A Metric for Adaptive Routing on Trustworthy Paths

Christoph Hofmann, Elke Franz, Silvia Santini

A Metric for Adaptive Routing on Trustworthy Paths

Christoph Hofmann[1], Elke Franz[1], and Silvia Santini[2]

[1] Chair of Privacy and Data Security,
[2] Chair of Embedded Systems,
Technische Universität Dresden, 01062 Dresden,
{christoph.hofmann, elke.franz, silvia.santini}@tu-dresden.de

Abstract. Any data transmission over multiple hops requires the use of routing algorithms to find a path from the sender to the receiver(s). Supporting secure data transmissions further requires the selected path to be trustworthy, i.e., not under the control of an attacker. Since attacks can occur at any time during data transmission, the current state of the network must be considered when selecting trustworthy paths. In this paper, we introduce a novel metric – called *Locally Evaluated Trust* (LET) – for local trust evaluation. The LET metric can be used by adaptive routing algorithms to select a trustworthy path. To evaluate our approach, we simulate different routing algorithms and compare their performance in the presence of one or more malicious nodes in a two-dimensional torus mesh. Our results show that the LET metric allows adaptive routing algorithms to effectively identify and circumvent malicious nodes.

Keywords: adaptive routing, attacker model, trustworthiness, local trust rating

1 Introduction

Data transmission in a communication network requires the use of routing algorithms to find a path from the sender to the receiver(s). Common requirements on routing algorithms are a minimal path length, high throughput, and low latency. Adaptive routing algorithms allow to consider the current state of the network in order to achieve these goals.

To select a path, existing algorithms often evaluate efficiency metrics such as the current load of neighboring nodes or the quality of links. But besides efficiency, the security of data transmission is also essential. A routing algorithm should thus be able to efficiently deliver packets to the intended recipients even in the presence of attackers.

An attacker may disturb the transmission in several ways, for instance by dropping or delaying packets. The sender can however typically detect whether a data transmission has been successful or not by using acknowledgments – short messages sent by the recipient to the sender upon the reception of a data transmission. If an acknowledgment is not received within a pre-specified time interval,

the routing algorithm assumes the message (or the acknowledgment) was lost. Lost messages are then often retransmitted but retransmissions increase network load, power consumption, and communication latency. To prevent attackers from causing this overhead, routing algorithms should thus select trustworthy paths. To prevent new attack possibilities and to limit the communication overhead, the selection of trustworthy paths should neither require the participation of a central entity in the network nor the exchange of information about possible attacks or attackers.

In this paper, we propose a novel metric to evaluate the trustworthiness of network nodes in a decentralized fashion. Our metric, called LET (Locally Evaluated Trust), is designed to guide the choice of trustworthy paths in adaptive routing algorithms. Every node in the network evaluates the trustworthiness of its one-hop neighbors using information about received (or missed) acknowledgments. Thereby, a missing acknowledgment lowers the trustworthiness of a neighbor and of every path passing through it. The lower the trustworthiness of a neighbor, the lower is the probability for this neighbor to be included in a routing path (and vice versa).

To evaluate the effectiveness of our metric, we implement four routing algorithms and simulate their behavior under the presence of one or more malicious nodes. Our results show that LET can describe the trustworthiness of paths even if it is based solely on local information about adjacent nodes. Algorithms that leverage our metric achieve a significant higher rate of successfully delivered packets in comparison to other algorithms.

The remainder of the paper is organized as follows. We review related work in Section 2. In Section 3, we describe our system model as well as the attacker model and introduce the LET metric. Section 4 describes our evaluation setup and the results of our simulations. Finally, Section 5 concludes the paper and provides an outlook on future work.

2 Related Work

The selection of a suitable path from a given source to a destination is the essential task of every routing algorithm. Based on the way algorithms select a routing path, they can be classified as deterministic, oblivious, or adaptive algorithms [4]. XY routing and Valliant's routing algorithm are examples for deterministic or oblivious routing algorithms. Both are based on the model of Dimension Order Routing (DOR) where packets are routed ordered by direction until they reach the closest point to the destination or the destination itself [4].

Deterministic and oblivious algorithms select a path only based on predefined rules. In contrast, adaptive algorithms are able to consider information about the current state of the network. This characteristic of routing algorithms is of course also necessary to select a trustworthy path.

Information about the current state of the network are described by means of metrics. Baumann et al [1] give an overview about metrics for different purposes.

Several metrics evaluate the efficiency of a path using various methods. A well known and often used metric is the minimum hop count which always selects a shortest path from source to destination. But selecting a minimal path is not reasonable all the time, especially if packet loss can occur during the transmission. In wired networks, packet loss is usually caused by congestion. The Transmission Control Protocol (TCP) was designed to cope with these problems in wired networks. Since packet losses indicate congestion, fewer packets are sent until the problems diminish.

In wireless networks, however, packet loss is usually caused by a bad quality of the links. Retransmissions are then used to recover from packet losses. Several metrics for the description of the quality of links have been proposed for Wireless Sensor Networks (WSN) and Mobile Ad-hoc Networks (MANETs).

A well known example is the Expected Transmission Count (ETX) [5]. The ETX of a link is an estimate of the number of packet transmissions that are necessary to deliver a packet from one node to the next. ETX is computed by considering information about the delivery ratio of previously sent data and control packets. The ETX of a path from a sender to a destination can then be computed as the sum of the ETX values for each link on the path.

ETX has been used also in the context of wireless sensor networks. For instance, Gnawali et al. [10] use the ETX to guide the construction of a collection tree along which packets are routed towards one or more sinks. Thereby, the next forwarder for a given packet is determined in a hop-by-hop fashion, i.e., the neighbor with the smallest ETX is chosen as the next hop. Generic routing algorithms for wireless sensor networks also use other metrics, like the EDC (Expected Duty Cycle) proposed in [12]. This metric enables the ORW (Opportunistic Routing in Wireless sensor network) protocol to select routes in the network in an opportunistic way, thereby minimizing the overall time during which nodes have their radios switched on. A number of other approaches and metrics have been presented in the wireless sensor networks literature, e.g. [7]. They however mainly focus on improving reliability and latency while minimizing energy consumption. Security issues are instead often neglected.

However, security of transmission is also essential. A maliciously acting node may launch several kinds of attacks, either alone or in cooperation with other malicious nodes. Such attacks like modifying packets, selective forwarding of packets, or sinkhole and blackhole attacks where packets are dropped can harm the network massively and have a severe influence on routing performance [6],[11]. Therefore, it should be possible to determine, identify, and circumvent these maliciously acting nodes. The routing algorithm should be able to select a trustworthy path that is not under the control of an attacker.

At first sight, it should be possible to use a metric that describes the quality of links also for the description of the trustworthiness, since both a bad link quality and active attacks may cause packet loss and require retransmission of packets. However, as Dong et al. discussed in [6], a metric that only focuses on maximizing the throughput can also introduce new attack possibilities. For example, nodes can try to manipulate local metrics as well as global metrics in

order to manipulate and control traffic. Another difference between routing in the presence of faults and in the presence of attackers is the question what is an adequate strategy to react on problems. In case of faults, the routing algorithm can still try to send packets over the path, maybe sending additional redundant packets, before an alternative path is selected. In case of malicious nodes, the routing algorithm should try to select a trustworthy path as soon as possible.

Another direction of research that directly considers malicious nodes is to apply reputation as basis for a metric. One example is a metric called CORE [14] that proposes a mechanism to rate the willingness of a node to cooperate in the network. Each node keeps a list containing the reputation of his adjacent neighbors. The overall reputation is calculated by subjective and indirect reputation and ranges from -1 (bad) to +1 (good). Subjective reputation is based on the nodes' own observations whereas indirect reputation is based on observations made by other nodes. Nodes that are not cooperating are penalized by bad reputation. Other examples for reputation based metrics are described in [2] and [16]. They also propose trust evaluation based on direct observations and information which is propagated by other nodes in the network.

A further approach for rating the trustworthiness is described in [13] for MANETs. A watchdog mechanism and a pathrater are defined to describe the reliability of a neighbor node. Nodes store information about recently sent packets in a buffer and use the watchdog mechanism to overhear the communication of their neighboring nodes by using the promiscuous mode. If a node notices that a formerly sent packet was forwarded by its neighbor correctly, the packet is removed from the buffer. Otherwise, if a packet remains in the buffer longer than a certain period, the responsible neighbor node gets a bad reputation. This mechanism is performed by every node on the path. If a node observes bad behavior of its neighboring node, it accuses this node to the source of the packet.

However, similar problems are relevant for metrics based on reputation as discussed in [6] for high-throughput metrics. If notifications about the misbehavior of nodes or accusations are used for the computation of the metric, an attacker can try to manipulate the metric. To cope with these problems, Dong et al. suggested a measurement-based detection of attacks combined with a temporary accusation-based reaction [6]. To limit the abuse of the accusation, nodes can only issue a new accusation after the previously issued one expired.

In the approach we introduce in this paper, we aim at excluding attacks that utilize indirect information delivered by other nodes such as probe packets, notifications, or accusations. Therefore, the proposed metric is solely based on local observations what additionally minimizes the overall communication overhead. The metric can be used by an adaptive routing algorithm that aims at selecting trustworthy paths.

A similar approach was introduced in [15, 9]. The authors also discuss ratings based on local observations by means on end-to-end acknowledgments. The main difference to our approach is the fact that senders and forwarders rate their neighbors what requires that the acknowledgments are sent back using the reverse path. In contrast, we aim at a solution that does not impose overhead

to the forwarders. Hence, rating is only done by the senders. Forwarders do not have to keep track of the success of forwarding data packets. Further, this also means that we do not need a special treatment of acknowledgments, they can be routed like data packets. Since we assume a system with a high volume of data traffic, all nodes will act as sender and rate their neighbors after a short time. A further difference is the system topology. While the authors of [15, 9] discuss ad hoc networks, we assume a fixed topology where the possible paths to the recipient are known from the beginning. This fixed topology allows to study different constellations of malicious nodes.

3 Our Approach

3.1 System Model

In this work we focus on a two-dimensional torus topology. This topology is found in several systems like for instance the HAEC Box (Highly Adaptive Energy-Efficient Computing), a novel high performance computing platform that is currently under development at TU Dresden [8].

The HAEC Box consists of several stacked boards each endowed with powerful compute nodes that cooperate to execute parallel applications. Nodes on each board are connected through optical waveguides in a two-dimensional torus. Each node can directly communicate with the compute nodes of adjacent boards by means of wireless links. During the execution of an application on the HAEC Box, compute nodes are expected to communicate intensively with each other. Hence, the efficiency of routing has a significant influence on the overall performance of the HAEC Box. In this paper, we focus on the topology of a single board, i.e., on a regular torus mesh network with m rows and n columns as illustrated in Fig. 1.

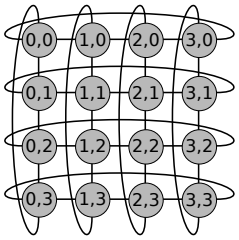


Fig. 1: 4x4 torus mesh

Further, we focus on unicast communication where one sender communicates with one receiver over one or more intermediate nodes (forwarders). Each node can be a sender, forwarder, or receiver. Receivers issue acknowledgments to inform the sender about the successful delivery of data packets. These end-to-end acknowledgments are routed through the network as data packets.

3.2 Attacker Model

In this work, we consider only active attackers. Active attackers can, e.g., modify or drop packets (data packets as well as acknowledgments), delay their transmission or replay formerly sent packets. The presence of these attackers can thus be detected using quantitative metrics. For instance, an increase in the average data delivery latency might hint at the presence of an attacker who delays packets.

Passive attackers only observe the data transmitted. However, as long as an attacker only observes, it is not possible to recognize the attack and, hence, not

possible to describe it in a metric. The confidentiality of the data can, however, be enforced through end-to-end encryption. We therefore assume the use of end-to-end encryption and do not further discuss passive attacks.

We further assume that appropriate security measures like digital signatures are in place that enable nodes to verify the origin and validity of received packets. Hence, modified packets can be recognized and will be discarded so that a modification implies a packet loss as well. Therefore, we do not explicitly distinguish between modification and dropping of packets in the following.

The sender recognizes the loss of a packet if a data transmission is not acknowledged within a pre-defined time interval (timeout). Since it is not possible to distinguish whether the data packet or the acknowledgment was lost, both cases are treated as the same. The delay of a transmission can also cause the detection of a packet loss, but when the acknowledgment eventually reaches the sender, this false detection can be corrected. The replay of acknowledgments to conceal an attack is not possible since we assume that the acknowledgment is digitally signed by the receiver and contains a unique reference to the acknowledged data packet.

An active attacker can also affect network availability by flooding the network with useless traffic (denial of service). The LET metric considers the trustworthiness of nodes and is not designed for this type of attacks. Thus, it is not helpful to prevent or detect them. We leave the consideration of denial of service attacks to future work.

We assume that both links or nodes may be attacked. Since the implications are the same, we model the active attacker by one or more malicious nodes. Malicious nodes drop packets with a certain probability (selective forwarding) or drop all packets in total (sinkhole attack). As we mentioned before, this packet dropping also covers the modification of packets.

Participants of a communication are the sender, the forwarder(s) and the receiver. Regarding the role of the attacker, we can distinguish different cases.

- *Only forwarders can be malicious:* Senders and receivers are trustworthy, they want to protect their communication. We assume that malicious nodes only want to disturb the traffic, they do not act as sender or receiver.
- *Forwarders and receivers can be malicious:* The receiving node may be controlled by an attacker and, therefore, malicious. In this case, the receiver does not send back an acknowledgment. Again, malicious nodes are only interested in disturbing the communication between other nodes and do not act as sender.
- *Senders, forwarders, and receivers can be malicious:* Malicious nodes are interested in disturbing the communication between other nodes, but they also want to communicate themselves.

For our first evaluations, we consider the first case. We mainly focus on malicious nodes that drop all packets but also consider malicious nodes that perform selective forwarding.

3.3 Locally Evaluated Trust (LET) Metric

In our approach, each node locally rates the trustworthiness of its four neighboring nodes (north, east, south, and west) by evaluating the delivery of previously sent packets. This implies that LET does not apply to a whole path but only to adjacent nodes. Before transmitting a packet, the sender and each forwarder consider their locally computed trust values to select the next node on the path.

The range for the trust value can be arbitrary. For the sake of simplicity, we let the LET range between 0 (untrusted) and 1 (trusted). A trust value of 0.5 indicates a neutral rating that is also used as the initial value of the metric.

Updates of the LET are triggered by local observations of the nodes. For each sent packet, the sender logs the identifier of the packet, the time of transmission, and the identifier of the neighbor that was selected as successor. The reception of an acknowledgment confirms the successful delivery of that packet. The sender does not know the path that was used, but knows that the node it selected as first forwarder processed the packet correctly. Hence, the acknowledgment triggers a positive rating of that neighbor.

On the other side, a missing acknowledgment indicates a problem, although the sender cannot identify the reason. The loss of a data packet or of the corresponding acknowledgment could have been caused by an attacker or by a fault in the network. The sender only knows to which of its neighbors it has sent the packet and, therefore, decreases the rating of that neighbor. This might be "unfair" since the neighbor may not be the cause of the problem. In this case, further successful routing over this node will improve its rating again.

LET values are updated as follows:

$$up\,(trust) = \begin{cases} trust + trust & \text{for } trust < 0.5 \\ trust + \dfrac{1 - trust}{2} & \text{for } trust \geq 0.5 \end{cases}$$

$$down\,(trust) = \begin{cases} trust - \dfrac{trust}{2} & \text{for } trust \leq 0.5 \\ trust - (1 - trust) & \text{for } trust > 0.5 \end{cases}$$

If a node changes its behavior, the metric should reflect this change as soon as possible. For our evaluation, we use static intervals and restricted the number of possible trust values to five positive and five negative ratings (Fig. 2).

3.4 Path Selection

To prevent problems of adaptive routing algorithms like deadlocks or livelocks, we use the Odd-Even-Turn-Model [3] as a basic path selection method. This model restricts the movements of a packet to certain allowed turns depending on the x coordinate of the node that takes the routing decision. These restrictions prevent cycles in the routing path and thus livelocks to occur.

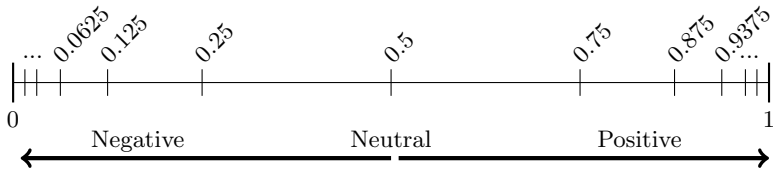


Fig. 2: Range of the trust value

In our decentralized adaptive approach, each forwarder selects the next node on the path to the receiver. To guide this selection, we first follow the rules of the Odd-Even-Turn-Model. The set of nodes that are not excluded by this model establish the set of *available nodes* (Av). Since we are also interested in selecting a short path, the neighbors that are on a minimal path to the receiver are selected from the available nodes ($Av_{min} \subseteq Av$).

We then consider the LET value computed by each node to further refine the choice of the next node. Simply selecting the node with the highest LET is however not beneficial. First, this node might become overloaded and thus cause latency to increase. Second, choosing always the node with the highest LET as the next successor prevents other nodes from getting the chance to improve their reputation by participating in a data transmission. Third, if the LET of available nodes that allow the selection of a minimal path are equal, we select one of these nodes at random irrespectively of the actual LET value.

We accordingly define two thresholds to classify the trustworthiness of the available nodes providing minimal paths. We refer to nodes with $trust \geq 0.5$ as *high trust nodes* ($Av_{min,h} \subseteq Av_{min}$) and to nodes with $0.06 \leq trust < 0.5$ as *low trust nodes* ($Av_{min,l} \subseteq Av_{min}$). The definition of these sets increases the chance for a node to be selected as successor and, thus, to get the opportunity to improve its rating. The node that makes the routing decision selects a high trust neighbor from the set $Av_{min,h}$. If there is no high trust neighbor ($Av_{min,h} = \emptyset$), the node selects a low trust neighbor from $Av_{min,l}$. If there is also no low trust neighbor ($Av_{min,l} = \emptyset$), the node selects the neighbor with the highest trust value from the set of available nodes (Av). In this latter case, the chosen neighbor does not lie on a minimal path.

4 Evaluation

4.1 Simulation

We simulated four routing algorithms in a 4×4 torus mesh network:

XY : Static XY routing,
OE_l : Odd-Even-Routing [3] with the current load as a metric,
OE_t : Odd-Even-Routing [3] with the proposed LET metric,
OE_o: Own implementation of the Odd-Even-Turn-Model with the proposed LET metric.

Two of these algorithms (OE_t, OE_o) apply the suggested LET metric, the other ones (XY, OE_l) serve as a reference. OE_o is an own implementation of the Odd-Even-Turn-Model since the implementation of this model given in [3] excludes some turns that are not forbidden by the Odd-Even-Turn-Model itself.

The simulation was done with Python using the discrete-event simulation framework SimPy [17], which allows a discrete time basis for the simulation. In our simulation, a node is able to process one packet per discrete time step (called tick in the following). At each node, incoming packets are stored in a queue and processed in a FIFO order.

The aim of our simulation was to observe the behavior of the four routing algorithms under the presence of one or more malicious nodes. A maliciously acting node either drops all packets given to it or selectively forwards them with a certain probability. To enable a fair comparison, each of the four routing algorithms is simulated in a separate mesh with identical settings for the selection of the malicious nodes and probabilities in case of selective forwarding. In every tick, a packet with random sender and receiver is created and given to each mesh.

We simulated different scenarios regarding number and constellation of malicious nodes. For each scenario, we performed 10 simulation runs with 10000 packets each. Preliminary tests have confirmed that this number of packets is sufficient to achieve a sufficiently small standard deviation of the results. During the simulation, the efficiency parameters described in the next section are determined and averaged over all runs.

4.2 Evaluation Parameters

The performance of the simulated routing algorithms was evaluated by a comparison of different evaluation parameters. One of these parameters is the *Delivery Ratio* that reflects the ability of the routing algorithm to deliver packets even in the presence of malicious nodes.

Another interesting parameter is the average total load (*AvrLoad*), which reflects the overall load of the network. We compute the load of a single node by the length of its queue and the total load of a path as sum of the loads of all nodes on that path. The load has a significant influence on the average time (*AvrTime*) needed to route a packet. Both *AvrLoad* and *AvrTime* can only be computed in relation to the delivered packets.

4.3 Results and Discussion

In first experiments, we assumed that malicious nodes drop all packets (sinkhole attacks). Fig. 3 compares the delivery ratios of the routing algorithms for different numbers and constellations of malicious nodes.

In case the network contains only one misbehaving malicious node, all algorithms can deliver more than 90% of the packets. Adding more malicious nodes to the network decreases the delivery ratio of algorithms not using our proposed

X

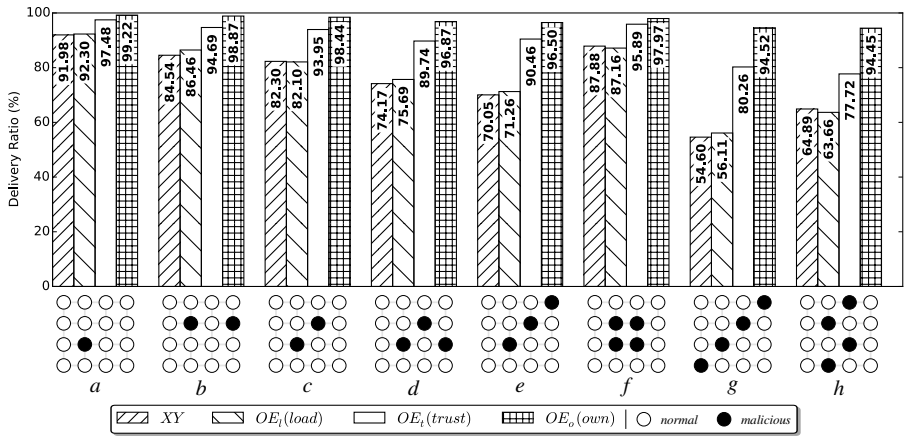


Fig. 3: Delivery ratios for different constellations of malicious nodes

trust metric massively. In the case of three malicious nodes arranged diagonally in the network (e), the delivery ratios of XY and OE_l only reach around 70%, whereas the algorithms using the LET metric still perform good reaching 90% (OE_t) and 96% (OE_o). Adding one more malicious node to the diagonal arrangement (g) causes the delivery ratio of XY and OE_l get even worse. Almost the half of all sent packets cannot reach their designated destinations in this scenario. While the delivery ratio of OE_t also drops by around 10%, the delivery ratio of our own implemented version of the Odd-Even-Turn-Model still reaches a good performance of 94%. This can be explained by the greater number of routing options offered by the own implementation.

Furthermore, we found out that the performance does not depend on the number of malicious nodes only. The arrangement of malicious nodes also has a significant influence on the results. While the algorithms reach relatively high delivery ratios in a network with four malicious nodes arranged in the middle of the mesh (f), the algorithms may already behave worse in a network with only two malicious nodes (e.g., c). This is mainly due to the fact that in the case of arrangement f, where malicious nodes are located in the center of the network, all honest nodes have only one misbehaving neighbor, whereas in case of arrangement c four nodes have one malicious neighbor and two nodes even have to deal with two malicious neighbors.

That also explains the low delivery ratios in constellations g and h. In arrangement g, eight honest nodes have to deal with two malicious neighbors. In case of arrangement h, four nodes have one malicious neighbor and four other nodes even have to deal with three malicious neighbors. If a node has three malicious neighbors, there is only one trustworthy link. In case of attackers that drop all packets, such a node can only send or receive packets. If the node has to forward a packet, it will always be dropped by one of its malicious neighbors.

Another interesting parameter for evaluating the different approaches is the average load (**AvrLoad**). Fig. 4 shows the average load on a path for each of the malicious node constellations presented in Fig. 3.

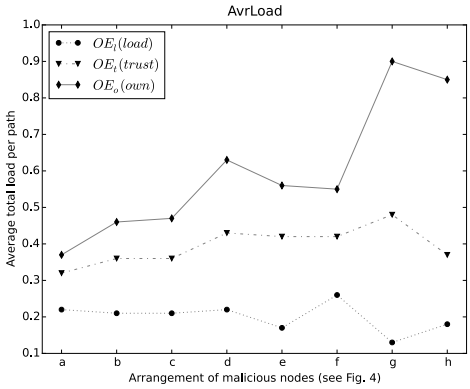


Fig. 4: Average total load per path for each arrangement

As the average load can only be calculated for successfully delivered packets, the values to some extend depend on the delivery ratios. This means that in the case of a lower delivery ratio the values base on fewer measurements and therefore are less meaningful. Nevertheless, it appears that the average load is constantly low for the Odd-Even algorithm using the load as a metric (OE_l).

Regarding the algorithms that use the trustworthiness as a metric (OE_t and OE_o), the average load is increasing according to the number of malicious nodes in the network. While the delivery ratio of OE_o is more or less constantly high, the average load increases rapidly from about 0.4 (one malicious node) to about 0.9 (four malicious nodes). This can be explained through the limitations imposed by the malicious nodes in the network. They force the nodes to use one of the few remaining trustworthy paths in the network for routing a packet. This causes higher load on these paths.

The average time needed to transmit a packet (**AvrTime**) is closely related to the average load on a path. Fig. 5 shows that the average time for routing a packet using OE_o increases with the number of malicious nodes. This is mainly due to the fact that the load and therefore the time a packet has to wait to get routed also increases. In addition, OE_o allows non-minimal routing paths in the absence of a minimal path with a certain trustworthiness. A non-minimal path also increases the time needed for routing. For OE_t, the average time remains mainly constant whereas the average time for OE_l decreases. Again, these values also depend on the delivery ratio since the average time can only be calculated for successfully delivered packets. The number of malicious nodes significantly influences the delivery ratio of OE_l. In case of many attackers, only packets with

a receiver close to the sender are routed successfully. Therefore, the path length is short and the average time decreases.

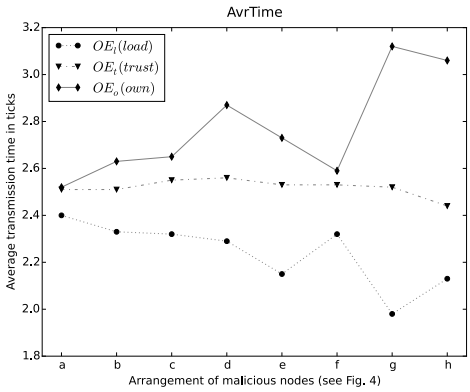
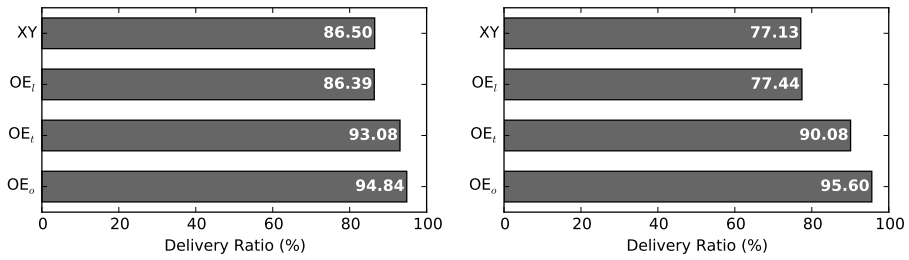


Fig. 5: Average time in ticks for the transmission of a packet

As described in our attacker model, a maliciously acting node may only forward some of the packets it receives. The rest is dropped. This malicious behavior is also known as selective forwarding. We also tested the LET metric under the presence of nodes that drop packets with a certain probability. Fig. 6 shows the delivery ratios for a network with four malicious nodes dropping packets with a probability of 30% (a) or 50% (b). The malicious nodes were arranged as shown in constellation g (see Fig. 3).



(a) 30 % (b) 50 %

Fig. 6: Delivery ratio for selective forwarding malicious nodes

Also in case of selective forwarding nodes, the two algorithms using our proposed metric (OE_t and OE_o) outperform the other tested algorithms. These results confirm that the metric is able to identify and circumvent the misbehav-

ing nodes even if they only drop parts of the traffic. Of course, the detection of malicious nodes is more complicated in comparison to sinkhole attacks and, therefore, takes more time.

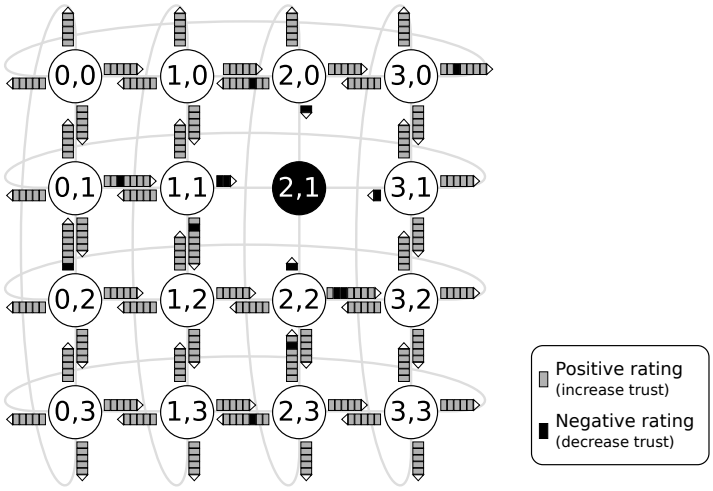


Fig. 7: Trust ratings performed in a network with one malicious node

Fig. 7 shows the ratings performed by each node for a simulation with one malicious node. A gray box represents a positive rating while a black box indicates a negative rating. Ratings are ordered chronologically in the direction of the arrow. Thus, boxes closer to the rating node imply an earlier rating.

This presentation of the ratings shows that all neighbors of the malicious node (2,1) were able to determine its malicious behavior. Because of alternative paths with good reputation, the decreased connections are not used anymore. An exception is the second negative rating of node (1,1): node (2,1) was selected as first forwarder by node (1,1) again before the time out for the first lost packet was reached.

As discussed earlier, a timeout can also falsely cause a negative rating since the sender cannot localize the reason of a packet loss. However, if a falsely decreased but honest node is used again, it will improve it's trust value.

4.4 Limitations

One limitation of our approach is that a rating is only issued by senders. As long as a node only acts as forwarder or receiver, it does not have the possibility to rate its neighbors. However, under the assumption of a high traffic volume, this issue is not crucial.

Another point to be considered is that we assume honest senders and receivers in our simulations. While a malicious sender that behaves selfishly and does not

forward packets but sends own packets is not a problem, a malicious receiver that constantly refuses to send acknowledgments affects a proper establishment of trust on sender side.

Since it is not possible to identify the reason of a timeout of an acknowledgment, the sender decreases the trust value of the selected successor. However, if that node processed the packet correctly, this downgrade is not justified.

Due to the high connectivity of the torus mesh, it may take some time before a wrongly degraded node will be selected as forwarder again. Either other available nodes are downgraded to the same trust level (low trust nodes), or the sender improves the trust value of the falsely accused node. The latter is only possible if the sender has to send a packet directly to this neighbor.

5 Summary and Outlook

Within this paper, we presented a novel metric (LET) for trust evaluation. Our proposed metric only utilizes local observations for the establishment of trust values. Simulations confirmed that adaptive algorithms using our proposed metric can successfully identify and circumvent misbehaving nodes. As a consequence, these algorithms are able to achieve higher delivery ratios in comparison to other routing algorithms. Furthermore, the simulations showed that the arrangement of malicious nodes in the network has a significant impact on the performance of the algorithms.

Besides cryptographic mechanisms like encryption, consideration of the trustworthiness provides a proactive method to protect packets in a communication network.

Until now, we only utilized end-to-end acknowledgments for trust evaluation. A topic of future work is to investigate the use of link-to-link acknowledgments. Even if link-to-link acknowledgments will increase the overall communication overhead, they may enhance the accuracy of the metric. Further, they allow to overcome the limitation that only senders can issue ratings. Of course, potential new attacks have to be carefully investigated as well. For future work, we will consider the usage of dynamic rating intervals to further improve the rating process and to ensure that LET reflects a changed node behavior as soon as possible. Another topic of future work is to study the influence of more dynamic attackers that change their behavior over time and to improve the rating accordingly. Future simulations will also consider other possible roles of the attacker (malicious sender or receiver) as mentioned in Section 3.2. Finally, we will investigate the performance of our metric in other topologies. First experiments for a slightly bigger 8×8 torus mesh network delivered promising results.

Acknowledgments

This work is partly supported by the German Research Foundation (DFG) in the CRC 912 “Highly Adapted Energy-Efficient Computing”. Further, the authors wish to thank the anonymous reviewers for their helpful comments.

References

1. R. Baumann, S. Heimlicher, M. Strasser, and A. Weibel. A Survey on Routing Metrics. *TIK report*, 262, 2007.
2. S. Buchegger and J.-Y. Le Boudec. Performance Analysis of the CONFIDANT Protocol. In *Proc. of the 3rd ACM Int. Symposium on Mobile A Hoc Networking & Computing*, pages 226–236. ACM, 2002.
3. G.-M. Chiu. The Odd-Even Turn Model for Adaptive Routing. *IEEE Trans. Parallel Distrib. Syst.*, 11(7):729–738, July 2000.
4. W. J. Dally and B. P. Towles. *Principles and Practices of Interconnection Networks*. Elsevier, 2004.
5. D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. *Wireless Networks*, 11(4):419–434, 2005.
6. J. Dong, R. Curtmola, and C. Nita-Rotaru. On the Pitfalls of High-Throughput Multicast Metrics in Adversarial Wireless Mesh Networks. In *Proc. IEEE SECON*, pages 224 – 232, 2008.
7. S. Duquennoy, O. Landsiedel, and T. Voigt. Let the Tree Bloom: Scalable Opportunistic Routing with ORPL. In *Proc. of the 11th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, 2013.
8. G. Fettweis, W. E. Nagel, and W. Lehner. Pathways to Servers of the Future. In *Proc. of DATE*, 2012.
9. W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, and W. Kellerer. Castor: Scalable Secure routing for Ad Hoc Networks. In *Proc. of IEEE INFOCOMM 2010*, 2010.
10. O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proc. of the 7th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, 2009.
11. C. Karlof and D. Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. *Ad hoc networks*, 1(2):293–315, 2003.
12. O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low Power, Low Delay: Opportunistic Routing meets Duty Cycling. In *Proc. of the Conf. on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, 2012.
13. S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. of the 6th Annual Int. Conf. on Mobile Computing and Networking*, pages 255–265. ACM, 2000.
14. P. Michiardi and R. Molva. CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks. In *Advanced Communications and Multimedia Security*, pages 107–121. Springer, 2002.
15. P. Papadimitratos and Z. J. Haas. Secure Data Communication in Mobile Ad Hoc Networks. *IEEE JSAC*, 24(2):343 – 356, 2006.
16. A. Rezgui and M. Eltoweissy. TARP: A Trust-Aware Routing Protocol for Sensor-Actuator Networks. In *Proc. of IEEE Int. Conf. on Mobile Adhoc and Sensor Systems*, pages 1–9. IEEE, 2007.
17. SimPy. simpy.readthedocs.org.