Demo: A Distributed Architecture For Heterogeneous Multi Sensor-Task Allocation

Diego Pizzocaro, Alun Preece
School of Computer Science
and Informatics
Cardiff University, UK
Email: D.Pizzocaro@cs.cardiff.ac.uk

Fangfei Chen, Tom La Porta
Dept. of Computer Science
and Engineering
Penn State University, US

Amotz Bar-Noy
Dept. of Computer Science
Graduate Center
City University of New York, US

I. Abstract

To complement our paper "A Distributed Architecture For Heterogeneous Multi-Sensor Task Allocation" published in DCOSS 2011 conference proceedings, we will demonstrate the current implementation of our architecture for Multi-Sensor Task Allocation (MSTA) [2] shown in Figure 1. This consists of a prototype interface for task submission, and a simulated sensor network serving other mobile users on the field competing for the same sensing resources. The prototype interface is implemented on two different mobile devices (iPhone & iPad[1]), dynamically interacting with a heterogeneous sensor network simulated in Java on top of the REPAST Symphony[2] discrete time simulation environment. A video of the demo is available at *http://www.youtube.com/watch?v=QzrpKRhGFRU*.

II. MSTA problem & Architecture

Heterogeneous sensor networks are increasingly deployed to support users in the field requiring many different kinds of sensing tasks. There may be multiple alternative kinds of sensors suitable for a given task. Sensing tasks might compete for the exclusive usage of available sensors. Such an environment is highly dynamic with users moving and generating tasks at different rates. Users typically lack the time and expertise to manually decide which are the best sensors for their tasks. We need therefore to design a distributed system to automatically allocate sensors to the tasks they best serve, considering the task information requirements and the sensor capabilities. We refer to this problem as *Multi-Sensor Task Allocation* (MSTA).

The architecture to solve the MSTA problem is comprised of four main components as shown in Figure 1: a mobile user in the field, a Knowledge-based bundle generator (KB bundle generator), an allocation protocol, and the sensor network. The mobile device represents the point of entry of tasks into the system: i.e. the mobile user submits sensing tasks through a mobile app interface, by specifying a local area-of-interest (represented by coloured circular areas in the interface in Fig. 1) and its information requirements. As an illustration-of-concept, we prototyped the interface to the system on Apple iPhone and iPad motivated mainly by their wide adoption and the quality of the development tools provided. The *Knowledge-based bundle generator* on the user device recommends bundles of sensors that are known to be "fit-for-purpose" for that particular type of task specified by the user. The user device then communicates directly to the sensors (e.g. using WiFi or Bluetooth) the best sensor bundle which would be needed to satisfy its newly created task. Then, the sensors autonomously negotiate through the *allocation protocol* which one is the best task to serve as part of a bundle, thus finding a solution to the MSTA problem instance. Note that a single sensor bundle fully satisfies the sensing requirements of the task, therefore we have a one-to-one relationship between sensor bundles and tasks. Finally, the sensor network is configured accordingly and begins serving tasks by delivering sensor data to users.

[1]http://www.apple.com/iphone/ & http://www.apple.com/ipad/
[2]http://repast.sourceforge.net/ - checked 15th April 2011

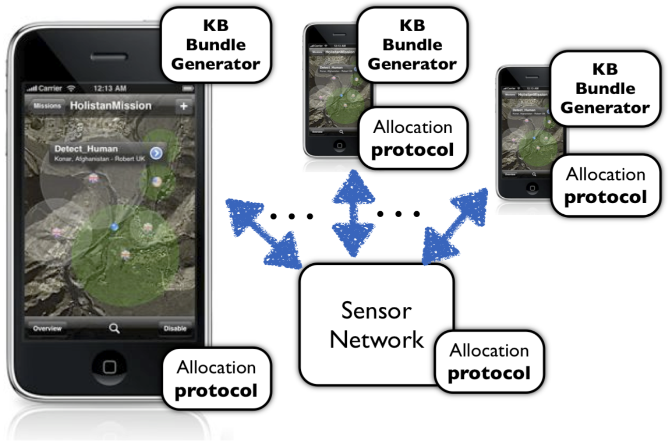


Fig. 1. Fully distributed system architecture.

III. Knowledge-Based Bundle Generator

Figure 2 shows the reasoning process enabled by the Knowledge-Based Bundle Generator in detail. For each newly created task T_j at time step t the KB recommends a Joint Utility Model (JUM) to compute the sensor bundle utility and a Bundle Type (BT) to select the sensor types compatible with the task. The identification of Bundle Types was originally implemented as described in [1], by dynamically matching the sensing capabilities provided by each single sensor and

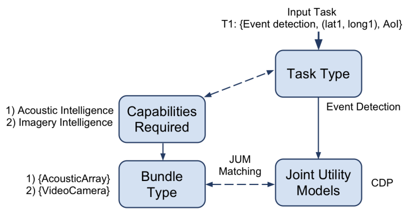
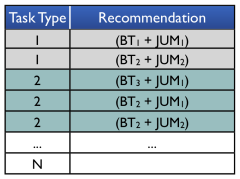


Fig. 2. Reasoning procedure.



Task Type	Recommendation
1	$(BT_1 + JUM_1)$
1	$(BT_2 + JUM_2)$
2	$(BT_3 + JUM_1)$
2	$(BT_2 + JUM_1)$
2	$(BT_2 + JUM_2)$
...	...
N	

Fig. 3. Lightweight KB bundle generator.

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[3]. The output of this step is a set of Bundle Types, where each of the entry is composed by a set of sensor types which altogether would satisfy the information requirements of the sensing task. To determine the number of instances of each sensor type assigned to a task we use a Joint Utility Model. Such 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 utility function which is used to compute the estimated value of utility for a group of sensor instances implementing the recommended BT. The KB indicates which JUMs are appropriate for each task type. Each JUM is only compatible with certain sensors, so the final step in the reasoning procedure is to match applicable JUMs with BTs.

The original implementation of the reasoning process is computationally expensive [1] 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 during an operation) 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. The only assumption that this approach makes is that the device will have a sufficiently large storage capacity, which is reasonable for modern mobile devices. The structure of the look up table is represented in Figure 3. Note that each row of the table is a tuple composed by a Task Type (represented as an ID), a Bundle Type and a JUM.

IV. Allocation Protocol

We used the *disjoint coalition formation protocol* in [3] as a starting point to design the allocation protocol in our architecture. The protocol runs on the two main entities composing our distributed system: sensors and user devices. When the user creates a task, the user device computes *feasible sensor bundles* and their *joint utilities* using the Knowledge-based 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 protocol consists of two main stages: in the preliminary stage - which we call *initial negotiation* - the user devices compute and distribute the bids to the sensors by first discovering which sensors are good candidates for each task; in the main stage — called *bundle formation* — the sensors decide upon which bundle to join in order to serve a particular sensing task. In addition we allow for preemption of sensing resources from ongoing tasks and a rebidding mechanism. A detailed description of the protocol is contained in our DCOSS 2011 paper.

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

V. The Demo

The demo will show the prototype mobile devices submitting sensing tasks to our simulated heterogeneous sensor network. The sensors and user devices will autonomously decide if the tasks can be served, considering also the simulated mobile users on the field who might compete for the same sensing resources. As described in our paper, our main contribution is an extension of a pre-existent well known coalition formation protocol [3] to implement a distributed system solving our MSTA problem instance. The novelty of our architecture consists in a layered approach which by integrating a knowledge base with the allocation protocol provides flexibility in the choice of sensors to use in order to satisfy the users' task requirements. Therefore we solve the allocation problem taking into account both qualitative and quantitative measures.

VI. Logistical Requirements

We will provide mobile devices (iPhone & iPad) and a laptop. The demo requires a wireless internet connection available to all three devices. A large screen would be desirable.

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. We acknowledge also Matthew P. Johnson & Yosef Alayev (CUNY), Matthew J. Williams & Konrad Borowiecki (Cardiff University).

References

[1] 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.

[2] 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.

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