DISTRIBUTED WIRELESS AD HOC GRIDS WITH BANDWIDTH CONTROL FOR COLLABORATIVE NODE LOCALISATION

Elisa Rondini and Stephen Hailes
University College London
London, UK

and

Li Li
Communications Research Centre Canada
Ottawa, Canada

ABSTRACT

In this paper, we investigate the problem of distributed computing in tactical Wireless Sensor Networks (WSNs). The constituent nodes of such WSNs are individually resource poor, but are collectively capable of carrying out significant calculations, if appropriately orchestrated. We propose a general Distributed Wireless Ad hoc Grids (DWAG) paradigm, exploiting concepts from the field of computational grids and modifying them in order to respect the constraints inherent in ad hoc sensor networking. In particular, we use our proposed paradigm to implement the CCA-MAP localisation algorithm [1] on a real WSN test-bed. We investigate the impact of radio interference on the performance of the system, and propose a technique to ameliorate its effects. Our experimental results, conducted using different network topologies, demonstrate that it is possible to achieve significant performance improvements in terms of job latency by considering both the computing load and the network constraints.

I. INTRODUCTION

In emergency rescue and military operations, tactical Wireless Sensor Networks (WSNs) are often deployed to gather in-field location and event information, an inherently distributed task that typically requires filtering, fusion and aggregation of data from multiple sensors. Such processing is typically performed within the network, to avoid overly congesting the shared radio medium by sending large amount of raw data to a central server. A class of applications taking advantage of distribution is the one providing localisation. Some of the best performing tactical localisation schemes, including cooperative Multi-Dimensional Scaling (MDS) [2] and Curvilinear Component Analysis (CCA) [1,3,4], produce satisfactory location estimates by employing distributed algorithms. However, such algorithms often incur a non trivial computational cost of $O(n^3)$ or $O(n^2)$. Previous work in this area has been focused on improving the location computing accuracy. In particular, these localisation schemes have mostly only been evaluated using simulators running on powerful machines, rather than in a real WSN [1,2,3,4]. Simulations also make strong assumptions on high computational (CPU/memory) capabilities and ideal radio network conditions [1,2,3,4]. Thus, the actual feasibility of implementing such complex distributed algorithms in a tactical WSN remains a question yet to be answered.

One might argue that a simple centralised implementation of the algorithms would avoid the issues of distribution. However, in many cases, a centralised implementation can overburden the sensor network, especially the links close to the central server. In other cases, the architecture of the algorithms is inherently distributed and thus centralisation is not a viable solution.

In this work, we develop a novel Distributed Wireless Ad hoc Grids (DWAG) paradigm that integrates Grid Computing principles with WSNs, to implement on real sensor networks an existing collaborative localisation algorithm (CCA-MAP [1]), previously tested solely on a centralised simulator. In this work, our specific contributions include the following:

1. We propose a novel, general approach to implement distributed algorithms in tactical WSNs formed by resource-constrained devices, through creating the DWAG. We implement the complex cooperative localisation algorithm CCA-MAP in a real distributed WSN, employing the DWAG paradigm.

2. We investigate the impact of radio interference on the performance of the system, and propose a Bandwidth-Aware Task Scheduling (BATS) scheme that significantly improves the performance. BATS load shares location-computing tasks among sensors by assessing both node computational capabilities and local network conditions. Our experimental results, conducted over different radio and network topology configurations, prove that significant performance improvements in terms of job latency can be achieved by adopting the BATS strategy.

The rest of the paper is organised as follows. Section II presents the DWAG paradigm. Section III describes the scenario and the application of the DWAG approach to the cooperative sensor localisation scheme presented in [1]. Section IV analyses the load distribution problem and its effects using experiments. It then presents the improved performance results obtained by applying our novel BATS scheme during load distribution. Finally, Section V discusses related work and Section VI concludes the paper.

II. A NOVEL PARADIGM FOR WSNs

A. WSNs Implementation Issues

In an earlier investigation, to verify the feasibility of porting an algorithm exclusively evaluated using simulations
to a real-world WSN, we selected the distributed localisation scheme from [5]. Though not particularly meeting the requirements of high accurate location results for tactical scenarios, this distributed localisation scheme is less computationally intensive than other highly accurate ones [1,2]. We implemented the approach onto a Tmote Sky\(^1\) device and we found that the computing code assigned to each sensor node could not be entirely supported by it. In particular, the following issues were encountered:

**Memory:** The localisation code assigned to each sensor node cannot fit within the memory of a single device. Thus, the code had to be split into two parts (see Table 1). The sensor was reprogrammed with each part in turn, with the partial results held in the external memory.

**Computation:** The MSP430 16-bit micro-controller does not support floating point libraries (nor any mathematical function more complex than the basic arithmetic operations) by default. However, floating point numbers are invariably used to achieve better localisation accuracy. Thus, we implemented a separate floating point library, again occupying scarce memory resources at runtime.

**Speed:** The results presented in Table 1 show the average time taken by a Tmote Sky sensor to compute each of the code parts with different numbers of neighbouring nodes. The results show that localisation can only practically be performed using this approach for very small node degrees, when delays of over a minute are acceptable and the nodes are exclusively dedicated to this task. Such an implementation is not viable for tactical military WSN scenarios, as it is too susceptible to prolonged computing delay and even failure.

Table 1. Statistics collected by porting the localisation application presented in [5] onto a Tmote Sky sensor device.

<table>
<thead>
<tr>
<th>Avg. Node Degree</th>
<th>1(^{st}) Part</th>
<th>2(^{nd}) Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.799</td>
<td>71 sec.</td>
<td>1 sec.</td>
</tr>
<tr>
<td>7.400</td>
<td>195 sec.</td>
<td>Not Terminated</td>
</tr>
<tr>
<td>8.199</td>
<td>255 sec.</td>
<td>Not Terminated</td>
</tr>
</tbody>
</table>

This porting experiment has further verified our assumption that some very essential functions in a tactical WSN, such as localisation, cannot be supported by simply porting the algorithm code to a resource poor sensor node. Given the trade-off between computational power, battery capacity and form factor, either a different implementation approach must be adopted or a more capable sensor node must be employed, or all the data must be shipped to a node that is more capable for processing.

One may argue for more powerful nodes in a tactical WSN to solve the implementation problem. It is worth noting that once procured, military hardware tends to be employed for a considerable time, since the assurance processes that drive procurement are time consuming and expensive. Thus, given that localisation is only one example (albeit a useful one) of the applications in WSNs, and that the uses of all technologies have tended to become more sophisticated over time, there is likely to come a point in the operational lifetime of any equipment in which nodes may be unable individually to compute the jobs that one wishes to allocate to them. In short, any deployment solution that relies solely on tailoring hardware to the requirements of existing algorithms is likely to be less effective in the long term than one that can make use of the collective power of the whole system.

For similar reasons, a requirement to ship unprocessed data to computing servers inevitably leads to network contention, increasing the potential of causing congestive network collapse, especially in the emergency case when mobile sensor nodes must be constantly localised and events must be reported in near real time. Once again, adequate support for distributed computation is required to extend the operational capabilities of a deployed system.

### B. Distributed Wireless Ad hoc Grids

To provide a general implementation platform for applications that are distributed over WSNs, and to study the effects of such distribution, we devised the Distributed Wireless Ad hoc Grids (DWAG) paradigm.

Assume a typical tactical WSN scenario, as illustrated in Fig.1. In this scenario, a large number of sensor nodes are distributed throughout a given space: some are static and pre-deployed, some are mobile (e.g., sensors worn by operational personnel, or carried on UAVs). Each node may choose either to execute a computing job by itself, or to distribute the job tasks to its neighbours. An essential issue encountered here is that the neighbourhood changes over time, and there is potentially a complex set of interactions that can result from the decision to offload a task. Not only does offloading a task have a cost in terms of transmission delay, but it may increase latencies for unrelated tasks, when occupying the shared broadcast radio medium and denying that medium to others.

We propose the DWAG paradigm, where nodes form dynamic ad hoc grids with nodes that are physical neighbours, and for whom there is an expectation of continued connectivity over the duration of the interaction. In Fig.1, the node responsible for the creation of the DWAG is said to assume the role of **Grid Activator (GA)**, whilst the auxiliary nearby nodes taking part in the ad hoc grid and responsible for sharing task execution will act as **Task Executors (TEs)**. Resource-intensive jobs are split (where possible) into smaller tasks and are then offloaded from GAs to TEs to create node-centric DWAG. In deciding to

---

\(^1\)http://www.moteiv.com/
which of the TEs to distribute tasks, the GA considers the computing cost, as well as the network cost of task offload, as measured by levels of network utilisation at the TEs. The result of such computations will be filtered/fused/aggregated information that is a summarisation of the data from which it was calculated, requiring less bandwidth to transmit throughout towards Command & Control (C2) than would the raw information.

Figure 1. Example of a tactical WSNs scenario involving nodes’ localisation to which the DWAG paradigm is applied.

In order to apply the DWAG paradigm to different sets of applications, three basic elements are required:

(1) It must be possible to parallelise applications, either by hand or automatically.

(2) The code for such tasks must be loaded onto the nodes in the WSN. This can happen in a number of ways: typically, code will be distributed, or in-execution tasks migrated, from the GA to the TEs on-the-fly. Clearly, this approach relies on the existence of an operating system or middleware capable of dynamically loading and linking code (i.e., [6]). Other approaches include pre-deployment of useful code fragments to nodes that are likely to be used as TEs, or the retrieval of such code from a repository. For reasons of space and simplicity, we confine ourselves to code pre-deployment within this paper.

(3) It must be possible to estimate the resource usage of tasks. Such a task profile can be determined statically, as a result of static analysis, or on the basis of dynamic profiles that are established and maintained through task execution. There is good reason to believe that such profiling is applicable to this environment. Tactical WSNs are not a general purpose computing resource and, although the superset from which extant tasks are drawn is expected to evolve over the long term, it will change relatively slowly. More- over, tasks in WSNs are frequently rather repetitive data analytic tasks. Thus, the conditions for both static analysis (slowly changing and restricted tasks amenable to pre-deployment code analysis and simulation) and dynamic analysis (repetitive tasks) hold.

At runtime, the GA starts the algorithm execution and, when locally overloaded, enters into the load distribution phase in which it negotiates to offload the specific tasks to TEs. The GA compares the profiles of the tasks with the information about candidate TEs’ current conditions, selects and then triggers task executions on TEs. Finally, TEs send the obtained results back to the GA, which aggregates them into the final result.

III. DWAG FOR NODE LOCALISATION

We now apply the DWAG paradigm to implement the distributed cooperative localisation scheme CCA-MAP [1,3,4] on a deployed WSN. CCA-MAP was chosen over others [5] because it is more challenging both from a computational and a communication point of view. This allows us to stress more our DWAG paradigm. CCA-MAP is also more flexible and robust, without requiring directional antennas or assuming that sensors already have an estimate of other node locations. CCA-MAP [1,3,4] applies the Curvilinear Component Analysis (CCA) to accurately estimate node positions in tactical WSNs where GPS support cannot be assumed. The CCA procedure is employed locally by each node to compute its own local map. Then, all the local maps created by each node in the network are patched together to form a global map. The CCA-MAP algorithm represents a typical and suitable candidate for the application of our DWAG paradigm because: (i) it is inherently distributed; (ii) it involves a computationally heavy phase (i.e., global map patching) which can benefit from nodes collaboration; and (iii) it is time sensitive, since out-of-date location information is of limited use.

All the original simulations for CCA-MAP [1,3,4] were run in a centralised way using Matlab V7.2 on a 1.60GHz Pentium M processor with 1GB RAM. On the contrary, in this work we implement the complete CCA-MAP localisation scheme on Tmote Sky sensors, each of which has a MSP430 8MHz 16-bit micro-controller, a ChipCon CC2420 radio module supporting IEEE 802.15.4 (with 250kbps as maximum bandwidth), 10kB RAM and 48kB Flash ROM. We thus use devices that are $O(200)$ times slower and $O(100,000)$ times less capable than the simulating computer.

We then map CCA-MAP algorithm requirements over the main actors of the DWAG as follows:

(a) Grid Activator: the GA drives the execution of the localisation job $J$ that has been split into several tasks $\{t_1, t_2, ..., t_M\}$. The GA performs the following tasks:

Task 1 ($t_1$): the GA broadcasts a message to discover its neighbours and collects from them their adjacency matrices. This phase was absent from the original algorithm implementation presented in [1,3,4] because of the centralised way in which it was simulated.

Deliverable: the GA establishes its own neighbourhood.

Almost all previous works proposing cooperative localisation schemes \([1,2,3,4]\) have been evaluated exclusively by running simulations with strongly simplified assumptions, especially regarding radio communications. For example,

**Task 2** \((t_2)\): the GA computes its own local map by applying the CCA procedure. In addition, the GA chooses among its neighbours (those stored in its adjacency matrix) the one that shares the greatest number of neighbours with itself, requests and obtains its local map.

**Deliverable:** the GA has two maps to be patched together.

**Task 3** \((t_3)\): the GA patches a global map in its own reference system by merging neighbouring node map with its own local one. This is done by using the Singular Value Decomposition (SVD) transformation matrices computed and received back from the elected TE node.

**Deliverable:** the GA outputs a global map containing its own and its neighbours’ computed locations.

**Task 4** \((t_4)\): the TE is elected to receive two maps from the GA and to perform the SVD transformation on them.

**Deliverable:** the TE outputs three transformation matrices that are then sent back to the GA to compute a global map.

Referring to Fig.1, each node that needs to patch together its global map acts as a GA in order to form an ad hoc grid with its neighbours. At the same time, the same GA can participate in other surrounding ad hoc grids (e.g., created by other GAs) by sending its local map. A node can thus have different roles in the flows of several applications, collaborating with each other in achieving better global performance under the current condition of the system.

### IV. SYSTEM PERFORMANCE EVALUATION

In evaluating the implementation of the CCA-MAP localisation scheme over our DWAG paradigm, we demonstrate: (i) that congestion of network links has a significant impact on task execution performance; (ii) that it is nevertheless possible to adopt simple bandwidth-aware task scheduling mechanisms to cope with the radio communication issues; and (iii) that considerable performance improvements can be gained during the task distribution process if one has even rudimentary knowledge about the state of the network.

#### A. The Bandwidth Problem

Before presenting our experimental set-up and measurement results, we highlight the following network bandwidth issues that we have encountered, which have a significant impact on the system performance.

**(1) Message collisions and radio interference generate a high probability of packet loss and corruption.** In WSNs, radio signal interference and very limited bandwidth often result in message collisions, packet corruption and data loss. It was thus necessary to add extra control code during the implementation to check whether the content of the exchanged messages (i.e., location maps) was corrupted. Such code adds to the computational load and latency of the algorithm.

**(2) Unpredictable hardware failures require control mechanisms.** On occasion, messages fail to reach the destination not only because of network collisions and interference, but also due to the hardware unreliability (e.g., temporary antenna failures, corrupted registers in the ChipCon CC2420 radio module). We therefore implemented several additional checks through the application to verify that the results were always correct and the lower levels (i.e., radio or memory) were behaving as expected. In case of errors, mechanisms were activated to perform corrupted data recovery, retransmission of matrices or even re-initialisation of sensor modules.

**Issues (1) and (2) significantly show that communication indirectly affects computational cost.** It is thus essentially important to account for network impact when evaluating the performance of distributed algorithms in WSNs.

**(3) Load distribution in the ad hoc grid needs to consider traffic congestion effects.** The very limited bandwidth resource and the resulting rather poor network support in the WSN make it critically important to consider network conditions when distributing computing tasks. For example, assume a GA node that needs to distribute a localisation task (see Fig.2) to two potentially selectable TEs (e.g., TE1 and TE2). TE2 is rather less loaded than TE1, but the latter is positioned in an environment with much lower network utilisation. This makes the choice of destination difficult, since job execution latency is affected by both factors. Moreover, a distributed task might itself add to the congestion of an area in several ways: through the traffic resulting from the initial distribution of data, through subsequent communication between nodes during the execution, and through control overhead. Thus, if the GA distributes a large task with substantial I/O requirements into a traffic heavy area, it will not only cause that task to run more slowly, but it will also retard other tasks either within the area, or using the area for communication.

![Figure 2. The bandwidth problem.](attachment:image379x150to512x226.png)
simulators often assume that sensor radio transmissions are error-free, unaffected by network interference, with a circular transmission radius, and bidirectional. However, it is widely known [7,8,9,10,11,12] that radio propagation (i) is non-isotropic, (ii) has non-monotonic distance decay, and (iii) results in asymmetric links. Thus, common simulator assumptions do not stand when implementing cooperative localisation algorithms in real WSN environments. Those simplified assumptions also make the validity of the work very questionable for even a small number of nodes [7]. It is thus crucial to handle the real radio communication effects in WSNs, to support localisation or other computing functions, as we are aiming to achieve in this work.

B. Bandwidth-Aware Task Scheduling

Driven by the issues of poor network capabilities described above, we devise a Bandwidth-Aware Task Scheduling (BATS) strategy that distributes localisation tasks according to both TEs’ computational capabilities and their local network conditions.

For task distribution, we selected a reactive, sender-initiated load sharing technique, namely the Auction algorithm. In our previous work [13], we conducted a comparative analysis between reactive and proactive scheduling schemes. The experimental results showed that whilst their performance were comparable, the Auction algorithm appeared to be much more flexible with respect to both the number of tasks of which a job is comprised and the number of GAs operating within the environment.

In the Auction algorithm, state information exchange is handled reactively. Each job is split into tasks; for each of these, the GA broadcasts a task request message containing the details of the data type (attributes that the peer has to match), the task CPU and estimated bandwidth requirements. Upon receiving the task request, each TE that finds itself meeting the task requirements sends a bid to the GA containing its CPU and bandwidth details. Once the GA has received the bids, it chooses the best TE, and starts to offload the task to the winning TE, which consequently launches a process to compute the task and returns the result to the GA. Multiple requests are handled by the TE on a First-Come First-Served (FCFS) basis.

Most of the existing load distribution methods in Grid Computing do not consider the bandwidth conditions in the underlying network [18], because they assume high-bandwidth network links. Thus, to adapt our DWAG paradigm to tactical WSNs, we propose and integrate within the Auction algorithm a novel BATS mechanism. In Eq.1 illustrated below, given a GA, for each one of its TEs $i$, a linearly weighted score function $S(i)$ is calculated on which to perform node selection. $C(i)$ and $B(i)$ are defined as the CPU and bandwidth availability of $i$, and $w_C$ and $w_B$ are weights associated to them, respectively. The TE with the highest score is selected for the task distribution.

$$S(i) = w_C * C(i) + w_B * B(i) \quad i = 1,...,N$$

When $w_C \neq 0$ and $w_B = 0$, Eq.1 considers only the computing resources of the candidate TEs. When $w_C \neq 0$ and $w_B \neq 0$ in Eq.1, network utilisation is also considered.

C. Experimental Set-Up

Our experimental test-bed consists of Tmote Sky sensors running the Contiki OS [14]. For our experiments, we used the fully distributed Heterogeneous Experimental Network (HEN) sensor test-bed deployed at the Dept. of Computer Science at UCL. Sensors are distributed in an indoor open plan area 20m long and 18m wide, as shown in Fig.3.

![HEN sensor test-bed at UCL.](http://www.cs.ucl.ac.uk/research/hen/)

The CCA-MAP localisation scheme is implemented over this test-bed using DWAG mappings, as described in Sec.III. In all the experiments, actual computation and actual network traffic were used. GAs run a localisation job that has been split into several tasks (i.e., $\{t_1,...,t_4\}$). Some of the tasks are directly executed by GAs (i.e., $\{t_1,t_2,t_3\}$), while the execution of the heaviest is distributed to TEs (i.e., $\{t_4\}$). Three communication phases are included in our experiments during task distribution: (i) offload of the data (but not code) associated with tasks from the GA to the TE; (ii) in progress communication exchanges between the GA and the TE needed to progress task execution; (iii) upload of results from the TE to the GA.

In our tests, we used different topological sets of 15 nodes belonging to the HEN test-bed, because this is usually the number either of first responders taking part in a single search and rescue operation, or of soldiers collaborating with each other within a unit of Military Operation on Urban Terrain (MOUT) [15]. Furthermore, the obtained results could be easily scaled to bigger networks with a similar connectivity degree, since their local traffic would be the same. In particular, for each test we selected uniformly distributed network topologies of up to 10 GAs, 4 TEs and 1 streaming node responsible for injecting generic background traffic within the environment. This latter streaming node emulates the presence within the environment of diversified computing tasks and communications that are

3 http://www.cs.ucl.ac.uk/research/hen/
potentially carried out in parallel within the WSN while the highly-demanding localisation computation is performed. This was done both to mimic a realistic and heterogeneous background level of communication, and to inject further real network traffic into the system, in order to study the effects of high network congestion. We set the radio power of the streaming sensor to level 0x03, which provides a physical packet reception range of ~250cm to congest only part of the overall area, reproducing a situation like the one described in Sec.IV.A (see Fig.2).

Job latency is used as our performance metric in all experiments because timeliness of information is vital in tactical scenarios. Moreover, latency captures the effects that tasks from one node have on the execution patterns of others. One may argue that battery life is an important metric. However, in the tactical scenario, battery lifetime is much less of an issue than the timeliness of information. In fact, information from sensors will be most useful within the first few tens of minutes of an incident. In the experiments, sensors were able to run the algorithms while battery powered for more than 48 hours between charges, thus fulfilling the battery requirements of our scenario. Job latency is defined to be the overall time spent by a GA to compute its global map. Task 1 is used by GAs to discover their neighbourhood, a process which is also required by functions such as routing, sensor health monitoring, etc. The adjacency matrices gathered in Task 1 can thus be obtained by simply piggybacking the messages employed by the routing or health monitoring functions. Thus, we did not include the time budget of Task 1 in our measurements.

In a fully connected network, each GA computes a global map by performing sequences of \(\{t_2,t_3\}\), delegating the execution of \(\{t_4\}\) to TEs, until its global map contains the locations of all nodes belonging to the network. It is important to note that this procedure is carried on at the same time by all GAs in the network. Each sample measurement was obtained as the latency of all GAs in the WSN to complete their global map computation, averaged over 30 runs of the experiment.

Although we did not intend to evaluate nor improve the accuracy of the CCA-MAP algorithm, we adopted the same set of input values (e.g., adjacency matrices, local map sizes) as those used in the previous simulations [1,3,4] and verified that sensors were indeed able to compute the positions with equal accuracy as reported by the simulator. Note that this work has no effect on the algorithm accuracy.

Finally, the parameters in Eq.1 were computed as follows. \(C(i)\) is the difference between the maximum number of processes executable on each TE (i.e., 5) and the ones actively running on it at the time of the request. \(B(i)\) is the percentage of local bandwidth availability, calculated as follows. We elected to read the Clear Channel Assessment value from pin 28 of the Chipcon CC2420 transceiver and maintain a sliding window containing information for the last 100 temporal slots in which the radio channel was clear or busy. The percentage was calculated by dividing the \(m\) number of times in which the channel was clear by 100. Thus, since \(C(i)\) is in the range \([0,5]\) and \(B(i)\) in \([0,1]\), we tuned the weights \(w_C\) and \(w_B\) to 1 and 5, respectively, to normalise both contributions within the range \([0,5]\).

D. Performance Results

We conducted four sets of experiments, varying both the number of GAs (i.e., 6 or 10 in Fig.4 and Fig.5, respectively) and the radio transmission power (i.e., \(0x04\) or \(0x1F\) in Fig.4-5 (a) and Fig.4-5 (b), respectively). This latter parameter was varied to generate both a loosely and a fully connected network. In fact, while \(0x04\) provides a radio range of \(~400-500cm\), with \(0x1F\) all the nodes are in range with each other. Thus, the variation of the radio transmission power affects both the size of the exchanged maps and the amount of radio transmissions to which every node is exposed. Furthermore, we changed the number of TEs for each set of the experiments.

![Graph](image)

Figure 4. Performance comparison of the Auction algorithm with and without BATS applied to a network of 6 GAs with radio transmission power level set to \(0x04\) (a) and \(0x1F\) (b).

Fig.4 and Fig.5 illustrate the measurements gathered. In the graphs, we compare the results obtained by using the existing Auction algorithm that collaboratively distributes
tasks exclusively according to TEs' computational capabilities ($w_C \neq 0$ and $w_B = 0$ in Eq.1), with those obtained by using our BATS mechanism in which the bandwidth resource is also considered in addition to the TEs' computational capacity ($w_C \neq 0$ and $w_B \neq 0$ in Eq.1). We thus compared our BATS strategy against the existing Grid Computing approaches where bandwidth is not considered.

Firstly, we examine the average job execution time spent by GAs to compute the global map of the nodes in the system. The time increases (i) when more nodes are involved in the process, because of the increased size of the exchanged local maps and the computed global map; and (ii) when the radio range grows bigger, because of the increased local map size and the increased radio interference. Thus, the latency in networks of 6 GAs with radio transmission power set to $0x04$ (Fig.4 (a)) is substantially lower than in networks of 10 GAs communicating with full radio power $0x1F$ (Fig.5 (b)). However, this behaviour must not lead to believe that having a lower radio transmission power is the best solution. As explained in [1,2,3,4], a bigger map or a greater connectivity degree, resulted from a longer radio range, can generate much more accurate location results.

In all the experiments, the results show that an increase in the number of TEs can be effectively exploited only if the BATS scheme is applied. If GAs are selecting TEs exclusively based on TEs' computational load, more candidate TEs do not lead to any significant performance improvement. In fact, the communication cost (i.e., lost packets and retransmitted messages) in offloading tasks to radio congested, although computationally capable TEs, results in longer overall job latency than selecting a congestion free TE, albeit it may be more computationally loaded.

When applying the BATS scheme, we observe performance improvements by about 22% and 50% (Fig.4 (a)-(b), respectively), for the cases that involved 6 GAs and 4 TEs, compared with the existing solutions without BATS. With 10 GAs and 4 TEs, BATS improves the performance by about 28% and 27% (Fig.5 (a)-(b), respectively). The biggest improvement on latency by about 50% (Fig.4 (b)), is detected when the network includes 6 GAs and 4 TEs, using the radio transmission power $0x1F$ that covers all nodes. In this case, each GA has more candidate TEs from which to choose, thus it can avoid congested ones. Moreover, if compared with larger networks of 10 GAs and 4 TEs where the bigger map size considerably increases the computing latency, in the situation depicted in Fig.4 the communication latency has a relatively bigger proportion in the overall job execution latency. Thus, the save on the communication latency by avoiding congested local area would lead to bigger overall improvements. In addition, 4 TEs assisting 6 GA nodes would experience less overall computing load than serving 10 GA nodes. Note that local maps of 6 nodes was among the recommended configuration for tactical WSNs [1,3,4], hence the improvement brought up by BATS can be practically useful.

These experimental results demonstrate that the DWAG paradigm with the BATS scheduling scheme makes it feasible to implement distributed computing in a practical tactical WSN, for even very complex algorithms. They also demonstrate that the BATS scheme brings significant performance improvements to job execution latency.

V. RELATED WORK

The problem of computation distribution has been extensively studied in the Grid Computing area [16,17,18]. Regrettably, the assumptions behind most existing approaches render them largely unsuitable for use in networks of highly constrained devices (WSNs). Recent research has explored collaborative computation distribution in WSNs. Some of these approaches are inspired by the client-server based paradigm, others by the mobile-agent based paradigm. Both can be combined with cluster based techniques. A range of approaches based on a cluster based computational model can be found in [19,20,21]. They all have in common the view that there is a hierarchical network architecture comprising of a high number of low-cost, less powerful sensors, and a small number of higher-cost, more powerful cluster heads. Such algorithms are
particularly interesting if the structure of the deployed network is indeed hierarchical, and if the geographic distribution of the cluster heads relative to the sensors is appropriate. However, cluster based approaches require often a more careful deployment process which cannot be always guaranteed in an urgent first response or a combat scenario. These cluster based schemes also fail to take into account issues of bandwidth utilisation, message collision and realistic radio modelling: communication is assumed to be collision-free. Furthermore, some of them [20] make two strong assumptions: firstly, when a certain event occurs, all sensor nodes can detect it and collect raw data; secondly, there are no events simultaneously occurring in the field. Moreover, tests are undertaken within oversimplified simulation environments that do not take into account real communication issues. Chiasserini et al. [22] proposed the replication of an algorithm on every node and to split the data to compute among the peers, but they do not tackle the congestion problem. Abrams et al. in [23] studied an optimisation algorithm for the assignment of tasks to microservers, but neither applied it to WSNs nor explicitly considered communications in their model.

VI. CONCLUSIONS

In this work, we presented a novel paradigm, the DWAG for WSNs which is a convergence of mobile ad hoc sensor networks and computational grids. We applied this paradigm to implement a previously published localisation algorithm (CCA-MAP [1,3,4]) on a real Tmote Sky testing network. CCA-MAP, developed for tactical WSNs with desirable performance features for mission critical scenarios, was previously tested on a centralised simulator only. We investigated the impacts of radio interference on the performance of the system, and proposed the BATS scheme to effectively distribute the location computing tasks among the sensor nodes, assessing both their load and traffic conditions. Our experimental results, conducted using different network topologies and radio configurations, prove that significant performance improvements in terms of latency can be achieved by applying BATS.

ACKNOWLEDGEMENT

We would like to thank the Contiki OS team for the technical support. This work is part of the Divergent Grid project funded by EPSRC under grant number EP/C534891. This research is also supported by Defence Research and Development Canada.

REFERENCES