Design of Multi-Agent UAV Simulator to Support the Development of the MARSNet Communication Protocol

Jonathan Ponniah* and Alexander K. Yee†
San Jose State University, San Jose, CA 95192

Or D. Dantsker‡ and Simon Yu§
University of Illinois at Urbana–Champaign, Urbana, IL 61801

Renato Mancuso¶
Boston University, Boston, MA 02215

Distributed multi-agent UAV systems motivate fundamental problems in control theory and distributed networking. Although these problems are often studied in isolation, emerging applications of UAV systems require an approach that jointly addresses the control and communication issues of interest. One major obstacle to this approach is the lack of simulation tools that capture the dynamics of both fields. This paper describes the design of a simulation tool that combines two independent simulators for robotics and networking. The proposed simulator combines ns-3 with ROS/Gazebo, through a middle-ware interface. Some of the fundamental challenges of this architecture are discussed and resolved.

I. Introduction

Distributed multi-agent UAV systems are the underlying technology for many emerging applications including environmental surveillance, topographical mapping, on-demand provisioning of wireless connectivity, and aerial delivery of goods to designated locations. However, several fundamental problems in control-theory and distributed communication systems must first be resolved before these applications become truly viable.

Control-theoretic issues generally center around the problems of task assignment and multi-agent motion-planning. The former refers to the problem of distilling high-level commands ("find the locations of all burning trees in this forest") to the low-level agent-specific instructions required to actually execute the given command. The latter refers to the problem of deriving non-intersecting trajectories for global and local path planning, in a computationally efficient manner. Communication-related issues stem from the challenge of ensuring individual agents can coordinate their activity without a centralized controller. Both sets of problems are separate areas of research in their own right.

Since the control and communication aspects of distributed multi-agent systems are tightly interwoven, an approach that jointly addresses these issues is preferable. However, efforts in this direction are stymied by the lack of tools that simultaneously simulate the all dynamics of individual interest. Instead, much of the research in control and communication theory depends on simulation tools that cater exclusively to one of these two communities. To further complicate matters, communication-oriented simulation tools are event-based, whereas control-oriented tools are time-based.

This paper describes the design of a simulator that interfaces ns-3,[1] an event-based network simulator, with ROS[2] and Gazebo,[3] a time-based robotics simulator and physics engine. The proposed simulator is not a replacement for either ROS or ns-3. Rather, it is part of broader efforts to develop a demand-response framework for the decentralized control of multi-agent systems that jointly addresses some of the challenges in task assignment, motion-planning, and distributed communication.

*Assistant Professor, Department of Electrical Engineering. jonathan.ponniah@sjsu.edu
†M.S. Student, Department of Electrical Engineering. alexander.k.yee@sjsu.edu
‡Ph.D. Student, Department of Aerospace Engineering, AIAA Student Member. dantske1@illinois.edu
§M.S. Student, Department of Electrical and Computer Engineering. jundayu2@illinois.edu
¶Assistant Professor, Department of Computer Science. rmancuso@bu.edu

The key idea in this framework is the concept of "demand", defined as the smallest spatially-confined unit of activity desired by an agent, where the desired activity is application dependent. For example, burning trees demand attention when fighting forest fires, mobile device locations demand attention when providing wireless connectivity, and sharp changes in elevation demand attention when performing aerial topographical mapping. The objective of the system is to match the spatial distribution of agents to the distribution of demand that requires service (see Figure 1).

The demand-response framework consists of two algorithmic components: *Recursive Cell Decomposition*, a computationally-efficient motion-planning/task-assignment algorithm for distributed multi-agent systems, and *MARSNet*, a cluster-based routing protocol that enables agents to reliably disseminate the appropriate mix of individual and aggregate state information required for system-wide coordination.

The simulator architecture consists of a ROS/Gazebo component tailored to Recursive Cell Decomposition, and an ns-3 component tailored to MARSNet. These two components interface through FlyNetSim,[4,5] third-party middle-ware that synchronizes the time-based dynamics on the control side with the event-based dynamics on the communication side. This focus of this paper is on the mechanism that makes the simulated control and communication dynamics mutually dependent even if ROS/Gazebo and ns-3 are independently standalone. The actual implementation of Recursive Cell Decomposition in ROS and MARSNet in ns-3 will be reserved for future work.

The rest of the paper is organized as follows. Section II, provides a more detailed description of the demand-response framework, including Recursive Cell Decomposition and MARSNet, Section III explains the simulator architecture, Section IV discusses some of the preliminary work done so far, and Section V describes work that will be forthcoming.

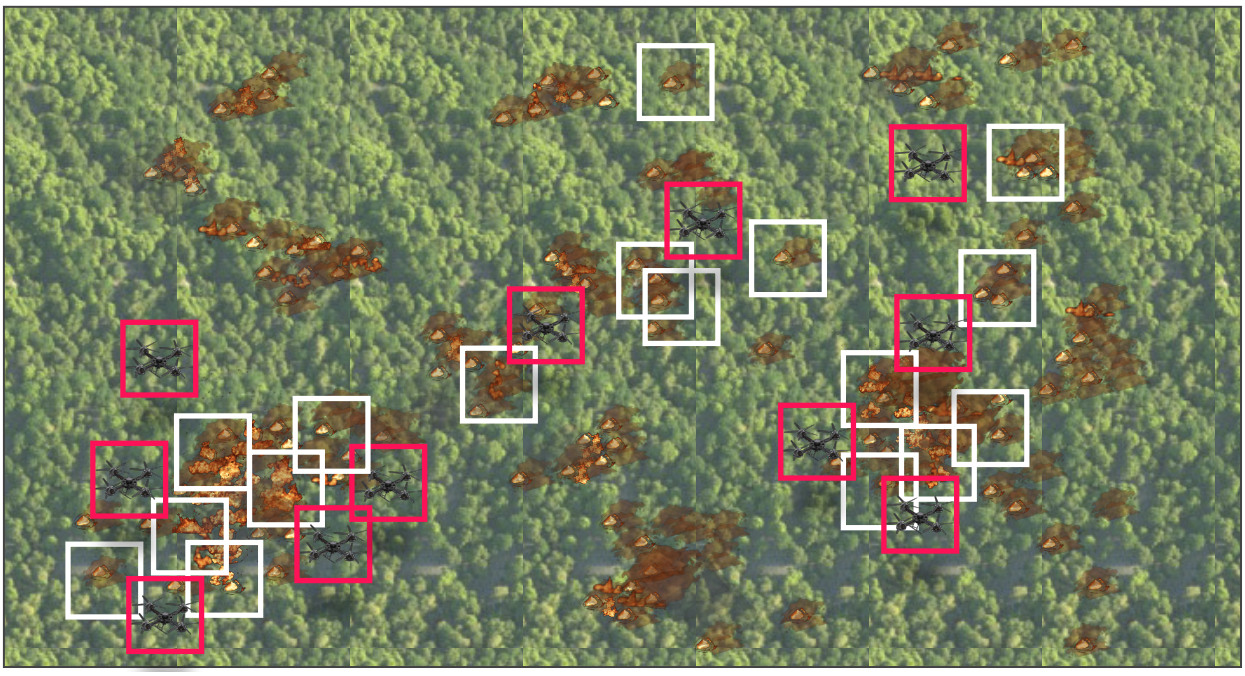

Figure 1. A distributed multi-agent UAV-based system tracking the spread of a forest fire. White boxes identify instances of "demand" (burning trees) whereas red boxes identify agents (UAVs). The system objective is to match the spatial distribution of agents to the actual distribution of demand. The agents only know the sampled distribution, that is, the burning trees that have been detected.

II. Background

The demand-response framework motivating the proposed simulator, enables decentralized control of multi-agent systems. Given an appropriate identification of demand, this framework interprets high-level instructions in terms of steering the spatial distribution of agents to the time-varying spatial distribution of demands requiring service, subject to certain feasibility constraints.

The two main components in this framework, MARSNet and Recursive Cell Decomposition, address the communication/networking problem in which agents agree on some common information about the distribution of demand, and the motion- planning/optimization problem in which agents determine safe trajectories that transport them between designated terminal points (charging stations) and the demands they must service. Both algorithms are depicted in Figure 2.

A. MARSNet

The MARSNet architecture is based on the principle of cluster stability in which agents form clusters that share relatively stable communication channels. Each cluster elects a leader responsible for facilitating intra-cluster communication. Cluster-based routing protocols in the existing literature never exceed more than one hierarchical level. Inter-cluster communication in these protocols occurs, if at all, through edge agents that straddle adjacent clusters.

By contrast, MARSNet extends the principle of stability upwards, forming a hierarchical communication graph based on relatively stable macro clusters of clusters at each hierarchical level. As a result, MARSNet creates the most

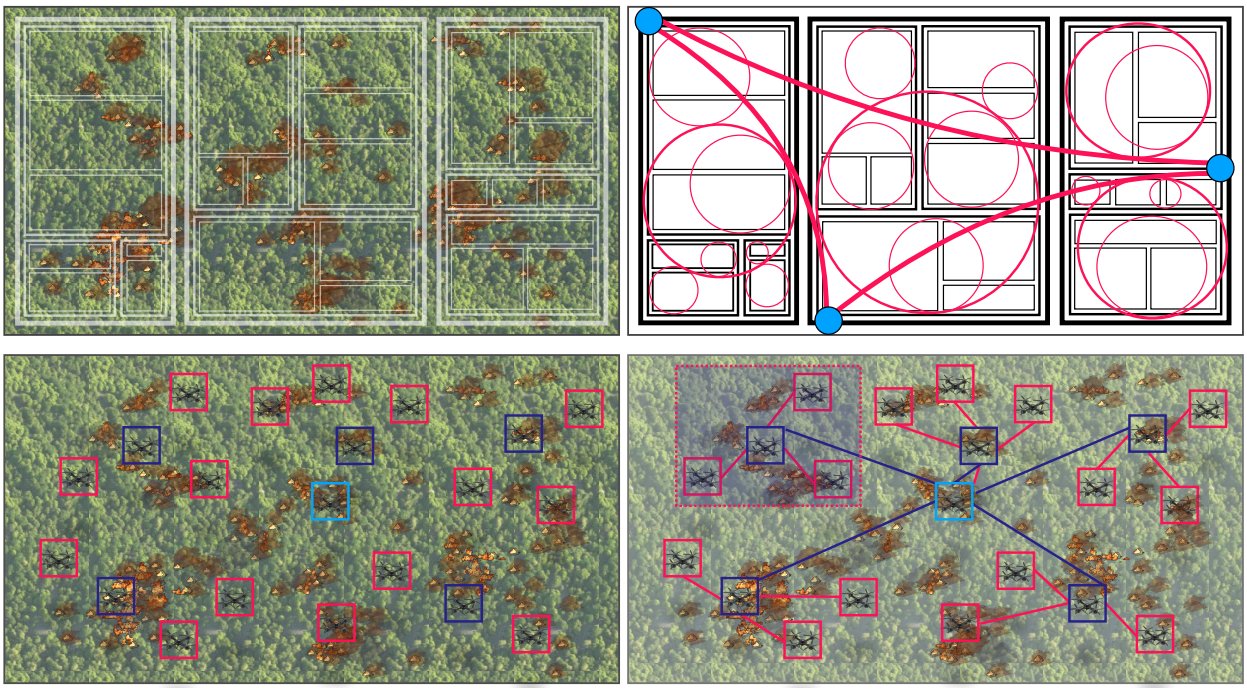

Figure 2. A Depiction of Recursive Cell Decomposition (top) and MARSNet (bottom). In the top left, the operating area is recursively divided into cells and sub-cells of equal demand. Three levels are shown with increasing cell border thickness. The cells at each level contain roughly the same amount of demand. The top right diagram depicts non-intersecting trajectories that traverse the cells and terminate at designated charging stations (in blue). The thickness of the trajectory matches the level of the cells traversed. Aggregate trajectories correspond to higher levels and transport agents across larger cells. The decomposition matches the spatial distribution of agents to the distribution of demand as depicted in Figure 1. The bottom left diagram depicts an arbitrary arrangement of UAVs over the operating area. The bottom right diagram shows a cluster-based communication network super-imposed on the UAVs. Red boxes indicate cluster members and blue boxes indicate cluster heads. Cluster heads also form a "macro-cluster" with a macro-cluster head indicated in light blue. Each cluster consists of agents that share relatively stable communication channels.

stable communication graph possible from a set of inherently unstable options. The trajectories of agents within a cluster may diverge so MARSNet requires periodic cluster updates to track relevant changes. The protocol consists of four phases: a neighbor discovery phase, a cluster discovery phase, a data transfer phase and a cluster maintenance phase. In the neighbor discovery phase, each agent broadcasts probe packets and measures the stability of the channel between itself and its neighbors. The agents use these measurements to form clusters. In the network discovery phase, cluster-heads broadcast probe packets to form macro-clusters. This process hierarchically repeats itself, terminating when the tree is complete.

In the data transfer phase, cluster-heads at all levels organize intra-cluster and inter-cluster communication. Cluster-heads schedule all communication in their clusters so that agents in the same cluster do not mutually interfere. They relay state information, such as position and velocity, between agents in their cluster. They aggregate state information and empirical demand in the cluster, and pass this information to the higher-level cluster heads. Finally, they relay to their clusters, any acquired external aggregate state, thus enabling the system to establish common information at each hierarchical level.

In the cluster maintenance phase, agents rebroadcast probe packets to measure the strength of the channel with their neighbors. These measurements may induce agents to leave their clusters and join others after proper notification. Cluster heads may also leave their cluster after releasing their responsibilities to a designated back-up agent. The protocol cycles through the data transfer phase and the cluster maintenance phase, as depicted in Figure 3, ensuring that messages are never dropped in transition while the communication graph dynamically adjusts to the motion of the agents.

MARSNet is related to the MobileIP [6] protocol widely-used in cellular networks. Both protocols support temporary routing addresses of mobile agents and maintain stable connections through hand-offs from one cell to the next. The key difference is that MobileIP does not support ad-hoc networks, but requires extensive pre-existing infrastructure between fixed densely distributed base stations. By contrast, MARSNet uses an imposed tree-structured hierarchy along with transient states to support the same capabilities in an ad-hoc setting.

B. Recursive Cell Decomposition

This algorithm, depicted in Figure 2, recursively decomposes the area of support into cells with equally distributed demand and restricts the motion-planning problem to cell traversal at each hierarchical level. This decomposition simplifies the task assignment problem from a global assignment to a hierarchical one of agent clusters to cells. All trajectories start and end at designated charging stations and the spatial distribution of demand evolves in time independently of where these stations are located.

First, the agents collectively decide on a coarse partition consisting of macro spatial-temporal cells of equal empirical demand. Given this partition, the agents compute aggregate trajectories with endpoints at charging stations, that safely traverse all macro-cells. These global trajectories transport agents en- masse to areas of significant demand, and reserve enough space-time volume to support disjoint individual local trajectories.

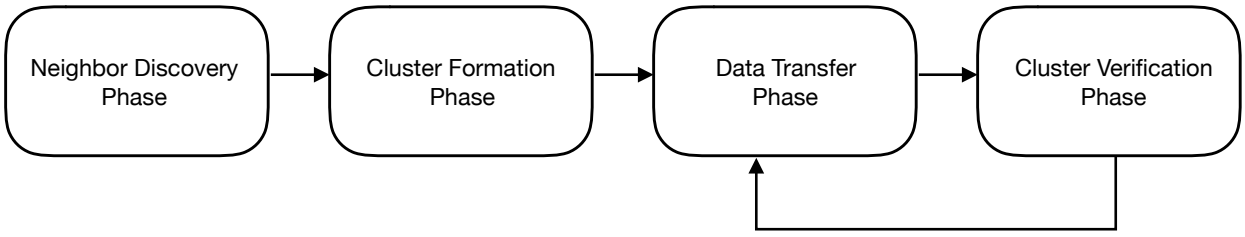

Figure 3. The MARSNet state diagram consisting of four phases: neighbor discovery phase, cluster discovery/formation phase, data transfer phase, and cluster maintenance/verification phase.

Next, agents within each macro cell collectively partition the cells into sub-cells of equal empirical demand and compute cellular-trajectories that safely traverse all the sub-cells and branch from appropriate entry and exit points on the proximal macro-trajectory. The agents repeat this decomposition, partitioning sub- cells into smaller sub cells and computing sub-cellular-trajectories that branch off the higher-volume trajectories. This decomposition terminates when the spatial distribution of agents is sufficiently close to the distribution of demand.

Every agent departing from some charging station, merges onto a macro-trajectory with other agents, travels to an assigned macro-cell, successively exits onto a sequence of lower-volume cellular trajectories until reaching the target location. After servicing the demand, the agent successively merges onto a sequence of higher-volume trajectories until returning to a charging station. This journey is analogous to that of a car traveling over residential streets, main streets, and highways during a single commute. The difference is that the trajectories of these roads evolve according to the estimated distribution of demand in real-time.

III. Simulator Design

The simulator architecture is depicted in Figure 4, and uses ns-3[7] for the ad-hoc networking dynamics, Gazebo[8] for the UAV swarm dynamics, and custom middleware to interface the two. Swarm control is implemented in ROS. Other middleware designs for interfacing network and UAV swarm simulators have previously appeared in the literature.[4,9,10] Of special relevance is the middleware[4] developed to interface ns-3 with Ardupilot.[11] This work differs in that the proposed simulator validates specific criteria that pertain exclusively to MARSNet and requires custom ns-3 modules and middleware for implementation. The cited work, on the other hand, is a more generic framework for testing existing ad-hoc routing protocols such as AODV,[12] OLSR,[13] and DSR[14] with existing ns-3 trace collection capabilities.

At its core, the proposed simulation framework emulates two core features, the management of distributed agents view of the global system state and dynamics as a result of said states. The management of states is split between the two simulator sub-components for networking and mobility. Abstractly speaking, the mobility simulation component will have a global view of multi-agent states and dynamics. Conversely, at the micro-level of each agent, each node will know its own state and through the networking component, gather a constrained local state view. Each agent then

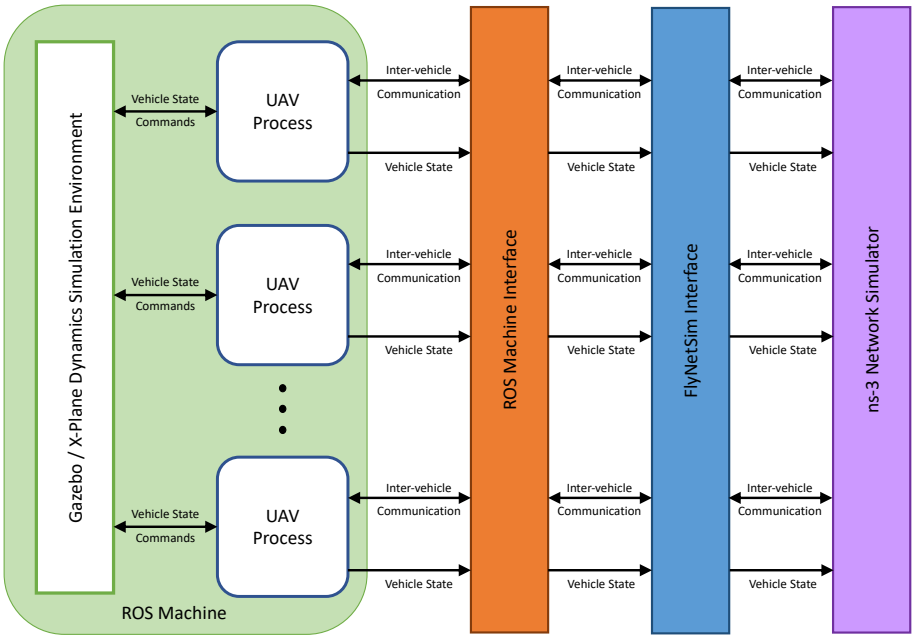

Figure 4. Simulator architecture diagram showing the ROS Machine handling dynamics simulation and the UAV processes, the communication interfaces, and the ns-3 network simulator.

uses the local view of the system to calculate an iterative state update. These responsibilities translate to a mobility component that simulates the dynamics of a multi-agent swarm and emulates distributed path planning decisions on each agent. And a networking component that is responsible for simulating the communication of state information between agents and then exposing a subset of the global state as a limited local state view to an agent to inform path planning decisions. Collectively, this behavior provides the tools to design and test our efforts towards a network assisted hybrid path planning system.

This exposition proposes the use of ns-3, ROS, and a modified FlyNetSim middleware approach to simulate our Demand-Response protocol. ns-3 will emulate the described networking component of the framework to aggregate, prioritize, and distribute state information. ROS will provide the foundation and plasticity to simulate the control, planning, and dynamics of our multi-agent UAV swarm system. Lastly, a modified FlyNetSim middleware interface will synchronize the state information known at each agent by facilitating communication between the two simulator components.

A. ns-3 Network Simulator

ns-3 is responsible for simulating the communication of state information in our Demand-Response protocol. However, there is a subtle distinction in the mechanisms for disseminating agent state information. The proposed Demand-Response protocol is not solely sending state data between agents. The communication of state information is a beneficial side-effect of cluster routing that takes a stateful approach to improve communication reliability in MANETs. Thus, to adequately model the communication of state information, we use ns-3 to simulate a hierarchical clustering framework.

In the larger context of the simulation framework, ns-3 will emulate, in isolation, the sharing of state information as part of a hierarchical clustering framework. Cluster routing protocols are different than traditional ad hoc routing methods such as, AODV, DSR, and DSDV as they use state information to construct a movement resilient ad hoc architecture. However, to do so, each simulated node in ns-3 will need state information about its corresponding self in the mobility component–ROS and Gazebo or X-plane. As show in Figure 4, ns-3 will interface with a modified FlyNetSim middleware to communicate with the ROS mobility component to retrieve state information for each corresponding node. Then, ns-3 will model in isolation the cluster communication of agents in the autonomous system. As a byproduct of the hierarchical nature of the cluster framework, the agents will associate with other agents that have similar movement patterns. These clusters of agents will then communicate through the hierarchy, prioritizing local intra-cluster and near-cluster communication over distant inter-cluster communication of states. As a result of clustering, each agent will then have a detailed local state view of its neighboring topological elements and a sparse understanding of distant ones.

ns-3 exports the local state view of each agent back to the mobility component to progress the path planning and dynamics simulation. ns-3 provides several mechanisms for exporting network gathered information to outside processes. In this simulation framework, we will use a method similar to FlyNetSim, a publish-subscribe model that interacts with virtual networking interface cards (NICs) attached to each ns-3 nodes.[4] The modified FlyNetSim middleware will interact with ns-3 through northbound and southbound publish-subscribe interfaces to send mobility state information to and extract local state subsets from ns-3. Since the mobility simulation component has global access to all states of the system, ns-3 will send back a subset of the global state that each node has access to, as seen in Figure 5. The primary responsibilities of the ns-3 networking component is to model the accurate clustering and communication of the agents. However, its global importance to the simulation framework is to manage agents local state knowledge to make informed movement decisions.

B. ROS Machine and Dynamics Simulation

For the design of our simulation architecture, we utilize ROS as a base framework for the simulation of planning and control of each UAV in the swarm, as well as the simulation of their dynamics in the high-fidelity dynamics simulation environments. Inside the ROS environment, each UAV in the swarm is represented by a UAV process node, as shown in Figure 4. The UAV process node communicates with the dynamics simulation environments via the ROS

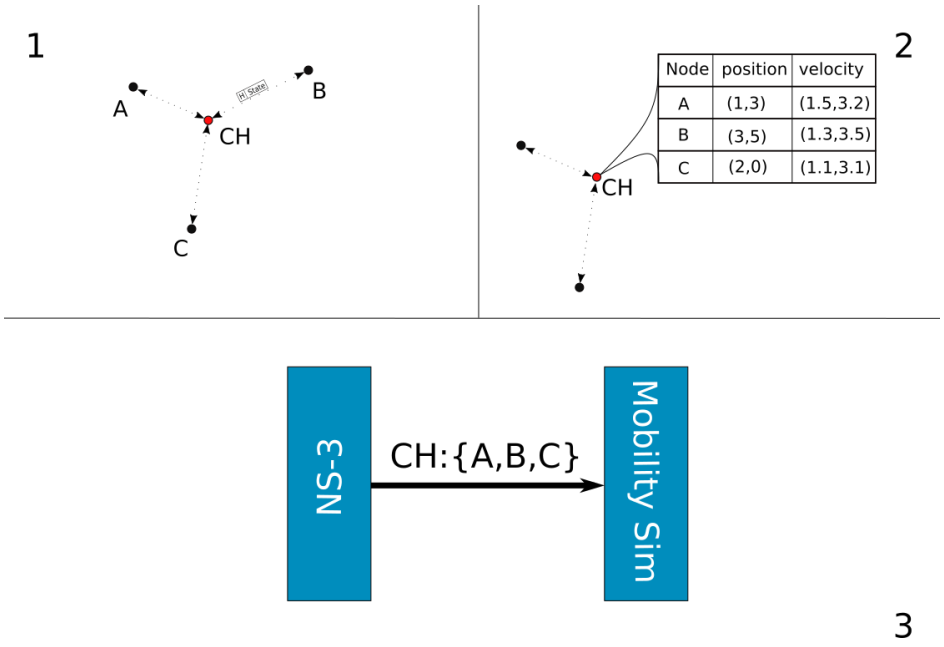

Figure 5. ns-3 state export model. 1) General cluster topology communicates states to intra-cluster neighbors. 2) Aggregated state information on Cluster Head node. Shows the communicated states of intra-cluster neighbors. 3) ns-3 exports the ID's of the states each node locally knows of due to network clustering. These collection of IDs represent the subset of agent states that a node learned from the communication framework.

publication-subscription framework, exchanging the control commands and the vehicle state information between each other and thus enabling the corresponding UAV process to control the simulated aircraft. The Gazebo dynamics simulation environment is integrated and readily available inside the ROS environment. Furthermore, other external simulation programs with high-fidelity dynamics models such as the X-Plane flight simulator[15] can also be integrated into the simulation architecture via a ROS machine interface which performs the conversion between the ROS messages and the custom data primitives of the X-Plane development library.

For each of the UAV process nodes, we intend to implement a typical planning and control stack which consists of a global mission planner, a local planner, and a controller. The global mission planner is responsible for generating a global trajectory for the aircraft. The local planner receives the global trajectories and outputs low-level targets for the controllers. And the controllers receive the local planner targets and tries to bring the current vehicle state to the targets by generating outputs to the vehicle control surfaces. In our simulation architecture, all the UAV processes receive the same copy of a global plan from either a base station or a lead UAV process. Then, each UAV process creates its own local plan based on the received global plan as well as the state of other nearby aircraft given by the ns-3 via the inter-vehicle communication. For example, if a global plan were to instruct the UAV swarm to fly to a specific spatial point, each UAV in the swarm cannot simply fly to the exact location as specified by the global plan without colliding into each other. As a result, the UAVs create a local plan for themselves based on the given global plan and the state of other UAVs such that the aircraft follows the general direction towards the global plan while keeping safe distances among each other.

C. Interface Design

To provide a simulated environment of the MARSNet protocol, we use the ns-3 network simulator for the simulation of the communication among the UAVs in the swarm. Each UAV process in the ROS machine provides its own vehicle states, or an aggregated vehicle states among a local group of UAVs, to the ns-3 network simulator. And the information about the states of nearby UAVs can be retrieved by each UAV from the ns-3 via the inter-vehicle communication. To achieve such communication setup, we use FlyNetSim[4] shown in Figure 4 as a communication interface between the

ns-3 network simulation and the ROS machine containing the UAV processes and the dynamics simulation. FlyNetSim, however, is originally designed to interface the ns-3 with the ArduPilot. As a result, a ROS machine interface, similar to the one required for the X-Plane integration discussed above, is needed for the conversion between the ROS messages and the external data types used by the FlyNetSim.

IV. Preliminary Work

Our initial attempt to implement the proposed simulation framework is limited to a local clustering application in ns-3. As clustering represents the core feature that enables us to enhance our path planning solution, we started with a local clustering algorithm based on SCalE.[16] The implementation models local associations of agents in the system that have similar movement patterns. The development of the clustering algorithm illuminated the need for a modular and sophisticated simulation framework. It informed our design to simulate communication between nodes in ns-3 and delegate the control, path planning, and dynamics to a mobility simulator.

V. Summary and Future Work

This paper proposes a mutli-simulator design to test and validate a network assisted path planning algorithm. As this solution is cross-disciplinary by nature, weve taken a multi-simulator approach to encapsulate simulation tasks to their respective domains. The proposed design uses ns-3 to model the aggregation, prioritization, and dissemination of state information between mobile agents as a result of clustering. Through a combination of ROS and its extensions such as, Gazebo or X-plane, the proposed simulation framework models the control, path planning, and dynamics of the multi-agent system. Lastly, the proposed framework takes a multi-interface middleware approach to synchronize agent knowledge across the multi-simulator framework.

Future work will be cross disciplinary and modular to reflect the complexity and modular nature of the proposed simulation framework. On the networking side, we plan to complete the other two main components of our hierarchical clustering algorithm in ns-3. Conversely, on the control side, we plan to clearly define the multi-agent scenario of interest in a ROS based simulation environment like Gazebo. Lastly, coordination of the required data format for agent state synchronization will tie the two components of the simulator into a holistic path planning simulation framework.

References

[1]nsnam, "ns-3 — a discrete-event network simulator for internet systems," https://www.nsnam.org/, Accessed May 2019.

[2]Open Source Robotics Foundation, "ROS — Powering the world's robots," https://www.ros.org/, Accessed May 2019.

[3]Open Source Robotics Foundation, "Gazebo," http://gazebosim.org/, Accessed May 2019.

[4]Baidya, S., Shaikh, Z., and Levorato, M., "FlyNetSim: An Open Source Synchronized UAV Network Simulator Based on ns-3 and Ardupilot," *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '18, ACM, New York, NY, USA, 2018, pp. 37–45.

[5]Kudelski, M., Gambardella, L. M., and Di Caro, G. A., "RoboNetSim: An integrated framework for multi-robot and network simulation," *Robotics and Autonomous Systems*, Vol. 61, No. 5, 2013, pp. 483–496.

[6]Perkins, C., "IP Mobility Support for IPv4, Revised," RFC 5944, RFC Editor, November 2010.

[7]Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y., and Yu, H., "Advances in Network Simulation," *IEEE Computer*, Vol. 33, No. 5, May 2000, pp. 59–67, Expanded version available as USC TR 99-702b at http://www.isi.edu/

[8]Koenig, N. and Howard, A., "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Vol. 3, Sept 2004, pp. 2149–2154 vol.3.

[9]Marconato, E., Rodrigues, M., Pires, R., Pigatto, D., Querino Filho, L., Pinto, A., and Castelo Branco, K., "AVENS A Novel Flying Ad Hoc Network Simulator with Automatic Code Generation for Unmanned Aircraft System," 01 2017.

[10]Zema, N. R., Trotta, A., Sanahuja, G., Natalizio, E., Felice, M. D., and Bononi, L., "CUSCUS: An integrated simulation architecture for distributed networked control systems," *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2017, pp. 287–292.

[11]"Ardupilot Autopilot suite," http://ardupilot.com, 2016.

[12]Perkins, C., "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, RFC Editor, July 2003.

[13]Jacquet, P., "Optimized Link State Routing Protocol (OLSR)," RFC 3561, RFC Editor, October 2003.

[14]Johnson, D. B., Maltz, D. A., and Broch, J., "Ad Hoc Networking," chap. DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001, pp. 139–172.

[15]Laminar Research, "X-Plane 11," http://www.x-plane.com/, Accessed May 2019.

[16]Rossi, G. V., Fan, Z., Chin, W. H., and Leung, K. K., "Stable Clustering for Ad-Hoc Vehicle Networking," *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2017, pp. 1–6.