Demo Abstract: In-network Training and Distributed Event Detection in Wireless Sensor Networks

Georg Wittenburg
wittenbu@inf.fu-berlin.de

Norman Dziengel
dziengel@inf.fu-berlin.de

Jochen Schiller
schiller@inf.fu-berlin.de

Department of Mathematics and Computer Science
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany

ABSTRACT

In order to avoid transmitting raw data to a base station, sensor nodes are trained to cooperatively recognize deployment-specific events based on the data sampled by their sensors. As both training and event detection are performed without the need for central coordination or processing, only information about the detected event needs to be reported.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; I.5.4 [**Pattern Recognition**]: Applications—*signal processing*

General Terms

Design, Experimentation, Performance

Keywords

Wireless Sensor Networks, Distributed Event Detection, In-network Data Processing, Pattern Recognition

1. INTRODUCTION

In-network data aggregation with the goal of avoiding excessive packet transmissions is commonly regarded as a key aspect for conserving energy and thus extending the lifetime of a Wireless Sensor Network (WSN). Data aggregation may be as simple as combining several sensor readings into one single packet or averaging data sampled on different nodes. However, even with limited per-node processing capabilities, more advanced methods are feasible.

In this demonstration, we present our system for distributed event detection *Patrec* that is capable of identifying complex application-specific events based on the raw sensory data collected by several sensor nodes. For instance, consider a WSN consisting of nodes equipped with accelerometers attached to a fence: Based on the acceleration data gathered by several nodes, the WSN is able to differentiate between events such as a person climbing over the fence or a person merely shaking the fence [6]. Event detection is performed collaboratively on those nodes whose sensors register the event and only a notification about which event was detected is transmitted back to the base station in order to alert the user. As events depend on the application as well as the deployment scenario, our system can be trained on-site as part of the deployment process.

This architecture improves upon other approaches that either consume additional energy by sending raw data back to the base station for centralized processing [5] and/or sacrifice accuracy by relying on simple heuristics, e.g. thresholds of sensor values [1] or number of affected nodes [3].

SenSys'08, November 5–7, 2008, Raleigh, North Carolina, USA.
ACM 978-1-59593-990-6/08/11.

2. SYSTEM ARCHITECTURE

The architecture of our system is depicted in Figure 1. We treat event detection as a layered process that runs in parallel on several sensor nodes. The general idea is to expand upon established methods from the area of pattern recognition by exchanging data about features extracted from the raw data between sensor nodes. The event detection process is triggered on any sensor node whose sensors register a noteworthy level of activity.

The tasks of the individual layers are as follows: During **raw data processing**, the stream of raw data is segmented using thresholds with hystereses and the data corresponding to one potential event is then filtered using an weighted moving average and normalized to optimize subsequent calculations on the microcontroller. The **feature extraction** uses the processed sensor data to calculate several descriptive features, e.g. duration, minimum/maximum/average values or distribution of frequencies. The numeric values for each feature are then concatenated into a *feature vector*. Additional normalizing of the extracted features makes sure that all features are comparable in the common feature space, thus allowing us to fuse features from different types of sensors. During **feature distribution and fusion**, the feature vectors of all sensor nodes on which the event detection has been triggered are sent to neighboring nodes and concatenated into a *combined feature vector*. Timeouts are used to compensate for the fact that per-node processing may have been triggered at different points in time depending on the sensory input. The **classification** compares the combined feature vector against a set of reference vectors with the Euclidean distance as metric. Each of the reference vectors corresponds to a previously trained class of events. If the event corresponding to nearest reference vector is deemed worthy of reporting, an identifier for this event is routed to the base station of the network.

The process of training the system works along the same lines. The lower layers are used without modification, however the final classification step is replaced by a corresponding training component: For each event to be classified later

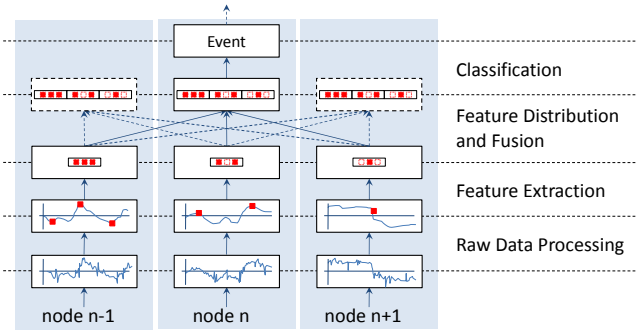

Figure 1: Distributed Event Detection Architecture of *Patrec* and Exemplary Detection Process

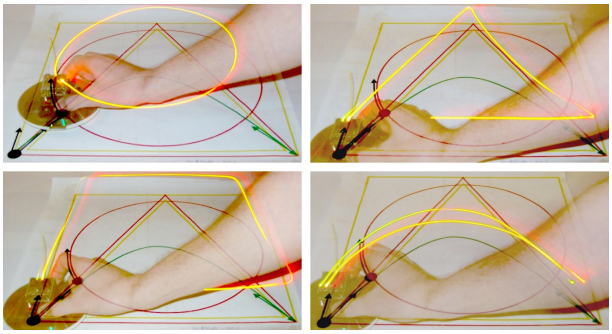

Figure 2: Sensor Node with Accelerometer Moved According to Four Previously Trained Shapes

on, a predetermined number of exemplary combined feature vectors are used to calculate an event-specific reference feature vector. An evaluation of further design alternatives is omitted for brevity but can be found in [2].

3. DEMONSTRATION SETUP

We have implemented the system for distributed event detection on ScatterWeb MSB sensor nodes [4] which are based on the TI MSP430 16-bit microcontroller with 5 kB of RAM. The sensor nodes are equipped with a Chipcon CC1020 868 MHz radio transceiver and a Freescale Semiconductor MMA7260Q accelerometer. The demonstration is completely self-contained, i.e. it requires no external components for coordination or processing.

Our demonstration consists of two distinct parts that respectively focus on training and distributed event detection. The training of the event detection system is started manually by a person holding the sensor node in one hand. The system will first calibrate the accelerometer and adjust the sensitivity parameters of the raw data processing layer based on how much the hand is trembling at its current position. After the calibration is completed, the user will be prompted by the LEDs of the node to teach the system three events in the form of motion patterns of his own choice. The beginning and end of each motion are detected automatically. The system subsequently extracts features from the acceleration data and constructs the feature vector. After repeating each motion three times, the feature vectors are combined to form the reference vector for the motion pattern in question. Once the training is complete, the user may repeat any of the previously learned motions and the sensor node will classify the motion pattern and report the result using its LEDs.

Distributed event detection is demonstrated using three sensor nodes that have been trained to recognize the four different geometric shapes depicted in Figure 2. A distributed event consists of the three nodes being moved simultaneously. To this end, three persons agree on one of the four shapes and then retrace the printed outline of the shape in question. Once again, each of the nodes detects automatically the beginning and end of each motion, extracts features and constructs a feature vector. These feature vectors are then exchanged via radio and on each node fused into a combined feature vector which is classified using previously trained reference vectors. The event resulting from the classification is reported using the LEDs of the nodes. In addition to the results of the distributed event detection, each node also reports which shape was recognized locally. This way, users can assess in how far distributed event detection increases the overall accuracy of the system as opposed to local event detection.

4. CONCLUSIONS

Our system for distributed event detection *Patrec* is capable of reliably detecting events based on sensory values sampled across multiple sensor nodes. It does not require central coordination or processing and expands upon other approaches by adapting pattern recognition methods to the distributed nature of WSNs. Furthermore, the system is capable of learning new events to detect by the means of a supervised training process.

Preliminary results from lab experiments are available in [2] and support our claim that distributed in-network event detection in WSNs increases the detection accuracy over existing approaches at a reasonable energy expenditure. We are currently in the process of preparing a large scale deployment of our system to verify our findings.

5. REFERENCES

[1] R. R. Brooks, P. Ramanathan, and A. M. Sayeed. Distributed Target Classification and Tracking in Sensor Networks. *IEEE*, 91(8):1163–1171, Aug. 2003.

[2] N. Dziengel, G. Wittenburg, and J. Schiller. Towards Distributed Event Detection in Wireless Sensor Networks. In *DCOSS '08*, Santorini, Greece, June 2008.

[3] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, and B. Krogh. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *Trans. Sensor Networks*, 2(1), Feb. 2006.

[4] J. Schiller, A. Liers, and H. Ritter. ScatterWeb: A Wireless Sensornet Platform for Research and Teaching. *Computer Communications*, 28:1545–1551, Apr. 2005.

[5] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *OSDI '06*, Seattle, USA, Nov. 2006.

[6] 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 *EWSN '07*, Delft, The Netherlands, Jan. 2007.