Chapter 1

Introduction

In recent time, the study of wireless sensor networks (WSNs) has become a rapidly developing research area. A WSN is a network of small sensing devices, called *motes*, which can communicate over a wireless channel.

The development of motes was a straightforward step after different technologies became available at very low cost: Small sensors, embedded micro-controllers, transceivers for wireless communications, and small power sources. Simply integrating these technologies into a single embedded device opens up a completely new field of applications. This lead to the "Smart Dust" project [KKP99], where the vision of WSNs was initially proposed. Consider millions of tiny devices, at a size comparable to dust particles, and at a negligable cost of individual devices. One could deploy such a network simply by throwing the nodes from planes or vehicles, for example in hazardous areas or places that are inaccessible to humans. The nodes would then run algorithms for self-organization, build a stable networking infrastructure by themselves, and eventually start to survey the area. They would react to changes in their sensor readings and compare findings with nearby nodes to distinguish actual phenomena in the environment from local misreadings. They would find base stations that collect the network's status reports and alarm messages, so that information about the surveyed area would eventually be available to whomever deployed the network. For an example, see Figure 1.1, which sketches a sensor network deployed all over a forest, which measures temperature, watches for forest fires and relays an alarm to firefighters. Because sensor nodes are located close to the fire, it is detected much earlier than what is possible by classical observation methods, e.g., satellites, manned watch towers, and alike. The network continues to work and provides live reports during the firefight operation, and may even help locating or guiding lost people.

As visions go, it was soon discovered that this one was at least slightly overenthusiastic. This has many reasons, two big issues being:

- Tiny batteries hardly exist with sufficient capacity.
- Developing, programming, and installing software for millions of small devices is cumbersome and requires completely new approaches.

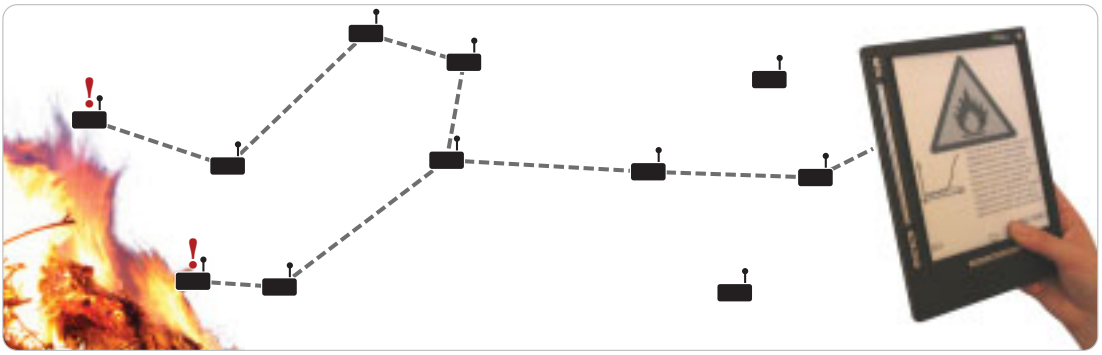


Figure 1.1: WSN observing a fire. *The two left-most nodes detect a fire in their vicinity. An alarm message is then routed over relay nodes to a base station.*

Over the last few years, research and development made great steps towards real-world sensor networks. Nowadays we see WSNs in practical applications in many different fields—from surveying glaciers to medical care. And while current devices are still large and quite costly, looking at the simpler yet similar RFID chips shows what is possible in the not-too-distant future. When RFID chips are already so small that they are indistinguishable from dust to the human eye, one can envision how small motes can become once they exist as highly integrated circuitry and are produced in large numbers.

With WSN research gaining popularity, it became evident that there were many algorithmic questions that needed to be addressed, especially because WSNs require a completely new kind of algorithms that did not exist previously. In the classic, centralized setting, an algorithm runs on a single processor, and has access to all problem data that exists at any point in time[1]. This makes it possible to use a unifying theory about algorithms, computability, problem complexity, and so on. A step towards WSNs is *parallel computing*, where a number of parallel processors jointly solve a problem. They still have shared access to some memory. *Distributed computing* takes this even further, assuming that each processor has only private memory, so nodes have to communicate to solve a problem together. With WSNs, several additional properties enter the stage:

- There is *geometry*—the network is placed in 2- or 3-dimensional space, and sensor values are only meaningful together with their location.
- Communication cannot happen between arbitrary nodes, as it is only possible when they are spatially close.
- Individual nodes may fail at any time. The loss of some nodes participating in

[1] this includes online algorithms, which merely cannot access *future* data.

a computation should not endanger the correct outcome of the overall algorithm.

- Each node is heavily limited, both in terms of processor speed and memory size. This comes from the design goal to have tiny motes at very low cost.
- Each node runs on a non-rechargeable tiny energy source, so heavy computation and communication is infeasible.

Therefore, there are many new problems associated with WSNs that require *distributed*, *geometry-aware*, and *energy-efficient* algorithms. This results in completely new (and exciting) paradigms for algorithmic research.

Our Vision: One of the basic algorithmic issues in WSNs is to let nodes know where they are. It is apparent that sensor readings are of little value, unless it is known where they were recorded. Furthermore, there is geometry in the network, so knowledge about it can be leveraged in many higher-level protocols and services, like multi-hop routing between nodes, tracking and addressing observed objects, generating a map showing hazard levels, and so on.

The first approaches to solve the localization problem were straightforward: Researchers added localization devices to the motes, for example GPS receivers. It quickly turned out that such receivers are quite costly and are not easily miniaturized. The second approach was to attach GPS to just a few nodes and let the other nodes "compute" their position from the known position of these so-called *anchor nodes*. Taking this to the next step, algorithms were developed that tried to assign positions to all nodes without using any external information at all. Unfortunately, it is now known that practically all variants of the localization problem are NP-hard, even in a classic, centralized setting. In distributed algorithms, there is the additional challenge that two nodes may get close positions by a localization algorithm, but are far away in reality—the localization is folded. Because they cannot communicate, they cannot detect this misplacement. They can only check that communication neighbors are indeed placed close to them, but this does not prevent folding at all. Using such position information can lead to all kinds of bad situations, e.g., a packet that arrives at a node seemingly close to the intended destination, yet far away in reality. Even worse, it may not be a packet but rather a firefighter, finally reaching the area where he incorrectly assumes the fire source to be.

There is one question that motivated most of our research for this thesis:

Why use Euclidean coordinates as localization information?

We tried to find an answer by looking at previous research and talking to engineers and computer scientists. We got many answers, but almost all of them boiled down to the following two points:

- Coordinates are naturally there—in simulations, visualizations, formulas, etc.
- It's been like that ever since motes were equipped with GPS devices.

Surprisingly, nobody could give us a reason why using coordinates as localization information would be better than alternative approaches. One frequently used argument was that coordinates allow to communicate an event's position to humans, e.g., marking the fire source on a map. This argument has two flaws: First, this presentation happens outside the network, through a full-fledged computer with a real processor and lots of memory. In this setting, it seems sub-optimal to put the computational burden of positioning onto the already crippled motes. Second, even if coordinates are required in-network one application, why should the network also use them for all other tasks that involve location?

We believe that Euclidean coordinates are a particularly poor choice of localization information. If a node knows it is at position 52°29'27"N 13°17'28"E, it can hardly use that information for anything useful. The numbers tells it nothing about the size of the network, its structure, the connectivity, the node's role in the network, or anything else. Even if it knows there is a base station waiting for reports at 52°26'39"N 13°21'31"E, it still has no clue how to relay a message towards it. This is a surprisingly hard task in coordinate-based WSNs, and a lot of research was necessary to provide actually working routing schemes. In the end, to get a picture of the whole network and its own position within it, a node needs to sample a lot of coordinates, which requires much communication, energy, and is counterproductive in the global goal of memory- and energy-efficiency.

Eventually, we came to the conclusion that the following question needs to be investigated further:

> *If we cannot (or don't want to) use global coordinates as localization information, how can we establish knowledge about the network's topology, and how can we use such knowledge to benefit the network's operation?*

This thesis describes most of the results stemming from this question. We focussed on scenarios where the network topology is complicated, with many holes and a complex shape of the area; this is where coordinate-based localization performs worst. We found a means to build *clusters*, that is, groups of nodes that claim to be a functional unit in the network, and construct a small geometric graph that precisely describes the network area. Every node knows to which cluster it belongs, and provable properties of the cluster decomposition make it possible to establish network services, say, routing with guaranteed delivery, at virtually no cost.

Organization of this Work: Chapter 2 describes the necessary fundamentals for the remainder of the thesis. This work uses bits and pieces from many disciplines,

so we focus on advanced topics, and assume the reader to be familiar with basic topics in mathematics and computer science.

Chapter 3 presents an algorithm to compute the boundary of a WSN without using coordinates; this is an important prerequisite for our clustering scheme. We also describe some competing algorithms both from ourselves and from the literature, and compare the different approaches by simulations.

Our topological clustering scheme is introduced in Chapter 4. We prove several beneficial properties of the clusters in the continuous case, and present algorithms for the discrete case. Furthermore, we describe how the clusters can be used for higher-level WSN services such as routing.

In Chapter 5, we introduce a novel algorithmic problem, the *Energy-Constrained Dynamic Flow* problem. While this was originally just another way to enhance clusters by computing characteristic properties, it became a solid piece of self-contained work.

Finally, Chapter 6 gives a short introduction into Shawn, a free WSN simulation software that we developed to run our algorithms in, and which is the only available software that serves the needs of algorithmic work in WSNs.

The work presented in this dissertation was not done in isolation. **Sándor Fekete** contributed to almost all aspects of this thesis. The chapter on dynamic flows (Chapter 5) is joint work with **Ekkehard Köhler** and **Alexander Hall**. The network simulator Shawn (see Chapter 6) came to live thanks to a fruitful collaboration with **Dennis Pfisterer** and contributions of many students, most notably **Tobias Baumgartner**.

Chapter 2

Basics

This chapter introduces fundamental concepts that are needed in later chapters. First there is an overview on today's motes and actual applications where they are used. Then we summarize basic results from selected topics in mathematics. We restrict this to just what we actually need in later chapters; it is not intended to be an complete introduction into the field. The third part of this chapter comprises models and basic properties related to sensor network theory, together with a short treatise on WSN localization.

2.1 Current Technology

In the eight years since the SmartDust project [KKP99] initially proposed the development of tiny motes, WSNs have left the field of pure academic research and are nowadays put into practice in a great variety of commercial applications.

2.1.1 Sensor Node Hardware

There are many ways to construct a mote. However, there is a basic set of components that are always present. They define the application space of WSNs, as well as the constraints under which theoretical research in this field has to operate.

Sensors: The purpose of most WSNs is sensing the environment, so there are sensors on the motes. They are the primary data source for the network. Actually, many WSN application are just that: Every node collects some data, which is then forwarded to a dedicated data sink, either directly or over relay nodes. A great variety of sensors is used in actual networks; they measure light, temperature, humidity, or acceleration, some motes even record audio. See the applications overview in Section 2.1.2 for more details.

Processor: By definition, a sensor node is an active computation device. It has a processor that is used for local computations on the data the node currently pos-

sesses, and to run communication protocols. Current motes feature anything from embedded micro-controllers running at 5 MHz, up to full-fledged CPUs, e.g., Intel's XScale series in the iMote 2, where the processor's speed can be adjusted between 13 and 400 MHz.

Memory: Motes have different kinds of memory. They often feature a small amount of RAM, usually between 5 and 256 kBytes as well as dedicated program storage. Some have flash-based memory to store large amounts of sensor data in, ranging up to 1 MByte in the iMote 2.

Communication: By definition, WSN motes can communicate with each other over a wireless channel. Currently, they often use standard communication technologies like WiFi/802.11, Bluetooth/802.15.1, or ZigBee/802.15.4. Some devices use alternatives such as sound or infrared light. Future devices will likely use different protocols, because none of the above are tailored for real low-energy communication with small protocol overhead.

Battery: The vision of sensor networks involves that each device has a small energy source that powers it for a while. When the battery is empty, the device simply dies, thereby removing itself from the network. The mote will not be recharged, it will just stay that way. Current motes usually use standard batteries. Because of the price, they are almost always collected after use, recharged, and reused.

Over the last years, several motes were developed for different kinds of requirements. Figures 2.1(a) to 2.1(d) show five different platforms that are currently in use. More devices, as well as detailed specifications, comparisons, and documentation can be found at the Sensor Network Museum[1]. It is noteworthy that none of these devices is highly integrated. All of them consist of off-the-shelf components. This adds to cost, size, and energy consumption: They usually cost around $50–100, meaning that not many organizations can afford a million-node network. They are large enough to strap two or three standard AA or AAA batteries to them, and they drain these batteries within a few days when no sleep-duty-cycle scheme is used.

Figure 2.1(d) shows a Spec mote. It is a highly integrated chip providing all circuitry that is needed for a sensor node. It was claimed that this chip can be produced in large numbers for less than $1. We are convinced that integrating single-chip motes, further miniaturization efforts, and time will yield much smaller and cheaper motes. Consider the μ-chip, see Figure 2.1(f). It is a complete RFID chip (shown here without antenna). It is much simpler than a sensor node: It is powered by the energy induced from a reading device communicating with it, and

[1] `http://www.btnode.ethz.ch/Projects/SensorNetworkMuseum`

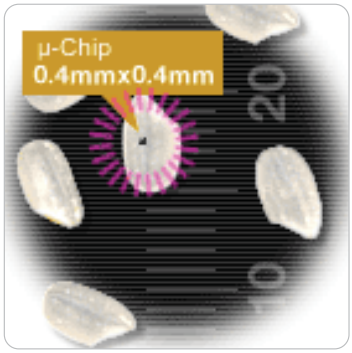
(a) MICA2 Mote, Crossbow Technology Inc.

(b) Intel Mote (iMote) 2, Intel Corp.

(c) MSB430, ScatterWeb GmbH

(d) Tmote Mini, Moteiv Corp.

(e) Spec mote next to a ball-point pen, UC Berkeley



(f) RFID μ-Chip (black square) on a grain of rice, Hitachi Ltd.

Figure 2.1: Five sensor node platforms, and an RFID chip. *Image sources: (a,b) www.xbow.com; (c) cst.mi.fu-berlin.de/projects/ScatterWeb; (d) www.moteiv.com; (e) www.jlhlabs.com/jhill_cs/spec; (f) www.hitachi.co.jp/Prod/mu-chip* .

all it can do is send back a 128-bit ID number stored in ROM. It does perform this task in such a small chip that it is not perceivable by the human eye though.

2.1.2 Sensor Network Applications

One of the envisioned "killer applications" for sensor networks is a system that can be used for catastrophe recovery, e.g., in a forest fire, in a contaminated area, after an earthquake, and so on. The idea is that a sensor network is distributed in the area, so there is no time for pre-configuration or placement of nodes in carefully chosen positions. Once spread out, the network automatically organizes itself, establishes a mode of communication, obtains location knowledge and provides supportive data to the disaster response team. It monitors for hazards, locates and tracks helpless people, finds safe routes to them, and guides people back out.