Revisiting the OLSRv2 Protocol Optimization in SDN-enabled Tactical MANETs

Ioannis Fourfouris*, Merkouris Karaliopoulos*, Dimitrios Kafetzis*,
Georgios Vardoulias†, Apostolos Georgiadis†, Spyridon Vassilaras†, Iordanis Koutsopoulos*
*Department of Informatics, Athens University of Economics and Business, Athens, Greece
†Intracom Defense, Koropi, Greece

***Abstract*—Despite their limited use in civilian applications, Mobile Ad Hoc Networks (MANETs) set a promising paradigm for decentralized and reliable communication in modern military operations. Of paramount importance in such networks is the capability of their routing protocols to react fast to network topology changes (*e.g.*, link breaks) without introducing excess routing message overhead. Dynamically selecting the protocol parameters to account for the instantaneous network topology can bear important benefits to this end. The addition of the Software Defined Networking (SDN) functionality to MANETs provides the otherwise missing global network view that such dynamic adaptations require and readily motivates the systematic optimization of the routing protocol parameters.**

In this paper, we follow a data-driven approach to decide on the best configuration of key parameters of the Optimized Link State Routing protocol version 2 (OLSRv2), a widely adopted routing protocol for (tactical) MANETs. We employ simple machine learning methods on a rich measurement dataset created with the CORE/EMANE emulator so as to train models that predict the protocol responsiveness and incurred overhead in different topologies and with different protocol configurations. Subsequently, we use the derived mapping between OLSRv2 responsiveness, overhead and MANET features to decide on the best OLSRv2 protocol parameter configuration. Our experimental results demonstrate that this approach can achieve up to 17.27% better responsiveness compared to using OLSRv2 current default parameter values, at the same overhead. Equivalently, with the optimized configuration, OLSRv2 can work at 15.55% lower overhead than with the current default values, without any loss in responsiveness.

***Index Terms*—Optimized Link State Routing version 2 (OLSRv2), Mobile Ad Hoc Networks (MANETs), Machine Learning, Emulation, Data-driven optimization.**

I. Introduction

Mobile Ad Hoc Networks (MANETs) have emerged, after years of research, as a promising solution for decentralized communication in tactical operations. However, their performance is highly dependent on the underlying routing protocols, which must be capable of (a) efficiently adapting to the dynamic topology of the network and promptly recovering from link breaks, in line with operational requirements, while (b) avoiding excessive routing overheads, in terms of protocol messages that need to be exchanged by network nodes to compute, set up and maintain routing paths in the network.

The Optimized Link State Routing protocol version 2 (OLSRv2) has been widely adopted in MANETs as a way to compromise these requirements. The protocol operates in decentralized manner and leverages the Multi-Point Relay (MPR) node concept to rationalize the overhead of message flooding in the network [1] [2]. However, there is still considerable room for improvement, particularly in the direction of systematically optimizing the parameters of the protocol that determine its responsiveness to the topological changes and/or the resulting routing message overhead. Dynamically adapting those parameters in response to topology and traffic patterns in the MANET can have a significant impact on both aspects of the MANET performance. Even a modest reduction in message overhead could be of much importance, especially in bandwidth-limited tactical environments.

Research work over the last decades has looked into the question of optimal routing protocol parameterization. Researchers have sought optimal values for OLSR protocol parameters either by devising simple analytical models [3] [4] or by combining simulations with metaheuristic optimization algorithms [5]. These studies have addressed different types of MANETs, such as vehicular ad hoc networks in [5], tactical MANETs in [6], and they have focused on various aspects of the MANET performance such as overhead, end-to-end delay of data traffic and responsiveness to topology changes. A common drawback of these studies is that they focus almost exclusively on the OLSR parameterization, either fully disregarding the impact of node mobility or considering it in a static simplified manner [3]. As a result, the proposed optimizations can hardly apply to different time instants of a given MANET, let alone across MANETs with different mobility patterns.

In MANETs, the topological information is fragmented across the network nodes and the distributed operation of the OLSRv2 protocol makes the coordination of the MANET nodes towards common settings a hard task to achieve. However, *in MANETs with Software Defined Networking (SDN) capabilities*, the situation is different. Earlier work in the literature has analysed the benefits of SDN for tactical MANETs, including the potential for global optimization of routing paths and the provision for policy-based routing, in particular in coalitional networks, where the "need-to-know" principle needs to be operationalized (see [7] and references within). However, it is argued that when the SDN Control Plane messages propagate through the same wireless links as the Data Plane packets, centralized SDN routing underperforms distributed MANET routing, especially in large and highly dynamic MANETs. We argue in this paper that *a*

side benefit of introducing SDN functionality in MANETs is precisely the capability of collecting topological and traffic information centrally and subsequently leveraging it towards dynamically configuring the OLSRv2 protocol parameters. This yields more degrees of freedom in control for better tactical network operation. For example, when the user data traffic load is higher, a configuration could be chosen that, for the current topology, keeps the message overhead within predefined acceptable bounds, while not penalizing heavily the protocol responsiveness.

Methodologically, our work draws on experimentation with an emulator and Machine Learning (ML). First, we use the CORE/EMANE network emulator to generate records of the OLSRv2 protocol responsiveness and routing message overhead, as we vary protocol parameters and mobility/topology properties. The experimentation with the emulator and the data generation process are phased, starting from a simple controlled experiment with a few nodes and scaling up to the Anglova traces [8], a celebrated benchmark for tactical network operations' scenario. These datasets are then used for training prediction models of the protocol responsiveness and overhead. We test various regressors, both linear and non-linear ones, and assess their training accuracy. In a final step, we generate another set of *testing* data, different from the one used for training the models, and use them to assess the prediction accuracy of these models. More importantly, we use these data to demonstrate how the prediction models can be used to choose OLSRv2 parameters that can better trade the protocol responsiveness with its overhead requirements when compared to its default parameters.

The remainder of the paper is organized as follows. Section II-A provides some background on the OLSRv2 protocol, insisting on key parameters and how they relate to its main tasks. Section II-B summarizes related work on OLSR parameter optimization. In section III, we describe our experimental methodology and the way we construct the datasets for training our ML-based models. In sections IV and V we derive our models and present additional experiments that evaluate the models on different (test) datasets. We conclude the paper in section VI.

II. Background and related work

A. The Optimized Link State Routing protocol version 2

OLSRv2 [2], is a proactive link-state (LS) routing protocol, the successor to the widely deployed OLSR version 1 [1] for wireless Ad Hoc Networks. OLSRv2 retains basic mechanisms and algorithms defined for its predecessor, while introducing several enhancements to it. One example is the adoption of a more modular structure, where, for instance, the responsibility for the exchange of HELLO messages is delegated to the independent Neighbour Discovery Protocol (NHDP) [9]. A second addition is the inclusion of link quality measurements for assessing the reliability and stability of wireless links. These measurements typically concern metrics such as packet loss and latency and help the protocol to dynamically adapt to changing network conditions and select the most reliable paths for data transmission.

Use of MPR nodes: OLSRv2 leverages the concept of 'Multipoint Relay (MPR)' node to reduce its routing message overhead. In typical LS protocols, nodes exchange topological information by flooding the network. As a result, each node receives the same message multiple times from different nodes so that many of the involved link transmissions are redundant. With MPR nodes in place, each network node n selects a subset of its 1-hop neighbour nodes that collectively provide node n with two-hop paths to all its symmetrically connected 2-hop neighbour routers. This set of 1-hop neighbour nodes forms the MPR set of node n. Hence, each MANET node determines its own MPR set and a single node may serve as MPR node for none, one, or more other network nodes, which are called its MPR selectors. There are two types of MPRs in OLSRv2, namely flooding MPRs and routing MPRs, which, in principle, may be different for each MANET node (see [10] for a detailed explanation).

Dissemination of topology information: OLSRv2 uses two types of routing packets to disseminate topological information: HELLO messages and Topology Control (TC) messages. Each network node n, broadcasts HELLO messages that list the identifiers of all nodes from which n has recently received a HELLO message as well as the "status" of the links. These messages are only processed by the 1-hop neighbors of n; they are not forwarded further. Each router sends HELLOs either periodically, every hl_int seconds, or upon certain events such as when it first becomes active or when a link is discovered or lost [9]. Through HELLO messages, the nodes check the availability of (bidirectional) connectivity to their 1-hop neighbours, discover their 2-hop neighbours, and signal their selection of flooding and routing MPRs.

TC messages on the other hand, allow each node to continuously track global changes in the network topology. They are sent periodically, every tc_int seconds, or in response to a change in the router node or its advertised 1-hop neighbourhood. They are broadcast to all other network nodes through flooding MPRs. TC messages carry link-state information in a reduced form. Each node holding the role of the routing MPR for at least one other node, advertises at least the links between itself and the nodes that selected it as MPR, in order to allow all other nodes to calculate shortest paths to those nodes through itself.

B. Related work

Interestingly, most studies focusing on the proper tuning of the protocol parameters. address the original version of OLSR. Table I summarizes the works that attempt to determine appropriate values for some OLSR protocol parameters.

Toutouh et al [5] combine offline metaheuristic optimization algorithms such as Particle Swarm Optimization (PSO), Differential Evolution (DE), Genetic Algorithms (GA) and Simulated Annealing (SA) with ns-2 simulations in searching for optimal OLSR parameterizations for Vehicular Ad hoc Networks (VANETs). They study how the routing overhead

correlates with the performance that data traffic achieves over the network, experimenting with eight protocol parameters, namely hl_int, tc_int, $refresh\_int$, $willingness$, $neighb\_hold\_time$, top_hold_time, mid_hold_time, dup_hold_time. The definition of these parameters can be found in [1]. The four algorithms above drive the search for new candidate vectors of the eight protocol parameter values, which are then assessed over predefined scenarios. Three performance metrics are used, namely, the end-to-end delay, the packet delivery ratio and the relative network overhead, defined as the ratio of routing packet transmissions over the number of delivered data packets. A weighted sum of those metrics then yields the fitness score of the candidate solution and triggers another iteration of the respective metaheuristic algorithm. In a second step, the optimized protocol parameterization is validated in a broader set of scenarios. The resulting values for the hl_int, tc_int and top_hold_time (see Table II) are higher than the ones recommended in [2].

In [3], Hosek *et al.* present an analytical model subject to certain assumptions about the speed of nodes' movement (10m/sec), the link BER (10^{-3}) and the node transmission range (200m). The model which is used to propose a different set of recommended values for four main OLSR parameters that are also relevant for OLSRv2, *i.e.*, hl_int, h_hold_time, tc_int, top_hold_time, with values shown in Table II. The OPNET Modeler simulation environment is used to compare the performance of OLSR under both default and optimized parameters, in a scenario with 72 mobile nodes and 5 traffic flows over a square area of $25km^2$. The paper reports up to 40% reduction of the total routing overhead, without penalties for the end-to-end delay experienced by data traffic.

In [4], Gómez *et al.* study the impact of six OLSR parameters (the four in Table II plus the $refresh\_interval$ and top_hold_time), on Route Change Latency (RCL), a newly introduced measure of the protocol's responsiveness to route changes. RCL is measured at the receiver side, as the total time that elapses from the moment when the last packet before the link failure is received till the moment the first packet after the route change is received. They analytically approximate RCL in a simple 4-node diamond topology as the sum of four delays: (*i*) the initial delay to detect the lost neighbour, (*ii*) the time spent at the node that detects the loss to recalculate a new MPR and signal it to its neighbours, (*iii*) the time it takes for the nodes to be informed about the changes and (*iv*) the time until the new route is actually activated and used. The authors experiment with three OLSR configurations and report a pure tradeoff between RCL and routing message overhead.

Maret *et al.* in [6] study the Directional AirTime (DAT) metric, the link quality metric that is used in OLSRv2 to assess the quality of links and calculate routes. They compare three options to compute the actual value, based on hopcount, expected number of transmissions (TxCount) and expected transmission time (RxAirTime). The three implementations of the OLSRv2 are compared against a benchmark centralized protocol, called Omniscient Dijkstra Routing (ODR) that holds perfect information about the network. They use the EMANE emulator to experiment with a reliable messaging service in tactical scenarios but also simple urban ones with static nodes. They find it challenging to single out one of the three options in terms of Round Trip Time and Message Completion Ratio performance.

III. Data collection experiments

A. CORE/EMANE emulator and OLSRv2 implementation

The emulation platform that we used for our experiments, integrates two open-source emulators, the Common Open Research Emulator (CORE) and the Extendable Mobile Ad-hoc Network Emulator (EMANE), [11], [12]. CORE relies on Linux Namespaces and can emulate realistic network topologies, protocols and devices such as routers, switches, and hosts. EMANE on the other hand, offers a pluggable Layer 1 (PHY) and Layer 2 (MAC, including TDMA) architecture that allows experimentation with PHY and MAC protocols. We used the CORE-EMANE environment to generate training data for our prediction models. The radio propagation model implemented in the emulator is the two-ray model. This does not account for more dynamic and distance-independent fading effects, so that the link quality variations are primarily due to variations of the physical distance between the link end points, as opposed to fades due to multipath propagation.

We utilize the OLSRv2 implementation provided by the OLSR.org Network Framework. This framework offers a modular approach for building network applications through its collection of plugins, also known as subsystems. We use the FF DAT Metric Plugin[1], which implements the Directional Airtime metric in OLSRv2. The plugin incorporates loss rate and link bit-rate to calculate a cost value for links between routers, which denotes the seconds per bit consumed by incoming frames. Details can be found in [13].

B. OLSRv2 protocol parameters

The goal of the experimentation is to identify optimal parameter sets for the OLSRv2 protocol in tactical scenarios, which best trade off the protocol responsiveness to topological changes with the subsequent overhead in terms of messages that need to be exchanged in the network. The main protocol parameters in this context are:

- the time interval between successive HELLO messages (hl_int)
- the time interval between successive Topology Control messages (tc_int)
- the length of the sliding window (L_w) which marks the time interval over which the link quality measurements are averaged when calculating the FF DAT metric.

Intuitively, as the frequency of the topology dissemination messages increases, the protocol responsiveness should improve at the expense of additional routing message overhead. Likewise, while not having impact on the protocol overhead, higher values of L_w delay the protocol response to topological changes for the sake of higher route stability.

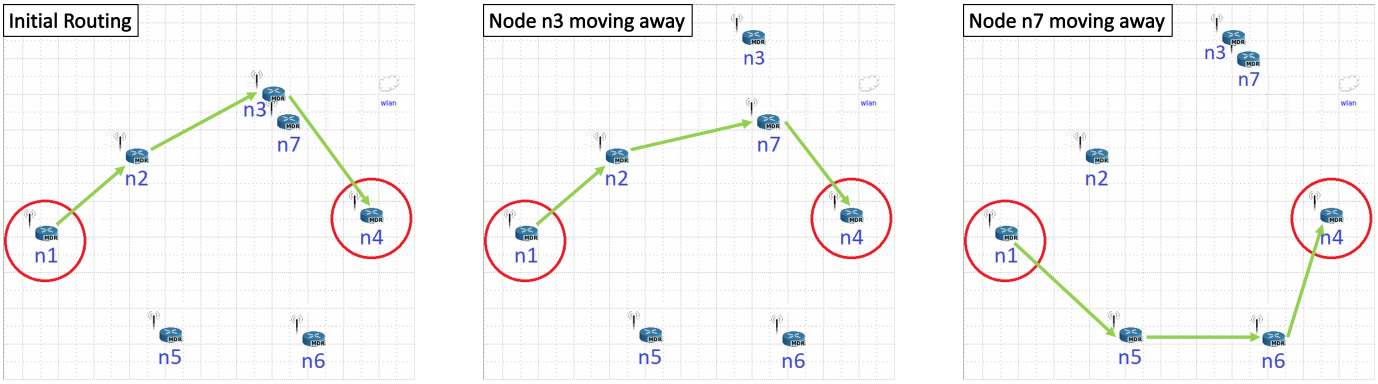
[1]http://www.olsr.org/mediawiki/index.php/FF_DAT_Metric_Plugin.

TABLE I: Characterization of literature on the OLSRv2 protocol.

Study	ver.	OLSR parameters	Performance metric	Methodology
Toutouh *et al.*	v1	hl_int, tc_int, refresh_int, willingness, neighb_hold, top_hold, mid_hold, dup_hold	end-to-end delay, packet delivery ratio (normalized) routing message overhead	Metaheuristic algorithms and ns-2 simulations
Hosek *et al.*	v1	hl_int, tc_int, neighb_hold, top_hold	Routing message overhead, end-to-end delay	Simple geometry analysis and OPNET simulations
Gomez *et al.*	v1	hl_int, tc_int, refresh_int, willingness neighb_hold, top_hold	Route change latency, probability of link/path availability	Analysis and testbed experiments

TABLE II: Recommended (in [5], [3]) or tested values (in [4]) for OLSR/OLSRv2 parameters in literature.

	RFC's Default		Toutouh et al. [5]				Hosek et al [3]	Gomez et al. [4]		
	OLSR	OLSRv2	DE	PSO	GA	SA		#1	#2	#3
hl_int [s]	2	2	8.48	8.91	8.57	9.01	3	0.5	0.5	4
tc_int [s]	5	5	7.25	7.19	5.3	6.75	6	1.25	2.5	10
willingness	3	7	0	1	1	0	3	3	3	3
top_hold_time [s]	15	15	99.06	72.7	67.62	80.97	12	3.75	7.5	20



(a) Original routing path. (b) Local recovery of the routing path. (c) Source recovery of the routing path.

Fig. 1: Controlled experiments: local and source recovery of the routing path.

C. Experiments

We carry out two types of experiments; in both types, we let hl_int take values in $[1, 7]$ sec with a step of 1 second, tc_int take values in [3, 11] sec with a step of 2 seconds and L_w range in $\{8, 16, 32\}$. Depending on the experiment type, we vary additional parameters, as we explain below.

1) Controlled link breaks: In this set of experiments, the network topology is semi-static and shown in Fig. 1. The network comprises seven nodes; five of them, nodes n_1, n_2, n_4, n_5 and n_6, are stationary, while nodes n_3 and n_7 move and, as a result, certain links break. We assume a constant UDP traffic flow of 100 kbps between the source node n_1 and the destination node n_4. The experiments proceed through the following distinct phases:

- *Baseline routing phase:* all nodes are static and the $n_1 \rightarrow n_4$ traffic flow is routed through nodes n_2 and n_3, as shown in Fig. 1a.
- *Local recovery of the routing path:* at time t_1, n_3 moves away and eventually link (n_2, n_3) breaks. The path is locally restored through a new link (n_2, n_7), as shown in Fig. 1b.
- *Source recovery of the routing path:* finally, at time $t_2 > t_1$, node n_7 moves away and the link (n_2, n_7) breaks. A new path between n_1 and n_4 is established, this time through nodes n_5 and n_6, see Fig. 1c. The term "Source" indicates that the initial path is now completely modified, only the source and final destination are preserved.

The experiments are repeated for different speeds and directions of movement of the two mobile nodes, n_3 and n_7. In particular, the two nodes move at three different speeds (20, 40, and 60 km/h) and in four different directions (up, $down$, $right$, $left$). Multiplying the 105 different protocol parameterization (hl_int, tc_int, L_w) combinations, with the 12 different (*direction, speed*) combinations, we get an overall of 1260 experiments for each of the two routing path recovery settings (local, source).

2) Trace-driven link breaks: The second type of experiments is based on the Anglova scenario [8], a well-established benchmark scenario that is widely used in military operations. Specifically, we explore the troop deployment vignette, where the 24 nodes (Company 1) are deployed over an approximate 2km by 2.5km area. This deployment generates dynamic connectivity patterns among nodes as they move with speeds ranging from 0 to 55 km/h. The 105 different combinations for the OLSRv2 parameters are also relevant for this type of experiments. The node movement pattern is fully determined by the Anglova mobility traces and we can vary the density of links between nodes by manipulating their transmission power

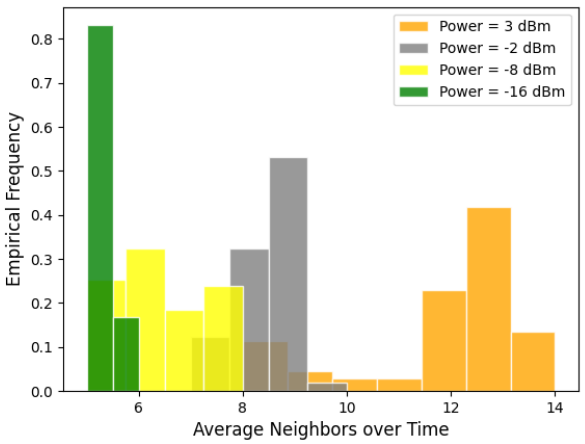


Fig. 2: Distribution of node degrees (number of OLSRv2 neighbour nodes) in the ANGLOVA trace for different values of the node transmission power.

(tx_power). Fig. 2 plots the distribution of neighbors across the network nodes for 4 different tx_power values.

Contrary to the controlled link break experiments, for the Anglova traces we need to *search for* instances of link breaks. Hence, we first run an experiment for a given set of $(hl\_int, tc\_int, L_w, tx\_power)$ parameters, we log down statistics of the node degree distribution (average, $avg(d)$ and standard deviation, $std(d)$), and then we parse offline the generated logs of the nodes' routing tables to identify link break events, their time epochs and how the network responded to them. We then re-run the experiment and introduce short-lived traffic flows traversing paths that include the broken links. These flows send small packets every 5 milliseconds, starting shortly before the link breaks and stopping a few seconds after a flow path is recovered. These packets serve as probes that allow us to more accurately measure the response time of the OLSRv2 protocol.

D. Performance metrics

In all experiments, we compute two quantities: the *Route Change Latency (RCL)* and the *Routing Message Overhead (RMO)*, which serve as measures of the OLSRv2 protocol responsiveness and overhead, respectively. RCL denotes the amount of time the OLSRv2 protocol needs to detect a link break, identify a new path to the destination and activate it. We calculate it for the probe flows as the negative time difference between the arrival time of the last received packet before a link break and the arrival time of the first packet after the flow path is recovered. RMO corresponds to the traffic load due to the control messages exchanged by the OLSRv2 protocol entities to maintain the network topology and derive forwarding paths. To measure it, we collect all the distinct control packets transmitted within a time period of x seconds (default $x = 30$) and determine the sum of their sizes. The RMO is then calculated in terms of bits per second.

IV. Machine Learning Prediction models for OLSRv2 performance

A. Controlled link breaks

The controlled link break experiments yield some first insights into the way the routing protocol and node mobility parameters affect the two performance indices. Fig. 3 plots indicative RCL *vs.* RMO tradeoff curves as we vary the three OLSRv2 protocol parameters hl_int, tc_int and L_w. Intuitively, increasing the frequency of HELLO and TC messages makes the protocol more responsive but also wastes more bandwidth. The depth of link quality measurements' averaging, on the other hand, affects the protocol responsiveness but has negligible impact on the routing message overhead.

The effect of the routing protocol and node mobility parameters is more evident in the statistical models of RCL and RMO. We trained least square-based Linear Regression models, with the OLSRv2 and mobility parameters as predictor variables and RCL and RMO as response variables. For the local recovery scenario, the regression equation for RCL turns out to be:

$$\begin{aligned} RCL_l &= -3.396 + 3.664 \cdot hl\_int + 0.658 \cdot tc\_int \\ &\quad +0.098 \cdot L_w - 0.014 \cdot speed \end{aligned} \tag{1}$$

and the routing message overhead equation is

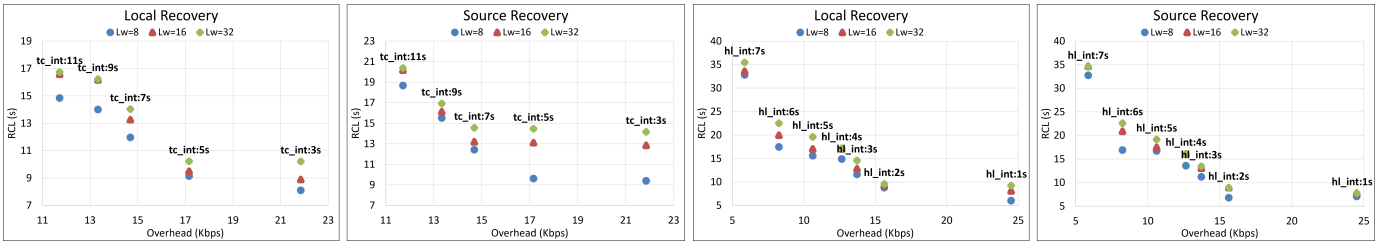
$$RMO = 15.714 - 1.169 \cdot hl\_int - 0.563 \cdot tc\_int \tag{2}$$

whereas for the source recovery scenario, the regression equation for RCL is

$$\begin{aligned} RCL_s &= -4.165 + 3.724 \cdot hl\_int + 0.794 \cdot tc\_int \\ &\quad +0.091 \cdot L_w - 1.279 \cdot dir\_right \end{aligned} \tag{3}$$

and the one for the routing message overhead is identical to equation (2) except for a 0.07 difference in the intercept term. The regression model parameters and fitness for the two equations are given in Tables III and IV.

We can see from equations (1)-(3), that the impact of the mobility parameters (node speed and direction of movement of mobile nodes n_3 and n_7) on the routing message overhead is negligible. A small effect on the protocol responsiveness is recorded for the movement speed in the case of local recovery, *i.e.*, the response time decreases by a tenth of second when the speed grows by 10km/h. This effect disappears in the case of source recovery. On the other hand, in this latter case we record an effect due to the direction of movement. Namely, when the mobile nodes move to the right, accelerating the rate of distancing from the other link endpoint (node n_2), the response time is approximately 1.28 smaller on average. The length of the link quality measurements' averaging window increases the response time, by approximately 1 second when it increases from 8 to 16 and another 1.5 seconds when it further increases to 32.



(a) Speed: 20 km/h, Direction: Down, Hello Interval: 3 s (b) Speed: 60 km/h, Direction: Right, Topology Control Interval: 7 s

Fig. 3: Controlled link breaks: RCL vs. routing overhead.

TABLE III: Regression coefficient estimates, their standard errors (SE) and statistical significance (p-values): controlled link breaks, local path recovery.

	RMO		RCL	
Term	**Estimate (SE)**	**t Ratio**	**Estimate (SE)**	**t Ratio**
Intercept	15.714 (0.113)	138.5*	-3.396 (0.335)	-10.12*
hl_int	-1.169 (0.017)	-68.74*	3.663 (0.037)	99.11*
tc_int	-0.563 (0.012)	-46.85*	0.658 (0.026)	25.18*
L_w			0.098 (0.007)	13.23*
$speed$			-0.014 (0.005)	-3.17**
	R^2 = 0.85, RMSE = 1.21		R^2 = 0.9, RMSE = 2.62	

* p-value < 0.001, ** p-value < 0.002

TABLE IV: Regression coefficient estimates, their standard errors (SE) and statistical significance (p-values): controlled link breaks, source path recovery.

	RMO		RCL	
Term	**Estimate (SE)**	**t Ratio**	**Estimate (SE)**	**t Ratio**
Intercept	15.64 (0.154)	101.9*	-4.165 (0.325)	-12.82*
hl_int	-1.169 (0.017)	-68.64*	3.724 (0.042)	87.68*
tc_int	-0.563 (0.012)	-46.78*	0.794 (0.03)	26.44*
L_w			0.091 (0.009)	10.66*
$direction$ (RIGHT)			-1.279(0.147)	-8.69*
	R^2 = 0.85, RMSE = 1.21		R^2 = 0.87, RMSE = 3.01	

* p-value < 0.001

B. Trace-driven link breaks

Tables V and VI report the outcome of the training processes for different regression models. When looking into the two tables, we can note the following:

- The topological features ($avg(d)$, $std(d)$) do not contribute much to the model capability to explain the variance in RCL. Starting from the full feature set, we can sequentially eliminate these two features (column 3 in Table V) with minimal penalization in terms of root mean square error (RMSE) and coefficient of determination (R^2). On the contrary, they are indispensable for predicting the routing message overhead with reasonable accuracy (compare columns 2 and 3 in Table VI).
- Applying regularization (columns 4-8 in Table V and columns 5-6 in Table VI) does not seem to improve the model fitting accuracy. In contrast, accounting for interaction effects (column 4 in Table VI) does yield an accuracy gain.
- Even higher accuracy is achieved with non-linear models such as regression trees. The accuracy gain (reduced RMSE, higher R^2) comes at the expense of model complexity. Both the model computations needed and the state that has to be stored increase fast with the number of splits allowed when constructing the trees (columns 7-10 in Table VI).

V. Use of the Prediction Models during Protocol Operation and Performance Evaluation

The prediction model(s) that emerged out of the training process in section IV-B are stored at a central node in the MANET. In SDN-enabled MANETs, a natural candidate location is the SDN Controller. The Controller stores up-to-date information about global network topology and ongoing traffic flows, which together with the model state, can drive the dynamic adaptation of the OLSRv2 protocol parameters.

Algorithm 1 Dynamic adjustment of OLSRv2 protocol parameters

Input: Topology information, routing message overhead constraint RMO_{max}, prediction models $\mathcal{P}_{RCL}$, $\mathcal{P}_{RMO}$

Output: Optimal OLSRv2 protocol parameter tuple $(hl\_int^*, tc\_int^*, L_w^*)$

Step 1: Process current topology information to compute the average and variance of node degree $(\widehat{avg(d)}, \widehat{std(d)})$

Step 2: Generate the set S of all possible combinations of parameter values $(hl\_int, tc\_int, L_w, \widehat{avg(d)}, \widehat{std(d)})$

Step 3: Identify the subset $S_c \subseteq S$ of combinations for which the $\mathcal{P}_{RMO}$-predicted $RMO \leq RMO_{max}$

Step 4: Within the set S_c find the combination $(hl\_int^*, tc\_int^*, L_w^*, \widehat{avg(d)}, \widehat{std(d)})$ that minimizes the predicted RCL

Step 5: Return $(hl\_int^*, tc\_int^*, L_w^*)$

Algorithm 1 illustrates the process, for identifying the protocol parameters that minimize RCL subject to a maximum RMO. The algorithm consists of five steps. In the first step, Algorithm 1 processes the current topology information that becomes available by the SDN Controller and computes the topology-related features of the prediction models, $\widehat{avg(d)}$ and $\widehat{std(d)}$. Then, it constructs the set S of all possible combinations of $\big($OLSRv2 parameters, $\widehat{avg(d)}$, $\widehat{std(d)}\big)$ and,

TABLE V: Coefficient estimates for different linear regression variants on RCL : trace-driven link breaks (p-value < 0.001 for all estimates).

Term	Ordinary Least Squares (OLS)	Stepwise remove terms	Lasso	Ridge	Elastic Nets ($\alpha = 0.25$)	Elastic Nets ($\alpha = 0.5$)	Elastic Nets ($\alpha = 0.75$)
Intercept	-5.939	-4.937	-5.934	-5.939	-5.894	-5.921	-5.93
hl_int	3.225	3.225	3.225	3.225	3.219	3.223	3.224
tc_int	0.565	0.565	0.565	0.565	0.563	0.565	0.565
L_w	0.142	0.142	0.142	0.142	0.142	0.142	0.142
$avg(d)$	-0.167	-	-0.167	-0.168	-0.166	-0.167	-0.167
$std(d)$	0.623	-	0.622	0.624	0.619	0.621	0.622
R^2	0.811	0.807	0.811	0.811	0.811	0.811	0.811
RMSE	3.29	3.33	3.293	3.286	3.293	3.292	3.293

TABLE VI: Regression models for predicting RMO : trace-driven link breaks (p-value < 0.001 for all estimates).

Term	OLS Linear (3 protocol terms)	OLS Linear (all 5 terms)	Stepwise Linear add terms	Lasso	Ridge	Regression trees (max split=x) x = 10	x = 25	x = 40	Random Forest Regression
Intercept	29.44	9.848	-3.616	9.849	9.848				
hl_int	-3.389	-3.388	0.41	-3.386	-3.388				
tc_int	-0.334	-0.336	-0.337	-0.336	-0.336				
L_w	0.071	0.071		0.070	0.071	N/A	N/A	N/A	N/A
$avg(d)$		2.143	3.92	2.143	2.143				
$std(d)$		0.206		0.206	0.206				
$hl\_int \cdot avg(d)$			-0.433						
R^2	0.424	0.758	0.805	0.758	0.758	0.714	0.782	0.811	0.992
RMSE	8.02	5.2	4.67	5.21	5.2	9.131	5.31	4.01	1.79

in step 3, it identifies the subset S_c of those that satisfy the constraint on the maximum tolerated RMO, RMO_{max}. This constraint may take into account specific policies and/or the current utilization of the network by user data traffic. It may become stricter under heavier data traffic load and be relaxed under lighter traffic load. The derivation of subset S_c involves repeated invocations of the prediction model $\mathcal{P}_{RMO}$. Then, in the fourth step, the prediction model $\mathcal{P}_{RCL}$ is called once for every combination in S_c. The tuple that results in the minimum RCL value is returned by the algorithm. Whenever this tuple differs from the one returned in the previous execution of the algorithm, the new tuple can be disseminated to the MANET nodes for activation. This can be achieved via additional headers in the SDN control plane messages sent by the SDN controller to the SDN entities (switches) at the MANET nodes.

The following remarks about the algorithm are in place:

- The algorithm can be executed either periodically or upon an alert that the network topology has changed, *e.g.*, when the SDN Controller identifies changes in the nodes' neighborhood sizes.
- Steps 3-5 outline exhaustive enumeration as a naive solution for the underlying constrained optimisation problem ("Find the protocol parameter combination that minimizes RCL subject to the constraint on the RMO"). For simple linear or even convex prediction models, we can leverage smarter optimization techniques.
- The setting of the OLSRv2 parameter L_w (the length of the averaging window) introduces a trade-off between responsiveness and stability. Smaller L_w values accelerate the link quality estimation process but result in higher variance in the obtained estimates. This could be taken into account in step 4, *e.g.*, by favoring feasible solutions with higher L_w values.

In what follows, we highlight the performance gains that are achievable through this dynamic adaptation of the OLSRv2 configuration.

A. Evaluation

We conduct experiments involving the same 24 nodes in the Anglova trace, only this time we set their tx_{power} to three values that are different from those used in the model training phase. Hence, we generate different tactical MANET topologies for the evaluation phase. We run a first set of experiments, measuring $RCL_{default}$ and $RMO_{default}$ values under the OLSRv2 protocol parameter values recommended in RFC 7181 [2]. We then repeat those experiments two times, with OLSRv2 protocol parameter values selected by Algorithm 1, with Random Forest Regression [14] and Lasso [15] as the prediction models for RMO and RCL, respectively.

First, we run Algorithm 1 with the RMO_{max} constraint set to $RMO_{default}$, thus asking how much gain can we get in terms of RCL if the routing message overhead is no more than the one under the default recommended OLSRv2 parameters. We measured an average 17.27% improvement in RCL values, while incurring only a marginal 0.03% increase in RMO.

The second time, we adapt Algorithm 1 to run with a constraint $RCL_{max} = RCL_{default}$ on the route change latency time and seek an OLSRv2 parameter combination that minimizes RMO. Essentially, we ask how much we can save in terms of routing message overhead if we match the RCL value obtained under the recommended OLSRv2 parameters. We measured an average 15.55% reduction in overhead, accompanied by a 1.29% increase in RCL.

In terms of prediction accuracy over this testing dataset, the RCL prediction model achieved $R^2 = 0.716, RMSE = 0.917$ in the first set of testing runs and $R^2 =$

$0.711, RMSE = 0.653$ in the second one. The RMO model resulted in $R^2 = 0.911, RMSE = 2.128$ and $R^2 = 0.947, RMSE = 2.203$, in the two sets of testing experiments, respectively.

VI. Conclusion and Future Work

Our work is an empirical study of data-driven OLSRv2 protocol parameterisation for simultaneously controlling the protocol responsiveness to topology changes and its overhead in terms of routing messages. We harness data from controlled experiments and the well-known Anglova traces to train machine learning models that capture the impact of mobility of tactical nodes and protocol parameters on OLSRv2 performance. We developed an algorithm that leverages these models to dynamically adjust protocol parameters based on real-time information about the network topology and routing message overhead constraints. The algorithm is shown to be better in trading the two protocol performance indicators, namely its responsiveness and the accompanying overhead. The findings of this study advance our understanding of OLSRv2 performance under various scenarios and enable better decision-making in optimizing the routing operation in tactical networks.

A critical assumption underlying our work is the existence and operation of the SDN layer. This is responsible for collecting and maintaining global information about the network topology, which is fed as input to our algorithm. The specifics of the SDN layer, *e.g.*, the number of SDN controller nodes has an impact on the amount of additional messaging needed to update the OLSRv2 agents on tactical nodes with the derived set of parameters. A hierarchical structure with multiple SDN controllers, each associated with a subset of tactical nodes, would reduce the amount of messages that need to be exchanged in the tactical network. In this paper, we have not elaborated on this aspect, which has been treated in the SDN literature, *e.g.*, [16].

One direction of extending this work relates to the feature selection process for the prediction models. Our prediction models can be enhanced with additional variables that impact the routing protocol responsiveness and overhead, such as traffic load patterns. We also intend to explore more sophisticated machine learning models such as Deep Learning ones.

Our research highlights the potential of using data from a real system (Anglova traces) to drive experimental studies with a static digital replica of the system (CORE/EMANE emulator implementation) and, then, use the outcomes of these studies to inform the real system parameters. The natural evolution of the concept is the creation of a tactical MANET *Digital Twin*. With a dynamic digital replica of the real network within an emulator such as CORE/EMANE, the prediction models for the OLSRv2 parameter optimisation could be continuously updated in real-time. This online training process would boost the models' responsiveness and adaptability to constantly changing environmental conditions, thus paving the way for truly efficient, adaptive tactical ad hoc networks. The challenge that has to be overcome relates to the implementation of the interfaces to/from the real world and the conversion of the emulator into a Digital Twin.

Acknowledgment

This work has been carried out in the context of the project entitled Software defined Mobile Tactical Ad hoc NeTwork (SMOTANET), which has received funding from the European Defence Industrial Development Programme (EDIDP) under grant agreement No EDIDP-CSAMN-SDN-2019-038-SMOTANET. This paper reflects only the authors' views and the European Commission is not responsible for any use that may be made of the information contained herein.

References

[1] T. H. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, Oct. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3626

[2] T. H. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, "The Optimized Link State Routing Protocol Version 2," RFC 7181, Apr. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7181

[3] J. Hosek and K. Molnar, "Investigation on OLSR routing protocol efficiency," *Recent advances in computers, communications, applied social science and mathematics*, pp. 147–153, 2011.

[4] C. Gomez, D. Garcia, and J. Paradells, "Improving performance of a real ad-hoc network by tuning OLSR parameters," in *Proc. 10th IEEE Symposium on Computers and Communications (ISCC'05)*, 2005, pp. 16–21.

[5] J. Toutouh, J. García-Nieto, and E. Alba, "Intelligent OLSR routing protocol optimization for VANETs," *IEEE transactions on vehicular technology*, vol. 61, no. 4, pp. 1884–1894, 2012.

[6] Y. Maret and J.-F. Wagen, "Preliminary Performance Benchmarking of OLSRd2 Using Emulated TDMA MANETs and an ODR," in *Proc. 92nd IEEE Vehicular Technology Conference (VTC2020-Fall)*, 2020, pp. 1–5.

[7] D. Kafetzis, S. Vassilaras, G. Vardoulias, and I. Koutsopoulos, "Software-Defined Networking Meets Software-Defined Radio in Mobile ad hoc Networks: State of the Art and Future Directions," *IEEE Access*, vol. 10, pp. 9989–10 014, 2022.

[8] N. Suri *et al.*, "The Anglova tactical military scenario and experimentation environment," in *Proc. International Conference on Military Communications and Information Systems (ICMCIS)*, 2018, pp. 1–8.

[9] T. H. Clausen, C. Dearlove, and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)," RFC 6130, Apr. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6130

[10] C. Dearlove, T. H. Clausen, and P. Jacquet, "Link Metrics for the Mobile Ad Hoc Network (MANET) Routing Protocol OLSRv2 - Rationale," RFC 7185, Apr. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7185

[11] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time network emulator," in *Proc. IEEE Military Communications Conference (MILCOM) 2008*, 2008, pp. 1–7.

[12] "The extendable mobile ad-hoc network emulator (EMANE)," https://www.nrl.navy.mil/itd/ncs/products/emane.

[13] H. Rogge and E. Baccelli, "Directional Airtime Metric Based on Packet Sequence Numbers for Optimized Link State Routing Version 2 (OLSRv2)," RFC 7779, Apr. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7779

[14] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[15] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. [Online]. Available: http://www.jstor.org/stable/2346178

[16] K. Choumas, D. Giatsios, P. Flegkas, and T. Korakis, "The SDN Control Plane Challenge for Minimum Control traffic: Distributed or Centralized?" in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–7.