











# Integrating a Multi-Agent System Simulator and a Network Emulator to Realistically Exercise Military Network Scenarios

Dante A. C. Barone<sup>1</sup><sup>a</sup>, Juliano Araujo Wickboldt<sup>1</sup><sup>b</sup>, Maria Claudia Reis Cavalcanti<sup>2</sup><sup>c</sup>,  
David Moura<sup>2</sup><sup>d</sup>, Julio Cesar C. Tesolin<sup>2</sup><sup>e</sup>, André M. Demori<sup>2</sup><sup>f</sup>, Julio C. S. dos Anjos<sup>3</sup><sup>g</sup>,  
Leonardo Filipe Batista Silva de Carvalho<sup>4</sup><sup>h</sup>, João Eduardo Costa Gomes<sup>5</sup><sup>i</sup>  
and Edison Pignaton de Freitas<sup>1</sup><sup>j</sup>

<sup>1</sup>Graduate Program on Computer Science, Federal University of Rio Grande do Sul, Porto Alegre, Brazil

<sup>2</sup>Military Institute of Engineering, Rio de Janeiro, Brazil

<sup>3</sup>Graduate Program in Teleinformatics Engineering (PPGETI/UFC), Federal University of Ceará, Fortaleza, Brazil

<sup>4</sup>Federal Institute of Rio Grande do Sul, Canoas, Brazil

<sup>5</sup>Graduate Program on Electrical Engineering, Federal University of Rio Grande do Sul, Porto Alegre, Brazil

Keywords: Integration, MAS, Military Network.

Abstract: Modern battlefield scenario are complex environment in which a myriad of equipment and people interact to accomplish a given mission. Most of this interaction is performed by means of wireless communication via Command and Control Systems, which efficiency represent a critical factor the mission success. The assessment of these systems, and their supporting networks, is of primal interest to decide for the best equipment and military maneuver approach. However, there is a lack of tools that provide all the necessary behavioral and network features to perform the task. Observing this fact, this work presents an alternative to simulate a battlefield environment model by means of integrating a network emulator and a Multi-Agent System simulator. By combining both software, it is possible to assess specific characteristics of each area without limiting the model, thus providing the necessary data for an informed military network setup assessment.


## 1 INTRODUCTION


Battlefields have been dynamic and hostile environments throughout history that are constantly subjected to unexpected changes. The advance of technology has increased the size and scope of battles. Now, battlefields can spread over multiple domains where adversaries contest their forces over land, air, sea, space and cyberspace. Such evolution has significantly in-


creased the role of communications in military activities and with it, has created the basic concept of Network-Centric Warfare (NCW) (Cebrowski, 1999). This has made it possible to deploy distributed Command and Control (C2) Systems to support Multi-Domain Operations (MDO) (Townsend, 2018).


The complexity and extension of modern military operations make them costly and cumbersome to assess. Therefore, simulation becomes a useful tool to evaluate military scenarios and help armed forces to test new approaches. However, the effectiveness and accuracy of a simulation depend on how close to the real-world operation its model is. Moreover, as new details are added, more complex becomes the model and the demands from the simulator.


An alternative is to break the model into smaller ones, isolating and assessing each aspect of the scenario with a specific simulator. This separation, however valid, removes the interaction between different aspects of the model from the assessment. In order to


<sup>a</sup> <https://orcid.org/0000-0002-5133-0144>


<sup>b</sup> <https://orcid.org/0000-0002-7686-8370>


<sup>c</sup> <https://orcid.org/0000-0003-4965-9941>


<sup>d</sup> <https://orcid.org/0000-0002-1153-3879>


<sup>e</sup> <https://orcid.org/0000-0002-0240-4506>

<sup>f</sup> <https://orcid.org/0000-0002-0533-3395>

<sup>g</sup> <https://orcid.org/0000-0003-3623-2762>

<sup>h</sup> <https://orcid.org/0009-0001-7032-5850>

<sup>i</sup> <https://orcid.org/0000-0003-1418-0658>

<sup>j</sup> <https://orcid.org/0000-0003-4655-8889>

get a better picture of what is happening in the military operation as a whole, it is necessary to broaden the scope instead of isolating its parts. Unfortunately, there are not many simulators capable of assessing very complex military operations models. This is the case when the goal is to assess the performance of troops in Network-Centric Operations, in which communication network aspects directly affect the unfolding of the military units on the field.

Thus, a solution to model such complex environment is to integrate two types of simulators, each covering one aspect of the scenario: unit behavior and network communication between units. Albeit simple, the idea of integrating two different simulators is easier said than done since both programs might run in different operating systems, use different programming languages, and have different time approaches (real-time, simulated time, event-driven), among many other specificities.

Observing the need for simulations that exercise C2 communication network features in realistic military operation scenarios, this work proposes the integration of a Multi-Agent System (MAS) simulator with a Network emulator. This paper presents an overview of the proposed Simulator-Emulator Integrated System, highlighting the main contributions:

- The proposal of an integration between a behavioral model based on Multi-Agents Systems with computer network paradigms able to deploy a realistic behavior of a military network;
- An integrated execution environment entitled System of Systems of Command and Control (S2C2) Emu-Sim;
- The design of timing and decision mechanisms that allows the joint operation of the integrated software.

This work is divided in the following structure: Section 2 brings the main concepts regarding the addressed military scenarios, then Section 3 presents the main challenges involved in integrating two different simulation paradigms, and Section 4 follows with the proposal of the integrated software. Section 5 presents the solutions for the challenges previously presented. Section 6 shows the results obtained on a case study and the paper is concluded with Section 7.

## 2 CONCEPTS REVIEW

The key concepts for military communications on the field are briefly summarized next.

### 2.1 Command and Control

The fundamental core of Command and Control regards the structure and decision making to enable a team of individuals to accomplish a mission (Alberts and Hayes, 2006).

The idea of military Command used to be projected into a single individual (the leader, the genius commander) but the Information Age has shifted that idea from centralized Command to de-centralized Command (Alberts and Hayes, 2003). Therefore, shifting authority and decision power to individuals at the edge of the organization (Boone, 2021) in a new approach that has been called Network-Centric Warfare (NCW) (Alberts and Hayes, 2006).

### 2.2 Military Communication Networks

To modern military conflicts, the communication network infrastructure is highly important. While it includes several methods, it relies mostly on wireless connections (Pawgasame and Wipusitwarakun, 2015). As consequence, there has been a considerable general increase in data rates to support modern applications such as maps, friendly positions and real-time video feeds (Jalui et al., 2019), which has become even more pressing after the introduction of Internet of Battle Things (IoBT) (Kott et al., 2016), which must be capable of dealing with different types of users and data requests. A contemporary example of that are the debilities of the Russian Armed Forces to communicate in the war of Ukraine, which points that most of the difficulties faced by the Russians are based on communication network problems, particularly in regard to acquiring timely data and to the coordination of their actions - clearly C2 business (Cranny-Evans and Withington, 2022).

To overcome the challenges of a variable network configuration and topology, new network approaches have been developed based on new paradigms that aim to improve network connectivity and data transfer between nodes while providing robustness, such as Information-Centric Network (ICN) (Campioni et al., 2019), Disruption/Delay-Tolerant Network (DTN) (Amin et al., 2015), Software Defined Network (SDN) (Nobre et al., 2016) and combinations of them (Wang et al., 2017), (Zacarias et al., 2017), (Leal et al., 2019).

### 2.3 Decision Making Process

Every military group has its own hierarchy and discipline. However, as NCW advances, responsibility and decision power shift to de-centralized individu-

als at the edge of the organization. Hence, each troop must be capable to make its own decisions in view of the main objective of the operation.

One way to simulate such behavior is to use Multi-Agents System (MAS) simulators. The core abstraction of agent is defined by (Dorri et al., 2018, p. 2) as “an entity which is placed in an environment and senses different parameters that are used to make a decision based on the goal of the entity. The entity performs the necessary action on the environment based on this decision”. This definition perfectly models the military units and the military operation scenario in which they are inserted.

However, while MAS simulators create communications between agents to allow them to interact with each other, those are simple and aim only to exchange knowledge of the environment. Thus, simulators usually do not assess real network parameters.

## 2.4 Conceptual Data Modeling

It is essential for the success of software development endeavors to conciliate both developers’ and experts’ perceptions about a domain while capturing it. Conceptual Data Modeling aims to provide such alignment, bringing semantic-rich concepts to represent reality as accurately as it can be, becoming more understandable and implementation independent. In this sense, the modeling language must provide the essential constructs for such task. Otherwise, some important concepts may be left out of the final model.

Ontological Conceptual Data Modeling (OCDM) can be used on large and complex information system, instead of Traditional Conceptual Modeling (TCM), to bring substantial benefits. In (Verdonck et al., 2019), authors observed in their empirical study that novice modelers (modelers without previous data modeling knowledge) using OCDM techniques brought higher quality models when compared to the ones brought by novice modelers using a TCM technique. Hence, ontology-based conceptual modeling tools, such as OntoUML (Guizzardi et al., 2015), are able to deliver better domain conceptual models, improving not only their reading clarity and implementation, as well to improve the reasoning capabilities of a system. Using ontology also reduces ambiguity and avoids semantic conflicts within the model.

## 3 PROBLEM STATEMENT

Modern battlefield scenarios cover aspects of decision-making and network communications that make realistic simulation models complex.

As more details and parameters are added, more is required from simulators. Although there are network simulators and emulators able to efficiently assess network parameters, and to evaluate different network paradigms, they have poor support to realistic represent the behavior of military troops on the field. This includes making decisions regarding the presence of enemies and performing coordinated maneuvers to avoid natural or man-made obstacles.

On the other hand, while making decisions and coordinating the movement of units are tasks better evaluated by MAS simulators, they are unfit to assess/test new network paradigms or parameters. Hence, to integrate these two types of simulators provides a way to improve complex military operation models to network communications and decision-making as well.

To use those two tools in synchrony requires to account some key factors, such as the simulation progress time having to occur at the same pace in both programs. Otherwise, events occurring in one program (and their consequences) are not properly reflected at the other. This is not a simple task when combining an emulator and a simulator. A simulator can accelerate (or slow down) time, while an emulator uses real-world clock time. This creates restrictions to how fast (or if) the scenario can be sped up to.

The kind of information shared between software, how it is shared and the physical position of units on the field are also matters to consider. Particularly, continuous and synchronous update of positions on both software is necessary so that aspects like signal loss/variation or loss of packages can be added to the model and to the decision-making process. That way, movement decisions are dealt by the MAS simulator while the network emulator uses the position of units to figure the network behavior. Thus, a common data interface to share information between these two types of tools becomes a requirement. However, that is not part of the design goals of either of them.

One way to synchronize data communication between those programs is to write and read data from external files, e.g. .txt or .csv files. That way, data generated and registered by one program can be accessed by to the other. Yet, this solution might lead to issues like semantic conflicts (see Section 2.4), concurrent file requests and file management problems.

To overcome such problems, an alternative is to write and read data from databases as the use of a well-conceived data model might facilitate interoperability. This makes it possible to register the configuration of the simulation and of each step of its steps, therefore, making it easier to compare, debug or identify outliers. The drawback of this approach is that if often requires the use of external *scripts* or third-

party software (such as DBMS) to enable direct access database files. Other than, that, the permission to read/write access into the database file is also an important concern.

The simulation of military scenarios requires to use information from real locations, with natural geographic elements such as rivers, mountains and valleys, as well as man-made elements such as bridges and roads. Hence, the use of maps with Geographic Information System (GIS) is a good option. This because, MAS simulators can be tuned to interpret each geographic information and use them in path-finding algorithms like Dijkstra or A\*. On that matter, weather conditions may also be another source of interference for units' movement and network communications that should be taken into account.

Combining all these concerns described above in a single setup that enables the assessment of network and military doctrine parameters is not a trivial task and requires the integration of different software systems. In turn, this integration has a number of challenges that have to be handled in order to work properly. The approach presented in this paper faces these problems as detailed in the following.

## 4 INTEGRATION PROPOSAL

This work presents the integration of a MAS simulator and a Network emulator to realistic represent battlefield environments and the accurate behavior of its troops, including the interoperability issues that might occur in communications in such scenarios.

### 4.1 Architectural Overview

The general structure of the proposed solution defined by the "S2C2 EmuSim - Command and Control Simulation Configuration and Orchestration" system is shown on the component diagram of Figure 1 and are described next from a objective point-of-view. Though most of the inner components of the diagram are omitted, Section 4.2 discusses the sub-components of the "EmuSim Script" component.

- "S2C2 Menu": the component tasked to initialize the system. It provides the "Start" interface used by other components to execute the each or theirs corresponding functionality.
- "ManageSimulation": the component used to create, load and edit a "Scenario (Simulation)" object. It uses the "Run" interface provided by the "EmuSim Script" to run the scenario over the "MAS Simulator" and the "Network Emulator".

- "Scenario (Simulation)": the battlefield scenario to be simulated. Each scenario is built over a battlefield ontology based on the Web Ontology Language (OWL)<sup>1</sup>, a domain ontology for the rich representation of entities, individuals, categories, inferences, attributes, and relationships on the battlefield. It also enables checking the logical consistency between the simulated entities and inferring rules and constraints of the battlefield scenario.
- "Simulation Log": an object created by "Manage Simulation" after a scenario has been run and had its data saved to the database.
- "S2C2 OWL Loader": a component used to convert to/from OWL data through its "Parse" interface.
- "MAS Simulator": the Multi-agent System Simulator used to simulate the troops and the battlefield environment of the military scenario under analysis, including the decision-making and path-finding algorithms used to choose routes and actions that should be taken to reach the goal.
- "Network Emulator": a program used to emulate the communication signals (and the success/failure of their delivery) of military troops simulated by the MAS Simulator.
- "EmuSim Script": a component tasked to synchronize the "MAS Simulator" and the "Network Emulator" and to run the input battlefield Scenario.
- "EmuSim Persistence": a component used to insert, update or read data base data through its provided "DB Query" interface.
- "Manage Simulation Report": a component used to read the "Simulation Log" object and view the resulting data from the execution of a Scenario in a user-friendly way. It is also used to configure the report, selecting which statistics should be shown.
- "Manage Simulation Set": a component tasked to build and read a "Simulation Set" object and to sequentially run each of its scenarios.
- "Simulation Set": the object file containing a set of different configurations of the same "Scenario (Simulation)" file that should be run to collect their data and to produce reports to identify the best resulting strategies.
- DBMS: the database of the system.

The need to synchronize the "MAS Simulator" and the "Network Emulator" comes from the fact that

<sup>1</sup><https://www.w3.org/OWL/>

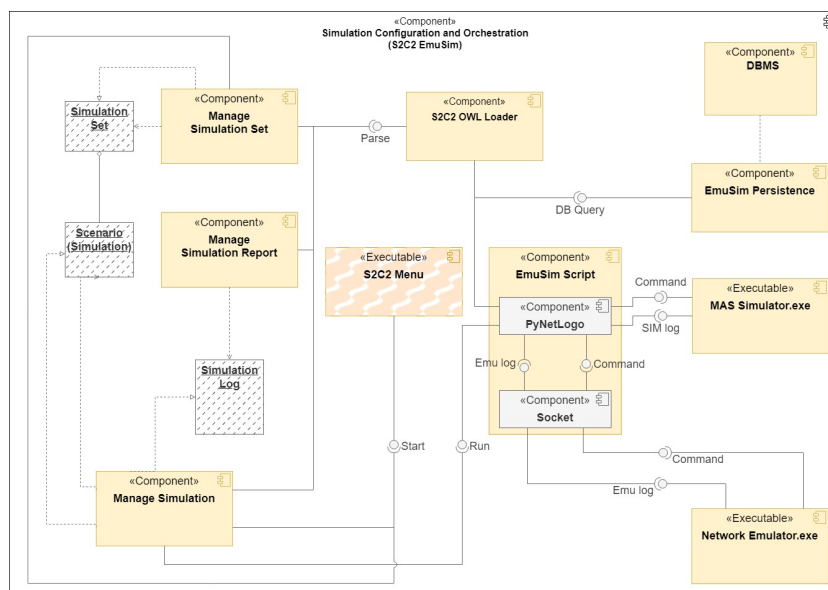


Figure 1: S2C2 EmuSim - Command and Control Simulation Configuration and Orchestration.

both use co-simulation (Gomes et al., 2018) to simultaneously run the same military scenario over their respective settings. This is critical to this work since that, different from a simulator, an emulator works in real-world time only.

### 4.2 EmuSim Script Component

The “EmuSim Script” component manages the general functioning of the S2C2 System. It also bridges the events triggered by the use of the system and the external applications responsible to them, such as the “MAS simulator” and the “Network emulator”, respectively NetLogo and Mininet WiFi.

NetLogo is a multi-agent programmable modeling environment (Wilensky, 1999). It is a robust open-source platform based on Logo programming language and implemented in Java and Scala. Mininet-WiFi is a lightweight open-source emulator developed in Python used to create realistic virtual networks that runs at the same computer real kernel, switch, and application code(Lantz et al., 2010), as well as wireless connections with access points, ad hoc communication, and mesh networks (Fontes et al., 2015).

When executing a scenario, the “MAS Simulator” is tasked to start and end the simulation, as well as to set the number of nodes (agents), their environment, primary objective, and behavior when encountering enemies. Meanwhile, the “Network Emulator” has the task of asserting the nodes’ success/failure to communicate in order to track the chance of friendly fire.

The “EmuSim Script” is triggered by the “Manage

Simulation” component whenever a scenario simulation must be run. Next, it simultaneously start the “MAS simulator” and the “Network emulator”. This task is carried out by its two sub-components seen in Figure 1: “PyNetLogo” and “Socket”.

The “PyNetLogo” sub-component is a Python Script to send instructions to NetLogo and to Mininet-Wifi. It provides the “Command” interface to send messages that deliver these instructions. However, the distributed nature of the Network Emulator requires that messages are delivered to network nodes, which may not be physically located on the same computer. For that reason, messages sent to the “Network Emulator” through the “Command” interface are first delivered to the “Socket” sub-component that, in turn, forwards them through its own provided “Command” interface to the network nodes emulated by the “Network Emulator”. Details of the complete flow of this process are described in Section 4.3.

Once the simulation is over the resulting data is collected through the “SIM log” interface connecting the “PyNetLogo” sub-component to the “MAS Simulator”, and the “Emu log” interface connecting it to the “Socket” sub-component. In turn, this component uses its own provided “Emu log” interface, gathers data from the emulator and delivers it back.

### 4.3 System Workflow

To run a simulation scenario on the “S2C2 EmuSim - Command and Control Simulation Configuration and Orchestration” system requires a number of steps. The flow of this functionality is detailed next.

1. The user starts the System using the “S2C2 Menu” component and selects to manage a simulation (“Manage Simulation component”).
2. The user requests the “Manage Simulation” component to load a previously built simulation.
3. The “Manage Simulation” component” retrieves the scenario data and gives the “EmuSim Script” component charge of the simulation.
4. The “EmuSim Script” initializes the database of the system, the “MAS Simulator” and the “Network Emulator” to set them ready to store data and to run commands.
5. The “Network Emulator” instantiates every network node of the emulated scenario, each with its own client and server objects, so it can monitor the success/failure of their exchange of messages. Each loaded scenario may have any number of emulated network nodes that, in turn, communicate with one another. Here, an abstraction is used to represent this unknown number of nodes as the “ClientNodeA”, “ClientNodeB”, “ServerNodeA” and “ServerNodeB” objects.
6. Next, the system runs the simulation to collect its results. This starts a loop that is repeated one tick<sup>2</sup> at a time, until all the nodes of the simulation have reached their goal. It should be noted that, while the simulation moves forward one tick at a time the simulator can be set to run any number of ticks per second. The next sub-items detail this loop.
  - (a) The “EmuSim Script” requests from the “Database” the communication messages of all nodes (if any) that were written at the last tick.
  - (b) The retrieved data is sent to the “MAS Simulator” to use it to update for every node of the simulation the information of each ally node they are aware of.
  - (c) The “MAS Simulator” calls itself to update the data about any hill between two simulated nodes. This step is essential to simulate geographical interference that may affect troop members and lead to friendly fire.
  - (d) The “EmuSim Script” calls the “MAS Simulator” to run the present tick. Next, it retrieves from it the current positions of the simulated nodes and writes it to the “Database”.
  - (e) The “EmuSim Script” calls the “MAS Simulator” to retrieve the information of every hill located between the current position of any two nodes. It then writes the data to the “Database”.
  - (f) The “EmuSim Script” calls the “Network Emulator” to start the communication of the nodes using the MQTT (Message Queuing Telemetry Transport) protocol, a machine-to-machine network protocol for message queue/message queuing service (FairCom, 1999). This action starts a new loop (detailed next) that runs on a single tick to every simulated node.
    - i. The “Network Emulator” sends an asynchronous message to “ClientNodeA” via MQTT protocol.
    - ii. The message is received by “ClientNodeA” which sends an asynchronous UDP message to “ServerNodeB” to assess communication success/failure.
    - iii. To every successfully received message, “ServerNodeB” sends an asynchronous message to “ClientNodeB” to write that message.
    - iv. “ClientNodeA” requests the database to persist every message it sent at the current tick.
    - v. “ServerNodeB” requests the current simulation tick from the database. Then, it writes to it every message it received at that tick.
    - vi. Next, steps 6(f)i to 6(f)v repeat themselves to the opposite client-server pairs. First, the “Network Emulator” asynchronously messages the “ClientNodeB” via MQTT protocol.
    - vii. When a message is received by “ClientNodeB” it sends an asynchronous UDP message to “ServerNodeA” to assess communication success/failure.
    - viii. To every message it successfully received, “ServerNodeA” sends an asynchronous message to “ClientNodeA” to write that message.
    - ix. “ClientNodeB” requests the database to persist every message it sent at the current tick.
    - x. “ServerNodeA” requests from the database the value of the current simulation tick. Afterward, it writes to the database every message it received at that tick.
7. The “EmuSim Script” fires messages to stop the “Network Emulator” and the “MAS Simulator” and to disconnect the database.

## 5 CHALLENGES

Time synchronization is crucial to integrate the “MAS simulator” and the “Network emulator”. Contrary to the parameters evaluated by the former, the ones assessed by the latter require real-world clock time. Thus, to optimize the execution time of the simulation the “EmuSim Script” acts as a pacemaker to keep both

<sup>2</sup>The internal time frame unit of the simulation.

tools running at real-world time speed. Hence, while this avoids disturbing the assessment of the parameters of the emulator (e.g. transmitted packets, established connection, etc) it allows the quick transfer of the position of units from the MAS to the emulator.

To enable the “EmuSim Script” to control the simulation pace and to achieve synchronization, the smallest time fraction of the “MAS simulator” was set to one (1) second. This also made it possible to track parameters of interest along the run of the simulation.

The management of the database is a task of the “DBMS” as it controls the read and write operations and the overall access. This is required due to the “MAS simulator”, “Network emulator” and “EmuSim Script” having different read/write privileges to most database tables. Other than that, the use of a database also allowed saving data to different configurations of the simulation, such as the number of units or broadcast communication intervals.

The last challenge faced in co-simulate the battlefield scenarios was scale. Mininet-WiFi uses real-life metrics while NetLogo uses different scales according to the map file. Yet, while NetLogo can use real geographic data from GIS files, the map scale has to be set within the simulator. To integrate these different scales, it was required to use a scale factor to keep the position of units proportional in both software.

## 6 CASE STUDY

Sections 4 and 5 show that the problem described in Section 3 is addressed by integrating Mininet-WiFi and NetLogo via database and orchestration scripts. Figure 2 illustrates the interfaces of the two synchronized software and is meant to represent a friendly fire case study scenario to validate that solution.

At the top of Figure 2 is the Mininet-WiFi emulator, displaying the telemetry of its emulated nodes in real-world distances. At the bottom is the NetLogo simulator, showing the geography of the current map being simulated and its agents/nodes. As shown, the two programs have the same nodes at the same positions (considering scale correction).

The scenario consists of 29 units (blue hexagons with military symbols) that carry broadcast radios and walk through a real-world terrain marked by plains (light green), hills (darker shades of green), and water bodies (blue). When in execution, NetLogo (bottom map) must consider these aspects and find either the shortest or fastest path to take troops from their initial position (bottom-left red square) over their mid-goals (red squares on the map diagonal) to their goal (top-right red square). To units, water bodies and steep

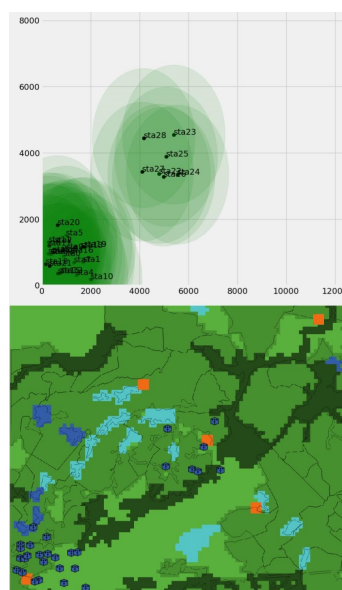


Figure 2: Mininet and NetLogo synchronized interfaces.

hills are impassable while plains and leaser hills have different effects over their movement speed.

To assess possible friendly fire situations, each unit broadcasts its current location every few seconds until it reaches its goal, thus, creating a *blue force tracking*. This is a necessity since there are elements of the map that might block the sight of an agent and lead to friendly fire whenever an agent detects an unknown agent in its presence. The same scenario can also be customized to run under different time intervals to analyze the impact these differences have over the friendly fire.

Particularly, longer intervals provide less information to agents about their peers, thus increasing the encounter of agents unaware of the other’s nature (friend or foe) and the potential of friendly fire. At the end of each simulation, the value of the used test-parameter interval and the results of the simulation are shown in a report with a set of batches that lists for each agent the average number of encounters leading to friendly fire and their standard deviation.

## 7 CONCLUSIONS

Despite the complexity to simulate modern military operations, this paper proposes how to handle some of that complexity by integrating two software. Moreover, while this architecture allows to simultaneously assess bigger complexities and larger number of parameters, it also prompts a model behavior closer to reality and avoids oversimplifications and loss of information.

To that end, aspects such as timing, data format, shared information, and the orchestration of the involved software had to be considered. Synchronizing these software was crucial so that the model could take into account communication and decision-making factors, while the database assured the evaluation and register of these factors for further analysis and comparison as well as the build of the case study.

Due to the success of this work, further developments are expected to communications and decision-making. This includes the testing of new network paradigms (such as ICN, DTN) and the addition of new elements to influence the behavior of agents (e.g. limited resources, enemies, etc.).

## ACKNOWLEDGEMENTS

This study was financed in part by CAPES - Brazil - Finance Code 001 and in part by CNPq - Brazil, Projects 309505/2020-8. We also thank the Brazilian Army via the research project S2C2, ref. 2904/20.

## REFERENCES

- Alberts, D. S. and Hayes, R. E. (2003). Power to the edge: Command... control... in the information age. Technical report, Office of the Assistant Secretary of Defense Washington DC Command and ....
- Alberts, D. S. and Hayes, R. E. (2006). Understanding command and control. Technical report, Assistant secretary of defense (C3I/Command Control Research Program ....
- Amin, R., Ripplinger, D., Mehta, D., and Cheng, B.-N. (2015). Design considerations in applying disruption tolerant networking to tactical edge networks. *IEEE Communications Magazine*, 53(10):32–38.
- Boone, C. M. C. (2021). Decentralized decision making. *Marine Corps Gazette*.
- Campioni, L., Hauge, M., Landmark, L., Suri, N., and Tortonesi, M. (2019). Considerations on the adoption of named data networking (ndn) in tactical environments. In *2019 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–8.
- Cebrowski, A. K. (1999). Network centric warfare: An emerging military response to the information age. *Military Technology*, 27(5):16.
- Cranny-Evans, S. and Withington, T. (2022). Russian comms in ukraine: A world of hertz. <https://rusi.org/explore-our-research/publications/commentary/russian-comms-ukraine-world-hertz>.
- Dorri, A., Kanhere, S. S., and Jurdak, R. (2018). Multi-agent systems: A survey. *Ieee Access*, 6:28573–28593.
- FairCom (1999). MQTT. <https://www.faircom.com/products/faircom-edge>. Accessed: 2023-02-20.
- Fontes, R. R., Afzal, S., Brito, S. H. B., Santos, M. A. S., and Rothenberg, C. E. (2015). Mininet-wifi: Emulating software-defined wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 384–389.
- Gomes, C., Thule, C., Broman, D., Larsen, P. G., and Vangheluwe, H. (2018). Co-simulation: a survey. *ACM Computing Surveys (CSUR)*, 51(3):1–33.
- Guizzardi, G., Wagner, G., Almeida, J. P. A., and Guizzardi, R. S. (2015). Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. *Applied Ontology*, 10(3-4):259–271.
- Jalui, S., Hait, T., Hathi, T., and Ghosh, S. (2019). Advanced military helmet aided with wireless live video transmission, sensor integration and augmented reality headset. In *2019 International Conference on Communication and Electronics Systems (ICCES)*, pages 123–127.
- Kott, A., Swami, A., and West, B. J. (2016). The internet of battle things. *Computer*, 49(12):70–75.
- Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 1–6.
- Leal, G. M., Zacarias, I., Stocchero, J. M., and Freitas, E. P. d. (2019). Empowering command and control through a combination of information-centric networking and software defined networking. *IEEE Communications Magazine*, 57(8):48–55.
- Nobre, J., Rosario, D., Both, C., Cerqueira, E., and Gerla, M. (2016). Toward software-defined battlefield networking. *IEEE Communications Magazine*, 54(10):152–157.
- Pawgasame, W. and Wipusitwarakun, K. (2015). Tactical wireless networks: A survey for issues and challenges. In *2015 Asian Conference on Defence Technology (ACDT)*, pages 97–102.
- Townsend, S. J. (2018). Accelerating multi-domain operations. *Military Review*, pages 4–7.
- Verdonck, M., Gailly, F., Pergl, R., Guizzardi, G., Martins, B., and Pastor, O. (2019). Comparing traditional conceptual modeling with ontology-driven conceptual modeling: An empirical study. *Information Systems*, 81:92 – 103.
- Wang, H., Tang, H., and Zhang, S. (2017). Joint optimization in software defined wireless networks with network coded opportunistic routing. In *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 298–302.
- Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo/>. Accessed: 2023-02-15.
- Zacarias, I., Gaspary, L. P., Kohl, A., Fernandes, R. Q. A., Stocchero, J. M., and de Freitas, E. P. (2017). Combining software-defined and delay-tolerant approaches in last-mile tactical edge networking. *IEEE Communications Magazine*, 55(10):22–29.