Decentralized Evaluation of Trust in Ad Hoc Networks using Neural Networks

Yelena Trofimova
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
yelena.trofimova@fit.cvut.cz

Viktor Černý
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
victor.cerny@fit.cvut.cz

Jan Fesl
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
jan.fesl@fit.cvut.cz

***Abstract*—Trust is an essential concept in ad hoc network security. Creating and maintaining trusted relationships between nodes is a challenging task. This paper proposes a decentralized method for evaluating trust in ad hoc networks. The method uses neural networks and local information to predict the trust of neighboring nodes. The method was compared with the original centralized version, showing that even without global information knowledge, the method has, on average, 97% accuracy in classification and 94% in regression problem. An important contribution of this paper is overcoming the main limitation of the original method, which is the centralized evaluation of trust. Moreover, the decentralized method output is a perfect fit to use as an input to enhance routing in ad hoc networks.**

***Index Terms*—ad hoc network, trust, neural network, packet delivery ratio, decentralized.**

I. Introduction

The inherent properties of ad hoc communication, such as the absence of central authority and the use of intermediate nodes to forward data, require cooperation between nodes for the network to function correctly. Trust in Wireless Ad hoc Networks (WANETs) is an important concept that helps improve the networks' reliability and performance by enforcing cooperation. Trust is a measure of confidence in the ability of a node to behave according to expectations. Trust values can be used by nodes in interactions with others or for access control, intrusion detection, or secure routing. Numerous methods to govern trust among communicating entities were proposed. One of them, a neural network (NN) approach, was proposed in [1] and utilized NNs to evaluate the trust of nodes in a WANET. The main drawback of the method was collecting information from all nodes. The primary motivation behind this paper was to solve this drawback by proposing a decentralized approach. Local information from the point of view of a particular node is used as an input for NNs.

The originally proposed NN method is referred to as Neural Network Trust Evaluation in Ad hoc networks (NeNTEA) in the further text. Let us call the decentralized version DeNeNTEA.

The rest of the paper is organized as follows. Subsection I-A provides an overview of the related work. Section II outlines a method to incorporate trust in ad hoc networks. Section III describes the original approach to the evaluation of trust. The following Section IV rectifies the original paper statistics. The description of a new decentralized method is covered in Section V. The results of simulations are presented in Section VI, where the performance is compared to the original method. Finally, Section VII concludes the paper.

A. Related work

The proposed trust management approaches are based on game theory [2], [3], fuzzy logic [4], [5], blockchain [6], artificial intelligence, and much more. Let us focus on the latter.

Ham et al. [7] proposed using NNs to estimate trust based on the internal attributes of the node rather than its observed behavior. The attributes are definable critical processes and physical characteristics associated with network security, e.g., operating system patches, encryption keys, hardware configurations, exposure and location of the node, and others.

The combination of fuzzy logic and NNs based tools was proposed by Sinha and Paul [8]. Trust is first calculated based on the packet drop of the node using fuzzy logic. NN is used in the second step to enhance accuracy. NN is trained offline on the scenario without attack and has five inputs: packet drop, packet forward, packet received, packet sent, and residual energy. The output is the probability that the node is malicious.

Siddiqui et al. [9] utilized several machine learning algorithms to allocate the optimal weights of parameters for trust calculations and select an optimal threshold value.

Repantis and Kalogeraki proposed decentralized trust management middleware for ad hoc peer-to-peer networks based on reputation [10]. Reputation is measured from direct interactions (ratings of the service provided) and information from other peers. They proposed to store the reputation information in a group of peers.

II. Trust-Aware Reactive Ad hoc routing (TARA)

The most efficient way to incorporate trust in ad hoc networks to increase network reliability is the usage of trust values to select more trustworthy paths. To incorporate the trust of nodes into reactive ad hoc routing protocols, we proposed a method called Trust-Aware Reactive Ad Hoc routing (TARA) [11].

The TARA method is advantageous because it does not require changes to the routing protocol, thus, it is not protocol-dependent. TARA influences the routing decisions from outside of the routing protocol. The route discovery phase is influenced by trust estimations to choose more trusted paths. During the route discovery phase, the Route Request (RREQ) packets over untrusted nodes are penalized by the delay. In [11], we proposed, simulated, and evaluated three different strategies for the optimal RREQ packets delay value setting. The delaying is implemented in an additional layer. For each RREQ packet this interlayer decides how long the packet will be delayed before sending it to the network layer routing protocol.

Our previous results proved that using TARA significantly improves the performance of reactive ad hoc routing protocols. Trust values suitable for TARA method calculations can be calculated by the NeNTEA method. The weak point of TARA stands in its centralization – the requirement of the sink node existence. The newly proposed method DeNeNTEA eliminates this fact and allows the TARA method to be functional in a fully decentralized version.

III. NeNTEA method description

A. Definitions and main idea

The *packet delivery ratio (PDR) of a node* N_i is defined as follows:

$$PDR(N_i) = \frac{\phi(N_i)}{\sigma(N_i)}, \tag{1}$$

where $\phi(N_i)$ is the number of data packets correctly forwarded by node N_i and $\sigma(N_i)$ is the total number of data packets sent to node N_i that were supposed to be forwarded.

The *trust of a node* N_i is its estimated $PDR(N_i)$ and it is a real number between 0 and 1: $PDR(N_i) \in \mathbb{R}, 0 \leq PDR(N_i) \leq 1$.

The main idea of the NeNTEA method is that the trust of the node can be determined from the trust of the paths in which the node participates. The precise weight of each path trust is determined by the NN through learning. The NN schema is recalled in Fig. 1.

Each node in the network, which forwards data of other nodes, is a different problem because the node participates in a particular set of paths $\mathcal{P}_{problem}$, and its $PDR_{problem\_node}$ is dependent on the PDRs of those paths. Another *problem node* participates in another set of paths, thus for each of the problem nodes, a particular neural network is constructed and trained. The source nodes of the paths from the set $\mathcal{P}_{problem}$ provide input for the neural network. The number of source nodes is denoted as γ. Each input for the neural network is calculated as follows. A source node has p_i paths, and it averages the PDRs of those paths. For example, p_1 means the number of paths that the first source node provides. PDR_{1i} is the PDR of the i^{th} path in the first set of paths. The number of inputs for a particular neural network is γ, given by the number of source nodes. The output is the PDR value of the problem node [12].

The primary advantage of the proposed approach is that trust calculations can be done on-the-fly, by utilizing statistics collected while ad hoc network nodes communicate, thus minimizing the related delay.

B. Simulation and Datasets

The original simulation considered ad hoc networks of 20 nodes. Topologies were generated in the OMNeT++ simulator. The positions of the nodes were randomly generated within an area of 600m $\times$ 600m. Each node has an omnidirectional antenna with a range of 250m. The AODV algorithm was used for routing.

Each problem node represents one problem instance with one instance file for which one NN was constructed and trained. In our experiments, we set the trust threshold $\tau = 0.5$. Later we performed experiments to analyze the influence of threshold value on the performance metrics of the NeNTEA method [12]. That work also contained a performance analysis of how the method is sensitive to incorrect input data.

Three datasets were constructed for the identical 100 WANET topologies with 20 nodes each. Datasets differ in node PDRs generation:

1) *Dataset 1,* ***Uniform-one****:* represents WANET with one untrusted node. Node PDRs are generated using a uniform distribution.
2) *Dataset 2,* ***Uniform-all****:* represents WANET where all nodes can be untrusted. Node PDRs are generated using a uniform distribution.
3) *Dataset 3,* ***Normal-one****:* represents WANET with one untrusted node (similar to Dataset 1), but node PDRs are generated using a normal distribution.

Various combinations of datasets were used for training and testing; the results of the most interesting experiments were published. Experiments 1-5 solve the classification problem, and Experiments 6 and 7 solve the regression problem.

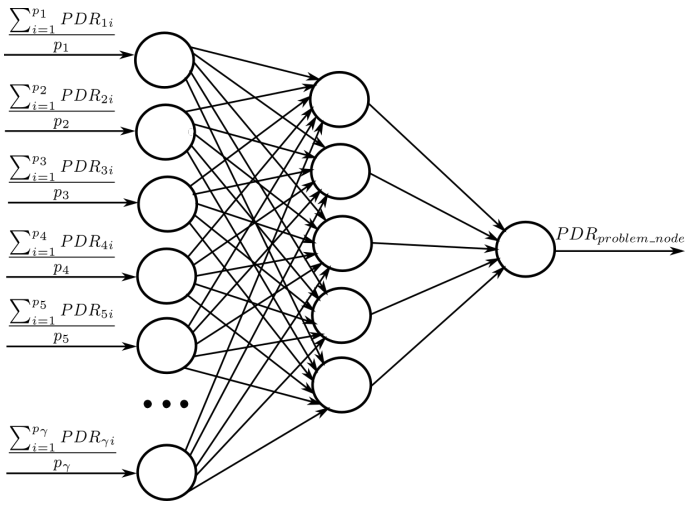

Fig. 1. Schema of the NN used in the original method.

C. Statistics calculation

The *metric* R_D for measuring the quality of the detection of untrusted nodes (classification problem) is defined as the ratio of the number $C_{success}$ of data instances that were correctly predicted to the total number C_{all} of data instances in the instance file and is measured in %:

$$R_D = \frac{C_{success}}{C_{all}} * 100\% \tag{2}$$

The *metric* R_E for measuring the quality of the estimation of a node trust value (regression problem) should represent how the obtained results differ from the correct values. It is defined as:

$$R_E = (1 - \frac{\sum_{i=1}^{m} |Y_i - Y_i'|}{C_{all}}) * 100\% \tag{3}$$

where m is the size of the instance file (number of data instances in the instance file), $|Y_i - Y_i'|$ is an absolute difference between the predicted output Y_i' value from the target output value Y_i.

The values of R_D or R_E are calculated for every instance file.

IV. Correction of the original paper statistics

There was a minor error in calculating the results statistics in the original paper. In more detail, each topology has a number of problem instances. For each problem instance, statistics R (R_D or R_E, depending on if classification or regression problem is solved) have been obtained. Then, originally, the results were calculated as follows:

- collect R of all problem instances of one topology;
- calculate their average;
- calculate the median, minimum, and maximum of those averages over 100 topologies.

The original statistical calculations are represented in Fig. 2.

To calculate the statistics correctly, we need first to calculate the median, minimum, and maximum over the problem instances in the same topology and then calculate the average through 100 topologies to minimize the statistical error; see Fig. 3.

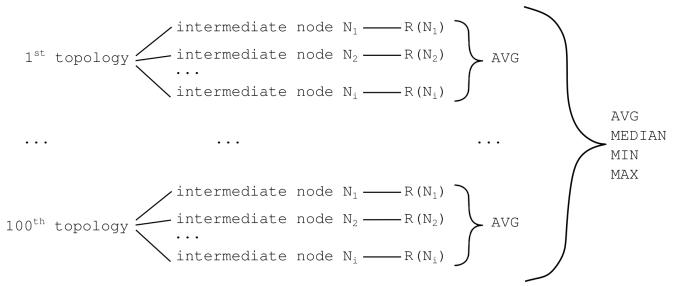

Fig. 2. Published calculation of statistics for NeNTEA.

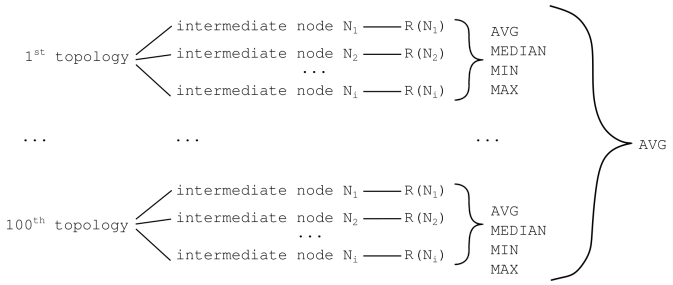

Fig. 3. Corrected calculation of statistics for NeNTEA.

Table I presents the published and corrected statistics of the results of the experiments.

Also, in the original paper, the statistics for experiment 7 were mistakenly switched with experiment 8, where the NNs were trained on Dataset 3 (Normal-one) and tested on Dataset 2 (Uniform-all). So, the values from the original paper for Experiment 7 (92.49, 92.63, 86.79, 97.73) should be actually (94.19, 94.49, 90.84, 95.81).

Recalculating the statistics showed that the results are even better than those published: although minimums are worse, medians and maximums are greater, see Fig. 4. The explanation for such an outcome follows. In the published paper, statistics (such as median, minimum and maximum) were calculated over the averages of each topology, meaning that we have lost information about extreme values. Thus corrected calculations showed lower minimums and greater maximums. Corrected calculations also produced better

TABLE I
Correction of results statistics published in [1]

Experiment number	Dataset trained	Dataset tested	Published				Corrected			
			Average	Median	Minimum	Maximum	Average	Median	Minimum	Maximum
Experiment 1	Uniform-one	Uniform-one	98.78	98.85	96.94	99.75	98.78	99.55	93.04	99.90
Experiment 2	Uniform-one	Normal-one	97.81	97.91	94.35	99.67	97.81	99.19	88.15	99.82
Experiment 3	Normal-one	Uniform-one	98.54	98.62	96.70	99.64	98.54	99.46	91.69	99.87
Experiment 4	Normal-one	Normal-one	98.22	98.34	95.14	99.74	98.22	99.49	88.96	99.88
Experiment 5	Uniform-one	Uniform-all	96.96	96.99	93.26	99.57	96.96	98.99	84.39	99.81
Experiment 6	Uniform-all	Uniform-all	96.60	96.75	94.12	97.76	96.60	97.63	88.57	98.05
Experiment 7	Uniform-all	Normal-one	94.19	94.49	90.84	95.81	94.19	95.52	84.62	96.08

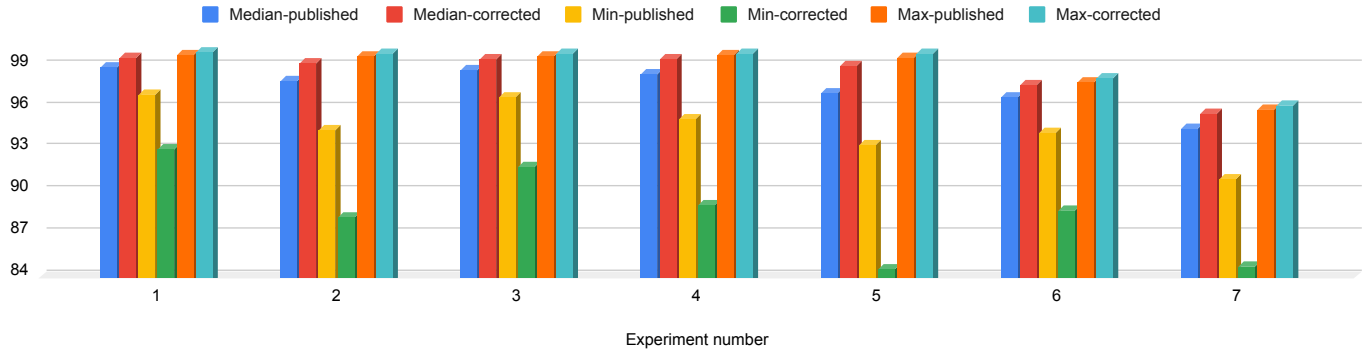

Fig. 4. Comparison of statistics published for the original method and corrected ones.

medians that confirmed the efficiency of the published method.

V. DeNeNTEA method description

The main idea behind the method is the same, but the decentralized version uses only local information that can be gathered on one particular node. Every node creates and trains a neural network for every neighbor. The input on this NN is the PDRs of the paths that pass through the neighbor node, and the output is the PDR (trust) of the particular neighbor, see Fig. 5. In this case, the number of inputs γ is given by the number of paths where the particular neighbor is the next hop.

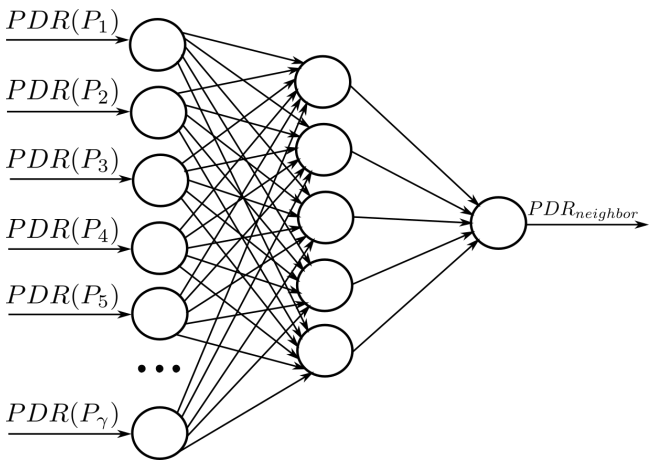

Fig. 5. Schema of the NN used for the decentralized version of our method.

Having NNs for the neighbors on each node leads to the possibility that the trust value of the particular node may differ in the separate instances of the method running on different nodes.

Before learning can be performed, it is necessary to gather information about the nodes along the paths to generate data for training. In the case of DSR, we have this information for free [13]. In the case of AODV, this could be provided either by some master/sink node or some probe packets can be sent through the network to collect data, or the information can be piggybacked on the data message packets. After this information is collected, data for learning can be generated.

VI. Performance comparison

DeNeNTEA experiments were carried out on the identical ad hoc network topologies generated for NeNTEA, for the relevance of the results.

Statistics for the decentralized version are calculated in two ways. First, *through topologies* (similar to the original solution), see Fig. 6:

- take all results of problem instances from the particular topology;
- calculate their average, median, minimum, and maximum;
- calculate average through 100 topologies to minimize statistical error and influence of the topological properties of the particular ad hoc network.

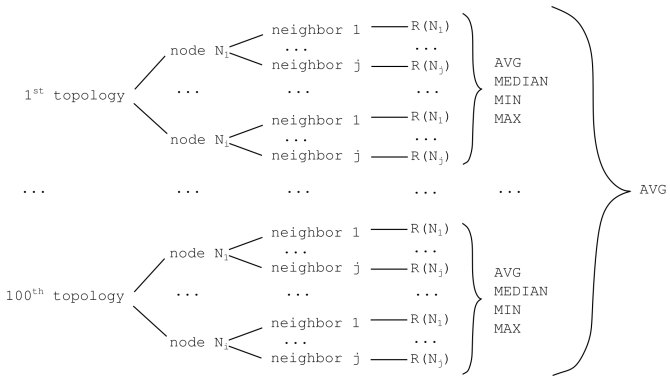

Fig. 6. Calculation of statistics for DeNeNTEA through topologies.

Secondly, through the decentralized instances of our method (running on each node) – *through nodes*. Each node in the network runs the same algorithm, and statistics are calculated as follows (see Fig. 7):

- average, median, minimum, and maximum are calculated among all problem instances of the particular node;
- their average is calculated through all the nodes in all topologies.

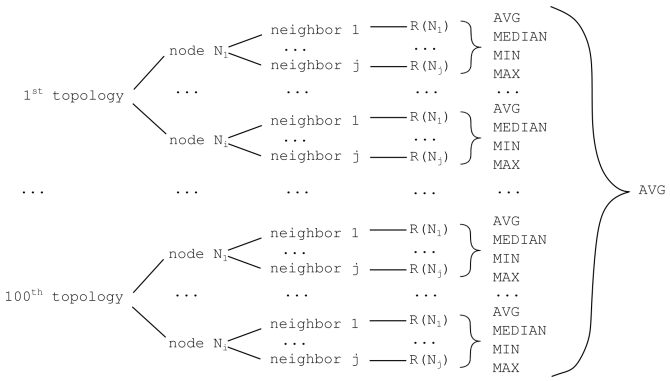

Fig. 7. Calculation of statistics for DeNeNTEA through nodes.

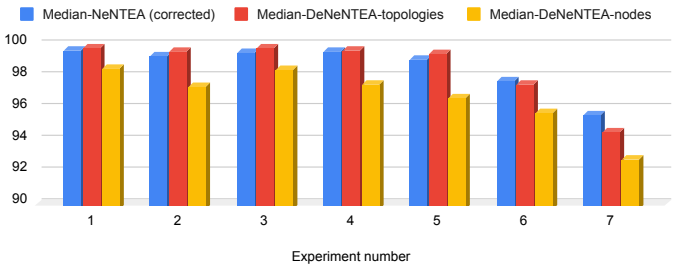

Fig. 8. Comparison of medians.

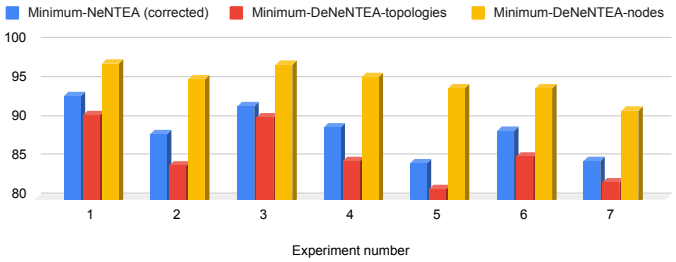

Fig. 9. Comparison of minimums.

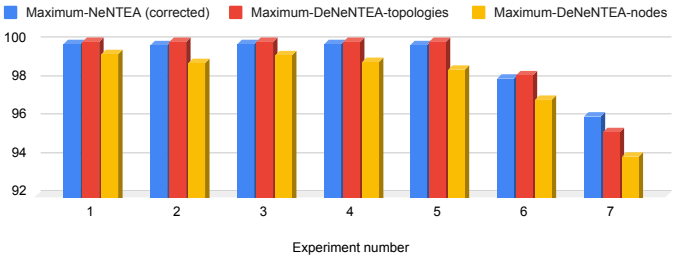

Fig. 10. Comparison of maximums.

Table II shows the statistics of the decentralized version calculated in both ways. The results show that even with only local information, DeNeNTEA performance is satisfactory and not much lower than for the centralized method.

The particular statistics comparison of DeNeNTEA method and the NeNTEA method (corrected in Section IV) are visualized in Figs. 8, 9, and 10.

Medians are generally worse when calculating through problem instances (nodes) compared to calculation through topologies, where all problem instances from the topology are put together, resulting in the prevailing of better results.

Minimums look worse for the calculations through topologies, as such calculations imply a greater sample set, thus a greater possibility of a worse minimum. When calculating through problem instances (nodes), this information is partially lost in the averaging. That also holds for maximums, but the other way around: having a greater sample set (through topologies) implies a greater possibility for a better maximum.

VII. Conclusions

The analysis of the original method results led to discovering the flaw in the statistics calculations. Correction of the calculations showed that the results are even better than published.

TABLE II
DeNeNTEA result statistics through topologies and problem instances(nodes))

Experiment number	Dataset trained	Dataset tested	Through topologies				Through nodes			
			Average	Median	Minimum	Maximum	Average	Median	Minimum	Maximum
Experiment 1	Uniform-one	Uniform-one	98.23	99.69	90.63	100.00	98.33	98.42	97.23	99.33
Experiment 2	Uniform-one	Normal-one	96.99	99.53	84.15	99.99	97.16	97.32	95.28	98.86
Experiment 3	Normal-one	Uniform-one	98.18	99.73	90.26	99.99	98.28	98.38	97.13	99.32
Experiment 4	Normal-one	Normal-one	97.15	99.61	84.64	100.00	97.30	97.46	95.50	98.94
Experiment 5	Uniform-one	Uniform-all	96.18	99.38	81.03	100.00	96.37	96.57	94.03	98.52
Experiment 6	Uniform-all	Uniform-all	95.41	97.43	85.33	98.25	95.55	95.67	94.01	96.96
Experiment 7	Uniform-all	Normal-one	92.50	94.45	81.98	95.28	92.61	92.74	91.11	94.00

The newly proposed decentralized method overcomes the limitations of the published one, preserving its benefits. Experimental results show the efficiency of the DeNeNTEA. Compared to the original, the performance is slightly worse, but the method has the benefit of decentralized processing.

Moreover, the DeNeNTEA method can be further applied to improve the performance of reactive ad hoc routing protocols using the TARA method [11]. Application of the TARA method requires knowledge of neighbor's trust, which is precisely the output of DeNeNTEA. This combination creates a fully decentralized mechanism to enhance any reactive ad hoc routing protocol with trust.

REFERENCES

[1] Y. Trofimova, A. M. Moucha, and P. Tvrdik, "Application of neural networks for decision making and evaluation of trust in ad-hoc networks," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, June 2017, pp. 371–377.

[2] H.-J. Li, Q. Wang, S. Liu, and J. Hu, "Exploring the trust management mechanism in self-organizing complex network based on game theory," *Physica A: Statistical Mechanics and its Applications*, vol. 542, p. 123514, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437119319600

[3] D. Das, K. Majumder, and A. Dasgupta, "Selfish node detection and low cost data transmission in manet using game theory," *Procedia Computer Science*, vol. 54, pp. 92–101, 2015, eleventh International Conference on Communication Networks, ICCN 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21-23, 2015, Bangalore, India. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915013356

[4] A. Beheshtiasl and A. Ghaffari, "Secure and Trust-Aware Routing Scheme in Wireless Sensor Networks," *Wireless Personal Communications*, vol. 107, no. 4, pp. 1799–1814, Aug. 2019. [Online]. Available: http://link.springer.com/10.1007/s11277-019-06357-3

[5] S. Guleng, C. Wu, X. Chen, X. Wang, T. Yoshinaga, and Y. Ji, "Decentralized Trust Evaluation in Vehicular Internet of Things," *IEEE Access*, vol. 7, pp. 15 980–15 988, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8616885/

[6] Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. M. Leung, "Blockchain-Based Decentralized Trust Management in Vehicular Networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1495–1505, Apr. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8358773/

[7] F. M. Ham, E. Y. Imana, A. Ondi, R. Ford, W. Allen, and M. Reedy, "Reputation prediction in mobile ad hoc networks using rbf neural networks," in *Engineering Applications of Neural Networks*, D. Palmer-Brown, C. Draganova, E. Pimenidis, and H. Mouratidis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 485–494.

[8] S. Sinha and A. Paul, "Neuro-fuzzy based intrusion detection system for wireless sensor network," *Wireless personal communications*, vol. 114, no. 1, pp. 835–851, September 2020.

[9] S. A. Siddiqui, A. Mahmood, W. E. Zhang, and Q. Z. Sheng, "Machine learning based trust model for misbehaviour detection in internet-of-vehicles," in *Neural Information Processing*, T. Gedeon, K. W. Wong, and M. Lee, Eds. Cham: Springer International Publishing, 2019, pp. 512–520.

[10] T. Repantis and V. Kalogeraki, "Decentralized trust management for ad-hoc peer-to-peer networks," in *MPAC '06*, 2006.

[11] Y. Trofimova and P. Tvrdík, "Enhancing reactive ad hoc routing protocols with trust," *Future Internet*, vol. 14, no. 1, p. 28, 2022.

[12] Y. Trofimova, J. Fesl, and A. M. Moucha, "Performance analysis of neural network approach for evaluation of trust in ad-hoc networks," in *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)*, 2021.

[13] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," RFC 4728, feb 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4728.txt