Efficient Data Gathering in Partially Connected and Delay-Tolerant Wireless Sensor Networks

Ruiyun Yu
Computing Center
Northeastern University
Shenyang 110004, China
yury@cc.neu.edu.cn

Qiang Chen
College of Information Science and Engineering
Northeastern University
Shenyang 110004, China
www00a@gmail.com

Xingwei Wang
College of Information Science and Engineering
Northeastern University
Shenyang 110004, China
wangxw@ise.neu.edu.cn

Sajal K. Das
Department of Computer Science and Engineering The University of Texas at Arlington
Arlington, TX 76019, USA
das@cse.uta.edu

ABSTRACT

Sparse sensor networks have emerged in recent studies. Relaying data with the help of mobile elements seems an effective way to bridge the gaps in such networks. In this paper, we propose the Grid-Based Mobile Element Scheduling (*GBMES*) approach that schedules a mobile element (*ME*) to periodically gather data from a partially connected sensor network. The *GBMES* algorithm performs well on avoiding data loss due to buffer overflow of sensor nodes through reducing the traveling delay of *ME*, and the data transferring delay at each data gathering point.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Algorithms, Design, Experimentation

Keywords

Data gathering, Mobile element, Sparse sensor networks

1. INTRODUCTION

Wireless sensor networks are usually characterized by resources constraint and dense deployment. But some sensor networks can not be connected due to the restriction of geographic conditions, and sometimes are not necessary to be connected in many application scenarios.

QShine 2008, July 28-31, 2008, Hong Kong, Hong Kong.
 ISBN 978-963-9799-26-4
DOI 10.4108/ICST.QSHINE2008.3902

Consequently sparse networks emerged as a special class of ad-hoc and sensor networks, which have created a number of new challenges, including absence of end to end paths, large delay, network partitioning, and so on. This kind of networks are also referred to as delay tolerant networks (DTNs) [1], intermittently connected networks, or partially connected networks.

Node mobility seems the only effective way to let nodes exchange messages in such sparse networks, and several related schemes are proposed in literature [2–8].

From network connectivity point of view, most current literature focus on totally sparse networks in which there are no end to end paths between most nodes, or even every two nodes are disconnected.

In this paper, we lay our effort on collecting data from partially connected sensor networks where there are a number of connected fragments, and any two of these fragments are disconnected from each other. Fig. 1 illustrates an instance of such deployments.

To demonstrate the significance of proposing such a problem, we can imagine several scenarios, such as wildlife sanctuaries monitoring networks, partially destroyed intrusion detection networks, and battlefield surveillance networks. In wildlife sanctuaries, we probably can't afford the expense of deploying a whole-area monitoring network, so an alternative is to build networks in some key zones. Intrusion detection networks and battlefield surveillance networks are perhaps partitioned into a few fragments due to attacks.

For dealing with such scenarios, we propose the Grid-Based Mobile Element Scheduling (*GBMES*) algorithm in which a mobile element *(ME)* is introduced to fulfill the task of collecting data from partially connected sensor networks.

The main purpose of our scheme is to avoid data loss due to buffer overflow of sensor nodes through scheduling the movement of *ME*, scaling the grid cell size, or adjusting some parameters of sensor nodes, such as data transmission rate, buffer size, sampling rate, etc. We evaluate the impact of these metrics on sensor buffer overflow in simulations, and the results demonstrate that the *GBMES* algorithm performs well and has high scalability.

The remainder of this paper is organized as follows. We survey some of related schemes in section 2. In section 3, we present the Grid-Based Mobile Element Scheduling (*GBMES*) algorithm. Then we evaluate the performance of the *GBMES* algorithm through simulations in section 4. Section 5 concludes this paper.

2. RELATED WORK

Vahdat and Becker proposed the Epidemic Routing in [2] in which mobile hosts move randomly from one connected portion to another and exchange messages whenever they meet, until the messages flood the network or reach the destination.

The approach proposed by Li and Rus [3] extends the concept of an "active message" introduced by [9]. The authors present two methods to change the trajectories of mobile nodes for transmitting messages in disconnected ad-hoc networks.

Different from the approaches in [2] and [3], the schemes presented in [4–8, 10–12] all impose a mobile-element layer on sparse networks for network connection or data collection.

Shah et al. in [4] model a three-tier architecture for sparse sensor networks where they exploit mobile data MULEs to bridge the gaps between sinks and sensors.

Zhao et al. propose several approaches which use mobile message ferries to relay messages in sparse and delay-tolerant networks. Literature [5] introduces the idea of message ferrying and studies its use in networks with stationary nodes. Networks with mobile nodes are considered in [6]. Literature [7] focuses on controlling the mobility of multiple message ferries for performance and robustness concerns.

The Mobile Element Scheduling (*MES*) problem is proposed in [10]. It considers a sensor network where sensor nodes operate at different sampling rates.

Gu et al. [8] propose the Partitioning-Based Scheduling (*PBS*) algorithm that schedules the movement of a mobile element for data harvesting, so that there is no data loss due to buffer overflow.

Harras and Almeroth [11] discuss inter-regional messenger scheduling in delay tolerant mobile networks.

Tariq et al. [12] propose the optimized way-points ferry routing method for designing message ferry routes in sparse networks where nodes has arbitrary movement.

Among the related work mentioned above, only literature [11] has the same network topology as ours, but it focuses more on scheduling the mobile messengers to provide inter-regional communication.

The *PBS* algorithm [8] and the *MES* problems [10] also lay much effort on reducing data loss caused by sensor buffer overflow as the *GBMES* algotithm does. But both schemes schedule mobile elements to traverse all sensor nodes for data gathering, which is not realistic in a large scale sensor network, and is not appropriate for our scenarios.

3. GRID-BASED MOBILE ELEMENT SCHEDULING ALGORITHM

3.1 Problem Description and Methodology

The network model of our scenario is shown in Fig. 1, where a *mobile element (ME)* travels along a carefully designed route, and periodically gathers data from all regions of the sensor network.

The sensor nodes are assumed static, and have equal transmission range. The *ME* has the same transmission range as regular sensor nodes, but it has sufficient energy, storage and processing capability.

All sensors and *ME* are aware of their own location through localization approaches, and every sensor node is equipped a buffer for caching data.

The Grid-Based Mobile Element Scheduling (*GBMES*) algorithm first partitions the sensor field into square grid cells. Then a *mobile element (ME)* periodically traverses all grid cells for data collection.

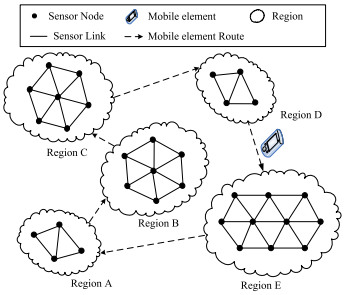


Figure 1: The network model

The goal of *GBMES* is to eliminate data loss due to buffer overflow of sensor nodes.

Suppose there are m regions in the network, the length of the inter-regional route is L_t, and the length of the intra-regional route in region i is ${L_s}^i$. The number of grid cells in all regions is assumed to be n. Let ${t_s}^j$ denote the data transferring delay of *ME* in grid cell j, and v denote the speed of *ME*. Then total time T_r for *ME* traveling a round (returning to the starting point) can be calculated through equation 1.

$$T_r = \frac{L_t + \sum_{i=1}^{m} {L_s}^i}{v} + \sum_{j=1}^{n} {t_s}^j \tag{1}$$

To avoid buffer overflow at node k, the following condition should be satisfied:

$$B_k \geqslant {r_s}^k \cdot T_r = \frac{{r_s}^k (L_t + \sum_{i=1}^{m} {L_s}^i)}{v} + {r_s}^k \sum_{j=1}^{n} {t_s}^j \tag{2}$$

Where B_k is the buffer size of node k, and ${r_s}^k$ is the sampling rate of node k.

According to equation 2, buffer overflow could be avoided by decreasing the sampling rate of sensor node, the length of traveling route, or the data transferring delay of *ME* in each grid cell. Increasing the speed of *ME* will also achieve the objective.

For the purpose of avoiding data loss due to buffer overflow, on one hand, we design a shortest route for *ME* to travel so that the traveling time is minimized, on the other hand, we introduce an efficient data gathering scheme that reduces the data transferring delay.

3.2 Grid Partitioning

In this part, we first partition the network into grid cells using *Grid Partitioning* technique, which is the basis of the *GBMES* algorithm.

In grid partitioning (*GP*) approach, each sensor maintains two predefined parameters: α (the size of each grid cell) and (X_{origin}, Y_{origin}) (the coordinate of the origin in a cartesian coordinate system).

Here we use a term *grid point* to denote the geometric center of each grid cell, and each grid cell is uniquely marked by the *grid point*. Suppose that the coordinate of a *grid point* is (X_c, Y_c). Then it satisfies equation 3.

$$X_c = X_{origin} + (i + 1/2) \cdot \alpha \tag{3a}$$
$$Y_c = Y_{origin} + (j + 1/2) \cdot \alpha \tag{3b}$$
$$(i, j = 0, 1, 2...)$$

In the process of grid partitioning, every sensor node calculates to know which grid cell it belongs to, and how far it is to the *grid point* of the grid cell.

A sensor node calculates the coordinate of its *grid point* through its own coordinate (X_s, Y_s) , and α (see equation 4).

$$X_c = \lfloor X_s/\alpha \rfloor \cdot \alpha + \alpha/2 \tag{4a}$$
$$Y_c = \lfloor Y_s/\alpha \rfloor \cdot \alpha + \alpha/2 \tag{4b}$$

The distance from the sensor node (X_s, Y_s) to its *grid point* is marked with d_{sc}, and its value is:

$$d_{sc} = \sqrt{(X_s - X_c)^2 + (Y_s - Y_c)^2} \tag{5}$$

Here another term *grid node* is introduced to denote the nearest sensor node to the *grid point* in the same grid cell compared with all its one-hop and two-hop neighbors, and d_{sc} of a *grid node* must be less than the transmission range of sensor nodes (marked with γ).

3.3 Traveling Route Design

Our goal of this section is to design a shortest loop tour for the mobile element (*ME*) collecting data periodically.

To simplify the route design issue, we assume that each region in the network has the approximate shape of a rectangle or a square whose sides are parallel to the axes of the coordinate system.

The route design problem is divided into two sub-problems: inter-regional design and intra-regional design.

3.3.1 Inter-Regional Design

The inter-regional route is designed to visit each region exactly once that has a minimum route length. Such a problem can be reduced to the traveling salesman problem (*TSP*). A number of algorithms are proposed in [13], [14], and [15] to solve this kind of problems.

Given the boundaries of *region k*, the grid domain that covers *region k* can be decided through calculating the *grid points* ($P_k{}^{bl}$, $P_k{}^{tl}$, $P_k{}^{tr}$, $P_k{}^{br}$) on the four corners of *region k*, where $P_k{}^{bl}$ is the *grid point* on the bottom left-hand corner of *region k*, $P_k{}^{tl}$ is the *grid point* on the top left-hand corner, $P_k{}^{tr}$ is the *grid point* on the top right-hand corner, and $P_k{}^{br}$ is the *grid point* on the bottom right-hand corner. We term these *grid points* as *corner grid points*.

$$(X_{P_k{}^{bl}}, Y_{P_k{}^{bl}}) = \begin{cases} (\lfloor V_k{}^l/\alpha \rfloor \cdot \alpha + \alpha/2, \lfloor H_k{}^b/\alpha \rfloor \cdot \alpha + 3\alpha/2), \\ \qquad V_k{}^l < X_c + \gamma, H_k{}^b \geqslant Y_c + \gamma & \text{(6a)} \\ (\lfloor V_k{}^l/\alpha \rfloor \cdot \alpha + 3\alpha/2, \lfloor H_k{}^b/\alpha \rfloor \cdot \alpha + 3\alpha/2), \\ \qquad V_k{}^l \geqslant X_c + \gamma, H_k{}^b \geqslant Y_c + \gamma & \text{(6b)} \\ (\lfloor V_k{}^l/\alpha \rfloor \cdot \alpha + 3\alpha/2, \lfloor H_k{}^b/\alpha \rfloor \cdot \alpha + \alpha/2), \\ \qquad V_k{}^l \geqslant X_c + \gamma, H_k{}^b < Y_c + \gamma & \text{(6c)} \\ (\lfloor V_k{}^l/\alpha \rfloor \cdot \alpha + \alpha/2, \lfloor H_k{}^b/\alpha \rfloor \cdot \alpha + \alpha/2), \\ \qquad V_k{}^l < X_c + \gamma, H_k{}^b < Y_c + \gamma & \text{(6d)} \end{cases}$$

The coordinate of *gird point* $P_k{}^{bl}$ can be calculated through equation 6, where $V_k{}^l$ is the left vertical boundary of region k, and $H_k{}^b$ is the bottom horizontal boundary of region k. Point (X_c, Y_c) marks the *grid point* of the grid cell that point $(V_k{}^l, H_k{}^b)$ belongs to (calculated by equation 4).

Fig. 2(a) depicts the strategy of deciding $P_k{}^{bl}$, the solid square denotes the *grid point* whose coordinate is (X_c, Y_c). When point $(V_k{}^l, H_k{}^b)$ falls in zone A, equation 6a applies. Equation 6b, 6c and 6d apply when point $(V_k{}^l, H_k{}^b)$ falls into zone B, C and D, respectively.

The coordinate of $P_k{}^{tr}$ is calculated similarly, and Fig. 2(b) shows the policy. The coordinates of $P_k{}^{tl}$ and $P_k{}^{br}$ could be deduced once the coordinates of $P_k{}^{bl}$ and $P_k{}^{tr}$ are calculated.

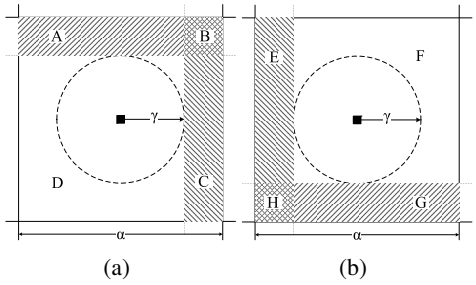


Figure 2: An illustration of calculating corner grid points

The Inter-Regional Route Construction (*IRRC*) algorithm is inspired from the Nearest Neighbor (*NN*) [15] algorithm, and is detailed in Algorithm 1.

Algorithm 1 IRRC algorithm

1: Put all *corner grid points* of each region into a vertex set V_{net}
2: Initialize an empty ordered set V_{tsp} for recording the node series of a *TSP* tour
3: Stand on an arbitrary region i. For each *corner grid point* of region i, find out the shortest edge connecting this *corner grid point* and a vertex of a different region in V_{net}. Put the *corner grid point* with the shortest edge (say u) and the corresponding endpoint (say v) into V_{tsp}. Remove from V_{net} all vertices belonging to the regions where u or v is located
4: Set current vertex be v, find out the shortest edge connecting v and a vertex w of a different region in V_{net}. Add w into V_{tsp}, then remove all vertices belonging to the same region as w's from V_{net}
5: If V_{net} is empty then terminate, otherwise set vertex w be v, and go to step 4

The vertices in the ordered set V_{tsp} generated by Algorithm 1 together form an inter-regional route.

3.3.2 Intra-Regional Design

After the *ME* goes into a region, it traverses the *grid node* of each grid cell to collect data, and then returns to the entering point.

As mentioned in section 3.2, every *grid node* is within γ distance from the *grid point* of the grid cell. So the intra-regional route is designed to be a hamiltonian cycle that visits each *region point* once, and it's also a *TSP* tour.

For example, if a region is covered by a grid with three rows and four columns (see Fig. 3), the *grid points* and the dash lines form a 3×4 grid graph. As illustrated in Fig.3, we can construct a hamiltonian cycle in the grid graph.

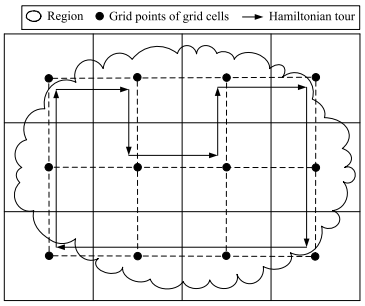


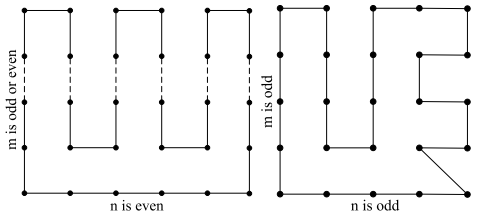
Figure 3: A hamiltonian cycle in a 3×4 grid graph

But not all grid graphs have a hamiltonian cycle. As proved in

literature [16], an $m \times n$ grid graph has a hamiltonian tour if and only if either m or n is even. Based on this theorem, we classify the intra-regional route design issue into two categories: (a) $m \times n$ is even; (b) $m \times n$ is odd.

If $m \times n$ is even, a hamiltonian cycle always exists in the grid graph. We assume n is even, and m is odd or even. The intra-regional route can be constructed as shown in Fig. 4(a).

For there is no a hamiltonian cycle in the grid graph when $m \times n$ is odd, we adjust the intra-regional route to take one diagonal of a grid cell in the grid graph(see Fig. 4(b)).



(a) $m \times n$ is even (b) $m \times n$ is odd

Figure 4: Hamiltonnian cycle construction in grid graphs

3.4 Data Gathering Scheme

After calculating the inter-regional route and the intra-regional routes, *ME* concatenates them to a loop tour, and travels along the loop tour periodically.

When *ME* arrives at each *grid point*, it stops and communicates with the *grid node*. The *grid node* is the root of a local spanning tree. It takes the responsibilities of disseminating queries over the tree, gathering data from the tree, and delivering gathered data to *ME*.

3.4.1 Multi-Point-Relay Tree Construction

The Multi-Point-Relay Tree Construction algorithm (*MPRTC*) constructs a local spanning tree in each grid cell. It is encouraged by the multi-point relays (*MPR*) [17] algorithm, and is stated in Algorithm 2.

Algorithm 2 MPRTC algorithm

1: For each grid node (say x), start with an empty set *MPR(x)*
2: Put the one-hop neighbors of x in current grid cell into set *N(x)*, and put the two-hop neighbors of x in current grid cell into set $N_2(x)$
3: First select those one-hop neighbor nodes in *N(x)* as *MPRs* which are the only neighbor of some node in $N_2(x)$, add these one-hop neighbor nodes to *MPR(x)*, and keep a mapping of each *MPR* and the dominated subset of $N_2(x)$ in *MPR(x)*
4: While there still exist some node in $N_2(x)$ which is not covered by *MPR(x)*:
(a) For each node in *N(x)* which is not in *MPR(x)*, compute the number of nodes that it covers among the uncovered nodes in the set $N_2(x)$
(b) Add that node of *N(x)* into *MPR(x)* for which this number is maximum, and keep a mapping of the *MPR* node and the dominated subset of $N_2(x)$ in *MPR(x)*
5: Construct an *MPR tree* rooted at x based on *MPR(x)*, where each two-hop neighbor of x selects its *MPR* node as its parent
6: After constructing *MPR* trees in all grid cells, every node not covered by any *MPR tree* requests to join an *MPR tree* as a three-hop neighbor of the *grid node* either in its grid cell or in a neighboring grid cell

Fig. 5 shows the *MPRTC* scheme in a grid cell, the left part of the figure is the initial network connection, and the right part illustrates the structure after the *MPR* selection. Node *A* is the *grid node*. Node *B*, *C*, and *D* are all *MPR* nodes selected by the *grid node*, and they together dominate all two-hop neighbors of the *grid node*.

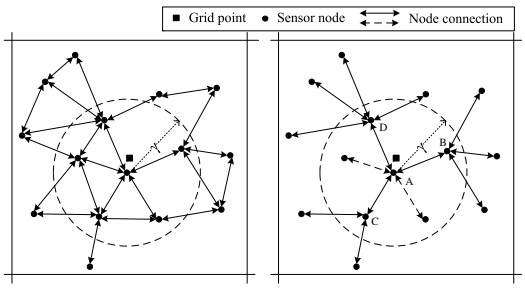


Figure 5: *MPRTC* scheme in a grid cell

For gaining good performance of *MPRTC*, we confine the diagonal length of each grid cell to no more than four times γ, so that most sensor nodes in a grid cell are within two hops away from the *grid node*. Given this constraint, the length of each side of the square grid cell is limited to $2\sqrt{2}\gamma$.

Apparently, the *MPRTC* algorithm divides the network into grid clusters. In each cluster, the *grid node* is a cluster head, it can deliver query message over its *MPR tree*, and gather sensed data from the sensor nodes.

3.4.2 In-Grid Data Fusion

The In-Grid Data Fusion (*IGDF*) technique is efficient for shortening data transferring delay.

Data generated by sensor nodes in a grid cell are geographically correlated, and could be fused at intersecting points of the *MPR tree* when needed.

For simplicity, we ignore the computation cost and the possibility of high rate fusion in *IGDF*. The fusion policy is in a simple step-by-step manner, which means the fusion point first aggregates its own data with one input and next fuses the aggregation result with another input until all the inputs are aggregated.

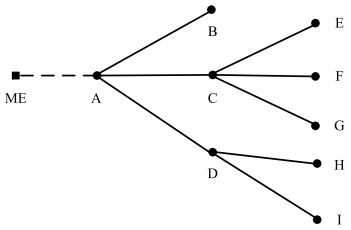


Figure 6: A typical *MPR tree*

For example, in the *MPR tree* shown in Fig. 6, *grid node A* first fuses its data with the data from node *B*, and waits for the data from *MPR* nodes *C* and *D*. Node *C* fuses data from the downstream nodes (node *E*, *F*, and *G*) one by one, and then sends the final result to node *A*. So does node *D*. After fusing all data from its downstream nodes (node *B*, *C*, and *D*), *grid node A* sends the final result to the *ME* staying at the *grid point*.

$$W_u' = (W_u + W_v)(1 - \delta) \tag{7}$$

The resulting amount of data after every fusion is exhibited in equation 7, where W_u and W_v are the amount of data in node u and

node v before current fusion, respectively. W_u' is the amount of data in node u after fusing the data from node v, and δ is the fusion factor.

3.4.3 Query Forwarding and Data Gathering

When *ME* arrives at each *grid point*, it stops and sends a QUERY message containing a *queryTime* parameter. If a *grid node* receives a QUERY message, it floods the QUERY message along the *MPR tree* rooted at itself. When receiving the QUERY message, from the end points of the *MPR tree*, all nodes send the data sensed before the time point of *queryTime* to its upstream sensor node, until all data is converged to the *grid node*. Then the *grid node* sends the data to *ME*, and *ME* moves to next *grid point* after collecting the data. Sensor nodes clear the data from its buffer after sending them out.

Data fusion could be executed at the fusion points using the *IGDF* algorithm.

4. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the *GBMES* algorithm. For comparison, we also observe the performance while implementing the (*PBS*) algorithm [8] to schedule the movement of *ME* in each region.

4.1 Simulation Model

In our simulations, we randomly deploy 500 sensor nodes into five predefined regions located in a $2000 \times 2000 m^2$ playground (see TABLE 1).

Table 1: Deployment of the network

Region	Scope (X_1, X_2, Y_1, Y_2)	Number of nodes
I	(90, 650, 170, 610)	100
II	(560, 1070, 780, 1150)	80
III	(170, 710, 1360, 1760)	90
IV	(1330, 1790, 1380, 1870)	80
V	(1340, 1910, 220, 910)	150

The grid cell size is set to 2.4γ, where γ is the transmission range of sensor nodes and is set to 100 m.

Each sensor node is equipped with a same size buffer (2 Mb), and the data transmission rate of sensor nodes and *ME* is 500 kb/s.

The data sampling rate of each sensor differs due to different locations or various stimuli. Without loosing generality, we randomly set the sampling rate of each sensor node to a value ranging from 0.128 kb/s to 0.512 kb/s.

The parameters used in the simulations are initially set to the default values above unless specified otherwise.

4.2 Impact of the *ME* speed

Fig. 7 illustrates the impact of the *ME* speed on buffer overflow of the *GBMES* scheme and the *PBS* algorithm, as well as the impact of fusion factors on buffer overflow in the *GBMES* scheme.

Given a fixed-length route, the *ME* will travel less time due to the increase of the speed, and therefore the buffer overflow occurrence rate is reduced. We can observe in Fig. 7, when the speed of *ME* ranges from 0 to 16 m/s, the buffer overflow rate gradually descends from 1 (the buffer of every sensor overflows) to 0 (no overflow occurs). In *PBS*, buffer overflow keeps existing until the speed of *ME* reaches about 12 m/s, while in *GBMES*, the critical speed is about 9 m/s.

When data fusion is exploited, *GBMES* performs much better than *PBS* on reducing buffer overflow rate at the same *ME* speed.

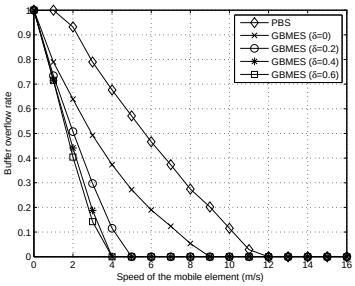


Figure 7: Impact of the *ME* speed on buffer overflow (δ is the fusion factor)

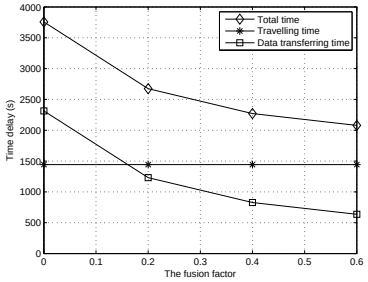


Figure 8: Impact of the fusion factor on the time delay of *GBMES* (at the *ME* speed of 10 m/s)

Fig. 7 shows three fusion instances with fusion factor 0.2, 0.4 and 0.6, respectively. When fusion factor is 0.4, *GBMES* can guarantee zero buffer overflow rate at the *ME* speed of 4 m/s.

The impact of fusion factor on the time delay of the *GBMES* scheme is shown in Fig. 8. In a round, the route length in *GBMES* is fixed, so the traveling time of *ME* running at a certain speed is also fixed. When the fusion factor increases, the amount of data transferred in the network decreases accordingly. Hence the data transferring time is shortened.

4.3 Impact of the grid cell size

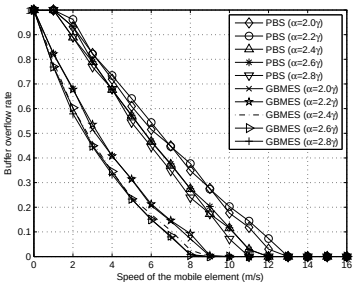


Figure 9: Impact of the grid cell size on buffer overflow

Obviously, increasing the size of grid cell will result in a shorter traveling route. But we can not increase the grid cell size arbitrarily due to the constraints of the *MPRTC* algorithm (see section 3.4.1). Within the acceptable range of the grid cell size (less than $2\sqrt{2}\gamma$, where γ is the transmission range of sensor nodes), we choose 5 points (2.0γ, 2.2γ, 2.4γ, 2.6γ, and 2.8γ) to simulate the impact of grid cell size on buffer overflow.

The results in Fig. 9 show that the performance is becoming better when increasing the grid cell size in both *GBMES* and *PBS* schemes, although not so obviously. Compared with *PBS*, *GBMES* performs better in each grid cell size setting.

The key difference between *PBS* and *GBMES* lies in route design. As illustrated in Fig. 10 , the average route length in *PBS* is about 3 times the one in *GBMES*.

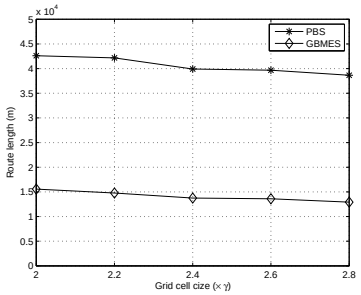


Figure 10: Impact of the grid cell size on traveling route length

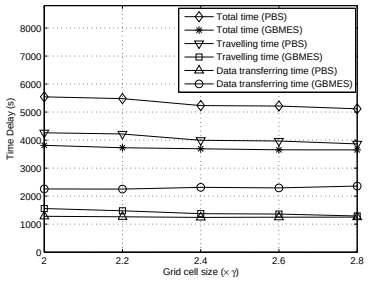


Figure 11: Impact of the grid cell size on time delay(at the *ME* speed of 10 m/s)

So *GBMES* scheme gains less traveling time than *PBS* due to the shorter route length (see Fig. 11).

Unfortunately, this benefit is earned with the augmentation of data transferring delay. In *GBMES*, nodes in the *MPR* tree need one to three hops to transfer data to the *grid node*, and one more hop to the *ME*. Compared with *PBS*, *GBMES* needs more data transferring time when the amount of data transferred is the same, which is confirmed in Fig. 11.

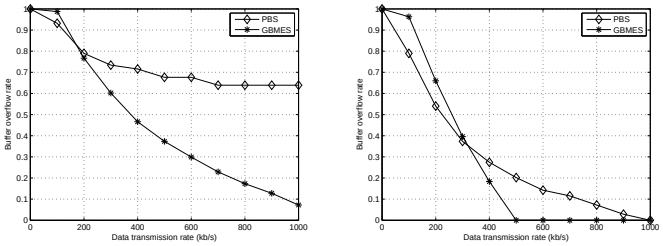
However, total time (the sum of traveling time and data transferring time) spent in *GBMES* is still less than that in *PBS* (see Fig. 11).

4.4 Impact of the data transmission rate

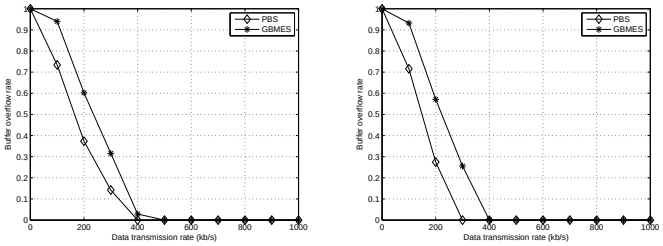
Both in *PBS* and in *GBMES*, increasing data transmission rate will obviously reduce the data transferring delay, hence affects the performance.

When the data transmission rate augments from 0 to 1000 kb/s, the buffer overflow rate descends remarkably. In Fig. 12(a) (*ME Speed = 5*) and Fig. 12(b) (*ME Speed = 10*), when data transmission rate increases, the buffer overflow rate of *GBMES* drops more rapidly than that of *PBS*. Also there is a critical point existing in each of these two instances, where *GBMES* and *PBS* have comparable performance. In Fig. 12(c) (*ME Speed = 15*) and Fig. 12(d) (*ME Speed = 20*), *PBS* has better performance than *GBMES*.

In general, *GBMES* scheme performs better in low-*ME*-speed and high-data-transmission-rate circumstances, while *PBS* performs better in high-*ME*-speed and high-data-transmission-rate situations.



(a) *ME speed=5m/s* (b) *ME speed=10m/s*

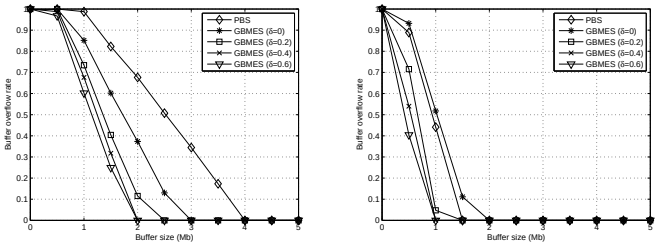


(c) *ME speed=15m/s* (d) *ME speed=20m/s*

Figure 12: Impact of the data transmission rate on buffer overflow

4.5 Impact of the buffer size of sensor nodes

Fig. 13 shows the impact of buffer size on the performance of *GBMES* and *PBS* at two different *ME* speeds. When the buffer size of sensor nodes varies from 0 to 5 Mb, the buffer overflow rate drops accordingly.



(a) *ME speed=5m/s* (b) *ME speed=20m/s*

Figure 13: Impact of the buffer size on buffer overflow

When the *ME* speed is 5 m/s (see Fig. 13(a)), there is no buffer overflow in *GBMES* at the buffer size of 3 Mb, while the buffer overflow rate goes to 0 until the buffer size reaches 4 Mb in *PBS*. *GBMES* schemes with data fusion perform much better.

As the *ME* speeds up, *PBS* gains good performance faster than *GBMES*. When the speed reaches 20 m/s (see Fig. 13(b)), both *GBMES* and *PBS* can keep zero buffer overflow with smaller buffer size, but the performance of *PBS* is slightly better than that of *GBMES*. The *GBMES* scheme still overwhelms *PBS* if data fusion is used.

4.6 Impact of the sampling rate of sensor nodes

To observe the impact of data sampling rate of sensor nodes, we raise a concept of *basic sampling rate* ($r_s{}^b$), and assume the sam-

pling rate in the network varies from $r_s{}^b$ to $4r_s{}^b$. Then we simulate the impact of sampling rate on buffer overflow by adjusting $r_s{}^b$ from 0 to 0.64 kb/s.

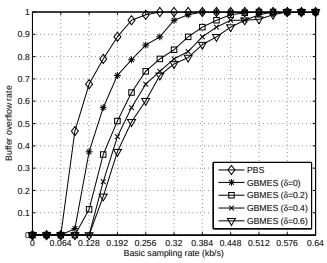


Figure 14: Impact of the sampling rate on buffer overflow

In these simulations, the speed of *ME* is chosen as 5 m/s. Fig. 14 illustrates the simulation results. The buffer overflow rate is increasing rapidly with the augmentation of the *basic sampling rate*. when the *basic sampling rate* reaches 0.64 kb/s, all buffers in *PBS* and *GBMES* (including the ones with data fusion) get overflowed. However the *GBMES* scheme still performs better than *PBS* at each sampling rate.

5. CONCLUSION

In this work, we propose the Grid-Based Mobile Element Scheduling (*GBMES*) approach which schedules a mobile element to periodically gather data from a multi-regional network. The *GBMES* algorithm reduces the time delay to an acceptable extent, and therefore gains good performance on reducing data loss due to the buffer overflow of sensor nodes.

6. ACKNOWLEDGMENTS

This work is supported by the National High-Tech Research and Development Plan of China under grant No. 2006AA01Z214, the National Natural Science Foundation of China under grants No. 60673159 and No. 70671020, the Program for New Century Excellent Talents in University, the Key Project of Chinese Ministry of Education under grant No. 108040, the Specialized Research Fund for the Doctoral Program of Higher Education under grants No. 20060145012 and No. 20070145017, and the Natural Science Foundation of Liaoning Province under grant No. 20062022.

7. REFERENCES

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03)*, Karlsruhe, Germany, Aug. 2003, pp. 27–34.

[2] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep. CS-200006, Apr. 2000.

[3] Q. Li and D. Rus, "Sending messages to mobile users in disconnected ad-hoc wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom'00)*, Boston, Massachusetts, USA, Aug. 2000, pp. 44–55.

[4] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling a three-tier architecture for sparse sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, Anchorage, AK, USA, May 2003, pp. 30–41.

[5] W. Zhao and M. Ammar, "Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks," in *Proceedings of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)*, Puerto Rico, May 2003, pp. 308–314.

[6] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'04)*, Tokyo, Japan, May 2004, pp. 187–198.

[7] W. Zhao, M. Ammar, and E. Zegura., "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom'05)*, Miami, USA, Mar. 2005, pp. 1407–1418.

[8] Y. Gu, D. Bozdag, R. Brewer, and E. Ekici, "Data harvesting with mobile elements in wireless sensor networks," *Computer Networks*, vol. 50, pp. 3449–3465, 2006.

[9] C.Okino and G.Cybenko, "Modeling and analysis of active messages in volatile networks," in *Proceedings of the 37th Allerton Conference on Communication, Control, Computing*, Monticello, IL, USA, Sep. 1999.

[10] A. Somasundara, A. Ramamoorthy, and M. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *Proceedings of the 25th IEEE International Real-Time Systems Symposium (RTSS'04)*, Lisbon, Portugal, Dec. 2004, pp. 296–305.

[11] K. Harras and K. Almeroth, "Inter-regional messenger scheduling in delay tolerant mobile networks," in *Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, Buffalo, NY, USA, Jun. 2006.

[12] M. Tariq, M. Ammar, and E. Zegura, "Message ferry route design for sparse ad hoc networks with mobile nodes," in *Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'06)*, Florence, Italy, May 2006, pp. 37–48.

[13] J. Bentley, "Fast algorithms for geometric traveling salesman problem," *ORSA Journal on Computing*, vol. 4, pp. 387–411, 1992.

[14] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. The MIT Press, 2002.

[15] G. Gutin and A. Punnen, *The Traveling Salesman Problem and Its Variations*. Boston, USA: Kluwer Academic Publishers, 2002.

[16] G. Thompson, "Hamiltonian tours and paths in rectangular lattice graphs," *Mathematics Magazine*, vol. 50, pp. 147–150, May 1977.

[17] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *The 35th Annual International Conference on System Sciences(HICSS'02)*, Hawaii, USA, Jan 2002, pp. 3866–3875.