УДК 681.3 https://doi.org/10.15407/pp2026.03.020

Д.В.Рагозін, К.А.Заіка

МОДЕЛЮВАННЯ І АНАЛІЗ АНОМАЛІЙ ПЕРЕДАЧІ ДАНИХ У БЕЗДРОТОВИХ МЕРЕЖАХ

У статті описана розробка та функціонування доступної системи моделювання для збору мережевого трафіку та аналізу його аномалій для бездротових мереж на основі протоколу TCP. Розглянуло мережі, що включають достатню кількість роботів, дронів та інших безпілотних літальних апаратів (БПЛА), які потребують аналізу щодо наявності проблем з транспортуванням даних у разі апаратних або програмних збоїв та зовнішніх впливів. Система моделювання повинна збирати інформацію про трафік у мережі, необхідну для подальшого аналізу за допомогою методів машинного навчання, та надання зворотного зв'язку мережевим адміністраторам. Розглянуто використання та розширення пакета GrADyS-SIM для середовища моделювання OMNeT++, який був спеціально розроблений і адаптований для мереж БПЛА. Модельований мережевий трафік попередньо обробляється для виділення необхідної інформації та додавання позначок часу для подальшого аналізу в класифікаторі на основі «навчання без підкріплення», побудованому за допомогою методів OPTICS. Додатковий інструментарій, інтегрований в систему OMNet++, забезпечує спрощене створення сценаріїв генерації трафіку та зручну візуалізацію. Класифікатор забезпечує кластеризацію подій з потоку трафіка та виявлення аномалій у потоці. Проведено серію експериментів, які доводять, що результати класифікації є цінними для керування роботи мережі та можуть бути використані для автоматичного налаштування обсягу трафіку від БПЛА до операторів та серверів.

Ключові слова: бездротові мережі, моделювання, аномалії у мережах, дрони, навчання без підкріплення.

D.V. Rahozin, K.A. Zaika

TRANSPORT ANOMALIES SIMULATION AND ANALYSIS IN WIRELESS NETWORKS

In the paper we research the affordable simulation system for gathering network traffic and analyzing its anomalies for TCP-based wireless networks. We consider networks that include enough number of robots, drones and other unmanned aerial vehicles (UAVs), and which need to be analyzed for traffic transport problems due to hardware/software failures or external impacts. The simulation system should be able to gather traffic information necessary for further analysis by machine learning techniques and providing feedback for network operators. We considered to use and extend GrADyS-SIM package for OMNeT++ simulation environment, as it was specially targeted for UAVs networks. The simulated networks traffic is preprocessed for extracting valuable information and adding timing marks for further analysis in unsupervised learning-based classifier built using the OPTICS methods. The rich tools, integrated into OMNet++ system provide simplified traffic scenarios creation and good visual presentation. The classifier provides traffic flow events clustering and anomalies detection. We made series of experiments, proving that the classification results are valuable for network operation and further may be used for automatically tuning traffic volume from UAVs to operators and servers.

Key words: wireless networks, simulation, network anomalies, drones, unsupervised learning.

Introduction

The major updates in wireless networking technologies enabled the construction of large networks of autonomous robots. These networks have static or ad-hoc configurations, with goal to gather data over the network area and transport this data to “sink” devices. Sinks redirect data flow to some accumulating data base storage for further analysis [1]. For the large number of unattended devices or robots it becomes critical to have fault-tolerant and robust network, as storage facility usually requires complete data sets and sustainable control over all network nodes [2]. Large ad-hoc networks may impose limitations on accurate and error-free data transport to sinks (we call it anomalies), which depends on network configuration, channels limitations or device failures. Such networks experience misconfiguration,

faults in network hardware or transporting structure, bandwidth limitations, external communication interference; hardware failures. This leads to data loss or corruption on simpler devices, for more complex devices malware or even software error may filter or affect data. Anyway, the patterns in data corruption or faults may be traced and analyzed [3]. Manual diagnostics of device network malfunctioning is practically impossible for a hundred-nodes network, so there is a need for automated methods which can detect and analyze various anomalies – data loss, data corruption, hardware failures, failed routes, access denials – and with minimum human intervention, finally providing summary for possible reaction scenarios. This kind of analysis requires the whole network statistics to define fault-related data. For our investigation this data can be gathered as a result of network behavior simulation.

Further, the task of fault analysis depends on anomaly detection methods, so the network simulation should not be up in the air, providing just all kinds of statistics. That's why we need to define which data should be provided for further analysis, and we should not consider just simulation itself, but instead - a linked pair of simulation and data analysis. In this paper we are researching methods, which can 1) check unusual activities happened inside the network; 2) identify the data we need for network analysis from simulation side; 3) propose a network failure simulator, adopted for our analysis tasks.

In chapter 1 we discuss requirement for faulty network simulation and the types of faults. Chapter 2 includes the review of existing network simulation software and the selection of the most applicable one. Also, chapter 2 elaborates the structure of simulated model. Chapter 3 discusses the simulation results and reviews the results/possibilities of fault/anomalies analysis for our model.

2. Network simulation requirements

We define and research the practical requirements based on practical configuration of device or robotics system, used now "in field". We are not original in our studies, there are many previous researches, for example [4], but we focusing on newest practical points. It should be noted, that the common networks may use several levels of interaction, but for many cases we can separate data processing center; backbone infrastructure; gates, connecting backbone infrastructure to real devices; device/robotic mesh or swarm. In some cases, several elements may be absent or gates may work in manual mode, but this network structure reflects the most common cases. Sometimes we have a mobile-network (3G/4G) back-bone as a base of data acquisition system, sometimes we have common network based on Ethernet physical layer, the emerging technology is the use of ancestors and modifications of Wi-Fi technology. Very often Wi-Fi-based transport is used on the last mile to gather high-bandwidth data for analysis. On the other side, low power technology such as LoRa or other proprietary transceivers are used on the "last mile" to provide limited bandwidth data acquisition over a big area for 24/7 analysis. So, the real field or industrial network may include a "zoo" of different transport technologies, and should be monitored, so that degraded nodes may be repaired or updated without massive denial of access caused by massive loss of transport abilities in the network. Here rises the common perspective, that some number of solutions, necessary for control robotic swarms in filed, will be established for research, for example [5], and soon moved to commerce. But we are not focusing just on control of devices and network.

2.1 Network protocols

Practical majority of wired and wireless networks use TCP/IP family of network protocols for data transmission over the "backbone" network [6]. Still the "last mile" may use local protocols, such as LoRa, specific Wi-Fi extensions, MAVLink over any physical layers, proprietary noise-like signal-based transmission or simplest proprietary transceivers. The "last mile" may require covering additional attention, as may experience signal jamming, signal shielding, signal power issues, periodic link losses and so on. Periodic transmission errors may affect higher protocol, causing hardly explained errors. Anyway, the big share of modern low power data acquisition devices uses LoRa communication protocol, or device swarm may be controlled by an extended

MAVLink, so the simulator somehow should be aware of "last mile" protocols. Full-featured TCP/IP requires more powerful computing units and energy and usually requires full-featured operation system (for example, embedded Linux): the most devices, which are equipped with motors or actuators have enough energy sources to support more computational resources, so they support TCP/IP. Less power-enabled devices or just simpler devices provide limited TCP/IP stacks, but application-level software is built to be fit into the transport restrictions. Still the heavy communication load on a device may be the source of malfunctioning.

The modern practice shows, that the big number of solutions include TCP/IP based backbone network; and it contains devices of many types. TCP/IP is used at least at the set of gates between backbone network and data acquisition devices. The network may include the other gate types, e.g. mobile network gates, Wi-Fi family gates; the number of end-points, which may be TCP/IP clients having local LoRa clients, MAVLink-based clients or other off-the-shelf protocols. We use this network structure type as a general case for our research Let's discuss a bit more details.

TCP/IP. Looking more precisely into TCP/IP we consider 4 layers – if compared to 7-level OSI network model TCP/IP looks simpler. Each layer presentation in a simulated network should be considered with respect to simulation goals – in our case for anomaly analysis and detection. The lowest level, #1, – physical layer – is usually well enough simulated by a simple data transmission with respect to real time but introducing programmable probability for packet loss - to simulate hardware errors or jamming in wireless networks. The next – internet layer, #2 – potentially should take into account external attacks to routing internal data, or protocol errors, which lead to packet loss and infrastructure problems in network hierarchy. Here the simulation of packet routing control is possible, as an example, via SNMP, so the standard protocol features are used; still the simulation system should support additional protocols such as SNMP and possibly more modern protocols, including vendor-specific protocols. The third – transport layer, #3 – reflects internal TCP/IP algorithms, that handle streamed protocol features, and here there are much less influence point than for previous internet layer, due to the nature of the layer. The #4 and final – the application layer – is the most complex and the most interesting for simulation, as it may represent all the device communication systems. Usually, the application layer is closely tied to application software, which forms the "local" or "last mile" network via LoRa or other protocol and may be represented as a data generator, which creates a data stream similar to real life data. Simulation may use some stored data tables to feed them to the network, or just generate the properly marked data here. Marked data means that some data may include the signaling values, forming some data sequence, where the data loss may be clearly visible, comparing to more expensive procedures needed for checking data loss for real sensor data. As these data should be stored on server within proper data sequence, Changes in values or in sequence of send data help us to detect the anomalies and check the possible source of malfunctioning. This technique looks like using floating point signaling NaN values, which cannot be generated during normal floating-point operations, but may be generated as a result of error in computations. Anyway, the discussion of the application layer requires a lot a trade-off, as running real-life application, closely tied to the layer may be extremely time-consuming activity for the simulation engine.

LoRa. It is mainly used at "last mile" or as a network part between the backbone and the "last mile", usually supporting data routes in the case of large scale "last mile", when this "mile" covers the dozens of square kilometers [7]. This level may constantly loose the data due to poor network connectivity, signal shielding, non-stable network due to device movement across land or in sky [8]. Complex LoRa network suffers from various kind of data losses, so the finally handled data on servers have empty records, void data, data non-synchronized in time and another various problems, specific to ad-hoc networks. This requires lowering the requirements for sustainable data transmission and lowering the limit where we treat the network as good working despite of some data loss. Scenarios alike [9], where traffic jamming may be caused intention-

ally by external actors, we treat as less dangerous, but still the anomaly detection system should report such kind accidents to the network owner. Note, that the LoRa infrastructure simulation may be emulated with a kind of data generator, which is able to pro-duce both correct and erroneous streams to check the transport between LoRa host, running a gate from LoRa network to TCP/IP layer, to the server storage. So, this application is separated from main simulation and separates "last mile" network from backbone for analysis simplicity.

MAVLink. The protocol with outstanding popularity in UAV, which should be viewed as kind of standard for modern system, but imposing many limitations to the system [10]. If compared to LoRa services, this protocol is a device control protocol and is used to achieve another goal – to control active devices with actuators. Due to protocol specifics, we have high risk for signal jamming, so anomaly detection should report issues fast, as the device may be lost or work improperly. Instead of producing additional data streams, MAVLink devices health can be controlled by extended connection diagnostics. Connection anomaly reports should be sent to the servers to reflect the device behavior. The network of MAVLink-controlled devices provides less diagnostic data, so anomalies should be detected based on this diagnostics of device connectivity or developed on-device reporting system. Here protocol spoofing is possible, which should be reflected in the anomaly analysis for tasks were protocol data are not protected some way. Also, MAVLink has the property, that it can be implemented over the any protocol, logical or physical, starting from RS-232-like pure physical links and finishing on high-level TCP/IP. So, when we talk about MAVLink as a "last mile" solution, this is mainly for compatibility reasons with other "last mile" protocols at application layer, but the level of anomalies detection may be or at MAVLink level, or at underlying protocol layer (TCP/IP).

Requirement summary. It should be noted, that the reviewed "zoo" of protocols and requirements involve quite a complex set of methods, detecting the anomalies due to different issues common to backbone, "last mile" and a communication mesh between backbone and "leaf" devices. For example, issues for wired connections, fast wireless connections and lower speed wireless connection have different details. For wired connections we may lower the anomaly representation down to "online/off-line" status. One of the authors experienced large packet loss issues due to bad electrical contacts in Ethernet cables, that was very similar to router problems, but this kind of error representation looks very rare in wired networks. So, from the first glance we need a simulation system with main goal to provide wide variety of communication issues cases for analysis. The question of necessary level of details for protocol events will be discussed further. Despite of discussed protocol cases, the network observations should be made separately for 1) backbone and 2) "last mile". For the easiest case we may use "flat" network representation, where "last mile" is also implemented over TCP/IP – this is real situation for robotic devices for MAVLink for now. On the other hand, backbone is a commutation structure, which only transport data from devices somewhere up to servers with negligible errors level.

3. Network analysis approaches

Before discussing the simulation software, we should check modern approaches for network behavior/anomaly/functioning analysis. The goal of our work includes the reflection of useful and real-life scenarios, and current on-the-field situation is that the real-life scenarios highly differentiate from the planned scenarios discussed in research papers even during the latest time. If we go back in time to the beginning of modern ad-hoc networks, we will see TinyOS/Tossim sensor operation system [11] and corresponding simulation software, and hundreds of related research articles, where different scenarios and metrics where considered. So now it's hard to find some theoretic question, related to sensor/wireless network performance, robustness, maintenance and different metrics calculation, which was unattended by researchers. Still the real and useful devices, e.g. agricultural and surveillance drones, transport drones, differ very much from the expectations due to technological advances. We even cannot say words "expert expectations", because the market niches for drones are formed very fast

and market is highly segmented for specialized drones targeted for "narrow" niches in market, due to technological improvements and price changes. Now drone market can redefine economic sectors, such as Chinese agricultural markets changes for automation with drones help. So, the robotic/drone industry development outreaches expert expectations practically in all areas. The core question which rises here is the data volume, which we need to gather in order to get valuable results for analysis. As it was stated before, TCP/IP protocol has four levels, and all four levels can provide valuable data: physical layer provides jamming statistics, transport layer provides data about algorithmic problems of protocol functioning, and these results are meaningful if robotic devices have problems with network functioning. We do not account the application on the drone, generating its own events. TCP/IP application layer is the most complex layer, because it potentially provides the big number of metrics, and the real analysis needs at least temporal information – the events sequence with appropriate time stamps, which are valuable for analysis. Application layer can pro-vide metrics related to system software, the most useful are denial in resource allocation, operation system alerts and hardware failures.

The important issue is filtering of events. Useless logging of events prevents the effective use of analysis system where the operator sees the same messages over months and really intellectual system should decrease this number down to zero.

Industrial application tends to fast and practical application of drones, so if we drone works incorrectly is should be changed to another one, similar to any industrial appliance. It's much simpler to throw away the device than to spend time to repair it, considering associated transporting cost to repair facility, so this effectively prevents accurate analysis of failures. So, one of the primary goals of analysis is the prediction of failures, which is strictly tied to robot types and construction, and need to be reviewed on filed data.

3.1 Analysis methods

Here should be discussed the use of supervised methods vs unsupervised for analysis of failures and anomalies. The supervised methods require some knowledge about the area, where a drone group operates. For a basic example we can compare open field somewhere in country (agricultural works) versus works inside town or some area of dense buildings, where wall blocks signal or reflects them. Depending on signal frequency, the physical processes of signal shielding on different frequences vary, so the analysis based on signal fading – may fail for changed frequency bands. Such obstacles require some kind of temporal logic, for example in case of signal blocking the robot may pass the data to servers once per some time, and this should be accounted into the model. For open country field we can use RSSI information (mean power of received signal) to evaluate distance between transmitter and receiver, but this does not work for dense building. But from our experience the harder problem with supervised learning is the additional cost necessary for building the model. Even if we have assumption about how we can analyze metrics and even applying some general neural net-work model – it is necessary to improve the model, as it can constantly give biased results which are not so valuable for day-by-day work. After some tuning time, these models can be used for particular drone swarm configurations, but the final system should be used friendly, giving short diagnostic messages but not long logs.

Anyway, the learning methods are successfully applied [12][13] for drone algorithmics, mostly for less or more swarm functioning, which can be formalized and not so related to random external processes – for example, building routes, finding some patterns in control application to optimize traffic, transport efficiency optimization and so on. Still we need to provide quite widely applied analysis, which may be narrowed somewhere to the list of events, which may represent event classes more clearly.

3.2 Unsupervised learning

Unsupervised learning as another option do not require pre-built database with defined cases and information how to classify them. Instead, we need accurately provide information gathered from the simulation and check if the selected unsupervised method is able to catch some pattern or select anomalies.

For the first hypothesis we decided to address the method we have previously used for different type of data, described in [14]. We proved, that a large data array, that definitely have relations in some groups – for example, database errors for node #12 are related to packet lost at the gate #5. Clusterization methods, used in [14], can work with such dependencies and can select such groups. The OPTICS method, used earlier, is improved constantly over time, for example [15], and can be used for clustering of the data gathered in TCP/IP network.

As the TCP/IP network events are developed in time, it is necessary to provide some linking of events for analysis either for supervised or unsupervised methods. In common sense for any event for supervised system, we should have a set of properties $P_i=\{p_1,p_2,\ldots,p_n\}$, which represents the state of the system – it is 2D-image for image recognition system, or an internal state of the communication system. For communication system each pi may be the traffic volume over the node interface, number of use TCP/IP buffers on the device, number of error interface after last check, time-to-live of the packet and some other property set we see useful for our investigation. Also, we have a classification definition $C_i=\{c_1,c_2,\ldots,c_n\}$ for each event, where c_i is at least a pair $\{e_i,r_i\}$, where e_i is the classificatory for the set P_i, for example "interface denial at port X", and the r_i is the probability of this event, as the resulting classification c_i is not the only solution, another classification exists. For example, packet loss at some interface may be caused by traffic overload, fail of router interface or simple hardware wiring fail. In order to resolve this loss issues, we have proposed in [14] the use of cascade of clustering-based classifiers. If the first classification provides a set C_i, defining the possible solution, there is the set of classifiers $\{ C_{i,j} \}$, where i refers to the classifiers for the problem, and j refers to the number of classifier results, which was got near the current system time. Here we need to observe time stamps of the system messages, for example event time stamps gathered withing +/- 100ms interval on the network. The second level unsupervised method can cluster the sets $\{ C_{i,j} \}$, giving even more useful info, but the potential of this method should be proven later on real data.

Another point is the time-linking of the events. Logically linked events should be linked in time, as some event with P_i (and C_i) will produce some event P_j on the neighbor node, as the outcoming packet at one node will be incoming at some other mode. As P_i is introduced at time t_i, and P_j at time t_j, where is no reason to link the linked in time events to some discrete times t_i, t_j. Instead, we introduce the time difference $dt_{i,j} = t_j - t_i$, so that the time link is the time difference or log of time difference. On the top of event properties P_i and P_i we form a joined event with properties $\{p_{i,0}, p_{i,1}, \ldots, p_{i,N}, p_{j,0}, p_{j,1}, \ldots, p_{j,n}, dt_{i,j}\}$, which makes the events P_i, P_j linked in time, so the clusterization algorithm will have a set of $\{\{p_{i,k}\},\{p_{j,k}\},dt_{i,j}\}$ events, where $dt_{i,j}$ changes from some minimum time interval to maximum values, depending on communication type. At the clusterization process, the event collection will treat rarely values of $dt_{i,j}$ as anomalies, selecting sets $\{p_{i,k}\}+\{p_{j,k}\}$, where possibly we have the decryption, what is wrong with the communication process. This time-linking method allows to move dependencies from temporal relationship to dataset, applicable for unsupervised methods. Time-linking is not a good way to link event on neighbor communication net-work elements, but also occasionally may link events on elements, located quite far from one another. This should be made during investigation of event details, when we make hypothesis on the root causes of the events in the network.

So, in order to define the basic requirements for simulation engine, we made an assumption what kind of data we need from simulation system. Our data include 1) selected fields from packets with data (at least sequence number from TCP); 2) packet headers; 3) interface identification; 4) temporal data $dt_{i,j}$; 5) software state. Depending on this we are selecting the proper simulation engine.

4. Simulation software

During last two decades we had no dramatic improvements in common net-work simulation software, related to TCP/IP software. The most valuable choices for out research are NS-3 network simulator and OMNeT++, which are widely used and have extreme number of appli-

cations. Both are discrete event simulators. NS-3 have the long story, well known predecessor NS-2 and integrates TCP/IP stack implementation derived from Linux/FreeBSD, so accurately simulates the network functions, and supports integration with analysis software, e.g. Wireshark. On the other hand, it has very rudimentary animation packages, that makes NS-3 not very pleasant for presentations. This issue looks raising, as if we compare NS-3 implementation with a species of simulation software, developed for UAV device models, the comprehensive list can be found in [16]. Many simulators listed in [16] provide 3D visualization interface, including virtual reality engines for wearable goggles or just interfaces to modern visualization engine such as Unity. OMNeT++ provides usual simulation support, but includes Inte-grated Developer Environment (IDE), based on Eclipse (IDE), which includes visual editor for network configurations, text editor with syntax highlight, rich interface for defining object hierarchies and rich logging abilities, including charts and graphs. Several examples are shown further in text. Simulation practically resembles process of usual software development in Eclipse, which is familiar to many programmers.

Additionally, we tried to use Cisco Packet Tracer for our simulation at the be-ginning of our study. The Cisco Packet Tracer is simplified at interface part, if compared to OMNeT++, and resembles rather some kind of SCADA system, as allows to control devices in real time, where device state is renderer over some map, giving a good visualization. After some time, we dropped this system for OMNeT++ use, as we experienced the Packet Tracer system to be more valuable for education processes and consuming too much resources for day-by-day use as simulation engine.

As a conclusion over systems, such as NS-3, OMNeT++, Cisco Packet Tracer, other was enlisted in [16] as a good examples of digital twin concepts applications and quality visualization in 3D, we should state that there are no so many choices for network simulation, so OMNet++ was selected for our studies.

4.1 Targeting simulation to OMNeT++

One of the big OMNeT++ advantages is that it can provide detailed simulation at TCP/IP level, as we need to learn many aspects of TCP/IP transport. It's well aligned with latest DJI/Autel drones, which have complex video capturing devices on the board, computer vision software, including collision avoidance with the use of stereo-camera, wireless technology, which streams HD video over big distance. Finally, SDK allows to have complex software on board, so complex software, utilizing many drone options can be implemented. Simplified application software may be used in model and integrated into OMNeT++. We state that the orientation for the complex drone use, as Autel/DJI is, is not redundant. For the price of several thousands of dollars, these drones provide a rich set of hardware, which can be tried for providing different real-time video processing algorithms. Although the swarm of such drones looks ahead of the time, our study makes closer the implementation of intellectual robot/drones swarms.

We have implemented a basic model for drone use over a GrADyS-SIM simulation package [17], as this package was already based on OMNet++ and targeted for UAV use. One of the advantages of OMNeT++ based system is the visualization of swarm functioning, which makes the simulation close to SCADA system. Fig. 1 shows the visualization.

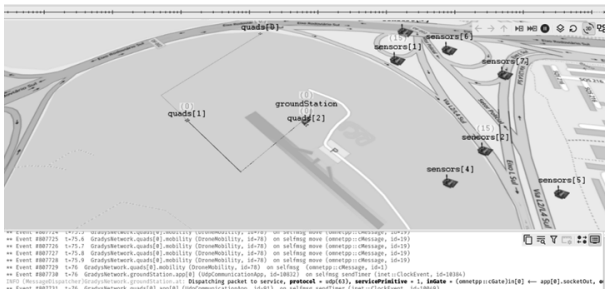

Fig. 1. Swarm simulation visualization

Fig. 1. shows the advanced Qt library-based interface, derived from GrADyS-SIM, which displays map, UAVs, its routes, ground stations and additional sensors over the map. Such kind of visualization consumes well enough computation resources, but it can be switched off. The log text, visible under the visualization, traces the packets. The log includes the packet contents, time mark, source and destination interfaces. Some robots may provide retranslation, and here the map is very valuable at big scale, as the user sees geograph-

ical placement of drone over the map, allowing to form desired network configuration during simulation. Additional modules allow to generate “issues” during the modeling, so the fig. 2 shows the example of the log with an error.

The log data are processed to form input data set for the classifier. We have not used direct error messages as in fig. 2: we see it in simulation log during simulation, but during the real run of swarm system, the error diagnostic not always is messaged to the base station or server. Instead, we are extracting basic statistics information from the packet transport: interface identifiers and time stamp, also if applicable critical application about packet sequences or data in packets. These data packets are transformed to n-dimensional vectors, and passed to the classifier. OPTICS classifier is able to distinguish packet with “incorrect time” and select them as anomalies. In reality this scenario sample looks very simple but working one for our simulation system.

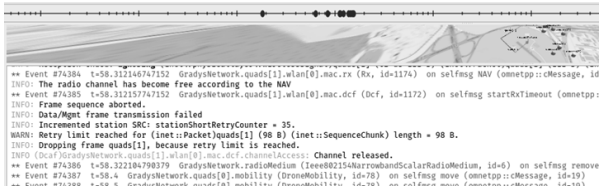

Fig. 2. Simulation error during packet transmission

Despite the simplicity of the sample, we are able to implement more complex cases. The case with frame drops looks obvious, but the method can hold more complex cases, for example drones inside radio-silence zones, packet prioritizing, changing robot software behavior depending on current state of the network transport. In the embedded devices world, it is known that TCP/IP has internal setting, which allows to handle packets in more sophisticated way if compared to default “big system” settings, at least for packets handling during long transport denials. Another case is regularizing TCP traffic in time, limiting flooding packet from particular drone. These questions are the topics for further research, as the proposed system, which consists of packet gathering for analysis and learning engine, can process networking information in real-time, as method, provided in [14] and [15], allows to provide feedback to devices in real-time, regularizing traffic in cases if, for example, the network transport cannot handle the all traffic. Huge traffic from some nodes can flood retranslator units, as wireless channel bandwidth is not as wide as cable channels have, so the precise analysis of distinct segments in the network via smaller classificators can detect upcoming transport problems practically in real-time, providing feedback for nodes and saving valuable information from being lost.

Conclusion

During our investigations we have looked for a good framework for analyzing potential problems in network transport simulation. We think we have moved towards our goal:

1) we have found the modern network simulation system with rich facilities, simulating a swarm of drones with enough level of details suitable for modern networks, including “last mile” such as MAVLink;

2) we are able to collect necessary data in this model and provide complex scenarios of robot interoperation;

3) we tried an unsupervised learning classifier, which is able to detect anomalies in providing data.

Our next goals are to extend working scenarios, make the model more complex and provide additional research for anomaly detection methods.

References

1. H. Wang, H. Zhao, J. Zhang, D. Ma, J. Li, and J. Wei, “Survey on unmanned aerial vehicle networks: A cyber physical system perspective,” IEEE Commun. Surveys Tuts., vol. 22, no. 2, pp. 1027–1070, 2nd Quart., 2020.
2. Abdelkader, M., Güler, S., Jaleel, H. et al. Aerial Swarms: Recent Applications and Challenges. Curr Robot Rep 2, 309–320 (2021). https://doi.org/10.1007/s43154-021-00063-4
3. Tsao, K.-Y., Girdler, T., Vassilakis, V. (2022). A survey of cyber security threats and solutions for UAV communications and flying ad-hoc networks. Ad Hoc Networks. 133. 102894. 10.1016/j.adhoc.2022.102894.
4. Gupta, L., Jain, R., Vaszkun, G. "Survey of Important Issues in UAV Communication

Networks," in IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1123-1152, Secondquarter 2016, doi: 10.1109/COMST.2015.2495297
5. Phadke, A., Medrano, A., Starek, M. (2024). U-SMART: unified swarm management and resource tracking framework for unoccupied aerial vehicles. Drone Systems and Applications. 12: 1-26. https://doi.org/10.1139/dsa-2024-0007
6. Alrayes, M., Elgaber, A., Khalifa, Z., Alfagi, A. (2021). Performance Study of TCP Variants in FANETs. International Science and Technology Journal. 26. 267-279. https://doi.org/10.62341/maza2627
7. Paredes, W. D., Kaushal, H., Vakilinia, I., & Prodanoff, Z. (2023). LoRa Technology in Flying Ad Hoc Networks: A Survey of Challenges and Open Issues. Sensors, 23(5), 2403. https://doi.org/10.3390/s23052403
8. Zirak, Q., Shashev, D., and Shidlovskiy, S. "Swarm of Drones Using LoRa Flying Ad-Hoc Network," 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 2021, pp. 400-405, doi: 10.1109/ICIT52682.2021.9491655.
9. Zilberman, A., Stulman, A., Dvir, A. "Identifying a Malicious Node in a UAV Network," in IEEE Transactions on Network and Service Management, vol. 21, no. 1, pp. 1226-1240, Feb. 2024, doi: 10.1109/TNSM.2023.3300809.
10. Koubâa, A., Allouch, A., Alajlan, M., Javed, Y., Belghith, A., Khalgui, M. (2019). Micro air vehicle link (mavlink) in a nutshell: A survey. IEEE Access, 7, 87658-87680.
11. Levis, P., Nadnarin, L., Welsh, M., Culler, David. (2003). TOSSIM: accurate and scalable simulation of entire TinyOS applications. In Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys '03). ACM, New York, NY, USA, pp. 126–137. https://doi.org/10.1145/958491.958506
12. Arranz, R., Carramiñana, D., Miguel, G. d., Besada, J. A., Bernardos, A. M. (2023). Application of Deep Reinforcement Learning to UAV Swarming for Ground Surveillance. Sensors, 23(21), 8766. https://doi.org/10.3390/s23218766
13. Tlili, F., Ayed, S., Fourati, L., Ouni, B. (2022). Artificial Intelligence Based Approach for Fault and Anomaly Detection Within UAVs. 297-308. 10.1007/978-3-030-99584-3_26.
14. Rahozin, D., Doroshenko, A. (2024) Low frequency signal classification using clustering methods. Problems in programming 2024; 1: p. 48-56. https://doi.org/10.15407/pp2024.01.048
15. Tang, C., Wang, H., Wang, Zh., Zeng, X., Yan, H., Xiao, Y. (2021). An improved OPTICS clustering algorithm for discovering clusters with uneven densities. Intelligent Data Analysis. 25. 1453-1471. https://doi.org/10.3233/IDA-205497.
16. C. A. Dimmig et al., "Survey of Simulators for Aerial Robots: An Overview and In-Depth Systematic Comparisons [Survey]," in IEEE Robotics & Automation Magazine, vol. 32, no. 2, pp. 153-166, June 2025, doi: 10.1109/MRA.2024.3433171.
17. Lamenza, T., Paulon, M., Perricone, B., Olivieri, B., Endler, M. (2022). GrADyS-SIM--A OMNET++/INET simulation framework for Internet of Flying things. arXiv preprint arXiv:2202.08134.

Дата першого надходження до видання: 16.06.2026
Внутрішня рецензія отримана: 10.07.2026
Зовнішня рецензія отримана: 11.07.2026
Дата прийняття до друку: 10.08.2026
Дата публікації: 29.08.2026

Про авторів:

Рагозін Дмитро Васильович,
старший науковий співробітник
Dmytro Rahozin, research scientist
http://orcid.org/0000-0002-8445-9921.

Заіка Кирило Анатолійович,
аспірант
Kyrylo Zaika, PhD student
http://orcid.org/0009-0006-1107-5106.

Місце роботи авторів:

Інститут програмних систем
НАН України,
Institute of Software Systems of National Academy of Sciences of Ukraine,
тел. +38-044-522-62-42
E-mail: Dmytro.Rahozin@ukr.net
www.iss.nas.gov.ua