東海大學資訊工程學系研究所

碩士論文

指導教授：呂芳懌 博士

Dr. Fang-Yie Leu

Cluster-RLM:植基於檔集之 RLM 在無線感測網路中建立繞境路徑以節省用電量

Cluster-RLM : Establishing a Routing Path with Cluster-based Redundant Link Minimization in Wireless Sensor Networks for Energy Saving

研究生：莊勝傑

中　華　民　國　一　〇　五　年　一　月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 莊 勝 傑 所提之論文

Cluster-RLM:植基於欖集之 RLM 在無線感測網路中建立繞境路徑以節省用電量

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 陳金鈴 簽章

委 員

指 導 教 授 簽章

中華民國 105 年 1 月 14 日

Outline

Figure of content:

Table of content:

中文摘要

在分散式非結構化網路中，經常使用洪水演算法(Flooding)來建立 routing path。但此演算法執行時會產生大量的冗餘訊息，浪費網路資源與能量。因此，學者專家提出了 RLM 及 RLM-T 演算法來減少這一些冗餘訊息的產生，但此二演算法仍會產生許多冗餘訊息及無法保證選取最佳化路徑的問題。於是本研究提出一個兩者的改進版本，稱為植基於叢集之 RLM 之繞境路徑建立方式(簡稱 C-RLM)，它是在一個基於事件觸發之無線感測器網路中，以 RLM 建立網路拓樸後，為每一節點 N 計算其鄰居指標 Δ，Δ 之定義為 N 和其每一直接鄰居節點擁有共同鄰居節點數量的總和。其後，由 Δ 值最大者開始分類每一節點，一個節點 P 若無其他相鄰節點之 Δ 值比 P 之 Δ 值大，則 P 為 Head-node (簡稱 H-node)，一個 H-node 周遭之相鄰節點為 Member-nodes (簡稱 M-nodes)，這一些 M-nodes 和該 H-node 共同組成一個 cluster。其後，一個 cluster 中，保留 M-node 和 H-node 之連線，而隱藏 M-node 與 M-node 之間的現有連線，使成為一個以 H-node 為核心之星狀網路拓撲。接著，任兩 cluster 之間若有 K 條連線，K1，則保留其中一條 Δ 值加總最小之連線，而隱藏其他者，目的在減少一個節點和 BS 建立路由路徑(routing path)時所發送及接收之 RREQ 封包。和洪水演算法相比，本方法可以有效地減少了無線感測器網路建立路由路徑時，所發送之多餘封包，且有效地降低了所耗費之電量、封包遺失率及點對點之封包傳送延遲時間，也提高了封包傳送之 throughputs。

關鍵字: 無線感測網路、洪水演算法、RLM、RLM-T、LEACH。

Abstract

In a wireless sensor network, the flooding algorithm is often invoked to establish a routing path between a sensor node and the sink (base station). But this algorithm generates a lot of redundant packets, thus wasting unnecessary network resources and packet delivery energy. Researchers have then presented at least two solutions, Redundant Link Minimization (RLM for short) and Redundant Link Minimization-Triangle (RLM-T for short), to mitigate the mentioned phenomena. But both algorithms still have their own problems, e.g., many redundant packets still exist, meaning that the established routing-path can be further optimized. Therefore, in this paper, we propose an improved version, called the Cluster-based Redundant Link Minimization scheme (C-RLM for short), with which an event-driven wireless sensor network (EDWSN for short) can build a spanning tree with the sink as its root. Then each node *N* has a path through which *N* can deliver the packets it generates to the sink. So when *N* defects an event, it does not need to issue a routing request packet to establish a routing path. In the C-RLM, Δ of a node *N*, denoted by Δ_N, is defined as the cumulative number of common neighbor nodes between *N* and each of *N*'s direct neighbors. After calculating Δ_N, the node with the highest Δ value is classified as a head node, denoted by H-node. This H-node's all direct neighbors are its member nodes, denoted by M-nodes. These M-nodes and the H-node are grouped together as a cluster. We then mark all of them as classified nodes. The unmarked node with the highest Δ value is again an H-node. All its unmarked direct neighbor nodes are its M-nodes, and these M-nodes and the H-node form another cluster, and all of them are then marked. This procedure repeats until all nodes are marked and classified into

either H-nodes or M-nodes. After that, in a cluster, the link between two neighbor M-nodes is hidden. If there are at least two paths connecting two adjacent clusters, we keep the path with the lowest $\sum\Delta$, and hide the remaining paths. The purpose is reducing energy consumption during packet delivery. Experimental results show that this method can effectively improve an EDWSN's throughputs, shorten its end-to-end delays, reduce its packet drop rates, and lower its packet delivery energy consumption.

Keyword: WSN, Flooding, LEACH, RLM, RLM-T

1.Introduction

In recent years, wireless sensor networks (WSNs) have been a part of our everyday lives. Those mechanisms that surround us, like the ones controlling temperature for air conditioners and refrigerators, and those turning on or off dehumidifiers, are all operated by sensors. Originally, sensors were developed by military and sprinkled to the battlefields to collect enemy information. Now, you can see sensors everywhere in different industrial and business domains, e.g., detecting defective products in a factory's production line to make sure the products' qualities [1], or monitoring health condition of a remote patient [2]. Some sensors on the other hand are used to monitor whether there is a fire event or not [3], and the concentration of carbon monoxide and carbon dioxide existing in our soundings [4]. The purpose is giving users the information of whether our environmental quality is good or not. On 31st July 2014, there was an underground-culvert gas explosion at Kaohsiung, Taiwan. If environmental sensors had been installed in the underground-culvert, it would be an alarm to warn people so as to avoid the explosion accident.

A WSN often consists of many sensors, a wireless data collector (i.e., the sink or called the base station) and its controlling system [5, 6]. The former senses environmental data or changes, the middle collects the sensed data, and the latter follows the system settings and parameters to turn on or off sensors or the WSN [7]. Recently, the design principles of sensors are mainly on saving electricity, lowering price, reducing their size and so on [8]. In the following, we use sensor, sensor node and node interchangeably.

In a wireless sensor network, we usually use flooding algorithm [9] to establish a routing path between a node, e.g., node *N*, and Base Station (BS). Due to the feature of flooding algorithm,

when a sensor, e.g., node *h*, senses the environmental change or data, it broadcasts an RREQ packet [10]. If *h* has *n* direct neighbors, each neighbor will receive the RREQ packet and resend it, implying that *h* will receive a total of *n* times of this RREQ packet from its *n* neighbors. In fact, *n*-1 times of which are useless. This on the contrary wastes energy for packet receiving.

When the flooding algorithm is invoked, in order to reduce a large number of packet delivery after a RREQ packet is sent, Yu [11] in 2008 proposed a scheme, named Redundant Link Minimization (RLM for short), which classifies and clusters nodes to reduce packet delivery. However, in the RLM, many R-nodes cannot be effectively clustered, thus making a sensing field can not be completely monitored. In 2010, Kao[12] proposed the Redundant Link Minimization –Triangle Optimization (RLM-T for short) to improve this problem. But the RLM-T doed not ensure choosing the best routing path for a sensor. Therefore, in this study, we propose an improving version of them, named Cluster-based Redundent Link Mininization (C-RLM for short) scheme. After establishing a network topology, we calculate an index I for node *N*, denoted by Δ_N which is defined as the cumulative number of common neighbors between *N* and each of its direct neighbors. After calculating Δ_N, we sorting all the nodes on their in an ascending order. The node with the highest I is an H-node, and its direct neighbors are M-nodes. These nodes are grouped together as a cluster. Among the remaining nodes, the one with the highest Δ_I is again an H-node. All its unclustered direct neighbors are classified as its M-nodes. These nodes are then grouped as a new cluster. This process repeats until all nodes are classified and clustered. The C-RLM utilizes an intra-cluster optimization algorithm to hide the link between two arbitrary M-nodes in the same cluster. Next, the C-RLM employs an inter-cluster optimization algorithm to choose a best path between two adjacent clusters and hide the remaining paths between the two clusters.

At last, the C-RLM treats a cluster as a big node, and invokes the breadth-first graph traversal algorithm to reduce the topology of big nodes to a spanning tree, aiming to decrease unnecessary data packet delivery and receiving, thus lowering energy wasting. Our simulation results show that C-RLM can effectively improve a WSN's throughputs, shorten its end-to-end delays and reduce its packet loss rates and packet delivery energy consumption compared with those when RLM and RLM-T are individually used.

The rest of this paper is organized as follows. Chapter 2 introduces the background and reviews related literature of this study. Chapter 3 presents the RLM and RLM-T. Chapter 4 describes the C-RLM. Experiments and their results are analyzed and discussed in Chapter 5. Chapter 6 concludes this paper and outlines our future studies.

2. Background and Related Work

2.1 Flooding algorithm

Flooding algorithm is one of the static routing schemes, in which a route-request packet is first broadcast by a source node, e.g., node *N*. If one of *N*'s neighbor nodes, e.g., node *K*, has ever received this route request packet, on receiving this packet, *K* throws it away. Otherwise, it adds its node ID to the packet and broadcasts this packet to its neighbors. The process repeats until BS receives the first route request packet, e.g., RREQ-*f*. Then BS backwards an Route Acknowledgement (RACK) packet to *N* via the reverse direction of the arriving path of RREQ-*f*.

One of the advantages of the scheme is reliable since if BS is reachable from *N*, at least one RREQ packet will arrive at BS. But this scheme has some problems, such as after the RREQ packet is transmitted, lots of duplicated packets will be generated and flown toward BS. As mentioned above, if a node *N* has *n* direct neighbor nodes, it will receive the route request packet *n* times. But only the first received packet is effective. In other words, other *n*-1 packets received will be thrown away, wasting (*n*-1)*e_r of receiving energy where e_r is the energy consumed by a node for receiving a packet. If the routing path *P* after established consists of m nodes, excluding the BS, the wasted receiving energy will be

where n_i is the number of neighbor nodes of node *i*. But if the multicast approach is used to transmit the RREQ packet, each node has to record its upstream and downstream nodes. When it would like to transmit an RREQ packet, it will check the neighbor-node table to see to which destination nodes it needs to send the packet. In fact, if

we construct a spanning tree with the shortest radius, no more RREQ packet needs to be sent before a data packet can be transmitted to BS.

2.2 Hierarchical routing

In a Hierarchical Routing scheme[13-15], all sensing nodes are grouped into clusters. Each cluster, having its own cluster head, builds a hierarchical route between its head node and BS. All intra-cluster sensors send the data they sense directly to their cluster head. After aggregating the data sent by its sensors, the head node sends the aggregated data to BS directly or through a multi-hop path. In fact, hierarchical routing can not only reduce the packet sending time, but also shorten the distance between sensing nodes and BS. As a result, it can reduce the energy consumed by sensing nodes effectively, particularly saving energy for delivering data packets to BS.

2.3 Flat routing

In a WSN, each sensor node plays the same role, i.e., collecting environmental data or changes, and relaying packets when necessary. The system initially receives a request packet issued by users through BS, and BS then checks to see whether the system owns the data or not. If yes, the data will be sensed by sensors and then sent back to the BS.
However several sensors may own or sense the data at the same time. They will transmit all the data to BS. This may cause network congestion, meaning this kind of routing is unsuitable for an extensive network. The advantage of flat routing [16, 17] is that it does not need to save too much routing data for sensors, particularly for those relay node on a routing path.

2.4 Location-based routing

Location-based Routing [18, 19] is a routing approach in which each sensor needs to install a global positioning system (GPS for short) so that a sensor can know its own location or others' locations. With these locations, a sensor network can find the best routing path to deliver data packets. Comparing it with broadcast, this routing approach will not produce unnecessary packets and consume energy to find a routing path. But it can be applied to different networks. Geographic Adapive Fidelity (GAF) [20] is a typical example. But the price is high since a GPS is required by each sensor.

3. The RLM and RLM-T

Notations and Terminology used in this study are listed and defined in Table 1.

Table 1 Notations and terminology used in this study.

Notation	Terminology
i	node i
	A link between node i and node j
	A path between nodes i and j with h intermediate nodes denoted by $\{k\}$, $h \geq 1$
$NS(i)$	The set of direct neighbors of node i
d	The distance defined as the number of links between two nodes
Δ_i	Cumulative numbers of common neighbors between node i and each of node i's direct neighbors
ClusterID(i)	The Cluster ID is i
NodeType(i)	The node i's type
NH(i)	A node i which is directly connected to j H-nodes, denoted by NH(i)={H_1, H_2,…,}, $H_i \in$ NS(i), $1 \leq i \leq j$, $2 \leq j \leq$ |NS(i)|+1
Dis (H_t, BS)	The distance between H_t to BS

3.1 Redundant link minimization

Yu [11] in 2008 proposed the RLM to cluster nodes in a network and hide the redundant links to the BS to simplify the network. In a multicast system, it can effectively decrease unnecessary delivered packet and the delivery energy.

The RLM assumes that all nodes in the network have the same processing capability and the same link bandwidth. The transmission time between two nodes is propotional to the distance between them. A node maintains two tables. One is the neighbor-node table which keeps the node's direct neighbors. The other, named multicast table, records which nodes should receive the following packets if multicast rather than broadcast is employed. Also, the live time of a packet is unlimited.

The RLM algorithm has four steps, including Δ calculation, clustering, intra-cluster optimization and inter-cluster optimization.

3.1.1 Δ calculation

Δ values are calculated as follows. Node *N*'s Δ, denoted by Δ_N, $\Delta_N = \sum_{j=1}^{K} CN_{Nj}$, where *K* is the number of *N*'s direct neighbors, and CN_{Nj} is the number of common direct neighbors of node *N* and *N*'s direct neighbor *j*, $1 \leq j \leq k$. In Figure 1, node 5's neighbor set contains nodes 4 and 6. Nodes 4 and 5 have a common neighbor, i.e., node 6. So Δ_5 is increased by one (the initial value is zero). Nodes 6 and 5 have a common neighbor, i.e., node 4. So Δ_5 is increased by one again. In other words, when three nodes form a triangle, the Δs of the three nodes will be individually increased by two.

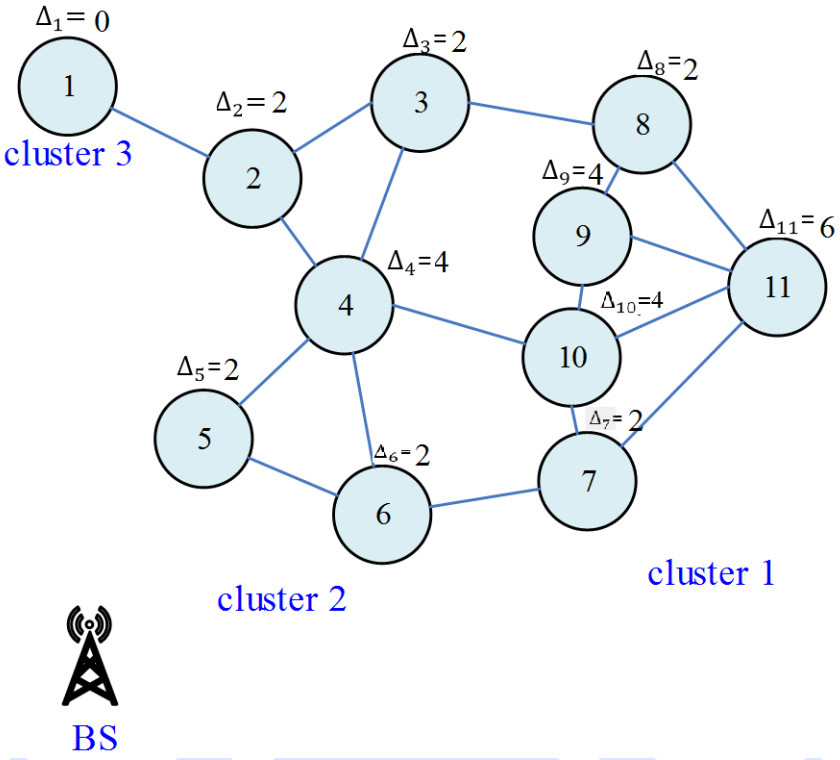

Figure 1After calculating the Δ values for a given topology with 11 nodes.

3.1.2 Clustering

When clustering nodes, if node *N*'s Δ, i.e., Δ_N, is the highest among all its direct neighbors and *N* itself, then *N* will be an H-node. Therefore in Figure 1, node 11 is an H-node. Node 11's direct neighbor nodes, i.e., nodes 7, 8, 9 and 10, are M-nodes which together with node 11 compose a cluster, denoted by cluster 1. After that, Δ_4 is the biggest among node 4 and its direct neighbors. So it is an H-node. Nodes 2, 3, 5 and 6 are its M-nodes. These nodes form another cluster, called cluster 2. The remainder, i.e., node 1, is an R-node. The result is shown in Figure 2.

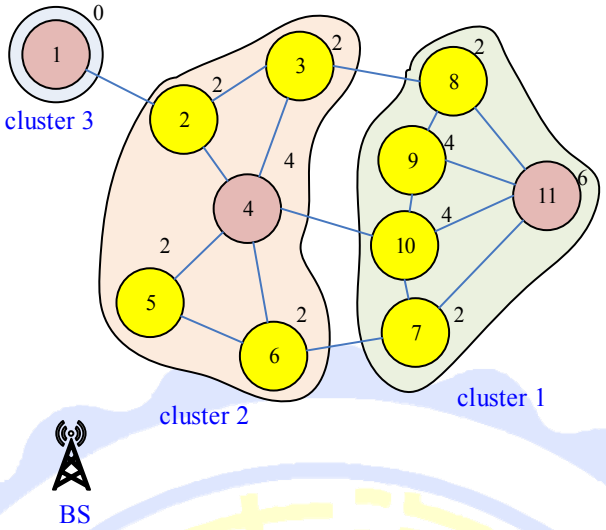

Figure 2 After classifying and clustering all nodes in the WSN shown in Figure 1.

3.1.3 Intra-cluster optimization and Inter-cluster optimization

In the intra-cluster optimization process, two links in cluster 2, i.e., the link between nodes 2 and 3 (denoted by) and that between nodes 5 and 6 (i.e.,) are hidden. In cluster 1, three links, i.e., , and , are hidden. The result is illustrated in Figure 3.

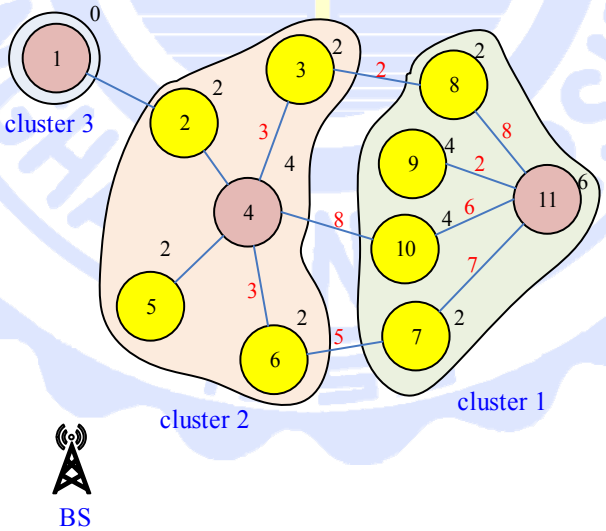

Figure 3 After the intra-cluster optimization for all clusters.

Further, there are three paths between clusters 1 and 2, i.e., with the sum of link weights of 13 (=3+2+8), with that of 14 (=8+6), and with that of 15 (=3+5+7). Of course, the sum of link weights of is the smallest. So the inter-cluster optimization will hide the links of the

other two paths. If some links, e.g., L_s of the is co-link with the remaining two paths, when hiding the links of the two paths, L_s will not be hidden.

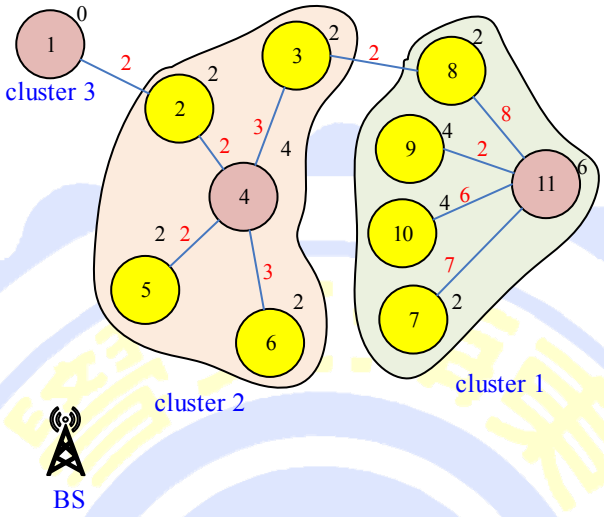

Figure 4 After the inter-cluster optimization on the inter-cluster paths of all pairs of adjacent clusters.

3.1.4 Node joint and leaving

The RLM considers the situations of node joint and leaving since a node may newly join the WSN or die, respectively.

A newly joining node is classified as follows.

Case 1: If at least one of the direct neighbors of a newly joining node *N* is H-node, then *N* will be an M-node of one of these H-nodes, even Δ_N is bigger than Δ of the H-node.

Case2: If *N*'s all direct neighbors are M-nodes, then *N* is an R-node and it will be a cluster by itself.

Case3: If *N*'s all direct neighbors are R-nodes, *N* is an H-node, its direct neighbors are M-nodes, and they form a cluster.

The process for the leaving of a node is below.

Case1: If the leaving node *N* is an H-node, *N* will notify its M-nodes, and dismiss all nodes in the cluster. The entire RLM process will be applied to re-organize these nodes.

Case2: If *N* is an M-node in cluster *C*, *N*'s direct neighbors belonging to cluster *C* will notify their H-node to delete *N* from *N*'s all direct neighbor's neighbor sets.

Case3: If *N* is an R-node, before leaving, *N* will notify each of *N*'s direct neighbor nodes, e.g., *Q*, to delete *N* from *Q*'s neighbor set.

3.1.5 The problems of the RLM

The RLM's problems are as follows.

(1). In the RLM, the relationship between two arbitrary nodes is peer to peer. So it does not consider a node's communication range. In this study, what we face is WSNs. So a node's communication range is a parameter needed to be dealt with.

(2). During classification [12], the RLM does not classify R-nodes, leaving each of them alone.

(3). In the inter-cluster optimization process, the distance between two nodes is proportional to the time required to deliver a packet between the two nodes. Practically, it is an infeasible approach since packets may be dropped due to beffer overflow. The delay time will be long or infinitive. Also, electromagnetic waves are transmitted very fast. It is hard for us to discriminate the difference of the individual timings among different pairs of neighbor-node transmission.

(4). The RLM cannot avoid the case in which after clustering, some nodes still have a route cycle among clusters (see Figure 5). If a route cycle exists, it means that at least three nodes, belonging to different clusters, are directly connected to each other. When a packet *P* is sent by one of them, *P* may be circulated onceamong them. This will consume unnecessary resources and energy.

(5). If *N* dies, immediately, in cases 1 and 3, it may not have chance to notify other node of its death. Its neighbor node is suitable to do this.

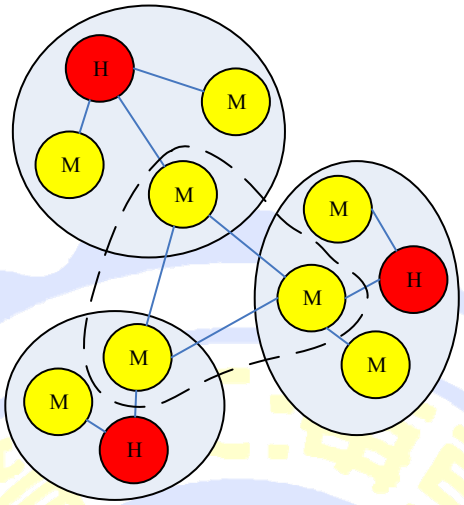

Figure 5 A cycle route among three clusters

3.2 Redundant Link Minimization - Triangle Optimization

The RLM-T improves the RLM by using Triangle optimization scheme to solve the problem of high threshold (see case 2 of newly-joining node classification) for changing an R-node to an H-node, and removes route cycles among clusters.

3.2.1 R-cluster Optimization

The method that the RLM-T solves the second problem of the RLM, called R-cluster optimization [12], is as follows.

A. For an R-node, e.g., *x*, the RLM-T compares Δ_x with those of *x*'s direct-neighbor R-nodes, and assigns the one with the highest , e.g., node *h*, as the H-node. However, if there are *k* R-nodes all with the Δ_1, *k*>1, the RLM-T chooses the one with the highest

degree, e.g., *d*, as the H-node. If there *m* nodes all with the degree of *d*, *m*>1, then RLM-T chooses the one with the lowest ID as the H-node. All the H-node's direct-neighbor R-nodes are M-nodes. Then these nodes form a new cluster, named R-cluster.

B. After an R-node is changed to an H-node, the network topology varies. So the intra-cluster optimization algorithm is invoked to hide the link between two arbitrary M-nodes.

C. Invoking the inter-cluster optimization algorithm to choose the best path between the new cluster with each of its adjacent clusters.

3.3.2 Triangle Optimization Algorithm

To avoid generating a route cycles among *k* adjacent clusters, *k*=3, the RLM-T employs the following Triangle Optimization which has four steps.

(1). In a cluster, a node transmits its own neighbor-node table (*T*) to its direct neighbor nodes.

(2). When receiving a neighbor node table (T_Q) sent by one of its direct neighbor, e.g., *Q*, node *N* checks to see whether in T_Q, there is a node *L* which is also one of *N*'s direct neighbors. If yes, meaning that each of *L*, *N* and *Q* is a common neighbor of the other two nodes, then *N* transmits a LOCK request to *Q* and *L*.

(3). When *Q* (or *L*) receivies the Lock request, it establishes the link between *Q* (or *L*) and *N*, and marks *L* (or *Q*) in its own neighbor-node table as temporarily deleting *L* (or *Q*). The purpose is to hide the link connecting *L* and *Q*.

(4). After that, both *Q* and *L* send a Unlock request to *N* to finish the algorithm.

3.3.3 The problems of the RLM-T

The problems of the RLM-T are as follows.

(1). An R-node can be itself a cluster. So R-cluster optimization is not required.

(2). Assume that nodes A, B and C form a route cycle. Let A be the starting node. Then the link will be hidden. If we choose B (or C) as the starting node, the hidden link is (or). That means the choice of the starting node will affect the resulting topology. In other words, this algorithm cannot ensure the generation of the best path for a cycle route among adjacent clusters.

(3). If there are p nodes in q adjacent clusters form a cycle route, $p>3$, and $p \geq q \geq 3$, the Triangle Optimization algorithm cannot identify it.

4. Cluster-based Redundant Link Minimization (C-RLM)

In this study, to solve the mentioned problems of the RLM and RLM-T, the C-RLM is proposed under the assumption that each node in the WSN knows the direction and position of the BS, has the same computation capability and the same network bandwith, and consumes the same amount of energy for transmitting (receiving) a packet.

The C-RLM has five steps, including Δ calculation, classification and clustering, Intra-cluster optimization, Inter-cluster optimization and establishing a spanning tree. Δ calculation, as the steps of the RLM and RLM-T, first calculates Δ values for all nodes in a given network topology. Classification and clustering will cluster all nodes into groups based on some rules. Intra-cluster optimization, as the steps of the RLM and RLM-T, removes the links connecting two M-nodes in a cluster. Inter-cluster optimization removes the links of a redundant path connecting two neighbor clusters. The final step, establishing a spanning tree, first shrinks a cluster into a big node, and then invokes a graph traversal algorithm [21, 22] to simplify the big-node topology into a spanning tree.

4.1 Parameters

In this study, five tables are created to record required information. The first, named the Node Information Table, is established to save nodes' information. The table has 8 fileds, including Node ID, Δ value, Degree, NeighborSet, NodeType, ClusterID, ClusterHead and ConnectingClusters, NodeID represents a node, e.g., node *N*'s ID, Δ is node *N*'s Δ value, Degree is the number of *N*'s direct neighbor nodes, NeighborSet records all direct neighbor nodes of *N*, NodeType is *N*'s node type, e.g., M-node or H-node, ClusterID is the identity of

node *N*'s cluster, and ClusterHead is the head node of the cluster to which *N* belongs, and ConnectingClusters field indicates the clusters that *N* connects on an inter-cluster path. For example, *N* connects clusters *i* and *j*. Then this field will be given C_{ij}. Of course, if a node, e.g., *Q*, in cluster *i* is not on an inter-cluster path, the field will be filled in C_i. But if a cluster, e.g., cluster *i*, has *k* neighbor clusters, the field will have *k* C_{ij}s where j is one of *i*'s neighbor clusters. Table A-1 shown in the Appendix of this thesis lists the Node Information Table established for the network topology shown in Figure 1.

Given a network topology, we first transform the topology into an adjacent matrix[23]. The set of *N*'s direct neighbor nodes is represented by NS(*N*)={*k*| *k* is a node in *N*'s NeighborSet field}, i.e., Neighbor Set of *N*.

Table 2 Node Information Table.

NodeID		Degree	NeighborSet	NodeType	ClusterID	ClusterHead	ConnectingClusters
1	0	1	2	H-node	3	1	
3	2	3	2, 6, 8	M-node	2	4	
5	2	2	4, 6	M-node	2	4	C_2

These second table is Intra-hidden-link Table which as shown in Table 3 is used to keep track of which link between two M-nodes is hidden during intra-cluster optimization. We call the link M-link. When a node dies, all hidden links need to be recovered so that we can re-intra-optimize the new topology. In this table, there are two fields, including ClusterID and HiddenLink, in which the HiddenLink = means that the link originally connects nodes *i* and *j*. Note that the table is sorted on ClusterID field in an accending order.

Table 3 Intra-hidden-link Table.

ClusterID	HiddenLink
2	
2	

We create another table, named Inter-hidden-link Table, which records the links hidden during the inter-cluster optimization. There are four fields, including Path, HiddenLink, TotalΔValue

and ConnectingClusters. In the ConnectingClusters field, a value, e.g., , indicates that the path, denoted by shown in the Path field, connects the two adjacent clusters *h* and *t* and node *i* (node *j*) belongs to cluster *h* (cluster *t*). {*q*} means there are *p* nodes between nodes *i* and *j* where *p* can be 1 or 2. We will derscribe this later. The HiddenLink field saves the links along the path which are hidden during the inter-cluster optimization. The TotalΔValue field keeps the cumulative Δ value for all nodes along . When a node dies, like that of the Intra-hidden-link Table, some related hidden links also need to be recovered for further processing.

Table 4 Inter-hidden-link Table

Path	HiddenLink	TotalΔValue	ConnectingClusters
		14	
		14	

The fourth table, called Big-node Inter-hidden-link Table, has the same relation schema as the one of Table 4, which will be described later.

4.2 Δ Calculation

Based on the Node Information Table, the C-RLM calculates Δ_N by counting the number of common nodes between *N* and *Q*, $Q \in$ NS(*N*), i.e., Δ_N is defined as

$$\Delta_N = \qquad . \qquad (1)$$

Figure 6 shows the Δ calculation algorithm, i.e., Algorithm 1.

Algorithm 1: Δ calculation

Input: Node Information Table

Output: Δ values of all nodes

{For (*i*=1;*i*$\leq$ *n*;*i*++){

/* *n*: the number of nodes in the given network topology*/

= 0;

For (each *j*, *j* $\in$ NS(*i*))

Δ_i = Δ_i +|NS(*i*)NS(*j*)| ; } /*cumulating node's Δ_i */

```
Sort the Node Information Table on    values;}
```

Figure 6 The algorithm that calculates Δ for all nodes in a WSN

After calculating Δ values for all nodes, the C-RLM sorts them on Δ in a descending order. Table A-2 illustrated in the Appendix of this thesis shows the result after sorting Table A-1. Currently the NodeType, ClusterID, ClusterHead and ConnectingClusters fields are all null, because nodes have not been classified and clustered.

4.3 Classification and Clustering

Now the node with the highest Δ value, e.g., node *N*, is classified as an H-node. All the nodes in NS(*N*) are M-nodes which together with *N* form a cluster, and the NodeType, ClusterID, and ClusterHead of these nodes as marked nodes are then filled in the table with the corresponding values, meaning they have been classified and clustered. Their ConnectingClusters fields are still empty. Next, among the unmarked nodes, the C-RLM chooses the node with the highest Δ, e.g., *P*, as an H-node and all unmarked nodes in NS(*P*) are *P*'s M-nodes. Again these nodes as marked nodes comprise another cluster and their corresponding values will be filled in the Node Information Table. The procedure repeats until all nodes are marked. Figure 7 lists Algorithm 2, named node classification and clustering algorithm.

```
Algirithm 2:Node classification and clustering
   Input: Node Information Table T
   Output: Node Information Table with all nodes classified and clustered, i.e., marked
{U=T ;
CID=1;       /*cluster ID*/
While (at least one node in U whose NodeType is null (i.e., unmarked)){
   Assume that  I   of node i is the highest among all  I   of unmarked nodes)
   If (there are  k
_28D28Dd28282828282828282828¥ĝ2228282828282828282828282828282828I ,
>1)
      If (in US, there are h nodes with the highest degree, k≥  h>1)
            Choose the one with the smallest NodeID, e.g., node i, from the h nodes;
      else    /*k>1 and h=1*/
            Choose the node with the highest degree, e.g., node;
   else /*k=1*/
         Choose node i as the H-node;
    NodeType (i) =H-node ;    /* node i is marked*/
    ClusterID (i) = CID ;
```

```
ClusterHead(i)=i ;
For (each j, j ∈ NS(i), and NodeType (j) is null, i.e., unmarked)
   {NodeType (j) =M-node;            /*classifying node j*/
    ClusterID (j) =CID;              /*clustering node j*/
    ClusterHead (j) = i;}
CID++; }}
```

Figure 7 The algorithm for node classification and clustering.

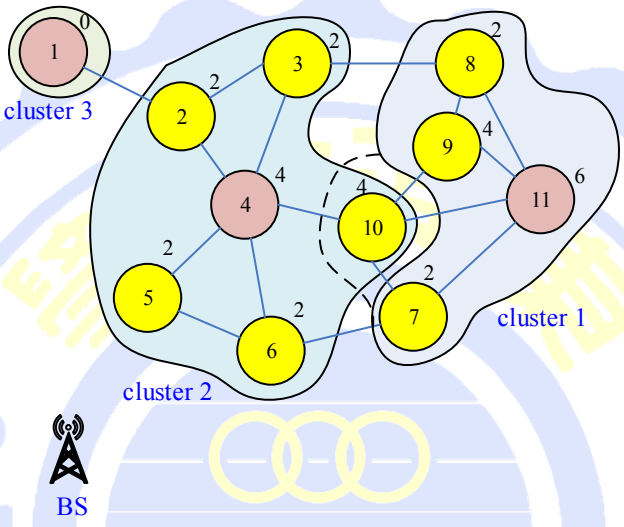

Figure 8 The topology after node classification and clustering

Choosing the node with the highest Δ value as an H-node may have three cases.

(1). If there is only one node with the highest Δ value, i.e., k=1 and h=1, choose this node as an H-node.

(2). If there are k nodes with the highest Δ value, k>1, choose the one with the highest degree.

(3). If there are h nodes with the same highest Δ value and the same highest degree, $k \geq h$>1, then choose the one with the smallest node ID as the H-node.

Assume node i is chosen as the H-node, and node j is an unmarked node, $j \in$ NS(i) and NodeType (j)=null, then j's is an M-node, ClusterID=CID and ClusterHead(j)=i. The result produced after applying Algorithms 1 and 2 to the figure shown in Figure 1 is illustrated in Figure 8, in which node 10 belongs to cluster 1 (see the dashed line).

After node classification and clustering, it is possible that an M-node, e.g., Q, is directly

connected to two H-nodes, e.g., H_i and H_j. Node 10 in Figure 8 is an example. In this case, based on Algorithm 2, *Q* will belong to the cluster, e.g., cluster *m*, with the H-node of higher Δ, e.g., H_i, since a node belonging to a cluster where H-node has a higher Δ value is classified before a node belonging to another cluster, e.g., clusters *r*, where H-node (e.g., H_j) has a lower Δ value is done. However, if the distance between H_i and BS, denoted by *Dis*(H_j, BS), is shorter than *Dis*(H_i, BS), it would be better to move *Q* from cluster *m* to cluster *r*. The purpose is to shorten the distance between *Q* and BS and the cumulative distance between all nodes and BS, defined as

(2)

where *n* is the number of nodes in the concerned network, and . So after reclustering, node *Q* belongs to cluster *m*, rather than cluster *r*, to reduce energy consumption for delivering those packets sent to BS by *Q*. Figure 9 lists the algorithm, i.e., Algorithm 3, named node re-clustering. Table A-3 shown in Appendix of this thesis illustrates the Node Information Table after node re-clustering.

Algorithm 3: Node re-clustering

Input: an M-node *Q* which is directly connected to *q* H-nodes, denoted by NH(*Q*)={H_1, H_2,…,}, $H_i \in$ _30D30Dd

/*if *q*=1, then Q has only one neighbor H-node. In this case, reclustering is not required.*/

Output :clustering *Q* to the cluster with H_j as its H-node where *Dis* (H_j, BS)==$\{_1 \le i \le q^{min}$ *Dis* (H_i, BS), $H_i, H_j \in$ ###_

{Let *R* = $|\{_1 \le i \le q^{min}$ *Dis* (H_i, BS)|;

If (*R*==1) Choose the only H-node, e.g., H_j, as *Q*'s H-node;

/*only one H-node with the shortest distance away from BS*/

else /**R*>1*/

Choose the H-node with the smallest NodeID, e.g., H_f, NH(Q), and ($1 \le i \le q^{min}$ *Dis*(H_i, BS) as the new H-node of Q;

If (ClusterID(Q)ClusterID(H_f)){

ClusterID(Q)=ClusterID(H_f); /* re-clustering*/

ClusterHead(Q) =H_f;}}

Figure 9 The algorithm for re-clustering nodes.

In Figure 8, node 2 (node 10) directly connects two H-nodes, i.e., nodes 1 and 4 (nodes 4 and 11). After invoking Algorithm 3, node 2 does not change the cluster to which it belongs. But node 10 is re-clustered to cluster 2 from cluster 1 (ignoring the dashed line) since *Dis*(node11, BS) > *Dis*(node4, BS).

4.4 Intra-cluster optimization

The Intra-cluster optimization algorithm (i.e., Algorithm 4) of the C-RLM is shown in Figure 10. After applying this algorithm to the figure shown in Figure 8 (of course, node 10 has been moved to cluster 2), the resulting figure is illustrated in Figure 11. The records generated in this step for intra-hidden links are listed in Table 5. In cluster 1, is hidden, and in cluster 2, and are hidden.

Algorithm 4 : Intra-cluster optimization

Input: Node Information Table T

Output: Node Information Table after being intra-cluster-optimized

{For (each cluster, e.g., cluster i, $1 \le i \le C$)

/*C is the number of clusters in the concerned network topology */

For (each pair of M-nodes, e.g., node P and node Q, in cluster i)

If ($Q \in$ NS(P){ /* checking to see whether P and Q are neighbor nodes or not) */

Insert (*i*,) as a new record in the Intra-hidden-link Table;

NS(*Q*)=NS(*Q*)-{*P*}; /*update Node Information Table*/

NS(*P*)=NS(*P*)-{*Q*};}}

Figure 10 The algorithm for Intra-cluster optimization.

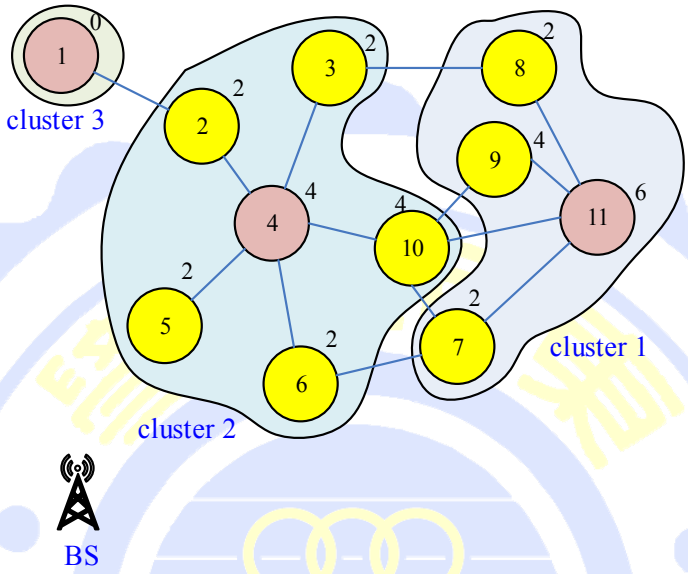

Figure 11 The resulting network topology after applying Intra-cluster optimization algorithm listed in Figure 10 to the figure shown in Figure 8 (node 10 has been moved to cluster 2)

Table 5 Intra-hidden-Link Table.

ClusterID	HiddenLink
1	
2	
2	

4.5 Inter-cluster optimization

If there are *u* paths connecting two neighbor clusters, in order to reduce the energy for delivering a packet from an upstream cluster to a downstream cluster, we choose the path with the smallest cumulative Δ value, e.g., $\sum \Delta_i$, as the inter-cluster routing path, hiding other

paths and recording links of the hidden paths in the Inter-hidden-link Table. The reason of choosing such a path is that a group of nodes with smaller $\sum \Delta_i$ consumes less energy than a group of nodes with higher $\sum \Delta_j$ does, *ij*, since less number of nodes will receive the packet. But if there are *k* paths, *k*1, with the same $\sum \Delta_i$ value, the C-RLM selects the one with the least number of nodes, e.g., *m* nodes, since we can then save message transmission and receiving energy. However, among them if there are *h* paths with the least $\sum$, $2 \leq h \leq k$, each of which consists of *m* nodes, the C-RLM randomly chooses one from the *h* paths.

Algorithm 5: Inter-cluster optimization
Input: The intra-cluster-optimized network topology
Output :The inter-cluster-optimized network topology with the best path between each pair of neighbor clusters
{For (each pair of neighbor clusters, e.g., cluster *i* with H_i as its H-node and cluster *j* with H_j as its H-node, are connected by K_{ij} paths, denoted by path(*i*, *j*) = {$\mathbf{path_1}$, $\mathbf{path_2}$, … $\mathbf{path_{K_{ij}}}$}, $K_{ij}>1$){
Let L(*i*, *j*) be the paths with , and $k=|L(i,j)| \leq K_{ij}$;
/* *q* is H_i, H_j or other node on $\mathbf{path_p}$ */
If (*k*== 1) =L(*i*, *j*); /*only one path with the least $\sum$ */
else /* *k* >1*/
{Let *h* be the number of paths with the least number of nodes, e.g., *m* nodes, $h \leq k$;
If (*h* > 1)
Randomly choose a path as the from the *h* paths;

else /*h=1*/

Choose the only path with m nodes as ;}

For (each path, e.g., $path_s$, $path_t$ ∈ _34D34Dd34343434 }, $1 \leq t \leq K_{ij}$)

$L_{P,R} \in path_s$

For(each link , connecting, e.g., nodes P and R, , and)

/*check to see whether is not a co-link of $path_t$ _*/

{ NS(R) = NS(R)-{P};

NS(P) = NS(P)- {R};

/*hiding of $path_t$ by deleting P from NS(Q) and deleting Q from NS(P) in the Node Information Table*/

Insert the record ($path_s$, , $\sum'$ and C_{ij}) to the Inter-hidden-link Table where is the cumulative value of the nodes on $path_s$; }}}

Figure 12 The algorithm for Inter-cluster optimization

Table 6 lists the inter hidden-links recorded in the Inter-hidden-link Table after applying Algorithm 5 (see Figure 12) to the figure shown in Figure 11. Figure 13 illustrates the result topology.

Table 6 Records generated in the Inter-hidden-link Table after applying Algorithm 5 to the figure shown in Figure 11.

Path	HiddenLink	TotalΔValue	ConnectingClusters
		14	
		14	
		16	
		18	

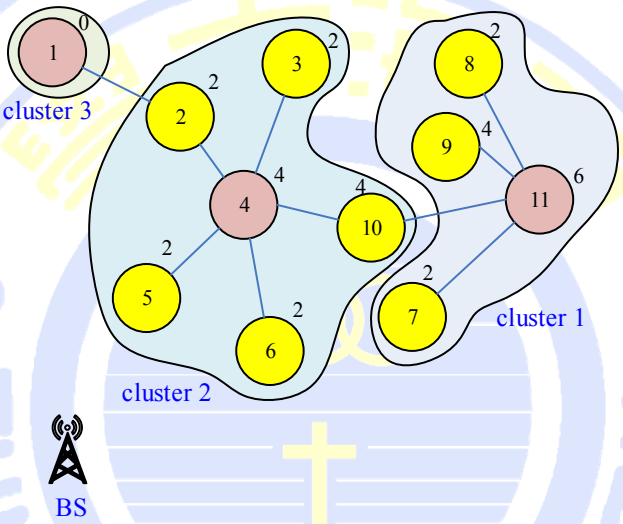

Figure 13 After applying Algorithm 5, i.e., Inter-cluster optimization algorithm, to the figure shown in Figure 11

When a node, e.g., *Q*, dies or leaves the sensing field, as mentioned above, we must recover the hidden links in the related clusters, including those in the cluster, e.g., DNC, to which *Q* belongs and those in the direct neighbor clusters of DNC, e.g., NCs, if *Q* is on the links connecting DNC and at least a member of NCs.

4.6 Cluster node

In this study, to avoid the route cycle among clusters, we shrink a cluster to a big node. If originally the topology is consisted of *C* clusters. The result topology will comprize *C* big nodes. We choose the big node whose H-node is the closest to BS, i.e., the one with $(1 \le f \le C^{min} Dis(H_f, BS)$, as the H-node, denoted by H_i, of the big-node topology.

Figure 14 illustrates an example of a big-node topology shrinked from its original figure (not

shown). Then, the Breadth-first graph traversal algorithm [21, 22] is invoked, starting at H_i, to generate a spanning tree which is one with the shortest radius where the radius of a tree is defined as the path length between H_i (root) and the farest big-node/clusters from H_i. All remaining inter-cluster links are hidden. Among them, a link on the path, e.g., $path_w$, connecting two big nodes, i.e., clusters *i* and *j*, has two link possibilities. The first is that *P* and *R* are two M-nodes, in which ClusterID(*P*)= and ClusterID(*R*)= or vice versa. In this case $Path_w$ in their original network topology is in which or , which contains four nodes, the two H-nodes at both ends of this path. This is the case in which $\{q\}$ in mentioned in section 4.1 includes two nodes, i.e., *P* and *R*. The second case is that one of them, e.g., *P* (or *R*) is an H-node of a cluster, e.g., cluster *i*, and the other, i.e., *R* (or *P*), is an M-node belonging to cluster *j*, i.e., $Path_w$ = or including three nodes, i.e., $\{q\}$ includes only one node. A record ($Path_w$, , ,) is then inserted into the Big-node Inter-hidden-link Table, the schema of which is the same as that of Inter-hidden-link Table (see Table 4). Like those links in the Intra-hidden-link Table and the Inter-hidden-link Table, the links will be recovered when necessary. Figure 15 shows the result, in which big node 1 is H_i. Figure 16 lists the algorithm, named Establishing a spanning tree (Algorithm 6).

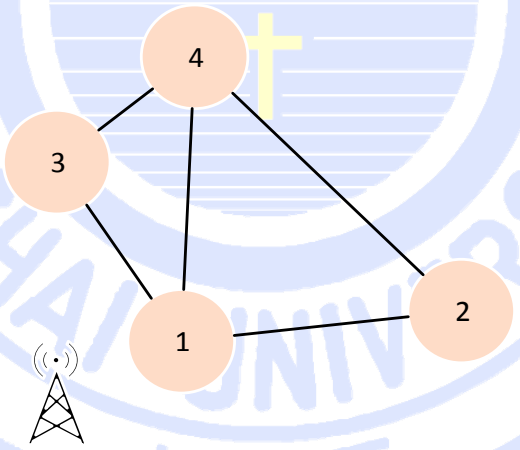

Figure 14 All clusters are shrinked to big nodes.

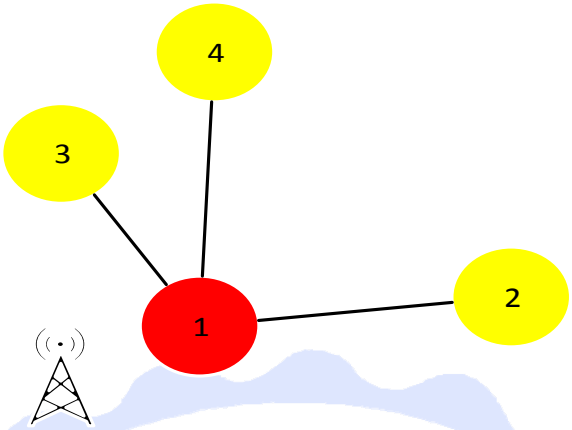

Figure 15 Establishing a spanning tree and hiding the remaining inter-cluster links.

Algorithm 6: Establishing a spanning tree

Input: The network topology *T* after inter-cluster optimization

Output: A spanning tree

{Each cluster in *T* is shrinked to a big node;

Choose the big node with $1 \le j \le C^{min} Dis(H_j, BS)$ as the H-node, denoted by H_i;

Invoke the breadth-first graph traversal algorithm which starts at H_i to generate a spanning tree;

Hide all other inter-cluster links;

Record the hidden links in the Big-node Inter-hidden-link Table;}

Figure 16 The algorithm for establishing a spanning tree for the big-node topology.

4.7 Node joint and leaving

In a wireless sensor network, a node may die or newly join the network. Of course, we can , like that in the RLM and RLM-T, use the method defined previously to re-process the topology. But this will consume a lot of energy. The atternative is that, we only re-process the affected clusters. In this study, we choose the latter.

4.7.1 Node joint

Assume that after a new node, node *N*, joins a network, it has *m* direct neighbors, i.e., NS(*N*)={Q_1 , Q_2 , …Q_m }. Then for each $Q_j \in$ NS(*N*), NS(Q_j)= NS(Q_j) ∩ {*N*}, $1 \leq j \leq m$. If there are *k* H-nodes in NS(*N*), denoted by NSH(*N*)={NSH_1, NSH_2, … NSH_k }, $k \leq m$. To simplify the following expressions, we assume that ClusterID(NSH_j)=*j*.

(1). If $k \geq 1$, the C-RLM chooses the H-node the closest to BS, e.g., NSH_i , i.e., *Dis*(NSH_i , BS)= , NSH_i , $NSH_1 \in$ NSH(*N*). Let ClusterID(*N*)=*i*, meaning *N* joins cluster *i* to be an M-node. Node 12 shown in Figure 17 is an example. Now for each NSH_v , $NSH_v \in$ NSH(*N*), excluding NSH_i , NS(NSH_v)=NS(NSH_v)-{*N*} since needs to be hidden, and then a record, e.g., (, , ∩ +∩ +∩ ,) for each of these H-node, e.g., NSH_v , $NSH_v \in$ NSH(*N*)-{NSH_i }, is inserted into the Inter-hidden-link Table. So a total of *k*-1 records are inserted. Also, NS(*N*)= NS(*N*)-(NSH(*N*)-{NSH_i }) which removes *k*-1 nodes from NS(*N*) in the Node Information Table. Among the remaining *m*-*k*+1 links, "1" means the link between *N* and NSH_i and *m*-*k* are connected to *N* from *m*-*k* M-nodes. Some are intra-cluster and some are inter-cluster connections. Let LN(*N*) = {Q_1 , Q_2 , … Q_{i_1}} be the nodes connecting to *N* where ClusterID(*N*)= ClusterID(Q_j),

$Q_j \in$ LN(*N*), $1 \leq j \leq l_1$, i.e., l_1 intra-cluster M-links (excluding the one between NSH_1 and *Q*), $l_1 \leq$ *m-k*. Now we need to hide the l_1 M-links in ClusterID(*N*), i.e., NS(*N*)=NS(*N*)-{Q_j}, and NS(Q_j)=NS(Q_j)-{*N*}, for all $Q_j \in$ LN(*N*). Also, for each Q_j, a record (*i*,) will be inserted into the Intra-hidden-link Table. A total of l_1 records will be inserted.

Now we hide Inter-cluster M-links due the joint of *N*. Let $l_2 = m\text{-}k\text{-}l_1$, and let NN(*N*)={$P_1$, P_2, …P_{l_2}} be the l_2 nodes connecting to *N*, of course from M-nodes in other clusters, where ClusterID(*N*)ClusterID((P_k), in which P_k is an M-node of ClusterID(P_k); then NS(P_k)=NS(P_k)-{*N*}, and NS(*N*)= NS(*N*)-{P_k}, for all $P_k \in$ NN(*N*), $1 \leq k \leq l_2$. Let H_k be the H-node of ClusterID(P_k), e.g., cluster *j*. o a total of l_2 records, denoted by (, l +l +l +l ,)s, will be inserted into the Inter-hidden-link Table. In Figure 17, the new node, i.e., node 12, is ultimately an M-node of cluster 3.

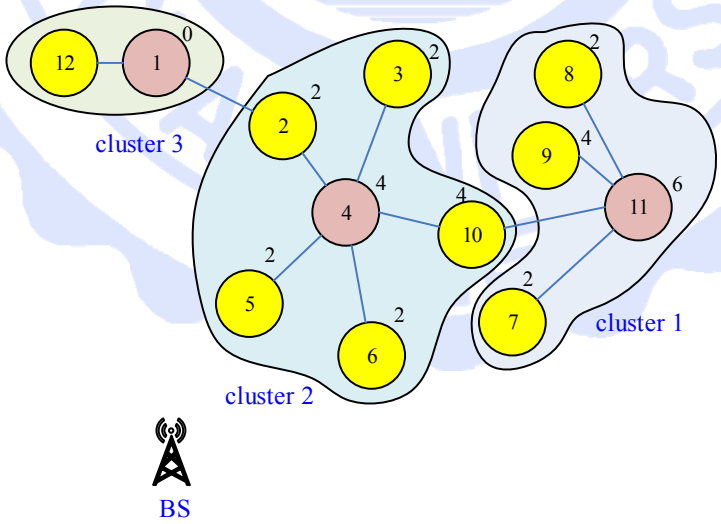

Figure 17 A node, e.g., node 12, newly joins the network as an M-node since one of its neighbors is an H-node.

(2). If *k*=0, it means that none of *N*'s direct neighbors is an H-node. Assume that all nodes in

NS(*N*) belong to *h* clusters, which are HH(*N*)={C_1, C_2,…C_l}, $m \geq h$. The C-RLM now recovers all those hidden links originally connected to at least one of the nodes in NS(*N*) based on the contents of the Intra-hidden-link Table, Inter-hidden-link Table and Big-node Inter-hidden-link Table (the three together are called three hidden-link tables), and then creates a new cluster, e.g., *NC*, in which *N* serves as the H-node. For each node *P*, $P \in$ NS(*N*), if *Dis*(ClusterHead(*P*), BS)> *Dis*(N, BS), *P* is then re-clustered to *NC* by invoking Algorithm 3. For each cluster *i*, $i \in$ HH(*N*) {*NC*}, the Intra-cluster optimization algorithm is invoked to hide the M-links in *i*. After that, for each pair of clusters in HH(*N*) $\cup$ {*NC*}, e.g., *i* and *j*, the Inter-cluster optimization algorithm is utilized to hide inter-cluster paths between them. Of course, the two algorithms will insert corresponding records to the three hidden-link tables. In Figure18, node 13 after newly joins the figure shown in Figure 17 has only one neighbor node, i.e., node 8. So itself as an H-node forms a cluster, i.e., cluster 4. Since Dis(node 11, BS) $\leq$ Dis(node 13, BS), node 8 remains in cluster 1.

(3). If *N* has no direct neighbors, it means that it cannot transmit sensed data to BS. That is, it is an unreachable node from BS. This case is beyond the scope of our study.

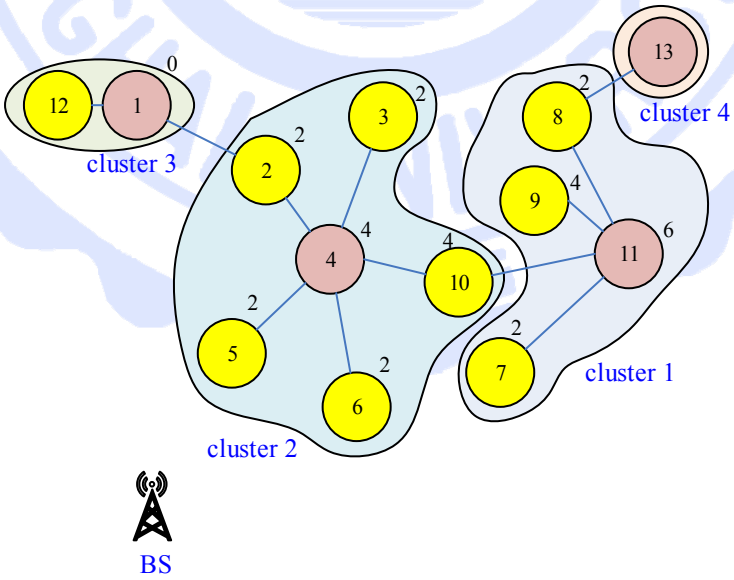

Figure 18 A node, e.g., node 13, newly joins the network as an H-node since none of its neighbors is an H-node, and it has only one neighbor node, i.e., node 8.

4.7.2 Node leaving

A leaving node *N* may be an H-node or M-node. We need to first delete all its related records in the four established tables. Figure19 lists the algorithm, named Deleting records for a leaving node (Algorithm 7).

Algorithm 7: Deleting records for a leaving node.

Input: A leaving node *N* and the four established tables

Output: the new network topology

{Duplicate the record of *N* in Node Information Table to a buffer-record R_N for later use;

Delete the record, of which NodeID=*N*, from Node Information Table;

For (each node *Q*, $Q \in$ NS(*N*))

NS(*Q*) = NS(*Q*)-{*N*}; /*a neighbor node *Q* removes *N* from its neighbor set NS(*Q*) */

For(each record, e.g., *V*, with *V*.ClusterID = ClusterID(*N*) in the Intra-hidden-link Table)

If(*V*.HiddenLink = or)

Delete *V* from the table; /* or will disappear due to the leaving of *N*/

For(each record, e.g., *T*, in the Inter-hidden-link Table)

If(*T*.HiddenLink= or)

Delete *T* from the table;

For(each record, e.g., *U*, in the Big-node Inter-hidden-link Table)

If(*U*.HiddenLink = or)

Delete *U* from the table;}

Figure 19 Deleting records from the four established tables for a leaving node.

Assume that *N* belongs to cluster i, and *i* has *q* neighbor clusters NC(*i*)={ NC_1, NC_2,… NC_q }. Hence, we need to recover all the intra-hidden links, of which the ClusterID=*i*, and all the inter-hidden links that are on the path connecting cluster *i* and NC_r,

$NC_r \in$ NC(i), $1 \leq r \leq q$.

(1). N is an H-node

If N is an H-node, the C-RLM chooses the node with the highest Δ value in cluster i, e.g., node Q, as the new H-node. But some nodes may be out of the communication range of Q, meaning that the remaining nodes in i may be classified into h clusters, $h \geq 1$. Now we need to recover all hidden links recorded in the three hidden-link tables. Figure 20 shows the algorithm, named Link recovery Algorithm, i.e., Algorithm 8.

Algorithm 8: Link recovery

Input : The buffer-record R_N of the leaving node N and the four established tables

Output : Node Information Table with the recovered intra-hidden links in i(=ClusterID(N)), and the inter-hidden links connecting i and one of its neighbor clusters.

{For(each record, e.g., U, with U.ClusterID==i under the assumption that

U.HiddenLink= or $L_{\downarrow}(R \mid F))$ _###) _###) 4242424242

{ /* ClusterID(N)=i, and recover the intra-hidden links originally in cluster i, PN and RN since N has been deleted when Algorithm 7 was invoked*/

NS(P)=NS(P){R}; /*In Node Information Table*/

NS(R)=NS(R){P};

Delete U from the Intra-hidden-link Table;}

For(each record, e.g., V, in the Inter-hidden-link Table)

If (V.ConnectingClusters = or has , $j \in$ _###) D42Dd42424242424242 or where V.ClusterID(P)==j and V.ClusterID(R)==i or V.ClusterID(R)==j and V.ClusterID(P)==i, and NC(i) is the neighbor cluster of cluster i)

{ /*recover inter-cluster links originally connecting i and j, $j \in$ 42

NS(P)=NS(P){R};

NS(R)=NS(R) | {P};

Delete V from the Inter-hidden-link Table;}

For(each record, e.g., W, in the Big-node Inter-hidden-link Table)

If (W.ConnectingClusters==, $j \in$ _###) _###) 42D42Dd4242424242424242424242´Ď2Ž2424242or where W.ClusterID(P)==j and W.ClusterID(R)==i or W.ClusterID(R)==j and W.ClusterID(P)=i)

{ /*recover inter-cluster-hidden links recorded in the Big-node Inter-hidden-link Table*/

NS(P)=NS(P)$\{R\}$;

NS(R)=NS(R) $\{P\}$;

Delete W from the Big-node Inter-hidden-link Table;}

Figure 20 The algorithm for recovering hidden links when the leaving node N is an H-node.

After that, the C-RLM invokes node classification and the clustering algorithm, i.e., Algorithm 2, to classify and cluster all nodes in cluster i. Assume that a total of r clusters SC(i)={SC_1, SC_2,…, SC_r} are generated from the remaining nodes in i, excluding the leaving node N. The reclustering algorithm, Algorithm 3, is then called to recluster nodes in each pair of clusters in NC(i)SC(i). After that, the Intra-cluster optimization algorithm (Algorithm 4) and the Inter-cluster optimization algorithm (Algorithm 5) will be sequentially performed to hide the intra-links in each of the cluster in NC(i)SC(i), and inter-cluster links in each pair of the clusters in NC(C_N) SC(C_N). Of course, these hidden links will be recorded in the Intra-hidden link Table and Inter-hidden link Table. The final step is establishing the spanning tree for big nodes by using Algorithm 6.

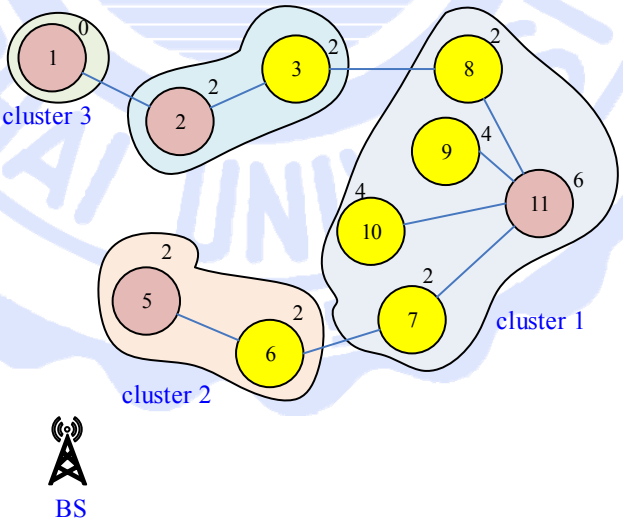

Figure 21 An H-node dies.

(2). N is an M-node

If the leaving node N is an M-node, there are two cases, i.e., N is or is not on the inter-cluster transmission path.

A. If N is not on the transmission path, Algorithm 7 is then invoked, i.e., for each $R \in$ NS(N), NS(R)=NS(R)-{N}, and the record of N is also deleted from the Node Information Table. Also, the records concerning the links connected to N in the Intra-hidden-link Table and in the Inter-hidden-link Table are all deleted. For example, if in the Intra-hidden-link Table (Inter-hidden-link-Table), there is a record, e.g., R, in which R.HiddenLink = or , R will be deleted from the table RN, Figure 22 shows an example, in which node 3 leaves the network.

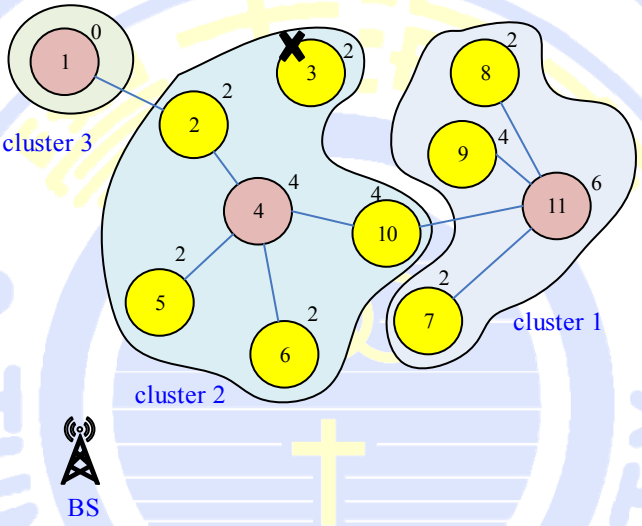

Figure 22 The leaving node is an M-node, which is not on any transmission path between two clusters.

B. If N is on the link l shared by q paths, denoted by path(i, X)={$path_1^i$, $path_2^i$,…,}, for each $path_x^i$, $path_y^i \in$ path(i, X), that connects cluster i and, e.g., cluster r where cluster i is the cluster to which N belongs, then the C-RLM invokes Algorithm 8 to recover all Intra-hidden links for cluster i by looking up the Intra-hidden-link Table, and inter-hidden-links that originally connect cluster i and r by checking the Inter-hidden-link Table. Of course, the records concerning the recovered links in the two tables will be deleted. Note that in Algorithm 7 which was performed prior to this stage, N was deleted from the Node Information Table, and all the links in i and j connected to N were also deleted. After that, we invoke

the intra-cluster optimization algorithm to hide intra-hidden links for cluster *i* and *j*. For each neighbor cluster *r*, assume that there are *k* remaining paths between *i* and *j*, path(*i*, *j*)={$path_1$, $path_2$, …$path_k$ }, $k \geq 1$. Then inter-cluster optimization algorithm is invoked to choose the best path between cluster *i* and *j* and hide links of the remaining paths. After that, Algorithm 6 is invoked to establish a spanning tree. Figure 23 is an example, in which node 10 dies. The new inter-cluster path between clusters 1 and 2 is the one going through nodes 6 and 7 since in original topology (see Figure 11), the of and the of are the same and the least.

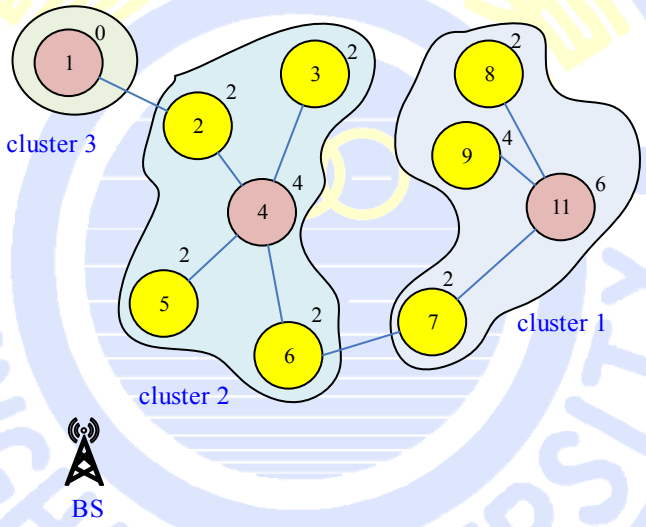

Figure 23 The leaving node is an M-node which is on the transmission path between two clusters.

5. Simulation Analysis

In this study, three schemes, including C-RLM, RLM and RLM-T, are tested, and four metrics are eraluated, including power comsumption defined as the energy consumed per second by all nodes, throughputs defined as the cumulative data size received per second by the sink, end-to-end delays defined as the time period from when a packet is sent by its source node to the time point when the sink receives the packet, and packet drop rates defined as the number of packets dropped on the way to the receiving node over the number of packets sent by source nodes Four experiments were performed. The first evaluated the four metrics given different packet sizes. The second, third and fouth experiments redid the first one given different packet rates, different numbers of nodes and different number of events, respectively. These experiments were simulations by using NS2 [24] as the simulation tool. The specifications of the network are shown in the Table 7. The default parameters for each experiment may be changed when necessary.

Table 7 The parameters and specifications used in the following experiments.

Parameters	Value
MAC layer protocol	IEEE 802.11
Number of sink node (BS)	1
Experimental field	150*150 m^2
Max bandwidth of a link	250Kbps
Number of nodes	30
Packet rate of a source node	10 pkts/sec
Init_energy (J)	100
Number of events occurs for each experiment	5
Energy consumption of packet transmission	1.075 (mJ/packet) on 1KB/pkt
Energy consumption of packet receiving	0.0512 (mJ/packet) on 1KB/pkt
Packet size	1KB

5.1 Different Packet Sizes

In the first experiment, different packet sizes ranging from 1KB to 5KB were given. Figure 24 shows the energy consumption. Due to transmitting packets though the best path, the C-RLM consumed the least energy among the three tested schemes. Generally, a higher packet size indicates higher energy consumption. This is why the three curves go up on larger packet sizes. Also, the energy consumption is linearly propotional to packet sizes. Particularly, the default bandwidth of a link is 250Kbps.

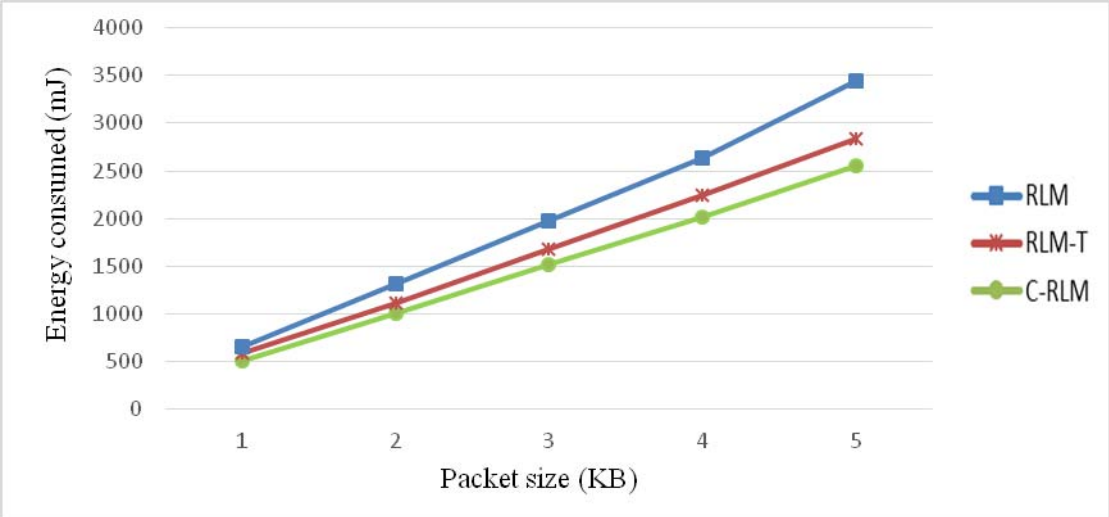

Figure 24 The total energy consumption of the tested schemes on packet rate = 10pkts/sec given different packet sizes.

Figures 25-27 illustrate the throughputs, packet drop rates and end-to-end delays, respectively. In each of the figures, the C-RLM outperforms the other two. The key reason is that for a node *N*, the path to BS is the shortest. The RLM may have route cycles, and both the RLM and RLM-T do not select the best path for delivering packets from *N* to BS. More detailedly, under the same packet rate, longer packets will cause higher throughputs since many data is delivered in a unit of time. So the three curves shown in Figure 25 increase on higher packet sizes. But when link bandwidth gradually saturates, the throughputs are approaching flat. Also, a longer packet will also result in a higher drop rate since the probabilities of transmission error and packets being dropped by downstream nodes increase.This is the reason why the three curves illustrated in Figure 26 rise when the bandwidths of all links of the paths are the same.On the other hand, when a routing path is longer, the drop rate generally will be higher. This is the reason why C-RLM has lower drop rates. Nevertheless, a longer packet needs a longer time to be delivered. Hence the three schemes' end-to-end delays are longer on longer

packets.

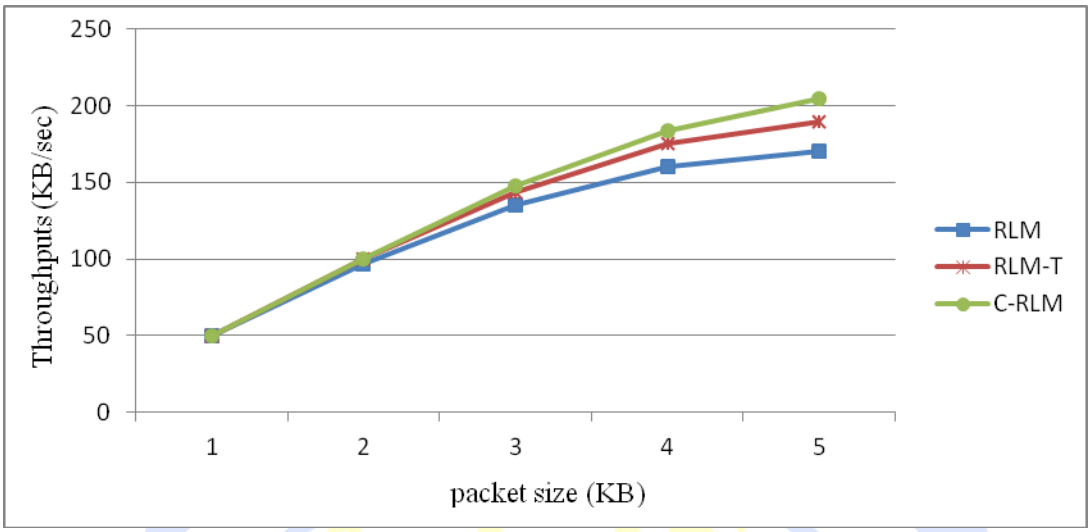

Figure 25 Throughputs at the sink on packet rate = 10 pkts/sec.

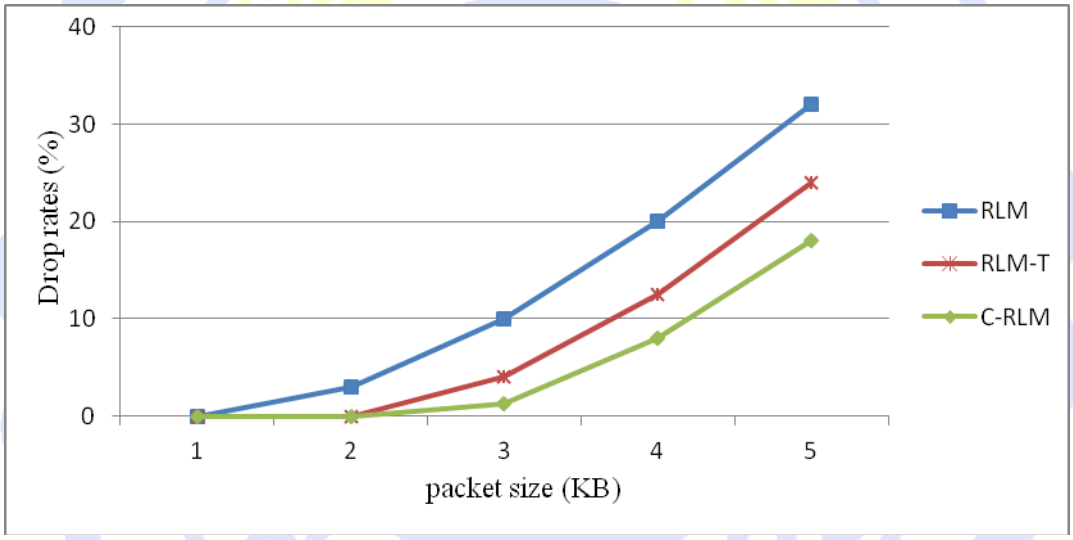

Figure 26 Packet drop rates on packet rate = 10 pkts/sec.

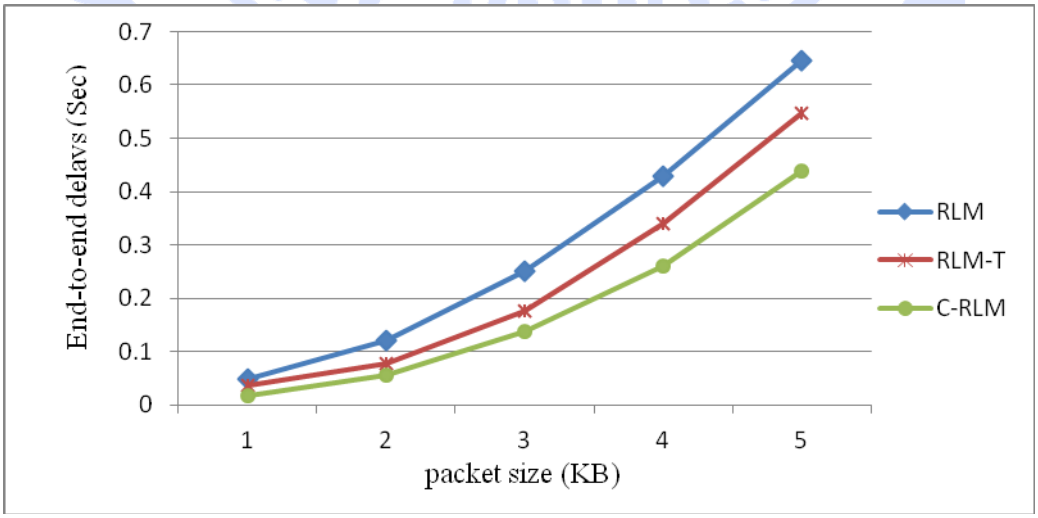

Figure 27 End-to-end delays on packet rate = 10 pkts/sec.

5.2 Different Packet Rates

In the second experiment, different packet rates ranging between 10 and 50 packets per second were given. Figure 28 shows the energy consumption, in which the C-RLM consumes the least energy. The reasons are mentioned above.

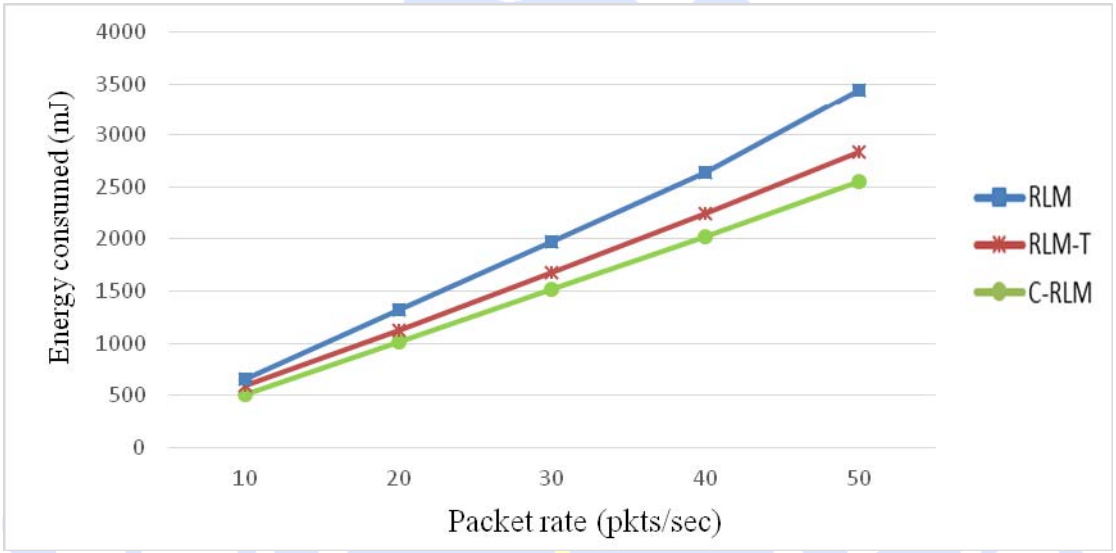

Figure 28 The total energy consumption of the three tested schemes on packet size = 1KB given different packet rates.

Figure 29 illustrates the throughputs of the tested schemes. The C-RLM has better throughputs than the other two have. Figure 30 shows their dropped rates. The shorter a routing path, the lower the probability of packets being dropped. When packet rates are higher, drop rates increase due to the higher probability of buffer overflow and network congestion. That is why the curves shown in Figure 30 go up quickly when packet rates are higher. Figure 31 plots the curves of end-to-end delay for the three tested schemes. Generally, higher packet drop rates often cause longer packet transmission delays, especially when dropped packets will be retransmitted. Now that the figures in Figures 28-31 are almost, respectively, the same as those in Figures 24-27. The reasons is that the bit rates sent in experiment 2 when packet rate = 10-50 pkts/sec(packet size = 1 KB/pkt) are the same as those in experiment when

packet size = 1-5 KB/pkt in experiment 1, $10 \leq i \leq 50$ is equal to j KB/pkt * 10 pkt/sec in experiment 2, $1 \leq j \leq 5$.

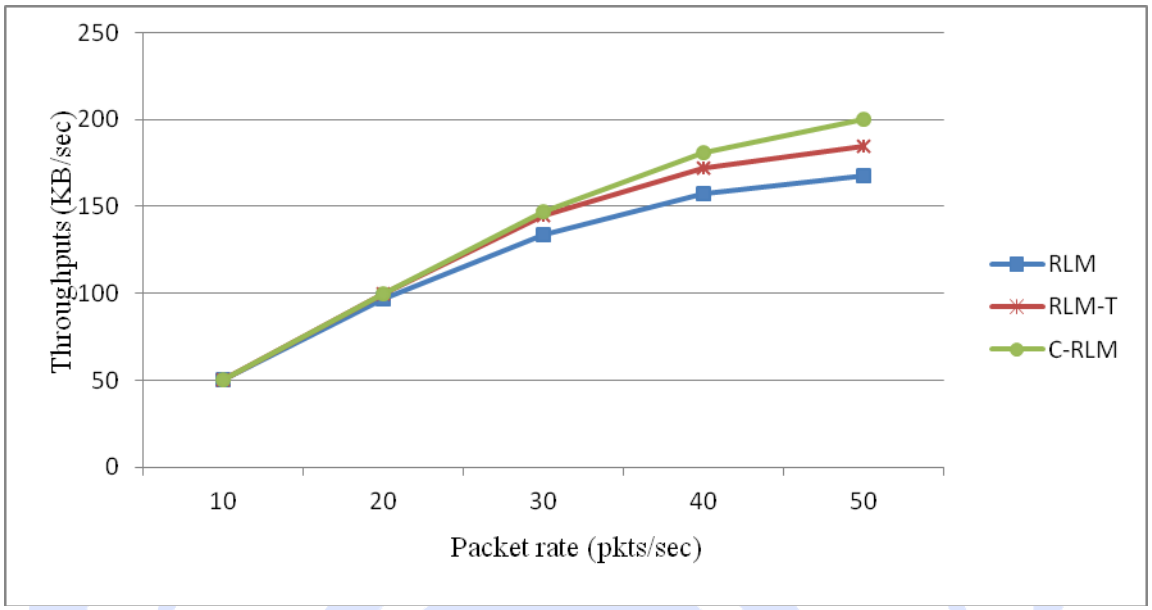

Figure 29 Throughputs at the sink on packet size = 1KB

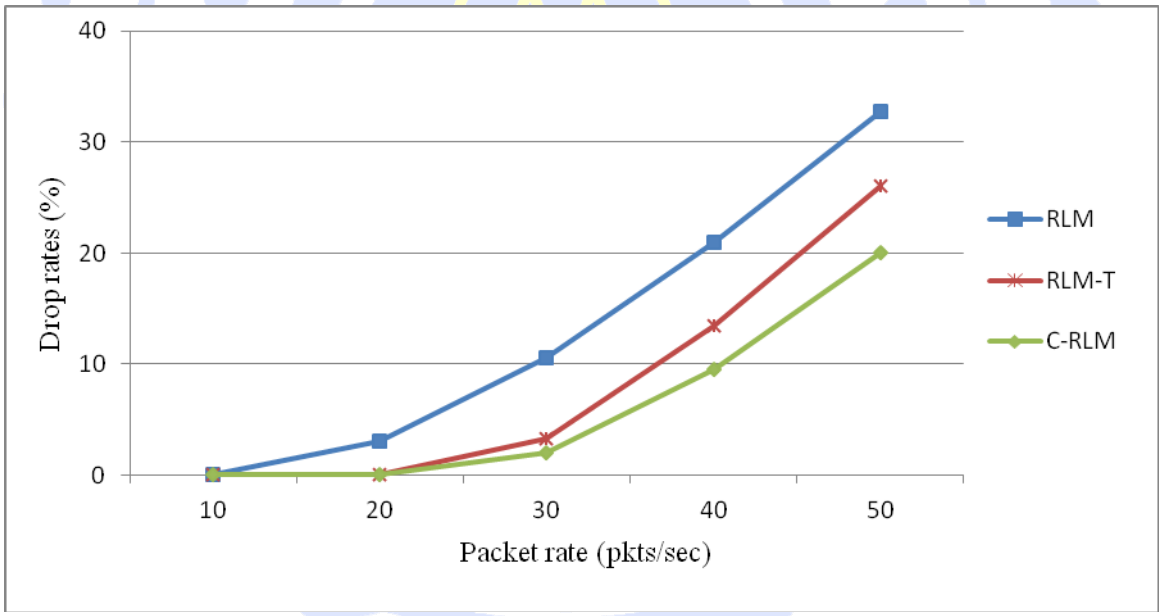

Figure 30 Drop rates on packet size = 1KB.

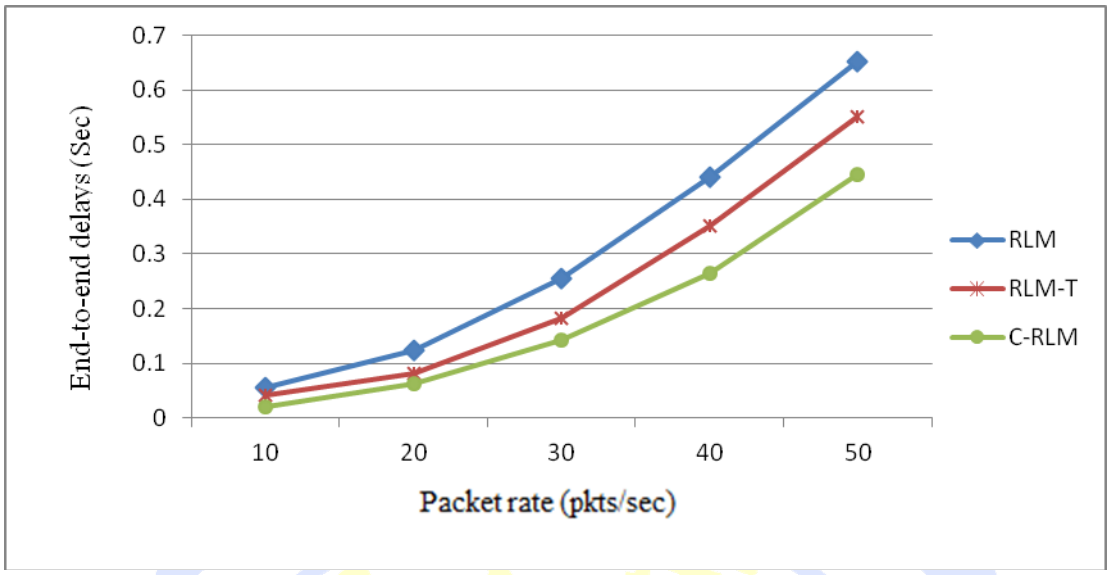

Figure 31 End-to-End Delay at the sink on packet size = 1KB.

5.3 Different Numbers of Nodes

In the third experiment, different numbers of nodes ranging from 30 to 70 are given to the sensing field. Table 8 lists the number. of nodes a packet passes through before it arrives at BS. Taking number of node = 70 as an example, in the C-RLM, a packet passes only 14 nodes before arriving at s BS. But a packet of the RLM-T (RLM) needs to go through 16 (22) nodes. The improvement of the C-RLM is significant.

Figure 32 shows the tested schemes' energy consumption. We can see that the C-RLM consumes the least energy. Figures 33-35 illustrate the throughputs, packet drop rates and end-to-end delays, respectively. The C-RLM outperforms the other two schemes. When comparing Figures 32 and 28, we can see that after many more nodes are distributed to the sensing field, the number of nodes that a packet has passed through before it arrives at BS increases. The energy consumption is then heavier. The range shown in Figure 32 linearly is between 500 and 1250 mJ. The reason is that we set the communication range of a node a constant. That means the distance between sander and receiver is not an energy-consumption parameter. But the more nodes a packet passes through before arriving at BS, the more energy the packet will consume, and the energy consumption is almost propotional to the number of the number of a packet has passed through.

Table 8 Number of nodes a packet passes through between the source node and BS

	Number of nodes				
	30	40	50	60	70
C-RLM	8	10	11	13	14
RLM-T	10	12	13	15	16
RLM	12	14	16	19	22

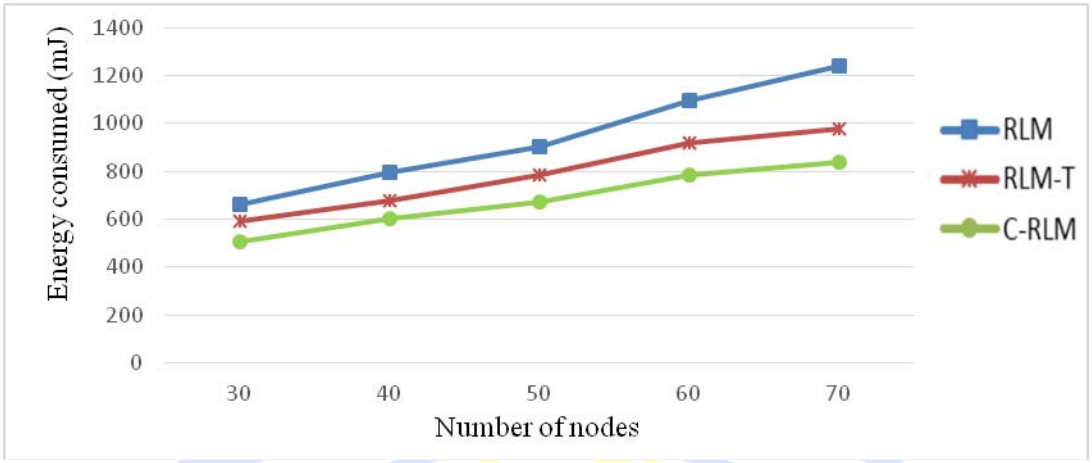

Figure 32 The energy consumption on packet size =1KB and packet rate = 10 pkts/sec given different numbers of nodes.

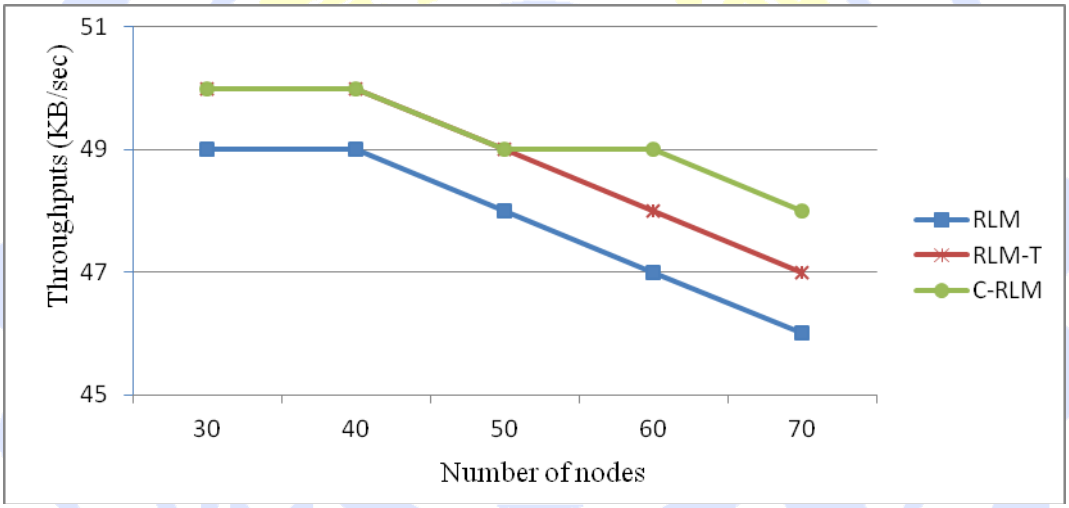

Figure 33 Thoughputs at the sink on packet size =1KB and packet rate = 10 pkts/sec.

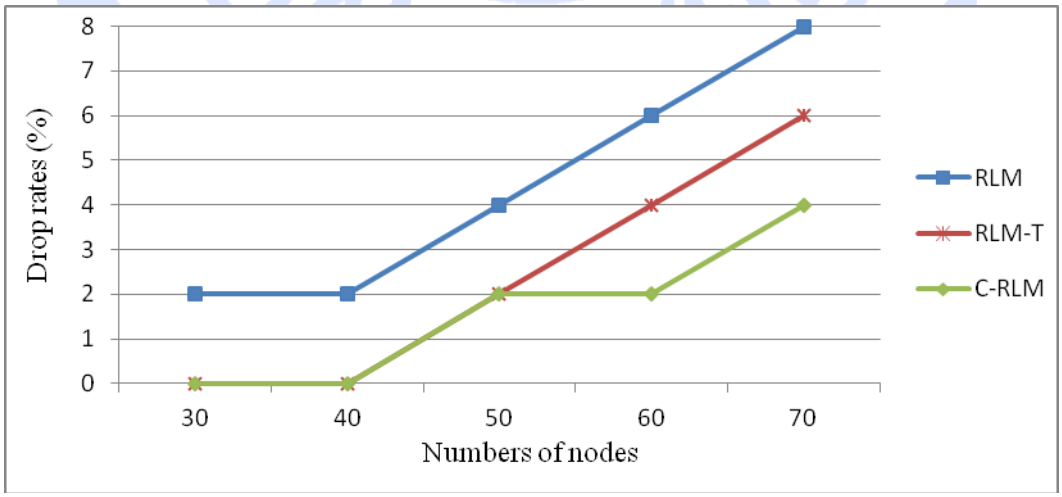

Figure 34 Packet drop rates on packet size =1KB and packet rate = 10 pkts/sec.

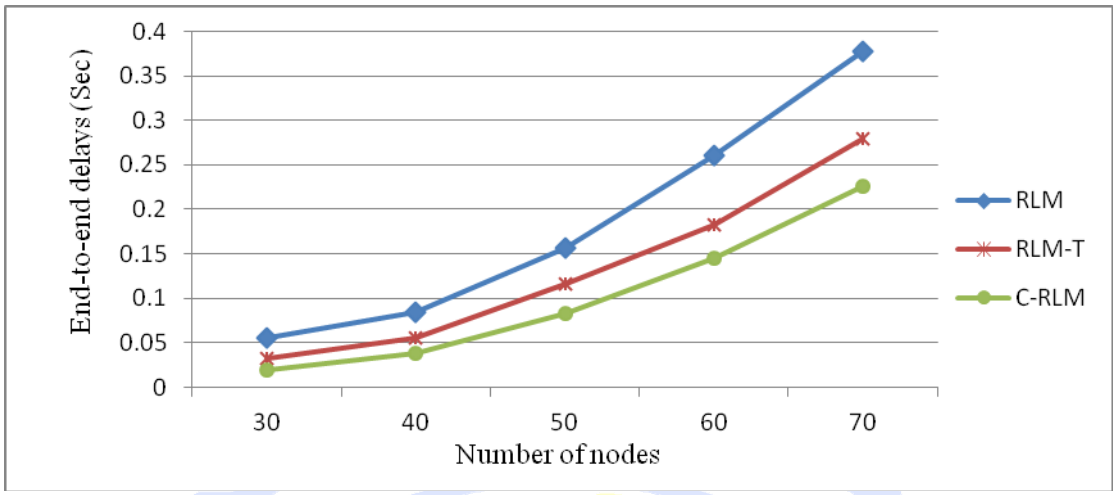

Figure 35 End-to-end delays on packet size =1KB and packet rate = 10 pkts/sec

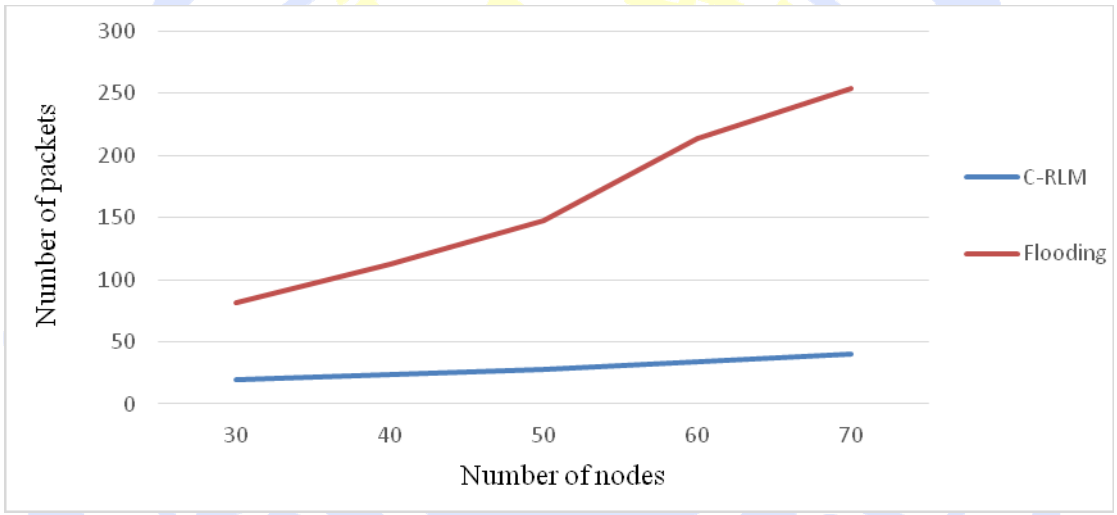

Figure 36 Packets number of C-RLM and Flooding algorithm.

Figure 36 compares C-RLM and AODV flooding routing approach [25]. We can see that the C-RLM can effectively reduce the number of generated Route-Request (RREQ) packets. Also, in the C-RLM, M-nodes transmit data packets to their H-node. The H-node then aggregates these packets to be one, and sends the one to BS via the path in the spanning tree established beforehand.

5.4 Different Numbers of Events

In the fourth experiment, different percentages of events ranging from 10% to 50% on a 30-node sensing field are given, i.e., the number of nodes ranging from 3 nodes to 15 nodes have detected events. Events are randomly generated in the field. Figure 37 shows the energy consumption of the three tested schemes. Due to generating many more data packets, in the energy consumed quickly rises, from 300 to 2100. Figures 38-40 illustrate the throughputs, drop rates and end-to-end delays, respectively. When

the number of events increases, the throughputs of course are then higher. But the probability of packet collision will be risen. So, the delays and drop rates are all higher. The RLM and RLM-T have some route cycles, the collision increases.

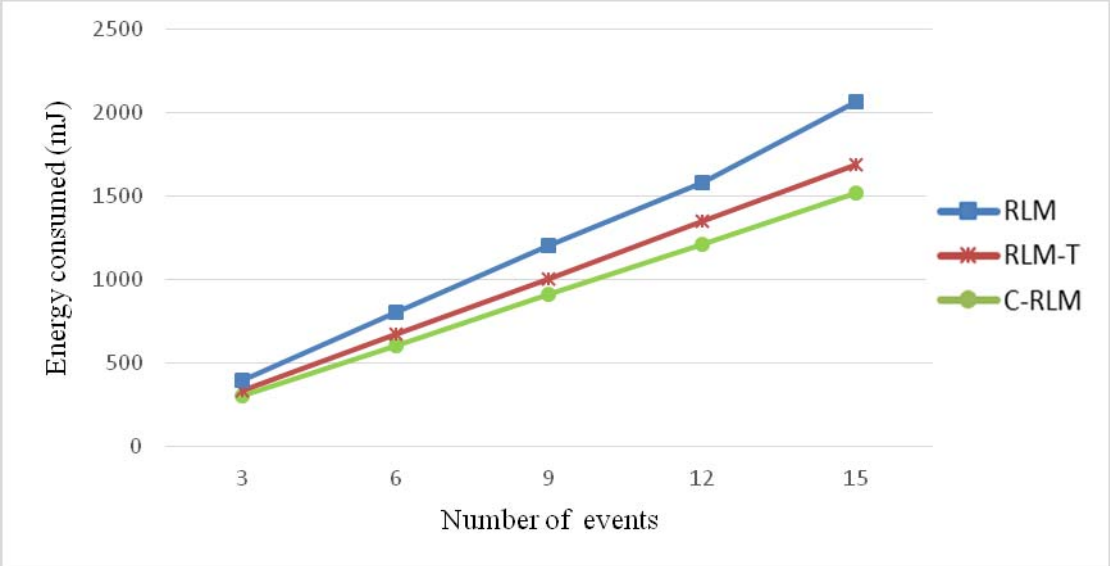

Figure 37 The energy consumption on packet size =1KB and packet rate = 10 (pkts/sec) given different numbers of nodes.

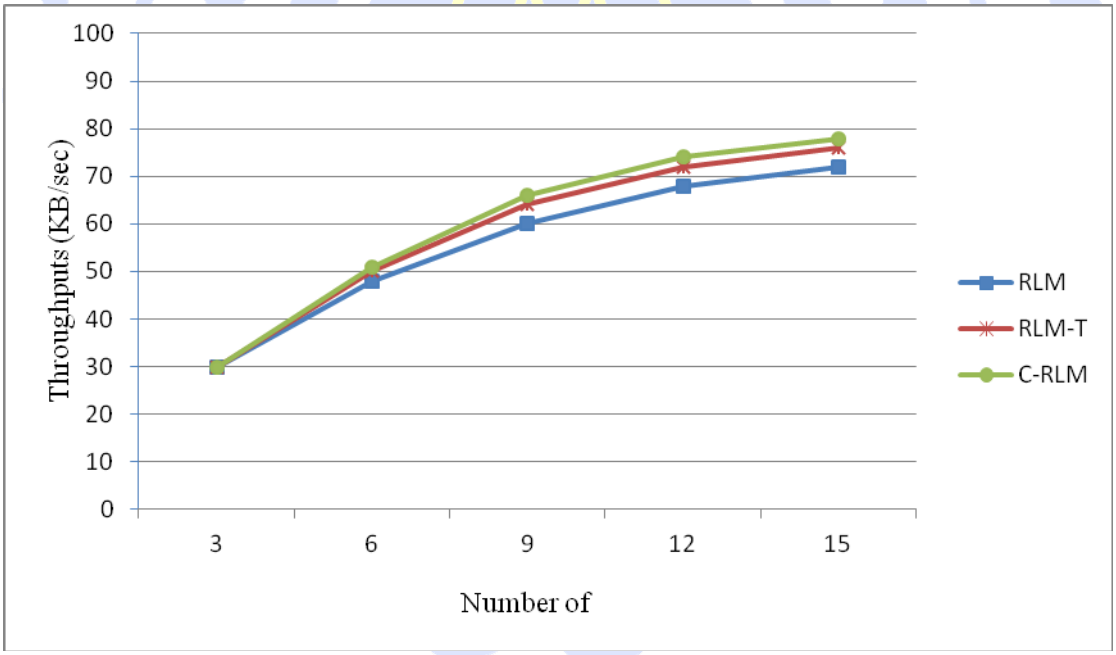

Figure 38 Throughputs at the sink on the packet size =1KB and packet rate = 10 pkts/sec.

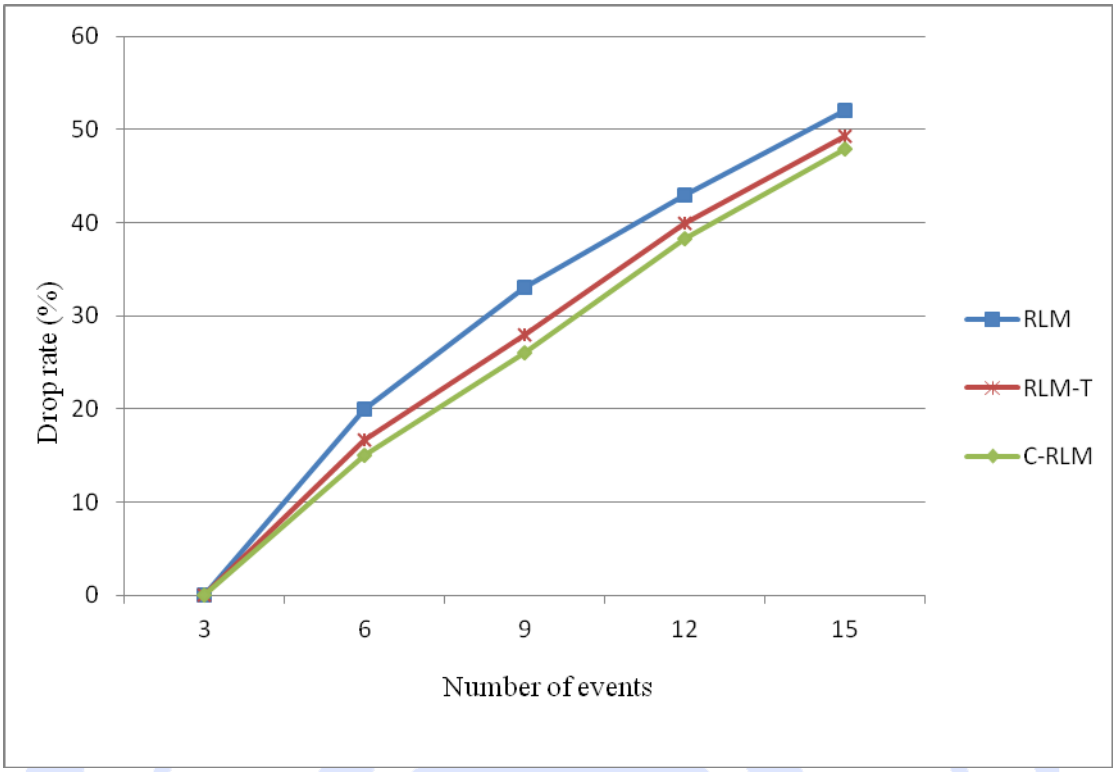

Figure 39 Drop rates on the packet size =1KB and packet rate = 10 pkts/sec.

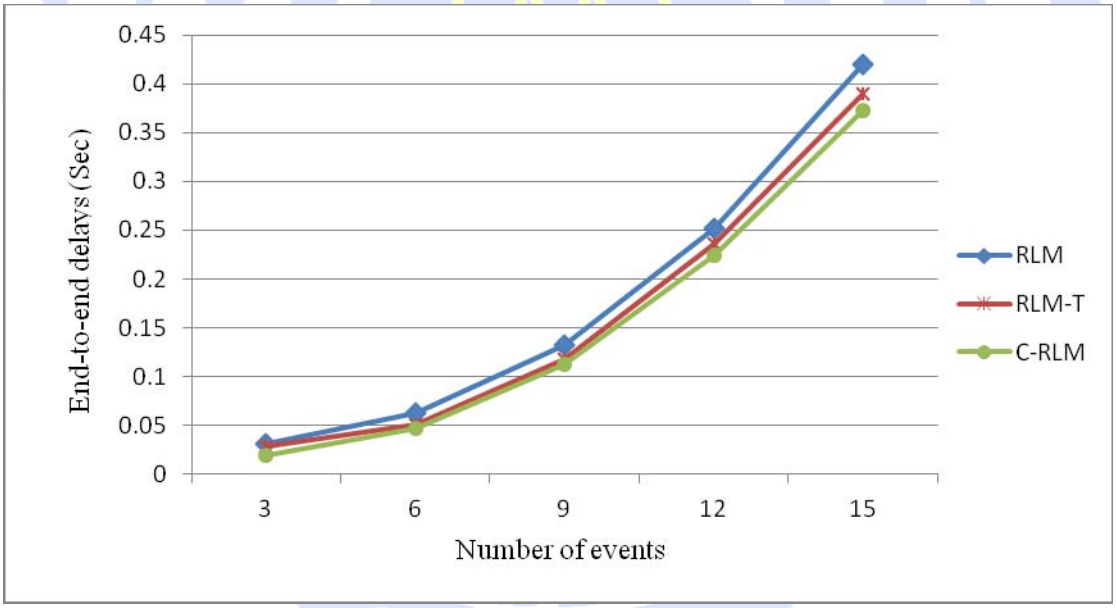

Figure 40 End-to-end delays on the packet size =1KB and packet rate = 10 pkts/sec.

6. Conclusions and Future Work

In this study, we propose the C-RLM to improve the RLM and RLM-T schemes and optimize the path that a node is connected to BS to further improve the drawbacks of the Flooding algorithm, i.e., sending too many unnecessary packets to establish a routing path. Our purpose is to lower wasted energy consumption on each transmission between two adjacent nodes since unnecessary packets wastes energy, consequently shortening the life time of a cluster head and that of the WSN, particularly when the length of the path between the cluster head and base station is long. However, the C-RLM establishes the best routing path for nodes, and minimizes the number of redundant links. Especially, in a multicast environment, this can effectively save much energy. Experimental results show that the energy consumption of C-RLM is less than that consumed by RLM-T. RLM and AODV, and its throughputs are higher than those of the former two schemes.

In the future, we would like to develop an effective and easy algorithm to find the cluster head, e.g., a node with higher and residual energy will be the cluster head in a periodical head-node selection process. In the C-RLM, restructuring the topology for node joint and deletion is a crucial work. We will try to simplify the procedures. We also want to derive the reliability model and behavior model for the C-RLM so that users can know the reliability and behaviors of the system before using it. These constitute our future studies.

7.References

[1] M. Gholami, M. Taboun, and R. W. Brennan, "Comparing alternative cluster management approaches for mobile node tracking in a factory Wireless Sensor Network," *The IEEE International Conference on Systems, Man and Cybernetics*, 2014, pp. 906-911.

[2] J. L. Minoi and A. W. Yeo, "Remote health monitoring system in a rural population: Challenges and opportunities," *IEEE Conference on Biomedical Engineering and Sciences*, 2014, pp. 895-900.

[3] S. Liu, W. Xie and Y. Zhang, "Research and implementation of WSN in fire safety applications," *International Conference on Wireless Communications Networking and Mobile Computing*, 2010, pp. 1-4.

[4] J.H. Liu, Y.F. Chen, T.S Lin, et al., "Developed urban air quality monitoring system based on wireless sensor networks," *International Conference on Sensing Technology*, 2011, pp. 549-554.

[5] https://en.wikipedia.org/wiki/Electronic_Toll_Collection_(Taiwan)

[6] Z. Bin, W. Jun, and L. Haiqing, "A data collection protocol for local mobile sensor network," *WRI International Conference on Communications and Mobile Computing*, 2009, pp. 523-527.

[7] WSN, http://en.wikipedia.org/wiki/Sensor

[8] S. Hnin Yu and P. H. J. Chong, "Cluster-based WSN routing protocol for smart buildings," *IEEE Vehicular Technology Conference*, 2015, pp. 1-5.

[9]Flooding, http://en.wikipedia.org/wiki/Flooding_%28computer_networking%29

[10] E. Kulla, M. Ikeda, L. Barolli, F. Xhafa, M. Younas, and M. Takizawa, "Investigation of AODV Throughput Considering RREQ, RREP and RERR Packets," *IEEE 27th International*

Conference on Advanced Information Networking and Applications, 2013, pp. 169-174.

[11] T.L. Yu, “On improving file searching in unstructured peer-to-peer systems,” Department of Computer Science and Information Engineering, PP.1-41 August 2008

[12] Y.P. Kao, “To improve the search cost of flooding in unstructured peer-to-peer network, ” PP.1-69 June 2010

[13] K. Iwanicki and M. van Steen, "On hierarchical routing in wireless sensor networks," *International Conference on Information Processing in Sensor Networks*, 2009, pp. 133-144.

[14] J. Sucec and I. Marsic, "Hierarchical routing overhead in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 3, 2004, pp. 46-56.

[15] Y. Qin, "Analysis of cluster-based hierarchical routing in ad hoc wireless networks," *Electronics Letters*, vol. 42, 2006, pp. 474-476.

[16] Z. Zhipu, L. Hui, P. Kai, Y. Chaoqi, C. Fuxing, and L. Dagang, "Centralized flat routing," *International Conference on Computing, Management and Telecommunications*, 2014, pp. 52-57.

[17] A. Kanavalli, D. Sserubiri, P. Deepa Shenoy, K. R. Venugopal, and L. M. Patnaik, "A flat routing protocol for sensor networks," *International Conference on Methods and Models in Computer Science*, , 2009, pp. 1-5.

[18] K. Prasanth and P. Sivakumar, "Location based routing protocol - A survey," *International Conference on Computer Communication and Informatics*, 2014, pp. 1-6.

[19] L. Blazevic, J. Y. Le Boudec, and S. Giordano, "A location-based routing method for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 4, 2005, pp. 97-110.

[20] J. Grover, Shikha, and M. Sharma, "Optimized GAF in wireless sensor network," *International Conference on Reliability, Infocom Technologies and Optimization* (Trends and Future Directions), 2014, pp. 1-6.

[21] https://en.wikipedia.org/wiki/Breadth-first_search

[22] http://www.cs.nott.ac.uk/~pszna/G5BADS04/graphs2.pdf

[23] E. Horowitz, S. Sahni, and D. Mehta, "Fundamentals of data structures in C++," Second Edition, Silicon Press, 2007.

[24] ns-2, http://www/isi.edu/nsnam/ns/

[25] AODV, https://tools.ietf.org/html/rfc3561

Appendix The Node Information Table

The Node Information Table established for the network topology shown in Figure 1.

Table A-1 An example Node Information Table.

NodeID		Degree	NeighborSet	NodeType	ClusterID	ClusterHead	ConnectingClusters
1	0	1	2				
2	2	3	1, 3, 4				
3	2	3	2, 4, 8				
4	4	5	2, 3, 5, 6, 10				
5	2	2	4, 6				
6	2	3	4, 5, 7				
7	2	3	6, 10, 11				
8	2	3	3, 9, 11				
9	4	3	8, 10, 11				
10	4	4	4, 7, 9, 11				
11	6	4	7, 8, 9, 10				

Table A-2 The Node Information Table after calculation step, i.e., all records shown in Table A-1 are sorted on in a descending order.

NodeID		Degree	NeighborSet	NodeType	ClusterID	ClusterHead	ConnectingClusters
11	6	4	7, 8, 9, 10				
4	4	5	2, 3, 5, 6, 10				
9	4	3	8, 10, 11				
10	4	4	4, 7, 9, 11				
2	2	3	1, 3, 4				
3	2	3	2, 4, 8				
5	2	2	4, 6				
6	2	3	4, 5, 7				
7	2	3	6, 10, 11				
8	2	3	3, 9, 11				
1	0	1	2				

Table A-3 The Node Information Table after node re-clustering.

NodeID		Degree	NeighborSet	NodeType	ClusterID	ClusterHead	ConnectingClusters
11	6	4	7, 8, 9, 10	H-node	1	11	
4	4	5	2, 3, 5, 6, 10	H-node	2	4	C_{10} , ,
9	4	3	8, 10, 11	M-node	1	11	
10	4	4	4, 7, 9, 11	M-node	2	4	
2	2	3	1, 3, 4	M-node	2	4	
3	2	3	2, 4, 8	M-node	2	4	
5	2	2	4, 6	M-node	2	4	C_2
6	2	3	4, 5, 7	M-node	2	4	
7	2	3	6, 10, 11	M-node	1	11	
8	2	3	3, 9, 11	M-node	1	11	
1	0	1	2	H-node	3	1	