# TAKE - Tactical Ad-Hoc Network Emulation

Simon Ruffieux*, Christophe Gisler*, Jean-Frédéric Wagen*, François Buntschu* and Gérôme Bovet†
*University of Applied Sciences and Arts of Western Switzerland
Email: {first}.{last}@hefr.ch
†Swiss Department of Defence
Armasuisse Science+Technology
Email: gerome.bovet@armasuisse.ch

***Abstract*—The Swiss Army uses tactical radios to communicate between troops on the field. Due to the operating environment, these radios experience specific characteristics such as limited bandwidth, in addition to high delay and packet losses. This paper presents the work achieved to develop, evaluate and test a novel application-layer routing algorithm specifically designed for tactical MANET networks. In order to evaluate the proposed algorithm in real conditions, two platforms and specific Quality of Experience metrics have been developed.**

## I. Introduction

The Swiss Army is currently evaluating new generation Wideband tactical radios for ground communication. These radios internally rely on the OLSR routing protocol. They are connected to a tactical router and PC. The whole system is referred to as a node in the context of this project. A node is generally located in a vehicle, which is potentially moving, or staying at a specific fixed position. Furthermore, we distinguish three types of communications services: user messages, Blue Force Tracking (BFT) messages and Red Force Tracking (RFT) messages. BFT and RFT messages correspond to the geographical position of friendly and enemy forces, respectively. User messages are generally sent to spread orders, but can also carry any kind of data sent by sensing devices (e.g. sensor-to-shooter loop). The Swiss army needs a robust messaging system that ensures all messages to be successfully delivered even with poor connectivity or in highly dynamic network topologies. Indeed moving vehicles constantly modify the network topology, which pushes the OLSR protocol to its limits and thus reduces the general quality of experience of the messaging system; i.e. messages fail to reach their recipient and positions are not up-to-date. Therefore, the Swiss Army is seeking to improve the current state of practice by developing smart messaging protocols that could mitigate these problems. In order to assess the value of the optimizations and the performance of the new generation radios, solutions to evaluate the quality of experience of the different services are proposed.

Our solution relies on a specific cross-layer messaging application, named *TAKE*, which allows us to take advantage of the provided OLSR routing information and to use this information at an application layer to improve the robustness of transmissions by using our proposed *GetCloser* algorithm. With *GetCloser*, a specific procedure is performed when the system detects timeouts in sent messages. We show that this process improves the robustness of the system through theory and experiments.

In order to test and evaluate the proposed system, we developed two different platforms. The *simTAKE* platforms has been developed to facilitate the development and analysis of the results in laboratory conditions, without the need to deploy the system on the field, which is very time-consuming and might produce almost no reproducible results. The platform provides the possibility to set the number of communication systems (called nodes) deployed in the scenario, their topology and the parameters of the network such as data bandwidth (bps), packets delay (s) and packet drop rate (%) as well as the distribution of messages emitted by each node during a simulation. The *emulTAKE* platform offers a system intended for the next level of tests, namely under field conditions. It allows to control the emulation of users for each node. With this platform, the field test manager can define scenarios, deploy them to each node, in order to generate messages according to specific distributions. Finally, results can be collected once a simulation is completed from a single web page.

This paper is structured as follows. In Section II, we provide an overview of a few investigations in network emulation / simulation platforms and routing algorithms. Four components developed in the context of this project are presented in their respective subsection of Section III. In Section IV, we provide a description of performed experiments and their respective results. The Section V provides a discussion of the results obtained from laboratory simulations and field experiments. Finally, we present our conclusions and possible future works in Section VI.

## II. Related work

The first subsection reviews existing emulation/simulation platforms, the second subsection provides an overview of MANET routing protocols while the third one depicts existing techniques inspired by Delay Tolerant Networking.

### A. Network Simulation/Emulation Platforms

Many open-source and commercial Network Simulator/Emulators exist. Patel et al. provide a survey on best-known emulation testbeds for MANET networks [9]. They notably outline the difference between simulators and emulators and the complexity of deploying and managing emulators for research purposes. A comprehensive list of frameworks can also be found in [1]. However, most of these frameworks do not offer out-of-the-box features that allows for the use of custom proprietary applications in the system or means for quickly reproducing emulated experiments in the field.

Some prior works demonstrated the possibility to develop emulation solutions close to real conditions using interconnected virtual machines. In their work, Zhang et al. proposed an emulator framework called *TapRouter* to run real-time applications on emulated networks [15]. To et al. used a similar approach and proposed a robust and dynamic framework called Dockemu [14]. Recently, Suri et al. have have worked on a complete emulation environment to alleviate the challenging task of having repeatable and controllable experimentations in conditions close to live exercises while having control over the different parameters. They worked on providing a realistic military scenario and emulation environment to evaluate network protocols, algorithms and components [13]. Several other research groups have been using this Dynamically Allocated Virtual Clustering (DAVC) restricted environment to conduct experiments and collect results [5]–[7], [11].

### B. MANET Routing Protocols

MANET (Mobile Ad-Hoc Network) routing protocols are generally located in the network layer and end-users often do not see them acting. They are traditionally divided into reactive and proactive ones. Mojamed et al. made a very informative survey on the subject [2].

With *proactive* (or *table-based*) routing methods, each node has routing information about other nodes in the network and the information is constantly updated through message exchanges, generating various degrees of background traffic. Examples of proactive routing protocols include OLSR, STAR, WRP and QOLSR. Proactive routing has many advantages such as low latency to access the route and a QoS path support and monitoring. Proactive routing is therefore best suited for applications requiring low message latency and high message throughput.

With *reactive* (or *on-demand*) routing, a route to the destination node is established only when necessary, and once it has been discovered, it is maintained by the source node as long as it is required or until it becomes unavailable. In reactive routing, the route discovery process is much more frequent than in proactive routing and the latency for sending a message to a destination is much higher due to the initial route discovery process. Reactive methods can significantly reduce the network control overhead if the required route discovery frequency is quite low. They are best suited for networks with low or medium traffic. Examples of reactive routing protocols include AODV and DSR.

### C. Delay Tolerant Networking

Several approaches have already been explored in the context of delay tolerant networks (DTN). When reliable protocols such as TCP cannot be used because of high latencies, then application-layer routing becomes an ideal candidate to ensure communication at the tactical level. The Bundle protocol [12] is a good example of such an applicative protocol striving to ensure that packets, or also named bundles are delivered to there destination. It offers a mechanism with an elaborated packet protocol specification to forward bundles from in a store-and-forward manner. Rules to decide whether bundles should be split, or by whom they should be forwarded are however not defined and left to the developer's discretion.

Hybrid approaches combining proactive MANET routing and store-and-forward techniques have been developed in order to benefit from end-to-end connectivity provided by MANETs and the reliability of DTN protocols [3], [8], [10]. The OLSR protocol has mostly been adapted such that it either carries itself additional information in order to allow its messages to be stored and forwarded, or the Bundle protocol directly encapsulates OLSR messages. This allows to increase the delivery ratio of packets, especially in sparse topologies. As it is not possible in our scenario to adapt OLSR in that way due to the fact that the tactical radios are considered as black-boxes, we follow another approach where the store-and-forward mechanism that we implemented at the application level will rely on information provided by OLSR.

## III. System overview

In this study, we have developed several elements that work together. A messaging application (*TAKE*) which relies on an applicative routing algorithm (*GetCloser*) and two platforms to manage tests and collect results under laboratory or field tests conditions; respectively named *simTAKE* and *emulTAKE*. We also proposed several novel metrics to characterize and quantify the quality of experience (QoE) of the studied networks.

### A. TAKE Application

The *TAKE* application is a messaging application, which provides the possibility to send and receive messages between nodes of the system. The application works either on the real nodes of the Swiss Army or on the nodes of the emulated virtual machines. The TAKE application relies on the radio network to exchange its messages through UDP sockets relying on the routing

tables provided by OLSR. Previous internal studies have shown that TCP performs poorly on networks with high latencies, which explains why UDP and an applicative acknowledgment mechanism is required. The TAKE application either uses normal operations to exchange messages (called *Direct* algorithm here) or uses the smart applicative routing algorithm developed in the context of this study: *GetCloser*.

The advantage of the TAKE application is that, when using the *GetCloser* algorithm, it adds an applicative layer on top of the networking layer for exchanges and transmissions of messages and can thus optimize the way the messages travel through the network independently of the underlying routing protocol in use.

### B. GetCloser Application-Layer Routing Algorithm

In highly dynamic networks, known routes between nodes may change rapidly and thus impair the success of transmissions due to broken links, in particular for messages exchanged between distant nodes and requiring multiples hops to reach their final destination. In a standard *direct* approach, the messages are directly sent to each of the recipients. When the sending of a message fails, this message is often resent to the same recipient(s). Just as the initial attempt, this has a high chance of failing again, mostly due to multiple hops in the path.

The *GetCloser* algorithm works as follows (see Figure 1). First, a message is sent from its sender to its recipient node(s) directly. Then, if no acknowledgment (ACK) has been received by the node after a given timeout, the message is sent to two intermediary nodes, which are supposed to be closer to the final recipients, considering the currently known network topology determined by the latest OLSR routing table by each node. When a message reaches a new node, the receiving node sends an ACK back to the previous one. When the new node is a recipient, an ACK is sent to the (original) sender as well, in order to improve the probability of notifying it. At last, if the sender has not received all expected ACKs from the recipient(s) after a certain time, the whole process is repeated a second time with the failed recipient only.

The *GetCloser* algorithm has several advantages. It is a simple strategy, easy to simulate, and it uses only the destination and next hop IP addresses of the routing table provided by OLSR. Moreover, it optimizes the sending of a message to multiple recipients by sending it only once on the common route trunks shared by some recipients. Hence, thanks to *GetCloser*, a message is only duplicated when a route splits into several routes towards the different message recipients. Another optimization consists in including the node list which a message has already been sent to, so that nodes in the network know whom not to (re)send it.

Blue Force Tracking (BFT) and Red Force Tracking (RFT) messages are specific messages containing time stamped GPS locations of all known ally nodes (BFT) and enemy military units (RFT), allowing for tactical situation awareness. BFT and RFT messages are handled specifically in order to minimize the consumption of the bandwidth and ensure reliable communications. In practice, these messages are sent periodically by each node to all its neighbour nodes in order to propagate and maintain the most up-to-date information in the entire ally network. However, when some up-to-date information was successfully sent to a neighbour (which will have sent an ACK for that), it will not be resent on the next time. Hence, only the last updated information is exchanged between nodes.

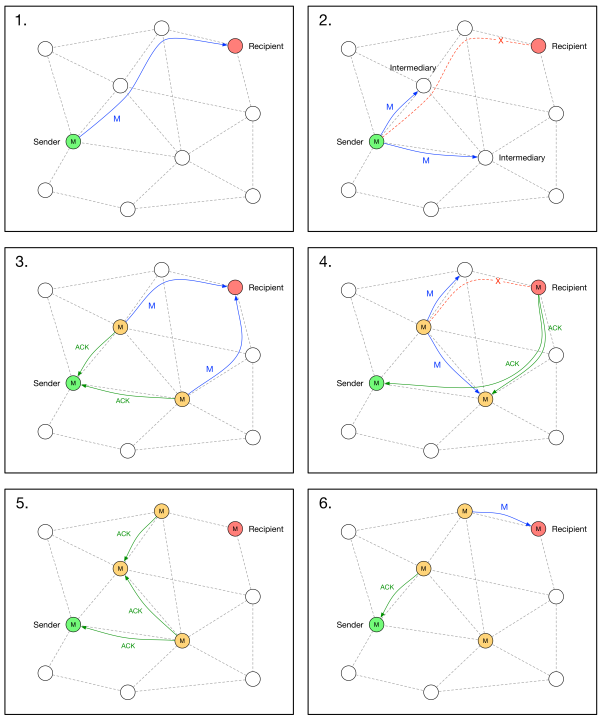


Fig. 1. Sending a message (M, blue arrows) from a sender (green circle) to a recipient node (red circle) using the GetCloser applicative routing algorithm over the existing OLSR network (gray links). Well received messages are acknowledged by nodes (ACK, green arrows).

### C. SimTAKE - Network Laboratory Simulation

The *simTAKE* platform has been developed to test and evaluate the *TAKE* application and applicative routing algorithm(s) in a laboratory setup while remaining in a context as similar as possible to field test conditions. It provides the possibility to shape the traffic of each node (i.e. by limiting the in/out throughput, setting latency and jitter, in addition to dropping a certain percentage of packets) and to define network topologies. The platform relies on Docker and ns-3 to generate virtual nodes and to interconnect them on a single computer, similarly to the works of Zhang [15] and To [14].

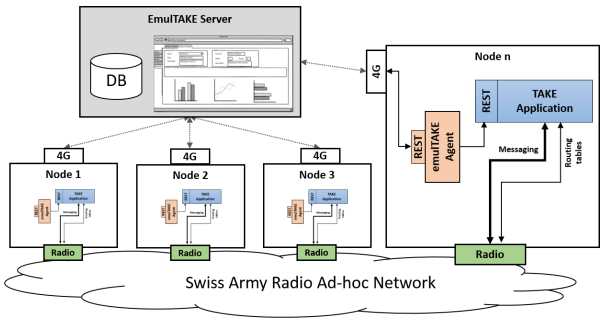


Fig. 2. Overview of the emulTAKE architecture. The emulTAKE Server has control over the complete system through LTE connection. The server consists in a Django Web-application providing a Web interface for user interaction and a database to store the data.

### D. EmulTAKE - Network Field Simulation

The *emulTAKE* platform has been developed to provide the possibility to control and manage the whole system while deployed in military vehicles from a single access point and to visualize the results of executed runs. The underlying idea is to emulate users interacting with the TAKE applications during field tests in order to obtain reproducible results. The platform relies on two components: a Web application providing control over nodes and scenarios, and the *emulTAKE* agent installed on every node of the system. The architecture is schematized in Figure 2.

The *emulTAKE* Agent acts as a middleware between the management Web interface and the *TAKE* application. The Web application and the agent communicate using structured JSON messages. With the Web interface, the field test supervisor can perform several distinct tasks: manage *nodes*, manage messaging *emission profiles*, manage *scenarios*, manage and start *runs*, and visualize *statistics*.

Managing *nodes* notably consists in inserting new nodes in the system. Each node must be configured with its own distinct attributes such as a name, a unique LTE management IP, etc. The Web interface allows to visualize the current state of all inserted nodes. Managing messaging emission *profiles* notably consists in defining a distribution (uniform, random, sequential) and a number of messages per second to emit as well as the size(s) of the payload that will be exchanged during a *run*. Managing *scenarios* consists in defining a name, a description, the involved *nodes*, the *phases* and a messaging *profile* for each "phase x node". Note that a scenario may have multiple *phases*; each phase having a duration and representing a different emission profile. Managing *runs* consists in defining a name and description, a start date and the related scenario. Then, using a specific page, the user can deploy the scenario of the run on all involved nodes and schedule the run, monitor its state and collect its results once completed. Finally, the user can visualize the detailed *statistics* of each involved node for a specific run.

### E. Performance Measures

We developed metrics to quantify the general quality of experience (QoE) of the three different services running on the network: messaging, BFT and RFT. Our approach is notably based on the Link Stability and Delay Metrics proposed by Lal et al. in their study on video streaming enhancement in MANET [4].

The **Messaging QoE** metric quantifies the quality of experience of the messaging service. It is based on the time it takes for a message to be known as received by the sender (aka. round-trip time, $RTT$) and the percentage of messages that have been successfully delivered (aka. Completion Rate, $CR$). We balanced and thresholded these two metrics according to the requirements defined by the field test supervisor. These metrics are computed at the node level and then averaged over all nodes involved in a simulation to obtain the final QoE measure.

$$CR_{metric}(CR) = \begin{cases} 1.0 \text{ if } (CR \geq 0.99) \\ 0.8 \text{ if } (0.99 > CR \geq 0.95) \\ 0.\ 3 \text{ if } (0.95 > CR \geq 0.80) \\ 0.\ 1 \text{ if } (0.80 > CR \geq 0.50) \\ 0 \ otherwise \end{cases}$$

$$RTT_{metric}(RTT) = \begin{cases} 1.0 \text{ if } (3000 < RTT) \\ 0.8 \text{ if } (5000 > RTT \geq 3000) \\ 0.\ 3 \text{ if } (10000 > RTT \geq 3000) \\ 0.\ 1 \text{ if } (15000 > RTT \geq 10000) \\ 0 \ otherwise \end{cases}$$

$$QoE_{msg} = \frac{w_{CR} * CR_{metric} + \ w_{RTT} * RTT_{metric}}{w_{CR} + w_{RTT}}$$

With $CR$ and $RTT$ being respectively the average Completion Rate and Round-Trip Time of a node. $w_{CR} = 0.75$ and $w_{RTT} = 0.25$ are constants, these can be tuned to put more emphasis on one or the other metric depending on the end-user requirements.

The **BFT QoE** metric quantifies the difference in time between the moment when a position of a friendly unit is sent by a node and the moment it is received by another node. Lost messages are implicitly taken into account by setting a time limit ($DeltaT_{max}$) after which a position message is not considered. At a larger scale, it represents the average time it takes for a node to be updated with the new positions of all other nodes in the network. In our experiments, we forced a constant periodical emission of messages in order to obtain BFT QoE measures at the node level. The final QoE measure is obtained at the end of an experiment, by averaging the measures of all involved nodes.

If we assume a constant emission period of $P$ seconds for all nodes and $N$ nodes involved in the simulation, we can assume that every node $(i)$ should receive the

position of all other nodes $(k)$ periodically every $P$ seconds. Using this assumption, we can compute the average delay between two received positions for every other nodes involved in the simulation. In our implementation, we limited the maximum delay to $DeltaT_{max}$ = 10*$P$:

$$DeltaT\,(i,k) = min\left(\overline{delay\,(k)}, DeltaT_{max}\right)$$

The average $DeltaT(i)$ for the node $i$ is then given by

$$DeltaT\,(i) = \frac{1}{N-1} * \sum_{k=0,\ \ i \neq k}^{N} DeltaT(i,k)$$

Then the averaged $DeltaT$ of every node $(i)$ involved in the field test is computed and normalized as follows to obtain a final BFT QoE metric ranging between 0 and 1.

$$QoE_{BFT} = 1 - \frac{\left(\frac{1}{N} * \sum_{i=1}^{N} DeltaT\,(i)\right) - P}{DeltaT_{max} - P}$$

The **RFT QoE** metric is similar to the $QoE_{BFT}$ metric: friendly unit (for BFT) is simply replaced by enemy (for RFT).

## IV. Experiments and Results

This section describes the main experiments performed in the context of this work and highlights the results for some of them. We distinguished experiments performed in the simulated environment with *simTAKE* allowing evaluation of algorithms with the simulation platform and the field test experiments performed by the Swiss armed forces on the field with real hardware mounted in vehicles, which were using *emulTAKE* to emulate real users.

### A. Network Simulations

For the simulations, a total of eight topologies were used in order to assess the performance of the *Direct* and *getCloser* algorithms. Due to space limitations, this paper covers only four of them. Three topologies are shown in Figure 3, the fourth being a 40 nodes fully meshed topology. The simulations were run under several conditions such as throughput, message and BFT/RFT rates. In this paper we limit the results to those matching with the parameters that were defined by Armasuisse with the intention to obtain results coping with real tactical operations. The throughput of the network was limited to a maximum of 50 kbit/s, which reflects what can be obtained with Wideband tactical radios under good channel conditions. Moreover, it was decided that each node would send messages at a rate of 0.2 message/s, which represents orders or various kind of sensor information. In addition to this, the BFT and RFT services were set to share their tactical situation every ten seconds with all the other nodes. The repeatability of the simulation tests has always been verified by running several times the same simulation.

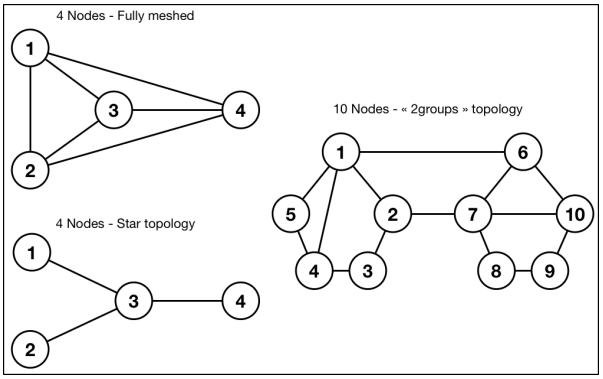


Fig. 3. Excerpt of the topologies used in the simulations.

TABLE I
Results of the tests performed with the simulation platform. A simulation lasted 300 seconds. Message emission rate = 0.2 message/s, BFT/RFT tactical situation sent every ten seconds to all nodes.

| Topology | Algorithm | QoE Msg | QoE BFT | QoE RFT | Average RTT [s] | Average CR [%] |
|---|---|---|---|---|---|---|
| 4 Nodes | Direct | 0.95 | 1 | 1 | 3.2 | 100 |
| Fully meshed | GetCloser | 0.95 | 1 | 1 | 3.2 | 100 |
| 4 Nodes | Direct | 0.78 | 1 | 1 | 4.6 | 99 |
| Star | GetCloser | 0.89 | 1 | 1 | 4.6 | 100 |
| 10 Nodes | Direct | 0.73 | 1 | 1 | 6.5 | 97 |
| 2Groups | GetCloser | 0.88 | 1 | 1 | 6.5 | 100 |
| 40 Nodes | Direct | 0.41 | 0.57 | 0.56 | 3.2 | 87 |
| Fully meshed | GetCloser | 0.62 | 0.57 | 0.54 | 4.3 | 95 |

In Table I we provide results for the four aforementioned topologies. Each scenario was run with the *Direct* and *GetGloser* algorithms. We can from this table see that there is no difference in terms of QoE, RTT and CR for the *4 Nodes - Fully meshed* scenario. As all nodes were directly connected to each other, the network indeed offered enough capacity for transporting the messages to their destination without loss or timeout. In this case the *GetCloser* algorithm did not active its specific retransmission capabilities. Although only four nodes were still used in the *Star* topology, we can clearly see that the RTT increases due to the fact that messages have to be relayed by Node 3. Some messages were considered as lost, certainly due to timeouts with the *Direct* algorithm. The *GetCloser* algorithm was in this scenario useful, as a CR of 100% could be obtained, which explains the better QoE. The same observation applies to the *2Groups* topology, where the *GetCloser* algorithm performed better. The difference is even more significant with the *40 Nodes - Fully meshed* topology. The *Direct* algorithm reaches a better RTT, but many messages were not successfully delivered. In contrary, the retransmissions of the *GetCloser* algorithm induces a higher RTT, but as a consequence allow to obtain a better CR thanks to the retransmissions. In terms of QoE,

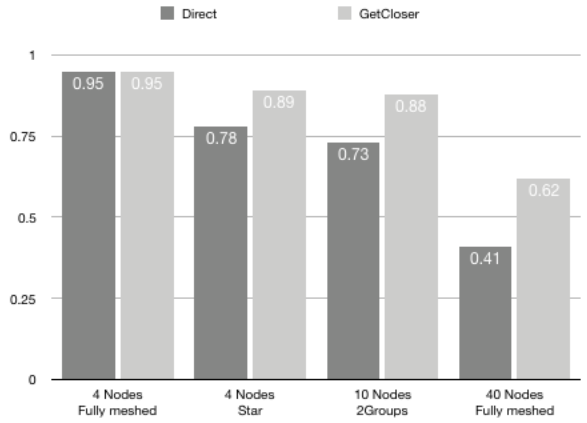


Fig. 4. Comparison of Messaging QoE between the Direct and GetCloser algorithms.

the difference is notable with a value about 0.2 (+50%) higher.

The results of our simulations demonstrate that *GetCloser* improves, as desired, the robustness of the network by improving the Completion Rate (CR) of the exchanged messages when possible. Therefore simulations using *GetCloser* consistently obtain better results for the messaging quality of experience (QoE MSG), thanks to the larger proportion of sent messages successfully reaching their destination(s).

### B. Field Tests

Several field test runs have been performed in real conditions by the Swiss Army using the *emulTAKE* platform and the *TAKE* software. All military vehicles were equipped with an embedded PC running both software. Each embedded PC was connected to a tactical radio, and had access over a LTE connection to backend systems. Armasuisse notably desired to perform an evaluation of two different radio systems. These tests consisted in 20 different scenarios involving 14 vehicles (nodes). The *GetCloser* algorithm was used in all scenarios. The duration of the scenarios ranged from 5 up to 20 minutes. In some of the scenarios, the vehicles were static and in others some or all vehicles were moving. Note that we report in this paper the results of 4 "most interesting" scenarios obtained with the *emulTAKE* platform.

All results are very poor from an end-user perspective but show that the *TAKE* software and the *emulTAKE* performed well despite the HW/SW problems of the radio prototypes under tests.

The four scenarios reported here had different conditions/parameters:

- Topology: Custom (automatically handled by the MANET layer of the radios)
- Message Emission Rate: 0.1 or 0.2 message/s
- Radio Model: A or B (real names are confidential)

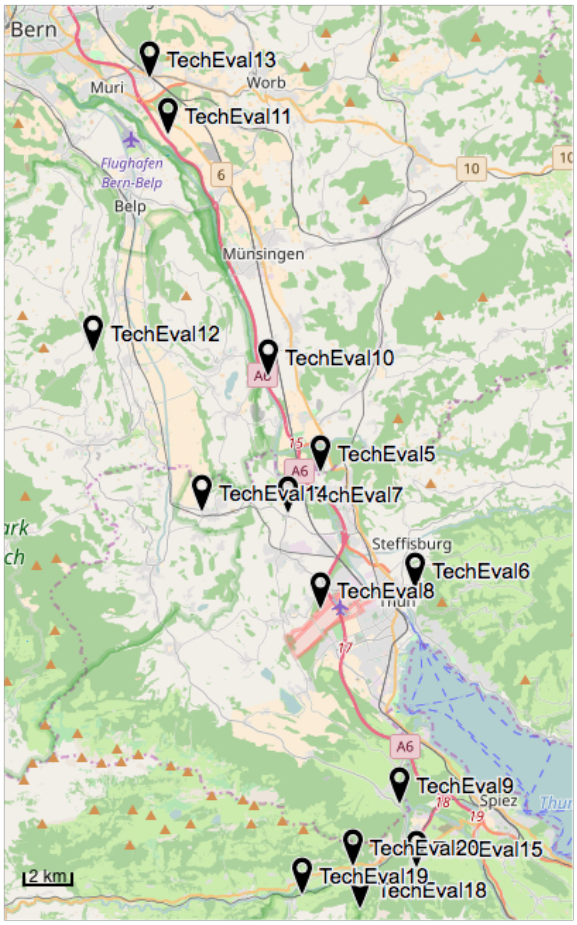


Fig. 5. The 14 vehicles were scattered around the region of Thun in Switzerland in order to obtain a custom topology.

Each scenario run had a duration of 5 minutes and BFT/RFT were set to emit messages periodically every 10 seconds. The 4 presented scenarios were conducted with a static custom topology; the vehicles were stationed in the region of Thun, as illustrated in Figure 5.

TABLE II
RESULTS OF ONE FIELD TEST PERFORMED WITH TACTICAL RADIOS AND VEHICLES SCATTERED IN THE THUN REGION.

| Messaging rate | Radio model | QoE Msg | QoE BFT | QoE RFT | Average RTT [s] | Average CR [%] |
|---|---|---|---|---|---|---|
| 0.1 | A | 0.31 | 0.43 | 0.43 | 20.8 | 87.14 |
| | B | 0.02 | 0.12 | 0.20 | 41.7 | 35.50 |
| 0.2 | A | 0.00 | 0.07 | 0.07 | 88.11 | 14.43 |
| | B | 0.02 | 0.03 | 0.04 | 45.6 | 6.76 |

The field tests conducted by Armasuisse yielded very low QoEs (<0.4) due to the high RTTs (>20s) and low CRs (<50%) obtained. The results indicate that radio A outperforms radio B by achieving better completion rates (87% vs. 35%) when few messages are emitted (messaging rate of 0.1 message/s). When more messages were emitted, radio model A also yielded poor results, although it had a CR twice better than the radio model

B. At the time of writing, these results could not be compared to measurements performed in simulation as we were not aware of the exact topology of the network.

We provide here a more detailed analysis of the results of the scenario with radio model A and a messaging rate of 0.1 message/s. Note that these plots were automatically generated by the Web interface of the *emulTAKE* platform. Figure 6 shows the RTT values for the different nodes. We can observe a high RTT value for the node TechEval18 with an average of approximately 40 seconds. The other nodes have similar mean RTT, which are around 20 seconds.

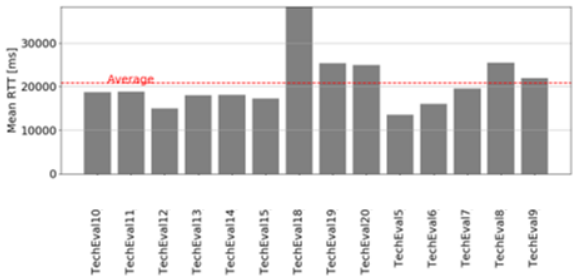


Fig. 6. RTT values for each of the nodes that were involved in the field test run.

Figure 7 shows the number of messages that could not be delivered during the scenario (Failed Messages). Note that during this scenario, a total of 60 messages were sent by each node. We can observe large variations of the results between the different nodes, with a few nodes having almost no failed messages (TE15, TE18, TE6, TE7 and TE8) and node TE19 having more than 25% of its messages that failed. It would be interesting to correlate these results with the actual topology of the network during the test; notably to observe if node TE19 was out of reach or had a particular position in the topology.

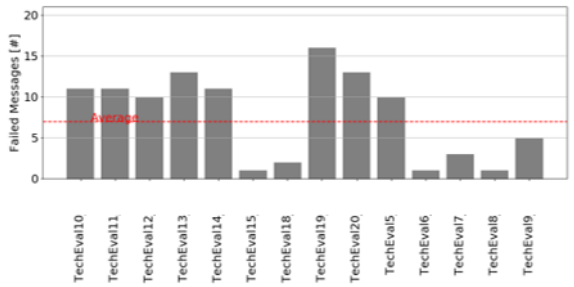


Fig. 7. Number of undelivered messages for each of the nodes that were involved in the field test run.

Figure 8 shows the computed Messaging QoE for each node; the metric combines RTT and CR results.

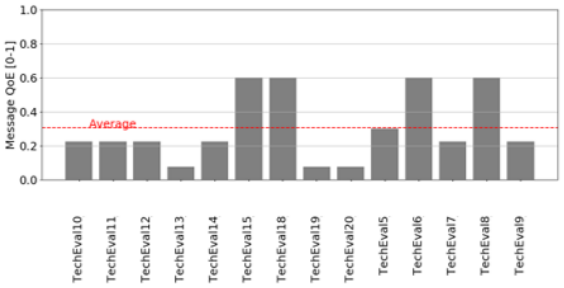


Fig. 8. QoE values for each of the nodes that were involved in the field test run.

## V. Discussion

In this project we developed several components. We notably proposed a specific algorithm named *GetCloser* to improve robustness of messages exchanged in highly dynamic networks, in which messages have a higher probability of being dropped with traditional (IP routing) approaches. We also developed a simulation tool in order to simulate a network with specific latency, drop rate and throughput. This tool allowed us to evaluate the *GetCloser* algorithm without the need of deploying a real network in the field. We also developed the *emulTAKE* platform to be able to manage and run scenarios which emulate users sending messages with real hardware during trials in the field. The *emulTAKE* platform is key to the handling of field experiments and provides the possibility to obtain reproducible results. Finally, we proposed several metrics to characterize the quality of experience (QoE) of two services: messaging and BFT/RFT.

### A. GetCloser

The *GetCloser* algorithm provides more robustness, i.e. a message completion rate closer to 100%, to the messaging exchange system by retransmitting messages that have been dropped to nodes near the destination. In contrary the *Direct* algorithm will fail if the end-to-end connectivity is not guaranteed. This behaviour improves the measured completion rate to the detriment of a lower round-trip time (as re-sent messages successfully received have their RTT increased by the timeout value, e.g. 20 seconds in our experiments). However, because our messaging metric (Messaging QoE) puts more emphasis on CR than RTT, the *GetCloser* algorithm performs better in terms of Messaging QoE in simulations where messages were dropped, as initially hypothesized.

### B. simTAKE

The *simTAKE* simulation platform provides a rapid solution to test and evaluate protocols and networks in simulation using a single computer. The possibility to virtualize nodes, set their topology and configure

their network characteristic from a single application allows to rapidly test various settings. This has been demonstrated with the evaluation of the *GetCloser* algorithm on multiple different topologies. Limitation in scalability are visible with 60 nodes, where forcing the periodical emission of 60 RFT and 60 BFT messages in burst by each node yielded dropped messages. We could not further investigate this point in the context of this project due to time limitations, but it is expected to be a behaviour due to the limitations of the network data throughput set in the experiments.

### C. emulTAKE

The *emulTAKE* platform successfully achieved its goals. Armasuisse developed and tested the application in a real context in the field. They were able to use the interface and tools developed to prepare and conduct their experiments. They notably used the platform to compare two different models of tactical radios. The platform allowed them to obtain quantitative results with vehicles deployed on the field. Although the results obtained during these tests were unsatisfactory in terms of QoEs, our platform provided the quantitative metrics which seem to demonstrate that the tested tactical Wideband radios (from suppliers A and B) are not yet ready to be operatively used in the field. Furthermore, the gained results served as basement for discussions with the suppliers. By analyzing the logs of nodes, some explanations could be found, therefore justifying the poor results. The *emulTAKE* platform will be further used by Armasuisse in future field tests.

## VI. Conclusion

In this work, we have presented the different elements developed for the Tactical Ad-hoc NetworK Emulation (TAKE) project. We developed a complete toolset allowing to test and evaluate routing algorithms, protocols and setups in laboratory and in field conditions using common metrics. Our system notably allowed to demonstrate the advantages of the proposed *GetCloser* applicative routing algorithm in laboratory conditions and also allowed Armasuisse to compare the efficiency of two different types of radios in field conditions. The results of these experiments have been presented with novel quality of experience (QoE) metrics for the different messaging channels under investigations: user messaging and BFT/RFT position tracking.

The proposed platforms provide the basic elements for a complete evaluation system but some of its functionalities could still be extended or new functionalities added in future steps of the project. The *simTAKE* platform does not yet provide the possibility to define mobility patterns for the nodes or to define advanced radio channel models (depending on distances, terrain topology, interferences, etc.). These functionalities would be required to evaluate more advanced scenarios in simulation and to obtain results closer to field conditions. The development of more advanced visualization tools could also be useful. Finally, a system to collect partial results of experiments while they are executed could be beneficial in order to facilitate the monitoring of running experiments.

## Acknowledgment

We would like to thank Armasuisse for project's funding and access to their infrastructure, under ARAMIS-No. 041-05. We would also like to acknowledge the work of Mathieu Dévaud and Loïc Bassang on the initial steps of the project, and Jean Hennebert for his advices.

## References

[1] Open-source routing and network simulation. http://www.brianlinkletter.com/open-source-network-simulators/.

[2] Mohammad Al Mojamed and Mario Kolberg. Structured Peer-to-Peer overlay deployment on MANET: A survey. *Computer Networks*, 96:29–47, 2016.

[3] Saaidal Razalli Azzuhri, Harith Ahmad, Marius Portmann, Ismail Ahmedy, and Ranjana Pathak. An efficient hybrid manet-dtn routing scheme for olsr. *Wirel. Pers. Commun.*, 89(4):1335–1354, August 2016.

[4] Chhagan Lal, Vijay Laxmi, Singh Gaur Manoj, and Conti Mauro. Enhancing QoE for video streaming in MANETs via multi-constraint routing. *Wireless Networks*, 2016.

[5] Kelvin Marcus. Application of the dynamically allocated virtual clustering management system to emulated tactical network experimentation. page 907904. International Society for Optics and Photonics, jun 2014.

[6] Kelvin Marcus, Christoph Barz, Jonathan Kirchhoff, Henning Rogge, Jan Nilsson, Anders Hansson, Ulf Sterner, Mariann Hauge, King Lee, Arjen Holtzer, Boyd Buchin, and Markus Peuhkuri. Evaluation of the Scalability of OLSRv2 in an Emulated Realistic Military Scenario. 2017.

[7] Kelvin Marcus and Jess Cannata. Dynamically allocated virtual clustering management system. *Proceedings of SPIE - The International Society for Optical Engineering*, 8742:87420X, 2013.

[8] Ranju Pant, Apinun Tunpan, Preechai Mekbungwan, Rujipol Virochpoka, and Kanchana Kanchanasut. Dtn overlay on olsr network, 01 2010.

[9] Kishan N. Patel and Rutvij h. Jhaveri. A Survey on Emulation Testbeds for Mobile Ad-hoc Networks. *Procedia Computer Science*, 45:581–591, 2015.

[10] C. Raffelsberger and H. Hellwagner. A hybrid manet-dtn routing scheme for emergency response scenarios. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 505–510, March 2013.

[11] Janusz Romanik and Krzysztof Zubel. Evaluation of OLSRv2-based Routing Mechanism for Tactical Networks. *ICMCIS*, 2017.

[12] K. Scott and S. Burleigh. Bundle protocol specification. RFC 5050, IETF, 2007.

[13] Niranjan Suri, Anders Hansson, Jan Nilsson, Piotr Lubkowski, Kelvin Marcus, Mariann Hauge, King Lee, Boyd Buchin, Levent Misirhoglu, and Markus Peuhkuri. A realistic military scenario and emulation environment for experimenting with tactical communications and heterogeneous networks. In *2016 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–8. IEEE, may 2016.

[14] Marco Antonio To, Marcos Cano, and Preng Biba. DOCKEMU – A Network Emulation Tool. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, number April, pages 593–598. IEEE, mar 2015.

[15] Jinxue Zhang and Zheng Qin. TapRouter: an emulating framework to run real applications on simulated mobile ad hoc network. pages 39–46, 2011.