TKN Telecommunication Networks Group

Technical University Berlin

Telecommunication Networks Group

Semantic Addressing for Wireless Sensor Networks

Vlado Handziski, Andreas Köpke, Christian Frank, Holger Karl

{handzisk, koepke}@tkn.tu-berlin.de, chfrank@inf.ethz.ch, hkarl@ieee.org

Berlin, November 2003

TKN Technical Report TKN-04-005

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

The EYES project (IST-2001-34734) is a three years European research project on self-organizing and collaborative energy-efficient sensor networks. It will address the convergence of distributed information processing, wireless communications, and mobile computing.

The goal of the project is to develop the architecture and the technology which enables the creation of a new generation of sensors that can effectively network together so as to provide a flexible platform for the support of a large variety of mobile sensor network applications.

This document explores the data-centric communication aspects of wireless sensor networks with an emphasis on the protocol entities and mechanisms considered for inclusion in the EYES semantic addressing architecture.

The document was submitted to the European Comission as EYES Deliverable 3.3: Semantic Addressing.

Contents

Introduction

In the envisioned scenarios, the Wireless Sensor Networks (WSNs) are going to be deeply embedded in the physical world, providing close range information about the sensed environment. This focus on the data is one of the defining characteristics that sets them apart from the more general purpose Mobile Ad-Hoc Networks (MANETs). Furthermore, the fact that the radio communication is usually the most costly operation in energy terms, and that the nodes are equipped with non-negligible processing capabilities, gives sense to performing *in-network processing* of the data in order to reduce the number of exchanged messages.

These specific WSNs properties result in applications with communication patterns that are inadequately supported by the current ID-centric MANET protocols. Their efficient programming and execution calls for a new middleware that is data-centric in nature and tightly tailored to the particular requirements of the WSNs.

The focus of this document is the incorporation of such capabilities into the EYES architecture. It is structured in four chapters: In Chapter 1 we discuss the main characteristics of the data-centric communication and computation in WSNs. This is followed with a short review of several existing frameworks that provide elements of data-centric

The second chapter explores in detail the properties of the publish/subscribe model. We give an emphasis to its content-based variant that is used as a basis for the EYES semantic architecture.

In Chapter 3 we start the discussion on the EYES approach with the requirements for a successful publish/subscribe based semantic addressing. After we identify some of the shortcomings in the current state-of-the-art, we proceed with the specifics of the EYES solution, concentrating on the proposed modifications and extensions.

Chapter 4 concludes the deliverable and outlines our future work schedule in the remaining duration of the Task 3.2 (Lookup Service).

Chapter 1

Data-centric communication

In the Introduction we hinted on the necessity of a middleware that will enable more rapid development of the typical WSN applications while improving the overall energy efficiency. In this chapter we start the search for the optimal solution to the problem by gaining some insight from several existing data-centric communication and computation frameworks. We evaluate their suitability as foundations for the data-centric EYES middleware by comparing their alignment with the specific WSN requirements.

1.1 Requirements

We classify the requirements over two axes: *functionality* and *performance*. For a data-centric middleware, functionality encompasses the expressive power of its naming and how well it supports the various data-centric application classes. These advanced features regularly come at a price in high complexity of the protocols and can lead to increased energy consumption. It is thus very important to also specify performance criteria that strike a balance between the required features and the energy-efficiency of the protocol implementation.

1.1.1 Functionality

The traditional communication middleware sits between the application and the networking layers. It is a level of abstraction intended as a shield from the intricacies of the networking architecture. It exposes to the application a set of higher-level communication primitives that are closer to the problem space the application is trying to address. These customized services significantly simplify the development process and increase the productivity of the programmer.

We want to achieve the same goals in our architecture. Because the WSNs are so tightly focused on extracting, processing and transferring data about the physical environment, our middleware has to be data-centric in nature. A top-down analysis that extracts some common properties of the envisioned WSNs applications can shed light on the concrete functional requirements:

- In most of the WSN applications the interest lies with the *data* and not with the Identification (ID) of the nodes that have produced it.
- Because the communication is the most energy-expensive operation there is strong incentive to perform *in-network processing* of the data, thus lowering the number of transmitted messages.

- The different in-network processing tasks rely on different *communication patterns*. One-to-many, many-to-one and many-to-many are very common communication patterns in WSNs applications.
- The applications operate in different *time coupling* modes. The long-term "sampling" applications produce higher levels of synchronization, while the event-based ones require a degree of temporal decoupling.
- The applications are not always interested in getting "exact" results. Most of the time, they can operate equally well with *estimates* of the real values.

A successful data-centric middleware for WSNs has to provide support for these properties. Most importantly, and as the name already suggests, it has to support a type of addressing that is independent from the network topology. The resulting *identity decoupling* is crucial in providing customized WSNs communication services. At the same time, it increases the energy efficiency by not requiring an explicit *name resolution* step that can be very costly in a multi-hop wireless network. The *semantic addressing* requires novel mechanisms to achieve the different types of many-to-many interactions arising in the various stages of in-network processing. Because of the changes in the addressing, the vast research in multicast for MANETs is not directly applicable. In addition to the different spatial distributions, the middleware has to support a spectra of patterns in the time domain: some of the interactions will require close request-response type of interaction, while others will operate in more loose, event-triggered mode.

1.1.2 Performance

The single most important design criterion in the WSNs is the energy-efficiency. The data-centric middleware influences the total energy dissipation in conflicting ways. On one side, the fulfilling of the functional requirements requires more complex communication protocols that can potentially lead to an increase in the energy use of the network. On the other side, by being closely adapted to the special WSNs conditions, they can also provide substantial energy savings in comparison with the more generic solutions. A good data-centric middleware has to reconcile this conflicting effects by ending with a *net-gain* in energy efficiency.

Another important performance characteristic in WSNs is the scalability of the solution. Some of the planned WSNs applications call for networks with thousands of nodes. Together with the limited storage on the nodes, this puts severe constraints on the amount of state the protocols can rely on. The protocols have to be designed in a way that minimizes the communication overhead of keeping the state synchronized. Owing to the broadcasting nature of the wireless channel, the protocols that are based on pure local (one-hop) interaction can achieve high levels of scalability. In the extreme, the scalability requirement may ask for unconventional communication techniques that are random and stateless in nature.

The exploitation of the accuracy/energy-efficiency trade-offs is another valuable tool for satisfying the performance requirements. Because it operates close to the application, the middleware is in a good position to judge on just where the balance point between these two is. The semantic nature of the communication simplifies the transfer of this knowledge to the communication components where the actual tradeoff can be exploited.

1.2 Data-centric frameworks

In the following we provide a short and non-exhaustive review of the most prominent data-centric frameworks that tackle some of the issues that we have mentioned in the previous section. Almost all of the presented models draw their roots outside the narrow WSNs field. Depending on their core ideas, they manage to satisfy the different requirements with different degrees of success. We try to identify their strong and week points and argue on their alignment with the Energy-efficient Sensor Network (EYES) objectives.

The publish/subscribe model also belongs to this group. Because we plan to use it as a basis of our semantic addressing infrastructure, we are examining this model more closely in a separate chapter (Chapter 2).

1.2.1 Overlay networks model

The increasing availability of high-speed connections at the core but also at the perimeter of the Internet has led to new applications with significantly different characteristics then the traditional client/server model. The so called Peer-to-Peer (P2P) file sharing, previously confined to the LAN environment, became possible on an Internet-wide scale. The limited support from the TCP/IP stack for these new applications was compensated with elaborate application level solutions [1, 2, 18], creating distributed *overlay networks* that run on top of the existing network infrastructure.

The P2P file sharing bears some similarities with the WSNs domain:

- The users are interested in the offered data, and not in the identity of the providers. In the case of the P2P file sharing application this is usually a file with a given name and of a given type. In the sensor networks space this is the sensed data or some other status information.
- The networks can be comprised of many nodes, so scalability and self-organization are required properties.
- The nodes in the network have more or less the same capabilities and act as end-systems and routers at the same time. Each node can be both a provider and a consumer of information.
- The responsibilities of the communication infrastructure are similar: it has to provide means for searching for the data (*lookup*) and then has to provide efficient means for distribution of that data to the interested parties.

Yet the degree of applicability of the solutions from the P2P field into the WSNs space is strongly determined by the actual implementation especially of the lookup and the routing component. The majority of the state-of-the-art solutions in this area are based on the concept of Distributed Hash Tables (DHTs) [34, 36, 40, 47]. In these systems, the required data (e.g. the files), are associated with a *key* that is usually created by hashing (e.g. hash of the filename). A small subset of the total key space is assigned to each node in the network for storage. The overlay network provides set of operations for searching, storing and retrieving data. The `lookup(key)` operation returns the underlying network ID (e.g. the IP address) of the node that stores the data corresponding to the requested key. This then allows the nodes to perform `get` and `put` operations based on the keys.

The main difference between the proposed solutions lies in the routing algorithm that routes the lookup from the source node to the node that most closely matches the associated key value. In a network with n nodes, most of them achieve average path lengths of $O(\log n)$ application-level hops with average $O(\log n)$ neighbors in the overlay address space.

In the above we can detect several properties that can pose as a limiting factors if one would like to build a data-centric framework for WSNs using the same mechanisms.

First of all, the key generation can not be directly applied to to the sensed data that is of interest in WSNs. Unlike the filenames that are strings, the sensed data is usually represented with real numbers. Directly hashing them into keys would result in a naming interface that is unsuitable for the WSNs applications. The problem arises from the nature of the matching. The lookup process performs *exact* matching of the key, thus only answering the equality relation (Is temperature $= 25$). It is impossible to specify a constraint based naming (e.g. Is temperature > 20) that is very frequently required in the WSN applications. One possible way around this limitation is to sacrifice the flexibility of the naming and create predetermined and fixed *categories* of data, and then create the key as a hash of the name of the category.

From performance point of view, a major weakness is the fact that the overlay is oblivious to the underlying network topology. While the DHT solutions provide good average number of hops in the overlay address space, this can translate into much larger number of hops on the networking level. Also, the potentially large number of neighbors in the overlay address space can translate to a large routing state at each node. In a failure prone and mobile sensor network keeping this state synchronized would require large number message exchanges between the neighbors.

Lately, the work on building *topology aware* overlay networks [22, 33, 43] shows promise in overcoming some of the identified limitations.

Using geographic routing is another way to increase the coupling between the overlay and the underlying wireless topology. The authors in [35] propose such a Geographic Hash Table (GHT) system as a basis for a Data Centric Storage (DCS) [37] for WSNs. While the DCS approach has the potential to lower the overhead in querying the network (by not requiring a flooding step), the expressiveness of the queries is still limited by the DHT properties, supporting at most category or event granularity.

1.2.2 Active networks and mobile agents model

In the active networks model, the users "inject" code in the nodes in order to customize the operation of the network. The high expressive power of the customization code and the fact that it can migrate into the network (in the form of mobile agents) enables the specification of arbitrarily complex in-network manipulation of the data.

This flexibility of the active networks approach can be put to a good use in the WSNs context:

- The possibility to modify or completely replace the software suite of a running WSN is desirable for several reasons. Many of the planned scenarios ask for a long term deployment of the network. During this time it is going to be necessary to implement fixes for the detected software bugs or introduce completely new services in order to support applications that were unforeseen at the time of the initial deployment.

- The mobile agents approach brings a customized processing closer to the data sources, cutting on the number of exchanged data messages. The code can be individually molded for each different application resulting in more efficient applications than using a more general purpose solutions. The efficiency can be further increased if the code is dynamically optimized during its roaming in the network.

But there are also several significant shortcomings that can hinder the active networks model when applied to the WSNs:

- While it offers almost unlimited power of expression, the complete burden of exploiting that flexibility is on the application programmer. One has to provide customized active code for each new application. The middleware as such does not provide much in the form of a general solutions that can be used for rapid development of different applications.

- Although the active code can cut the number of required messages for a given in-network data processing, this comes at a cost for the additional code-distribution messages.

- The mobile agents model can introduce tight temporal coupling between the agents. This complicates the correct operation in the face of link/node failures or other types of transient node unavailability.

- The introduction of active code in the network can increase the security risk, making the network more vulnerable to external and internal attacks.

The first problem can be overcome by providing additional APIs that wrap some of the standard and recurring operations: sensing, networking, mobility, etc. But even with these APIs there is still a trade-off between the programmer efficiency and the efficiency of the solution. The model is best suited for application scenarios that require complex distributed algorithms and dynamic in-network processing.

The authors in [5] present *SensorWare*, an active code architecture based on mobile Tool Command Language (TCL) scripts. These scripts define both the data processing and the replication/migration pattern of the mobile code. The effectiveness of the approach is demonstrated on a target tracking application that relies on complex relations between the nodes participating in the execution of this cooperative task. The use of a scripting language instead of a full-blown compiled language also simplifies the creation of a protected “sandbox” execution environment that can limit the unwanted effects of malicious or misbehaving code.

Another interesting attempt to tackle the problem with the introduced code-distribution overhead is the introduction of *virtual machines* on the nodes. These virtual machines support a customized WSN instructions set that allows quite dense representation of even very large programs.

In [24] a virtual machine for the TinyOS [17], *Maté*, is presented. The instruction set has separate instructions for the most frequently used WSN operations: reading a sensor, sending a message, turning the Light Emitting Diodes (LEDs) on an off, for example, take just a single instruction.

Similar to the work in [4], additional savings in the code-overhead is achieved by segmentation of the program into smaller building blocks or *capsules* and then implementing smart capsule caching on the nodes. Depending on the application dynamics, the caching of the capsules can significantly reduce the number of required messages. Nevertheless, the capsule dissemination approach in itself, can pose a serious problem in the current Maté implementation. Unlike SensorWare, the distribution of the capsules in Maté is preformed only via broadcasting. Without a proper Medium Access Control (MAC) support this can lead to significant congestion-inflicted capsule loss, and consequently to long settling time for the new program version.

1.2.3 Database model

Many of the WSN applications, or at least some of their parts, can be expressed in the form of queries:

- *What is the current average temperature in the building?*
- *What is the maximum humidity in Room 439?*
- *How many sensors in Room 943 detect movement?*
- *What is the average humidity in the areas with temperature higher then 25 degrees?*

This is quite similar to the way the users interact with the traditional databases. The realization is fueling a very active research direction that treats the sensor network as a database [11, 12, 20].

According to this model, each sensor reading represents a database *tuple*. The combination of these tuples creates a append-only, distributed relational table that can be operated on. The specification of the queries is usually performed using the standard Structured Query Language (SQL). There are some benefits from using this model:

- It provides an application-independent data-centric interface that has established itself as an indispensable tool for increasing the efficiency of the application programmer.
- This interface already provides support for selection, grouping and simple aggregation operations, very common tasks in the envisioned WSNs applications.
- It shields the programmer from the volatility of the network, letting him concentrate on the application requirements of the end-user.

As an illustration, the above queries can all be expressed using standard SQL constructs over a virtual *sensors* table [26]:

```
SELECT AVG(temperature)
FROM sensors

SELECT MAX(humidity)
FROM sensors
WHERE location = "Room␣439"

SELECT COUNT(movement)
FROM sensors
WHERE location = "Room␣943"

SELECT AVG(humidity), temperature
FROM sensors
WHERE temperature > 25
```

Yet, the introduction of a “common denominator” interface can be a limiting factor in the application specific WSNs:

- Not all of the in-network processing operations can be specified using the simple aggregation functions of the SQL.

- The lack of support for tasking (see Section 3.3.2) can limit the model's applicability in the WSNs with actuators.
- The SQL interface completely masks the possible trade-offs between the accuracy of the answer and the energy spent for executing the query.

Observed just like interface shortcomings, the above problems can be easily remedied with simple extensions of the standard SQL syntax [39]. But in the WSNs the interface can not be separated from the implementation of the supporting query execution back-end.

The simplest and naive way to support the illusion of the WSNs as a database is to send each sensor reading to a central processing node where they are collected in a relational table over which the queries are run. Because of the WSNs limitations, this will seldom be acceptable but for the most simple cases with few nodes. For a scalable and energy-efficient solution, the execution of the queries has to be pushed inside the network, closer to the data sources, where the filtering effect can have maximal impact in lowering the number of messages. How to perform this task optimally is in the core of the current research attempts in this area. Leaning on previous work in the areas of distributed databases, and the recent developments on operating over "data steams", attempts are being made to specify WSNs optimized in-network implementation of the main database operators [12, 45].

MADDEN et al. [26] discuss some of the peculiarities on performing the simple stateless aggregation operations in a WSNs setting. Their aggregation service for the TinyOS architecture runs on top of a tree, routed in the "base-station" that issues the queries. As the values propagate from the leaves back to the root node, they are aggregated according to the aggregation function in the query.

The authors classify the aggregates along several dimensions: *duplicate-sensitive* or *insensitive*; *summary* or *exemplary* in nature; *monotone* or *not-monotone* and with *distributed*, *algebraic* or *holistic*. These properties have direct impact on the achievable energy-savings. While the in-network calculation of the aggregates with distributed and algebraic state (MAX, MIN, COUNT, AVG) can bring significant energy savings, the same does not apply for the holistic aggregates (MEDIAN) that require the same number of messages as the centralized approach.

From the above, it is clear that the classical database problem of query-execution optimization is transformed into a *routing* problem. The overall success of the database model for WSNs will largely depend on the development of an efficient and adaptive tuple-routing mechanism that can support the basic database operators.

Chapter 2

The publish/subscribe model

In this chapter we turn our attention to the publish/subscribe model. After discussing its general properties we focus on the characteristics of one particular type of publish/subscribe systems, namely the Content-Based Publish/Subscribe (CBPS). This variant is the basis on which we build our semantic addressing architecture presented in the next chapter.

2.1 Introduction

The rapid growth of the distributed systems in the last decade has prompted the development of more flexible communication schemes that closely mirror the emerging dynamic and decoupled nature of the applications. The *publish/subscribe* is one such scheme that supports a form of loose interaction between the entities in a distributed system [31]. It is event-based in nature, that is very different from the *request/reply* behavior of the “classical” *point-to-point* and *synchronized* protocols.

Instead of specifying the identities of the communicating parties, in the publish/subscribe model, the parties specify the nature of the data/events that they are prepared to provide or consume. The abstract *brokering* service that sits between them coordinates the interaction, making sure that the matching data is delivered to the interested parties.

The receivers declare their interest in certain types of data by subscribing. This is done by sending a *subscribe* message that codifies the nature of the requested data. From this point on, all the produced data that is matching the subscription is delivered to the subscriber. This continues until the subscriber declares that he is no longer interested by unsubscribing with an *unsubscribe* message that codifies the nature of the data he does not want to receive anymore. The providers disseminate the data by *publishing* it in the form of notification messages. In addition, they can issue *advertisment* messages in which they declare the nature of their future notifications. These messages are then used to improve the brokering performance and to inform the subscribers about the availability of new information types.

2.2 General properties

The above communication scheme is well adjusted to the specific requirements of the large-scale distributed applications. Its scalability stems from the indirect and asynchronous type of the interaction between the parties. This interaction is characterized by decoupling along two dimensions: identity and time. In the following we discuss how these different decoupling types distinctly contribute to the overall properties of the publish/subscribe model.

2.2.1 Identity decoupling

The *identity decoupling* is the foundation on which the flexibility of the model rests. It represents the fact that the communication between the subscribers and the publishers is performed without knowing the identity of the other side. The whole communication is performed based on the characteristics of the provided or requested *data*. The publishers do not keep references to the subscribers and vice versa. Even the number of the parties participating in the interaction is not known. It can as easily be one-to-one, one-to-many, many-to-one or many-to-many interaction.

This *anonymous* interaction makes the applications built using the publish/subscribe model easily adaptive to the run-time changes in the network. New subscribers can be introduced in the network at any time, and they will start immediately receiving the notifications containing data matching their interest. Similarly, new publishers can be introduced in the system and the existing matching subscribers will start receiving their notifications without any additional readdressing.

In contrast, in the traditional distributed-systems, after the introduction of new services or after changes in the existing ones, the *name service* must be updated. To use the new service, the application has to first contact the name service in order to obtain the identity of the provider and then establish a direct communication. In the publish/subscribe model, as long as the subscribers and the publishers use the same data specification model, there is no need for a naming service, as the data is self-describing.

2.2.2 Time decoupling

In addition to being identity oblivious, the publish/subscribe is *time decoupled*, i.e. it does not require that the parties are actively participating in the interaction at the same time. This is to say that a subscribers can subscribe to a given event even when there are no available publishers, and start receiving them once a matching publisher comes on-line. It can also happen, that a subscriber is notified about some data only after the original publisher is disconnected. Similarly, a publisher can start publishing while the subscriber is disconnected, and the subscriber will start receiving the data once it comes on-line.

The same asynchronism is present inside the producers and the subscribers. The publications and the notifications are not usually performed in the main program flow of the application. They are *non-blocking* operations. The publisher is not blocked while producing events, and the subscribers are asynchronously notified when a matching publication is received.

The above decoupling of the production and the consummation of the information increases the scalability by reducing the required coordination between the communicating parties. This makes the publish/subscribe model a suitable framework for implementing large-scale asynchronous applications.

2.3 Classification

The specific implementations of the above general ideas can be grouped along several axes. For the purpose of this document, the classification along the nature of the broker system and along the different ways for specifying the interests are of highest importance. A more in-depth survey of the publish/subscribe model can be found in [10], for example.

2.3.1 Architecture

One possible classification of the publish/subscribe systems is depending on the architecture of the brokering system that performs the matching between the subscribers and the publishers. Three different architectures can be identified:

Centralized In this architecture, there is a single entity in the system that acts as a broker for all subscriptions and notifications. When providing data, the publishers first send the notifications to the broker. The broker then matches the publication with the previously received subscriptions and sends the matching parts to the subscribers. This approach is suitable for the applications that require high levels of reliability, data consistency and transactional support. On the other hand, the centralized architecture introduces a bottleneck in the system that can severely limit the data throughput. At the same time, the centralized broker is a single point of failure that makes the whole system vulnerable.

Distributed On the other extreme, in the distributed architecture, the functionality of the broker service is shared between all entities in the system. Using intelligent store and forward methods, the subscribers and the publishers interact directly while maintaining the appearance of anonymity and asynchrony. Consequently, there is no bottleneck that stands in the way of the scalability. The main shortcomings of this approach is the fact that it requires complex communication primitives that are more error prone that in the centralized approach. As a result, it is much harder to implement the higher level services discussed above.

Hybrid The hybrid approach tries to combine the characteristics of the both approaches. Instead of using a single dedicated broker system, the broker functionality is distributed among several networked broker servers. The subscribers and the publishers act as clients and associate themselves with a given broker. Each broker server acts as proxy for the subscriptions/notifications of the locally attached clients. The broker network performs a distributed algorithm for matching the various notifications and interests. Many of the properties of the hybrid architecture depend on the topology of this broker network that can be in the form of a tree, acyclic graph or completely peer-to-peer.

2.3.2 Expressiveness

Equally important is the classification of the publish/subscribe systems based on the *expressiveness* of their data description styles. This has profound influence on the flexibility of the system and limits the type of the applications that can be effectively supported. Currently, three different categories of publish/subscribe systems can be identified based on the type of the *naming* that is used when expressing the interests for the data:

Subject-based This group encompasses the representatives of the earliest publish/subscribe systems that defined subscriptions based on specific *subjects* or *topics*. It represents a publish/subscribe wrapper around the *group* concept in the multicast communications. Each subject defines one such group. The act of subscribing to a subject is the same as becoming a member to a multicast group, while the publishing an event on a given subject is equivalent to multicasting the event to all members of the group.

Although the tight integration enables straightforward implementation of the middleware (by direct mapping to the multicast support in the network layer), it also results in an inflexible and coarse naming style. The partitioning of the event space using the subject *keywords* has to be performed upfront, and the parties have to agree on this set before they can effectively communicate.

One useful extension is the introduction of *hierarchical* dependencies between the subjects. In this type of systems, subscribing to a node in the subject hierarchy tree also implies the subscription to the events published under the keywords in the subtree. The subject tree provides a mechanism for controlling the coarseness of the subscriptions and in that sense amortizes some of the negative properties, but it the nature of the partitioning remains static.

Type-based The type-based publish/subscribe naming casts the subjects approach in an Object Oriented (OO)-framework. It enables filtering of the notifications based on their *type*. The subject tree is here replaced by an *inheritance graph*. The act of subscribing to notifications of a given type means an automatic subscription the inherited notifications types.

While similar with the hierarchical subjects model, the type-based systems provide some additional conveniences on the programming side. Through a tight integration of the middleware and an OO-programming language, a compile time type checking can be performed, significantly simplifying the debugging of the applications.

Content-based This is the most flexible of the publish/subscribe schemes. It allows the specification of the interests in the form of predicates over the content of the notification message. Because of its expressiveness, we believe that it is a suitable basis for the EYES semantic addressing component. We examine its properties in the succeeding section.

2.4 Content-based publish/subscribe

Following the same approach as in Section 1.1, we discuss the properties of the Content-Based Publish/Subscribe (CBPS) model along the axes of functionality and performance. Assuming a distributed solution, the first is primarily determined by the richness of the event naming style, and the second by the subscription/event routing mechanism.

2.4.1 Content-based naming

In contrast to the static *channel* abstraction of the subject-based publish/subscribe, the interests in CBPS are defined as predicates over the whole content of the notification messages [8]. The subscription represents a *content-based address* that *filters-in* the data of interest, and *filters-out* the rest. This results in a very flexible and expressive naming that enables the subscribers to select the interesting notifications with an arbitrary level of detail.

Theoretically, the filters can be defined by any function that evaluates to *true* or *false*, when applied to the content of the notification message. In practical systems this freedom has to be somewhat constrained in order to facilitate the effective execution of the matching and the routing/forwarding tasks. One of the most widely used content-based naming subclass is the so called *name/value* or *attribute* system [7]. In this type of naming, the elementary subscriptions are defined as *conjunction* of simple attribute constraints in the form of (*attribute, value, operator*) tuples. The event notifications, on the other hand, are conjunction of (*attribute, value*) tuples.

The various parts of the attribute filter have the following meaning:

Attribute This part of the *content-based address* specifies the property of the notification that will be subjected to inspection during the filtering process. The usage of a preselected set of attributes does limit the expressiveness to a degree, but the complexity of the matching and forwarding tasks is largely simplified.

The attributes are usually typed. Most of the implementations line-up the attribute types to those available in the used programming language. In addition to the basic number and boolean types, almost all of the implementations support string attributes and provide means for extending in the form of User Defined Types (UDTs).

Value The value is selected from the allowed range for the given attribute type and has to be aligned with the operation part of the name, i.e it has to belong to its valid domain. Some of the operations also allow the use of values with *special* meaning like “ANY”, “ALL”, etc.

Operator The last part of the name tuple contains the binary predicate operator used for the filtering. Primarily, it is one of the common equality and ordering relations ($=$, $>$, $\geq$, $<$, $\leq$). For the more complex attributes, type-specific operators can be defined (e.g. *substring, prefix, suffix* operators for strings).

A notification $n = (attribute_n, value_n)$ *matches* a given attribute constraint $a = (attribute_a, value_a, operator_a)$ if and only if:

$$attribute_n = attribute_a \wedge operator_a(value_n, value_a) = \top$$

When a notification n matches a subscription s (i.e. $n \prec s$), it is also said that the subscription *covers* the notification.

Generally speaking [29], when a filter F is applied to an notification it evaluates to true or false: $F(n) \rightarrow \{\top, \bot\}$. The set of the matching notifications $N(F)$ is defined as $\{n|F(n) = \top\}$.

Given this, two filters F_1 and F_2 are *identical* ($F_1 \equiv F_2$) if and only if $N(F_1) = N(F_2)$. The filters are *overlapping* if $N(F_1) \cap N(F_2) \neq \emptyset$, otherwise they are *disjoint*.

The filter F_1 *covers* the filter F_2, written $F_1 \supseteq F_2$, if and only if $N(F_1) \supseteq N(F_2)$. The relation is transitive and ($F_1 \supseteq F_2 \wedge F_1 \subseteq F_2$) is equivalent ($F_1 \equiv F_2$).

As discussed in the rest of the section, these properties can effectively be used in the optimization of the routing algorithm.

2.4.2 Content-based routing

The brokering system has to guarantee the delivery of the relevant notifications to the interested parties. The naive approach would be to *flood* each notification in the broker network, and then locally apply the subscription filters to sort out the relevant notifications from the rest. While stateless, this leads to wasteful usage of the communication resources as the notifications are delivered to brokers whose clients are not interested.

A much better alternative is to perform *content-based routing* where each broker maintains a *routing table* whose entries associate *filters* and *destinations*. Because of its expressiveness, the CBPS has to be supported by a much more complex routing structure then the subject-based model. As there is no longer a direct mapping between the subscriptions and the *multicast group* concept, the implementation can not leverage on the prolific research in this area.

In the following we briefly discuss some of the possible schemes for performing the content routing in order of increasing complexity [28]:

Subscription flooding The reverse approach to the naive solution is to flood the subscriptions instead of the notifications. This ensures that each broker has *global* information about every subscription in the network. Having this information, the broker network can *guide* the subscriptions only to the interested clients. The result is a reduction in the number of messages that comes at the cost of substantial state in the brokers that severely limits the scalability of the system.

Identity-based Building on the above approach, one simple way to reduce the sizes of the routing tables is to make sure that only one entry is kept for identical filters [30]. This means that a subscription is not forwarded to the neighboring brokers if that was done for an identical one in the past, because they match the same notifications.

Covering-based The next step is to leverage the covering property in reducing the number of forwarded subscriptions [6, 7]. In this approach, the new subscriptions and unsubscriptions are not distributed to the neighbors if covering ones have already been distributed. Depending on the level of overlap this can noticeably increase the scalability of the system. A significant drawback is the need to resubmit some of the subscriptions when an unsubscription is issued for the covering subscription.

Merging-based Instead of passively examining the relation between the filters, the merge-based model [30] suggests a more active approach where a group of subscriptions are first *merged* in a single covering subscription, and then this single subscription is distributed to the neighbors. In addition to the problem with the unsubscriptions, the merge-routing also requires some heuristic about when and to what extent this merging of the subscriptions is performed.

The performance gains from the above optimizations are dependent both on the level of overlapping between the subscriptions as well as on their spatial distribution in the network. To maximize their effectiveness, they can be augmented with *advertisement-based* pruning [6]. Here the forwarding of the subscriptions can be further constrained only to the areas where the publishers have previously *advertised* the possibility of a matching data.

Chapter 3

EYES semantic addressing

This chapter outlines the foundations of the data-centric middleware for the EYES project. We begin the discussion by summarizing why we feel that the CBPS represents a good basis in this sense. Afterwords, we turn our attention on the necessary enhancements to the basic model so that it can match the WSN requirements stipulated in Section 1.1. Finlay, we conclude the chapter by identifying some of the remaining open issues and sketching out the future work plan.

3.1 Introduction

The WSNs are tightly coupled with the physical environment. Their main task is to gather information about the phenomena in their surrounding. The ultimate aim is to process and transmit the observed data efficiently to the interested parties. An important characteristic of the WSNs is their volatility. It is expected that the sensor nodes will frequently fail/recover or go in a sleep mode to preserve energy. One way to counter these issues is by over-provisioning the network with nodes (in relation to the minimum number of nodes required for the requested quality of information gathering), thus introducing increased redundancy. This approach poses some interesting challenges to the messaging sublayer in the network. A mechanism is needed that will provide an efficient level of indirection, masking this dynamics from the upper layers.

The ID-independent nature of the Content-Based Publish/Subscribe (CBPS) can successfully accomplish this role. The attribute-based naming provides enough expressiveness and can simplify the application development by providing to the programmer a high-level construct that is closer to the application domain than the classical socket service, for example.

Besides, the CBPS does not require an additional name resolution step, substantially reducing the required message traffic. The different WSNs traffic types: Data-dissemination (periodic or event-triggered); Control-information (neighborhood information, time synchronization, localization, management, service discovery) etc., will benefit in various degrees. The gains can be *maximized* if the message exchange:

- needs name resolution (so we save on the name resolution triangle).
- can be optimized if one has access to application level data (so we save by in-network processing)

The specific WSN requirement of exposing the accuracy/energy-efficiency trade-offs to the application programmer arises as a conflicting requirement with the above discussed “shielding” task. Nevertheless, using smart extensions to the CBPS naming, the most important control levers can be provided to the programmer without compromising the basic properties of the model.

To remain a feasible choice for the EYES data-centric middleware, the increased complexity of the CBPS naming has to be matched with more efficient implementation of the underlying protocol mechanics. The content-based routing requires the highest attention in this sense. In the design of the protocol we have to address, but also use for our benefit, the peculiar characteristics of the WSNs environment. In particular, we intend to use the broadcasting property of the wireless medium and heavily rely on the tight integration between the layers for increased efficiency.

After treating the related work, in the next sections we examine some of these issues more closely. The suggested extensions to the CBPS model are with different levels of maturity: for some of them we have completed functionality and performance evaluations, while for the others the evaluations are under way. Our future work schedule is laid out towards the end of the chapter.

3.2 Related work

Recently, there has been an increased research activity relating to the application of the publish/subscribe model in MANETs and WSNs. In the following we present a short review on some of the work in this area that is relevant for the EYES project:

- The *Directed Diffusion (DD)* protocol presented by INTANAGONWIWAT et al. [21] is the most prominent example of a new class of *data-centric routing* protocols for WSNs. It applies some of the CBPS ideas in the domain of sensor networks. In the following, we provide a brief description of the core protocol mechanism that are important for the deliberations in Section 3.4: [1]

  1. The *sinks* flood their subscriptions in the form of `INTEREST` messages with $(attribute, value, operation)$ semantics. The nodes establish *gradients* towards the originators of the `INTEREST` messages. Assuming symmetric links and no message loss, this will create gradients between each two neighbor nodes in the network.
  2. The matching *sources* publish `EXPLORATORY DATA` along these gradients. When the data reaches the subscriber, a `REINFORCEMENT` messages are sent that reinforce one/-couple of the paths based on some metric. The resulting tree can also be pruned using `NEGATIVE REINFORCEMENTS`.
  3. The *normal* data is sent *only* along these reinforced links.
  4. The routing structure is maintained by *periodic re-flooding* of the `INTEREST` and `EXPLORATORY DATA` messages.

  As discussed in Section 2.4.2, the main problem with this approach is the excessive control overhead. The network is overwhelmed by the flooding of the `INTERESTS` and `EXPLORATORY DATA`. When due to synchronization (and in the case of multiple subscribers) these phases coincide, the network becomes fully congested for long periods of time.

[1] The reader is referred to [15, 16, 38] for more background material

Noticing this, in a subsequent publication [14], the original authors suggest that the `EXPLORATORY DATA` phase should be skipped and that the normal data should be sent *directly* to the subscriber along a *single path*. For example, along the gradient to the neighbor that first sent the `INTEREST` message. This change can reduce the amount of control traffic, on the expense to the robustness of the algorithm. It strengthens the symmetrical channel assumption and it requires that flows are marked with a flow ID, thus losing the "pure local" nature of the algorithm. Additionally, it complicates the protocol in the case of multiple subscribers, as each source can send data only towards a single sink at a given point in time.

- MARATHE and HERMAN [27] present a publish/subscribe communications protocol for WSN concentrating on the routing aspects. Building on a previous proposal that uses the notification flooding approach, they suggest the use of the *interval routing* concept as a method for maintaining small routing tables at the nodes. The interval routing tree requires only one entry per neighbor in the routing table (the unique interval of node IDs reachable through this link). Although this increases the scalability of the protocol, it introduces some serious deficiencies from a reliability point of view. In this single-tree approach, the root node becomes a bottleneck that can limit the performance of the distribution network. At the same time, if this node fails, it will result in multiple disjoint networks rooted at his children. To overcome this shortcomings, the authors argue for keeping k separate interval routing trees, with roots that are uniformly distributed in the network.

- HUANG and GARCIA-MOLINA [19] also address the problem of building Publish/Subscribe Trees (PSTs) in the MANET environment. The emphasis in the paper is not on the routing state, but on the *efficiency* with which this tree can route the notifications to the interested subscribers. Assuming that the notification is injected from the root node, the authors evaluate different tree construction approaches that minimize the *cost*, i.e. the collective work necessary to deliver that notification. The main idea is to use the available subscription information in order to guide the tree building process so that the joint *overhead*[2] is minimized. Since the optimal solution to the problem requires exhaustive search trough all spanning trees of the connectivity graph, for practical applications a heuristic is needed that will yield comparative results. The authors propose a greedy distributed algorithm based exclusively on local information that attempts to reach this optimal solution. The presented simulation results confirm the advantage of using application level information in the tree creation phase over random selection of parents from the neighbors pool. Nonetheless, the solution is still burdened with the intrinsic deficiencies of the single-tree approach. The assumption that the notifications stem from the root is also problematic. For systems with many publishers this can necessitate either of tunneling the notifications to the root (using separate unicast routing) or creation of per-publisher trees.

- Instead of using tree structures, ANCEAUME et al. [3] propose a distributed algorithm for *diffusion* of the subscriptions and the notifications in a mobile network. Assuming that the underlying communication graph is a Directed Acyclic Graph (DAG), the authors propose that the publish/subscribe interaction is implemented using the classical *link reversal* concept. According to this approach, at a given time, just the node that has exclusively incoming edges (so called *sink*) has the privilege to perform tasks (like issuing or forwarding notifications). After

[2]additional work that a node has to perform on behalf of its children in the PST

this atomic action is executed, the direction of the incident edges towards this node are reversed, giving chance to the other nodes. While decentralized and modular, the protocols based on the edge reversal method can experience problems with slow convergence in real life settings. Furthermore, this approach does not leverage the existing application level information in optimizing the distribution process.

3.3 Functionality enhancements

In this section we examine more closely the proposed improvements in the functionality of the CBPS so that it can better match the individual requirements of the WSNs environment.

3.3.1 Tunable accuracy

The scant energy resources is an intrinsic property of WSNs. Operating under such conditions often requires making compromises between the desired level of performance and the available energy. Yet, *controlling* this compromise is proving to be a rather complicated task. While the application component has the most intimate knowledge about the needs of the end-user and consequently about the amount of performance penalty he/she is ready to pay in order to increase the energy-efficiency, the majority of the trade-offs happen on the lower components.

A good WSNs middleware has to be able to *expose* these trade-offs to the application programmer, while maintaining the coherence of the API. The standard (*attribute*, *value*, *operator*) naming, as the primary interface between the application and the publish/subscribe component, is lacking in this important aspect.

In order to address this issue we propose to extend the name by introducing a new *accuracy* parameter:

$$(attribute, value, operator, accuracy)$$

The resulting increased expressiveness of the CBPS address can be exploited in several different ways:

- In most basic terms, it can be used to control the amount of matching notifications that the subscriber is willing to receive. For example, the address:

  $$(\texttt{temperature}, 25, >, 50\%)$$

  can specify that the subscriber is ultimately interested to receive only 50% of the notifications that match the attribute constraint, i.e. that carry data with temperature higher then 20 degrees. This crude method enables direct application control over the number of transfered messages in the network. This simultaneously impacts both the accuracy of the result (lower number of input samples results in decreased accuracy) and the energy dissipation in the network (lower number of transfered messages lowers the energy dissipation).

- The accuracy constraint can be observed also as a method for *scoping* the effects of the CBPS subscription. In many of the WSN applications, the most relevant data resides in the proximity of the interested node. Based on previous information (e.g. from advertisements) the routing layer can limit the distribution of the subscription just to the region that is necessary for reaching the specified accuracy.

- The combination of the reduced target set, and the identity obliviousness of the publish/subscribe model, provides space for performing *load balancing*. The additional constraining of the accuracy parameter can be applied to different subsets of matching nodes, thus spreading the load in a more fair way between the “semantically” equivalent nodes.

The *probabilistic k-out-of-m* approach discussed in the Deliverable 3.2 (Information Services) can be one potential way of instantiating the interface. We are currently working on how to obtain comparable functionality in the case of structured (clustered) topology and more deterministic requirements.

3.3.2 Tasking support

As elaborated in the previous section, the *scoping* of the semantic multicast can be used as one method for trading between accuracy and energy efficiency in a WSN. On an even finer level, this compromise can be performed by directly controlling the data *sampling* process. Intuitively, a more frequent sampling of the phenomena results in higher accuracy, at the expense of more messages. By controlling the sampling rate, the application can directly influence this trade-off. Obviously, this approach is best suited to the scenarios that require long-lived, periodic sampling of data.

As it stands, the CBPS interface does not provide any means to the application to control this process more closely. When explaining the basic operation in Section 3.1, we said that for performance reasons, the publishers start issuing the notifications only after they are informed that there is at least one matching subscription in the system. This may be considered as a crude binary control of the sampling process: as long as there are no subscriptions, the nodes do not sample the environment and do not needlessly inject notifications. But in many WSN scenarios, significant gains in efficiency can be achieved if the application can directly control the nature of the notification generation.

The most straightforward way to implement this *tasking* capability is by using dedicated attributes into the CBPS address [21]. Instead of being used as constraints on the content of the incoming notifications, these attributes, and the associated values, act as simple commands. By combining several of them in one subscription message, more complex tasks can be specified. For example:

$$(\mathtt{temperature}, \mathtt{ANY}, =) \wedge (\mathtt{rate}, 2, =) \wedge (\mathtt{duration}, 60, =)$$

can be used to specify that the subscriber is interested in all temperature data generated in the next 60 s with a rate of 2 samples per second.

So far, we have treated the WSN as a passive distribution architecture for the data sampled from in-situ monitoring of the surroundings. Yet, this closeness to the environment puts the network in a good position also to *actively* influence it. This action is performed by *actuating* elements, while the sensor network provides the necessary feed-back control loop. For the same reasons as in the data distribution, the addressing of these *actuators* should be data-centric in nature. The above tasking-extended CBPS naming can be used as an appropriate interface for this scenarios too. Having a unified middleware both for monitoring and actuation will simplify the programming and will maximize the potential gains from the cooperative task execution between the actuators and the sensor network.

3.4 Scalability enhancements

In the Section 2.4.2 we have already stressed the shortcomings of using simple flooding as a routing mechanism for the CBPS. The reduction of the control overhead is crucial in obtaining an energy-efficient solution. To this end, besides the measures that can be taken on the publish/subscribe level (e.g. covering and merge routing), we can also make use of the characteristic of the wireless medium to further reduce the number of unnecessary control messages.

In the wireless networks, the significant overlap between the neighborhoods of two nodes in immediate vicinity, can lead to a large number of unnecessary rebroadcasts. This additionally results in increased channel contention and waste of bandwidth that take further toll on the scarce energy resources of the nodes. The problem of identifying a set of re-forwarding nodes that still guarantees the distribution of the messages to all of the nodes in the network is very well researched. The proposed techniques vary from probability-based approaches to area- and neighborhood-knowledge methods [44].

The neighborhood-knowledge based group mainly comprises the distributed algorithms for efficient approximation of the minimum connected dominating set of nodes. They are based on the partitioning of the network into logical substructures called *clusters* that confine and mask local interactions from the rest of the network. In the process of defining these substructures (*clustering*) a single node (*clusterhead)* can be selected from every cluster to exert local control and coordination. The interfacing between the different clusters is then realized only by dedicated *gateway* nodes. The selection of the clusterheads and the gateways is such as to guarantee the connectivity of the resulting, reduced topology.

Clustering reduces the flooding overhead by limiting the re-forwarding of the messages to this reduced topology. Yet, for the approach to be overall effective, the cost of building and maintaining the clustered structure has to be lower then the energy savings from reducing the number of sent and received messages.

This requirement is not easily met if clustering is performed for the sole reason of limiting the flooding overhead, especially if one has in mind the pro-active and resource intensive nature of the majority of clustering algorithms. To overcome this problem, Kwon and Gerla [23] proposed the Passive Clustering (PC) algorithm for on-demand creation and maintenance of the clustered substrate.

3.4.1 Passive Clustering

Passive clustering has several unique properties that increase its viability as a flooding overhead control mechanism for on-demand wireless networking protocols. In the following, we briefly summarize these characteristics.

Unlike in “classical” clustering algorithms, the formation of clusters here is dynamic and is initiated by the first data message to be flooded. In this way, the potentially long initial set-up period is avoided, and the benefits of the reduction of the forwarding set can be felt after a very small number of data message rounds.

Because the main function of the clusters is to optimize the exchange of flooded messages, there is no point in wasting valuable resources to pro-actively maintain such an elaborate structure between floods, when there is no traffic that can make use of it. Consequently, passive clustering refrains from using explicit control messages to support its functionality and all protocol-specific information is piggybacked on the exchanged data messages. This approach joins the frequency of exchanged messages with the quality of the clustered substrate resulting in a graceful trade-off between the “freshness” of the clustered substrate and the introduced overhead for its maintenance.

An additional trade-off is also made over the issue of selection of a clusterhead between several potential candidates. Passive clustering does not have the benefit of exchanging specific control messages to optimally resolve this conflict, as it is usually done in other clustering schemes. It introduces a simple novel rule, called *first declaration wins*. Under this rule, the first aspirer to become a clusterhead is immediately and without further neighborhood checks declared as such and allowed to dominate the radio coverage area. This may at first sound suboptimal, but is more then compensated for by the decreased latency, the prevention of "chain re-clusterings" and the natural correlation that emerges between the traffic flow patterns and the resulting clustered topology.

The effectives of any clustering algorithm as a tool for reducing the number of redundant flood messages directly depends on its ability to select the minimal number of gateways while still maintaining the connected property of the topology. In order to avoid the large overhead of collecting full two-hop neighborhood information that is required for optimal solution of this clustering subproblem, passive clustering resorts to a counter-based gateway and distributed gateway selection heuristic. Under this heuristic, the probability of a node that belongs to two or more clusters to become a gateway is directly proportional to the number of clusterheads in its radio range and inversely proportional to the number of gateways already present in the same area. By controlling the coefficients of proportionality, one can make trade-offs between the flooding overhead reduction and the level of connectivity in the reduced topology.

As demonstrated by the authors in [46], the combination of these design decisions results in a lightweight and flexible algorithm that can significantly lower the overhead associated with flooding in wireless networks.

3.4.2 Directed Diffusion and Passive Clustering

To investigate the applicability of the passive clustering as a method for reducing the control overhead in content-based publish/subscribe systems, we performed a detailed performance evaluation taking Directed Diffusion as an example. The two protocols share some common properties:

1. They are on-demand mechanisms that maintain the operation only while there is application traffic in need of their services
2. They rely on purely local information for performing their functions

As a result, their combination is perfectly adapted to the operating conditions that are common in WSNs.

For the initial exploration, we decided to limit the interaction between them just to a single point in the operation of the original Directed Diffusion:

Ordinary (non-clusterhead, non-gateway) nodes do not forward the interest and exploratory data messages that they receive.

To explain the effects of this interaction, we demonstrate in Fig. 3.1 the establishment of the Directed Diffusion routing structures when used in combination with passive clustering.

Let the operation start with the topology depicted in Fig. 3.1(a). In passive clustering, nodes that are first able to piggyback information onto an outgoing message will declare themselves as clusterheads (assuming that they have not heard of other clusterheads before). Nodes that have heard of two clusterheads but of no other gateway assigned to these same clusterheads will become a gateway, resulting in the clustered topology shown in Fig. 3.1(b). Assuming this constellation, the next diffusion

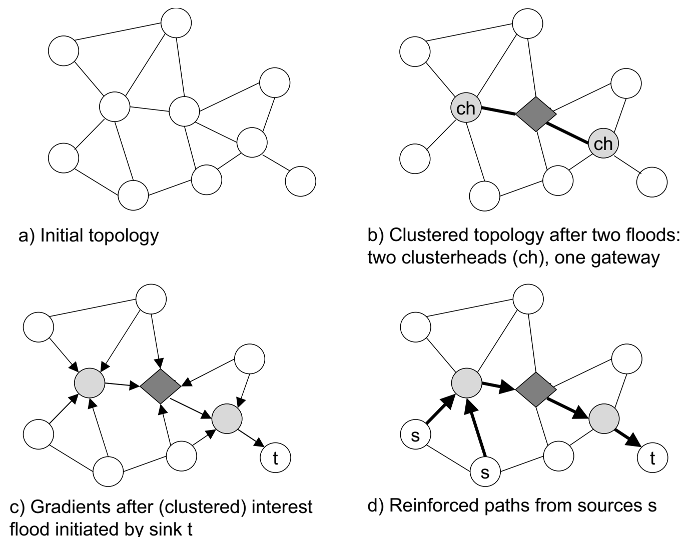
ch
ch
a) Initial topology
b) Clustered topology after two floods: two clusterheads (ch), one gateway
t
s
s
t
c) Gradients after (clustered) interest flood initiated by sink t
d) Reinforced paths from sources s

Figure 3.1: Establishing the clustered diffusion structure (gray circles indicate clusterheads, diamonds indicate gateways)

interest flood will establish a sparse gradient structure shown in Fig. 3.1(c). The reinforced data paths are then shown in Fig. 3.1(d).

Based on this description, we can make a couple of observations:

1. The number of redundant transmissions during the broadcasting tasks is significantly reduced.
2. The size of all Directed Diffusion messages that are broadcasted is increased by the size of the Passive Clustering header.
3. Passive Clustering reduces the level of connectivity available to Directed Diffusion.

This means that the overall effectiveness of our approach depends on the answers to the following crucial questions:

- Does the combination of the reduced number of messages with the increased message size result in net gain in energy efficiency?
- Does the reduced level of connectivity result in degradation of the data distribution capability?
- Does the reduced level of connectivity result in increase of the latency as sometimes less optimal paths have to be followed?

3.4.3 Effects of the network topology

We start the presentation of our results with the five sources/five sinks case in networks of either fixed area or fixed density. Figure 3.2 depicts the average dissipated energy per unique event for the original Directed Diffusion protocol and for the Passive Clustering augmented Directed Diffusion

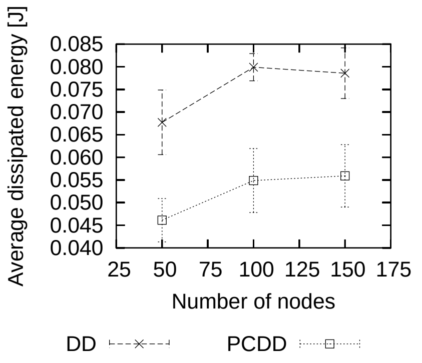

Figure 3.2: Energy efficiency – fixed density scenario

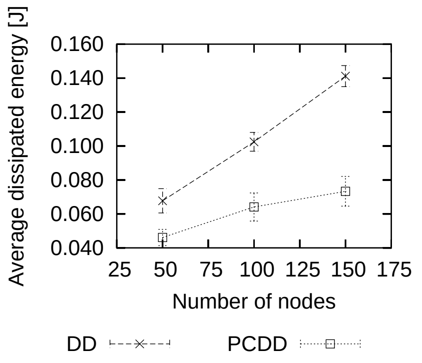

Figure 3.3: Energy efficiency – fixed area scenario

protocol, under three different network sizes. For each different network size, the simulation area is scaled so that the average node degree in the network remains constant at around 10. This enables us to examine the scaling properties of the protocols as a function of the number of the nodes in the network in isolation. In all of the topology experiments the load of the network is generated by five sources sending unique events every two seconds.

The results clearly show the effectiveness of Passive Clustering in controlling the unnecessary re-forwarding of messages in Directed Diffusion. While both protocols show slow degradation of their performance with the increase in number of nodes, the relative savings of the passive-clustering augmented directed diffusion over the original protocol is evident even for the 50 nodes scenario and is maintained as the size of the network increases. This is a confirmation to our assumption that the reduced number of flooded messages results in a net gain in the energy metric, despite the increased size of the remaining messages.

As described in Sect. 3.4.1, both Directed Diffusion and Passive Clustering are based on pure local interaction. Many of the elementary operations of which they are comprised operate over the set of neighboring nodes that are in radio range. As a result, their complexity, and by that, their performance is strongly determined by the size of that set, i.e., on the average node degree in the network. Again, we have examined our performance metrics for three different network sizes as previously, but now keeping the simulation area constant to the initial 160 m by 160 m. This gives us an insight into the performance of the evaluated protocols under an average number of around 10, 20, and 30 neighbors.

In Fig. 3.3 we can see the average dissipated energy per unique event for an increasing number of nodes in this fixed area. The original Directed Diffusion shows rather linear increase in the required energy with the increase of the number of neighbors. This was to be expected, as the increased number of neighbors results in higher number of flooded interest messages and an increase in the number of unicasted exploratory data messages. Passive Clustering augmented Directed Diffusion, on the other hand, behaves much better in dense networks. Like in the fixed degree case, the increased overhead due to the PC headers is more then compensated by the reduction of the costly send and receive operations. The denser the network, the larger the advantage over pure Directed Diffusion as more redundant paths can be suppressed by passive clustering.

Moreover, these advantages in energy efficiency are not paid for in a reduced delivery rate. As Fig. 3.4 shows, the contrary is true: the delivery rate actually is considerably better with passive clustering. Compared with the fixed degree scenario, the delivery ratio of plane Directed Diffusion

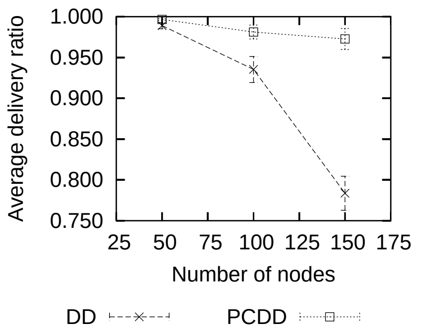

Figure 3.4: Delivery ratio – fixed area scenario

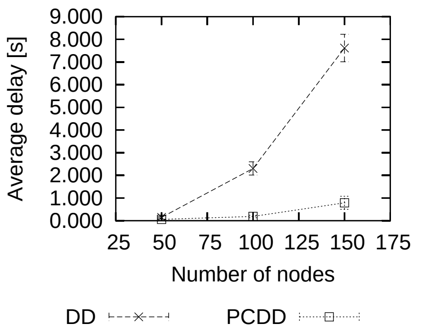

Figure 3.5: Delay – fixed area scenario

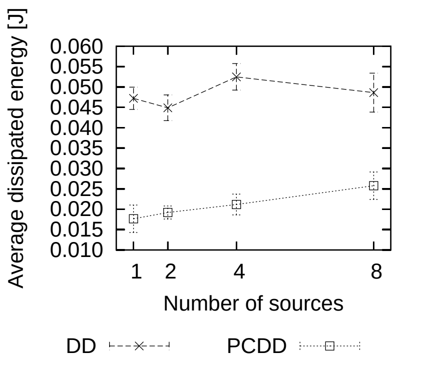

Figure 3.6: Energy efficiency – single sink scenario

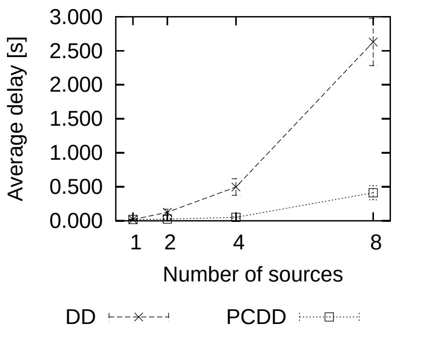

Figure 3.7: Delay – single sink scenario

experiences significant degradation as the size of the neighborhood increases. By reducing the superfluous message exchanges, passive clustering leads to a decrease in the collision rate that results in a very good delivery ratio even for the highest density case.

Similarly, Fig. 3.5 shows that also the delay is much better with passive clustering, despite the potential problem of the longer paths. The reduced load in the network more than compensates for this potentially negative aspect. Without this reduction in the amount of traffic in the network, plain Directed Diffusion faces extensive delays in the higher degree scenarios.

Overall, we can conclude that Passive Clustering augmented Directed Diffusion scales much better with respect to the neighborhood size, maintaining satisfying performance in the face of the increasingly harsh network environment.

3.4.4 Effects of the traffic pattern

But what happens when the number of sinks or sources is varied?

As one could expect, the relative gain of the passive clustering enhancements compared to plain Directed Diffusion increases with the number of sources: More sources issue more interest floods, which in turn benefit from passive clustering. Our energy metric shown on Fig. 3.6 somewhat masks these effects because the average dissipated energy *per event* tends to drop as the number of generated events due to the multiple sources increases. The effects of the increased load and the performance

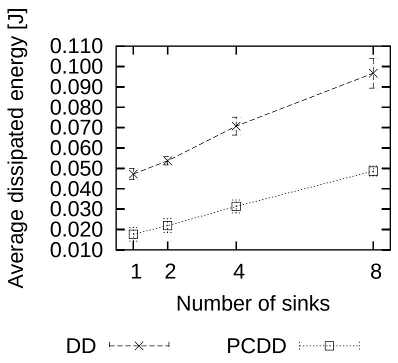

Figure 3.8: Energy efficiency – single source scenario

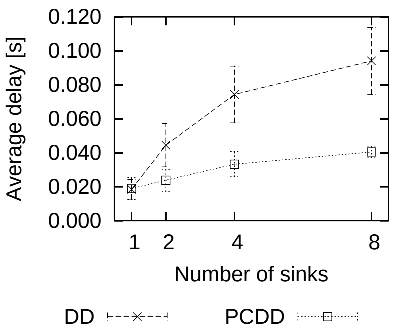

Figure 3.9: Delay – single source scenario

gains from the passive clustering are much more visible on the delay plot shown on Fig. 3.7.

A similar pattern becomes apparent when the number of sinks is increased: We fixed one single source and increased the number of sinks. Here the number of reported data is not increased, but each data message needs to be reported to more areas of the network. This, combined with the increase in the “interests” traffic, leads to an increase in the energy consumption as shown in Fig. 3.8. The delay (Fig. 3.9) also rises with the number of sinks, but not as dramatically as with the number of sources.

3.5 Higher level services

Owing to the nature of the wireless medium and to the limited hardware resources on the nodes, the WSNs experience both node and communication failures that lead to loss and reordering of messages. Naturally, this also affects the functioning of the data-centric middleware. As the different applications can have a wide range of reliability requirements, the middleware service should remain *best-effort* in nature. Depending on the application needs, it is left to the higher level components to implement advanced services as: reliable delivery, exactly once delivery, temporal order delivery, etc. Although the CBPS routing is not directly mappable to the multicast concept, the extended research in *reliable multicast* can be used as a starting point in their implementation. Additional services can also be inserted between the middleware and the application because of performance reasons. The publish/subscribe model is an appropriate base for many of the common distributed services [25].

The *service discovery* is one such functionality that is indispensable for the internal functioning of the WSNs and for interfacing with the outside Internet Protocol (IP)-world. One may argue on the need of a separate *service discovery* component in a network with a CBPS middleware [25]. From a functionality point of view, the naming part (Section 2.4.1) provides support for almost all of the standard service discovery tasks. Nevertheless, spreading new subscriptions to all nodes each time one wants to learn about the available services may not be the optimal solution. If these requests are going to be frequent enough, it makes sense to collect the necessary service description information for easy future access. A natural and energy-efficient way of approaching the problem is to make use of information that is already present in the network.

The *advertisements* are important tool for routing optimization in the CBPS systems. They carry information about the potential data that the publishers can offer. This knowledge is then used to limit the propagation of the subscription messages only to those areas of the network that have the potential to satisfy the filter. But the same knowledge can also be exploited to build a distributed repository of node capabilities that forms the basis for quite energy-efficient *service discovery* functionality. We believe that this cross-level solution will prove itself as superior in comparison with the "classical" distributed service discovery architectures [9, 13, 42]. The results from the functionality and the performance evaluations of the proposed approach are going to be presented in the deliverable D3.4 (Service Discovery).

3.6 Future work

We can draw several conclusions from the results in Section 3.4.3 and 3.4.4. First, the simulations confirm that flooding is one of the major energy consumers in this type of content-based routing. Second, they show that when the flooding of the subscription and the notification messages is limited using Passive Clustering, it results in significant energy savings. At the same time, the decreased load leads to a better delivery ratio and lower delay, as fewer messages are lost due to collisions.

While the Passive Clustering in the form presented here is very beneficial, it can introduce hot spots in the network, namely the gateways between two clusterheads and the clusterheads themselves. Their energy expenditure is higher than those of ordinary nodes. How to deal with this asymmetry between nodes on top of an *intentionally* passive clustering is an open problem. We intend to compare the PC approach with the active clustering algorithms developed in the EYES framework under this perspective.

The routing structure in the traditional distributed publish/subscribe models are largely oblivious to the underlying network topology. Because of their tight integration with the physical environment, we expect that the notifications in the WSNs are going to exhibit high level of locality. This physical locality is transformed into locality in the wireless topology, raising the issue of the degree of alignment between it and the publish/subscribe distribution structure [32]. This is the same problem that overlay networks are facing (Section 1.2.1), but the penalty for the mismatch is substantially greater in WSNs then in internets.

We propose to address the issue by influencing the creation of the routing structure using a joint "physical" and "semantic" cost function. This cost function can be added to the existing heuristic for clusterhead/gateway selection presented in Section 3.4.1. As a result, we will promote the creation of gateways (and by that of routing paths) between the clusters in physical range that have the highest degree of overlap in the subscriptions and the advertisements. Apart from the expected decrease in the delay, the existence of these routes will maximize the gains from the "covering" and the "merging" operations (Section 2.4.2), thus reducing the amount of traffic in the network and increasing its operational lifetime.

The flexibility and the robustness of the network will also depend on its structure. The majority of proposed approaches in the literature are based on trees routed in the subscriber or publisher, depending on the expected ratio between subscriptions and publications. The issue of reusing the existing structure when adding new subscriptions is seldom tackled. Recent publications in the ad hoc networks field, show that the mesh structure is superior (in terms of reliability) to the shared trees approach, at the cost of increased overhead [41]. Whether this is true in the case of sensor networks under the publish/subscribe communication pattern is still unclear and warrants careful deliberation.

Another open issue is the trade-off between building the distribution network completely on-demand and relying just on local (one-hop) information, thus resulting in suboptimal solutions, and the introduction of limited active collection of “semantic” information in the extended neighborhood, that can potentially result in distribution network with much better filtering capabilities.

In the remaining duration of the Task 3.2 (Lookup services) we will continue our research on the suitability of an advertisement-based service discovery protocol. In parallel, we will carry out performance evaluation on the EYES extensions for which this was not already done.

Our work is burdened by the fact that some of the extensions reflect themselves in different gains in the visible metrics depending on the network and traffic characteristics. Owing to the limited deployment of WSNs today, there is a lack of established models (probability distributions, parameters) of the subscription and the notification generation process. Even more elusive are the models that define the amount of locality, overlapping and the general redundancy of the traffic in a typical EYES sensor network.

Based on the evaluation, suitable candidates will be chosen for implementation in the final testbed. The results from the correctness and performance tests will be the subject of the deliverable D3.7 (Software implementation of Information Services and Lookup Service).

Chapter 4

Conclusion

The data-centric communication and computation is a natural fit to the characteristics of the WSNs. They are intimately connected with the environment and one of their main tasks is to provide sensed data reflecting the occurring phenomena in this environment. The effective programming of the future distributed data-centric applications requires introduction of new abstractions that are adapted to the specifics of the WSNs architecture. Founding these programming abstractions on the current ID-centric communication protocols can lead to sub-optimal solutions from energy-efficiency point of view. The middleware supporting this type of WSN applications has to be optimized both for programmer and communication efficiency.

As elaborated in Chapter 2, we believe that the publish/subscribe model has characteristics making it a suitable framework for large class of data-centric operations. The Content-Based Publish/-Subscribe (CBPS) model in particular (Section 2.4), enables high flexibility in the subscription specification, acting as a distributed filter for selecting and streaming data to the interested parties.

From a functionality point of view, the CBPS is flexible enough to support the majority of the needed data-centric applications. But this flexibility comes at a price in increased complexity of the underlying protocol mechanics. Consequently, it requires customization before it can be successfully applied in the WSNs space where the energy-efficiency is the single most significant design criterion.

In the EYES framework we have proposed and tested several extensions that try to increase the functionality and the performance of the CBPS, molding its characteristics to the specific requirements of the WSNs.

The proposed *accuracy* parameter (Section 3.3) exposes some of the possible accuracy/energy trade-offs on the communication level to the application programmer. This scoping of the “semantic multicast” can also be used as a load balancing measure as it provides space for fair sharing of the load between the semantically equivalent nodes in the network. Furthermore, the tasking support (Section 3.3.2) enables even finer application control over the notification generation process. The introduction of attributes with a *command* function is also necessary as an unifying interface towards the WSNs with actuating elements.

The simulation studies, presented in Section 3.4, prove that the excessive control overhead is the main deficiency of the CBPS model that can limit its scalability and by that its usefulness in the WSN context. The successful application of the Passive Clustering (PC) approach in solving this issue gives us confidence that at the end, we can emerge with a net gain in energy efficiency terms.

In parallel with the improvements to the basic CBPS model, we have started to explore how we can use this framework to implement even higher level services that some applications require. The design of an advertisements-based *service discovery* functionality is the first step in this direction and will be the subject of the next deliverable in the Task 3.2.

Bibliography

[1] Gnutella Home Page. http://gnutella.wego.com.

[2] Napster Home Page. http://www.napster.com.

[3] Emmanuelle Anceaume, Ajoy K. Datta, Maria Gradinariu, and Gwendal Simon. Publish/subscribe scheme for mobile networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 74–81. ACM Press, 2002.

[4] Cristian Borcea, Chalermek Intanagonwiwat, Akhiles Saxena, and Liviu Iftode. Self-Routing in Pervasive Computing Environments Using Smart Messages. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, page 87. IEEE Computer Society, 2003.

[5] Athanassios Boulis and Mani B. Srivastava. A Framework for Efficient and Programmable Sensor Networks. In *Proceedings of the 5th International Conference on Open Architectures and Network Programming (OPENARCH 2002)*, pages 117–128, June 2002.

[6] Antonio Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks*. PhD thesis, Politecnico di Milano, Milano, Italy, December 1998.

[7] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, 19(3): 332–383, 2001.

[8] Antonio Carzaniga and Alexander L. Wolf. Content-based Networking: A New Communication Infrastructure. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, number 2538 in Lecture Notes in Computer Science, pages 59–68, Scottsdale, Arizona, October 2001. Springer-Verlag.

[9] Microsoft Corporation. Universal Plug and Play Device Architecture. http://www.upnp.org, June 2000.

[10] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[11] Wai Fu Fung, David Sun, and Johannes Gehrke. COUGAR: the network is the database. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 621–621. ACM Press, 2002.

[12] Ramesh Govindan, Joseph M. Hellerstein, Wei Hong, Sam Madden, Michael Franklin, and Scott Shenker. The Sensor Network as a Database. Technical Report 02-771, USC/Information Sciences Institute, September 2002.

[13] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2, 1999.

[14] John Heidemann, Fabio Silva, and Deborah Estrin. Matching Data Dissemination Algorithms to Application Requirements. Technical Report ISI-TR-571, USC/Information Sciences Institute, April 2003.

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

[16] John Heidemann, Fabio Silva, Yan Yu, Deborah Estrin, and Padmaparma Haldar. Diffusion Filters as a Flexible Architecture for Event Notification in Wireless Sensor Networks. Technical Report ISI-TR-556, USC/Information Sciences Institute, April 2002.

[17] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104. ACM Press, 2000.

[18] Yang hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast (keynote address). In *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12. ACM Press, 2000.

[19] Yongqiang Huang and Hector Garcia-Molina. Publish/Subscribe Tree Construction in Wireless Ad-Hoc Networks. In *Proceedings of the 4th International Conference on Mobile Data Management*, pages 122–140. Springer-Verlag, 2003.

[20] Tomasz Imielinski and Samir Goel. DataSpace-querying and monitoring deeply networked collections in physical space. In *Proceedings of the ACM international workshop on Data engineering for wireless and mobile access*, pages 44–51. ACM Press, 1999.

[21] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking (TON)*, 11(1):2–16, 2003.

[22] Minseok Kwon and Sonia Fahmy. Topology-aware overlay networks for group communication. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 127–136. ACM Press, 2002.

[23] Taek Jin Kwon and Mario Gerla. Efficient flooding with Passive Clustering (PC) in ad hoc networks. *ACM SIGCOMM Computer Communication Review*, 32(1):44–56, 2002.

[24] Philip Levis and David Culler. Maté: a tiny virtual machine for sensor networks. In *Tenth international conference on architectural support for programming languages and operating systems*

on Proceedings of the 10th international conference on architectural support for programming languages and operating systems (ASPLOS-X), pages 85–95. ACM Press, 2002.

[25] Alvin Lim. Distributed Services for Information Dissemination in Self-Organizing Sensor Networks. *Journal of Franklin Institute, Special Issue on Distributed Sensor Networks for Real-Time Systems with Adaptive Reconfiguration*, 338:707–727, 2001.

[26] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36 (SI):131–146, 2002.

[27] Virendra Marathe and Ted Herman. Design of Interval Routing Enabled Publish/Subscribe Communication Protocol for Ad Hoc Sensor Networks. In *Midwest Society for Programming Languages and Systems*, 2002.

[28] G. Mühl, L. Fiege, F.C. Gartner, and A. Buchmann. Evaluating advanced routing algorithms for content-based publish/subscribe systems. In *Proceedings. 10th IEEE International Symposium onModeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, pages 167–176. IEEE, 2002.

[29] Gero Mühl. Generic Constraints for Content-Based Publish/Subscribe Systems. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS '01)*, volume 2172 of *LNCS*, pages 211–225, Trento, Italy, 2001. Springer-Verlag.

[30] Gero Mühl, Ludger Fiege, and Alejandro P. Buchmann. Filter Similarities in Content-Based Publish/Subscribe Systems. In H. Schmeck, T. Ungerer, and L. Wolf, editors, *International Conference on Architecture of Computing Systems (ARCS)*, volume 2299 of *Lecture Notes in Computer Science*, pages 224–238, Karlsruhe, Germany, 2002. Springer-Verlag.

[31] Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen. The Information Bus: an architecture for extensible distributed systems. In *Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 58–68. ACM Press, 1993.

[32] Lukasz Opyrchal, Mark Astley, Joshua Auerbach, Guruduth Banavar, Robert Strom, and Daniel Sturman. Exploiting IP multicast in content-based publish-subscribe systems. In *IFIP/ACM International Conference on Distributed systems platforms*, pages 185–207. Springer-Verlag New York, Inc., 2000.

[33] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2002*, volume 3, pages 1190–1199. IEEE, 2002.

[34] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[35] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: a geographic hash table for data-centric storage. In *Proceedings of the first ACM international workshop on Wireless sensor networks and applications*, pages 78–87. ACM Press, 2002.

[36] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[37] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, and Deborah Estrin. Data-centric storage in sensornets. *ACM SIGCOMM Computer Communication Review*, 33(1):137–142, 2003.

[38] Fabio Silva, John Heidemann, and Ramesh Govindan. Network Routing Application Programmer's Interface (API) and Walk Through 9.0.1. Technical report, USC/ISI, December 2002.

[39] Chavalit Srisathapornphat, Chaiporn Jaikaeo, and Chien-Chung Shen. Sensor Information Networking Architecture. In *Proceedings of the 2000 International Workshop on Parallel Processing*, page 23. IEEE Computer Society, 2000.

[40] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.

[41] Lee Sung-Ju, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, volume 2, pages 565–574. IEEE, march 2000.

[42] Jim Waldo. The Jini architecture for network-centric computing. *Communications of the ACM*, 42(7):76–82, 1999.

[43] Marcel Waldvogel and Roberto Rinaldi. Efficient topology-aware overlay network. *ACM SIGCOMM Computer Communication Review*, 33(1):101–106, 2003.

[44] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the third ACM international symposium on Mobile ad hoc networking & computing*, pages 194–205. ACM Press, 2002.

[45] Yong Yao and Johannes Gehrke. Query Processing for Sensor Networks. In *First Biennial Conference on Innovative Data Systems Research(CIDR 2003)*, January 2003.

[46] Yunjung Yi and Mario Gerla. Scalable AODV with efficient flooding based on on-demand clustering. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(3):98–99, 2002.

[47] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: a fault-tolerant wide-area application infrastructure. *ACM SIGCOMM Computer Communication Review*, 32(1):81–81, 2002.

Acronyms

API	Application Programming Interface
CBPS	Content-Based Publish/Subscribe
DAG	Directed Acyclic Graph
DCR	Data Centric Routing
DCS	Data Centric Storage
DD	Directed Diffusion
DHT	Distributed Hash Table
EYES	Energy-efficient Sensor Network
GHT	Geographic Hash Table
ID	Identification
IP	Internet Protocol
IST	Information Society Technologies
LAN	Local Area Network
LED	Light Emitting Diode
MAC	Medium Access Control
MANET	Mobile Ad-Hoc Network
OO	Object Oriented
P2P	Peer-to-Peer
PCDD	Passive Clustering augmented Directed Diffusion
PC	Passive Clustering
PST	Publish/Subscribe Tree
SQL	Structured Query Language

SQTL	Sensor Query and Tasking Language
TCL	Tool Command Language
TCP/IP	Transmission Control Protocol/Internet Protocol
UDT	User Defined Type
WSN	Wireless Sensor Network