A Trust Based Distributed Intrusion Detection Mechanism for Internet of Things

Zeeshan Ali Khan and Peter Herrmann
Department of Telematics
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
Email: {zakhan, herrmann}@ntnu.no

***Abstract*—Many Internet of Things (IoT) networks comprise tiny devices with limited processing power and tight energy restrictions. These limitations make it difficult to use established security mechanisms protecting the devices against malicious attacks. This holds particularly for Intrusion Detection Systems (IDS) that help to detect various net-based attacks but often demand significant network and computing resources. In this article, we design and evaluate some IDS mechanisms for IoT Networks that are suited to small devices. They use a trust management mechanism that allows devices to manage reputation information about their neighbors. This mechanism makes it possible to single out maliciously behaving units in a processing and energy-friendly way. The approach is explained in the context of the healthcare domain.**

I. Introduction and Related Work

According to most predictions, several billions of "things" will be connected with the Internet in the coming few years (see, e.g., [1]). Many of these connected devices will be very small and cheap such that they can be placed wherever they can be of avail. They form an Internet of Things (IoT) that will lead to a completely new set of applications revolutionizing the use of ICT technology in various areas of our living. The interconnectedness of the IoT networks, however, poses a significant risk since the systems will be subject to malicious attacks. An example are denial of service attacks precluding the devices from communicating with other stations. Therefore, security issues must be considered for the engineering and deployment of IoT networks [2]. Further, the applied security mechanisms have to address the limitations of many devices in terms of memory, power, and bandwidth [3]. Otherwise, the devices can be depleted within a short period of time.

Our approach centers on Intrusion Detection Systems (IDS) that help to retrieve various forms of network attacks for distributed systems, e.g., the denial of service attacks mentioned above [4]. IDS can be signature-based such that they check the network traffic for certain attack patterns. In contrast, anomaly-based IDS try to find abnormalities in the overall behavior which may be an indication for attacks. Thus, unlike the signature-based IDS they can also detect previously unknown attacks but are often subject to "false positives", i.e., reports about incidents that are not attacks. Of course, a good IDS shall minimize the number of false positives but, one the other hand, detect as many real attacks as possible. Unfortunately, many IDS tend to demand a lot of communication overhead to coordinate the different nodes in a network. Moreover, the comparison with known signatures as well as the analysis of anomalies often afford complex computations that may exceed the abilities of smaller devices.

Our solution concentrates on attacks that actively try to omit crucial communication between nodes. In ad-hoc networks, a way for that are insider attacks (see [5]) in which the intruder manages to compromise the routing capability of devices [6], [7]. To reduce the computing efforts for the small devices, we use a trust management technique. That enables IoT devices to build up a measurement about the trustworthiness of adjacent nodes in a resource-friendly way. For that, the neighbors are monitored and depending on positive and negative experiences, trust values are built. In addition, one can use special trust management policies which, for example, exclude certain nodes when their trust values display malicious behavior. Jøsang's Subjective Logic [8] provide a number of easily processable functions to maintain such trust values.

In the literature, there are a few articles that propose intrusion detection for IoT based networks. In [9], the authors investigated an IDS for IoT. In contrast to our solution, it is a centralized strategy and requires a lot of packets for detection of only few intruder nodes. Further, this approach is criticised by [10] who claim an extremely high number of false positives due to time inconsistency. In [11], a distributed mechanism for intrusion detection is proposed that, however, targets mainly mobile nodes. In contrast, we want to identify an architecture considering the nature of an IoT application as well as the topology and dynamics of the used network.

II. IoT Network Architecture for a Healthcare System

An important IoT application domain is Medical Care. Various medical imaging and diagnostic devices are in the market that facilitate elderly care, fitness care, and a better treatment of chronic diseases. In particular, IoT networks enable real-time monitoring of patients which reduces costs and improves the quality for the patients [12]. In the following, we suppose an ad-hoc style IoT network in which nodes may forward received data. Some of the devices have a functionality as *border routers* that are able to communicate via the Internet with external healthcare servers that, amongst

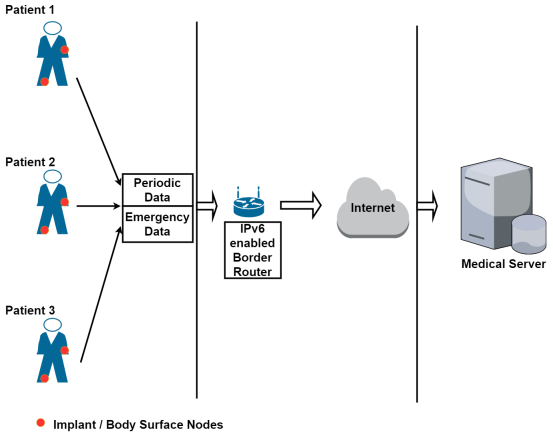

Fig. 1. IoT based Healthcare system

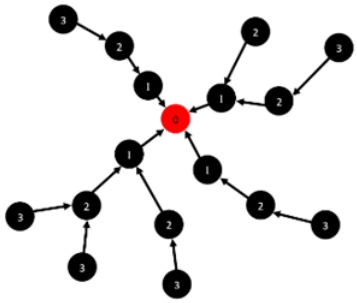

Fig. 2. DODAG Network

others, may access health records. The different phases of patient data transfer are shown in Figure 1 (see [13]). In the first phase, vital patient data such as temperature or blood pressure is sensed and forwarded via a sequence of devices until reaching a border router that supports the IPv6 protocol. In the second phase, the border router forwards the received data to the medical server using the Internet. The third phase is the collection of data and the initiation of remedial measures if necessary.

We assume that the devices sensing and forwarding the patient data in the first phase are those with limited energy and processing resources. Many of them will be worn by the patients. The lack of suitable security mechanisms make the devices targets for denial of service attacks which may have catastrophic consequences. Therefore, we need lightweight and resource-friendly solutions to protect these nodes without depleting them.

Before introducing our ideas to solve this problem, we first introduce the popular RPL protocol that is used to connect the devices. Thereafter we present the attack types against which the devices shall be safeguarded followed by some assumptions, we made about the network.

A. RPL Protocol

The *Routing Protocol for Low power and Lossy networks* (RPL) [14] is a distance vector routing protocol for small devices that uses a hierarchical topology called Destination Oriented Directed Acyclic Graph (DODAG) [15], [16]. A typical DODAG is depicted in Fig. 2. The nodes have a particular *rank*[1] indicating the number of hops to the root of the DODAG. Devices are provided with an Objective Function (OF) that allows them to select the best possible path to a node of the DODAG [9]. While there may be several DODAGs in a RPL-based network, each node may only join a single DODAG at any point of time. When a new node wants to attach a DODAG, it broadcasts special advertisement messages and selects those replying stations that have the lowest rank, as its parents.

Loops are avoided since all messages are only forwarded between parents and children [17]. The dynamicity of the protocol may, however, lead to inconsistencies in the ranks which are corrected by a data path validation mechanism [18]. Further, nodes can disappear due to the lack of battery power or other technical issues. In that case, either a global or local repair mechanism is initiated. In a global rebuild, the entire DODAG is built again which is expressed by an incremented *version number* in the corresponding messages. Each receiving node compares its existing version to the one received from its parent and if it is higher than current one, it must ignore the current rank and initiate the procedure for joining the DODAG node anew. Unfortunately, the global rebuild is quite costly in terms of energy consumption. Therefore, some local rebuild mechanisms exist that can be used when a node disappears. These mechanisms allow a temporary routing of messages through neighbors of the same rank as well as the switching of parents [19].

B. Routing attacks against RPL networks

There are three main kinds of routing attacks in the literature that can be carried out on the RPL routing protocol:

- *Selective-Forwarding Attacks [5], [20]:* Malicious nodes can launch denial of service attacks by forwarding packets selectively, e.g., by sending only control but not data messages.
- *Sinkhole Attacks [5], [20]:* In this type of attack, a malicious node falsely claims a lower rank in order to let its neighbors select this node as their parent. Thus, the malicious node receives more traffic that can be assailed by Selective-Forwarding Attacks.
- *Version Number Attacks [17]:* Here, the malicious node changes the version number in their messages which will lead to a global rebuild that slows down the network and strains the energy consumption of its nodes. Further, during the rebuild, messages of both versions will be active at the same time which may lead to temporary inconsistencies and loops.

C. Amended Network Model

To realize the first step in our medical IoT network, we arrange the devices in a DODAG tree that uses the border

[1]When describing a message exchange, we call a node closer to the root *parent* and the adjacent partner *child*.

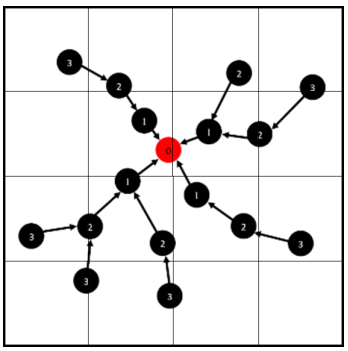

Fig. 3. Clustered DODAG Tree

router as its root. Further, in some of our algorithms described in Sect. III-D, we cluster the DODAG in order to reduce the number of messages to be transmitted. The clusters are based on the geographical location of the nodes in the network. One of the nodes in a cluster is allocated the role of a cluster-head that will take a particular role in the trust-based mechanism. Figure 3 shows an example of a clustered DODAG that is divided into 16 equally sized clusters.

III. Trust-based Intrusion Detection for IoT

In the following, we describe the different aspects of the trust management functions used to allow DODAG nodes build up trust relations with their neighbors that guide the routing of messages through the network (see also [21]). In particular, we introduce the trust management technique used followed by presenting the trust gaining mechanism carried out in the nodes. Afterwards, we discuss how the border router or a cluster head combines trust values in order to build a general reputation of the nodes. Finally, three algorithms addressing different layouts will be described.

A. Trust Evaluator

In computers, trust between entities can be represented by *trust values*. These values can be discrete, e.g., the different shaped and colored stars describing the reputation of sales partners in eBay, or continuous. A mature technique is the Subjective Logic [8]. Here, the trust values are so-called *opinion triangles* that not only refer to trust or distrust but also consider uncertainty about a trustee. An opinion triangle is represented by the three variables b (*belief*, i.e., trust), d (*disbelief*, i.e., distrust), and u (*uncertainty*). All variables are real numbers in the interval between 0 and 1, and their values must always add to 1.

The trust values can be computed from positive and negative experiences with a trustee by metrics like the following [22]:

$$b = \frac{p}{p+n+k} \qquad d = \frac{n}{p+n+k} \qquad u = \frac{k}{p+n+k}$$

The number of positive experiences are expressed by variable p and those from negative experiences by variable n. The constant k for which often the values 1 or 2 are used, determines how fast certainty about a trustee is built. This metric can be adapted by deducing older experiences with a forgetting factor such that more recent incidents are rated higher than older ones (see [23]).

For enabling the nodes to rate their neighbors according to this technique, we assume that they use transceivers that support idle mode listening of 1-hop neighbors data traffic. The trust evaluator functionality of a node can then listen to the transmission of its neighbors and rate them positively if they behave as expected according to the RPL protocol and negatively if they deviate from it. Let us assume that the nodes x, y, and z participate in a transmission. Since node y is a neighbor of x, it cannot only monitor the communication of y towards x but also that directed to z. Thus, x can determine whether a packet planned to run via y, z, and possibly other nodes to the border router, is indeed correctly forwarded by y after receiving it from x. If the distrust variable d in the trust value for y in x increases, x may reduce the communication via y. It can even avoid to send messages to y at all if d exceeds a certain threshold.

Trust values can be sent to the border router that may aggregate them to *reputation values*. If a bad reputation value indicates a node as a potential intruder, the border router removes it from the network and notifies the operators of the network.

B. Direct Trust of Neighboring Member Nodes

One defines *direct trust* as the kind of trust one has in the benevolence of a trustee while *recommendation trust* refers to trust that a party gives correct recommendations about a third one. Here, we discuss direct trust of a node x in another node y by monitoring messages sent by y. In particular, x performs three types of checks:

- *Forwarding Check:* In order to detect Selective-Forwarding Attacks (see Sect. II-B), node x sets its transceiver to idle listening mode after sending a packet to y. If x receives the forwarded message from y directed to z within a certain time interval, y seems to follow the RPL protocol correctly and x increases variable p in the trust value of y. If x does not receive the forwarded packet of y in time, it increases its n value indicating wrong behavior.
- *Ranking Check:* To detect Sinkhole Attacks, a node x checks if the packets of a parent resp. child y contain the correct rank of y and increases y's p or n values accordingly.
- *Version Number Check:* The version number update packets are received from the root node of a DODAG when it initiates a global rebuild. One should further expect that a node does not receive such a notification too long after its own children. Therefore, if a node x detects that its child y uses a new version number but does not receive a corresponding rebuild message from the root within a certain amount of time, it assumes that y commits a Version Number Attack. In consequence, x increases variable n in the trust value of y. This attack is considered as more serious than the other ones such

Algorithm 1 Trust Computing in a Node

while $True$ **do**
  if $packet\_sent$ **then**
    Store $packet$ in $send\_stack$
  end if
  if $packet\_received$ **then**
    if $packet\_transmitter \in (parents \cup children)$ **then**
      if $packet \in send\_stack$ **then**
        Remove $packet$ from $send\_stack$
        Increment value p of $packet\_transmitter$
      end if
      if $packet\_transmitter \in parents$ **then**
        if $packet\_hop\_rank = my\_rank - 1$ **then**
          Increment value p of $packet\_transmitter$
        else
          Increment value n of $packet\_transmitter$
        end if
      else
        if $packet\_hop\_rank = my\_rank + 1$ **then**
          Increment value p of $packet\_transmitter$
        else
          Increment value n of $packet\_transmitter$
        end if
        if $packet\_version > current\_version$ **then**
          Store $packet$ in $new\_version\_stack$
        end if
      end if
    end if
    if $packet$ is a rebuild message from the border router **then**
      Empty $new\_version\_stack$
    end if
  end if
  if Timeout of a $packet$ in $send\_stack$ **then**
    Remove $packet$ from $send\_stack$
    Increment value n of $packet\_transmitter$
  end if
  if Timeout of a $packet$ in $new\_version\_stack$ **then**
    Remove $packet$ from $new\_version\_stack$
    Increase value n of $packet\_transmitter$
  end if
end while

that n is increased more heavily (for weighting ratings see [24]).

To evaluate new events higher and to keep the values of p and n small, we only consider the last r last events. The value of the variable r depends on the capabilities of a device. For the most restricted ones, we take a value of 10. The procedures a node carries out in order to build trust values about its neighbors, is exemplified in Algorithm 1.

C. Trust Value Combination

As already mentioned, the nodes also forward their trust values to the border router or a cluster-head that aggregates them to the reputation values in the network or a cluster. For this aggregation, the Subjective Logic defines a consensus operator $\oplus$ [8]. Be $v_1 = (b_1, d_1, u_1)$ the trust value of node x in node y and $v_2 = (b_2, d_2, u_2)$ the trust value of another node w in the same y. Then the combined trust of the x and w in y is expressed by $v_1 \oplus v_2$ which is defined as follows:

$$\left(\frac{b_1 u_2 + b_2 u_1}{u_1 + u_2 - u_1 u_2}, \frac{d_1 u_2 + d_2 u_1}{u_1 + u_2 - u_1 u_2}, \frac{u_1 u_2}{u_1 + u_2 - u_1 u_2} \right)$$

Since this operator is commutative and associative, the border router or cluster-head can combine all incoming trust values about y. If the combined reputation value shows a large degree of distrust in y by its neighbors, the border router can assume that y is used for a malicious attack and notify the operator of the IoT system accordingly. Further, it may block the perceived intruder. This functionality is described in greater detail in Algorithm 2.

Algorithm 2 Trust Computing in the Border Router resp. Cluster-head

if Periodic Trust packets are received from network nodes resp. cluster members **then**
  Combine trust values for every node to its reputation value
  for All Nodes **do**
    if $Disbelief > intruder\_threshold$ **then**
      Block the node as an intruder
      Notify operator resp. border router
    end if
  end for
end if

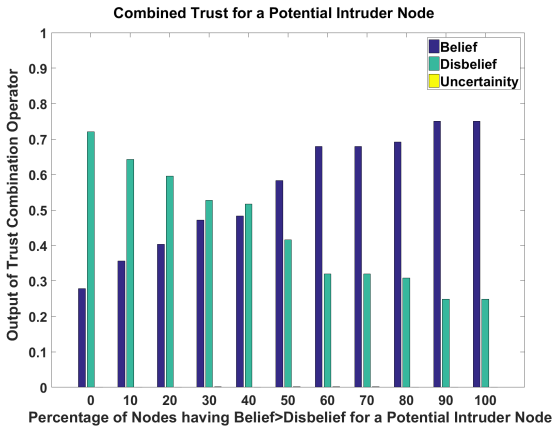

Fig. 4. Combined Trust for a Potential Intruder Node

As a first trust management policy, we decided that the border router shall notify the operator when variable d in the reputation value of a node is larger than b. We simulated a small IoT network containing an intruder and evaluated the inputs of its neighbors. The result is depicted in Fig. 4. It shows that d is larger than b in the reputation value of a node when more than 50% of the other nodes have more disbelief as well.

D. Three Algorithms to Manage Reputation

To realize our approach, we developed three different algorithms that are all based on Algorithm 2:

- *Neighbor Based Trust Dissemination (NBTD):* In this algorithm, trust is implemented in a centralized manner. The border router calculates trust values periodically based on the trust inputs received from the nodes in the DODAG. Thus, it is the sole manager of reputation ratings for all the nodes in the DODAG.
- *Clustered Neighbor Based Trust Dissemination (CNTD):* This algorithm is a distributed approach to collect and evaluate the trust values from the network. It assumes that the DODAG is segmented into several clusters that all contains a cluster-head as discussed in Sect. II-C. Instead of the border router, the cluster-head is responsible to gather and compute the reputation of each node in its cluster. It receives the trust values from the other nodes in the cluster periodically and aggregates them with its own trust values. If the d variable of a reputation value exceeds a threshold, the node will be blocked and the border router is notified.
- *Tree Based Trust Dissemination (TTD):* This algorithm uses the same topology as CNTD but reduces the surveillance of nodes. A node only supervises its parents but not children in order to save network overhead. In consequence, the leaf nodes of a DODAG will not be monitored since they have no children. We assume, however, that after a rebuild the position of the various nodes change such that former leaves will often get children that can track them.

IV. Simulation Results

The mentioned techniques are analyzed with help of simulations in MATLAB. We simulated all three algorithms introduced in Sect. III-D. Due to the lack of research in trust development for RPL networks, however, we did not compare the algorithms with the state-of-the-art.

The evaluation scenario consists of 1000 nodes that have been randomly deployed in a 100 $\times$ 100 m network. For CNTD and TTD, the network was divided into nine equally sized clusters. We carried out two different sets of simulations to compare the results of the three algorithms. In the first one, we conducted one simulation round using varying numbers of intruder nodes in the network to find out about the effect of the share of malicious nodes in the network. In the second simulation set, we executed several simulation rounds in which intruders detected in a certain round are excluded in the next one. Further, we checked in both simulation sets the effect of the algorithms on the overall network load. In our simulations, we assumed that an intruder node randomly gives misleading trust values for its neighbors in order to spoil the IDS.

The main focus of the simulation was on the following parameters:

- The *Number of Intruders Detected* identified in the network.

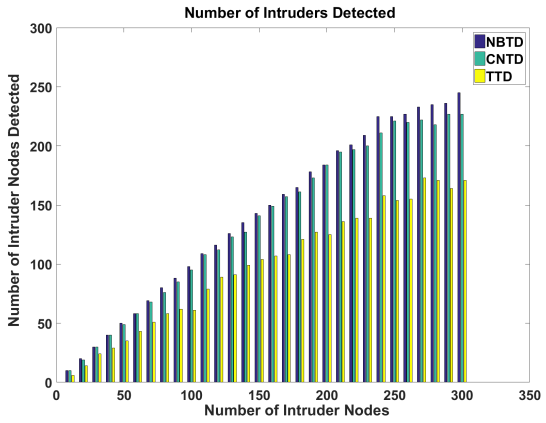

Fig. 5. Number of Intruders Detected

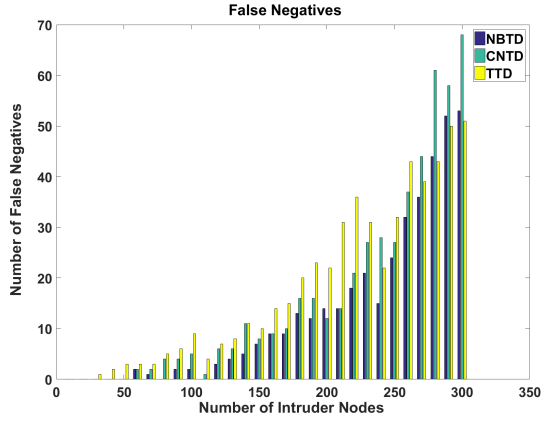

Fig. 6. False Negatives versus Number of Intruder Nodes

- *False Positives*, i.e., the number of non-intruder nodes that have falsely been identified as intruders.
- *False Negatives*, i.e., the number of intruder nodes that have falsely been identified as non-intruders.
- *Undetected Positives* indicating the number of intruder nodes that could neither be identified as an intruder nor as a non-intruder due to few observations leading to a high degree of uncertainty in the trust values.
- *Undetected Negatives* determining the number of non-intruder nodes that could not be identified as intruder or non-intruder due to a scarce number of observations.

Below, we discuss the different simulations followed by a discussion about the results.

A. Number of Intruder Nodes

The number of intruder nodes varied from 0 to 300 and the effect of these variations on the network parameters was observed. If the number of intruder nodes were increased, the number of detected intruders increased as well, see Fig. 5. Here, NBTD and CNTD showed better results than TTD since TTD cannot detect if a leaf node, i.e., one without children, is malicious.

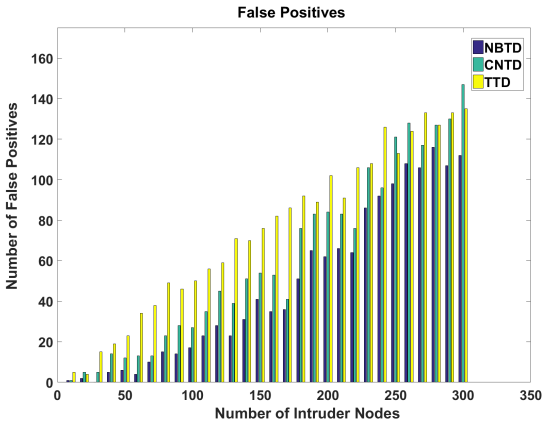

Fig. 7. False Positives versus Number of Intruder Nodes

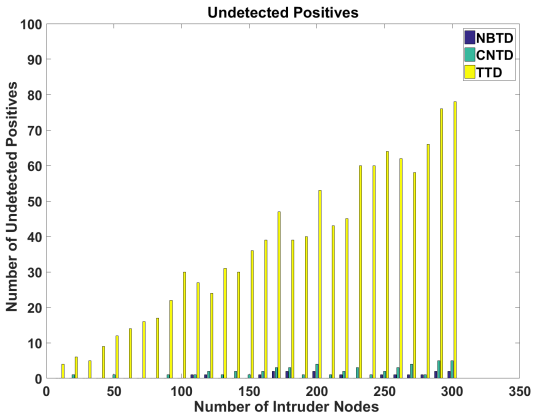

Fig. 9. Undetected Positives versus Number of Intruder Nodes

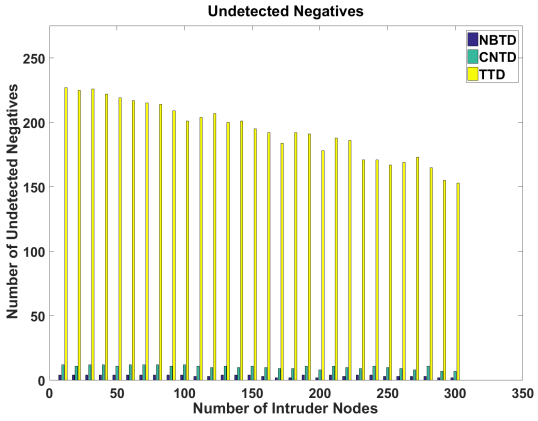

Fig. 8. Undetected Negatives versus Number of Intruder Nodes

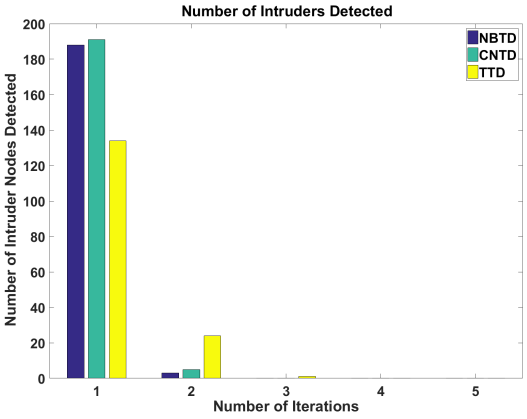

Fig. 10. Number of Intruders Detected

Figure 6 shows the false negatives for the three algorithms. A result is that the number of false negatives is very low if the number of intruder nodes are less than 50. For larger numbers of intruder nodes, the number of false negatives went up to 7%. That holds particularly for CNTD. The reason here is that also cluster-heads can be intruder nodes which can forge their own trust value and those of other members in the cluster.

A similar behavior is the number of false positives that are depicted in Fig. 7. Here, CNTD as well but also TTD are more problematic which also results from malicious cluster-heads. We need to mention that we did not implement a policy that a cluster head actively incriminates other members of his cluster which would have lead to much higher numbers of false positives.

In Figs. 8 and 9, we show the numbers of undetected positives and negatives. In these simulations, NTBD and CNTD are much better than TTD since that does not detect malicious leaf nodes.

B. Number of Rounds

In this set of simulations, the algorithms are executed simulating five different rounds. Figures 10 to 14 describe the progression of the five parameters over the rounds. Initially, 200 of the nodes were intruders. According to Fig. 10, all three algorithms detected intruders nearly only in the first two rounds. NBTD missed to detect around 5% of all intruders, CNTD around 3%, and TTD around 20%. The reasons for the misses are revealed in Figs. 11 and 14. Interestingly, the main reason for the misses in NBTD was false negatives, i.e., a too good rating in the reputations values. In contrast, the sole reason for such misses in CNTD was undetected positives, i.e., a high degree of uncertainty due to few experiences of adjacent nodes. A cause for that could be that, in average, trust value reports in NBTD run through more nodes than in CNTD. Thus, the likelihood of passing intruders falsifying the reports is higher. Nevertheless, we have to study this effect closer. We were not surprised, however, that TTD is much worse since the position of a node in the network may imply that it is always a leaf. In this case, it can never be detected as an intruder.

Overall, we do not take the number of not detected intruders in all three algorithms too seriously. An infection rate of 20% is typical for a heavily infected network. Here, however, the network operators will probably start a complete reset of

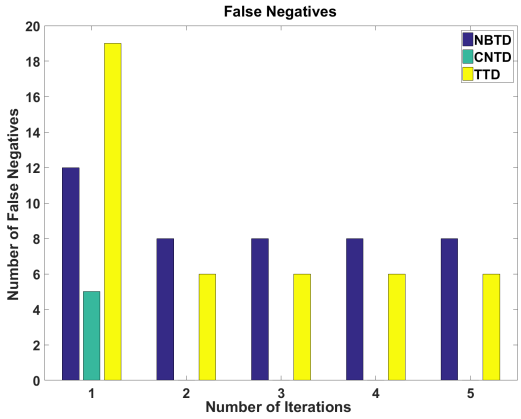

Fig. 11. False Negatives versus Number of Iterations

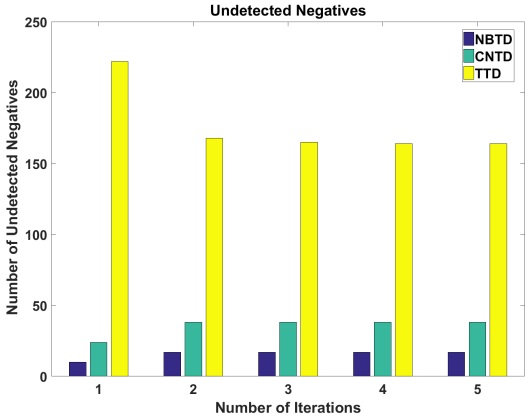

Fig. 13. Undetected Negatives versus Number of Iterations

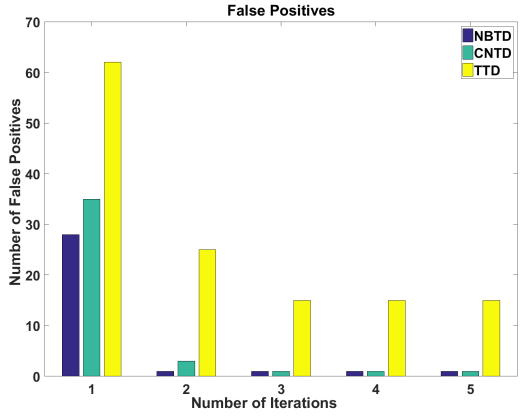

Fig. 12. False Positives versus Number of Iterations

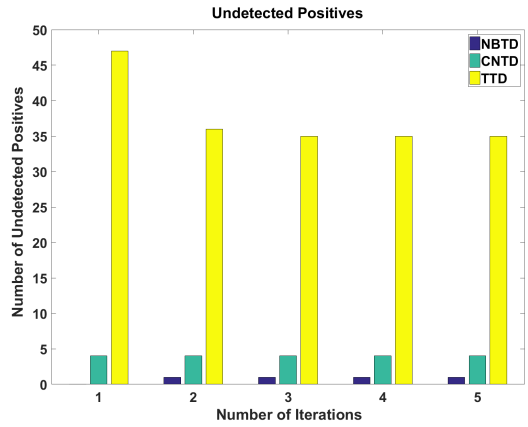

Fig. 14. Undetected Positives versus Number of Iterations

all devices. Figure 5 depicts that NBTD and CNTD detect nearly all intruders already in the first round when the overall percentage of infections is lower.

C. Network Load

In general, NBTD and CNTD give much better results than TTD with respect to the five parameters introduced above. However, as depicted in Figure 15, this is at the cost of network load. As expected, the centralized approach NBTD is worst since all experience reports have to be send all the way to the border router. CNTD has lesser overhead as the experience reports from the nodes in a cluster are sent only the relatively short way to its cluster-head and only the cluster-heads forward their aggregated results to the border router. At best, of course, is TTD since it manages trust values only for its parents but not its children. Thus, less data have to be forwarded to the cluster-head.

D. Discussion of the Three Algorithms

The decision about which of the three algorithms to take, depends on several properties of the real system, e.g., the likelihood of malicious nodes in the network and the quality of service provided by the underlying communication network. The centralized solution NBTD shows the best results in terms of intruder detection and error rate but has a very high overhead in terms of network load. On the other hand, TTD has the least amount of network load and yields good results when the network size is small. CNTD requires more packets on the network than TTD but has better results with respect to the error detection.

Altogether, we like to propose TTD only for relatively small networks with extra high communication costs. With respect to health care, that can be the monitoring of relatively few people in remote areas like Arctic regions or the Australian Outback where expensive satellite communication is used. For other cases, NBTD or CNTD should be used. An advantage of NBTD is that the border router is a larger unit that can be protected better against malicious attacks than smaller units. It is definitely to be preferred when the bandwidth is large and communication is cheap but the likelihood of malicious intruders is relatively high. CNTD, in contrast, saves packet exchanges but on the cost that potentially small devices with limited energy must take the role of a cluster-head. They can be easier attacked than a border router which may spoil the overall algorithm. Yet, CNTD can be of greater advantage if

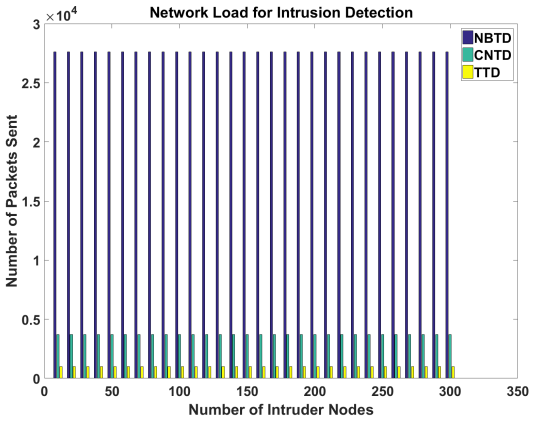

Fig. 15. Network Load

clusters not only consist of small devices but also incorporate some larger ones with better processing power and energy supply. Such units can take the role of cluster-heads and due to more in-depth protection mechanisms are similar save as border routers such that one can utilize the lower amount of packet exchange they offer.

V. Conclusion

Considering the Internet of Things (IoT) applications involving resource constrained devices, it is important to secure these devices by having a distributed IDS mechanism. We presented three distributed resp. centralized mechanisms that all use the trust management technique Subjective Logic [8] to detect intruder nodes in the system. Once an intruder is detected, it needs is removed from the network. The presented mechanism is suited to three prominent attacks against the RPL protocol described in the literature [5], [17], [20]. Nevertheless, our approach is quite flexible and can be easily accommodated to other types of attacks. For that, the trust metric computation needs just to be updated by adapting Algorithms 1 and 2.

We assume that the computations to build trust values according to the metric of [22] and the consensus operator of the Subjective Logic that are both presented in Sect. III, are not computing intensive. To get evidence, however, we are in the process to create a test-bed consisting of Z1 devices running the operating system Contiki. That will allow us to validate the results received from our MatLab simulations.

References

[1] K. Bloede, G. Mischou, A. Senan, and R. Koontz, "The Internet of Things," http://www.woodsidecap.com/wp-content/uploads/2015/03/WCP-IOT-M_and_A-REPORT-2015-3.pdf, Woodside Capital Partners, 2015, accessed: 2016-10-27.

[2] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the internet of things: Perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, 2014.

[3] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure CoAP for the internet of things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3711–3720, 2013.

[4] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 266–282, 2014.

[5] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad hoc networks*, vol. 1, no. 2, pp. 293–315, 2003.

[6] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[7] Z. A. Khan and U. Abbasi, "Evolution of Wireless Sensor Networks toward Internet of Things," in *Emerging Communication Technologies Based on Wireless Sensor Networks: Current Research and Future Applications*. CRC Press, 2016, ch. 7, pp. 179–200.

[8] A. Jøsang, "A Logic for Uncertain Probabilities," *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 9, pp. 279–311, 2001.

[9] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[10] T. Matsunaga, K. Toyoda, and I. Sasase, "Low false alarm rate RPL network monitoring system by considering timing inconstancy between the rank measurements," in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*. IEEE, 2014, pp. 427–431.

[11] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 606–611.

[12] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[13] S. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The internet of things for health care: a comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[14] IETF, "RfC 6550 — RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," https://tools.ietf.org/html/rfc6550, 2012, accessed: 2016-10-24.

[15] ——, "Neighbor discovery for IP version 6 (IPv6), IETF RFC 4861," https://tools.ietf.org/html/rfc4861, 2007, accessed: 2016-10-28.

[16] B. Mohamed and F. Mohamed, "QoS routing RPL for Low power and Lossy Networks," *International Journal of Distributed Sensor Networks*, vol. 2015, p. 6, 2015.

[17] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "A study of RPL DODAG version attacks," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2014, pp. 92–104.

[18] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proc. of the 1st USENIX/ACM Symp. on Networked Systems Design and Implementation*, 2004.

[19] K. D. Korte, A. Sehgal, and J. Schönwälder, "A study of the RPL repair process using ContikiRPL," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2012, pp. 50–61.

[20] 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. 2013, 2013.

[21] T. Zahariadis, P. Trakadas, H. C. Leligou, S. Maniatis, and P. Karkazis, "A novel trust-aware geographical routing scheme for wireless sensor networks," *Wireless personal communications*, vol. 69, no. 2, pp. 805–826, 2013.

[22] A. Jøsang and S. J. Knapskog, "A Metric for Trusted Systems," in *21st National Security Conference*. NSA, 1998.

[23] P. Herrmann, "Temporal Logic-Based Specification and Verification of Trust Models," in *4th International Conference on Trust Management (iTrust)*, ser. LNCS 3986. SpringerVerlag, 2006, pp. 105–119.

[24] M. Tavakolifard, P. Herrmann, and S. J. Knapskog, "Inferring Trust based on Similarity with TILLIT," in *3rd IFIP WG 11.11 International Conference on Trust Management (IFIPTM)*, ser. IFIP AICT 300. Springer-Verlag, 2009, pp. 133–148.