BotNet: A Simulator for Studying the Effects of Accurate Communication Models on Multi-agent and Swarm Control

Mark Selden*, Jason Zhou*, Felipe Campos*, Nathan Lambert*, Daniel Drew‡, Kristofer S. J. Pister*

***Abstract*— Decentralized control in multi-robot systems is dependent on accurate and reliable communication between agents. Important communication factors, such as latency and packet delivery ratio, are strong functions of the number of agents in the network. Findings from studies of mobile and high node-count radio-frequency (RF) mesh networks have only been transferred to the domain of multi-robot systems to a limited extent, and typical multi-agent robotic simulators often depend on simple propagation models that do not reflect the behavior of realistic RF networks. In this paper, we present a new open source swarm robotics simulator, BotNet, with an embedded standards-compliant time-synchronized channel hopping (6TiSCH) RF mesh network simulator. Using this simulator we show how more accurate communications models can limit even simple multi-robot control tasks such as flocking and formation control, with agent counts ranging from 10 up to 2500 agents. The experimental results are used to motivate changes to the inter-robot communication propagation models and other networking components currently used in practice in order to bridge the sim-to-real gap.**

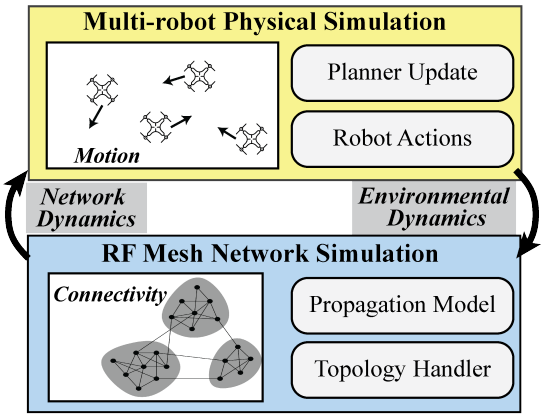


Fig. 1: System diagram for *BotNet*, a simulator for studying the effects of accurate communications models on multi-agent control. The simulator exchanges information between a standards compliant networking simulator with a modular, lightweight robotics simulator.

I. Introduction

Multi-agent systems are becoming more prevalent and interconnected with the emergence of low-cost radios and capable robots [1], [2]. As agent count increases, the classification of the research domain transitions from that of multi-robot systems, up to about 10 agents, to one of swarm robotics, with 10s, 100s, or even 1000s of agents [3]. Historically, the most successful method for control with lower agent counts has been centralized planning and optimization, with actions sent to all agents simultaneously [4], [5]. Imperfect sharing of information between agents in widely distributed systems and the significant computation requirements for planning with high agent counts are two major barriers to centralized control approaches at the swarm level. Decentralized control, where decisions are made locally per-agent by communicating with one's neighbors [6], is a common solution because it is more readily scalable to higher agent counts. The success of decentralized control, however, relies on the communication between an agent and all of its neighbors, the success of which in turn depends heavily on the number and density of agents, data transfer rate, and communication network structure – elements that are not typically studied at the scale of swarms nor with the dynamical complexity of mobile robots. In this work, we introduce a tool, BotNet, for studying the effects of local radio frequency (RF) communication on multi-agent control, and use it to demonstrate how the challenges of realistic communication can manifest during example tasks.

RF communication is a popular solution within the realm of decentralized control due to its relatively high information transmission rate, few restrictions on environment, existing commercially-available communication hardware, and more [7]. RF networks have been studied extensively, especially in the context of stationary wireless sensor networks (WSNs) [8], but directly translating findings for use in mobile systems has proven difficult. Communication has proven to be a limiting factor for deployment of novel multi-agent controllers, as the dynamics of mesh networks can be difficult to predict and change rapidly [9], [10], [11]. Challenges to network reliability can come from environmental disturbances, such as fixed or mobile obstacles, task related challenges, such as large coverage areas, and from network-specific limitations, such as transmission collisions. For example, in the recent DARPA Subterranean Challenge [12], RF communications were extremely unreliable in underground environments. In disaster relief scenarios, the ad hoc wireless networks established by first responders can quickly become saturated and lead to decreased data throughput from teleoperated and semi-autonomous robots [9].

Owing to the complexities of RF communications, many robotics simulators do not include the option to evaluate control with realistic communications models. The most frequently used models for peer-to-peer links – including line-of-sight, nearest neighbor communication, and perfect

Corresponding author: Nathan Lambert, nol@berkeley.edu.
*Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA.
‡Department of Electrical and Computer Engineering, University of Utah, UT, USA.

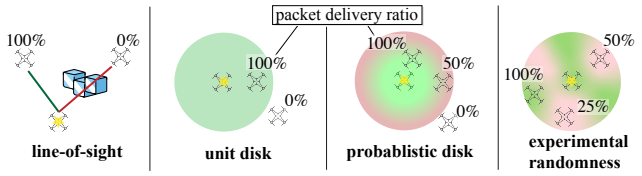


Fig. 2: Different propagation models used in wireless sensor network and multi-agent control simulators. From left to right: *line-of-sight* models allows agents to connect within their field of vision; *unit disk* models connect with any agent within a radius r; *probabilistic disk* models reduce RSSI uniformly as distance increases from communicator; and *experimental randomness* models are derived from logging RF connections in real-world environments. The experimental randomness model shows that there is a much weaker correlation between position and connection than is often assumed.

communication within a certain radius – can limit effective translation to the real world. Experiments show that many RF networks can be realistically described by extending the Friis free-space model (transmission power is inversely proportional to radius) to an *experimentally random* model for RF Received Signal Strength Indicator (RSSI) due to the complicated dynamics of a given environment [13], [14], [15]. Accurately determining who can communicate with whom is only one of the many parts required for translating simulated multi-agent controllers to systems with real RF communication. Integration between the controller design and the dynamics of information routing must be considered in order to determine how precisely each agent understands the intents of its neighbors and each agent in the swarm understands progress towards the group's task. An accurate simulation of the network is important; if the simulated network is too optimistic, the control tuning will not necessarily translate to the real world. Conversely, if the simulation is pessimistic and the controller is tuned to accommodate a poorly performing network, it can result in sub-optimal behavior such as slow or inefficient task performance.

In this work we examine the effects that simulating accurate communications has on canonical swarm robotics tasks. We detail the construction and use of BotNet (framework shown in Fig. 1), an extensible open source swarm robotics simulator with an embedded RF networking simulator, which is scalable to 1000s of agents with detailed logging and visualization. Our experiments show that in decentralized formation control and flocking, varying the assumptions on RF communication (via the propagation model and scheduling functions) can make conceptually simple tasks difficult to reliably perform.

II. Background

In this section we detail the background needed for understanding the interplay of network dynamics with multi-agent systems.

Wireless sensor networks (WSNs) are connections between many individual radios, referred to as *motes* or *nodes*. Two

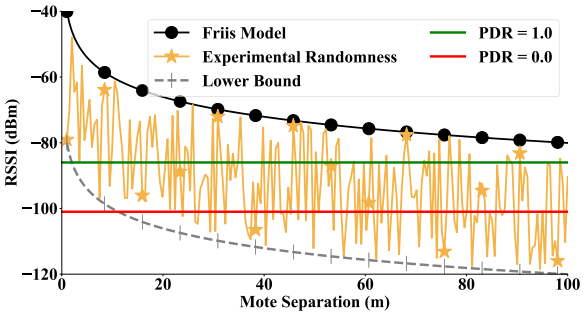


Fig. 3: Received signal strength from a 0dBm transmitter as a function of different propagation models. The Friis free space transmission loss model is an optimistic best case for unobstructed transmission. In reality, environmental factors lead to significant deviation from this ideal. The *experimental randomness* model, which is based on empirical data, can be used to approximate the behavior of signals in real world environments. The lower bound of this experimental data, approximately the Friis model prediction minus 40dBm, can be used as a conservative estimate. The empirical model, however, shows that the true signal strength will often be better than this, providing opportunity for more optimistic control patterns[1].

motes have a Packet Delivery Ratio (PDR), denoting the number of received packets out of the total sent, which is often heavily influenced by the Received Signal Strength Indicator (RSSI). In simulated networks, the relationship between physical distance and RSSI or PDR is called a *propagation model*. The algorithm controlling which mote is speaking at a given time-step is called a *schedule function*. For a more detailed review of sensor network behavior, see [13].

A. RF Propagation Models

Determining the RSSI between network nodes is crucial to accurately simulating communication, as it is the primary driver of PDR. Using theoretical abstractions for predicted signal strength in simulators is necessary because RF signal propagation is extremely difficult to model. Different propagation models are visualized in Fig. 2: line-of-sight sets PDR to be 1 for any agents with an unobstructed visual link, a unit disk delivers all packets within a radius r, and a probabilistic disk creates a lower bound on communication radius by matching the lower bound of free path loss. Historically, the RSSI, which underpins propagation models, is very hard to model due to the complexities of electromagnetic propagation, but recent work has shown the accuracy of an *experimental randomness* model [13], [14], [15], shown in Fig. 3. In this work we show the effects of different unit disk radii, but focus on a radius of $r = 10$m because that is near to where the PDR hits 0 for the lower bound in Fig. 3. Changing

[1]The connectivity traces were obtained from the `http://wsn.berkeley.edu/connectivity/project` at the University of California, Berkeley. The data set used is "soda," created by Jorge Ortiz and Prof. David Culler.

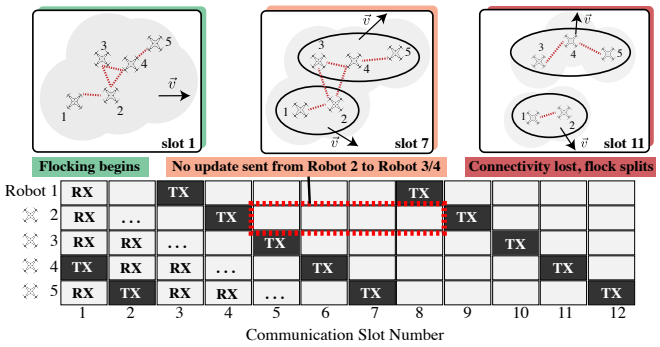


Fig. 4: Suboptimal schedule functions rapidly deteriorate swarm performance. Below is a round robin scheduling function (RRSF) grid for per-agent transmit or receive behavior, where agents broadcast position to all agents within range. Due to the asynchronicity of networked communication, a delay in broadcasting can result in the position of the important agents not being updated in a crucial timewindow.

communications hardware (e.g., to higher power transmitters or more sensitive receivers) would correlate to a vertical shift of the RSSI at a given distance.

B. 6TiSCH Networks

In this work we focus on 6TiSCH, a standards-compliant RF mesh networking protocol which is designed to be low-power, demand minimal computation overhead, and able to integrate with existing internet services. Together, these features make it well-suited for use in high agent-count, low-cost autonomous systems. 6TiSCH combines Time Synchronized Channel Hopping (TSCH) with Internet Protocol version 6 (IPv6) [13]. Time Synchronized Channel Hopping (TSCH) changes when and which channels are used in wireless communications to provide more reliable communications. A group of channel options combine to form a slotframe, where a slot is a fixed time width interval at a specified channel frequency.

C. Network Scheduling Functions

The control mechanism for who transmits (*TX*) and receives (*RX*) during each slot in a slotframe is called the schedule function. This schedule is necessary because multiple motes transmitting on the same channel, or a mote designated to transmit and receive simultaneously, regularly results in dropped packets. When starting a task, the scheduling function determines the initial communications network structure as peers negotiate who will speak when. Different scheduling functions have substantial variance in the amounts of time to form and the resulting connection structures. How best to form a network is an open avenue of research investigation beyond the scope of this paper, and such structure changes can be very important for the downstream task.

The 6TiSCH network protocol was designed with a Minimal Scheduling Function (MSF) to negotiate connections over time with stationary nodes. Our experiments show limitations with using MSF for communication between mobile nodes, namely because MSF is designed to slowly converge to a slotframe schedule for static nodes based on negotiations, transmission collisions, and PDR measurements over several slotframes. For mobile nodes using this scheduling function, communication links will frequently break down due to rapid position changes of agents. This issue becomes further exacerbated when scaling to larger numbers of agents because there are only a finite number of slots where any given agent can update its neighbors with its current position data.

To improve on network stability with the MSF, we have implemented a round robin scheduler function (RRSF), shown in the bottom of Fig. 4, where every new slot schedules one mote to transmit while all other motes are in receive mode, cycling through each mote over the course of a slotframe. Compared to this new RRSF, the MSF is slower to converge to an initial network topology because it entails peer-to-peer negotiation over transmit-receive slots, and re-negotiations make it substantially less stable to the changes in network architecture which arise with mobile WSNs. Even with RRSF, challenges in multi-agent control can arise when crucial agents do not send a packet, resulting in link loss; this challenge of communications in swarm control is shown in Fig. 4.

III. Related Works

A. Mobility in Wireless Networks

A typical wireless sensor network (WSN) is designed and optimized as a stationary system. In networks composed of mobile nodes, many events can result directly from physical movement, including connection topology changes, removing or adding of network members, unexpected node failures, signal strength change, and more [16], [17]. Methods to improve sensor network performance in the presence of mobile nodes come at many layers of the communication stack, but primarily can be characterized at the Medium Access Control (MAC) layer – who communicates when – and at the network layer when routing – determining how to distribute information over multiple steps through the network. Solutions at both of these layers are highly application-specific and difficult to design and apply to the general case. For example, tools at the MAC layer optimize for mobility by being selective about which packets are sent, for example by using tools for avoiding packet collisions [18], [19]. Scheduling functions are used to control for transmission timings and are also crucial for maintaining a network during movement, so they can also be tuned for mobility, such as efficient scheduling with delay constraints [20]. Multi-hop routing for mobile WSNs is an open area of research, with solutions investigating moving nodes to pass information to another area of the network [21], but little work has been performed optimizing specifically for multi-robot control tasks.

Network node mobility can also be used in a complimentary optimization problem, for example by incorporating network maintenance and health directly into robotic task planning. Route Swarm [22] adds a second optimization to a coordination problem in order to maximize information flow across a

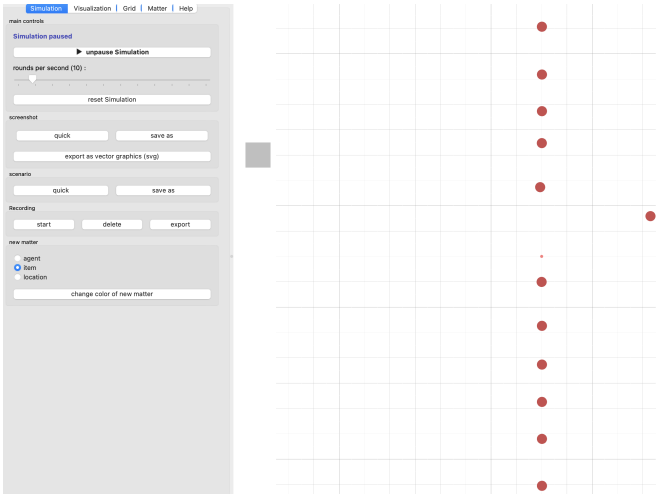
network. Mobile nodes can also be used to ferry messages between stationary nodes [23].

B. Communication in Multi-agent Robotics

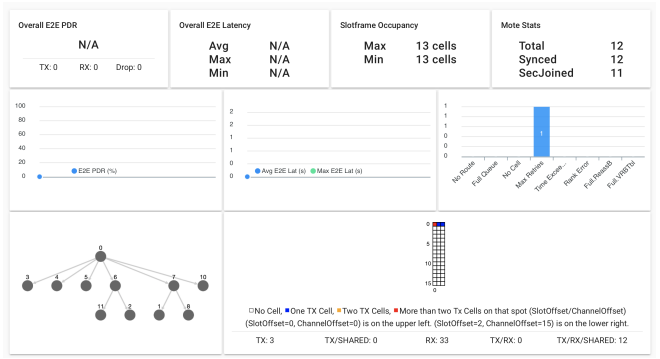
Groups of robots can work together and accomplish tasks unrealistic for a single unit, often by leveraging the information shared by their neighbors [24]. The potential of peer-to-peer communication is highlighted by progress in decentralized control, but recent work has incorporated only a limited study of realistic propagation models, schedule functions, multi-hop behavior, and other real-world aspects of complex networks. The simplest network abstractions used in robotics studies focus on swarms as a graph structure, using a model where the nearest neighbors can communicate fully [25], [26]. Some work uses line-of-sight (LOS) instead of purely geometric distance in order to determine which agents can communicate [27], [28], but LOS is not an accurate model for RF networks. The most common approach to communications may be a radius of connectivity model, where all neighbors within a radius can communicate [29], [30], [31], [32], but this assumption uses inconsistent communication radii rather than information inspired by real systems. More advanced communications models used in the literature include a Gaussian well model, where RSSI and PDR decrease with increasing radius, but examples incorporating these remain rare [33]. BotNet provides a tool that allows for simple and modular manipulation of the propagation models to show how the aforementioned communications assumptions change behavior of multi-agent tasks. The different propagation models common in the robotics literature and studied with our framework are shown in Fig. 2.

C. Simulating Networked Swarm Robotics

Accurately simulating all aspects of a robotic swarm, including the network, robots, and environment, is a challenging task. Many tools for multi-agent simulation specialize at one function – such as Gazebo [34] or AirSim [35] for accurate robot dynamics, SwarmSim [36], SwarmLab [37], or Scrimmage [38] for swarm control, and the OpenWSN emulator [39] or the 6TiSCH Simulator [36] for realistic communications models. There has been relatively little development of simulation tools at the intersection of robotic control and accurate network modelling. RoboNetSim [40] and ROS-NetSim [41] are two frameworks designed to mediate external simulators and synchronize data for accurate communications in robotics. RoboNetSim acts as an integration tool between any two robotics and networking simulators. For example, RoboNetSim synchronizes time and robot position between Network Simulator 2 (NS-2) [42] and ARGoS [43], a popular multi-agent simulator. RoboNetSim is no longer actively supported. ROS-NetSim performs a similar function to RoboNetSim by creating a central Robot Operating System (ROS [44]) node between the network and robotic simulators. ROS-NetSim has promise for improving sim-to-real transfer because many real systems are deployed on ROS [44], but is not designed to handle high agent counts nor standards compliant communication models, which is



(a) Robot dynamics visualizer.



(b) 6TiSCH network visualizer.

Fig. 5: BotNet is configurable to visualize continuous network and/or robotic dynamics. A core component of the network visualization, the connection graph, is shown in the bottom left. Lopsided network configurations can make a swarm sensitive to dropping members from the network during complicated multi-agent tasks. For more details on the robotic simulation, see [36], and for details on simulating 6TiSCH networks, see [13].

helpful for transitioning to real WSNs. Our simulator, BotNet, allows rapid experimentation and modularization by having built-in network and robotic simulation, is written entirely in Python for ease of development, and is scalable to 1000s of agents.

IV. BotNet: Swarms with 6TiSCH Networks

In this section we describe the open source simulator we have released, BotNet, which is suitable for studying large numbers of mobile networked whileagents with realistic communications models. The simulator, visualized in Fig. 1, uses the environment and agent abstractions from SwarmSim [36] and simultaneously integrates a IEEE802.15.4 6TiSCH simulator [13]. In building the simulation framework, we define an abstract class through which 6TiSCH can interface with a multi-agent robotic simulator using remote procedure calling (RPC). This allows for the transfer of the minimal set of necessary data to guarantee time and state

synchronization between simulators. It can be extended to transfer any type of useful information between the network and physical layer simulators, for example allowing for robotic and network routing control algorithms to depend on one another, network failure information being exchanged to take appropriate actions at the robotic controller level, and for physical environment information to be communicated to the network simulator to inform propagation modelling. The code for the simulator and experiments can be found at `https://github.com/PisterLab/BotNet`.

A. Design

Building on SwarmSim: The dynamics simulation for BotNet extends SwarmSim's minimal infrastructure [36]. To capture more realistic dynamics, we created an agent structure `VeloAgent` that has continuous velocity-controlled dynamics instead of the SwarmSim default with only positional dynamics. This extension can be abstracted further to accommodate more realistic robotic simulations. We modified the `World` class to pass arguments into the simulator environment for configurable experimentation. Additionally, we fixed multiple bugs preventing out-of-the-box examples in SwarmSim from running.

World Dynamics: BotNet has a world consisting of two portions: network and environment dynamics. The network dynamics dictate which agents can communicate and what information is passed between them. For example, with the experimental randomness propagation model, when the robot simulator moves an agent in space, the propagation model updates its RSSI distribution per each mote-to-mote link as a function of separation to determine the network dynamics. The communications dynamics can be expanded to study how data flows through multi-hop networks. The environment dynamics controls how each agent moves through space, and adds disturbances or constraints on motion. These can be modified by further instantiating the class `Scenario` (inherited from SwarmSim), such as for when the robotic task is heavily dependant on the environment (e.g., indoor space exploration). Finally, additional varieties of robot dynamics can be encoded by adding subclasses of `Agent`.

Controllers: We expand the original SwarmSim architecture of a controller class called `Solution` [36]. Agents can be controlled using both the global environmental information and via their internal belief states, for example of who their nearest neighbors are based on networking data. Alternatively, both the networking simulator and the environmental simulator can send controls directly to agents through their RPC endpoints in order to study event-driven control.

Communications: The communications stack is designed to study both low-level RF dynamics (who can communicate with whom) and multi-hop network behavior (how to route packets through the swarm). This paper focuses on the peer-to-peer communication problem, dictating to each agent which neighbors it has successfully transferred data with in a given slotframe, but the 6TiSCH network simulator is designed to be able to study multi-hop packet routing. The low-level encoding of bits into packets and maintaining IPv6 standards are handled entirely within the 6TiSCH simulator. The communications stack can interface with controller design in multiple ways. For example, included in BotNet are different options for when controls are updated with respect to communication – including after each packet, after each slotframe, or at a user-specified fixed slot-frequency. For high agent-counts, network initialization is intractable with simple scheduling functions (e.g., due to frequency of packet collisions), so experiments can also be run only evaluating the difference between propagation models.

B. Usage

Experimentation: The publicly available BotNet repository includes many environment scenarios and control solutions included at launch to foster use in multiple application domains of multi-agent systems. The included scenarios and solutions for flocking and formation control are easily used with a different agent, world, or controller: all environments and controllers are modular and changed via a configuration file. A quickstart guide and instructions for running batched experiments are included in the repository.

Visualization: The 6TiSCH network visualizer runs in the browser and dynamically shows which agents are connected, the schedule matrix, packet delivery over time, and more [13]. SwarmSim includes a OpenGL based video rendering tool to show the physical dynamics in 2- or 3-dimensions [36]. With the RPC server, we can synchronize and visualize these two diagnostic tools simultaneously, as shown in Fig. 5. Currently, only the dynamics visualization can be saved and exported as a video, but all network and dynamic data are saved individually after each experiment as compressed arrays in `.dat` files (the frequency of saving and file type is configurable). Future work will address simplifying and merging this logging system.

Scaling & Performance: BotNet is capable of scaling to 100s of agents for rapid iteration on experiments with different control tasks, communication models, and robots. The simulator update period per dynamics step is shown for different numbers of agents in Tab. I. To show the challenge of network formation, we have shown the number of timesteps it takes for the network to converge and indicate which configurations do not converge with the 6TiSCH simulator. For experiments where the network does not converge within a reasonable time, we have included the per-agent simulator scaling when running the dynamics engine with only propagation models (assuming all agents can send and receive packets simultaneously). Performance scaling per-step degrades dramatically when using the real-time visualization tool.

V. Results

We have quantified and compared the performance of multi-agent flocking and formation control over a set of different communication paradigms (full connectivity, unit and probabilistic disks, and the experimental randomness model). The agents used for our experiments have simple point-mass dynamics with a maximum velocity of $3.0 \times 10^1 \, \mathrm{m\,s^{-1}}$. All

Agent Count	RRSF Network Formation (Simulation Timesteps)	Full Network Simulation Speed (Clock Time Per-step, ms)	Propagation Model Simulation Speed (Clock Time Per-step, ms)
5	141.4 ± 21.8	6.255	8.5×10^{-1}
10	187.8 ± 71.4	2.145×10^{1}	1.6
25	418.8 ± 488.2	1.272×10^{2}	4.1
50	2485.4 ± 2221.4	4.955×10^{2}	8.3
100			1.79×10^{1}
250			5.52×10^{1}
500			1.538×10^{2}
1000			5.603×10^{2}
2500			2.273×10^{3}

TABLE I: BotNet performance characteristics. The mean and standard deviation of steps reported for network formation are shown to highlight the variance of network formation with a set scheduling function. 10 trials per configuration were executed with agents initialized in a line with $2\,\mathrm{m}$ spacing. Due to the limitations of schedule functions with a full network simulator, high agent count (>100) networks failed to converge with the full network simulator and the RRSF regardless of propagation model. Columns two and three indicate the mean period per dynamics step when running different numbers of agents (the standard deviations of clock-times per step are extremely low and omitted). The Full Network Simulation Speed is when using the full 6TiSCH simulator, but this does not scale well to high agent counts due to the challenges of scheduling function design. To accommodate high agent counts, the simulator can instead run with a limited communications module that can vary the propagation function ("Propagation Model Simulation Speed"), which is where experiments can scale to 1000s of agents. These experiments are run with a 2.3 GHz 8-Core 9th-generation Intel Core i9.

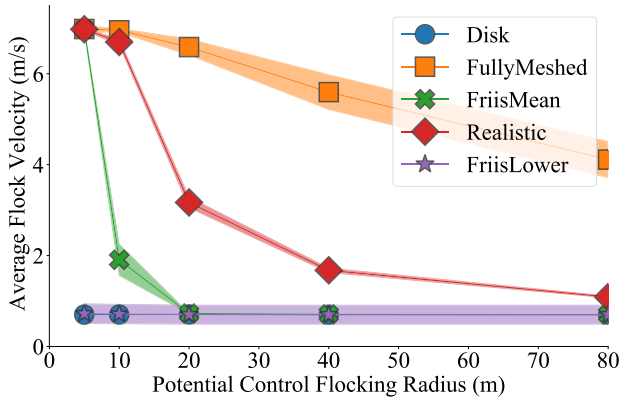


Fig. 6: Median and standard deviation flocking velocity for different potential control connectivity radii (10 trials per configuration, 10 agents per trial, RRSF). Average velocities close to zero generally indicate that the flock fails to follow the leader agent. The results show that using more realistic propagation models in the networking simulation stack sets bounds on the potential control flocking radius; at higher radii, agents lose connection from the leader and potentially their neighbors (depending on collision radius).

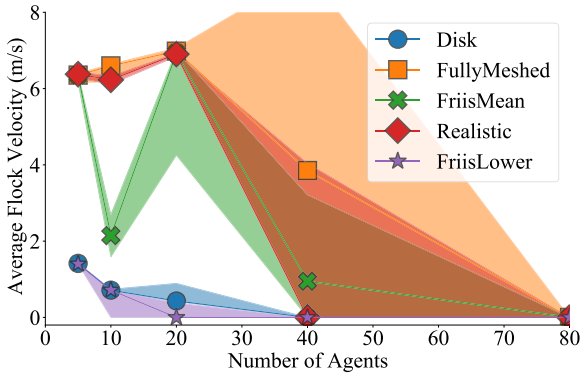


Fig. 7: Flock speed (median with maximum and minimum across trials) versus number of agents with ideal vs real communications models (10 trials per configuration, $r_{flock} =$ 10m, RRSF). There is a complicated relationship between number of agents and agent separation that results in a large variance in potential outcomes of flocking. A minimum of 0 is a trial where the agents failed to form a stable network, prohibiting flocking from starting.

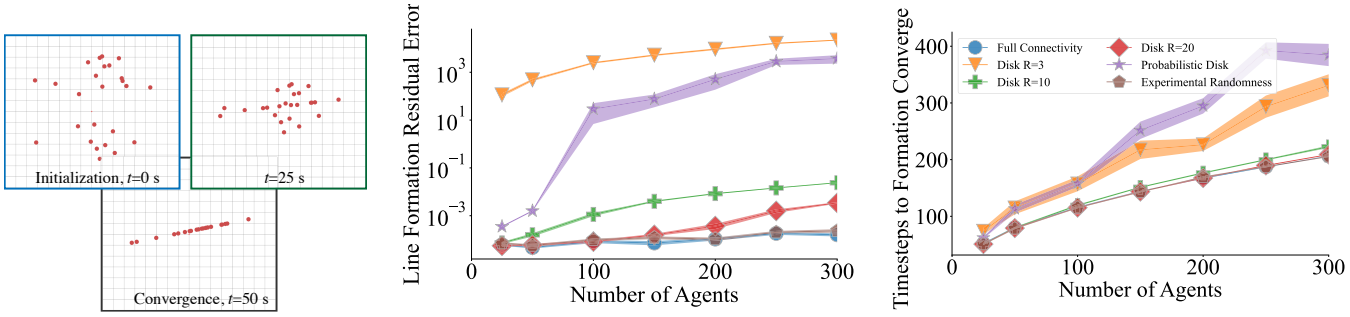
experiments use a round robin scheduling function (RRSF). The reported performances are averaged over the timesteps after networks are formed, which varies with different scheduling functions, propagation models, and control tasks.

A. *Flocking*

Flocking is a common task in decentralized multi-agent control where members of the group must co-align with their neighbors and therefore maximize the average velocity in a desired direction. In this section, we show how the task can fail, either via the flock failing to form or moving with very low velocities, due to imperfect communications and the resulting delayed or erroneous control updates. Given a potential function, V_{ij}, valuing a set distance between any two agents, flocking can be formulated as an optimization problem over the controls of each agent u_i, given the velocity $\vec{v}_i$ and position $\vec{x}_i$ of each agent [45]:

$$u_i = -\sum_{j \neq i} (\vec{v}_i - \vec{v}_j) - \sum_{j \neq i} \nabla_{\vec{x}_i} V_{ij} \tag{1}$$

$$V_{ij} = \frac{1}{\|\vec{x}_i - \vec{x}_j\|_2^2} + \frac{1}{r_{\text{flock}}^2 - \|\vec{x}_i - \vec{x}_j\|_2^2} \tag{2}$$



(a) Example with 25 agents. (b) Log-scale error from the desired formation. (c) Time until convergence.

Fig. 8: Example visualization and quantitative performance of decentralized line formation control with different communications models and number of agents (16 trials per configuration and using propagation model only to avoid the intractability high-agent count network formation; median and standard deviation are shown). (a) three time-snapshots from an example run of the line-formation control task with BotNet's built in visualizer. (b) the magnitude of error with different communications models during line formation. This error arises when sub-groups of agents form multiple lines due to dropped communication with their neighbors. (c) the time to completion of the task, *i.e.* when no agent is off their projected target at a given time, varies more closely with the number of agents. Full connectivity in this example represents the minimum time to completion and error for the randomly initialized agents, and the experimental randomness model maintains performance at high agent counts.

The first term is for directional alignment and the second term minimizes collisions (outward pressure) and maintains network connectivity (inward pressure). In our simulations with robots capable of high velocities, the discrete control update often reached a singularity at the flocking radius in Eq. (2) when agent separation increased beyond the desired r^2_{flock} before a position update, leading to flock divergence (even if connected). For this paper, we use a modified, singularity-free flocking value function V_{ij} to enable flocking with discretized control inputs with varying network connectivity:

$$\widetilde{V}_{ij} = \sum_j K_{\text{col.}} e^{-\frac{\|\vec{x}_i - \vec{x}_j\|_2^2}{r^2_{\text{collision}}}} + K_{\text{conn.}} e^{\frac{\|\vec{x}_i - \vec{x}_j\|_2^2}{r^2_{\text{flock}}}}, \qquad (3)$$

where $K_{\text{col.}} = r^2_{\text{collision}} + r_{\text{flock}}$ and $K_{\text{conn.}} = r_{\text{flock}}$. Two characteristic radii are important to flocking behavior: the collision radius between agents, $r_{\text{collision}} = 0.8\text{m}$ for all experiments, and the maximum desired separation between connected members, r_{flock}, which we vary in experiments. Even with the stabilized flocking gradient in Eq. (3), emergent flocking failed to converge regularly when not using the full connectivity propagation model. To create a more stable flocking task, we set the initial magnitude and direction of one *master* agent, $\vec{v}^*$, and direct the remainder of the flock to maintain connectivity with the following velocity-based control inputs:

$$u_k = \vec{v}^*, \quad u_i = -\sum_{j \neq i} \nabla_{\vec{x}_i} V_{ij}. \qquad (4)$$

When using over 10 agents, in order to maintain the flock, the weighting of the leader's velocity in the follower's flocking summation is increased by $\frac{n_{\text{agents}}}{10}$. With this new follow-the-leader flocking, where we are testing a network's ability to dynamically maintain connectivity, we examine the mean flocking velocity versus the desired flocking radius and connectivity models. A higher flocking radius leaves the agents less likely to collide, but makes agents more likely to lose connection. The behavior of agents and stability of the behavior is sensitive to the relative magnitudes of these radii.

The flock speeds with different maximum flocking radii (*i.e.* the separation that agents are pressured to stay below) are shown for the studied propagation models in Fig. 6. A similar experiment varying the number of agents is shown in Fig. 7, where the realistic propagation model again improves on a disk model, though the relationship to flock velocity is noisy and warrants further investigation. Mean velocity shows the capability of all agents to follow the identified leading agent, but it does not explicitly capture some emergent flocking behavior such as oscillating agents towards and away from the current swarm centroid (these mechanism do slow down the mean velocity by directing movement of the agents to be less co-aligned with the leader). The experimental randomness model represents a substantial gain over unit disk models designed for a conservative representation of communication abilities. With the MSF, properly quantifying the performance of flocking is difficult because the network often failed to register an agent's nearest neighbors, removing the separating force of the potential function and collapsing the flock. While this would result in a higher average velocity, it represents a divergence from desired behavior.

B. Formation Control

A common area of study in multi-agent systems is that of forming and maintaining pre-defined shapes through local communication [5]. The experiments in this section only use the propagation model to allow scaling to higher agent counts (recall the challenge of network formation with simple propagation models discussed in Sec. IV). When communication is limited, the task can take more time to

converge or even fail, as without a global critic, local agents do not know the success of their controller. In this paper, we deploy agents that seek to organize themselves into a single line, irrespective of any particular ordering of agents or orientation of the line itself. The decentralized line formation algorithm follows from [46]: at each timestep, every agent locally applies least-squares to its own position and that of all neighbors it is connected to; this produces a local approximation of the optimal global line of convergence for each agent, which it uses to update its next control input. An example of the task is shown in Fig. 8a. When agents drop from communication, consensus among desired planes of convergence breaks down, and sometimes leads to multiple sub-formations being formed.

We benchmark the effects of communication modeling on this task through two metrics: convergence time and residual error. Convergence time is defined as the number of timesteps taken until all agents stop moving (i.e., until all agents individually believe the line formation task has been achieved). Note that this captures only the local belief state of the agents, and not the true global quality or convergence of the agent formation into a line. Residual error, calculated directly from applying a standard line-fitting least square procedure over the positions of all agents, shows the relative accuracy of a given formation. During initialization, we randomly spawn a given number of agents within an adaptive circular region centered around the origin such that the density of agents is approximately $5\,\mathrm{agents}/\mathrm{m}^2$. The log residual error for a group of n agents where the coordinates of each agent are (x_i, y_i) is calculated as:

$$\log \sum_{i=1}^{n} \big(y_i - (a_0 + a_1 x_i)\big)^2, \tag{5}$$

where (a_0, a_1) are the learned least squares parameters over the n agents.

For formation control, the experimental randomness model performs very closely to ideal communications, but simpler models show a wider variation of performance. The log-scale formation residual error is shown in Fig. 8b and the time taken to converge to said formation is shown in Fig. 8c. Larger effects on performance are shown between the uniform and probabilistic disk models, where the lower communication range has a strong effect on performance accuracy, but the probabilistic model takes longer to converge as the agents lose and re-establish communications with nearby agents in the formation more frequently.

VI. Discussion and Future Work

Our experiments demonstrate the importance of the network propagation model and schedule function on multi-agent task performance and agent scaling. There are two ways to view the results: first, if the prior network model is full connectivity, introducing the more realistic model may seriously degrade performance and require adapting existing controllers. On the other hand, the experimental randomness propagation model, as an improvement from the most conservative model using the lower bound of RF path loss, can represent an increase in potential task performance (e.g., average velocity while flocking as shown in Fig. 6). This change also motivates the need for refinement of network schedule functions, as allowing agents previously represented by a conservative unit-disc model to connect with more neighbors and move through space with more velocity presents a higher likelihood of losing connection with any given neighbor due to the physical dynamics.

The experiments presented here operate at a high level of abstraction, and do not necessarily point to specific improvements to be made in applications of networking and control. For example, this work has done little to optimize the scheduling function or inform how these communications models pose a risk for tasks where multi-hop connections are needed to send data to a central node, such as routing to a centralized controller during a search and rescue or exploration task. Creating methods for networks to add and drop members is also important, especially in the context of realistic communications. BotNethas been designed in order to help answer these, and other, questions in a more focused way in the future.

Although other communication methods used in swarms such as short-range infrared links or WiFi are not explored here, the RPC-based synchronization scheme that links the 6TiSCH simulator to the agent physical dynamics could be extended to include additional networking mediums in future work exploring multi-modal communication.

VII. Conclusion

This work presents a framework for understanding the performance and agent-count scaling of networked multi-agent robotic systems by more closely studying the effects of mobile RF communications. Although RF communication has been used for many experimental and simulated robotic tasks, improvements to the networking implementation using findings from the mobile wireless sensor networks community has been limited. Using our simulator, we have shown that even relatively simple multi-agent control tasks become substantially more difficult when including realistic propagation models and scheduling functions, and these difficulties are compounded when increasing the number of agents. The challenges we present in this paper will only be exacerbated when transitioning to the real world, and we hope BotNet will aid in designing new tools and techniques for networked, multi-agent control that scale to large, flexible, robot swarms.

Acknowledgments

The authors would like to thank Brian Kilberg for his work on an earlier form of this simulator and the reviewers for their helpful feedback.

REFERENCES

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[2] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[3] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *International workshop on swarm robotics*. Springer, 2004, pp. 10–20.

[4] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE transactions on robotics and automation*, vol. 17, no. 6, pp. 947–951, 2001.

[5] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[6] D. D. Siljak, *Decentralized control of complex systems*. Courier Corporation, 2011.

[7] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[8] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks*. John Wiley & Sons, 2010, vol. 4.

[9] R. R. Murphy, S. Tadokoro, and A. Kleiner, "Disaster Robotics," in *Springer Handbook of Robotics*, ser. Springer Handbooks, B. Siciliano and O. Khatib, Eds. Cham: Springer International Publishing, 2016, pp. 1577–1604. [Online]. Available: https://doi.org/10.1007/978-3-319-32552-1_60

[10] R. Shakeri, M. A. Al-Garadi, A. Badawy, A. Mohamed, T. Khattab, A. K. Al-Ali, K. A. Harras, and M. Guizani, "Design Challenges of Multi-UAV Systems in Cyber-Physical Applications: A Comprehensive Survey and Future Directions," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3340–3385, 2019.

[11] D. S. Drew, "Multi-agent systems for search and rescue applications," *Current Robotics Reports*, pp. 1–12, 2021.

[12] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanskỳ, D. Heřt, M. Petrlík, T. Báča, V. Spurnỳ *et al.*, "Darpa subterranean challenge: Multi-robotic exploration of underground environments," in *International Conference on Modelling and Simulation for Autonomous Systesm*. Springer, 2019, pp. 274–290.

[13] E. Municio, G. Daneels, M. Vučinić, S. Latré, J. Famaey, Y. Tanaka, K. Brun, K. Muraoka, X. Vilajosana, and T. Watteyne, "Simulating 6tisch networks," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 3, p. e3494, 2019.

[14] A. K. Demir and S. Bilgili, "Diva: a distributed divergecast scheduling algorithm for ieee 802.15. 4e tsch networks," *Wireless Networks*, vol. 25, no. 2, pp. 625–635, 2019.

[15] Y. Tanaka, K. Brun-Laguna, and T. Watteyne, "Trace-based simulation for 6tisch," *Internet Technology Letters*, vol. 3, no. 4, p. e162, 2020.

[16] A. M. Mehta, "Mobility in wireless sensor networks," Ph.D. dissertation, UC Berkeley, 2012.

[17] R. Silva, J. S. Silva, and F. Boavida, "Mobility in wireless sensor networks–survey and proposal," *Computer Communications*, vol. 52, pp. 1–20, 2014.

[18] M. Ali, T. Suleman, and Z. A. Uzmi, "Mmac: A mobility-adaptive, collision-free mac protocol for wireless sensor networks," in *IEEE International Performance, Computing, and Communications Conference*. IEEE, 2005, pp. 401–407.

[19] B. M. Khan and F. H. Ali, "Collision free mobility adaptive (cfma) mac for wireless sensor networks," *Telecommunication Systems*, vol. 52, no. 4, pp. 2459–2474, 2013.

[20] Y. Gu, Y. Ji, J. Li, and B. Zhao, "Eswc: Efficient scheduling for the mobile sink in wireless sensor networks with delay constraint," *Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1310–1320, 2012.

[21] Z. Vincze and R. Vida, "Multi-hop wireless sensor networks with mobile sink," in *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, 2005, pp. 302–303.

[22] R. K. Williams, A. Gasparri, and B. Krishnamachari, "Route swarm: Wireless network optimization through mobility," in *IEEE International Conference on Intelligent Robots and Systems*, 2014, pp. 3775–3781.

[23] S. Wang, A. Gasparri, and B. Krishnamachari, "Robotic message ferrying for wireless networks using coarse-grained backpressure control," *IEEE Transactions on Mobile Computing*, vol. 16, no. 2, pp. 498–510, 2017.

[24] T. Balch and R. C. Arkin, "Communication in reactive multiagent robotic systems," *Autonomous robots*, vol. 1, no. 1, pp. 27–52, 1994.

[25] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on automatic control*, vol. 50, no. 2, pp. 169–182, 2005.

[26] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv preprint arXiv:1802.08757*, 2018.

[27] R. C. Arkin and J. Diaz, "Line-of-sight constrained exploration for reactive multiagent robotic teams," in *International Workshop on Advanced Motion Control*. IEEE, 2002, pp. 455–461.

[28] D. Zelazo, A. Franchi, F. Allgöwer, H. H. Bülthoff, and P. R. Giordano, "Rigidity maintenance control for multi-robot systems," in *Robotics: science and systems*, 2012, pp. 473–480.

[29] M. C. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *IEEE Conference on Decision and Control*. IEEE, 2006, pp. 3628–3633.

[30] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2480–2496, 2008.

[31] H. Su, X. Wang, and Z. Lin, "Flocking of multi-agents with a virtual leader," *IEEE transactions on automatic control*, vol. 54, no. 2, pp. 293–307, 2009.

[32] I. Filippidis, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Decentralized multi-agent control from local ltl specifications," in *IEEE Conference on Decision and Control*. IEEE, 2012, pp. 6235–6240.

[33] A. C. Jiménez, V. García-Díaz, and S. Bolaños, "A decentralized framework for multi-agent robotic systems," *Sensors*, vol. 18, no. 2, p. 417, 2018.

[34] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154.

[35] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[36] A. R. Cheraghi, K. Actun, S. Shahzad, and K. Graffi, "Swarm-sim: A 2d 3d simulation core for swarm agents," in *International Conference on Intelligent Robotic and Control Engineering*, 2020, pp. 1–10.

[37] E. Soria, F. Schiano, and D. Floreano, "Swarmlab: a matlab drone swarm simulator," 2020.

[38] K. DeMarco, E. Squires, M. Day, and C. Pippin, "Simulating collaborative robots in a massive multi-agent game environment (SCRIMMAGE)," in *Int. Symp. on Distributed Autonomous Robotic Systems*, 2018.

[39] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "Openwsn: a standards-based low-power wireless development environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[40] M. Kudelski, L. M. Gambardella, and G. A. Di Caro, "Robonetsim: An integrated framework for multi-robot and network simulation," *Robot. Auton. Syst.*, vol. 61, no. 5, p. 483–496, May 2013. [Online]. Available: https://doi.org/10.1016/j.robot.2013.01.003

[41] M. Calvo-Fullana, D. Mox, A. Pyattaev, J. Fink, V. Kumar, and A. Ribeiro, "Ros-netsim: A framework for the integration of robotic and network simulators," *Preprint*, 2020.

[42] T. Issariyakul and E. Hossain, "Introduction to network simulator 2 (ns2)," in *Introduction to network simulator NS2*. Springer, 2009, pp. 1–18.

[43] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[44] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[45] M. M. Zavlanos, A. Jadbabaie, and G. J. Pappas, "Flocking while preserving network connectivity," in *IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2919–2924.

[46] X. Yun, G. Alptekin, and O. Albayrak, "Line and circle formation of distributed physical mobile robots," *Journal of Robotic Systems*, vol. 14, no. 2, pp. 63–76, 1997.