repeat 250 times:
    1) Choose the locations of gateways randomly
    2) Choose the location of the attack randomly
    3) Run simulation
- Calculate the average of all 5,000 instances

The impact of the attack might be reduced by installing more gateways. This would allow a smaller average size for each tree and shorter average routing paths. Thus, fewer nodes are likely to be affected by an attack. However, installing a gateway is costly because it requires physical connectivity to the backbone. Because of the trade-off between cost and performance involved in adding more gateways, the x-axis in most of our graphs represents the number of gateways in the network.

We consider the periodicity of the DIO messages, i.e., the time during which every parent sends one such message, as 1 time unit. Each simulated attack consists of 200 time units of the ON period, during which the jamming is on, followed by 400 time units of OFF period, during which jamming is off. We choose the ON period length such that it maximizes the damage proposed in Section V.

In addition to the number of gateways in the network, the impact of the attack depends on the following two important parameters:

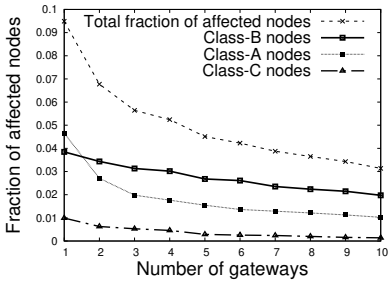
1) The network density, as reflected by the average node degree.
2) The jamming power.

To cover many possibilities, we use 4 combinations of network density and jamming power, each representing a different scenario:

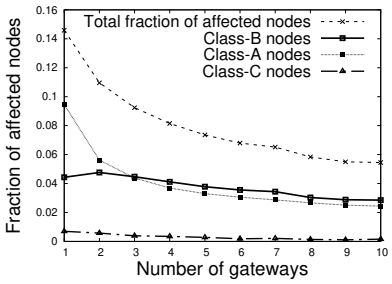
	Sparse network average node degree ∼4	**Dense network average node degree ∼14**
Low jamming 0.25*TP	Scenario-1	Scenario-3
High jamming 0.5*TP	Scenario-2	Scenario-4

In this table, TP refers to the transmission power of a legitimate node. The average degree of a node in a network, i.e., the number of neighbors it has, defines the network density. A neighbor of node v is a node that can hear the DIOs of v and can therefore consider v as a parent. Due to lack of space, in some graphs we omit the results of Scenario-4.

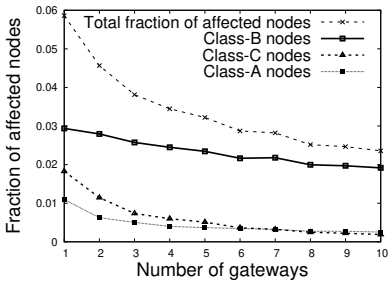
Figure 6 shows the fraction of affected nodes from each class, and the total number of affected nodes, as a function of the number of gateways for the various scenarios. The graphs clearly show that the fraction of affected nodes decreases sharply when the number of gateways varies from 1 to 8. This indicates that increasing the number of gateways is important not only for the performance of the network in general, but also as a remedy for the proposed attack. For Scenario-1 (Figure 6(a)), we see that despite the random location of the attack and the light jamming (it takes down only two wireless links on the average), almost 40% of these nodes are from Class-A. Recall that these nodes are completely disconnected from any gateway during the entire ON period.



(a) Scenario-1



(b) Scenario-2



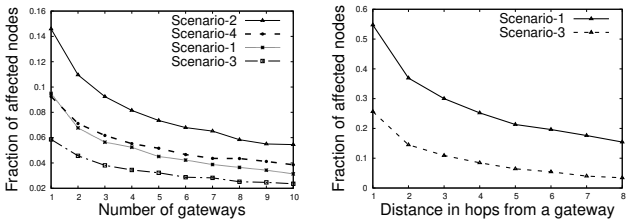
(c) Scenario-3

Fig. 6: The fraction of affected nodes in various scenarios.

When comparing Figure 6(a) to Figure 6(b), we see that the number of affected nodes increases by 50%: e.g., from 10% to 15% for 1 gateway, and from 5% to 8% for 4 gateways. We can also see that the number of Class-A and Class-B nodes is closer in Scenario-2 than in Scenario-1. This is because when the jamming power increases, more nodes are likely to remain without an alternative parent.

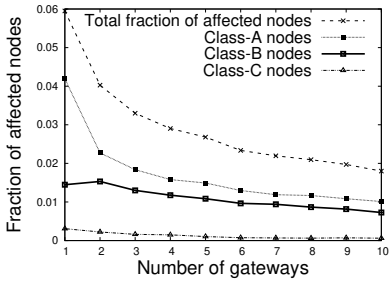
When the network density increases, the impact of the attack decreases. This is evident from comparing the results of Scenario-1 to those of Scenario-3. We can also see that this decrease is mainly attributed to Class-A nodes. This is because when the network is denser, each node has more paths towards the gateways, and is less likely to have nodes that cannot find an alternative parent.

Figure 7(a) compares the total fractions of affected nodes

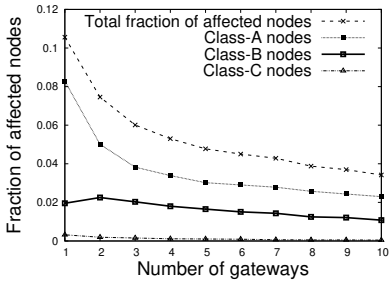


(a) (b)

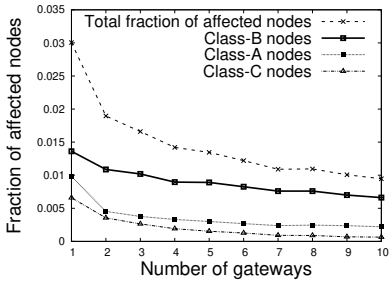
Fig. 7: (a) The total fraction of affected nodes; (b) The impact of the jamming location on the number of affected nodes.



(a) Scenario-1



(b) Scenario-2



(c) Scenario-3

Fig. 8: The fraction of affected nodes in the various scenarios when the remedy technique is used.

in the four scenarios. We can clearly see that increasing the jamming power and decreasing the network density both increase the impact of the attack. Thus, Scenario-2 yields the maximum number of affected nodes and Scenario-3 yields the minimum number.

In the previous graphs we chose a random location for the jamming. Nevertheless, we found that a significant fraction of the nodes are affected by the attack. **An attacker who knows where the gateways are located** can significantly increase the damage to the network (of course, jamming the area of the gateway would affect 100% of the nodes). This is evident from Figure 7(b), which considers a network with only one gateway, and shows the total fraction of affected nodes as a function of the distance between the jamming location and the gateway for Scenario-1 and Scenario-3. A distance of 1 indicates that the attack is 1-hop away from the gateway. We can see the dramatic increase in the number of affected nodes. In Scenario-1 for example, the increase is from 0.1 when the location is random to 0.55 when the distance is 1 or 0.35 when the distance is 2. In Scenario-3, the increase is from 0.06 when the location is random to 0.25 when the distance is 1.

Finally, Figure 8 shows simulation results when the remedy technique proposed in Section IV is implemented. The remedy works when the attack is repeated, by significantly reducing the number of Class-B and Class-C nodes. To produce these graphs, each attack is repeated 3 times, namely, 3 ON periods, each followed by an OFF period.

VII. Conclusions

This paper presented a novel attack on wireless mesh networks that use distance vector protocols, when the new IETF protocol RPL is used to demonstrate the effect of the attack. The proposed attack is as easy to conduct as a Phy/MAC layer attack yet is as effective as a Network layer attack. We also proposed a remedy technique and studied the attack and the remedy technique using simulations.

References

[1] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, 2012.

[2] G. Montenegro, C. Schumacher, and N. Kushalnagar, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC 4919, 2007.

[3] A. Mayzaud, R. Badonnel, and I. Chrisment, "A taxonomy of attacks in RPL-based internet of things," *International Journal of Network Security*, vol. 18, no. 3, 2016.

[4] D. R. Raymond and S. F. Midkiff, "Denial-of-service in wireless sensor networks: Attacks and defenses," *IEEE Pervasive Computing*, vol. 7, no. 1, 2008.

[5] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *"Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing"*. ACM, 2005.

[6] C. Dearlove and T. H. Clausen, "Multi-Topology Extension for the Optimized Link State Routing Protocol Version 2 (OLSRv2)," RFC 7722, 2015.

[7] K. Pelechrinis, M. Iliofotou, and S. V. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, 2011.

[8] L. Lazos and M. Krunz, "Selective jamming/dropping insider attacks in wireless mesh networks," *IEEE network*, vol. 25, no. 1, 2011.

[9] T. Tsao, R. Alexander, M. Dohler, V. Daza, A. Lozano, and M. Richardson, "A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs)," RFC 7416, 2015.

[10] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the RPL-based internet of things," *International Journal of Distributed Sensor Networks*, vol. 9, no. 8, 2013.

[11] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *"Ad Hoc Networks"*, vol. 1, no. 2, 2003.

[12] B. Milic and M. Malek, "NPART-node placement algorithm for realistic topologies in wireless multihop network simulation," in *"Proceedings of the 2nd International Conference on Simulation Tools and Techniques"*. ICST, 2009.