A Software Ecosystem for Autonomous UAV Swarms

Vivek Shankar Varadharajan, David St-Onge, Ivan Švogor and Giovanni Beltrame
Department of Computer and Software Engineering, MISTLab, Polytechnique Montréal
Montréal, QC, Canada
Email: {*vivek-shankar.varadharajan, d.st-onge, ivan.svogor, giovanni.beltrame*}*@polymtl.com*

Abstract—**Despite previous successes and growing research interest in swarm robotics there are still many challenges to be addressed. We focus on compatibility, code portability and programability of swarms. Most research teams customize existing hardware platforms and develop very specific software platforms with little to no regard to previously mentioned issues. In this paper we present our ongoing efforts to create a platform-agnostic software ecosystem for swarms of UAVs that can be used with off-the-shelf hardware, and with existing simulators.**

I. Introduction

Multi-robot systems and swarms of unmanned aerial vehicles (UAVs) in particular have many practical applications. In recent years there have been significant advancements in this research field. However, very rarely do UAV swarms leave the controlled and safe environment of laboratories, and when they do it is for short-duration experiments. The current use of swarms is generally based on custom, centralized solutions, in environments with reliable communication. Our work is focused on swarm deployment, fully distributed autonomy, and programability of heterogeneous swarms. Our goal is to develop a platform-agnostic software ecosystem for swarms which provides a way to reuse existing swarm behaviors, regardless of the underlying fleet.

II. The software ecosystem

Figure 1 shows ROSBuzz[1], a platform-agnostic software ecosystem for swarms created to address previously described challenges. It consists of four independent layers based on ROS[2] and MAVLink[3].

1) Swarm control layer: This is the topmost layer in charge for implementation of swarm behavior algorithms through a of a set of Buzz programs [1]. Buzz is a domain specific language (DSL) inspired by Python and Lua which provides swarm specific data structures and control sequences independently from the underlying platform. The desired swarm behavior is defined in a Buzz script that is shared with the fleet, and executed by all the members of the swarm. Typically, the script implements behaviors by allocating tasks to groups of UAVs, based on neighbor communication (broadcast, querying specific topics, and overall swarm consensus [2]).

[1]https://github.com/MISTLab/ROSBuzz
[2]http://www.ros.org
[3]http://qgroundcontrol.org/mavlink/start

2) Swarm communication layer: This layer is used to provide a communication platform between robots in the swarm and it is designed to minimize data transfers. To implement swarm algorithms, we need a form of situated communication [3]. All outgoing messages are tagged with the position information of the source (from GPS or other localization system). When messages are received, the layer updates the local information about neighbors locations. It is worth noting that there is no assumption that the all swarm members can communicate at all times. This layer is implemented in C++ as a ROS node, while the low-level communication is realized by XBee modules. However, since it is independent from both top and bottom layers, it can be easily replaced with any other hardware (e.g. WiFi or Bluetooth).

3) Control distribution layer: This is the key node of the entire ecosystem. It publishes a ROS node (`rosbuzz_node`) which acts as a mediator between ROS and Buzz. This layer is in charge of obtaining sensor data by subscribing to specific ROS topics and delivering the content to Buzz scripts. In turn, Buzz scripts use this information to propagate control messages which are then translated into the MAVLink protocol in ROSBuzz. The subsequent layers deliver these messages to the actual or simulated robot hardware. When Buzz scripts are compiled, the result is an executable bytecode for the platform-independent Buzz Virtual Machine (BVM). The BVM is a stack-based virtual machine written in C, open–source, and highly customizable and portable. The key aspect of ROSBuzz

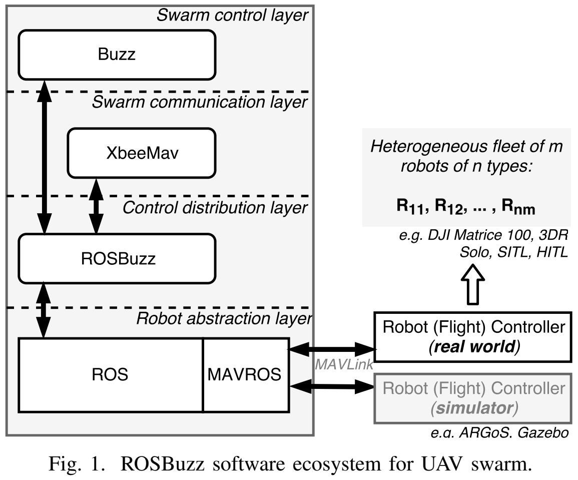
Swarm control layer
Buzz
Swarm communication layer
XbeeMav
Control distribution layer
ROSBuzz
Robot abstraction layer
ROS
MAVROS
MAVLink
Heterogeneous fleet of m robots of n types:
$R_{11}, R_{12}, \dots, R_{nm}$
e.g. DJI Matrice 100, 3DR Solo, SITL, HITL
Robot (Flight) Controller (real world)
Robot (Flight) Controller (simulator)
e.g. ARGoS, Gazebo

Fig. 1. ROSBuzz software ecosystem for UAV swarm.

is its usage of BVM to update and maintain swarm related information. Most importantly, there are two key data sets maintained by the BVM: a) the *swarm data holder* which contains all swarm specific information, and b) the *update monitor* which contains the bytecode, current system state, standby code and currently executed code. Buzz scripts can be updated during operations through the *update monitor*, which is a gateway for continuous integration of Buzz scripts. It manages current and new releases and provides an integrity test of the code. The main idea with this approach is to have the ability to change the swarm behavior while operating in conditions of long–term autonomy and to simplify the code deployment procedure across a large fleet.
A new code release can be activated by two triggers: a *file system trigger* which occurs upon change of the script, and a *neighbor trigger* which occurs when any of the neighbors have a new code release. To achieve consistency across the swarm, a *barrier* synchronization mechanism is used to wait until a sufficient number of robots are ready to proceed [4]. This is achieved with Buzz primitives for swarm-level consensus.

The main goal of this layer is to hide the complex interaction between ROS and the BVM, and allow software developers to fully focus on development of swarm behavior and swarm related algorithms through Buzz scripts, while ROSBuzz handles Buzz compilation and message propagation towards ROS.

4) Robot Abstraction Layer: The connection between ROS and UAVs is realized through MAVROS[4], which is an extendable ROS communication node for the MAVLink protocol. MAVLink is used for marshaling and communication between upper layers and the robot hardware by a set of standardized messages. Due to its widespread use, MAVLink is continuously maintained and empowered by various third party tools for monitoring and control of (semi– or fully–) autonomous vehicles (e.g APM Planner).

ROSBuzz leverages the loose-coupling of a publisher-subscriber architectural style to allow robotics researchers and practitioners to adapt any of the layers for their particular use, without affecting other layers. However, since ROSBuzz is designed as a mediator between Buzz and ROS, there should hardly be any necessity for this. ROSBuzz places the focus on defining the swarm behavior, rather than to hardware and software infrastructure which realize it. ROSBuzz also allows for Buzz execution on both simulated and real drones, *hardware–in–the–loop* or *software–in–the–loop*.

III. Preliminary results

1) The heterogeneous platform: Our current experimental platform consists of a simulated environment and a heterogeneous swarm platform. For the simulated environment we use both ARGoS[5] and Gazebo since both have advantages and drawbacks, and for our heterogeneous swarm platform we use six DJI Matrice 100 (equipped with either NVidia Jetson TK1 or TX1 on-board computers, Zenmuse X3 camera, and a collision avoidance module) and four 3DR Solo (with a mounted Raspberry Pi3) quadcopters.

[4]https://github.com/mavlink/mavros

2) Ongoing experiments: We validated the implementation and performance of the entire software ecosystem in simulation and with real robots. Our initial experiments, targeted at validating the communication model, demonstrated flocking of multiple, heterogeneous UAVs shown in Figure 2.

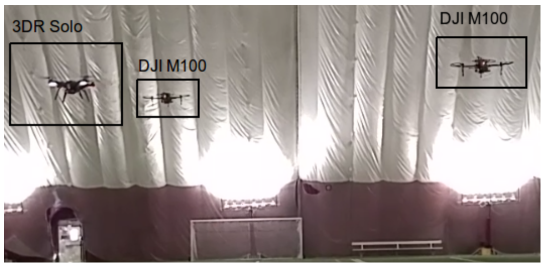

Fig. 2. Heterogeneous swarm consisting of one 3DR Solo and two DJI M100 UAVs.

At the moment we are improving ROSBuzz and performing experiments related to *heterogeneous swarm pattern formation* and movement.

IV. Conclusions

The goal of this paper is to briefly present our ongoing efforts to develop a platform-agnostic software ecosystem for heterogeneous UAV swarms. To design swarm behavior we use Buzz, a domain-specific language with special primitives for swarm programming. A major concern related to portability and compatibility with existing platforms is addressed through MAVROS. Our ROS node, ROSBuzz mediates the communication between Buzz and ROS and enables robotics researchers and practitioners to focus on the swarms with little concern to hardware and software infrastructure. This allows Buzz to be executed on a whole variety of real and virtual robots available through simulation environments like Gazebo, or ARGoS which support MAVLink. Finally, the main contributions of this work are a) a complete open source software ecosystem, ready to fly on distributed UAV systems, b) platform agnostic solution realized through multiple independent, exchangeable loosely coupled layers and c) swarm behavior update system for in–the–air safe manipulation of the fleets' intelligence.

References

[1] C. Pinciroli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 2016, pp. 3794–3800.

[2] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*, ser. BICT'15, ICST, Brussels, Belgium, Belgium, 2016, pp. 287–294.

[3] K. Støy, "Using situated communication in distributed autonomous mobile robots," *Proceedings of the 7th Scandinavian Conference on Artificial Intelligence*, pp. 44–52, 2001.

[4] R. Nagpal, "A catalog of biologically-inspired primitives for engineering self-organization," *Engineering Self-Organising Systems*, pp. 53–62, 2004.

[5] C. Pinciroli, V. Trianni, R. O'Grady, and et.al., "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.