University of Wollongong

Research Online

Faculty of Engineering and Information Sciences - Papers: Part A

Faculty of Engineering and Information Sciences

1-1-2015

A multi-agent framework for packet routing in wireless sensor networks

Dayong Ye
University of Wollongong, dayong@uow.edu.au

Minjie Zhang
University of Wollongong, minjie@uow.edu.au

Yun Yang
Swinburne University of Technology, yyang@it.swin.edu.au

Follow this and additional works at: https://ro.uow.edu.au/eispapers

 Part of the Engineering Commons, and the Science and Technology Studies Commons

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

A multi-agent framework for packet routing in wireless sensor networks

Keywords

routing, framework, wireless, packet, agent, networks, sensor, multi

Disciplines

Engineering | Science and Technology Studies

Publication Details

Ye, D., Zhang, M. & Yang, Y. (2015). A multi-agent framework for packet routing in wireless sensor networks. Sensors, 15 10026-10047.

This journal article is available at Research Online: https://ro.uow.edu.au/eispapers/4180

A Multi-Agent Framework for Packet Routing in Wireless Sensor Networks

Dayong Ye [1,*], Minjie Zhang [2] and Yun Yang [1]

[1] School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne 3122, Australia; E-Mail: yyang@swin.edu.au

[2] School of Computer Science and Software Engineering, University of Wollongong, Wollongong 2522, Australia; E-Mail: minjie@uow.edu.au

* Author to whom correspondence should be addressed; E-Mail: dye@swin.edu.au; Tel: +61-3-9214-4899.

Academic Editor: Leonhard M. Reindl

Received: 5 March 2015 / Accepted: 22 April 2015 / Published: 28 April 2015

Abstract: Wireless sensor networks (WSNs) have been widely investigated in recent years. One of the fundamental issues in WSNs is packet routing, because in many application domains, packets have to be routed from source nodes to destination nodes as soon and as energy efficiently as possible. To address this issue, a large number of routing approaches have been proposed. Although every existing routing approach has advantages, they also have some disadvantages. In this paper, a multi-agent framework is proposed that can assist existing routing approaches to improve their routing performance. This framework enables each sensor node to build a cooperative neighbour set based on past routing experience. Such cooperative neighbours, in turn, can help the sensor to effectively relay packets in the future. This framework is independent of existing routing approaches and can be used to assist many existing routing approaches. Simulation results demonstrate the good performance of this framework in terms of four metrics: average delivery latency, successful delivery ratio, number of live nodes and total sensing coverage.

Keywords: multi-agent systems; wireless sensor networks; packet routing

---

1. Introduction

Due to recent technological advances, the manufacturing of small, low-power, low-cost and highly integrated sensors has become technically and economically feasible. These sensors are generally equipped with sensing, data processing and communicating components. Thus, such sensors can be used to measure ambient conditions in the environment surrounding them and then transform these measurements into signals. The signals can be processed further to reveal some properties about objects located in the vicinity of the sensors. The sensors then send such collected data, usually via a radio transmitter, to a command centre (also known as a sink or a base station) either directly or via several relaying sensors. A large number of these sensors can be networked in many applications that require unattended operations, hence producing a WSN. Currently, there are various applications of WSNs. These applications include battlefield surveillance [1], video capture, processing and communication [2,3], *etc*. Typically, WSNs contain hundreds or thousands of sensors, and these sensors have the ability to communicate with each other. Thus, packet routing is an important issue in WSNs. In addition, the routing issue in WSNs is more challenging than in other types of wireless networks, e.g., mobile *ad hoc* networks or cellular networks [4], because the power energy, storage capability and processing capacity of each sensor are usually constrained. Moreover, unlike general computer networks, where the destination of data is variable, in WSNs, the destination of nearly all data is a sink node. Therefore, designing routing approaches in such sensor networks needs careful resource management.

Over the last decade, many approaches have been proposed in order to solve the routing problem in WSNs [5,6]. These approaches have taken into account the inherent features of WSNs along with particular applications and architecture requirements. It is not a simple task to find and maintain routes in WSNs due to energy constraints and sensor failures. To mitigate energy consumption, existing routing approaches proposed for WSNs have employed some well-known routing techniques from other wireless networks and special techniques in WSNs, such as data aggregation, clustering, different node role assignment, *etc*. According to Al-Karaki and Kamal [7], almost all of the routing approaches can be classified into three major categories: flat routing approaches, hierarchical routing approaches and location-based (also known as geographic) routing approaches, although there are a few exceptional ones based on network flow or quality of service (QoS) awareness [8]. Routing approaches belonging to each of the three categories have advantages and disadvantages. In flat routing approaches, all sensors play the same role, and each sensor could be a source node or a relay node. Hence, the routing delay can be minimised, if the routing algorithm is carefully designed. The energy consumption is an outstanding problem, however, compared with the routing approaches belonging to other two categories, because sensors lack the knowledge of the global network, and thus, broadcasting is often required or data have to be repeatedly transferred in the network. Hierarchical routing approaches aim to cluster sensor nodes so that cluster heads can aggregate and reduce data to be transmitted to the sink node in order to save energy. Because packet transmission is the most energy expensive task [9], the reduction of packets is vital for the overall network. However, the packet routing delay will be lengthened, as cluster heads have to wait for a period for sensors in their clusters to transfer packets to them for aggregation. Location-based routing approaches use the position information to relay the packets to the desired regions rather than the whole network. Location-based routing approaches have good scalability, since the information

of the whole network is not required. Such location-based approaches, however, often need a device, such as a GPS, to provide the necessary position information. Furthermore, the cost and weight of a sensor will be increased if each sensor is equipped with such a device. Although several researchers have suggested using a virtual coordinate to provide the position information [10], this would incur extra energy consumption due to the communication process, which is necessary for acquiring such position information. In addition, to obtain such virtual coordinate-based position information, global information has to be used, e.g., the maximum ID among all sensors with the maximum hop distance from a specific node.

As can be seen from the above, since all existing routing approaches have some disadvantages, it is more important to develop an assistant tool to enhance existing routing approaches than to develop a new routing approach. To the best of our knowledge, this paper is the first one to propose a multi-agent framework that can be used to assist and enhance many existing routing approaches. The merits of this framework include that: (i) it is independent of individual existing routing approaches, so it can assist many existing routing approaches and enhance their performance; (ii) it is decentralised and operated based only on local information, and it does not need to change the physical network topology; and (iii) it does not only suit static WSNs, but also works well in mobile and open WSNs, where sensors are mobile and new sensors may join or existing sensors may leave the network.

The rest of the paper is organised as follows. In the next section, a detailed literature review is provided, which includes routing approaches belonging to the aforementioned three categories, especially those approaches developed after the publication of the survey papers in [4,7]. Then, the proposed framework is described in detail in Section 3. A simulation regarding the performance of the proposed framework is given in Section 4. Finally, this paper is concluded in Section 5.

2. Literature Review

As the routing issue is one of the most significant issues in WSNs, there is a large number of published prominent studies that focus on this issue. Akkaya and Younis [4] and Al-Karaki and Kamal [7] provided a thorough survey on routing approaches nearly at the same time. Thus, in this section, our review focuses primarily on the routing approaches developed after their survey papers were published.

2.1. Flat Routing Approaches

The first category of routing approaches is multi-hop flat routing. In such routing approaches, each sensor node typically plays the same role, and sensor nodes collaborate to perform sensing tasks.

Flooding and gossiping [11] are two classical approaches to relay packets in sensor networks without the need for any routing algorithms or topology maintenance. In flooding, each sensor that receives a packet broadcasts the packet to all of its neighbours, and this process continues until the packet arrives at the destination or the maximum number of hops for the packet is reached. On the other hand, gossiping is a slightly enhanced version of flooding where the receiving node sends the packet to a randomly-selected neighbour, which picks another random neighbour to forward the packet to, and so on, until the destination or the maximum hop is reached.

Heinzelman *et al.* [12,13] proposed a family of adaptive protocols, called sensor protocols for information via negotiation (SPIN). The SPIN family is designed to address the deficiencies of classical routing approaches, such as flooding and gossiping, which waste much energy and bandwidth when sending extra and unnecessary copies of packets by sensors that cover overlapping areas. SPIN consists of three steps. First, when a sensor has a data packet to transfer, it broadcasts the metadata of the data packet to its neighbours. The size of the meta packets is much smaller than that of the data packet. Then, if any neighbours are interested in the data packet, they will send back a request message to the sensor. Once the sensor receives the request message, it will transfer the data packet to those neighbours that are interested in that packet.

Rogers *et al.* [14] devised an energy-aware self-organised routing algorithm for WSNs. Their algorithm enables individual sensors to follow locally selfish strategies, which, in turn, result in a framework of a routing network with desirable global properties. In addition, the algorithm can adaptively deal with the changing sensor numbers and network topology.

Beraldi *et al.* [15] studied the lowest latency path problem and then devised a forwarding protocol based on biased random walks, where sensors use only local information about their neighbours and their next active period to make forwarding decisions. Specifically, their protocol, lukewarm potato, is a compromise between the shortest path forwarding and hot potato forwarding. Hot potato forwarding is a greedy approach to the fast propagation of a message toward the sink: a node, upon receiving a message, forwards the message to the neighbour that is the first to become active.

Quang and Kim [16] devised a gradient routing algorithm that uses two-hop information for industrial WSNs to enhance real-time performance with energy efficiency. The routing algorithm is based on the number of hops to the sink instead of distance. Specifically, the routing algorithm applied two-hop velocity-based routing [17] to a gradient-based WSNs so as to reduce both energy consumption and end-to-end delay.

Niu *et al.* [18] introduced a routing enhancement scheme. Their scheme first finds a robust guide path during the route discovery phase. Then, along this guide path, data packets are greedily forwarded toward the destination through nodes' cooperation without utilising location information. The forwarding procedure is based on the priorities of forwarding candidates, where nodes with the highest priorities will be picked up to forward data packets.

2.2. Hierarchical Routing Approaches

Hierarchical or cluster-based routing approaches, originally proposed in wired networks, are well-known techniques with special advantages related to efficient communication. The concept of hierarchical routing is also utilised to perform energy-efficient routing in WSNs. In a hierarchical architecture, high-energy sensors can be used to process and transmit information, whereas low-energy sensors can be employed to perform sensing tasks in the proximity of the target. Hierarchical routing is usually a two-layer routing approach, where one layer is composed of cluster heads and the other layer consists of cluster members. Cluster members are responsible for sensing tasks, while cluster heads are used to aggregate packets and transmit the aggregated packets to the sink node. Since the size

of aggregated packets is much less than that of raw packets, the energy consumption used for packet transmission can be significantly reduced, and thus, the overall network lifetime can be prolonged.

The most famous hierarchical routing protocol is LEACH (low-energy adaptive clustering hierarchy) [19]. LEACH randomly selects a few sensor nodes as cluster heads and rotates this role to evenly distribute the energy load among the sensors in the network. In LEACH, cluster heads compress packets arriving from cluster members that belong to the respective clusters. Cluster heads then send aggregate packets to a sink in order to reduce the number of packets that must be transmitted to the sink.

In [20], an enhancement over the LEACH protocol was proposed, *i.e.*, power-efficient gathering in sensor information systems (PEGASIS). PEGASIS is a near optimal chain-based protocol. The basic idea is that in order to extend network lifetime, sensor nodes need to communicate only with their closest neighbours, and these neighbours take turns in communicating with the sink. Unlike LEACH, PEGASIS avoids cluster formation and uses only one sensor node in a chain instead of multiple nodes to transmit the sink.

Tsai [21] presented a coverage-preserving routing protocol that was a modified version of LEACH. Unlike most routing protocols, the aim of his protocol is to preserve sensing coverage when some sensor nodes are no longer available, because, for example, their energy is used up. Tsai considered that simply increasing the lifetime of sensor nodes might not automatically achieve good sensing coverage. Thus, unlike LEACH, Tsai's protocol considered not only energy consumption, but also sensing coverage when clusters were formed.

Valentini *et al.* [22] devised a non-dominated algorithm to improve a simple hybrid routing protocol (SHRP) [23] in choosing the best route towards the sink node. Their algorithm allows simultaneous analysis of energy- and latency-related metrics to generate a Pareto-optimal solution, which has better time convergence and reliability than SHRP.

Mottola and Picco [24] developed an adaptive energy-aware routing protocol that was expressly designed for many-to-many communication, *i.e.*, simultaneously routing from multiple sources to multiple sinks. To increase network lifetime, their protocol minimises the number of nodes involved in many-to-many routing and balances forwarding load, reduces the amount of redundant information flowing in the network and decreases contention on the wireless medium and packet collisions.

2.3. Location-Based Routing Approaches

In this kind of routing, also known as geographic routing, sensor nodes are addressed in terms of their locations. The distance between neighbouring nodes can be estimated on the basis of incoming signal strength. The relative coordinates of neighbouring nodes can be obtained by exchanging information between neighbours. Alternatively, the location of sensor nodes may be achieved directly by communicating with a satellite using GPS if each node is equipped with a small low-power GPS receiver.

GAF (geographic adaptive fidelity) [25] is an energy-aware location-based routing algorithm designed primarily for mobile *ad hoc* networks, but may be applicable to WSNs, as well. In GAF, the network area is first divided into fixed zones. These zones then form a virtual grid. Inside each zone, nodes collaborate with each other to play different roles, and each node uses its GPS-indicated location to associate itself with a point on the virtual grid.

Yu *et al.*'s protocol [26], *i.e.*, GEAR (geographic and energy-aware routing), uses energy-aware and geographically-informed neighbour selection heuristics to route a packet toward the destination region. The key idea is to restrict the number of messages in directed diffusion by considering only a certain region rather than sending the messages to the whole network. Thus, the energy for transmission can be conserved.

Huang *et al.* [27] presented an energy-aware and interference-sensitive geographic routing protocol that focuses on minimising the total network energy consumption and reducing interference. Their routing protocol adaptively uses an anchor list to guide packet delivery and selects the minimum-interference link from the energy-optimal relay region for packet delivery.

Awad *et al.* [28] proposed the virtual cord protocol (VCP), which exploits virtual coordinates to provide efficient and failure-tolerant routing and packet management in sensor networks. Specifically, VCP uses two mechanisms to find paths to nodes and associated packet items. First, it relies on the virtual cord that always provides a path toward the destination. Second, locally available neighbourhood information is exploited for greedy routing.

Petrioli *et al.* [29] presented a geographic routing protocol that considered three important, but often neglected design challenges: routing around connectivity holes, resilience to localisation errors and efficient relay selection. In addition, their protocol integrates awake/asleep schedules, MAC, routing, traffic, load balancing and back-to-back transmissions. During the forwarding procedure, a sender checks the availability of its awake neighbouring nodes and channels. Then, the sender chooses the best relaying node based on its neighbouring nodes' forwarding performance. If several nodes have the same performance, positive geographic advancement toward the sink is used to discriminate among them. Such joint optimisation can make the protocol robust and realistic.

After reviewing related work, it can be seen that each approach has its own strengths and limitations. We propose, therefore, that it is important to devise a framework that can be applied to many existing routing approaches in order to improve their performance. This paper proposes such a framework, which will be described in detail in the next section.

3. The Multi-Agent Framework

In order to realise the framework, a two-layer architecture is introduced, where the first layer is the wireless sensor network and the second layer is a multi-agent cooperation network, as shown in Figure 1.

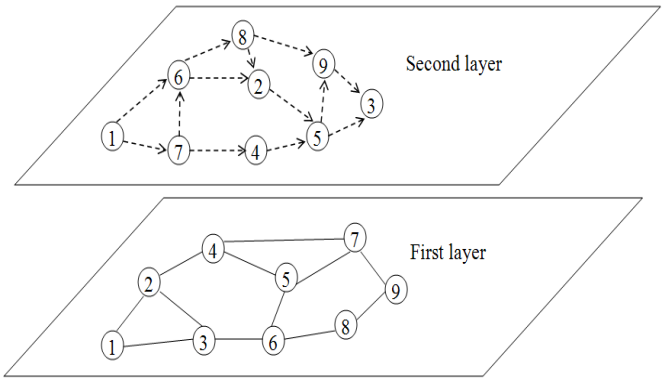

Figure 1. A sample two-layer architecture.

In the first layer, the wireless sensor network layer, sensors are connected by a wireless medium, such as infrared devices or radio, which is represented by the solid lines connecting the two sensor nodes. The first layer is the physical network, which really exists. In the second layer, each sensor is modelled as an agent, and agents are linked by a cooperative relation, represented as dashed lines. In the rest of the paper, the two terms, sensor and agent, are used interchangeably. The second layer, the multi-agent cooperation network layer, is an abstract network, which does not really exist. The multi-agent cooperation network is formed on the basis of the sensors' past cooperation. For example, in Figure 1, if many packets sent from Sensor 1 have to be forwarded by Sensor 7, Sensor 1 may add Sensor 7 as one of its cooperative neighbours. Then, in the future, if Sensor 1 has packets to be sent, it first sends the packets to Sensor 7, and Sensor 7 then forwards the packets to the destinations for Sensor 1. For the process that Sensor 1 sends packets to Sensor 7 and Sensor 7 forwards packets to the destinations, Sensors 1 and 7 can use any existing routing approaches. It can be seen that the multi-agent cooperation network layer is used to guide the packet routing process in the wireless sensor network layer in order to improve the routing efficiency, instead of becoming a concrete routing protocol that directly operates in the wireless sensor network layer. Thus, in this case, many existing routing protocols can be employed in the wireless sensor network layer. Such a two-layer architecture is necessary, because in the first layer, the topology of a WSN cannot be altered (unless some nodes move or leave the network, or new nodes join the network), and hence, the proposed framework has to work in the second layer, the multi-agent cooperation network layer, which enables each agent to keep the most useful cooperative neighbours and, in turn, assists existing routing approaches to achieve better performance.

In this paper, the sensor network is defined as a graph, $G = (V, E)$, where $V = \{v_1, ..., v_n\}$ is the set of n sensors in the network and $E \subseteq V \times V$ is a set of physical links. Two sensors, v_i and v_j, are physical neighbours if $(v_i, v_j) \in E$ and they can directly transmit packets between each other. $N_i = \{v_j | (v_i, v_j) \in E\}$ indicates sensor v_i's physical neighbour set. Similarly, the cooperation network is defined as a graph, as well, $G' = (A, E')$, where A is the set of agents and $E' \subseteq V \times V$ is a set of cooperation links. Each agent represents a sensor node. Two agents, ag_i and ag_j, are cooperative neighbours if $(ag_i, ag_j) \in E'$. $N'_i = \{ag_j | (ag_i, ag_j) \in E'\}$ indicates agent ag_i's cooperative neighbour set. It should be noticed that physical neighbours are symmetrical, namely that if $(v_i, v_j) \in E$, then $(v_j, v_i) \in E$ holds, but cooperative neighbours are asymmetrical, namely that when $(ag_i, ag_j) \in E'$, it is not necessary that $(ag_j, ag_i) \in E'$ holds. Initially, each sensor's physical neighbour set is identical to its cooperative neighbour set (although later, its cooperative neighbour set may be adapted).

Before devising the framework, it is necessary to introduce some evaluation indices to estimate each sensor, as well as the sensor network. These indices include energy consumption, storage consumption and sensing coverage.

3.1. Energy Consumption

Energy consumption relies on how many packets are being transmitted and the transmission distance of each packet. For a single sensor, say v_i, its energy consumption for transmitting a k-bit message to a receiver, say v_j, with a distance d away can be calculated using Equation (1) [19].

$$Ene_{T_i}(k,d) = k(E_{elec} + \epsilon_{amp} d^{\beta}) \tag{1}$$

where E_{elec} represents the energy being dissipated to operate the transmitter or receiver circuitry per bit and ϵ_{amp} denotes the energy dissipation of the transmitter power amplifier for transmitting a bit to a receiver with a distance of $d = 1$ unit away, and β is a constant coefficient that denotes the path loss exponent.

For the receiver, v_j, its energy consumption for receiving the k-bit message is:

$$Ene_{R_j}(k) = kE_{elec} \tag{2}$$

Thus, for the whole sensor network, its energy consumption in a pre-defined period is the sum of the energy consumption caused by sensor nodes transmitting and receiving data packets in that period.

Certainly, other factors, such as idle listening, overhearing and synchronisation, can also incur energy consumption. To simplify the model, such factors are not taken into consideration.

3.2. Storage Consumption

Storage consumption involves two types of consumption. The first type of storage consumption is for maintaining packets in sensors' waiting lists. For example, if a sensor has several packets to send, but in a time slot, it can only send one packet, it has to keep the remaining packets in its waiting list and transmit them in the succeeding time slots. It is important for a sensor to keep as few packets in its waiting list and as short of a period of time as possible, so that both the sensor's storage consumption and the packet transmission latency can be reduced. For a single sensor, v_i, its storage consumption for maintaining packets can be calculated using Equation (3).

$$Stor_i^{(1)} = S^{(1)} \cdot (\sum_{j=1}^{|W_i|} t_{ij}^{(1)}) \tag{3}$$

where W_i denotes the waiting list of sensor v_i, $t_{ij}^{(1)}$ demonstrates the number of time slots during which sensor v_i maintains a packet j and $S^{(1)}$ is a constant to represent the storage coefficient for keeping packets.

The second type of storage consumption for an individual sensor is to keep (both physical and cooperative) neighbours, which can be calculated by employing Equation (4).

$$Stor_i^{(2)} = S^{(2)} \cdot (\sum_{j=1}^{|N_i|} t_{ij}^{(2)} + \sum_{l=1}^{|N_i'|} t_{il}^{(3)}) \tag{4}$$

where N_i and N_i' are the physical and cooperative neighbours of v_i, respectively, which have been described above; $t_{ij}^{(2)}$ indicates the time slots during which sensor v_i keeps v_j as its physical neighbour; $t_{il}^{(3)}$ indicates the time slots during which sensor v_i, *i.e.*, agent ag_i, keeps ag_l as its cooperative neighbour; $S^{(2)}$ is a constant to represent the storage coefficient for keeping neighbours.

The reason for taking time into account when computing storage consumption is that in Equation (3), the time used to keep packets can indirectly reflect transmission latency, and in Equation (4), the time

used to keep neighbours can, to some extent, reflect the computational overhead associated with taking each neighbour into consideration whenever a sensor sends a packet.

Likewise, for the whole sensor network, the storage consumption is the sum of each sensor's storage consumption.

3.3. Sensing Coverage

Sensing coverage is also an important property of a sensor network, although this property was overlooked by many researchers. As described in [21], saving unnecessary power consumption can prolong the lifetime of sensor nodes, but does not necessarily imply that a better sensing coverage can be achieved. Hence, it is necessary to consider sensing coverage as a separate property.

In many applications, sensor nodes are deployed randomly over the entire desired area. In this way, the sensing areas of different nodes may partially overlap. When a local area has a much higher node density than the average node density, a target location in that area may be covered by multiple sensors. On the other hand, if the node density of a local area is much lower than the average node density, a target location is more likely covered by only one sensor. The normalised effective sensing area, η, of a sensor, v_i, is calculated using Equation (5) [21].

$$\eta = \eta_0 + \sum_{m=1}^{\infty} \frac{\eta_m}{m+1} \tag{5}$$

where η_0 is the percentage of v_i's sensing area that is covered by v_i only and η_m represents the percentage of v_i's sensing area that is covered by v_i and any other m neighbouring nodes. The value of η is within $(0, 1]$. If the sensing area of a node does not overlap with that of any other nodes, the value of η is one. On the other hand, if the sensing area of multiple nodes overlaps the sensing area of a node, the value of η may be much less than one.

For the whole sensor network, the sensing coverage is the ratio between the total sensing area covered by all sensor nodes and the target area. For example, if the target is 1000 m^2 and all of the sensors totally cover 900 m^2 (due to some sensors failure), the sensing coverage of this sensor network now is 0.9.

3.4. Framework Design

The aim of the multi-agent framework is to improve the overall routing performance of existing routing approaches. In this framework, each agent can autonomously create a cooperative neighbour set and each agent can dynamically add or remove cooperative neighbours into or from its cooperative neighbour set. Afterwards, when a sensor has a packet to send, it sends the packet to one of its cooperative neighbours, and the cooperative neighbour then relays the packet to one of the cooperative neighbour's cooperative neighbours until the packet reaches the sink node. For the packet transmission between cooperative neighbours, any existing routing approach can be used.

3.4.1. Cooperative Neighbour Set Creation

The creation of a cooperative neighbour set is based on the sensors' historical information. We use an example to describe the creation procedure. A sensor, v_j, often relays packets for v_i. Then, v_j may

want itself to be v_i's cooperative neighbour. v_j then sends a request message to v_i to suggest v_i add v_j as one of v_i's cooperative neighbours. The request message includes the information regarding v_j itself, such as the remaining energy of v_j, the normalised effective sensing area of v_j, the distance between v_j to the sink node and the average number of hops from v_i to v_j (that is derived from the previously relayed packets, each of which contains a hop-count field). Once v_i receives this request packet, v_i will evaluate the reward of adding v_j as one of v_i's cooperative neighbours. v_i estimates that if it adds v_j as a cooperative neighbour, v_i's storage consumption may increase and v_j's energy consumption may rise (because in the future, v_i will send more packets to v_j than before). Such a consumption is a negative reward. On the other hand, however, in the future, the packet routing delay could be reduced, because the routing procedure becomes more targeted. Furthermore, the energy consumption of other sensors could be reduced, since packets would be relayed by fewer sensors. The storage and energy consumption can be estimated using Equations (1), (2) and (4). The estimation of packet routing delay is based on two factors: the communication radius of a sensor, r, and the distance between sensor node v_i and sink node s, which is recorded as $d_{v_i \to s}$. If there is more than one sink in the sensor network, such a distance demonstrates the shortest distance. In this study, it is assumed that each sensor has the same communication radius, r. The distance between sensor nodes to the sink node can be obtained during the initialisation stage of a sensor network. Once the two factors, r and $d_{v_i \to s}$, are known, the estimation of packet routing delay can be obtained using Equation (6), where δ is a constant coefficient, which is greater than one.

$$del = \delta \cdot \frac{d_{v_i \to s}}{r} \tag{6}$$

The reason for using r and $d_{v_i \to s}$ to calculate the estimation of the packet routing delay is that the result of using $d_{v_i \to s}$ to divide r can approximately indicate the number of hops from sensor v_i to sink s. Since a packet transmitted between two neighbouring sensors needs one time slot, the number of hops from sensor v_i to sink s can approximately reflect the number of time slots needed to transmit a packet from v_i to s, namely the packet delay from v_i to s.

Hence, if v_i wants to add v_j as one of its cooperative neighbours, the estimated increase of storage consumption is $S^{(2)} \cdot \hat{t}_{ij}^{(3)}$, where $\hat{t}_{ij}^{(3)}$ indicates the intended time slots during which v_i keeps v_j as one of its cooperative neighbours; the estimated increase of energy consumption on v_j is $\zeta \cdot Ene_{R_j}(k)$, where k denotes the total size of messages (measured in bits) passed from v_i to v_j and ζ is a constant coefficient, which is greater than one; the estimated routing delay reduction is $\zeta \cdot (del_i - del_j)$; the estimated reduction of energy consumption on other sensor nodes is $\zeta \cdot hop_{i \to j} \cdot (Ene_T(k, \frac{r}{\delta}) + Ene_R(k))$. After the estimation, the reward of v_i, namely $\mathcal{R}_i$, is derived by summing up the four estimated values. Here, the four estimated values can be assigned different weights in different situations to demonstrate their different importance levels. For example, if in some situations, energy consumption is the most important, the energy consumption will be assigned the highest weight among these estimated values. In this paper, we do not consider special use cases. Thus, the four estimated values are assigned the same weight, *i.e.*, one.

Once the reward value, $\mathcal{R}_i$, is obtained, sensor v_i has to make a decision regarding whether to add v_j as one of its cooperative neighbours. A Q-learning algorithm is developed for this decision making process (see Algorithm 1). Q-learning is one of the most-studied reinforcement learning (RL) algorithms and has been applied with success in several domains, from relatively simple toy

problems, such as cliff-walking [30], to more complex ones, such as web-based education [31] and face recognition [32]. One of the attractive features of Q-learning is that it assumes no knowledge about the environment (such as state transition functions or reward functions).

Algorithm 1. Learning progress of sensor v_i

1 Initialise Q value for each available action arbitrarily;
2 for $k = 0$ to a predefined integer **do**;
3 | calculate π;
4 | **for each** available action $a \in A_i$ **do**;
5 | | $Q_{k+1}(a) = Q_k(a) + \pi(a)\alpha_1(\sum_a \mathcal{R}(a)\pi(a) - Q_k(a))$;
6 | **end for**
7 end for
8 $a_{opti} \leftarrow argMax(Q)$;
9 v_i takes the action a_{opti};

Sensor v_i first arbitrarily initialises the Q-value of each available action (Line 1). There are two available actions for sensor v_i: whether or not to add v_j as one of its cooperative neighbours. Next, v_i launches the learning process for the Q-value of each available action (Lines 2–7). In Line 2, the number of iterations is set to 30, which can yield adequate results. In Line 3, π is calculated using Equation (7) for each available action, a, which is the ϵ-greedy exploration method [33], where $0 < \epsilon < 1$ is a small positive number and n is the number of available actions of a sensor. The ϵ-greedy exploration method defines a semi-uniform probability distribution, in which the current best action is selected with probability $1 - \epsilon$ and a random action is chosen with probability ϵ.

$$\pi(a) = \begin{cases} (1-\epsilon) + (\epsilon/n), \text{if } Q(a) \text{ is the highest value} \\ \epsilon/n \text{ , otherwise} \end{cases} \tag{7}$$

Line 5 demonstrates the Q-value update equation, where $0 < \alpha_1 < 1$ is the learning rate and $\mathcal{R}(a)$ is the estimated reward received by v_i after taking action a. After finishing learning, v_i executes the action that could maximise the Q-value (Lines 8–9). Finally, if v_i decides to add v_j as one of its cooperative neighbours, v_i will send an acknowledgement message back to v_j to let v_j know that v_j has been added to v_i's cooperative neighbour set. Conversely, if v_i does not decide to add v_j as its cooperative neighbour, v_i just ignores v_j's request message and does not send any message back in order to save energy consumption. Having this algorithm, the expected behaviour of sensors can be derived by tracking the actions with the highest Q-value over the learning process of each sensor. The importance of obtaining a Q-learning algorithm with ϵ-greedy exploration is also justified through a large number of applications. For example, Galstyan *et al.* [34] applied a Q-learning algorithm with ϵ-greedy exploration to develop a decentralised resource allocation mechanism; Gomes and Kowalczyk [35] studied the problem of learning demand functions; and Ziogos *et al.* [36] investigated the development of bidding strategies.

Similarly, if v_j has been a cooperative neighbour of v_i, but v_j seldom relays packets for v_i, then v_i may remove v_j from its cooperative neighbour set so as to reduce v_i computational overhead associated with

taking v_j into account whenever v_i transfers a packet. The decision making of v_i, regarding removing v_j from its cooperative neighbour set, is analogous to the decision making process about adding v_j as a cooperative neighbour.

Of course, a sensor can directly choose an action that could maximise its reward, and thus, the learning process is not really necessary. Kaelbling *et al.* [37] note, however, that this kind of algorithm, which always takes the highest rewards action and overlooks the trade-off between exploitation and exploration, may converge to a sub-optimal state.

3.4.2. Selecting a Cooperative Neighbour for Relaying

When a sensor, say v_i, has a packet to transfer, it needs to decide which cooperative neighbour is the best choice for relaying the packet. The selection of a cooperative neighbour is based on each cooperative neighbour's properties, which are embodied in the previous request message (As time progresses, the information embodied in the request message may be outdated. Thus, if a cooperative neighbour's status significantly changes, it will send a new message to v_i regarding its current status.). These properties include the remaining energy of v_j, the normalised effective sensing area of v_j, the distance between v_j to the sink node and the average number of hops from v_i to v_j. Generally, sensor v_i prefers to forward packets to a cooperative neighbour, which has more remaining energy, because such a cooperative neighbour is not likely to run out of energy. v_i also prefers a small normalised effective sensing area, because even if such a cooperative neighbour is depleted, the overall coverage area will not be greatly affected, since the sensing area of that cooperative neighbour is overlapped by multiple sensors. A short distance from v_j to the sink and a small number of hops from v_i to v_j are also preferred, as a short distance and a small number of hops usually imply little transmission delay. Therefore, for v_i, there are different benefits for choosing different cooperative neighbours to relay a packet. Such a benefit, for v_i choosing v_j, can be calculated using Equation (8).

$$\mathcal{B}(v_j) = \gamma_1 \cdot Rem_Eng(v_j) - \gamma_2 \cdot \eta(v_j) - \gamma_3 \cdot d_{v_j \to s} - \gamma_4 \cdot hop_{i \to j}, \tag{8}$$

where $Rem_Eng(v_j)$ means the remaining energy of v_j, $\eta(v_j)$ indicates the normalised effective sensing area of v_j, $d_{v_j \to s}$ is the distance from v_j to the sink node, $hop_{i \to j}$ denotes the number of hops from v_i to v_j and γ_1, γ_2, γ_3, γ_4 are coefficients that are used to normalise the four values in the same magnitude.

After calculation of the benefit for selecting each cooperative neighbour, sensor v_i employs another Q-learning algorithm to select a cooperative neighbour to forward the packet (Algorithm 2).

In Line 1, v_i initialises the Q-value of each cooperative neighbour as zero. $\boldsymbol{\pi}$ is a vector, which demonstrates the probability with regard to selecting each cooperative neighbour to transfer the packet, *i.e.*, $\boldsymbol{\pi} = \langle \pi(v_1), ..., \pi(v_{|N_i'|}) \rangle$. v_i initially sets each element in $\boldsymbol{\pi}$ to $\frac{1}{|N_i'|}$ for simplicity. Then, in Lines 2–4, v_i updates the Q-value of each cooperative neighbour based on the benefit of selecting each cooperative neighbour, $\mathcal{B}(v_j)$, where α_2 is the Q-learning rate. v_i averages the benefits of selecting its cooperative neighbours and uses this average value and the newly created Q-values to update the probability vector $\boldsymbol{\pi}$, where τ is the policy learning rate (Lines 5–8). The probability vector update method was developed by Zhang *et al.* [38]. In Line 9, the function $Normalise()$ is used to constrain $\boldsymbol{\pi}$ to a legal probability distribution, *i.e.*, $\sum_{j=1}^{|N_i'|} \pi(v_j) = 1$, and $Normalise(\boldsymbol{\pi})$ has the minimum Euclidean distance to $\boldsymbol{\pi}$. Finally, in Line 10, v_i selects a cooperative neighbour to transfer the packet based on the probability

vector $\boldsymbol{\pi}$. This selection is akin to the roulette, namely that cooperative neighbour v_j with a large value of $\pi(v_j)$ is more likely to be selected. Algorithm 2 is a direct policy search algorithm that learns stochastic policies. As argued in [39], stochastic policies can work better than deterministic policies in partially observable environments (such as packet routing in WSNs, where each sensor has only partial information about the entire WSN). In this algorithm, the policy search is based on the difference between the benefit of an action and the current average benefit. Thus, this algorithm encourages each sensor to select the cooperative neighbours, which are associated with high benefits, for packet relaying. Since the benefit of selecting each cooperative neighbour is dynamic, a sensor does not always select relaying neighbours from a small part of its cooperative neighbours. Therefore, the relaying role can be automatically rotated among the sensor's cooperative neighbours, which can then prolong the lifetime of each cooperative neighbour.

Algorithm 2. Choice of a cooperative neighbour for relaying

\\ according to v_i's view \\
1 v_i initialises Q-values and $\boldsymbol{\pi}$;
2 for each cooperative neighbour of v_i, *i.e.*, $v_j \in N_i'$ **do**
3 $\quad Q(v_j) \leftarrow (1-\alpha_2)Q(v_j) + \alpha_2 \cdot \mathcal{B}(v_j)$;
4 end for
5 $\bar{\mathcal{B}} \leftarrow \frac{1}{|N_i'|}\sum_{v_j \in N_i'} \mathcal{B}(v_j)$;
6 for each cooperative neighbour of v_i, *i.e.*, $v_j \in N_i'$ **do**
7 $\quad \pi(v_j) \leftarrow \pi(v_j) + \tau(Q(v_j) - \bar{\mathcal{B}})$;
8 end for
9 $\boldsymbol{\pi} \leftarrow Normalise(\boldsymbol{\pi})$;
10 v_i selects a cooperative neighbour based on $\boldsymbol{\pi}$;

In this framework, each sensor node in the network has to have a global unique ID to distinguish different nodes in the network (see Figure 1). Although some addressing schemes, e.g., [40,41], can be used to handle this problem, it is still sometimes difficult for each node to have a global unique ID, especially in a large WSN. This is a drawback of this framework, and overcoming this drawback will be the subject of one of our future studies.

3.5. Complexity Analysis of the Framework

This framework is proposed as an assistant tool to enhance the performance of existing routing approaches, but meanwhile, the framework incurs extra complexity, which is independent of the complexities of specific underlying routing approaches. Such complexity is incurred during the creation and use of cooperative neighbour sets. As described above, a node, e.g., v_i, first estimates the reward of adding another node, e.g., v_j, to v_i's cooperative neighbour set. Then, node v_i uses Algorithm 1 to decide whether or not to add v_j as one of its cooperative neighbours. In the future, when v_i has a packet to send out, it will choose one of its cooperative neighbours to relay the packet.

The estimation of reward does not incur much extra complexity, as there is only the computation of simple equations (Equations (1)–(5)). Then, for the decision making of a node, in Algorithm 1, there

is a nesting loop (Lines 2–7). The number of iterations of the loop, Lines 2–7, is 30. The number of iterations of the loop, Lines 4–6, is two, as there are two available actions: adding a node as a cooperative neighbour or not. Because the number of iterations of both of the loops is constant, the complexity of Algorithm 1 is $\mathcal{O}(1)$. For the selection of cooperative neighbours, in Algorithm 2, there are two loops. The number of iterations of each loop equals the number of cooperative neighbours of a sensor node. In the worst case, the number of cooperative neighbours of a sensor node is the number of sensor nodes in the WSN. As the number of sensor nodes in the WSN is n, the complexity of Algorithm 2 is $\mathcal{O}(n)$. It should be noted that to alleviate the potential worst case, sensor nodes are allowed to remove inefficient cooperative neighbours (recall Section 3.4.1).

In addition, there is also storage and communication consumption. Every time a node adds another node as a cooperative neighbour, the focal node has to use some storage space to store such a neighbouring relationship. In the worst case, a node adds all of the n nodes in the WSN to be its cooperative neighbours. Thus, the storage complexity is $\mathcal{O}(n)$. Furthermore, before a node adds another node as a cooperative neighbour, a message has to be sent. Similarly, in the worst case, a node adds all of the n nodes in the WSN to be its cooperative neighbours, which means that n messages have to be sent. Thus, the communication complexity is $\mathcal{O}(n)$, as well. Certainly, the potential worst case can be alleviated by allowing sensor nodes to remove inefficient cooperative neighbours.

4. Simulation and Analysis

As illustrated in the previous section, the proposed framework is not a concrete routing approach, but rather a framework used to enhance existing routing approaches. Therefore, to assess the performance of the proposed framework, three existing routing approaches are selected, and each routing approach belongs to one of the three categories described in Section 2. Specifically, the selected routing approaches are gossiping [11] (a flat routing approach), LEACH-revised (a hierarchical routing approach) and GEAR [26] (a location-based routing approach). The reason for choosing the three routing approaches is that they are classical and easy to implement. Certainly, other routing approaches can also be employed in our simulation, but using the three classical approaches is sufficient to demonstrate the effectiveness of the proposed framework. Another of our future studies will take other more complex routing approaches into simulation. The three selected approaches are described in detail as follows.

1. Gossiping: In gossiping, a sensor node sends the packet to a randomly selected neighbour, which picks another random neighbour to forward the packet to, and so on, until the expiry time of the packet or the destination is reached, whichever comes first.
2. LEACH-revised: LEACH [19] randomly selects a pre-determined number of sensor nodes as cluster heads and rotates this role to evenly distribute the energy load among the sensors in the network. The role rotation is depicted as below. A sensor node chooses a random number, r, between zero and one. If this random number is less than a threshold value, $T(n)$, the node becomes a cluster head for the current round. The threshold value is calculated based on an equation that incorporates the desired percentage, p, to become a cluster head in the current round and the set of nodes that have not been selected as a cluster head in the last $\frac{1}{p}$ rounds. Then, all elected cluster heads broadcast an advertisement message to the rest of the nodes in the network

that they are the new cluster heads. The problem with LEACH, however, is that it assumes that the cluster heads can directly send packets to the sink node, which is infeasible in large-scale WSNs. Thus, we introduce Yu *et al.*'s multi-hop routing algorithm [42] used by cluster heads to transfer packets to the sink node. Yu *et al.*'s algorithm is based on the cost of each wireless link, which is evaluated by using the remaining power of each node.

3. GEAR: There are two phases in GEAR. First, forwarding packets toward the target region: upon receiving a packet, a node relays the packet to the neighbour which is closer to the target region than itself. If there is more than one neighbour, the nearest neighbour to the target region is selected. However, if all of the neighbours are further from the destination region than the node itself, one of the neighbours is picked based on their evaluated cost to the destination region. Second, forwarding packets within the region: if the packet has arrived at the target region, it is diffused in the region by either recursive geographic forwarding or restricted flooding, until the packet reaches the sink node.

4.1. Simulation Setup

The simulation is divided into two parts. One is in a stationary WSN, and the other is in a dynamic WSN. In the stationary WSN, the position of each sensor node, including sink nodes, is fixed, meaning that sensors cannot move, and the WSN is closed, which means that no new sensors will join the network. On the other hand, in the dynamic WSN, each sensor node, involving sink nodes, has a pre-defined probability to move, and the WSN is open, which means that at each time slot, with a pre-determined probability, a new sensor may join the network. Performance is measured by the four quantitative metrics: average delivery latency, successful delivery ratio, number of live nodes and total sensing coverage. The meanings of the four metrics can be easily deduced from their names. For packet delivery, it is supposed that a packet is transmitted and then received in one time slot. The simulation is operated in a simulated 1000 m $\times$ 1000 m area, and 200 sensor nodes and five sink nodes are deployed randomly in this area. The communication radius, that is the furthest delivery radius, of each sensor is 60 m, and the sensing coverage radius of each sensor is 40 m. The initial energy of each sensor is set to 5 J. The average size of a packet is 50 bytes, and the actual size of a packet is based on the normal distribution with variance equal to 10. At each time slot, each node generates a packet based on a pre-defined probability, and the expiry time of a packet is based on exponential distributions. If a packet cannot reach a sink before its expiry time, the delivery of this packet is considered as a failure. Undelivered packets are also accounted for in the average delivery latency computation. The values and meanings of those parameters mentioned in the previous section, are listed in Table 1. These values were chosen experimentally to provide the best results. Each simulation was executed 200-times, and the average results are displayed in the following figures. We used JAVA to build the simulation platform, where two two-dimensional arrays were used to represent the two layers of a sensor network and another two-dimensional array was defined to indicate the distance between two neighbouring sensors. 'Sensor' was programmed as a class, and each sensor is an object of this class.

Table 1. Parameter settings.

Parameters	Values	Explanations
E_{elec}	50 nJ/bit	Energy dissipation parameter
ϵ_{amp}	0.1 $nJ/bit/unit$	Energy dissipation parameter
β	3	Path loss exponent
$S^{(1)}$	0.01	Storage coefficient for keeping packets
$S^{(2)}$	0.0015	Storage coefficient for keeping neighbours
δ	2.5	Path length coefficient
ζ	2.2	Energy consumption coefficient
α_1,α_2,τ	0.1, 0.3, 0.2	Learning rate
ϵ	0.4	Action selection distribution probability
k	20	Learning rounds
γ_1,γ_2,γ_3,γ_4	1.7, 8.2, 0.013, 1.5	Benefit coefficients

4.2. Simulation Results: The Stationary Occasion

Figure 2 demonstrates the performance of the three routing approaches and their multi-agent framework-based enhanced versions in a stationary WSN. The solid lines denote the performance variation of the three routing approaches, while the dashed lines indicate the performance of their multi-agent framework-based counterparts.

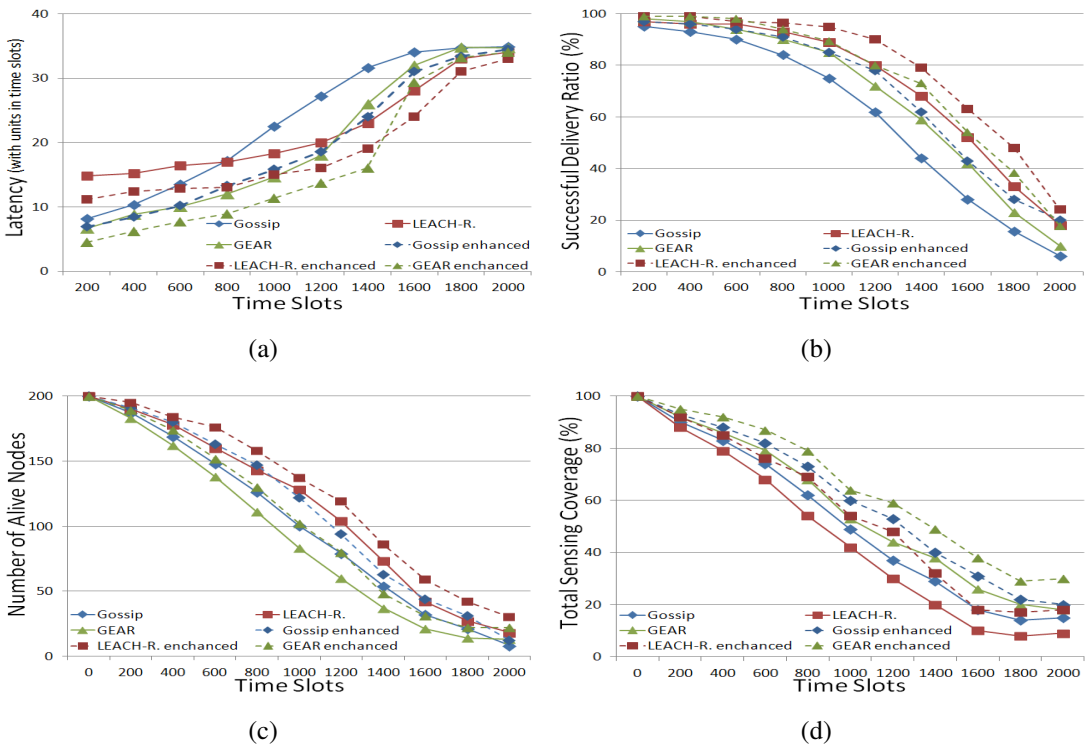

Figure 2. Performance of routing approaches in a stationary WSN. (**a**) Average delivery latency; (**b**) successful delivery ration (%); (**c**) number of live nodes; (**d**) total sensing coverage (%).

In Figure 2a, initially, the delivery latency values in gossiping and GEAR are less than that in LEACH-revised. However, as time progresses, LEACH-revised turns out to be better than both gossiping and GEAR. This can be explained as follows. Initially, in LEACH-revised, the cluster heads usually have to wait for a period for their cluster members to send packets to them and then aggregate and condense these packets to save transmission energy consumption. As time progresses, some sensor nodes are depleted. In gossiping and GEAR, some pre-existing routing paths will disappear, but in LEACH-revised, the impact, caused by sensor node failure, is much less than that in gossiping and GEAR, because routing in LEACH-revised is based on only a few nodes, *i.e.*, cluster heads, and even if these cluster heads are exhausted, some other nodes can take their roles immediately by taking the measure developed in LEACH.

In Figure 2b, LEACH-revised achieved the highest successful delivery ratio. This is due to the fact that packets in LEACH-revised use the fewest hops from a source node to the sink node compared to gossiping and GEAR, because, as described above, routing in LEACH-revised is based only on cluster heads. Thus, it is more reliable for LEACH-revised to successfully transmit a packet to the sink node than both gossiping and GEAR.

In Figure 2c,d, it can be found that in gossiping and LEACH-revised, more sensor nodes can be saved than that in gossiping (Figure 2c), but in GEAR, the total sensing coverage is larger than that in both gossiping and LEACH-revised (Figure 2d). This finding confirms the argument in [21] that saving energy consumption can prolong the lifetime of sensors, but does not necessarily achieve better sensing coverage. This can be explained by the fact that GEAR's transmission method can easily deplete the nodes that are closer to the sink node, but gossiping's transmission method, randomly choosing a neighbour for transmission, can distribute the energy consumption over a larger area than GEAR's method does; so, in gossiping, fewer nodes are depleted compared to GEAR in the same time span. Likewise, in LEACH-revised, routing only relies on cluster heads, and the role of a cluster head is rotated among different sensor nodes. Thus, the lifetime of each sensor node in LEACH-revised could be further prolonged. However, due to the same reason, the total sensing coverage in GEAR is larger than that in gossiping, because gossiping's transmission method consumes energy over a larger area, which implies that depleted sensors are distributed over a larger area; so, the total sensing coverage reduction is more than that in GEAR. Similarly, in LEACH-revised, the energy dissipation in the network is distributed more broadly than it is in gossiping, because in LEACH-revised, sensors are organised based on clusters that are broadly distributed over the network. In this way, when sensors in different clusters are depleted, the total sensing coverage will be decreased more than that in gossiping. Finally, it can be seen that the performance of the three routing approaches is improved by the use of the proposed framework, measured by the four quantitative metrics. This is due to the fact that the framework enables each sensor to find the most useful cooperative neighbours, and therefore, the routing is more targeted. In addition, as described in the previous section, in the framework, each sensor takes into account energy consumption, storage consumption and sensing coverage, when it builds its cooperative neighbour set. Thus, by using the framework, the performance of the three existing routing approaches can be raised in all four quantitative metrics.

4.3. Simulation Results: The Dynamic Occasion

In this simulation, at each time slot, each sensor has the probability, 0.05, to move from its current position to a random position, and at each time slot, a new sensor will arrive at the network with the probability 0.2 at a random position. When an existing sensor moves to a new position or a new sensor joins the network, its physical neighbour set and cooperative neighbour set will be initialised. The node's physical neighbour set is built based on its communication radius. This means that any node in this node's communication radius will become this node's physical neighbours. The node's cooperative neighbour set is initialised in the same way as the physical neighbour set. In practice, the movement of a sensor is usually based on some purpose, so the sensor does not move to a random position. In this simulation, the aim is just to demonstrate how the proposed framework works in a dynamic WSN, so we simplified the movement of a sensor by setting its movement destination to a random position.

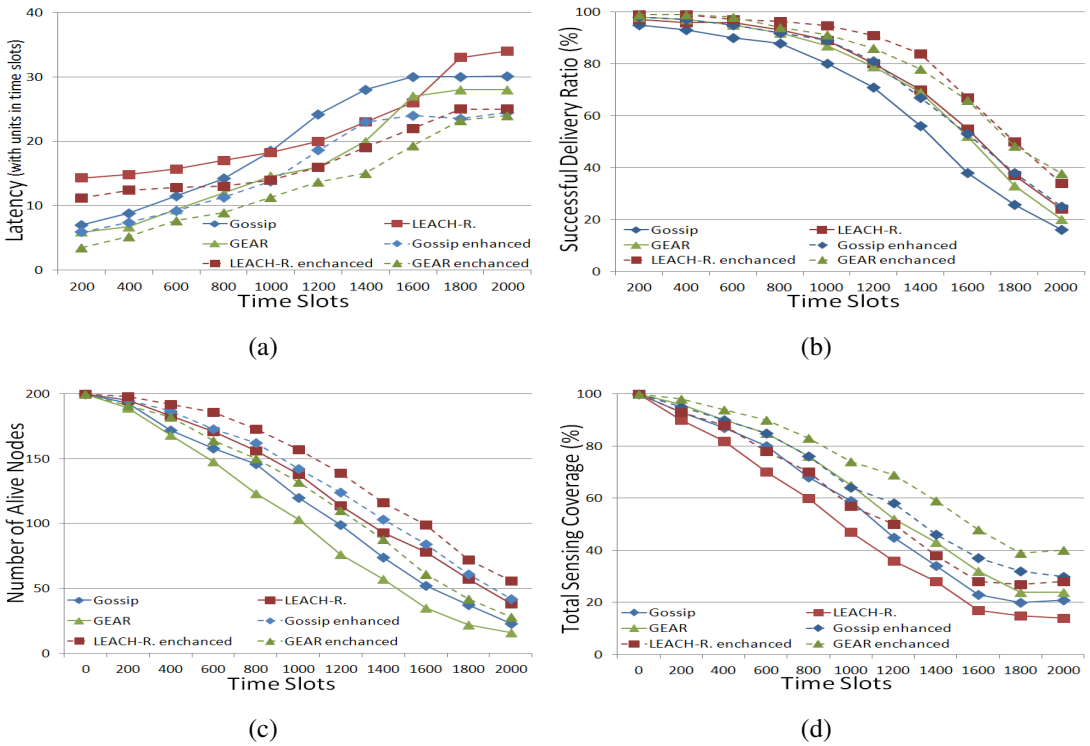

Figure 3. Performance of routing approaches in a dynamic WSN. (**a**) Average delivery latency; (**b**) successful delivery ration (%); (**c**) number of live nodes; (**d**) total sensing coverage (%).

In Figure 3, the main trend of the simulation results is similar to that in a stationary WSN, but the overall performance of the three routing approaches and their multi-agent framework-based enhanced versions is improved compared to that in a stationary WSN. This is because sensors movement can, to some extent, balance the energy consumption in the network. In a stationary WSN, those sensors that are neighbours of the sink nodes are very likely to be depleted, because they are the final hops to the sink nodes and all of the packets have to be relayed by them. Therefore, if sensors can move, those final hop nodes will most likely move away and, likewise, other sensors will most likely be the final hop nodes; so, the energy consumption can be balanced to some extent. Analogously, if new sensors can join the

network, the energy of the whole network increases, which is good for the network in various aspects. There are still some specific phenomena that should be noted. In Figure 3a, the average delivery latency in LEACH-revised in the final time slots is longer (*i.e.*, worse) than that in gossiping and GEAR, which is different from Figure 2a. This is ascribed to the fact that routing in LEACH-revised is based on cluster heads, so new sensors joining or sensor movement cannot effectively shrink the routing path, and thus, the latency of packet routing cannot be improved very much. Due to the same reason, the difference of the final performance, regarding successful delivery ratio (Figure 3b) between LEACH-revised and gossiping is reduced compared to Figure 2b. On the other hand, in Figure 3c, the difference in the number of live nodes in LEACH-revised and gossiping in the final time slots is enlarged compared to Figure 2c, which can reflect the fact that LEACH-revised is quite good at balancing energy-consumption. Finally, again, using the proposed framework can improve the performance of all three routing approaches in a dynamic WSN.

In summary, using the proposed framework, the performance of the selected three routing approaches in both stationary and dynamic WNSs can be improved by around 10%–15% with regard to the four quantitative metrics, which demonstrates the potential usefulness of this framework.

5. Conclusions and Future Work

In this paper, a multi-agent framework was developed, which to the best of our knowledge, is the first one used to assist many existing routing approaches to improve their routing performance. In addition, this framework is decentralised and does not need global information. The simulation results exhibited the effectiveness of this framework in both static and open and dynamic WSNs.

In the future, we first intend to overcome the drawback that an addressing scheme has to be used. Another flaw is that this framework uses a 'unit disk' communication model, where sensors can communicate bi-directionally if they are in each other's communication radius. However, in real-world WSNs, several physical links are uni-directional. Additionally, more existing routing approaches have to be taken into our simulation to further validate the effectiveness of our framework, and it will be very interesting to allow nodes to use different routing approaches in different situations. We are also interested in validating the framework in heterogeneous WSNs, where different sensors have different sensing coverage and initial energy. Finally, when such studies are finished, we intend to test this framework on a specific platform, e.g., NS2 [43], and further in a real environment.

Acknowledgments

This work was originally created at University of Wollongong, Wollongong 2522, Australia. It was partly supported by an ARC Discovery Project (DP150101775) from Australian Research Council, Australia. We also greatly appreciate Mrs. Madeleine Strong Cincotta, who provided professional language editing for us.

Author Contributions

Dayong Ye and Minjie Zhang created the major part of this paper. Yun Yang contributed a lot during the revision process by providing complexity analysis.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Winkler, M.; Tuchs, K.D.; Hughes, K.; Barclay, G. Theoretical and practical aspects of military wireless sensor networks. *J. Telecommun. Inf. Technol.* **2008**, *2*, 37–45.
2. Seema, A.; Reisslein, M. Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a flexi-wvsnp design. *IEEE Commun. Surv. Tutor.* **2011**, *13*, 462–486.
3. Tavli, B.; Bicakci, K.; Zilan, R.; Barcelo-Ordinas, J.M. A survey of visual sensor network platforms. *Multimed. Tools Appl.* **2012**, *60*, 689–726.
4. Akkaya, K.; Younis, M. A survey on routing protocols for wireless sensor networks. *Ad Hoc Netw.* **2005**, *3*, 325–349.
5. Li, C.; Zhang, H.; Hao, B.; Li, J. A Survey on Routing Protocols for Large-Scale Wireless Sensor Networks. *Sensors* **2011**, *11*, 3498–3526.
6. Anisi, M.H.; Abdullah, A.H.; Razak, S.A.; Ngadi, M.A. An Overview of Data Routing Approaches for Wireless Sensor Networks. *Sensors* **2012**, *12*, 3964–3996.
7. Al-Karaki, J.N.; Kamal, A.E. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wirel. Commun.* **2004**, *11*, 6–28.
8. Sohrabi, K.; Gao, J.; Ailawadhi, V.; Pottie, G.J. Protocols for Self-organization of a Wireless Sensor Network. *IEEE Pers. Commun.* **2000**, *7*, 16–27.
9. Polastre, J.; Hui, J.; Levis, P.; Zhao, J.; Culler, D.; Shenker, S.; Stoica, I. A unifying link abstraction for wireless sensor networks. In Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems, San Diego, CA, USA, 2–4 November 2005; pp. 76–89.
10. Tsai, M.; Yang, H.; Liu, B.; Huang, W. Virtual-Coordinate-Based Delivery-Guaranteed Routing Protocol in Wireless Sensor Networks. *IEEE/ACM Trans. Netw.* **2009**, *17*, 1228–1241.
11. Hedetniemi, S.; Liestman, A. A Survey of gossiping and broadcasting in communication networks. *Networks* **1988**, *18*, 319–349.
12. Heinzelman, W.; 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.
13. Kulik, J.; Heinzelman, W.R.; Balakrishnan, H. Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks. *Wirel. Netw.* **2002**, *8*, 169–185.
14. Rogers, A.; David, E.; Jennings, N.R. Self-organized routing for wireless microsensor networks. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2005**, *35*, 349–359.
15. Beraldi, R.; Baldoni, R.; Prakash, R. A biased random walk routing protocol for wireless sensor networks: The lukewarm potato protocol. *IEEE Trans. Mob. Comput.* **2010**, *9*, 1649–1661.
16. Quang, P.T.A.; Kim, D.S. Enhancing Real-Time Delivery of Gradient Routing for Industrial Wireless Sensor Networks. *IEEE Trans. Ind. Electron.* **2012**, *8*, 61–68.

17. Li, Y.; Chen, C.S.; Song, Y.Q.; Wang, Z.; Sun, Y. Enhancing realtime delivery in wireless sensor networks with two-hop information. *IEEE Trans. Ind. Electron.* **2009**, *5*, 113–122.
18. Niu, J.; Cheng, L.; Gu, Y.; Shu, L.; Das, S.K. R3E: Reliable Reactive Routing Enhancement for Wireless Sensor Networks. *IEEE Trans. Ind. Inform.* **2014**, *10*, 784–794.
19. Heinzelman, W.; Chandrakasan, A.; Balakrishnan, H. Enerty efficient communication protocol for wireless sensor networks. In Proceedings of the 33rd Hawaii International Conference System Sciences, Maui, HI, USA, 4–7 January 2000; pp. 3005–3014.
20. Lindsey, S.; Raghavendra, C. PEGASIS: Power-Efficient Gathering in Sensor Information Systems. *IEEE Aerosp. Conf. Proc.* **2002**, *3*, 1125–1130.
21. Tsai, Y.R. Coverage-Preserving Routing Protocol for Randomly Distributed Wireless Sensor Networks. *IEEE Trans. Wirel. Commun.* **2007**, *6*, 1240–1245.
22. Valentini, G.; Abbas, C.J.B.; Villalba, L.J.G.; Astorga, L. Dynamic Multi-Objective Routing Algorithm: A Multi-Objective Routing Algorithm for the Simple Hybrid Routing Protocol on Wireless Sensor Networks. *IET Commun.* **2010**, *4*, 1732–1741.
23. Villalba, L.J.G.; Orozco, A.L.S.; Cabrera, A.T.; Abbas, C.J.B. Routing protocols in wireless sensor networks. *Sensors* **2009**, *9*, 8399–8421.
24. Mottola, L.; Picco, G.P. MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2011**, *10*, 1694–1709.
25. Xu, Y.; Heidemann, J.; Estrin, D. Geography-informed energy conservation for ad hoc routing. In Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Rome, Italy, 16–21 July 2001; pp. 70–84.
26. Yu, Y.; Estrin, D.; Govindan, R. *Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks*; Technical Report; UCLA Computer Science Department: Los Angeles, CA, USA, 2001.
27. H.Huang.; Hu, G.; Yu, F.; Zhang, Z. Energy-aware interference-sensitive geographic routing in wireless sensor networks. *IET Commun.* **2011**, *5*, 2692–2702.
28. Awad, A.; German, R.; Dressler, F. Exploiting Virtual Coordinates for Improved Routing Performance in Sensor Networks. *IEEE Trans. Mob. Comput.* **2011**, *10*, 1214–1226.
29. Petrioli, C.; Nati, M.; Casari, P.; Zorzi, M.; Basagni, S. ALBA-R: Load-Balancing Geographic Routing Around Connectivity Holes in Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 529–539.
30. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998.
31. Iglesias, A.; Martnez, P.; Aler, R.; Fernndez, F. Learning teaching strategies in an Adaptive and Intelligent Educational System through Reinforcement Learning. *Artif. Intell.* **2009**, *31*, 89–106.
32. Harandi, M.T.; Ahmadabadi, M.N.; Araabi, B.N. Optimal Local Basis: A Reinforcement Learning Approach for Face Recognition. *Int.J. Comput. Vis.* **2008**, *81*, 191–204.
33. Gomes, E.R.; Kowalczyk, R. Dynamic Analysis of Multiagent Q-Learning with E-Greedy Exploration. In Proceedings of the ICML'09, Montreal, QC, Canada, June 2009.
34. Galstyan, A.; Czajkowski, K.; Lerman, K. Resource allocation in the grid using reinforcement learning. In Proceedings of the AAMAS'04, New York, NY, USA, July 2004; pp. 1314–1315.

35. Gomes, E.R.; Kowalczyk, R. Learning the IPA Market with Individual and Social Rewards. In Proceedings of the International Conference on Intelligent Agent Technology (IAT'07), Fremont, CA, USA, 2–5 November 2007; pp. 328–334.
36. Ziogos, N.P.; Tellidou, A.C.; Gountis, V.P.; Bakirtzis, A.G. A reinforcement learning algorithm for market participants in FTR auctions. In Proceedings of the 7th IEEE Power Tech Conference, Lausanne, Switzerland, 1–5 July 2007; pp. 943–948.
37. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement Learning: A Survey. *J. AI Res.* **1996**, *4*, 237–285.
38. Zhang, C.; Lesser, V.; Shenoy, P. A Multi-Agent Learning Approach to Online Distributed Resource Allocation. In Proceedings of the IJCAI'09, Pasadena, CA, USA, July 2009; pp. 361–366.
39. Singh, S.P.; Jaakkola, T.; Littman, M.L.; Szepesvari, C. Convergence Results for Single-Step On-Policy Reinforcement Learning Algorithms. *Mach. Learn.* **2000**, *38*, 287–308.
40. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422.
41. Shah, R.; Rabaey, J.M. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. *Proc. IEEE Wirel. Commun. Netw. Conf.* **2002**, *1*, 350–355.
42. Yu, M.; Leung, K.K.; Malvankar, A. A Dynamic Clustering and Energy Efficient Routing Technique for Sensor Neworks. *IEEE Trans. Wirel. Commun.* **2007**, *6*, 3069–3079.
43. The Network Simulator. Available online: http://www.isi.edu/nsnam/ns/ (accessed on 28 April 2015).

© 2015 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).