Effecting Large-Scale Adaptive Swarms Through Intelligent Collaboration (ELASTIC)

James Edmondson

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

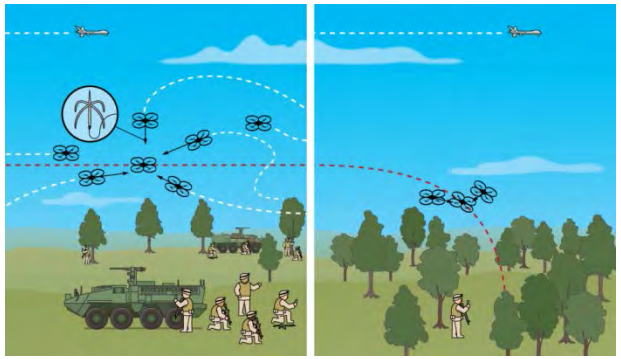
DM-0002816

Autonomous systems are difficult

1. Current unmanned systems (UAS) are **individually controlled by a handful of pilots and potentially dozens of analysts** micromanaging every aspect of the device
2. This control paradigm results in **poor scalability and high training costs**
3. A centralized control station is also **prone to failure, bottlenecks, and enemy attacks** taking out all UAS managed by that station
4. Missions change but **AI tends to be static and preset**

We want to make controlling autonomous systems scalable, effective, and predictable

1. Allow **one person** to **command an entire swarm** of UAS to do mission-level tasks
2. Focus on **1) scalability**, 2) bringing **simulated distributed AI to reality**, and providing 3) **predictable control of UAS** logic, threads, sensors, actuators and software components
3. **Provide all distributed algorithms and capabilities through open source** release of middleware and software via BSD-style licenses at Sourceforge and Github

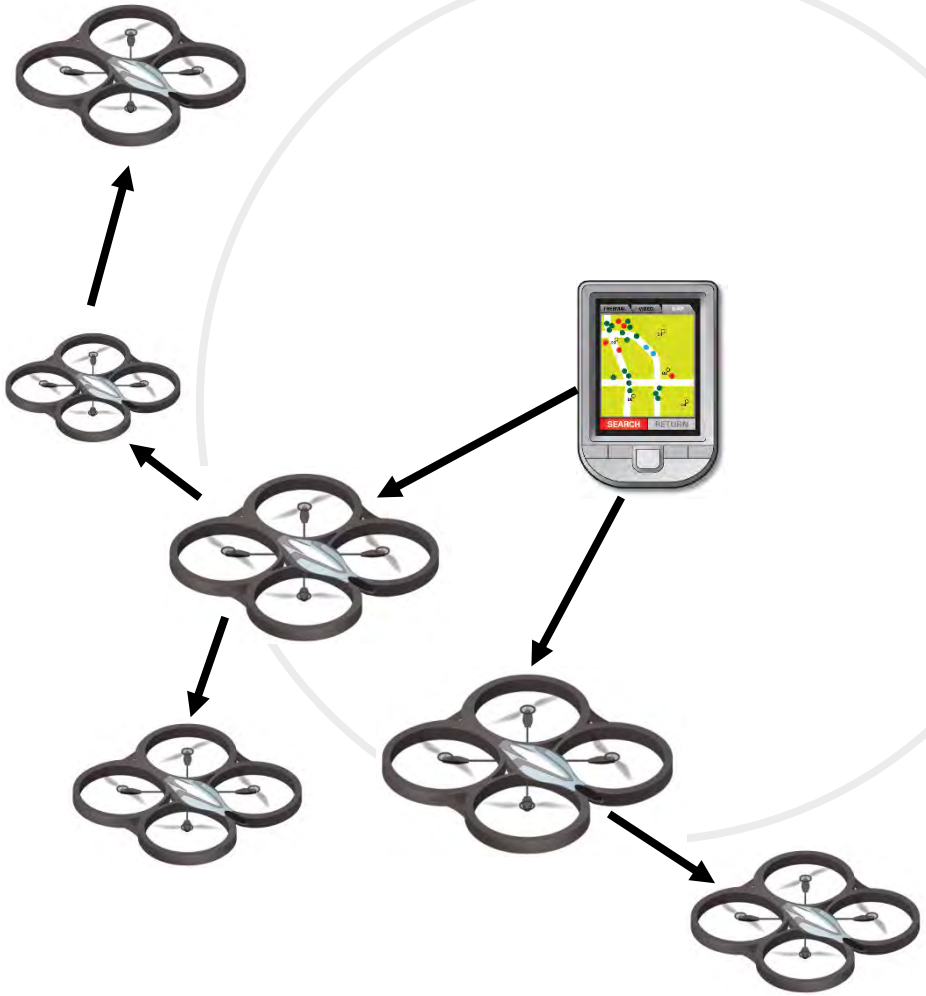


The state of the art in robotics control

Feature	ROS	Stage/Player	ELASTIC *
Scalability	Low. Dozens.	Low. Dozens.	**High**. Thousands.
Potential in partially disconnected environments	Low. Blocking (TCP)	Low. Blocking (TCP)	**High**. Non-blocking (UDP). Expects disconnects
Consistency/QoS	None	None	Lamport clocks, QoS-enabled threads
Knowledge Abstraction	Socket based	Message based	Containers for native language debugging, real-time scripting language
Simulation transition to reality	Some. Mostly Custom	Some. Mostly Custom	Built-in support

* Via the GAMS (http://jredmondson.github.io/gams/) and MADARA (http://madara.sourceforge.net/) open source middlewares that we are creating and extending

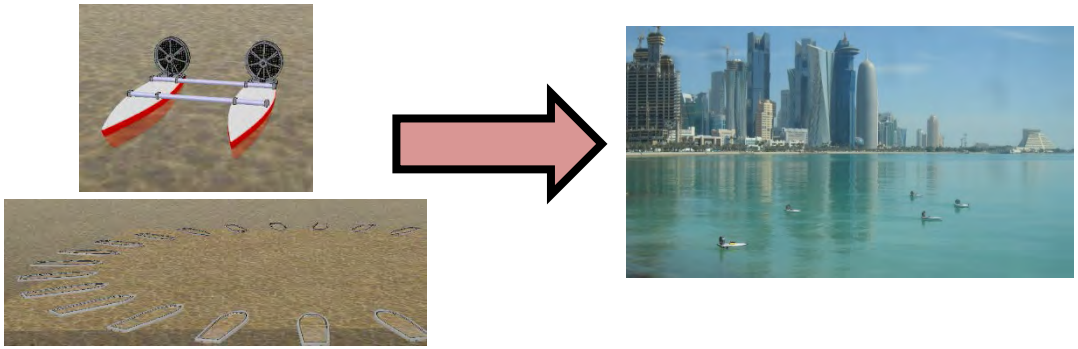
Challenge 1: Make distributed AI that is extensible, scalable and not hindered by disconnects

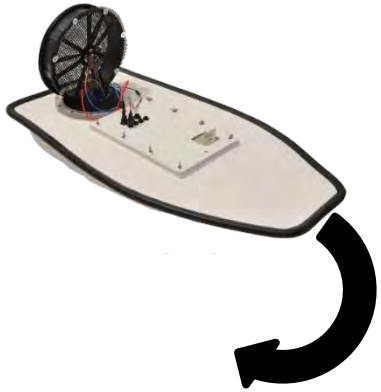


Solution

- **Use UDP** instead of TCP and DDS
- **Minimize blocking behaviors** within the knowledge engine, threads
- Revamp middleware to **provide O(1) access to all knowledge** to minimize access times
- Add **scalable read threads, controlled publish/receive rates**
- Add **controllable AI execution** via MAPE constructs and controllable execution and threading
- Frequently **resend important information**

Challenge 2: Move Distributed AI from simulation to reality





GAMS can now provide automatic translation of coordinate planes, references and rotations in simulations to real-world

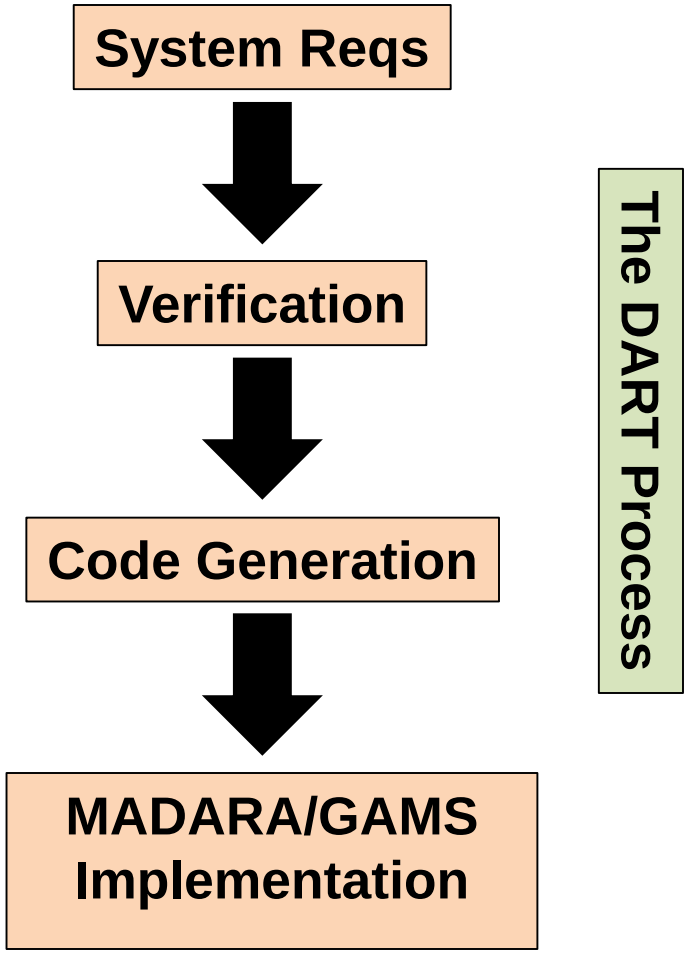
Solution

- **Provide Pose abstraction** that resolves **Cartesian-based sims** (the most common type of reference frame for sims) **to GPS** and other types of real-world reference frames
- Only use simulator engines that **provide real-world physics and accurate environments**
  - We developed boat models for VREP simulator that were **so accurate that they were used to develop PID controllers** for the boat

Challenge 3: Make AI controllable to aid verification techniques

Solution

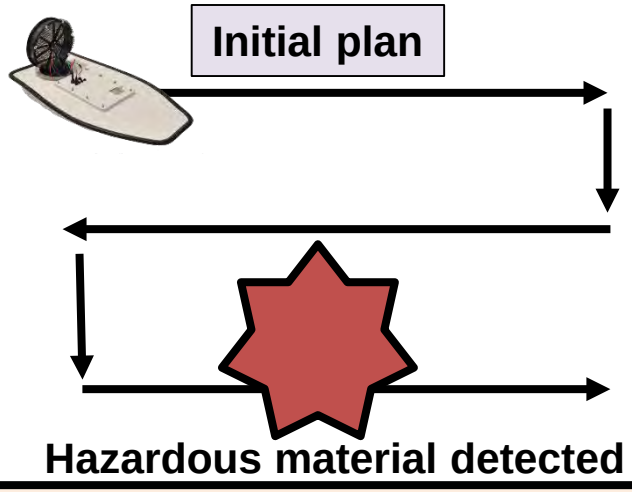
- **Modify threading models** with predictable execution
- **Modify networking layers to have guaranteed consistency** in knowledge updates
- **Create algorithms with synchronous models of computations** to help with predictability and consistency



1. Chaki, S., & Edmondson, J. (2014, July). Toward parameterized verification of synchronous distributed applications. *In Proceedings of the 2014 International SPIN Symposium on Model Checking of Software* (pp. 109-112). ACM.

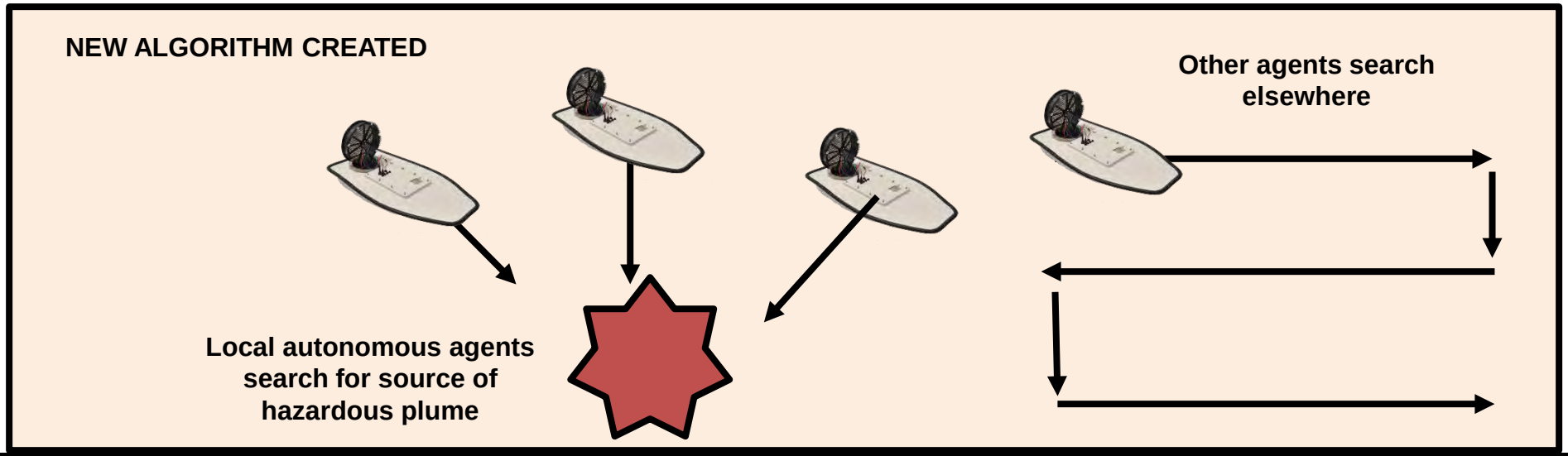
2. Chaki, S., Edmondson, J. (2014). Model-Driven Verifying Compilation of Synchronous Distributed Applications. *Model-Driven Engineering Languages and Systems*, Springer, LNCS, v.8767, pp. 201-217.

Challenge 4: Change AI when the mission or environment changes



Solution

- **Standardize command and control messages** between users and UAS
- **Modify controller** to allow distributed **algorithms** to be **changed at run time**
- **Create algorithm factory infrastructure** that allows users to **bind C++ and Java™ algorithms to command and control messages**



Scalability results with 20-node ODROID cluster

- Using ODROID boards that are similar to the internals of Platypus LLC boats to test scalability of real world distributed autonomy
- We believe current middleware and hardware on boat can scale to 280 before we miss optimal hertz rates

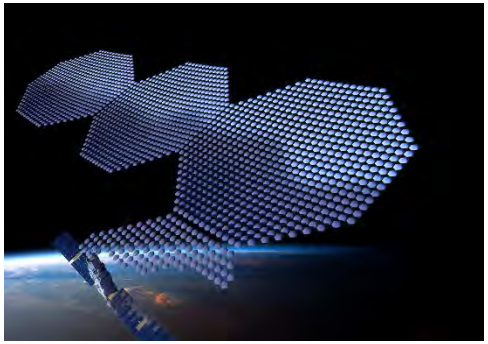
	Min Hz	Max Hz	Avg Hz
Per Node Publish	500.00	517.30	507.46
Per Node Receive	432.00	490.00	468.12
Per Node Total Receive	8,820.00	9,082.00	**8,921.38**
Total Throughput			133,868.00
Platypus Node Publish	5.00	35.00	31.00
Platypus Node Receive	5.00	35.00	31.00
Platypus Per Node Total Receive	100.00	700.00	**620.00**
Platypus Total Throughput			12,400.00
Estimated boat scaling	1764	240	**280**

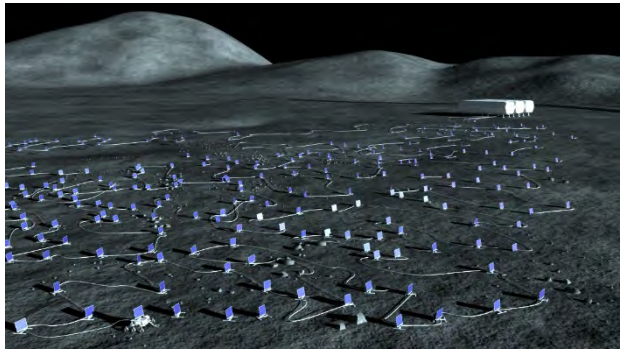
Platform and program development

- Platypus LLC has incorporated ELASTIC middlewares into their autonomous boat product
  - 100k+ lines of code in C++ were fully ported to Android™
  - Deployments of MADARA and GAMS in North America, South America, Middle East, and Europe
  - Deployed 20 boats in Doha, Qatar harbor

Platform and program development

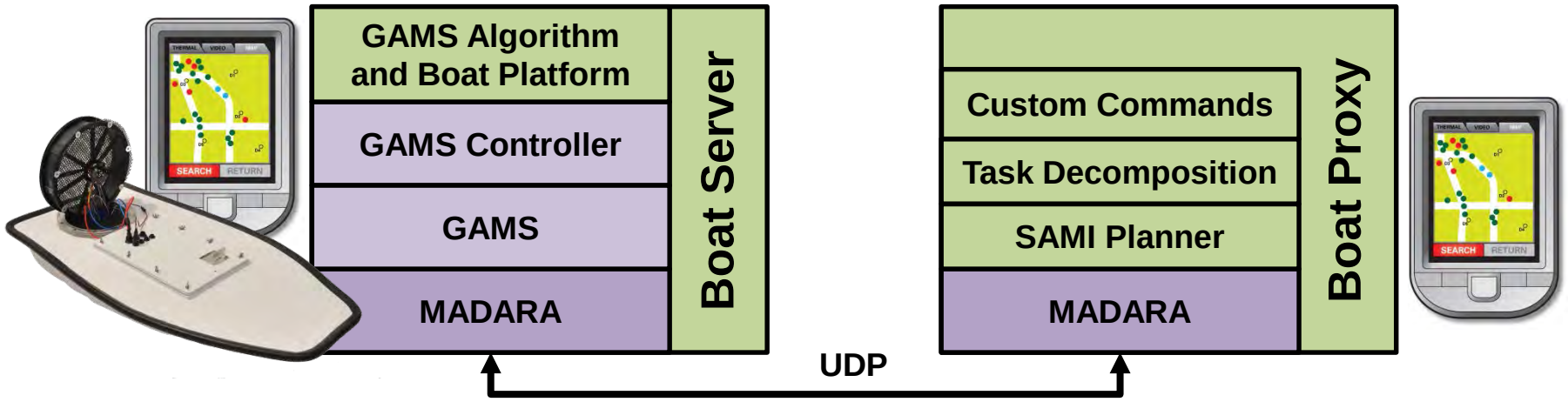
- ELASTIC middleware results have been incorporated into the Keck Institute for Space Studies Multi-Planetary Smart Tile proposal
  - Three phased project that puts 2,000+ tiles into 1) orbit, 2) the Moon, and 3) Mars
  - Energy harvesting, localization, power beaming, and oxygen/fuel harvesting
  - MADARA and GAMS provide knowledge, reasoning, and group autonomy for tile swarm
  - Created simulations and code for the Smart Tile as part of the planning and proposal process





How our technologies are being used

FY 2015 architecture for Platypus LLC and CMU collaboration



Boat Images © 2014-2015 Platypus LLC

FY16 Technical goals (next steps MADPARTS line)

1. Scale to 20+ boats working collaboratively
2. Split boat swarm into competitive groups
3. Extend GAMS middleware to include adversarial models
4. Create algorithms that are adversary aware, with a jamming adversary model
5. Extend poses and coordinate systems to individual actuator/sensor intrinsic translations
6. Make reliable knowledge transfer seamless (e.g., with commands)

FY16 collaborators

1. James Edmondson (CMU SEI)
2. Manuela Veloso (CMU CS)
3. Paul Scerri (CMU RI/Platypus LLC)

Closing remarks

In this presentation, we've discussed scalability and adaptivity results of the GAMS project

- An **extensible framework called GAMS for distributed algorithms and platforms**
  - MAPE-based for predictable AI execution and interaction with platform sensors and actuators
  - Extensible algorithms that can be changed at runtime by users or even the swarm itself
  - Translation between coordinate systems for use with algorithms that need to work in Cartesian, GPS or other reference frames
  - Predictable execution and application of knowledge in the presence of threading, multi-processing, networking, latency, and jamming
  - High concentration on quality-of-service and realization of practical swarm algorithms and platforms

Carnegie Mellon University

Software Engineering Institute
Carnegie Mellon

FY 2015 Open Source Release

The algorithms, tools, and middleware created at SEI are released via BSD-style licenses through the following projects:

- Multi-Agent Distributed Adaptive Resource Allocation (MADARA) for the distributed OS layer: http://madara.sourceforge.net/
- Group Autonomy for Mobile Systems (GAMS) for the algorithms and UIs: http://jredmondson.github.io/gams/
- Model Checking for Distributed Applications (MCDA) http://cps-sei.github.io/mcda/
- Drone-RK for the UAV device drivers: http://www.drone-rk.org
- Contact: jredmondson@sei.cmu.edu

SEI Project Members

James Edmondson

David Kyle

CMU Project Members

Paul Scerri

Nate Brooks

Christopher Tomaszewski

Jason Blum

Cormac O'Meadhra

Vanderbilt Students

Anton Dukeman (CS)

Carnegie Mellon University

Software Engineering Institute
Carnegie Mellon

Backup Slides

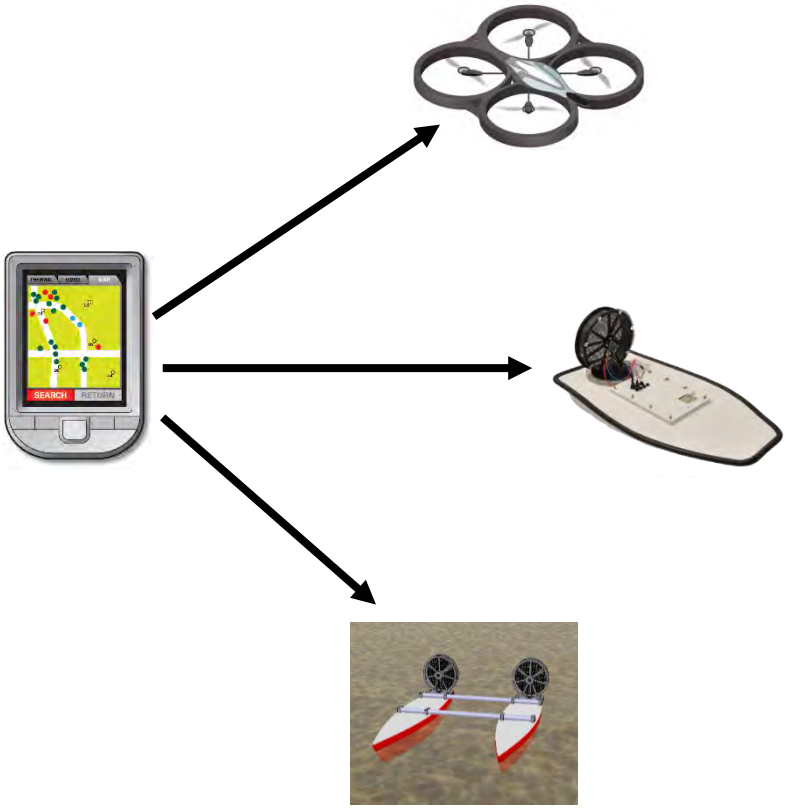
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

What is Group Autonomy for Mobile Systems?

An open-source middleware that provides extensible algorithms and platforms, knowledge sharing, timing, and control to distributed, heterogeneous autonomous systems in potentially disconnected, contested environments

What can GAMS do?

1. Allows one or more persons to control a swarm of UAS in simulation or in the real-world
2. Allows a distributed algorithm to be ran across any supported platform
3. Supports advanced networking features such as encryption, authentication, bandwidth shaping, deadline filtering, rebroadcasting of knowledge across a swarm, arbitrary filtering
4. Supports fine-grained control and predictable execution and consistency



What algorithms are supported?

Formation Coverage
Prioritized Region Coverage
Minimum Time Coverage
Serpentine Coverage
Waypoints
Formation Follow
Synchronized Formations
Convoy Shielding

What platforms are supported?

VREP Boat
VREP Quadcopter
VREP Ant Robot
ROS Pioneer 3DX
Platypus LLC Lutra Boat
C++, Java, Android, Python
ARM, Intel