# Towards Multi-Purpose Wireless Sensor Networks

Jan Steffan Ludger Fiege Mariano Cilia Alejandro Buchmann
Darmstadt University of Technology
D-64289 Darmstadt, Germany
steffan@ito.tu-darmstadt.de
{fiege,cilia,buchmann}@dvs1.informatik.tu-darmstadt.de

## Abstract

*Current wireless sensor network (WSN) architectures are based on the assumption that all sensor nodes are participating in a single global task. In many scenarios, however, it will be desirable to use a single sensor network for multiple concurrent applications. In order to enable such multi-purpose WSNs efficiently, delimiting each application to its specific set of relevant nodes is one of the key issues that needs to be solved.*

*We present scoping as a general concept for the creation and maintenance of network-wide node subsets and describe a flexible and modular architecture that meets the requirements of multi-purpose WSNs.*

## 1. Introduction

Wireless sensor networks (WSNs) consist of hundreds or thousands of low-power nodes that form dynamic ad-hoc multi-hop networks. The desired behavior of WSNs have to be achieved by local algorithms running on each sensor node with very limited knowledge about the whole sensor network. A central theme of past and current research is the improvement of the robustness and resource utilization of such algorithms. Both robustness and efficiency are important and often conflicting optimization goals under the dynamic behavior and resource constraints of the proposed WSNs.

In this paper we look at WSNs from a slightly different angle. In many commercial deployments of WSNs the energy consumption and thus the unattended lifetime of sensor nodes may not be the dominating cost, but only one aspect contributing to the total cost of a WSN. Other cost factors include the development of WSN applications, maintenance and the return-of-investment (ROI). ROI does not only depends on the lifetime of sensor nodes, but also on the "usefulness" of a WSN. Considering this, the focus shifts from energy efficient special-purpose WSN solutions towards cost efficient flexible and modular multi-purpose WSN infrastructures. A similar transition can be found in the area of wired sensor and actuator networks such as field busses in cars, buildings or plants, where multiple independent applications are using the same infrastructure today.

An important building block for the implementation of multi-purpose WSNs is the separation of different tasks both at node and networking level [19]. There are multiple solutions for multitasking-like functionality at node level, e.g., query engines capable of executing multiple concurrent queries, the Maté virtual machine developed at Berkeley [9], or the SOS operating system [13].

In order to avoid resource conflicts between concurrent applications and to use large scale WSN installations efficiently [3] it is necessary to delimit the scope of each application to the subset of relevant nodes. Mechanisms are required for the selection and discovery of these groups of nodes. Changes of group membership and connectivity due to various changes in the WSN have to be handled. In [15] we have introduced *scoping* as a middleware building block and abstraction layer for these tasks. This builds on earlier work on scoping in large scale publish/subscribe systems [5] where similar problems arise. Here we describe a modular architecture that utilizes scoping in order to meet the requirements of multi-purpose WSNs.

The remainder of the paper is structured as follows: The following section gives a practical example of an application scenario for multi-purpose WSN and illustrates potential uses of scoping. Section 2 discusses related work on grouping and node selection in WSNs. An outline of our scoping architecture for multi-purpose WSNs is presented in Section 3. Section 4 describes access control as an example of binding additional services to scopes. Section 5 closes with a summary and directions for future work.

### 1.1. Motivating scenario

We consider a freight container monitoring scenario as an example for a multi-purpose WSN. We will use

it throughout this paper to illustrate various cases of structuring through groups of nodes.

Sensors deployed inside containers can monitor environmental conditions for perishable goods; detect tampering or leakage of dangerous goods; or provide an RFID-based real-time inventory [2, 6, 10]. Nodes that are capable of communicating with the outside of a container connect the sensors to data-sinks by forming an inter-container ad-hoc network. The layout of this network is regular, forming a grid-like two- or three dimensional matrix.

This inter-container network can be used for additional purposes such as tracking containers during their journey, detecting containers that went over-board or were forgotten somewhere, or determining the position of a specific container.

Obviously all of these applications make use of the WSN in a distinct way. Environmental condition monitoring, for instance, makes use of temperature and humidity sensors which are not relevant for the tracking application. Moreover, there are various aspects that require further differentiation (see also figure 2):

- for monitoring environmental conditions, different sensor types, sampling rates and thresholds are appropriate for different goods.
- a higher temperature sampling rate might be necessary for containers at positions which are exposed to sunshine. In containers at inaccessible inner positions the sensors for tampering detection can be deactivated to save energy.
- multiple parties are involved, such as the owners of the containers, the goods providers, or official authorities. Much of the collected information such as inventory or condition of goods is business-critical and should not be available to competitors.

While technically each application is feasible with current systems, the efficient and economical integration of several of these applications within the same WSN infrastructure is only possible if the following properties can be offered:

- application-specific grouping of sensor nodes based on various conditions such as position, capabilities or other meta-data
- a high-level abstraction of node groups in order to enable the economical development of multiple applications
- modular and extensible functionality in order to support different group formation schemes and other application-specific requirements
- means of restricting sensor access or the visibility of sensor data to authorized parties

## 2. Node selection and grouping in WSNs

Early work about attribute based addressing of nodes in WSNs was published in [7]. This is a general approach towards attribute based matching of queries and data provided by sensor nodes. Considered attributes include the type of data, its accuracy and the geographic location of the originating node. Most other query languages for WSNs also include some mechanism for selecting nodes as data sources based on their properties such as the geographic location. Node selection here usually is an integral part of the query execution. The group of nodes matching a condition is not considered as a logical structure of its own. This prevents the reuse of node groups for multiple queries and the association of abstractions and services such as data sharing or access control with node groups.

Recently a number of approaches proposed dynamically selected groups of nodes as an abstraction in order to simplify the programming of WSNs. For instance, Hood [18] and Abstract Regions [17] are motivated by the observation that many data-collection algorithms for WSNs are based on the creation of local node-clusters. They allow the selection of a node subset amongst the neighbors of a node by means of a declarative condition. Both provide abstractions for data sharing between selected nodes. Hood and Abstract Regions require group specifications and algorithms to be fixed at compile time. Moreover, all sensor nodes must share the same code base. A differentiation of nodes happens at run-time based on the local evaluation of node selection conditions. RegionsVM was mentioned in [9] as a more flexible implementation of Abstract Regions based on the Maté virtual machine.

While Hood and Abstract Regions are restricted to neighboring nodes for group formation, the role based approach presented in [14] allows the network-wide selection of nodes. Nodes are preprogrammed for a number of roles with associated tasks. Nodes select their current roles based on some conditions at runtime. However, there is no structure that would connect nodes sharing a common role to an abstract group.

Regiment [11] is a conceptual approach that operates on the set of all sensor nodes as a whole. All sensor readings originating from any node are viewed as a single so called region stream. This region stream can be transformed and narrowed down to data originating from a partial set of nodes through filters and mappings.

There is more work that implicitly uses groups of nodes in order to implement efficient data collection algorithms based on aggregation and clustering. A list of algorithms based on neighborhood groups can be found in [18].

Multicast algorithms for mobile ad-hoc networks [1, 8] are related, too. These approaches, however, focus on transferring the multicast paradigm of wired networks to mobile

ad-hoc network environments. It is assumed that nodes decide upon their group membership themselves by subscribing to the appropriate multicast id. This is contrary to the appropriate model for WSNs where nodes are selected based on their properties.

None of the approaches mentioned above facilitates network-wide selection of nodes to establish structures that can be used to delimit applications and perform the actual tasks. Moreover, it is not possible to instantiate new groups of nodes at run-time, which is inevitable for multi-purpose WSNs where applications are added and changed after node deployment.

## 3. Scoping architecture for multi-purpose WSNs

This section gives an overview of the scope-based architecture we have developed to meet the previously mentioned requirements. Scoping as a means of structuring distributed event-oriented systems was first developed in the context of publish/subscribe systems [4, 5]. A *scope* is basically a group of nodes that is specified through a membership condition. Considering scopes as first class objects provides an orthogonal means for structuring distributed systems and allows additional services such as access control or different communication semantics to be bound to scopes. Orthogonality is important in order to achieve the required degree of modularity and flexibility. Once created, a scope can be reused multiple times for different queries or tasks.

This is reflected in our modular architecture which is roughly divided into four layers as depicted in figure 1. The bottom layer comprises all low level hardware abstractions for network and sensor access as they are provided by WSN operating systems such as TinyOS. The top layer is responsible for node level multitasking capabilities, i.e., the lifecycle and memory-management necessary to execute and instantiate multiple concurrent tasks at runtime. This layer can be implemented, for instance, as a query-engine or virtual machine and is relatively independent of the scoping functionality. Scoping is provided in the two middle layers: *a)* the scope-state layer for handling the scope membership status of each node; *b)* the routing layer for the dissemination of scope creation requests and intra scope communication. These are described in more detail in the following subsections.

### 3.1. Scope specification language

A fundamental difference between WSNs and most other networks is that at application level nodes are not addressed individually by some address, but by their properties or context. A scope therefore is defined as the group of nodes that

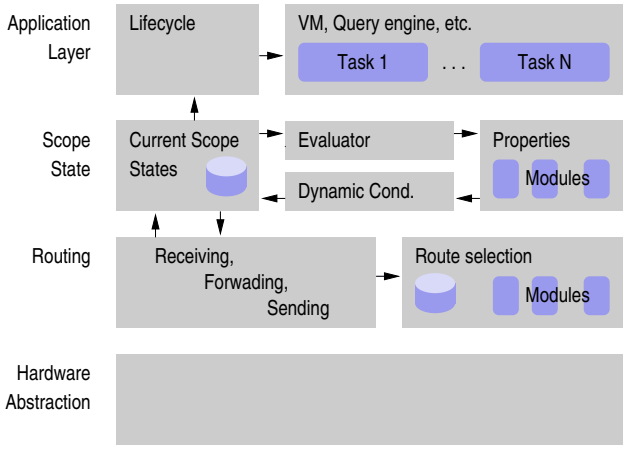



**Figure 1. Layered architecture of scoping implementation for WSNs**

match a given condition. A central part of a scope specification language for WSNs is therefore the specification of scope membership conditions.

**3.1.1. Node properties** Membership conditions are based on node-properties, which can be static or dynamic. A typical static property is “node has a temperature sensor”. Dynamic properties are “node is in geographic region *X*” or “node has no more than three neighbors”. Node properties are obviously in part scenario specific. In the container example it makes sense to have properties reflecting the presence of a container at each of the six possible neighboring positions. Properties are therefore implemented as exchangeable modules. Dynamic properties are of particular interest as their alteration has to trigger a reevaluation of the scope membership condition. They are implemented as specialized modules that trigger an event when the property has changed.

**3.1.2. Namespace** In order to enable a conflict-free addition of scenario-specific properties and to make property implementations distinguishable, we use hierarchical namespaces for naming properties. The top-level part of a property name identifies the module that implements the property. A partial list of top-level identifiers might be `sensor` for sensor related properties, `geo` for geographic location, `neighb` for neighborhood properties and `meta` for meta-information. A static boolean property reflecting the existence of a temperature sensor could be named `sensor.type.temperature`. For the container scenario a specific property could be `meta.container.owner` for the owner of container the sensor node was placed in.

In order to comply with the resource restrictions of WSNs, property identifiers and all other aspects of the scope

specification language are mapped to a more compact bytecode at compile time.

**3.1.3. Expressiveness** An important question is the expressiveness of a specification language for scope membership conditions based on node properties. There are three major levels of expressiveness to be considered:

- Expressions that can be evaluated locally are based only on constants and locally available current node properties. Expressions can be formed from functions over properties and constants (e.g. +, -, `log()`), predicates and comparison operators (e.g. ==, <) and boolean expressions over these.
- Expressions over historic values require local memory. Examples are derivations of properties over time such as velocity or acceleration and aggregations over time windows such as average or variance.
- Aggregations over non-local properties require the synchronization of properties with multiple nodes. Conditions like these are useful to select the most suitable from a set of candidate nodes. This is feasible when the set of candidate nodes is restricted to a neighborhood. Whether aggregations over candidate nodes can be handled efficiently in general is an open question. An alternative solution would be to implement certain aggregated values as local properties which are handled by special modules. Those modules can handle the data synchronization in the most appropriate and efficient way for the given property.

We chose the first option as it results in a simple architecture and interface without serious restrictions of expressiveness. If aggregations over time or space are required in certain cases, these should be implemented within a property module.

**3.1.4. Nested scopes** In many applications it makes sense to use an existing scope as the candidate set for a second scope. In the container scenario for instance there are many potential uses of nested scopes. For example there could be a scope *A* over all nodes that are placed inside containers belonging a company. Within this scope the company could create its own sub-scope *B*. Figure 2 shows an example of nested scopes. Moreover, nested scopes can be used to enforce security policies as we will show later in section 4.

The main benefit of nested scopes is the increased efficiency of message propagation and group maintenance due to the delimited candidate set. Each scope specification includes the definition of a *basescope*. The default base-scope is *world* which includes all available nodes. Scope propagation is limited to the basescope.

**3.1.5. Life-cycle** In the approaches discussed in section 2 the specification of node groups remains valid during the whole deployment time of the WSN. When multiple concurrent applications are considered, this is not feasible as the life-cycle of all of the applications is not likely to coincide with the life-cycle of the WSN.

We provide therefore mechanisms for the creation and destruction of scopes at run-time. Scope creation can be initiated by the gateway node or by any other node within the network. Issues of propagating a request for the creation of a new scope towards the potential member nodes are discussed in section 3.2 below.

The lifetime of scopes is delimited through predefined leases that may be extended. This way member nodes can decide locally when a scope should be removed by dropping all information about it. Local decision avoids communication overhead and prevents stale scope instantiations in disconnected areas. The lifetime of a scope can be adjusted according to its usage pattern such as single-shot queries or long term applications.

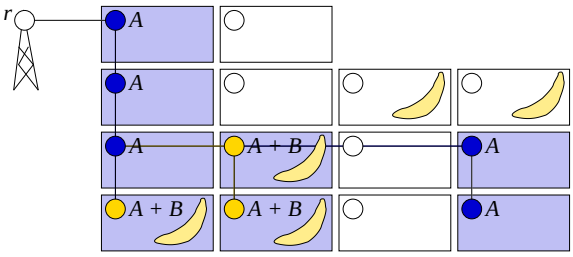
## 3.2. Scope propagation and maintenance

Scope creation can be requested through a gateway node or by sensor nodes within the network. We are calling this node the scope's *root*. Inner sensor nodes in most cases will create local neighborhood scopes for tasks such as data aggregation. Scopes created by gateway nodes will typically have global character. As mentioned previously a scope request consists of a membership condition, a basescope and the time-to-live of the scope.

During scope propagation a routing tree is constructed from the root node to each node within the basescope that matches the membership condition. During the lifetime of the scope both the connectivity of the routing tree and scope membership has to be maintained. The main task of the routing tree is the dissemination of queries or mobile code to the members of a scope and the transmission of query results or events from the scope members back to the scope root. An example of such a routing tree is shown in figure 2.

A new scope creation request is disseminated through the routing tree of an already instantiated basescope. The basescope *world* refers to a special global scope that covers all nodes of the WSN. The world scope is persistent and provides connectivity between all sensor nodes. Scenario specific parameters such as the frequency of neighbor updates are encapsulated within the mesh routing algorithms of the world scope.

Scope propagation can be optimized by exploiting membership conditions that are related to some globally known structure. This is the case for properties such as the geographic location of the hop count from the scope root. Membership conditions are therefore not only used for lo-




Scope over sensors in containers leased by “Blue Co”:
$A := \{v \in world : v.meta.owner == \text{“Blue Co”}\}$

Sensors in Blue Co’s Containers carrying bananas:
$B := \{v \in A : v.meta.cargo == \text{“Bananas”}\}$

**Figure 2. Node $r$ is the scope root of both $A$ and $B$. Scope $A$ is partitioned into two areas and connected through its routing tree. Scope $B$ is a nested scope within $A$.**

cal membership decisions, but also to estimate the chance of finding matching nodes in a certain direction. To this end the scoping layer provides an interface to aid the routing layer at route selection based on the scope membership condition.

When a scope request reaches a node that meets the membership condition, the scope’s ID and specification is recorded in a local table. The node will then handle messages that are disseminated through this scope as long as the scope is valid and the node still meets the membership condition.

Candidate nodes that currently do not meet the membership condition store the scope specification during the scope’s lifetime if membership depends on dynamic properties that might change at a later point of time. Changes on nodes’ properties that may affect their membership are evaluated against the scope’s membership condition.

Nodes which are not member of a scope themselves but lie on a routing path between scope root and a scope member store an appropriate entry in their routing table.

If a node gets a new neighbor due to, for instance, node mobility, it first exchanges all scope requests with this neighbor that are within the global basescope. This is repeated recursively for all matching nested sub-scopes.

## 4. Access control

There are many potential configurations and services that might be bound to scopes. We present here access control as one example. More examples have been published in [15].

There are many potential security threats against WSNs [12]. Most work on security measures for WSNs so far has focused on the protection of the sensor network as a whole. New security issues arise when multi-purpose WSNs are considered. If multiple applications are active concurrently, they have to be protected from each other—especially if these applications are run by different parties. Data and sensor-nodes belonging to or associated with one party should not be accessible by another. This can be achieved easily with the help of scopes and a public key authentication scheme:

Before scope creation the root node of a scope generates a public/private key pair. The secret private key stays at the root node. The public key is distributed together with the scope creation request to all member nodes of the scope.

All following messages such as queries or requests for nested scopes that are sent from the root node are signed with its private key. Scope members can then verify that messages really originate from the same node that created the scope. Other nodes cannot request the execution of queries or code within the scope neither request the creation of a nested sub-scope. In the opposite direction the root node’s public key can be used by the scope members to encrypt data such as query results before it is sent back to the root node.

Although this scheme does not cover mutual authentication between any pair of sensor nodes it covers the two most important security aspects of multi-purpose sensor networks: differentiated access control for sensor nodes that are grouped within a scope and encryption of sensitive data originating within the scope.

If keys are maintained at a gateway node, a finer grained access control could be realized outside of the WSN. Parties that wish to send some request within a certain scope would first have to acquire a signature from the owner of this scope.

As only a single key per root node or scope is required the usual key distribution problems within WSNs do not arise. Member nodes of scopes have to perform only relatively lightweight public-key operations which has been shown to be feasible with current WSN hardware [16].

## 5. Conclusion and future work

Today WSNs focus on dealing with a single global task. We argue in favor of a single WSN for multiple concurrent applications and we present a scenario in this sense. For this purpose a modular multi-purpose WSN infrastructure is required, helping to reduce development and maintenance costs.

Multi-application support requires means of separating applications both at node and network level. Application deployment must be possible dynamically at run-time. In order to achieve the desired efficiency and scalability it is necessary to delimit the scope of applications to a subset of relevant nodes. A multi-purpose WSN architecture like

this also needs a high degree of adaptability and modularity in order to satisfy the requirements of different applications and scenarios.

We presented a modular architecture based on the concept of scoping that meets these requirements. Scopes are both an abstraction layer for global and local groups of nodes and a middleware building block. They may serve as an extension point that allows to integrate other solutions like access control or quality of service mechanisms.

We are currently working on a prototype implementation based on the SOS operating system [13] for the Mica mote platform. One goal of this implementation is to assess the influence of multiple applications on the routing overhead. On the one hand deployment and maintenance of scopes causes the transmission of additional messages. On the other hand the relation of routing overhead versus payload is improved by sharing a routing tree between multiple scopes and using scopes for multiple purposes.

Future work includes the improvement of routing efficiency by exploiting properties of scope membership conditions and by sharing routing information between scopes. The extension of scopes with services like access control also provides an interesting field for future research.

## References


[1] C. Cordeiro, H. Gossain, and D. Agrawal. Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Network*, 17(1), January 2003.

[2] C. Decker, M. Beigl, A. Krohn, U. Kubach, and P. Robinson. eSeal - a system for enhanced electronic assertion of authenticity and integrity of sealed items. In *Proceedings of the Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science (LNCS)*, pages 254–268. Springer Verlag, 2004.

[3] Henri Dubois-Ferrière and Thanos Stathopoulos. Efficient and practical query scoping in sensor networks. In *1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004.

[4] Ludger Fiege. *Visibility in Event-based Systems*. PhD thesis, Department of Computer Science, Darmstadt University of Technology, April 2005.

[5] Ludger Fiege, Mira Mezini, Gero Mühl, and Alejandro P. Buchmann. Engineering event-based systems with scopes. In B. Magnusson, editor, *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, volume 2374 of *LNCS*, pages 309–333, Malaga, Spain, June 2002. Springer-Verlag.

[6] Chunlong Guo and Andy Fano. Cargo container security using ad hoc sensor networks. In *Fourth International Conference on Information Processing in Sensor Networks IPSN 2005*, Los Angeles, CA, April 2005. (to appear).

[7] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 146–159. ACM Press, 2001.

[8] Sung-Ju Lee, William Su, Julian Hsu, Mario Gerla, and Rajive Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *INFOCOM 2000*, March 2000.

[9] Philip Levis, David Gay, and David Culler. Bridging the gap: Programming sensor networks with application specific virtual machines. Technical Report UCB//CSD-04-1343, UC Berkeley, August 2004.

[10] Daniel Machalaba and Andy Pasztor. Thinking inside the box: Shipping containers get 'smart'. *The Wall Street Journal*, January 2004. `http://www.skybitz.com/newsroom/pdf/WSJ1.15.04.pdf`.

[11] Ryan Newton and Matt Welsh. Region streams: Functional macroprogramming for sensor networks. In *Proceedings of the First International Workshop on Data Management for Sensor Networks (DMSN)*, Toronto, Canada, August 2004.

[12] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM special issue: Wireless sensor networks*, 47(6):53–57, 2004.

[13] Ram Kumar Rengaswamy, Roy Shea, Eddie Kohler, and Mani Srivastava. SOS: A dynamic operating system for sensor networks. In *MobiSYS '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM Press, 2005.

[14] Kay Römer, Christian Frank, Pedro José Marrón, and Christian Becker. Generic role assignment for wireless sensor networks. In *Proceedings of the 11th ACM SIGOPS European Workshop*, pages 7–12, Leuven, Belgium, September 2004.

[15] Jan Steffan, Ludger Fiege, Mariano Cilia, and Alejandro Buchmann. Scoping in wireless sensor networks: A position paper. In *Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-hoc Computing*, pages 167–171. ACM Press, October 2004.

[16] Ronald Watro, Derrick Kong, Sue fen Cuti, Charles Gardiner, Charles Lynn, and Peter Kruus. Tinypk: securing sensor networks with public key technology. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 59–64, New York, NY, USA, 2004. ACM Press.

[17] Matt Welsh and Geoff Mainland. Programming sensor networks using abstract regions. In *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, 2004.

[18] Kamin Whitehouse, Cory Sharp, Eric Brewer, and David Culler. Hood: a neighborhood abstraction for sensor networks. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 99–110. ACM Press, 2004.

[19] Alec Woo, Sam Madden, and Ramesh Govindan. Networking support for query processing in sensor networks. *Commun. ACM special issue: Wireless sensor networks*, 47(6):47–52, 2004.