Sensor-based Clustering for Indoor Applications

Matthias Gauger, Olga Saukh, Marcus Handte, Pedro José Marrón
Universität Bonn, Bonn, Germany and Fraunhofer IAIS, St. Augustin, Germany
{gauger, saukh, handte, pjmarron}@cs.uni-bonn.de

Andreas Heydlauff, Kurt Rothermel
Universität Stuttgart, Germany
{heydlaas, rothermel}@ipvs.uni-stuttgart.de

***Abstract*—The lifetime requirements on wireless sensor networks often require the redundant deployment of sensor nodes with appropriate management mechanisms based on node clustering. Yet, existing clustering approaches do not take the primary task of sensor networks into account: performing relevant measurements. They usually form 'arbitrary' clusters, e.g., using connectivity information, and thus, the resulting measurements are often of only limited use to the applications. This problem can be avoided by considering application-specific semantics. For indoor applications, the notion of a room provides a natural unit of clustering since walls are constructed deliberately to ensure locality. This paper shows that it is feasible to automatically create clusters that reflect boundaries between rooms by analyzing the measurements of inexpensive, broadly available sensors. The paper first analyzes the applicability of statistical clustering methods and based on this analysis, it proposes and evaluates a lightweight approach to determine clusters in real deployments.**

I. Introduction

The lifetime requirements of wireless sensor network deployments continue to exceed the capacity of today's battery technology by orders of magnitude. A frequently used solution to this problem is the redundant deployment of sensor nodes in combination with appropriate management mechanisms such as node clustering with temporary cluster-node deactivation and cluster-head rotation. However, existing clustering approaches usually form 'arbitrary' groups, for example based on connectivity information. As a result, the deactivation of clustered nodes or the aggregation of values in clusters can severely distort measurements since the results are heavily dependent on the respective clustering.

This problem can be avoided by forming clusters in such a way that they reflect real world semantics that are meaningful to the application. For indoor applications, the notion of a room provides a natural unit of clustering that is particularly suitable for sensing, since walls are deliberately constructed to ensure locality by blocking sound, light, temperature, humidity and so on. Yet, the sheer number of sensor nodes in sensor network deployments with sufficient redundancy renders manual cluster configuration a prohibitively tedious and sometimes even impossible task. At the same time, adequate location information is often not available or too costly to acquire indoors so that cluster membership cannot be inferred from the position of nodes in the building. Moreover, neither connectivity information nor distance measurement techniques like RSSI or TDOA are able to separate neighboring nodes located in adjacent areas: Despite all of their limitations indoors, these techniques rather estimate distances or express closeness than detecting area boundaries like walls. Thus, there is a need to come up with other means of obtaining meaningful clusters in sensor networks automatically.

This paper shows that it is feasible to automatically create clusters (groups of nodes) that adhere to room boundaries using inexpensive and broadly available sensors. To do this, the paper first presents an off-line analysis of the applicability of relevant data filtering and statistical data clustering methods for the task of detecting room boundaries through sensor measurements. Using the results of the analysis, the paper proposes and evaluates a lightweight approach to compute clusters in real world wireless sensor network deployments.

It is worth noting that apart from redundancy management to improve the network lifetime, clustering information derived on the basis of real world semantics can also be useful for a broad spectrum of additional tasks. Examples include role assignment as described in Frank and Römer [1], anomaly detection [2] and room level querying.

The remainder of this paper is organized as follows. The following section briefly reviews related work. Section III analyzes the applicability of different data filtering and statistical data clustering methods to determine clusters of nodes that reside within the same room of a building. Based on this analysis, Section IV derives the approach to determine clusters of nodes in real sensor network deployments and Section V evaluates it. Finally, Section VI concludes the paper and discusses future work.

II. Related Work

Clustering is a frequently used mechanism in wireless sensor networks. Applications include routing, aggregation and coordination. In most cases, clusters are formed on the basis of connectivity information or on the basis of geographical positions of sensor nodes. LEACH [3], for example, was one of the first clustering protocols for wireless sensor networks that employed the idea of cluster-head rotation in order to improve lifetime. However, the clusters formed by LEACH are purely based on a probability value to become a cluster-head and the topology of the network. The same holds true for other protocols such as HEED [4].

Published in Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON '08. 16-20 June 2008, San Francisco, CA

http://dx.doi.org/10.1109/SAHCN.2008.64

The middleware presented in [5] exhibits a close relationship to our work as it is also targeted at clustering nodes on the basis of sensor measurements. However, the actual comparison and classification of measurements to derive clusters is an aspect not considered by the middleware. This missing building block is the focus of this paper.

The work presented in [6] considers a grouping of sensor nodes that is – in a broader sense – quite similar to the one we are aiming at. The grouping aims at identifying sensors that are worn by the same person. To do this, Lester et al. rely on acceleration sensors. Their results show that a successful grouping can be established with high accuracy if the sensors are worn on the same part of the body. However, the utilized methods are not applicable to our scenario with static sensor nodes, since the characteristics of the available sensor data are very different.

Meka and Singh [7] cluster sensor nodes based on similarity of sensor data. However, their approach is not oriented on real world semantics. Instead, nodes in their clusters must only conform to a specified minimum similarity level. Moreover, their distributed computation is quite sophisticated, for example requiring the nodes to organize and coordinate in a quadtree structure.

As part of this work we use different approaches from the area of statistical data clustering in order to classify objects so that objects of the same class have a high degree of similarity while objects in different classes have a low degree. An overview of such methods and their application areas can be found in [8]. Examples include data mining [9], health psychology [10] and gene analysis.

An important application field for our approach is data aggregation [11]. One aggregation approach that could specifically benefit from clusters formed on the basis of real world semantics is [12] as it aims at optimizing the transmission structure of the network based on correlations between sensor measurements. It can be expected that such correlations are frequently present inside the same room of a building.

III. Analysis

Before we describe the approach to cluster sensor nodes in real wireless sensor network deployments in the next section, we first analyse the applicability of relevant data filtering and statistical data clustering methods to determine the groups of sensor nodes that reside within the same room. The results of this analysis are important for two reasons. Firstly, they are used to select appropriate methods for the approach and, secondly, they are used as the best-case scenario in order to evaluate the quality of the approach.

To perform the analysis, we use a centralized experimental setup to collect real sensor data in a building. During the experiment, each participating sensor node first performs a series of n measurements that are stored in its local flash memory. To globally synchronize the measurements of different sensor nodes, we use the Flooding Time Synchronization Protocol (FTSP) [13] for TinyOS. After the measurement series has been completed, each sensor node forwards its stored data to a central base station. After all measurements have been collected, we start with an off-line analysis using a PC. Clearly, this setting is not suitable for real world wireless sensor networks. However, it eliminates a number of complications as discussed later on and ensures that multiple methods can be run on the same set of input data.

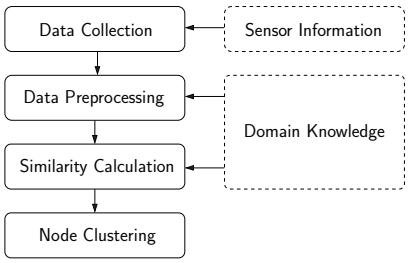


Fig. 1: Processing steps

The off-line analysis consists of three steps, namely data preprocessing, similarity calculation and node clustering. Since domain knowledge can greatly improve the clustering process [8], we also discuss how it can be integrated into the off-line analysis. Fig. 1 shows the individual steps of the overall experiment. In the following, we describe the details of the off-line analysis for each step.

A. Data Preprocessing

Data preprocessing or data filtering modifies or transforms the acquired sensor data in a way that supports the following steps. Besides overcoming some effects of the lack of calibration, data preprocessing is one way of including domain knowledge into the analysis by emphasizing specific features hidden in the sensor data. From the vast selection of preprocessing steps we have opted for four basic methods, since they are particularly adequate for filtering sensor data. Each of the filters presented below is oriented towards some specific characteristic of sensor data:

- The deviation between the output of uncalibrated sensor chips can increase linearly with the values. The **normalization filter** deals with this problem by bringing the sensor data from all data sources to one common scale.
- In order to remove small amplitude jittering of sensor output, a **data smoothing filter** can be applied. It smoothes the incoming data signal by calculating the average of the previous x values and uses this average as the output.
- Abstracting from more complex irregularities is the task of the **curve tendency filter**. This filter solely records whether the current sensor value is higher or lower than the previous value, thereby producing binary output.
- The motivation for the **event detection filter** originates from the observation that some types of sensors occasionally experience significant changes of their values in between two samples. An exemplary source of such a change can be a light that is switched on or off. We record such sudden changes using the event detection filter. Thus, this type of filter is a way of incorporating

$$\phi = \frac{ad-bc}{\sqrt{efgh}}$$

(a) Formula

	x^-	x^+	Total
y^-	a	b	e
y^+	c	d	f
Total	g	h	n

(b) Counter values

Fig. 2: Phi coefficient

domain knowledge in the preprocessing step.

B. Similarity Calculation

For the calculation of similarities there are distance metrics and correlation coefficients. Distance metrics express how far apart two variables are according to a certain criterion. A correlation coefficient measures the strength of a relationship between two variables. A large set of different distance and correlation measures have been developed in the field of data clustering. For the analysis, we use a set of elementary metrics that cover the different available classes. Specifically, we consider the Euclidean distance and the Manhattan distance as well as the Pearson coefficient and the Phi coefficient for binary data.

The **Euclidean distance** and the **Manhattan distance** are two distance metrics that are most popular for their use in the field of geometry:

$$d_{Eucl}(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}, \quad d_{Man}(x,y)=\sum_{i=1}^{n}|x_i-y_i| \qquad (1)$$

The main idea of **Pearson's product-moment correlation coefficient** is to measure the tendency of two variables to increase and decrease together assuming a linear relationship of these variables. Equation 2 shows how it is calculated for two variables x and y when a series of n samples has been taken. $\overline{x}$ and $\overline{y}$ are the mean values and σ_x and σ_y the standard deviations calculated over the n samples of the variables x and y respectively.

$$r_{xy} = \frac{\sum_{i=1}^{n}((x_i-\overline{x})(y_i-\overline{y}))}{n\sigma_x\sigma_y} \qquad (2)$$

The **Phi coefficient** works on data with only two magnitudes like the data generated by the event filter or the curve tendency filter. It is calculated as shown in the equation in Fig. 2 (a) based on information about how often the binary values of x and y match and how often they conflict (see Fig. 2 (b)).

C. Node Clustering

The last step is to use a data clustering algorithm to calculate a clustering of nodes based on the acquired similarity information. For this step, we use hierarchical and partitional clustering.

The basic idea of **hierarchical clustering** is to arrange the individual elements of a set in a tree so that short branch distances between elements in the tree express a high level of similarity and long branch distances express a small level of similarity. The clustering process can either be done agglomerative or divisive. Agglomerative methods start with each element being its own cluster and then form larger clusters step by step. Divisive clustering works the other way around. We only discuss agglomerative clustering here since it outperformed divisive clustering in our experiments.

In each step of the clustering process, the two most similar elements are joined together to form a new aggregated element. This is repeated until all elements are part of one large aggregated element. This creates a tree data structure with the individual elements forming the leafs of the tree and the union of all elements forming the root of the tree. Different methods are available to compare the similarity among aggregated elements, e.g., minimum similarity (**single linkage**), maximum similarity (**complete linkage**) or average similarity (**average linkage**). If the number of clusters is known, it is possible to deduce the desired set of clusters by stopping the agglomerative clustering once the desired number is reached.

The class of **partitional clustering** approaches represents all algorithms that directly calculate a specified number of clusters. For our analysis, we used the k-means clustering algorithm which is the most popular representative of the square error clustering methods. Applying it requires knowledge about the number of clusters to be found in the system. Initially, the individual elements are randomly assigned to one of the k clusters. Then, for each cluster, the mean vector of the cluster elements (the "centroid") is calculated. Each element is then reassigned to the cluster whose centroid has the highest similarity to the data vector of the respective element. The computation of the cluster centroids and the reassignments of the elements is repeated until some convergence criterion is met (e.g., the assignment does not change anymore). Due to the random initialization, each run of the k-means algorithm can yield different results.

D. Combining Clustering Trees

With more than one information source available (e.g., multiple sensors), it is desirable to be able to combine information acquired from these sources in order to improve the resulting clustering quality and to balance weaknesses and strengths of different criteria. An example of a weakness of one criterion in a specific scenario is light sensor data in a room with extremely irregular lighting conditions like some sensors being exposed to direct sunlight while other sensors lie in much darker areas.

However, combining distance or correlation information from different sources is non-trivial and cannot be done by simply calculating the average distance or correlation matrices as the range of values strongly depends on the properties of the sensor, the preprocessing methods and the similarity measure.

We found a suitable solution to this problem coming from the field of biology where the average consensus supertree (ACS) [14] method has been developed to find a consensus based on multiple clustering trees. The basic idea of ACS is to calculate path length matrices for each of the input trees then take the average matrix of these path length matrices and use it as input for a second round of clustering. The path length

TABLE I: Average correlation

	Home scenario 1		Home scenario 2	
	Inside	Between rooms	Inside	Between rooms
Humidity	0.87	0.22	0.86	-0.02
Light PAR	0.96	0.50	0.93	0.20
Light TSR	0.98	0.39	0.95	0.34
Temperature	0.83	0.35	0.77	0.18

(a) inside rooms / between rooms

	Home scenario 1	Home scenario 2
Temp. – Light TSR	0.52	0.17
Temp. – Light PAR	0.44	0.15
Temp. – Humidity	0.17	0.28
Li. TSR – Li. PAR	0.88	0.94
Li. TSR – Humidity	0.08	0.07
Li. PAR – Humidity	0.03	0.04

(b) among different sensors

matrix of a tree contains for each pair of nodes the number of branches between the nodes in the input tree. While the original version of ACS used a least-squares algorithm for the second cluster calculation we simply use the same clustering algorithm that was used to calculate the input clustering trees. Due to space limitations, we refer to [14] for a more detailed description of ACS.

E. Experimental Analysis

For the analysis of the described methods we used Tmote Sky sensor nodes providing temperature, humidity and light sensors in the form of photosynthetically active radiation (PAR) as well as total solar radiation (TSR). In each scenario, we distributed twelve nodes to four different rooms with three nodes being placed in each room. The sensor nodes were attached to the wall or to furniture in different parts of the monitored rooms at different heights (ranging from the floor to the ceiling) and with different orientations. We paid attention to the fact that none of the sensor chips was directly covered by other artifacts of the room. However, effects like some nodes lying in the shadow of an artifact or lying in the airflow of a window were deliberately not avoided.

1) Validating assumptions: In a first step we validated the assumption that there exists a significantly higher correlation among sensor values of nodes located in the same area than of nodes located in different areas using the complete data sets collected in our exemplary scenarios and the Pearson coefficient. The results shown in Table I (a) confirm this and also illustrate interesting differences among different sensors and scenarios.

As a second step, Table I (b) shows potential correlations among different types of sensors again using the average value of the Pearson coefficient. This time it is calculated for all pairs of different sensors on the same sensor node. If two types of sensor output are highly correlated then operating both sensors on the same node cannot provide much additional information. Such a correlation only exists between the TSR light sensor and the PAR light sensor. But even in this case the correlation tends to be smaller than the correlation among different nodes in the same area when considering a single sensor type.

2) Results: We evaluated different reasonable combinations of preprocessing, similarity calculation and data clustering approaches. The basic evaluation criterion in this analysis is the correctness of the clustering result expressed by the percentage of groups that have been calculated correctly. A correctly calculated group contains all nodes belonging to this group (in our case: all nodes lying in the same room) and does not contain any node belonging to another group. This is more complicated for hierarchical clustering as we need to differentiate between node groups that form a correct part of the clustering tree and node groups that additionally can be correctly deduced during the calculation of the clustering tree as described in Section III-C.

Among the four sensor types we experimented with, the TSR light sensor proved to be the most reliable one that provided useful results over a large number of cases. While the PAR light sensor and the humidity sensor are also useful in many cases, temperature showed to be a surprisingly weak criterion.

The event detection filter is the only preprocessing step that consistently proved to be useful in a large set of cases across all experiments. The curve tendency filter also provided good clustering results, however, it required a large number of samples or tended not to stabilize which disqualifies it for use in real applications. Improvements by data smoothing and normalization were not reproducible across experiments.

Concerning the similarity among nodes, the analysis of our collected data showed that the Pearson correlation coefficient is the most reliable similarity metric. It works well for all sensor types albeit requiring different numbers of data samples to stabilize (See data in Table II (a)). The Euclidean and the Manhattan metrics work well in a smaller set of scenarios with only some types of data. No clear advantage of one of the two over the other can be detected. The Phi coefficient works well in combination with the event filter.

Both the hierarchical clustering algorithms and the k-means clustering algorithm worked well with a similar quality. For that reason the following discussions will only consider complete linkage hierarchical clustering and only show its results. Due to space limitations, we defer the analysis of combining multiple criteria using ACS to the evaluation in Section V.

Table II (a) shows some exemplary node clustering results with different combinations of filters and similarity measures recorded for the two home scenarios. Both experiments were started in the evening and the analysis is based on data samples taken every 200 seconds. For each combination the table shows the percentage of clusters correctly identified and the number of steps (= the number of data samples) required for the algorithm to stabilize. We use a dash to indicate the case where the algorithm fails to find a stable solution. [1] Table II (b) summarizes the results of our analysis of the centralized approach considering the usefulness of the data

[1] If the number of steps is supplemented by a star (*) then the result shown is for the number of correct clusters being part of the tree and not for the number of clusters correctly deduced during the calculation of the clustering tree as described above.

TABLE II: Results of the analysis

	Home scenario 1		Home scenario 2	
	Success	Steps	Success	Steps
Humidity Pearson	100%	50	75-100%	-
PAR Pearson	100%	70	100%	70
TSR Pearson	100%	70	100%	70
Temperature Pearson	100%	500	75%	900*
TSR Euclidean	75-100%	-	100%	10*
TSR Manhattan	75-100%	-	100%	10*
TSR Event10.0 Phi	100%	90*	100%	80*
TSR Event20.0 Phi	100%	40*	100%	10*
TSR Event30.0 Phi	100%	60	100%	70

(a) Exemplary results

	$T(n)$	$S(n)$	**Usefulness**
Normalization	$O(n)$	$O(n)$	Limited
Data smoothing	$O(n)$	$O(1)$	Limited
Curve tendency	$O(n)$	$O(1)$	Limited
Event detection	$O(n)$	$O(1)$	High
Euclidean	$O(n)$	$O(1)$	High
Manhattan	$O(n)$	$O(1)$	High
Pearson	$O(n)$	$O(1)$	High
Phi	$O(n)$	$O(1)$	High
Hier. single link.	$O(m^2 \log m)$	$O(m^2)$	High
Hier. compl. link.	$O(m^2 \log m)$	$O(m^2)$	High
Hier. avg. link.	$O(m^2 \log m)$	$O(m^2)$	High
k-means	$O(klmn)$	$O(kn)$	High

(b) Result overview

preprocessing steps, the similarity calculation methods and the node clustering algorithms as well as their respective time complexities ($T(n)$) and space complexities ($S(n)$).

While Table II (a) illustrates that a correct clustering of nodes can be found with the help of different criteria using only a small number of data samples, it does not show the dependency of the clustering result on the start time of the analysis. Our experiments showed that node clustering works best when the data collection is started in the evening with human activity present in the rooms. To illustrate this, Fig. 3 (a) shows the progress of the Pearson coefficient over time for TSR light sensor data collected in home scenario 1 over a period of approximately two days. The figure distinguishes between node pairs of the same group and of different groups. While the average correlation coefficient quickly approaches 1 for node pairs of the same group, one can also see two time periods (colored in the figure) that significantly increase the average correlation among nodes of different groups which in turn complicates the clustering.

The first colored time period corresponds to the first night during which all light sensors record similar values in the range of 0 – irrespective of their group affiliation. This makes the nodes appear more similar and expresses itself in an increasing correlation coefficient as can be seen in the figure.

We can use this knowledge about the existence of time periods without sensor activity to remove the data samples from the calculation of the Pearson coefficient for a pair of nodes using a simple rule: Interrupt the recording of data samples if both values of the data sample pair have been below a specified threshold th for more than l consecutive data sample pairs. Resume the calculation once one value lies above th again. Fig. 3 (b) shows the result of applying this heuristic (using $th = 3$ and $l = 5$) to the collected data. As can be seen, the average correlation among nodes belonging to the same group still grows quickly while the average correlation among nodes belonging to different groups does not grow anymore during the night time (Maximum of 0.37 instead of 0.59).

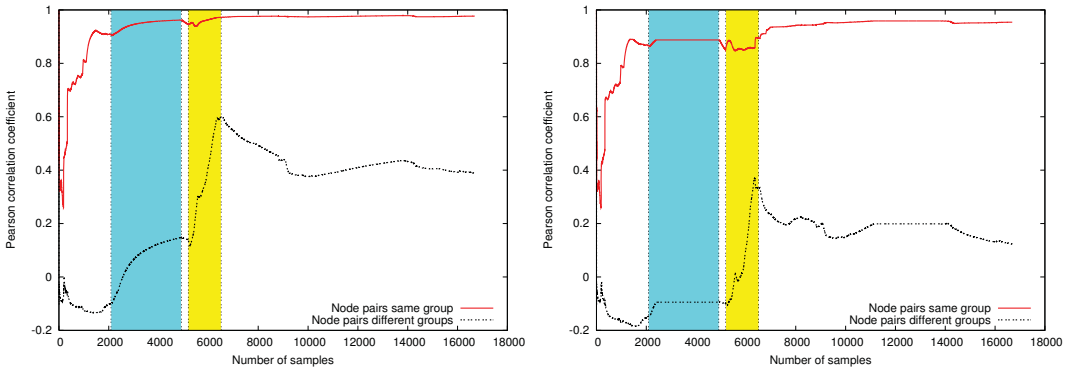
The second colored time interval represents the time period from forenoon to noon. Here, the problem is that bright sunlight experienced by many nodes causes high data values that quickly dominate the much smaller values collected during the evening and the night. A small number of these samples then suffices to loose much of the useful similarity information acquired previously. Only after some time this is compensated and the average correlation among nodes located in different rooms starts to decrease again.

Coping with the negative effects of the second time interval is more challenging. The simplest solution is to time the data collection appropriately. As an alternative, it is possible to collect and analyze the data from different value ranges separately. As shown later in the evaluation, a final option is to balance weaknesses of individual sensors by combining node clustering data using ACS. Interestingly, our experiments showed that the performance of the humidity and the temperature sensor also depend on the time of the day and on the presence of human activity. This indicates that the recording of phenomena (for example triggered by human activity) is an important factor in the successful clustering of nodes.

Another result of our analysis is the insight that it is difficult to achieve a correct clustering quickly using a high sampling rate. Instead, it is possible to save on the number of required data values and consequently on the number of required messages by increasing the data sampling interval. This is illustrated by Fig. 4 which shows the number of steps required for stabilization for the TSR light and the humidity sensors depending on the sampling rate in home scenario 1. It shows that the number of samples required for stabilization increases with the sampling rate due to the comparatively low volatility of the recorded criteria.

3) Complexity: The complexity values shown in Table II (b) are based on the number of clusters k, the number of iterations l, the number of nodes m and the number of data samples n. The complexities given for the data preprocessing methods are per node and the complexities for the similarity calculation methods are per node pair.

The literature on statistical data clustering (e.g., [8]) identifies k-means clustering as more efficient to calculate than hierarchical clustering. On the one hand, this is due to the fact that for most applications the size of the data vector n of each cluster element is usually small and fixed. Consequently, it is ignored in the complexity calculations. On the other hand, the number of elements to be clustered m can be extremely large. The situation is different in our scenario where the number of elements (nodes) stays comparatively small whereas the size of the data vectors collected can be significant. The consequence for our approach is that hierarchical clustering is typically



(a) without removing samples (b) with removing samples

Fig. 3: Average progress of the Pearson coefficient for the TSR light sensor

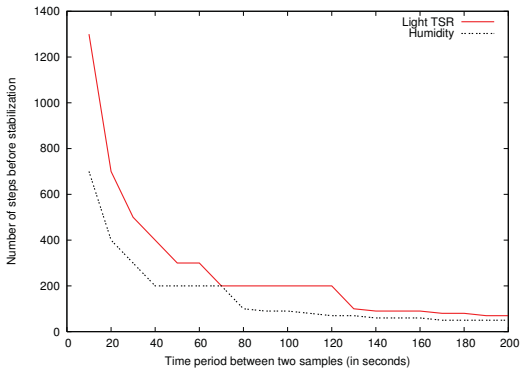


Fig. 4: Clustering stabilization depending on the sampling interval

less costly than k-means clustering as the number of samples n only influences the similarity calculation and not the node clustering itself.

IV. Distributed Approach

Forwarding the collected sensor data from all nodes in the network with a sufficiently high sampling rate creates a high traffic load despite potential optimizations like aggregation. Thus, to be applicable to real sensor network deployments, significant parts of the previously described off-line analysis must be implemented on the sensor nodes in a distributed manner. Thereby, it is imperative to avoid complex and expensive tasks such as time synchronization or measurement rounds. Note that this type of distribution is not related to established distributed data clustering methods whose primary goal is to distribute calculations in order to achieve a higher performance.

The basic idea behind our approach is that each sensor node periodically reads its own sensor data every p seconds (beacon send interval). The node broadcasts the collected data in a beacon message to its neighbors. When a node receives a beacon message it reads its own sensor data and compares the two data samples to extract information required for the similarity calculation. This way, data preprocessing and similarity calculation are directly done on the individual sensor nodes. Only the last step of the clustering process – the actual computation of the node clustering – cannot be distributed as it requires a global view on similarity information among all nodes. However, collecting the similarity information at a central point is much less expensive than collecting complete vectors of sensor data.

Based on the results of our previous analysis the only data preprocessing step still supported for our exemplary scenario is the detection of events. For other scenarios, most other filters could be added if needed. The only exception is the normalization filter whose space complexity impedes a distributed implementation.

Implementing similarity measurements and clustering techniques on sensor nodes requires that they can be calculated on-line with a single pass, i.e., without storing the previous measurements. Both the **Euclidean and the Manhattan distance** can fulfill this requirement. A sensor node solely needs to store two values, a sum for the distance and a counter for the samples. Similarly, the **Pearson correlation coefficient** can be used by reformulating Equation 2 into Equation 3. Together with a reformulation of the formula for the standard deviation as shown in Equation 4 this results in the following list of six values that must be collected by a sensor node:

$$\sum_{i=1}^{n} x_i \quad \sum_{i=1}^{n} x_i^2 \quad \sum_{i=1}^{n} x_i y_i \quad \sum_{i=1}^{n} y_i \quad \sum_{i=1}^{n} y_i^2 \quad n$$

$$r_{xy} = \frac{\sum_{i=1}^{n} x_i y_i - \overline{y}\sum_{i=1}^{n} x_i - \overline{x}\sum_{i=1}^{n} y_i + n\overline{x}\overline{y}}{n\sigma_x \sigma_y} \tag{3}$$

$$\sigma_x = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2 - \frac{1}{n} 2\overline{x}\sum_{i=1}^{n} x_i + \overline{x}^2} \tag{4}$$

To compute the **Phi coefficient**, the sensor nodes only need to collect the four counter values from the table in Fig. 2 (b). Compared to other collected values, the advantage of the Phi counter values is that their range is small since the value of all four counters is less than or equal to the number of samples.

The different types of **hierarchical clustering** discussed in section III-C base their calculation on the distance or correlation matrix containing similarity information among all nodes. If the similarity values collected by the individual nodes are collected at a central base station then the hierarchical

clustering can be calculated as discussed in the previous section.

The situation is different for the **k-means clustering** algorithm since the algorithm requires the computation the mean vector of the data vectors of all objects in a cluster in each iteration. This is not possible when the sensor nodes only store and provide similarity values calculated out of the data vectors.

A. Inaccuracies

The distributed approach described above introduces a new set of inaccuracies that can affect the clustering results. The first inaccuracy is introduced by the delay between the measurements on a pair of nodes caused by the time required for the message transfer. The impact of this delay depends on the volatility of the respective sensor data. The second inaccuracy is caused by lost beacon messages, e.g., due to collisions. Without precautions, this can result in differing numbers of samples recorded at different node pairs. Given a sufficient number of samples, this is can be neglected for the Phi and the Pearson coefficient as their values are by definition normalized between -1 and 1. However, the values of the Euclidean or Manhattan distance increase monotonically with the number of samples so that only distance values based on the same number of samples are comparable. An efficient solution for this problem is to collect a predefined number of samples for each node pair. However, this can still lead to samples being recorded in different time periods for different node pairs.

A more fundamental potential source of inaccuracies results from the fact that it is not feasible to compute the similarities between arbitrary node pairs in a distributed manner. Doing so would require the global distribution of measurements, e.g., by flooding beacons. However, in many application scenarios, this problem can be mitigated by limiting the beaconing scope (i.e., its hop count) using domain knowledge. In our particular indoor scenario for instance it is feasible to limit the scope to 1 since it is reasonable to assume that sensors residing within the same room are direct neighbors and nodes that are not direct neighbors are not within the same room. In such cases we use this domain knowledge to assign an infinite distance (or a correlation value of 0) to the remaining node pairs.

Special care is also required for the detection of events in a distributed manner as the time interval between two data samples recorded by a node can be arbitrarily small as data is sampled whenever a beacon message from a neighbor arrives. This causes unbalanced conditions for the detection of events. Our solution to this problem is to let the node store the previous two sensor values it collected in reaction to its own time trigger (and not in reaction to receiving a beacon message) as reference values. When sensor data is sampled following a received beacon message the node selects one of the two reference values for the comparison according to the following rule: If the difference of the current time and the time stamp of the younger reference value is smaller than half of the beacon send interval then use the old reference value. Use the younger reference value otherwise.

B. Optimizations

The sampling overhead, which depends on the number of beacon messages, can be reduced using the following two optimizations: First, it is not necessary to sample if the last sample is still very fresh (less than q seconds old). Second, sensor data originally sampled for calculations can be directly sent out in a beacon message if the next scheduled send time is less than r seconds away.

Another possible optimization reduces the amount of state stored by each sensor for the calculation of similarity information for neighboring nodes. In the basic approach, two neighboring nodes x and y both collect information about their similarity with respect to each other. While in most cases the two similarity values will not be identical (as they are based on different data sample pairs) their information should be the same or very similar. So we can practically split the number of neighboring nodes clustering information must be collected for in half by selecting only one node to store the information. This can either be done explicitly through negotiation or by applying a static rule based on node identifiers.

C. Implementation

As for the experimental analysis, we used Tmote Sky sensor nodes to implement the distributed approach. The implementation was done in nesC for the TinyOS 2.0 operating system. The implementation supports the Euclidean distance, the Pearson correlation coefficient and the Phi correlation coefficient. The three criteria differ in the amount of memory consumed by the data required for the calculation: 7 bytes per sensor and node pair for the Euclidean distance, 25 bytes for the Pearson coefficient and 8 bytes for the Phi coefficient (reserving between 2 and 5 bytes per value to avoid overflows). The considerably higher memory consumption of the Pearson coefficient data motivates why considering the other criteria is still worthwhile despite the superior performance the Pearson coefficient showed in the centralized analysis.

V. Evaluation

This section evaluates the performance of the distributed node clustering approach in different situations using real-world measurements. For this purpose, nodes were distributed in two home scenarios with 15 nodes in 5 different rooms with 3 nodes per room in each scenario. We prepared five experiments in the two scenarios (200 seconds beacon send interval) started at different times in the evening reflecting our knowledge that this time period provides the best data for the clustering of nodes.

One important result of the experiments is that basing the clustering decision on a single criterion allows us to reach good node clustering results with acceptable effort only in a limited share of the experiments. For example, while considering the TSR light sensor alone led to correct and stable clustering after around 100 data samples in some experiments it failed to detect more than 80% of the groups correctly or failed to stabilize at all in other experiments. This highlights the need for combining node clustering information

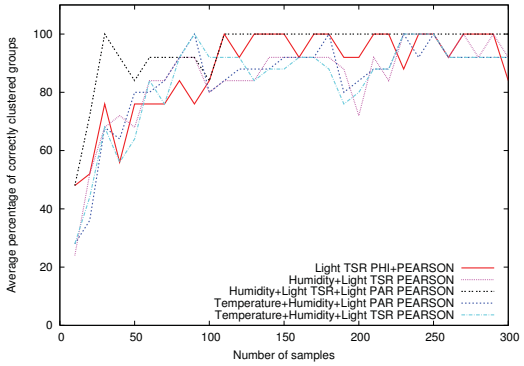


Fig. 5: Percentage of groups clustered correctly for various criteria

from different criteria with the help of ACS. While such combinations also show weaknesses in a subset of experiments their results vary less across experiments and are in general more stable over time. Consequently, the best performing node clusterings all combine two or more different criteria. An analysis of the average percentage of correctly clustered groups calculated over all experiments is shown for five particularly well performing combinations in Fig. 5.

In our scenarios, the light sensors are again indispensable information sources. However, both temperature and humidity sensors can play an important role as stabilizing elements that can compensate periods of weaknesses of the light sensors. Relying on temperature or humidity alone is difficult in the settings we tested and would increase the required number of samples considerably.

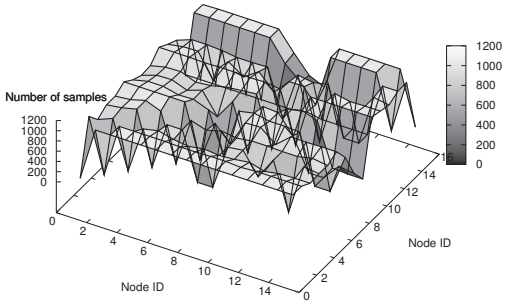
Due to space limitations Fig. 5 can only present a small part of the result space. Note, however, that a lot of different criteria and combinations of criteria were able to correctly cluster 80% and more of the groups correctly using a relatively small number of samples. In general, the experiments confirmed our expectations that a successful node clustering is more difficult to achieve due to the challenges of the distributed approach but still possible with only a limited amount of effort.

We have identified different numbers of samples recorded by different node pairs as a source of inaccuracies and as a challenge to the successful clustering of nodes. The table in Fig. 6 (a) analyzes the severity of this problem for four different data collections showing the range of the number of samples recorded by different node pairs. Fig. 6 (b) exemplifies this for one data collection from the first scenario and shows the number of samples recorded for each node pair. The analysis shows that such irregularities in the number of samples recorded are the normal case rather than an exception and that the standard deviation is significant in some cases. Interestingly, there are even considerable differences among experiments conducted in the same scenario.

We see a direct connection between the radio irregularities and an inferior performance of the Euclidean distance metric that fails to confirm the promising results from the centralized analysis. While the calculation is based on the same number

	Max	Min	Avg	Std dev
Scenario 1 - Experiment 1	1040	0	829	335
Scenario 1 - Experiment 2	1130	0	924	332
Scenario 2 - Experiment 1	660	0	582	152
Scenario 2 - Experiment 2	670	410	630	31

(a) Analysis



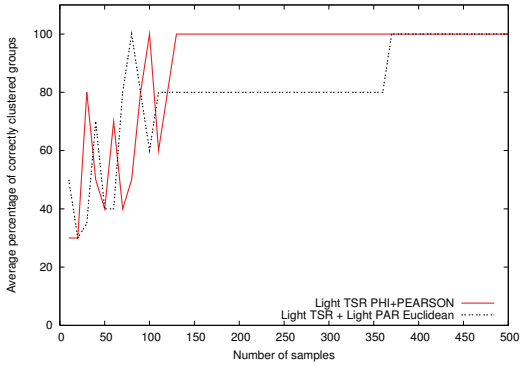
(b) First home scenario

Fig. 6: Number of samples

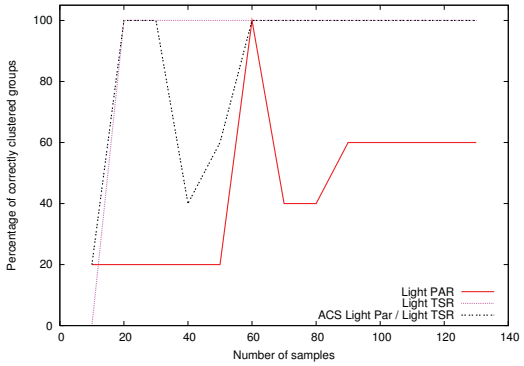
of samples in all cases, the samples of different node pairs might have been recorded at different points in time under completely different external conditions. Consequently, the Euclidean distance metric only produced useful results in experiments with a low variance in the number of samples and should therefore only be used in scenarios with relatively stable communication links between all node pairs.

Node clustering works best when started in a time period with artificial light and considerable human activity. To explore how well our method works under common, less optimal conditions – normal daylight scenarios without any human activity – we started two experiments (home scenario 2, data collected every 200 seconds) in the morning and made sure that there was no human activity in the covered rooms for at least the first eight hours. The most interesting result of these experiments is that all types of sensors are negatively affected by the lack of human activity and consequently the list of criteria stabilizing on a successful clustering in an acceptable amount of time is smaller. Fig. 7 a) shows the number of groups correctly detected over the number of samples considered for a set of criteria performing reasonably well in this setting.

In another experiment we explored the other extreme – how explicit human activity can positively influence the clustering of nodes. We recorded sensor data every 10 seconds and alternately turned on or turned off the light in one of the rooms every 20 seconds as an explicit trigger. While the temperature and humidity sensors were not able to generate useful results in a short time interval, with the TSR light sensor the clustering stabilized on a correct result after only a little more than three minutes. The PAR light sensor and the combination of both light sensors took a little longer to stabilize as shown in Fig. 7 b) and the PAR light sensor alone was not able to identify all groups correctly.



(a) without activity



(b) supported by explicit triggers

Fig. 7: Node clustering

VI. Conclusion

Clustering is a powerful technique. For many sensor network applications, however, it is beneficial and sometimes even mandatory to form clusters on the basis of real world semantics.

In this paper, we have shown that it is feasible to automatically create clusters that reflect rooms by analyzing the measurements of inexpensive and broadly available sensors. To this end, we have performed an extensive analysis of the suitability of relevant data filtering and statistical data clustering methods. Based on the results of this analysis, we have devised an approach to automatically determine clusters that adhere to room boundaries in real deployments. The evaluation shows that added domain knowledge leads to significant quality improvements that allow for a successful clustering under real world conditions.

At the present time, we are investigating whether it is possible to perform the data clustering in a completely distributed fashion. A key problem thereby is the definition and reliable detection of adequate stabilization criteria. In the future, we would like to extend the presented approach to other scenarios that do not exhibit the comparatively clear correlations as present in and between individual rooms. In this context it might be useful to perform a thorough analysis of fuzzy data clustering approaches.

We are sure that the idea of clustering devices based on sensor data does not have to be limited to sensor nodes but could also be useful in other application domains. With the availability of cheap, high-quality sensor chips it might even be reasonable to add sensor chips to devices for the sole purpose of grouping devices together. Potential applications that come to mind include self-configuring home entertainment systems, home automation systems or alarm systems.

We also plan to generalize our concept beyond the application of clustering nodes. For example, information about correlations among the sensor data of nodes could be used for lossy data aggregation or for doing node duty cycling without calculating a clustering first. Nodes can keep histories of their sensor values correlation to neighboring nodes and use this data to decide whether the data (or the node itself) is redundant.

References

[1] C. Frank and K. Römer, "Algorithms for generic role assignment in wireless sensor networks," in *Proc. of the 3rd Int. Conf. on Embedded Networked Sensor Systems*, 2005.

[2] V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and B. Maglaris, "Hierarchical anomaly detection in distributed large-scale sensor networks," in *Proc. of the 11th Symp. on Computers and Communications*, 2006, pp. 761–767.

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS '00: Proc. of the 33rd Hawaii Int. Conf. on System Sciences-Volume 8*, 2000.

[4] O. Younis and S. Fahmy, "Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[5] T. Nagata, H. Oguma, and K. Yamazaki, "A sensor networking middleware for clustering similar things," in *Proc. of the W'shop on Smart Object Systems*, 2005.

[6] J. Lester, B. Hannaford, and G. Borriello, "Are you with me? using accelerometers to determine if two devices are carried by the same person," in *Proceedings of the 2nd International Conference on Pervasive Computing*, 2004.

[7] A. Meka and A. K. Singh, "Distributed spatial clustering in sensor networks." in *Proc. of the 10th Int. Conf. on Extending Database Technology*, 2006.

[8] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[9] P. Berkhin, "Survey of clustering data mining techniques," Accrue Software, San Jose, CA, Tech. Rep., 2002.

[10] J. Clatworthy, D. Buick, M. Hankins, J. Weinman, and R. Horne, "The use and reporting of cluster analysis in health psychology: A review." *British Journal of Health Psychology*, vol. 10, no. Part 3, pp. 329–358, 2005.

[11] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[12] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "On network correlated data gathering," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2004)*, 2004.

[13] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. of the 2nd Int. Conf. on Embedded Networked Sensor Systems*, 2004.

[14] F.-J. Lapointe and G. Cucumel, "The average consensus procedure: Combination of weighted trees containing identical or overlapping sets of taxa," *Systematic Biology*, vol. 46, 1997.