Cooperative Event Detection in Wireless Sensor Networks

Georg Wittenburg

INRIA
LIX, École Polytechnique
Route de Saclay, 91128 Palaiseau, France
georg.wittenburg@inria.fr

Norman Dziengel, Stephan Adler, Zakaria Kasmi, Marco Ziegert, and Jochen Schiller

Department of Mathematics and Computer Science
Freie Universität Berlin
Takustraße 9, 14195 Berlin, Germany
[dziengel,adler,kasmi,ziegert,schiller]@inf.fu-berlin.de

Abstract

Event detection in wireless sensor networks is a sophisticated method for processing sampled data directly on the sensor nodes, thereby reducing the need for multi-hop communication with the base station of the network. In contrast to application-agnostic compression or aggregation techniques, event detection pushes application-level knowledge into the network. In-network event detection – especially the distributed form involving multiple sensor nodes – has thus an exceptional potential to increase energy efficiency, thus prolonging the lifetime of the network.

In this paper, we summarize recently proposed system architectures and algorithms employed for event detection in wireless sensor networks. On the example of the AVS-Extrem platform, we illustrate how energy-efficient event detection can be implemented through a combination of custom hardware design and distributed event detection algorithms. We then continue to present a brief evaluation of the detection accuracy and the energy consumption that is achievable by current systems.

1 Introduction

In order to construct highly energy-efficient Wireless Sensor Networks (WSNs), it is paramount to push application-level data processing as deeply into the network as possible. If sampled data is processed and evaluated close to its source, communication with the base station of the network is reduced, energy consumption is minimized, and the lifetime of the network is thus extended. Event detection in WSNs follows this strategy by programming the sensor nodes in such a way that the occurrence of deployment-specific, semantic events can be established directly on the sensor nodes, relying only on the locally available data from the sensors. This transformation of data from the physical sensory domain into the scenario-specific application domain stands in contrast to other techniques for in-network data processing, e.g., compression and aggregation, which are agnostic to the application-level properties of the data. Fundamentally, event detection trades in the generality of these latter approaches in favor of higher data reduction rates and thus higher energy efficiency.

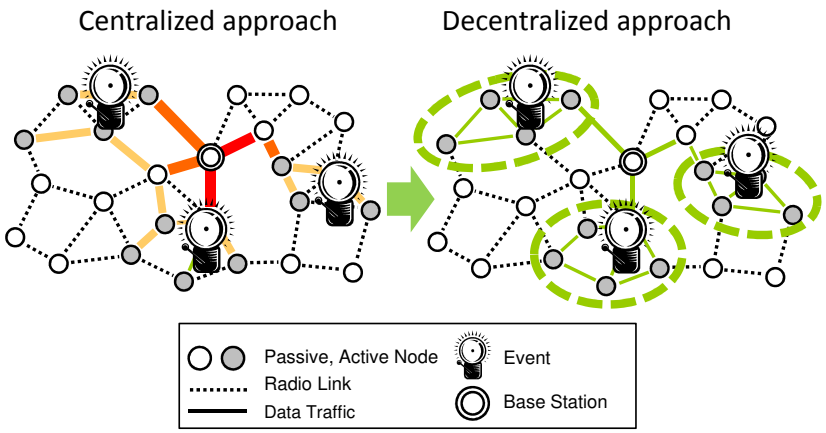


Figure 1: Centralized vs. decentralized event detection

Event detection in WSNs is commonly used for tasks such as fire or hazard detection [1], vehicle tracking [2], area surveillance [13], undersea monitoring [10], or the classification of human motion sequences [15]. Depending on the application, a variety of sensors can be employed, including light and temperature sensors, accelerometers, microphones, or cameras. The raw data gathered by these sensors is – depending on the sampling rate – too voluminous to be transmitted to the base station of the WSN. Instead, one or multiple sensor nodes evaluate the raw data locally, seeking application-specific patterns, e.g., a temperature reading exceeding a threshold value (for a fire detection system), a transient change in a video image (for area surveillance), or the occurrence of a specific motion path (when monitoring human movement). Only the result of this localized evaluation is then sent to the base station, and thus, as illustrated in Figure 1, reduces the required network traffic substantially.

In general, designers of a WSN-based event detection system need to make two major architectural choices: First, the question arises in how far the sensor nodes should cooperate during the detection process, and second, how the semantically relevant information is to be extracted from the sampled raw data. The first of these questions involves trade-offs related to the communication architecture, while the second one deals with algorithmic issues during data processing. The overall goals when tackling these questions are, of course, to minimize communication overhead (and thus energy expenditure) and to maximize the detection accuracy for a variety of use cases.

In this paper, we summarize and contextualize our findings in this field of research based on our previous work on the *AVS-Extrem* platform for cooperative event detection in WSNs, which we built, refined, and evaluated in multiple lab experiments [4, 5] and deployments [14, 13]. We begin with qualitative comparisons of possible communication architectures and algorithms for data processing in Sections 2 and 3 respectively, as part of which we also review the state of the art. We then illustrate the typical design choices that arise when building a deployment-ready event detection system in Section 4. In Section 5, we summarize experimental results from several deployments of the AVS-Extrem platform, and finally conclude and motivate future work in Section 6.

2 Architectures

As data processing and event detection play a key role in WSNs, several different approaches have been developed and deployed to process and to transport data and events within a WSN.

2.1 Local Detection

The most basic approach is a local event detection. In this approach, each node gathers data from its local sensors and employs local algorithms (cf. Sec. 3) to decide whether a specific event has occurred. Data of neighboring nodes is not taken into account and all signal processing is performed on the local node itself. If an event is detected, each node signals the results directly to the base station of the sensor network.

This approach is very easy to implement, but incurs a non-negligible communication overhead and is only applicable to fairly simple types of events. An exemplary implementation can be found in the fence surveillance system proposed by Kim et al. [8].

2.2 Centralized Evaluation

Centralized event detection is widely used in current real-world deployments. All nodes send either raw or preprocessed sensory data directly to the base station, which has significant computational and energy resources. The data is interpreted exclusively on the base station – individual sensor nodes have no knowledge about the semantics of the collected data.

Although this method has several advantages, e.g., good detection accuracy resulting from the global knowledge available at the base station, the network does not scale well and the continuous data stream quickly depletes the available energy. Furthermore, the dependency on the base station node introduces a single point of failure, thus rendering this architecture unsuitable for security-relevant applications.

Event detection using a centralized evaluation architecture has been implemented, amongst many others, by Gu et al. [7] for vehicle tracking.

2.3 Decentralized Evaluation

Decentralized evaluation is one way of mitigating the aforementioned problems of the centralized approach. In this approach, the network is clustered into smaller subnetworks which operate autonomously. Sensory data is gathered, exchanged, and processed within each of these clusters. One node in the cluster – the so-called *cluster head* – takes on the task of communicating with the base station when an event is detected or raw sensor data needs to be transmitted. The cluster head is selected at deployment time and may have additional computational and energy resources. Furthermore, cluster heads may also be equipped with a separate network interface to communicate directly with the base station, e.g., via the cellular phone network.

The main advantage of this concept is its reliability since the network remains mostly operational even if individual subnetworks fail. Another advantage is the possibility to save energy: As only the nodes in one cluster need to exchange and process data when an event occurs, nodes in other clusters can remain in a low-power state. The limitations of this approach become apparent in ad hoc scenarios, as the network is impaired by the initial, event-agnostic selection of cluster heads.

Event detection employing decentralized evaluation has been used in large-scale deployments by Duarte and Hu [2] for vehicle classification.

2.4 Distributed Evaluation

In a distributed evaluation, each node processes data on its own and can take different roles during the event detection process. If several nodes detect an event, they communicate with each other and decide autonomously, i.e., without the support of a base station or cluster head, which type of event has occurred. The decision about the event type is made by a distributed algorithm running on each node: Nodes wake up as a result of an event, sample and process the sensory data, and afterwards exchange the results with other nodes that also woke up. It is not predefined, but rather established during the distributed detection process, which node reports the detected event to the base station. Depending on the application, it is possible that all nodes in the network remain in a low-energy state and only wake up upon being exposed to an event. If a node of the network fails to operate, the network itself remains operational and only a minor drop in detection accuracy is to be expected.

The major advantages of this architecture are its reliability, robustness and the possibility for substan-

Table 1: Trade-offs of architectural approaches to event detection in WSNs

Architecture	**Energy Efficiency**	**Detection Accuracy**	**Exemplary Implementations**
Local detection	Medium, since nodes process data locally and no communication is necessary. However, a single event may be reported to the base station multiple times by different nodes.	Only applicable to applications with simple events, e.g., those detectable by monitoring data for exceeded threshold values.	Fence surveillance (Kim et al. [8])
Centralized evaluation	Low, since a continuous data stream is transmitted from all nodes to the base station.	High, since all data is available at the base station and a variety of complex algorithms can easily be applied.	Detection, tracking and classification of vehicles (Gu et al. [7])
Decentralized evaluation	Medium, since this is a combination between the previous two approaches.	High, since all data is available on each cluster head.	Distributed vehicle classification (Duarte and Hu [2])
Distributed evaluation	High, since data is only exchanged locally between nodes and events are only reported once to the base station.	High, since distributed versions of complex algorithms can be employed.	Event detection (Martincic and Schwiebert [9]), Fence surveillance (Wittenburg et al. [13])

tial energy savings. A drawback is the system complexity that results from its reliance on ad hoc reconfigurations. In particular, the autonomous evaluation needs to be very reliable as the nodes change roles and the user has no access to the sensor data and cannot easily verify the correctness of the distributed evaluation.

A system using fully distributed evaluation architecture has been proposed and simulated by Martincic and Schwiebert [9], and implemented and deployed as part of our own work on fence monitoring (cf. Sec. 4).

The trade-offs between these architectures as well as exemplary implementations are summarized in Table 1. As WSNs pose challenges in energy optimization and therefore need to communicate very sparingly, not every architecture is suitable for the particular requirements in this domain. Specifically, the distributed evaluation approach seems very promising for WSNs that operate on an unknown topology and require extensive energy savings.

3 Algorithmic Approaches

Expanding upon the architectural view presented in the previous section, we now shift our focus from the networking aspects to the data processing aspects of event detection in WSNs. Several alternatives to analyzing the collected raw data in order to establish whether a particular event has occurred have been proposed in the literature. The overall trade-off in this area consists of – as we will explore in this section – weighting resource consumption against the complexity of the events that the system is able to detect. Another interesting aspect to consider is whether a system is capable of learning events on its own, or whether external expert knowledge needs to be embedded into the detection algorithm.

3.1 Threshold-based Decisions

Event detection based on comparisons against threshold values is the most basic approach to event detection WSNs. Obviously, it is applicable to all

Table 2: Trade-offs of algorithmic approaches to event detection in WSNs

Algorithm	**Computational Overhead**	**Applicability**	**Exemplary Implementations**
Threshold-based decisions	Low, since simple monitoring of raw data is sufficient.	Only detects simple events, e.g, "It's very hot, hence there must be a fire."	Vehicle tracking (Gu et al. [7]), fence surveillance (Kim et al. [8])
Pattern recognition	High, since approaches of this type involve complex feature extraction and classification.	Applicable to a variety of complex events, depending on platform support for feature extraction and training.	Vehicle classification (Duarte and Hu [2]), human motions (Ghasemzadeh et al. [6]), fence surveillance (Wittenburg et al. [13])
Anomaly detection	Varies depending on the complexity of the events during normal operation.	Differentiates between known and unknown, i.e, anomalous, patterns.	Light tracking (Wälchli [11])

kinds of sensors that produce a continuous sampling range, e.g., temperature sensors, magnetometers, motion sensors, microphones. Threshold values, which correspond in their dimensionality to the number of sensed values, are usually established by a domain expert and set either at deployment or at run time. Dynamically adjusting threshold values depending on the observed events is possible, however rarely implemented in practice.

The key advantage of this approach is its simplicity. Comparisons against threshold values can be implemented directly on the sensor nodes that sample the data. Decentralized or distributed detection architectures, as discussed in the previous section, are thus not directly applicable to this detection scheme.

Event detection based on threshold values has been implemented by Gu et al. [7] as part of the VigilNet project to track vehicles, and by Kim et al. [8] as part of their fence surveillance system.

3.2 Pattern Recognition

Pattern recognition is the process of classifying a set of data samples into one of multiple classes. Various techniques for pattern recognition have been employed in different WSN deployments, and we are merely able to cover the fundamentals in this section. The process of pattern recognition is typically subdivided into four steps: sampling, segmentation, feature extraction, and classification. During *sampling*, raw data related to an event is gathered by a sensor and optionally preprocessed, e.g., for noise reduction. *Segmentation* employs thresholds to detect beginning and end of an event. *Feature extraction* computes a set of highly descriptive features from the data and combines them into a feature vector. Features may comprise data such as minimum, maximum, average, histogram, and/or discrete Fourier transform, and their extraction from the raw data reduces the dimensionality while preserving characteristic information. The feature vector may combine features from multiple different sensors built into the sensor node; each feature is extracted separately from the respective raw data. During *classification*, feature vectors are evaluated, either through statistics or with a priori knowledge from a preceding training session. Algorithms for classification include, among many others, decision trees, neural networks, support vector machines, and k-nearest neighbors. Since these algorithms operate on the feature vector, different numbers and types of sensors are supported transparently. A detailed introduction into these techniques can be found in Duda et al. [3].

Most of these algorithms require significant computational resources and thus need to be modified heavily in order to run on sensor nodes. Furthermore, approaches to event detection that employ pattern recognition techniques usually require

a scenario-specific training to initialize the classifier. Pre-configured threshold values can be used to simplify the initialization, but tend to reduce the event detection accuracy.

Event detection employing pattern recognition techniques has been used in large-scale deployments by Duarte and Hu [2] for vehicle classification, by Ghasemzadeh et al. [6] to detect specific human movement patterns, and by our own system for fence monitoring (cf. Sec. 4).

3.3 Anomaly Detection

Anomaly detection is a term used for approaches that focus on the specific case of detecting whether particularly unusual event has occurred. This is achieved by learning typical system behavior over time and then classifying specific events as either normal or anomalous. Approaches with this goal expand upon the methods of the previous two approaches and incorporate techniques from the fields of intrusion detection and bio-inspired artificial immune systems.

One example of a system that specifically detects anomalies is the Distributed Event Localization and Tracking Algorithm (DELTA) proposed by Wälchli [11] which uses light sensors to track a moving flashlight.

Our discussion of algorithmic approaches is summarized in Table 2. The crucial properties of event detection algorithms are the computational overhead (in terms of processing and memory usage) and the scenario-dependent applicability. The computational overhead obviously increases as the algorithms become more complex. The applicability describes in which circumstances an acceptable accuracy can be expected from an algorithm. Since applicability varies depending on the deployment scenario, there is not a single preferable algorithmic approach to event detection in general.

4 An Exemplary Event Detection System: AVS-Extrem

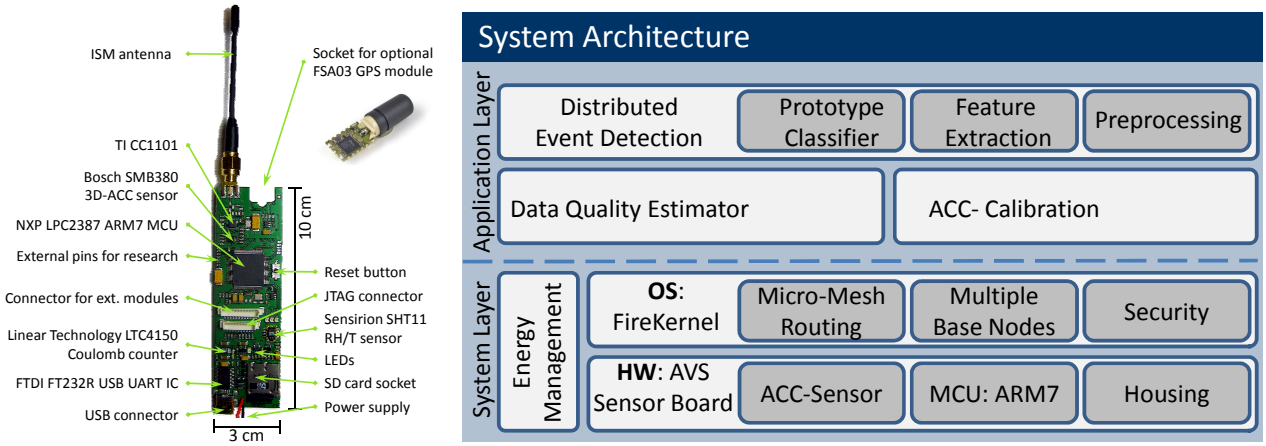
Based on the general discussion of architectures and algorithms, we now present an exemplary platform for distributed event detection called *AVS-Extrem*. This platform is employed in a project in which wireless sensor nodes equipped with accelerometers are integrated into the fence surrounding a security-sensitive area. The goal of this deployment is to detect intrusions into this area. The sensor nodes are thus programmed to collaboratively distinguish between numerous events, e.g., opening the fence or climbing over the fence, based on the movement of the fence elements as measured by the sensors.

On the algorithmic side, the system implements a pattern recognition approach (cf. Sec. 3.2). Each event class is described by a *prototype* that incorporates the most descriptive features which are extracted from the training data. Training data is gathered during on-site training as part of which the sensor network is repeatedly exposed to events of each class. When deployed, the prototypes allow the sensor nodes to distinguish between event classes. Architecturally, this is achieved using a distributed evaluation method (cf. Sec. 2.4) during which sensor nodes exchange extracted features and merge them into prototypes that span multiple sensor nodes.

4.1 Hardware Platform

The AVS-Extrem sensor nodes, as depicted in Figure 2a, were developed with motion-centric and localization-dependent applications in mind [5]. Each node is equipped with an ARM7-based NXP LPC2387 MCU with 96 KB RAM and 512 KB ROM running at a maximum of 72 MHz. For communications, a Texas Instruments CC1101 radio transceiver operating in the 868 MHz ISM band is used. A Bosch SMB380 low-power triaxial acceleration sensor with a range of up to 8 g, 10-bit resolution, and a sampling rate of up to 1,500 Hz is employed for acquiring motion data.

In addition to these core components, the PCB also includes a coulomb counter to measure battery usage,



(a) Sensor node

(b) System architecture

Figure 2: AVS-Extrem platform

an SD card slot to store trained motion patterns, and optionally a GPS receiver to establish the location of the node. The rugged housing fixes the node in place within vertical fence bars and enables us to conduct highly repeatable experiments.

The MCU can be woken up from power-saving status by the radio transceiver, the accelerometer, and a scheduled interrupt timer from the RTC. These external interrupts are raised when a packet is received (wake on radio, WOR), the accelerometer recognizes a change in position of the node, or a scheduled OS-level task needs to be run. Hence, only the radio transceiver, the accelerometer, and the RTC consume energy, while all other components of the node remain in low-power modes most of the time.

4.2 System Architecture

As illustrated in Figure 2b, the system architecture is divided into two layers: the application layer and the system layer. The system layer contains the AVS-Extrem sensor board, the operating system and the energy management. The FireKernel OS [12] is a low-power operating system that supports threading and priority-based preemptive multitasking. The latter is necessary to support uninterrupted sampling of the sensor while at the same time communicating with other nodes. A flexible energy management system supports WOR and numerous MCU-specific energy saving techniques.

For multi-hop communications, the nodes employ the reactive Micro Mesh Routing (MMR) protocol. We have adapted this protocol to our use case by adding the capability to reliably communicate with the base station of the sensor network via a set of multiple base nodes. Each of these nodes can be addressed by the network as a virtual base station and transparently relays packets to the real base station, which is connected to all base nodes via wireless or wired connections. This way, a fault-tolerant connection between any node in the WSN and the base station is provided. For secured communications, we have implemented the *ParSec* security layer that ensures confidentiality, authenticity, and integrity of transmitted data by using a symmetric cipher-algorithm in CBC mode.

The application layer encompasses a calibration routine for the acceleration sensor, optionally a data quality estimator, and the algorithmic core of our distributed event detection system. The task of the calibration routine is to compensate for the individual positional setup of the fence elements after every event. The data quality estimator employs the

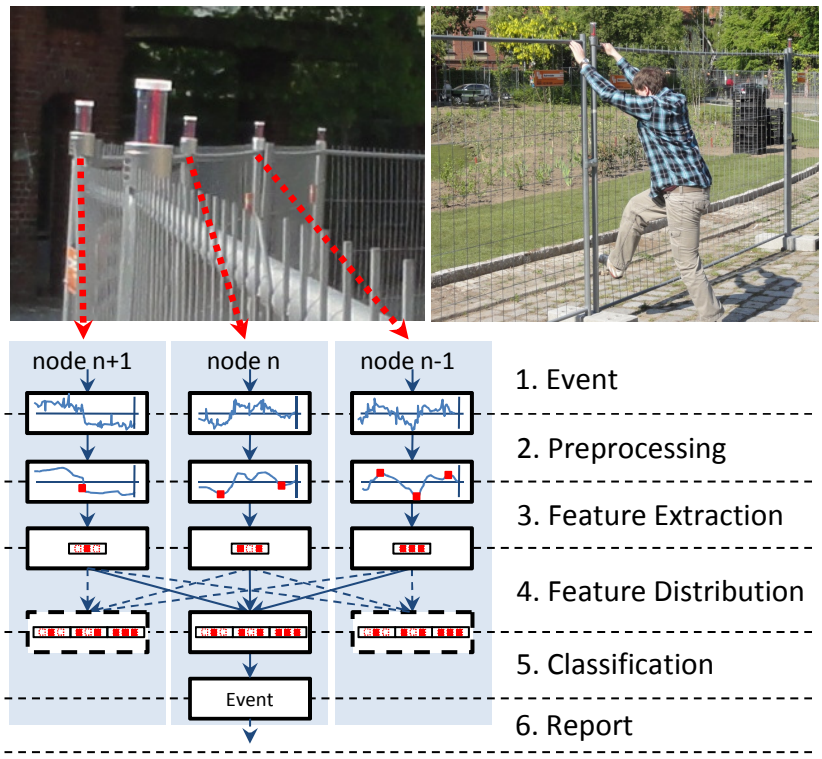


Figure 3: The distributed event detection process

Dempster-Shafer theory in order to establish confidence levels for each measured data point, and, by doing so, allows the system to ignore the readings of malfunctioning sensors. This provides additional input for the feature fusion module in the event detection system, as detailed in the next section.

4.3 Event Detection Algorithm

The distributed event detection system observes and evaluates events using multiple sensor nodes [13]. Events are recognized directly on the nodes in the network, i.e., without the need for a base station or other means of central coordination and processing. This is advantageous over other approaches to event detection in WSNs which transmit raw data to an external entity for evaluation or rely on simplistic pattern recognition schemes (cf. Secs. 2 and 3).

The general idea of our event detection system is that each node has its own view of any given event, but only for one node this view corresponds to one of the events it was trained to recognize. As illustrated in Figure 3, the nodes are attached to the fence at fixed inter-node distances. Whenever an event occurs, a lateral oscillation is propagated through the interconnected fence elements. The nodes thus sample the movement of the fence at different relative positions to the source of the event. If an event occurs, all affected nodes exchange the extracted features and concatenate their own features and the received features from the neighbors based on the relative sender position. This concatenated feature vector is then compared to a set of previously trained prototype vectors, and an event is reported in case of a match.

For a typical deployment, the system operates in three distinct phases: In the *calibration phase*, the

aforementioned calibration routine automatically calibrates the acceleration sensors. The *training phase* is initiated by a user via a control station to setup a newly deployed system. The purpose of this phase is to train the WSN to recognize application-specific events by executing a set of representative training events for each event class. Every event class is trained by extracting the same set of features from the sampled raw data on all sensor nodes, and then transmitting it to the control station. An adequately large pool of available features is crucial for the overall performance of the event detection system, since – depending on the characteristics of the deployment – some subsets of features is vastly superior in distinguishing between events than others. The control station performs a leave-one-out cross validation across all collected features to automatically select the optimal subset of features for this specific deployment. A prototype containing the most expressive features for each event class is then calculated and transmitted back to each sensor node.

Once the training is complete, the system enters the *detection phase*. In this phase, the nodes are configured to autonomously recognize the trained events and the control station is no longer needed. As illustrated in Figure 3, the sensors gather data, which is then preprocessed, segmented, and normalized. Only features used in the prototype vectors are extracted from the raw data, and then combined to form a feature vector. The extracted feature vector is then flooded to all sensor nodes in the n-hop neighborhood. In our deployments, n is usually set to 1 since the radio range of our sensor nodes is significantly larger than the spacial expansion of the fence motion caused by an event. After receiving all required feature vectors, each node performs a feature fusion by combining the feature vectors based and the relative position of the sender. Each node establishes the relative position of other nodes by evaluating the unique node IDs of received packets. As part of this overall process, each node builds its own view of the event using feature vectors from its neighboring nodes. Obviously, the combined feature vector is different for each node, because the respective neighboring nodes perceive the event differently depending on their location. The prototype classifier running on the node whose view of the event matches the trained view classifies the event correctly, while the feature vector represents an unknown pattern for the other nodes, which in turn ignore the event. If deemed relevant to the use case, e.g., the event of climbing over the fence in our scenario, the node that correctly classified the event transmits a packet reporting the detection to the base station.

5 Experimental Evaluation

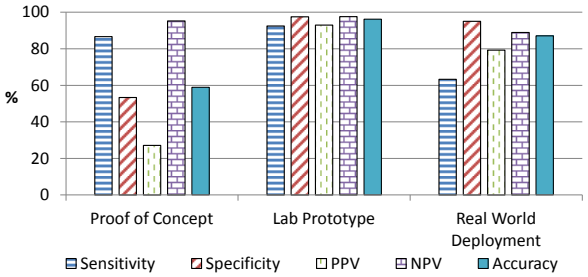
In this section, we present the results from our experimental evaluation of the detection accuracy and the energy efficiency of the AVS-Extrem platform. These results summarize four major experiments that we conducted over the past years [14, 4, 13, 5].

5.1 Detection Accuracy

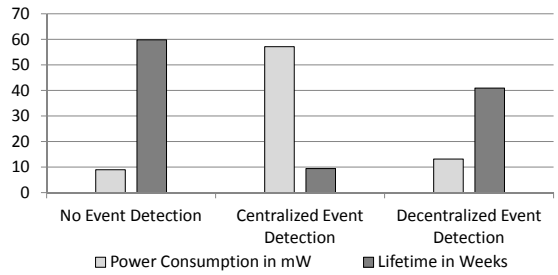
We evaluate the detection accuracy using three experiments: an initial proof-of-concept deployment, a lab prototype, and a real-world deployment.

In the proof-of-concept deployment [14], we attached ten ScatterWeb MSB sensor nodes, the predecessor of our current AVS-Extrem node, to a fence and exposed them to six different events: long and short shaking of the fence, kicking against the fence, leaning against the fence, and peeking and climbing over the fence. The proof-of-concept system did not support autonomous training, and hence we relied on a custom-built heuristic classifier implemented in a rule-based middleware. This classifier was manually configured to classify events based on visible patterns in the raw data.

As part of the lab experiment [4], we trained three sensor nodes to cooperatively recognize four different motion paths based on the acceleration data measured by the sensor. The four motion paths, comprising a square, a triangle, a circle and the capital letter U, were drawn simultaneously by three persons on a flat surface by moving the sensor nodes. Event detection was implemented using a cooperative fusion classifier which, for the first time, allowed for the feature vectors to include features from multiple sensor nodes.



(a) Detection quality of distributed event detection for different deployments



(b) Energy consumption and network lifetime of different event detection systems

Figure 4: Detection quality and energy consumption of different approaches to event detection

Finally, in our real-world deployment [13], we attached 100 sensor nodes to the fence of a construction site. We exposed the WSN to four different classes of events: shaking the fence, kicking against the fence, leaning against the fence, and climbing over the fence. This deployment utilized the same event detection algorithm as the lab prototype, but additionally introduced automated feature selection based on leave-one-out cross-validation (cf. Sec. 4.3) during the training phase. For the training, we chose a region of the fence that was free of any external obstructions and trained the WSN with 15 events of each class.

Figure 4a shows sensitivity, specificity, Positive Predictive Value (PPV), Negative Predictive Value (NPV), and accuracy for these three experiments.[1] The figure illustrates that AVS-Extrem achieves near-perfect results under lab conditions. Further, the final real-world deployment performs substantially better then the initial proof-of-concept setup with the rule-based classifier. The current system achieves an overall detection accuracy of 87.1%, an improvement of 28.8% over the proof-of-concept.

5.2 Energy Consumption

In contrast to the detection accuracy experiment, the energy consumption is evaluated with the succeeding hardware platform AVS-Extrem as introduced in Section 4. In our experiments in [5], we evaluated the energy consumption of a sensor node during the distributed event detection and recognition process. To measure the energy consumption of the whole board as accurately as possible, we soldered a 10 Ω shunt resistor into the supply line which is powered by a reference voltage of 5 V. The voltage of the shunt resistor was measured using a digital sampling oscilloscope (DSO). As the resistor and the voltage are known, we can calculate the value of the current and use it to calculate the electric power used by the sensor node over the time of one DSO sample. By integrating the electric power over the time of one system state, e.g., packet transmission or IDLE mode, we can measure the energy needed. This information can then be used to approximate the energy consumption of the sensor node over a given time (cf. [5]). During the event detection phase, the sensor nodes use the MCU power down (PD) mode, that also shuts down all internal peripherals. The wireless transceiver uses the wake-on-radio (WOR) mode and thus remains available for processing incoming data. The acceleration sensor is active and monitors the movement of the fence.

Figure 4b illustrates the average energy consumption and the resulting extrapolated network lifetime. The underlying scenario for this calculation is a deployment of seven nodes for which an event is generated and processed five times per hour. For comparison, we also plot the numbers for a deployment of the same sensor network, but without any event-related activity. As can been seen in the figure, the lifetime

[1] Definitions of these metrics are given in [13].

of the network that employs centralized event detection is drastically reduced in comparison to the idle network due to increase energy consumption. In contrast, distributed event detection is able to reduce the per-node energy consumption by 77%, thus increasing the lifetime of the network to an extrapolated total of 41 weeks.

These results underline the validity of our initial hypothesis, that distributed event detection is able jointly achieve the otherwise conflicting goals of high-accuracy event detection and long network lifetime.

6 Conclusion and Outlook

In this paper, we described how event detection in wireless sensor networks comes in several different forms, each with its specific trade-offs and, at times, particularly suited for certain types of deployments. The overall dominating factor is the trade-off between energy efficiency and event detection accuracy, two goals which are mutually exclusive for simple approaches. Systems that operate distributively, such as the AVS-Extrem platform which we used as an example in this paper, pave the way towards deployments that can provide both, highly accurate event detection as well as a long network lifetime.

As distributed approaches to event detection mature, we expect that a new, deployment-time trade-off will emerge: System that adapt techniques from machine learning (such as AVS-Extrem) profit from their inherent adaptability to specific scenarios, but on the downside also require extensive training for each new deployment. Other types of systems rely on on-site parametrization which can be performed quickly, but is unable to detect complex events. It will thus be interesting to explore in how far event detection systems can not only achieve high accuracy at low power consumption, but furthermore also be efficiently deployed without sacrificing either of the aforementioned properties.

Acknowledgments

This work was funded in part by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF) through the project AVS-Extrem.

References

[1] D. M. Doolin and N. Sitar. Wireless Sensors for Wildfire Monitoring. In *Proceedings of the SPIE Symposium on Smart Structures & Materials / NDE '05*, San Diego, CA, USA, Mar. 2005.

[2] M. F. Duarte and Y. H. Hu. Vehicle Classification in Distributed Sensor Networks. *Journal of Parallel and Distributed Computing*, 64(7), July 2004.

[3] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, Nov. 2000.

[4] N. Dziengel, G. Wittenburg, and J. Schiller. Towards Distributed Event Detection in Wireless Sensor Networks. In *Adjunct Proceedings of the Forth IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS '08)*, Santorini Island, Greece, June 2008.

[5] N. Dziengel, M. Ziegert, S. Adler, Z. Kasmi, S. Pfeiffer, and J. Schiller. Energy-Aware Distributed Fence Surveillance for Wireless Sensor Networks. In *Proceedings of the Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '11)*, Adelaide, Australia, Dec. 2011.

[6] H. Ghasemzadeh, V. Loseu, and R. Jafari. Collaborative Signal Processing for Action Recognition in Body Sensor Networks: A Distributed Classification Algorithm Using Motion Transcripts. In *Proceedings of the Ninth ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*, Stockholm, Sweden, Apr. 2010.

[7] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh. Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In *Proceedings of the Third International Conference on Embedded Networked Sensor Systems (SenSys '05)*, San Diego, CA, USA, Nov. 2005.

[8] Y. Kim, J. Kang, D. Kim, E. Kim, P. Chong, and S. Seo. Design of a Fence Surveillance System Based on Wireless Sensor Networks. In *Proceedings of the Second International Conference on Autonomic Computing and Communication Systems (Autonomics '08)*, Turin, Italy, Sept. 2008.

[9] F. Martincic and L. Schwiebert. Distributed Event Detection in Sensor Networks. In *Proceedings of the International Conference on Systems and Networks Communications (ICSNC '06)*, Tahiti, French Polynesia, Oct. 2006.

[10] A. Tavakoli, J. Zhang, and S. H. Son. Group-Based Event Detection in Undersea Sensor Networks. In *Proceedings of the Second International Workshop on Networked Sensing Systems (INSS '05)*, San Diego, CA, USA, June 2005.

[11] M. Wälchli, P. Skoczylas, M. Meer, and T. Braun. Distributed Event Localization and Tracking with Wireless Sensors. In *Proceedings of the Fifth International Conference on Wired/Wireless Internet Communications (WWIC '07)*, Coimbra, Portugal, May 2007.

[12] H. Will, K. Schleiser, and J. Schiller. A Real-time Kernel for Wireless Sensor Networks Employed in Rescue Scenarios. In *Proceedings of the 34th IEEE Conference on Local Computer Networks (LCN '09)*, Zürich, Switzerland, Oct. 2009.

[13] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller. A System for Distributed Event Detection in Wireless Sensor Networks. In *Proceedings of the Ninth ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*, Stockholm, Sweden, Apr. 2010.

[14] G. Wittenburg, K. Terfloth, F. L. Villafuerte, T. Naumowicz, H. Ritter, and J. Schiller. Fence Monitoring - Experimental Evaluation of a Use Case for Wireless Sensor Networks. In *Proceedings of the Forth European Conference on Wireless Sensor Networks (EWSN '07)*, Delft, The Netherlands, Jan. 2007.

[15] A. Yang, S. Iyengar, S. S. Sastry, R. Bajcsy, P. Kuryloski, and R. Jafari. Distributed Segmentation and Classification of Human Actions Using a Wearable Motion Sensor Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '08)*, Anchorage, AK, USA, June 2008.