Coordination in Micro-Aerial Vehicle Sensor Networks

Jason Waterman
School of Engineering and Applied Sciences, Harvard University
waterman@eecs.harvard.edu

Abstract

Research in micro-aerial vehicles (MAVs) is enabling insect-sized MAV swarms as a new class of mobile sensing systems. These MAVs are extremely resource constrained, leaving little energy for sensing, computation, and communication. Managing and coordinating these limited resources is a key challenge in MAV sensor networks. To achieve high efficiency, it is necessary to orchestrate resource management decisions across the network as a whole. This research summary presents a centralized system for coordinating and programming MAV swarms and also discusses future work on a decentralized coordination model which allows for more dynamic operation of MAVs in the field.

1 Introduction

Research in micro-aerial vehicles (MAVs) is enabling insect-sized MAV swarms as a new class of mobile sensing systems [2, 3, 14]. Due to their small size, these MAVs are extremely resource constrained, with most of the power budget going towards actuation [9]. This leaves even less energy for sensing, computation, and communication than traditional static sensor network nodes. While each individual MAV may not be very powerful, their small size means large numbers of MAVs can be deployed in a given area such as urban surveillance, crop pollination, and disaster recovery. While untethered controlled flight has yet to be achieved at this scale, Figure 1 shows one promising MAV prototype.

Managing and coordinating these limited resources is a key challenge in MAV sensor networks. To achieve high efficiency, it is necessary to orchestrate resource management decisions across the network as a whole. For example, consider an application such as crop pollination. MAVs must decide how much of their limited energy to invest in locating areas to be pollinated, storing and sharing that information with others, and performing the actual pollination. The resource load on each MAV is a complex function of the where the MAV is going, environmental conditions (e.g. wind), and the work performed when the MAV arrives at its target location. It is important to note that both resource load and resource availability fluctuate over time: an off-line static solution cannot sufce.

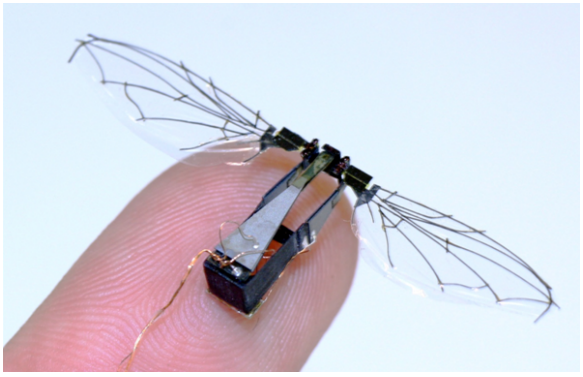

Figure 1. Example of a prototype insect-sized MAV.

Existing sensor node OSs provide little support for collective resource management. TinyOS [8] and SOS [6] provide only low-level interfaces for managing the hardware state of the node. Pixie [10] and Eon [13] provide greater control over node-level resource availability and tuning, but still focus only on individual nodes.

The need for coordination in a sensor network to achieve good resource efficiency has been recognized in the literature [1, 7]. However, previous approaches have been *ad hoc* in nature, focusing on point cases of specific problems, such as routing, tracking, or sensor coverage. Most of the proposed algorithms have only been studied in simulations, and the few implementations would have required substantial effort to build, given only low-level messaging support provided by the OS. As a result, general-purpose abstractions for coordinated resource management have yet to emerge.

Similar resource management problems arise in many other distributed systems, including Internet-based services [11], grids [5] and clusters [4, 12]. MAV sensor networks present new challenges in this space due to mobility; vastly different workloads; severe constraints on resources; and the need for low-overhead coordination mechanisms.

This research summary briefly presents *Karma*, a cen-

SenSys'11, November 1–4, 2011, Seattle, WA, USA.

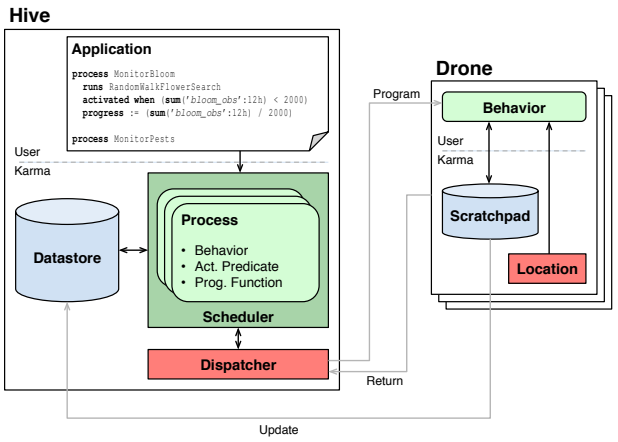

Figure 2. System block diagram

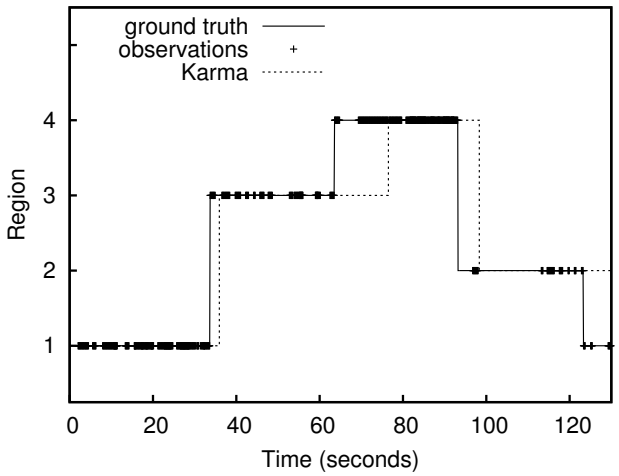

Figure 3. Testbed vehicle tracking results. The reported location lags behind the ground truth due to the latency of reporting data to the hive.

tralized system for coordinating and programming MAV swarms that runs on limited MAV resources, is robust to individual MAV failure, and is able to adapt to changes in the environment. Karma is evaluated with a custom simulator and in a testbed of toy helicopters controlled by the simulator. This paper also discusses future work on a decentralized coordination model which allows more dynamic operation of MAVs in the field.

2 Current Work

The extreme resource limitations of the insect-scale MAV platforms restrict the complexity of the programs that can be executed on the MAV. The design principles for controlling MAV swarms in this research is built around answering the following questions:

- How can the behaviors that execute on MAVs be simplified to match the resource constraints while maintaining global utility to the swarm?
- In what ways can simple MAV behaviors be combined to accomplish a swarm-level goal?
- To what extent can the burden of low-level MAV coordination be lifted from the end user?

Answering these questions leads to system architecture based on a *hive-drone* model in which individual MAVs, called *drones*, perform the simple sensing and manipulation tasks required to fulfill the goals of the swarm. Coordination is done by a centralized *hive*, which contains sufficient sensing, computation, and storage capabilities to manage the swarm. To simplify coordination, drones do not communicate with each other in the field or with the hive once they are dispatched. This simplifies drone programs and allows the application programmer can focus on implementing the correct behaviors for the application (what to do) while Karma reasons about the execution of the application on the swarm (where and when to do it).

To achieve this decoupling, the programming model allows the application programmer to compose programs from simple drone-level behaviors by relating the behaviors that produce information to the ones that consume it. This model allows for an easy and flexible composition of programs and enables the system to reason about drone scheduling efficiently.

Drones are stationed at the *hive*, which has a physical presence in the environment and the capability to recharge drone batteries. The hive *Scheduler* determines how drones should be used to accomplish the swarm objectives. The hive *Dispatcher* programs drones to execute specific tasks in the environment. The *Datastore* is the hive's repository for all information dumped by the drones from their local *Scratchpad* upon returning to the hive. A block diagram of the system is shown in Figure 2.

This system design is coupled with a programming model that allows the application programmer to specify the desired swarm behaviors as sequences of location agnostic sensing, sensor processing, and actuation commands without concern for coordination. The programmer defines two functions for each drone behavior:

- **Activation Predicate**: A boolean function based on the information in the Datastore. The hive can allocate a drone to execute this behavior if the function evaluates to true.
- **Progress Function**: A function based on the information in the Datastore that evaluates to a real number between 0 and 1, indicating the progress made toward the application goal associated with the behavior. When this function returns a value of 1, the application has achieved the behavior's goal.

The activation predicates allows the hive to determine when it is appropriate to execute each behavior and the progress function allows the scheduler to decide how many drones to allocate to a behavior. The burden of low-level coordination has now been shifted from the user to the system.

2.1 Testbed Results

In addition to simulation experiments, which are omitted here for brevity, the system is evaluated on a testbed of small, commercially available MAVs, the $120 E-flite Blade mCX2

micro coaxial helicopter. The mCX2 is 20 cm long, 12 cm high, weighs 28 grams. It has a flight time of 6–8 minutes depending on flight maneuvers. The testbed experiments show how the system performs under real-world control and navigation conditions.

In one testbed experiment, a vehicle tracking application is evaluated. Three helicopters are used to search and track one remote controlled ground vehicle traveling along the testbed floor. The testbed space is divided into four equal regions and the helicopters are initially dispatched to a region with a `Search` behavior, looking for the ground vehicle. When a helicopter has located a vehicle, it writes the location of the vehicle to its local Scratchpad. When the helicopter returns to the hive and uploads its data to the Datastore, the tracking process is activated. The system then dispatches a helicopter with the `Track` behavior to the last known vehicle location to resume observation. A trace of the tracking experiment is shown in Figure 3.

3 Future Work

The centralized model of coordination presented above simplifies the programming individual drones at the cost of some flexibility. Once a drone has been dispatched to a given location with a given behavior, it will continue to perform that behavior until it returns to the hive.

Consider a simple pollination application where the hive has a map of flowers to be pollinated in the Datastore and drones are dispatched to pollinate them. In the current coordination model the hive is not updated with new information until drones return to the hive. Drones in the field will have a more up to date view of the world in their own local scratchpad. This can lead to inefficient and redundant work being done by the drones.

This seems to suggest that for some applications in-field coordination can improve the overall efficiency of the work performed. Of course, while in-field coordination has the potential for greater optimization, this must be traded off against the higher overhead for communicating resource demand and availability information.

One natural extension of the previous model is to allow drones to share their local Scratchpads with each other in the field and allow drones to modify their behavior based on this information. Given the fleeting and potentially random interactions between the drones in the field, a global but weakly-consistent view of the Datastore would seem to provide a good trade-off between efficiency (in terms of radio messages) and consistency within local neighborhoods. Given the inherent spatial nature of the applications, drones that are nearer to each other are more likely to require more careful coordination than drones that are far away from each other.

My future research hopes to answer the following open questions:

- What is the right abstraction for sharing data?
- Given the limited memory of the drones how do your prioritize which information gets shared among the local nodes?
- How how frequently should drones exchange data?
- How does new information influence existing drone behaviors?
- Can the programming model still be kept simple with in-field coordination?

4 References

[1] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 450–459, New York, NY, USA, 2007. ACM.

[2] X. Deng, L. Schenato, W. C. Wu, and S. Sastry. Flapping flight for biomimetic robotic insects: Part i-system modeling. *IEEE Transactions on Robotics*, 22(4):776 –788, Aug 2006.

[3] M. H. Dickinson, F.-O. Lehmann, and S. P. Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999.

[4] D. G. Feitelson and L. Rudolph. Gang scheduling performance benefits for fine-grain synchronization. *Journal of Parallel and Distributed Computing*, 16:306–318, 1992.

[5] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing*, number 3779 in LNCS, pages 2–13, 2005.

[6] C.-C. Han, R. K. Rengaswamy, R. Shea, E. Kohler, and M. Srivastava. SOS: A dynamic operating system for sensor networks. In *Proc. Third International Conference on Mobile Systems, Applications, And Services (Mobisys)*, 2005.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. the 33rd Hawaii International Conference on System Sciences (HICSS)*, January 2000.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Proc. the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, Boston, MA, USA, Nov. 2000.

[9] M. Karpelson, J. Whitney, G.-Y. Wei, and R.J.Wood. Energetics of flapping-wing robotic insects: Towards autonomous hovering flight. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[10] K. Lorincz, B. rong Chen, J. Waterman, G. Werner-Allen, and M. Welsh. Resource aware programming in the pixie os. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 211–224, 2008.

[11] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Design and implementation tradeoffs for wide-area resource discovery. In *HPDC '05: Proceedings of the High Performance Distributed Computing, 2005. HPDC-14. Proceedings. 14th IEEE International Symposium*, pages 113–124, Washington, DC, USA, 2005. IEEE Computer Society.

[12] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum. Locality-aware request distribution in cluster-based network servers. In *ASPLOS-VIII: Proceedings of the eighth international conference on Architectural support for programming languages and operating systems*, pages 205–216, New York, NY, USA, 1998. ACM.

[13] J. Sorber, A. Kostadinov, M. Brennan, M. Garber, M. Corner, and E. D. Berger. Eon: A Language and Runtime System for Perpetual Systems. In *Proc. ACM SenSys*, November 2007.

[14] R. J. Wood. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE Transactions on Robotics*, 24(2):341 – 347, April 2008.

Biographical Sketch

Jason Waterman is a 5th year PhD student in Computer Science at the School of Engineering and Applied Sciences, Harvard University. His research involves the design and implementation of systems for coordinated resource management in wireless and mobile sensor systems. His is part of Harvard Sensor Networks Laboratory and his adviser is Matt Welsh, currently at Google. He holds a B.S. and M.S. in Electrical Engineering from Tufts University. From 2001 to 2005 he was research staff at MITs Computer Science and Artificial Intelligence Laboratory. His expected date of dissertation submission is August 2012.