Scoping in Wireless Sensor Networks

A Position Paper

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

ABSTRACT

One of the trends of wireless sensor networks (WSN) is to allow multiple applications to run on top of the same sensor network. This will have an enormous impact on the way WSN applications are developed, deployed and maintained. Many applications for WSN are still developed on very low level functions provided by simple operating systems or bare hardware. Alternatively, generic WSN middleware focuses on very high-level system abstractions, such as declarative query languages, and acts as black box that tries to automatically map applications to the underlying resources.

In this paper, we propose scopes as a generic abstraction for the definition of groups of nodes. They bridge the gap between high- and low-level interfaces and enable the partitioning of WSN functionality. As middleware building block they facilitate the construction of tailored services in multipurpose WSNs.

Keywords

Multi purpose wireless sensor networks, Scopes

Categories and Subject Descriptors

C.2.4 [**Computer-Communication networks**]: Distributed Systems—*Distributed applications*; D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*

1. INTRODUCTION

Wireless sensor networks (WSNs) are discussed as platforms for many new kinds of applications, like habitat or health monitoring, building automation, and logistics. And with falling hardware costs they promise to become widely applicable. However, today many WSNs and applications are still designed and deployed for one specific purpose. The lack of versatility increases the complexity and cost of developing WSNs and applications, preventing their industrial adoption.

In view of hardware improvements, sensor networks will typically be used for longer periods of time and for multiple purposes [14]. In some cases they may even undergo evolutionary development and continuous adaption. Multipurpose WSNs will be heterogeneous in most cases: for each application only a specific type of sensor node or some parts of the monitored environment may be relevant. The two main challenges of multipurpose WSNs are the support of divergent application requirements and the efficient partitioning of the WSNs with regard to these requirements.

The envisioned multipurpose networks [13] are tackled by current research in two quite different approaches. On the one hand, there are low level implementations that focus on programming individual sensors and their sensing/acting and communication facilities directly. Examples are simple operating systems like TinyOS [8] On the other hand, there is work on declarative query processors offering high-level interfaces [9, 1], which do not allow explicit control on the level of individual nodes.

The difficulty lies in finding a good trade-off between the universality of such high-level interfaces and the degree to which application specific details can be passed and utilized for the optimization of routing and resource scheduling. In order to resolve this problem we propose a scoping approach that partitions WSNs and their functionality. This combines capabilities of high-level specifications with low level programming of nodes. Multiple applications with divergent requirements can coexist within the same WSN and still use state-of-the art algorithms for the respective implementations. Scopes serve as a building block for creating WSN middleware and adapting it to application needs.

1.1 Motivating Scenario

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

Sensors deployed inside containers can monitor environmental conditions for perishable goods; detect tampering [3] or leakage of dangerous goods; or provide an RFID-based real-time inventory. 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 these networks is regular, forming a grid-like two- or three dimensional matrix.

2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing Toronto, Canada

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 establishing the position of a specific container in a stack.

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:

- For environmental conditions monitoring, 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.
- Multiple parties are involved, such as the owners of the containers, the cargo, or the ship 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 single application is feasible with current sensor hardware, the efficient and economical integration of several of these applications within the same WSN infrastructure is only possible if the following properties can be achieved:

- application-specific grouping of sensor nodes based on various conditions such as position, capabilities or other kinds of 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. SCOPES IN WSN

Traditional publish/subscribe systems exhibit problems that are similar to the ones mentioned in the introduction in terms of missing control. Scoping has been successfully introduced to amend these problems [6], and the basic ideas are described and applied to WSN here. In a pub/sub system producers and consumers of data interact indirectly by publishing notifications about events they have observed, and consumers announce their interest in certain kinds of notifications by issuing subscriptions. An intermediate notification service conveys notifications in a network of routers to those consumers having a matching subscription. The data-centric communication in sensor networks is similar in nature.

2.1 Scope Model

In the original scoping model, producers and consumers are the components of the system, i.e., the software artifacts that operate and communicate. Within the domain of sensor networks, components implement all kinds of high-level functionality within sensor nodes, such as query processing or in-network processing of data. The idea of scopes is to provide means to define groups of components and limit the visibility of messages sent within groups. In pub/sub systems, a published notification is visible to a consumer only if it is in the same scope as the producer. In a more general sense, scopes serve two purposes: a) as design tool, they identify and delimit groups of components, b) as part of the infrastructure, they control the dissemination of data in the network. A component can participate in multiple scopes simultaneously, and scopes can be nested. The resulting structure of the system is given by a directed, acyclic graph of simple and complex components. However, for the purpose of this paper nested scopes are not considered.

There are two versions of a scope graph. The *descriptive* scope graph describes an application, the components and the scopes it comprises and how they interrelate. The graph is thus comparable to source code specified by the programmer. The *deployed* scope graph describes the components actually running in the system; note that the graph is still a conceptual data structure that is not necessarily instantiated in the nodes. It conceptually includes information on where and how many instances of a scope/component are running. From an abstract point of view, the difference between descriptive and instantiated state can be seen as the result of a transformation. And one of the main advantages of scopes is that this transformation can be governed by annotations to scopes. These annotations are generic, they are interpreted and implemented by the middleware at deployment and at runtime.

2.2 Applications of Scopes in WSN

As already recognized, for example in [13], the ability to model various logical or geographical node-subsets is one of the main design dimensions of sensor network applications. There is a need for both network-wide and local node-subsets such as the neighborhoods proposed by Hood [12] and abstract regions [11]. In our example, nodes within a container are composed in one scope, nodes with temperature sensors would be composed in a different scope (independent of the actual containers they belong to), and the set of containers belonging to one transport company, containing cargo of one owner, or carrying certain kinds of dangerous goods would each constitute a scope of its own.

The presented scoping concept supports these kinds of node selection. Annotations on scopes carry rules that determine at deployment time which nodes belong to the scope (cf. Section 3). In addition to [13], scopes may also carry specifications determining the most suitable routing algorithm for the communication between scope members or special requirements regarding timeliness, fault-tolerance or security. In the container example, messages can simply be broadcasted within a container scope. The temperature scope, on the other hand, constitutes an overlay that must be mapped to underlying communication facilities (cf. Section 4).

The separation of communication on the level of scopes allows us to control the (side) effects of independent ap-

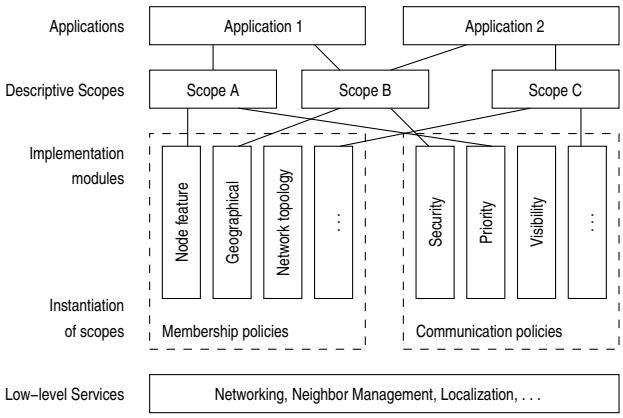

Figure 1: Abstract model of scope application in WSN

plications, which would otherwise not be possible without adapting potentially *all* involved components. Furthermore, application development is made easier. Consider a temperature reading emitted via a `send()` call. The actual functionality needed here is determined by the reader's scope membership. On deployment, this call is bound to middleware primitives that obey the overlay. Moreover, if the set of required services is given by the scope definition, the footprint of the WSN can be reduced. Unnecessary services need not be active/deployed on the node. A future extension would be to even allow for dynamic application *and* service deployment.

Scoping is an abstraction for system programming that is located on a higher level than application source code, but which does not hide all the system details. And it defers deployment-specific decisions until resources and constraints are known.

3. NODE SELECTION

Perhaps the most apparent use of scope annotations is node selection. Rules specify on which nodes a component is to be deployed, and thus determine scope membership of the transformed, i.e., instantiated scope graph components.

3.1 Selection Rules

Various kinds of selection rules determine at deployment time and/or at runtime on which nodes of the sensor network the scope's components are to be deployed. Examples are selection based on node features, geographic coordinates, topology information, etc.

Node features are typically static information about the node's capabilities in terms of hardware, such as sensor types, maximum sampling rate, maximum accuracy, persistent storage or extended processing capabilities and available software modules. Selection can also be based on node-specific meta-data, such as the owner of the node. This could be used to separate the monitoring of goods or containers that belong to different companies, possibly in combination with security requirements. Dynamic properties, like the current battery level or load, are often too generic, because they relate selection on node state in general.

Geographical scoping is already an implicit feature of many WSN implementations, e.g. [7]. A geographic scope can be defined by absolute or relative coordinates, such as "all nodes within 100m of the current node". The latter is often useful within event-triggered queries that collect and aggregate additional data locally.

Selection based on *network topology* also considers node positions, but on different measures. There are various network topology properties used in current WSN algorithms, such as the number of neighbors, the hop-count from another node or the node density. A typical application would be to select a low density subset of nodes that sufficiently covers an area, which could be used for habitat monitoring with a minimal number of nodes. The formation of local clusters is another example [4].

While the above selection rules are ad-hoc examples of those proposed in [13]. Our main idea is to have an extensible set of rules to which application-specific rules can be added. This requires the availability of a module that handles the deployment and maintenance of a scope as specified by its node selection rule.

3.2 Scope Deployment and Maintenance

For each type of rule there is a specialized implementation module that maps the scope definition to the underlying network. So, new rule types can be added and existing solutions can evolve independently. The task of these modules is to perform the part of the transformation of the descriptive scope graph, which comprises two steps. First, *scope deployment* creates a scope and installs its components for the first time on the selected nodes. Second, *scope maintenance* updates the scope and adds (removes) nodes that are now (no longer) covered by the selection rules. Both steps install, activate, deactivate and deinstall components, which is described in Section 4.

3.2.1 Scope deployment

This step includes discovering nodes that should run some components of the scope, installing the components, and finally a strategy to instantiate the scope itself. *Discovery* obviously relies on some routing algorithms to search for the appropriate nodes, and the choice which algorithm to use is guided by the node selection rule associated with the scope. For instance, to implement geographic scopes, existing geographic routing algorithms can be reused. The most simple implementation is flooding with a local evaluation of the scope membership rule.

For the *strategy* to instantiate the scope we differentiate in this paper only two alternatives. First, there may be no strategy/no instantiation, that is, besides node selection and component installation the scope does not affect system operation. Second, the scope is instantiated and an overlay structure is established that connects scope members and allows for scope internal communication. This realization of visibility control is described later.

3.2.2 Scope maintenance

Scope maintenance involves adapting a scope to external changes so that its membership selection remains valid. Changes that have to be considered include

- changes in the network topology typical for ad-hoc networks, such as node or link failures,
- node mobility,

- dynamic aspects of the membership rule itself, such as the current battery level or load.

For the first two points, the routing algorithms employed for node discovery have to be re-evaluated partially to assess the selection rules after these changes. It depends on the actual algorithm whether this reassessment can be done locally or some distributed algorithm (e.g., measuring path lengths, group diameter) is required. As for the communication overlay constructed for a scope, it must possibly be updated to reflect topology changes, at least when the routing cannot cope with these changes itself. For example, directed diffusion maintains alternative paths and even in classic pub/sub networks, epidemic algorithms [2] are being investigated to improve failure behavior.

Finally, scope maintenance includes the task of removing a scope from the network when it is no longer used.

4. ANNOTATIONS ON COMMUNICATION SEMANTICS

In multipurpose sensor networks, a variety of services may be provided as part of the WSN infrastructure (routing, caching, etc.) and, generally, they will likely come in different flavors, offering different levels of quality of service or security. Some examples are given below:

- The original purpose of scopes is to delimit the visibility of messages. This can be used as a means of concentrating tasks on a certain node set, thereby decreasing network load, or in order to restrict the visibility of information to authorized parties.
- Encryption is CPU-intensive and therefore conflicts with the resource constraints of WSNs [10]. Scopes can be used to delimit encryption to those areas where it really is required. Moreover different scopes may be adjusted to individual trade-offs between security level and cost.
- In the same way selecting the appropriate QoS requirements on scope level allows for an efficient utilization of resources and bandwidth.
- Messages originating in different scopes may also have different priorities. Messages related of the monitoring of dangerous goods for instance should have a higher priority than tracking-related messages.

Scope annotations govern the selection of services appropriate for the installed application in the current environment.

4.1 Implementation Issues

Take security as an example. Messages are sent by application components via a `send()` function, which is provided by different WSN services with diverse QoS characteristics. If all communication between members of a scope shall be subjected to (light-weight) encryption, `send()` must be called on the appropriate service. Instead of manually customizing the involved components, system engineers annotate the respective scope to guide the deployment process.

In general, this approach separates applications more clearly from the underlying services and communication hardware. It utilizes modern engineering principles (separation of concerns) and makes it easier to modularize both application and infrastructure functionality.

4.1.1 Overlay Structures

In order to implement communication policies it is often required that some overlay structure be established and maintained. This may be the distribution of encryption keys or the discovery of a routing tree that meets certain QoS requirements. Normally these structures would be established with the deployment of the sensor network or during query dissemination. Annotating a scope with the appropriate communication policies allows us to combine the benefits of both approaches: an overlay structure can be established selectively between the required node set during scope deployment. This structure can then be reused for multiple tasks within the same scope. Considering our container scenario a routing tree with extra redundancy and low latency might be established once in order to connect sensors in the "dangerous-goods scope" with a sink.

4.2 Scoped Communication:

If scopes are used in their original sense of limiting the visibility of messages, it is possible to incorporate available routing schemes and exploit their functionality to constrain the set of receivers much like the scope deployment approaches described in section 3.2.

Another approach is to instantiate administrative instances of scopes in the network and let them maintain the necessary routing and membership information; various alternatives are investigated in [5] for general pub/sub systems.

So far, scoping is considered as functionality between application and routing layers. But the management of groups of nodes can also offer modularization of the lower layers. When scoping is introduced in an extra layer between the routing algorithm and the basic communication primitives, the nodes they see and use for routing is restricted. Moreover, multiple instances of (different) routing algorithms can run simultaneously. This reduces the amount of state information needed for each instance. If tunneling between distant scope nodes is part of the core middleware, the design of routing algorithms is made easier (i.e., modularized), and it may even mask topology changes due to movements or link failures to some extent.

5. CONCLUSIONS

This paper is basically motivated by the need for middleware in multipurpose WSNs. With this in mind, we have proposed scopes as a generic abstraction for grouping nodes and/or components. Scopes serve as a building block for creating WSNs and adapting them to different application requirements. They support system engineering on a higher level than application source code. Deployment-specific decisions are deferred until resources and constraints are known, and thus reduce system footprint.

We have exemplified the potential of scoping in a multipurpose sensor network with the help of a a freight container scenario. The ability to integrate quite different forms of routing and node clustering demonstrates the relevance and usefulness of the scope concept.

This paper outlines many promising directions of future work. The original scope concept includes nested scopes with message filters and transformers between them. This could be used to address heterogeneity issues and in-network processing. If we treat scopes as first class objects in the WSN, many alternatives exists for implementing scope rep-

resentatives, which can be used for many purposes, like placing caches or data aggregation and composition.

Acknowledgments

We gratefully acknowledge the fruitful discussions with Jack Stankovic, Sang Son and the sensor networks group at the University of Virginia.

6. REFERENCES

[1] S. Babu and J. Widom. Continuous queries over data streams. *SIGMOD Record*, 30(3):109–120, 2001.

[2] Paolo Costa, Matteo Migliavacca, Gian Pietro Picco, and Gianpaolo Cugola. Introducing reliability in content-based publish-subscribe through epidemic algorithms. In *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, pages 1–8, San Diego, CA, USA, 2003. ACM Press.

[3] 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.

[4] Henri Dubois-Ferriere and Deborah Estrin. Efficient and practical query scoping in sensor networks. Technical Report 39, CENS, UCLA, Los Angeles, CA, USA, April 2004.

[5] Ludger Fiege. *Visibility in Event-Based Systems*. PhD thesis, Technical University of Darmstadt, Darmstadt, Germany, 2004.

[6] 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.

[7] B. Karp. *Geographic Routing for Wireless Networks*. PhD thesis, Harvard University, Cambridge, MA, October 2000.

[8] Philip Levis, Sam Madden, David Gay, Joe Polastre, Robert Szewczyk, Eric Brewer Alec Woo, and David Culler. The emergence of networking abstractions and techniques in tinyos. In *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, 2004.

[9] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a Tiny AGgregation service for ad-hoc sensor networks. *ACM SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

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

[11] 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)*, March 2004.

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

[13] 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.

[14] Yang Yu, Bhaskar Krishnamachari, and V. E. Prasanna. Issues in designing middleware for wireless sensor networks. *IEEE Network*, 18(1):15–21, 2004.