*future internet*

MDPI

*Article*

# Enhancing Reactive Ad Hoc Routing Protocols with Trust

**Yelena Trofimova * and Pavel Tvrdík**

Faculty of Information Technology, Czech Technical University in Prague, Thákurova 9, 160 00 Prague, Czech Republic; pavel.tvrdik@fit.cvut.cz

* Correspondence: yelena.trofimova@fit.cvut.cz

**Abstract:** In wireless ad hoc networks, security and communication challenges are frequently addressed by deploying a trust mechanism. A number of approaches for evaluating trust of ad hoc network nodes have been proposed, including the one that uses neural networks. We proposed to use packet delivery ratios as input to the neural network. In this article, we present a new method, called TARA (Trust-Aware Reactive Ad Hoc routing), to incorporate node trusts into reactive ad hoc routing protocols. The novelty of the TARA method is that it does not require changes to the routing protocol itself. Instead, it influences the routing choice from outside by delaying the route request messages of untrusted nodes. The performance of the method was evaluated on the use case of sensor nodes sending data to a sink node. The experiments showed that the method improves the packet delivery ratio in the network by about 70%. Performance analysis of the TARA method provided recommendations for its application in a particular ad hoc network.

**Keywords:** ad hoc network; trust; reactive routing; route discovery; packet delivery ratio



**Citation:** Trofimova, Y.; Tvrdík, P. Enhancing Reactive Ad Hoc Routing Protocols with Trust. *Future Internet* **2022**, *14*, 28. https://doi.org/10.3390/fi14010028

Academic Editor: Paolo Bellavista

Received: 14 December 2021
Accepted: 12 January 2022
Published: 15 January 2022



## 1. Introduction

Ad hoc networks are a broad category that includes a variety of network types based on their intended use. A wireless network of mobile devices, vehicles, sensors, or devices for Internet of Things (IoT) could be involved. Ad hoc communication allows to create networks without infrastructure components, such as routers or access points. Instead, each node participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is made dynamically based on the network connectivity and the routing algorithm in use. On the one hand, ad hoc networks can be quickly deployed and are naturally scalable. On the other hand, as nodes forward the data of others, these networks need some mechanism to ensure functionality, fault tolerance, and reliability of communication. There is no central authority in the network. Nodes need to establish secure communication by themselves, which requires cooperation among nodes. Cooperation is unfeasible without some notion of trust.

Trust is a measure of confidence that a node behaves according to expectations. Trust is absolutely fundamental for the future of wireless communication [1]. Trust values can be used directly by nodes in interactions with others or for access control, intrusion detection, or secure routing. A considerable amount of methods to establish and maintain trust relationships in ad hoc networks have been proposed [2–11].

Various ad hoc routing protocols have been proposed over the past 20 years. The Mobile Ad Hoc Network (MANET) working group [12] published Request for Comments (RFC) for three ad hoc routing protocols: Ad Hoc On-Demand Distance Vector (AODV), Optimized Link State Routing (OLSR), and Dynamic Source Routing (DSR). AODVv2 (former Dynamic MANET On-demand (DYMO)) is a work in progress. OLSR is a *proactive* protocol, meaning that the information about routes is periodically exchanged. On the contrary, AODV and DSR are *reactive* routing protocols, meaning that the route discovery process is started only when a route to some destination is needed. This way, the commu-

nication overhead is lower, and node batteries are saved compared to proactive routing protocols [13].

### *1.1. Motivation, Problem Statement, and Main Contributions*

Ad hoc routing protocols, by design, rely on the fact that nodes cooperate in route discovery. For instance, the RFC of AODV says that the protocol is intended for usage in networks where all nodes can trust each other [14]. Nevertheless, it is not always the case. The reasons for the nodes' failure to cooperate can be different. However, the result is the same: the network performance degrades due to unexpected node behavior (i.e., the existence of untrusted nodes).

This research aims to improve the functionality of ad hoc routing protocols to work more reliably in ad hoc networks with untrusted nodes. Incorporating trust in reactive ad hoc routing protocols can be solved by changing the protocol, which implies that the protocol algorithms and structures must be changed (e.g., in [4–6], authors introduced new message structures; in [3], the algorithm for updating the routing table was changed; in [4,5,15], the routing table structure was changed).

The problem we will tackle is enabling trust-aware ad hoc routing without changing the routing protocol.

The main idea of our solution is to enforce route discovery through trusted nodes from outside of the routing protocol. The utilized trust mechanism [11] uses a neural network to estimate the trust of nodes. These trust estimations are then used to influence the route discovery phase of a reactive ad hoc routing protocol to prefer more trusted routes. Untrusted nodes are penalized during the route discovery phase by delaying the Route Request messages of those nodes. Thus, the node trust values influence the routing decisions indirectly to improve the performance of reactive ad hoc routing protocols.

A simulation proof-of-concept framework was built, and an extensive experimental evaluation of the proposed solution was conducted. As the ad hoc routing protocol, the AODV protocol was chosen, but the proposed method to enhance reactive ad hoc routing with trust can be used with any reactive ad hoc routing protocol. In addition, a study of how the proposed solution influences the routing protocol metrics was performed. Another problem is finding the best combination of parameters and creating recommendations on how to tune the method settings according to the particular use cases.

To the best of our knowledge, all previous methods of enhancing ad hoc routing with trust used the trust values directly during route discovery [5–7,9,10,15] or in routing decisions [4,8,16]. Our approach differs because the route discovery messages through the untrusted nodes are penalized by delays, but the routing algorithm and structures remain unchanged. The main contribution of this research is creating a method for enhancing reactive ad hoc routing protocols with trust that does not require changes to the routing protocol.

### *1.2. Article Organization*

The rest of the article is organized as follows. Section 2 provides an overview of the related work, description of reactive routing protocols and the trust estimation method. Section 3 describes our proposed method of enhancing reactive ad hoc routing protocols with trust. Section 4 presents the simulator, and Section 5 describes the experiments and their results. Section 6 discusses the considerations regarding the implementation of the proposed method. Finally, Section 7 evaluates the results of the experiments, provides conclusions, and outlines the future work.

## 2. Background

### *2.1. Related Work*

Methods to enhance ad hoc routing protocols with trust can be divided into two groups. The first represents approaches that apply trust during the route discovery phase. Either the control messages are rejected based on the trust, or they contain a special field in

the header that collects the trust value of the path during the dissemination of the discovery message. Examples of such approaches are Eissa et al. [15], Marchang and Datta [5], Venkanna et al. [6], Simaremare et al. [7], and Hatzivasilis et al. [10]. The second group represents approaches that select routes according to their trust values stored in routing tables. Examples are Li et al. [4], Wang et al. [17], and Pathan et al. [3]. Our approach falls into the first group.

#### 2.1.1. Trust-Based AODV Protocol Approaches

In [4], routing tables of nodes contain trust value for each path. Therefore, if there are more entries for some destination with the smallest hop count, the more trusted path is chosen. The trust of a path is calculated as a product of the node trusts along the path.

Authors in [15] proposed a trust-based scheme for AODV which uses a friendship mechanism. Nodes keep lists of friends and their friendship values, calculated over some features. Route Request (RREQ) and Route Reply (RREP) messages are rejected based on trust values of previous and next hops. Furthermore, a friendship of the route is evaluated, and the route is registered if it is more friendly than the existing one.

The method proposed in [5] changes the original neighbor table entries and routing table entries to keep information about trust. Neighbor trust values are used to handle RREQ and RREP messages from those neighbors, while route trusts are used to select routes.

Authors in [16] have a slightly different approach to incorporate trust. Each node has its trust table, and when receiving a packet, it checks the trust value of the source. If it is untrusted, then the packet is dropped.

In [6], the path trust value is computed during the spread of RREQ and RREP messages. Each node on the way adds the trust value of the next hop while the control message is spread over the network. The destination node computes the average and that is considered as the trust of the path.

In [7], when a node receives a RREQ message, it checks the trust of the source. If the source is untrusted, the RREQ message is ignored. Otherwise, the trust of the neighbors is calculated, and RREQ is forwarded only to the trusted ones. RREP messages are only forwarded.

#### 2.1.2. Trust-Based DSR Protocol Approaches

In [8], a trust scheme is presented to extend DSR routing with detection and isolation of misbehaving nodes. When non-cooperative behavior is detected, the misbehaving node is excluded from routing.

Authors in [17] evaluate trust based on the similarity of nodes. The similarity is a weighted sum of different node attributes, such as velocity or moving direction. The decision to forward the data is based on the similarity degree.

In [9], the trust mechanism is integrated into the route discovery process. Trust is a cumulative sum of the normalized values for different categories of behavior. In the DSR protocol, nodes add their IP addresses in the RREQ message. During the propagation of the RREP message, each node adds the trust of the preceding node.

SCOTRES system [10] was created for wireless ad hoc networks deployed for monitoring environmental parameters. The routing protocol uses three metrics to evaluate the node trust. Based on the information from the routing table, a topological metric of a node is evaluated, and the rating system will tolerate failures of significant nodes. Energy and channel-health are the other two metrics. Paths that contain malicious nodes are excluded. The method was integrated into the DSR protocol.

### *2.2. Use Case and Communication Model*

The use case of the proposed method is an ad hoc network of IoT/sensor nodes monitoring, measuring, and collecting environmental parameters. The communication model assumes that all nodes keep sending data periodically to a single node, called a *sink*. This is a typical case for wireless sensor networks that represent a special class of ad hoc

networks [18]. The real-world applications of such networks include monitoring of air pollution, destruction phenomena, agricultural monitoring, and more [19]. The common property of all these use cases is that there is always one dedicated node collecting data from other nodes that act as sensors. These sensors cannot communicate directly with the sink, they need to use intermediate nodes to deliver their messages to the sink. Each route from a node to the sink is traversed periodically, so often that it does not expire.

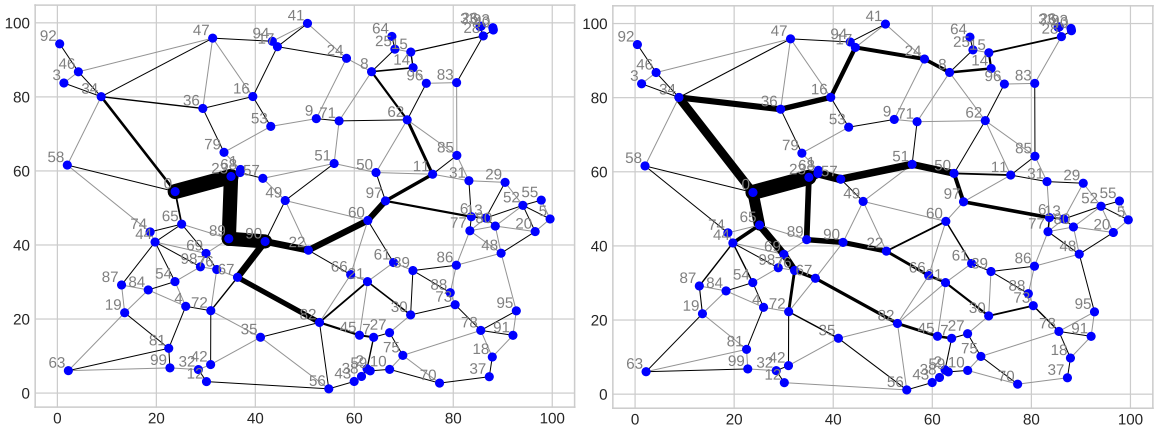
This communication model was used for performing an experimental evaluation of our solution, but the solution is not limited to this model. Other models of ad hoc communication are suitable for its application.

### *2.3. Description of Routing Protocols*

#### 2.3.1. The AODV Protocol

The AODV protocol [14] is a reactive routing protocol, which implies that a route discovery process is initiated when a route to a specific destination is needed. Route discovery is accomplished by flooding the network with RREQ messages. Upon receiving RREP messages the route with the smallest hop count will be recorded in the routing table. The routing table on each node contains information about the next hop to the destination, the number of hops, and the expiration time of the routing table entry. Each time the route is used for forwarding data, its expiration time is updated. In the case of periodic communication, the routing table entry never expires.

There are two ways of creating routes in the AODV protocol. Protocol's behavior depends on the flag called *destination only*. When the flag is set (further referenced as DFT), only the destination node can reply to a RREQ message. This implies that every time a new route is requested, a new original route from the source to the destination is created, without any influence of already existing routes. On the other hand, when the flag is not set (further referenced as DFF), any node that has a route to the destination can reply to the RREQ message. As a result, the routes that already exist influence the searching process. The existing routes are then used by many nodes and even though a better route exists, it is not discovered. The difference in the created routes can be seen in Figure 1. Nodes are depicted as circles with their IDs written up left from them. The node with $ID = 0$ is the sink. Grey links, although they exist, are not used in any route. The thicker the link is, the more routes are going through it. Topology in Figure 1a was created with the *destination only* flag not set. Compared to Figure 1b, routes in Figure 1a were attached to the existing ones, and thus the topology has fewer "branches".



(**a**) (**b**)

**Figure 1.** Topologies created by AODV. (**a**) *Destination only* flag is set to false (DFF), (**b**) *Destination only* flag is set to true (DFT).

AODV scales well to large networks, as shown in [20]. It is a widely accepted choice for reactive routing [21].

#### 2.3.2. The DSR Protocol

Dynamic Source Routing protocol (DSR) [22] is also a reactive protocol. Its route discovery process is similar to AODV [23]. Thus, the results from the simulations with the AODV protocol can be applied to the DSR protocol, too.

Generally, DSR is a source routing protocol. It uses so-called *source routing*, where each data packet contains a list of intermediate nodes it will traverse on its way to the destination.

Compared to AODV, the DSR protocol keeps the information about multiple routes to the destination. A node may choose to select the shortest sequence of hops to the destination, or it may use an alternative metric to select the route [22]. According to its RFC, DSR is designed for networks of up to two hundred nodes and of small diameter up to 10 hops [22]. The next difference is that there is no expiration time for routes.

To store routing information, DSR protocol implements Route Cache data structure. It can also implement Link Cache data structure, meaning each individual link learned from the DSR Options header is stored. Then, to search for a route, the source node performs Dijkstra's shortest path algorithm to find the best path to the destination [22].

#### 2.3.3. Other Protocols

Another ad hoc routing algorithm, called TORA, was proposed in [24], but it never made it to RFC. To the best of our knowledge, it is not used.

The last reactive protocol, DYMO, was renamed to AODVv2 [25]. RFC for AODVv2 is a work in progress. The document expired in 2019, and it has not been approved yet.

### *2.4. Description of the Trust Estimation Method*

The method published in [11] demonstrated the applicability of neural networks (NNs) for the evaluation of trust.

We defined trust as the confidence that a given node will forward the data correctly [26]. To measure that, we use the *packet delivery ratio (PDR)* metric. A node is considered trusted if its *PDR* is greater than a given threshold ($THR$). Otherwise, it is considered untrusted.

A path between a source node and a destination node is an ordered sequence of intermediate nodes. Let $P = N_1, N_2, \ldots, N_p$ be a path from source $N_1$ to destination $N_p$, where $p$ is the length of the path. Paths are constructed by routing protocols.

The *PDR* of a node $N_i$ is defined as follows:

$$PDR(N_i) = \frac{\text{forwarded-by-}N_i}{\text{sent-through-}N_i}, \tag{1}$$

where forwarded-by-$N_i$ is the number of packets correctly forwarded by node $N_i$ and sent-through-$N_i$ is the total number of packets sent to node $N_i$ that were supposed to be forwarded.

The advantage of the NN-based trust estimation method is fast NN learning with generated data. To generate the data, we used the fact that *PDR* of a path $P$ is defined as the product of *PDR*s of intermediate nodes along the path [11]:

$$PDR(P) = \prod_{i=2}^{p-1} PDR(N_i) \tag{2}$$

Let us describe the method. We have a network topology where nodes can measure the *PDR*s of the paths. Using NNs, we can learn the underlying dependencies and predict *PDR*s of nodes from *PDR*s of paths. A detailed example of the problem and its solution are described in [11], Subsection III-A.

A formal mathematical definition of the NN-based trust estimation method is presented in [26]. The trust of a node is its estimated *PDR*, and it is a real number between 0 and 1.

The evaluation of method performance showed that in 98% of cases NN was successful in estimating the trust of a node in an ad hoc network.

The advantage of the NN-based estimation of node trusts is that data can be incomplete or incorrect, and the NN still predicts trust with high success [26].

However, for the NN-based trust estimation method to perform reasonably, network topology should change slowly. As the method proposed in this article is designed to run in symbiosis with the NN-based trust estimation method, we assume that the positions of all network nodes are fixed. Considering the communication model, the environmental conditions may change (e.g., some obstacles appear), so even with the fixed topology, *PDR*s of nodes, and thus their trusts, may change in time.

## 3. The Proposed Method of Trust-Aware Reactive Ad Hoc Routing

### *3.1. The Main Idea*

The main idea is that the RREQ message delivery is delayed according to the trust of the previous hop node. The goal of the trust-aware route discovery is to prefer routes consisting of more trusted nodes. We will call this method TARA (Trust-Aware Reactive Ad Hoc routing).

In the rest of the article, we will use the AODV protocol as the reactive ad hoc routing protocol. However, the proposed adaptivity of route discovery on trust values of nodes is independent of routing protocol implementation. It should work with any reactive ad hoc routing protocol that accepts the first obtained route.

To make the TARA method independent of the particular ad hoc routing protocol, the delaying of RREQ messages needs to be implemented as an interlayer of the ISO-OSI model. Said interlayer needs to be placed between the link and the network layer, and its job is to filter the RREQ messages coming from the link to the network layer. For each RREQ message, this interlayer decides how long the message will be delayed before sending it to the network layer. The delay value is derived from the estimated *PDR* value of the neighbor node that sent or resent the RREQ message. In this article, we consider three functions to calculate these delays, see Section 3.5.

The method description follows. Section 3.2 outlines the assumptions for the distribution of trust values that are necessary for computation of the delay. Section 3.3 introduces the time model for delay and metrics related to time. Section 3.4 describes the method performance metrics. Finally, Section 3.5 provides algorithms for the delay functions.

### *3.2. Trust Distribution*

In order to successfully deploy the suggested solution, each node needs to know the estimated *PDR*s of its neighbors. In sink-based networks, the sink node always communicates with all other nodes. Thus, it can collect all necessary statistical data, estimate the trust of the nodes using a NN, and distribute trust estimations to all nodes in the network. We assume that the topology of the network does not change, or changes slowly in time, and each route found by the AODV protocol is used periodically or very often, therefore it does not expire.

In this article, we do not consider scenarios in which routes are rediscovered and the sink node recomputes the impacted trust values as a reaction to network changes. We will simply assume that all nodes receive the trust estimation, and we will focus on how the TARA method performs compared to the standard AODV protocol.

### *3.3. Definition of the Time Unit*

Our simulation does not model real devices. Instead, it reproduces the results of their behavior. Thus, the simulation does not include the precise timing of the communication in a network. We decided to establish a reference operation that takes for simplicity constant time. This operation is the retransmission of the message to the next hop under ideal conditions. In reality the retransmission is not an atomic operation, and it is influenced by many factors such as node performance or the amount of data traffic going through that node. We expect that all nodes are identical and the data traffic is not too dense, therefore

we could neglect these factors. We define *Time Unit (TU)* to represent the retransmission time. All metrics, which are related to time, use *TU* as a unit.

### *3.4. Performance Metrics*

To better understand and evaluate the TARA method performance, we first specify the performance metrics. We define four metrics as follows.

#### 3.4.1. Network-Wide Average PDR

In order to investigate the overall losses caused by the low *PDR* of some nodes, we define the *average PDR* metric ($PDR_{avg}$), which is the mean of all path *PDR*s in the network:

$$PDR_{avg} = \frac{\sum_{i=1}^{m} PDR(P_i)}{m}, \tag{3}$$

where $P_i$ is the $i$-th path and $m$ is the number of paths in the network. For our use case of sensors sending data to one sink, there are $m = n - 1$, where $n$ is the number of nodes in the ad hoc network.

#### 3.4.2. Network-Wide Average Path Length

When we incorporate trust into the routing protocol, it does not find the shortest paths anymore. It may pick more trusted but longer alternative routes. The metric called *average path length* ($PL_{avg}$) helps us understand how the average path length was influenced by incorporating trust into the route discovery mechanism. The length is the number of hops from the source node to the destination node. $PL_{avg}$ is formally defined as:

$$PL_{avg} = \frac{\sum_{i=1}^{m} PL(P_i)}{m}, \tag{4}$$

where $m$ is the number of paths $P_i$ in the ad hoc network.

#### 3.4.3. Average and Maximum Delivery Delay of RREQ Messages

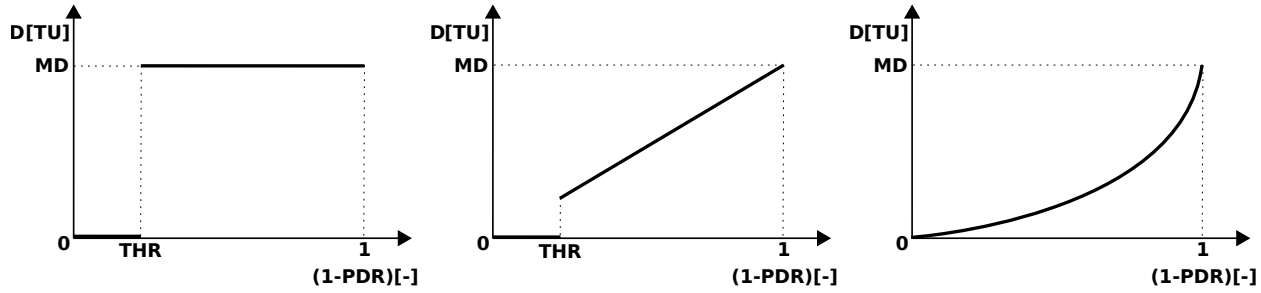
As the TARA method delays the RREQ messages through nodes with low PDR and thus penalizes them during route discovery to improve the *PDR* of the paths in the network, it, on the other hand, prolongs the time of the new route discovery. We call it *route discovery delay (RDD)*. So metrics $PDR_{avg}$ and $RDD_{avg}$ are counteractive: improvement of one can deteriorate the other. In order to investigate these dependencies, we define two metrics:

- average route discovery delay ($RDD_{avg}$);
- maximum route discovery delay ($RDD_{max}$).

Both metrics are measured in *TU*s. The reason we picked two metrics of *RDD* instead of one is that, for some topologies, the maximum reaches really impractical values, thus some routes cannot be found. This could be a problem for network applications when the routes to all nodes need to be found. For these purposes, it is more relevant to consider $RDD_{max}$. On the other hand, some network applications do not require to communicate with all nodes (for example, it is expected that some sensors are lost or no longer operational), and then the $RDD_{avg}$ is a more useful metric.

### *3.5. Delay Functions*

We consider three functions for delaying the RREQ messages depending on the node *PDR*. Each delay function input consists of the estimated *PDR* and some function parameters, and the output is the calculated *delay D* of the RREQ message in *TU*. Each function is described in detail in the following sections. The graphical representation of all functions is in Figure 2.




(**a**) (**b**) (**c**)

**Figure 2.** Delay functions. (**a**) Constant Delay function, (**b**) Linear Delay function, (**c**) Exponential Delay function.

#### 3.5.1. Constant Delay Function

The most straightforward function that we experimented with is the Constant Delay function. The idea is to set the delay value to the maximal delay (*MD*) every time the estimated *PDR* of the neighbor node drops below the $THR$. The purpose of the $THR$ value is not to penalize the nodes with relatively high *PDR*s. The value of *D* is calculated as a function of *THR*, *MD*, and estimated *PDR* (Algorithm 1).

**Algorithm 1** Constant Delay function specification

**Input:** $THR \in [0,1]$; $MD > 1$; $PDR \in [0,1]$
**Output:** $D \geq 0$
  **if** $PDR < THR$ **then**
    $D \leftarrow MD$
  **else**
    $D \leftarrow 0$
  **end if**
  **return** $D$

We expect this function to have the worst results among all metrics. However, it helps us see whether delaying the RREQ messages can be successful at all. We also expect that higher *MD* values increase the resulting $PDR_{avg}$ and both *RDD* metrics.

#### 3.5.2. Linear Delay Function

The previous function does not take into account the value of the estimated *PDR*. The $THR$ value helps solve this insufficiency only partially. The Linear Delay function delays RREQ messages linearly according to the *PDR* of the node that sent or resent the RREQ message up to the *MD* value; we use the threshold, too. The value of *D* is calculated as a function of *THR*, *MD*, and estimated *PDR* (Algorithm 2).

**Algorithm 2** Linear Delay function specification

**Input:** $THR \in [0,1]$; $MD > 1$; $PDR \in [0,1]$
**Output:** $D \geq 0$
  **if** $PDR < THR$ **then**
    $D \leftarrow MD * (1 - PDR)$
  **else**
    $D \leftarrow 0$
  **end if**
  **return** $D$

In this case, we expect that the *THR* value does not have a positive impact on the results. We also expect that higher *MD* values increase $PDR_{avg}$ and both *RDD* metrics.

3.5.3. Exponential Delay Function

The last suggested function delays RREQ messages more for the nodes with small estimated *PDR* and less for nodes with higher estimated *PDR* in comparison with the Linear Delay function. The differences are clear from Figure 2. The shape of the curve is determined by the following function depending on three parameters, *BASE*, *MD*, and *PDR* (Algorithm 3).

**Algorithm 3** Exponential Delay function specification

**Input:** $BASE \geq 2; MD > 1; PDR \in [0,1]$
**Output:** $D \geq 0$
$D \leftarrow MD * (BASE^{1-PDR} - 1)/(BASE - 1)$
**return** $D$

As we can see from the function specification, the *BASE* value influences the bend of the curve, and the *MD* value scales the curve in the vertical direction. We expect that this function could have better results than the Linear Delay function because it penalizes more the nodes with worse *PDR*. We also expect that the *RDD* metrics values could be smaller in comparison with the Linear Delay function for the similar $PDR_{avg}$ values.

## 4. Simulator

The simulator is written in Python and has a modular structure. Each logical component of the simulated problem was implemented as a separate module. Therefore, functionality can be easily changed by replacing the module. The basic structure is shown in Figure 3. The logical functioning and behavior of individual modules are described in the following subsections.

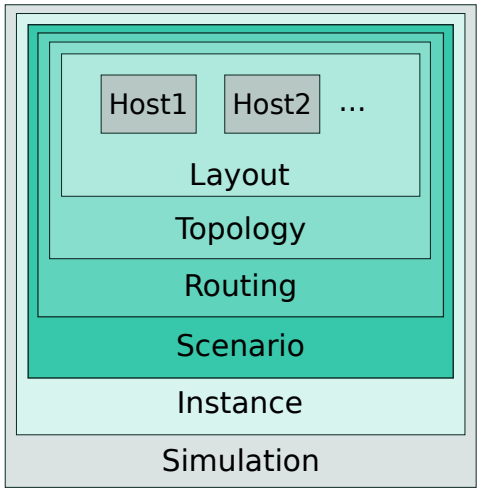



**Figure 3.** Module architecture of the simulator.

### *4.1. Layout Module: Modeling the Network*

An ad hoc network is modeled as a graph with set of vertices $\mathcal{N} = \{N_1, N_2, ..., N_n\}$ that represent network nodes and set of edges $\mathcal{L} = \{L_1, L_2, ..., L_e\}$. Every edge represents a bidirectional link between exactly two nodes. The set of nodes and links together create an ad hoc network topology.

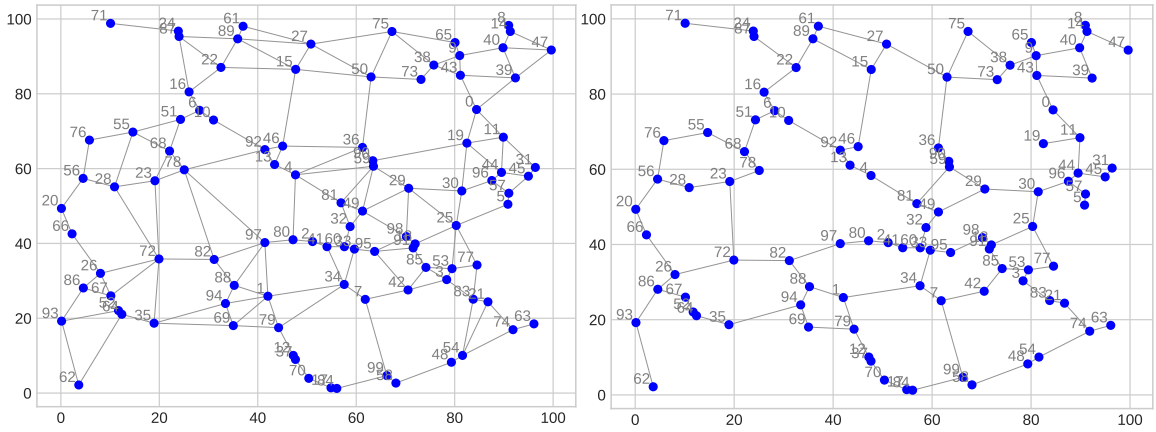
The parameters for generating the set of network nodes are the 2D area size (the default value is $100 \times 100$) and the number of nodes (the default value is 100). We use the uniformly random distribution to generate the positions of nodes in the selected 2D area.

### *4.2. Topology Module: Generating Topologies*

One of the main properties of each topology is its *density* that is related to the number of links between nodes. Density highly influences the number of alternative paths between

any two nodes in the network. Together with the total number of nodes, density is the most important parameter to observe when observing the behavior of routing mechanisms. To describe the density, we borrowed the definition from the graph theory. We use the *average vertex degree (AVD)*, which tells us how many neighbors each node has on average.

When generating a topology for a given 2D random distribution of node positions, we want to create a connected topology and, at the same time, we want to achieve a specific density. We decided to use the topology control algorithm called Lune $\beta$-skeleton [27]. The performance of this algorithm is influenced by parameter $\beta$ that produces the topologies with various densities. A smaller $\beta$ value creates a higher density network while a greater value produces a lower density network. An example of the difference between topologies created with various $\beta$ parameters is in Figure 4. $\beta = 1$ produces a topology with more links and higher *AVD* compared to $\beta = 2$.



(**a**) $\beta = 1$ (**b**) $\beta = 2$

**Figure 4.** Two networks with the same distribution of nodes in the area but with different densities controlled by the $\beta$ parameter.

The advantage of this method is that, for $\beta \leq 2$, it is ensured that the resulting topology is always connected if it was connected before the application of the algorithm. In our case, we started with the complete graph topology and then reduced the number of links with the help of this algorithm.

The resulting *AVD* of each topology depends on the $\beta$ parameter and on the initial positions of all nodes. Therefore, the same $\beta$ value produces topologies with slightly different *AVD*s. The dependency of the average vertex degree on the $\beta$ value is shown in Figure 5.

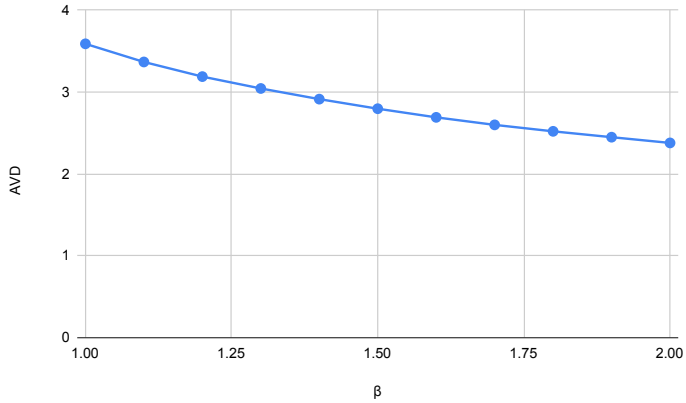



**Figure 5.** Average vertex degree as a function of the $\beta$ parameter.

### *4.3. Routing Module: AODV Implementation*

Our simulator does not implement the full specification of the AODV protocol. We implemented only the result of the AODV protocol operation. No node-to-node communication is simulated. We focused only on the way the AODV protocol discovers the new paths and how this process is influenced by our delay mechanism.

As the AODV route discovery process is based on flooding, it always finds the fastest discovered path. To simulate this behavior, we use Dijkstra's algorithm for searching shortest paths where the distance is the delay in *TU*s. If the trust penalization of a RREQ message is not involved, the delay between neighboring nodes is exactly 1 *TU*. Otherwise, the delay is 1 *TU* plus penalization $D$ computed by the given delay function (Constant, Linear, or Exponential). If Dijkstra's algorithm returns more paths with the same delay, one of them is randomly selected. This simulates the real ad hoc routing where the fastest discovery depends (randomly) on various physical constraints (e.g., communication load).

We also implemented both AODV modes DFT and DFF.

### *4.4. Scenario Module: Trust Distribution*

This module allows us to change the behavior of nodes in terms of trust.

In order to investigate the performance of the TARA method, we prepared several scenarios that differ by the number of nodes with $PDR < 1$, i.e., potentially untrusted nodes (depending on the threshold value). Nodes with $PDR < 1$ were selected randomly. The *PDR* values for these nodes were generated uniformly from the interval $[0, 1]$. We have limited the simulation scenarios to cases where the percentage of nodes with $PDR < 1$ are 10%, 20%, 30%, 40%, and 50%. In the following sections, we denoted these trust scenarios as SC-10, SC-20, SC-30, SC-40, and SC-50, respectively.

### *4.5. Instance Module*

Instance module defines the parameters of one specific simulation instance. Specific values of parameters of one simulation instance are called its *configuration*. Table 1 lists all parameters of the simulation instance configuration and their default values.

**Table 1.** Parameters of simulation instance.

| Parameter | Default Value |
|---|---|
| 2D area size | $100 \times 100$ |
| distribution of node positions | uniform |
| link type | bidirectional |
| number of nodes $n$ | 100 |
| $\beta$-parameter | 1.0 |
| trust distribution scenario | SC-50 |
| delay function | none |
| *destination only* flag | DFT |

Default configuration simulates a general use case of reasonably large networks. Area and distribution of node positions allow to create variety of topologies in order to test fairly the TARA performance.

### *4.6. Simulation Module*

The purpose of Simulation module is to run simulation instances and collect, process, and save the simulation results. The unit of the simulation is 1 *experiment* which is 100 runs of one simulation instance configuration, each with different node positions and therefore with different topologies, generated by the Lune $\beta$-skeleton for the given $\beta$.

## 5. Experiments and Results

Within each experiment, we calculate $PDR_{avg}$, $PL_{avg}$, and $RDD_{avg}$ (see Section 3.4) as average over all generated 100 topologies of the experiment. $RDD_{max}$ is the maximum

over all the 100 topologies, representing the worst *RDD*. This allows us to choose the best parameters of the delay function. More specifically, if some particular combination of parameters produces a reasonable $PDR_{avg}$, but at the same time the value of $RDD_{max}$ is too high, it just means that the overhead of the route discovery is too high, and such combination of parameters is not acceptable. The goal of experiments is to find values of configuration parameters that maximize $PDR_{avg}$ and minimize $RDD_{avg}$ at the same time.

For each combination of parameters, we simulate two versions of the outputs depending on whether the *destination only* flag is enabled (DFT) or not (DFF).

The *reference experiment* is an experiment with default values (see Table 1) of configuration. In particular, no delay function is used and, therefore, the AODV protocol is run in the standard mode, and untrusted nodes are not penalized.

The performance of each experiment will be compared relative to the performance of the reference experiment. We will enumerate the relative change for a particular metric $\delta_{metric}$ via Formula (5).

$$\delta_{metric} = \frac{Value_{metric} - ReferenceValue_{metric}}{ReferenceValue_{metric}} * 100\% \quad (5)$$

$\delta_{metric} = 0\%$ means that the metric value has not changed compared to the reference value. Additionally, $\delta_{metric}$ can be more than 100%.

### 5.1. Dependence of the TARA Method Performance on the Delay Function

The configuration of the reference experiment is 2D area $100 \times 100$, uniform distribution of 100 nodes, bidirectional links, $\beta = 1.0$, and no delay function is applied (trust distribution scenario is irrelevant). The performance results of the reference experiment are shown in Table 2.

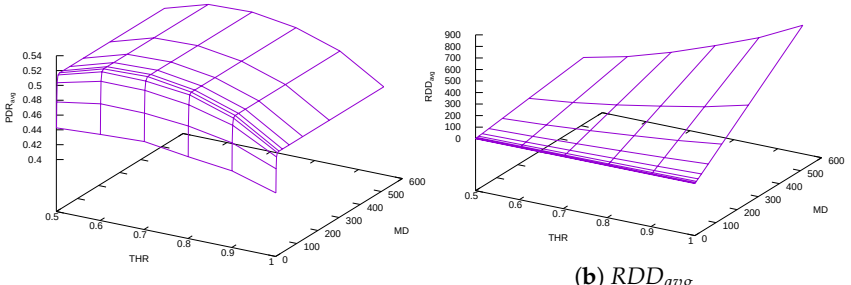
**Table 2.** Performance metrics for the reference experiment.

| Flag | $PDR_{avg}$ | $PL_{avg}$ | $RDD_{avg}$ | $RDD_{max}$ |
|---|---|---|---|---|
| DFT | 0.35 | 6.13 | 6.13 | 16 |
| DFF | 0.34 | 7.28 | 1.42 | 15 |

When no delay function is applied, delay at each hop is 1 *TU*. Thus, $RDD_{avg} = PL_{avg}$ if DFT, whereas $RDD_{avg} < PL_{avg}$ if DFF.

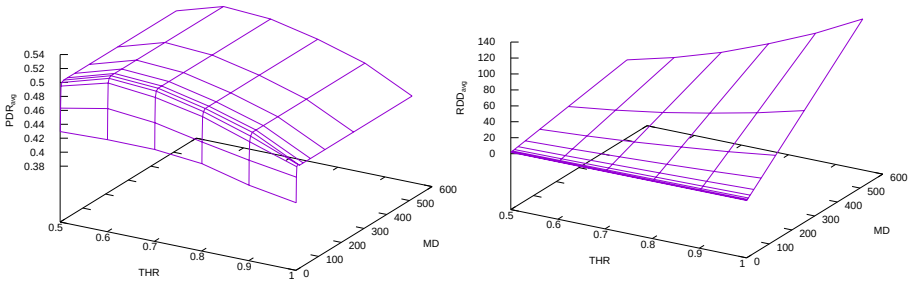
#### 5.1.1. Constant Delay Function

The Constant Delay function serves as a proof of concept that should show us whether the TARA method has the potential to improve the performance of an ad hoc network with untrusted nodes. For the experiments with the Constant Delay function, we chose the most pessimistic scenario SC-50. Experiments were conducted with several combinations of *MD* and *THR* parameter values for both DFT and DFF. The performance results for DFT are plotted in Figure 6 and for DFF in Figure 7.




(**a**) $PDR_{avg}$ (**b**) $RDD_{avg}$

**Figure 6.** Constant Delay function + DFT.




(**a**) $PDR_{avg}$ (**b**) $RDD_{avg}$

**Figure 7.** Constant Delay function + DFF.

It follows from these figures that $PDR_{avg}$ achieves maximum for $THR$ around 0.7. The reason is that the Constant Delay function penalizes nodes with $PDR < THR$. If $THR = 1$, then nodes with high $PDR$ values are penalized by the function, making it impossible to find routes through them. On the other hand, $THR = 0.5$ is a bad choice, as nodes with relatively low $PDR$ values are not penalized. Thus $THR$ value of these boundary values gives worse results.

The maximum value of $PDR_{avg}$ and the minimum value of $RDD_{avg}$ and combinations of delay function parameters to reach them are presented in Table 3.

**Table 3.** Constant Delay function parameters that provided the best values of metrics.

| Flag | Perfor. Metrics | Best Value | $\delta_{metric}$ | Parameters | | Other Metric Values for Context | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $THR$ | $MD$ | $PDR_{avg}$ | $PL_{avg}$ | $RDD_{avg}$ | $RDD_{max}$ |
| DFT | $PDR_{avg}$ | 0.53 | 51% | 0.7 | 16 | - | 7.60 | 17.90 | 102 |
| | $RDD_{avg}$ | 7.22 | 18% | 0.6 | 1 | 0.45 | 6.37 | - | 19 |
| DFF | $PDR_{avg}$ | 0.52 | 54% | 0.6 | 16 | - | 8.57 | 2.82 | 63 |
| | $RDD_{avg}$ | 1.57 | 11% | 0.6 | 1 | 0.43 | 7.70 | - | 18 |

The Constant Delay function does not reflect differences in trust values at all. Smaller values of $MD$ are not enough for trusted path selection; on the other hand, too great $MD$ value does not make a significant improvement of $PDR_{avg}$, but instead worsens route discovery delay $RDD_{avg}$.

The best $PDR_{avg}$ value is achieved for $MD = 16$. Further increase in $MD$ does not bring additional improvement, only worsening the $RDD_{avg}$. Results of the experiments with the Constant Delay function validated the proof of concept. The TARA method improved $PDR_{avg}$ by 51% (DFT) and 54% (DFF) compared to the reference experiment.

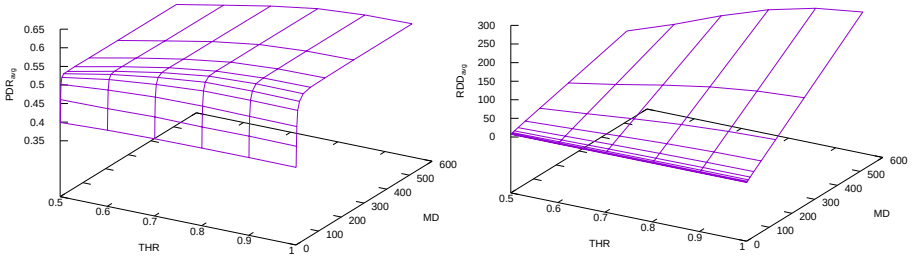
#### 5.1.2. Linear Delay Function

Result plots for the Linear Delay function have similar shapes compared to the Constant Delay function, compare Figures 8 and 9 with Figures 6 and 7. However, with the same configuration parameters, the TARA method with the Linear Delay function achieves significantly better $PDR_{avg}$ and $RDD_{avg}$.

For the Linear Delay function, smaller $MD$ values penalize all nodes almost the same way, and penalization became fairer with greater values.

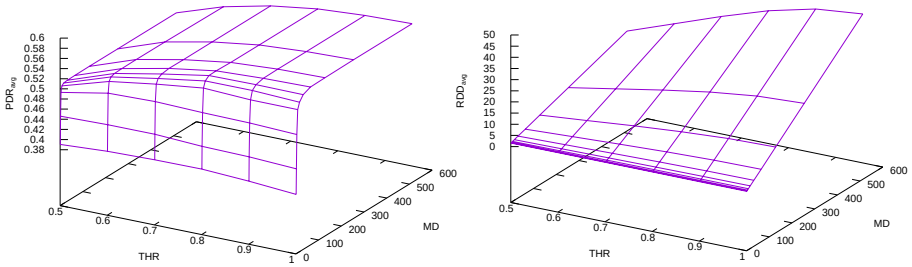
$THR$ value does not influence $PDR_{avg}$ significantly. That is because the linearity of the delay function already plays the same role as the $THR$ in Constant Delay function.

Further dependence found in results is that the $RDD_{avg}$ depends on $MD$ of the Linear Delay function. Increasing $MD$ prolongs the path length and increases the route discovery time. $RDD$s for the Linear Delay function are better than for the Constant Delay one. The explanation is that by penalizing worse nodes more, the Linear Delay function generally delays RREQ messages less in comparison to the Constant Delay function.




(**a**) $PDR_{avg}$ (**b**) $RDD_{avg}$

**Figure 8.** Linear Delay function + DFT.




(**a**) $PDR_{avg}$ (**b**) $RDD_{avg}$

**Figure 9.** Linear Delay function + DFF.

The best metric values and combinations of the Linear Delay function parameters to reach them are presented in Table 4.

**Table 4.** Linear Delay function parameters that provided the best values of metrics.

| Flag | Perfor. Metrics | Best Value | $\delta_{metric}$ | Parameters | | Other Metric Values for Context | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $THR$ | $MD$ | $PDR_{avg}$ | $PL_{avg}$ | $RDD_{avg}$ | $RDD_{max}$ |
| DFT | $PDR_{avg}$ | 0.60 | 71% | 1 | 512 | - | 8.64 | 273.75 | 1626 |
| | $RDD_{avg}$ | 6.79 | 11% | 0.5 | 1 | 0.40 | 6.17 | - | 19 |
| DFF | $PDR_{avg}$ | 0.58 | 72% | 1 | 512 | - | 9.56 | 49.05 | 1129 |
| | $RDD_{avg}$ | 1.50 | 6% | 0.5 | 1 | 0.39 | 7.81 | - | 18 |

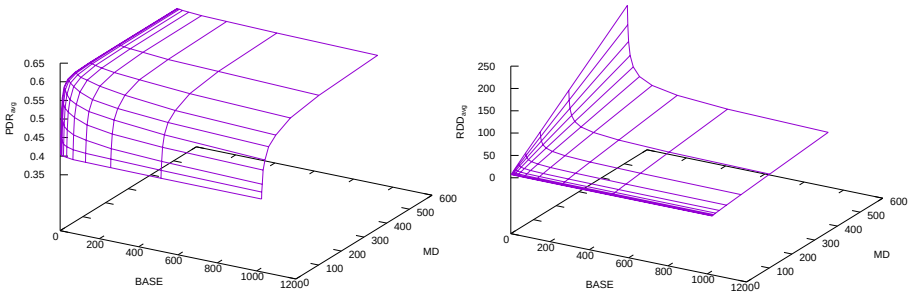
Table 5 presents the combinations of parameters for the Linear Delay function to find the best trade-off between contradictory requirements to maximize $PDR_{avg}$ and minimize $RDD_{avg}$. Clearly, large $MD$ increases $RDD_{avg}$ significantly while $PDR_{avg}$ improves very little. This table can be used to find the best configuration parameters once the real-world application gives us the relative importance of maximizing $PDR_{avg}$ vs minimizing $RDD_{avg}$.

**Table 5.** Linear Delay function parameter combinations for the best trade-off.

| Flag | Parameters | | Results | | | |
|---|---|---|---|---|---|---|
| | $THR$ | $MD$ | $PDR_{avg}$ | $\delta_{PDR_{avg}}$ | $RDD_{avg}$ | $\delta_{RDD_{avg}}$ |
| DFT | 0.5 | 4 | 0.50 | 42% | 7.97 | 30% |
| | 0.6 | 8 | 0.54 | 52% | 9.82 | 60% |
| | 0.8 | 4 | 0.52 | 47% | 9.09 | 48% |
| | 0.9 | 8 | 0.56 | 59% | 11.66 | 90% |
| | 0.9 | 16 | 0.58 | 65% | 15.92 | 160% |
| | 1 | 32 | 0.59 | 69% | 24.90 | 306% |
| DFF | 0.6 | 4 | 0.51 | 51% | 1.73 | 22% |
| | 0.6 | 8 | 0.53 | 57% | 1.95 | 37% |
| | 0.7 | 4 | 0.51 | 51% | 1.80 | 26% |
| | 0.7 | 8 | 0.55 | 61% | 2.07 | 46% |
| | 0.8 | 8 | 0.56 | 65% | 2.17 | 53% |
| | 0.8 | 16 | 0.57 | 68% | 2.82 | 98% |
| | 0.9 | 32 | 0.58 | 70% | 4.41 | 210% |

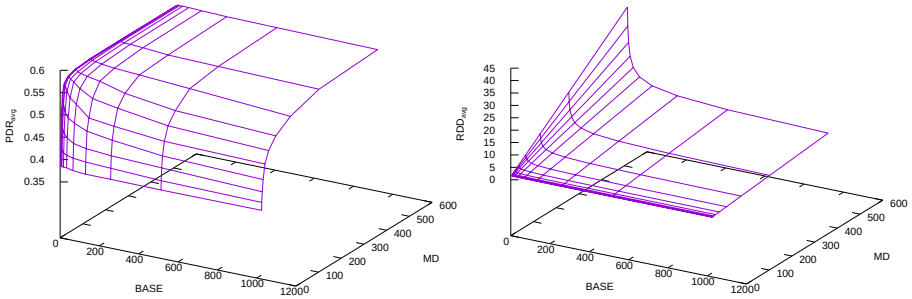
#### 5.1.3. Exponential Delay Function

Results for the Exponential Delay function are shown in Figures 10 and 11.




(**a**) $PDR_{avg}$ (**b**) $RDD_{avg}$

**Figure 10.** Exponential Delay function + DFT.




(**a**) $PDR_{avg}$ (**b**) $RDD_{avg}$

**Figure 11.** Exponential Delay function + DFF.

The best metric values and combinations of Exponential Delay function parameters to reach them are presented in Table 6.

**Table 6.** Exponential Delay function parameters that provided the best values of metrics.

| Flag | Perfor. Metrics | Best Value | $\delta_{metric}$ | Parameters | | Other Metric Values for Context | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | *BASE* | *MD* | $PDR_{avg}$ | $PL_{avg}$ | $RDD_{avg}$ | $RDD_{max}$ |
| DFT | $PDR_{avg}$ | 0.61 | 72% | 8 | 512 | - | 8.82 | 145.93 | 1105 |
| | $RDD_{avg}$ | 6.38 | 4% | 1024 | 1 | 0.40 | 6.14 | - | 17 |
| DFF | $PDR_{avg}$ | 0.59 | 74% | 8 | 512 | - | 9.81 | 26.74 | 892 |
| | $RDD_{avg}$ | 1.45 | 2% | 1024 | 1 | 0.38 | 8.32 | - | 16 |

Table 7 represents the combination of parameters for the Exponential Delay function for the best trade-off between maximizing $PDR_{avg}$ and minimizing $RDD_{avg}$ for the practical application of the TARA method.

If the Exponential Delay function is compared with the Linear Delay function, Exponential Delay penalizes nodes with high *PDR* less, depending on the *BASE* value.

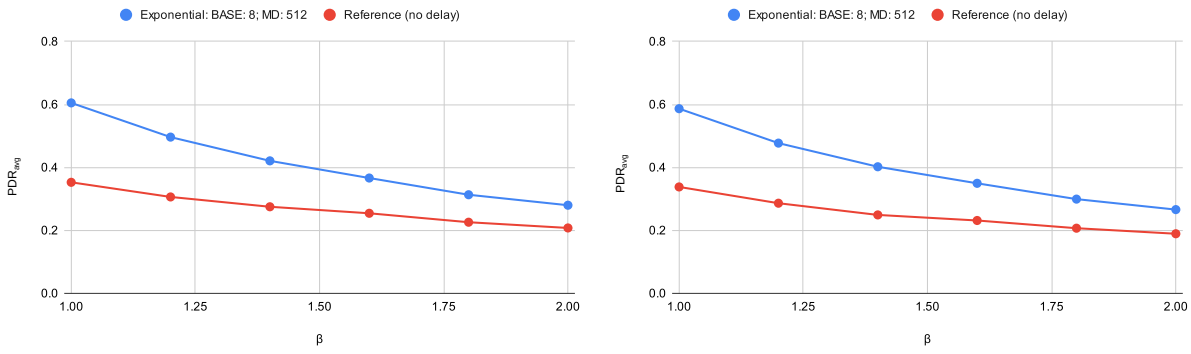
As with previous functions, greater *MD* increases $RDD_{avg}$, while using a greater *BASE* with the same *MD* helps reduce the delay but, at the same time, produces worse $PDR_{avg}$.

### *5.2. Dependence of the TARA Method Performance on the Network Density*

The importance of the $\beta$ parameter that determines the density of the network is shown in Figure 12 for the Exponential Delay function. In networks with greater $\beta$ and, therefore, smaller density, fewer alternative routes can be found, thus less space is left for improving $PDR_{avg}$. Models DFT and DFF differ marginally.

**Table 7.** Exponential Delay function parameter combinations for the best trade-off.

| Flag | Parameters | | Results | | | |
|---|---|---|---|---|---|---|
| | $BASE$ | $MD$ | $PDR_{avg}$ | $\delta_{PDR_{avg}}$ | $RDD_{avg}$ | $\delta_{RDD_{avg}}$ |
| DFT | 4 | 8 | 0.55 | 55% | 10.38 | 69% |
| | 4 | 32 | 0.59 | 69% | 19.26 | 214% |
| | 8 | 16 | 0.57 | 62% | 12.29 | 100% |
| | 16 | 16 | 0.56 | 59% | 11.26 | 84% |
| | 16 | 32 | 0.59 | 66% | 14.94 | 144% |
| | 32 | 8 | 0.52 | 47% | 8.71 | 42% |
| DFF | 4 | 32 | 0.58 | 71% | 3.56 | 151% |
| | 8 | 16 | 0.56 | 66% | 2.35 | 66% |
| | 8 | 64 | 0.58 | 73% | 4.73 | 233% |
| | 16 | 16 | 0.55 | 63% | 2.18 | 53% |
| | 32 | 8 | 0.49 | 46% | 1.78 | 25% |
| | 1024 | 8 | 0.45 | 32% | 1.58 | 11% |

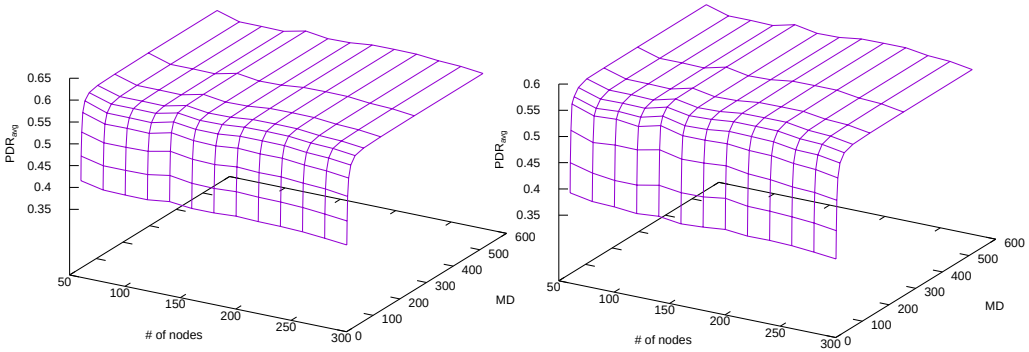



(**a**) DFT (**b**) DFF

**Figure 12.** Dependence of $PDR_{avg}$ on the $\beta$ parameter compared to the reference experiment (with no delay).

### *5.3. Dependence of the TARA Method Performance on the Network Size*

From the experiments, it became clear that the size of the network has no effect on the performance of the TARA method. Figure 13 shows the results for the Linear Delay function.



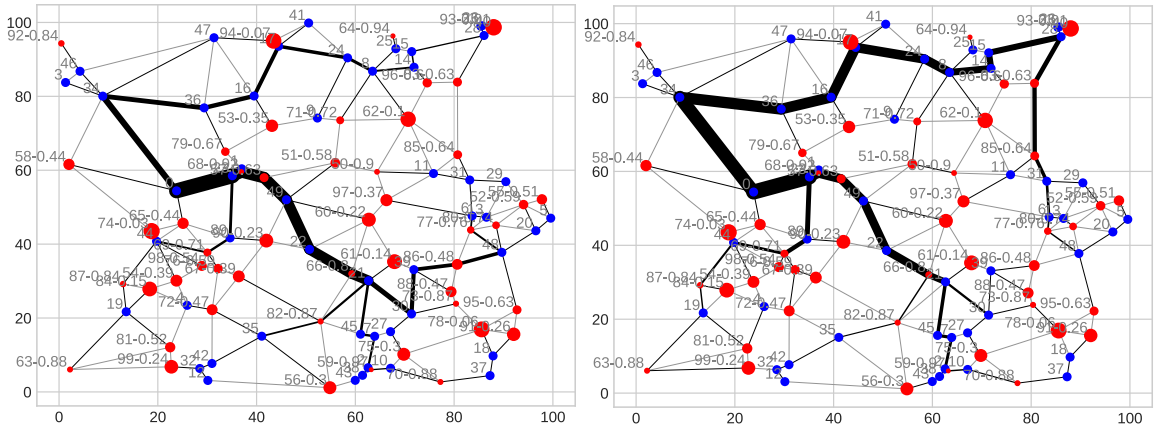

(**a**) DFT (**b**) DFF

**Figure 13.** Dependence of $PDR_{avg}$ on the network size (Linear Delay: $THR = 1$).

### *5.4. Dependence of the TARA Method Performance on the Destination Only Flag*

Figure 14 shows how the setting of the *destination only* flag influences construction of paths. Blue nodes have $PDR = 1$ and red nodes have $PDR < 1$. The size of the red nodes grows with decreasing of their $PDR$ value. The value of $PDR$ is written next to the node ID. Paths in Figure 14 were constructed with the default configuration and Exponential Delay function with $BASE = 8$ and $MD = 512$. Grey links are not used. Links are thicker when

they are used in more paths. DFF topology has longer paths and fewer branches because paths are built on existing ones (see Section 2.3.1).



(**a**) DFF (**b**) DFT

**Figure 14.** Paths created with different *destination only* flag setting.

Performance metrics for Constant, Linear, and Exponential Delay functions with DFT vs. DFF are depicted on Figure 6 vs. Figure 7, Figure 8 vs. Figure 9, and Figure 10 vs. Figure 11, respectively. DFT allows to reach greater values of $PDR_{avg}$ for the price of longer $RDD_{avg}$. $RDD_{avg}$ values of DFT are by one order greater than the corresponding results of DFF (see Table 8, compare DFT and DFF sections).

### *5.5. Dependence of the TARA Method Performance on Trust Distribution Scenarios*

We have conducted several experiments to analyze how scenarios of malicious behavior influence the performance of the TARA method. The percentage of potentially untrusted nodes is fixed in each scenario, ranging from 10% to 50%. Experiments with all combinations of the *BASE* and *MD* parameters were helpful, but for simplicity and clear comparison, Table 8 shows the results only for $BASE = 8$ and $MD = 512$.

**Table 8.** Different trust distribution scenarios.

| Flag | Scen. | $PDR_{avg}$ | Ref.$PDR_{avg}$ | $\delta_{PDR_{avg}}$ | $RDD_{avg}$ | Ref.$RDD_{avg}$ | $\delta_{RDD_{avg}}$ |
|---|---|---|---|---|---|---|---|
| DFT | SC-10 | 0.99 | 0.83 | 19% | 10.71 | 6.13 | 75% |
| | SC-20 | 0.95 | 0.69 | 39% | 23.08 | 6.13 | 276% |
| | SC-30 | 0.89 | 0.55 | 63% | 39.77 | 6.13 | 549% |
| | SC-40 | 0.75 | 0.44 | 71% | 84.63 | 6.13 | 1281% |
| | SC-50 | 0.61 | 0.35 | 72% | 145.93 | 6.13 | 2281% |
| DFF | SC-10 | 0.99 | 0.80 | 24% | 2.56 | 1.42 | 80% |
| | SC-20 | 0.95 | 0.64 | 49% | 4.54 | 1.42 | 220% |
| | SC-30 | 0.89 | 0.50 | 76% | 8.92 | 1.42 | 527% |
| | SC-40 | 0.74 | 0.42 | 78% | 16.79 | 1.42 | 1081% |
| | SC-50 | 0.59 | 0.34 | 74% | 26.74 | 1.42 | 1781% |

For scenarios with few untrusted nodes, the TARA method is not as efficient. Its benefit increases with the percentage of untrusted nodes.

### *5.6. Summary Evaluation of Experiment Results*

Results of experiments for all delay functions show that increasing maximum delay value *MD* improves network-wide average *PDR* significantly, but the effect weakens with greater values. Moreover, the initial increase in *MD* gives a quick boost for the *PDR* metric, but a further increase in *MD* is not effective. Improvement of *PDR* by the TARA method, compared to the reference experiment with no delay function applied, is significant for all delay functions, but Exponential Delay provided the best improvement.

*THR* values do not influence $PDR_{avg}$ as significantly as the *MD* values do. For the Linear Delay function, *THR* has influence mainly on the $RDD_{avg}$. The reason is that nodes with *PDR* above *THR* are not penalized, so no delay is added for them. Thus, route discovery, in general, is faster.

If the Exponential Delay function is compared with the Linear Delay function, Exponential Delay penalization better reflects the differences in trust values of nodes. Depending on the bending of the exponential curve, penalization of nodes with high *PDR* starts slowly, which is sufficient, and penalization of nodes with low *PDR* value grows rapidly.

$PL_{avg}$ and $RDD_{avg}$ metrics need to be minimized, contrary to the average $PDR_{avg}$, which needs to be maximized. Due to the delay function, improving the *PDR* means worsening *PL* and *RDD*. Greater *MD* value prolongs the path length and increases the route discovery time.

In general, applying the TARA method approach in more dense networks gives greater $PDR_{avg}$ improvement. At the same time, the network size, under the assumption of the same density, has no influence on the performance of the TARA method.

DFT provides more trusted and shorter paths for the cost of much greater $RDD_{avg}$. DFF copes better with discovering paths quickly but produces longer paths and cannot reach so high $PDR_{avg}$ as DFT.

Experiments with different scenarios showed that networks with a greater amount of untrusted nodes have a greater potential for improvement.

The aim of the experiments was to find the best combination of TARA method parameters to achieve two contradictory goals, which should also be kept in mind when applying the TARA method:

- maximize the average packet delivery ratio in the network $PDR_{avg}$;
- minimize the average route discovery delay $RDD_{avg}$.

Combinations of delay functions parameters that produce relatively high $PDR_{avg}$ while $RDD_{avg}$ is low are presented in Tables 5 and 7.

## 6. Discussion

According to the AODV RFC [14], if a node obtains more replies for its RREQ message, it also processes them, and if they have a fresher sequence number, or the same sequence number, but the number of hops is smaller, then the routing table is updated with the new route. This is something to keep in mind when implementing the TARA method.

If there are alternative paths and the route discovery is delayed according to the trust of the nodes along those paths, the first RREP message is the fastest, meaning that path is the most trusted one. RREP for the other alternative paths may take longer to arrive, implying that those paths pass through less trusted nodes. The less trusted path could be shorter (have fewer hops), and AODV will accept it. The goal of our method is not to change the implementation of AODV. Our interlayer that delays packets should discard the repeated RREP messages to the same RREQ message.

Another aspect to consider is the behavior of the DSR protocol, which stores several route entries for the same destination. The problem is identical to AODV, and the interlayer can address it in the same way.

## 7. Conclusions

Cooperation and trust establishment are the significant challenges in ad hoc network security. Earlier, we have proposed a method to evaluate the trust of the particular node in an ad hoc network using neural networks. Research presented in this article is devoted to designing a way to integrate trust in reactive routing protocols. The challenge was to do it without changing the routing algorithm, i.e., to influence the routing decision from the outside.

This article introduces a method to enhance trust in reactive routing protocols, called TARA, and analyses the implementation and settings of the TARA method. The method's

main idea is to delay the route discovery messages of untrusted nodes, forcing the more trusted paths to be chosen.

Results of experiments show that different delay functions improve the average packet delivery ratio up to 78% for DFF and 72% for DFT (Table 8). Generally, the *destination only* flag should be activated to provide greater trust if route discovery time is not critical. In essence, when using the method, parameters should be selected to trade-off between maximizing trust and minimizing route discovery delay at the same time.

The TARA method fulfilled the research goal, namely, enhancement of reactive ad hoc routing protocol with trust mechanism without the need to change the implementation of the routing itself. Various parameter combinations were tested and analyzed to provide recommendations for choosing TARA method parameters, depending on the application in a specific ad hoc network.

The TARA method was discussed in the context of implementation in an ad hoc network, and future work could address the concerns.

**Author Contributions:** Conceptualization, Y.T.; methodology, Y.T.; software, Y.T.; validation, Y.T. and P.T.; formal analysis, Y.T.; investigation, Y.T.; writing—original draft preparation, Y.T.; writing—review and editing, Y.T. and P.T.; visualization, Y.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Link to the git repository with simulator source code and results: https://gitlab.fit.cvut.cz/trofiyel/TARA (accessed on 13 December 2021).

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| AODV | Ad hoc On-demand Distance Vector |
| DFF | *Destination only* Flag False |
| DFT | *Destination only* Flag True |
| DSR | Dynamic Source Routing |
| DYMO | DYnamic Manet On-demand |
| MANET | Mobile Ad hoc NETwork |
| MD | Maximum Delay value |
| NN | Neural Network |
| OLSR | Optimized Link State Routing |
| PDR | Packet Delivery Ratio |
| RFC | Request For Comments |
| RREP | Route REPly |
| RREQ | Route REQuest |
| TARA | Trust-Aware Reactive Ad hoc routing |
| THR | THReshold value |

## References

1. Buttyán, L.; Hubaux, J.P. Trust assumptions and adversary models. In *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*; Cambridge University Press: Cambridge, UK, 2007; pp. 74–82. [CrossRef]
2. Alnumay, W.; Ghosh, U.; Chatterjee, P. A Trust-Based Predictive Model for Mobile Ad Hoc Network in Internet of Things. *Sensors* **2019**, *19*, 1467. [CrossRef] [PubMed]
3. Pathan, M.S.; Zhu, N.; He, J.; Zardari, Z.A.; Memon, M.Q.; Hussain, M.I. An Efficient Trust-Based Scheme for Secure and Quality of Service Routing in MANETs. *Future Internet* **2018**, *10*, 16. [CrossRef]
4. Li, X.; Jia, Z.; Zhang, P.; Zhang, R.; Wang, H. Trust-based on-demand multipath routing in mobile ad hoc networks. *IET Inf. Secur.* **2010**, *4*, 212. [CrossRef]
5. Marchang, N.; Datta, R. Light-weight trust-based routing protocol for mobile ad hoc networks. *IET Inf. Secur.* **2012**, *6*, 77. [CrossRef]

6. Venkanna, U.; Agarwal, J.K.; Velusamy, R.L. A Cooperative Routing for MANET Based on Distributed Trust and Energy Management. *Wirel. Pers. Commun.* **2015**, *81*, 961–979. [CrossRef]
7. Simaremare, H.; Abouaissa, A.; Sari, R.F.; Lorenz, P. Security and performance enhancement of AODV routing protocol. *Int. J. Commun. Syst.* **2015**, *28*, 2003–2019. [CrossRef]
8. Buchegger, S.; Le Boudec, J.Y. Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks. In Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands, Spain, 9–11 January 2002; IEEE Computer Society: Canary Islands, Spain, 2002; pp. 403–410. [CrossRef]
9. Pirzada, A.A.; Datta, A.; McDonald, C. Incorporating trust and reputation in the DSR protocol for dependable routing. *Comput. Commun.* **2006**, *29*, 2806–2821. [CrossRef]
10. Hatzivasilis, G.; Papaefstathiou, I.; Manifavas, C. SCOTRES: Secure Routing for IoT and CPS. *IEEE Internet Things J.* **2017**, *4*, 2129–2141. [CrossRef]
11. Trofimova, Y.; Moucha, A.M.; Tvrdik, P. Application of neural networks for decision making and evaluation of trust in ad-hoc networks. In Proceedings of the 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 26–30 June 2017; pp. 371–377. [CrossRef]
12. Retana, A. Mobile Ad-hoc Networks (MANET) Working Group Charter. Available online: https://datatracker.ietf.org/doc/charter-ietf-manet/ (accessed on 13 December 2021).
13. Loo, J.; Mauri, J.L.; Ortiz, J.H. (Eds.) *Mobile Ad Hoc Networks: Current Status and Future Trends*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2012. doi: 10.1201/b11447. [CrossRef]
14. Das, S.R.; Perkins, C.E.; Belding-Royer, E.M. *RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing*; Internet Engineering Task Force: Fremont, CA, USA, 2003. Available online: https://datatracker.ietf.org/doc/rfc3561/ (accessed on 13 December 2021).
15. Eissa, T.; Abdul Razak, S.; Khokhar, R.H.; Samian, N. Trust-based routing mechanism in MANET: Design and implementation. *Mob. Networks Appl.* **2013**, *18*, 666–677. [CrossRef]
16. Aggarwal, A.; Gandhi, S.; Chaubey, N.; Jani, K.A. Trust Based Secure on Demand Routing Protocol (TSDRP) for MANETs. In Proceedings of the 2014 Fourth International Conference on Advanced Computing Communication Technologies, Rohtak, India, 8–9 February 2014; pp. 432–438. [CrossRef]
17. Wang, J.; Liu, Y.; Jiao, Y. Building a trusted route in a mobile ad hoc network considering communication reliability and path length. *J. Netw. Comput. Appl.* **2011**, *34*, 1138–1149. [CrossRef]
18. Conti, M.; Giordano, S. Mobile ad hoc networking: Milestones, challenges, and new research directions. *IEEE Commun. Mag.* **2014**, *52*, 85–96. [CrossRef]
19. Karapistoli, E.; Mampentzidou, I.; Economides, A.A. Environmental monitoring based on the wireless sensor networking technology: A survey of real-world applications. *Int. J. Agric. Environ. Inf. Syst. (IJAEIS)* **2014**, *5*, 1–39. [CrossRef]
20. Perkins, C.; Royer, E. Ad-hoc on-demand distance vector routing. In Proceedings of the WMCSA'99—Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, USA, 25–26 February 1999; pp. 90–100. [CrossRef]
21. Saini, T.K.; Sharma, S.C. Recent advancements, review analysis, and extensions of the AODV with the illustration of the applied concept. *Ad Hoc Netw.* **2020**, *103*, 102148. [CrossRef]
22. Johnson, D.; Hu, Y.; Maltz, D. *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728*; Internet Engineering Task Force: Fremont, CA, USA, 2007. Available online: https://datatracker.ietf.org/doc/html/rfc4728 (accessed on 13 December 2021).
23. Perkins, C.; Royer, E.; Das, S.; Marina, M. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Pers. Commun.* **2001**, *8*, 16–28. [CrossRef]
24. Park, V.; Corson, M. A highly adaptive distributed routing algorithm for mobile wireless networks. In Proceedings of the INFOCOM '97, Kobe, Japan, 7–11 April 1997; IEEE Computer Society Press: Kobe, Japan, 1997; Volume 3, pp. 1405–1413. [CrossRef]
25. Perkins, C.E.; Ratliff, S.; Dowdell, J.; Steenbrink, L.; Pritchard, V. *Ad Hoc On-Demand Distance Vector Version 2 (AODVv2) Routing*; Work in Progress; Internet Engineering Task Force: Fremont, CA, USA, 2019. Available online: https://datatracker.ietf.org/doc/draft-perkins-manet-aodvv2/03/ (accessed on 13 December 2021).
26. Trofimova, Y.; Fesl, J.; Moucha, A.M. Performance Analysis of Neural Network Approach for Evaluation of Trust in Ad-hoc Networks. In Proceedings of the 2021 11th International Conference on Advanced Computer Information Technologies (ACIT), Deggendorf, Germany, 15–17 September 2021.
27. Bhardwaj, M.; Misra, S.; Xue, G. Distributed Topology Control in Wireless Ad Hoc Networks using $\beta$-Skeletons. In Proceedings of the IEEE Workshop on High Performance Switching and Routing (HPSR), Hong Kong, China, 12–14 May 2005; p. 5.