Adaptive Opportunistic Airborne Sensor Sharing

JACOB BEAL, KYLE USBECK, and JOSEPH LOYALL, Raytheon BBN Technologies, USA
MASON ROWE and JAMES METZLER, Air Force Research Laboratory, USA

Airborne sensor platforms are becoming increasingly significant for both civilian and military operations; yet, at present, their sensors are typically idle for much of their flight time, e.g., while the sensor-equipped platform is in transit to and from the locations of sensing tasks. The sensing needs of many other potential information consumers might thus be served by sharing such sensors, thereby allowing other information consumers to opportunistically task them during their otherwise unscheduled time, as well as enabling other improvements, such as decreasing the number of platforms needed to achieve a goal and increasing the resilience of sensor tasks through duplication. We have implemented a prototype system realizing these goals in Mission-Driven Tasking of Information Producers (MTIP), which leverages an agent-based representation of tasks and sensors to enable fast, effective, and adaptive opportunistic sharing of airborne sensors. Using a simulated large-scale disaster-response scenario populated with publicly available Geographic Information System (GIS) datasets, we demonstrate that correlations in task location are likely to lead to a high degree of potential for sensor-sharing. We then validate that our implementation of MTIP can successfully carry out such sharing, showing that it increases the number of sensor tasks served, reduces the number of platforms required to serve a given set of sensor tasks, and adapts well to radical changes in flight path.

CCS Concepts: • **Information systems → Information systems applications**; • **Computing methodologies → Distributed algorithms**;

Additional Key Words and Phrases: Field calculus, self-organization, self-stabilization, spatial computing, large-scale coordination

ACM Reference format:
Jacob Beal, Kyle Usbeck, Joseph Loyall, Mason Rowe, and James Metzler. 2018. Adaptive Opportunistic Airborne Sensor Sharing. *ACM Trans. Auton. Adapt. Syst.* 13, 1, Article 6 (April 2018), 29 pages.
https://doi.org/10.1145/3179994

1 INTRODUCTION

The use of airborne sensor platforms is rapidly increasing across a large number of commercial, civilian, law-enforcement, and military organizations, particularly in time-critical situations such

This work has been supported by the United States Air Force under Contract No. FA8750-14-C-0096. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views, opinions, and/or findings contained in this article are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Cleared for public release. Case number: 88ABW-2016-6037, AFRL, 23-Nov-2016. RGEMS E16-M54H Approved for public release; distribution is unlimited.
Authors' addresses: J. Beal, K. Usbeck, and J. Loyall, Raytheon BBN Technologies, 10 Moulton Street, Cambridge, MA 02138, USA; emails: jakebeal@ieee.org, {kyle.usbeck, joseph.loyall}@raytheon.com; M. Rowe and J. Metzler, Air Force Research Laboratory, USAF AFRL/RISA, 525 Brooks Road, Rome, New York 13441-4505, USA; emails: {mason.rowe, james.metzler}@us.af.mil.

https://doi.org/10.1145/3179994

as management of mass public events or responding to natural or industrial disasters. Enabled by advances in the automation of both flight control and sensor packages and increased availability of platforms making use of such techniques, a great diversity of platforms are being deployed, large and small, fixed and rotary-wing, manned and unmanned, and mounting a wide variety of sensors, particularly for imaging. Even so, much of the potential of these airborne sensor platforms remains untapped. Typically, any given platform is controlled by a single individual or organization with a focused set of goals, and as a result, an airborne sensor typically spends a large fraction of its flight time idle, e.g., while the platform is in transit to, from, and between the locations where sensor tasks are intended to be executed.

This spare sensor capacity provides an opportunity: if the operator of an airborne platform is willing to share their sensor, then it can be used to execute tasks for other organizations during its "down time," thereby potentially realizing the following benefits:

- —Airborne sensing capabilities can be made available to organizations that lack their own airborne platform.
- —The number of airborne platforms necessary to carry out a given set of missions can be reduced.
- —Mission resilience and execution speed can both be increased by assigning multiple platforms to perform the same task and/or by dynamically shifting tasks between platforms to adapt to changing circumstances in the field.

Consider, for example, the scenario illustrated in Figure 1, in which a disaster response team is using a ScanEagle UAV to check the San Luis Dam for damage following a large earthquake in the San Francisco Bay area. The planned flight path to check the dam also flies close to many other pieces of critical infrastructure, including airports, cell phone towers, and fire departments. Sensor sharing can allow other disaster response teams, which may want information about these other locations, to survey them while the UAV is in transit to and from the dam; complementarily, other UAVs flying nearby can use their own spare sensor capacity to conduct additional surveys of the dam, hedging against problems and possibly obtaining the information more quickly than with the original mission plan.

To realize these potentials for usage of spare sensor capacity, we have developed the protoype system Mission-Driven Tasking of Information Producers (MTIP) for the sharing of airborne sensors, which obtains sensor tasks from explicit information requests or requests implicit in other user interactions with information management systems, then uses an agent-based representation of tasks and sensors both to allocate tasks to sensors and to dynamically adapt that allocation at runtime. This article makes a full presentation of MTIP, with particular focus on its agent-based adaptive task allocation methods, combining and extending prior presentations in Beal et al. (2016a) and Beal et al. (2016b). Following a brief review of related work in Section 2, Section 3 presents the MTIP architecture and Section 4 presents and analyzes MTIP's agent-based method for adaptive task allocation. Section 5 then presents a validation of the proposed benefits of airborne sensor sharing, followed by experiments in Section 6 that confirm that our MTIP prototype can both effectively allocate tasks and also efficiently adapt those allocations at runtime. Finally, Section 7 summarizes conclusions and discusses future work.

2 BACKGROUND AND RELATED WORK

Many organizations already implement schemes for sharing information (including airborne sensor data) through a variety of information management systems, many of which are also specifically focused on maintaining situational awareness in emergency or military situations (e.g., Gillen et al. (2012), OMG Data Distribution Service Portal (2012), Xiong et al. (2007), Carvalho et al. (2011),

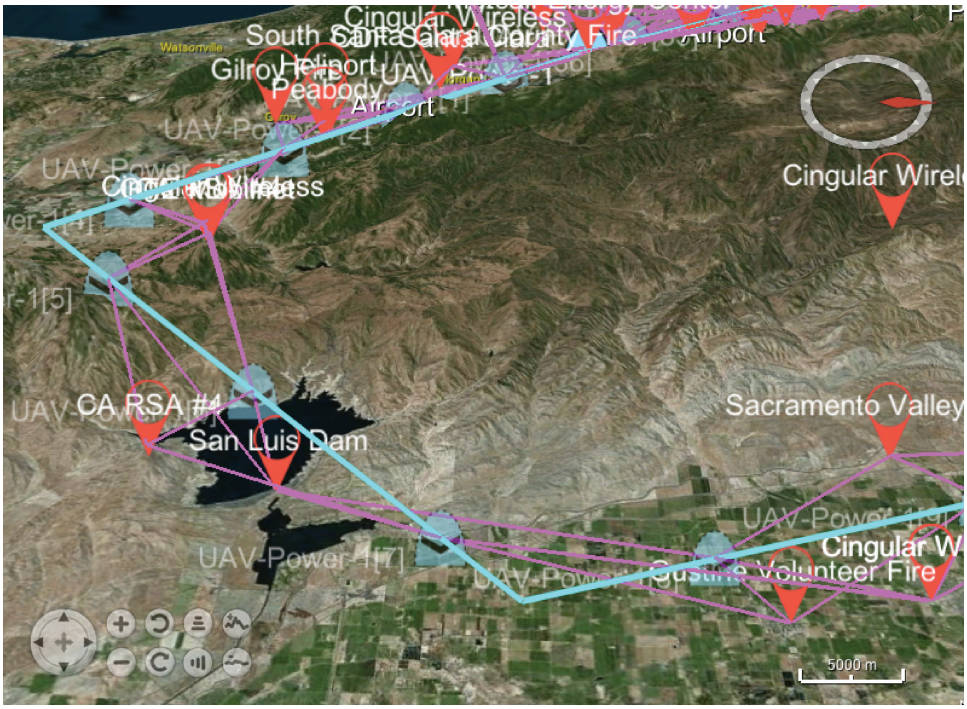


Fig. 1. Example of sensor sharing with MTIP: the planned path of a UAV flight intended to check the San Luis Dam for earthquake damage (cyan line with translucent blue air-asset tactical symbols) also flies close enough (magenta sight lines) to survey other critical infrastructure (red icons), including the San Martin airport, the South Valley Hospital and its heliport, and a number of fire departments and cell phone towers.

Usbeck et al. (2015), Rosenblum and Wolf (1997), Ma and Bacon (1998), and Essendorfer and Mueller (2009)). MTIP itself is built upon one of these systems (Gillen et al. 2012), but while these systems are good at enabling interested parties to share information, actually tasking sensors to gather that information must still be manually accomplished outside of the system by humans. Sensor sharing is thus still typically implemented by interactions between humans, which tends to create problems in discovery and to compete for attention with other critical tasks, contributing to sensor under-utilization.

Another closely related area is collective sensing, which has been studied for many years in a number of contexts, both for airborne and other types of platforms (Chong and Kumar 2003). Collective sensing also considers multiple sensor platforms, but often those platforms are all assumed to be controlled by a single organization that can task them and often also place and move them at will (e.g., Delin (2002), Burne et al. (1999), Allred et al. (2007), and Esterle et al. (2014)), rather than needing to be shared within the constraints imposed by an independent platform owner. Recently, smartphones and other human-carried sensors have begun to drive interest toward opportunistic sensor systems (Lane et al. 2008; Krause et al. 2008; Ma et al. 2014). While a number of such systems and mechanisms for such systems have been developed (e.g., Shin et al. (2011), Eisenman et al. (2008), Sheng et al. (2012), Jayaraman et al. (2013), and Liang et al. (2014)), all of these systems have focused on diffuse tasks, such as noise or pollution monitoring, in which sensors are non-directional and any given sensor contributes only a small portion of the sensing capability needed for the task. As a result, prior work on opportunistic sensing systems has not considered how to manage conflict between tasks for sensor resources.

MTIP, by contrast, focuses on highly directional sensors such as airborne cameras, and on more specific and localized tasks such as using a camera to image a particular target of interest. Furthermore, the requirements of information consumers (e.g., high-quality imagery) often mean that sensors can effectively serve only one task at a time. Thus, rather than one diffuse task extended over space and time, we are instead dealing with a potentially large number of discrete tasks with specific and conflicting requirements. As a result, a key problem, which cannot be addressed with prior opportunistic sensing systems, is to effectively manage the allocation of large numbers of competing tasks to individual sensors, both taking advantage of opportunities as they appear and rapidly adapting the allocation when tasks or sensor availability changes.

3 MISSION-DRIVEN TASKING of INFORMATION PRODUCERS (MTIP)

At a high level, the motivating purpose of MTIP is to address (in the context of airborne sensor platforms) a deficiency common to most publish-subscribe (pub-sub) Information Management Systems (IMS). Publish-subscribe systems decouple information producers from information consumers (thereby enabling scalability and resilience desirable in an IMS). Subscriptions are made by consumers as a way to inform the IMS what information is of interest to that consumer: clients make subscriptions into a pub-sub system with a *filter* or *predicate* that limits the information they receive to that which matches their interest. However, because a pub-sub IMS decouples information producers from information consumers, it is often the case that information producers will collect/provide information that is not of interest to any consumers, while remaining unaware of what information the consumers are actually interested in, and thus causing consumer's information needs to be poorly met or not met at all.

MTIP addresses this deficiency by assigning tasks to information producers that have the capability and availability to collect information that is of interest to a consumer. To accomplish this, MTIP monitors subscription predicates (including explicit requests for information about a location or other target) and other information organically provided by information consumers in the course of normal system usage. MTIP takes these expressed user interests and converts them into tasking, which is sent to the information producers, thus increasing the chances of meeting users' information needs, as shown in Figure 2. Specifically, the current MTIP prototype provides the following functionality:

- *Extraction of information needs from existing user interfaces.* MTIP is integrated with several user interfaces, including map-based interfaces to Geographic Information System (GIS) datasets, such as OpenStreetMap, to support both explicit user requests for information and to extract implicit user interests from commonly performed actions. For example, MTIP identifies placement of a Point of Interest (POI), drawing of an Area of Interest (AOI), or creation of a route on a map as indicating user interest in getting imagery at the POI, within the AOI, or along the route, respectively. To this end, MTIP includes a rich set of *semantic extractors* that extract user interests from these and other commonly used interface interactions, such as requests and queries.
- *Extraction of sensor tasks from information needs.* MTIP turns the information needs it has extracted into a set of inferred user goals that are desirable to achieve by sensor-carrying airborne platforms and, in turn, generates a set of specific tasks that can be performed to satisfy these goals.
- *Allocation of tasks to available platforms.* MTIP assigns the tasks it has derived to the set of available platforms with the objective of satisfying the largest number of tasks within the constraints of the expected flight plans of the platforms.

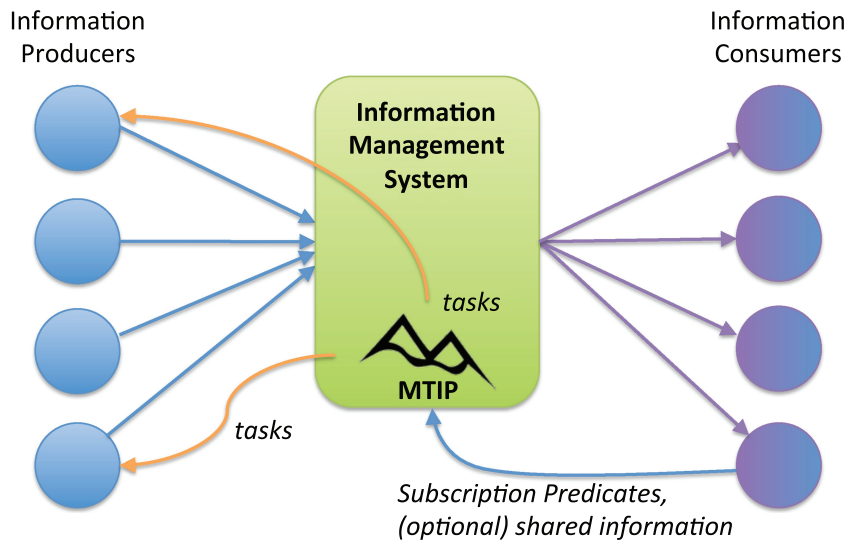


Fig. 2. Traditional Publish-Subscribe IMS decouple information producers from consumers, potentially causing consumers' information, needs to go unmet. MTIP uses organically-collected information from consumers (e.g., the predicates/filters controlling the information they receive from the IMS) to task information producers to collect and share information that is of interest to end-users.

Thus, by opportunistically increasing sensor utilization, MTIP can be used to achieve any or all of the following goals (depending on choices in deployment):

—Increasing the number of information gathering tasks that can be served by a given set of sensor platforms.
—Reducing the number of sensor platforms required to perform a given set of information gathering tasks.
—Increasing resilience of information gathering tasks by serving them on multiple platforms.

3.1 Context: Marti and TAK

In order to give a clear path to adoption and deployment, MTIP is designed to operate with and extend the capabilities of existing systems. In particular, MTIP is designed to augment the Team Awareness Kit (TAK) ecosystem of Situation Awareness (SA) tools (though it can potentially interact with any software that uses the Cursor on Target information exchange standard (Miller 2004; Kristan et al. 2009; Robbins 2007)). The TAK ecosystem is composed of a set of clients such as ATAK (Usbeck et al. 2015) and/or its Windows or iOS ports (WinTAK and iTAK, respectively) and an IMS, *Marti* a.k.a. TAKServer, which is an advanced tactical IMS that can be deployed on a variety of hardware platforms and any IP network (Gillen et al. 2011, 2012). While ATAK can function in serverless environments, there are times when a server is needed to bridge networks to provide Beyond Line-of-Sight (BLOS) communication (Gillen et al. 2012) or provide a communications substrate when networks otherwise prevent clients from communicating directly (e.g., those separated by firewalls or lacking IP addressability, such as commercial cellular) in austere conditions (Loyall et al. 2012). In those cases, Marti acts as the IMS to decouple information producers from information consumers, allowing new producers or consumers to be added on demand.

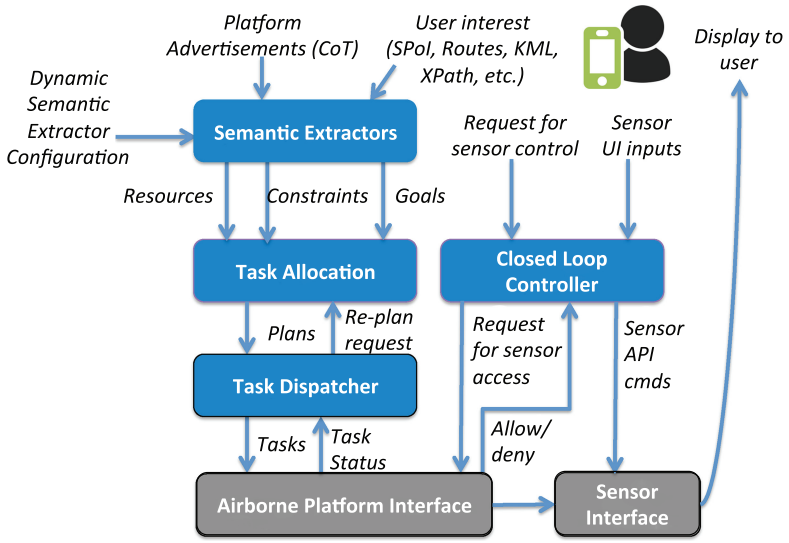


Fig. 3. Architecture of the MTIP prototype (blue), showing flow from situation information to sensor task allocation, dispatching of tasks to a set of airborne platforms (grey), and feedback of task status and results from platforms, triggering plan adaptation.

Marti uses a publish-subscribe model: information producers (publishers) are decoupled from information consumers (subscribers). Publishers submit information to Marti with metadata describing the time, location, and content of the information. Subscribers register a request for their information needs (i.e., their *subscription*) and Marti delivers published information matching that subscription. Marti can also archive published information for consumers to query for information that was published in the past. Further, Marti provides QoS management to prioritize and shape traffic to the network's available bandwidth. Full details of the Marti system are available in Gillen et al. (2011) and Gillen et al. (2012).

3.2 MTIP Architecture

The overall architecture of MTIP is shown in Figure 3, comprising the following components:

- A set of *Semantic Extractors* that gather explicit and implicit information on user interest (i.e., tasks) from user interfaces and activities, as well as receiving publications of platform announcements and route plans.
- A *Task Allocation* component that assigns tasks to platforms and receives and acts on platform responses and situation changes that may affect the assignment.
- A *Task Dispatcher* that sends task assignments to sensor platforms and receives back from those platforms messages accepting or rejecting tasks and providing information on the progress of tasks toward completion.
- A *Closed Loop Controller* that responds to requests for direct control of an airborne platform's sensor by a user (a separate means of control not further discussed in this article; its interaction with the rest of MTIP is similar to pilot override).

At present, MTIP's semantic extractors recognize five types of information:

- Sensor Points of Interest (SPoI),
- Routes (i.e., paths),

—AoI,
—Information Requests (Subscriptions/Queries),
—Geospatial Annotations (e.g., Keyhole Markup Language (KML)).

These types of information can be produced both directly as information requests through appropriate interfaces in TAK clients or indirectly as a side effect of other collaboration activities by TAK users, such as assignment of responsibilities or discussion of tactics regarding a target.

From these sources of information, the semantic extractors produce three classes of sensor tasking requests compliant to a specified platform interface: "point" tasks focused on a discrete location, route tasks, and polygonal area tasks. Each task request can also be annotated with additional requirements information derived from the initial communication flow by the semantic extractors, including the following properties:

Property	Description
Name	Textual description of the request for display
Priority	Importance of the request (e.g., based on user identity)
Start Time	Time by which the request must start execution
End Time	Time by which the request must be completed
Resolution	Desired ground resolution of the resultant image(s)
Sensor Mode	Desired mode for multi-modal sensors
Radius	The radius of interest surrounding the AoI
Revisit Time	Desired repetition frequency of a recurring task

These task requests are passed to the Task Allocation component in order to be assigned to particular platforms whose sensors can fulfill the requests. The task allocation component also receives advisements of the position, status, and path plan (if available) of all the platforms via advertisements from the platforms themselves. With this information and the current state of task allocation, the task allocation component determines which tasks will be assigned to which platforms (as elaborated in Section 4): any given task might be assigned to one platform, multiple platforms, or even no platforms at all if none is available that can satisfy the task. Task requests are then dispatched to platforms whose operators decide whether and how to carry them out, returning information to the task allocation component about whether they accept or reject requests they have been given, when tasks will be started and completed, and the expected quality of the resultant product.[1] With feedback from the platforms passed through the task dispatcher, the task allocation component can then adjust its assignment in response to this feedback, canceling or reassigning tasks as necessary.

Finally, as platforms carry out their assigned tasks, the sensor data that they produce is placed under management by the Marti system, where it is ultimately delivered to users in accordance with their subscriptions.

4 AGENT-BASED TASK ALLOCATION

We now turn our focus to the task allocation component in particular, the workflow for which is illustrated in Figure 4: as MTIP receives information about available resources (airborne sensor

[1]Note that MTIP does not currently include any explicit representation of costs, payments, or other information about incentives: this is because for the types of scenarios initially considered, these issues are typically handled separately at an organizational level through mechanisms such as mutual aid agreements and directives to organization members. Likewise, every platform is assumed to be "owned" by some organization, which has ultimate authority over its route and availability for tasking; thus, MTIP does not currently attempt to plan routes or otherwise override the operation of a platform.

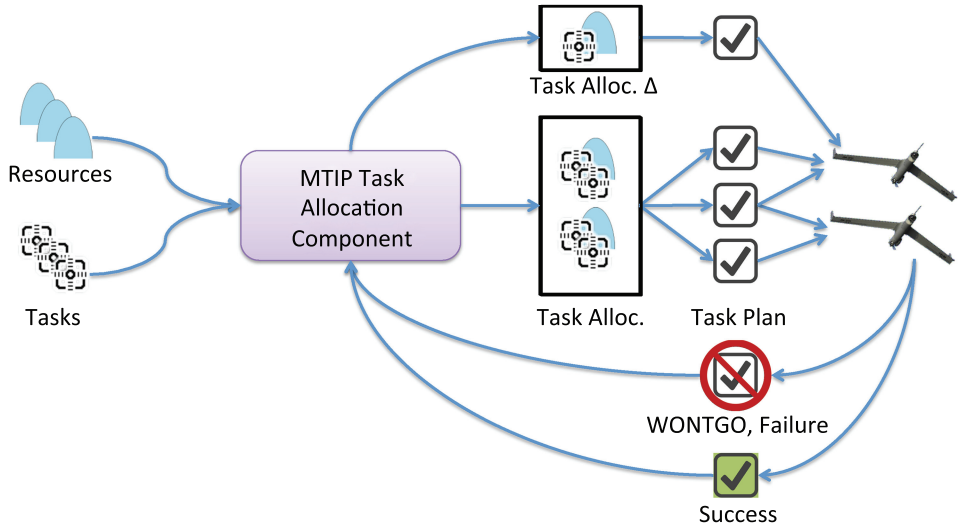


Fig. 4. MTIP (re)allocation workflow: whenever any change happens to a platform, a task, or the condition of a task at a platform, the task allocation system is triggered to run.

platforms) and tasks, it generates a task allocation plan that is passed to the dispatcher so that it can inform the platforms that have been assigned tasks. Those platforms may choose to accept or reject tasks (either automatically or via an operator query), and also inform the dispatcher about the progress of a task, including when a task has failed or has been successfully completed. Task allocation does not distinguish any "pre-planning" phase, but rather operates continuously on the evolving situation and set of tasks: upon any receipt of rejection, failure, or success messages, or upon receipt of new information about tasks and resources (e.g., the changing location of a platform over time), the task allocation component is triggered to update its plan and any allocation changes are sent to the platforms. The goal is to execute as many tasks as possible, and to ensure that those tasks that are executed as redundantly as possible, within the constraints of available platforms and possible rejection by operators (which may force tasks to be reassigned elsewhere).

To implement task allocation, MTIP uses an agent-based approach, which we have chosen over typical planning or optimization algorithms for three key reasons: First, with an appropriate choice of algorithm, agent-based planning executes rapidly and its current state always provides a viable (though possibly still improving) allocation plan, which means that it can be safely used in real-time situations with strict time constraints. The cost of this speed is that task allocations will generally be somewhat sub-optimal (possibly including some tasks being unable to be served at all), though this need not be a significant cost (as will be shown in Section 6).

Second, our agent-based approach is also well-suited for rapid dynamic replanning, which in MTIP is triggered by runtime monitoring of execution in response to changes in tasks, priorities, platform routes, and availability of platform sensors. With an agent-based approach, replanning need not begin from scratch, but can be implemented simply by modifying the agent representation and allowing the agents to continue interacting. Third, although currently all agents are located within a single server, the agent-based approach provides a natural path toward further decentralization whereby platform agents and nearby task agents are proxied to particular platform agents or mirrored and synchronized across multiple platforms. Such distribution of agents

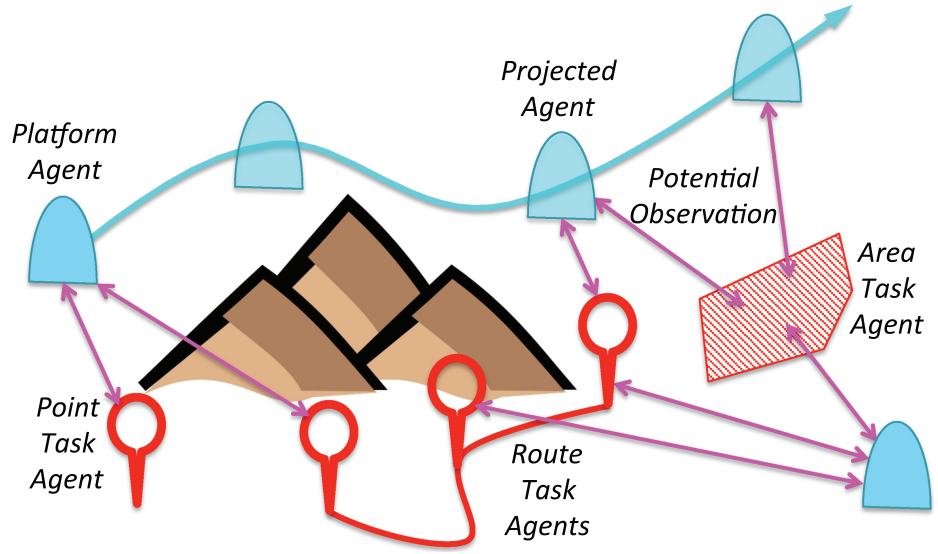


Fig. 5. In MTIP's agent-based task allocation, agents for current platforms (light blue) and projections (translucent blue) along the platform's anticipated trajectory communicate (purple arrows) with task agents (red) within line-of-sight and sensor range limitations to determine which tasks will be assigned to which platforms and on which segments of their anticipated route.

would then enable platform clients to dynamically reallocate tasks amongst themselves, even when communication with the task allocator is limited or unavailable, yet still allow fast and efficient planning under normal communication conditions.

4.1 Agent-Based Representation

The MTIP agent-based task allocation mechanism uses two classes of interacting agents: platform agents and task agents (Figure 5). For each airborne sensor resource,[2] an agent is created for its current position along with a set of projected agents at intervals of a parametrically specified resolution along its anticipated future trajectory (similar to the polyagents approach presented in Parunak and Brueckner (2007)). Likewise, one or more agents is created for each task: point tasks and other tasks with small extent in space and time are represented by a single agent; tasks with larger extent can be broken up into intervals (routes) or tiles (areas and circles) at a parametrically specified resolution.

A bipartite network between tasks and platform agents is then formed by creating an edge between each task agent and every current or projected platform agent that could plausibly carry out that task. This is determined by sensor range (e.g., close enough to achieve a specified camera resolution) and by line of sight, as computed from the planetary horizon and GIS terrain data—in particular, our implementation accomplishes this with the aid of NASA's WorldWind GIS environment (Bel et al. 2007). The agents then communicate with one another along the edges of the bipartite network in order to determine which tasks will be allocated to which platforms and on which segments of their anticipated route.

[2]The work presented in this article assumes one sensor per platform: this can be readily generalized to multiple sensors per platform by the use of multiple agents, one per sensor, which share the trajectory of their joint platform.

Each platform agent and projected agent is responsible for computing the sensor tasks that it is best equipped to perform. Finally, the collection of individual agents' decisions produces an overall task allocation for the whole system, i.e., a mapping between resources and tasks. Note that the cardinality of resources and tasks in a task allocation is many-to-many, where a single platform agent may be assigned many tasks and a single task may be assigned to multiple platform agents.

4.2 Stable Task Allocation Algorithm

For actually allocating tasks to platforms and adjusting those allocations as the situation changes, MTIP must balance the concerns of efficiency and stability. On the one hand, it is obviously important to assign tasks in an efficient and timely manner, in order to ensure the coverage of as many targets as possible by assigning each task to the best set of platforms available to cover it at the earliest and best opportunities. On the other hand, reassigning tasks to different platforms adds its own costs and inefficiencies in the form of additional network traffic, increased risk to the mission (that opportunities will be lost during reassignment or that re-assignment will fail), and additional cognitive load for the platform operator.

We have thus developed a stable task allocation algorithm that balances between these two concerns.[3] The basic idea behind our algorithm is that it acts as a negotiation between task agents and platform agents, in which each tries to improve its current situation. Each platform agent considers its set of neighboring task agents and chooses whether to send a coverage proposal to one (possibly dropping a currently covered task in order to do so, if the potential gain is big enough); task agents, in turn, consider their set of incoming proposals and choose which is the best option to accept. Iterating this process eventually produces a stable allocation in which no more proposals are being made, as we prove below.

Adaptation is done the same way as initial planning: as this agent-based algorithm is fairly fast to execute, task allocations are adjusted simply by adjusting the situation model (e.g., updating airborne platform positions, adding tasks, changing task position or priority), at which point the platform agents may choose to begin making new proposals, adjusting the allocation as necessary to match the changing situation. Inefficient reassignments are controlled by inclusion of state projection into the proposal process: in particular, when considering whether to drop a currently covered task in order to make a new proposal, the proposed task is compared as it would be ranked if accepted, rather than in its current state.

Figures 6 and 7 show detailed pseudocode for our algorithm, the first for platform agents, the second for task agents (currently in MTIP, the code is implemented in a mixture of Java and the Protelis aggregate programming framework (Pianini et al. 2015; Beal et al. 2015)).

In this algorithm, each device tracks two state variables—one for the current assignment and one for proposed changes—plus information on the platform or task it represents. In particular, platform agents use variable `assignment` to track the set of tasks they are currently assigned and `proposal` as a slot that can hold one proposal of a pair `[platform, task]` proposing adding a task to its assignment. Complementarily, task agents use variable `assignedBy` to track the set of platform agents (current or virtual) assigned to execute the task, and `acceptedProposal` as a slot to report a proposal from a platform that the agent wishes to accept.

Agents execute in rounds: in this case, we consider synchronous execution, first all platform agents and then all task agents, though the synchrony is not necessary. In each round, platform agents begin by gathering information from their task agent neighbors: how many platform agents

[3]Our prior work in Beal et al. (2016a) and Beal et al. (2016b) used a naive algorithm that could produce unstable allocations in certain situations.

```
// Platform Agent:
variables:
  assignment = []
  proposal = []
  type = 'Platform'
  platform = getLocalPlatform()

loop:
  // Collect pairs of [task, # platform agents assigned] from neighbors
  taskCover = nbrMap(nbrs.filterBy(type=='Task'), (nbr) -> [nbr.task,nbr.assignedBy.size()])
  // Check if agent has a proposal accepted by any neighbor
  acceptedProposal = (proposal != []) and
                     anyNeighbor( (nbr) -> proposal==nbr.acceptedProposal )
  // If accepted proposal is not already in the assignment, add it
  if acceptedProposal and !assignment.contains( proposal[1] )
    assignment.append(proposal[1];
  // Either way, reset proposal variable before recomputing it
  proposal = []

  // Filter out tasks that are not assignable to this platform
  candidates = taskCover.keys.filter( (t) -> platform.validTask(t) )
  // Remove any invalid tasks from current assignment
  assignment = assignment.intersection(candidates)
  // Remove current tasks from candidates for new assignment
  candidates = candidates.subtract(assignment)

  // Assignment priority is lexicographic on # platform agents assigned,
  // then task priority, and finally a stable tiebreaker
  metric = (t) -> [taskCover.get(t), t.getPriority(), t.hashCode()]
  candidates.sortBy(metric)
  assignment.sortBy(metric)

  // Initial budget is the seconds of the entire planning interval
  budget = planningInterval()

  // If there are any candidates, consider the first as a new proposal
  if candidates.size() > 0
    c = candidates[0];
    // Projected state: when inserting, add one to effective candidate coverage
    priorityOverCandidate = assignment.filter( (t) -> metric(t) > (metric(c) + [1 0 0]) )
    cost = sum(priorityOverCandidate.map( (t) -> t.secondsNeeded() ))
    // If there is space enough to insert, do so, pre-allocating budget
    if budget >= cost + c.secondsNeeded()
      proposal = [platform,c]
      budget = budget - c.secondsNeeded()

  // Finally, trim anything in the assignment that does not fit in the budget
  for t in assignment
    budget = budget - t.secondsNeeded()
    if budget < 0
      assignment.remove(t)
```

Fig. 6. Pseudocode for MTIP platform agents.

cover each task and whether their current proposal (if they have one) has been accepted. When a proposal is accepted, the platform agent adds it to its `assignment`. The platform agent then goes on to adjust its `assignment` and `proposal` variables, based on the updated situation. First, it filters the set of neighboring tasks to remove any "invalid" tasks that cannot be serviced from this platform, e.g., because they require sensors that are not available or have been rejected by the platform's pilot. If any tasks in its `assignment` are no longer valid neighbors (e.g., having been rejected or left line-of-sight), those tasks are removed.

In order to determine which tasks should be serviced within the limited budget of sensor time available, tasks are sorted by a metric $M(C_T, P_T, U_T)$ that ranks each task in three ways:

(1) Number of covering platforms C_T: a task with less platforms covering it will always be assigned before a task with more platforms covering it.

```
// Sensor Task Agent:
variables:
  assignedBy = []
  acceptedProposal = []
  type = 'Task'
  task = getLocalTask()

loop:
  // Collect set of neighbors assigned to this task
  assignedBy = nbrValues(nbrs.filterBy(type=='Platform'),
                         (nbr) -> nbr.assignment.contains(task) )
  // Collect set of all non-null proposals at platform neighbors
  proposals = nbrValues(nbrs.filterBy(type=='Platform'), (nbr) -> nbr.proposal )
  // Find the subset of proposals relevant to this agent
  validProposals = proposals.filter( (p) -> p[1] == task )

  // If there are any proposals for this task, assign them stable hash codes and pick lowest
  if(validProposals.size() > 0)
    validProposals.sortBy(validProposals.map( (p) -> p.hashCode()));
    acceptedProposal = validProposals[0];
  else // Otherwise, just clear acceptedProposal variable
    acceptedProposal = [];
```

Fig. 7. Pseudocode for MTIP sensor task agents.

(2) Task priority P_T: more important tasks (indicated by smaller numerical priority) are served before less important tasks.
(3) Arbitrary unique identifier U_T: ties are broken stably with a task unique identifier so that they will be broken the same way from round to round.

In other words, the system considers it more important to have one platform cover a low-importance task than to put redundant coverage on a high-important task; and for tasks that have equal redundancy and importance, the system just wants to keep a stable order on which tasks are preferred.

In order to provide monotonicity in decision making and to ensure stability across the collection of agents, only the one minimally ranked candidate task (if any) is considered for addition at any time, and it is evaluated as though its coverage is one higher than actually observed, i.e., as it will appear if the proposal is accepted. Evaluating a task as it will appear when accepted means that if accepting a task causes another task currently assigned to be dropped, then an agent will have no reason to reverse that decision. The task agent determines where this task would fall in the current assignment and if there is sufficient time budget to include it (where the estimated time to complete the task is determined as a function of task and sensor properties). If there is sufficient time available in the budget, the available budget is reduced accordingly and a pair of `[platform,task]` is placed into `proposal` to see if the task agent will accept it. Finally, the `assignment` is compared against the (possibly reduced) budget, which may result in the worst-ranked current tasks being dropped for lack of available sensor time.

Task agents run a complementary algorithm that is simpler because there is no constraint on the number of platform agents that can redundantly cover a given task. In each round, task agents begin by gathering information from their platform agent neighbors: which platform agents currently have the task in their assignment (updating `assignedBy`) and the current set of proposals to cover this task. If there are any such proposals, the agent then sorts them by an arbitrary unique identifier in order to stably break ties in choosing which proposal to accept.

The algorithm is executed until it converges or until a set number of rounds or timeouts have elapsed, at which point any changes to the set of task assignments is dispatched to platform clients to allow them to update their execution plans. When the situation is updated, the algorithm executes again, adjusting the distribution as necessary within the constraints of the algorithm.

Note that we are not asserting optimality for this algorithm, but rather its sufficiency to produce fast and effective task allocation by selecting one of the many reasonable points of balance between adaptivity and stability. We prove its balance between adaptivity and stability by proving that this algorithm is self-stabilizing (Schneider 1993; Dolev 2000), meaning that when run, starting from any arbitrary initial state or perturbation, it always converges to a correct state—here, defined as obtaining a task assignment that can be executed and that no agent wishes to change.

THEOREM 1. *The MTIP task allocation algorithm is self-stabilizing in within $O(T \cdot D_T)$ rounds, where T is the number of tasks and D_T is the maximum degree of any task agent, i.e., the maximum number of neighbors for any task agent.*

PROOF. Without loss of generality, we assume a connected network of agents: in the case of a disconnected network, each connected component will resolve independently according to the same logic. We also assume, without loss of generality, that every task can be executed within the planning interval, meaning it is possible to assign.

Consider starting the algorithm from an arbitrary, possibly invalid state. In the first round, all invalid tasks are removed from platform agent `assignment` variables, and `proposal` variables become either empty or filled with a potentially valid proposal. Subsequently, in the second half of the first round, all invalid platforms are removed from task agent `assignedBy` variables and `acceptedProposal` variables become either empty or filled with a potentially valid proposal. Thereafter, we need only show that the algorithm converges to a stable assignment.

To show this, let us create an "assignment quality metric" $Q(T_1, T_2, \dots)$ over the whole set of tasks based on the platform agent metric, $M(C_T, P_T, U_T)$. Tasks are sorted by P_T and U_T, so that T_1 is the task with the lowest $[P_T, U_T]$, T_2 the second-lowest, etc., and $Q(T_1, T_2, \dots)$ is equal to the vector $[C_{T_1}, C_{T_2}, \dots]$. Any two sets of platform agent `assignment` values, then, can be compared lexicographically to determine which is better, based on giving more coverage to more important tasks.

Consider, first, the case in which all platform agents make proposals that do not cause them to drop anything from their current assignments. Given a set of such proposals, the quality metric Q is guaranteed to increase within one round: a task agent presented with a set of proposals will always accept one of them, and such an accepted proposal will be added to a platform agent's `assignment` in the next round. Thus, C_{T_i} will increase for at least one task (this minimum occurs if all platform agents propose to the same task agent). Since C_{T_i} increases monotonically, there are no more than $T \cdot D_T$ steps of monotonic increase possible before no proposals will be made and the algorithm stabilizes.

Finally, we generalize to the case where a platform agent makes a proposal that causes it to drop tasks from its current assignment. In this case, Q may drop temporarily, but is guaranteed to eventually increase again, as every execution falls into one of three cases:

- —The agent's proposal is accepted: Q must increase after acceptance because the proposal could not be made unless accepting it would increase coverage of a better-ranked task than the tasks that were dropped.
- —The agent's proposal is not accepted, in which case each dropped task is eventually:
  - —not covered by the same agent because its budget is consumed by a better-ranked task, increasing Q.
  - —proposed and accepted again by the same agent: a task that was dropped is a task that once fit in the budget, so the budget is guaranteed to be able to accommodate it unless a better-ranked task is assigned. This neutral state still implies to an increase in Q: if this platform agent's proposal was not accepted by the task agent it was proposed to,

Infrastructure Class	# Objects	# UAVs	UAV Base (Lat/Lon)
Airports	25	2	37.625°, -122.383°
Cell phone towers	251	3	37.418°, -121.883°
Dams	152	2	37.941°, -122.261°
Fire Departments	160	3	37.779°, -122.390°
Heliports	28	1	38.466°, -121.423°
Hospitals	28	1	37.432°, -122.178°
Military Installations	8	1	37.404°, -122.028°
Power Plants	14	1	37.219°, -121.747°
Total	666	14	

Fig. 8. The San Francisco disaster response scenario considers eight classes of critical infrastructure: for each class of critical infrastructure, we use a publicly available GIS dataset for survey targets and specify a manually-planned set of UAV routes to provide coverage of all of the targets in that class.

that means the task agent accepted some other platform agent's proposal instead, which must lead to an increased Q.

In all cases, we are again guaranteed a monotonic increase in Q associated with each round of proposals, though the time at which the increase is able to actually be observed may be deferred. □

5 SENSOR SHARING POTENTIAL

In order to validate the MTIP sensor-sharing concept and to test our system implementation, we created a simulation of a disaster response scenario using publicly available GIS data about critical infrastructure.[4] Evaluating this scenario, we find that there is a high potential for sensor sharing between different classes of infrastructure, providing evidence in support of the general sensor-sharing concept. We further find that our MTIP architecture implementation produces fast and effective task allocations in this scenario, as expected.

5.1 San Francisco Disaster Response Scenario

For evaluation of MTIP, we consider a scenario in which a major earthquake has just hit the San Francisco Bay area and a number of different disaster response teams are attempting to assess damage with UAVs, prefatory to preparing a response.

In particular, we consider eight classes of critical infrastructure, as listed in Figure 8. Given typical organization structures and the specialization needed to deal with managing different classes of infrastructure, it is reasonable to assume that each class of critical infrastructure will have its own dedicated response team or teams. Thus, for instance, a disaster response team evaluating the integrity of dams will likely be different personnel in a different organization than one that is attempting to ensure that the wireless infrastructure on cell towers is operational.

For our scenario to have realistic complexity and distribution of survey targets, we populate the set of survey targets for each class of infrastructure from publicly available GIS datasets, filtered to consider only locations in the range latitude 37.0° to 38.5° and longitude −123.0° to −121.0° (roughly, North/South from Santa Cruz to Sonoma County and inland through the Sacramento and San Joaquin Valleys). As these tasks are assumed to be for an initial assessment, all are assigned the same priority, mode, and resolution requirements, and have no specific timing requirements. Furthermore, since the purpose of our test is to investigate the efficacy of cooperative task allocation and not the user interface, in our experiments, we supply this survey target information as

[4]The validation of sensor-sharing presented in this article differs from that in Beal et al. (2016a) in that we have corrected a minor bug in line-of-sight calculations from GIS data, making visibility slightly more strict, and UAV routes have been adjusted slightly in compensation in order to maintain the stated preconditions.

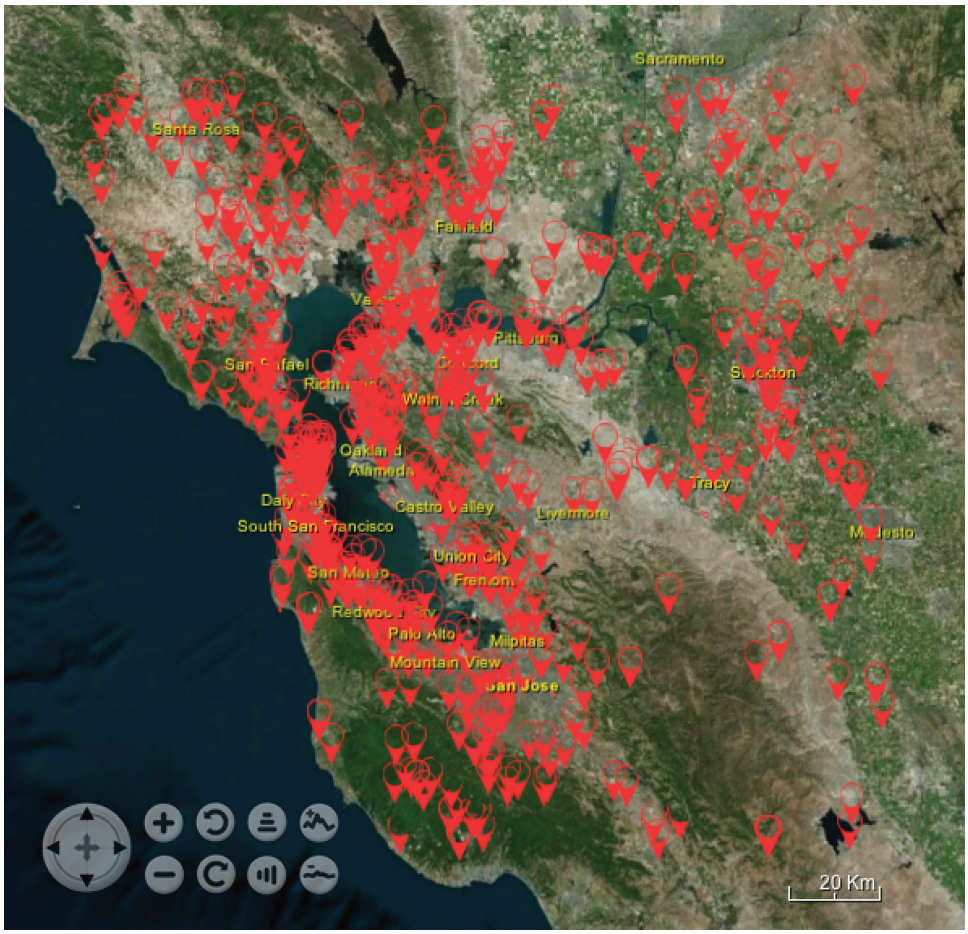


Fig. 9. Set of all 666 critical infrastructure survey targets in our scenario, indicated by standard emergency management "point of interest" icons or polygonal areas.

explicit KML survey target lists, rather than implicitly through other TAK interfaces. A summary of all eight classes of critical infrastructure survey targets is shown in Figure 9.

For each class of critical infrastructure, we assumed one of the survey targets is the operating base for a disaster response team and manually planned a set survey route for 1-3 UAVs from that base to cover the rest of the survey targets in the class, with the number of routes depending on the number and geographical dispersion of targets to be surveyed. Manual planning was carried out for each infrastructure class independently, on a map showing only survey targets and UAV routes for that infrastructure class. The preferred altitude planned for surveys is 500 meters, but can reach as high as a maximum of 1,500 meters as the UAVs pass over the various coastal mountain ranges. Figure 10 shows the set of all 14 UAV routes, with each color indicating a different infrastructure class, while Figure 11 shows an example of how UAVs cover a set of infrastructure, in the form of the single UAV route planned for surveying the 14 critical infrastructure power plants. Hereafter, when we refer to a "set of UAVs," it means the group of UAVs associated with a particular infrastructure class.

Notice that the routes are strongly affected by infrastructure class. Some of the UAV routes are relatively smooth and simple, such as the UAV routes for airports (orange lines in Figure 10), which

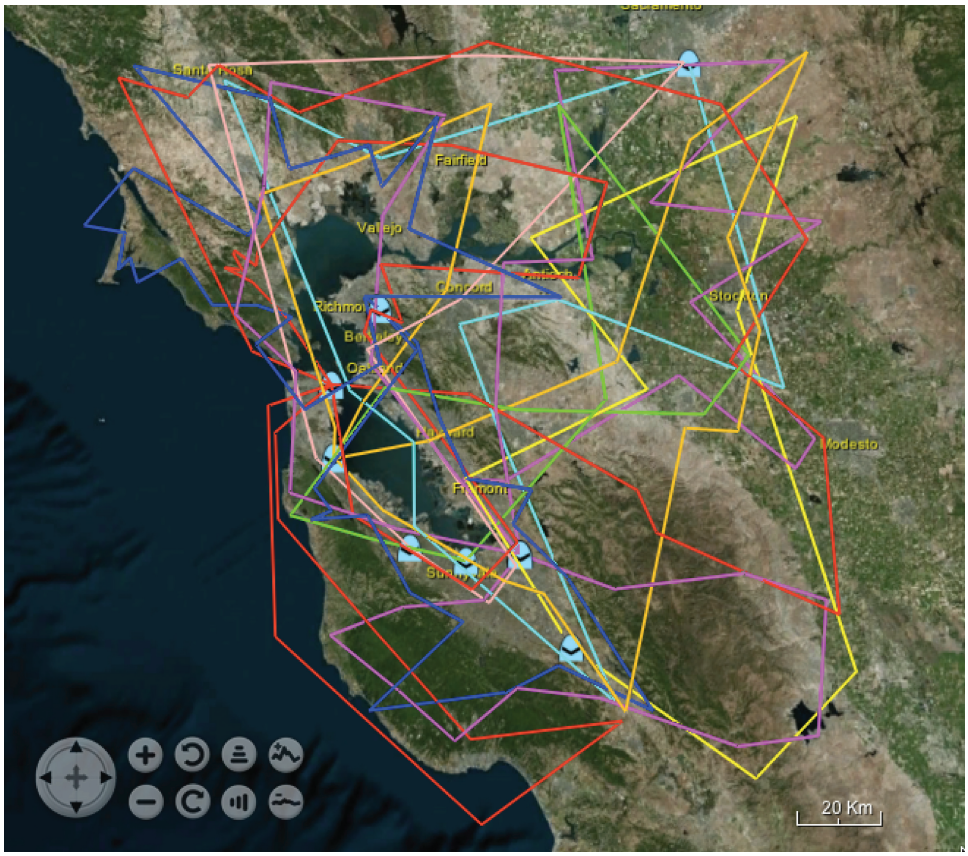


Fig. 10. Set of all 14 UAV routes, with each color indicating a different infrastructure class and air asset icons indicating class bases.

tend to be in flat and unobstructed areas visible from far away. Others, such as the UAV routes for dams (blue lines in Figure 10), are much more intricate, since many dams are deep in mountainous valleys and can, thus, only be seen from certain perspectives.

For survey UAVs, we chose to consider the Boeing ScanEagle, a small high-endurance UAV with an approximately 3-meter wingspan and 20kg mass, used by a number of military and civilian organizations around the world. Based on its published specifications, we assume a flight speed of 40m/s, a 6,000 meter operating ceiling, and up to 24 hours endurance (though the planned survey routes of our scenario are all less than four hours). Standard equipment options for ScanEagles include a high-resolution imager with up to 170x zoom, so we also assume an effective visual survey range of up to 20 kilometers for initial damage assessment.

For planning purposes the survey routes are quantized into projections at intervals of 5 minutes between projections (i.e., for UAV locations 12km apart along the planned route). Finally, each UAV is assumed to be able to adequately survey one target every 20 seconds: with a three site per minute sensor, this implies a maximum of 15 tasks per planning location.

All told, the set of planned survey routes spans nearly 500 planning locations, with the average mission planned to survey approximately 50 targets over a period of approximately two and a half

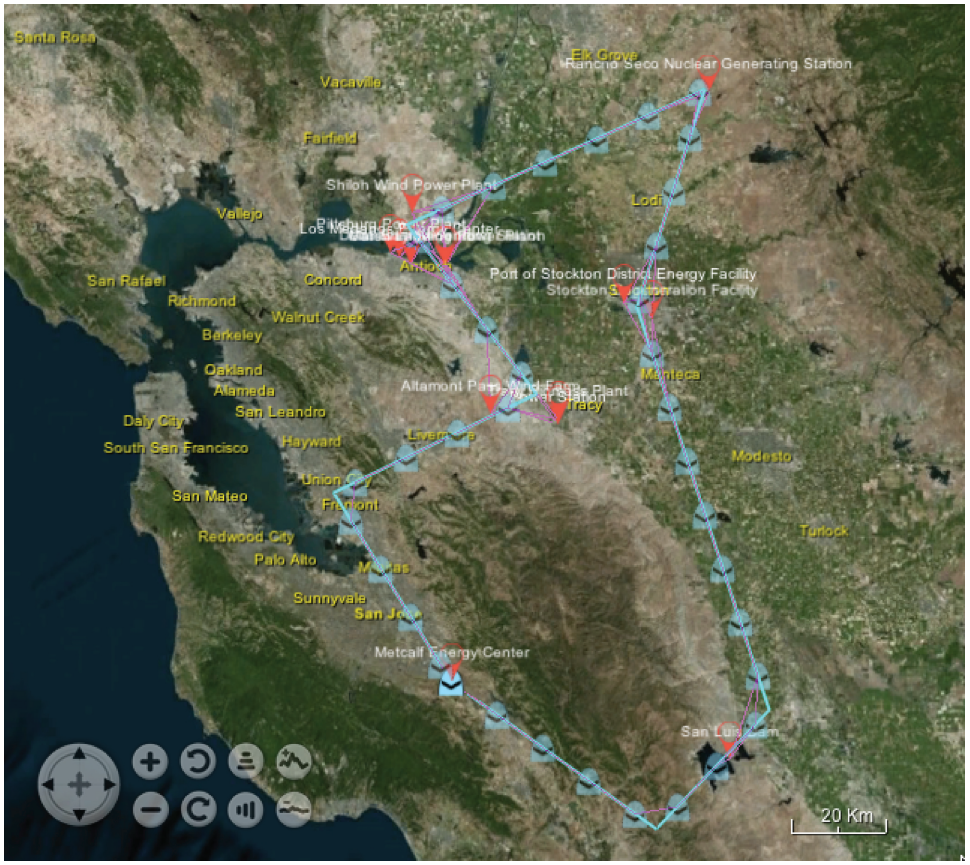


Fig. 11. UAV route example, of a route for a single UAV planned for surveying the set of 14 power plants in our collection of critical infrastructure sites. UAV route and positions are blue, with the base position indicated by a standard airborne asset icon and projected positions indicated by translucent icons, while survey targets are red emergency management "point of interest" icons.

hours. Even for the densest planned route, surveying dams, the mean number of targets per planning location is only 2.81, meaning that under these conditions, most time is indeed spent in transit rather than on mission execution, and that there is ample opportunity for sensor sharing.

This is, of course, dependent on our assumptions: slower surveys or faster platforms will reduce the opportunity for sharing, and differences in the number or distribution of targets will impact sharing opportunities as well. These numbers, however, are drawn from real GIS data and UAV specifications. Notably, we have intentionally chosen a region with a high degree of terrain and settlement heterogeneity, including dense urban environments, suburban and exurban sprawl, agricultural lands, coastline, and mountainous wilderness. Likewise, the infrastructure sets we have chosen are highly heterogeneous in both density and spatial distribution. We thus argue that it is reasonable to expect qualitatively similar sparseness to be the case for many real-world deployments, and that good performance in our presented scenario indicates that we should expect good performance in such real-world deployments as well.

5.2 Potential for Sensor Sharing

We begin by assessing the theoretical potential for sensor sharing across different organizations, regardless of our particular implementation of this capability in the MTIP platform. The key hypothesis of MTIP is that commonalities in the physical and human geographical environment are likely to mean that an aerial platform route planned for one set of sensor tasks will likely pass close to other classes of sensor tasks as well. Critical infrastructure, for example, is not placed arbitrarily, but tends to be highly constrained by the distribution of population and major physical features such as rivers and mountains. In our scenario, we would thus expect that any given set of UAVs is likely to have good visibility on many of the survey targets for other classes of infrastructure, despite being planned independently. This is particularly likely to be true in the case of UAV routes that have been planned for the more numerous and pervasive classes of infrastructure: Figure 12, for example, shows that between them, the three UAV routes planned for covering cell phone towers also covers all airports and power plants.

To evaluate this hypothesis, we ran two experiments in which we ran the MTIP system with no constraints on the number of targets that could be covered for a given route segment. In this configuration, every survey target is assigned to every route segment from which it is visible and within sensor range, thereby showing the maximum potential for sensor sharing.

For the first test, we ran the planned UAV routes for each class of infrastructure individually against the survey targets of each class of infrastructure. Figure 13(a) shows the fraction of survey targets in each infrastructure class that can be observed for each of the 64 pairings. On the diagonal, of course, every set of UAVs perfectly covers the infrastructure class for which its routes were planned. UAVs planned to cover highly dispersed classes of infrastructure, such as cell phone towers and fire departments, cover most other infrastructure classes quite well. Even the most small and geographically constrained classes of infrastructure, however, such as power plants and military installations, turn out to provide coverage of a fairly high proportion of most other classes of infrastructure.

Complementarily, for the second test we ran a "leave one out" scenario in which we ran the survey targets for each infrastructure class against the set of planned UAV routes for all other classes of infrastructure, showing how well an infrastructure class might potentially be covered by relying entirely on sensor sharing from the UAVs of other teams. Figure 13(b) shows the fraction of survey targets in each infrastructure class that can be observed by the UAV routes planned for the other classes. Here we find that 98% of all targets are covered, with every class of survey targets better than 90% covered, and, in fact, only three not 100% covered. The targets that are not covered are one cell tower, nine dams, and four fire stations, nearly all of which are in isolated mountain or coastal valleys.

These experiments, thus, indicate that the MTIP sensor sharing concept is indeed likely to be valid and of significant use in real-world tasks such as disaster response, in which there is significant geographical correlation in the likely interests of different sensor users.

6 EFFICACY AND ADAPTATION OF MTIP SENSOR SHARING

Having established that most targets can potentially be covered by multiple sets of UAVs, we now evaluate the efficacy of our MTIP implementation in allocating survey targets to UAVs with constrained time resources and in adapting that allocation over time.[5] First, we evaluate the efficacy of MTIP when platforms follow their planned flight paths, thus requiring no adaptation over the

[5]The experimental results presented in this article differ from those in Beal et al. (2016a) and Beal et al. (2016b) in three ways: line-of-sight calculations and UAV routes are adjusted as described in the previous section, we use the algorithm presented in Section 4.2, and we gather additional data and perform additional analyses.

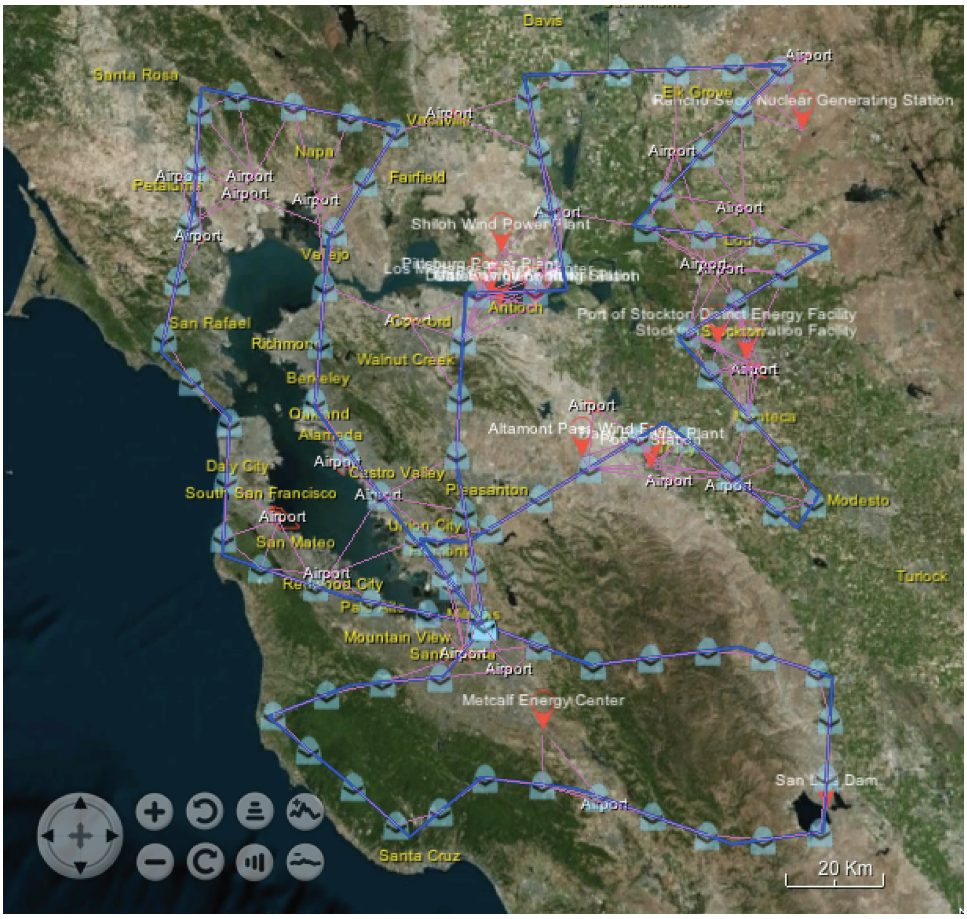
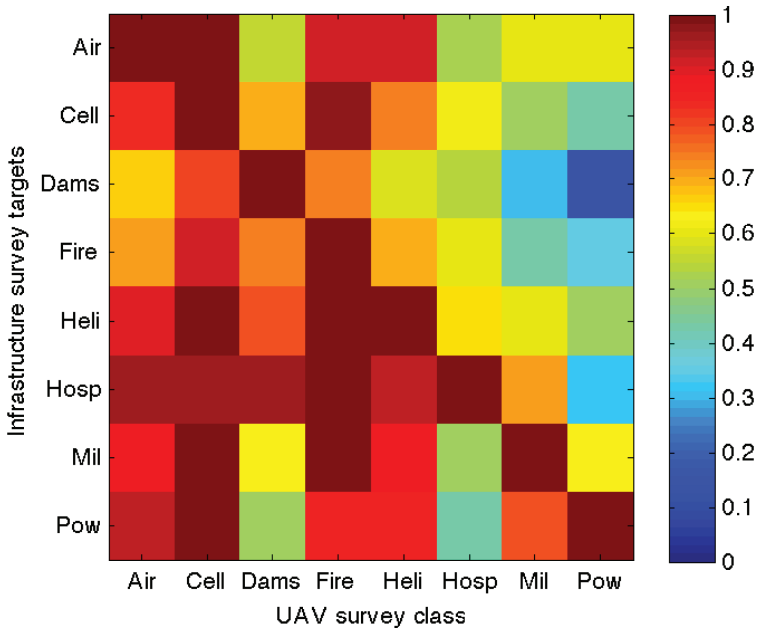


Fig. 12. A shared geographical environment makes it likely that UAVs planned for one task will provide good coverage for others, as well: for example, UAVs tasked to survey cell towers in the San Francisco area also cover all of the airports and power plants. The planned UAV path is shown as a blue line with blue air-asset tactical symbols for UAV positions (solid for current position, translucent for projected future positions), survey targets are red, and magenta lines show line-of-sight relations between UAV positions and survey targets.

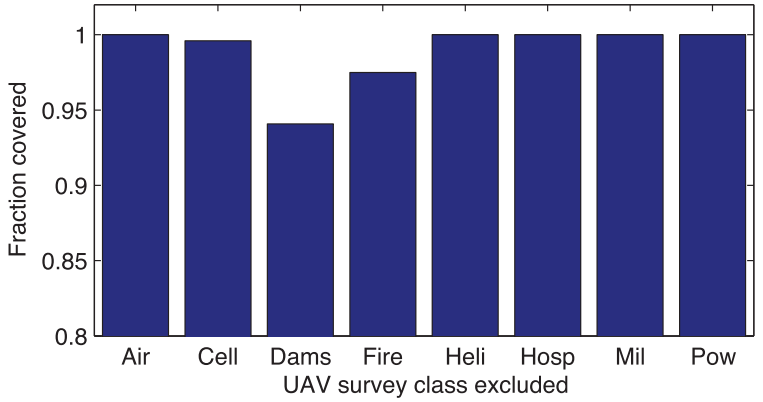
course of a mission. Following this, we evaluate the ability of MTIP to use sensor-sharing to effectively adapt to changing circumstances by having simulated platforms execute missions in real time against the running system, with various degrees of unpredicted divergence from each platform's initial planned flight route.

6.1 Sensor Sharing with Predictable Flight Paths

For our first set of experiments, we run the full MTIP system in an emulated network environment, in which each UAV is represented by a dummy process implementing a simplified version of a Marti platform client, and with the previously stated limit of 15 survey targets per UAV planning location. Each simulated UAV launches a server that receives and responds to task allocation messages following the MTIP protocol, with each UAV accepting all tasks that it is sent by MTIP.



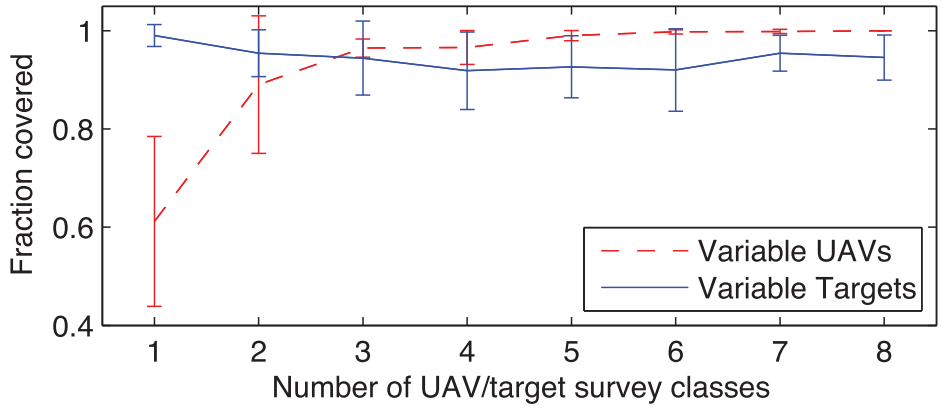
(a) Potential for groups of UAVs to cover groups of survey targets.



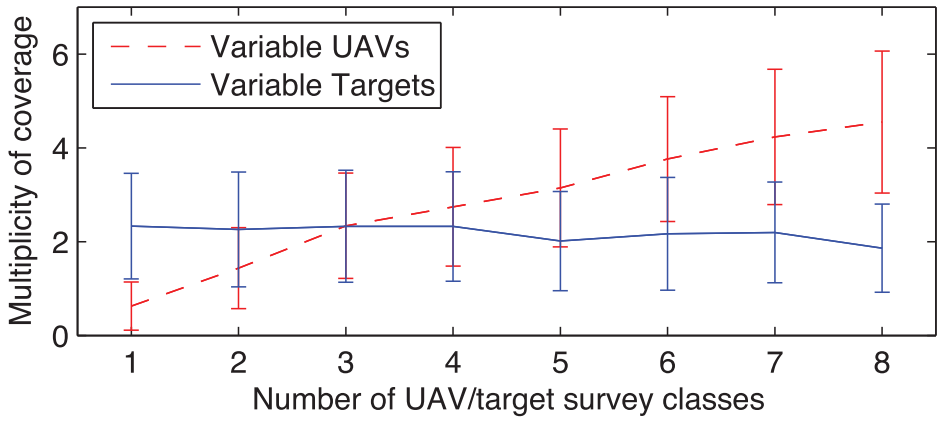
(b) Potential for survey targets to be entirely covered by other UAVs

Fig. 13. Unconstrained task assignment shows sensor sharing potential (a) for individual sets of UAVs and classes of survey targets, and (b) for a class of targets to be covered by some UAV from any other class.

We evaluate both the efficacy and scalability of the system through two tests with randomly selected subsets of the system. For the first test, we randomly select u sets of UAVs and execute MTIP to plan for surveying all eight sets of critical infrastructure, ranging u from 1 to 8 and running 10 trials for each condition. The second test fixes the number of randomly selected sets of UAVs to $u = 3$ and executes MTIP to plan for surveying t randomly selected sets of survey targets, ranging t from 1 to 8 and running 10 trials for each condition. The UAVs are then assumed to faithfully execute the plans they have been assigned, and the result analyzed to determine how well MTIP can realize the theoretical potential for sensor sharing under ideal conditions.



(a) Fraction of targets covered



(b) UAVs per target

Fig. 14. Performance of MTIP sensor sharing with respect to variable numbers of UAVs (u) or survey targets (t): (a) even with few UAVs and many targets, the vast majority of targets are covered by at least one UAV, and (b) most targets receive coverage from multiple UAVs. All graphs show the mean over 10 trials, ±1 standard deviation.

Figure 14 summarizes the results of these two experiments. Figure 14(a) shows that MTIP effectively implements sensor sharing even with many consumers and few airborne resources: just two sets of UAVs is enough to reliably cover the vast majority of survey targets and five UAV sets are enough to reliably cover all but a few particularly difficult to see survey targets. Complementarily, with a fixed number of sets of UAVs, performance degrades by only a small amount as the most densely populated areas begin to saturate UAV sensor sharing capacity. Indeed, as Figure 14(b) shows, there is enough excess sensor capacity to allow most targets to be surveyed by multiple UAVs. This indicates that one of the ways that we can expect mission resilience to be improved is by "backing up" UAVs with spare sensor capacity on other UAVs that fly nearby routes. This conclusion is reinforced by examining the distribution of multiple coverage with respect to the number of UAVs, as shown in Figure 15: here, we find that the number of UAVs covering each target is distributed in a remarkably smooth and balanced manner, indicating that the mean UAVs per target does indeed indicate a high potential for covering most individual targets multiple times.

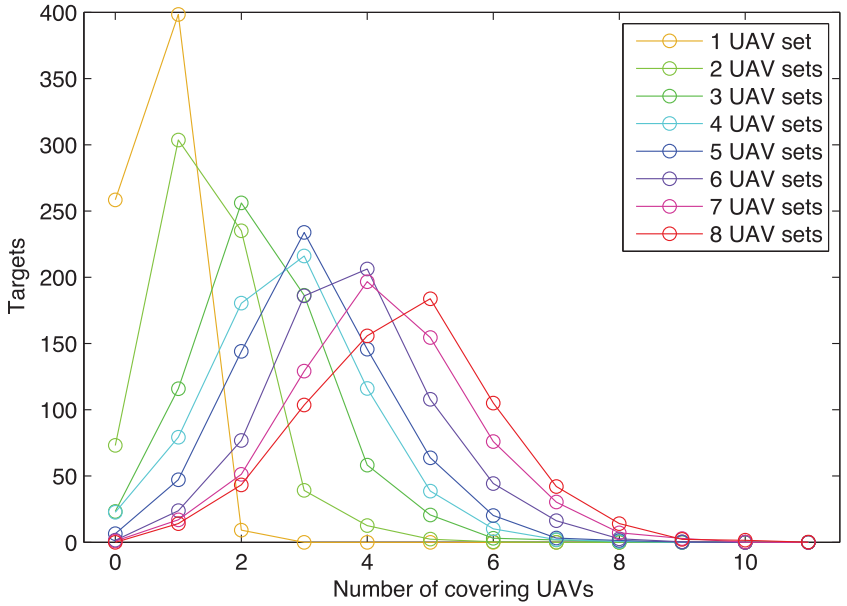


Fig. 15. The mean distribution of the number of UAVs covering each target over 10 trials with varying number of UAVs u. The color indicates the number of sets of UAVs u, ranging in hue from 1 (yellow) to 8 (red).

The time required for our agent-based system to achieve these results is reasonable. Figure 16 shows the time required for MTIP to make and dispatch allocation plans for these complex scenarios on a small portable COTS machine (MacBook Air with 1.7GHz Intel Core i7 and 8GB RAM). We find that time scales approximately linearly with both the number of UAVs and the number of targets, remaining quite reasonable, even for cases comprising more than 1,000 interacting agents. Note also that in the current prototype implementation, a significant majority of this time is spent in computation of terrain intersections via a fairly simple and unoptimized method and in dispatching thousands of assignments of survey targets to UAVs using a separate HTTP session for each assignment. There is, thus, much room for improvement to enable even larger-scale sensor sharing.

6.2 Adaptation During Mission Execution

In order to test the behavior of MTIP in a dynamically evolving mission environment, we simulated each UAV as an independent process, running in real-time with its own set of threads independent of the operation of the MTIP system. Since the UAV imagers are assumed to take 20 seconds to effectively survey a site, UAV operations are simulated in 20 second steps. At each step the UAV moves 20 seconds (800 meters) along its flight path, sends its new position to MTIP (as a standard Cursor on Target (CoT) packet), and attempts to select a task site for imaging. From its list of assigned tasks, the UAV finds the set of tasks that are both close enough for imaging (less than 20km from the UAV) and within line of sight (i.e., not blocked by terrain). If any of these task sites has not yet been imaged, one of the non-imaged task sites is selected arbitrarily for imaging. Otherwise, the task site that has been imaged the fewest times is selected for additional imaging (again breaking ties arbitrarily), and if no task sites are nearby and visible, then the sensor is idle for that timestep.

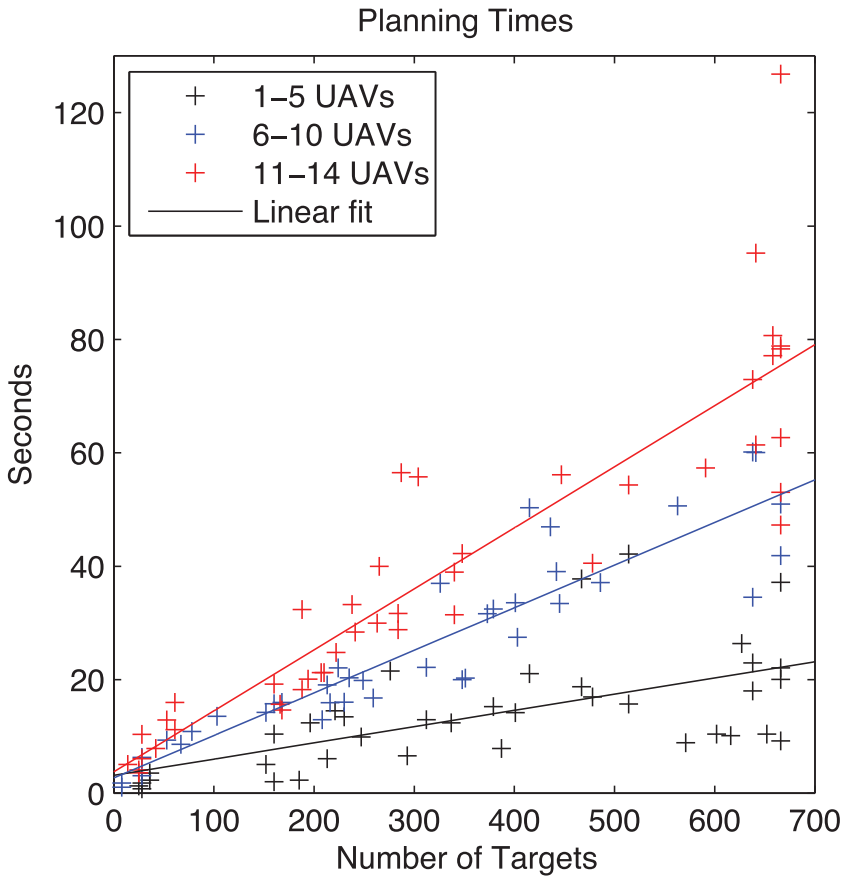


Fig. 16. The time required for MTIP to make and dispatch allocation plans scales approximately linearly with both the number of UAVs and the number of targets.

In order both to further stress the system in our experiments and to make their running time tractable, however, we actually run most of each mission at a greatly accelerated rate. For each simulation, we first run the system for 120 seconds in real time in order to allow for MTIP initialization, the first round of planning, and assignment dispatch. Thereafter, we shift the simulated platforms to run at a 100:1 rate, taking a 20-second simulated step every 200 milliseconds of real time.

A dynamic environment of adaptation challenges is created by random deletion of waypoints from each UAV's planned route: the route to be flown is created by taking the planned route and giving each waypoint an independent probability of being deleted d (except the first and last waypoints at the base where the UAV takes off and lands). An example of random waypoint deletion is shown in Figure 17: note that the change of flight plan leaves a number of tasks unable to be completed. Random waypoint deletions thus create a situation similar to what might happen if UAV operators are receiving emergency requests that lead them to skip planned survey sites and instead send their UAVs directly to sites that had been scheduled for later observation.

We evaluate the adaptivity of the MTIP system by means of two experiments. In the first experiment, we evaluate the impact of various levels of dynamism by varying the rate of deletion

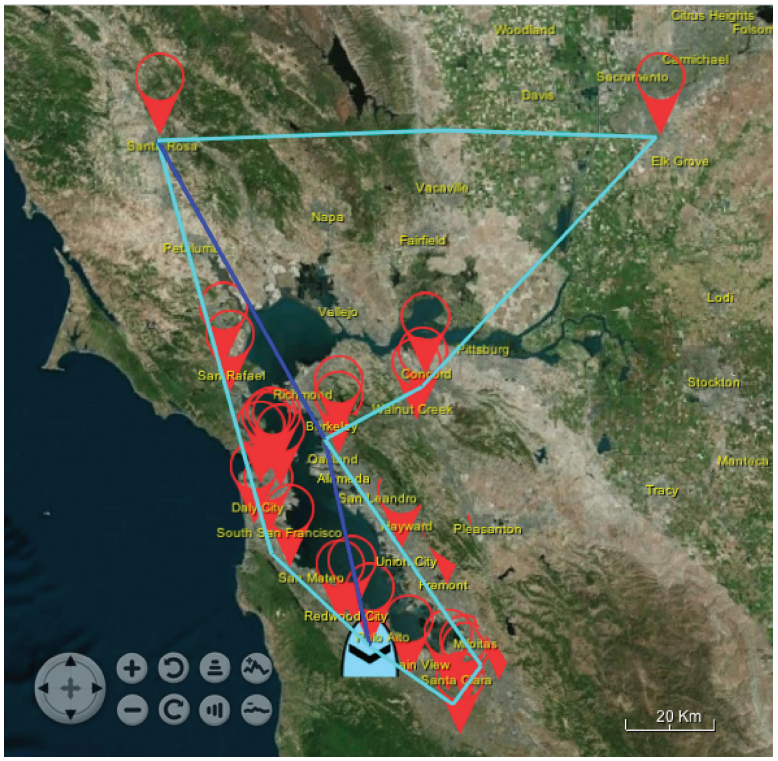


Fig. 17. An example of planned path vs. adaptation challenge path generated by random waypoint deletion: here, a planned UAV path (light blue) to survey hospital infrastructure (red) is subjected to 50% random waypoint deletion. In this case, the route loses 5 of its 8 non-terminal waypoints, causing both of its Eastward excursions to be replaced with the truncated path segments shown in dark blue and leaving 11 of 28 survey tasks without coverage.

d from 0.0 to 0.7 in steps of 0.1, comparing the full scenario of eight infrastructure classes under MTIP to the situation without MTIP, in which UAVs do not share their sensors but only image their own originally assigned tasks. The second experiment evaluates the degree to which each available sensor increases resilience by running MTIP with a fixed high rate of deletion $d = 0.5$ and selecting a random subset of n infrastructure classes (adding both UAVs and targets for each class), varying n incrementally from 1 to 8. We run 20 trials for each condition in each experiment.

As anticipated, the results of these experiments show that MTIP greatly increases the resilience of sensing missions. Figure 18 shows the effect of varying the fraction of waypoints deleted on the fraction of tasks that are able to be successfully surveyed. With MTIP's dynamic sensor sharing, the fraction of tasks surveyed degrades much more slowly than when UAVs do not share their sensors. Not until more than 30% of waypoints are deleted does the fraction surveyed show any significant decrease, and even at the extreme value of 70% waypoint deletion, the system is consistently able to survey around 90% of tasks. Performance is much less variable as well, another indicator of reduced fragility.

Analysis of individual tests shows that this increased resilience comes from two different sources: first, the fact that MTIP "backs up" the survey plan for each task with the spare capacity of other UAVs whenever possible; and second, the fact that the fast agent-based planning system

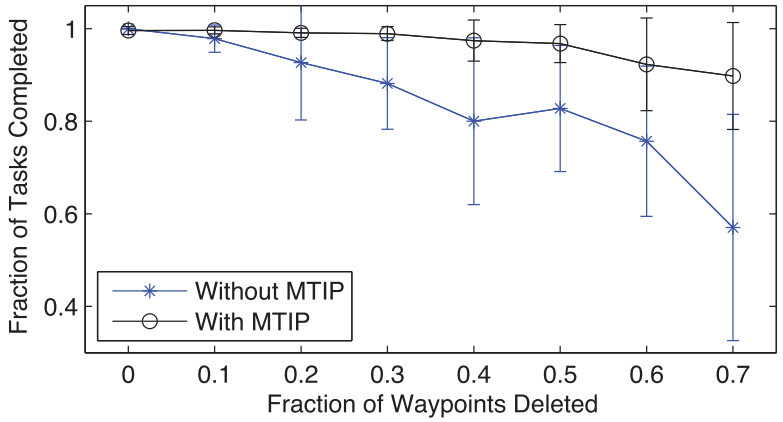


Fig. 18. MTIP dynamic sensor sharing allows coverage to be sustained well even when UAVs diverge greatly from their anticipated routes.

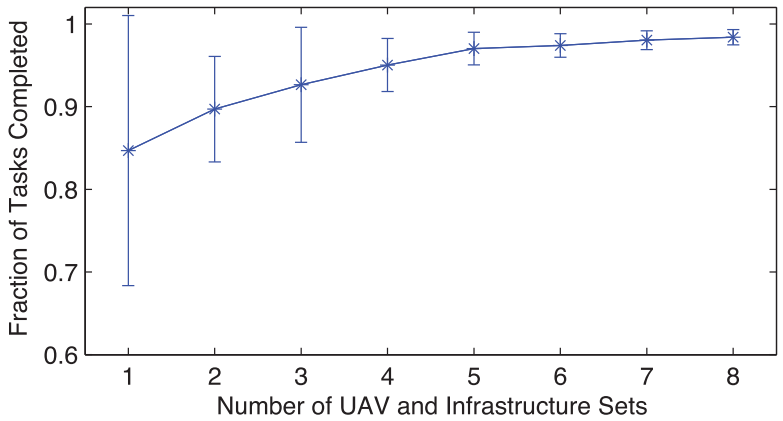
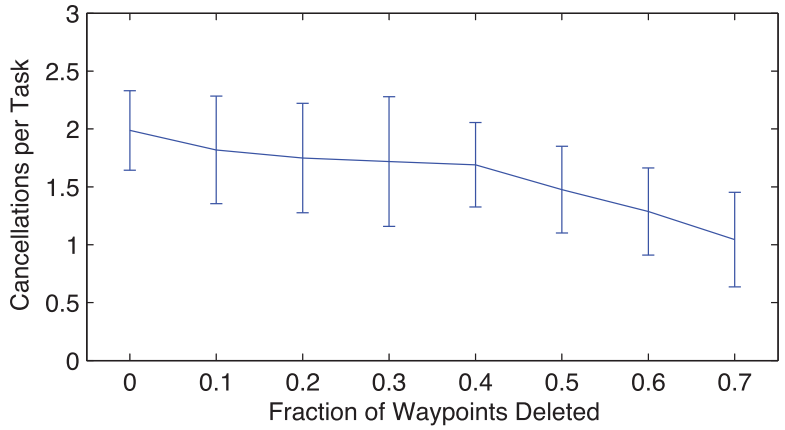


Fig. 19. Ability to adapt in coverage of sensor tasks is significantly improved by sensor sharing, even between a small number of UAVs. Results show adaptation to 50% deletion.

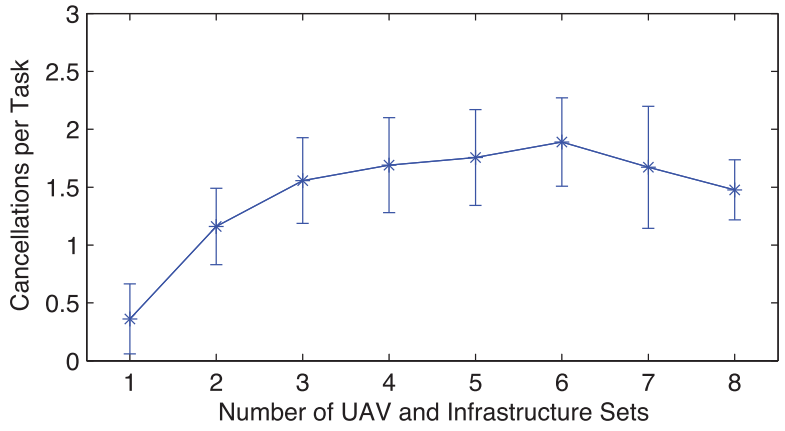
can rapidly reallocate when unexpected, UAV movements cause tasks to become unassigned or create new observation opportunities, almost as soon as those movements are observed.

The results of our second experiment show that even a small amount of sensor sharing can greatly improve resilience. Figure 19 shows that sharing between even just two sets of UAVs eliminates approximately half of the observed degradation in task completion, as well as greatly decreasing variability of performance. More sets of UAVs continue to incrementally improve the situation, up until five sets of UAVs, at which point nearly all survey targets are covered and additional sets of UAVs necessarily provide diminishing returns.

Rapid adaptation to changing circumstances, however, may come at a cost. Whenever a survey task is allocated to a UAV or cancelled on that UAV, it must be informed of the change, adding network traffic. Worse, if MTIP is in interactive mode, which queries platform operators to accept or reject each task, every new allocation produces an operator query, which can be a significant burden on the scarce resource of operator attention. Of particular concern is the possibility of adaptation resulting in a "thrashing" condition, a well-known problem of replanning (e.g., Nebel



(a) Variable deletion rate



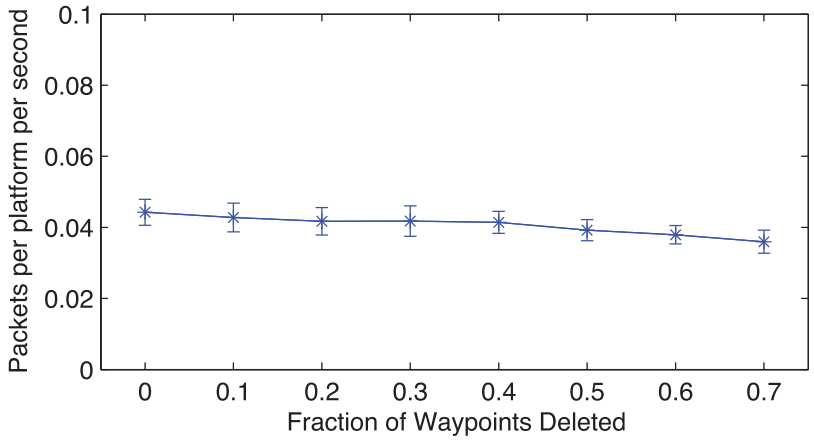
(b) Variable number of sharing participants

Fig. 20. The number of cancellations per task is fairly low, but highest with no waypoint deletion, indicating that some degree of thrashing in task allocation is still occurring.

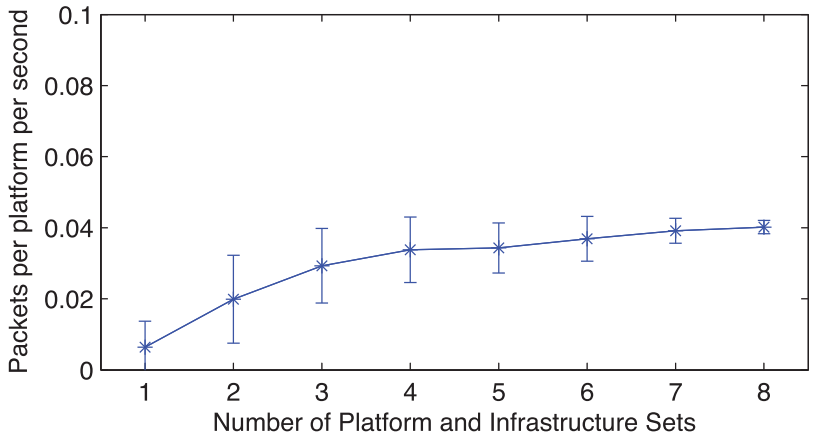
and Koehler (1995) and Fox et al. (2006)) in which a system shifts rapidly between plans with significantly different content but nearly equivalent utility (e.g., target coverage).

As noted in Section 4.2, however, the state projection included in the allocation algorithm is intended to prevent such thrashing. To validate its performance, we quantify the adaptation cost for MTIP by measuring two statistics: how frequently a survey task assignment is cancelled (which is also linked to the rate at which tasks are reassigned), and the number of communication packets per platform per simulated second (either assigning or canceling; both are typically less than 1KB). Cancellation rates for our two experiments are shown in Figure 20, while communication rates are shown in Figure 21. As can be seen, cancellations and communications are both fairly low, indicating that MTIP is unlikely to pose a significant burden on either network capacity or operator attention.

Note, however, that the number of cancellations per task is highest when there is no waypoint deletion: this indicates that there is some degree of thrashing still occurring in task allocation. Inspection of individual cases indicates that this is primarily due to the quantization of planning



(a) Variable deletion rate



(b) Variable number of sharing participants

Fig. 21. The rate of communication required per platform is fairly low, indicating that MTIP is unlikely to pose a significant burden on either network capacity or operator attention.

projections: as platforms move along their routes, line-of-sight projections are affected for survey targets that are either at the extremes of sensor range or in complex terrain such as mountain valleys, causing these targets to switch back and forth between allocatable and not allocatable for a particular platform. Thus, while plans are indeed stable for nearly all targets, a small population of targets are being repeatedly allocated and cancelled. This does not appear, however, to reflect any inherent problem with the current approach, only a shortcoming of the current implementation's handling of line-of-sight relations; preliminary tests indicate it can be mitigated by adding a short-term memory for line-of-sight relations to smooth out this side effect of quantization.

7 CONTRIBUTIONS

As we have demonstrated, there are significant opportunities to improve the overall efficacy of airborne sensor platforms by making their sensors available for opportunistic use by other information consumers. Our prototype MTIP system provides an implementation of such a framework,

in the context of a publish-subscribe IMS, and we have validated both this system and the overall sensor sharing concept using an emulated scenario of disaster response in the San Francisco area, demonstrating that we can produce effective plans for sensor sharing and quickly and effectively adapt those plans on the fly in response to changing circumstances. While the present work tests only this region, the region tested is highly heterogeneous, including the flat, rural lands of the Sacramento and San Joaquin Valleys, extremely high population density urban areas near San Francisco, and largely unpopulated regions of high topographic relief. Furthermore, spot inspection of some infrastructure classes in other areas indicates the observed patterns of overlap are likely to hold elsewhere as well.

In future work, we aim to improve the performance of the MTIP allocation system by adding short-term memory for line-of-sight calculations to mitigate for the quantization of planning, as well as by optimizations in the handling of GIS data and redundant agent executions to improve efficiency and scalability. Future work will also address generalizations of the core allocation system, such as support for multi-sensor platforms, and repeating tasks. The agent-based architecture can also be leveraged to decentralize MTIP, allowing nearby platforms to communicate and replan even when their communication with the central dispatcher is limited or unavailable. We also aim to transition MTIP toward deployment on various fielded airborne platforms (including validation against other types of dynamism, such as changing tasks and platform loss), where its ability to improve situational awareness through sensor-sharing can be put to real use and be validated in the field. Finally, the MTIP approach of lightweight agent-based planning may be applicable to other complex and dynamical environments as well, such as smart transportation systems or services for mass public events, and these results may form a foundation on which to investigate such further expansions. It may also be of interest to consider adaptation of the MTIP approach to other architectures, such as incentive-driven sensing or non-publish-subscribe IMS.

REFERENCES

Jude Allred, Ahmad Bilal Hasan, Saroch Panichsakul, William Pisano, Peter Gray, Jyh Huang, Richard Han, Dale Lawrence, and Kamran Mohseni. 2007. SensorFlock: An airborne wireless sensor network of micro-air vehicles. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys'07)*. ACM, New York, NY, 117–129.

Jacob Beal, Danilo Pianini, and Mirko Viroli. 2015. Aggregate programming for the internet of things. *IEEE Computer* 48, 9 (2015), 22–30.

Jacob Beal, Kyle Usbeck, Joseph Loyall, and James Metzler. 2016a. Opportunistic sharing of airborne sensors. In *Proceedings of the 12th International Conference on Distributed Computing in Sensor Systems*. 25–32.

Jacob Beal, Kyle Usbeck, Joseph Loyall, Mason Rowe, and James Metzler. 2016b. Adaptive task reallocation for airborne sensor sharing. In *Proceedings of the Workshop on Engineering Collective Adaptive Systems*. 168–173.

David G. Bel, Frank Kuehne, Chris Maxwel, Randy Kim, Kushyar Kasraie, Tom Gaskins, Patrick Hogan, and Joe Coughlan. 2007. NASA world wind: Opensource GIS for mission operations. In *Proceedings of the IEEE Aerospace Conference*. 1–9.

Richard A. Burne, Anna L. Buczak, Vikram R. Jamalabad, Ivan Kadar, and Eitan R. Eadan. 1999. Self-organizing cooperative sensor network for remote surveillance. In *Enabling Technologies for Law Enforcement and Security*. International Society for Optics and Photonics, 124–134.

Marco Carvalho, Adrian Granados, Kyle Usbeck, Joseph Loyall, Matthew Gillen, Asher Sinclair, and James Hanna. 2011. Integrated information and network management for end-to-end quality of service. In *Proceedings of MILCOM*. 1604–1609.

Chee-Yee Chong and Srikanta P. Kumar. 2003. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE* 91, 8 (2003), 1247–1256.

Kevin A. Delin. 2002. The sensor web: A macro-instrument for coordinated sensing. *Sensors* 2, 7 (2002), 270–285.

Shlomi Dolev. 2000. *Self-Stabilization*. MIT Press.

Shane B. Eisenman, Nicholas D. Lane, and Andrew T. Campbell. 2008. Techniques for improving opportunistic sensor networking performance. In *Distributed Computing in Sensor Systems*. Springer, 157–175.

Barbara Essendorfer and Wilmuth Mueller. 2009. Interoperable sharing of data with the coalition shared data (CSD) server. In *North Atlantic Treaty Organization (NATO)/Research and Technology Organization (RTO): C3I in Crisis, Emergency and Consequence Management*. 7–1–7–12.

Lukas Esterle, Peter R. Lewis, Xin Yao, and Bernhard Rinner. 2014. Socio-economic vision graph generation and handover in distributed smart camera networks. *ACM Transactions on Sensor Networks (TOSN)* 10, 2 (2014), 20.

Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina. 2006. Plan stability: Replanning versus plan repair. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*. 212–221.

Matthew Gillen, Joseph Loyall, Kyle Usbeck, Kelly Hanlon, Andrew Scally, Joshua Sterling, Richard Newkirk, and Ralph Kohler. 2012. Beyond line-of-sight information dissemination for force protection. In *Proceedings of the Military Communications Conference (MILCOM)*.

Matthew Gillen, Joseph P. Loyall, and Joshua Sterling. 2011. Dynamic quality of service management for multicast tactical communications. In *Proceedings of the 14th IEEE Computer Society Symposium on Object/Component/Service-oriented Real-time Distributed Computing (ISORC)*. 11–18.

Prem Prakash Jayaraman, Charith Perera, Dimitrios Georgakopoulos, and Arkady Zaslavsky. 2013. Efficient opportunistic sensing using mobile collaborative platform mosden. In *Proceedings of the Inernational Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 77–86.

Andreas Krause, Eric Horvitz, Aman Kansal, and Feng Zhao. 2008. Toward community sensing. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*. IEEE Computer Society, 481–492.

Michael J. Kristan, Jeffrey T. Hamalainen, Douglas P. Robbins, and Patrick Newell. 2009. *Cursor-on-Target Message Router User's Guide*. Technical Report MITRE Product–MP090284. MITRE.

Nicholas D. Lane, Shane B. Eisenman, Mirco Musolesi, Emiliano Miluzzo, and Andrew T. Campbell. 2008. Urban sensing systems: Opportunistic or participatory? In *9th Workshop on Mobile Computing Sys. and App.* 11–16.

Qilian Liang, Xiuzhen Cheng, S. C. Huang, and Dechang Chen. 2014. Opportunistic sensing in wireless sensor networks: Theory and application. *IEEE Transactions on Computers* 63, 8 (2014), 2002–2010.

Joseph Loyall, Matthew Gillen, Jeffrey Cleveland, Kyle Usbeck, Joshua Sterling, Richard Newkirk, and Ralph Kohler. 2012. Information ubiquity in austere locations. *Procedia Computer Science* 10 (2012), 170–178. DOI: http://dx.doi.org/10.1016/j.procs.2012.06.025 ANT 2012.

Chaoying Ma and Jean Bacon. 1998. COBEA: A CORBA-based event architecture. In *Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems*. Volume 4. 9–9.

Huadong Ma, Dong Zhao, and Peiyan Yuan. 2014. Opportunities in mobile crowd sensing. *IEEE Communications Magazine* 52, 8 (2014), 29–35.

William Miller. 2004. Cursor-on-target. *Military Information Technology Online* 8, 7 (2004).

B. Nebel and J. Koehler. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence* 76, 1–2 (1995), 427–454.

OMG Data Distribution Service Portal. Retrieved March 6, 2012 from http://portals.omg.org/dds/.

H. Van Dyke Parunak and Sven Brueckner. 2007. Concurrent modeling of alternative worlds with polyagents. In *Multi-Agent-Based Simulation VII*. Springer, 128–141.

Danilo Pianini, Mirko Viroli, and Jacob Beal. 2015. Protelis: Practical aggregate programming. In *Proceedings of the 2015 ACM Symposium on Applied Computing*. 1846–1853.

Doug Robbins. 2007. Unmanned aircraft operational integration using MITRE's cursor on target. *The Edge* 10, 2 (2007).

David S. Rosenblum and Alexander L. Wolf. 1997. A design framework for internet-scale event observation and notification. *SIGSOFT Software Engineering Notes* 22, 6 (Nov. 1997), 344–360. DOI: http://dx.doi.org/10.1145/267896.267920

M. Schneider. 1993. Self-stabilization. *Computing Surveys* 25 (1993), 45–67.

Xiang Sheng, Xuejie Xiao, Jian Tang, and Guoliang Xue. 2012. Sensing as a service: A cloud computing system for mobile phone sensing. In *Sensors, 2012 IEEE*. IEEE, 1–4.

Minho Shin, Cory Cornelius, Dan Peebles, Apu Kapadia, David Kotz, and Nikos Triandopoulos. 2011. AnonySense: A system for anonymous opportunistic sensing. *Pervasive and Mobile Computing* 7, 1 (2011), 16–30.

Kyle Usbeck, Matthew Gillen, Joseph Loyall, Andrew Gronosky, Joshua Sterling, Ralph Kohler, Kelly Hanlon, Andrew Scally, Richard Newkirk, and David Canestrare. 2015. Improving situation awareness with the android team awareness kit (ATAK). In *SPIE Defense+ Security*. International Society for Optics and Photonics, 1–22.

Ming Xiong, Jeff Parsons, James Edmondson, Hieu Nguyen, and Douglas Schmidt. 2007. Evaluating technologies for tactical information management in net-centric systems. In *Proceedings of the Defense and Security Symposium*. International Society for Optics and Photonics, 1–11.

Received March 2017; revised August 2017; accepted January 2018