J. Parallel Distrib. Comput. 74 (2014) 1833–1847

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Hyperspherical cluster based distributed anomaly detection in wireless sensor networks☆

Sutharshan Rajasegarar [a,*], Christopher Leckie [b], Marimuthu Palaniswami [a]

[a] Department of Electrical and Electronic Engineering, The University of Melbourne, Victoria 3010, Australia
[b] Department of Computing and Information Systems, The University of Melbourne, Victoria 3010, Australia

HIGHLIGHTS

- Detecting anomalies in data is challenging on resource constrained networks.
- We propose a distributed algorithm using hyperspherical cluster based data models.
- The scheme is capable of identifying global anomalies at an individual node level.
- Comparable detection accuracy with significant reduction in communication overhead.
- Implemented and demonstrated on a real wireless sensor network test-bed.

ARTICLE INFO

Article history:
Received 25 July 2011
Received in revised form
27 June 2013
Accepted 13 September 2013
Available online 23 September 2013

Keywords:
Distributed processing
Wireless sensor networks
Anomaly detection

ABSTRACT

This article describes a distributed hyperspherical cluster based algorithm for identifying anomalies in measurements from a wireless sensor network, and an implementation on a real wireless sensor network testbed. The communication overhead incurred in the network is minimised by clustering sensor measurements and merging clusters before sending a compact description of the clusters to other nodes. An evaluation on several real and synthetic datasets demonstrates that the distributed hyperspherical cluster-based scheme achieves comparable detection accuracy with a significant reduction in communication overhead compared to a centralised scheme, where all the sensor node measurements are communicated to a central node for processing.

1. Introduction

Wireless sensor networks that are deployed for monitoring purposes aim to identify interesting events that have occurred in the area under observation. Furthermore, the sensor nodes are resource constrained, in terms of their energy, memory and computational capabilities. An important factor concerning the radio communication in sensor nodes is that it constitutes the majority of the power consumption in sensor networks [24,8]. For example, the ratio between communication and computation energy consumption per bit ranges from 10^3 to 10^4 in Sensoria sensors and Berkeley motes [45]. Moreover, Marthur et al. have observed that using the parallel NAND Flash technology reduces energy consumption, and enables large storage capacities for sensor nodes [20]. Therefore, it is advantageous to perform in-network processing at the nodes and reduce communication overhead in the network in order to prolong the lifetime of the wireless sensor network.

For robust and reliable functioning of the network, it is essential to identify and mitigate any misbehaviours in the network in an accurate and timely manner. Misbehaviour in sensor networks can occur due to faulty sensors, gradual drift in the sensor elements, loss of calibration, noisy sensor transducers, movement of sensors, changes in observed phenomena or malicious attacks such as denial of service attacks [22]. A key challenge in wireless sensor networks is to identify any misbehaviours or interesting events (*anomalies*) with minimal communication overhead within the network while achieving high detection accuracy.

An *anomaly* (or *outlier*) in a set of data is defined by Barnett et al. [2] as "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data". In

☆ We acknowledge the support from Australian Research Council (ARC) Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP); REDUCE project grant (EP/I000232/1) under the Digital Economy Programme run by Research Councils UK - a cross council initiative led by EPSRC and contributed to by AHRC, ESRC and MRC; the ARC Linkage project grant (LP120100529) and the ARC Linkage Infrastructure, Equipment and Facilities scheme (LIEF) grant (LE120100129)..

* Corresponding author.
E-mail addresses: sraja@unimelb.edu.au (S. Rajasegarar), caleckie@unimelb.edu.au (C. Leckie), palani@unimelb.edu.au (M. Palaniswami).

0743-7315/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jpdc.2013.09.005

sensor networks, anomalies can be at the level of individual measurements with respect to the other measurements at the same sensor node, or at the level of the measurements of one node with respect to other sensor nodes in the network. Henceforth, we have identified three categories of anomalies in sensor networks [26,29]. First are the anomalies in individual measurements or network traffic attributes at *a* sensor node, i.e., some observations at a node are anomalous with respect to the rest of the data. Second, all of the data at a sensor node may be anomalous with respect to its neighbouring nodes. In this case that sensor node will be identified as an anomalous node. Third, a set of sensor nodes in the network can be anomalous. These three types of anomalies are called first, second and third order anomalies. In this paper, we propose a distributed scheme to detect the first and second order anomalies.

The approach proposed in this paper is a distributed, non-parametric anomaly detection algorithm that identifies anomalous measurements at nodes. It uses data clustering to model the data at each node. Data clustering (hereafter called clustering) is the process of finding groups of similar data points, such that each group of data points is well separated [12]. In sensor networks, measurements or traffic data collected by the sensor nodes can be *clustered* by identifying groups of similar measurements in the data. Here *similarity* means the closeness of data vectors to each other. Then the anomaly detection can be performed on the clustered data to classify data vectors as either normal or anomalous. Rather than communicating raw sensor measurements to a central node for analysis, the data at each node are clustered using hyperspherical clusters. Sensor nodes then report cluster summaries, rather than individual measurements to other nodes in the network. Intermediate sensor nodes then merge the cluster summaries before communicating with other nodes. This distributed scheme minimises the communication overhead, which is a major source of energy consumption for wireless sensor networks. Previous attempts at distributed data clustering in sensor networks such as [1,18] have not considered co-operation between nodes for anomaly detection.

The effectiveness of our proposed approach is demonstrated using several real and synthetic datasets. Comparison to a centralised approach shows that this distributed approach can achieve significant reductions in communication overhead, while achieving comparable detection accuracy. Further, we compared our detection scheme with other existing schemes in the literature. Also, we provide a mechanism to select the parameter settings of the algorithm such that an appropriate trade-off between the detection accuracy and the communication overhead can be found. Moreover, our distributed algorithm is implemented on a real wireless sensor network consisting of SunSPOT nodes.

2. Related work

Anomaly or outlier detection has long been a research topic in the data mining and machine learning communities. Several alternative definitions of anomalies have been proposed in the literature, as well as a variety of detection algorithms. Several surveys of these algorithms can be obtained from [7,13,19,23,2]. Furthermore, data aggregation has received considerable attention in the sensor network community and several algorithms [25,32,35] have been proposed in the literature. However, anomaly detection that utilises such in-network processing capability is still an open research challenge.

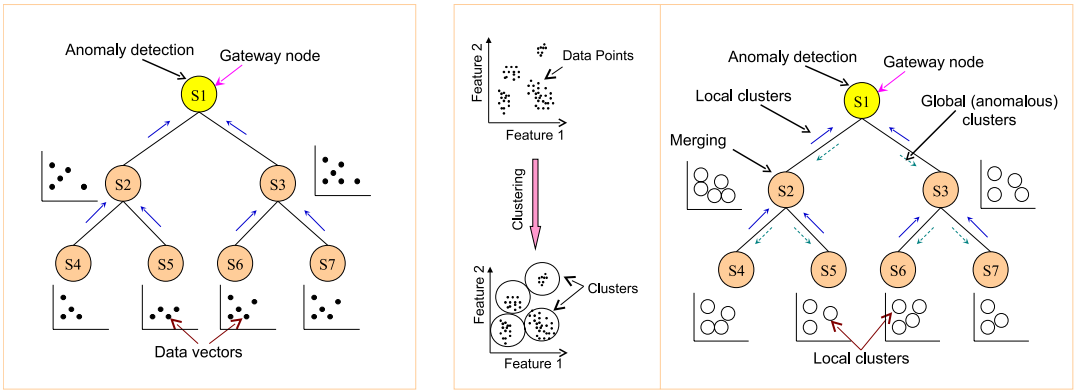
Anomaly or outlier detection mechanisms used in sensor nodes to detect anomalies can be categorised into three general approaches depending on the type of background knowledge of the data that is available [13]. The first approach finds outliers without prior knowledge of the underlying data. This approach uses unsupervised learning or clustering, and assumes that the outliers are well separated from the data points that are normal. The second approach uses supervised classification, where a classifier is trained with labelled data, i.e., the training data is marked as normal or abnormal. Then the trained classifier can be used to classify new data as either normal or abnormal. This approach requires the classifier to be retrained if the characteristics of normal and abnormal data changes in the system. The third approach is the novelty detection, which is analogous to semi-supervised recognition. Here a classifier learns a succinct generalisation of a given set of data, which can then be used to recognise anomalies. This approach can incrementally learn the normal model as and when data is available, so that it can adapt to changes in the distribution of the data.

We can also categorise the anomaly detection techniques in terms of the type of model that they learn, e.g., a parametric or non-parametric model. In parametric techniques, the underlying assumption is that the density distribution of the data points that are being analysed for anomalies is known a priori, e.g., a Gaussian distribution. Anomaly detection techniques that do not assume any prior knowledge about the distribution of the data are called non-parametric anomaly detection techniques. These techniques are suitable for resource constrained sensor networks where the data distribution may change frequently. For example, these changes can be caused by energy depletion of sensors over the lifetime of the network, which affects the stability of the routing topology, and can thus affect the anomaly-based intrusion detection. Changes or uncertainty about the type of monitored environment can also affect the distribution of measurement values. Below we look in more detail at some of the recent non-parametric approaches that have been proposed for sensor networks, namely, clustering and nearest neighbour based approaches. Further details about the anomaly detection in wireless sensor network can be seen in [29,28].

Data clustering is a process of finding groups of similar data points such that each group of data points is well separated [12]. In this approach the data are first clustered and then anomaly detection is performed using those clusters. In [18], a cluster based technique is used to detect routing attacks in sensor networks. In this approach, each sensor node monitors the routing messages that it receives. At regular intervals, each sensor node characterises the set of routing records it has seen in terms of a vector of features, such as the number of route requests, and the average hop count to the base station. Each vector of features can be viewed as a point in a multidimensional feature space. It is assumed that attack traffic occurs far less frequently than normal traffic, and that it is statistically different to normal traffic samples, hence they appear as anomalies in the feature space.

The anomaly detection process comprises two phases. The first phase is the training phase, where a training dataset (with normal and abnormal traffic) is used to model the distribution of the traffic using clusters. The clustering algorithm used here is called fixed-width clustering. This algorithm creates hyperspherical clusters of fixed radius for a dataset. Then the clusters are labelled as normal if they contain more than a given fraction of the total data points, or are otherwise labelled as anomalous. If any new traffic data points fall outside the normal clusters, they are labelled as anomalous. It is demonstrated that this algorithm effectively detects sinkhole attacks in sensor networks. An issue that was identified for further work was how to introduce collaboration between sensors when making a decision about the anomalies. In [1], a distributed *k*-means clustering algorithm is proposed for peer to peer sensor network environments. Clusters are formed at a sensor node using cluster statistics communicated by neighbouring nodes, but are not used for anomaly detection.

Nearest neighbour based approaches use distances among data vectors as the similarity metric. Examples of such metrics include the distance to a nearest neighbour data vector (NN), the distance



(a) Centralised approach. (b) Distributed approach.

Fig. 1. (a) Centralised approach (b) Distributed approach: (i) Clusters formed at each node, (ii) Distributed detection.

to the kth nearest neighbour data vector (kNNM), and the distance to the average of the k nearest data vectors (Average kNN). Note that k is a user defined parameter. These similarity measures are used to sort the data vectors and to classify them as normal or anomalous.

Branch et al. [4] proposed a distributed unsupervised anomaly detection algorithm to identify the top-n outliers in sensor network data. They considered a sensor network topology where all the nodes can communicate with each other in the network. They used several distance metrics such as nearest neighbour distance and average k-nearest neighbour distance to compute the similarity between data vectors at every node. Then the identified top-n anomalies and a subset of data, called a support set, is broadcast to all its neighbour nodes. After several iterations of communication among the nodes, each node identifies global anomalies. This involves multiple rounds of raw data communication in the network.

Zhang et al. [44] proposed an improved framework for the above scheme by Branch et al. [4] for detecting the top-n outliers using the k-nearest neighbour approach. In this approach, all the nodes are arranged in an aggregation tree topology, where a parent–child relationship exists. In order to find global anomalies in the network, it uses three phases iteratively, namely: *commit*, *disseminate* and *verify*.

During the first *commit* phase each node identifies the top-n anomalies on their local data. It then sends the identified anomalies and a subset of data, called a support set, to its intermediate parent node. The intermediate parent node recomputes the top-n anomalies and the support set considering all the received information from its children. Intermediate parent nodes then forward this information to their parent. This process continues until the sink node (the top-most parent node of the aggregation tree) is reached. The sink node uses all the received information from its children and computes the global anomalies, and their deviation rank, which is a metric used to measure how far each anomaly is deviating from its closest data in the support set.

During the *disseminate* phase, the sink node then broadcasts the global anomalies and the deviation rank to all its children along the tree for verification by its children. Then, during the *verify* phase, each node verifies the received global anomalies from the sink node with their local data by using the deviation rank metric. If it finds that it has another set of data vectors in its local data that may modify the global results, it then sends those data vectors and the corresponding deviation rank to the sink node to compute the global anomalies again. After several iterations of the commit, disseminate and verify phases, all the nodes will agree on a global result for the anomalies. This scheme incurs lower communication overhead than the algorithm proposed by Branch et al. [4], even though it involves some form of raw data communication along the tree.

In [30] a cluster based distributed scheme is proposed for anomaly detection. Here the data vectors are collected at individual nodes and clustered using hyperspherical clusters. These clusters are communicated to a gateway node along the hierarchy. Anomaly detection is performed at the gateway node to identify anomalous clusters at the gateway node. However, anomalous data vectors are not identified at each node level. Furthermore, the clusters are merged at the intermediate parent node level along the hierarchy using a single level merging operation. This can be further improved to reduce the communication overhead in the network. In this paper, we propose a distributed anomaly detection scheme using a hyperspherical clustering algorithm and k-nearest neighbour scheme to collaboratively detect anomalies in wireless sensor network data. Further, the anomalous data vectors are identified at each node level, and no raw data measurements are communicated in the network. A recursive cluster merging operation is used to merge the maximum number of clusters at the intermediate parent node level, consequently reducing the communication overhead in the network. The proposed scheme is implemented and evaluated on a real wireless sensor network testbed using SunSpot nodes.

Potential applications of our proposed anomaly detection scheme include identifying inefficient energy consumption behaviour in a indoor (work) environment [33,34], monitoring the health of coral reefs such as the Great Barrier Reef, Australia [10], Internet of Things (IoT) [11] applications such as Smart City monitoring for noise, pollution and environmental parameters in Melbourne, Australia [15,27,16] and SmartSantander, Spain [36]. Below we explain our proposed scheme in detail.

3. Network architecture and problem statement

We consider a set of sensor nodes $S = \{s_j : j = 1 \ldots s\}$ having a hierarchical topology as shown in Fig. 1. The sensors are deployed in a homogeneous environment, in which the measurements taken have the same unknown distribution. All the sensor nodes are time synchronised. Note that, there are several time synchronisation algorithms [38] available for sensor networks that can be utilised for this purpose. At every time interval Δ_i each sensor node s_j measures a data vector x_i^j (we sometimes refer to x_i^j as a measurement). Each data vector is composed of attributes x_{ik}^j, where $x_i^j = \{x_{ik}^j : k = 1 \ldots p\}$ and $x_i^j \in \Re^p$. After a window of n measurements, each sensor s_j has collected a set of measurements $X_j = \{x_i^j : i = 1 \ldots n\}$.

The aim is to find *local* and *global* anomalies in the data measurements collected by the nodes in the network. Local anomalies are anomalous measurements in a sensor nodes own (local) data measurements, where only the measurements at the same node are used as a basis for comparison. However, we are also interested

in cases where the majority of measurements at a sensor node are anomalous in comparison to other nodes in the network. These global anomalies $O \subset X$ are anomalous measurements in the union of the measurements collected from multiple sensor nodes in the network $X = \bigcup_{j=1..s} X_j$. Local anomalies can be detected by considering local measurements of a sensor node without incurring any energy intensive communication overhead in the network. However, detecting global anomalies requires all the measurements from multiple nodes to be considered. Below we describe the approaches to detect the local and global anomalies in detail.

4. Anomaly detection approaches

4.1. Centralised approach

A simple approach to detect the global anomalies is to use a centralised approach. In this approach, at the end of every time window of measurements, each sensor node s_j sends all its data to its gateway node $s_g \in S$. The gateway node s_g combines its own data X_g with the received dataset $X_R = \bigcup_{j=1\ldots s-1, j\neq g} X_j$ to form a combined dataset $X = X_R \cup X_g$. A clustering algorithm is run on X to find a set of clusters $C = \{c_r : r = 1 \ldots c\}$. The clustering algorithm used here is based on fixed-width clustering, used by Eskin et al. [9] for anomaly detection. Euclidean distance is used as the dissimilarity measure between pairs of data [12]. Once the clusters are formed, outlier clusters (anomalous clusters) are classified using an anomaly detection algorithm. This algorithm identifies the anomalous clusters using the K nearest neighbour distance between clusters (refer to Section 4.2.3).

Fig. 1(a) shows an example of the centralised approach for a multi-level hierarchical topology. The data vectors at each node $S2$, $S3$, $S4$, $S5$, $S6$ and $S7$ are transmitted to the gateway node $S1$. The combined data vectors at $S1$ are used for clustering the data. Finally, the anomaly detection algorithm is run at node $S1$ on those clusters. However, the centralised approach has several drawbacks, including its high communication overhead and processing delay at the central node. Hence, we propose a distributed anomaly detection scheme that overcomes these limitations by minimising the communication overhead requirement in the network.

4.2. Distributed approach

In the distributed approach, the anomaly detection process is distributed to all sensors in the network. First, we give the steps involved in the distributed approach and then, in the following subsections, we explain the main functions in the steps in detail.

In the distributed approach, at every time window of n measurements the following operations are performed.

Step 1 Each sensor $s_j \in S$ performs the clustering operation (see Section 4.2.1) on its data X_j and produces clusters $C_j = \{c_r^j : r = 1 \ldots \rho_j\}$.

Note that, the number of clusters ρ_j is determined algorithmically. Also, a unique identifier (ID) is assigned to each of the clusters produced at that node. This unique ID not only identifies the cluster uniquely, but also identifies the sensor node from which it is generated. This can be easily achieved by associating the cluster ID with the unique node ID. We assume that a localisation protocol used in the network assigns unique node IDs to each node.

In order to detect the *local* anomalous data vectors, the anomaly detection algorithm (see Section 4.2.3) can be applied to the clusters C_j to identify the *local* anomalous clusters, and subsequently the local anomalous data vectors in the node. In order to detect the *global* anomalies, the following need to be performed.

Step 2 Sensor s_j sends a *summary* of its clusters (including its unique ID) to its immediate parent $s_u = Parent(s_j)$, i.e., each sensor sends a tuple $\langle LS_{jr}, n_{jr}, ID_{jr} \rangle$, where LS_{jr} is the linear sum of the data vectors in cluster c_r^j, n_{jr} is the number of data vectors in cluster c_r^j, and ID_{jr} is the unique cluster ID of cluster c_r^j.

Each (hyperspherical) cluster $c_r^j \in C_j$ can be completely represented by its centroid and the number of data vectors it contains [43]. Let the number of data vectors in c_r^j be $n_{jr} \leq n$ and the set of data vectors contained in c_r^j be $X_r^j = \{x_q^j : q = 1 \ldots n_{jr}\}$. Then the linear sum of the data vectors of that cluster can be defined as $LS_{jr} = \sum_{q=1}^{n_{jr}} x_q^j$. Hence, the centroid of c_r^j is LS_{jr}/n_{jr}. Thus, c_r^j is *summarised* by n_{jr} and the linear sum of the data vectors LS_{jr} of that cluster.

Step 3 The parent node s_u combines its clusters C_u with the clusters $C = \bigcup_{j \in children(s_u)} C_j$ from its immediate children and forms a combined set of clusters $C_\vartheta = C \cup C_u$.

Step 4 The parent node s_u merges the combined cluster set C_ϑ to produce a merged cluster set $C_h = \{c_r^h : r = 1 \ldots e\}$, where $e \leq |C_\vartheta|$.

See Section 4.2.2 for the cluster merging algorithm. Also note that whenever a pair of similar clusters c_1 and c_2 are merged to produce a merged cluster c_3, a new cluster ID is assigned for c_3. In addition a new field called ⟨Merged cluster IDs⟩ is also added. This field contains a combination of cluster and node IDs of c_1 and c_2. Hence, if the cluster c_3 is identified as anomalous, then the Merged cluster IDs would reveal from which clusters and from which nodes this merged cluster c_3 has been produced.

Step 5 Then the parent node s_u sends the summaries of C_h to its immediate parent.

Step 6 This process continues recursively up to the gateway node $s_g \in S$, where an anomaly detection algorithm is applied to its merged clusters C_g to identify the anomalous cluster set $C_a \subset C_g$ (Section 4.2.3).

Step 7 Summary of anomalous global clusters C_a are communicated to all the nodes in the network.

Step 8 Each node s_j, using the anomalous global cluster C_a, identifies the corresponding globally anomalous data vectors at the node.

Fig. 1(b) shows an example of our distributed approach for a multi-level hierarchical topology with leaf nodes $S4$, $S5$, $S6$ and $S7$, intermediate parent nodes $S2$ and $S3$ and gateway node $S1$. Initially, all sensor nodes perform the clustering operation on their own data. Nodes $S4$ and $S5$ send cluster summaries to their parent node $S2$. Nodes $S6$ and $S7$ send cluster summaries to their parent node $S3$. Nodes $S2$ and $S3$ combine the received summaries from their respective children with their own clusters, and then produce merged clusters C_h. Nodes $S2$ and $S3$ send their cluster summaries to the gateway node $S1$. Finally, node $S1$ merges the clusters from its children and its own clusters producing a merged cluster set C_g. The anomaly detection algorithm is run on the C_g to identify the globally anomalous cluster set C_a. Then the globally anomalous cluster summaries (C_a) are communicated back to all the sensor nodes. Each node from $S2$ to $S7$ uses C_a to identify the globally anomalous data vectors at their node.

In this approach, globally anomalous data vectors are identified at each node level. Further, better load balancing is achieved in the network by distributing the clustering process between all the nodes. Also the communication overhead is reduced by only sending cluster summaries instead of (all of) the raw data. This helps to extend the lifetime of the network. Note that, in the centralised case, the gateway node has complete information about the data in the network, whereas in the distributed case, the gateway node only has the merged cluster information. Therefore, there may be a slight reduction in detection accuracy for the distributed case compared to the centralised case.

In general, the computation of the global clusters can be performed at any parent node (or at any level) of the hierarchy. For example, in Fig. 1(b), the global model computed at the parent node $S3$ will consider the clusters from its children $S6$ and $S7$ only. Then, $S6$ and $S7$ will perform global detection using the global clusters sent by $S3$. In this case, the region considered for distributed detection is the region covered by the nodes $S3$, $S6$ and $S7$. If the global clusters are computed at the top most parent node, i.e., the base station $S1$, then it will consider the cluster information from all the children from $S2$ to $S7$. Here, the region considered for distributed detection will be all of the nodes in the topology. This method of distribution over a hierarchical topology provides flexibility to the user in selecting the coverage region for distributed computation.

Moreover, the distributed detection approach is not limited to hierarchical topologies, such as the Tenet architecture [21]. It is applicable to any network topology where a set of sensors can communicate with their neighbours in the network. In this case, any sensor node in a connected group of sensors can be selected as a leader node for performing the global clustering and the anomaly detection of the clusters. This also gives flexibility for the leader node to select or ignore any participating neighbours for the computation of the global clustering. This provides robustness against faulty or malicious nodes in the network.

In addition to detecting interesting events in the data collected from environmental monitoring applications, the proposed approach can be used for detecting some types of malicious attacks caused by adversaries, such as routing attacks in sensor networks. Periodic route error attacks, active sinkhole attacks and passive sinkhole attacks are some of the examples of routing attacks that can be detected using this scheme in a similar way to [18]. Moreover, alternative routing trees can be constructed to generate alternative ways of aggregating the data. If an inconsistency is found between these alternative aggregation results, then the presence of a malicious node can potentially be detected. A detailed investigation of such an approach is beyond the scope of this paper, and is a direction for the future research.

The following sections describe the fixed-width clustering algorithm, merging algorithm and the anomaly detection algorithm in detail.

4.2.1. Fixed-width clustering algorithm

The clustering algorithm used here is based on the fixed-width clustering algorithm used in [9,18]. Fixed-width clustering creates a set of hyperspherical clusters of fixed radius (width) w. The width w is a parameter specified by the user. The procedure begins by randomly choosing any data point as the centroid (centre) of the first cluster with radius w. In the general step, the Euclidean distance is computed between the centroids of the current clusters to the next remaining data vector. If the distance to the closest cluster centre from a data vector is *less* than the radius w, the data vector is added to that cluster and the centroid is updated. If the distance to the closest cluster centre is *more* than the radius w, then a new cluster is formed with that data vector as its centroid. This procedure is shown in Algorithm 1.

The principle advantage of this simple approach is that only one pass is required through the data vectors, thus minimising storage, computation and energy consumption. This efficiency is traded against the loss of flexibility and possible accuracy engendered by using a single threshold (w) to determine all the clusters. Another disadvantage of this scheme is the (implicit) assumption that clusters in the input space are hyperspherical in shape.

4.2.2. Merging clusters

Two clusters are merged when they are *similar*. We define a pair of clusters c_1 and c_2 to be *similar* if the inter-cluster distance $d(c_1, c_2)$ between their centres is less than a given merging threshold τ ($\leq w$). If c_1 and c_2 are similar, then a new cluster c_3

Algorithm 1 Fixed-width Clustering

Require: Cluster width w and Dataset $X = \{x_i : i = 1...n\}$
  $C \Leftarrow \oslash$ {empty cluster set C}
  $N_c \Leftarrow 0$ {Number of clusters}

  Create the first cluster c_1 with the centre as x_1
  $C \Leftarrow c_1$
  $N_c \Leftarrow 1$
  for $i = 2$ to n **do**
    Compute Euclidean distance d_i between x_i and the *closest* cluster $c_r \in C$ with centre m_r
    if $d_i \leq w$ **then**
      $c_r \Leftarrow x_i$ {Add x_i to c_r}
      $m_r \Leftarrow$ mean of data vectors in c_r
    else
      Create a new cluster c_υ with x_i as the centre
      $C \Leftarrow c_\upsilon$
      $N_c \Leftarrow N_c + 1$
    end if
  end for
  Output: Cluster set $C = \{c_r : r = 1...N_c\}$

Algorithm 2 Merge Clusters

Require: Cluster set $C = \{c_r : r = 1...N_c\}$, merging threshold τ and cluster width w.
  $C_m \Leftarrow \oslash$ {*empty merged cluster set* C_m}
  $C_m \Leftarrow C$ {*copy all the clusters from* C *to* C_m}
  MergeClusters(C_m) {*recursively merges clusters*}
  Output: Merged cluster set C_m

  Procedure *MergeClusters*(C_m)
  $l = |C_m|$ {*store initial size of* C_m}
  $C_u \Leftarrow \oslash$ {*empty merged cluster set* C_u}
  for $r = 1$ to N_c **do**
    if c_r is not merged **then**
      IsMergingDone = false
      for $j = r + 1$ to N_c **do**
        if c_j is not merged **then**
          Compute Euclidean distance D_r between c_r and c_j
          if $D_r \leq \tau$ **then**
            $n_v \Leftarrow n_r + n_j$ {n_r *and* n_j *are number of data vectors in clusters* c_r *and* c_j *respectively*}
            $m_v \Leftarrow \frac{1}{n_v}(LS_r + LS_j)$ {LS_r *and* LS_j *are linear sum of data vectors of clusters* c_r *and* c_j *respectively*}
            Create a merged cluster c_v using m_v and n_v {m_v *is the cluster centre and* n_v *is the number of data vectors*}
            $C_u \Leftarrow c_v$
            Mark c_r and c_j as merged
            IsMergingDone = true
            break
          end if
        end if
      end for
      if IsMergingDone == false **then**
        $C_u \Leftarrow c_r$ {*add* c_r *to merged cluster set* C_u}
      end if
    end if
  end for
  if $l < |C_u|$ **then**
    MergeClusters(C_u)
  end if

is produced whose centre is the mean of the data vectors in the clusters with centre c_1 and c_2 and whose number of data vectors is the sum of those in c_1 and c_2. This merging procedure is performed recursively until there is no more merging possible (Refer to Fig. 2). The merging algorithm is shown in Algorithm 2.

When two clusters are merged into a new cluster with fixed width w, a small error can be introduced due to finding the new cluster while keeping the cluster width fixed. When the merging operation is performed recursively until there are no more clusters to merge, this error would start to accumulate. This error can be reduced by setting a small value for τ, i.e., merge only if the pair of clusters are very close to each other. However, the trade-off is the increased communication overhead in the network. In the future, the merging algorithm and the clustering algorithm will be improved to accommodate clusters with varying widths, however, with an increased computational overhead.

4.2.3. Anomaly detection algorithm

The anomaly detection algorithm classifies *clusters* as either normal or anomalous. The average inter-cluster distance of the K nearest neighbour (KNN) clusters is used [31] to identify anomalous clusters. The algorithm is as follows:

- for each cluster c_β in the cluster set C, a set of inter-cluster distances $D_{c_\beta} = \{d(c_\beta, c_\gamma) : \gamma = 1 \ldots (|C| - 1), \gamma \neq \beta\}$ is computed. Here $d(c_\beta, c_\gamma)$ is the Euclidean distance between the centroids of c_β and c_γ, and $|C|$ is the number of clusters in the cluster set C.
- among the set of inter-cluster distances D_{c_β} for cluster c_β, the shortest K (parameter of KNN) distances are selected and using those, the *average inter-cluster distance* ICD_β of cluster c_β is computed as follows,

$$ICD_\beta = \begin{cases} \dfrac{1}{K} \displaystyle\sum_{\gamma=1,\neq\beta}^{K} d(c_\beta, c_\gamma) & K \leq |C| - 1 \\ \dfrac{1}{|C| - 1} \displaystyle\sum_{\gamma=1,\neq\beta}^{|C|-1} d(c_\beta, c_\gamma) & K > |C| - 1. \end{cases}$$

  Our average inter-cluster distance computation differs from the one proposed by Chan et al. [6] in the following way. Chan et al. used the whole cluster set C to compute the average inter-cluster distance ICD_β for a cluster c_β, whereas here only the K nearest neighbour clusters of c_β are used to compute the average inter-cluster distance ICD_β. The advantage of this approach is that clusters at the edge of a dense region are not overly penalised compared to the clusters in the centre of the region.
- a cluster is identified as anomalous if its average inter-cluster distance ICD_β is more than ψ number of standard deviations of the inter-cluster distance $SD(ICD)$ from the mean inter-cluster distance $AVG(ICD)$, i.e., the set of anomalous clusters $C_a \subset C$ is defined as

$$C_a = \{c_\beta \in C | ICD_\beta > AVG(ICD) + \psi \times SD(ICD)\} \quad (1)$$

  where ICD is the set of average inter-cluster distances.

Fig. 3 shows an example of normal and anomalous clusters.

4.3. Complexity analysis

The centralised approach requires memory for storing data measurements of $O(np)$ at each node and $O(snp)$ at the gateway node, where s is the number of sensor nodes in the network, p is the number of dimensions in a data vector, and n is the number of data vectors at a sensor node. The computational overhead is only for the gateway node and its complexity is $O(snN_c)$, where $N_c (\ll n)$ is the number of clusters. The communication complexity per link (i.e., for communicating between a pair of nodes) is $O(np)$.

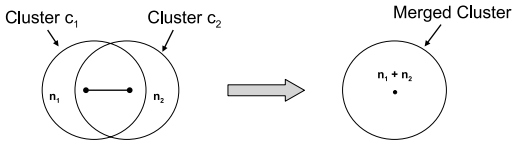


Fig. 2. Cluster merging, where n_1 and n_2 are the number of data vectors in each cluster c_1 and c_2 respectively.

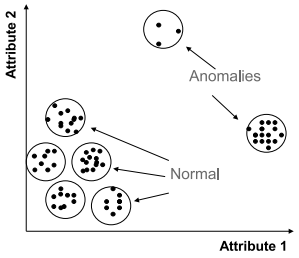


Fig. 3. Example of normal and anomalous clusters.

Table 1
Comparison of the complexities at a sensor node for the distributed and centralised scheme, where, n is the number of data vectors at node, p is the dimension of a data vector, N_c is the number of clusters at a node ($p, N_c \ll n$), N_a is the number of global anomalous clusters ($N_a \ll n$), and s is the number of sensors in a network.

Scheme	Complexity		
	Computation (per node)	Memory (per node)	Communication (per node)
Distributed	$O(nN_c)$	$O(N_cp + np)$	$O((N_c+N_a)(p+2))$
Centralised	$O(snN_c)$ (at central node)	$O(snp)$ (at central node)	$O(np)$

The complexity for the distributed approach is as follows.

The *fixed-width clustering algorithm* requires a single pass over the data at each node. For each data vector, it computes the distance to each existing cluster. Hence the computational complexity is $O(nN_c)$. The memory complexity for each sensor node is $O(N_cp)$, since each cluster requires a fixed length record.

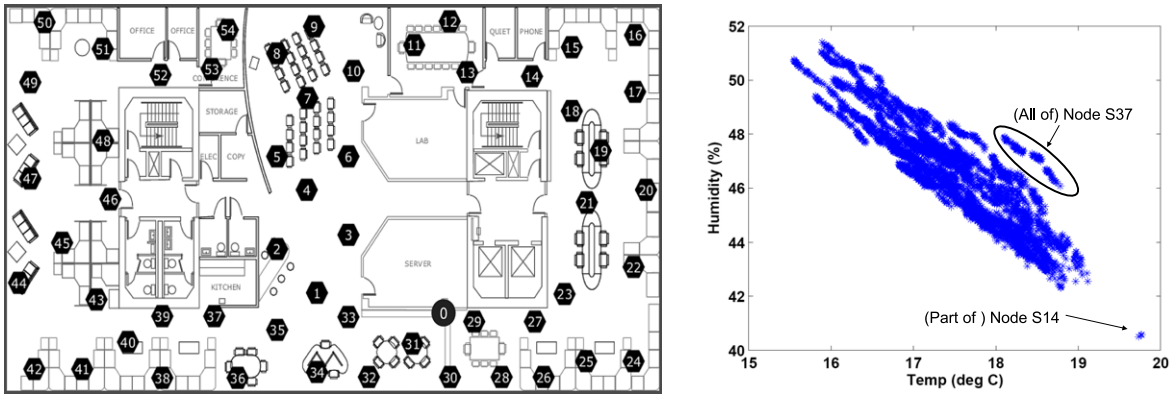
The *cluster merging* operation compares the cluster pairs recursively in ς iterations with computational complexity of $O(N_c^2\varsigma)$. The anomaly detection algorithm compares each cluster to all other clusters in the cluster set C to find the K nearest neighbours with computational complexity $O(N_c^2)$.

In the *distributed algorithm*, each node j, for each cluster c_r^j, needs to communicate the linear sum of data vectors LS_{jr}, the number of data vectors in the cluster n_{jr}, and the cluster ID ID_{jr} between nodes. Further, after the global anomalous clusters are found at the central node, N_a number of anomalous clusters are sent back to the nodes. Therefore, the communication complexity per link is $O(N_cp + 2N_c + N_ap + 2N_p)$.

In summary, in the distributed anomaly detection algorithm, each sensor node requires memory to keep N_c clusters with maximum complexity of $O(N_cp)$ and memory for data measurements $O(np)$. Each sensor incurs a maximum $O(nN_c)$ computational complexity and a maximum $O((N_c + N_a)(p + 2))$ communication complexity. Table 1 provides a summary of the complexity per sensor node.

5. Evaluation

The aim of this evaluation is to compare the performance of our proposed cluster-based distributed anomaly detection approach with the centralised approach and other existing schemes in the literature. Furthermore, we aim to provide a mechanism to select the



(a) Sensor node locations. (b) IBRL data.

Fig. 4. (a) Sensor node locations in the IBRL deployment. Node locations are shown in black with their corresponding node IDs. Node 0 is the gateway node. (b) Scatter plot of IBRL data at the centralised node S_H.
Source: Reproduced from [14]).

parameter settings of the algorithm such that a suitable trade-off between the detection accuracy and the communication overheads is found. Moreover, we aim to investigate whether our proposed algorithm can be applied in a real life scenario by implementing on a network consisting of real wireless sensor nodes.

We use two real sensor network deployment datasets and two synthetic datasets for evaluation purposes, namely the *IBRL*, *GDI*, *Banana* and *Gaussmix* datasets. We now describe each dataset in detail.

5.1. Methodology

The *IBRL* data is a publicly available set of sensor measurements from a wireless sensor network consisting of 55 *Mica2Dot* sensor nodes (including a gateway node), which was deployed in the Intel Berkeley Research Laboratory (IBRL) in Berkeley, California, USA [5,14]. This deployment is an example of indoor environmental monitoring using sensor networks. The sensors collect five measurements at 31 s intervals: temperature in degrees Celsius, light in Lux, temperature corrected relative humidity as a percentage, voltage in volts and topology information. Fig. 4(a) shows the sensor deployment in the laboratory. Sensor nodes are labelled from 0 to 54, where Node 0 is the gateway node, which only transmits data to a host computer and does not collect any sensor measurements. The sensors collected measurements for a period between 28th February 2004 and 5th April 2004 [14]. All the measurements were communicated to the gateway node via multiple hops. We consider a four hour time window of measurements collected on 1st March 2004 during the time interval from 00:00 am to 03.59 am. During this window, nodes 5 and 15 did not contain any data. For ease of visualisation, we focus on two types of features: temperature and humidity. Since the original data did not contain any labels as to which data is normal and anomalous, we use scatter plots to visually identify and label them as normal and anomalous using the following procedure. We first make a visual inspection of the observed data to highlight the presence of any apparent anomalies in the data. Fig. 4(b) is the scatter plot of the combined measurements from all 54 nodes. In this plot, two apparent anomalies can be observed. One possible anomaly is a small collection of data vectors in the lower right hand corner of the plot that differ significantly from the mass of the data. This anomalous data constitutes a part of the data from node 14. Second, there are three visual "islands" of data coming from node 37 (whole data) which lie "above" the mass of the data. We label these visually apparent anomalous data vectors as *anomalies*, and the rest of the data vectors as *normal* for our evaluation purposes. We formed a single level hierarchical topology with node 1 as the gateway node and the others (node 2 to node 54) as leaf nodes. Note that we omit the node 0 (the gateway node), node 5 and node 15 which did not contain any data (see Fig. 4(a)) in our evaluation.

The other real dataset that we used was the sensor measurements collected at the Great Duck Island Project [40]. The network was deployed for monitoring the habitat of a sea bird called the *Leach's Storm Petrel* [41]. This is an example of an outdoor environmental monitoring deployment. In every five minute interval, each sensor recorded light, temperature, humidity, and pressure readings. Seven sensor nodes are selected for the evaluation, namely nodes 101, 109, 111, 116, 118, 122 and 123 over a 24 h period on 1st July 2003. Three features were used: humidity, temperature and pressure readings from each sensor node. 3D Scatter plots were made for the data at each node, and each dataset was manually cleaned to remove overly excessive values for the attributes in the data, such as negative values for the humidity readings. The cleaned data were labelled as *Normal* and used for the evaluation. A three level hierarchical topology as shown in Fig. 1 was formed by using node 101 as the gateway node, nodes 109 and 111 as the intermediate parent nodes, and the other four nodes as leaf nodes.

We generated two types of anomalous data to use with this *normal* GDI dataset. First, a uniformly distributed, randomly generated set of anomalous data (of 20 data vectors for each node) were introduced into the tails of the distribution of each feature for *two* of the nodes (nodes 118 and 123). These introduced anomalous data measurements were labelled as *Anomalies*, and we refer to this dataset as *GDI–ISO*, which stands for *GDI data with isolated anomalies*. The second type of anomalous data is a randomly generated set of anomalous data added to the *normal* GDI data at each node. The anomalous data for each attribute comprised a set of vectors drawn from a uniform distribution over the *normal* measurements of each attribute. The number of introduced anomalous data were 10% of the normal measurements and were labelled as *Anomalies*. We refer to this dataset as *GDI–UNI*, which stands for *GDI data with uniform anomalies*. Both GDI–ISO and GDI–UNI were normalised to the range [0, 1]. The scatter plots of the combined data vectors from all of the seven nodes are shown in Fig. 5. The first synthetic data used was a *Banana* dataset [42], where the *normal* and the *anomalous* data is distributed in the shape of a banana. A single level hierarchical network topology consisting of 15 nodes was used for the evaluation. Each node comprises 100 data vectors of two dimensions, each with 10% of anomalous data. This *Banana* data set (including the anomalous data) was generated using the *gendatb* function from DDtools [42]. Fig. 6(a) shows a scatter plot of the Banana dataset, which is the combined data vectors from all of the 15 nodes. The second synthetic dataset is called *Gaussmix*. The Gaussmix dataset was similar to the one used by Subramaniam et al. [37]. This consists of two features, each generated

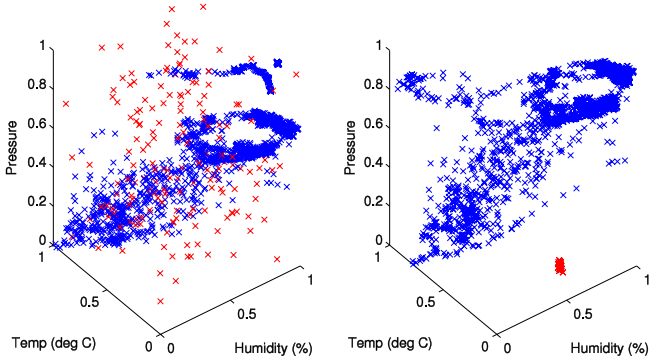


Fig. 5. Scatter plots of GDI–UNI (left) and GDI–ISO (right) datasets. Blue crosses (x) represent normal data and red stars (*) represent anomalous data. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

Table 2
Dataset description.

Dataset type	Dataset name	Network topology (levels of hierarchy)	Number of dimensions	Number of nodes
Real	IBRL	Single	2	52
	GDI–ISO	Three	3	7
	GDI–UNI	Three	3	7
Synthetic	Banana	Single	2	15
	Gaussmix	Single	2	15

from a mixture of Gaussian distributions with means randomly selected from (0.3, 0.35, 0.45) and with a standard deviation of 0.03. Uniformly distributed noise (*anomalies*) ranging between [0.50, 1] was introduced to each feature of the data. The data for 15 sensor nodes were created including 5% noise data at every node. Each node comprises 105 data vectors. The whole dataset is normalised to the range [0, 1]. A single level hierarchical network topology was formed using these nodes.

Table 2 gives a summary of the datasets used in our evaluation.

5.2. Results

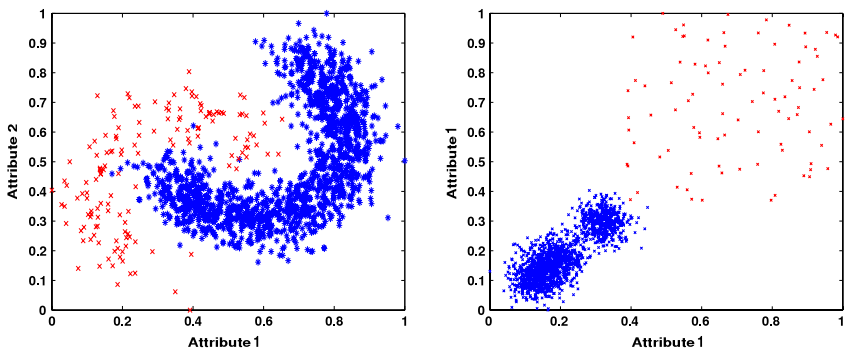
The centralised and distributed cluster-based anomaly detection algorithms were implemented in C++. All the datasets were normalised by subtracting from the mean and dividing by the standard deviation before use in the evaluation. First we evaluate the distributed and centralised scheme using the *GDI–ISO* dataset, by changing one of the parameters (w or K) at a time while the other is fixed. The cluster width value w used ranged from 0.02 to 2.02 in 0.02 intervals, and the KNN parameter value K ranged from 1 to 51. We used $\tau = w$, and $\psi = 1$ in our evaluations. For each simulation, the false positive and false negative rates were calculated.

A *false positive* occurs when a normal measurement is identified as anomalous by an algorithm, and a *false negative* occurs when an anomalous measurement is identified as normal. The *false positive rate* (FPR) is the ratio between false positives and the actual normal measurements, and the *false negative rate* (FNR) is the ratio between false negatives and the actual anomalous measurements. The *detection rate* (DR) is given by $DR = 100 - FNR$. Also during the evaluation, the number of data vectors and the number of clusters communicated were recorded as a measure of the reduction in communication overhead. Here, results are reported for the distributed and centralised detection at the top most parent node (gateway node) of the network topology.

Fig. 7(a) and (b) shows the graphs of the false positive and the detection rates as a function of the cluster width for the centralised and distributed schemes. The KNN parameter is fixed at $k = 3$, and the trends seen in these two graphs are very similar. The lower cluster widths (e.g., $w = 0.02$ for the centralised scheme) yield higher false positive rates because at lower cluster widths, a larger number of clusters are produced (some of the clusters may even be singletons). Consequently, there is a greater chance of the anomalous data being divided into multiple clusters in close proximity. This results in the average inter-cluster distances being smaller for the anomalous data, rendering them undetectable.

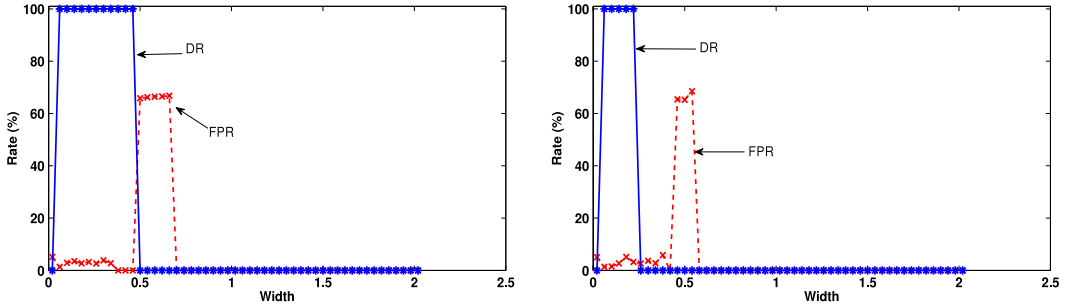
Similarly, at higher cluster widths (>0.7 for the centralised scheme) the detection rate again decreases while the false positive rate becomes zero. This is because, at larger cluster widths, a small number of large clusters are produced. Hence there is a higher chance of the anomalous data being included in the larger normal clusters. The range of cluster widths at which the system performs better is $w \in [0.06, 0.46]$ for the centralised scheme and $w \in [0.06, 0.22]$ for the distributed scheme. This shows that the cluster width (w) is an important parameter for achieving better detection performance. In practice a good cluster width can be selected by training the system before deployment and then periodically adjusting w based on the detection accuracy and phenomenal changes in the environment of the network.

Fig. 8(a) and (b) shows the graphs of the false positive and detection rates as a function of the KNN parameter for the centralised and distributed schemes. Here the cluster width is fixed at $w = 0.18$. These graphs show there is a threshold for the KNN parameter beyond which detection performance begins to diminish. In this evaluation, the threshold KNN value for the centralised scheme is 10, and for the distributed scheme is 3, i.e., for KNN values ≤ 10 (for the centralised scheme), false positive rates are much lower. This shows that the KNN value (K) is also an important parameter for achieving better detection performance. In practice a good KNN value can be selected by training the system before deployment. Furthermore, the number of global clusters produced after the merging operation for the distributed scheme is 15 (see Fig. 9(a)). From the empirical results, the KNN parameter may be



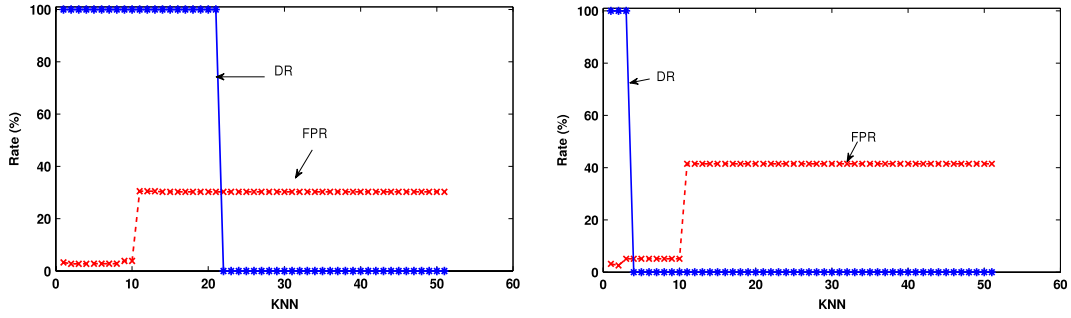
(a) Banana. (b) Gaussmix.

Fig. 6. Scatter plots of Banana and Gaussmix datasets. The blue stars (*) represent the normal vectors and the red crosses (x) represent the anomalous vectors.



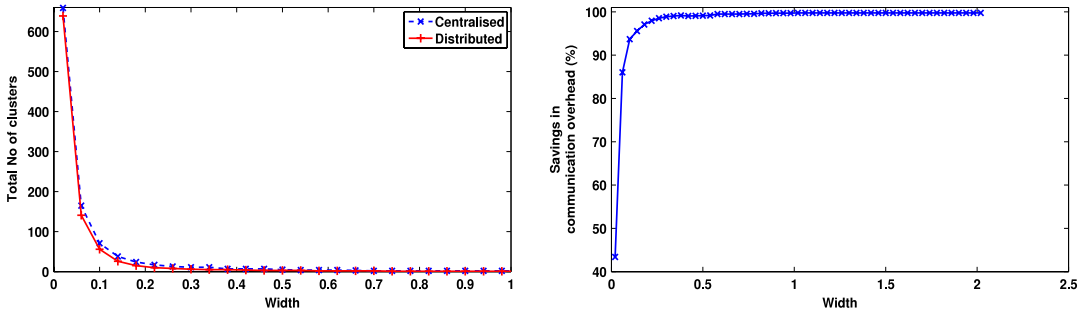
(a) Centralised scheme. (b) Distributed scheme.

Fig. 7. Detection rate (DR) and False positive rate (FPR) (%) with Cluster width (w) for GDI–ISO dataset.



(a) Centralised scheme. (b) Distributed scheme.

Fig. 8. Detection rate (DR) and False positive rate (FPR) (%) with KNN parameter (k) for GDI–ISO dataset.



(a) Number of clusters. (b) Savings in communication overhead.

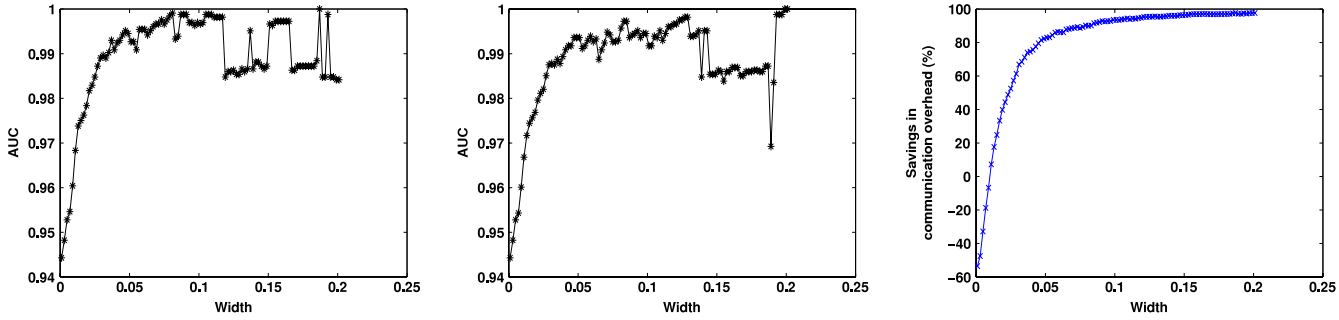
Fig. 9. (a) Number of clusters produced for the centralised and distributed scenarios with cluster width (w). (b) Percentage reduction in the communication overhead in the network vs Cluster width (w). The reduction in the communication overhead in the network is calculated as (the number of data vectors transmitted in the centralised case − the number of clusters transmitted in the distributed case)/(the number of data vectors transmitted in the centralised case).

selected as a percentage of the number of clusters produced in the scheme (e.g., 20% of the final number of clusters produced by the distributed detection algorithm).

When comparing the distributed and the centralised schemes in Figs. 7 and 8, it can be noticed that a higher detection performance is observed for the larger range of cluster widths and a larger range of KNN values for the centralised scheme compared to the distributed scheme. This is because the number of clusters produced for the centralised scheme (24) is larger than the number of clusters produced in the distributed scheme (15) at cluster width 0.18 (see Fig. 9(a)). Note that in the distributed scheme, the cluster merging process is performed to combine similar clusters in order to minimise the communication overhead in the network. Hence the useful window of values for the cluster width and the KNN parameters are smaller for the distributed scheme compared to the centralised scheme.

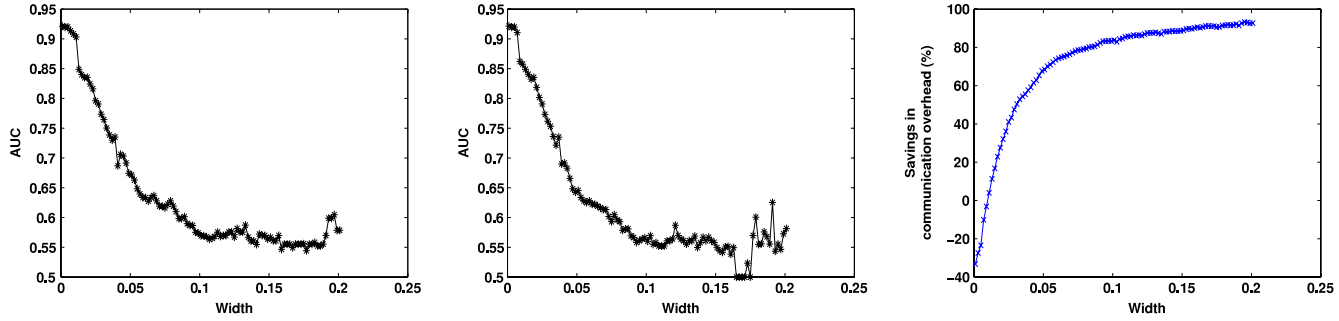
The average false positive rate for the distributed and the centralised case is about 3% (refer to Fig. 7). From this it follows that the distributed and centralised schemes achieve roughly the same accuracy. Moreover, a significant saving in communication overhead is achieved by the distributed approach compared to the centralised approach. Fig. 9(b) shows that the distributed approach, when compared to the centralised case (with $K = 3$), realises savings in communication overhead that ranges from 86.04% to 97.95% over the cluster width range 0.06 to 0.22.

Next, we compare the detection performance using the other datasets. We performed simulations for the following parameters. Both distributed and centralised algorithms were evaluated for different cluster width values w ranging from 0.001 to 0.201 in 0.002 intervals, and for different KNN parameter values K ranging from 1 to 25. In each simulation, the following measures are recorded for each (w, K) pair: the FPR and DR. *Receiver operating characteristic* (ROC) curves, which are DR vs FPR curves, are drawn by varying one of the parameters while the other is fixed. The *area under the ROC curve* (AUC) is computed while varying these parameters. The closer the AUC value is to 1, the better the classifier performance is. AUC values of 0.5 and lower indicate that the performance of a classifier is poorer than random guessing. Here, results are reported for the distributed and centralised detection at the top most parent node (gateway node) of the network topology.



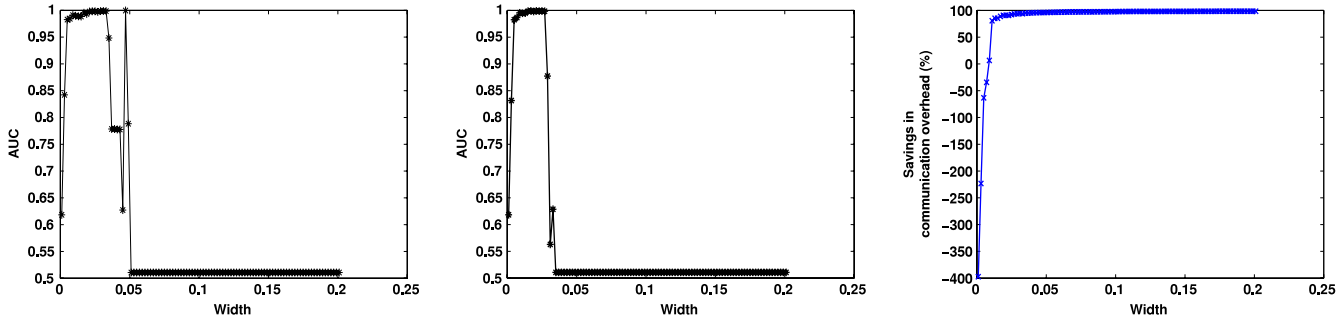
(a) Centralised detection. (b) Distributed detection. (c) Savings in communication overhead.

Fig. 10. Graphs for the area under the ROC curve (AUC) with cluster width (w), and the percentage savings in communication overhead for the centralised and distributed detection algorithm on the GDI–ISO data.



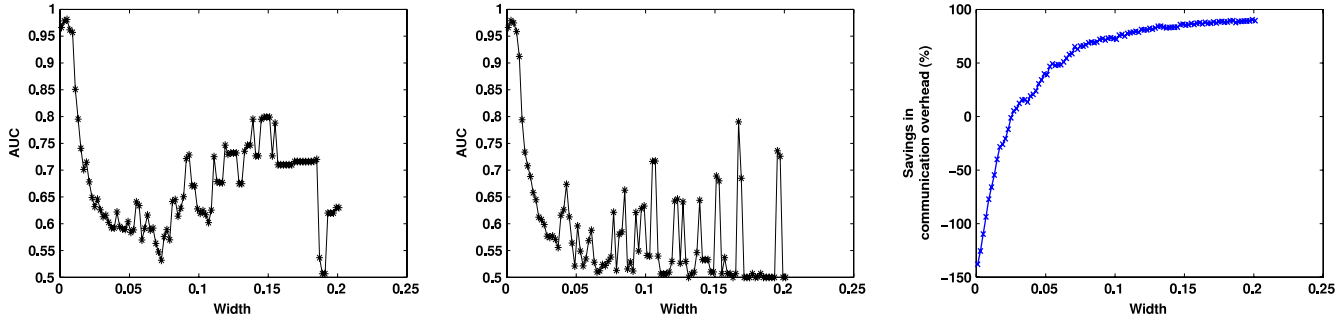
(a) Centralised detection. (b) Distributed detection. (c) Savings in communication overhead.

Fig. 11. Graphs for the area under the ROC curve (AUC) with cluster width (w), and the percentage savings in communication overhead for the centralised and distributed detection algorithm on the GDI–UNI data.



(a) Centralised detection. (b) Distributed detection. (c) Savings in communication overhead.

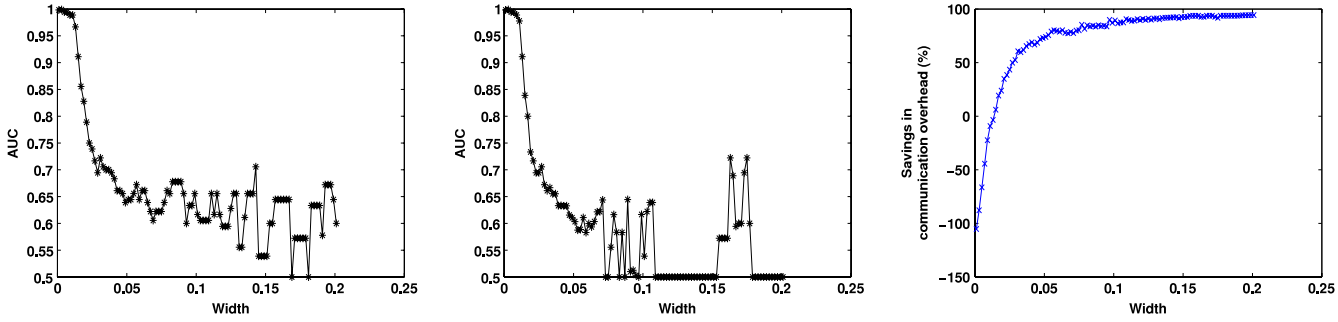
Fig. 12. Graphs for the area under the ROC curve (AUC) with cluster width (w), and the percentage savings in communication overhead for the centralised and distributed detection algorithm on the IBRL data.



(a) Centralised detection. (b) Distributed detection. (c) Savings in communication overhead.

Fig. 13. Graphs for the area under the ROC curve (AUC) with cluster width (w), and the percentage savings in communication overhead for the centralised and distributed detection algorithm on the Banana data.

Figs. 10–14 show graphs for all five datasets separately. When the AUC values obtained for the centralised and distributed schemes are compared for each dataset, they reveal that the distributed algorithm generally achieves a comparable detection



(a) Centralised detection. (b) Distributed detection. (c) Savings in communication overhead.

Fig. 14. Graphs for the area under the ROC curve (AUC) with cluster width (w), and the percentage savings in communication overhead for the centralised and distributed detection algorithm on the Gaussmix data.

accuracy to the centralised scheme with significant savings in communication overhead. Overall, the savings in communication overhead depend on the cluster width parameter, which affects the number of clusters generated.

In the figures showing the percentage savings in communication overhead (SCO) vs (cluster) width graphs (see Figs. 10(c), 11(c), 12(c), 13(c) and 14(c)), negative values for the SCO can be observed for smaller cluster widths. The reason is as follows. In the centralised scheme, the data vectors from each node are communicated to the gateway node once. Whereas, in the distributed scheme, there are two communication overheads incurred. One, when each sensor node sends their data clusters to gateway node up along the hierarchy. Second, when the gateway node sends the global anomalous clusters to all the sensor nodes down along the hierarchy. Therefore, if the cluster width is small, there will be a larger number of clusters produced in each node (sometimes singleton clusters are produced for very small cluster widths). Hence, in total, the number of clusters communicated up and down along the hierarchy for the distributed scheme becomes more than the number of raw measurements communicated up along the hierarchy (only once) for the centralised scheme. This will result in negative values for the SCO. It is advantageous to operate the distributed scheme with the range of cluster widths that give a positive value for the SCO and more than 0.5 for the AUC values.

The communication savings obtained for the GDI–ISO and IBRL datasets (Figs. 10(c) and 12(c)) are higher than those obtained for the GDI–UNI, Banana, and Gaussmix datasets (refer to Figs. 11(c), 13(c) and 14(c)). This is because, in the GDI–ISO and IBRL datasets, the anomalies lie a larger distance apart from the normal data, and are also *concentrated* in a particular location in the space (for example, see the anomalies shown in Fig. 4(b)). Whereas, for the Gaussmix and GDI–UNI datasets, the anomalies are spread over a much larger space, hence requiring more clusters to capture them. However for the Banana dataset (refer to Fig. 6(a)), the anomalies are not so distant from the normal data. Hence, to achieve a higher detection rate, smaller width clusters are needed to capture the less distant anomalies while ensuring that normal data are not included. This is evident from the AUC plots in Fig. 13(a) and (b), where AUC values of above 0.7 are achieved for cluster width values $w \leq 0.015$ for the centralised and distributed cases. The smaller cluster widths needed for good detection performance results in a larger number of clusters being produced. This is the reason for the lower savings in communication overhead compared to the GDI–ISO and IBRL datasets.

The above evaluation demonstrates that our distributed algorithm is capable of detecting a variety of anomalies with comparable detection accuracy to that of a centralised scheme, but with varying efficiency in terms of communication overhead, depending on how the anomalies are spread and located in the dataset. If the anomalies are distant and concentrated, our scheme achieves very high savings in communication overhead (e.g., 50% savings in communication overhead with an AUC of about 98% in the case of the GDI–ISO dataset). If the anomalies are less concentrated and spread over a large space or if they lie much closer to the normal data, the savings in communication will be less. Furthermore, the above graphs provide a mechanism to choose the right parameter values (e.g., cluster width) such that a given detection accuracy is achieved for a known communication overhead in the network. This is useful in practice as a variety of commercial sensor nodes (sensor networks) exist with varying resources, such as computation and battery life. Hence this gives a guide to select the right parameter settings for a selected sensor network platform, and to achieve the desired detection accuracy with knowledge of the amount of communication overhead that will be incurred in the network.

6. Comparison with other schemes

Next, we compare the effectiveness of our hyperspherical cluster based scheme (HSCBS) with other existing anomaly detection schemes from the literature. Three of the comparison methods search for top N anomalies (in the terminology of [4]) using three different distance metrics: (1) nearest neighbour distance (*NN*) (2) Kth nearest neighbour distance method (*KNNM*) and (3) average of K nearest neighbour distances (*AvgKNN*). The fourth comparison method is a distance based outliers algorithm proposed by Knorr et al. [17], which we denote as *DBO*. In *DBO*, a data vector $\mathbf{x}$ is anomalous if at most a given α number of the data vectors in the dataset X lie within a given distance r from $\mathbf{x}$. The N value for the top-N anomalies is set as a percentage of the number of data vectors. The K value for KNN and AvgKNN are selected after performing a systematic search for each algorithm. All the results are reported for the centralised scenario.

Figs. 5 and 4(b) show the scatterplots of the labelled (data vectors are pre-marked as normal or anomalous) real data of each dataset used in our comparison evaluation. Table 3 provides the detection performance of each of the detection schemes in terms of the detection rate (DR%) and the false positive rate (FPR%) for different datasets, along with the parameter settings (Params) used for each of the schemes. The parameters for each of the algorithms are selected after performing a systematic search for the parameter values that yield the highest detection rate with a lower false positive rate.

For the IBRL dataset, we see that the HSCBS and DBO algorithms successfully detect the anomalies (both first and second order anomalies), but DBO incurs many false positives. This suggests that the proposed method (HSCBS) is more successful than the other methods in detecting these anomalies in the IBRL dataset.

In the GDI–ISO dataset, the anomalies are far apart from the greater mass of the data. Hence, all schemes successfully separate

Table 3
Detection rate (DR%) and false positive rate (FPR%) of various detection schemes for the IBRL, GDI–UNI and GDI–ISO datasets.

Scheme		HSCBS	NN	KNNM	AvgKNN	DBO
IBRL	DR	100	6.43	11.43	6.43	100
	FPR	1.05	5.87	5.87	5.98	2.41
	Params	W = 0.027, KNN = 4	N = 6%	N = 6%, KNN = 10	N = 6%, KNN = 10	$r = 0.1105, \alpha = 282$
GDI–UNI	DR	85.47	87.21	86.62	88.37	92.44
	FPR	1.48	4.27	4.33	4.15	7.29
	Params	W = 0.005, KNN = 4	N = 12%	N = 12%, KNN = 10	N = 12%, KNN = 10	$r = 0.0601, \alpha = 5$
GDI–ISO	DR	100	0	100	100	100
	FPR	1.462	10.11	7.73	7.73	0.24
	Params	W = 0.059, KNN = 4	N = 10%	N = 10%, KNN = 60	N = 10%, KNN = 60	$r = 0.4098, \alpha = 63$

these anomalies, except the NN scheme, which also incurs higher false positives. This suggests that the NN scheme struggles to detect easily identifiable anomalies that are quite isolated.

In the GDI–UNI dataset, the anomalies are uniformly distributed over the greater mass of the data. All schemes successfully separate these anomalies with a higher detection rate. However, the HSCBS method incurs the lowest false positive rate.

The major advantage of the HSCBS algorithm over DBO comes from the reduced communication overhead when used in the distributed model. The DBO scheme is used in a distributed manner in [37]. In their scheme, each sensor performs local anomaly detection using DBO on its local data. Then, all the local anomalies are communicated to the parent node to verify whether they are globally anomalous. This scheme involves communication of raw data vectors between nodes. Further, if the number of anomalies is high, it causes substantial energy consumption in the network due to increased communication activity. Our distributed scheme transmits only a summary information about the hyperspheres (normal model), which is communication efficient. Hence, the distributed HSCBS scheme achieves accuracy comparable to the DBO scheme, but has reduced communication overhead. Moreover, our scheme is capable of identifying both local or global anomalies at the individual node level.

7. Implementation on a real wireless sensor network testbed

In this section, we investigate whether our proposed distributed anomaly detection algorithm can be implemented on a real wireless sensor network.

We implemented our distributed scheme on a real wireless sensor network testbed consisting of SunSPOT [3,39] sensor nodes. SunSPOT (Sun Small Programmable Object Technology) nodes consist of a 180 MHz, 32-bit ARM920T core processor, 512 KB RAM and 4 MB Flash. It uses a TI CC2420 (formerly ChipCon) radio and communicates wirelessly at 2.4 GHz. It runs on a rechargeable 3.7 V Lithium-ion battery. The code is written using Java and executed using the Squawk Virtual Machine on board. Each node is equipped with on-board sensing elements measuring temperature, light and acceleration (3 axes). We used temperature (in degrees Celsius) and light (in Lux) measurements for our experiment.

A multi-hop hierarchical network is created using 10 nodes as shown in Fig. 16(a). The base station node is connected to a computer via Universal Serial Bus (USB). The base station node only provides processing and communication between the computer and the rest of the nodes in the network, and does not participate in collecting any data measurements on-board.

There are two modules implemented in Java. The first module runs on the computer and communicates with the other nodes via the base station node. A graphical user interface (GUI) is implemented as shown in Fig. 15, which provides an interface to communicate with the sensor nodes using messages sent via the wireless medium from the base station node. The GUI provides functionalities to set parameters in the nodes, such as cluster width w, KNN parameter K and merging threshold τ, and execute commands such as collect data into flash, run local clustering, run global clustering and collect labelled data. It also provides graphs to visualise the labelled data vectors and the labelled clusters (local and global) collected from each node in the network.

The second module runs on each node and performs all the processing required for anomaly detection. Each SunSPOT node in the network has been provided with a unique address, which is a 64-bit IEEE extended MAC address expressed as four sets of four-digit hexadecimal. We used the last four digits of the IEEE extended MAC address for the node ID. Each node is assigned a parent node ID and its immediate children node IDs along with the port for communication.

The following functionalities are provided in the implementation. Refer to Fig. 16 for a summary of these functionalities.

- *parameter settings*: parameters required for distributed detection can be set from the GUI, such as cluster width, KNN parameter, sample time, number of records to collect, merge threshold and connection timeout. When an appropriate value for a parameter is entered in the GUI, a message is communicated to all the nodes via the hierarchy and the corresponding parameter value is set at each node. Furthermore, the current parameter values set at each node can also be requested and viewed on the GUI;
- *collect data*: when a collect data command is issued from the GUI, each node samples the measurements at the pre-specified interval (set using parameter settings from the GUI) and saves them in its flash memory along with the date and time;
- *local clusters*: when the local clusters command is issued from the GUI, fixed width clustering is performed at each node on the local data saved in its flash memory using the predefined parameters. Then the anomaly detection algorithm is run on the local clusters to identify and label them as locally normal or anomalous clusters. The locally anomalous data vectors are identified using the anomalous local clusters and the local labels (locally normal or anomalous) are recorded in the flash memory against each data vector together with the Cluster ID. Note that, the Cluster IDs are unique numbers. The labelled local clusters from each node are communicated to the base station for viewing purposes. When communicating the cluster information, each cluster has the ID in a concatenated form as ⟨Node ID:Cluster ID⟩ so that a cluster can be uniquely identified as to which node it belongs;
- *global clusters*: when the global clusters command is issued from the GUI, each node first performs local clustering on the data it has collected and saved in its flash memory. Then the summary of the local clusters are communicated along the hierarchy to the top-most parent node (in this case to node ID 3550; see Fig. 15). Cluster merging is performed at the intermediate parent nodes and at the top-most parent node. The anomaly detection algorithm is run on the global clusters at the top-most parent node. The anomalous global clusters are then communicated back to all the nodes. Each node then uses these

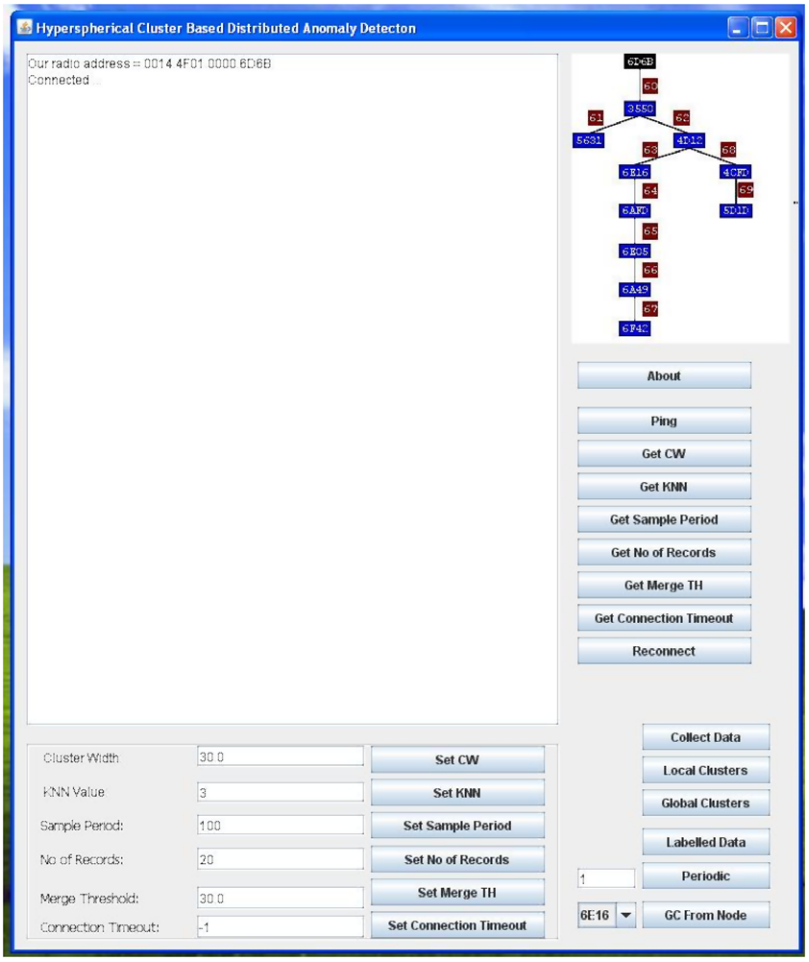
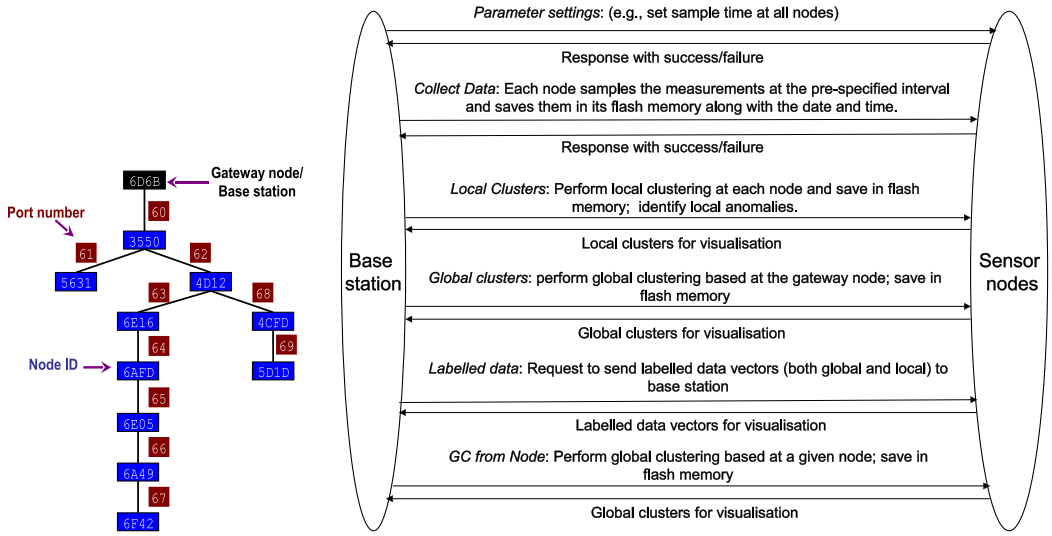


Fig. 15. Graphical user interface of the distributed anomaly detection scheme.



(a) Hierarchy. (b) Functionalities.

Fig. 16. Node hierarchy and the functionalities implemented for the distributed anomaly detection scheme using SunSPOT sensor nodes.

anomalous global clusters to classify and label its data vectors as globally normal or anomalous. The global labels are recorded in the flash memory against each data vector. The labelled global clusters are communicated to the base station from the top-most parent node to the GUI for viewing purposes.

Each data record saved in the flash memory at each node is in the format of ⟨Date/Time, Temperature, Light, ClusterID, Local label, Global label⟩. The data packet format used for communicating each cluster information is of the form ⟨ClusterID, Node ID, Merged Cluster IDs, No of Features, Label, Sum of Data

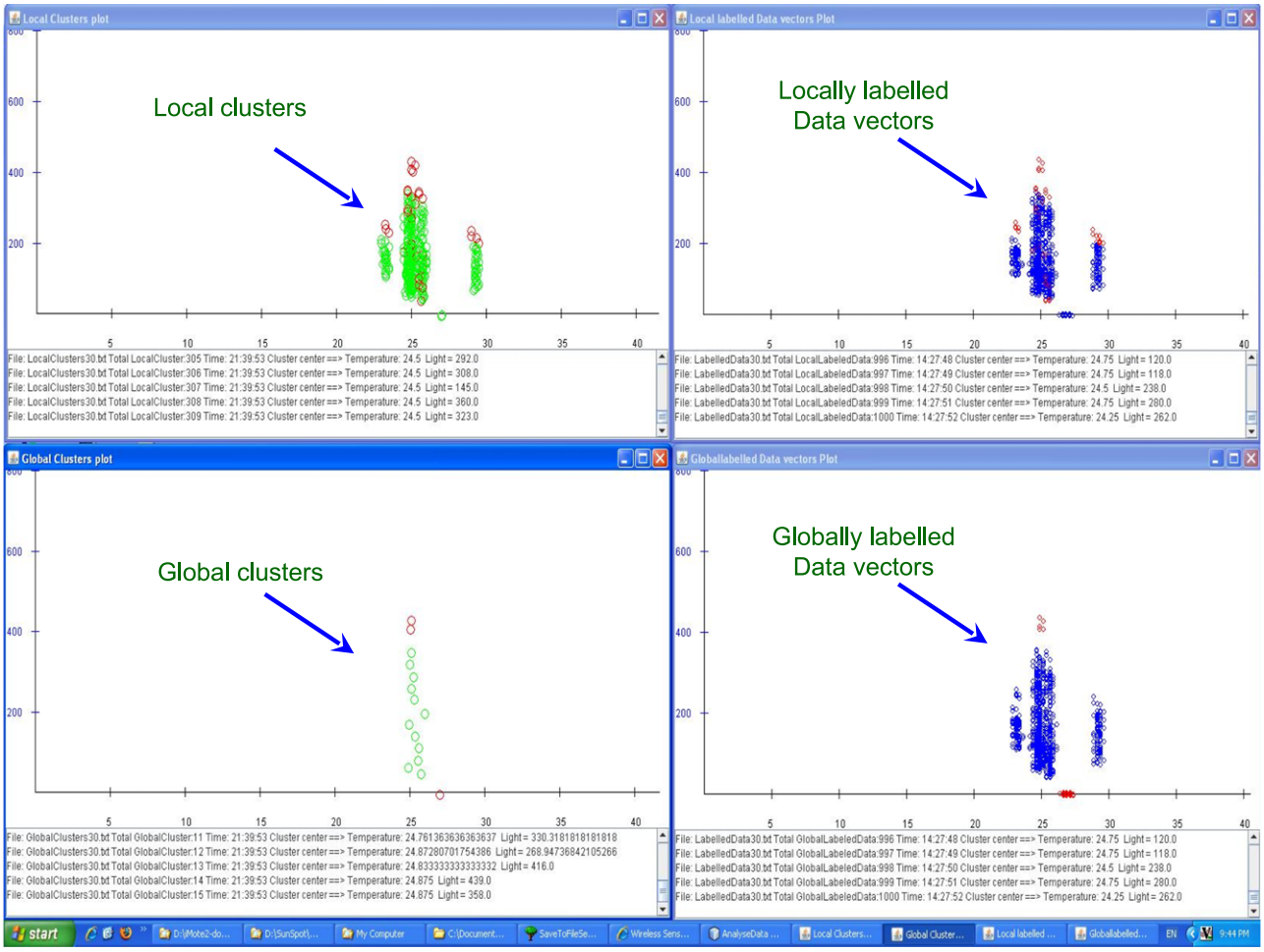


Fig. 17. Screenshots of distributed anomaly detection results. Green and blue circles represent normal clusters (not to scale) and normal data vectors respectively, and red circles represent the anomalies. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Vectors, No of Data Vectors⟩. Note that when clusters are merged recursively at a node, a new merged cluster is produced with a new cluster ID (i.e., its Node ID and Cluster ID). In addition a new field is also added to the merged cluster information called "Merged Cluster IDs", which is of the form ⟨Node ID_1:Cluster ID_1; Node ID_2: Cluster ID_2; Node ID_3: Cluster ID_3; . . .⟩. This new field enables identification of the participating clusters that have been involved in the formation of this new merged cluster. This information will be used by each node to backtrack and identify the corresponding data vectors, so that the global labels of the merged clusters can be assigned to the corresponding data vectors in the flash memory;

- *labelled data*: when the labelled data command is issued from the GUI, the labelled data vectors saved in the flash memory of each sensor node are communicated to the base station and viewed on the GUI;
- *GC from node*: this command from the GUI is used to perform the above explained "Global Clusters" command functionalities *at a specified node level*. Note that, the distributed anomaly detection can be performed at any intermediate parent level. This command enables to specify the node ID at which the "Global Clusters" should be performed;
- *periodic*: this command is provided to iteratively perform the "Collect Data", "Global Clusters" and "Labelled Data" commands in the sequel for a specified number of times. The results from each iteration are saved in a text file for later visualisation purposes.

Fig. 17 provides a graph of the local and global clusters and the corresponding labelled data vectors obtained from the experiment. All 10 nodes were deployed inside a lab (indoor environment). The anomalies were introduced by means of controlling the light falling on to some of the nodes as follows. One of the nodes was covered with a paper to simulate a low lighting condition, and two of the nodes were kept outside the window to simulate high lighting conditions. Note that the clusters shown in the graphs are not to scale. The parameters used for the experiment are as follows: Cluster width: 15.0, KNN Value: 3, Cluster Merge TH: 15.0, Data Sample Period (ms):1000, No of Data records: 100. The code footprint is approximately 50 k bytes. The Global detection is performed at the top most parent node (Node ID 3550). This implementation demonstrates that our proposed anomaly detection scheme can be used in real wireless sensor network hardware with limited resources.

8. Conclusion

In this article, a distributed anomaly detection algorithm based on the data clustering using hyperspherical clusters is presented to identify anomalies in wireless sensor network data. The scheme is capable of identifying global anomalies at an individual node level. The evaluation on several real and synthetic datasets shows that the distributed approach achieves comparable detection performance to a centralised approach, while achieving a significant reduction in communication overhead. Furthermore, our distributed scheme has been implemented on a real wireless sensor network

testbed and demonstrated its functionalities. In the future, more sophisticated clustering methods (which would incur more computational overhead in each node) to capture the pattern of normal data in a node will be considered.

References

[1] S. Bandyopadhyay, C. Gianella, U. Maulik, H. Kargupta, K. Liu, S. Datta, Clustering distributed data streams in peer-to-peer environments, Inf. Sci. 176 (14) (2006) 1952–1985.
[2] V. Barnett, T. Lewis, Outliers in Statistical Data, third ed., John Wiley and Sons, 1994.
[3] BigNet, 2013. http://issnip.unimelb.edu.au/research_program/sensor_networks/infrastructure/bignet_testbed.
[4] J. Branch, B. Szymanski, C. Giannella, R. Wolff, H. Kargupta, In-network outlier detection in wireless sesnor networks, in: Proceedings of the IEEE International Conference on Distributed Computing Systems, ICDCS, 2006, pp. 51–58.
[5] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, S. Madden, TASK: Sensor network in a box, in: Proceedings of the Second European Workshop on Wireless Sensor Networks, 2005, pp. 133–144.
[6] P.K. Chan, M.V. Mahoney, M.H. Arshad, Learning rules and clusters for anomaly detection in network traffic, in: Managing Cyber Threats: Issues, Approaches and Challenges, Vol. 5, Springer, 2005, pp. 81–99.
[7] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, ACM Comput. Surv. 41 (3) (2009) 1–58. http://doi.acm.org/10.1145/1541880.1541882.
[8] L. Doherty, B. Warneke, B. Boser, K. Pister, Energy and performance considerations for smart dust, Int. J. Parallel Distrib. Syst. Netw. 4 (3) (2001) 121–133.
[9] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo, A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data, in: Data Mining for Security Applications, 2002.
[10] Great Barrier Reef, Australia, 2013. http://www.issnip.unimelb.edu.au/research_program/sensor_networks/environmental_monitoring/gbroos.
[11] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): a vision, architectural elements, and future directions, Future Gener. Comput. Syst. 29 (7) (2013) 1645–1660. http://dx.doi.org/10.1016/j.future.2013.01.010.
[12] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, 2001.
[13] V. Hodge, J. Austin, A survey of outlier detection methodologies, Artif. Intell. Rev. (2004) 85–126.
[14] IBRL web, 2011. http://db.lcs.mit.edu/labdata/labdata.html.
[15] Internet of Things, 2013. http://issnip.unimelb.edu.au/research_program/sensor_networks/Internet_of_Things.
[16] J. Jin, J. Gubbi, T. Luo, M. Palaniswami, Network architecture and QoS issues in the internet of things for a smart city, in: International Symposium on Communications and Information Technologies, ISCIT, 2012, pp. 974–979.
[17] E.M. Knorr, R.T. Ng, Algorithms for mining distance based outliers in large datasets, in: Proceedings of the 24th International Conference on Very Large Data Bases, VLDB, 1998, pp. 392–403.
[18] C. Loo, M. Ng, C. Leckie, M. Palaniswami, Intrusion detection for routing attacks in sensor networks, Int. J. Distrib. Sensor Netw. 2 (4) (2006) 313–332.
[19] M. Markos, S. Sameer, Novelty detection: a review part 1: statistical approaches, Signal Process. 83 (12) (2003) 2481–2497.
[20] G. Mathur, P. Desnoyers, D. Ganesan, P. Shenoy, Ultra low power data storage for sensor networks, in: Proceedings of the Fifth international Conference on Information Processing in Sensor Networks, IPSN, USA, 2006, pp. 374–381.
[21] J. Paek, B. Greenstein, O. Gnawali, K.Y. Jang, A. Joki, M. Vieira, et al., The tenet architecture for tiered sensor networks, ACM Trans. Sensor Netw. 6 (2010) 34:1–34:44.
[22] A. Perrig, J. Stankovic, D. Wagner, Security in wireless sensor networks, Commun. ACM 47 (6) (2004) 53–57.
[23] M.I. Petrovskiy, Outlier detection algorithms in data mining systems, Program. Comput. Softw. 29 (4) (2003) 228–237.
[24] V. Raghunathan, C. Schurgers, S. Park, M. Srivastava, Energy-aware wireless microsensor networks, IEEE Signal Process. Mag. 19 (2) (2002) 40–50.
[25] R. Rajagopalan, P. Varshney, Data-aggregation techniques in sensor networks: a survey, IEEE Commun. Surv. Tutor. 8 (4) (2006) 48–63.
[26] S. Rajasegarar, J.C. Bezdek, C. Leckie, M. Palaniswami, Elliptical anomalies in wireless sensor networks, ACM Trans. Sensor Netw. (ACM TOSN) 6 (1) (2009) 28.
[27] S. Rajasegarar, T.C. Havens, S. Karunasekera, C. Leckie, J.C. Bezdek, M. Jamriska, A. Gunatilaka, A. Skvortsov, M. Palaniswami, High-resolution monitoring of atmospheric pollutants using a system of low-cost sensors, IEEE Trans. Geosci. Remote Sens. PP (99) (2013) pp. 1, 10, 0. http://dx.doi.org/10.1109/TGRS.2013.2276431.
[28] S. Rajasegarar, C. Leckie, M. Palaniswami, Anomaly detection in wireless sensor networks, IEEE Wirel. Commun. 15 (4) (2008) 34–40.
[29] S. Rajasegarar, C. Leckie, M. Palaniswami, Detecting data anomalies in sensor networks, in: R. Beyah, J. McNair, C. Corbett (Eds.), Security in Ad-hoc and Sensor Networks, World Scientific Publishing, Inc., 2009, pp. 231–260.
[30] S. Rajasegarar, C. Leckie, M. Palaniswami, J.C. Bezdek, Distributed anomaly detection in wireless sensor networks, in: Proceedings of the IEEE International Conference on Communications Systems, ICCS, Singapore, 2006.
[31] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: Proceedings of the ACM Special Interest Group on Management Of Data, ACM SIGMOD, 2000, pp. 427–438.
[32] S. Ranjani, S. Krishnan, C. Thangaraj, Energy-efficient cluster based data aggregation for wireless sensor networks, in: International Conference on Recent Advances in Computing and Software Systems, RACSS, 2012, pp. 174–179.
[33] REDUCE, 2013. http://info.ee.surrey.ac.uk/CCSR/REDUCE/.
[34] REDUCE network architecture, 2013. http://www.smartsantander.eu/wiki/index.php/Testbeds/Guilford.
[35] S. Roy, M. Conti, S. Setia, S. Jajodia, Secure data aggregation in wireless sensor networks, IEEE Trans. Inf. Forensics Secur. 7 (3) (2012) 1040–1052.
[36] Smart Santander, 2013. http://www.smartsantander.eu/.
[37] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos, Online outlier detection in sensor data using non-parametric models, in: Proceedings of the International Conference on Very Large Data Bases, VLDB, 2006, pp. 187–198.
[38] B. Sundararaman, U. Buy, A.D. Kshemkalyani, Clock synchronization for wireless sensor networks: a survey, Ad Hoc Netw. 3 (3) (2005) 281–323. Elsevier.
[39] SunSPOT, 2013. https://www.sunspotworld.com/.
[40] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, D. Culler, An analysis of a large scale habitat monitoring application, in: Proceedings of the International Conference on Embedded Networked Sensor Systems, SenSys, 2004, pp. 214–226.
[41] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, D. Estrin, Habitat monitoring with sensor networks, Commun. ACM 47 (6) (2004) 34–40.
[42] D.M.J. Tax, DDtools, the data description toolbox for Matlab, version 1.5.4, 2006. http://ict.ewi.tudelft.nl/davidt/dd_tools.html.
[43] T. Zhang, R. Ramakrishnan, M. Livny, Birch: a new data clustering algorithm and its applications, Data Min. Knowl. Discov. 1 (2) (1997) 141–182.
[44] K. Zhang, S. Shi, H. Gao, J. Li, Unsupervised outlier detection in sensor networks using aggregation tree, in: Proceedings of the Advanced Data Mining and Applications, ADMA, 2007, pp. 158–169.
[45] F. Zhao, J. Liu, J. Liu, L. Guibas, J. Reich, Collaborative signal and information processing: an information-directed approach, Proc. IEEE 91 (8) (2003) 1199–1209.

Sutharshan Rajasegarar received his B.Sc. Engineering degree in Electronic and Telecommunication Engineering (with first class honours) in 2002, from the University of Moratuwa, Sri Lanka, and the Ph.D. degree in 2009 from the University of Melbourne, Australia. He is currently a Research Fellow with the Department of Electrical and Electronic Engineering, the University of Melbourne, Australia. His research interests include Internet of Things (IoT), wireless sensor networks, distributed anomaly/outlier detection, machine learning, pattern recognition, signal processing and wireless communication.

Christopher Leckie is a professor at the Department of Computing and Information Systems, the University of Melbourne, Australia. He received the B.Sc. degree in 1985, the B.E. degree in electrical and computer systems engineering (with first class honours) in 1987, and the Ph.D. degree in computer Science in 1992, all from Monash University, Australia. He is currently the deputy director of NICTAVictoria Research Laboratory. His research interests include the scalable data mining, network intrusion detection, wireless sensor networks, artificial intelligence (AI), telecommunications, machine learning, fault diagnosis, distributed systems and design automation.

Marimuthu Palaniswami is a professor at the Department of Electrical and Electronic Engineering, the University of Melbourne, Australia. He received his B.E. (Hons) from the University of Madras, India in 1977, ME from the Indian Institute of science, India in 1979, MEngSc from the University of Melbourne in 1983 and the PhD from the University of Newcastle, Australia in 1987. He currently leads the ARC Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP) programme. His research interests include the Internet of Things (IoT), SVMs, Sensors and Sensor Networks, Machine Learning, Neural Network, Pattern Recognition, Signal Processing and Control.