Time-bounded and Space-bounded Sensing in Wireless Sensor Networks

Olga Saukh, Robert Sauter, and Pedro José Marrón

Universität Bonn, Bonn, Germany and Fraunhofer IAIS, St. Augustin, Germany
{saukh, sauter, pjmarron}@cs.uni-bonn.de

Abstract. Most papers on sensing in wireless sensor networks use only very simple sensors, e.g. humidity or temperature, to illustrate their concepts. However, in a large number of scenarios including structural health monitoring, more complex sensors that usually employ medium to high frequency sampling and post-processing are required. Additionally, to capture an event completely several sensors of different types are needed which have to be in range of the event and used in a timely manner. We study the problem of time-bounded and space-bounded sensing where parallel use of different sensors on the same node is impossible and not all nodes possess all required sensors. We provide a model formalizing the requirements and present algorithms for spatial grouping and temporal scheduling to tackle these problems.

1 Introduction

Many wireless sensor network (WSN) applications focus on monitoring scenarios like structural health monitoring of bridges, tunnels and buildings, environmental monitoring and tracking of mobile objects. Complex events, which are common in these scenarios, can only be detected by a combination of several environmental characteristics captured at one point in time and space. Therefore, these applications require sampling and in-network processing of several kinds of complex data which together provide a complete description of a complex event. However, it is practically impossible to install a sensor node with all required sensors to cover every point of the target area. Therefore, distributed wireless sensor nodes usually provide approximations of the description of events.

Although it is often theoretically possible to equip the sensor nodes with all required complex sensors, this might cause several problems. First, each additional complex sensor requires energy, which degrades the lifetime of the individual sensor node and of the whole network. Second, it is often difficult or impossible to trigger and sample several complex sensors at the same time. Moreover, the triggers of two separate sensors activated by an event will not happen simultaneously, which makes simultaneous sampling quite complex. Third, each sensor has its own area of regard. Therefore, some sensors might register abnormality of one environmental characteristic when capturing events, with no confirmation from other attached sensors with a smaller sensing range. Fourth, attaching all sensors to every sensor node still requires much cable to place the

sensors at meaningful locations. For example in bridge monitoring scenario, an acceleration sensor used to measure cable forces must be attached to the corresponding cable whereas an acoustic sensor must be embedded into the bridge floor to be able to acquire acoustic waves which might indicate cracks in the construction. Therefore, in many cases there is a strong need for the distribution of sensors among several sensor nodes and for further in-network cooperation of space-bounded sensors and grouping of time-bounded sensor values in order to detect and characterize an event. Fifth, sensing takes time and for highly dynamic events, like the occurrence of a crack in a bridge, it is impossible to inquire several sensors sequentially within the event duration.

In this paper we assume that every sensor node has one or several sensors attached. These sensors try to capture events that can only be detected within a limited range in space and within a limited period in time. On the one hand, we present algorithms that establish space-bounded non-disjoint groups of sensor nodes which can be seen as one logical sensor node for recognizing an event. On the other hand, we present a scheduling algorithm that allows the distribution of sensing tasks in every group and creates a local task schedule for every sensor on every individual sensor node.

The rest of this paper is structured as follows. In Section 2 we present related approaches. We provide the definition of the problem of time-bounded and space-bounded sensing and discuss the model and assumptions in Section 3. In Section 4 we present algorithms to build groups and generate schedules followed by a thorough evaluation of these algorithms in Section 5. Conclusions and an outlook to future work in Section 6 conclude this paper.

2 Related Work

Event detection is a popular research area in WSN [1–4]. The event detection system presented in [5] allows the detection of composite events in case nodes have heterogeneous sensing capabilities. The results described in [1, 2, 6] provide algorithms for the detection of k-watched composite events, where each event occurrence can be detected by at least k sensors. In [3], the authors concentrate on state transitions of the environment rather than on states only and discuss a generalized view on event detection in WSNs. They model state transitions with finite automata. However, this model is impractical due to its complexity. The authors of [4] consider the problem of describing events or states and state transitions of the environment with an event description language. The main difference of the mentioned works to our approach is that only the spatial characteristic of event detection has been considered. Since an event happens at some point in space and *time*, we also consider its temporal characteristic. Moreover, most papers in this group consider spatial node grouping to increase the confidence of the sensing results. In this paper we group nodes in order to be able to process complex queries and detect complex events.

A number of research projects including TinyDB [7] and Cougar [8] have considered a query-based database abstraction of the WSN. However, these works

assume that the sensor nodes possess the same unordered set of sensors and that the actual access of these sensors is not time limited. Therefore, the sequential execution of the sensing tasks that compose the query on every node is always possible. In this paper we motivate and provide a solution for the case when sequential execution of sensing tasks on every node is impossible.

There are a number of papers that consider the problem of spatial node grouping or clustering [9, 10] in WSN. The usual reason for this grouping is to allow for efficient data aggregation in sensor networks and, therefore, save energy of individual nodes. In this paper we present node grouping algorithms that try to construct the maximum number of complete groups – groups that have all required sensing capabilities to fulfil the query (or detect an event)

The problem of job scheduling is closely related to our work. This problem is usually formulated as follows: Given a directed acyclic graph, where vertices represent jobs and an edge (u, v) indicates that task u must be completed before task v. The goal is to find a schedule of tasks which requires the minimum amount of time or machines. Additional resource constraints on every machine and resource requirements for every task may exist. There are also a number of solutions to this problem in different formulations [11]. However, in this paper we also consider concurrency constraints between tasks, which reduces the applicability of existing solutions to our problem. Therefore, we present a new scheduling algorithm for sensor networks which allows the scheduling of sensing tasks between different sensor nodes taking concurrency constraints between individual tasks into account.

3 Distributed Sensing

3.1 Terminology, Assumptions and Problem Statement

The *sensor network* is usually modeled as an undirected graph $G(V, E)$ embedded into the plane, where V is a set of nodes and E is the set of edges between nodes that can communicate. Every node $v_i \in V$ in this *embedding* $p : V \to \mathbb{R}^2$ has coordinates $(x_i, y_i) \in \mathbb{R}^2$ on the plane. However, as explained later, it is not necessary for our approach that the nodes are aware of their coordinates. We consider that all sensor nodes are embedded within some *target area* $A \subset \mathbb{R}^2$ on the plane. Additionally, the function $v : V \to 2^{\mathbb{S}}$ defines the sensor types each node possesses where $\mathbb{S}$ is the domain of sensor types used in the scenario. This implies that each node contains at most one sensor of each type.

A *query* or *event description* $Q = (S, R, D, C, Pred)$ is defined as a 5-tuple. The set $S \subseteq \mathbb{S}$ describes the sensor types used in the query. The functions $R : S \to \mathbb{R}$ and $D : S \to \mathbb{D}$ describe the sensing range and sensing time respectively for each sensor type in S. We assume a sensing area to be a disc with radius $r \in \mathbb{R}$. The sensing time only describes the actual time needed to access the sensor readings. This duration does not include any data processing, filtering or aggregation actions which can be arbitrarily postponed. Additionally, the concurrency constraints between different sensor types are captured by

$C : S \times S \rightarrow R$. For each pair of sensor types, an element of C defines the maximum duration between starting the sensing of the corresponding sensors. *Pred* is a predicate which maps the sensor values of each sensor type and the combination of these values to {true,false}. We do not further discuss possible definitions of this predicate since we do not extend the large amount of prior work regarding this topic. The combination of these characteristics is extremely important because, for example, a break in a concrete structure event can be captured by an acoustic sensor only within several microseconds in a range of several meters, whereas a fire event results in a temperature increase that can be measured by a temperature sensor within several minutes within a range dependent on the fire event itself (centimetres for a fire of a candle till hundreds of meters for a fire in a forest). We refer to the sensing duration and concurrency constraints and to the sensing range of a query or event description as *event time and space constraints*.

The sensing range of an event with respect to a specific sensor coincides with a sensing range the sensor has and depends on the sensitivity of the sensor. The event duration is, however, usually longer than the sensing duration of the sensor. For example, an acoustic event results in a series of acoustic emissions or fire event lasts longer than needed by a temperature sensor to read the value. The event duration as well as the additional dependencies between the sensing with different sensor types have to be captured by the concurrency constraints.

Our main assumption is that if an event happens at some location $P \in \mathbb{R}^2$ and at some point in time $T_0 \in \mathbb{R}$ and is sensed by a group of sensors which comprises all required sensor types S and satisfies time and space constraints D, C, R with respect to the starting point of the event in time and the point in space respectively, then the sensor readings obtained are a good approximation of the event characteristics. This assumption is very natural for sensor networks.

We define a group of sensors $G_i \subseteq S \times V$, if $\forall (s_j, v_k^i) \in G_i, s_j \in \upsilon(v_k^i)$ and $\forall s_j \in S, \exists (s_j, v_k^i) \in G_i$. This ensures, that only sensor types of a node are used if this node actually possesses the type and that all sensor types required by the query are contained at least once in the group. Let dist denote the Euclidean distance, we say that a group G_i is *space-bounded* with respect to a certain event, if $\forall (s_j, v_k^i) \in G_i, \mathrm{dist}(P, p(v_k^i)) \leq R(s_j)$. This requires that all sensors of each type are in range of the event. However, the definition of a group does not require that for every node all sensor types it possesses are used in the group. This allows the creation of groups where for some nodes only the long-range sensors (e.g. temperature) are used and the short range sensors are ignored. The missing sensor types have to be supplied by nodes closer to the event.

We assume, that parallel sampling of two sensors on the same sensor node is impossible. Two sensors can be accessed sequentially only. Every sensor s_j has its sensing duration $D(s_j)$ – the time the sensor node requires to obtain the sensor readings from this sensor.

We refer to a *local schedule* J_i^{local} for a sensor node v_i as a sequence of sensing tasks that need to be executed for the complete or a fraction of an event description Q. The schedule contains the start times for the access of every

sensor type. As described above, the sensing tasks may not overlap; however, it is possible to insert idle times between two sensing tasks. A local schedule is *time-bounded* if the sequence of sensing tasks fulfils the concurrency constraints C. A *group schedule* J_i for a group G_i is defined correspondingly to the local schedule but includes information for all sensing tasks of all group members. A group schedule is time-bounded if the local schedules it includes are time-bounded and additionally the concurrency constraints C between sensor types on different nodes are met, which can require the inclusion of idle times on individual nodes.

The problem of time-bounded and space-bounded sensing is: *Define a group of nodes and a schedule that enable the execution of the query Q under the given time, concurrency and space constraints.*

In the following subsections we consider the problem of time-bounded and space-bounded sensing separately as well as in combination with distributed coordination.

3.2 Time-Bounded Sensing

Consider the case when an individual sensor node is equipped with all required sensors to process the query Q. It is easy to construct a schedule that sequentially fetches sensor readings from each of these sensors in order to evaluate Q and send the aggregated result to the data sink. Here a simple algorithm can be used to try every possible permutation in order to find a schedule that fulfils the concurrency constraints. Since the number of sensors on a node is usually quite low (≤ 5), it is easy even on resource constrained sensor network devices to enumerate all possible permutations.

Additional concurrency dependencies between the individual nodes must also be satisfied when distributing sensing tasks including the distribution delays. We model the scheduling problem as a graph in which the vertices represent sensing tasks with their corresponding task durations and the edges represent the concurrency constraints between each pair of tasks. The concurrency constraints express the maximum difference in time between the start times of each pair of tasks.

We define the duration $\mathrm{Dur}(J)$ of a schedule as the time from starting to sense the first task to the completion of the last task. Additionally, we define a general metric $\varphi(J)$ indicating the badness of a schedule. We use a badness metric instead of a quality metric since for most intuitive approaches, e.g. the duration of a schedule, higher values mean worse results. Using this metric, we describe the challenge of time-bounded sensing as an optimization problem:

$$\varphi(J_i) \rightarrow \mathbf{min},\ J_i \text{ is a valid schedule as defined above} \tag{1}$$

In particular, J_i must fulfil the concurrency constraints between the execution start times of different tasks. Note that classical scheduling algorithms from the literature do not assume any concurrency dependencies between the tasks.

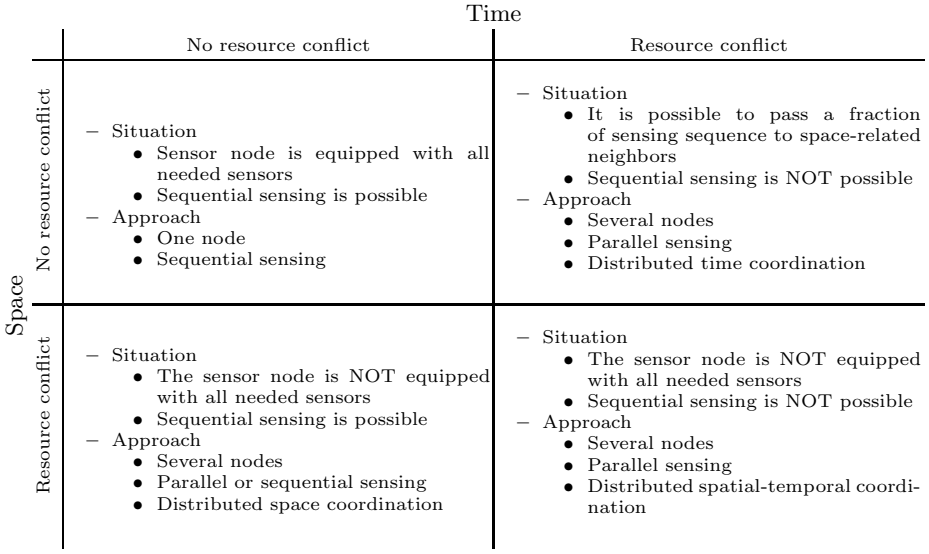


Fig. 1. Problem Space

3.3 Space-Bounded Sensing

If an individual sensor node does not possess all required sensors to fulfil the query or check the event description, popular approaches, e.g. TinyDB [7], fail to consider this node. However, it might be possible to combine two nodes, each with an incomplete set of sensors, located close to each other. From the viewpoint of the application, a space-bounded group plays the role of *one logical sensor node* which possesses all sensors that the group has and allows to perform operations over such space-bounded sensor readings, e.g. calculate if the sensor readings confirm to a certain event description.

Since the location of an event is not known, we introduce a dispersion metric $\psi : S \times V \to \mathbb{R}$. Our approach can work with different metrics. Besides metrics based on actual node coordinates, e.g. the maximum distance from the centroid of the group, metrics based only on some proximity information such as RSSI or the number of shared neighbors are possible. Therefore, we formulate the optimization problem of space-bounded sensing as follows:

$$\psi(G_i) \to \mathbf{min},\ G_i \text{ is a valid group as defined above} \tag{2}$$

3.4 Distributed Coordination

The formulated time-bounded and space-bounded sensing problem considers time and space as limited resources in the context of event detection or phenomena monitoring.

Consider the case when all nodes are equipped with a complete set of sensors. A *time resource conflict* occurs, when it is impossible to acquire an event with a single sensor node via sequential scheduling of sensing tasks due to concurrency constraints. This problem can be solved by intelligent relocation of a fraction of the local schedule to the nodes close-by.

A *spatial resource conflict* occurs if a node does not possess the required set of sensors to process the query. In this case, the event must be captured by in-network cooperation of a group of sensor nodes. If the sequential scheduling on every individual node does not result in any problems, the solution involves group coordination mechanism which partitions the sensor network in space-bounded groups equipped with all required sensors to process the query. Then every group operates as a single logical sensor node.

In real world deployments both problems might occur. In this case, the distribution of sensing tasks in space by space-bounded group coordination and intelligent distributed scheduling of sensing tasks is the only possible solution. Fig. 1 summarizes the described problem space.

4 Algorithms

4.1 Ordering of Sensors

As described in the problem statement, both algorithms have to solve an optimization problem. We introduce a total order among sensor types to greatly reduce the complexity of the algorithms. This order does not pose any unduly restrictions for the query definition, since the priorities can often be derived quite naturally from the characteristics of the employed sensor types. For example, humidity and temperature sensors are not very time sensitive and can be sampled with low priority and are thus among the last of the order. Very complex sensors, e.g. acoustic emission sensors, that may even rely on hardware triggers and thus initiate the complete event detection process, should be placed first in the order.

To express the total order, we redefine a query for our algorithms in terms of vectors and matrices. A query $\tilde{Q} = (\boldsymbol{S}, \boldsymbol{R}, \boldsymbol{D}, C, \mathit{Pred})$ on n sensor types comprises the sensor types $\boldsymbol{S} = (s_1, \ldots, s_n), s_i \in \mathbb{S}$, the associated sensing ranges $\boldsymbol{R} = (r_1, \ldots, r_n), r_i \in \mathbb{R}$ and sensing durations $\boldsymbol{D} = (d_1, \ldots, d_n), d_i \in \mathbb{R}$ as well as a symmetric $n \times n$ matrix C where each element $c_{ij} \in \mathbb{R}$ defines the maximum duration between the start times of sensor type i and sensor type j.

Additionally, we reduce the number of allowed sensors of the same type in a group to one. Therefore, a group $\boldsymbol{G} = (v_1, \ldots, v_n), v_i \in V$ simply indicates for each sensor type the corresponding node. Accordingly, a schedule $\boldsymbol{J} = (t_1, \ldots, t_n), t_i \in \mathbb{R}$ defines the start times of each sensor type.

4.2 Space-Bounded Group Establishment

As stated above, the goal of space-bounded group establishment is to build *tight* groups, where tight is defined in terms of a dispersion metric $\psi : S \times V \rightarrow \mathbb{R}$.

We distinguish between three parts of our algorithm, which we describe in detail below. For each part, we have developed two alternatives. Since all combinations are possible, we have evaluated eight different algorithm combinations.

For each group, one node is distinguished as the leader of the group. This is always the node of which the first sensor type is used. The groups are built iteratively on the basis of the order of sensor types. At initialization, all nodes that possess the first sensor start forming groups by looking for nodes with the second sensor type. Which nodes are available for addition to a group is determined by the *selection rule*. Of course only nodes which have the correct sensor type are considered. The best node is then selected based on the *dispersion metric*. This step is performed for the following sensor types until the group is complete or as long as a suitable node is found. Each sensor of each node can only be used in one group. We now describe each step in more detail and show the implementations we used.

Dispersion metric The dispersion metric has to be computed at each step and with each candidate node. Let $\boldsymbol{G}_p$ denote a partially complete group consisting of the already selected sensor types and a candidate node for the sensor type that is currently required. Let $k = |\boldsymbol{G}_p|$ denote the number of sensor types in this partial group. Then $\text{centroid}(\boldsymbol{G}_p) = (\frac{1}{k}\sum_{v_i \in \boldsymbol{G}_p} x_i, \frac{1}{k}\sum_{v_i \in \boldsymbol{G}_p} y_i)$ defines the centroid of the partial group. We define the standard deviation $SDev = (\frac{1}{k}\sum_{v_i \in \boldsymbol{G}_p} dist(p(v_i), \text{centroid}_x(\boldsymbol{G}_p))^2)^{\frac{1}{2}}$ and the maximum of the distances to the centroids $Rad = \max_{v_i \in \boldsymbol{G}_p} dist(p(v_i), \text{centroid}_x(\boldsymbol{G}_p))$ as our two metrics.

Selection rule The selection rule defines which nodes to consider for the next sensor type. We define the *Group* selection rule where all neighbors of all group members are considered. The *Leader* selection rule allows only the neighbors of the leader of a group to be added. These rules differ obviously in communication cost, since the collection of candidates as well as the actual selection requires message transmissions. Therefore, the *Group* selection rule allows choosing from a larger number of candidates at the expense of increased communication costs.

Grouping algorithm The grouping algorithm defines the overall process of group building. We implemented a rather simple *FirstChoice* algorithm. A leader simply selects the next node based on the selection rule and the dispersion metric. The first leader that chooses the sensor on another node wins. Our second approach *BestChoice* is more complicated. A leader informs a selected node not only with its ID but also with the value of the dispersion metric, which the new node saves. If another leader later selects the same sensor on the same node and the dispersion metric of this group is better than the saved one, the node changes the group and informs the former leader about its decision. The former leader then discards all selected sensors of a lower priority and begins rebuilding its group. This algorithm is detailed in Fig. 2[1].

[1] The *BestChoice* algorithm converges if the dispersion metric has a lower bound. Proof by induction on the number of sensor nodes in the group. The transitivity of the min relation ensures that no cycles occur.

```
Procedure bestChoice
  for j = 2 to n do
  |  leader_j ← inf
  |  leaderMetric_j ← inf
  endfor
  if isLeader then
  |  search(1)
  endif
```

```
Procedure search(sensor)
  (v_cand, metric) ←
  findCand(sensor, G_i, V_ignore^sensor)
  if metric > 0 then
  |  send(v_cand, SELECT, sensor,
  |  metric)
  endif
```

```
Procedure onReceive(what,
sender, sensor)
  if what = ACCEPT then
  |  G_k ← sender
  |  if sensor < n then
  |  |  search(sensor+1)
  |  endif
  endif
  if what = REJECT then
  |  V_ignore^sensor ← V_ignore^sensor ∪ {v_sender}
  |  search(sensor)
  endif
  if what = SELECT then
  |  if leaderMetric_sensor > metric
  |  then
  |  |  if leader_sensor ≠ inf then
  |  |  |  send(leader_sensor,
  |  |  |  REJECT, sensor)
  |  |  endif
  |  |  leaderMetric_sensor ← metric
  |  |  leader_sensor ← sender
  |  |  send(sender, ACCEPT,
  |  |  sensor)
  |  else
  |  |  send(sender, REJECT,
  |  |  sensor)
  |  endif
  endif
```

```
Procedure schedule
  /* returns start times t of all
     sensor tasks                  */
  t ← 0
  minStart ← 0
  current ← 2
  for 1 = 1 to n do
  |  nodeBusy_{G_i} ← 0
  endfor
  nodeBusy_{G_1} ← d_1
  while current ≤ n do
  |  minTime ←
  |  max(nodeBusy_{G_current}, minStart)
  |  good ← TRUE
  |  for i = current-1 to n step −1
  |  do
  |  |  minTime ← max(minTime,
  |  |  t_i − C_currenti))
  |  |  if minTime > t_i − C_currenti
  |  |  then
  |  |  |  if i = 1 then
  |  |  |  // impossible
  |  |  |  |  return −1
  |  |  |  endif
  |  |  |  minStart ← minTime
  |  |  |  −C_currenti
  |  |  |  nodeBusy ← 0
  |  |  |  for j ← 1 . . . i − 1 do
  |  |  |  |  nodeBusy_{G_j} ← t_j+d_j
  |  |  |  endfor
  |  |  |  current ← i
  |  |  |  good ← FALSE
  |  |  |  break
  |  |  endif
  |  endfor
  |  if good then
  |  |  t_current ← minTime
  |  |  nodeBusy_{G_current} ←
  |  |  minTime + d_current
  |  |  current ← current+1
  |  |  minStart ← 0
  |  endif
  endw
  return t
```

Fig. 2. The BestChoice algorithm and the scheduling algorithm

4.3 Time-Bounded Scheduling

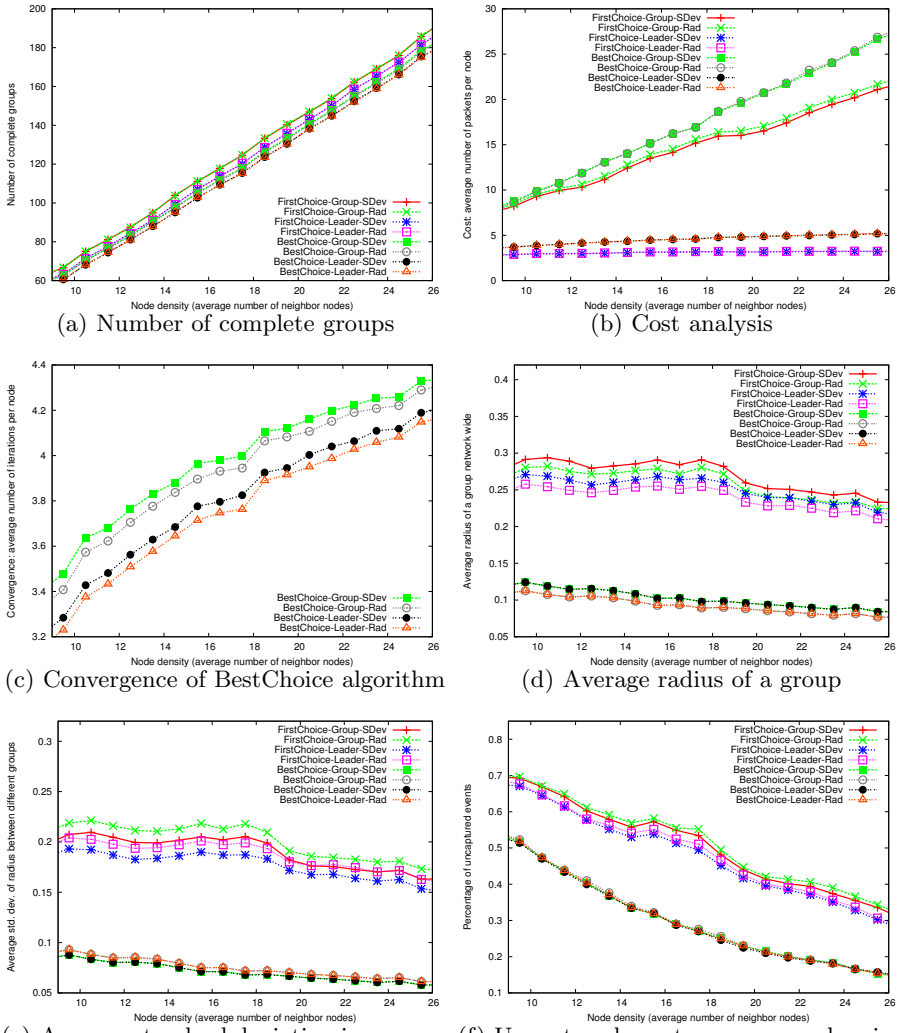
We have developed an algorithm that generates a group schedule fulfiling the concurrency constraints and minimizing the duration $\mathrm{Dur}(J)$ of the whole schedule. The total order of sensor types allows a very efficient non-recursive algorithm, which is detailed in Fig. 2. The algorithm is based on backtracking. However, due to the priorities, the start time of a task may only monotonically increase, which limits the number of backtrackings. The algorithm always finds the schedule with the minimal duration. As a slight simplification, we do not include the time that is necessary to notify all members of the group. However, this interval can simply be inserted at the beginning of the local schedules. The problem of inter-group conflicts when an event can be sensed by several groups and the sensors of one node are used in different groups is left for future work. However, priorities provide a deterministic method to select which sensor on a node is used first in case of a conflict.

5 Evaluation

The evaluation of our approach is based on thorough simulations with different node densities, different numbers of sensor types required to detect an event or process the query and two settings of concurrency constraints. The sensor nodes are distributed in an area of 900×530 m^2. In all figures we distributed between 150 and 450 sensor nodes in the area which results in average node degrees between 9 and 26. We constructed topologies using perturbed grids to create the scenarios: the nodes are placed randomly inside of circles arranged in a regular grid (the radius is equal to the grid spacing). This type of topologies is a good match for real-world deployments where the goal usually is a more or less regular coverage of the sensing area. We assume that two nodes can communicate with each other, if they are within a distance of at most $R_c = 100$m (communication radius). Therefore, the communication graph is a *unit disk graph*. Every sensor node possesses several different sensor types. Every type of sensor is uniformly distributed over the sensing area so that a certain percentage of nodes possess this sensor. Typically, the communication radius is larger than the sensing radius R_s [12], therefore, we chose $R_s = \frac{1}{2}R_c$ as the sensing radius for all sensor types.

5.1 Space-bounded Quality of Sensing

Fig. 3 shows the performance of spatial node grouping algorithms under the setting, that every sensor node can be equipped with more than one sensor. In this scenario, 50% of the sensor nodes possess the same sensor type. In Fig. 3a) we present the number of complete groups constructed by different algorithms. The number of complete groups increases with higher node densities and a greater number of deployed sensor nodes. Here, the *FirstChoice* algorithm dominates by constructing a slightly higher number of groups. As expected, both grouping algorithms combined with the *Group* selection rule require a higher number of



(a) Number of complete groups (b) Cost analysis

(c) Convergence of BestChoice algorithm (d) Average radius of a group

(e) Average standard deviation in a group (f) Uncaptured events or uncovered region

Fig. 3. Evaluation of the spatial grouping algorithms: 4 sensor types (50% of nodes possess the same sensor type, each node might possess more than 1 sensor types)

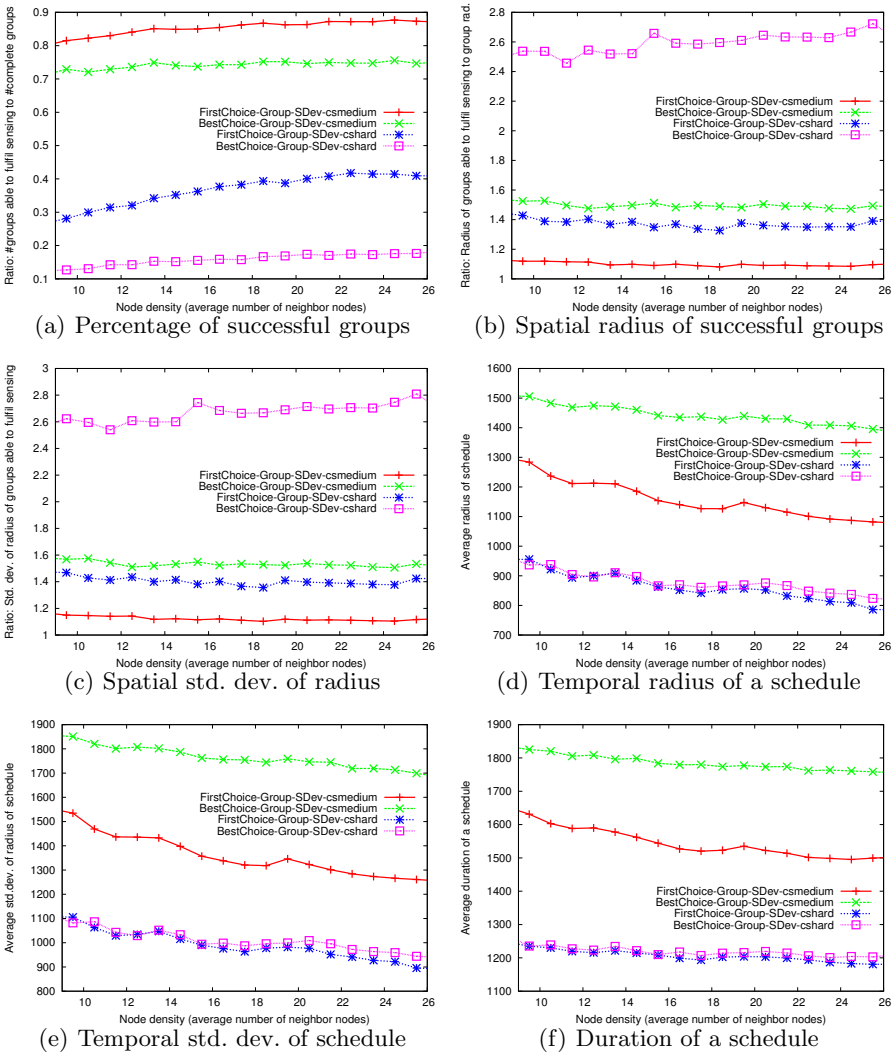
sent packets per node (Fig. 3b)). Interestingly, the algorithms combined with the *Leader* metric produce almost constant message overhead irrespective to the node density in the network, which makes this selection rule more applicable to dense scenarios than *Group*. Moreover, although the *BestChoice* algorithm requires several steps to converge, in combination with *Single* it does not produce a large increase in communication cost (Fig. 3c)). Fig. 3d,e) present the average radius and its standard deviation of constructed groups relative to the communication range of the sensor nodes R_c. In all combinations, the *BestChoice* algorithm generates the best solution. Finally, we evaluate the coverage of the monitored area by the constructed groups and plot in Fig. 3f) the percentage of events that remain uncaptured. For this test, we generated 1000 events at random points within the monitoring area. We consider that an event is captured if there is at least one spatial group nearby such that the event is in the sensing range of all sensors of this group. Obviously, the complement to the percentage of uncaptured events represents the percentage of the covered area by the constructed groups. A small group radius and a high number of successfully constructed spatial groups result in only 10% of uncaptured events and, therefore, to 90% of area coverage as presented in Fig. 3f). Again, the *BestChoice* algorithm outperforms the *FirstChoice* algorithm, as it tends to construct tighter groups and, therefore, increases the common area covered by all sensors of the group.

5.2 Time-bounded Quality of Sensing

In this section we evaluate the presented scheduling algorithm. There are five types of sensors involved in the evaluation. The first three types of sensors are distributed uniformly among the sensor nodes in the network so that 50% of the sensor nodes possess the same type of sensor. The last two types of sensors are distributed randomly over the network so that 60% of the nodes have the same sensor. These sensors might be more common ones like temperature or humidity. The first four sensors have a sensing range of $R_s = \frac{1}{2}R_c$ and the last sensor $R_s = R_c$.

The *BestChoice* and *FirstChoice* grouping algorithms combined with the *Group* selection rule and the *SDev* dispersion metric were used to build spatial groups. Moreover, we constructed two sets of concurrency constraints to evaluate the performance of the scheduling algorithms: “hard” and “medium” presented in Fig. 5. We assume that the sensing tasks are ordered from left to right in the presented graphs. Notice, that neither of these sets of concurrency constraints can be fulfilled when considering sequential scheduling on one sensor node equipped with all required sensors. Therefore, in both cases the schedule must consider the relocation of some sensing tasks to other sensor nodes in the spatial group.

In Fig. 4 we present the evaluation results of the constructed schedules in every complete group in the network. Fig. 4a) plots the percentage of groups able to fulfil the schedule compared to the overall number of complete groups constructed by the spatial grouping algorithms. This graph shows a big difference in the difficulty of both schedules: up to 90% of groups were able to compute a



(a) Percentage of successful groups

(b) Spatial radius of successful groups

(c) Spatial std. dev. of radius

(d) Temporal radius of a schedule

(e) Temporal std. dev. of schedule

(f) Duration of a schedule

Fig. 4. Spatial and temporal evaluation of the scheduling algorithm

(a) "Hard" (b) "Medium"

Fig. 5. Two sets of concurrency constraints used for evaluation: "hard" and "medium"

valid "medium" schedule, whereas less than 30% of groups succeeded to execute the "hard" one. However, for both schedules the *BestChoice* algorithm performs worse than *FirstChoice*. Moreover, in Fig. 4b,c) we evaluate changes to the average spatial radius of the groups able to compute a valid schedule compared to the average spatial radius of all constructed groups. Only the worst groups constructed by the *BestChoice* algorithms computed a valid schedule. The problem is, that the *BestChoice* algorithm constructs tighter groups than *FirstChoice* and, therefore, tends to group fewer nodes where more sensor types on each node are used. This shows that the *BestChoice* algorithm is better suited for events that require distributed space coordination whereas the *FirstChoice* algorithm is better suited for "hard" concurrency constraints and, therefore, for distributed time coordination.

In Fig. 4d-f) we evaluate the temporal characteristics of the constructed schedules: its average radius, standard deviation of the average radius in every group and average duration of a schedule. We define *radius of a schedule* as the maximum difference between the start execution times of the earliest and the latest task in the schedule. If the number of sensors in the group is equal to the number of sensor nodes in this group, the radius of a valid schedule equals 0. The *FirstChoice* algorithm outperforms the *BestChoice* algorithm also in the temporal characteristics of constructed schedules due to the reasons explained above.

6 Conclusions and Future Work

In this paper we formulate and provide a solution for the problem of time-bounded and space-bounded sensing in wireless sensor networks in order to enable complex event detection or query execution. Two cases are considered that require the distribution of sensing tasks among several nodes in vincinity: when a node is not equipped with all required sensors and when the event duration and the concurrency dependencies between sensing tasks preclude sequential sensing on one node. We analyse several spatial grouping algorithms for constructing groups of sensor nodes that can act together as one logical node equipped with all needed sensors to recognize an event. Additionally, we provide a scheduling algorithm to enable the efficient relocation of sensing tasks to group members.

Our evaluation results show that time-bounded and space-bounded sensing provides good results for complex event detection even in case when no single sensor node is able to accomplish this task on its own.

We plan to implement the described concepts for real sensor nodes as part of our future work. We also want to look into integrating these concepts into existing query and event detection systems for wireless sensor networks.

References

1. Vu, C.T., Beyah, R.A., Li, Y.: Composite event detection in wireless sensor networks. In: In Proc. of the IEEE International Performance, Computing, and Communications Conference. (2007)
2. Ould-Ahmed-Vall, E., Riley, G.F., Heck, B.S.: Distributed fault-tolerance for event detection using heterogeneous wireless sensor networks. Technical report, Georgia Institute of Technology (2006)
3. Römer, K., Mattern, F.: Event-based systems for detecting real-world states with sensor networks: A critical analysis. In: DEST Workshop on Signal Processing in Sensor Networks at ISSNIP. (2004) 389–395
4. Mansouri-Samani, M., Sloman, M.: GEM: a generalized event monitoring language for distributed systems. Distributed Systems Engineering **4**(2) (1997) 96–108
5. Janakiram, D., A. V. U. Phani Kumar, Adi Mallikarjuna Reddy V: Component oriented middleware for distributed collaboration event detection in wireless sensor networks. In: In Proc. of the 3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC'05). (2005)
6. Krishnamachari, B., Iyengar, S.: Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. IEEE Trans. Comput. **53**(3) (2004) 241–250
7. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: TinyDB: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. **30**(1) (2005) 122–173
8. Fung, W.F., Sun, D., Gehrke, J.: Cougar: the network is the database. In: Proc. of the 2002 ACM SIGMOD international conference on Management of data (SIGMOD'02), New York, NY, USA, ACM (2002) 621–621
9. Yoon, S., Shahabi, C.: The clustered aggregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks. ACM Trans. on Sensor Networks **3**(1) (2007) 3
10. Handy, M., Haase, M., Timmermann, D.: Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In: 4th International Workshop on Mobile and Wireless Communications Networks. (2002) 368–372
11. Brucker, P.: Scheduling Algorithms. Springer, 5th Edition (2007)
12. Funke, S., Klein, C.: Hole detection or: "How much geometry hides in connectivity?". In: Proc. of the 22nd Symp. on Computational Geometry. (2006)