Movement-based Group Awareness with Wireless Sensor Networks

Raluca Marin-Perianu, Mihai Marin-Perianu, Paul Havinga, Hans Scholten

University of Twente, Enschede, The Netherlands
{r.s.marinperianu, m.marinperianu, p.j.m.havinga, j.scholten}@utwente.nl

Abstract. We propose a method through which dynamic sensor nodes determine that they move together, by communicating and correlating their movement information. We describe two possible solutions, one using inexpensive tilt switches, and another one using low-cost MEMS accelerometers. We implement a fast, incremental correlation algorithm, with an execution time of $\approx$ 6ms, which can run on resource constrained devices. The tests with the implementation on real sensor nodes show that the method is reliable and distinguishes between joint and separate movements. In addition, we analyze the scalability from four different perspectives: communication, energy, memory and execution speed. The solution using tilt switches proves to be simpler, cheaper and more energy efficient, while the accelerometer-based solution is more reliable, more robust to sensor alignment problems and, potentially, more accurate by using extended features, such as speed and distance.

1 Introduction

Emerging applications of wireless sensor networks (WSNs) demand an increasing degree of *dynamics*. The sensor nodes are expected to take decisions autonomously, by using *context-aware reasoning*, and to provide an overall solution that is more reliable, accurate and responsive than traditional approaches. Examples of recent application domains include industrial processes, transport and logistics, user guidance in emergency situations [8]. In all these scenarios, we can note a rapidly growing interest in having many small, cheap devices that self-organize and cooperate, in order to supervise and actively support the actual processes. The challenges shift, accordingly, from small-scale user-to-device interaction toward large-scale device-with-device *collaboration*. In parallel, the design choices migrate from complex, centralized approaches toward simple, distributed techniques, that can be implemented on very resource constrained devices.

In this paper we propose to construct dynamic groups based on common context, by using common movement information. More specifically, nodes are considered to be *together* if their movement *correlates* for a certain amount of time. We argue that such a method opens perspectives for a large variety of applications, ranging from user entertainment (people hiking or skying together) to healthcare (body area networks), and smart vehicles carrying smart goods (in the field of transport and logistics, as we describe in Section 2). There are, however, a number of questions that such a solution should answer:

1. How to extract and communicate the movement information?
2. How to compute the correlation, taking into account the resource limitations of the sensor nodes?
3. How does the method scale with the number of nodes?
4. How reliable is the solution and which are the benefits and limitations?

The contribution of this paper is a lightweight, fast and cheap method for correlating the movement data among sensor nodes, for the purpose of clustering nodes moving together. Each node correlates the movement data generated by the local movement sensor with the movement data broadcast periodically by its neighbors. The result of the correlation is a measure of the confidence that one node shares the same context with its neighbors, for example that they are placed in the same car. We focus in this paper on correlating sensor nodes carried by vehicles on wheels.

We describe two possible practical solutions, one using tilt switches, and another one using MEMS accelerometers. In order to answer the aforementioned questions in detail, we analyze the scalability from several different perspectives (communication, energy, memory and execution speed), and discuss the most relevant advantages and limitations. The analysis is based on the experimental results obtained from testing with real sensor nodes. We use the Ambient μNode 2.0 platform [1], with the low-power MSP430 micro-controller produced by Texas Instruments, which offers 48kB of Flash memory and 10kB of RAM. The radio transceiver has a maximum data rate of 100kbps. Figure 1 shows the sensors used for extracting the movement information and the sensor node platform.

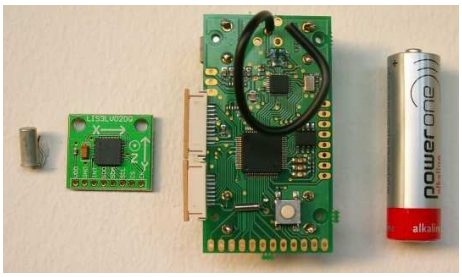

Fig. 1. Movement sensors and sensor node platform.

In the following section we describe a concrete application setting in the field of transport and logistics, which best illustrates the idea of movement-based group awareness. Section 3 overviews the relevant related work. The general correlation method is described in Section 4. In Sections 5 and 6 we present the two practical solutions for autonomous group formation. Section 7 covers the analysis, advantages and limitations of both solutions, giving also comparative details whenever relevant. Finally, Section 8 formulates the conclusions.

2 Application Setting

Transport and logistics represent large-scale processes that ensure the delivery of goods from producers to shops. The distribution process starts at a *warehouse*, where an *order picker* gets an order list, assembles a rolling container (Returnable Transport Item - RTI), picks the requested products from the warehouse shelves, and loads them in the RTI. Next, the order picker moves the RTI to the *expedition floor*, a large area used for temporary storage (see Figure 2). The expedition floor is seen as a grid, where each cell of the grid is associated with a certain shop. At loading time, the *loading operators* place the RTIs into trailers, according to a *loading list*, derived from the delivery orders. Eventually, a truck pulls the trailer and delivers the goods to the shops.

Due to the large scale of the process, the transport company personnel (e.g. order pickers, loading operators) is prone to errors. It often happens that the order pickers make mistakes when filling the RTIs with goods, or that the RTIs are loaded in the wrong trailer. In addition, the products are sometimes stored in improper climate conditions, which is a serious problem in the case of perishable goods. WSN technology can be a solution to these problems, as sensor nodes offer precise control over the status (e.g. location, storage temperature) and history of the goods. Consequently, the RTIs and the products carried in them, equipped with sensor nodes, can check for errors and trigger alerts at the point of action. Movement-based group awareness is an essential component for achieving this vision of smart RTIs and goods.

The solution that we propose targets two specific problems. First, the goods from an RTI correlate their movement as the RTI is pushed, and report as a group to the device carried by the order picker. In this way, a missing or wrong item can be detected before arriving on the expedition floor. Second, any RTI placed in the wrong trailer should be signaled as the truck approaches the exit gate (see Figure 2). Since the distance between two RTIs or two trucks is quite short, the localization of the goods inside the RTI, or of the RTIs inside the truck cannot be done reliably with radio signal strength proximity techniques. We consider, however, highly probable that two different vehicles move differently in a certain time interval. Therefore, we propose to group the nodes based on the similarities and differences in the data generated by the movement sensors.

3 Related work

Grouping of devices into clusters has been a topic of major interest in the field of wireless networks [9]. The clusterhead node is usually chosen based on different node properties such as the node ID, capability, degree of dynamics, or topological characteristics, such as the node degree. The main goal is to achieve energy efficiency at the networking layer, and therefore the application-level attributes are usually not of concern. Nevertheless, grouping based on application-specific attributes has been studied in the field of service discovery [4]. Nodes organize into groups, in order to efficiently search for services. However, the grouping

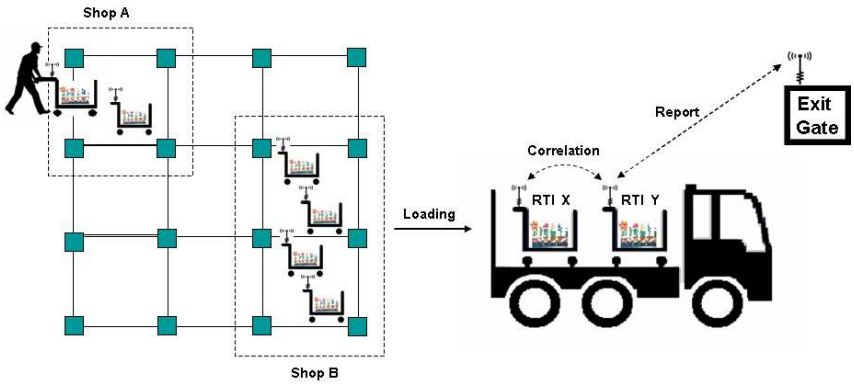

Fig. 2. Transport and logistics process diagram.

criteria is usually based on statically-assigned semantic descriptions, while we are interested in context-dependent dynamically changing attributes.

Lam et al. [6] propose an algorithm for dynamic grouping based on the position and speed of the nodes. Each node is attached to a GPS sensor, which keeps track of the existing location and movement information. Nodes that can communicate with one another and move together in a similar manner form a group. However, equipping each node with a GPS sensor is not a viable solution for WSNs, because of price and power consumption considerations.

In the project Smart-Its, Gellersen et al. [5] formulate the notion of context sharing. The idea is to associate two smart objects by shaking them together. As a result, the user can establish an application-level connection between two devices by imposing a brief, similar movement. In our work, we are interested in extending the idea of "moving together" at the group level, within large-scale industrial and business scenarios. Therefore, we propose a fast algorithm that correlates the movement over a larger time history, and analyze the scalability, performance and limitation factors.

Lester et al. [7] use accelerometer data to determine if two devices are carried by the same person. Human locomotions represent a repeated activity that makes an analysis in the frequency domain possible. The authors use a coherence function to derive whether the two signals are correlated at a particular frequency. Our application domain poses, however, quite different challenges. There is no regularity in the movement of the RTIs that can facilitate an analysis in the frequency domain. Moreover, the computations involved in the frequency analysis can easily overcome the resources available on sensor nodes.

Spatial and temporal correlation of sensor data in a WSN has been studied for data aggregation [11], which assures that the readings are transmitted in an energy-efficient manner to the sink node. Our approach is different, in the sense that we provide a method for nodes to detect dynamically a common context, and set up accordingly an ad-hoc group.

4 General Method

The algorithm correlates the movement data generated by the movement sensors attached to different sensor nodes. Regardless of the movement sensor type, we use the same general method.

4.1 Computing the Correlation

Let **x** be one of the sensor nodes, which receives the data from another sensor node **y**. Node **x** stores the latest sample values produced by the local movement sensor in a circular buffer X_C of size k. The buffer X_C is periodically transmitted to the neighbors at intervals $k\Delta t$, where Δt is the sampling interval. At step $i \geq 1$, **x** receives from **y** the buffer $Y_i = \{y_{(i-1)k+1}, y_{(i-1)k+2}, ..., y_{ik}\}$. Node **x** then copies the buffer X_C into a working copy $X_i = \{x_{(i-1)k+1}, x_{(i-1)k+2}, ..., x_{ik}\}$ and calculates the correlation coefficient over the last n sequences of data X_i and Y_i. More precisely, at each step i, the correlation coefficient is calculated over the data $X = (x_{(i-n)k+1}, x_{(i-n)k+2}, ..., x_{ik})$ and $Y = (y_{(i-n)k+1}, y_{(i-n)k+2}, ..., y_{ik})$, which represents the last $N = nk$ samples. Note that for $j \leq 0$, $x_j = y_j = 0$. If we denote the means of X and Y as $\bar{X}$ and $\bar{Y}$, respectively, the correlation coefficient is computed as follows:

$$\rho(X,Y) = \frac{cov(X,Y)}{\sqrt{var(X)\ var(Y)}} = \frac{\sum_{j=(i-n)k+1}^{ik}(x_j - \bar{X})(y_j - \bar{Y})}{\sqrt{\sum_{j=(i-n)k+1}^{ik}(x_j - \bar{X})^2 \sum_{j=(i-n)k+1}^{ik}(y_j - \bar{Y})^2}} \tag{1}$$

Table 1 shows the execution time for computing the correlation coefficient on one sensor node, with two sets of random samples of size $N = 128$. We conclude that using the direct computation from Eq. 1 generates slow execution times, so it is not feasible for implementation on resource-constraint devices. Therefore, we propose a fast algorithm that updates the correlation coefficient at each step. For large data sequences (large k), the memory consumption is also reduced by storing only intermediate values (see Section 7.1 for an evaluation of the memory consumption).

The algorithm is the following. At step i, node **x** receives the buffer Y_i from node **y**. Node **x** then calculates the following sums:

1. $S_i^x = \sum_{j=(i-1)k+1}^{ik} x_j$ and $S_i^y = \sum_{j=(i-1)k+1}^{ik} y_j$
2. $\sigma_i^x = \sum_{j=(i-1)k+1}^{ik} x_j^2$ and $\sigma_i^y = \sum_{j=(i-1)k+1}^{ik} y_j^2$
3. $S_i^{xy} = \sum_{j=(i-1)k+1}^{ik} x_j y_j$

Afterward, node **x** computes the following values:

$$\bar{X}_i = \bar{X}_{i-1} + \frac{S_i^x - S_{i-n}^x}{N} \tag{2}$$

$$\bar{Y}_i = \bar{Y}_{i-1} + \frac{S_i^y - S_{i-n}^y}{N}$$

Table 1. Execution times on MSP430 microcontroller, for N=128, k=16.

Method	**Operation**	**Time** [ms]
Direct computation	Average	68.91
	Variance, covariance and correlation coefficient	275.65
	Total	**344.56**
Incremental algorithm	Auxiliary sums	0.81
	Variance, covariance and correlation coefficient	5.47
	Total	**6.28**

$$var_i(X) = var_{i-1}(X) + \frac{\sigma_i^x - \sigma_{i-n}^x}{N} - (\bar{X}_i^2 - \bar{X}_{i-1}^2) \tag{3}$$

$$var_i(Y) = var_{i-1}(Y) + \frac{\sigma_i^y - \sigma_{i-n}^y}{N} - (\bar{Y}_i^2 - \bar{Y}_{i-1}^2)$$

$$cov_i(X,Y) = cov_{i-1}(X,Y) + \frac{S_i^{xy} - S_{i-n}^{xy}}{N} - (\bar{X}_i\bar{Y}_i - \bar{X}_{i-1}\bar{Y}_{i-1}) \tag{4}$$

Finally, node **x** computes the new value of the correlation coefficient:

$$\rho_i(X,Y) = \frac{cov_i(X,Y)}{\sqrt{var_i(X)\ var_i(Y)}} \tag{5}$$

The proofs of Eq. 3 and 4 are given in Appendix A. We make the following observations:

- For all $j \leq 0$, S_j^x, S_j^y, σ_j^x, σ_j^y, S_j^{xy}, $var_j(X)$, $var_j(Y)$ and $cov_j(X,Y)$ are 0.
- If $var_i(X) = 0$ and $var_i(Y) = 0$, we take $\rho_i(X,Y) = \rho_{i-1}(X,Y)$. If $var_i(X) = 0$ and $var_i(Y) \neq 0$ or the other way around, we decrease $\rho_i(X,Y)$ with a value proportional to the positive variance.

The algorithm proves to be much faster (by a factor of about 55) than the direct computation of Eq. 1, as shown in Table 1. This result makes the implementation of the online correlation on sensor nodes possible.

4.2 Experimental Setting

We perform two types of experiments, in which we show that the proposed method yields similar results regardless of the movement characteristics:

1. The first type of experiment is intended to reproduce the movement pattern of the smart goods, in which items equipped with sensor nodes are placed in RTIs maneuvered by people. Throughout the tests, we use two RTIs on wheels, which we push on a flat surface, and also lift occasionally, similar to loading them into the trailer. For detecting joint movement, two sensor nodes are placed on the same RTI, while for separate movement, each sensor node is placed on a different RTI.

2. The second type of experiment maps to the setting where RTIs are loaded into and carried by trucks. We use instead two bicycles, which move on a flat paving. For joint movement, the two sensor nodes are placed on the same bicycle, while for separate movements, nodes are attached to different bicycles.

The sensor nodes broadcast the movement data together with the correlation coefficient calculated locally. A gateway node logs the coefficients and the samples from both sensor nodes to a computer through a serial interface.

4.3 Parameters

Table 2 lists the values of the parameters used in the experiments, which are chosen considering the platform constraints (sampling interval Δt and data size k) and the scenario particularities (time history T).

Table 2. Experimental values for correlating data from tilt switches.

Parameter	**Explanation**	**Value**
k	Size of current data sequence	16 (2s)
n	No. of data sequences in data queue	8
$N = nk$	Size of queue	128
Δt	Time unit (sampling interval)	125ms
$T = N\Delta t$	Time history	16s

4.4 Synchronization

The synchronization between two sets of data to be correlated is very important for accurately calculating the correlation coefficient. We assume that the communication delay, plus the time for processing the incoming and outgoing buffers is $<< \Delta t = 125ms$. Therefore, we ignore the time spent on communication, and we make sure that each node copies its last k samples in the local working buffer, at the moment it receives the samples from the neighboring nodes. If a transmission error occurs, the next message contains again the latest k samples. Using this method, we can achieve *implicit synchronization* between the two sensors.

5 Solution I - Tilt Switches

A ball-contact tilt switch (also referred to as ball switch or tilt switch) is a very simple and cheap sensor, used in a large range of applications for coarse movement detection. Usually, the sensor is expected to provide binary information on the status of the device it is attached to (e.g. stationary/moving).

5.1 Extracting the Movement Information

In our experiments, we are using the ASSEMTECH CW1300-1 tilt switch [2]. The price is below 2 EUR and the power consumption is approximately 2μW. Our solution is based on counting the number of contacts made by the switch ball per time unit, as the RTI is moved. We make the following observations:

1. It is possible to distinguish the starting and stopping states (acceleration and deceleration) from the constant movement.
2. The sensitivity depends on the position of the ball switch.
3. The results are reproducible with other switches of the same type. Although the actual values vary due to the inherent sensitivity differences and imperfect alignment of the sensitive axis, the movement pattern remains similar.

5.2 Experimental Results

Figures 3 and 4 show the behavior of the algorithm running on two nodes attached to RTIs moving, first together, then separately, over a period of 40 seconds. Figures 5 and 6 show the results when the nodes are attached to bicycles. The plots at the top of the figures show the correlation coefficients calculated by the sensor nodes over the time history T, while the two bottom plots show the sampled data from the tilt switches. We make the following observations:

- The sampled data from Figure 3 shows a movement pattern, while in the experiments with bicycles from Figure 5, there is no pattern. The reason is that the experiments with RTIs are based on movements combined with stationary periods, while on the bicycles the movement is rather constant.
- There is a clear distinction between the moving and stationary cases. For example, in Figure 3, during the first 4 seconds the number of ball contacts is 0, which indicates that the sensor nodes are static. Therefore, in a static situation, nodes may not need to send the whole movement buffer, but just a short indication of their state, saving thus energy.
- In Figure 4, the nodes change their status from moving together to moving separately. The transition is slower, due to the correlated time history.
- The method distinguishes between correlated and uncorrelated movements, during both types of experiments. A correlation coefficient above 0.6 indicates that the nodes move together, while if it is below 0.4, the nodes move separately.

6 Solution II - Accelerometers

MEMS accelerometers have become increasingly popular recently, due to their relatively low price compared with the performance offered. The range of applications is quite broad, from movement or free-fall detection to gaming or virtual reality, and inertial navigation systems (INS) [10]. The operating principle is based on measuring the displacement of a proof mass when an acceleration

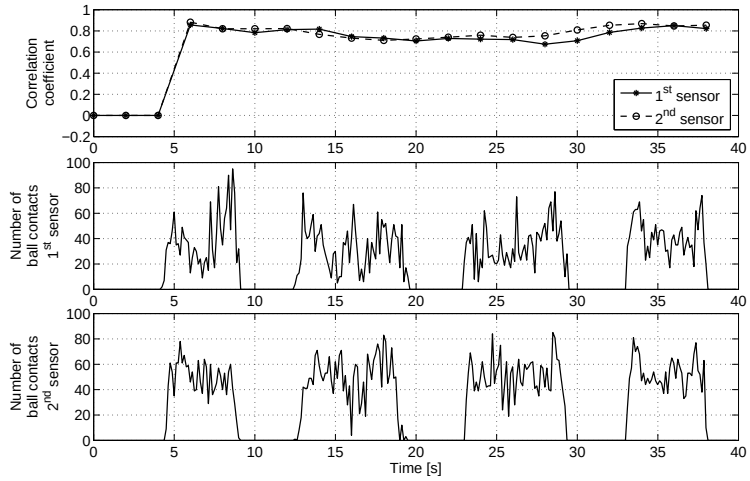

Fig. 3. Two nodes on RTIs with tilt switches moving together.

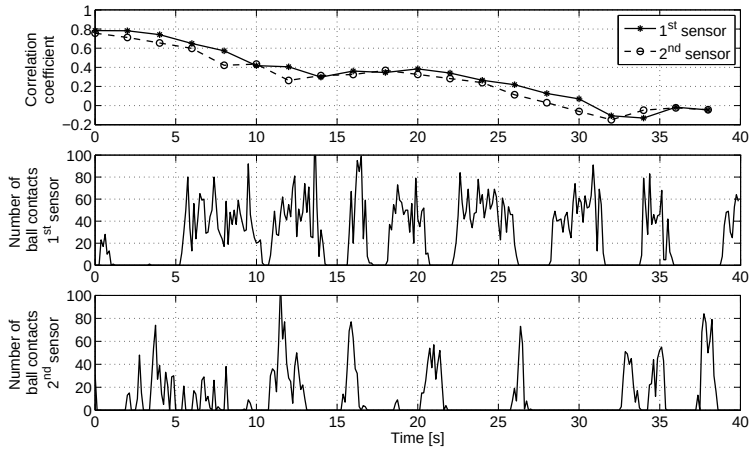

Fig. 4. Two nodes on RTIs with tilt switches moving separately.

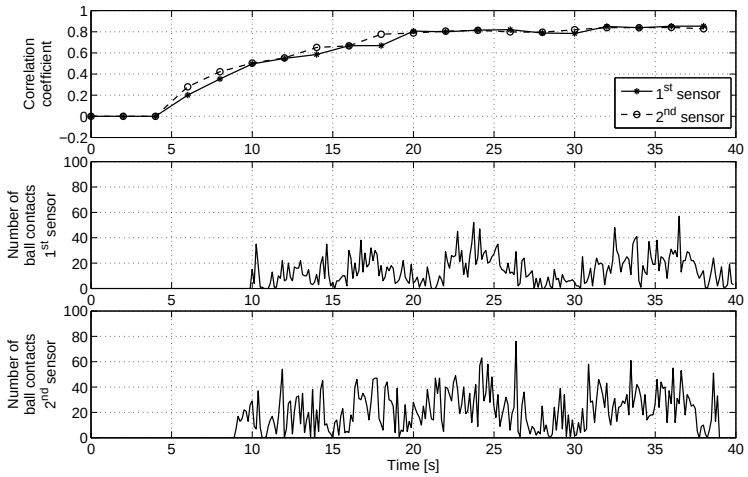

Fig. 5. Two nodes on bicycles with tilt switches moving together.

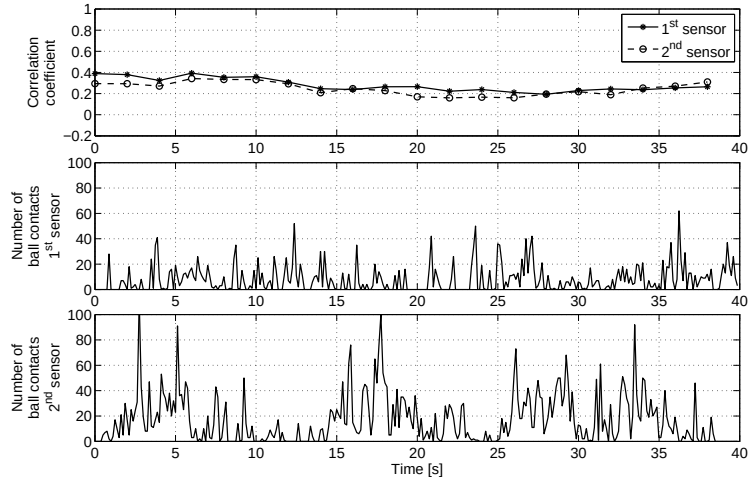

Fig. 6. Two nodes on bicycles with tilt switches moving separately.

is applied. The accelerometer measures, therefore, the applied acceleration (including gravitation), and outputs the values of the projections along its sensitive axis.

6.1 Extracting the Movement Information

By using accelerometers, it is possible to extract elaborate information about the movement, such as the speed and distance. However, to calculate the speed and position accurately, information provided by gyroscopes has to be used for maintaining an absolute positional reference. In this way, the overall complexity and price of the system increase significantly. Moreover, the accumulation of errors require elaborated filtering and prediction techniques.

From these considerations, it appears that the resource-constraint sensor nodes are not yet capable of extracting and correlating speed or distance information. Therefore, we propose a simplified solution, which considers the magnitude of the acceleration vector $\| \mathbf{a} \| = \sqrt{a_x^2 + a_y^2 + a_z^2}$. The reason is that the magnitude of the sensed acceleration is the same in any frame of reference. Consequently, the alignment and orientation of the sensors are no longer important.

In our experiments, we are using the LIS3LV02DQ three-axis accelerometer from STMicroelectronics [3]. The price is around 15 USD and the typical power consumption is 2mW. The list of features include user selectable full scale of $\pm$2g, $\pm$6g, I^2C/SPI digital interface, programmable threshold for wake-up/free-fall and various sample rates up to 2.56kHz.

6.2 Experimental Results

Figures 7 and 8 show the behavior of the algorithm running on two nodes attached to RTIs moving, first together, then separately. Figures 9 and 10 show the results when the nodes are attached to bicycles. The plots at the top of the figures show the correlation coefficients calculated by the sensor nodes over the time history T, while the two bottom plots show the magnitude of the acceleration calculated by the sensors, relative to 1g (the constant gravitational component). We make the following observations:

- In Figures 7 and 9, the nodes start from the initial state, where the auxiliary sums and coefficients are 0. The transition to the correlated state is immediate.
- In Figure 8, the nodes are first correlated, and then change their status from moving together to moving separately. The transition is slower, due to the correlated time history.
- A node can deduce that it is static by comparing the sample values with the zero offset. Therefore, similar to the tilt switch case, in the static situations nodes may just send a short indication of their state.
- The method is successful in distinguishing between correlated and uncorrelated movements, for both types of experiments. The separation interval is larger, from 0.2 to 0.6, which makes this method more reliable.

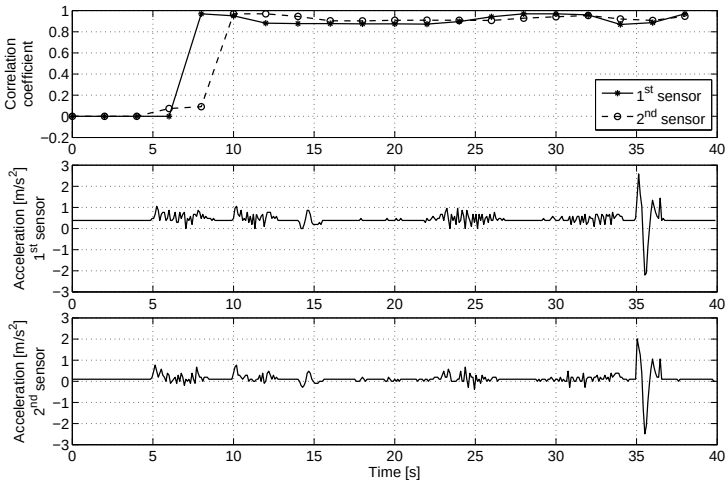

Fig. 7. Two nodes on RTIs with accelerometers moving together.

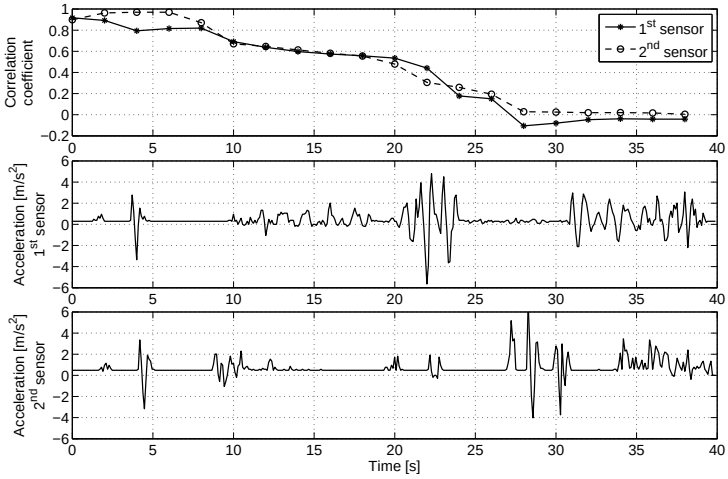

Fig. 8. Two nodes on RTIs with accelerometers moving separately.

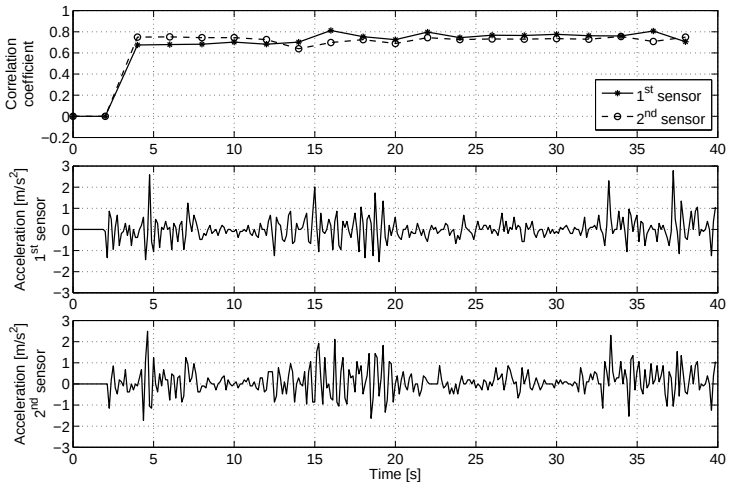

Fig. 9. Two nodes on bicycles with accelerometers moving together.

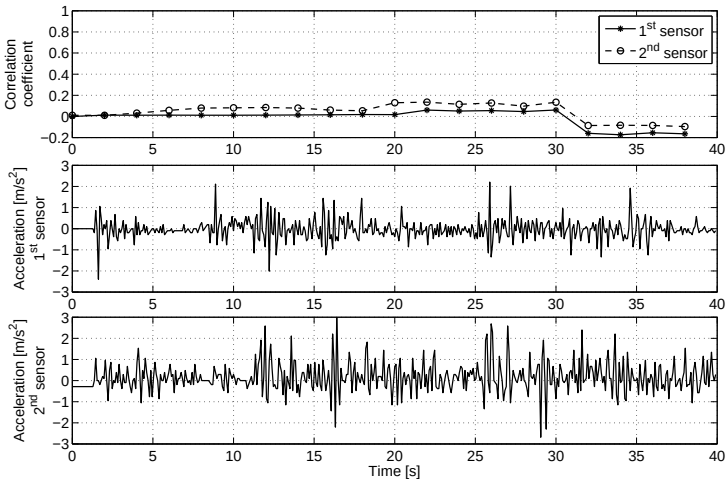

Fig. 10. Two nodes on bicycles with accelerometers moving separately.

7 Analysis

In this section, we discuss the two proposed solutions, and analyze the scalability and reliability problems, pointing out the advantages and limitations.

7.1 Scalability

We present the factors that influence the maximum number of nodes supported by our proposed correlation methods. We denote the maximum number neighboring nodes as M. It follows that a node has maximum $M-1$ neighbors, for which it computes the correlation coefficients.

Communication (Medium Access). We estimate the maximum number of neighboring nodes M as follows. Each node transmits a data sequence every $k\Delta t$. If a TDMA-based MAC protocol is used, then the frame length T_f has to be at most $k\Delta t$, so that each node has a chance to transmit the data. The slot time T_s of a node is therefore bounded by $MT_s = T_f \leq k\Delta t$. Depending on the radio chip used, the slot time for sending a data packet can be computed. In our experiments, $T_s = 20$ms, which leads to $M = 100$.

Memory. The available memory (RAM and FLASH) is usually a critical resource on sensor nodes. The FLASH usage is not a problem, since the code memory footprint of our implementation on the sensor node platform amounts to 2.1kB out of 48kB available. Considering the RAM, Table 3 shows the data structures required by the correlation method, and the associated sizes. In the case of recent low-power controllers equipped with 10kB RAM, the maximum number of nodes is $M = 106$.

Table 3. Memory requirements.

Data structure	**Size** [bytes]
Data sequence to send (X_i)	16
Received data sequence (Y_i)	16
S_i^x, S_i^y	2
σ_i^x, σ_i^y, S_i^{xy}	4
$\bar{X}_i$, $\bar{Y}_i$, $var_i(X)$, $var_i(Y)$, $cov_i(X,Y)$, $\rho_i(X,Y)$	4
Auxiliary sums	n × M × 10
Correlation data (Eq. 3-5)	$M \times 16 - 8$

Execution Time. Since the correlation algorithm runs online, the nodes must have enough time within one slot to receive and process the incoming data. It follows that the execution time T_e must be much smaller than the slot time: $T_e << T_s = T_f/M \Rightarrow M << T_f/T_e$. For the values used in our experiments, we get $M << 318$. This shows that the speed of the algorithm is not a limiting factor from the scalability point of view.

Energy. Estimating the energy consumption is always important for the battery powered sensor nodes. We consider the radio communication and sensor operation as the most costly functions in terms of energy. For communicating the sampled data, a node performs $M-1$ receptions and one transmission every frame T_f. Typical radio current consumption on Ambient μNodes is 12.8mA for reception and 11mA for transmission. In addition, the current consumed with operating the sensors is 0.64μA for tilt switches and 0.65mA for accelerometers. Figure 11 shows the operating time for a node with a typical 1000mAh battery. The running time is represented depending on the number of neighboring nodes M. The maximum values for M are deduced from the previous analysis, as being $M = 100$. As an example, for $M = 50$, the system can operate for approximatively 156 hours of continuous movement when using tilt switches and 142 hours when using accelerometers. The overall lifetime of a node is, however, much longer, as the movement periods are expected to be short (40s in our experiments) and rare (several times a day in our scenario).

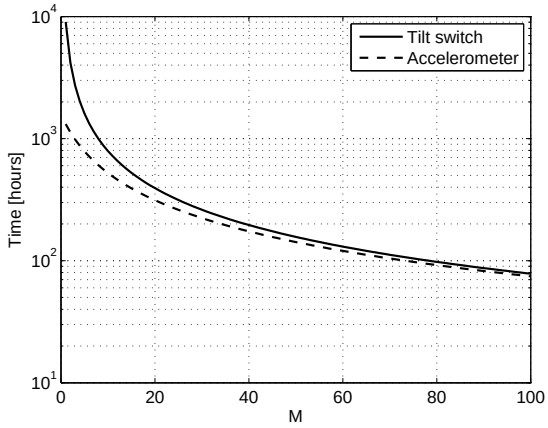

Fig. 11. The running time for continuous movement.

7.2 Discussion

In what follows, we comment on the most important advantages and limitations of both solutions, giving also comparative details whenever relevant.

Advantages

1. *Autonomous group awareness.* The proposed methods aim at establishing groups autonomously, based on common dynamic properties of the group members. No infrastructure support is needed.

2. *Simplicity.* The overall system (hardware and software) is kept very simple. This implies both a low price range and the feasibility of the implementation on resource constrained devices.
3. *Robustness to constructive differences of sensors.* The correlation coefficient gives an indication on the degree of similitude of two signals. It is known that the result is neither affected by scaling the signals with a certain factor, nor by adding/subtracting a constant value. This makes our method inherently robust to constructive differences of sensors, such as: calibration factors, zero-offset values, differences in sensitivity, etc.
4. *Clear distinction between ensemble and separate movements.* Figures 3 - 10 indicate a good behavior, with separable thresholds for distinguishing between ensemble and separate movement. However, the solution employing accelerometers is more reliable, having a larger separation interval between joint and separate movements.
5. *Implicit synchronization.* There is one important factor that can adversely affect the correctness of the correlation result, and that is time synchronization. It is therefore essential that the data sequences X_i, Y_i are synchronized when computing the correlation coefficient. For this reason, X_C is implemented as a circular buffer, so that the incoming data from neighbors is correlated with the latest values sampled on the current node. Moreover, it is preferable not to use any retransmission mechanisms, since occasional packet losses do not affect the synchronization.
6. *Saving power while stationary.* In static situations, nodes can save energy by transmitting just a short indication of their status.
7. *Extended features.* More accurate results may be obtained by correlating extended movement features, such as direction or heading, speed, distance, etc. In this sense, the accelerometer-based solution is much richer in possibilities.

Limitations

1. *Energy efficiency versus accuracy.* When the nodes are moving, they are expected to exchange a large amount of data compared with the static mode. Consequently, the energy consumption on sensing, communicating and processing is significantly higher for movement situations. However, more movement and faster sampling rates yield more granular correlation results, so better accuracy.
2. *Alignment and orientation.* Because movement is always relative to a frame of reference, different alignment or orientation of the sensors may lead to misleading results. In the case of tilt switches, a similar alignment is necessary for obtaining a correct behavior, such as in Fig. 3. In contrast, for the accelerometer-based solution, no alignment is needed, since we are correlating the magnitude of the acceleration vector.
3. *Reliability.* One of the major questions is how reliable and precise are the proposed methods. Quantitative results, such as the ratio of false positives and negatives, are not feasible at the moment, as they require large-scale experiments. Nevertheless, several qualitative considerations can be made.

First, there are separable thresholds for distinguishing between ensemble and separate movement for both tilt switches and accelerometers. Second, the solution using accelerometers is more reliable due to larger separation interval. Finally, in order to achieve better results, data from more sensors can be combined.

8 Conclusions

This paper proposes a method for constructing dynamic groups based on movement information. Nodes are considered part of the same group if their movement correlate for a certain amount of time. For extracting the movement information we investigate two solutions, one using tilt switches, the other one using accelerometers. On the one hand, the solution using tilt switches proves to be cheaper, simpler and less energy consuming. On the other hand, the solution using accelerometers is more reliable in distinguishing between ensemble and separate movements and it does not need any sensor alignment. Nevertheless, the solution is more complex, as the magnitude of the acceleration has to be calculated from the three samples corresponding to the three axes. The scalability analysis shows a maximal network density of 100 nodes for both solutions.

For future work, we intend to test our solution with different types of movements, including various accelerations and vibrations. A large-scale experiment is required, in order to give quantitative estimates of the reliability. Since such an experiment may imply a multi-hop network, and because it is not feasible to propagate the movement data over multiple hops, we plan to investigate the reliability of the solution when using a transitive correlation relation. We further intend to make use of other types of sensors for improving reliability and achieving better results.

9 Acknowledgments

We would like to thank Tjerk Hofmeijer and Stefan Dulman for their assistance with the Ambient μNode platform, Marlous Weghorst for her support during the experiments and Pieter Hartel for reviewing this paper.

References

1. Ambient Systems. http://www.ambient-systems.net.
2. ASSEMTECH CW1300-1 ball-contact tilt switch. http://www.farnell.com/datasheets/67723.pdf.
3. STMicroelectronics LIS3LV02DQ 3-axis linear accelerometer. http://www.st.com/stonline/products/literature/ds/11115.pdf.
4. Dipanjan Chakraborty, Anupam Joshi, Tim Finin, and Yelena Yesha. GSD: A novel group-based service discovery protocol for MANETs. In *MWCN'02*, pages 140–144. IEEE, September 2002.

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. Gary Hoi Kit Lam, Hong Va Leong, and Stephen Chi fai Chan. Gbl: Group-based location updating in mobile environment. In *DASFAA*, pages 762–774, 2004.
7. 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, 2004.
8. M. Marin-Perianu, N. Meratnia, M. Lijding, and P. J. M.Havinga. Being aware in wireless sensor networks. In *15th IST Mobile and Wireless Communication Summit, Capturing Context and Context Aware Systems and Platforms Workshop*, Myconos, Greece, 2006.
9. 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.
10. C.W. Tan, S. Park, K. Mostov, and P. Varaiya. Design of gyroscope-free navigation systems. In *Intelligent Transportation Systems, Oakland, USA*, pages 286–291, 2001.
11. Mehmet C. Vuran, Ozgur B. Akan, and Ian F. Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259, 2004.

A Appendix

The proofs of Eq. 3 and 4 are the following:

$$
\begin{aligned}
var_i(X) = \sum_{j=(i-n)k+1}^{ik} \frac{(x_j - \bar{X}_i)^2}{N} = \sum_{j=(i-n)k+1}^{ik} \frac{x_j^2}{N} - \bar{X}_i^2 = \\
= (\sum_{j=(i-n-1)k+1}^{(i-1)k} \frac{x_j^2}{N} - \bar{X}_{i-1}^2) + \sum_{j=(i-1)k+1}^{ik} \frac{x_j^2}{N} - \\
- \sum_{j=(i-n+1)k+1}^{(i-n)k} \frac{x_j^2}{N} - \bar{X}_i^2 + \bar{X}_{i-1}^2 = \\
= var_{i-1}(X) + \frac{\sigma_i^x - \sigma_{i-n}^x}{N} - (\bar{X}_i^2 - \bar{X}_{i-1}^2)
\end{aligned}
$$

$$
\begin{aligned}
cov_i(X,Y) = \sum_{j=(i-n)k+1}^{ik} \frac{(x_j - \bar{X}_i)(y_j - \bar{Y}_i)}{N} = \sum_{j=(i-n)k+1}^{ik} \frac{x_j y_j}{N} - \bar{X}_i \bar{Y}_i = \\
= (\sum_{j=(i-n-1)k+1}^{(i-1)k} \frac{x_j y_j}{N} - \bar{X}_{i-1} \bar{Y}_{i-1}) + \sum_{j=(i-1)k+1}^{ik} \frac{x_j y_j}{N} -
\end{aligned}
$$

$$- \sum_{j=(i-n-1)k+1}^{(i-n)k} \frac{x_j y_j}{N} - \bar{X}_i \bar{Y}_i + \bar{X}_{i-1} \bar{Y}_{i-1} =$$

$$= cov_{i-1}(X, Y) + \frac{S_i^{xy} - S_{i-n}^{xy}}{N} - (\bar{X}_i \bar{Y}_i - \bar{X}_{i-1} \bar{Y}_{i-1})$$