sensors

MDPI

Article

A Matrix-Based Proactive Data Relay Algorithm for Large Distributed Sensor Networks

Yang Xu *, Xuemei Hu, Haixiao Hu and Ming Liu

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; huxuemei1990@126.com (X.H.); 201511060118@std.uestc.edu.cn (H.H.); mingliu.uestc@gmail.com (M.L.)

* Correspondence: xuyang@uestc.edu.cn; Tel.: +86-189-8094-7876

Academic Editor: Xue-Bo Jin

Received: 21 June 2016; Accepted: 2 August 2016; Published: 16 August 2016

Abstract: In large-scale distributed sensor networks, sensed data is required to be relayed around the network so that one or few sensors can gather adequate relative data to produce high quality information for decision-making. In regards to very high energy-constraint sensor nodes, data transmission should be extremely economical. However, traditional data delivery protocols are potentially inefficient relaying unpredictable sensor readings for data fusion in large distributed networks for either overwhelming query transmissions or unnecessary data coverage. By building sensors' local model from their previously transmitted data in three matrixes, we have developed a novel energy-saving data relay algorithm, which allows sensors to proactively make broadcast decisions by using a neat matrix computation to provide balance between transmission and energy-saving. In addition, we designed a heuristic maintenance algorithm to efficiently update these three matrices. This can easily be deployed to large-scale mobile networks in which decisions of sensors are based on their local matrix models no matter how large the network is, and the local models of these sensors are updated constantly. Compared with some traditional approaches based on our simulations, the efficiency of this approach is manifested in uncertain environment. The results show that our approach is scalable and can effectively balance aggregating data with minimizing energy consumption.

Keywords: proactive data relay; information fusion; large distributed sensor networks; matrix-based computing

1. Introduction

Large distributed sensor networks have been widely used in both military and civilian applications [1], such as target tracking [2], disaster response [3] and field surveillance [4]. In these applications, although each sensor is able to get some crude information, the data is usually imprecise or noisy with very low fidelity. As a consequence, the data in this form cannot be directly used for automatic planning or supporting human decisions, and has to be fused with other relevant data [5].

To fuse distributed data in a network, the key is to relay multiple-source data and aggregate enough amount of data to a single node in order to achieve high confidence information fusion [6]. However, with the progress of the development of state-of-the-art mobile sensor applications, networks emerge with new characteristics that pose challenges to existing information fusion approaches. The astronomical growth of networks involving thousands of sensors is a typical challenge in this respect. In these large networks, no single sensor can respond as the center and gain complete statistic of states of the entire network. In addition, because these networks are dynamically changing due to the mobility of sensors, or movements as a result of the surrounding air or ocean currents [7], node

failures are not rare. Additionally, most sensors in these large networks have lightweight processing units, which make high constraint of energy consumption ideal for their operations.

Following the importance of information fusion, many data relay protocols have been developed to diffuse data in sensor networks. Flooding [8] adopts a straightforward protocol for sensors to rebroadcast any new data received, but suffers from the intrinsic excessive energy consumption and network congestion due to the large number of duplicated messages. Traditional hierarchical [8] or centralized routing [9–12] approaches, which are designed for fixed structure or where only sink nodes respond to information fusion are not also ideal for data fusion in mobile sensor networks. The predefined backbone nodes in these approaches will exhaust their power very quickly because of the heavy transmission burden and enormous cost on routing maintenance. To avoid these problems, many other distributed approaches have been developed for cases whereby all nodes can be treated as sink nodes. Those approaches include Information via Negotiation(SPIN) [13], Scalable Broadcast Algorithm (SBA) [14], The Lightweight and Efficient Network Wide Broadcast (LENWB) [15], Dominant Pruning [16] and Dynamic Probabilistic Flooding Algorithm [17]. Even though these approaches can guarantee a high quality information fusion, not all sensors would need all the data because any piece of information can only be fused by the sensor that aggregates enough relative data first. Therefore, the unnecessary data coverage may end with huge amount of energy consumption and redundant communication cost. SPIN can address this limitation to some extent by negotiating before broadcast, and allows sensors to transmit only data that others would need. Unfortunately, it is a reactive protocol, and a large number of negotiated messages makes it time-consuming and cost-ineffective for distributed data fusion. Some algorithms [18] can proactively relay data by forecasting the needs before a given set of data is broadcast. For example, a sensor would rather like to rebroadcast the data that is relative to the data of its neighbors only. However, when sensors cannot gain a complete view over an entire network, an intelligent proactive algorithm could be very difficult to implement [19].

In this paper, we present a novel matrix-based energy-saving and proactive algorithm to relay data in large distributed sensor networks. To deliver the core competence of this approach, we employ no querying process, and each node proactively forwards data to its neighbors by estimating the needs of neighbors based on its local knowledge about itself and the neighbors. From this viewpoint, we encode local knowledge of sensors into three neat matrices: local connection matrix (C); local data distribution matrix (D); and utility matrix (U) to encode utility of adding data into a data set. With these matrices, a compact and light-weight algorithm is proposed, where each node makes a series of simple all-in-one matrix computation to evaluate the benefits and cost of actions. Comparing the benefits against cost allows serious decisions to be made whether a piece of sensor data should be broadcast to make a good tradeoff between data relay to get high quality information and prolonging the lifetime of the network. In addition, energy consumption in data relay is taken into consideration towards cautious decision-making processes. As sensors do not have global knowledge of the network, a heuristic algorithm is proposed to maintain the size and value of these matrices from incoming messages to adapt to sensor mobility. By introducing matrix computations, and as our matrices only store local knowledge and updated in time, our approach is lightweight and can easily be deployed in large mobile sensor networks. Further, to manifest feasibility of this approach, we discuss the simulations carried out, and the results show that our approach can perform well in dynamic environments to effectively balance aggregating data with minimizing energy consumption.

2. Related Work

Many data relay protocols have been designed for distribute data fusion in wireless sensor network [20–22]. They can be categorized as two groups according to where the fusion occurs.

One strand of protocols are based on fix or predefined sink nodes. They are designed for multiple sensors to forward data to specific sink nodes for data fusion, such as Directed Diffusion [9], in which a sink floods its interests to build reverse paths from all potential sources to the sink. Rumor

Routing [10], Constrained anisotropic diffusion routing (CADR) [11] and GRAdient Broadcast [12] are some variations of Directed Diffusion. In Rumor routing, sink floods queries while sensors flood events which make Rumor routing performs better than Directed Diffusion when number of events is small. CADR introduces an information utility measure to select which sensors query and dynamically guide data routing. GRAB builds and maintains a cost field to provide sensors the direction to forward sensing data. It is a robust data delivery algorithm addressed nodes failures and link failures. However, since these protocols are based on single-gateway architecture that makes them not fit for large scale mobile sensor network [21]. First of all, sensors near sink nodes have heavy burden to relay data and their energy will be exhausted in short time. Second, sensors are typically not capable of long-haul communication and the latency in communication can not be ignored. Third, energy on finding ways to sink nodes is huge which makes them not suitable for mobile networks where network topology dynamically changes.

To allow the system to be able to cover a large area of interest without degrading the service, networking clustering has been pursued in some routing approaches. LEACH [21] and LEACH series [23] are hierarchical routing algorithms for sensor networks. Cluster heads change randomly over time in order to balance the energy dissipation of nodes. Each node transmits directly to the cluster-head. They are completely distributed and requires no global knowledge of network. However, they use single-hop routing where each node transmits directly to the cluster-head and the sink. Therefore, it is not applicable to networks deployed in large regions . Further more, the idea of dynamic clustering brings extra overhead, e.g. head changes, advertisements etc., which may diminish the gain in energy consumption.

For the second strand of protocols, there is no predefined sink nodes, sensors are teated equally as probable sink nodes, and data fusion can be done by any sensor unless it aggregates enough relative data. The most straight forward protocol is flooding [8], in which sensors rebroadcast any new data it receives. Apparently, FLOODING consumes too much energy on the transmission of redundant data. To reduce the redundant data of flooding, some reactive and proactive protocols are proposed. Sensor Protocols for Information via Negotiation(SPIN) [13] is a reactive protocol which avoids redundant data transmission by meta-data negotiation with neighbors. sensors only forward data to the neighbors that need it. However, it is infeasible for distributed fusion where the size of meta-data is close to useful data because the frequent query data transmission is not cost-effective and introduces non-negligible time delay.

Proactive protocols do not need queries to avoid redundant data transmission but proactively decide if rebroadcasting data received based on the local topology and static information of redundant data, such as Scalable Broadcast Algorithm (SBA) [14], The Lightweight and Efficient Network Wide Broadcast (LENWB) [15], Dominant Pruning [16] and Dynamic Probabilistic Flooding Algorithm [17]. SBA and Dominant Pruning maintain the local network topology by hello messages. In SBA, sensors only rebroadcast the data that at least one of its neighbors do not know. In Dominant Pruning, rebroadcasting nodes proactively choose one or more of its neighbors as rebroadcasting nodes by Greedy Set Cover algorithm. In Dynamic Probabilistic Flooding Algorithm, sensors obtain neighbor information by basic flooding and divide neighbors into three types: parent (upper level), sibling (same level), and child (lower level) nodes. A node with the more children nodes and less siblings needs the lower retransmission probability. These three protocols try to cover all the nodes with all the data to guarantee at least one node aggregates enough relative data to fuse into information. However, in large sensor network, too much energy is consumed on unnecessary coverage of such a huge number of sensor nodes, for a piece of information is only needed to be fused by one sensor. In addition, in mobile sensor network, too much energy is consumed on the transmission of hello message to maintain local topology.

3. Problem Description

A typical scenario of a large scale mobile sensor domain is illustrated in Figure 1, where distributed sensors are randomly deployed for remote operations in a large unstructured geographical area to detect events. For their limited communication ranges and wide distribution, each of them can only relay data to a few of the others directly. But for the low quality of those sensor readings, data has to be transmitted until a single node get enough relevant data to produce high confident information, which denoted as a double circled sensor in the figure.

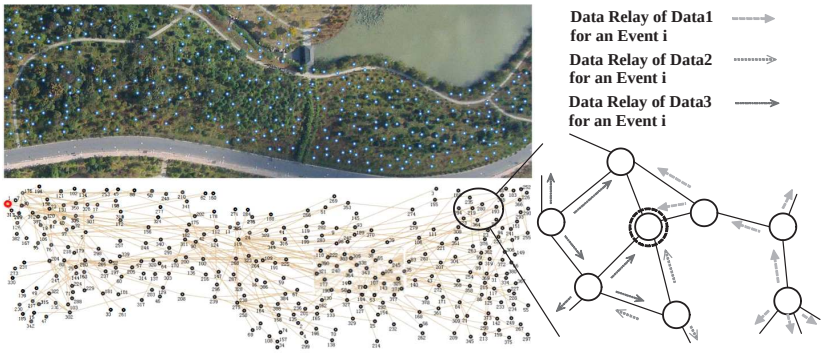


Figure 1. An overlook on the large-scale sensor network deployment and the real topology.

Let $G(V, E)$ be the topology graph of this network. V=$\{v_1, v_2, ..., v_i, ...\}$ denotes the set of mobile sensor nodes such as sonics, microwave, infrared and x-ray sensors. $E = \{e_{ij} | \forall v_i, v_j \in V, P(v_i, v_j) > 0\}$ consists of edges between any two sensors, where $P(v_i, v_j)$ is the connection probability between two sensors. Supposed that each sensor has an identical communication range r, $P(v_i, v_j)$ can be calculated according to the channel propagation model [24,25]: $P(v_i, v_j) = (1 - f(d_{ij}, 1/r))$ where d_{ij} is the distance between v_i and v_j. Based on E, the neighbor set $N(v_i)$ of each sensor $v_i \in V$ is $\{v_j | \forall v_j \in V, e_{ij} \in E\}$.

Let us consider a case of sensors deployed to track multiple stationary or moving targets $Target = \{T_1, T_2, ..., T_m\}$. It is possible that a given target T_k can be detected by multiple sensors at the same time. For example, when a hostile vehicle is moving through a given intersection, a shock sensor detects the vibration when it passes and an infrared sensor nearby receives the infrared signal from the vehicle. After detecting a target, sensor v_i will analyze the raw data(vibrates, infrared signal and so on), and generates a data d_j about this target. Data d_j can be denoted as a tuple $< sourceID, identity, location, timestamp, path >$:

- *sourceID* is the ID of the sensor that senses this data.
- *identity* is the confidence hypotheses about the target identities T_k, which can be expressed as $cd_j = \{c_{d_j}(T_1), c_{d_j}(T_2), ..., c_{d_j}(CLUTTER), \}$. An example is shown in Table 1.
- *location* is the geographical location of the target when detected.
- *timestamp* is the system time when the target is detected.
- *path* records sensors that pass this data.

Table 1. An example of the body of a piece of data.

Target	Confidence	Target Type	Confidence
USSR T80	0.4	US M977	0.001
WSSR T72M	0.3	US M35	0.001
US M1	0.1	US AVENGER	0.001
US M1A1	0.05	US HMMWV	0.001
USSR 2S6	0.02	USSR SA9	0.001
USSR ZSU23	0.03	CLUTTER	0.095

The *location* and *timestamp* of a target are used to determine if two pieces of data are referring to the same target. If that is the case, they are called relevant. Since the data from a sensor is always noisy, uncertain and cannot be used directly for automatic planning or supporting human decisions, data should be relayed around the network to meet more relevant data to produce a higher quality information [26].

The basic distributed data relay process for a given sensor v_i can be briefly described by Algorithm 1. Each sensor keeps a local data set L_i to store the up-to-date data it receives and senses. For any data d_j sensed or received by v_i, d_j is first put into the local data set L_i of v_i. Next, v_i tries to fuse this data with relevant data in L_i based on pre-selected fusion rules such as Bayesian inference method [27] and Dempster-Shafer theory [28]. The fusion rules are out of the scope of this paper, and so no explanation to that effect is provided. If the quality of these fused data is beyond the predefined threshold, v_i will stop the propagation of this data and fuse them into a piece of valuable information I_h with a credible confidence about a target T_k. Otherwise, v_i will add the data into the pending queue $pendingQue_i$, and make communication decisions for all data in this queue.

Algorithm 1 Distributed data relay process.

```
1:  while true do
2:      L_i ← data received or sensed by v_i
3:      for all data d_j received or sensed by v_i do
4:          Try to fuse it with relevant data in L_i;
5:          if the quality of the fused data meets the threshold then
6:              Fuse them into a piece of information;
7:              Inform other nodes data d_j is outdated;
8:          else
9:              PendingQue_i ← d_j;
10:     Make communication decisions for each data in PendingQue_i;
```

The objective of data relay is to aggregate relevant data to some single node to fuse more high quality information while minimizing the energy consumption of the network. However, sensors cannot take optimal actions since they do not have global view of the network. They make data communication decisions based on local knowledge to maximize incremental quality of relevant data set of neighbors in local data set, and minimize the energy cost. For data d_j, there are two communication choices, broadcast $act_{v_i}^{d_j} = 1$, or not $act_{v_i}^{d_j} = 0$. If broadcasting, it is not absolutely sure that all the neighbors will receive this data because of the uncertainty of network connection. We can explain the objective function of data relay as following:

$$\underset{act_{v_i}^{d_j}}{\operatorname{argmax}}\, act_{v_i}^{d_j} \times \Big(\sum_{v_k \in N(v_i)} P(v_i, v_k) \times \Delta Q(d_j, L_k) - \beta \times Energy(v_i, d_j) \Big) \tag{1}$$

where $\Delta Q(d_j, L_k)$ is the incremental quality of knowledge base L_k after receiving data d_j. $Energy(v_i, d_j)$ is the energy consumption on transmitting d_j, and β is a coefficient to balance the energy cost and information cost in decisions.

3.1. Information Quality

We use $L_i^h \subseteq L_i$ to express the subset of data that is relevant, and indicates information by I_h. The quality $Q(L_i^h)$ of fused data in subset L_i^h can be calculated based on the fusion rule of sensors. Let us take Dempster-Shafer rule [28] as an example,

$$Q(L_i^h) = \max_{T_k \in T} c_{L_i^h}(T_k) + c_{L_i^h}(CLUTTER) \tag{2}$$

where $c_{L_i^h}(T_k)$ and $c_{L_i^h}(CLUTTER)$ are elements of $c_{L_i^h} = \{c_{L_i^h}(T_1), .., c_{L_i^h}(T_M), c_{L_i^h}(CLUTTER)\}$, representing the basic probability assignment for the fused data indicating information I_h. This can be calculated as $c_{L_i^h} = \oplus_{d_k \in L_i^h} c_{d_k}$, where " $\oplus$ " is the operator in D-S rule of combination. If the number of these data reaches the minimum number threshold ω_n and the quality of fused data is more than the minimum value threshold ω_Q: $Q(L_i^h) > \omega_Q$ &$|L_i^h| > \omega_n$, these data will be fused into a piece of information I_h. The quality of the whole local data set is the sum of the quality of data sets:

$$Q(L_i) = \sum_{L_i^h \subseteq L_i} Q(L_i^h) \tag{3}$$

As data is only fused with relevant data, adding one piece of data to a local dataset can only affect the quality of its related data set. The incremental quality of a database after receiving d_j can be calculated as

$$\Delta Q(d_j, L_i) = Q(L_i \cup \{d_j\}) - Q(L_i) = Q(L_i^h \cup \{d_j\}) - Q(L_i^h) \tag{4}$$

Maintaining a precise value of the incremental quality based on Equation (4) for each neighbors is computational high. However, it is unnecessary to keep it so precise and an estimation of this value is enough for the relay decision. What we need is to obtain those data that have higher confidence and have more relevant data in the local data set. To simplify, we use the utility of a data set to approximately estimate the quality of same. The utility can easily be calculated as the sum of utility between data:

$$U(L_i) = 1/2 \sum_{d_j \in L_i} \sum_{d_l \in L_i} U(d_j, d_l) \propto Q(L_i) \tag{5}$$

where $U(d_j, d_l)$ is the utility between data. The value can be computed either according to the fusion rule of sensors, or given by an expert knowledge system. For example, utility can be looked up in a utility table stored in sensors as their domain knowledge. Particularly, $U(d_j, d_j) = 0$, and if d_j and d_l are not relative, $U(d_j, d_l) = 0$. Therefore the incremental quality can be approximately represented by the incremental utility:

$$\begin{aligned} &\Delta Q(d_j, L_i) \propto \\ &\Delta U(d_j, L_i) = U(L_i \cup \{d_j\}) - U(L_i) \quad = \begin{cases} \sum_{d_l \in L_i} U(d_j, d_l) & \text{if } d_j \notin L_k; \\ 0 & \text{if } d_j \in L_i \end{cases} \end{aligned} \tag{6}$$

If v_i already has observation about d_h, this data becomes redundant and makes no contribution to fusion, and as such, the benefit of broadcasting is 0. If v_i does not have this data, it may be helpful to sensor v_i to increase the fusion probability. The benefit in this case can be represented by the sum of utility between this data and other data in local data set.

3.2. Energy Consumption on Communication

In a deployed sensor network, the sensor nodes are usually battery powered [29], and they have to operate on an extremely frugal energy budget. Since communication is the major source of energy consumption in sensor networks, to prolong the lifetime of the sensor network requires a careful consideration of the energy cost in each transmission.

For each sensor v_i, the energy cost on communication is mainly composed of two parts: the energy of broadcasting data d_j: $Eb(v_i, d_j)$, and the energy of receiving data d_j: $Er(v_i, d_j)$. They can be computed as follows [24]:

$$\begin{aligned} Eb(v_i, d_j) &= (E_{elec}(v_i) + E_{amp}(v_i)) \times d_j.length \\ Er(v_i, d_j) &= E_{elec}(v_i) \times d_j.length \end{aligned} \tag{7}$$

where $E_{elec}(v_i)$ is the energy consumed by v_i's transmit electronics or receive electronics for digital coding, modulation and filtering of the signal, $E_{amp}(v_i)$ is the energy consumed by its TX amplifier, and $d_j.length$ (bits) is the length of data pieces. Suppose that the sensors are homogenous, and the size of data is identical. The energy cost of broadcasting and receiving a piece of data can be substituted by two constants Eb and Er respectively. The energy consumption on transmitting a piece of data is the sum of energy on broadcasting and all neighbors' receiving:

$$Energy(v_i, d_j) = Eb + Er \times \sum_{v_j \in N(v_i)} Pr(v_i, v_j) \tag{8}$$

4. Matrix-Based Data Relay Algorithm

In large-scale distributed sensor networks, because of the huge system size and energy constraints of sensors, sensors are unlikely to have a global observation to support optimal communication decisions. Sensors only make rational communication decisions soley based on their local dataset from their previously transmitted messages around the network. To make rational communication decisions, sensors need local topology of the network to indicate who are potential receivers, the local data distribution to indicate what data neighbors have, and the utility between data pieces to figure out the benefit of broadcasting one piece of data.

In this section, a Proactive Energy-Saving Data Relay algorithm *CDU* will be proposed to help sensors compute the benefit and cost for all the data in *pendingQue* by neat matrix computations. The framework is shown in Figure 2. First, sensors need local states to support their decisions. In our model, three parts are necessary to a sensor: connection matrix **C** denotes its local network topology, data distribution matrix **D** about the local data distribution of the sensor and its neighbors and the data Utility matrix **U**. For each time step t, each sensor i is required to update CDU matrixes by the model maintenance function introduced in the next section. Next, a neat computation with the CDU matrixes can produce the expected benefit of transmitting pending data B_i^t. In addition, connection matrix C_i^t also be used to predict the cost of transmitting data E_i^t. By balancing between B_i^t and E_i^t, if the sensor find that the expected utility F_i^t is positive, it will broadcast the data as the network will be benefit of more likely fusing valuable information.

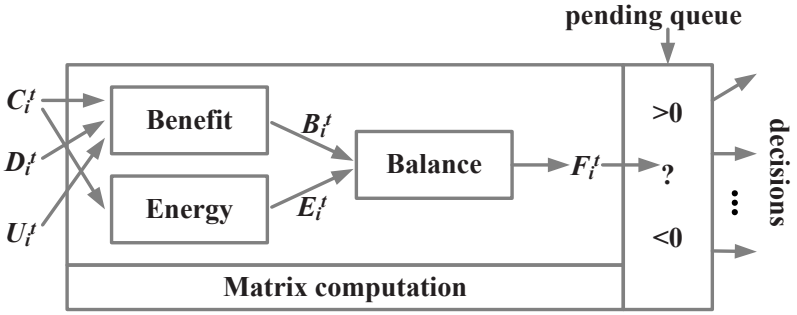


Figure 2. The frame of matrix-based Data Relay algorithm.

4.1. Basic Matrix Model

Before introducing these three matrices, we first need one data structure N_i to store the sensors known by v_i, and it can be updated by the path of messages it receives. Since sensors have different local information, we take sensor v_i as an example to describe these three matrices.

$\mathbf{C} : 1 \times |N_i| \rightarrow [0, 1]$ is the connection matrix to illustrate the connections between v_i and other sensors recorded in N_i. Each element $\mathbf{C}_{v_i,v_k}$ represents the connection probability between v_i and v_k estimated by v_i. Especially, $\forall v_i \in V,\ C_{v_i,v_i} = 0$.

$\mathbf{D} : |N_i| \times |L_i| \rightarrow [0, 1]$ is the local data distribution matrix, and each element $\mathbf{D}_{v_k,d_j}$ shows the probability of d_j in v_k's local data set by v_i's estimation.

$\mathbf{U} : |L_i| \times |L_i| \to [0, c]$ is the utility matrix of data known by v_i, and each element $\mathbf{U}_{d_l,d_k} = U(d_l, d_k)$ shows the incremental utility when adding d_l meets $\{d_k\}$.

4.2. Benefit of Broadcasting

The benefit of broadcasting one piece of data d_j is the sum of increased expected utility of all receivers. By multiplying Equation (1) by $(1 - D_{v_k,d_j})$, the increased expected utility of v_k receiving d_j can be calculated in a unified manner by matrix computation.

$$\begin{aligned} \Delta EU(d_j, L_k) =& (1 - D_{v_k,d_j}) \cdot \sum_{d_l \in L_k} U_{d_h,d_l} \\ =& (1 - D_{v_k,d_j}) \cdot \sum_{d_l \in L_i} D_{v_k,d_l} \cdot U_{d_h,d_l} \\ =& (1 - D_{v_k,d_h}) \cdot (\overrightarrow{D_{v_k,*}} \cdot \overrightarrow{U_{*,d_h}}) \end{aligned} \tag{9}$$

The benefit of broadcasting d_h is the sum of increased information utility of sensors that can receive this data. However, for v_i, it does not know which sensor exactly receives this piece of data broadcast. Only the connection probability stored in matrix C indicates the probability of receiving this data. Therefore, the benefit of v_i broadcasting d_h can be computed as:

$$Benefit(v_i, d_h) = \sum_{v_k \in N_i} C_{v_i,v_k} \cdot \Delta EU(d_h, L_k) = \quad C \cdot ((\overrightarrow{\Lambda} - \overrightarrow{D_{*,d_k}}) \circ (D \cdot \overrightarrow{U_{*,d_k}})) \tag{10}$$

where $D_{v_k,*}$ is a row vector of matrix D, $\overrightarrow{\Lambda} = [1]_{|N_i| \times 1}$ is a column vector of ones, the operator " $\circ$ " is the Hadamard product that takes two matrices with identical dimensions but only produces their corresponding elements. In matrix representation, one sensor can compute the benefit of broadcasting any of its data by a single matrix computation:

$$B = C \cdot ((\Gamma - D) \circ (D \cdot U)) \tag{11}$$

where $B = [Benefit(v_i, d_h)]_{1 \times |L_i|}$, and $\Gamma = [1]_{|N_i| \times |L_i|}$.

4.3. Energy Cost of Transmission

The energy cost of transmission is composed of two parts: cost of broadcasting, and cost of receiving. The broadcast energy cost of a piece of data is Eb. Before transmission, v_i does not know exactly which sensor can receive this data, and can only estimate the whole receiving energy cost according to the connection probability in matrix C_i. The cost of v_i transmitting d_h can be estimated as follows,

$$Energy(v_i, d_h) = Eb + \sum_{v_k \in N_i} C_{v_i,v_k} \cdot Er \quad = Eb + Er \cdot (C \cdot \overrightarrow{\Lambda}) \tag{12}$$

Let matrix $E = [Energy(v_i, d_h)]_{1 \cdot |L_i|}$, which represents the energy cost of v_i for transmitting any data in $pendingQue_i$. Thus, this matrix can be calculated as follows:

$$E = (Eb + Er \cdot (C \cdot \overrightarrow{\Lambda})) \cdot \overrightarrow{\Lambda}^T \tag{13}$$

4.4. The Balance

To make communication decisions for each sensor data in pending queue, sensors balance the benefit of broadcasting with the energy cost as follows.

$$F = B - \beta E = \quad C \times ((\Gamma - D) \bullet (D \times U)) - \beta \cdot (Eb + Er \cdot (C \times \overrightarrow{\Lambda})) \times \overrightarrow{\Lambda}^T \tag{14}$$

where F is a $1 \times |L_i|$ matrix and each element in F indicates the difference between the benefit and the energy cost of transmitting each data. Essentially, this can also be calculated in a single matrix computation.

In the decision process, for each data in $pendingQue_i$, if $F_{v_i,d_h} > 0$, which represents the benefit of broadcasting this data is bigger than the energy cost, this data is worth to be broadcast at this moment. Otherwise, it may not be broadcast. By ensuring a balance between increasing the quality of receivers and saving energy,

- the data that has higher confidence (that can make a higher contribution to fuse into an information) has a higher priority to be broadcast, which can avoid the energy consumption on unnecessary data retransmission.
- the data that is more relative to data of neighbors has a high probability to be broadcast. This guarantees that the related data are only transmitted in a small part of the whole network and aggregated toward some node rather than blind coverage.

5. Model Maintenance Algorithm

Sensors can make rational decisions by comparing the benefits and energy consumption of broadcast through matrix computation introduced in the previous section. Because of the mobility of sensors, and the dynamic changing nature of data distribution, the three matrices C,D and U need to be updated in time. The more precise the model is, the better the decisions. However, in large sensor networks, considering the massive energy cost on communication, no single sensor can get the precise and global connection and data distribution model but only partial observations.

In this section, considering the intrinsic energy cost in the operations of these networks, we propose heuristic updating approaches to maintain a local model for each sensor from its incoming messages. With these updating approaches, an integrated decision algorithm is considered to help each sensor make rational decisions with partial observations.

5.1. Initialization

Before deployment in large scale sensor networks, locations of sensors are not pre-determined, and the network topology is unknown to any sensor. Sat this stage, the three matrices of each sensor are initialized to empty matrices. After deployment, sensors start the introduction phase by broadcasting hello messages for once to initialize their connection matrix C.

5.2. The Rules to Maintain The Dimensions

As we described in Section 3, the size of these three matrices is related to N_i and L_i. The column number of C and row number of D correspond to the size of N_i. The column number of D and the row and column number of U also correspond to the size of L_i. When an element is added into any of these two sets, one column or row of 0 will be added to the corresponding matrices. If one element is moved out of these sets, the corresponding column or row of matrices will be deleted. Therefore, the principles to maintain these two sets are paramount, and given below:

- v_i will add an element v_j in set N_i: when receiving a data d_k and $v_j \in d_k.path$ is not in N_i.
- v_j will be removed from N_i: when v_i neither has positive connection probability with it nor has any knowledge of it, C_{v_i,v_j}=0 and $\sum_{d_h \in L_i} S_{v_j,d_h}$=0.
- v_i will add an element d_k into L_i:

  - when receiving data d_k that is not in L_i.
  - when generating data d_k based on its detection.
- v_i will delete the element d_k from L_i: when $d_k \in dataO$, where $dataO$ stores data that has been fused or outdated.

5.3. Updating the Connection Matrix C

In wireless sensor networks, the connection matrix C is initialized by hello messages. However, because of the dynamically changing nature of the network topology, the connections between sensors may change. In this subsection, some heuristic rules are proposed to update matrix C from sensors' incoming messages. When sensor v_i receives a piece of data from sensor v_j, first, it indicates that v_j is well connected with it:

$$C_{v_i,v_j} = \sigma$$

where σ is a high probability that the two sensors are connected. Second, any two adjacent sensors in $d_k.path$ are well connected for their successful transmission in the last time step. The connection between them can be updated by σ^2 such that $\forall v_j \in d_h.path$, $v_k = d_h.path.next(v_j)$, and

$$C_{v_j,v_k} = C_{v_k,v_j} = \sigma^2$$

Also, v_i will assume any connection probability in C fades as sensors move and failure of sensors occurs. This implies, the connection probability decays for a given period of time T. We can describe this process as follows:

$$C \leftarrow C \circ C$$

5.4. Updating the Data Distribution Matrix D

The better sensors have knowledge about the knowledge bases of their neighbors, the better decisions they can make. In this subsection, we focus on the data distribution updating approaches based on the data pieces sensors sense, broadcast and receive.

Algorithm 2 presents the updating process of data distribution matrix D for sensor v_i, where $dataG$, $dataB$, and $dataR$ are data sets, which respectively store data sensed , data broadcast and data received by v_i as well as outdated data. First, for each data generated by v_i based on its own detection, obviously, it will update element D_{v_i,d_h} to 1 (line 1–2). Second, for each data broadcast by v_i, and for any neighbor v_j, the probability of receiving this data is C_{v_i,v_j}. Therefore, according to the standard probability function, v_j's probability of having data d_h should be updated, $D_{v_j,d_h} = 1 - (1 - D_{v_j,d_h})(1 - C_{v_i,v_j})$ (line 3–4). Third, for each data received by v_i, v_i, update D_{v_i,d_h} to 1 (line 6). Also, all nodes on $d_j.path$ have this data (line 8), and the nodes that are neighbors of nodes on the path have a probability of having this data, which can be calculated according to the standard probability function (line 9). Finally, for data, which relative information has been fused and outdated, its corresponding column in D should be deleted to save storage (line 10–11).

Algorithm 2 $updateD(D, C, dataG, dataR, dataB)$.

1: **for all** $d_h \in dataG$ **do**
2: $\quad D_{v_i,d_h} \leftarrow 1$;
3: **for all** $d_h \in dataB$ **do**
4: $\quad \overrightarrow{D_{*,d_h}} \leftarrow \overrightarrow{\Lambda} - (\overrightarrow{\Lambda} - \overrightarrow{D_{*,d_h}}) \circ (\overrightarrow{\Lambda} - \overrightarrow{C_{v_i,*}})$;
5: **for all** $d_h \in dataR$ **do**
6: $\quad D_{v_i,d_h} \leftarrow 1$;
7: $\quad$ **for all** $v_j \in d_h.path$ **do**
8: $\quad\quad D_{v_i,d_h} \leftarrow 1$;
9: $\quad\quad \overrightarrow{D_{*,d_h}} \leftarrow \overrightarrow{\Lambda} - (\overrightarrow{\Lambda} - \overrightarrow{D_{*,d_h}}) \circ (\overrightarrow{\Lambda} - C_{v_j,*})$;

5.5. Updating The Utility Matrix U

When generating or receiving a new piece of data, the utility between this data and the data in L_i can be looked up in a table, which is stored as background knowledge before sensor v_i was deployed according to their *identity*, *location* and *timestamp*. The corresponding elements in matrix U will then be updated. The details are shown as Algorithm 3. When receiving new data d_h, first, v_i will judge if it is relative to the data in L_i(line 2,3) based on their *timestamp* and *location*. If that is the case, the utility between it and any relative data can be looked up in a table according to their *identity* (line 4). Else, the corresponding utility will be set to 0 since they indicate different information (line 6).

Algorithm 3 $updateU(U, dataG, dataR, utilityTable)$.

1: **for all** $d_h \in dataG \cup dataR$ **do**
2: **for all** $d_j \in L_i$; **do**
3: **if** $relative(d_h, d_j)$ **then**
4: $U_{d_h,d_i} \leftarrow utility(d_h, d_i, utilityTable)$;
5: **else**
6: $U_{d_h,d_i} \leftarrow 0$;

The example below shows the update process of the utility matrix. At time t, the data set of v_1 is $L_i = \{d_1.d_2\}$, and the sensor set in the knowledge of v_1 is $N_i = \{v_2, v_3\}$. The details of d_1, d_2 are shown as follows:

$d_1 =< v_4, (T_A, 0.5)(T_B, 0.2)(CLUTTER, 0.3), (1.1, 2.1), 2013/05/08/16:00, path = \{v_4, v_2\} >$,
$d_2 =< v_7, (T_A, 0.4)(T_B, 0.5)(CLUTTER, 0.1), (3.2, 7.1), 2013/05/08/13:00, path = \{v_7, v_3\} >$.

As the *location* and *timestamp* of these two data pieces are different, they are not relevant, and matrix $U = \begin{pmatrix} & d_1 & d_2 \\ d_1 & 0 & 0 \\ d_2 & 0 & 0 \end{pmatrix}$. At time $t+1$, v_1 receives d_3 from v_3, and d_3 is shown in details as follows:

$d_3 =< v_8, (T_A, 0.6)(T_B, 0.3)(CLUTTER, 0.1), (1.3, 2.0), 2013/05/08/16:03, path = \{v_8, v_5, v_3\} >$.

According to its *location* : $(1.3, 2.0)$ and *timestamp* : $2013/05/08/16:03$, which is close to d_1 but not d_2, v_1 can confirm that d_3 is related to d_1 but not d_2. It becomes obvious now that $U_{d_3,d_2} = U_{d_2,d_3} = 0$. Right from here, v_1 will look up the utility between d_3 and d_1 in its reward table according to their *identity*.

- For target A, $c_{d_1}(T_A) = 0.5$, $c_{d_3}(T_A) = 0.6$. After checking *utilitytable*, v_2 can get $value(T_A) = 5.5$. Doing the same for target B, v_2 can get $value(T_B) = 3.5$
- The utility is a function of these two values. One possible way:

$$U_{d_3,d_1} = value(T_A) + \omega \cdot value(T_B) = 5.5 + 0.3 \cdot 3.5 = 6.55$$

Finally, the matrix $U = \begin{pmatrix} & d_1 & d_2 & d_3 \\ d_1 & 0 & 0 & 6.55 \\ d_2 & 0 & 0 & 0 \\ d_3 & 6.55 & 0 & 0 \end{pmatrix}$ is obtained.

5.6. Integrated Algorithm

With these updating algorithms, network connection, data distribution and utility matrix can be updated to help sensors get local observations to support rational communication decisions even in dynamically changing networks. The whole data relay process with local knowledge can be seen in Algorithm 4.

This algorithm consists of two parts. In the first part, sensors update these three matrices based on the approaches mentioned in last three subsections (line 1–7). In the second part, v_i will try to fuse with these new data and make communication decisions for those that have not been fused (line 8–21). Principally, v_i checks them one by one to see if any new information can be fused (line 8–11). If the quality of the new information exceeds the predefined threshold, data related to this information will

be fused and added into the outdated data set. v_i will then inform other sensors to stop propagation of these outdated data pieces (line 11–13). If no information can be fused, this data will be added into the pending queue(line 15). After the information fusion process completes, v_i makes decisions for data in pending queue based on our lightweight matrix computation (line 16) by balancing between the benefits and energy cost of broadcasting. If the benefit is higher than the energy cost (line 18), v_i will add itself onto the path of this data and broadcast it (line 19–20), and update the data set $dataB$ with this data. Otherwise, this data will be ignored (line 21).

Algorithm 4 Data relay process for a sensor v_i.

1: **while** *true* **do**
2: $dataG \leftarrow sensedData()$;
3: $dataR \leftarrow receivedData()$;
4: $dataO \leftarrow outdatedData()$
5: $updateC(C, dataR)$;
6: $updateD(D, dataG, dataR, dataB, dataO)$;
7: $updateU(U, dataG, dataR, utilityTable)$;
8: **for all** $d_h \in dataG \cup dataR$ **do**
9: **if** $d_h \notin L_i$ **then**
10: $L_i \leftarrow L_i \cup \{d_h\}$;
11: **if** $Quality(I_j | L_i) \geq \tau_j)$ **then**
12: fuse into information I_j;
13: $dataO \leftarrow datarelatedto I_j$;
14: **else**
15: $pendingQue.add(d_h)$;
16: $F \leftarrow C \cdot ((\Gamma - D) \circ (D \times U)) - \beta \cdot (Eb + Er(C \cdot \overrightarrow{\Lambda}))$;
17: **for all** $d_h \in pendingQue$ **do**
18: **if** $F(d_h) > 0$ **then**
19: $d_h.path.add(v_i)$;
20: $broadcast(d_h)$;
21: $dataB.add(d_h)$

6. Experimental Section

In this section, we evaluate the performance of our proactive data relay algorithm CDU through simulations. In most scenarios, we used a field size of 600 $\times$ 600 m^2 where 500 mobile nodes were randomly scattered for target detection, and fused data detected into information. For each time step, 1% of sensors were made to move. Sensors communicated with each other in a broadcast medium, and the power of sensor radio transmitter was fixed so that any node within a 25 meter radius was within communication range. Sensor nodes within a communication range of another sensor are described as the neighbors. The power consumption (0.66 W in transmit mode, 0.395 W in receive mode) were chosen based on data from currently available radios [12]. The transmission time for a packet was fixed at 10ms. In each run, 100 events about targets randomly occurred. Each target could be detected by 9 sensors, and each detection generated a piece of data with a confidence c, $c \in [0, 1]$ is related to the distance between detected sensor and target [25]. For each event, one piece of information can be fused only when more than 6 data related to this target is aggregated by one sensor and the combined confidence is higher than 0.75.

We mainly compare the performance of our algorithm CDU with the Flooding [8], the Scalable Broadcast Algorithm (SBA) [14], and GRAdient Broadcast(GRAB) [12] algorithms. GRAB is on behalf of routing algorithms where only predefined sink nodes are able to fuse while the other three treat all sensors as potential sink nodes. Flooding is the most straightforward data relay algorithm, where each sensor broadcasts whatever new data it receives immediately without any reasoning. For SBA, only data that can reach new neighbors is broadcast. In SBA, neighbor knowledge in two hops is maintained by periodic "hello" packages. Also, in GRAB, each sensor maintains a cost field, and records the cost to the sink node in proportion to the distance to the sink node. Sensors only

broadcast data received from the sensor where cost is higher and relay data to sink nodes to fuse. Typical of most simulations, the network has one random sink node. When the number of nodes is 1000, 2 sink nodes are randomly chosen. In CDU, Flooding, and SBA, any sensor can fuse data into information if enough data is aggregated, while in GRAB only sink node can fuse. The value of U matrix in CDU is generated based on the D-S fusion rule [30]. Other relevant parameters defined are $\sigma = 0.97, \beta = 2.0, lengthOfPath = 2$.

To test if CDU works well, we measured the **Number of fused information** by all the sensors, the **Total energy cost** on communication of all sensors, and the **efficiency** = Number of fused information/Total energy cost, calculated as the energy consumed on communicating to fuse each piece of information. The efficiency indicates how the algorithms balance broadcasting to fusing more information and minimizing energy consumption. Results for each experiment are based on one hundred trials.

In the simulations, we first evaluate the impact of control parameters of CDU, including β and the length of path. Then we compare the performance of CDU with the other three related algorithms mentioned above in default settings with the progress of simulations. After that we compare these algorithms while independently changing the following environmental factors: the size of the field to test if CDU is scalable; the ratio of moved sensors to test if CDU can adapt to dynamic networks; and the threshold of fused targets. Finally, since GRAB is one of the routing protocols, we compare the final energy distribution between CDU and that of GRAB.

6.1. Different β and Different Length of Path

The value of β is a parameter used by CDU to balance forwarding data and minimizing energy consumption, which is defined in Equation (1). The path is a data structure used in CDU to store the sensor visited in each data. The length of path affects rich degree of the local matrix model.

To understand how β and the length of path affect efficient data relay, we varied β from 0 to 4, and varied the length also from 1 to 4. Experimental results of a number of fused information, energy cost on communication, and efficiency are illustrated in Figure 3. As we can see, the higher the value of β, the definition of useless data becomes more strict, and more data will be stopped from broadcasting especially those data with lower confidence or well known by neighbors. To some degree, it affects the number of fused information. As shown in Figure 3a,b, the number of fused information is proportional to β and the energy cost decreases as β increases. When $\beta = 0$, any data received will be broadcast and all the information can be fused. Figure 3c shows the efficiency for different β. When $\beta = 2.0$, the efficiency reaches a high position. The balance of data relay and saving energy is good. When $\beta > 2.0$, the efficiency still keeps a high value. However, the information fused is much less.

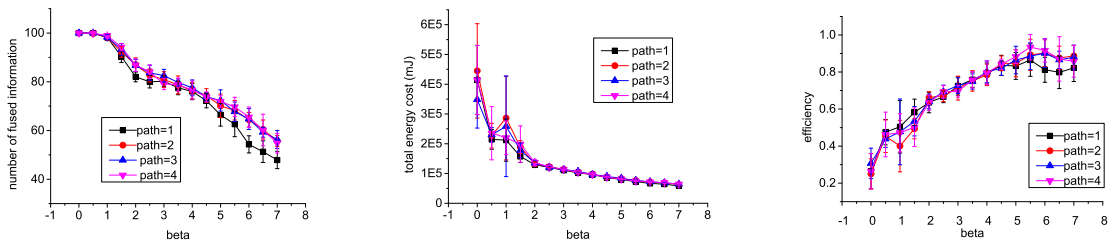
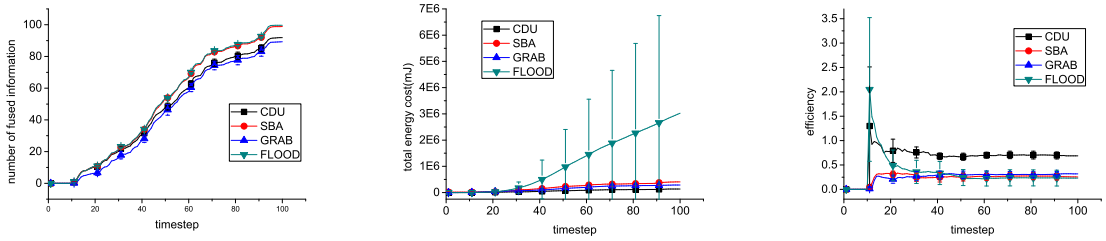


Figure 3. Experimental results with different β and different length of path. (**a**) introduces the comparisons of the number of fused information; (**b**) introduces the comparisons of the energy cost on communication; (**c**) introduces the comparisons of the fusion efficiency.

In the next simulations, the default value of β is 2.0. We notice that the performance of CDU is slightly affected by the length of the path. Regardless of the value of β, when the length of path is longer than 1, it performs better than when the length is 1. While the other values work almost the same for sensors, broadcast decisions can only affect their neighbors, and neighbors' neighbors have higher probabilities to be neighbors than nodes far away with sensors moving around. However, the longer the path, the more storage space needed to store these matrices. Therefore in the following simulation, the default length of a path is 2.

6.2. Different Algorithms

In this experiment, we compared the performance of our data relay algorithm CDU, with the Flooding algorithm, SBA and GRAB. 100 events about targets randomly occurred during 10th–95th time steps. The results are shown in Figure 4. At the first 10 time steps, no events exist, and no information is fused as Figure 4a shows. During this period, sensors in SBA broadcast hello messages to maintain 2-hop neighbor knowledge [16] and sensors in GRAB maintain their cost field. Therefore, in Figure 4b the cost of these two algorithms is more than 0. In the next 90 time steps, as time goes by, information is fused as more and more energy is consumed on data relay.



(a) (b) (c)

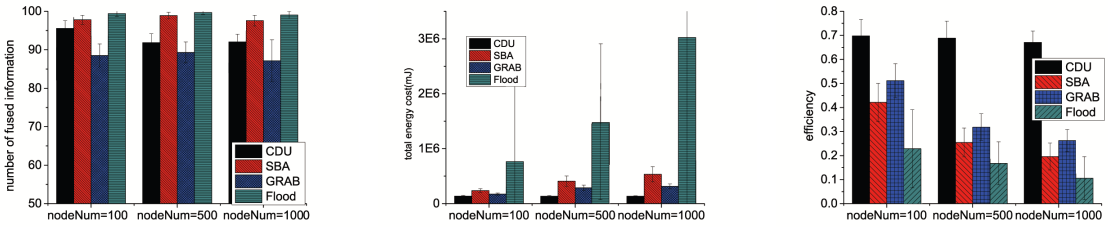
Figure 4. Experimental results for 500 sensors using different algorithms to fuse revelent data. (**a**) introduces the comparisons of the number of fused information. (**b**) introduces the comparisons of the energy cost on communication. (**c**) introduces the comparisons of the fusion efficiency.

In general, Flooding and SBA can fuse almost all the information. Correspondingly, their energy cost is higher than CDU because of flooding's no limitations on broadcast and SBA's purpose to cover all sensors with all data until it is possible to fuse them. GRAB fuse less information than CDU while consuming more for the long path to relay data to the fixed sink node. The efficiency of CDU is the highest of these algorithms for its effective constraining broadcast of the useless data.

6.3. Impact of Environmental Settings

6.3.1. Network Size

In this experiment, we studied how CDU scales to large networks for data relay. The number of sensors in the system are 100, 500 and 1000 while the side length of the squares are 250, 600, 850 to guarantee the same node density. The number of events is kept at 100. Figure 5 shows that regardless of the network size, CDU can well balance the broadcasting and saving energy, and its efficiency is always about 2 times as efficiency of the other algorithms.



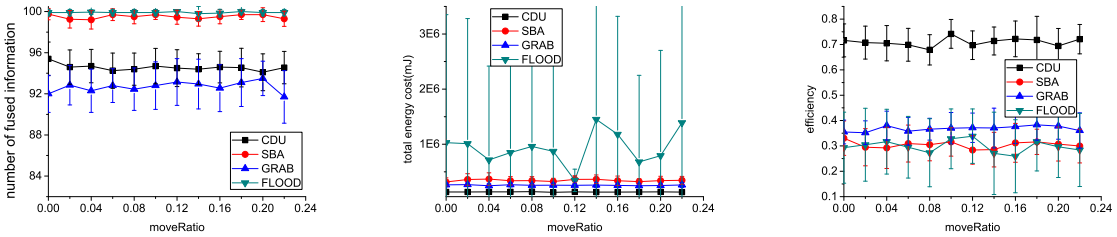
(a) (b) (c)

Figure 5. Experimental results with different network size using different algorithms. (**a**) introduces the comparisons of the number of fused information; (**b**) introduces the comparisons of the energy cost on communication; (**c**) introduces the comparisons of the fusion efficiency.

6.3.2. The Ratio of Moved Sensors

In this experiment, we evaluated the performance of these four algorithm to investigate if CDU can adapt to the dynamic network. We varied the move ratio of sensors at each time step from 0 to 0.22.

In SBA, sensors update two hop neighbor knowledge by periodic hello messages. For GRAB, sink nodes periodically broadcast advertisement messages, which help sensors update their cost field. However, in CDU , there is no periodic hello messages. A sensor will not broadcast hello message until it moves. Figure 6 shows that the performance of these four algorithms is unaffected by the variation of move ratio and CDU is always way ahead in efficiency.



(a) (b) (c)

Figure 6. Experimental results with different move ratio using different algorithms. (**a**) introduces the comparisons of the number of fused information; (**b**) introduces the comparisons of the energy cost on communication; (**c**) introduces the comparisons of the fusion efficiency.

6.3.3. Threshold

We first varied the minifusedNum from 3 to 8, while using a fixed 0.75 minifusedConfidence. Then, we varied the minifusedConfidence from 0.4 to 0.85 while using a fixed 6 minifusedNum threshold.

Figure 7 shows that when the threshold of minifusedNum or minifusedConfidence is higher, less information can be fused in the same 100 time steps. However, more energy consumption and lower efficiency on data relay for a sensor is needed to aggregate more data to reach the number and confidence threshold. Figure 7c,f show CDU has the highest efficiency of all these algorithms.

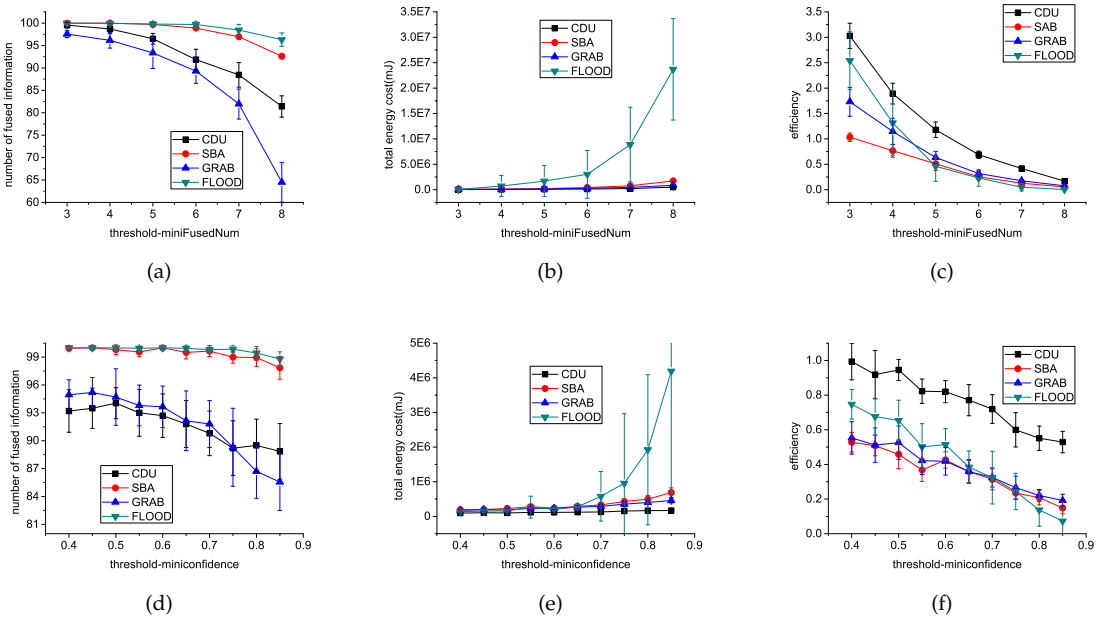


Figure 7. Experimental results for different threshold using different algorithms. (**a**) introduces the comparisons of the number of fused information with minifusedNum from 3 to 8; (**b**) introduces the comparisons of the energy cost on communication with minifusedNum from 3 to 8; (**c**) introduces the comparisons of the fusion efficiency with minifusedNum from 3 to 8; (**d**) introduces the comparisons of the number of fused information with minifusedNum from 0.4 to 0.85; (**e**) introduces the comparisons of the energy cost on communication with minifusedNum from 0.4 to 0.85; (**f**) introduces the comparisons of the fusion efficiency with minifusedNum from 0.4 to 0.85.

6.4. Energy Distribution

In this subsection, we compare the energy distribution between CDU and SBA, and the routing protocol GRAB. Figure 8 shows the final energy distribution after relaying data for 100 and 200 events while the network size is 500. As shown in these figures, the three algorithms all consume more energy in Figure 8b, where more events happened than in Figure 8a. In SBA and CDU, energy consumption is better distributed than in GRAB. For GRAB, data should be relayed to sink node to fuse and sink nodes, and nodes near sink nodes distinctly consume more energy. However, in SBA, sensors try to cover every node with data until fused, and the energy is much more than CDU.

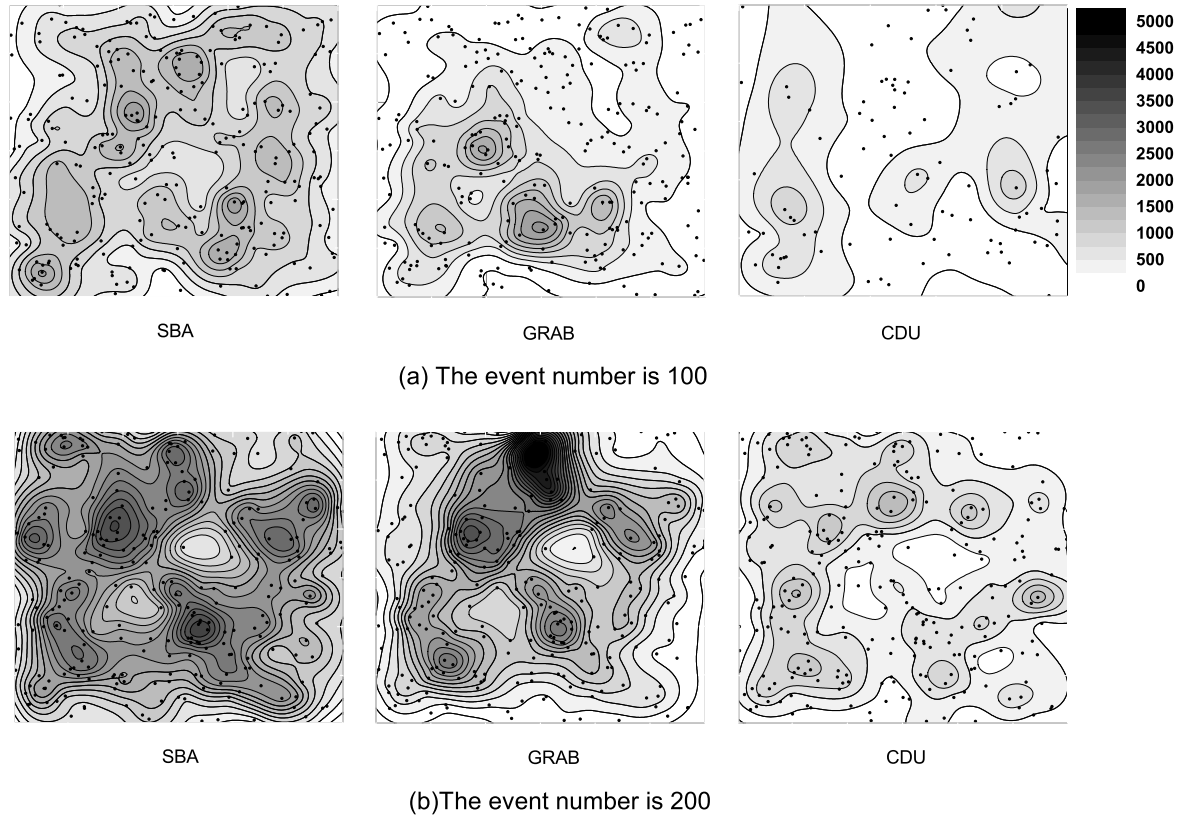


Figure 8. Energy consumption distribution(mJ) for different scenarios. (**a**) introduces the comparisons of the energy distribution after relaying data with 100 events; (**b**) introduces the comparisons of the energy distribution after relaying data with 200 events.

7. Conclusions

Large scale multi-sensor fusion is a very important issue in future networks and internet of things, especially in the domains of disaster response and military operations. However, previous centralized data aggregation algorithms for small sensor network are no longer feasible in considering of huge network expenses as well as the energy expenses for central nodes. In this paper, we have proposed an extremely economic data relay algorithm that sensors could proactively make broadcast decisions by neat matrix computation to balance transmission and save energy. By encoding sensors' local knowledge to three matrixes: network connection, data distribution and data utility matrix, they can do the reasoning for all the pending data by only a neat matrix computation. We also built heuristic algorithms for sensors to well maintain those matrixes with only a local view to the network so that this design can be adaptive to the scalable and dynamic environment. Our experimental simulations manifested that our approach is scalable and effectively balance between promoting data fusion process and saving energy to prolong the life time of the whole network.

Acknowledgments: This study was funded by National Natural Science Foundation of China 61370151 and 61202211, National Science and Technology Major Project of China 2015ZX03003012, Central University Basic Research Funds Foundation of China ZYGX2014J055, Huawei Technology Foundation YB2013120141, YB2015070068, and the Science and Technology on Electronic Information Control Laboratory Project.

Author Contributions: Y.X. and H.X.M. conceived and designed the experiments; Y.X. and H.X.M. and H.H.X. performed the experiments; Y.X. and H.X.M. and H.H.X. analyzed the data; Y.X. and M.L. contributed analysis tools; H.X.M. and M.L. wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tifenn, R.; Bouabdallah, A.; Challal, Y. Energy efficiency in wireless sensor networks: A top-down survey. *Comput. Netw.* **2014**, *67*, 104–122.
2. Maurizio, B.; Ossi, K.; Neal, P.; Suresh, V. Multiple target tracking with RF sensor networks. *IEEE Trans. Mobile Comput.* **2014**, *13*, 1787–1800.
3. George, S.M.; Zhou, W.; Chenji, H.; Won, M.; Lee, Y.O.; Pazarloglou, A.; Stoleru, R.; Barooah, P. DistressNet: A wireless ad hoc and sensor network architecture for situation management in disaster response. *IEEE Commun. Mag.* **2010**, *48*, 128–136.
4. Vicaire, P.; He, T.; Cao, Q.; Yan, T.; Zhou, G.; Gu, L.; Luo, L.; Stoleru, R.; Stankovic, J.A.; Abdelzaher, T.F. Achieving long-term surveillance in vigilnet. *ACM Trans. Sens. Netw.* **2009**, *5*, doi:10.1145/1464420.1464429.
5. Alexei, M.; Hugh, D.W. Decentralized data fusion and control in active sensor networks. In Proceedings of the Seventh International Conference on Information Fusion, Stockholm, Sweden, 28 June–1 July 2004; pp. 479–486.
6. Ochoa, S.F.; Santos, R. Human-centric wireless sensor networks to improve information availability during urban search and rescue activities. *Inf. Fus.* **2015**, *22*, 71–84.
7. Daniela, B.; Sytze, D.B.; Bregt, A.K. Value of information and mobility constraints for sampling with mobile sensors. *Comput. Geosci.* **2012**, *49*, 102–111.
8. Borges, L.M.; Velez, F.J.; Lebres, A.S. Survey on the characterization and classification of wireless sensor network applications. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1860–1890.
9. Intanagonwiwat, C.; Govindan, R.; Estrin, D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In Proceedings of the 6th annual international conference on Mobile computing and networking, Boston, MA, USA, 6–11 August 2000; pp. 56–67.
10. David, B.; Deborah, E. Rumor routing algorthim for sensor networks. In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, Atlanta, GA, USA, 28 September 2002; pp. 22–31.
11. Chu, M.; Haussecker, H.; Zhao, F. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *Int. J. High Perform. Comput. Appl.* **2002**, *16*, 293–313.
12. Liu, M.; Xu, Y.; Mohammed, A.W. Decentralized Opportunistic Spectrum Resources Access Model and Algorithm toward Cooperative Ad-Hoc Networks. Available online: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0145526 (accessed on 3 August 2016).
13. Heinzelman, W.R.; Kulik, J.; Balakrishnan, H. Adaptive protocols for information dissemination in wireless sensor networks. In Proceedings of the 5th Annual ACM/IEEE International Conference On Mobile Computing and Networking, Seattle, WA, USA, 15–19 August 1999; pp. 174–185.
14. Peng, W.; Lu, X.C. On the reduction of broadcast redundancy in mobile ad hoc networks. In Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing, Boston, MA, USA, 11 August 2000; pp. 129–130.
15. John, S.; Ivan, M. An Efficient Distributed Network-Wide Broadcast Algorithm for Mobile Ad Hoc Networks. Avaiable online: http://www.ece.rutgers.edu/~marsic/Publications/infocom2001nwb.pdf (accessed on 3 August 2016).
16. Hyojun, L.; Chongkwon, K. Multicast tree construction and flooding in wireless ad hoc networks. In Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Boston, MA, USA, 20 August 2000; pp. 61–68.
17. Hyocheol, J.; Hyeonjun, J.; Younghwan, Y. Dynamic probabilistic flooding algorithm based-on neighbor information in wireless sensor networks. In Proceedings of the International Conference on Information Network 2012, Bali, India, 1–3 Feburary 2012; pp. 340–345.
18. Williamson, S.A.; Gerding, E.H.; Jennings, N.R. Reward shaping for valuing communications during multi-agent coordination. In *International Foundation for Autonomous Agents and Multiagent Systems*, Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, Budapest, Hungary, 10–15 May 2009; Volume 1, pp. 641–648.
19. Raphen, B.; Alan, C.; Victor, L.; Shlomo, Z. Analyzing myopic approaches for multi-agent communication. *Comput. Intell.* **2009**, *25*, 31–50.

20. Karthikeyan, N.; Palanisamy, V.; Duraiswamy, K. Performance comparison of broadcasting methods in mobile ad hoc network. *Int. J. Future Gener. Commun. Netw.* **2009**, *2*, 47–58.
21. Akkaya, K.; Younis, M. A survey on routing protocols for wireless sensor networks. *Ad Hoc Netw.* **2005**, *3*, 325–349.
22. Chen, W.; Guha, R.K.; Kwon, T.J.; Lee, J.; Hsu, Y.Y. A survey and challenges in routing and data dissemination in vehicular ad hoc networks. *Wirel. Commun. Mobile Comput.* **2011**, *11*, 787–795.
23. Nishi, S.; Vandna, V. Energy efficient LEACH protocol for wireless sensor network. *Int. J. Inf. Netw. Secur.* **2013**, *2*, 333–338.
24. Heinzelman, W.B. Application-Specific Protocol Architectures for Wireless Networks. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.
25. Eriksson, O. Error Control in Wireless Sensor Networks: A Process Control Perspective. Ph.D. Thesis, Uppsala University, Uppsala, Sweden, 2011.
26. Bin, Y.; Paul, S.; Katia, S.; Yang, X.; Michael, L. Scalable and reliable data delivery in mobile ad hoc sensor networks. In Proceedings of the Fifth International Joint Conference On Autonomous Agents and Multiagent Systems, Hakodate, Japan, 8–12 May 2006; pp. 1071–1078.
27. Ondrej, K.; Neuzil, J.; Smid, R. Quality-based multiple-sensor fusion in an industrial wireless sensor network for MCM. *IEEE Trans. Ind. Electron.* **2014**, *61*, 145–157.
28. Wang, H.T.; Jia, Q.S.; Song, C.; Yuan, R.; Guan, X. Building occupant level estimation based on heterogeneous information fusion. *Inf. Sci.* **2014**, *272*, 145–157.
29. Ling, D.; Weili, W.; James, W.; Lidong, W.; Zaixin, L.; Wonjun, L. Constant-approximation for target coverage problem in wireless sensor networks. In Proceedings of the 31st Annual IEEE International Conference on Computer Communications, Orlando, FL, USA, 25–30 March 2012; pp. 1584–1592.
30. Bin, Y.; Katia, S. Learning the quality of sensor data in distributed decision fusion. In Proceedings of the 2006 9th International Conference on Information Fusion, Florence, Italy, 10–13 July 2006; pp. 1–8.