Tandem: A Context-Aware Method for Spontaneous Clustering of Dynamic Wireless Sensor Nodes

Raluca Marin-Perianu[1], Clemens Lombriser[2],
Paul Havinga[1], Hans Scholten[1], Gerhard Tröster[2]

[1] University of Twente, The Netherlands
[2] Wearable Computing Lab, ETH Zürich

Abstract. Wireless sensor nodes attached to everyday objects and worn by people are able to collaborate and actively assist users in their activities. We propose a method through which wireless sensor nodes organize spontaneously into clusters based on a common context. Provided that the confidence of sharing a common context varies in time, the algorithm takes into account a window-based history of believes. We approximate the behaviour of the algorithm using a Markov chain model and we analyse theoretically the cluster stability. We compare the theoretical results with simulations, by making use of three sets of experimental data reported from field tests. The results show the tradeoff between the time history necessary to achieve a certain stability and the responsiveness of the clustering algorithm.

1 Introduction

Wireless sensor networks, smart everyday objects and cooperative artefacts represent all different facets of the ubiquitous computing vision, where sensor-enabled devices become integrated in the environment and provide context-aware services to the users. Various systems have already been demonstrated to be able to retrieve the context, such as the physical context (e.g. position, movement [4]), the situation (e.g. meeting [16]) and even the emotional context (e.g. mood detection [3]). One step further is to have sensor nodes that interact and use common contextual information for reasoning and taking decisions at the point of action. Such a "networked world" opens perspectives for novel applications in numerous fields, including transport and logistics [9], industrial manufacturing [12], healthcare [7], civil security and disaster management [8].

In this paper we explore a non-traditional networking paradigm, based on context sharing. Previous work showed that sensor nodes can recognize online a common context and build associations of the type "moving together" [6, 9]. Starting from this result, we address the problem of organizing the nodes sharing a common context into stable clusters, given the dynamics of the network and the fact that the accuracy of the context-recognition varies in time.

The contributions of this paper are as follows. Firstly, we propose Tandem, an algorithm for spontaneous clustering of mobile wireless sensor nodes. The

algorithm allows reclustering in case of topological or contextual changes, and tries to achieve stable clusters if there are no changes in the network, by analysing the similarity of the context over a time history. Secondly, we approximate the behaviour of the algorithm using a Markov chain model, which allows us to estimate the percentage of time the clustering structure is correct given that the topology is stable. Thus, we are able to analyse the cluster stability both theoretically and through simulations, using experimental data reported from real field tests. Thirdly, we study the tradeoff between the time history necessary to achieve a certain stability and the responsiveness of the clustering algorihm. As a result, we estimate the delay induced by the time history, given a desired cluster stability.

2 Application scenarios

We describe two applications where wireless sensor nodes are able to extract and communicate the general contextual information for creating a dynamic cluster. The cluster is able to provide services such as reporting the group membership, analysing the cluster activity, recognizing fine-grained events and actions.

2.1 Transport and logistics

Transport and logistics represent large-scale processes that ensure the delivery of goods from producers to shops [9]. Using the wireless sensor networks technology in transport and logistics is particularly interesting for dynamically locating the goods, generating automatic packing lists, as well as for monitoring the storage condition of a product (e.g. temperature, light) or its surroundings.

The delivery process starts at a warehouse, where the transport company personnel assemble rolling containers (Returnable Transport Items - RTIs), pick the requested products from the warehouse shelves, and load them in the RTIs. Next, the RTIs are moved on the *expedition floor*, a large area used for temporary storage (see Figure 1). From the expedition floor, the RTIs are loaded into trailers according to the shop orders. In order to avoid the errors made during loading of items in the RTIs, a node equipped with a movement sensor can be attached to each product. At the moment the RTIs are pushed on the expedition floor, the sensors from each container correlate their movement and report as a group to the devices carried by the company personnel. In this way, a missing or wrong item can be detected before arriving on the expedition floor. The same solution can be applied for checking the correct loading of the RTIs into the trailers that deliver the goods to the shops.

In a similar manner works the list generator for automatic packing [1]. Various order items are packed in a box and an invoice is generated. In order to find out where a certain item is, nodes with movement sensors can be attached to each good. When the box is moved around, the items inside can correlate their movement and decide that they form a group. When the objects are grouped, checking on the goods and packing lists can be generated automatically.

Fig. 1. Transport and logistics scenario.

The scenario can be further extended, for example to the supermarket carts, where automated counters can automatically "see" the contents of the carts.

2.2 Body area networks (BAN)

Wearable computing aims at supporting workers or people in everyday life by delivering context-aware services. One important aspect is the recognition of human activities, which can be inferred from sensor integrated into garments and objects people are interacting with. The usage of context information enables more natural interaction between humans and computers and might assist workers with complex tasks with just now relevant information. Examples include training unskilled workers for assembly tasks [12], monitoring the health and activity of patients [7], assisting firefighters engaged in rescue operations in unknown environments with poor visibility.

Clustering nodes related to the activity of the persons simplifies the selection of relevant sensors in the environment which can contribute to the recognition of the currently performed activity. The advantage is that the activity recognition processing can be kept within the cluster, which is important for environments where multiple people are present. This provides: (1) identification of the body wearing the sensors, (2) a better recognition stability when selecting sensors moving with a person for the recognition task, and (3) potentially a trusted network where private data can be communicated only to nodes within the cluster.

3 Related work

Clustering in ad-hoc and sensor networks is an effective technique for achieving prolonged network lifetime and scalability [13]. Parameters such as the node degree, transmission power, battery level, processor load or degree of dynamics

usually serve as metrics for choosing the optimal clustering structure. Nevertheless, recent initiatives address the problem of grouping based on application-specific attributes. For instance, Bouhafs et al. [2] propose a semantic clustering algorithm for energy-efficient routing in wireless sensor networks. Nodes join the clusters depending on whether they satisfy a particular query inserted in the network. The output of the algorithm is called a *semantic tree*, which allows for layered data aggregation.

The Smart-Its project [5] first introduces the notion of context sharing: two smart objects are associated by shaking them together. Using this explicit interaction between the two devices, an application-level connection can be established. For example, the two devices can authenticate using secret keys that are generated based on the movement data [10]. Siegemund [11] proposes a communication platform for smart objects that adapts the networking structure depending on the context. A clusterhead node decides which nodes can join the cluster, based on similar symbolic location. Strohbach and Gellersen [14] propose an algorithm for grouping smart objects based on physical relationships. They use associations of the type "objects on the table" for constructing the clusters. A master node (the table) has to be able to detect the relationships for adding/deleting the nodes to/from the cluster. The solution described is suitable for static networks, where the master node is stationary for a long period of time. In our work, the network is dynamic, the context is permanently changing and every pair of nodes is capable of understanding the physical relationships, and thus the common context.

We now give two examples of shared-context recognition algorithms, where the practical results help us evaluate the clustering algorithm. First, Lester et al. [6] use the accelerometer data to determine if two devices are carried by the same person. The authors use a coherence function to derive whether the two signals are correlated at a particular frequency. Second, Marin-Perianu et al. [9] propose a correlation algorithm which determine whether dynamic sensor nodes attached to vehicles on wheels move together. We use the real-world experimental results reported by both algorithms to analyse the performance of Tandem. For each case, we evaluate the minimum time history necessary to achieve stable clusters.

4 Algorithm description

The goal of Tandem is to organize the nodes that share the same context, so that they can subsequently collaborate to provide a service. Tandem assumes that each node runs a shared-context recognition algorithm, which provides a number on a scale, representing the *confidence value* that two nodes are together. This algorithm can be for example a *coherence* function, which measures the extent to which two signals are linearly related at each frequency [6] (on a scale from 0 to 1), or the *correlation coefficient*, which indicates the strength and direction of a linear relationship [9] (on the scale from -1 to 1). Such an algorithm permanently evaluates the context, so that at each time step every node has an updated image

of the current situation, reflected in a new set of confidence values (one for every neighbour).

4.1 Requirements

Following the scenarios from Section 2, the nodes sharing the same context are within each-others transmission range, so we consider only one-hop clusters. The environment is dynamic, with frequent contextual and topological changes. The requirements and design choices for the clustering algorithm are the following:

- *Incorporate dynamics.* The clusters can merge or split, depending on the context changes. Nodes can join and leave the cluster at any time if the topology or context changes accordingly. For example, in the BAN scenario, people can pick up and use different tools, and then return or exchange them with other people. In this case, the nodes attached to the tools have to join and leave the BAN clusters. Contextual and topological changes cannot be predicted, so the clustering algorithm cannot assume a stable situation during cluster formation.
- *Stability.* If there are no contextual or topological changes, the clustering structure has to be stable. Following the remark that every node periodically re-evaluates the shared context with its neighbours, the fluctuations of the confidence values may lead to unwanted changes in the cluster structure. Therefore, the cluster has to cope with these fluctuations in order to keep the structure as stable as possible. A possible solution to increase the stability is to analyse the similarity of the context over a larger time history. In this sense, a tradeoff has to be found between the spontaneity in accommodating changes and the desired cluster stability.
- *Energy-efficiency.* The communication overhead should be kept to a minimum, for prolonging the lifetime of the wireless network.
- *Facilitate service provisioning.* The clusters have to be able to easily interact with the higher-layer applications and provide context-aware services to the user.

4.2 Cluster formation algorithm

For the sake of simplicity, we assume that a node is aware of its neighbours (every node within the range) and the one-hop communication is reliable (this can be achieved for example by using a simple stop-and-wait ARQ protocol).

Each node v periodically computes the confidence of sharing the same context with its neighbours. If the confidence with a neighbour u exceeds a certain threshold, then v considers that it shares the same context with u for the given time step. The final decision for sharing the same context with u is founded on the confidence values from a number of previous time steps, called the *time history* (see Section 5.1).

The fact that the perception of the shared context may vary from one node to another leads to nodes having different views of the cluster membership. To

Algorithm 1 Tandem - node v (events/actions)

Initialization:

1. $r(v) \leftarrow \perp,\ r(u) \leftarrow \perp,\ \forall u \in \Gamma(v)$

GetContext:

1. $r_0(v) \leftarrow r(v)$ // Store the root of v
2. Update $h(v,m), \forall m \in \Gamma(v)$ // Update the history
3. $M \leftarrow \{m \in \Gamma(v) \mid h(v,m) > h_{min}\}$ // Select the nodes sharing the same context with v
4. **if** $M \neq \emptyset$ **then**
5. $\quad M_0 \leftarrow \{\ m \in M \mid r(m) = m\}$ // Select from M the nodes that are clusterheads
6. $\quad$ **if** $M_0 \neq \emptyset$ **then**
7. $\quad\quad$ Choose $u \in M_0$ such that $pn(u) = max\{pn(m) \mid m \in M_0\}$
8. $\quad\quad$ **if** $(r(v) = \perp) \vee (r(v) \notin M_0) \vee (r(v) = v \wedge pn(v) < pn(u))$ **then**
9. $\quad\quad\quad r(v) \leftarrow u$ // Choose u as the root of v
10. $\quad\quad$ **end if**
11. $\quad$ **else if** $r(v) \neq v$ **then**
12. $\quad\quad M_1 \leftarrow \{\ m \in M \mid r(m) \neq \perp\}$ // Select from M the nodes that have a valid root
13. $\quad\quad$ **if** $M_1 \neq \emptyset$ **then**
14. $\quad\quad\quad r(v) \leftarrow \perp$
15. $\quad\quad$ **else**
16. $\quad\quad\quad r(v) \leftarrow v$ // Become root
17. $\quad\quad\quad$ Generate $pn(v) > 0$
18. $\quad\quad$ **end if**
19. $\quad$ **end if**
20. **else**
21. $\quad r(v) \leftarrow \perp$ // v is unassigned
22. **end if**
23. **if** $r_0(v) \neq r(v)$ **then**
24. $\quad$ Send $SetRoot\ (v, r(v), pn(v))$ to neighbours // Announce root change
25. **end if**

provide a consistent management of the membership list and cluster organization, a *clusterhead* or *root* node is dynamically elected among the nodes that share the same context. This also assists the service provisioning and the interaction the higher-layer applications. In order to allow merging of clusters and to facilitate the election process, the candidate clusterheads dynamically generate unique *priority numbers*, either based on the unique hardware addresses, or as a context-dependant measure, such as the rapidity in occupying the wireless medium. A regular node subscribes to the clusterhead with which it shares a common context and has the highest priority number.

We use the following notation:

- V is the the set of nodes in the network.
- n is the number of nodes in the network, $n = |V|$.
- $r(v)$ is the root or clusterhead of node v.
- $pn(v)$ is the priority number of node v.
- $\Gamma(v)$ is the neighbourhood of node v.
- $h(v,u)$ represents the number of times v and u are sharing a common context over a total time history of H steps.
- h_{min} is the minimum amount of times when two nodes are sharing a common context, such that they can safely be considered part of the same cluster.

The algorithm constructs a set of one-hop clusters, based on the context information shared by the nodes. A node v can be: (1) *unassigned*, where v is

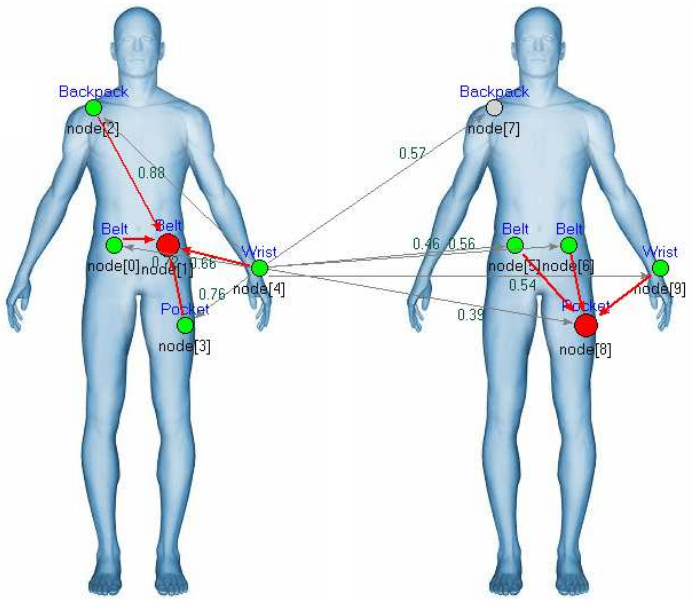


Fig. 2. Graphical simulation of the clustering algorithm on BANs.

not part of any cluster, (2) *root*, where v is clusterhead, or (3) *assigned*, where v is assigned to a cluster where the root node is one of its neighbours.

Algorithm 1 gives the detailed description of the cluster formation and update of knowledge among neighbouring nodes. Every node has the following information about its neighbours: the root, the priority number and whether it shares a common context for a specified time history. Let v be an arbitrary node in the network. At each time step, node v changes or chooses its root node in the following cases: (1) v is unassigned, (2) v does not share a common context with its root, (3) the root of v is no longer a root or (4) v is root and there is another neighbour root, sharing the same context with v, that has a higher priority number. In any of these cases, v chooses as root node the neighbour root u with which it shares a common context and which has the highest priority number. If such a neighbour does not exist, v competes for clusterhead or becomes unassigned. The decision is based on the current status of the neighbours and tries to minimize the effect of the following erroneous situation: due to context fluctuations, an assigned node v may loose its root node and cannot join another cluster because none of its neighbours is root. Therefore, v may become root, form a new cluster and attract other nodes in that cluster. To avoid this undesirable outcome, a node declares itself root only if all its neighbours with which it shares a common context are unassigned. If there exists at least one neighbour u with which v shares a common context and u has a valid root node, then v becomes unassigned.

Node v announces the changes in choosing the root node by sending a local broadcast message *SetRoot* to its neighbours. In case of topological changes, this message is also used to announce the new neighbours of the current structure.

Let us consider the example from Figure 2. A BAN is associated with each person, consisting of five nodes placed in various locations: backpack, belt, pocket and wrist. The clustering structure is seen from the perspective of node 4, which is attached to the wrist of the left person. The clusterheads are represented with bigger, red-filled circles (nodes 1 and 8). The red arrows indicate the assignment of the other nodes to the current clusterheads. Node 7 is unassigned, as the shared-context recognition algorithm did not associate this node with any of the neighbouring clusterheads at the previous time step. The grey arrows show the confidence values computed by node 4 at the current time step in this particular case. The confidence values for the nodes on the same body with node 4 range between 0.66 and 0.88, while for the other body they lie between 0.39 and 0.57. Because the confidence of sharing the same context with the root node 1 is 0.66 and above the threshold, node 4 keeps the current root. Otherwise, it would become unassigned (node 4 has some neighbours with the same context, having a valid root node), or assigned to the other cluster, if the confidence value for the neighbouring root node 8 was higher than the threshold.

5 Cluster stability analysis

Several algorithms for context sharing have been proposed in the literature, using various sensors and providing different accuracies (see Section 3). However, none of them gives a measure of the overall accuracy of the system, when multiple nodes sharing different contexts come together. We would like to analyse the cluster stability from both the theoretical point of view, by giving average approximation, upper and lower bounds, and through simulations. In addition, we are interested in the tradeoff between the time history necessary to achieve a certain stability and the responsiveness of the clustering algorithm. First, we compute the probabilities of correctly assessing the context, given the distribution of the confidence values. Second, we model the algorithm using Markov chains and we derive an approximation for the proportion of time the clustering structure is in a correct state.

5.1 Determination of common context

In this section, we give an example of how the probabilities of correct detection of the shared context can be computed.

Let v be a node in the network and u a neighbour of v. If v does not share the same context with u (e.g. they represent sensor nodes attached to different persons), we model the confidence value computed by the shared-context recognition algorithm with the random variable $X_1(v, u)$. If v shares the same context with u (e.g. they are attached to the same person), we model the confidence value as a random variable $X_2(v, u)$. We take the distribution encountered during the experiments as the reference Probability Density Function (PDF): we associate the random variables $X_1(v, u)$ with the PDF φ_1 and the corresponding

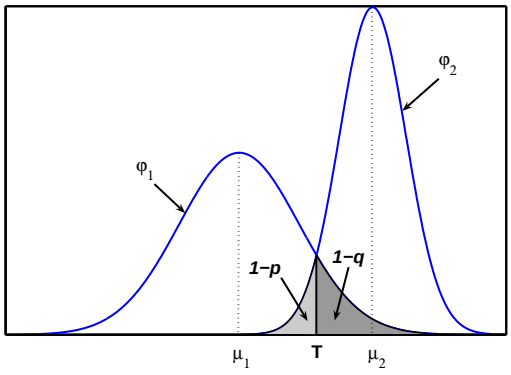


Fig. 3. The calculation of the threshold value T and the probabilities p and q.

Cumulative Distribution Function (CDF) Φ_1. Similarly, we associate the random variables $X_2(v,u)$ with the PDF φ_2 and CDF Φ_2.

Node v selects the subset of its neighbours with which it shares a common context based on a threshold value T. We choose T as the intersection point of the two PDFs φ_1 and φ_2, which minimizes the sum of probabilities of an incorrect determination. We denote p as the probability of the correct detection of the *common* context and q as the probability of the correct detection of *different* contexts. The probabilities p and q are computed in the following way (see Figure 3):

$$p = 1 - \Phi_2(T), \qquad q = \Phi_1(T) \tag{1}$$

We compute the threshold value for the case where the distributions are normal, which is valid for the applications described in Section 2 (see the experimental data reported by [6, 9]). Let us consider two normal distributions, $\varphi_1(\mu_1, \sigma_1)$ and $\varphi_2(\mu_2, \sigma_2)$. The intersection point of φ_1 and φ_2 which lies between μ_1 and μ_2 is the following:

$$T = \frac{\mu_1\sigma_2^2 - \mu_2\sigma_1^2 + \sigma_1\sigma_2\sqrt{(\mu_1-\mu_2)^2 + 2(\sigma_2^2-\sigma_1^2)ln(\sigma_2/\sigma_1)}}{\sigma_2^2 - \sigma_1^2} \tag{2}$$

Using Eq. 1 and 2, it is straightforward to compute p and q, knowing the characteristics of φ_1 and φ_2. We are now interested in how these probabilities change if we involve the time history in the decision process. The probability p_h of the correct detection that two nodes share a common context for a minimum time history h_{min} out of a total of H time steps is given by the CDF of the binomial distribution:

$$p_h(h_{min}, H) = \sum_{k=h_{min}}^{H} \binom{H}{k} p^k (1-p)^{H-k} \tag{3}$$

Similarly, the probability q_h of the correct detection of different contexts for a minimum time history h_{min} out of a total of H time steps is:

$$q_h(h_{min}, H) = \sum_{k=h_{min}}^{H} \binom{H}{k} q^k (1-q)^{H-k} \tag{4}$$

We have therefore $p = p_h(1,1)$ and $q = q_h(1,1)$.

5.2 Modelling with Markov chains

We approximate the behaviour of the algorithm with a Markov chain, which allows us to estimate the global probability of having a correct cluster. We stress on the difference between a *time step* and a *Markov chain step*. A time step is related to the periodic update of the context information by the shared-context recognition algorithm which runs on every node. For improving the probabilities of correct detection of the shared context, the algorithm looks over a time history H, composed of a number of time steps (see Section 5.1). A Markov chain step is the "memoryless" transition from one state to another, which happens on intervals equal to the total time history H.

We define a *group* G as the collection of nodes that share the same context in reality. We define a *cluster* C as the collection of nodes which have the same root node (as a result of Agorithm 1). The goal of the clustering algorithm is that for any group of nodes G, there exists a cluster with the root $r_0 \in G$ such that $\forall v \in V, r(v) = r_0 \Leftrightarrow v \in G$. Taking the example from Figure 2, we have two groups: $G_1 = \{0,1,2,3,4\}$, $G_2 = \{5,6,7,8,9\}$ and two clusters: $C_1 = \{0,1,2,3,4\}$ with root node 1, $C_2 = \{5,6,8,9\}$ with root node 8.

We define the following states for cluster C:

1. *Correct:* The cluster has exactly one root node from the group and all the members of the group are part of the cluster[1].
2. *Has root:* At least one group member is root.
3. *No root:* None of the group members is root.
4. *Election:* After reaching state 3, members of the group start an election process for choosing the root node.

For example, cluster C_1 from Figure 2 is *Correct*, while C_2 is in the state *Has root*, since node 7 is unassigned.

The transition matrix that determines the Markov chain is the following:

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{pmatrix}$$

Let $m \geq 0$ be the number of root nodes with higher priority than the current root and $k \geq 1$ the number of nodes in the cluster. If the cluster has a root, let r_0 be the root node. The probabilities p_{ij} are evaluated in a worst case scenario, by minimizing the chance to get in the *Correct* state.

[1] We intentionally take $G \subseteq C$. The nodes from $C \setminus G$ are part of other groups, which have the corresponding clusters in an incorrect state.

Table 1. Transition probabilities p_{ij}

Probability	**Value**	**Probability**	**Value**
p_{11}	$q^m p^{k-1}$	p_{21}	$q^m p^{k-1} q^{m(k-1)}$
p_{12}	$q^m(1-p^{k-1})$	p_{22}	$q^m(1-p^{k-1}q^{m(k-1)})$
p_{13}	$1-q^m$	p_{23}	$1-q^m$
p_{14}	0	p_{24}	0
p_{31}	0	p_{41}	0
p_{32}	0	p_{42}	$q^{mk}(1-(1-p)^{k(k-1)})$
p_{33}	0	p_{43}	0
p_{34}	1	p_{44}	$1-q^{mk}(1-(1-p)^{k(k-1)})$

The conditions which determine the probabilities p_{ij} are the following:

- p_{11}: (a) r_0 remains root in the next step, as r_0 does not share the same context with other root nodes with higher priority, and (b) all the nodes in the group share the same context with r_0.
- p_{12}: (a) r_0 remains root in the next step, and (b) there exists at least one node in the group that does not share the same context with r_0.
- p_{13}: r_0 shares the same context with a root node with higher priority, so that it gives up its role and joins another cluster.
- p_{21}: (a) r_0 remains root in the next step, (b) all the nodes in the group do not share the same context with other root nodes with higher priority and (c) all the nodes in the group share the same context with r_0.
- p_{23}: r_0 shares the same context with a root node with higher priority, so that it gives up its role and joins another cluster.
- p_{34}: from state *No root* the system goes at the next step to state *Election*.
- p_{42}: (a) all the nodes in the group do not share the same context with any root node with higher priority, and (b) there are at least two nodes in the group that share the same context.

Table 1 gives the computed probabilities for each transition of the Markov chain. We notice that the *Correct* state can be reached only from the *Has root* state. If $m > 0$, the probability p_{21} is minimized, so that in the stationary distribution of the Markov chain, the probability to be in the *Correct* state is lower than the real probability. Calculating the fixed row vector of the Markov chain yields the following result:

$$p_1(m,k,q,p) = \frac{p_{21}p_{42}}{(1+p_{21}-p_{11})(p_{42}+p_{42}p_{13}+p_{13})} \tag{5}$$

We define the *cluster stability* P_S as the probability of a cluster to be in the *Correct* state. Given that there are c clusters in the network, we have the following lower and upper bounds:

$$p_1(c-1,k,q,p) \leq P_S \leq p_1(0,k,q,p) \tag{6}$$

An estimation of the average case is given by:

$$P_S \approx p_1(\frac{c-1}{2}, k, q, p) \tag{7}$$

6 Results

A typical example of context sharing is the similarity of movement, which we analyse in this section using real experimental data corresponding to the scenarios described in Section 2. In general, the movement information is extracted from accelerometers. Simpler sensors such as tilt switches can be also used, but with less accurate results. We have the following two concrete examples of wireless objects moving together:

1. **RTI** - wireless sensor nodes used in a transport scenario, which correlate their movement pattern; both tilt switches and accelerometers are used to extract the movement information [9].
2. **BAN** - smart devices that decide whether they are carried by the same person, using the coherence between the movement data provided by accelerometers [6].

Table 2 shows the characteristics of the normal distributions derived from the concrete experiments conducted in both application examples, together with the computed threshold from Eq. 2 and the probabilities p and q. Contrary to the RTI scenario, where the nodes moving together experience exactly the same movement pattern, in the BAN scenario different parts of the body are engaged in different types of movements during walking. For a realistic evaluation, we choose the worse case experimental results from the BAN scenario [6], where the nodes are attached to the pocket and wrist of the subjects (Pocket/Wrist experimental trial).

Table 2. Statistics from the experiments and computed probabilities.

Application	μ_1	σ_1	μ_2	σ_2	T	p	q
RTI - tilt switch	0.641	0.087	-0.017	0.249	0.438	0.9902	0.9662
RTI - accelerometer	0.817	0.106	0.009	0.124	0.442	0.9998	0.9998
BAN - Pocket/Wrist	0.757	0.065	0.519	0.069	0.640	0.9636	0.9607

For our experiments we use the OMNeT++ [15] simulation environment. The scalability analysis of the movement correlation method proposed for the RTI scenario indicate a maximum network density of 100 nodes [9], imposed by the shared wireless medium. Because the same periodic transmission of the movement data is needed for the BAN scenario [6], we use in our simulations the same maximum network density for both applications. As the cluster formation

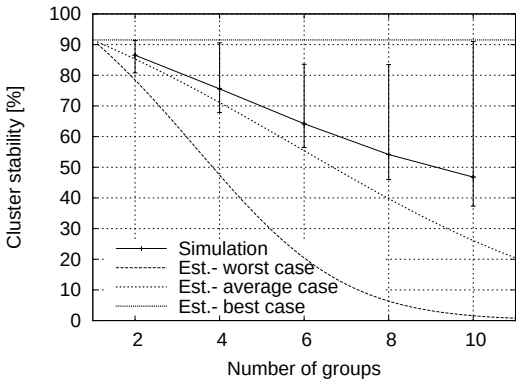


Fig. 4. Cluster stability in the RTI with tilt switches scenario ($h_{min} = H = 1$).

and stability is affected only by the nodes in the one-hop neighbourhood, we simulate a network of maximum 100 nodes that are attached to different mobile objects or people, and where there is a wireless link between any two nodes. The cluster stability is directly influenced by the number of nodes in the group and by the number of groups: the more nodes in one group and the more groups, the less stable are the clusters. We focus on a mobile scenario with clustered nodes moving around and passing each other, and thus we analyse how the cluster stability changes when we vary the number of groups. We have 10 nodes in each group moving together and interacting with other groups, and we vary the number of groups between 2 and 10 and also the time history. We recall from Section 5.2 that the cluster stability is the probability that the cluster is in the *Correct* state. The graphs from Figures 4-8 represent the cluster stability in percentage, for the following cases: (1) average simulation results, (2) estimation of the worst case, derived from Eq. 6, (3) estimation of the average case, derived from Eq. 7, and (4) estimation of the best case, derived from Eq. 6.

For each point on the simulation plots we run up to 10 simulations of $10^4 - 10^5$ time steps. In order to study the influence of the history size, we take $H = 2h_{min} - 1$ and we vary h_{min} from 1 to 4.

6.1 RTI with tilt switches scenario.

Figure 4 shows the cluster stability depending on the number of groups present in the network, given that the time history is $h_{min} = 1$. The error bars represent the absolute minimum and maximum stability recorded during the simulations. We notice that the results respect the upper and lower bounds calculated theoretically. The estimation of the average case is close to the simulations for a small number of groups. However, increasing the number of groups decreases the precision of the approximation, due to the minimization of the transition probabilities to get in the *Correct* state (see Section 5.2).

Figure 5 shows the cluster stability depending on the time history, for a network composed of 10 groups. We notice that increasing the time history considerably improves the cluster stability and the theoretical approximation.

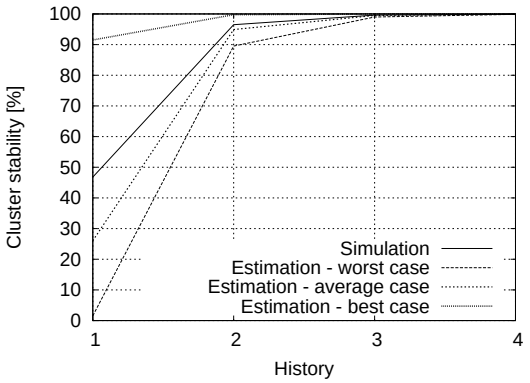


Fig. 5. Cluster stability depending on the time history in the RTI with tilt switches scenario (10 groups).

For a time history $h_{min} = 4$ ($H = 7$), a stability of 99.92 is achieved, while the lower bound is 99.89. Therefore, for achieving a stability close to 100%, the necessary delay is $H \times 16 = 112$ seconds (the size of the data sequence is 16 seconds [9]).

6.2 RTI with accelerometers scenario.

The solution in using accelerometers is more reliable, resulting in higher probabilities for the correct detection of the context (see Table 2) and consequently, higher cluster stability. Figure 6 shows the cluster stability depending on the number of groups present in the network, given that the time history is $h_{min} = 1$. We also represent the error bars for the absolute minimum and maximum values. We notice the high stability obtained even for large number of groups (99.5 for 10 groups). Due to the fact that the clusters stay in the *Correct* state for most of the time, the approximations are close to the simulation results. For this scenario, a high cluster stability can be achieved even considering the time history 1, reaching a responsiveness of only 16 seconds.

6.3 BAN scenario.

Figure 7 shows the cluster stability in the BAN scenario, depending on the number of groups, given that the time history is $h_{min} = 1$. Similarly with the two scenarios presented above, we notice that the results respect the upper and lower bounds calculated theoretically. The average stability is lower than in the previous cases, with a maximum of 67% and less than 50% for a network composed of more than 6 groups.

Figure 8 shows the cluster stability depending on the time history, for a network composed of 10 groups. The time history significantly improves the cluster stability: for the time history $h_{min} = 4$ ($H = 7$), the stability is 99.84 and the lower bound is 99.74. For achieving this, the delay is $H \times 8 = 56$ seconds (the window size is 8 seconds [6]).

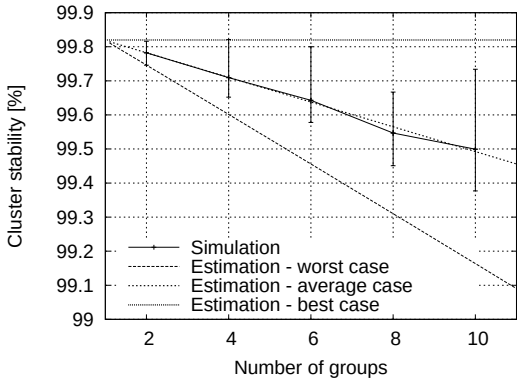


Fig. 6. Cluster stability in the RTI with accelerometers scenario ($h_{min} = H = 1$).

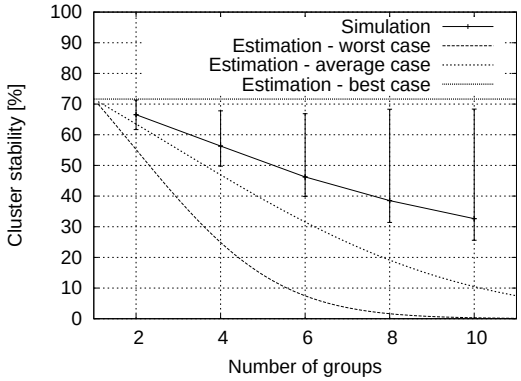


Fig. 7. Cluster stability in the BAN scenario ($h_{min} = H = 1$).

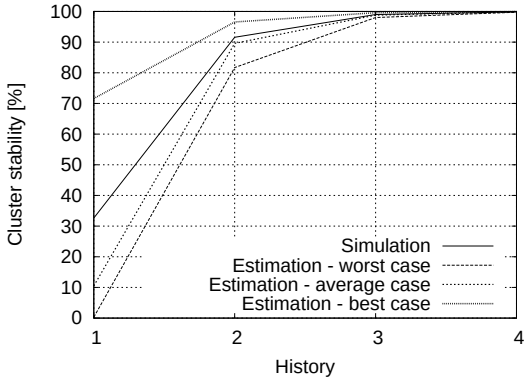
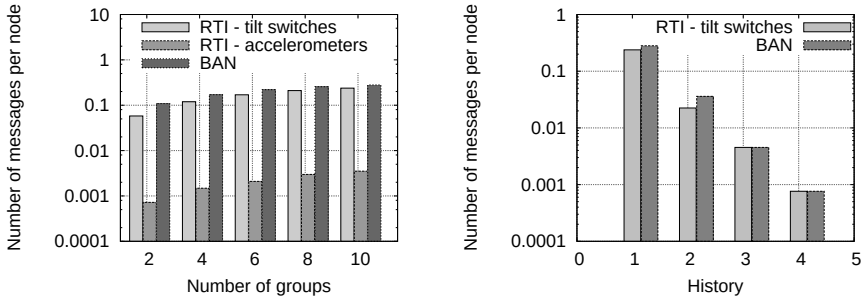


Fig. 8. Cluster stability depending on the time history in the BAN scenario (10 groups).



(a) Number of groups. (b) History.

Fig. 9. Average number of *SetRoot* messages sent per node in 100 steps.

6.4 Communication overhead.

The communication overhead is induced by the *SetRoot* message, sent by every node when the current root node changes. In case the lower networking layers use a periodic message in order to maintain for example the synchronization (e.g. in case of a TDMA MAC protocol), Tandem can use this heartbeat to piggyback the small piece of information present in the *SetRoot* message (root address and priority number). Assuming that there is no such periodic message, we count the average number of *SetRoot* messages sent by each node in one step of simulation on average.

Figure 9(a) shows the number of messages on a logarithmic scale, depending on the number of groups. The results correspond to each of the three scenarios, where the time history is 1. We notice that the more stable the structure is, the less communication overhead is needed. For the RTI with accelerometers scenario, less than 1 message is sent per node in 100 time steps, even for a large number of groups. The overhead is increasing as the number of groups increases, due to the diminishing cluster stability.

Figure 9(b) shows the number of messages depending on the time history, for the RTI with tilt switches and BAN scenarios. Increasing the time history improves the stability and thus reduces the communication overhead. For the time history 4, the overhead is less than 10^{-3} messages per node.

7 Discussion and conclusions

We presented Tandem, a context-aware method for spontaneous clustering of wireless sensor nodes. The algorithm allows reclustering in case of topological or contextual changes. By analysing the similarity of the context over a time history, Tandem tries to achieve stable clusters. We approximate the behaviour of the algorithm using a Markov chain model and we analyse the cluster stability theoretically and through simulations, using experimental data reported from real field tests. The analysis gives the possibility to theoretically estimate the stability of the structure and the responsiveness of the algorithm. Computing

the worse case stability via the Markov chain approximation, we can deduce the time history necessary to achieve stable clusters.

In what follows, we discuss the main advantages and limitations of the proposed clustering method.

Advantages

- *Responsiveness.* The clustering structure reacts quickly to topological and contextual changes: nodes decide based only on the current situation of their neighbourhood, without the need of any negotiation with other parties.
- *Small-scale experiment required.* For computing the probabilities p and q that can be used to estimate the cluster stability, only a small-scale reproducible experiment is required. For example, two nodes moving together and another two moving separately are enough to generate the statistical distributions of the confidence values.
- *Delay estimation.* By deducing the time history required to achieve a certain stability, the delay in accommodating the topological or contextual changes can be easily estimated.

Limitations

- *Rough approximation for many groups.* As we notice from Figures 4 and 7, the difference between the approximation that we derive using Markov chains and the real situation is increasing with the number of groups. However, the model offers a good approximation in case of highly accurate context detection methods (see Figure 6). Therefore, the approximation can be successfully used for deducing the minimum time history for a cluster stability close to 100%.
- *Multihop clusters.* The method that we propose is valid only for one-hop clusters, which is justified taking into account the scenarios from Section 2. Nevertheless, other applications may require multihop clusters, even several layers of clustering. For example, groups of people skiing together, forming multihop clusters, where each person is wearing a BAN that is a one-hop cluster. The algorithm can be easily extended to accommodate multihop clusters: instead of choosing directly the clusterhead node, every node selects a parent and thus joins the cluster associated with the parent node.

For future work, we intend to extend the algorithm for multihop clusters and to investigate the cluster stability in this case. We also plan a series of experiments involving clustering of nodes on body area networks. Subsequent to clustering, a task allocation mechanism distributes various tasks to the nodes which are part of the cluster, depending on their capabilities. The final goal is to have a distributed activity recognition algorithm running on the BAN, which allows dynamically entering and leaving of nodes to/from the context-aware cluster.

References

1. S. Antifakos, B. Schiele, and L. E. Holmquist. Grouping mechanisms for smart objects based on implicit interaction and context proximity. In *UBICOMP 2003 Interactive Posters*, pages 207 – 208, 2003.
2. F. Bouhafs, M. Merabti, and H. Mokhtar. A semantic clustering routing protocol for wireless sensor networks. In *Consumer Communications and Networking Conference*, pages 351– 355. IEEE Computer Society, 2006.
3. A. Gluhak, M. Presser, L. Zhu, S. Esfandiyari, and S. Kupschick. Towards mood based mobile services and applications. In *2nd European Conference on Smart Sensing and Context (EuroSSC)*, 2007.
4. Lin Gu, Dong Jia, Pascal Vicaire, Ting Yan, Liqian Luo, Ajay Tirumala, Qing Cao, Tian He, John A. Stankovic, Tarek Abdelzaher, and Bruce H. Krogh. Lightweight detection and classification for wireless sensor networks in realistic environments. In *SenSys*, pages 205–217, New York, NY, USA, 2005. ACM Press.
5. Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-Werner Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *UbiComp'01*, pages 116–122, London, UK, 2001. Springer-Verlag.
6. Jonathan Lester, Blake Hannaford, and Gaetano Borriello. "Are You with Me?" - using accelerometers to determine if two devices are carried by the same person. In *Pervasive*, pages 33–50. Springer Verlag, 2004.
7. Konrad Lorincz, David J. Malan, Thaddeus R.F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, 03(4):16–23, 2004.
8. M. Marin-Perianu and P. J. M.Havinga. D-FLER: A distributed fuzzy logic engine for rule-based wireless sensor networks. Technical Report TR-CTIT-07-54, CTIT, University of Twente, 2007.
9. R. S. Marin-Perianu, M. Marin-Perianu, P. J. M. Havinga, and J. Scholten. Movement-based group awareness with wireless sensor networks. In *Pervasive*, pages 298–315. Springer Verlag, 2007.
10. R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In *Pervasive*, pages 144–161. Springer-Verlag, 2007.
11. Frank Siegemund. A context-aware communication platform for smart objects. In *Pervasive*, pages 69–86, 2004.
12. T. Stiefmeier, C. Lombriser, D. Roggen, H. Junker, G. Ogris, and G. Tröster. Event-based activity tracking in work environments. In *Proceedings of the 3rd International Forum on Applied Wearable Computing (IFAWC)*, March 2006.
13. Ivan Stojmenovic, Mahtab Seddigh, and Jovisa Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, 2002.
14. M. Strohbach and H. Gellersen. Smart clustering - networking smart objects based on their physical relationships. In *Proceedings of the 5th IEEE International Workshop on Networked Appliances*, pages 151– 155. IEEE Computer Society, 2002.
15. A. Varga. The omnet++ discrete event simulation system. In *ESM'01*, Prague, Czech Republic, June 2001.
16. Jue Wang, Guanling Chen, and David Kotz. A sensor fusion approach for meeting detection. In *MobiSys 2004 Workshop on Context Awareness*, June 2004.