Wireless Netw (2010) 16:141–152
DOI 10.1007/s11276-008-0120-7

# Mobility control for achieving optimal configuration in wireless sensor networks

**Zhen Jiang · Jie Wu · Robert Kline**

Published online: 28 June 2008


**Abstract** Recent work in wireless sensor networks, or simply called WSNs, has drawn attention to the mobility capability of each node. In Stojmenovic and Lin (IEEE Trans Parallel Distrib Syst 12: 1023–1032, 2001), it is proved that the optimal positions of the relay nodes along a single active flow must lie entirely on the line between the source and destination with each node spaced evenly along such a line. Based on this, we propose two practical solutions to control the relay nodes in WSNs to approach their optimal positions in the local relative coordinate system. One uses one-hop neighbor information and the other one uses two-hop neighbor information. Basically, each relay node will approach the midpoint on the line composed of neighbors. For the latter control scheme, we also discuss its different implementation with outdated two-hop neighbor information (lagged by one-round neighbor information exchange and update). This is an improvement since given nodes only reuse the two-hop neighbor information previously saved at its one-hop neighbors and does not require any extra neighbor information collection. All the new methods prevent oscillations by demanding minimal moving distance per round (*MDPR*), otherwise the node does not move. Unlike the one presented in Goldenberg et al. (Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc'04), pp 163–174 2004) using only one-hop neighbor information, our methods will converge more quickly. The experimental results show a substantial improvement on the speed of achieving the optimal configuration and the total moving distance of nodes.

**Keywords** Wireless sensor networks (WSNs) · Distributed algorithm · Information model · Mobility control · Self-configuration

Z. Jiang (✉) · R. Kline
Department of Computer Science, Information Security Center, West Chester University, West Chester, PA 19383, USA
e-mail: zjiang@wcupa.edu

R. Kline
e-mail: rkline@wcupa.edu

J. Wu
Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA
e-mail: jie@cse.fau.edu

## 1 Introduction

As the capability of mobility becomes more readily available to nodes in WSNs [14, 17], recent research work has drawn attention to control schemes in order to achieve optimal configuration in data flows for improving communication performance [8, 12, 21–24]. The optimal configuration of a single active flow is established in [19] and then in [8], a complete distributed iterative scheme is proposed to move each node toward its optimal position. Simply, in a synchronous round-based system, every node is required to compute the average of its two neighbors and then move to that new position at each round. As we will discuss later in this paper, the round-off error [1] that is affiliated with this method may cause the oscillation problem. In [8], instead of reaching the expected target position, the node only moves toward that point. The movement is damped and will not suffer from many oscillations. But the oscillation problem is not solved completely. Indeed, the damping slows down the convergence and causes a delay in achieving optimal configuration. This will decrease the

effectiveness of such mobility control in some real time applications, such as target tracking systems [5, 9, 25], in which the tracking data must be transferred quickly enough to catch up with the moving target. Thus, providing a practical scheme which can form a stable data flow quickly while keeping the relay nodes at or close to their optimal positions is becoming increasingly important.

A short summary of our approach follows. First, in the averaging process using 1-hop neighbor information [8], the minimum moving distance per round, *MDPR*, as a certain parameter for each node deployed, is introduced to avoid the oscillation caused by the round-off error. Instead of using the damping process, a node will reach its target position immediately. However, it will be moved only if the distance from the current position to the target position is larger than this *MDPR*. As a result, the nodes will be moved close to their optimal positions and such an averaging process, denoted by *MC*1, will converge in a few rounds without oscillation. Then, by using 2-hop neighbor information, the averaging process, which is denoted by *MC*2, can converge more quickly and each node has less movement. Finally, to reduce the cost in collection and distribution of such 2-hop neighbor information, an averaging process using inconsistent information *MC*2*A* is presented. Our experimental results show the substantial improvement of our control schemes on the number of rounds needed in the converging process in the synchronous round-based system (i.e., the speed of achieving stable configuration) and the corresponding total moving distance of nodes. The results also show that our local relative coordinate system schemes will achieve a position configuration very close to the optimal one (*MCM*) while the memory requirement for storing location information can be reduced greatly due to the use of local views of neighborhood.

In this paper, we focus on a way to collect and distribute the location information in order for the above averaging algorithm to converge quickly. The location is discovered in the GPS-free positioning algorithm [4]. By using the relative coordinate system, we can keep the connectivity of neighbors during the node movements, which is a key performance aspect in sensor location adjustment indicated by [7, 15, 16]. Relying on such systems will bring a more stable solution without any disconnection. It is noted that in *MCM*, *MC*1, *MC*2 and *MC*2*A*, the relay node will reach its expected target position. In this paper, we suggest not using damping in those averaging processes because it causes the final stabilized positions of relay nodes to be farther from the optimal ones. The challenge here is twofold. First, the information must be accurate enough to represent the exact configuration. The effect of the round-off error, which may cause oscillation in the implementation, must be considered. Second, the collection and distribution process must be practical and energy efficient. In other words, it must be simple, have no complex computation, and must not require any costly information which demands expensive equipment or a lot of multicasting/broadcasting.

The remainder of the paper is organized as follows: Section 2 introduces some necessary notions and preliminaries, including the related control schemes and the problems in their implementation. Our control schemes in the local relative coordinate system are presented in Sect. 3. Section 4 shows the experimental results. Section 5 concludes this paper and provides ideas for future research.

## 2 Preliminary

We assume that all the nodes have a fixed communication range $r$. The nodes inside such a range are called neighbors and two neighboring nodes are directly connected by a beaconing mechanism [13]. This range is also called *beaconing range*. To send the data in an efficient way, the power spent at each node for sending out a data packet is adjustable and will be determined by its physical distance to the target neighbor, which is called *transmission range*. Thus, a network can be represented by a simple undirected graph $G = (V, E)$, where $V$ is a set of vertices (nodes) and $E$ is a set of undirected edges. An undirected edge $(u, v)$ denotes the connection between two neighboring nodes $u$ and $v$. The *neighbor set* $N(u)$ of node $u$ is defined as $\{w|(w, u) \in E \text{ or } (u, w) \in E\}$. Each node $u$ has the location $(x_u, y_u)$, simply denoted by $L(u)$. $|L(u) - L(v)|$ is the distance between two nodes $u$ and $v$ · $L'(u)$ denotes the target location of $u$ in its movement. $L_u(v)$ is the location of node $v$ in the relative coordinate system at node $u$. Specifically, $L_u(u) = (0, 0)$.

Our mobility control algorithm is orthogonal to the routing discovery protocol. We assume that a path from the source $s$ to the destination $d$ has been discovered using a routing protocol, e.g., a greedy routing protocol or one of the ad hoc routing protocols. We also assume that neither $s$ nor $d$ can move during the averaging process. Otherwise, the path is always broken and a new routing path is needed, regardless of whether the averaging algorithm is applied or not. We label the nodes from the source to the destination 0, 1, … ,$n$. We call node $u_0$ the source, node $u_n$ the destination, and nodes $u_1, \ldots, u_{n-1}$ relay nodes. For each relay node $u_i$, we have $u_{i-1}, u_{i+1} \in N(u_i)$. Thus, any information of $u_i$ can be shared with $u_{i-1}$ and $u_{i+1}$ by the information exchange among neighboring nodes in the beaconing process. To simplify the discussion, we describe the schemes in a synchronous round-based system. All the schemes presented in this paper can be extended easily to an asynchronous round-based system. However, to make our schemes clear, we do not pursue the relaxation.

After each relay node knows its position, the optimal configuration of relay nodes for a single active flow is established in [19] as follows. Assume that the energy cost function is a non-decreasing convex function.

**Algorithm 1** (*MCD*): Mobility control at each relay node $u_i$ [8].

1. Exchange $L(u_i)$ with $u_{i-1}$ and $u_{i+1}$.
2. Receive $L(u_{i-1})$ and $L(u_{i+1})$. Set $L'(u) = \frac{L(u_{i-1})+L(u_{i+1})}{2}$.
3. Set damping factor $g$ a random value $\in (0,1]$, move toward $L(u_i) + g \cdot (L'(u_i) - L(u_i))$.

Then the optimal positions of the relay nodes must lie entirely on the line between $s$ and $d$. Furthermore, the relay nodes must be evenly spaced along the line. A uniform distributed algorithm that allows the relay nodes to move to their optimal position is also introduced in [8] (see Algorithm 1). The key ingredient of this algorithm is the simple averaging step. Note that although a relay node computes the average of its two neighbors, the node only moves toward this point, instead of reaching it in one round. In other words, the movement is damped. The damping process is used to avoid over-reaction of each node. Such an algorithm is denoted by *MCD*.

Goldenberg et al. [8] also claimed that *MCD* will converge and eventually evenly distribute all the relay nodes on the line between $s$ and $d$. However, in the final converging stage, it only requires a node to move a very short distance. Because of the round-off error, such a distance cannot be expressed precisely in most computer languages, such as *C*, and will cause inappropriate round-in or round-out. For example, a 5-hops path contains nodes $s$(92.···34, 3.···32), $u_1$(86.···14, 9.···64), $u_2$(80.···93, 16.···95), $u_3$ (74.···75, 22.···28), $u_4$(69.···55, 28···60), and $d$(63.···37, 34.···94), where *XX*.···*YY* stands for the coordinate value beginning with *XX* as integer part and ending with *YY* in its decimal part. In the first round, node $u_1$ is expected to move to (86.···135, 9.···635). However, the target location of $u_1$ comes out as (86.···13, 9.···63) because a round-off error occurs. Respectively, $u_2$, $u_3$, and $u_4$ will move to (80.···94, 16.···96), (74.···74, 22.···27), and (69.···56,28.···61). In the next round, all the nodes will bounce back to the positions in the previous round; that is, an oscillation occurs. According to our experimental results, the oscillation will occur in most cases (>70%) when $g$ is fixed and set to 1. Our experimental results also show that when $g$ is randomly generated, the percent chance of achieving stabilization can be improved, but the problem cannot be solved completely.

Alternatively, after the locations $L(s)$ and $L(d)$ in the absolute coordinate system are collected at a relay node $u_i$, its optimal position can be determined as $L'(u_i) = L(s) + i \cdot \frac{L(d)-L(s)}{n}$, and this node can move to the optimal position directly without oscillation. Such an algorithm is denoted by *MCM* and its details are shown in Algorithm 2.

**Algorithm 2** (*MCM*): Mobility control with minimum total moving distance.

1. The source node $s$ sends $L(s)$ and its label 0 to $u_1$. When each relay node $u_i$ receive $L(s)$ and the label $i-1$, it will pass $L(s)$ and its own label $i$ to the succeeding node along the path. Such a propagation will end at $d$.
2. Once $L(s)$ is received at the destination node $d$, send a message carrying $L(d)$ back to $s$ along the path.
3. At each relay node $u_i$, once both $L(s)$ and $L(d)$ are received, set $L'(u_i) = L(s) + i \cdot \frac{L(d)-L(s)}{n}$ according to the result of [19] and move $u_i$ to $L'(u)$.

First, a message is initiated at $s$. Along with $L(s)$, this message will be propagated to $d$ in iterative rounds. In each round, it will advance one hop along the path and the label counter attached increases by 1. Every relay node will read out its label and the location information of $s$ from this message. When such a message reaches the destination $d$, it will carry $L(d)$ and bounce back to $s$. In this way, every relay node will know its label and receive both $L(s)$ and $L(d)$. In one round, the relay node can calculate the target location and move to its optimal position without any unnecessary movement. Therefore, *MCM* has the properties listed as follows:

**Property 1** *The total moving distance in MCM is minimum (when the routing path has been initialized).*

**Property 2** *MCM will converge in exactly $(2n - 1)$ communication rounds where $n$ is the number of nodes along the single flow.*

It is noted that step 1 in *MCM* can be integrated with the routing process when the routing message is sent from $s$ to $d$. Step 2 in *MCM* can also be integrated with a reception process of routing path acknowledgement which is sent from $d$ back to $s$. In this way, the construction of the optimal path in *MCM* may not require any extra time and can be finished before the data transmission starts. This guarantees that all the data can be transferred through the optimal path.

However, *MCM* moves the relay nodes without considering their neighbor connections. Such a movement may cause the disconnection of the network [6]. Obviously, relying on the local relative coordinate system will bring a more stable solution. Capkun et al. [4] introduces a distributed algorithm that enables the nodes to find their neighbors' positions using only one round of information exchange. Such an algorithm does not rely on GPS (Global Positioning System) [10, 11, 18] and only uses the distances between the nodes to build a relative coordinate system in which the nodes' positions are computed. In the

next section, we use this positioning algorithm to obtain the location information for the control of node mobility. We assume that the nodes are deployed densely enough for each relay node to find neighbors to apply such an algorithm.

## 3 Mobility control in the local relative coordinate system

In this section, we rewrite the control scheme using 1-hop neighbor information in [8] in the local relative coordinate system, and solve its oscillation problem by introducing the constraint of *MDPR* on mobility of each node. Then, to move the nodes closer to their optimal positions more quickly, a control scheme which requires each node keep its 2-hop neighbor information up-to-date, i.e., keep consistent 2-hop neighbor information, is presented. After that, we extend this control by introducing a more efficient method of neighbor information collection.

**Algorithm 3** *(MC1)*: Mobility control based on 1-hop neighbor information at relay node $u_i$, subject to *MDPR*.

1. Set $L_{u_i}(u_i) = (0,0)$ and build the local coordinate system.
2. Apply the procedure in [4] to obtain $\{L_{u_i}(v) | v \in N(u_i)\}$ in the local coordinate system, which only needs one round information exchange.
3. Get $L_{u_i}(u_{i-1})$ and $L_{u_i}(u_{i+1})$. $L'_{u_i}(u_i) = \frac{L_{u_i}(u_{i-1}) + L_{u_i}(u_{i+1})}{2}$.
4. If $|L'_{u_i}(u_i) - (0,0)| > MDPR$, move to $L'_{u_i}(u)$.

### 3.1 Mobility control based on 1-hop neighbor information

At each relay node $u_i$, the location of neighbors $u_{i-1}$ and $u_{i+1}$ in the relative coordinate system of $u_i$, $L_{u_i}(u_{i-1})$ and $L_{u_i}(u_{i+1})$, can be determined in the positioning algorithm in [4]. Then, we can apply the averaging algorithm in [8] to estimate the target location of $u_i$, i.e., $L'_{u_i}(u_i) = \frac{L_{u_i}(u_{i-1}) + L_{u_i}(u_{i+1})}{2}$. Without damping, node $u_i$ will reach the new position in our scheme before the next round starts. As we discussed in Sect. 2, caused by the round-off error, the relay nodes may oscillate between two consecutive positions but within a small range. To prevent the node from falling into unstoppable movement and wasting energy on such oscillation, the oscillation range is set as the threshold called *Minimum Moving Distance per Round*, *MDPR*. A relay node will move only if the distance from the current position to the target position is larger than this *MDPR*. This is sufficient to avoid oscillation while keeping the nodes close to their optimal positions. The oscillation range relies on the difference between the calculated approximation of a number and its exact mathematical value. The detailed process can be seen in Algorithm 3, which is denoted as *MC*1.

According to [2], the round-off error is caused by the difference between the approximation and the exact value. If *MDPR* is two times larger than any possible difference, we have the following theorem to ensure that the oscillation can be avoided. According to the differences listed in [2], *MDPR* can simply be set as 0.0001 (>2 * 0.00003) to cover all cases.

**Theorem 1** *If MDPR* $> 2\, d_r$, *the oscillation in averaging process can be avoided, where* $d_r$ *is the maximum difference causing round-off error.*

*Proof* Denote the exact target position of an intermediate node $u$ as $L_t(u)$ and the corresponding difference causing round-off error in node $u$'s movement as $|L'(u) - L_t(u)|$. Obviously, we have $d_r \geq |L'(u) - L_t(u)|$. If the round-off error causes the oscillation of node $u$ in the averaging process, we have $|L'(u) - L_t(u)| > 0$ and can always find $|L(u) - L_t(u)| \leq |L'(u) - L_t(u)| \leq d_r$ after certain time.

By using the *MDPR* ($>2d_r$), node $u$ changes its position in the averaging process if and only if $|L(u) - L'(u)| > MDPR$. Therefore, we have

$$|L(u) - L_t(u)| \geq |L(u) - L'(u)| - |L'(u) - L_t(u)| > MDPR - d_r > d_r.$$

In other words, the above oscillation can be avoided. □

After applying *MC*1, the relay nodes may not be located at their exact optimal positions due to the use of *MDPR*. However, we will show in the experimental results in the next section that such a gap is very small and can be ignored. Actually, the existence of such a gap can be accepted as the tradeoff cost for achieving a stable path quickly. It is noted that a relay node in *MC*1 will move exactly like the one in *MCD* when $g = 1$, because they all move to the midpoint on line composed of 1-hop neighbors, no matter whether the absolute coordinate system (in *MCD*) or the relative coordinate system (in *MC*1) is used. When the *MDPR* blocks its movement, the other nodes will move like those in *MCD* in an asynchronous round-based system. Due to the use of *MDPR*, the nodes in *MC*1 will stop moving before they reach the final converging stage in *MCD*; that is, each node will reach its stable status earlier. Thus, the properties of *MCD* discussed in [8], such as the connection of all the relay nodes, still hold in *MC*1.

### 3.2 Mobility control based on consistent 2-hop neighbor information

To quickly approach the optimal position, each relay node needs to know more accurate neighbor information for the

averaging algorithm. After collecting the location information of its 1-hop neighbors, a node can send such information back and share it with them after one more round of information exchange. In this way, the information can be exchanged between nodes that are 2 hops away. That is, 2-hop neighbor information can be collected, which is consistent with the exact location of the corresponding nodes. The details of the averaging algorithm, based on such consistent neighbor information, are shown in Algorithm 4, which is denoted by *MC*2.

In *MC*2, for any relay node $u_i$ and its 2-hop neighbor node, say node $u_{i-2}$, the location of $u_{i-2}$ is determined in the coordinate system at $u_i$. Denoted by $L_{u_i}(u_{i-2})$, such a location depends on the location of $u_{i-2}$ in the coordinate system at $u_{i-1}$ and the location of $u_{i-1}$ in the coordinate system at $u_i$. That is, $L_{u_i}(u_{i-2}) = L_{u_{i-1}}(u_{i-2}) + L_{u_i}(u_{i-1})$. Respectively, $L_{u_i}(u_{i+2}) = L_{u_{i+1}}(u_{i+2}) + L_{u_i}(u_{i+1})$. For the relay neighbor of the source and the destination, nodes $u_1$ and $u_{n-1}$, only 1-hop neighbor information is available. At these two nodes and only at these nodes, the 1-hop neighbor information is still in use.

**Algorithm 4** *(MC2)*: Mobility control based on 2-hop neighbor information at relay node $u_i$, subject to *MDPR*.

1. Same as step 1 in Algorithm 3.
2. Same as step 2 in Algorithm 3.
3. Get $L_{u_i}(u_{i-1})$ and $L_{u_i}(u_{i+1})$. Share this with nodes $u_{i-1}$ and $u_{i+1}$ by one more round information exchange. Thus, 2-hop neighbor information is collected if any: $L_{u_i}(u_{i-2}) = L_{u_{i-1}}(u_{i-2}) + L_{u_i}(u_{i-1})$ and $L_{u_i}(u_{i+2}) = L_{u_{i+1}}(u_{i+2}) + L_{u_i}(u_{i+1})$.
4. Set the target location of each relay node $u_i$ as the midpoint on line composed of nodes two hops away (i.e., $L'_{u_i}(u_i) = \frac{L_{u_i}(u_{i-2}) + L_{u_i}(u_{i+2})}{2}$). Otherwise, only 1-hop neighbor information is available and $L'_{u_i}(u_i) = \frac{L_{u_i}(u_{i-1}) + L_{u_i}(u_{i+1})}{2}$.
5. Same as step 4 in Algorithm 3.

Figure 1 shows the difference between *MC*1 and *MC*2 in a sample of 7-hop single flow ($n = 7$). Among six relay nodes, $u_1$ and $u_6$ use 1-hop neighbor information. For any of other relay nodes, $u_2$, $u_3$, $u_4$, and $u_5$, its action is based on 2-hop neighbor information in *MC*2 (see Fig. 1(b)). Their new locations are much closer to the optimal positions, compared with the ones achieved in *MC*1 (see Fig. 1(a)). In other words, the use of 2-hop neighbor information leads to a faster converging process in the averaging algorithm. In the following theorem, we prove that the connection of relay nodes still holds in *MC*2 with such a combination of information of 1-hop neighbors and 2-hop neighbors.

**Theorem 2** Connection between neighbors along the path is not lost.

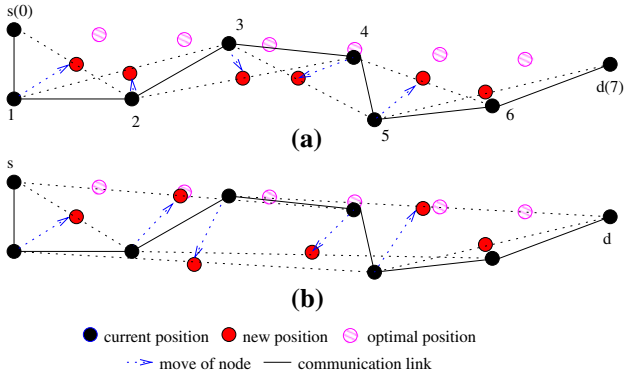



**Fig. 1** (**a**) Averaging algorithm based on 1-hop neighbor information (*MC*1). (**b**) Averaging algorithm based on 2-hop (consistent) neighbor information (*MC*2)

*Proof* Each node $i$ has the beaconing range $r$. Assume the path has been connected originally; that is, for each $i$, $0 \le i < n$, $|L_{u_i}(u_i) - L_{u_i}(u_{i+1})| < r$. We prove that any adjustment won't break the connection of $u_i$ to $u_{i+1}$ by considering the following cases.

1. When $n < 4$, only 1-hop neighbor information is used. The corresponding proof is the same as that for *MC*1 and can be found in [8]. In the following, we prove the above statement in the case where $n \ge 4$.
2. When $i = 0$ or $n-1$, only 1-hop neighbor information is used at $u_1$. The proof can be found in [8].
3. When $i = 1$, 1-hop neighbor information is used at $u_i$ but 2-hop neighbor information is used at $u_{i+1}$. Thus,

$$\begin{aligned}|L'_{u_i}(u_i) - L'_{u_i}(u_{i+1})| &= |L'_{u_1}(u_1) - L'_{u_1}(u_2)| \\ &= \left|\frac{L_{u_1}(u_0) + L_{u_1}(u_2)}{2} - \frac{L_{u_1}(u_0) + L_{u_1}(u_4)}{2}\right| \\ &= \left|\frac{L_{u_1}(u_2) - L_{u_1}(u_4)}{2}\right| \le \frac{|L_{u_1}(u_2) - L_{u_1}(u_3)|}{2} \\ &+ \frac{|L_{u_1}(u_3) - L_{u_1}(u_4)|}{2} < \frac{r}{2} + \frac{r}{2} = r.\end{aligned}$$

   when $i = n-2$, $|L'_{u_i}(u_i) - L'_{u_i}(u_{i+1})| < r$ similarly.
4. When $2 \le i \le n-4$, 2-hop neighbor information is used at both $u_i$ and $u_{i+1}$. Thus,

$$\begin{aligned}&|L'_{u_i}(u_i) - L'_{u_i}(u_{i+1})| \\ &= \left|\frac{L_{u_i}(u_{i-2}) + L_{u_i}(u_{i+2})}{2} - \frac{L_{u_i}(u_{i-1}) + L_{u_i}(u_{i+3})}{2}\right| \\ &= \left|\frac{L_{u_i}(u_{i-2}) - L_{u_i}(u_{i-1})}{2} + \frac{L_{u_i}(u_{i+2}) - L_{u_i}(u_{i+3})}{2}\right| \\ &\le \frac{|L_{u_i}(u_{i-2}) - L_{u_i}(u_{i-1})|}{2} + \frac{|L_{u_i}(u_{i+2}) - L_{u_i}(u_{i+3})|}{2} < r.\end{aligned}$$

Since nodes move along straight paths to their target points, as shown in [8], that connectivity is guaranteed not only throughout the moving period but also in the case when the *MDPR* blocks the movement of a node. □

### 3.3 Mobility control based on inconsistent 2-hop neighbor information

At each round in the above *MC*2, a relay node is required to calculate its neighbors' positions and conduct two rounds of information exchange. In a more efficient way, each node applies the positioning algorithm only once at the beginning of network construction and then uses only one round of information exchange at each round to update the position record reactively when the movement occurs. When two neighboring relay nodes share not only their updated locations but also the recorded locations of all their neighbors, a relay node can collect the location information of all its 2-hop neighbors from the corresponding 1-hop neighbors. In this way, a significant amount of positioning computation and communication for neighbor information collection can be reduced while the averaging algorithm based on 2-hop neighbor information can still be applied. The details are shown in Algorithm 5, which is denoted by *MC*2*A*.

At each round in *MC*2*A*, the update of the location of a 1-hop neighbor node is triggered by the corresponding position change in the previous round. For a relay node $u$ and its 1-hop relay neighbor $v$, when $u$ moves from (0,0) to $L'_u(u)$ in its local system, its relative coordinate system is rebuilt and the original position of $v$ can be re-calculated as $L_u(v) = L_u(v) - L'_u(u)$ (see the update of $L_{u_i}(u_{i-1})$ at node $u_i$ in Fig. 2(a)). After $u$ receives the information of the movement of $v$ in the previous round, $L'_v(v)$ , the exact location of $v$ in the view from $u$ is determined as $L_u(v) = L_u(v) + L'_v(v)$ (see the update of $L_{u_i}(u_{i-1})$ at node $u_i$ in Fig. 2(b)). Therefore, the 1-hop neighbor information is up-to-date at $u$ before the next round of averaging process is applied. It is noted that the update only occurs at relay nodes because only they can move.

**Algorithm 5** *(MC2A)*: Mobility control based on inconsistent 2-hop neighbor information at a relay node $u_i$, subject to *MDPR*.

1. Send $L_{u_i}(u_{i-1}), L_{u_i}(u_{i+1})$, and the position change of node $u_i$ in the previous round $L'_{u_i}(u_i)$ to both $u_{i-1}$ and $u_{i+1}$.
2. Receive information from $u_{i-1}$ and $u_{i+1}$ and determine their relative locations: $L_{u_i}(u_{i-1}) = L_{u_i}(u_{i-1}) + L'_{u_i}(u_{i-1})$ and $L_{u_i}(u_{i+1}) = L_{u_i}(u_{i+1}) + L'_{u_i}(u_{i+1})$.
3. Determine the 2-hop neighbor information if any: $L_{u_i}(u_{i-2}) = L_{u_{i-1}}(u_{i-2}) + L_{u_i}(u_{i-1})$ and $L_{u_i}(u_{i+2}) = L_{u_{i+1}}(u_{i+2}) + L_{u_i}(u_{i+1})$.
4. Determine $L'_{u_i}(u_i)$, same as step 4 in Algorithm 4.
5. If $|L'_{u_i}(u_i) - (0,0)| > MDPR$, move to $L'_{u_i}(u_i)$ and save the position change for updates at neighbors in the next round.
6. Rebuild the relative coordinate system of $u_i$ and update each position record according to the new position of origin point.

In *MC*2*A*, the location of 1-hop neighbors can be collected precisely. However, the 2-hop neighbor information collected in the previous round at that 1-hop neighbor may not represent the exact position in the current round (see $L_{u_{i-1}}(u_{i-2})$ in Fig. 2(b)). In other words, the 2-hop neighbor information collected via the corresponding 1-hop neighbor node in *MC*2*A* is outdated (see the update of $L_{u_i}(u_{i-2})$ in Fig. 2(b)), lagging by one round of information exchange and update. Thus, *MC*2*A* will converge as an asynchronous *MC*2. It is noted that the averaging algorithm using such inconsistent information can still achieve the configuration like *MC*2 does, but it needs a few more rounds.

Figure 3 shows a sample of the convergence of *MC*2*A* in a 7-hop data flow: $s = u_0$(2.3187, 14.5268), $u_1$(2.3187, 19.73513), $u_2$(11.17287, 19.73513), $u_3$(18.46453, 15.56847), $u_4$(27.83953, 16,61013), $u_5$ (29.40203, 21.29763), $u_6$(38.2562, 20.25597), and $d = u_7$(47.1104, 17.13097). To simplify the discussion, we assume that before the averaging process starts, there is no relay node in this flow involved in any other movement. At the first round, each relay node will receive consistent neighbor information and determine the corresponding target position as they do in *MC*2 (see Fig. 1(b)). After they move to new positions, their relative coordinate systems are rebuilt. For instance, node $u_2$ has the location information of $u_3$, $L_{u_2}(u_3) = (7.29166, -4.16666)$, before it moves to the target position $L'_{u_2}(u_2) = (3.906245,$

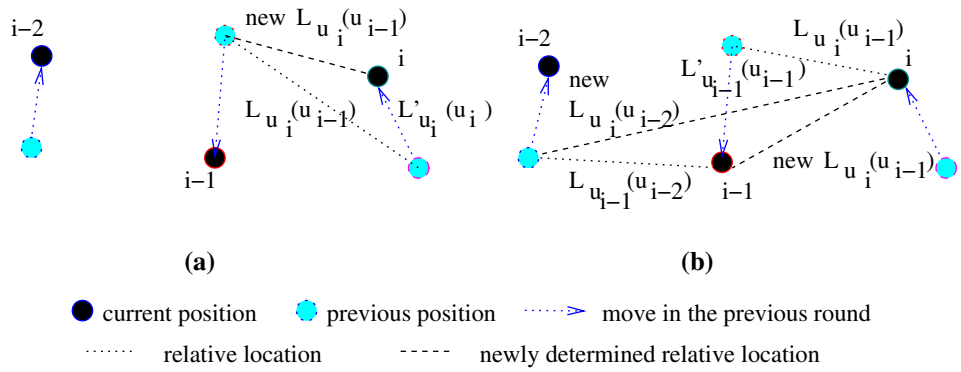



**Fig. 2** Illustration of information collection at node $u_i$ in *MC*2*A*.(**a**) Coordinate rebuilt for node $u_{i-1}$ in the previous round. (**b**) Neighbor information collected ($L_{u_i}(u_{i-1})$ and $L_{u_i}(u_{i-2})$) after information exchange

$-4.166665$). In the new coordinate system, $L_{u_2}(u_3) = L_{u_2}(u_3) - L'_{u_2}(u_2) = (3.385415, 0.000005)$. After node $u_2$ receives $L'_{u_3}(u_3) = (-2.604165, 4.94791)$ from $u_3$, $u_2$ has the consistent 1-hop neighbor information $L_{u_2}(u_3) = (0.78125, 4.947915)$ before the averaging process in round 2. However, node $u_3$ cannot detect the position change of node $u_4$ before it exchanges the location information with node $u_2$. Therefore, from the view of node $u_2$, the location of $u_4$ is not changed. As a result, $u_2$ will not change its position in the averaging process in round 2 (see Fig. 3(a)). Similarly, $u_3$, $u_4$, and $u_5$ keep their positions in the same round. In round 3, $u_1$ and $u_6$ will not change their positions because $u_2$ and $u_5$ do not change their positions in the previous round. Meanwhile, $u_2$, $u_3$, $u_4$, and $u_5$ receive the information of the movement of their 2-hop neighbors in round 1; they will make the corresponding adjustment movement via the averaging process (see Fig. 3(b)). The vacillating movement of relay nodes will continue through an asynchronous converging process (see Fig. 3(a, c)). Eventually, the flow will stabilize at $s$(2.3187, 14.5268), $u_1$(8.71743, 14.898892), $u_2$(15.1162, 15.270985), $u_3$(21.515, 15.64296), $u_4$(27.9138, 16.01499), $u_5$(34.3126, 16.38702), $u_6$(40.7115, 16.75899), and $d$(47.1104, 17.13097). The further movement of each relay node, which would make the transmission range closer to the optimal one, is avoided by the use of *MDPR* (set as 0.0001 in Sect. 3.1). For instance, $L'(u_1) = (8.71745, 14.8988925)$ and $|L'(u_1) - L(u_1)| \not> MDPR$. However, the transmission range achieved, 6.40958, is very close to the optimal one, 6.40962.

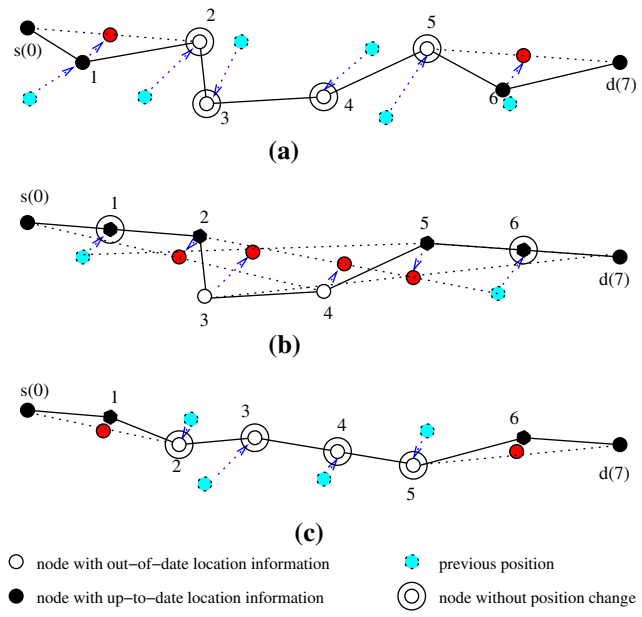



**Fig. 3** Asynchronous movements in *MC*2*A* caused by inconsistent neighbor information. (**a**) Unchanged locations caused by outdated neighbor information in round 2. (**b**) Unchanged locations caused by lagged movement of the previous round in round 3. (**c**) The use of outdated neighbor information in another round, round 4

In the above sample, each relay node has a stable status with consistent neighbor information initially. After the first round, the 2-hop neighbors' movement is detected one round late in *MC*2*A*, causing the relay node approaching the optimal position vacillatingly. Whether a relay node moves or stays stable is determined by the target location calculated from the averaging process. This depends on two factors: one is the relative positions of its two-hop neighbors and the other is the use of consistent information at each relay node in the initial round (i.e., round 1). Usually, the use of inconsistent information at a relay node in round 1 is caused by moving any of its two-hop neighbors (involved in other moving process) right before the neighbor information collection. Even when starting from the same configuration, this will cause a different sequence of alternation of node moving and stabilizing. On the other hand, when the neighbors move, the averaging result may not change and the corresponding relay node does not need to move. This will also cause a difference of converging process. In the following theorem, we prove the connection still holds in *MC*2*A* with the consideration of all these factors. Thus, by using the local relative systems, the data transmission can be conducted along the flow while the averaging process converges.

**Theorem 3** Connection between neighbors along the path is not lost.

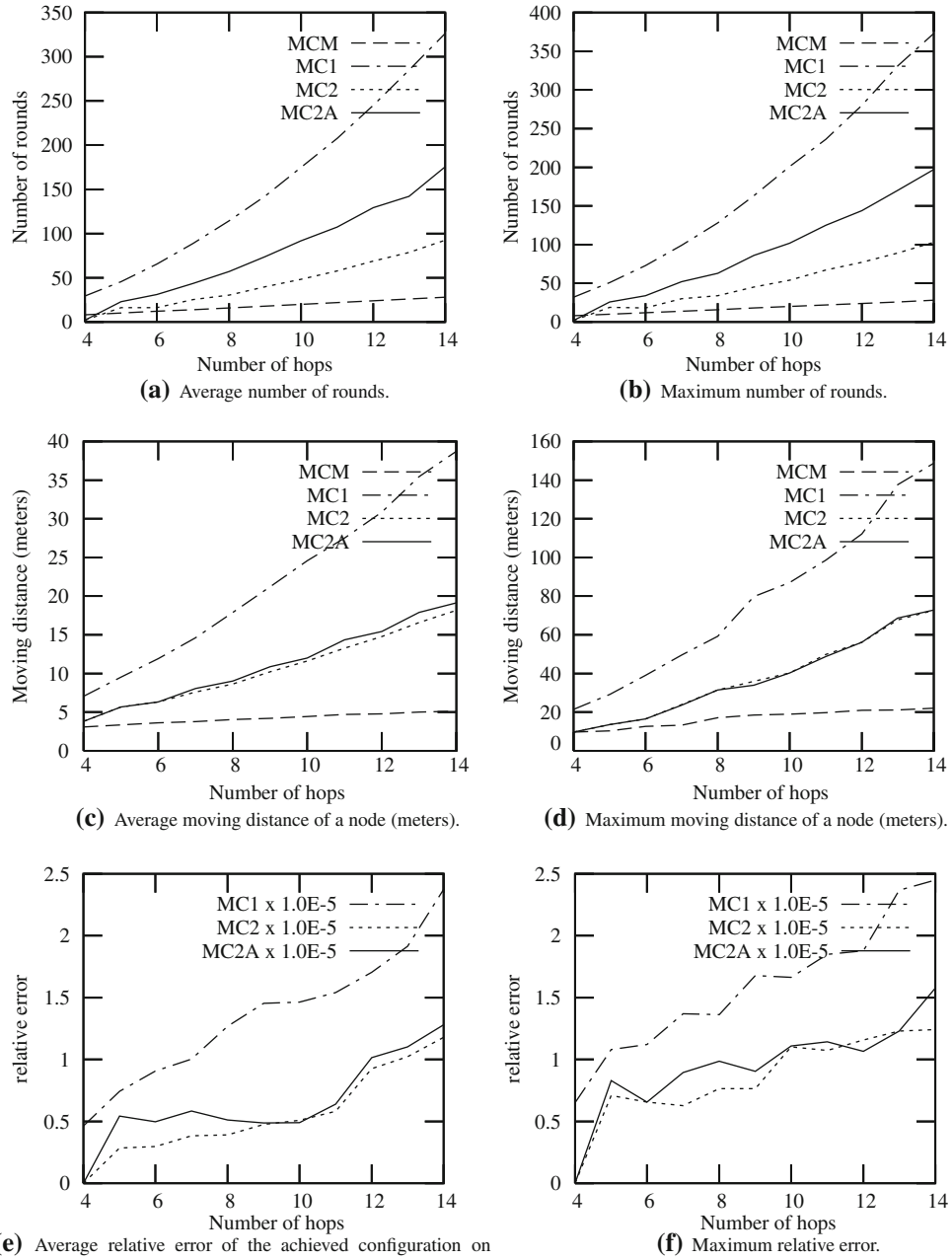
*Proof* [8] proves that the connection in *MCD* in the asynchronous round based system is not lost. Based on that, it is easy to prove that the connection in our *MC*1 and *MC*2 in the asynchronous system is not lost. The relay nodes in *MC*2*A* (which is applied in a synchronous round-based system) act as they do in an asynchronous system (as seen in the sample in Fig. 3). Thus, the connection in *MC*2*A* along the flow path will not be lost. □

For the sample shown in Fig. 1, before the configuration stabilizes, the transmission range of each hop may have been very close to the optimal one (<0.1) in 6 rounds in *MC*2 and in 10 rounds in *MC*2*A*. In the next section, we will show in the experimental results that the performance of *MC*1, *MC*2, or *MC*2*A* is very close to that of *MCM*.

## 4 Experimental results

In this section, we verify the improvement of our new schemes on speed and cost of convergence in the Monte Carlo estimate method with experimental results. In a synchronous round-based system, the speed of achieving stabilization is measured by the number of rounds needed for convergence. The cost of mobility control schemes comes primarily from the energy consumed in node

movement, which is determined by the distance a node moves. The customer simulation tool is developed in Java/C++ to test the number of rounds needed and the total distance the nodes move in flows with different length from 4-hops to 14-hops in *MCM*, *MC*1, *MC*2 and *MC*2*A* (see Fig. 4). The transmission range of each relay node eventually achieved in different schemes is also tested to compare with the minimal range in the optimal configuration. Such a comparison is shown in the rate of the relative error, which measures the ability of each scheme to achieve optimal configuration. The test is also conducted for *MCD*. However, due to the oscillation that occurs, the results are not comparable and are not shown in this paper. In the averaging process presented in [8], after the target position is determined in each movement, a node only moves toward this point, instead of reaching it in one round. The damping factor $g \in (0,1]$ is used to avoid over-reaction of each node. When the same damping factor is applied in *MC*1, *MC*2, and *MC*2*A*, the corresponding averaging processes are denoted as *MC*1*G*, *MC*2*G*, and *MC*2*AG*, respectively. According to our early discussion on *MDPR* in Sect. 3.1, *MDPR* can be set as 0.0001. In the test, we also try a larger value, 0.1. The test results of the averaging processes *MC*1, *MC*2, and *MC*2*A* and their




**(a)** Average number of rounds.

**(b)** Maximum number of rounds.

**(c)** Average moving distance of a node (meters).

**(d)** Maximum moving distance of a node (meters).

**(e)** Average relative error of the achieved configuration on transmission range to the optimal one.

**(f)** Maximum relative error.

**Fig. 4** Experimental results of MCM, MC1, MC2, and MC2A with $MDPR = 0.0001$

different implementations with damping factor (see Fig. 5) and different *MDPR* value (see Fig. 6) are compared to find a practical mobility control scheme for achieving the optimal configuration in WSNs.

In the simulation, the radius of beaconing range $r$ is set to 10 m [20] for each node. We randomly generate each single flow, with the source $s$, the destination $d$, and the corresponding intermediate nodes along path connecting them. In this way, the worst case in each routing protocol can be generated and tested. Then, we apply all the above averaging processes on those flows, and compare the results in the following figures. It is noted that a flow with 4-hops is the least case in which we can apply *MC*2 and *MC*2*A* and we will probably never see any flow longer than 14-hops in an application system with a 10-m beaconing range.

We make the following observations from the comparison shown in these figures.

1. *Optimal configuration*. With the information of node label and the locations of $s$ and $d$, each node in *MCM* will move to its optimal position directly in one round and will not make any unnecessary movement. The

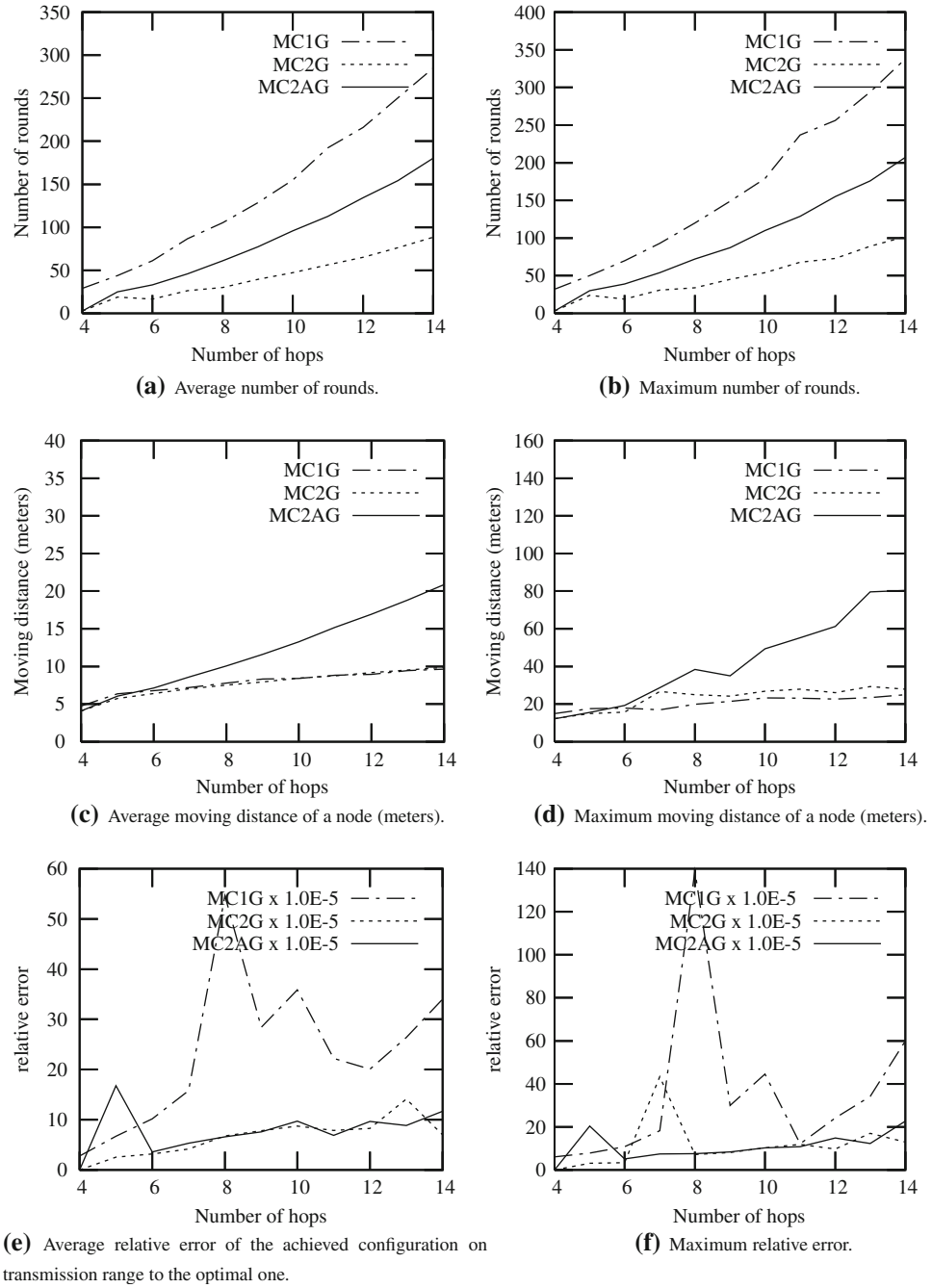



**Fig. 5** Experimental results of MC1G, MC2G, and MC2AG with $MDPR = 0.0001$

**(a)** Average number of rounds.

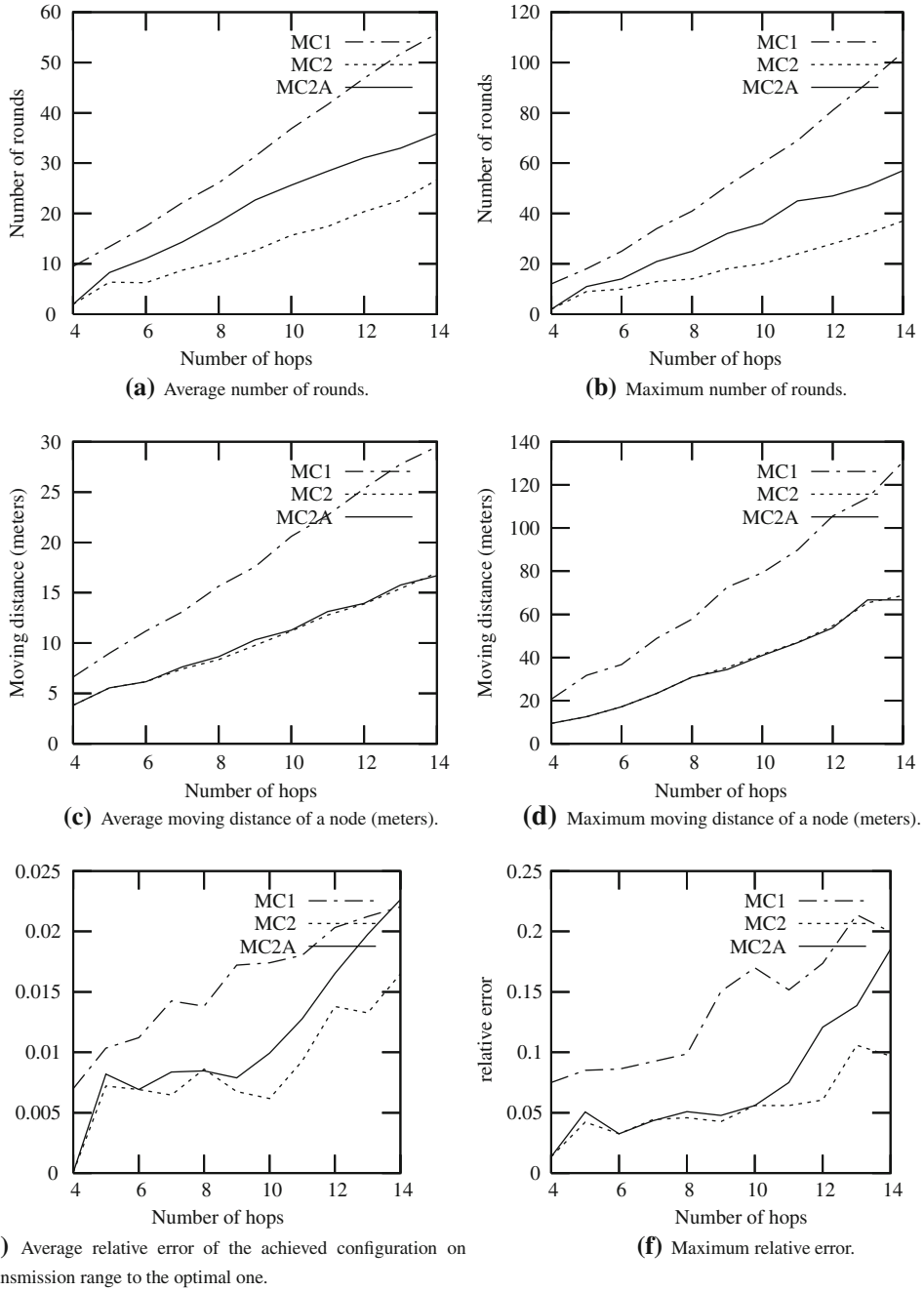
**(b)** Maximum number of rounds.

**(c)** Average moving distance of a node (meters).

**(d)** Maximum moving distance of a node (meters).

**(e)** Average relative error of the achieved configuration on transmission range to the optimal one.

**(f)** Maximum relative error.




**(a)** Average number of rounds.

**(b)** Maximum number of rounds.

**(c)** Average moving distance of a node (meters).

**(d)** Maximum moving distance of a node (meters).

**(e)** Average relative error of the achieved configuration on transmission range to the optimal one.

**(f)** Maximum relative error.

**Fig. 6** Experimental results of MC1, MC2, and MC2A with $MDPR = 0.1$

moving distance of *MCM* shown in Fig. 4(c) represents the average difference between the original configuration of a flow and its optimal one.

2. *Converging speed in averaging processes*. By using the averaging algorithm, *MC*1, *MC*2, and *MC*2*A* need more rounds to form a stable flow and waste some movement caused by the non-optimal intermediate configuration of neighbors. With 2-hop neighbor information, each node in *MC*2 and *MC*2*A* needs fewer rounds to become stable (see Fig. 4(a, b)) and makes more efficient movement (see Fig. 4(c, d)) than the one does in *MC*1. Because of the use of consistent information, the performance results of *MC*2 are better than those of *MC*2*A*. However, considering the computation and communication cost saved in *MC*2*A*, *MC*2*A* is preferred, as its performance is still acceptable.
3. *Acceptable results*. *MCM* will achieve the optimal configuration with the optimal transmission range in each hop. The experimental results show that *MC*1, *MC*2, and *MC*2*A* can move the relay nodes close to the optimal position. That is, the relative errors are all less than 0.00025 when $MDPR = 0.0001$ and are all less than 0.25 when $MDPR = 0.1$, although the *MDPR*

prevents them from reaching their expected target positions. With more accurate information, *MC*2 and *MC*2*A* can move nodes closer to optimal positions than *MC*1 does. However, due to the use of inconsistent information, the nodes in *MC*2*A* will stabilize at positions a little bit farther away from the optimal position than the nodes in *MC*2 do (see Fig. 4(e, f)).

4. *Use of MDPR*. *MCD* presented in [8] has oscillation in more than 70% of flows. All our new schemes are proved to converge within a certain number of rounds in the experimental results. They are oscillation-free and can perform better than *MCD*. *MC*1 is extended directly from *MCD* by introducing the constraint of *MDPR*. The convergence of *MC*1 and its performance shown in Fig. 4 prove the effectiveness of this *MDPR*.
5. *Damping factor*. The damping factor really works in reducing the total moving distance of relay nodes in both *MC*1 and *MC*2 (see Figs. 5(c) and 4(c)). Although its use can speed up the convergence of *MC*1, the convergence of *MC*2 is not affected so much (see Figs. 5(a) and 4(a)). It is noted that for our preferred *MC*2*A* process, the use of damping factor will not only increase the total moving distance of relay nodes but also slow down the convergence, because of the use of inconsistent (lagged) neighbor information or incorrect location information. Moreover, after the use of damping factor, the final stabilized positions of relay nodes in *MC*1, *MC*2, and *MC*2*A*, are farther from the optimal ones, compared with the ones achieved in the corresponding averaging processes without using damping factor (see Figs. 5(e) and 4(e)). As a result of our study [3]), not using the damping factor is preferred.
6. *MDPR value selection*. When the value of this *MDPR* becomes larger, the nodes may be located a little bit farther away from their optimal positions. However, the number of rounds needed for convergence and the moving distance of all the nodes will decrease greatly in our control schemes (see the corresponding results in Fig. 6).

## 5 Conclusion

In this paper, we introduce the *MDPR* in the implementation of the averaging algorithm so that oscillation caused by a round-off error can be avoided. New mobility controls using 2-hop neighbor information, consistent and inconsistent with the exact position, are proposed. The process to collect and distribute each kind of location information is also presented. In each control scheme, the connection of relay nodes is guaranteed. The experimental results of the performance and the cost of each control scheme illustrate substantial improvement of our schemes on achieving optimal configuration in a data flow. In our future work, we will apply our results to create a new routing process so that when the nodes move and form dynamic networks, not only can the connected path be constructed but also the data can be transmitted in an energy efficient way. Also, the mobility control will be extended for multiple flows as the connectivity constraints are considered in *MC*1, *MC*2, and *MC*2*A*.

**Acknowledgments** The work was supported in part by NSF grants ANI 0083836, CCR9900646, CNS 0422762, CNS 0434533, and EIA 0130806.

## References

1. http://mathworld.wolfram.com/RoundoffError.html.
2. http://en.wikipedia.org/wiki/Rounding_error.
3. http://www.cs.wcupa.edu/rkline/mobility/.
4. Capkun, S., Hamdi, M., & Hubaux, J. (2001). GPS-free positioning in mobile ad hoc networks. In *Proceeding of the 34th Annual Hawaii International Conference on System Sciences*, pp. 3481–3490.
5. Cerpa, A., Elson, J., Hamilton, M., Zhao, J., Estrin, D., & Girod, L. (2001). Habitat monitoring: Application driver for wireless communications technology. In *Proceedings of ACM SIGCOMM Workshop on Data Communications in Latin America and The Caribbean Costa Rica*, pp. 3–5.
6. Chen, X., Jiang, Z., & Wu, J. (2008). Mobility control schemes with quick convergence in wireless sensor networks. In *Proceedings of the 10th International Workshop on Advances in Parallel and Distributed Computational Models (IPDPS APDCM'08)*, (CD-ROM).
7. Cortes, J., Martinez, S., Karatas, T., & Bullo, F. (2004). Coverge control for mobile sensing networks. *IEEE Transactions on Robotics and Automation, 20*(2), 243–255.
8. Goldenberg, D., Lin, J., Morse, A., Rosen, B., & Yang, Y. (2004). Towards mobility as a network control primitive. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc'04)*, pp. 163–174.
9. Gupta, R., & Das, S. (2003). Tracking moving targets in a smart sensor network. *Proceedings of the 58th IEEE Vehicular Technology Conference (VTC'03), 5*, 3035–3039.
10. Hofmann-Wellenhof, B., Lichtenegger, H., & Collins, J. (1997). *Global Positioning System: Theory and Practice* (4th ed.). Springer-Verlag.
11. Jadbabaie, A. (2004). On geographic routing without location information. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pp. 4764–4769.
12. Li, L., & Halpern, J. (2001). Minimum-energy mobile wireless networks revisited. In *Proceedings of IEEE Conference on Communications (ICC2001), 1*, 11–14.
13. Li, M., & Liu, Y. (2007). Underground structure monitoring with wireless sensor networks. In *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN'07)*, pp. 69–78.
14. Liu, B., Brass, P., Dousse, O., Nain, P., & Towsley, D. (2005). Mobility improves coverage of sensor networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc'05)*, pp. 300–308.

15. Mao, Y., & Wu, M. (2005). Coordinated sensor deployment for improving secure communications and sensing coverage. In *Proceedings of the 3rd Workshop on Security of Ad Hoc and Sensor Networks*, pp. 117–128.
16. Poduri, S., & Sukhatme, G. (2004). Constrained coverage for mobile sensor networks. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 165–171.
17. Rodoplu, V., & Meng, T. (1999). Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications, 17*(8), 1333–1344.
18. Sichitiu, M., & Ramadurai, V. (2004). Localization of wireless sensor networks with a mobile beacon. In *Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 174–183.
19. Stojmenovic, I., & Lin, X. (2001). Power-aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems, 12*(10), 1023–1032.
20. Tian, D., & Georganas, N. (2002). A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proceedings of the 1st ACM Workshop on Wireless Sensor Networks and Applications*, pp. 32–41.
21. Wang, W., Srinivasan, V., & Chua, K. (2005). Using mobile relays to prolong the lifetime of wireless sensor networks. In *Proceedings of the 11th ACM International Conference on Mobile Computing and Networking (MobiCom'05)*, pp. 270–283.
22. Wang, W., Srinivasan, V., & Chua, K. (2007). Trade-offs between mobility and density for coverage in wireless sensor networks. In *Proceedings of the 13th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'07)*, pp. 39–50.
23. Wang, Y., Wu, H., Li, F., & Tzeng, N. (2007). Protocol design and optimization for delay/fault-tolerant mobile sensor. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS'07)*, (CD-ROM).
24. Wu, J., & Yang, S. (2007). Polylogarithmic store-carry-forwarding routing using mobile nodes. In *Proceedings of the 1st IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS'07)*, CD-ROM, available at http://www.cse.fau.edu/jie/small-world11%5B1%5D.pdf
25. Zhao, F., Shin, J., & Reich, J. (2002). Information-driven dynamic sensor colaboration for tracking applications. *IEEE Signal Processing Magazine, 19*, 68–77.

## Author Biographies

**Zhen Jiang** received the B.S. degree in Computer Engineering from Shanghai Jiaotong University, P.R. China, in 1992, the Master degree in Computer Science from Nanjing University, P.R. China, in 1998, and the Ph.D. degree in Computer Engineering from Florida Atlantic University in 2002. He is currently an associate professor in the Department of Computer Science at West Chester University of Pennsylvania, and a faculty member of the Information Security Center at West Chester University. His research interests are in the area of computer and network security, fault tolerant systems, and wireless sensor networks. He is a member of IEEE.

**Jie Wu** is a distinguished professor in the Department of Computer Science and Engineering at Florida Atlantic University and a program director at the US National Science Foundation. He has published more than 300 papers in various journals and conference proceedings. His research interests include mobile computing, routing protocols, faulttolerant computing, and interconnection networks. He was a co-guest editor of a special issue in computer on ad hoc networks. He was also an editor of several special issues of the Journal of Parallel and Distributing Computing (JPDC) and IEEE Transactions on Parallel and Distributed Systems (TPDS). He is the author of the text Distributed System Design (CRC, 1998) and is the editor of the text Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks (Auerbach, 2005). He served as an associate editor of the IEEE Transactions on Parallel and Distributed Systems and is currently on the editorial board of several international journals. He is a recipient of the 1996–1997, 2001–2002, and 2006–2007 Researcher of the Year Awards at Florida Atlantic University. He served as an IEEE Computer Society distinguished visitor and is the chairman of IEEE Technical Committee on Distributed Processing (TCDP). He is a senior member of the IEEE.

**Robert Kline** received his B.A. degree in Mathematics from Millersville University, Pennsylvania in 1974 and his Ph.D. degree in Mathematics from Washington University in 1981. He is currently an associate professor in the Computer Science Department at West Chester University, where he has been teaching and doing research in Computer Science fields since 1991. He is a faculty member of the Information Security Center at West Chester. His research interests are in the areas of computer and network security, wireless sensor networks, operating systems, and distributed programming.