A Clustering-Based Characteristic Model for Unreliable Sensor Network Data

Ulf Kulau, Tobias Breuer and Lars Wolf
Institute of Operating Systems and Computer Networks
TU Braunschweig
Email: [kulau|breuer|wolf]@ibr.cs.tu-bs.de

***Abstract*—Wireless Sensor Networks (WSNs) that are deployed outdoors suffer from rough environmental conditions. Moreover, cheap hardware or energy management techniques like undervolting might lead to inaccurate sensing results and, thus, unreliable data. Hence we propose a characteristic model of every sensor node's data to derive the 'normal' behavior of sensors statistically. Beside massively reducing the total amount of sensed data, this representative model can be used to detect discordant values and redundancies between nodes.**
Theoretical considerations as well as a functionality test of a server implementation with real WSN nodes show the features of this approach.

I. Introduction

WSNs are an important part of upcoming Internet of Things (IoT) applications as they can be used for environmental sensing, structural health monitoring or smart agriculture. Here the mentioned application areas have in common that they are primarily deployed outdoors. However, recent experiences in real world applications revealed that the reliability of WSN nodes suffer from rough environmental conditions. Where a node works well at normal room temperatures, an increased temperature could lead to unpredictable errors like resets, calculation errors or clock drifts [1], [2]. Moreover, it could not be guaranteed that sensed data is not modified during its processing on the node itself e.g. when using energy management techniques like undervolting [3].
A more practical and indeed popular example of a WSN application, where such issues could arise, is the detection of a forest fire [4]. When a sensor detects a suddenly increased temperature value, how does the sink know whether this 'abnormal' temperature data is caused by a possible fire or a faulty WSN node?
For this purpose we propose a data clustering approach which can help to avoid possible costs of wrong decisions. Moreover, a statistic model of sensor data is more lightweight. When dealing with hundreds of sensors, a handy model of sensor data leads to an efficient processing and storage. Besides a more precise interpretation of the context, clustering of sensed data offers several advantages. An exemplary use case presented in this paper is the detection of redundancy between sensed data. Matching the clusters of several sensor nodes would give direct information about redundant sensing. If data sensed by several nodes are very similar, it could be considered as equivalent. Finally this would allow to alternate between these sensor nodes to save energy.

II. Related Work

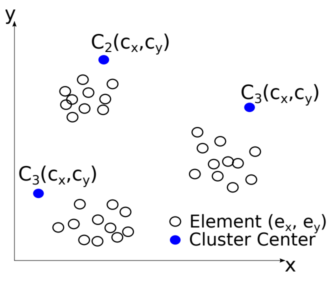
Indeed, data clustering and the used methods to enable this work are not new. A detailed insight in the advantages of data clustering is given in [5]. The classification methods, the Principal Component Analysis (PCA) and distance functions used in this paper are common techniques to analyze a given amount of data [6], [7]. In [8] also the k-means algorithm is considered to cluster the data of WSNs. While the efficiency of the k-means algorithm is optimized, the special need for clustering the sensor data is almost vague. A more frequent use case for clustering in WSN is not focussed on the data but much more on the structure of the network itself. In terms of energy awareness, the nodes of a WSN are clustered to build up hierarchical structures for lower duty-cycles, or optimize the transmit power of the nodes [9], [10], [11].
Nevertheless, clustering methods are also considered for data aggregation in WSNs [12], [13]. One use of data clustering is that correlation between data can reduce the overall traffic of the WSN. This intention is comparable with the presented approach of finding redundant data to share the resource of energy for a given task.
Actually, in [14], [15] also clusters are used to detect outliers of WSN data, where the proceeding of [14] is comparable with the *cluster-in-cluster* mapping of Section V. Sequential time slots are defined and distances of cluster centres of different slots are compared.

III. Classification of Data – Clustering

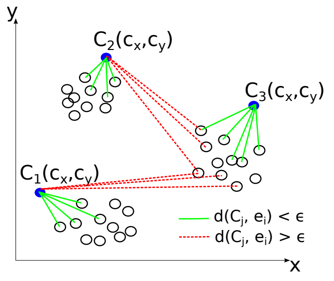
In general the classification of elements is defined as the possibility to arrange these objects by their characteristics. These characteristics are defined by a n-dimensional feature vector, indeed, in this work we deal with 2-dimensional feature-spaces $\mathbb{D}^2$ only. A group of elements with similar characteristics is called a cluster, where the elements within a cluster show a low standard deviation to each other. *Hard clustering* postulates that an element belongs to only one cluster. However, in this paper we deal with *soft clustering*, so that an element can belong to more than one cluster. Thus, *soft clustering* allows overlapping clusters.

A. Classification using k-means Algorithm

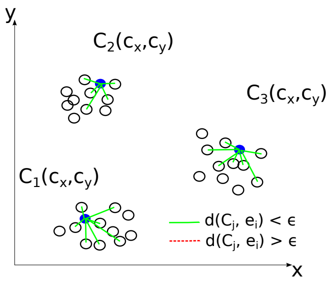
Not every clustering algorithm is sufficient to cluster continuously sampled data. For example the classification of elements using the Gaussian Distribution is not sufficient as it leads to oversized clusters.
For this purpose the k-means algorithm is applied as it is a common method to classify given elements [6].



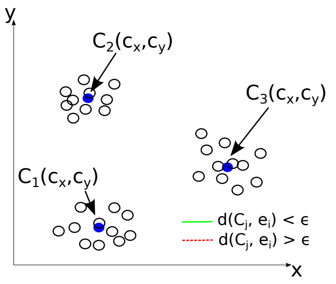
(a) step 1



(b) step 2



(c) step 3



(d) step 4

Figure 1. Principle execution of the k-means algorithm to classify elements.

The series of pictures of Figure 1 shows the principle of k-means classification algorithm. Firstly, the algorithm generates several potential clusters C_j with a random center $C_j(c_x, c_y)$. The quantity of generated clusters within the feature space $\mathbb{D}^2$ is variable and can be increased or decreased according to the specific use case. In Figure 1(a) three clusters, C_1, C_2 and C_3 are generated exemplarily.
Subsequently, for each element $e_i \in \mathbb{D}^2$ the distance to the clusters C_j is calculated. The distance between an element and a cluster is given by a distance function $d(e_i, C_j)$ which is equivalent to the classification criterion. Within a 2-dimensional feature space, it was shown that the Euclidean distance is an adequate method to classify the elements [7].

$$d_{euc}(e_i, C_j) = \left\| \begin{pmatrix} x_e - c_x \\ y_e - c_y \end{pmatrix} \right\| \tag{1}$$

The Euclidean distance d_{euc} is a useful tool to get the distance between an element e_i and a cluster C_j. Indeed, the simple information if an element is far away from a center of a cluster or not might not be sufficient. To decide whether an element belongs to a cluster or not, it is hard to define which distance should be undershot to join a cluster. For this reason the distance function is slightly extended.

$$d_{sqr}(e_i, C_j) = \left\| \begin{pmatrix} x_e - c_x \\ y_e - c_y \end{pmatrix} \right\|^2 \tag{2}$$

The advantage of the quadratic euclidean distance is that distances are weighted. Thus, distances of elements which are far away from a cluster become much greater, whereas small distances remain small. As a result the elements of the feature space convert faster to a cluster, which decreases the execution time of the k-means algorithm.
Now, if the distance $d(C_j, e_i)$ of an element undershots an specific threshold ϵ, the element e_i is assigned to the cluster C_j. This process is illustrated in Figure 1(b), where the dashed lines (red) indicate that the threshold was exceeded and the solid lines (green) that an element fits to a cluster. This step is repeated while every element of the feature space is either part of a cluster or not assignable. Afterwards, the mean values of every cluster C_j are recalculated with the related elements $e_i \in C_j$. In this case, as seen in Figure 1(c), the centres of the clusters moved towards the elements of the feature space. Step 2 and step 3 are repeated which leads to the following optimization criterion:

$$Minimize \sum_{j=1}^{M} \sum_{e_i \in C_j} \|e_i - C_j\|^2 \tag{3}$$

$$\text{with } e_i(e_x, e_y), C_j(c_x, c_y) \in \mathbb{D}^2$$

According to [16] the k-means algorithm is np-hard, which would lead to a polynomial runtime. For an adequate termination criteria, the movement of the clusters gives a good measure whether an almost optimal classification is reached or not. Hence, after every iteration cycle the ratio between the current clusters C_j^n and the previous clusters C_j^{n-1} is calculated. Whenever the movement of clusters remains almost constant, it can be assumed that the optimization is finished. It should be mentioned, that it is not advisable to define a fixed number of clusters M. The coverage of elements by these clusters could be too low, so that a termination criteria might be violated. A more useful method is to add new random clusters whenever a predefined coverage criterion is not reached. Thus, the total amount of clusters which represents the data depends on the characteristic of the data themselves.

IV. Characteristic Model

Basically a characteristic model means that this model is representative for the whole dataset yet more handy. In the following, based on the k-means classification, a statistic characteristic model of a dataset can be derived with the aid of the common PCA method [17]. For this purpose the covariance matrices of the previously defined clusters are derived to calculate their eigenvalues and eigenvectors. These eigenvalues and eigenvectors form an ellipse around the center of clusters, which means that the resulting ellipse is representative for all elements within a cluster.
In statistics the covariance describes the connection between two random values X and Y. The calculation of the covariance is given in the following equation and results from the expectancy values $E[X]$ and $E[Y]$ as well as the values X and Y.

$$\begin{aligned} Cov(X) = Cov(X,X) = E((X - E[X])(X - E[X])) = VAR(X) \\ Cov(Y) = Cov(Y,Y) = E((Y - E[Y])(Y - E[Y])) = VAR(Y) \\ Cov(X,Y) = Cov(Y,X) = E((X - E[X])(Y - E[Y])) \end{aligned}$$

The covariance matrix holds the pairwise covariances of X and Y. Hence, with regard to the previously defined clusters, the covariance matrix is representative for the elements e_i within a cluster C_j. For a particular cluster C_j, the covariance matrix is given as follows:

$$Cov_{C_j} = \begin{pmatrix} Cov(X_{C_j}, X_{C_j}) & Cov(X_{C_j}, Y_{C_j}) \\ Cov(Y_{C_j}, X_{C_j}) & Cov(Y_{C_j}, Y_{C_j}) \end{pmatrix} \tag{4}$$

To derive the eigenvalues and eigenvectors of the covariance matrix, the system of equations $Cov_{C_j} - \lambda E = 0$ has to be solved. However, the determinant $det(Cov_{C_j} - \lambda E)$ leads to a simple quadratic equation. This equation can be solved e.g.

by the pq-formula to get the eigenvalues $\lambda_{1,2}$. Hence together with the derived eigenvalues the eigenvectors $\vec{e_1}$, $\vec{e_2}$ are given by:

$$\vec{e_1} = \begin{pmatrix} Cov(X,X) - \lambda_1 \\ Cov(Y,X) \end{pmatrix}, \vec{e_2} = \begin{pmatrix} Cov(X,Y) \\ Cov(Y,Y) - \lambda_2 \end{pmatrix} \quad (5)$$

Finally, the normalization of the eigenvectors leads to a statistical representation of the cluster C_j. As an example, the following Figure 2 illustrates the description of clusters by using covariance ellipses.

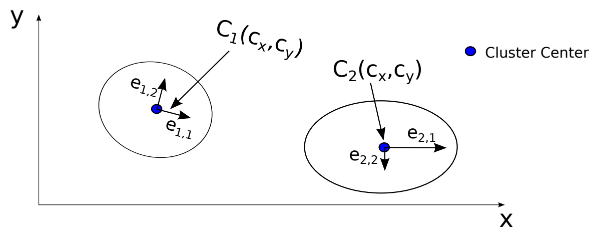


Figure 2. Exemplary covariance-ellipses of two clusters.

The definition of a characteristic model based on the covariance ellipses offers a more flexible integration of new data. By using a statistical description of a cluster, a range of tolerance is added to the model per se. If necessary, this tolerance range can be adapted by the eigenvalues, respectively eigenvectors.

V. Detection of invalid Data

Indeed, the trend of data can change rapidly which leads to the question if this fluctuations are still covered by the covariance ellipses to avoid misinterpretations. For example, when measuring the temperature of nodes deployed outdoors, the radiation of the sun could lead to erratic rises of temperature values. In consequence when the proposed approach is used for online monitoring the hitherto static characteristic model has to be updated continuously.
In the following C_j describes a cluster within the characteristic model $\bar{C}$ as defined in the previous Section IV. Whenever new sampled data $e_i^{new}(e_x, e_y)$ arrive at the sink, it has to be checked if an element is part of the characteristic model or not. A simple but naive approach is the *point-in-cluster* method.

$$\forall e_i^{new} \begin{cases} e_i^{new} \in C_j, \text{ if } e_i^{new} \text{ within } C_j \\ e_i^{new} \notin C_j, \text{ if } e_i^{new} \text{ not within } C_j \end{cases} \quad (6)$$

With the aid of the ellispe equation this distinction shows a low computational complexity. In addition the *cluster-in-cluster* method utilizes more than one new element for comparison with the characteristic model. In this case a whole cluster C_j^{new} of new elements $e_i^{new} \in C_i^{new}$ is compared with the clusters $C_j \in \bar{C}$. For this purpose it is assumed that new data share a common context so that it is allowed to condense them to a single cluster. However, these simple methods do not consider the statistical deviation of the model. The *Mahalanobis-Distance* might be a better measure to define the distance between clusters and new elements [7].

$$d_{Maha}(C_j, e_i) = \sqrt{\begin{pmatrix} c_x - e_x \\ c_y - e_y \end{pmatrix}^T Cov_{C_j}^{-1} \begin{pmatrix} c_x - e_x \\ c_y - e_y \end{pmatrix}} \quad (7)$$

In contrast to the euclidean distance, the *Mahalanobis-Distance* includes the covariance of the cluster. Thus, the *Mahalanobis-Distance* gives a measure of how many standard deviations an element e_i is away from the cluster C_j.
At least the *Manhattan-Distance* is considered. The distance between two points is given as the sum of the distances of their Cartesian coordinates [7]. Thus, the Manhattan-Distance results in an echelon form:

$$d_{Man}(C_j, e_i) = \left\| \begin{pmatrix} c_x - e_x \\ c_y - e_y \end{pmatrix} \right\|_1 \quad (8)$$

Later on in Section V-B the capabilities of the presented distance methods are evaluated.

A. Updating the Characteristic Model

The characteristic model is always based on the already sampled data set only. Hence, the ellipses have to be updated continuously to keep the model up to date and allow validation of incoming data.
Initially, by using the previously described distance functions, it can be decided if new data fit to the current model. Statistically, if new data fit into the current model, they are added and the model is updated afterwards. This process of updating the model is illustrated in the following Figure 3:

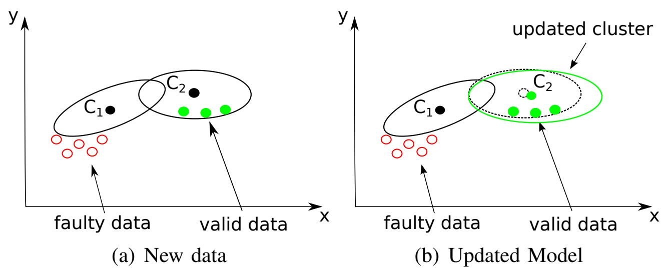


Figure 3. Update of the characteristic model by adding new and valid data.

The non-filled dots (red) represent potentially faulty data, whereas the filled dots (green) symbolize valid data. Valid data are added to the corresponding clusters and the model is updated by recalculating the involved covariance ellipses. Nevertheless, it should be avoided that the model expands too much over time as it can get far too tolerant or it could evolve in a wrong direction. Thus, after a period of time the whole classification should be executed instead of only updating the same model over and over again.

B. Interim evaluation

To test the quality of the characteristic model, the update of the model and the usability of the distance functions, an exemplary dataset was generated. This data correspond to a one day measurement of temperature values with a sample rate of $1Hz$ which leads to 86400 elements. A characteristic model $\bar{C_m}$ of this dataset was derived. Afterwards $\bar{C_m}$ was fed with the same dataset but random noise $0 \leq \Delta N_i$ was added iteratively. Thus, the detection rate, respectively the absorption of the new elements can be checked. In practice, at the first run no noise was added ($\Delta N_0 = 0$). Hence, $\bar{C_m}$ was fed by its own underlying dataset. For the second iteration every element was distorted by a random value between $\in [0, \Delta N_1]$. This noised data were added to the model $\bar{C_m}$ and the detection rate was checked again. Iteratively, the noise level was increased and the described process was repeated.

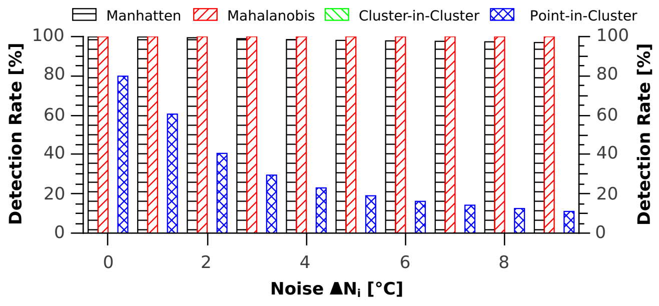


Figure 4. Detection rate of noisy data and different distance functions. (Note: Cluster-in-Cluster always $< 0.05\%$)

By using the previously introduced distance functions, Figure 4 shows the result of this test. The noise is given on the x-axis as an absolute temperature value, whereas the detection rate is given on the y-axis. In this case the detection rate gives a measure how many elements have been added to the model. It can be seen that the detection rate depends on the used distance function. Due to the consideration of the covariance, the Mahalanobis-Distance is more robust against noisy data. The other distance functions loose their ability to match new elements with increasing noise.

VI. Redundancy between Data

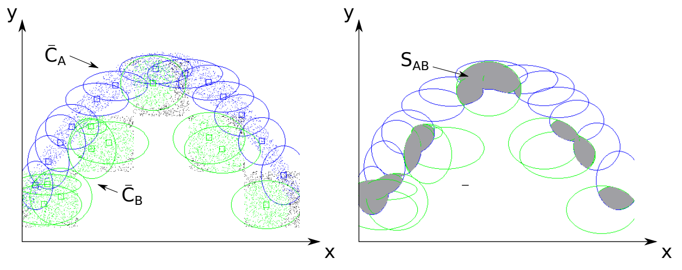
In some application areas of WSNs the deployment of nodes might be rather unstructured and random. A few nodes of the network could be located close together so that these nodes could alternate their sensing activity to save energy. Without any knowledge about the network structure, the characteristic models of these nodes can be used to identify such areas of redundant sensing. We assume two characteristic models $\bar{C_A}$ and $\bar{C_B}$ which belong to the sensor nodes A and B. To get the overlapping areas of $\bar{C_A}$ and $\bar{C_B}$ the intersection of models is calculated.

$$S_{AB} = \bar{C}_A \cap \bar{C}_B \tag{9}$$

As described above, the clusters are described by covariance ellipses. Thus, equation 9 leads to the problem that the overlap of two ellipses $C_j^A \in \bar{C_A}$ and $C_j^B \in \bar{C_B}$ has to be calculated. In the end the resulting area of intersections gives a measure how similar the sensed data of sensor nodes are. Figure 5 illustrates this principle of finding redundancies between sensor nodes. In Figure 5(a) the exemplary data set of two independent sensor nodes is classified. Figure 5(b) highlights the intersection of the two characteristic models.

VII. Functionality and Evaluation

To show the functionality of the presented approach we implemented a server application which is able to handle incoming data of several nodes. We initially tested the implementation and derived methods with a small benchmark consisting of five wireless sensor nodes. For every node which sends its data to the sink, a corresponding characteristic model is generated and updated continuously. With a sample rate of



(a) Charateristic Models (b) Intersection

Figure 5. Classified data and corresponding redundancy of two exemplary sensor nodes.

$1Hz$, every node samples the room temperature and sends this value to the sink immediately. The nodes were deployed within an apartment and placed on the window sill of different rooms. Thus, although the nodes were placed indoors, the radiation of the sun still affects the temperature measurements.

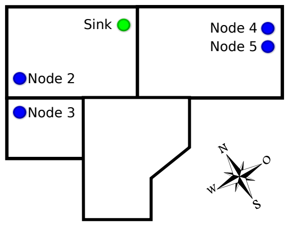


Figure 6. Draft of the test setup and ground-plan of the used apartment with round positions of the nodes.

With the intention that node 4 and node 5 should sample similar temperatures, these nodes were wilfully placed in the same room. Thus, the nodes should sample almost redundant data so that the detection of redundancy between data can be evaluated.

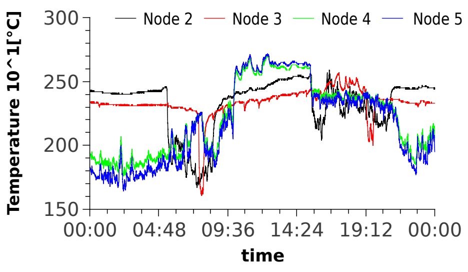
A. *Classification*

To analyse the computational overhead when executing the k-means algorithm, real temperature data from the test-setup are classified by the server. A normal laptop with an Intel Core 2 Duo 1,6 Ghz Central Processing Unit (CPU) and 4GB RAM was used for all evaluations. The following Table I shows the results.

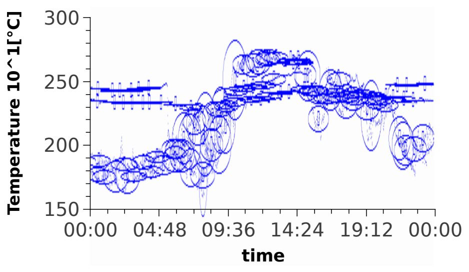
Table I. Runtime of the classification of real measurements (Day 3)

Node	Samples	Runtime	Iterations	Cluster
2	83162	429ms	17	22
3	83635	521ms	19	22
4	83592	581ms	22	28
5	83635	457ms	23	24

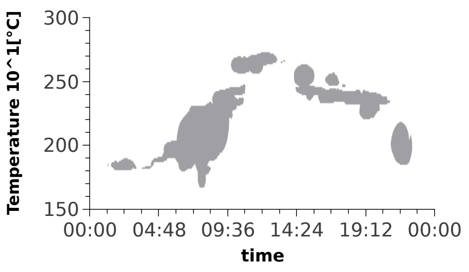
This evaluation was repeated for 7 more days. All in all the averaged runtime for the classification of ≈ 83500 temperature data amounts to $504.57ms$. Moreover, this evaluation shows the massive reduction of total sensor data. Arround 83500 data points can be represented by ≈ 25 clusters.
The final classification of four nodes over five days is depicted in Figure 7.



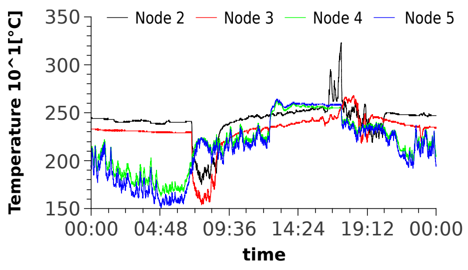
(a) Temperature (Day 1)



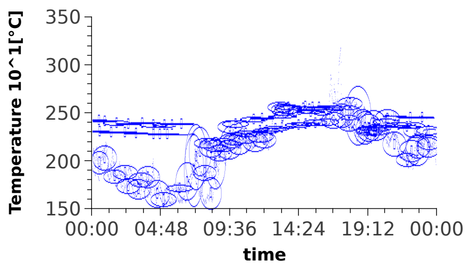
(b) Characteristic Model (Day 1)



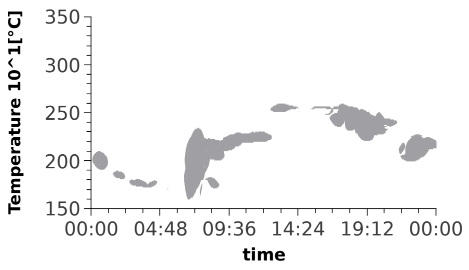
(c) Redundancy (Day 1)



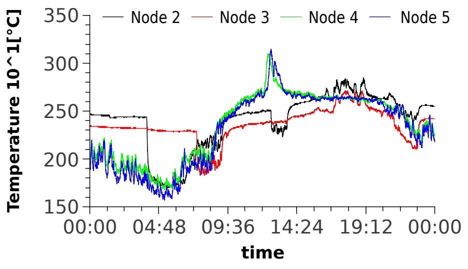
(d) Temperature (Day 2)



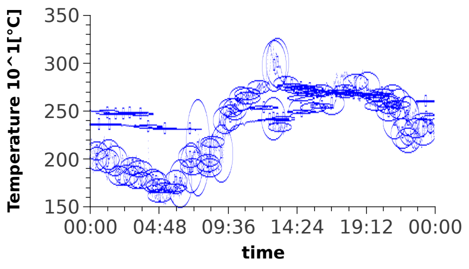
(e) Characteristic Model (Day 2)



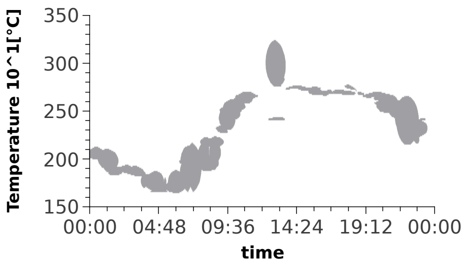
(f) Redundancy (Day 2)



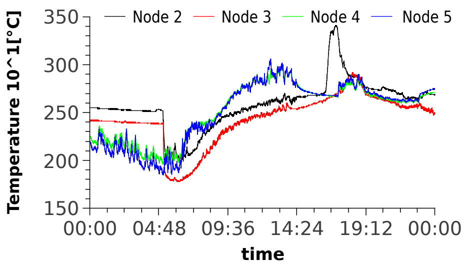
(g) Temperature (Day 3)



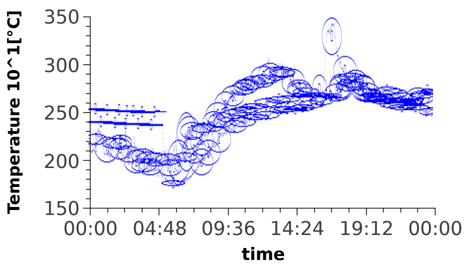
(h) Characteristic Model (Day 3)



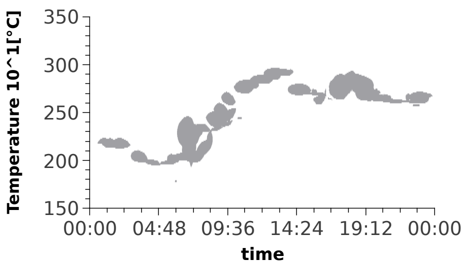
(i) Redundancy (Day 3)



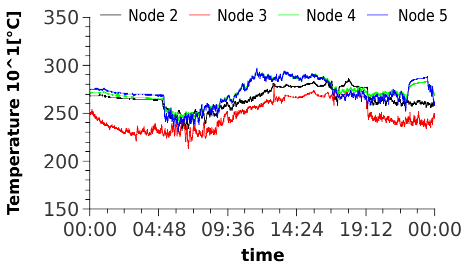
(j) Temperature (Day 4)



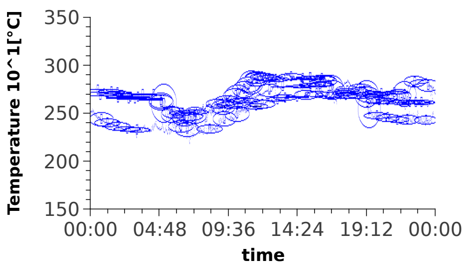
(k) Characteristic Model (Day 4)



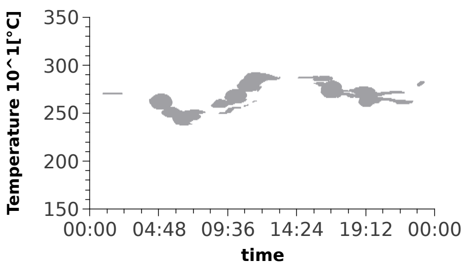
(l) Redundancy (Day 4)



(m) Temperature (Day 5)



(n) Characteristic Model (Day 5)



(o) Redundancy (Day 5)

Figure 7. Five consecutive days of temperature measurement, classification and redundancy analysis.

B. Redundancy

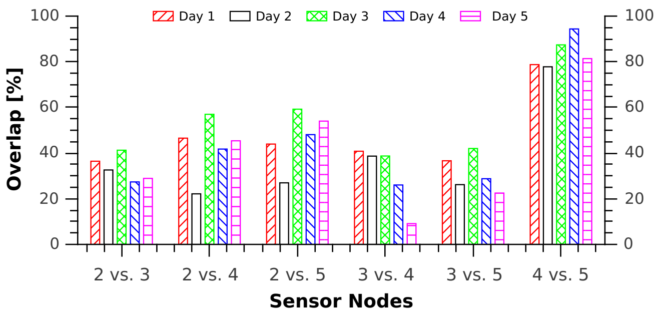


Figure 8. Redundancy between nodes is indicated by their percent overlap.

From five different days of measurement the characteristic models of the sensor nodes were compared to each other. The redundancy between the nodes was extracted as described in Section VI. As already mentioned the nodes 4 and 5 were placed close to each other to show the effects of redundant sensing. Figure 8 depicts the percent overlap of the characteristic models of the nodes. The similar environmental conditions and intersections between the temperature profiles lead to a remarkable redundancy also between disjunct nodes. However, the obviously redundant nodes 4 and 5 show a significant conformity.

VIII. Conclusion

In this paper we proposed a characteristic model for sensor data of potentially unreliable WSN nodes. Based on a classification using k-means clustering algorithm, this model describes the nominal behaviour of every node statistically through covariance-ellipses. Due to a continuous update of the ellipses, the characteristic model is more flexible and can be used to detect outliers or off-nominal behaviour which is essential for unreliable WSNs Moreover, it is more efficient in handling a huge amount of sensed data as the characteristic model represents the whole dataset.
When matching several of the characteristic models, potential redundant sensing can be detected. Thus, redundant nodes can lower their duty-cycle by alternating their sensing.
To test the derived methods we implemented a server application and performed a small test with a real WSN. The results showed that characteristic models of the nodes can be derived with a low computational overhead. In addition, intentionally redundant nodes show many overlapping regions when comparing this models with each other.

However, beside the proposed general functionality and documentation of continuous online clustering: When comparing the models of two consecutive days, the similarity between these days is remarkable. Hence, another advantage for WSNs applications can be achieved by maintaining the charateristic model of every sensor node. The models could help to predict further system states by taking the trend of temperature into account. Thus, some kind of local weather forecast for every sensor node can be implemented to avoid unstable states or a breakdown of the network structure.

References

[1] C. A. Boano, M. Zúñiga, J. Brown, U. Roedig, C. Keppitiyagama, and K. Römer, "Templab: a testbed infrastructure to study the impact of temperature on wireless sensor networks," in *Proceedings of the 13th international symposium on Information processing in sensor networks*. IEEE Press, 2014, pp. 95–106.

[2] K. Bannister, G. Giorgetti, and S. Gupta, "Wireless sensor networking for hot applications: Effects of temperature on signal strength, data collection and localization," in *Proceedings of the 5th Workshop on Embedded Networked Sensors (HotEmNets' 08)*, 2008.

[3] U. Kulau, F. Busching, and L. Wolf, "Undervolting in wsns—a feasibility analysis," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 553–558.

[4] L. Guang-Hui, Z. Jun, and W. Zhi, "Research on forest fire detection based on wireless sensor network," in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 1, 2006, pp. 275–279.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999. [Online]. Available: http://doi.acm.org/10.1145/331499.331504

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent robotics and autonomous agents series. Mit Press, 2005.

[7] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational geometry*. Springer, 2000.

[8] W. Zhang, H.-I. Yang, H. yi Jiang, and C. Chang, "Automatic data clustering analysis of arbitrary shape with k-means and enhanced ant-based template mechanism," in *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, July 2012, pp. 452–460.

[9] M. Younis, M. Youssef, and K. Arisha, "Energy-aware routing in cluster-based sensor networks," in *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*. IEEE, 2002, pp. 129–136.

[10] G. Gupta and M. Younis, "Fault-tolerant clustering of wireless sensor networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 3. IEEE, 2003, pp. 1579–1584.

[11] Y. Liu, J. Gao, L. Zhu, and Y. Zhang, "A clustering algorithm based on communication facility in wsn," in *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, vol. 2, Jan 2009, pp. 76–80.

[12] J. Yuan and H. Chen, "The optimized clustering technique based on spatial-correlation in wireless sensor networks," in *Information, Computing and Telecommunication, 2009. YC-ICT '09. IEEE Youth Conference on*, Sept 2009, pp. 411–414.

[13] F. Yuan, Y. Zhan, and Y. Wang, "Data density correlation degree clustering method for data aggregation in wsn," *Sensors Journal, IEEE*, vol. 14, no. 4, pp. 1089–1098, April 2014.

[14] K. Niu, F. Zhao, and X. Qiao, "An outlier detection algorithm in wireless sensor network based on clustering," in *Communication Technology (ICCT), 2013 15th IEEE International Conference on*, Nov 2013, pp. 433–437.

[15] S.-Y. Jiang and A. min Yang, "Framework of clustering-based outlier detection," in *Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on*, vol. 1, Aug 2009, pp. 475–479.

[16] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *WALCOM: Algorithms and Computation*. Springer Berlin Heidelberg, 2009, vol. 5431, pp. 274–285.

[17] I. Jolliffe, *Principal component analysis*, ser. Springer series in statistics. Springer-Verlag, 1986. [Online]. Available: http://books.google.de/books?id=cN5UAAAAYAAJ