Energy-Efficient Processing of Spatio-Temporal Queries in Wireless Sensor Networks

Markus Bestehorn Klemens Böhm Erik Buchmann Stephan Kessler
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany

ABSTRACT

Research on Moving Object Databases (MOD) has resulted in sophisticated query mechanisms for moving objects and regions. Wireless Sensor Networks (WSN) support a wide range of applications that track or monitor moving objects. However, applying the concepts of MOD to WSN is difficult: While MOD tend to require precise object positions, the information acquired in WSN may be incomplete or inaccurate. This may be because of limited detection ranges, node failures or detection mechanisms that only determine if an object is in the vicinity of a node, but not its exact position. In this paper, we study the processing of spatio-temporal queries in WSN. First, we adapt the models used in MOD to WSN while keeping their semantical depth. Second, we propose two approaches for processing such queries in WSN in-network instead of collecting all data at the base station. Our experimental evaluations using simulation as well as a Sun SPOT deployment show that our measures reduce communication by up to 89%, compared to collecting all information at the base station.

Categories and Subject Descriptors

H.2.4 [**Information Systems**]: Systems—*Distributed Databases*; H.2.8 [**Information Systems**]: Database Applications—*Spatial databases and GIS*

Keywords

Sensor Networks, Spatio-Temporal Queries

1. INTRODUCTION

Wireless Sensor Networks (WSN) have a broad range of applications. Many of them track moving objects and have spatio-temporal semantics. For example, environmental protection agencies track animals or hazardous materials in nature-protection areas [1, 25]. Authorities observe if unauthorized persons enter sensitive regions [21, 2].

Research on WSN has demonstrated that the declarativeness of queries (e.g., [37, 24]) is advantageous, but has focused on relational queries so far. However, [17, 35] have shown that expressing spatio-temporal semantics using relational query languages results in complex queries that are "hopelessly inefficient to process". This is because modeling the movement of objects and regions requires data types and operators not offered by purely relational database systems. To solve this problem, researchers on *Moving Object Databases (MOD)* have proposed languages to query moving objects and regions. To our knowledge, the question how to process such queries efficiently in WSN is untouched so far.

Well-known limitations of WSN render the problem difficult: MOD tend to assume precise and complete information on objects and regions queried. A region is modelled as a set of points that meet a user-defined criterion, e.g., all positions inside a forest. Applying this model to WSN would require precise knowledge which points fulfill such a criterion. This frequently is impractical. Instead, for many WSN applications, one can say for each node if a certain user-defined criterion is met, e.g., if the node has been deployed inside or outside the forest. Thus, spatio-temporal queries in such settings typically target the topological relationship of objects detected and a set of nodes. The semantics of such spatio-temporal queries have not been defined yet. Additionally, object detection mechanisms in WSN have limited accuracy. Thus, even if precise information on the location of a region was available, nodes might be unable to determine if objects are inside, on the border or outside of a region. Finally, sensor nodes have limited energy resources. Communication in particular dominates energy consumption. Thus, it is important to minimize communication when processing spatio-temporal queries in WSN.

In this paper, we show how to compute meaningful results for spatio-temporal queries in WSN. We first provide semantics for such queries related to object movement in relation to a set of nodes. These semantics also take the limited accuracy of object detection into account, while keeping the semantical depth of MOD. We then show how these semantics are integrated into the theoretical foundations for MOD, the 9-intersection model [10] in particular. This is important because it allows the re-use of existing concepts, e.g., spatio-temporal query languages [11].

To compute detection scenarios and query results, nodes must collect information on objects. A simple way to do so is sending all information about objects detected to the base station. This is prohibitive regarding energy consumption. To avoid this, we propose two strategies which compute results in-network. They differ in the way they collect information from nodes close to each other. By combining the spatial correlation of object detections and spatio-temporal semantics, both strategies reduce the number of messages

ACM GIS '10, November 2-5, 2010. San Jose, CA, USA

exchanged significantly. It depends on the query in question which strategy requires less communication. Our approach also addresses node failures. We provide mechanisms for the detection of such failures.

Our study includes an evaluation using simulations as well as a Sun SPOT deployment. It shows that our in-network strategies reduce communication by 45% to 89%, compared to collecting all information at the base station.

Summing up, we are the first to consider the following questions:

Q1 How can point-set based semantics of MOD be translated to node sets in WSN?

Q2 How can WSN efficiently derive results for spatio-temporal queries using the semantics just mentioned?

We answer **Q1** by adapting the point-set model for WSN in Section 4. Section 5 answers **Q2** with execution strategies for spatio-temporal predicates in WSN.

2. APPLICATION EXAMPLE

This section introduces our running example. Several biology projects track the movement of individuals at large spatial and temporal scale [19]. An example of a species studied in this way are caribous [26, 28]. The following query is an example of a spatio-temporal query scientists studying caribous could issue: "Which caribous have moved into the tree-covered swamp area on the south-western side of the river?" See Figure 1. The swamp area on the south-western side of the river that is covered by trees is a set of points. For most WSN deployments, recording the exact location of trees, swamp and river is impractical and unnecessarily complex. Instead, most WSN use a *controlled deployment* [16]: Before the nodes [36, 31] start sensing, node positions and properties of their surroundings are recorded. Examples of such properties are if a node has been deployed inside the forest or in a treeless area, close to food resources, in the swamp or in a calving area. This information allows users to derive a set of nodes that are, say, in a tree-covered swamp area on the south-western side of the river (black-colored circles in Figure 1). It is sufficiently accurate for the purpose of such an installation if the WSN observes caribou movement in relation to this set of nodes. Our paper studies this case, i.e., users are interested in object movements in relation to a set of nodes. We refer to this set of nodes as *zone* to distinguish it from the term 'region', which is a set of points. Thus, users of WSN typically express their interest described above as follows: "Which caribous c have entered the zone Z?" We define the exact meaning of such a query in Section 4.

This current work relies on several assumptions: Nodes are stationary, and they can distinguish between query-relevant objects and irrelevant ones. There exist several approaches to detect animals, e.g., acoustic recognition [22] or radio receivers which detect caribous wearing radio collars [26]. Typically these mechanisms only determine if an animal is in the vicinity of a node, but not its exact position. While our approach can be applied to more accurate mechanisms, we do not require such mechanisms. Further, nodes are able to identify objects. For instance, if Sensor `Si` detects a certain object, and `Sj` detects the object later on, the WSN knows that it is the same object. Such an identification is usually available, e.g., through identification numbers on the radio collars or noise patterns that are characteristic. Finally, we limit this paper to queries regarding the relationship between moving objects and one zone.

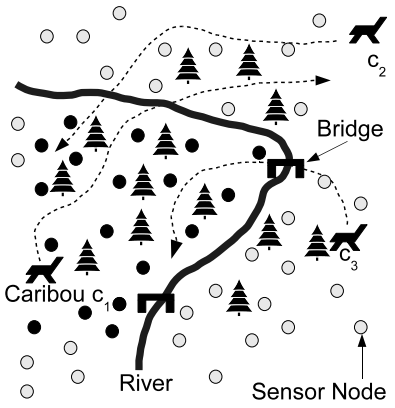


Figure 1: Illustration of the application scenario

3. PRELIMINARIES

3.1 Foundations of MOD

We only review the foundations of MOD as far as relevant for this paper; see [17, 12] for further details. In MOD there exist three spatial entities: *objects*, *lines* and *regions*. An entity is a set of points of the d-dimensional Euclidean space $\mathbf{E^d}$ [15]. To ease presentation, we focus on $d = 2$, and leave aside lines in the following.

An entity divides the space into three pair-wise disjoint partitions: The *interior*, the *border* and the *exterior*. For a region $\mathbf{R}$, the border $\mathbf{R}^B$ contains all points of the line encompassing the interior $\mathbf{R}^I$. All points that are neither in $\mathbf{R}^B$ nor in $\mathbf{R}^I$ are part of the exterior $\mathbf{R}^O$. Typically, an object $\mathbf{x}$ is represented by its position $p \in \mathbf{E^2}$ and partitions the space as follows: The interior $\mathbf{x}^I$ contains only p, the border $\mathbf{x}^B$ is empty, and all points except p are the exterior $\mathbf{x}^O$. See [10] for formal definitions.

The 9-intersection model [10] describes the topological relationship of two entities A and B: As illustrated in Figure 2, there are nine possible intersections of the exterior, the border and the interior of A with the exterior, the border and the interior of B, respectively. Each of these intersections is either empty or not. Hence, a matrix of nine boolean values identifies the relationship of A and B.

There exist three predicates to describe the relationship of an object $\mathbf{x}$ and a region $\mathbf{R}$: $Inside\,(\mathbf{x}, \mathbf{R})$, $Meet\,(\mathbf{x}, \mathbf{R})$ and $Disjoint\,(\mathbf{x}, \mathbf{R})$. Figure 3 illustrates each predicate.

Example 1: The rightmost matrix in Figure 4 describes $Inside\,(p_3, \mathbf{R})$. Since the border ${p_3}^B$ is empty, it does not intersect with any partition of $\mathbf{R}$, as reflected by the first row of the matrix. ${p_3}^I$ contains one point, and the second row implies that $p_3^I \cap \mathbf{R}^I \neq \varnothing$, i.e., p_3 is inside $\mathbf{R}$. The last row shows that ${p_3}^O$ intersects with all partitions of $\mathbf{R}$. The matrices for $Meet\,(p_2, \mathbf{R})$ and $Disjoint\,(p_1, \mathbf{R})$ only differ from the matrix for $Inside\,(p_3, \mathbf{R})$ in the second row: The topological relation of p_2 and $\mathbf{R}$ conforms to $Meet\,(p_2, \mathbf{R})$ if $p_2^I \cap \mathbf{R}^B \neq \varnothing$. Similarly, $p_1^I \cap \mathbf{R}^O \neq \varnothing$ implies that p_1 is outside of $\mathbf{x}$, i.e., $Disjoint\,(p_1, \mathbf{x})$. ■

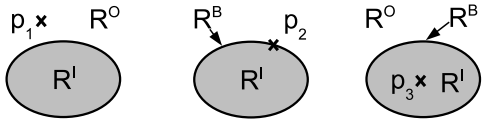
Objects can move, and the topological relation of an object and a region can change over time. To deal with such changes, [12] defines the concatenation operator:

Definition 1 (Concatenation): The *concatenation of two predicates P and Q*, referred to as $P \triangleright Q$, is true if P is true for some time interval $[t_0; t_1[$, and Q is true at t_1. □

Concatenation allows users to formulate *sequences of predicates* $P_1 \triangleright P_2 \triangleright \ldots \triangleright P_p$. These sequences, also called *spatio-*

$$\begin{pmatrix} A^B \cap B^B \neq \varnothing & A^B \cap B^I \neq \varnothing & A^B \cap B^O \neq \varnothing \\ A^I \cap B^B \neq \varnothing & A^I \cap B^I \neq \varnothing & A^I \cap B^O \neq \varnothing \\ A^O \cap B^B \neq \varnothing & A^O \cap B^I \neq \varnothing & A^O \cap B^O \neq \varnothing \end{pmatrix}$$

Figure 2: 9-Intersection Model for two spatial entities A and B



$Disjoint\,(p_1, \mathbf{R})$ $Meet\,(p_2, \mathbf{R})$ $Inside\,(p_3, \mathbf{R})$

Figure 3: Spatial Predicates for Point/Region relations

$$\begin{pmatrix} F & F & F \\ F & F & T \\ T & T & T \end{pmatrix} \quad \begin{pmatrix} F & F & F \\ T & F & F \\ T & T & T \end{pmatrix} \quad \begin{pmatrix} F & F & F \\ F & T & F \\ T & T & T \end{pmatrix}$$

$Disjoint\,(p_1, \mathbf{R})$ $Meet\,(p_2, \mathbf{R})$ $Inside\,(p_3, \mathbf{R})$

Figure 4: 9-Intersection representation of spatial predicates ($A = p_i$ and $B = \mathbf{R}$)

temporal developments, express the interest in the spatio-temporal relationship of an object and a region over time.

Example 2: A user interested in all objects $\mathbf{x}$ that move into a region $\mathbf{R}$ can express this using the following spatio-temporal development:

$$Disjoint\,(\mathbf{x}, \mathbf{R}) \triangleright Meet\,(\mathbf{x}, \mathbf{R}) \triangleright Inside\,(\mathbf{x}, \mathbf{R}) \quad (1)$$

This predicate sequence is paraphrased as $Enter\,(\mathbf{x}, \mathbf{R})$. Similar sequences like $Touch\,(\mathbf{x}, \mathbf{R})$ are possible as well:

$$Disjoint\,(\mathbf{x}, \mathbf{R}) \triangleright Meet\,(\mathbf{x}, \mathbf{R}) \triangleright Disjoint\,(\mathbf{x}, \mathbf{R}) \quad (2)$$

3.2 Related Work

[6, 8] have shown that accessing WSN declaratively is desirable for several reasons: Users formulate queries instead of writing code on sensor-node level. Such query engines can be re-used for different applications, which reduces development costs. Furthermore, these systems prolong the lifetime of battery-powered nodes by reducing communication [37, 24]. So far, relational queries have been the focus of research on query processing in WSN [7, 9, 29, 23]. Relational approaches are sufficient for queries like "What is the average temperature measured by all nodes?". But even simple spatio-temporal queries become very complex [35] and inefficient to process [17]. This is because expressing spatio-temporal semantics with relational operators typically results in a large number of joins. Research on join processing [38, 20] in WSN has shown that, except for special cases, joins should be processed at the base station. Examples of such cases are join queries with high selectivity or queries where source nodes of tuples that join are close to each other. For example, [38] concludes that the distance between source nodes must be less than a few hops. This is because determining which nodes store tuples that meet the join condition possibly requires communication among many nodes. For spatio-temporal queries one join condition would be a selection that distinguishes between objects detected that are relevant and those that are not. First, the number of nodes detecting an arbitrary number of query-relevant objects over time can be arbitrarily high, i.e., selectivity is low. Second, the distance between these nodes may be large. Hence, existing join-processing approaches for WSN are not applicable, or they send all tuples to the base station to compute the join. Therefore, our evaluation compares our approach to collecting all information on object movements at the base station. Additionally, we show in Section 6 that taking spatio-temporal semantics into account reduces communication significantly. This occurs even if nodes detecting an object are close to each other.

A straightforward application of the existing results on MOD [12, 14, 17, 18], instead of using a conventional relational query processor for WSN, is not possible as well: First, the semantics of spatio-temporal queries in MOD are defined for regions, i.e., point sets, but not for zones, i.e., node sets. Second, sensor nodes typically cannot determine an object position precisely. [34, 33] have shown how to process spatio-temporal queries in MOD if only some points of object trajectories are known. While this helps if object positions are determined periodically, it still does require precise object positions from time to time. Our approach tries to derive meaningful results based on inaccurate object detections by sensor nodes.

In [4], we have shown how to acquire meaningful results for spatio-temporal queries in WSN referring to regions. As mentioned in Section 2, acquiring precise information on the location of a region is sometimes impractical. Furthermore, [4] does not address the processing of queries at all, in contrast to the approach presented here.

4. PREDICATE SEMANTICS IN WSN

4.1 Network Model and Notations

Notation (WSN): A *wireless sensor network* is a set $\mathsf{N} = \{\mathtt{S1}, \ldots, \mathtt{Sn}\}$ of sensor nodes. p_i is the position of $\mathtt{Si}$.

Processing spatio-temporal predicates requires detection of objects moving in the area where the WSN has been deployed. To detect an object, it must be "in range", i.e., in the area observed by the detection function.

Definition 2 (Detection Area): The *detection area of node* $\mathtt{Si}$ is the set of points $\mathbf{D}_i \subseteq \mathbf{E^2}$ where $\mathtt{Si}$ can detect an object. □

Definition 3 (Detection Function): The *detection function* $detect\,(\mathtt{Si}, \mathbf{x}, t)$ is defined as follows:

$$detect\,(\mathtt{Si}, \mathbf{x}, t) = \begin{cases} T & \text{iff } \mathbf{x} \in \mathbf{D}_i \text{ at } t \\ F & \text{otherwise} \end{cases} \quad (3)$$

We say that *object* $\mathbf{x}$ *is detected at time* t if $detect\,(\mathtt{Si}, \mathbf{x}, t)$ returns T for at least one $i \in \{1, \ldots, n\}$. The detection area can have any shape or size, may overlap with detection areas of other nodes and may change over time. This is illustrated by Figure 5 where a rock prevents a sensor node from detecting objects behind it. Overlap of detection areas results in *simultaneous detection*.

Notation (Detection Set): The *detection set* $\mathsf{D}_t^{\mathbf{x}} \subseteq \mathsf{N}$ is the set of nodes that detect $\mathbf{x}$ at some time t.

$$\mathsf{D}_t^{\mathbf{x}} = \{\mathtt{Si} \in \mathsf{N} \mid detect\,(\mathtt{Si}, \mathbf{x}, t)\} \quad (4)$$

Sensor nodes typically cannot determine their detection area. However, the maximum detection range of the detection mechanism is typically available prior to deployment.

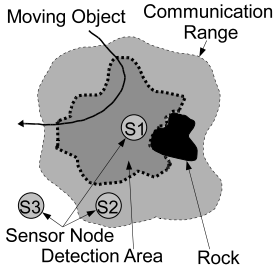


Figure 5: Illustration of the node model

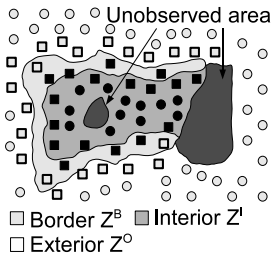


Figure 6: Point set model for zones

Detection Scenario	Predicate	Partition
$\mathtt{DS}^{\mathtt{N}}$	-	Z^N
$\mathtt{DS}^{\mathtt{O}}$	$Disjoint\,(\mathbf{x},\mathsf{Z})$	Z^O
$\mathtt{DS}^{\mathtt{I}}$	$Inside\,(\mathbf{x},\mathsf{Z})$	Z^I
$\mathtt{DS}^{\mathtt{B}}$	$Meet\,(\mathbf{x},\mathsf{Z})$	Z^B

Table 1: Mapping detection of an object x to predicates/space partitions

Definition 4 (Maximum Detection Range): The *maximum detection range* $\mathcal{D}_{max}$ is the maximum distance of an object to a node to be detected. □

Definition 5 (Communication Area): The *communication area of node* Si is the set of points $\mathbf{C}_i \subseteq \mathbf{E^2}$ where a node Sj with $i \neq j$ can receive messages sent by Si. □

Communication areas may change over time and can have any shape or size. We say that *a node* Si *can directly communicate with another node* Sj if $p_j \in \mathbf{C}_i$.

Definition 6 (Communication Neighbors): The *communication neighbors* $\mathsf{CN_i}$ *of a node* Si are the nodes that Si can directly communicate with. □

4.2 Spatio-Temporal Predicates for Zones

In line with our application example in Section 2, users are interested in the spatio-temporal relationship between objects and a *zone*, i.e., a set of nodes.

Definition 7 (Zone): A *zone* Z is a non-empty set of nodes. We say *a node* Si *is inside of* Z if $\mathtt{Si} \in \mathsf{Z}$. Si *is outside of* Z otherwise. □

To process spatio-temporal predicates in WSN, one must consider which nodes inside and outside of the zone detect an object. I.e., we are interested in the intersections of $\mathsf{D}_\mathsf{t}^\mathbf{x}$ with Z and $\overline{\mathsf{Z}}$. We refer to the different cases as *detection scenarios*, and there are four of them:

$\mathtt{DS}^{\mathtt{O}}$**:** Only nodes outside of Z currently detect **x**.
$\mathtt{DS}^{\mathtt{I}}$**:** Only nodes inside of Z detect **x**.
$\mathtt{DS}^{\mathtt{B}}$**:** $\mathsf{D}_\mathsf{t}^\mathbf{x}$ contains nodes from Z as well as $\overline{\mathsf{Z}}$.
$\mathtt{DS}^{\mathtt{N}}$**:** $\mathsf{D}_\mathsf{t}^\mathbf{x}$ neither intersects with Z nor with $\overline{\mathsf{Z}}$, i.e., **x** is currently undetected.

See [5] for formal definitions of these detection scenarios and proofs of all lemmas.

LEMMA 1. *For any point of time, exactly one detection scenario holds.*

Detection scenarios abstract from the details of object detection, i.e., they are applicable to any detection hardware or mechanism. Based on detection scenarios, we define the semantics of spatio-temporal predicates as follows: When an object **x** is undetected, the WSN cannot make any statement on the topological relation of **x** and Z. Thus, when $\mathtt{DS}^{\mathtt{N}}$ occurs, no predicate is true. $\mathtt{DS}^{\mathtt{O}}$ occurs if **x** is exclusively detected by nodes outside of the zone. Likewise, $\mathtt{DS}^{\mathtt{I}}$ occurs if **x** is exclusively detected by nodes in the zone.

Definition 8 (*Inside* (x,Z)): Object **x** conforms to the predicate $Inside\,(\mathbf{x},\mathsf{Z})$ if $\mathtt{DS}^{\mathtt{I}}$ occurs. □

Definition 9 (*Disjoint* (x,Z)): Object **x** conforms to the predicate $Disjoint\,(\mathbf{x},\mathsf{Z})$ if $\mathtt{DS}^{\mathtt{O}}$ occurs. □

If nodes from Z and $\overline{\mathsf{Z}}$ detect an object, i.e., $\mathtt{DS}^{\mathtt{B}}$ occurs, the WSN can derive that the object is between being inside and outside of the zone. Hence, we define:

Definition 10 (*Meet* (x,Z)): The object **x** conforms to $Meet\,(\mathbf{x},\mathsf{Z})$ if $\mathtt{DS}^{\mathtt{B}}$ occurs. □

Table 1 serves as a summary. ▷ and other concepts from MOD, e.g., lifting [17], are applicable to these predicates as well. Thus, one can construct developments that query the spatio-temporal relationship of objects and a zone. For instance, one could define:

$$Enter\,(\mathbf{x},\mathsf{Z}) = Disjoint\,(\mathbf{x},\mathsf{Z}) \triangleright Meet\,(\mathbf{x},\mathsf{Z}) \triangleright Inside\,(\mathbf{x},\mathsf{Z}) \quad (5)$$

4.3 Point-Set Topology for Zones

According to point-set topology, a region partitions the (Euclidean) space into three subsets of points. By definition, this partitioning is *regular* [32]: This means that a region **R** does not contain holes, is continuous, and the border $\mathbf{R}^B$ encompasses the interior $\mathbf{R}^I$ completely. The space partitioning of zones in turn is not regular, as we will explain. We show how this affects the semantics of spatio-temporal queries in WSN and how to deal with this non-regularity.

First we derive a space partitioning based on the predicate definitions and detection scenarios above using the following idea: Without loss of generality, let **x** be detected according to $\mathtt{DS}^{\mathtt{O}}$, i.e., $Disjoint\,(\mathbf{x},\mathsf{Z})$. From this, we infer that the position of **x** is some $p \in \mathbf{E^2}$ exclusively observed by nodes outside of Z, i.e., *the exterior of the zone.* For each predicate we can derive such a subset of the space.

Definition 11 (Unobserved Area): All points not contained in a detection area form *the unobserved area* Z^N:

$$\mathsf{Z}^N = \left\{p \in \mathbf{E^2} \mid \nexists \mathtt{Si} \in \mathsf{N} : p \in \mathbf{D}_i\right\} \quad (6)$$

Definition 12 (Exterior): All points exclusively observed by nodes in $\overline{\mathsf{Z}}$ constitute *the exterior of a zone* Z^O:

$$\mathsf{Z}^O = \left\{p \in \mathbf{E^2} \mid p \notin \mathsf{Z}^N \wedge \nexists \mathtt{Si} \in \mathsf{Z} : p \in \mathbf{D}_i\right\} \quad (7)$$

Definition 13 (Interior): All points of space exclusively observed by nodes in Z constitute *the interior of a zone* Z^I:

$$\mathsf{Z}^I = \left\{p \in \mathbf{E^2} \mid p \notin \mathsf{Z}^N \wedge \nexists \mathtt{Si} \in \overline{\mathsf{Z}} : p \in \mathbf{D}_i\right\} \quad (8)$$

Definition 14 (Border): All points of space observed by nodes in Z and $\overline{\mathsf{Z}}$ form the *border of a zone* Z^B:

$$\mathsf{Z}^B = \left\{p \in \mathbf{E^2} \mid \exists \mathtt{Si} \in \mathsf{Z}, \exists \mathtt{Sj} \in \overline{\mathsf{Z}} : p \in \mathbf{D}_i, p \in \mathbf{D}_j\right\} \quad (9)$$

LEMMA 2. *The point sets* Z^N, Z^O, Z^I *and* Z^B *partition the space.*

Figure 6 illustrates the space partitioning for zones. Black-colored circles or squares[1] represent nodes in the zone. Ex-

[1]The difference between squares and circles is irrelevant here and will be explained in Section 5.3.

ternal factors influence detection areas which in turn determine the partitions. This may result in situations where the border does not encompass the zone or "holes" in the zone. Hence, space partitioning based on zones is not regular.

Non-regularity has an impact on spatio-temporal developments, as we illustrate with *Enter*: MOD can assume that point sets are regular, i.e., an object $\mathbf{x}$ has to cross the border $\mathbf{R}^B$. This is not true for WSN: $\mathbf{x}$ may be detected according to $\mathtt{DS}^0$ first, move through an unobserved area and then appear inside of Z. Users interested in objects moving in this way cannot express this as follows:

$$\mathbb{P}(\mathbf{x},\mathsf{Z}) = \mathit{Disjoint}(\mathbf{x},\mathsf{Z}) \triangleright \mathit{Inside}(\mathbf{x},\mathsf{Z}) \quad (10)$$

This sequence never occurs, because $\triangleright$ requires $\mathit{Inside}(\mathbf{x},\mathsf{Z})$ to follow $\mathit{Disjoint}(\mathbf{x},\mathsf{Z})$ immediately. On the other hand, $\mathit{Enter}(\mathbf{x},\mathsf{Z})$ excludes all objects that are unobserved while moving into the zone. In other words, users cannot express such a query given the three predicates and $\triangleright$.

Definition 15 (Relaxed Concatenation): The *relaxed concatenation of two predicates*, $P \mathbin{\widetilde{\triangleright}} Q$, is true if P is true for some interval $[t_0, t_1[$, and Q is true at $t_2 \geq t_1$. □

(11) expresses the query discussed above, and Example 3 shows that this new primitive increases the semantical depth:

$$\mathit{WSNEnter}(\mathbf{x},\mathsf{Z}) = \mathit{Disjoint}(\mathbf{x},\mathsf{Z}) \mathbin{\widetilde{\triangleright}} \mathit{Inside}(\mathbf{x},\mathsf{Z}) \quad (11)$$

Example 3: The area where the WSN in Figure 1 is deployed contains a river with several bridges. Suppose that nodes are deployed so that caribous moving over a bridge are detected, but caribous swimming are not, i.e., the river itself is unobserved. A user only interested in caribous c entering Z by crossing bridges can use $\mathit{Enter}(c,\mathsf{Z})$. A user interested in all caribous can express this as $\mathit{WSNEnter}(\mathbf{x},\mathsf{Z})$. ■

LEMMA 3. *$\widetilde{\triangleright}$ is associative and can be combined with $\triangleright$:*

$$P_1 \triangleright P_2 \Rightarrow P_1 \mathbin{\widetilde{\triangleright}} P_2$$
$$P_1 \mathbin{\widetilde{\triangleright}} (P_2 \mathbin{\widetilde{\triangleright}} P_3) = (P_1 \mathbin{\widetilde{\triangleright}} P_2) \mathbin{\widetilde{\triangleright}} P_3$$
$$P_1 \triangleright (P_2 \mathbin{\widetilde{\triangleright}} P_3) = (P_1 \triangleright P_2) \mathbin{\widetilde{\triangleright}} P_3$$

Definition 16 (Spatio-Temporal Development): A *spatio-temporal development* $\mathbb{P}(\mathbf{x},\mathsf{Z})$ is a sequence of predicates $P_1(\mathbf{x},\mathsf{Z})\,\theta \ldots \theta P_q(\mathbf{x},\mathsf{Z})$ with $\theta \in \{\triangleright, \mathbin{\widetilde{\triangleright}}\}$, and $P_i(\mathbf{x},\mathsf{Z}) \in \{\mathit{Inside}(\mathbf{x},\mathsf{Z}), \mathit{Meet}(\mathbf{x},\mathsf{Z}), \mathit{Disjoint}(\mathbf{x},\mathsf{Z})\}$. □

This concludes our study regarding **Q1**. We have defined the semantics of spatio-temporal predicates for SN that express the topological relation of an object and a zone. Furthermore, we have shown how to take unobserved areas into account when expressing spatio-temporal queries for SN and provided a space partitioning for zones.

5. EFFICIENT PROCESSING OF SPATIO-TEMPORAL QUERIES

In this section, we provide an answer to **Q2**, i.e., process spatio-temporal queries efficiently: We propose two strategies that reduce the number of messages exchanged between nodes, compared to a straightforward approach that collects all information at the base station. We present our approach in the following steps: Section 5.1 describes data structures needed and says how to compute detection scenarios from them. The remainder of the section describes three strategies for acquiring the necessary information. All information either is collected at the base station (Section 5.2), or collection is distributed among sensor nodes to compute detection scenarios in-network (Section 5.3). Failures of nodes may impact query results. Section 5.4 shows how to detect these failures and determines if they might have an impact on the query result.

We assume that the following steps have been completed before the WSN starts to process a query:

1. Definition of a zone Z.
2. Specification of the movement of interest as a spatio-temporal development $\mathbb{P}(\mathbf{x},\mathsf{Z})$.
3. Dissemination of a list of nodes representing Z and the query $\mathbb{P}(\mathbf{x},\mathsf{Z})$ to all nodes.

The query result returned to the user by the base station includes every object whose movement conforms to $\mathbb{P}(\mathbf{x},\mathsf{Z})$. To accomplish this, the WSN must compute the detection scenario when an object is detected. Using Table 1, any node can determine which predicate the detected object conforms after the detection scenario has been computed.

An object $\mathbf{x}$ fulfills $\mathbb{P}(\mathbf{x},\mathsf{Z})$ if the WSN derives from the detection scenarios of $\mathbf{x}$ that the predicates have occurred in the correct order. The distributed strategies notify the base station whenever a predicate $P(\mathbf{x},\mathsf{Z})$ in $\mathbb{P}(\mathbf{x},\mathsf{Z})$ is satisfied. Thus, the base station determines if $\mathbf{x}$ has fulfilled $\mathbb{P}(\mathbf{x},\mathsf{Z})$. Note that a node `Si` may send several notifications regarding a predicate to the base station because it detects the same object more than once. This is intended, for two reasons: First, the query may be a predicate sequence that contains a predicate twice, e.g., $\mathit{Touch}(\mathbf{x},\mathsf{Z})$ (cf. (2)). Second, coordinating nodes such that they only send notifications regarding predicates that have not occurred on any other node requires communication. A preliminary study of ours has shown that the communication effort for such coordination only pays off if the network is very small, the zone is small, and if the object moves through detection areas of many nodes repeatedly. Thus, we do not intend to prevent this. On the other hand, we show in Section 5.3 how to exploit spatio-temporal semantics to reduce the number of notifications, e.g., only few nodes send notifications for queries like $\mathbb{P}(\mathbf{x},\mathsf{Z}) = \mathit{Enter}(\mathbf{x},\mathsf{Z})$.

5.1 Data Structures and Algorithms

A relation `Detections` contains data on objects detected. It depends on the strategy where `Detections` is stored: All tuples are sent to the base station (centralized), or they are distributed and/or replicated among the nodes in the WSN (distributed). `Detections` has the following attributes:

- `NodeID`: Identifier of the node `Si` detecting the object identified by attribute `ObjectID`.
- `ObjectID`: Identifier of the object detected by `Si`.
- t_{entry}: $t \in \mathit{time}$ when `Si` starts to detect the object.
- t_{exit}: This value is either $\top$ or a $t > t_{entry}$. If it is $\top$, `Si` is still detecting the object. Otherwise, `Si` has detected the object during the time interval $[t_{entry}; t_{exit}[$.

We say that *a tuple `T` originates from node* `Si` if `T.NodeID` = `Si`. We refer to the moment an object $\mathbf{x}$ moves into the detection area of a node `Si` as *entry event*. When such an entry occurs at time t, a tuple $[\mathtt{Si}, \mathbf{x}, t, \top]$ is added to `Detections`. Then the object may be inside the detection area for an arbitrary interval of time. We refer to the moment when $\mathbf{x}$ leaves the detection area as *exit event*. When $\mathbf{x}$ this happens at t', the existing tuple is updated, i.e., becomes $[\mathtt{Si}, \mathbf{x}, t, t']$.

To determine the detection scenario, the system must compute how the detection set $\mathsf{D}_t^{\mathbf{x}}$ intersects with Z and $\overline{\mathsf{Z}}$ at t based on `Detections`. We refer to this computation as `isDetecting(S*,x,t)`, which is defined in (12).

$$\texttt{isDetecting(S}^*\texttt{,x,t)} = \begin{cases} T & \text{if } \exists \texttt{Si} \in \mathsf{S}^* : \mathit{detect}\,(\texttt{Si}, \mathbf{x}, t) \\ F & \mathit{otherwise} \end{cases} \tag{12}$$

The input parameter S^* is either Z or $\overline{\mathsf{Z}}$. [5] provides an algorithm that implements this function. We use this function twice to compute the detection scenario: First to determine `isDetecting(Z,x,t)` and then for `isDetecting(`$\overline{\mathsf{Z}}$`,x,t)`. Based on this, one can derive the detection scenario according to Table 2. Each cell corresponds to a pair [`isDetecting(Z,x,t)` , `isDetecting(`$\overline{\mathsf{Z}}$`,x,t)`] and contains the corresponding detection scenario.

		`isDetecting(`$\overline{\mathsf{Z}}$`,x,t)`	
		T	F
`isDetecting(Z,x,t)`	T	DS^{B}	DS^{I}
	F	DS^{0}	DS^{N}

Table 2: Deriving detection scenarios from `Detections`

In the following, we deal with the collection of tuples in `Detections` to ensure a correct computation of detection scenarios.

Definition 17 (Correctness): The *computation of the detection scenario is correct* if the space partition corresponding to the detection scenario computed (cf. Table 1) contains the position $p \in \mathbf{E^2}$ of the object detected. □

Definition 18 (Completeness): `Detections` *is complete regarding an object* $\mathbf{x}$ *and a time* t if it contains all tuples $\{\texttt{Si}, \mathbf{x}, t_1, t_2\}$ with $t_1 \leq t$ and $t \leq t_2$ or $t_2 = \top$. □

LEMMA 4. *If the relation* `Detections` *is complete, the detection scenario computed according to Table 2 is correct.*

Summing up, the base station or an arbitrary node must store a complete set of tuples locally to compute a detection scenario for an object $\mathbf{x}$ and a time t. In the following we deal with acquiring these tuples.

5.2 Centralized Data Collection

A straightforward approach is that every node notifies the base station whenever an object enters or leaves a detection area. The base station then modifies `Detections` as shown and computes a detection scenario. See Algorithm 5.1.

Algorithm 5.1: Centralized Data Collection

```
1 When x enters/leaves D_i of Si at t do
2 |  Si sends corresponding notification to base station
3 end
4 When base station receives notification from Si do
5 |  Modify Detections at base station
6 |  Wait t_delay
7 |  Compute
  |  [isDetecting(S^R,x,t), isDetecting(S^R̄,x,t)]
8 end
```

Arbitrary nodes detecting an object execute the first part of Algorithm 5.1. Sending tuples from `Si` to the base station requires routing protocols [27, 13]. These protocols forward messages via multiple hops if `Si` is not a communication neighbor of the base station.

The base station executes the second part. Line 5 modifies `Detections` as described previously. The base station then has to wait a timeout t_{delay} before it computes the detection scenario according to Table 2. The timeout ensures that notifications of nodes which simultaneously detect an object have arrived before the detection scenario is computed. t_{delay} is the maximum time a notification may need to be forwarded to the base station. Its actual value depends on factors such as communication hardware, WSN size, routing protocol etc. For our reference implementation we use a delay of 30 seconds.

LEMMA 5. *If* t_{delay} *is the maximum time a notification needs to travel from a node* `Si` *to the base station,* `Detections` *stored at the base station is complete at* $t + t_{delay}$.

Thus, the computation of the detection scenario based on centralized data collection is correct.

5.3 Distributed Data Collection

In the following, we propose two strategies which distribute the relation `Detections`. The distribution is done in such a way that any node `Si` detecting an object $\mathbf{x}$ can compute the detection scenario. As we show, this reduces communication for two reasons:

- Nodes only notify the base station on objects that possibly fulfill the query.
- There are fewer nodes from which data must be collected, i.e., only some nodes communicate.

The latter point stems from the following idea: When a node `Si` detects an object $\mathbf{x}$, only nodes in its vicinity can detect the object simultaneously. The problem is that detection mechanisms in WSN typically do not allow precise localization of the object detected. But in turn, `Si` can derive that only nodes whose detection area overlaps with $\mathbf{D}_i$ could possibly detect $\mathbf{x}$ simultaneously.

Definition 19 (Detection Neighbor): Node `Sj` is a *detection neighbor of* `Si` if the detection areas of both nodes overlap. $\mathsf{DN_i}$ is the set of detection neighbors of `Si`. □

Section 5.3.1 shows how to approximate the detection neighbors if detection areas are indeterminable. Since Z is disseminated to all nodes, every `Si` can derive for each detection neighbor $\texttt{Sj} \in \mathsf{DN_i}$ if it is in Z or not.

Notation (Detection-Neighbor Subsets): We refer to the subset of detection neighbors of `Si` in the zone Z as $\mathsf{DN_i^Z}$. $\mathsf{DN_i^{\overline{Z}}}$ contains all detection neighbors of node `Si` that are outside of Z.

LEMMA 6. `Detections` *stored at* `Si` *is complete regarding* $\mathbf{x}$ *and* t *if* `Si` *detects* $\mathbf{x}$ *at* t *and obtains all tuples on* $\mathbf{x}$ *originating from its detection neighbors* $\mathsf{DN_i}$.

[5] provides proofs this and all other lemmas presented here. By taking into account that `Si` is either in Z or $\overline{\mathsf{Z}}$ we actually can compute a correct detection scenario without `Detections` being complete.

Definition 20 (Semi-Completeness): `Detections` *regarding* $\mathbf{x}$ *and* t *stored at a node* $\texttt{Si} \in \mathsf{Z}$ *is semi-complete* if it contains all tuples $[\texttt{Sj}, \mathbf{x}, t_1, t_2]$ with $t_1 \leq t \leq t_2$ where $\texttt{Sj} \in \mathsf{DN_i^{\overline{Z}}}$.

`Detections` *regarding* $\mathbf{x}$ *and* t *stored at a node* $\texttt{Si} \in \overline{\mathsf{Z}}$ *is semi-complete* if it contains all tuples $[\texttt{Sj}, \mathbf{x}, t_1, t_2]$ with $t_1 \leq t \leq t_2$ where $\texttt{Sj} \in \mathsf{DN_i^Z}$. □

LEMMA 7. *Let* `Si` *detect* $\mathbf{x}$ *at* t. *Without loss of generality, let* $\texttt{Si} \in \mathsf{Z}$. *If* `Detections` *stored at* `Si` *is semi-complete regarding* $\mathbf{x}$ *and* t, *the computation of the detection scenario at* `Si` *according to Table 1 is correct.*

Based on Lemma 7, a node can reduce the number of detection neighbors it has to obtain tuples from even more by using the following idea:

Definition 21 (Border Node): $\mathtt{Si}$ is a *border node* if

- $\mathtt{Si} \in \mathsf{Z}$ and $\mathsf{DN_i}^{\overline{\mathsf{Z}}} \neq \varnothing$, or
- $\mathtt{Si} \in \overline{\mathsf{Z}}$ and $\mathsf{DN_i}^{\mathsf{Z}} \neq \varnothing$. □

Figure 6 illustrates a WSN, a zone Z and the resulting space partitioning based on detection areas. Nodes are represented as squares or circles, and there are four kinds of nodes: Non-border nodes inside Z are represented by black-colored circles. Black-colored squares correspond to border nodes inside Z. Similarly, grey-colored squares and circles correspond to border and non-border nodes outside of Z, respectively. A significant share of the nodes in this scenario are non-border nodes. According to Lemma 8, non-border nodes can compute detection scenarios without obtaining tuples originating from any detection neighbor.

LEMMA 8. *If a non-border node* $\mathtt{Si}$ *detects* $\mathbf{x}$ *at* t *and modifies* $\mathtt{Detections}$ *accordingly,* $\mathtt{Detections}$ *stored at* $\mathtt{Si}$ *is semi-complete.*

LEMMA 9. *Let* $\mathbb{P}(\mathbf{x}, \mathsf{Z}) = P_1(\mathbf{x}, \mathsf{Z}) \triangleright P_2(\mathbf{x}, \mathsf{Z})$. *Tuples originating from non-border nodes are not necessary to process* $\mathbb{P}(\mathbf{x}, \mathsf{Z})$.

Lemma 9 refers to developments constructed using $\triangleright$ exclusively. Sensor nodes typically have a deep-sleep modus [30] which reduces their energy consumption significantly. This is important since non-border nodes can use deep-sleep while developments like $Enter(\mathbf{x}, \mathsf{Z})$ are processed.

5.3.1 Approximation of Detection Neighbors

As stated in Section 4.1, there exist detection mechanisms where the detection area is indeterminable. In this case, nodes cannot determine their detection neighbors. To solve this problem, we use a superset $\mathsf{DN_i}^{approx}$ which contains at least all detection neighbors $\mathsf{DN_i}$, i.e., $\mathsf{DN_i} \subseteq \mathsf{DN_i}^{approx}$. Using $\mathsf{DN_i}^{approx}$ instead of $\mathsf{DN_i}$ obviously still yields a correct result, because those nodes in $\mathsf{DN_i}^{approx}$ that are not detection neighbors of $\mathtt{Si}$ cannot detect an object simultaneously. Several approaches to derive such a superset are conceivable, and we outline two of them:

Communication Neighbors: If the communication range can be assumed to be much larger than the maximum detection range, a valid superset is $\mathsf{CN_i}$, i.e., $\mathsf{DN_i}^{approx} = \mathsf{CN_i}$. In this case, all detection neighbors are communication neighbors as well. This approach is applicable to most detection mechanisms used in WSN, and we use it for our evaluation.

Node Positions: Another approach is applicable if nodes know their position: The set $\mathsf{DN_i}^{approx}$ contains all nodes with a distance of at most $2 \cdot \mathcal{D}_{max}$ to $\mathtt{Si}$.

Next, we propose two strategies which allow a node $\mathtt{Si}$ that detects $\mathbf{x}$ to obtain tuples originating from detection neighbors efficiently to compute the detection scenario.

5.3.2 Reactive Strategy

The core idea of the *reactive strategy* is as follows: At query-dissemination time, each node has received $\mathbb{P}(\mathbf{x}, \mathsf{Z})$. Each predicate of the development is related to a detection scenario according to Table 1. For instance, for $WSNEnter(\mathbf{x}, \mathsf{Z})$ each node knows that only $\mathtt{DS}^{\mathtt{0}}$ and $\mathtt{DS}^{\mathtt{I}}$ are relevant. When an object $\mathbf{x}$ enters or leaves the detection area of $\mathtt{Si}$ at time t, $\mathtt{Si}$ checks if this possibly results in a predicate $P(\mathbf{x}, \mathsf{Z})$ of the query being true. If so, $\mathtt{Si}$ requests tuples on $\mathbf{x}$ from some or all of its detection neighbors. $\mathtt{Si}$ stores each such tuple and computes the detection scenario after the tuples requested have arrived. If the detection scenario computed results in one predicate of $\mathbb{P}(\mathbf{x}, \mathsf{Z})$ being true, the base station is notified. A core question is: "When $\mathtt{Si}$ detects $\mathbf{x}$, which detection neighbors could have tuples that are relevant to compute the detection scenario?" This depends on three points:

- The predicates $P(\mathbf{x}, \mathsf{Z})$ that form $\mathbb{P}(\mathbf{x}, \mathsf{Z})$.
- Whether $\mathtt{Si} \in \mathsf{Z}$ or $\mathtt{Si} \in \overline{\mathsf{Z}}$.
- Whether $\mathbf{x}$ has entered or left the detection area $\mathbf{D}_i$.

Table 3 summarizes from which detection neighbors a node has to request tuples to check if a given detection scenario has occurred when it detects $\mathbf{x}$. In the following, we explain these cells, using Figure 7 as an illustration. Figure 7 shows two nodes and their detection areas as well as two objects that enter/leave these detection areas at different times t_i.

The first row of Table 3 is related to $\mathtt{DS}^{\mathtt{I}}$, i.e., $\mathbb{P}(\mathbf{x}, \mathsf{Z})$ contains $Inside(\mathbf{x}, \mathsf{Z})$. There are two cases that can lead to $\mathtt{DS}^{\mathtt{I}}$: (1) an object enters the detection area $\mathbf{D}_i$ of a node $\mathtt{Si} \in \mathsf{Z}$ or (2) an object leaves $\mathbf{D}_j$ of $\mathtt{Sj} \in \overline{\mathsf{Z}}$. For all other detection events, no communication is required, as reflected by the '$\varnothing$' entries. Case (1) occurs at t_2 and t_5 in Figure 7 since $\mathtt{S2} \in \mathsf{Z}$. Applying Lemma 7, $\mathtt{S2}$ only requires tuples from $\mathsf{DN_i}^{\overline{\mathsf{Z}}}$ to compute the detection scenario. The corresponding $\mathsf{DN_i}^{\overline{\mathsf{Z}}}$ entry in Table 3 reflects this. For t_2, $\mathtt{S1} \in \mathsf{DN_i}^{\overline{\mathsf{Z}}}$ returns a tuple $[\mathtt{S1}, \mathbf{x_1}, t_1, \top]$. From this, $\mathtt{S2}$ can derive that $\mathtt{S1}$ and $\mathtt{S2}$ detect $\mathbf{x_1}$ simultaneously, i.e., $\mathtt{DS}^{\mathtt{I}}$ did not occur. Contrary to this, $\mathtt{S2}$ derives $\mathtt{DS}^{\mathtt{I}}$ for $\mathbf{x_2}$ for t_5. Case (2) is different, because objects leave $\mathbf{D}_j$ of $\mathtt{Sj} \in \overline{\mathsf{Z}}$, i.e., $\mathtt{Sj}$ does not detect the object any more and thus cannot apply Lemma 7. Hence, $\mathtt{Detections}$ stored at $\mathtt{Sj}$ has to be complete, i.e., $\mathtt{Sj}$ must request tuples from all detection neighbors. This is reflected by the $\mathsf{DN_i}$ entry in the first row of Table 3. This case occurs at t_3 and t_8 in Figure 7 since $\mathtt{S2} \in \overline{\mathsf{Z}}$. In both cases, $\mathtt{S1}$ must verify that no other node outside of Z still detects the object, and that there is at least one node in the zone detecting it. Hence, $\mathtt{DS}^{\mathtt{I}}$ occurs at t_3 but not at t_8.

Algorithm 5.2: Reactive Strategy

```
1 When x enters or leaves the detection area of Si do
2     Modify Detections as described in Section 5.1;
3     DN* ← Set of detection neighbors that must be
      queried according to Table 3;
4     Request tuples on x from every node in DN*;
5     Wait for response from every node in DN*;
6     Determine detection scenario d according to Table 2;
7     Compute predicate result from d based on Table 1;
8     Notify base station if x fulfills a predicate of the
      query;
9 end
```

The second row of Table 3 is related to $\mathtt{DS}^{\mathtt{B}}$, i.e., $Meet(\mathbf{x}, \mathsf{Z})$ is part of $\mathbb{P}(\mathbf{x}, \mathsf{Z})$. $\mathtt{DS}^{\mathtt{B}}$ requires simultaneous detection of $\mathbf{x}$ by nodes inside and outside of the zone. Thus, when an object leaves a detection area, $\mathtt{DS}^{\mathtt{B}}$ either already has occurred or does not occur at all, i.e., no communication is required. Contrary to that, objects entering a detection area can result in $\mathtt{DS}^{\mathtt{B}}$. This allows applying Lemma 7. Thus, if $\mathtt{Si} \in \mathsf{Z}$, only tuples from $\mathsf{DN_i}^{\overline{\mathsf{Z}}}$ are required and vice versa.

The entries for $\mathtt{DS}^{\mathtt{0}}$, i.e., $\mathbb{P}(\mathbf{x}, \mathsf{Z})$ contains $Disjoint(\mathbf{x}, \mathsf{Z})$, are derived analogously to those for $\mathtt{DS}^{\mathtt{I}}$.

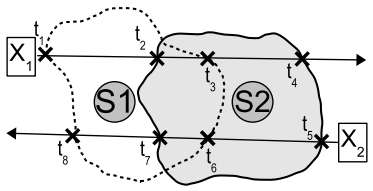


Figure 7: Detection Events ($\texttt{S1} \in \overline{\texttt{Z}}$, $\texttt{S2} \in \texttt{Z}$)

Reactive		$\texttt{Si} \in \texttt{Z}$	$\texttt{Si} \in \overline{\texttt{Z}}$
$\texttt{DS}^{\texttt{I}}$	Entry	$\texttt{DN}_\texttt{i}^{\overline{\texttt{Z}}}$	$\varnothing$
	Exit	$\varnothing$	$\texttt{DN}_\texttt{i}$
$\texttt{DS}^{\texttt{B}}$	Entry	$\texttt{DN}_\texttt{i}^{\overline{\texttt{Z}}}$	$\texttt{DN}_\texttt{i}^{\texttt{Z}}$
	Exit	$\varnothing$	$\varnothing$
$\texttt{DS}^{\texttt{0}}$	Entry	$\varnothing$	$\texttt{DN}_\texttt{i}^{\texttt{Z}}$
	Exit	$\texttt{DN}_\texttt{i}$	$\varnothing$

Table 3: Detection-neighbor partitions for the reactive strategy

Proactive		$\texttt{Si} \in \texttt{Z}$	$\texttt{Si} \in \overline{\texttt{Z}}$
$\texttt{DS}^{\texttt{I}}$	Entry	$\varnothing$	$\texttt{DN}_\texttt{i}^{\texttt{Z}}$
	Exit	$\varnothing$	$\texttt{DN}_\texttt{i}^{\texttt{Z}}$
$\texttt{DS}^{\texttt{B}}$	Entry	$\texttt{DN}_\texttt{i}^{\overline{\texttt{Z}}}$	$\texttt{DN}_\texttt{i}^{\texttt{Z}}$
	Exit	$\texttt{DN}_\texttt{i}^{\overline{\texttt{Z}}}$	$\texttt{DN}_\texttt{i}^{\texttt{Z}}$
$\texttt{DS}^{\texttt{0}}$	Entry	$\texttt{DN}_\texttt{i}^{\overline{\texttt{Z}}}$	$\varnothing$
	Exit	$\texttt{DN}_\texttt{i}^{\overline{\texttt{Z}}}$	$\varnothing$

Table 4: Detection-neighbor partitions for the proactive strategy

5.3.3 Proactive Strategy

As illustrated in Algorithm 5.2, the reactive strategy requires communication for requesting tuples and for responding to these requests. The proactive strategy tries to avoid responses. Algorithm 5.3 outlines the strategy, and the core idea is as follows: When $\mathbf{x}$ enters or leaves at t the detection area of $\texttt{Si}$, $\texttt{Detections}$ stored at $\texttt{Si}$ is modified. This modification is either an insertion of a tuple $[\texttt{Si}, \mathbf{x}, t, \top]$ or an update of a tuple $[\texttt{Si}, \mathbf{x}, t' < t, \top]$ to $[\texttt{Si}, \mathbf{x}, t', t]$ (cf. Section 5.1). Afterwards, $\texttt{Si}$ immediately sends the modified tuple to a subset $\texttt{DN}^*$ of its detection neighbors. Each detection neighbor $\texttt{Sj} \in \texttt{DN}^*$ stores the modified tuple. This ensures that $\texttt{Detections}$ stored at $\texttt{Si}$ and each $\texttt{Sj}$ is semi-complete. According to Lemma 7, $\texttt{Si}$ and any $\texttt{Sj} \in \texttt{DN}^*$ that currently detects $\mathbf{x}$ can compute the detection scenario. Again, the important step is determining the set $\texttt{DN}^*$ in Line 3 since it determines the number of messages. Analogously to the reactive strategy, Table 4 lists which detection neighbors must receive an update to ensure semi-completeness, for each detection scenario. We explain each cell in the following using Figure 7.

Algorithm 5.3: Proactive Strategy

```
1  When x enters/leaves detection area of Si do
2  |  Modify Detections as described in Section 5.1;
3  |  DN* ← Set of detection neighbors whose
   |       information must be updated according to Table 4;
4  |  Send updated tuple(s) to every node in DN*;
5  |  Goto Line 9;
6  end
7  When Si receives updated tuples about x do
8  |  Insert updated tuples into Detections;
9  |  Determine detection scenario d according to Table 2;
10 |  Compute predicate result from d based on Table 1;
11 |  Notify base station if x fulfills a predicate of the
   |       query;
12 end
```

Recall that $\texttt{DS}^{\texttt{I}}$ can either occur (1) when an object enters the detection area of a node inside the region or (2) when the detection area of a node outside of the region is left. An object detection conforming to $\texttt{DS}^{\texttt{I}}$ requires at least one node $\texttt{Si} \in \texttt{Z}$ to detect the object. If such a detection occurs, $\texttt{Si}$ must determine if there exists a simultaneous detection by another node $\texttt{Sj} \in \overline{\texttt{Z}}$. Using Figure 7 again, Case (1) occurs at t_2 and t_5. To compute the detection scenario correctly at t_2, $\texttt{S2}$ must know that $\texttt{S1} \in \overline{\texttt{Z}}$ currently detects $\mathbf{x_1}$. Case (2) occurs when $\mathbf{x_1}$ leaves the detection area of $\texttt{S1}$ at t_3. In this case, the information at $\texttt{S2}$ is updated, and $\texttt{S1}$ then correctly determines $\texttt{DS}^{\texttt{I}}$ for $\mathbf{x_1}$. Regarding $\mathbf{x_2}$, $\texttt{S2}$ computes $\texttt{DS}^{\texttt{I}}$ at t_5, because there do not exist any relevant detections by any $\texttt{Si} \in \overline{\texttt{Z}}$. Thus, if the query requires $\texttt{DS}^{\texttt{I}}$, nodes outside of the zone must send updates to their detection neighbors inside the zone when objects enter/leave their detection areas.

$\texttt{DS}^{\texttt{B}}$ requires simultaneous detection by nodes in $\texttt{Z}$ as well as $\overline{\texttt{Z}}$. Thus, every $\texttt{Si} \in \texttt{Z}$ must be informed about detections of detection neighbors in $\overline{\texttt{Z}}$ and vice versa.

5.4 Node Failures

When a node fails, there are two possible consequences: (1) An object $\mathbf{x}$ that would have been detected is not detected. (2) Nodes detect $\mathbf{x}$, but the detection-scenario computation is possibly incorrect because it is based on an incomplete relation $\texttt{Detections}$. We have shown how users can express queries if they are interested in objects that are temporarily unobserved in Section 4.3. Therefore we focus on (2), i.e., we notify the user if query results returned could be incorrect due to node failures. We discuss the detection of failures first and continue with failure handling.

5.4.1 Failure Detection

It depends on the strategy used for data collection how failures are detected. A node $\texttt{Si}$ using the reactive strategy requests tuples from its detection neighbors $\texttt{DN}^*$ and expects a response from each of them. If no such response has been received after a timeout, $\texttt{Si}$ derives that the detection neighbors whose responses are missing have failed.

The drawback of the proactive strategy is that nodes cannot detect failures of detection neighbors using missing responses. Without further measures, a failed node might not send updates to detection neighbors and thus affect query results. A practical approach to solve this is sending beacon messages periodically to detection neighbors and assuming node failure if beacons are missing. Our evaluation includes the additional messages induced by this. Note that this problem also occurs with the centralized strategy, i.e., additional messages are required to detect node failures.

5.4.2 Failure Handling

The user must be notified of a node failure if it could have an impact on the query result, i.e., if the computation of the detection scenario is incorrect. In the following, we refer to the node whose failure has been detected as $\texttt{Sf}$. When $\texttt{Si}$ detects the failure of $\texttt{Sf} \in \texttt{DN}_\texttt{i}$ and computes a detection scenario later, the result is possibly incorrect. We denote the detection scenario computed based on an incomplete relation $\texttt{Detections}$ with DS_{fail}.

LEMMA 10. *If $DS_{fail} = \texttt{DS}^{\texttt{B}}$, the failure of $\texttt{Sf}$ did not affect the computation of the detection scenario.*

LEMMA 11. *If $DS_{fail} = \texttt{DS}^{\texttt{I}}$ and $\texttt{Sf} \in \texttt{Z}$ or $DS_{fail} = \texttt{DS}^{\texttt{0}}$ and $\texttt{Sf} \in \overline{\texttt{Z}}$, the failure of $\texttt{Sf}$ did not affect the computation of the detection scenario.*

Summing up, the base station must be notified of node failures in the following two cases:

- $DS_{fail} = \texttt{DS}^{\texttt{I}}$, and $\texttt{Sf} \in \overline{\texttt{Z}}$

- $DS_{fail} = \mathtt{DS}^{\mathtt{O}}$, and $\mathtt{Sf} \in \mathtt{Z}$

This notification is a message that contains DS_{fail} and an identifier of Sf.

6. EVALUATION

We have evaluated our approach thoroughly using simulations and a Sun SPOT deployment to investigate the following hypotheses:

H1 Both distributed strategies scale better with the number of nodes than the centralized strategy.

H2 The proactive strategy is the most energy-efficient for $Inside\,(\mathbf{x}, \mathtt{Z})$ and $Disjoint\,(\mathbf{x}, \mathtt{Z})$.

H3 The reactive strategy is the most energy-efficient for $Meet\,(\mathbf{x}, \mathtt{Z})$.

H4 The centralized strategy is energy-efficient for small networks and nodes around the base station.

H5 Distributed strategies reduce communication required for processing spatio-temporal developments like $Enter\,(\mathbf{x}, \mathtt{Z})$ or $WSNEnter\,(\mathbf{x}, \mathtt{Z})$.

6.1 Simulation Setup

To run exactly the same software for simulations and case study, we used the Sun SPOT simulator of the KSN project [3]. Each simulation run consists of the following steps:

1. Generate a WSN of 100-300 nodes that are randomly deployed over an area. The size of the area is constant to account for different node densities, i.e., varying numbers of detection and communication neighbors.
2. Define a zone of varying size. Zones contain between 2 and 30 nodes.
3. Generate 50 different object paths using a random walk model with starting points randomly chosen.
4. For each object path evaluate each detection scenario using each strategy.
5. Count the number of messages sent and received.

Overall, the results presented here are based on more than 100.000 simulation runs.

Since detection areas tend to be indeterminable, we have approximated the set of detection neighbors with the set of communication neighbors: To do so, a node sends a beacon message periodically. Each node receiving it adds the sender to the list of detection neighbors. We graph the communication required for these beacons for distributed strategies separately. For the proactive approach, these periodic beacon messages would allow the detection of failures and notification of the base station as well.

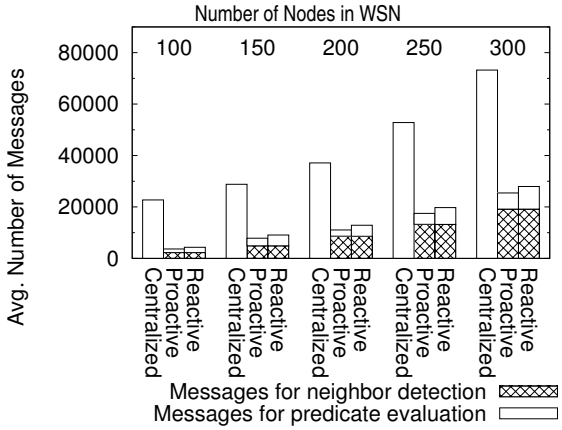


Figure 8: Scalability of data-collection strategies

6.2 Simulation Results

Figure 8 shows the average number of messages per simulation run for WSN of 100-300 nodes to compute $\mathtt{DS}^{\mathtt{I}}$. Graphs for other detection scenarios are similar and omitted here. As expected, the number of messages required by the centralized strategy increases linearly with network size. Contrary to this, network size only affects both distributed strategies marginally. The reason for this is the increasing node density, i.e., more detection neighbors per node. Even the added overhead for the approximation of detection neighbors does not change this. The large share of communication related to detection-neighbor approximation suggests that more sophisticated mechanisms for this could reduce energy-consumption even further. Thus, we conclude that **H1** is true. Detection-neighbor approximation should be investigated in future work.

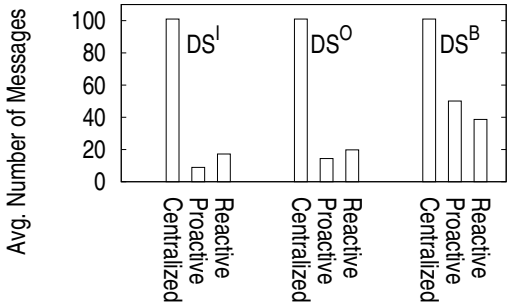


Figure 9: Communication per detection-scenario

Figure 9 shows the average number of messages per detection-scenario computation. The result is that distributed strategies require between $45\% - 85\%$ less messages than the centralized strategy. Comparing both distributed strategies shows that the proactive strategy is advantageous for $\mathtt{DS}^{\mathtt{I}}$ and $\mathtt{DS}^{\mathtt{O}}$. This is expected, because $\mathtt{S}^*$ is smaller for the proactive strategy when objects leave the detection area of a node (cf. Tables 3 and 4). These roles are reversed for $\mathtt{DS}^{\mathtt{B}}$, because the proactive strategy is triggered more often than the reactive one. Summing up, these results confirm **H2** and **H3**.

Strategy	**Number of Messages per Object for**	
	$Enter\,(\mathbf{x}, \mathtt{Z})$	$WSNEnter\,(\mathbf{x}, \mathtt{Z})$
centralized	334	334
proactive	44,3	123,8
reactive	39,1	163,1

Table 5: Avg. number of messages for $Enter\,(\mathbf{x}, \mathtt{Z})$ and $WSNEnter\,(\mathbf{x}, \mathtt{Z})$

The distributed strategies reduce communication to process spatio-temporal developments as well. Table 5 shows the average number of messages to determine that $\mathbf{x}$ conforms to $Enter\,(\mathbf{x}, \mathtt{Z})$ or $WSNEnter\,(\mathbf{x}, \mathtt{Z})$ (cf. (5) and (11)), respectively. As expected, the centralized strategy requires at least twice as much communication since every detection event must be forwarded to the base station. For $WSNEnter\,(\mathbf{x}, \mathtt{Z})$, the proactive strategy is most efficient. This is because this development does not contain $Meet\,(\mathbf{x}, \mathtt{Z})$. The difference between $Enter\,(\mathbf{x}, \mathtt{Z})$ and $WSNEnter\,(\mathbf{x}, \mathtt{Z})$ must be attributed to Lemma 9 because all non-border nodes are basically inactive for $Enter\,(\mathbf{x}, \mathtt{Z})$. Compared to the centralized strategy the savings of distributed strategies are between 51% and 89%. This confirms **H5**.

6.3 Sun SPOT Case Study

Since simulations always abstract from certain real-world phenomena and these may impact performance, e.g., interferences or collisions, we conducted a case study using real sensor nodes. For our case study, we have deployed 26 Sun SPOT sensor nodes and a base station on our office floors. [5] provides exact node positions, the object trajectory and further information on the case study. The query was $Inside\,(\mathbf{x}, \mathsf{Z})$. In analogy to the simulations, we assumed that nodes cannot determine their detection areas by themselves. Thus, a node periodically sent beacons to approximate the set of its detection neighbors, i.e., $\mathsf{DN_i} = \mathsf{CN_i}$.

Strategy	Number of Messages		
	Collect	Result Forward.	Total
centralized	137	0	137
proactive	115	42	157
reactive	145	33	178

Table 6: Case study results

Table 6 shows the result of the case study: The rightmost column contains the total number of messages sent, i.e., the sum of the two columns in the middle which reflect messages for data collection (left) and result forwarding (right). Since the centralized strategy computes all results at the base station, the number of messages sent to forward the result is 0. A simulation that replicated the node setup and object movement of the case study had the same results. This indicates that real-world phenomena from which simulations have abstracted would not significantly change the findings based on simulations.

The centralized strategy required 137 messages, the distributed approaches 20 respectively 41 more. This is expected because the network was relatively small, i.e., messages were forwarded 5 hops at most to reach the base station. Considering the simulation results and the result of the case study, we conclude that **H4** is true. Summing up, our evaluation confirms all of our hypotheses.

7. CONCLUSIONS

For many applications, WSN are used to track moving objects. Research has shown that accessing WSN declaratively is important. But research so far has focused on relational queries which are insufficient to express spatio-temporal semantics inherently required by these applications. This paper is the first that addresses the processing of declarative queries interested in the spatio-temporal relationship of objects detected by WSN and zones. First, we have defined the fundamental concepts of spatio-temporal queries in WSN and the semantics of spatio-temporal predicates for zones. Second, we have provided a space partitioning for zones which allows the application of the 9-intersection model and other existing research results to WSN. Processing of spatio-temporal queries requires nodes to exchange information on objects detected. We have shown how to reduce the respective communication significantly by considering detection neighbors and have proposed two execution strategies for processing predicates in-network. Our strategies can deal with node failures that could affect the query result. We have evaluated our approach using both simulations and a Sun SPOT deployment. Our evaluation shows that distributed strategies perform well, particularly for WSN consisting of hundreds of nodes.

References

[1] A. Ailamaki et al. An Environmental Sensor Network to Determine Drinking Water Quality and Security. *SIGMOD Rec.*, 2003.

[2] A. Arora et al. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 2004.

[3] M. Bestehorn. Karlsruhe Sensor Networking Project, 2010. http://www.ipd.kit.edu/KSN/.

[4] M. Bestehorn et al. Deriving Spatio-Temporal Query Results in Sensor Networks. In *SSDBM*, 2010.

[5] M. Bestehorn et al. Energy-Efficient Processing of Spatio-Temporal Queries in Wireless Sensor Networks (Extended Version). *Technical Report of the Karlsruhe Institute of Technology*, 2010. http://dbis.ipd.kit.edu/english/1588.php.

[6] P. Bonnet et al. Querying the Physical World. *Personal Communications, IEEE*, 2000.

[7] P. Bonnet et al. Towards sensor database systems. In *MDM '01*, 2001.

[8] D. Chu et al. The Design and Implementation of a Declarative Sensor Network System. In *SenSys*, 2007.

[9] J. Considine et al. Approximate aggregation techniques for sensor databases. In *ICDE '04*, 2004.

[10] M. J. Egenhofer and R. D. Franzosa. Point set topological relations. *IJGIS*, 1991.

[11] M. Erwig and M. Schneider. Developments in spatio-temporal query languages. In *DEXA*, 1999.

[12] M. Erwig and M. Schneider. Spatio-temporal predicates. *IEEE TKDE*, 2002.

[13] R. Fonseca et al. The collection tree protocol (ctp), 2007.

[14] L. Forlizzi et al. A data model and data structures for moving objects databases. In *SIGMOD*, 2000.

[15] S. Gaal. Point set topology. *Academic Press*, 1964.

[16] C. Gamage et al. Security for the mythical air-dropped sensor network. In *ISCC*, 2006.

[17] R. H. Güting et al. A Foundation for Representing and Querying Moving Objects. *ACM TODS*, 2000.

[18] R. H. Güting et al. Modeling and querying moving objects in networks. *VLDB J.*, 2006.

[19] W. Koenig et al. Detectability, Philopatry, and the Distribution of Dispersal Distances in Vertebrates. *Trends in Ecology & Evolution*, 1996.

[20] Y. Kotidis. Processing proximity queries in sensor networks. In *DMSN '06*, 2006.

[21] P. Langendorfer et al. A Wireless Sensor Network Reliable Architecture for Intrusion Detection. In *NGI*, 2008.

[22] N.-H. Liu et al. Long-term animal observation by wireless sensor networks with sound recognition. In *WASA '09*, 2009.

[23] S. Madden et al. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS OSDI*, 2002.

[24] S. Madden et al. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM TODS*, 2005.

[25] M. U. Mahfuz and K. M. Ahmed. A Review of Micro-Nano-Scale Wireless Sensor Networks for Environmental Protection: Prospects and Challenges. *STAM*, 2005.

[26] J. M. Metsaranta. Assessing Factors Influencing the Space Use of a Woodland Caribou Rangifer Tarandus Caribou Population using an Individual-Based Model. *Wildlife Biology*, 2008.

[27] C. E. Perkins et al. Internet Connectivity for Ad Hoc Mobile Networks, 2002.

[28] J. W. Rettie and F. Messier. Hierarchical Habitat Selection by Woodland Caribou: Its Relationship to Limiting Factors. *Ecography*, 2000.

[29] M. A. Sharaf et al. Balancing energy efficiency and quality of aggregate data in sensor networks. *The VLDB Journal*, 2004.

[30] E. Shih et al. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *MOBICOM '01*, 2001.

[31] SUN Microsystems Inc. SPOT, 2009.

[32] R. B. Tilove. Set Membership Classification: A Unified Approach to Geometric Intersection Problems. *IEEE TC*, 1980.

[33] G. Trajcevski et al. The geometry of uncertainty in moving objects databases. In *EDBT*, 2002.

[34] G. Trajcevski et al. Managing uncertainty in moving objects databases. *ACM TODS*, 2004.

[35] O. Wolfson et al. Moving objects databases: Issues and solutions. *SSDBM*, 1998.

[36] XBow Technology Inc. Wireless sensor networks, 2009.

[37] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Rec.*, 2002.

[38] M. L. Yiu et al. Retrieval of spatial join pattern instances from sensor networks. *Geoinformatica*, 2009.