Addressing DODAG Inconsistency Attacks in RPL Networks

Anuj Sehgal[†], Anthéa Mayzaud[*], Rémi Badonnel[*], Isabelle Chrisment[*], Jürgen Schönwälder[†]
[†]Computer Science, Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany
Email: {s.anuj, j.schoenwaelder}@jacobs-university.de
[*]TELECOM Nancy, Université de Lorraine, LORIA UMR 7503, Inria Nancy-Grand Est, Villers-lès-Nancy, F-54600, France
Email: {anthea.mayzaud, remi.badonnel}@inria.fr, isabelle.chrisment@loria.fr

***Abstract*—RPL is a routing protocol for low-power and lossy constrained node networks. A malicious node can manipulate header options used by RPL to track DODAG inconsistencies, thereby causing denial of service attacks, increased control message overhead, and black-holes at the targeted node. RPL counteracts DODAG inconsistencies by using a fixed threshold, upon reaching which all subsequent packets with erroneous header options are ignored. However, the fixed threshold is arbitrary and does not resolve the black-hole issue either. To address this we present a mitigation strategy that allows nodes to dynamically adapt against a DODAG inconsistency attack. We also present the forced black-hole attack problem and present a solution that can be used to mitigate it. Results from our experiments show that our proposed approach mitigates these attacks without any significant overhead.**

I. Introduction

The Routing Protocol for Low-power Lossy Networks (RPL) was designed by the IETF RoLL working group, with capabilities of resource constrained nodes in mind [1]. This protocol is expected to form the basis of many Internet of Things (IoT) applications and is also useful in the wireless sensor networks (WSN) area.

RPL includes security mechanisms that can be used to ensure integrity and confidentiality of messages, however, important features like key-management are left out by the current standard [2]. Furthermore, cryptographic algorithms are known to occupy the most memory and take many CPU cycles, thereby greatly affecting the performance of constrained devices [3] likely to be used in IoT and WSN applications. Current RPL implementations, as such, do not enable secure operation modes [4]. This leaves RPL open to multiple attacks wherein a malicious node can manipulate contents of a packet to adversely affect the network.

One such attack is the DODAG inconsistency attack, which entails a malicious node manipulating the RPL IPv6 header options [5] used to keep track of DODAG inconsistencies in order to force the target to drop packets. This can lead to denial-of-service and increase in control overhead, which impacts limited energy reserves of constrained devices. A malicious node can use a DODAG inconsistency attack to modify all packets it forwards such that the next-hop node always drops them. This leads to the creation of a black-hole, which is hard to detect and counteract. Without addressing the black-hole scenario, RPL makes use of a fixed threshold to counteract DODAG inconsistency attacks. Once a node receives a certain number of packets with the appropriate RPL IPv6 header options it ignores all such future messages. The standard proposes a value of 20 for this threshold, however, does not provide any reasoning as to why this value is recommended.

Our contribution is an analysis of the DODAG inconsistency attack to obtain a better understanding of its consequences. Based on this we propose a dynamic threshold to mitigate effects of such attacks. We also study the black-hole scenario and develop a strategy to counter it. Our experimental results show that our proposed approach can lead to significant improvement.

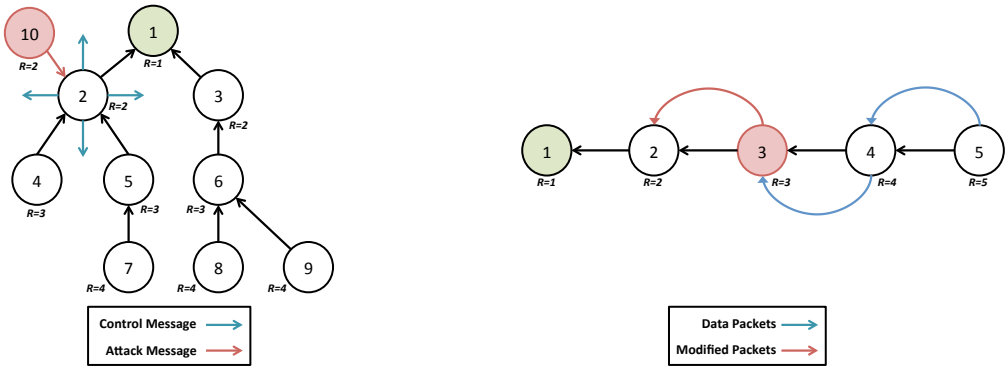
The paper is organized as follows. In Section 2 an overview of relevant related work is provided. This is followed by an overview of the RPL protocol in Section 3. The DODAG inconsistency attacks are presented in Section 4, along with proposed strategies that can be used to mitigate them in Section 5. An evaluation of the DODAG inconsistency mitigation approach follows in Section 6, before we draw conclusions in Section 7.

II. Related Work

While the study of RPL security is relatively new, the wireless sensor networks (WSNs) community has investigated security in similar environments. The authors of [6] investigate trust to enhance the security of WSNs and also propose to extend their approach to RPL networks [7]. However, such an approach is not useful if malicious nodes perform DODAG inconsistency attacks, because they can easily remain undetected owing to the unaltered control messages they broadcast.

A framework proposed in [8] classifies a large variety of attacks according to the confidentiality, integrity and availability (CIA) model. General countermeasures are suggested to address each attack, however, mitigation and effect of DODAG inconsistency attacks is not covered.

In recent years, some studies on attacks in RPL networks have also been carried out. The authors in [9] explored black-hole attacks in RPL-based networks and highlighted specific measurable parameters to detect such threats. Other works such as [10], [11], [12] studied attacks mainly related to field packet modification resulting in topology modification or resource depletion, and proposed security techniques to prevent them. However, none of them consider the situation wherein a malicious node may cause black-holes to be formed at targeted nodes; nor are straightforward DODAG inconsistency attacks discussed.



(a) (b)

Fig. 1. DODAG inconsistency attack scenarios. (a) The attacker, node 10, targets node 2 by sending packets with the 'R' flag. (b) The attacker, node 3, modifies packets received from its descendents to contain the 'R' flag before forwarding them. Packets with the 'R' flag are dropped by the receiving node and cause a reset of the trickle timer governing control message transmission, which increases network overhead. (R represents node's rank)

III. The RPL Protocol

Using RPL in a network leads to the formation of a loop-free tree like topology termed a DODAG, wherein nodes are organized into a hierarchical structure with a root, children, grandchildren and so on. RPL can use multiple objective functions to optimize the topology based on a set of goals, e.g., energy conservation, reduced hop count or link-quality. Each network can have multiple DODAGs, each optimized with its own objective function, root and appropriate topology. RPL can also run multiple instances within a network, which leads to each instance having its own set of DODAGs [1]. A node may be a member of only one single DODAG in an instance at any given point in time, but it may join multiple instances.

Formation and maintenance of the RPL DODAG are carried out using control messages. These are: (1) DAG Information Solicitation (DIS), (2) DODAG Information Object (DIO) and (3) Destination Advertisement Object (DAO). A new node may join an existing network by broadcasting a DIS message in order to solicit DIO messages, which carry information about the DODAG such as node ID and objective code point. A node can also wait to receive DIO messages from one of its neighbors, which are periodically broadcast. The periodicity is determined via the use of the trickle algorithm [13].

Upon receiving a DIO message the node calculates its rank by using the objective code point specified in this message. In case DIO messages are received from multiple nodes, the sender that results in the best rank calculated is chosen as the parent and informed of the decision. To build downwards routes a node must send the DAO message, containing routable prefixes, up the tree [1]. As the message propagates upwards, the prefixes are aggregated and downward routes become available to parents.

To avoid possible loops, RPL utilizes two basic rules. Any messages traveling downwards in the DODAG, but having originated at a descendant node, are ignored. Also, nodes can normally only change their parents and rank in case of upwards movement. Downwards movement is strictly prohibited, unless it occurs during loop avoidance measures or when a new version of the DODAG is created by the root.

For situations when a loop might occur, RPL provides the *data path validation* mechanism to detect and repair rank related DODAG inconsistencies. This mechanism works by carrying the following flags in the RPL IPv6 header options [5] of multi-hop data packets:

- The *'O' flag* — indicates the expected direction of a packet. When set, the packet is intended for a descendant. Otherwise it is intended for a parent, towards the DODAG root.
- The *'R' flag* — indicates that a rank error was detected by a node forwarding the packet. A mismatch between the direction indicated by the 'O' flag and the rank of sending/forwarding node causes the flag to be set.

The 'R' flag is used to repair this problem by setting it, in case it was not set previously, and forwarding the packet. Upon receiving a packet with the 'R' flag already set, the packet is discarded and the trickle timer used by RPL is reset [13].

IV. DODAG Inconsistency Attacks

Normally the data path validation mechanism is used to improve reliability of RPL, however, it can be used by a malicious node to attack a network as well. These DODAG inconsistency attacks can either be used to manipulate a targeted node, or a malicious node in the routing path can manipulate packet contents as well.

A. Direct Attack Scenario

A malicious node, part of an RPL network, can directly attack its parent by sending data packets that have the 'O' and 'R' flags set. Since packets with the 'O' flag are intended for descendent nodes, the receiving parent detects a DODAG inconsistency. If the 'R' flag is also set, which is the case during the attack, the received packet is dropped and the trickle timer is reset.

Trickle timer reset causes control messages to be broadcast more frequently, thereby causing local instability in the network. This increases the overhead of maintaining the RPL DODAG, reduces channel availability and can increase energy

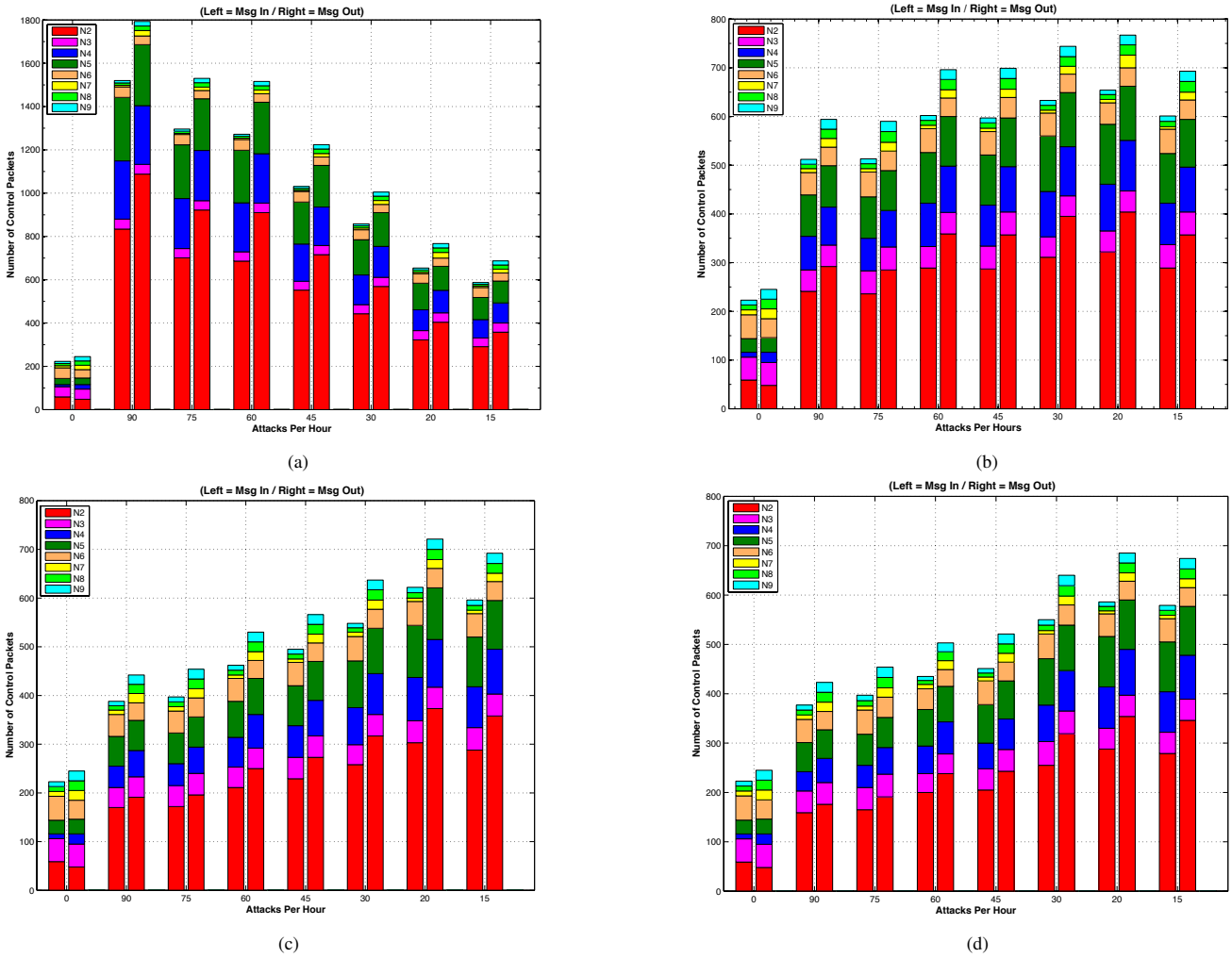


Fig. 2. Total control message overhead experienced by a network when (a) no mitigation strategy, (b) the default mitigation strategy, adaptive threshold (c) $\gamma = 20$ and (d) $\gamma = 25$ is used. ($N2 \ldots N9$ represent nodes $2 \ldots 9$ in Fig. 1(a))

consumption leading to reduced network lifetime. Since nodes in RPL networks are likely to be resource constrained, they are unlikely to support multi-tasking or large packet buffers. As such, time spent on processing malicious packets could lead to loss of genuine ones.

An RPL network where this scenario plays out can be seen in Figure 1(a). In this case, a stable network topology of ten nodes is formed using RPL. At some point, node 10 assumes the role of an attacker by sending messages, with the 'O' and 'R' flags set, to node 2, its parent. Node 2 resets its trickle timer, thereby flooding its neighborhood with control messages and effecting nodes 4 and 5 as well.

B. Packet Manipulation Scenario

In this case, the malicious node modifies packets it forwards such that the 'O' flag represents the incorrect direction and the 'R' flag is set. When forwarded, the receiving node will detect a DODAG inconsistency and drop the packet. As a result, the malicious node succeeds in forming a black-hole at the next-hop node.

This forced black-hole is difficult for nodes originating the message to detect because the packet is not dropped by their next-hop, but by a node that is at least two hops away. If the malicious node itself were to drop the packets, its children could detect this by enabling promiscuous mode [9], which causes packets addressed to nodes in the mode to be dropped. This makes it preferable for the attacker to force another node to drop the packets. Furthermore, if the control packets originating from the malicious node are normal, then the attack cannot be discovered. In this scenario, not only does the delivery ratio reduce, but the control overhead of RPL increases, availability of the channel decreases and the energy consumption may also increase.

For example, in the DODAG depicted by Figure 1(b) node 3 is the malicious node. Before forwarding data packets from its descendants, nodes 4 and 5, it modifies them such that the 'O' and 'R' flags are set. Node 2 drops them, thereby becoming akin to a black-hole. This causes the delivery ratio for descendants of node 3 to be severely harmed. Node 2 also resets its trickle timer causing an increase in overhead as well.

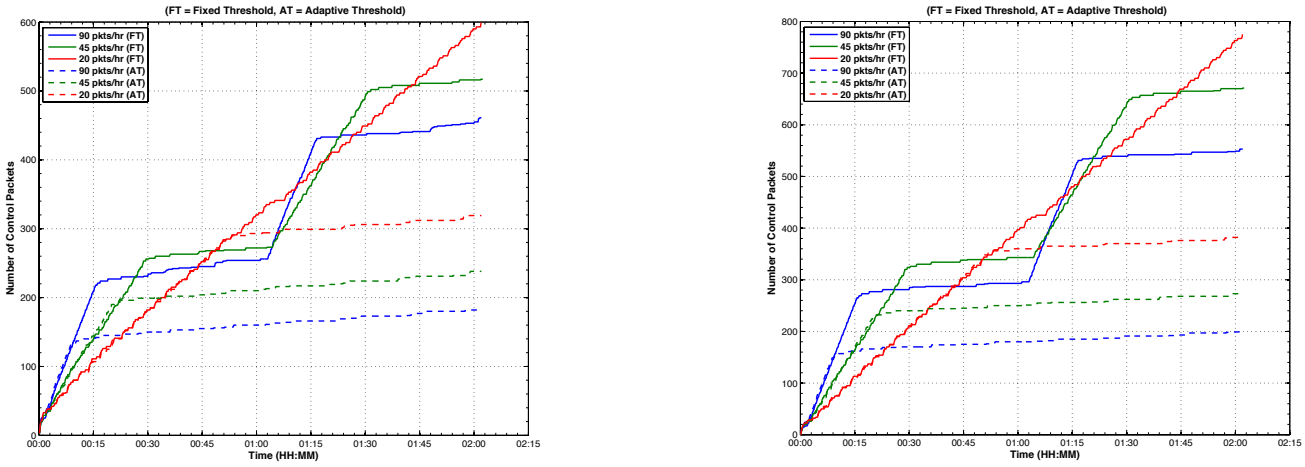


Fig. 3. Time-line of (a) incoming and (b) outgoing message overhead experienced by the attacked, i.e., node 2 in Fig. 1, when the rate of attack is varied from 20 to 90 packets per hour. ($\gamma = 25$)

V. Mitigation Approach

The default DODAG inconsistency attack mitigation strategy of RPL can be seen in Algorithm 1, where i is a node within the graph with a rank of r_i. M represents a packet received by node i from a neighbor j with rank r_j. O and R represent the 'O' and 'R' flags present in M. The variable $count$ is initialized to 0 and λ is a constant set to 20.

Upon receiving a packet with an inconsistency indicating 'O' flag, and the 'R' flag also set, the node drops the packet and resets its trickle timer. To limit the effects of an attack, the number of trickle timer resets is limited to the recommended constant $\lambda = 20$ [5]. Upon reaching this threshold, malformed packets are dropped but the trickle timer is not reset. $count$ is reset every hour, allowing attackers to once again have a higher impact.

This approach can limit the effects of a DODAG inconsistency attack, but $\lambda = 20$ is recommended arbitrarily. No reasoning is provided to justify this choice or how performance could be improved in case of varying attack scenarios. Packet manipulation attacks cannot be mitigated by this approach either. As such, we develop an adaptive threshold mechanism that can be useful in countering multiple attack patterns and mitigating packet manipulation.

A. Adaptive Threshold

In order to take into account the actual state of the network and react to varying attack patterns we propose utilizing an adaptive threshold to determine when to stop resetting the trickle timer. A function $\lambda(r)$ is used, instead of a constant λ, which takes the following form:

$$\lambda(r) = \lfloor \alpha + \beta \cdot e^{1-\gamma \cdot r} \rfloor \tag{1}$$

where,

$$r = \frac{E_{pkt}}{D_{pkt}}, \quad \alpha = 5, \quad \beta = \frac{15}{e}$$

Algorithm 1 The default DODAG inconsistency mitigation strategy of a node.

```
if (O = 1 and r_i < r_j) or (O = 0 and r_i > r_j) then
    if R = 1 then
        count++
        drop(M)
        if count < λ then
            reset_trickle_timer()
        end if
    end if
end if
```

E_{pkts} is the number of received data packets with the 'R' flag set and D_{pkt} represents normal forwarded data packets. To allow comparison with the default strategy, the value of β was chosen such that the default $\lambda(r) = 20$. α is an asymptote to ensure that threshold never reaches 0. This ensures that data packet validation is not disabled upon encountering the first packet with an 'R' flag, but only when the situation is deemed an attack.

Since γ has an impact on the threshold's rate of change, a value is not chosen here. In general, a larger value for γ leads to a smaller threshold being reached quicker. The effect of choosing different values for γ is discussed in Section VI-C1.

The adaptive threshold causes $\lambda(r)$ to change based on network conditions. If an attacker is aggressive, the threshold drops quickly and increases slowly once the attacks stop. Unlike with the fixed threshold, $count$ is not reset every hour, but rather allowed to increase in the absence of attacks. As such, not only is this approach likely to be better than a fixed threshold within the first hour of an attack, but it should perform significantly better against long running attacks.

B. Countering Packet Manipulation

Rather than only increasing control overhead by targeting nodes with malformed packets, a malicious node could manipulate forwarded packet header contents by inappropriately

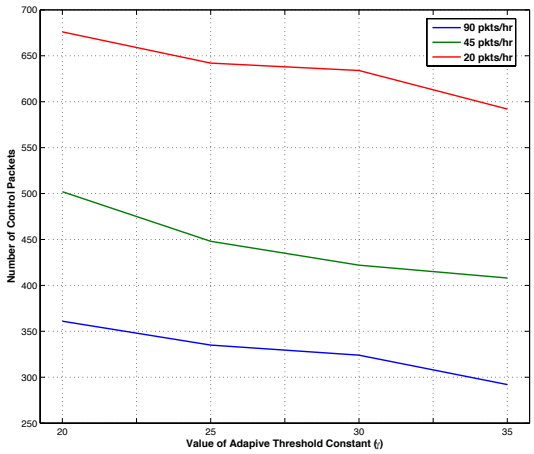


Fig. 4. The effect of different values for γ on the total control packet overhead experienced by the attacked node, i.e., node 2 in Fig. 1(a), under multiple attack patterns.

Algorithm 2 The adaptive DODAG inconsistency mitigation, along with the packet manipulation mitigation strategy.

```
if (O = 1 and r_i < r_j) or (O = 0 and r_i > r_j) then
    if R = 1 then
        if count < λ(r) then
            count++
            drop(M)
            reset_trickle_timer()
        else if λ(r) ≤ α then
            O ← 0
            R ← 0
            forward(M)
        end if
    end if
end if
```

setting the 'O' and 'R' flags. This forces the next-hop node to always drop these forwarded data packets, thereby forming a black-hole like scenario, where the packets of the attacker's descendants never reach the sink.

To counter this form of DODAG inconsistency attack we reuse the adaptive threshold. Nodes behave normally until the number of messages indicating inconsistency becomes greater than the threshold obtained from Equation 1. This indicates situation either an attack against the node, or malfunction of the node forwarding such packets. To rectify the situation, the clears the 'O' and 'R' flags before forwarding the packets normally. The complete packet manipulation mitigation strategy, combined with adaptive threshold mitigation approach, can be seen in Algorithm 2.

Since no additional resources are used by this approach, the cost of protecting the network against black-hole scenarios is quite low.

VI. Evaluation

The Contiki 2.6 operating system was used for evaluating the DODAG consistency attacks since it provides an RPL implementation that works on multiple platforms. However,

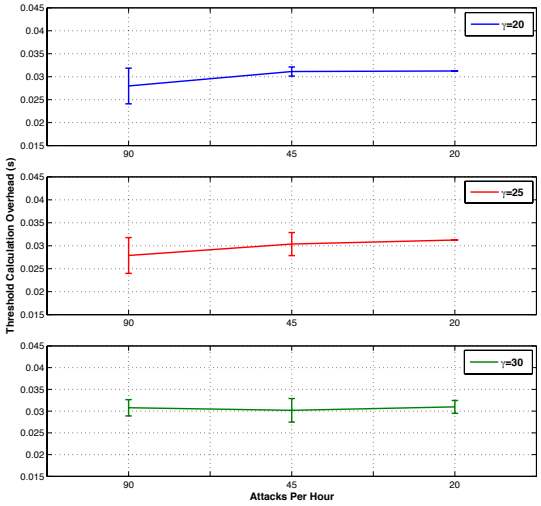


Fig. 5. The computation overhead of the adaptive threshold. (Clock Frequency 1 MHz)

the default DODAG inconsistency attack mitigation approach is not supported by Contiki. While packets with the 'R' flag are dropped, the trickle timers are not reset. As such, the implementation was modified to enable the use of a threshold to determine when the trickle timers should be reset.

The TelosB platform was used for development since it's computational resources are sufficient to function as an RPL router. The compiled binary for a TelosB was used in the COOJA simulator provided by Contiki. This allow us to rapidly test multiple scenarios, and also provides a method of testing our approach without the lossy IEEE 802.15.4 channel causing packets to disappear, thereby evaluating our approach under ideal conditions. Furthermore, utilizing COOJA is quite close to actual hardware since it uses MSPSim to emulate the architecture and performance of the MSP430F1611 microcontroller, utilized by the TelosB.

A. Overhead Without Mitigation

To evaluate the control message overhead caused by DODAG inconsistency attacks, the topology shown in Figure 1(a) was setup in COOJA, with node 1, the DODAG root, acting as the sink. Other nodes were configured to send messages to the sink every six seconds. To avoid packet collisions and add a degree of irregularity to the transmission scenario, an additional back-off period of up to six seconds was added. The RPL implementation was setup to ignore the threshold and always reset trickle timers.

The attacker, i.e., node 10 in Figure 1(a), was setup to periodically send packets, between 15 to 90 msgs/hr, with the 'O' and 'R' flags towards the sink. Each simulation was executed for one hour and the attacks began after 2 mins so as to allow the network to stabilize. A simulation with no attacks was also performed to obtain a baseline measurement for comparison.

More complex topologies and scenarios were not studied because the effects of such attacks are limited to the target and it's immediate neighborhood, except delivery ratio, which is

TABLE I. ENERGY MODEL FOR THE CC2420 RADIO AND MSP430F1611 MICROCONTROLLER OPERATING AT 1 MHZ ON THE TELOSB PLATFORM.

Operation	*Current*[1]	*Voltage*[1]	*Part*[1]
Transmit (T_x)	18.8 mA	2.2 V	CC2420
Receive (R_x)	17.4 mA	2.2 V	CC2420
Processing	0.33 mA	2.2 V	MSP430F1611

impacted for all descendants of the targeted node. As such, our scenarios are carefully designed to study the possible effects of such attacks.

Results of this experiment can be seen in Figure 2(a). As expected, the more aggressive the attacker, the higher the overall message overhead in the network. Node 2 experiences the largest increase in control messages since it is directly targeted. Nodes 4 and 5 also experience an increase due to being direct descendants of node 2. The most aggressive attacker tested can increase overhead at the targeted node by over 2000%. As such, it is very important to mitigate DODAG inconsistency attacks.

B. Default Mitigation

The setup from Section VI-A was used to evaluate the default RPL mitigation approach, however, with the fixed threshold now enabled. The control message overhead in this experiment can be seen in Figure 2(b).

When using a threshold to mitigate DODAG inconsistency attacks the worst case overhead reduces by nearly 60%. Aggressive attacks cause the threshold to be reached faster, causing lower overhead in these scenarios. As such, the best strategy for an attack is to remain as close to the threshold as possible, as is evident from the 20 attacks/hr scenario. Since the counter indicating DODAG inconsistencies is reset every hour, by remaining close to the fixed threshold the attacker can do maximum damage and the nodes have no recourse.

While a threshold is useful, adapting it to network conditions would not allow an attacker to keep just below a known value and neither would counter resets give the attacker another window of opportunity.

C. Adaptive Threshold Mitigation

The adaptive threshold mitigation approach was also evaluated using the same setup as in Section VI-A. An implementation of Equation 1 replaced the fixed threshold.

By comparing Figures 2(b), (c) and (d), it is clear that the adaptive threshold is more successful in reducing control message overhead than a fixed threshold. An aggressive attack causes the adaptive threshold to reduce rapidly, thereby limiting the impact of the attack. This results in slower attacks being the best strategy. In Figures 2(c) and (d), 20 attacks/hr is the best strategy because the values of α and β were chosen to model the recommended default of 20 in a steady state. However, if the values of these coefficients are changed, so will the periodicity of the optimal attack pattern. The adaptive threshold is between 8% ($\gamma = 20$) to 13% ($\gamma = 25$) better, even in the worst case scenarios. The results for other attack periodicities are better.

[1]Operating values were obtained from relevant data-sheets.

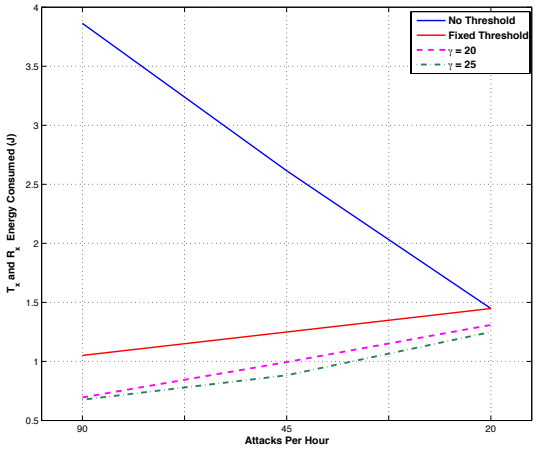


Fig. 6. The energy consumption caused by the control message overhead resulting from different attack patterns at the attacked node, i.e., node 2 from Fig. 1(a).

Since the value of *count* is not reset every hour the attacker does not have a future window of opportunity for causing increased damage. The threshold increases automatically in the absence of an attack, and as such the adaptive approach mitigates long running attacks even better. Results from a two hour long experiment can be seen in Figure 3. Only results from the directly attacked, node 2, are depicted.

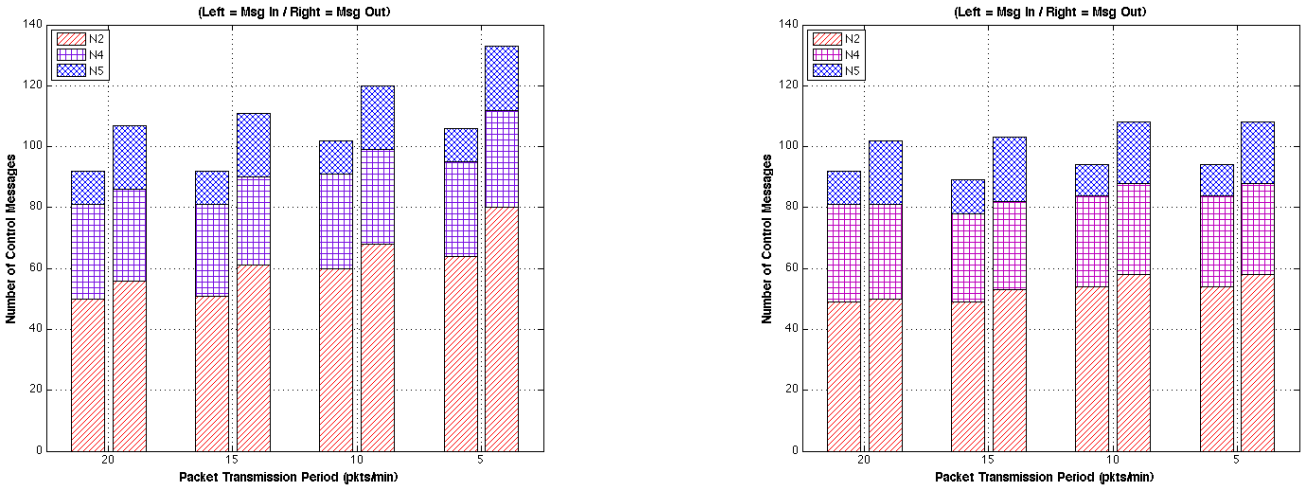
When using a fixed threshold the control messages increase quickly till the threshold is encountered. They then grow at a slow rate, following the trickle timer pattern until the 1 hr mark, when the counter is reset. Once again, the control messages increase quickly until the threshold is encountered. This behavior causes a high control message overhead. The only exception is the period of 20 attacks per hour, because at this rate the threshold is never encountered, thereby causing the largest overhead growth.

On the other hand, the limit is reached much faster with the adaptive threshold, due to the exponential growth of the function. Coupled with a non-reseting counter, this leads to between 45%-55% savings in the control message overhead.

1) The effect of γ: Given the same attack periodicity, the value of γ in Equation 1 determines the rate at which the threshold changes. Experiments were run with $20 \leq \gamma \leq 25$ to gain insights into it's impact. As can be seen in Figure 4, by increasing the value of γ the overhead, even in the case of the most efficient attacker, can be further reduced by around 10%. This means that higher values of γ are able to offer more significant savings in the overhead.

As such, the temptation to use a larger value for γ might be high, but it is important to keep in mind that a rapidly reducing threshold might also impact the repair of genuine loop conditions. It would, therefore, be best to remain conservative in choosing a value for γ.

2) Computational overhead: Since Equation 1 replaces a constant threshold, the complexity of which is $\mathcal{O}(1)$, it is important to also quantify the impact using an exponential function has upon the overall computation costs. Measuring this impact is even more important since this approach is



(a) (b)

Fig. 7. The total control message overhead experienced by a network operating with (a) a fixed threshold and (b) dynamic threshold ($\gamma = 25$), when a black-hole is formed at node 2 from Fig. 1(b). ($N2 \ldots N5$ represent nodes $2 \ldots 5$ in Fig. 1(b))

expected to be used on resource constrained devices with limited computing abilities. The two arguments that effect the number of instructions required to calculate the value for the exponential part of Equation 1 are r and γ.

The value of r is dependent upon the attack pattern, and as such, these should be varied in order to quantify the computational cost of calculating the adaptive threshold. The time taken to calculate the threshold is a good measure of complexity because it is directly related to the processor architecture and number of instructions.

As such, while running the aforementioned experiments with different γ values, the time taken to calculate the threshold was also obtained. Figure 5 shows the average computation time required to obtain the threshold for multiple attack patterns and γ values. While at first glance the variances observed might seem to be large, it is important to note that the MSP430F1611 microcontroller on the TelosB has a timing precision of 7.8125 ms[2]. The largest observed variance, across all scenarios, is 3.8919 ms, which is less than half of the precision provided by the microcontroller. Being within the precision limit this variance is negligible and it could be said that calculating the threshold adds a maximum computational overhead of approximately 31.25 ms.

Using the `msp430-size` tool it was determined that a node based on the fixed threshold approach occupies 41.96 kB (87.4%) of Flash memory and 8.63 kB (86.3%) of statically allocated RAM. On the other hand, the adaptive threshold approach requires 45.61 kB (95%) of Flash memory and 8.62 kB (86.2%) of statically allocated RAM. The amount of Flash required grows by about 8%, but there is almost no change in the amount of statically allocated RAM required.

From the measured values, it can be said that the overall impact of the adaptive threshold approach on computational overhead is quite minimal, especially when taking the gains into consideration.

[2]Obtained from MSP430F1611 data-sheet.

3) Energy consumption: Besides the extra processing time on the constrained devices and the reduction in channel availability, the increased message overhead can also cause an increase in the energy utilized by the nodes. Since many deployments of networks utilizing RPL are expected to have nodes that work on battery power, it is essential to evaluate whether using the adaptive threshold provides a net gain or loss in energy consumption.

From the energy model shown in Table I it can be determined that the amount of energy taken up by the adaptive threshold computation is approximately 22.68 μJ, which is the amount of energy required to keep the processor running for the computation time of 31.25 ms. This indicates that the energy required for this computation is small enough to not have any impact upon the overall operation of the network. This means that for frequencies of 20, 45 and 90 attacks per hour, the total energy spent over a period of one hour on computing the adaptive threshold is about 0.45 mJ, 1.02 mJ and 2.04 mJ respectively.

The energy consumption for the transmission and reception of all control messages, over a period of one hour, at the attacked node can be seen in Figure 6. Firstly, it is quite clear from these results that the energy consumed in calculation of the threshold is several orders of magnitude lesser than what is expended in transmitting and receiving control messages. As such, using the adaptive threshold reduces energy consumption and can lead to longer network lifetimes.

It is also worth noting that in case of aggressive attack scenarios, as much as 40% energy can be saved by using an adaptive threshold. Even in case of the most efficient attack period, about 10% energy can be saved at the targeted node.

D. Packet Manipulation Mitigation

To evaluate the effect of our packet manipulation mitigation approach, the topology shown in 1(b) was setup in COOJA, with node 1, the DODAG root, acting as the sink. All other

nodes, except the attacker were configured to send messages to the sink at rates varying from 5 to 20 packets per minute. The packet sending rate is varied, because the attacker, i.e., node 3, silently modifies the option headers of the packets it forwards, rather than originating a direct attack.

The nodes were configured to use the adaptive threshold for mitigating packet manipulation and the overhead caused by DODAG inconsistency attacks. Results in Figures 7(a) and (b) show that the adaptive threshold reduces overhead in the network. Compared to the default fixed threshold approach a reduction of up to 30% can be achieved.

However, the primary effect of packet manipulation attacks is not increased overhead, but to cause the next-hop node to drop all packets of the attacker's descendants. The emergence of this black-hole can severely impact the overall delivery ratio of packets, since none of the packets from the attacker's descendants will ever reach the sink. Without a black-hole mitigation approach the overall delivery ratio is only about 33%. This is because only packets from node 2 reach the sink, while the attacker forces node 2 to drop all packets sent by nodes 4 and 5. On the other hand with mitigation strategy employed the overall delivery ratio increases to just above 99%, because node 2 no longer drops packets from node 4 and 5 once the threshold is reached. This speaks strongly in favor of mitigating packet manipulation based DODAG inconsistency attacks via the dynamic threshold approach.

An analysis of the computational overhead associated with this approach is not necessary since no additional resources are used. The computational overhead remains the same as that presented in Section VI-C2.

VII. Future Work

Our adaptive threshold approach introduces configurable values, i.e., α, β and γ. Our experiments show that an increasing γ improves performance by reducing the threshold quickly, however, an optimal value of γ needs to be investigated. It might even be possible to dynamically scale γ based on network conditions. Similarly, optimal values for α and β must also still be calculated. The impact of varying these coefficients of Equation 1 on the success of identifying genuine loop conditions also needs to be studied. The impact of α, β and γ on mobile scenarios, where genuine loops can occur, must also be studied.

VIII. Conclusions

An analysis of the DODAG inconsistency attacks against RPL networks, showed that utilizing a threshold to mitigate such attacks is useful. We proposed an adaptive threshold to take changing network characteristics into account. We also discovered that packet manipulation inconsistency attacks can lead to the formation of a black-hole, and significantly impact the delivery ratio. An extension of our adaptive threshold approach was proposed to mitigate such attacks.

Through our experimental evaluation it becomes clear that the adaptive threshold is able to reduce the control message overhead, compared to fixed threshold, by up to 13% in short lived and 55% in long-lived networks. This leads to large reductions, i.e., between 10%-40%, in energy consumption, which extends the network lifetime when battery operated nodes are used. Our strategy is also effective against packet manipulation black-holes, since it is able to increase the delivery ratio of the effected network segments up to 99%.

The implementation and computational complexity costs of these approaches are minimal, thereby providing a good strategy for mitigating DODAG consistency attacks in RPL networks. Such attacks would not be possible with secure modes of RPL. However, due to resource constraints and lack of complete information secure modes of RPL are not implemented[3], making such a mitigation approach important.

Acknowledgments

This work was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Program.

References

[1] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, 'RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *IETF RFC 6550*, March 2012.

[2] S. Seeber, A. Sehgal, B. Stelte, G. D. Rodosek, and J. Schönwälder, 'Towards A Trust Computing Architecture for RPL in Cyber Physical Systems," in *IFIP/IEEE International Conference on Network and Service Management (CNSM)*, Zürich, Switzerland, October 2013.

[3] A. Sehgal, V. Perelman, S. Kuryla, and J. Schönwälder, 'Management of Resource Constrained Devices in the Internet of Things," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 144–149, 2012.

[4] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, 'Evaluating the Performance of RPL and 6LoWPAN in TinyOS," in *Workshop on Extending the Internet to Low Power and Lossy Networks (IP+SN)*, Chicago, IL, USA, April 2011.

[5] J. Hui and J. Vasseur, 'The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams," IETF RFC 6553, Mar. 2012.

[6] H.-C. Leligou, P. Trakadas, S. Maniatis, P. Karkazis, and T. Zahariadis, 'Combining Trust with Location Information for Routing in Wireless Sensor Networks," *Wireless Communications and Mobile Computing*, vol. 12, no. 12, pp. 1091–1103, 2012.

[7] P. Karkazis, P. Trakadas, T. Zahariadis, A. Hatziefremidis, and H. Leligou, 'RPL Modeling in J-Sim Platform," in *Ninth International Conference on Networked Sensing Systems (INSS)*. Antwerp, Belgium: IEEE, June 2012, pp. 1–2.

[8] T. Tsao, R. Alexander, M. Dohler, V. Daza, and A. Lozano, 'A Security Framework for Routing over Low Power and Lossy Networks," *IETF I-D <draft-ietf-roll-security-framework-07>*, July 2012.

[9] K. Chugh, L. Aboubaker, and J. Loo, 'Case Study of a Black Hole Attack on LoWPAN-RPL," in *Proc. of the Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*. Rome, Italy: IARIA, August 2012, pp. 157–162.

[10] A. Dvir, T. Holczer, and L. Buttyan, 'VeRA - Version Number and Rank Authentication in RPL," in *8th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, Hangzhou, China, October 2011, pp. 709–714.

[11] A. Le, J. Loo, Y. Luo, and A. Lasebae, 'Specification-based IDS for Securing RPL from Topology Attacks," in *IFIP Wireless Days (WD)*, Niagara Falls, Canada, October 2011, pp. 1–3.

[12] K. Weekly and K. Pister, 'Evaluating Sinkhole Defense Techniques in RPL Networks," in *20th IEEE International Conference on Network Protocols (ICNP)*, Austin, TX, USA, November 2012, pp. 1–6.

[13] P. A. Levis, N. Patel, D. Culler, and S. Shenker, 'Trickle: A Self Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *1st Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, USA, March 2004.

[3] ContikiRPL, TinyRPL, SimpleRPL for Linux and RIOT OS were surveyed.