# System Architectures for Multi-Sensor Task Allocation

Diego Pizzocaro*,
Alun Preece,
School of Computer Science & Informatics,
Cardiff University, UK
(*D.Pizzocaro@cs.cf.ac.uk)

Fangfei Chen,
Thomas La Porta,
Department of Computer Science and Engineering,
Pennsylvania State University, USA

Matthew P. Johnson,
Amotz Bar-Noy
Department of Computer Science,
Graduate Center,
City University of New York, USA

***Abstract*—Heterogeneous sensor networks are increasingly being deployed to support users on the field who require many different kinds of sensing tasks, such as detecting people who may need help or monitoring collapsing buildings. Sensors may be scarce and in high demand. Sensing tasks might compete for the exclusive usage of available sensors. Such an environment is highly dynamic with, for example, sensor failures and changing sets of tasks. First responders usually lack the time and expertise to manually decide which are the best sensors for each task. We need therefore to automate the allocation of sensors to tasks. We formalize this problem, which we call Multi-Sensor Task Allocation (MSTA). We propose different system architectures to solve MSTA and we discuss the tradeoffs that each architectural choice implies. We show a proof-of-concept implementation of the system interface deployable on mobile devices in the field.**

## I. Introduction

Heterogeneous sensor networks are increasingly being deployed to support users on the field who require many different kinds of sensing tasks. For example, in an emergency response scenario, tasks will include detecting people who may need help and monitoring collapsing buildings. Consider the recent case of Haiti earthquake or also the Hurricane Katrina disaster in New Orleans. In both of these cases the military used sensors mounted on Unmanned Aerial Vehicles (UAV) to scan the disaster areas and organize humanitarian relief operations.[1] In the near future such operations will be supported by a *sensor network* consisting of a large number of heterogeneous sensing devices of many different types. These could vary from very simple sensors with limited capabilities such as motes, to very complex unmanned platforms such as UAVs. This direction is seen as the way forward in research to provide sensor network support to future multinational coalition operations[2].

Upon deployment on the field of an emergency response operation, heterogeneous sensing devices will form an ad hoc network using wireless links or cables to communicate with each other. Given that sensing devices can be static or mobile, and that sensors may fail or be damaged, it is clear that the network configuration is highly dynamic. This heterogenous sensor network is required to support multiple sensing tasks of different types to be accomplished simultaneously in order to give support to many different emergency response operations at the same time. Sensing tasks might share the usage of a sensing resource, but more often they compete to exclusively control it. Consider for example a search-and-rescue scenario as the one in Figure 1, where the sensor network needs to support the task of detecting people who may need help, and at the same time to monitor a collapsing building which might represent threats to the life of other people. Assuming that all the other sensors in the network are already tasked, if only one UAV is available and can serve exclusively one of these two tasks (e.g. because the two tasks are far apart) then the question becomes, where is it better to send that UAV?

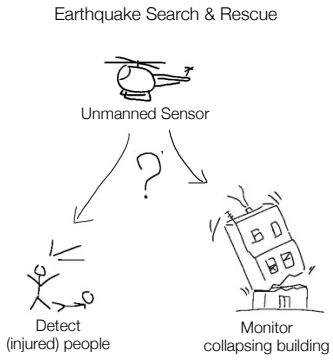


Fig. 1. Multi-Sensor Task Allocation problem.

A mobile user on the field creating many different sensing tasks during an emergency response operation would not have the time to manually decide what is the best set of sensors to use for each of his own tasks. In addition, he might not have the expertise to decide what type of sensors could best match the sensing requirements of each task. Even assuming that the user was an expert and had enough time to manually decide which sensors to allocate to his tasks, they would probably not consider all the pertinent features of the problem. We need therefore to automatically allocate individual sensors to the task they best serve, considering all the relevant parameters (e.g. sensor distance, sensor lifetime, task priority, task duration, etc). In general we can have tasks assigned to multiple sensors and sensors assigned to multiple tasks, but the fundamental question remains, which sensor should be allocated to which task? This summarizes the

[1] "Global Hawk collects reconnaissance data during Haiti relief efforts", http://www.af.mil/news/story.asp?id=123185754; "Small, Unmanned Aircraft Search for Survivors in Katrina Wreckage", http://www.nsf.gov/news/news_summ.jsp?cntn_id=104453

[2] International Technology Alliance in Network & Information Sciences (ITA), http://www.usukita.org

problem that we call Multi-Sensor Task Allocation (MSTA). The problem is exacerbated in coalition contexts — for example, a humanitarian relief operation requiring a multi-agency international response — where users will have little idea of what sensor assets are available across the coalition, or what their ownership and access rights might be.

In this paper we present and analyze a variety of different system architectures which could solve different MSTA problem instances inspired by generic emergency response scenarios in a coalition context. The novelty of our architectures consists mainly in a modular approach which solves the MSTA problem step by step, by integrating a knowledge base module with an allocation mechanism. Together, these two components offer flexibility in the automatic choice of sensors while also hiding the complexity of this choice from the user.

The remainder of the paper is organized as follows. In Section II we describe and formalize the MSTA problem in the context of background work on sensor allocation systems. In Section III we provide an overview of our conceptual system architecture, including the knowledge base module. Then in Section IV we describe and analyze the alternative architectural choices available to support sensor allocation on the field. Section V concludes the paper with a discussion and outline of future work.

## II. Multi-Sensor Task Allocation Problems

There are many possible instances of the MSTA problem, depending on a number of key parameters. For this reason, we propose a taxonomy of MSTA problems through which we can classify problem instances, and thus identify more easily the underlying optimization problem. The MSTA taxonomy which we propose consists of the following four main axes, introduced in [12]:

- **Single-task sensors (ST)** vs **multi-task sensors (MT)**: in ST each sensor is able to execute at most one task at a time; in MT a sensor can execute multiple simultaneous tasks. This is equivalent to what researchers have previously called exclusive sensors [13] vs shared sensors [10].
- **Single-sensor tasks (SS)** vs **multi-sensor tasks (MS)**: SS means that each task needs exactly one sensor to be accomplished; MS means that some tasks could require multiple sensors (also referred to as the coalition formation problem [17]).
- **Instantaneous assignment (IA)** vs. **time-extended assignment (TA)**: IA means that the information concerning the sensors, tasks and the environment does not allow for planning of future allocation; instead in TA sensors can be allocated considering the future. For example, in [6] this dimension corresponds to lifetime-unaware approach vs lifetime-aware (where it is available a model of how task are expected to arrive over time).
- **Homogeneous sensor network (HO)** vs **heterogeneous sensor network (HE)**: HO means that the network is composed only by sensors with the same features; HE represents the case in which the sensor network consists of sensing devices of different types each with different sensing capabilities, lifetimes and other unique characteristics.

To categorize MSTA problem instances we identify the corresponding quadruple of two-letter abbreviations which parameterize our problem scenario. Given that military and emergency response operations generally use heterogeneous sensing devices in order to coordinate missions, we focus on heterogeneous sensor networks (HE); also given that the dynamic environment does not provide enough information to plan for future allocation we aim at Instantaneous Allocation (IA). In the most general case each task usually requires a group of sensors therefore we consider just Multi-Sensor tasks (MS) which require a more complex way of evaluating the joint utilities of sensor bundles, often involving non-additive joint utility functions (i.e. functions in which the utility coming from each sensor does not sum-up linearly). For these reasons, we believe that two specific instances of MSTA are actually most relevant in our scenario: *ST-MS-IA-HE* when sensors can serve exclusively one task at a time and *MT-MS-IA-HE* when sensors can be shared among tasks. Below we include a brief analysis of the optimization problems underlying the two MSTA instances on which we focus, summarizing and porting to our scenario some of the work done in [3]. Note that here we consider tasks as independent (not linked).

### A. ST-MS-IA-HE: Single-task sensors

*ST-MS-IA-HE* considers Single-Task sensors supporting Multi-Sensor tasks. This is a very common scenario because often sensors are scarce and in high demand and most importantly sensors cannot always be shared; here therefore we focus on a restricted type of sensors such as directional sensors. This problem is referred to as *coalition formation* in the multi-agent community and it has been formally studied in [15] and [17]. Following [3], ST-MS-IA-HE can be modeled as a Set Partitioning Problem (SPP) [1] in which is given a set of sensors $S$ and a family $F$ of acceptable subsets of $S$ and a utility $u : F \rightarrow \mathbb{R}$. The objective is to find a maximum-utility family $X$ of elements in $F$ such that X is a partition of $S$. Note that $F$ represents the set of all feasible sensor bundle-task pairs and the utility $u$ represents the utility estimate for each pair. It is clear that for each different type of sensing task we will have a different utility function, for example for a localization task we will have a utility dependent on the angle between sensors and instead for a video reconnaissance task we will have a utility based on the distance and resolution of the video sensors. Finally note that casting the *ST-MS-IA-HE* problem as an instance of SPP does not imply that all sensors should be allocated to at least one task or that all the tasks should be assigned at least a bundle of sensors, in fact the family $F$ includes all the subsets of $S$ where for some the utility might be zero.

### B. MT-MS-IA-HE: Multi-task sensors

We can easily imagine cases in which a sensor can be shared among different tasks, e.g. a UAV surveilling two areas that

are close enough. This can happen much more frequently in an heterogeneous sensor network and therefore it is important to consider also the *MT-MS-IA-HE* problem in which we allow to assign a sensor to multiple tasks. Of course each sensor will have a limited maximum number of tasks able to serve, due to sensory limitations: e.g. a camera will be able to detect up to a certain number of suspicious objects in a certain area, given for example that the camera might miss out-of-focus objects. *MT-MS-IA-HE* is also well studied by the multi-agent community and it is referred to as *overlapping coalition formation* problem [17]. Such a problem can be cast as a variant of the Set Covering Problem (SCP) [18] in which is given a set of sensors $S$ forming the network, a family $F$ of acceptable subsets of $S$ representing possible overlapping coalitions, and a utility function $u : F \rightarrow \mathbb{R}$ as an estimate of the utility of assigning a subset of sensors to a task. The objective is to find the maximum-utility family $X \in F$ such that $X$ is a cover of $S$. Note that also in this case casting the *MT-MS-IA-HE* problem as an instance of SCP does not enforce that all sensors should be allocated to at least one task and viceversa; in fact by definition the family $F$ includes also subsets of $S$ for which the utility is zero (representing unassigned sensors or unsupported tasks). If full coverage of the tasks is required there could be several ways to include it in the problem formulation, but in our case we allow for tasks being dropped due to the features of our environment in which sensing resources are scarce and in high demand.

MSTA is closely related to the Multi-Robot Task Allocation (MRTA) problem which consists of answering the question: which robot should execute which task in a generic Multi-Robot System (MRS). Gerkey et al [3] offered a framework to categorize different instances of the MRTA problem, proposing a *domain-independent taxonomy* of problems. The MSTA taxonomy is designed as an extension of the MRTA taxonomy, to include domain specific features of the sensor network environment. Most importantly in MRTA the distinction between a MRS composed by homogeneous devices vs one composed by heterogeneous devices does not represent one of the main axes. This distinction is instead extremely important when developing solutions to MSTA problems [19], [9]: as a matter of fact in order to correctly allocate sensor assets to tasks in a heterogenous environment, it is necessary to have more complex knowledge about the sensing capabilities provided by each sensor and the sensing requirements requested by each task.

## III. Conceptual Architecture

In both the MSTA problem instances we have to deal with coalitions of sensors which we call *sensor bundles*. Given this common feature we propose a conceptual architecture valid for both problem instances which highlights the steps necessary in order to find a solution. The architecture is comprised of four main components as shown in Figure 2: a mobile user on the field, a knowledge-based bundle generator (KB bundle generator), an allocation mechanism, and the sensor network

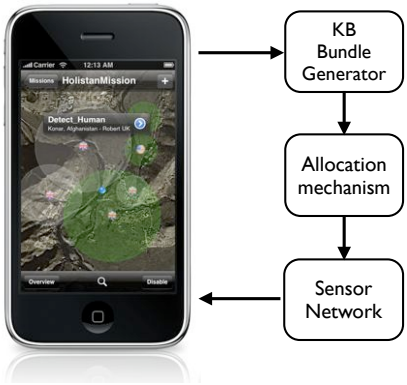


Fig. 2. Conceptual system architecture.

itself. The mobile device represents the point of entry of tasks into the system: basically the mobile user can express their sensing requirements, including a local area-of-interest, represented in Figure 2 as a coloured circular area on the smartphone GUI. Note that we assume that users on the field are provided with mobile devices (such as smartphones, PDAs or tablets) to access the system.[3]

The *KB bundle generator* recommends bundles of sensors that are known to be "fit-for-purpose" for that particular type of task specified by the user. Then, the *allocation mechanism* finds a solution either to the *ST-MS-IA-HE* or to the *MT-MS-IA-HE* problem depending on the scenario. The allocation mechanism considers the output of the KB bundle generator for each task generated; this consists of a set of coalitions of sensors that, if allocated to the task, would satisfy the sensing requirements demanded by the user. Note that a single sensor bundle recommended by the KB bundle generator is enough to satisfy the sensing requirements of the task; therefore we may have a one-to-one (in *ST-MS-IA-HE*) or a one-to-many (in *ST-MS-IA-HE*) relationship between sensor bundles and tasks. Finally, the sensor network is configured according to the output of the allocation mechanism, and begins serving the tasks by delivering sensor data to the user.

The functionality of the knowledge-based bundle generator is independent of the architectural choices we will present in Section IV so we discuss it in further detail here, including implementation options.

### A. Knowledge-Based Bundle Generator

Task allocation in heterogeneous sensor networks requires knowledge of which sensor types are applicable to which kinds of task. For example, certain kinds of object localisation tasks can be achieved through acoustic sensing (provided that there are sufficient sensors to triangulate the signals and determine object location) or visual sensing (provided that the image is of sufficient quality to determine the presence of an object). We separate two issues: whether a type (or combination of types)

[3]As an illustration-of-concept, we prototyped the interface to the system on the Apple iPhone (http://www.apple.com/iphone/) motivated mainly by its popularity and the quality of the development tools provided.

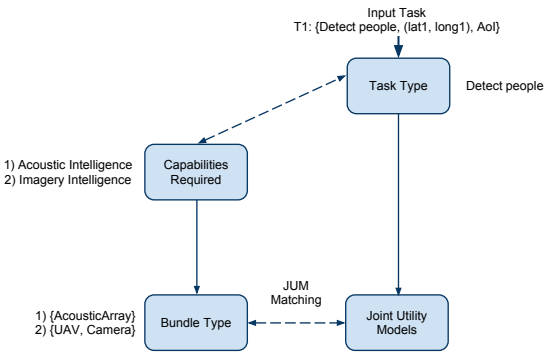


Fig. 3. Reasoning procedure.

of sensor *can* potentially satisfy a task, and *how well* might particular sensor instances perform on a given task. Addressing these issues requires knowledge from the literature, and we encode this knowledge in a knowledge base (KB). The KB stores, for each kind of task, each type (or combination of types) of sensor that can theoretically achieve that task, and a *joint utility function* that allows us to compute the utility of particular sensor instances for that task. For example, based on [14] we define a *2D-localization* function that returns a utility based on the locations of two sensors of the same type (in order to triangulate signals in two dimensions) where the theoretical utility is derived from the spread of the sensors. This function can be used with, for example, acoustic or seismic sensors. We refer to the types of applicable sensors as *bundle types* (because they determine the types of sensors that form our allocated bundles). These types may be defined using a *sensor ontology*, such as the one described in [4].

Figure 3 shows the reasoning process enabled by the KB in more detail. For each newly created task $T_j$ at time step $t$ the KB recommends a Joint Utility Model composed by a Bundle Type and a Joint Utility Function. This identification of Bundle Types was originally implemented as described in [4], by dynamically matching the sensing capabilities provided by each single sensor and the capabilities required by each task type. Sensor and task features were defined using ontologies expressed with the Web Ontology Language (OWL); the matching process was implemented using the Pellet open source OWL reasoner[4]. The output of this step is a set of Bundle Types (BT), where each of the entry is composed by a set of sensor types which altogether would satisfy the information requirements of the sensing task. The BTs recommended at this stage are just sets of sensor types, and they do not contain any kind of information regarding the number of instances to choose for each sensor type in the BT.

This is of crucial importance for the generation of bundles of sensors to be assigned to a task in order to satisfy it and it is solved through the matching with a Joint Utility Model (JUM). Such a model consists of a suggested maximum, minimum or exact number of sensor instances for each of the sensor types forming a BT. Moreover it includes a Joint Utility Function (JUF) which is used to compute the estimated value ($e_{kj}$) for a group of sensor instances implementing the recommended BT. The KB indicates which JUMs are appropriate for each task type. Each JUF is only compatible with certain sensors, so the final step in the reasoning procedure is to match applicable JUMs with BTs.

To illustrate the above, consider the example mentioned in the previous section, in which an emergency responder creates a task on the field in the coordinates $(lat_1, long_1)$ and the type of the task is *Detection of Injured People*. The reasoner suggests to use either $BT_1 = \langle\{AcousticArray\}$ or $BT_2 = \langle\{UAV, Camera\}\rangle$, as result of the first reasoning stage. Later on it associates one or more JUMs to each recommended BT by searching the knowledge base. In this case, we could assume that in the knowledge base there are just two utility models $JUM_1$ and $JUM_2$, like the ones proposed in [14], compatible with these two BTs. Therefore we obtain the following output from the reasoner $(BT_1, JUM_1)$, $(BT_1, JUM_2)$, $(BT_2, JUM_1)$, $(BT_2, JUM_2)$. This allows us to be very flexible in terms of allocation as it is clear that the same task can be satisfied using different Bundle Types, and therefore increasing the chances of satisfying that task.

### B. Lightweight KB Bundle Generator

The original implementation of the reasoning process is computationally expensive [4] due to the exponential-time complexity of the classification algorithm used by the Pellet reasoner. However, because the task types and sensor types are relatively stable (it is rare for new kinds of sensor or task to become available) it is feasible to pre-compute the results of the reasoner and store these in a look up table; such an implementation is more suitable for deployment on a mobile device, to avoid wasting battery life on expensive computation (allowing us to save the device battery for more important things such as the delivery of the information from the sensor network). Note that all the reasoning operations using a lookup table will be $O(1)$. The only assumption that this approach makes is that the device will have a sufficiently large storage capacity, which is reasonable for modern smartphones. The structure of the look up table is represented in Figure 4. Note that each row of the table is a tuple composed by a task type (represented as an ID), a Bundle Type and a Joint Utility Function.

Our initial experiments show that memory requirements and lookup performance on the mobile device are acceptable. A realistic knowledge base (including around 4000 different task types with 5 different BT+JUM on average per task type, i.e. around 20,000 table entries) occupies 12 megabytes of the flash memory of an iPhone. Its size grows linearly with the number of entries in the table. The time required to retrieve an entry from this lookup table on the iPhone is around 20 milliseconds, which increases logarithmically with the number of entries.

[4]http://pellet.owldl.com/

| Task Type | Recommendation (BT+JUFs) |
|---|---|
| 1 | ($BT_1$ + $JUF_1$) |
| 1 | ($BT_2$ + $JUF_1$) |
| 1 | ($BT_2$ + $JUF_2$) |
| 2 | ($BT_3$ + $JUF_1$) |
| 2 | ($BT_2$ + $JUF_1$) |
| 2 | ($BT_2$ + $JUF_2$) |
| ... | ... |
| N | |

Fig. 4. Knowledge Base: lookup table on user device.

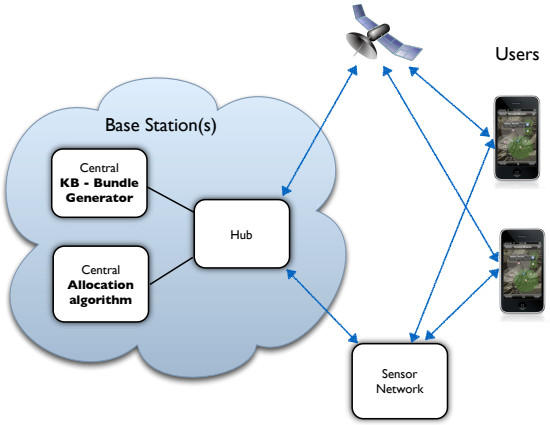


Fig. 5. Centralized system architecture.

## IV. Allocation Mechanisms

Unlike the KB bundle generator, which is independent from architectural choices, the allocation mechanism varies greatly depending on whether we use a centralized or a distributed architecture. In this section we describe three architectures: one fully centralized, one fully distributed, and another partially distributed (or hybrid). Note that all of them can support both *ST-MS-IA-HE* (i.e. exclusive sensor assignment to tasks) and *MT-MS-IA-HE* (i.e. sensors sharable between tasks).

### A. Centralized architecture

Currently many mobile applications are cloud-based, which means there is a centralized cloud server able to cope with all the mobile requests coming from different user's devices. In many emergency response and military operations, we might assume the presence of one or more base stations collecting information about the environment, thereby having a global view of the ongoing tasks. The mobile devices used on the field are usually provided with cellular/GPRS/3G connection and therefore could use these technologies to remotely communicate with a base station. As a consequence we might be able to implement both the KB bundle generator and the allocation mechanism on a central server and use mobile devices only to create new tasks and push them to the central server, which will then run periodically a centralized algorithm to find a feasible allocation, and also to receive data from the sensors.

Describing in more detail this architecture, shown in Figure 5: during a certain timestep, we have a set of mobile users requesting different sensing tasks in the field. These newly created tasks are then posted to a central server which passes each of the task to the KB bundle generator. Using different joint utility models for each task, this component computes for each task a set of feasible sensor bundles, each of which satisfies the task's sensing requirements. Then a centralized allocation algorithm is run to find a solution. Both the *ST-MS-IA-HE* and the *MT-MS-IA-HE* are NP-hard, therefore considering the size of our scenario which usually is in the order of hundreds of sensors and dozens of tasks the problem instance is in general too large to find an optimal solution, as stated in [7] and [3]. In Section II we analyzed how the *ST-MS-IA-HE* problem (where sensors can be part of exclusively one bundle) can be cast as a Set Partitioning Problem (SPP), so we could use centralized SPP approximation algorithms like the one in [5]. To apply this algorithm we have to be able to enumerate a set of feasible bundle-task combinations, which in our system is the output of the KB bundle generator. Another reasonable way of modeling this problem is as a *combinatorial auction* in which the bidders are the users who bid for bundles of sensors, basically combinations of items for which we can express just a joint utility value in the auction. In practice, each user generates a set of tasks, and then places a bid for the highest utility sensor bundles which could potentially satisfy the information requirements of the newly created tasks. Note that the value of the bid can easily coincide with the joint utility value computed by the KB bundle generator. There are many approximation and anytime algorithms developed to solve a combinatorial auction, among these the most relevant are CABOB [16] and CASS [2].

Considering the *MT-MS-IA-HE* problem (i.e. when tasks can share sensors), we can apply Set Covering Problem (SCP) approximation algorithms such as the one proposed in [8] since, as we explained in Section II, we can cast this problem as an instance of SCP. As highlighted in [3], many approximation algorithms have been proposed, most requiring a set of feasible bundles of sensors for each task as in the previous problem. More importantly, the general trend is that they perform poorly when the number of allowed sensor bundle for each task is large, that is when it is almost equal to the number of all possible subsets of sensors ($2^{|S|}$). This means that these algorithm actually behave better in a heterogeneous environment where the natural diversity of sensors limits the number of possible bundles for each task.

The key benefit of a centralized architecture for sensor-task allocation is that usually it is able to provide a better overall allocation than a distributed allocation, by virtue of having a global rather than local view of the situation [7]. A secondary benefit of this architecture is that there is no "heavy" computation happening on the mobile devices nor on the sensor nodes — these need only to communicate to the base station their tasks and current status (location, battery life, whether they are serving a task or served by a sensor bundle, etc). This improves the lifetime of the sensor network and of the user devices.

Implementing a centralized architecture relies on (1) the Hub having an up-to-date picture of the status of the sensor network and tasks on the field and (2) users having con-

nectivity with the Hub. We would argue that (1) needs to happen anyway in order to monitor what is happening on the field for mission command and control purposes. (2) is particularly problematic if a certain area becomes "hot", e.g. there is an explosion and/or a building collapses, and suddenly many mobile users (military or emergency responders) may occupy the same area, leading to an overload of the mobile telecommunication network. We need to ensure that our system supports sensing tasks requested by mobile users also when there is no connectivity with the base station. Note that messages posted on/sent from the server can travel both via the "conventional" telecommunication network or via the sensor network. However, while we might be able to reroute the traffic on the sensor network if the telecommunication network crashes due to network overload, this would reduce the lifetime and capacity of the sensor network.

### B. Distributed architecture

In this section we describe a fully distributed allocation system, shown in Figure 6, which is able to work autonomously without the presence of a base station, thus addressing the main problem with the centralized approach. This architecture moves the KB bundle generator and the allocation mechanism onto the users' mobile devices and the sensors composing the network. This is based on the idea that nowadays the computational power and storage on a mobile device, such as a smartphone, is comparable to that of an average personal computer, albeit with more limited running time due to the limited battery life. Therefore, in each of the mobile devices we include a light version of the KB bundle generator as described in Section III-B.

The users' devices communicate directly (e.g. using WiFi or Bluetooth) the tasks created to the sensors composing the network; then the sensors are able to autonomously negotiate through an allocation protocol which is the best task to serve as part of a bundle. Well known efficient allocation protocols for *ST-MS-IA-HE* and *MT-MS-IA-HE* have been proposed in [17] for the *coalition formation* and *overlapping coalition formation* problems. Note that these protocols were designed for generic multi-agent systems, so we need to adapt them in order to include the mobile device as a "task entry point" and as the only entity able to generate a list of sensor bundle candidates for each task. Note also that since these are dynamic protocols they are much more flexible and able to deal with the dynamic nature of our environment. For example, there is no need to synchronize events into timesteps, unlike in the centralized approach where we need to run the algorithm on all tasks created during a certain timestep and, while doing so, to queue any other newly-created tasks for the next timestep.

We give a sketch of how we may adapt the protocol for the *disjoint coalition formation problem* [17] to our scenario. The adapted protocol runs on the two main entities composing our distributed system: sensors and user devices. When the user creates a task (e.g. using the iPhone interface illustrated in Figure 2) the user device computes *feasible sensor bundles* and their *joint utilities* using the KB Bundle Generator, we call these pairs *bids*. These are then sent to the sensors which choose greedily the task to serve based on the average utility per sensor until there are no more bids. The modification of the protocol has also to take into account multi-hop communication channels between nodes and unreliable message delivery (due to package loss, congestion, delays, etc).

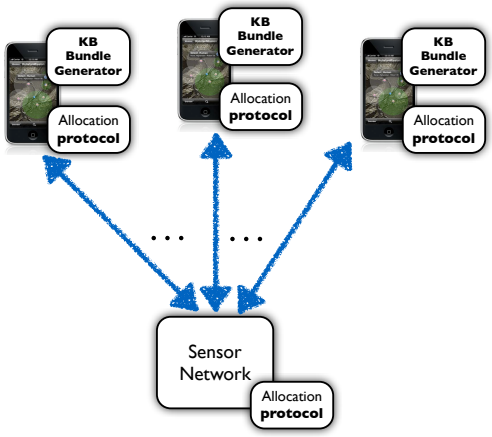


Fig. 6. Fully distributed system architecture.

In more detail, the protocol is as follows:

- At time $t$ the users create tasks on their devices which then generate bids of the type $bid_\alpha = \langle U_\alpha : (T_j, B_k, v_{\alpha,t})\rangle$, where $U_\alpha$ is a user, $T_j$ is a task generated by $U_\alpha$, and $B_k$ is the bundle of sensors with highest utility value $v_{\alpha,t}$ generated as an output of the *KB Bundle Generator*.
- The user device sends to all the sensors included in each of the bundles $B_k$ the bids $bid_\alpha$.
- Each sensor node keeps a list of bids in which it is involved sorted by decreasing average utility value, and chooses the one to which it can contribute the most.
- The sensor then sends an ACCEPT message to the sensors that are present in this bid.
- The sensor clears this bid (i.e. it officially commit to that task) only when it receives an ACCEPT message from all the sensors in this bid. This ensures that a bid is cleared if and only if all the sensors in the bid agree to clear it.
- When a sensor node clears a bid, it sends a CLEARED to all its neighbors — the set of sensors with which it shares some bids and the users who were in competition for it — telling them that it has cleared and all bids including the sensor should be dropped from consideration.
- The sensors that receive a CLEARED message from another sensor, delete the bids in which the sender sensor is included. The sensors stop execution when they clear a bid or their list of bids is empty.
- The users that receive a CLEARED message from a sensor assigned to another user's task, delete the sensor from the set of neighbor sensors and recompute a new bid with the remaining sensors. The users stop execution when they obtain an ACCEPT message or when a convergence

timeout is expired since the beginning of the negotiation.

It is clear from the protocol description that also in this case the KB bundle generator is crucial in the enumeration of the feasible sensor bundles and in the evaluation of their utility. By including this component on the user's mobile device it is also possible to easily extend the system by simply extending the KB with new types of tasks/sensors and new joint utility functions. This is very different from previous approaches in sensor networks where if a new task/sensor type had to be supported a new protocol has to be uploaded to the sensors in the network, which is usually referred to as wireless reprogramming of sensors as in [11]. In this case we would just need to update the KB of the device which can be easily done offline when recharging it, as opposed to the expensive operation of uploading a new protocol to the sensors.

The procedure for each sensor implementing the protocol has a computational complexity of order $O(n^k \cdot |T|)$ to arrive at a decision during each timestep (as proved in [17]), where $n$ is the number of sensors in the network, $k$ is the maximum size allowed for a sensor bundle and $|T|$ is the number of current tasks. Based on this result and on [7], we hypothesize that the traffic generated by this architecture should be less then the traffic generated by the centralized architecture described in the previous section; part of our future work is to confirm this by running experiments and analyzing the network traffic generated by the two architectures. As noted above, we would expect the centralized architecture to perform better than this distributed architecture with respect to quality of the allocation in terms of total welfare (calculated as the sum of the utilities of all the allocated bundle-task pairs at each timestep). Nonetheless, we anticipate the solution to MSTA found by an allocation protocol to be comparable to the centralized solution.

### C. Hybrid architecture

In view of the relative strengths and weaknesses of the fully centralized and fully distributed architectures, a third option is to combine aspects of both into a hybrid architecture, shown in Figure 7. This is essentially the distributed architecture connected with a base station having a central allocation engine as described in Section IV-A. The motivation behind this architecture is that when there is a connection to the base station users can post new tasks directly to the central server and wait for an allocation decided by the central allocation engine, and instead when the connectivity to the base station is interrupted (e.g. because of telecommunication network overload or physical damage) the system is able to find autonomously a feasible allocation using the KB bundle generator and allocation protocol on their mobile device.

This approach is prone to race conditions where the central and distributed mechanisms interfere with each other. For example, consider the case where a mobile device $A$ loses connection to the base station while a neighbour's device $B$ remains connected. In this case device $A$ will start to run the local allocation protocol (after having found a set of feasible sensor bundles through its local KB bundle Generator, while

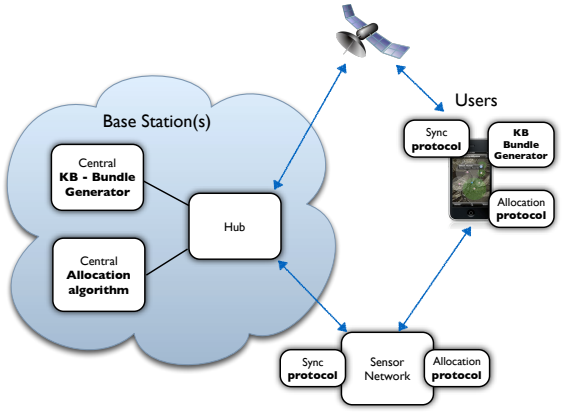


Fig. 7. Hybrid system architecture.

$B$ will post its newly-created task to the central server. The central server would need to know that the sensors in the surroundings of the disconnected user ($A$) are currently running a local allocation protocol to avoid allocating sensors twice to different tasks. We would therefore need a synchronisation protocol to prevent such race conditions, which comes at a cost in terms of increasing network traffic. Moreover, such conditions undermine the advantage of partial centralization: the better solution obtainable from the centralised system is not guaranteed given that the central and distributed mechanisms would be working in potential conflict.

A more desirable hybrid architecture is shown in Figure 8. The basic idea is to provide the base station with the big picture of what is happening on the field leaving the allocation to the autonomous distributed system composed by sensors and mobile user devices. To allow this, either the mobile devices or the sensors should send an update on the current situation to the base station when something changes in the network. A few examples are: when a bundle of sensors is allocated to a task created by a user, either the sensors in the bundle or the mobile device of the owner of the task should notify the central server of the new allocation; the same thing should happen if either a sensor or a user is out of communication range with the sensor network maybe due to hardware failure or battery depletion. This approach would generate more traffic in the sensor network which would shorten its lifetime, but implementing this would allow for the base station to collect relatively live data which could also then be used for command and control, and subsequent replay and analysis of the entire operation.

## V. Discussion and Conclusion

The previous section considered the various architectural options largely from a technical point-of-view. We can also consider them in terms of their fit to coalition command and control structures, and levels of command. In a military context, for example, the centralized approach fits well with a high-echelon command centre, where decisions are made on the deployment or intelligence, surveillance, and reconnaissance (ISR) assets across a whole coalition. Here,

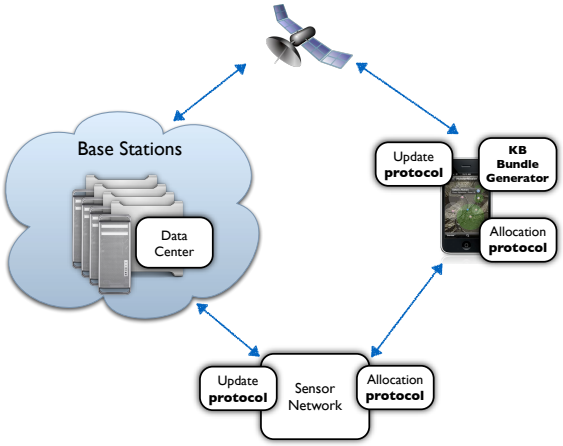


Fig. 8. Hybrid system with remote data logs.

tasks are submitted in the form of "requests for information" from local commanders on the field, and the ISR command centre attempts to assign assets for the good of the coalition operation as a whole. The command centre is typically staffed by ISR experts who have knowledge of all appropriate means to satisfy particular kinds of task. In our approach, much of this knowledge is encoded in the KB bundle generator. The aim here is not to replace human expertise, but to assist commanders when the number of tasks and assets is large, and demand (tasks) exceeds supply (assets) requiring trade-offs to be made in terms of utility, and the allocation problem therefore requires automated assistance.

In contrast, the fully distributed approach most obviously resembles a case where a local commander has a number of ISR assets in their vicinity, and aims to choose between these to satisfy the task at hand. For example, this could be a platoon commander with a small UAV or ground vehicle at their disposal. However, the local commander does not usually have knowledge of all the ways a particular task can be satisfied. In this case, having the KB bundle generator available locally to the commander offsets their lack of knowledge. The obvious disadvantage of the distributed approach from a coalition command and control point-of-view is its lack of a "big picture" — an overview of assets and tasks — which is part of our motivation for proposing the hybrid architecture.

We have evidence from our previous work [14] that a distributed mechanism can handle sensor allocation among heterogeneous tasks effectively. In that work, the association between task types and utility models was hard-coded, rather than being given by an extensible knowledge base. We are currently conducting experiments to compare solution quality between the fully centralized architecture and the fully distributed approach, using the lightweight implementation of the KB bundle generator deployable on mobile devices.

**Acknowledgement** This research was sponsored by the U.S.Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF- 06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

## REFERENCES

[1] E. Balas and M. W. Padberg. Set partitioning: A survey. *SIAM Review*, 18(4):710–760, Oct. 1976. ArticleType: primary_article / Full publication date: Oct., 1976 / Copyright 1976 Society for Industrial and Applied Mathematics.

[2] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: optimal and approximate approaches. In *Proceedings of the 16th international joint conference on Artifical intelligence - Volume 1*, pages 548–553, Stockholm, Sweden, 1999. Morgan Kaufmann Publishers Inc.

[3] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in Multi-Robot systems. *The International Journal of Robotics Research*, 23(9):939, 2004.

[4] M. Gomez, A. D. Preece, M. P. Johnson, G. de Mel, W. Vasconcelos, C. Gibson, A. Bar-Noy, K. Borowiecki, T. F. L. Porta, D. Pizzocaro, H. Rowaihy, G. Pearson, and T. Pham. An ontology-centric approach to sensor-mission assignment. In *EKAW'08*, volume 5268 of *Lecture Notes in Computer Science*, pages 347–363. Springer, 2008.

[5] K. L. Hoffman and M. Padberg. Solving airline crew scheduling problems by Branch-and-Cut. *Management Science*, 39(6):657–682, June 1993. ArticleType: primary_article / Full publication date: Jun., 1993 / Copyright 1993 INFORMS.

[6] M. Johnson, H. Rowaihy, D. Pizzocaro, A. Bar-Noy, S. Chalmers, T. La Porta, and A. Preece. Frugal sensor assignment. In *DCOSS '08*.

[7] M. P. Johnson, H. Rowaihy, D. Pizzocaro, A. Bar-Noy, S. Chalmers, T. L. Porta, and A. Preece. Sensor-Mission assignment in constrained environments. *IEEE Transactions on Parallel and Distributed Systems*, 2010.

[8] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 2007.

[9] K. H. Low, W. K. Leow, and M. H. A. Jr. Autonomic mobile sensor network with self-coordinated task allocation and execution. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36(3):315327, 2006.

[10] J. Ostwald, V. Lesser, and S. Abdallah. Combinatorial auction for resource allocation in a distributed sensor network. In *RTSS '05*.

[11] R. K. Panta, I. Khalil, and S. Bagchi. Stream: low overhead wireless reprogramming for sensor networks. In *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, page 928936, 2007.

[12] D. Pizzocaro and A. Preece. Towards a taxonomy of task allocation in sensor networks. In *Proceedings of the 28th IEEE international conference on Computer Communications Workshops (INFOCOM)*, pages 413–414, Rio de Janeiro, Brazil, 2009. IEEE Press.

[13] H. Rowaihy, M. Johnson, A. Bar-Noy, T. Brown, and T. L. Porta. Assigning sensors to competing missions. In *IEEE Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008*, page 16, 2008.

[14] H. Rowaihy, M. Johnson, D. Pizzocaro, A. Bar-Noy, L. Kaplan, T. L. Porta, and A. Preece. Detection and localization sensor assignment with exact and fuzzy locations. In *DCOSS'09, Marina Del Rey, California, USA*, pages 28–43, June 2009.

[15] T. W. Sandhlom and V. R. T. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1-2):99–137, July 1997.

[16] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: a fast optimal algorithm for combinatorial auctions. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*, pages 1102–1108, Seattle, WA, USA, 2001. Morgan Kaufmann Publishers Inc.

[17] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, May 1998.

[18] V. V. Vazirani. *Approximation algorithms*. Springer Verlag, 2001.

[19] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In *Proceedings IEEE INFOCOM 2005.*, volume 2, 2005.