A Fault Tolerance Spider-Net Zone Routing Protocol for Mobile Sinks Wireless Sensor Networks

Shih-Hao Chang, *IEEE Member, Intel-NTU Connected Context Computing Center, National Taiwan University, Taiwan, sh.chang@ieee.org*

Ping-Tsai Chung, *IEEE Senior Member, Dept. of Computer Science, Long Island University, Brooklyn, New York, USA, pchung@liu.edu*

Abstract—In this paper, we introduce a Fault Tolerance Spider-Net Zone (FTSNZ) routing protocol that combines spider-net zone routing and consensus-based fault tolerance to provide high performance, energy-efficiency and reliable communications in MSWSN (Mobile Sinks Wireless Sensor Networks). MSWSN consists of a large number of sensors with several mobile monitoring terminals, called mobile sinks. This algorithm utilizes a spatio-temporal topology that we call 'Spider-Net', which is inspired by the way that a spider hunts using its own web. The spider uses spider catching silk, probably in some form of a simple sheet web, to capture their prey. When spider makes its own web, it will make radii lines of the web first, and then makes the circular threads around the web until whole web is finished. We use this phenomenon to build our network topology and routing for MSWSN. The sinks similar as spiders can have mobile capability and access data from network. For concerning the data reliability of spider-net, we provide consensus-based fault tolerance to avoid data failure in network. Through our simulation study, we show that FTSNZ scheme achieves less energy consumption, less average delay and better packet success ratio than other related protocols.

Index Terms—Mobile sink, wireless sensor networks, spider-net zone routing, network topology, consensus-based fault tolerance, data dissemination, network reliability, performance evaluation.

I. INTRODUCTION

Spiders are one of the most technical predators you can observe in the natural world. A spider normally will build a web as a trap; this web is made of spider silk, a thin, strong strand extruded by the spider. The silk composes a spider web which can be used to wrap prey and for many other applications. Consider how the spider builds the spider-net; it will make radii lines of the web first. After finishing the radii lines of the web, the spider start to make the circular threads around the web to build the whole web. Wireless sensor networks (WSN) consist of a large number of sensors that have a variety of applications such as battlefield surveillance, environment monitoring, and target tracking. WSN normally use monitoring terminals called sinks, these sinks could have mobile capability, similar as the spiders that catch the sources (prey) from the network.

Mobile sinks wireless sensor networks (MSWSN) consist of a large number of sensors with several mobile monitoring terminals, called mobile sinks. Example of a mobile sink is a smart phone or PDA (Personal Digital Assistant) device carried by users to gather sensor readings, process and aggregate sensed data from local environment. This promising solution would enhance performance, extend network lifetime and reduce consumed energy and latency by routing data to a nearby mobile sink. However, it also introduces many problems and research challenges such as the high communication overhead for updating the dynamic routing paths to connect to mobile sinks, and the recovery of lost data because of sink mobility. In this paper, we introduce a new routing topology for data dissemination in a heterogeneous MSWSN. This algorithm is called Fault Tolerance Spider-Net Zone (FTSNZ) routing protocol.

We use the concept of spider-net zone routing protocol [1], which provide an energy efficient and low latency routing protocol for multiple events propagation in wireless sensor networks with mobile sinks. We assumed that these mobile sinks can notify their neighboring cluster heads about their current location, and then these neighboring cluster heads will relay the data to mobile sinks. These neighboring cluster heads, which called moles, can provide current mobile sinks locations and maintain routing paths according to their movements.

The major contributions of this paper are listed below. First, we use spider-net topology proposed in [1] to provide high performance routing through a spider-net zone sensor network. Second, we use consensus-based solutions to provide reliable network communication. Our algorithm adapts the magnetic query dissemination and extends it to provide a spider-net zone topology to improve network efficiency. Instead of changing the whole routing path when the mobile sink moves, we propose a partial update of the routing path based upon the new location of the mobile sink. Third, we provide a fault tolerance mechanism to increase the network reliability since sensor networks are prone to failure due to hardware, software, and environment issues. The main idea of this fault tolerance

Dr. Shih-Hao Chang, Research Fellow, Intel-NTU Connected Context Computing Center, National Taiwan University, Room F, 7F, Barry Lam Hall, No. 1, Sec. 4, Roosevelt Road, Taipei, 10617 Taiwan, Email: sh.chang@ieee.org.

Prof. Ping-Tsai Chung, Department of Computer Science, Associate Professor and Chair, Long Island University, 1 University Plaza, Brooklyn, New York 11201, USA, E-mail: pchung@liu.edu.

mechanism is to ensure that faulty cluster heads do not share in data dissemination process, thus guarantee that mobile sinks gather correct data from proper cluster heads only.

FTSNZ is a new routing algorithm utilize spider-net zone routing to achieve high performance, energy efficiency and reliability in MSWSN, our algorithm adapts of QoS pattern, and data centric dissemination. Our design criteria are described as follows:

- High performance: Routing mechanisms can provide high performance data communication with extended networks lifetime by application requirements in terms and network conditions.
- Energy-efficiency: Given data update demands, the protocol should be able to satisfy them with lower energy dissipation and ultimately extend the network lifetime.
- Reliable communications: The protocol should be designed to achieve high data reliability in the presence of network dynamics.

The rest of the paper is organized as follows. In Section II, background and related works are briefly described. In Section III, we present the Fault Tolerance Spider-Net Zone (FTSNZ) routing algorithm, we first discuss the general assumptions, we then organize the randomly deployed sensor network to form a spider-net topology mechanism and present our FTSNZ data dissemination mechanisms. In Section IV, we discuss the network management of the Fault Tolerance Spider-Net Zone (FTSNZ) routing protocol. We first present data collection and redirect tunnel for supporting mobile sinks random movement. We then discuss the Quality of Service (QoS) pattern control to provide efficient routing, better guaranteed scheduling. Finally, we conclusion and further work are explored in Section V.

II. Background & Related Works

In this section, we provide an overview of a selection of protocols and mechanisms that have been developed to improve the performance of MSWSN. We will focus on routing protocols; path planning, and reliable data transfer mechanisms. To the best of our knowledge so far, only limited research has tried to address high performance issues in MSWSN. The Cluster-Based Routing Protocol (CBRP) is a well-known WSN protocol. The main strategy of the CBRP is to use a clustering approach to aggregate data and minimize on-demand route discovery traffic, reduce travel time and latency [1]. Cluster-heads normally have more power strength than normal sensor node in WSN.

Zone Routing Protocol (ZRP) [2] is a hybrid routing scheme that contains both reactive and proactive components. ZRP uses proactive Intra Zone Routing Protocol (IARP) to maintain routing information for nodes that are within the routing zone of the node that can reduce the traffic overhead in the network. ZRP uses reactive Inter Zone Routing Protocol (IERP) that offers enhanced routing discovery and zone routing services based on local connectivity monitored by IARP. Fisheye Zone Routing Protocol (FZRP) [3] is an extension of ZRP adopting the concept of Fisheye State Routing (FSR) [9]. The idea of FZRP uses the ZRP concept to update routing information in its defined basic zone and to add a reduction factor to reduce the frequency of transmission updates in the extend zone. However, they do not concern the traffic overhead and reliability issues for link failure in WSN.

A number of research efforts which focus on routing protocols in MSWSN aim at achieving power efficiency, load balancing and extended system lifetime in hostile environments. The TTDD [4] protocol builds a grid structure and divides the network into cells with several dissemination nodes. The dissemination nodes are responsible for relaying the query and data to and from the proper sources. ODDD [5] is another protocol that improves the TTDD [4] that is more suitable for dynamic large wireless sensor networks. In ODDD, a source does not proactively construct a virtual grid. Instead, a source sends a data announce message along the X-axis only. Therefore, ODDD reduces the amount of communication overhead for creating and maintaining virtual grid structures over the entire network. However, due to ODDD sends along the X-axis only, therefore it may take grid node fail risk that lead to network failure. In this routing mechanism, there is no fault tolerance algorithm for any grid node fail on X-axis.

SAFE [6] and Directed diffusion [7] utilize data dissemination path sharing among multiple data sinks and attempt to minimize message exchanges to achieve energy saving over the network. They flood "query" messages to the entire network and find out the source node and junction nodes that help to setup the dissemination path. Flooding is geographically limited to forward the query to nodes along the direction of the source. In contrast to directed diffusion, SAFE uses geographically limited flooding to find the gate connecting itself to the tree. Directed diffusion and SAFE are most effective in small-to-medium size sensor networks. In a very large network, the initial sensor flooding may consume too much energy. Due to their junction node's algorithm, it will be resulted in long-term waiting for the source information when the path is broken or congested. Our approach aims at providing energy efficiency, and high performance data dissemination for MSWSN.

III. Fault Tolerance Spider-Net Zone (FTSNZ) Routing Algorithm

Fault Tolerance Spider-Net Zone routing protocol (FTSNZ) is a hierarchical hybrid scheme for query based data dissemination for mobile sinks in WSN. In this section, we first give the assumptions and environment requirements. Then we explain the FTSNZ in the following four subsections: In the section A, we provide a set of general assumptions of the network model for FTSNZ. We then discuss the construction spider-net topology mechanism in section B, the spider-net data dissemination mechanism in section C.

A. General Assumptions

We consider the specific case where there is an inherent pattern to the sink's movement through the sensor field with allowance for some variance. The network model for FTSNZ makes the following general assumptions:

- The sensor nodes and cluster-heads are knowledgeable about their neighbor nodes and location aware of their geographic locations. Several algorithms exist to estimate the locations of the individual nodes that provide route messages towards geographic locations.
- The sensor nodes and cluster heads are regularly deployed over the entire sensing region. The sensor nodes and cluster heads are heterogeneous with constrained energy resource. Their wireless communication channels are bidirectional.
- After deployment these sensor nodes and cluster heads will all remain stationary at their initial in a flat two-dimensional space.
- Each mobile sink only broadcasts the hello message to the neighboring cluster-heads and concentrates on continue listening to reply messages around it.
- The sensor nodes will aggregate data or event messages and send across to vicinity cluster-heads then forward to mobile sinks. That is to say, sources and mobile sinks are typically much further apart than a single radio radius (multi-hop).

B. Spider-Net Zone Topology Mechanism

We use the spider-net topology mechanism [1] to form the dynamic network in randomly deployed sensor nodes. The idea of the spider-net zone [1] is based on assumption which all nodes in the network can aware of their own coordinates location, and each node also can obtain neighbor nodes coordinate information by periodically exchange coordinates information. The key of this construction is any GCH in the network can broadcast a spider-net zone construction message to the center area of the network by directional antenna. This construction message will be forwarded to it specific direction 1-hop away neighboring nodes inside range *R*, which has angle θ. The nodes receive this message will check its location information whether it is a center node. If it is the center node, it will broadcast this construction message to its 1-hop away node. Otherwise, it will continuously forward this construction message to its next neighboring nodes with same angle θ.

To form the CCHs in the spider net, each cluster-head can use peer-to-peer technology which provides advertisement, listening and publication functions. Each cluster head will broadcast advertisements and listen to neighboring cluster-heads before its control timer is expired. The advertisement and listening functions for each cluster-head becomes aware of its coverage level to neighboring nodes. The timer function in each cluster-head closely monitors the status of neighboring cluster-heads and values can be set to the communication interval of the query message. This coverage level is to decide the cluster head role in the spider-net. CCH can be defining as a cluster-head that covers three or more cluster-heads in one web.

Core Cluster Head (CCH) Election: When initiating a randomly deployed sensor network, each cluster head will broadcast its advertisement and listen to neighboring cluster heads' responses. Therefore, it is likely to have multiple CCHs or non-CCHs in a region.

(1) For multiple CCHs in a region: These CCHs can be aware themselves when the timer of advertisement and listening is expired. If there are multiple CCHs in a region, each candidate cluster head will announce it to be the CCH and will also receive the announcement messages of other CCHs. This announcement message will include their ID numbers. We assume that every sensor node has an ID number. Therefore, when the candidate cluster head receives other candidate cluster heads with ID larger than its own, it will stop sending announcement messages and change its state to become a normal cluster head. Finally, there will be only one CCH in the spider-net.

For non-CCHs in a region: The announcement message approach also applies to the situation of non-CCHs in a region. When the timer has expired, if no cluster head has received any announcement in this region, then the cluster heads covered by two cluster heads can be the candidate CCHs. Therefore, if the candidate cluster head receives other candidate cluster head IDs bigger than its own, it will stop sending announcement messages and change its state to become a normal cluster head.

(2) Gateway Cluster Head (GCH) Election: After CCHs for every spider-net have been elected; CCH will broadcast announcement messages within α hop distance to notify the border cluster heads. This "α" hop distance can be decided by the designer that defines the size of the spider-net. In this situation, those border cluster heads receiving two or more different CCH ID announcement messages are the GCHs. After the GCHs have been decided, the CCH can manage all the routing information to the GCHs in the spider-net. At this stage, the shape of the spider-net is now similar to the radii lines of a web built by a genuine spider, as shown in Fig. 1 (a). GCHs are mainly for the maintenance of link changes in two different webs. If the topologies between two webs go down or up, a GCH will inform the CCHs in these two regions.

(3) Intermediate Cluster Head (ICH) Formation: After the GCH has been elected, the CCH will broadcast to its 1-hop away cluster heads. These cluster heads are similar to the first ring that circulates around the CCH, which we call the CCH group. This CCH group will start to organize the network similar to the chain protocol [11], which uses the location information to greedy broadcast to its vicinity neighbors to form the network. Therefore, the cluster heads on the CCH to GCH paths that also receive location based greedy broadcast will become the ICHs

in the network. This network formation is like the spider building the circular threads on its web until it reaches the GCHs. Each ICH will have routing information for all the ICHs, CCH and terminal GCH on the path information. Therefore, this spider-net can divide the network into different zones.

CCH Group Functionality: the central bold circle is the CCH group. This bold circle reduces the distances to the destination. The cluster heads belonging to this group will share routing information; therefore the CCH group can provide efficient data dissemination in the network. The CCH group also provides many benefits in terms of maintenance and reliable communication. When an ICH sends a 'hello' packet to its neighbor ICH, it starts a hold timer, which is the amount of time that a router treats a neighbor as reachable. If a 'hello' packet is not received within the hold timer, the hold timer expires and this cluster head will inform the CCH group to maintain the route. Due to the fact that it only monitors neighboring nodes, the energy cost in maintenance overhead is small. When an event occurs in the spider-net, the event message will be sent to the CCH group. The CCH group can know the event source ID and event message. When the query message arrives, the CCH group can respond with source information to the mobile sink, and then the sink can decide to query or discard. In addition, due to the shape of the spider-net, it can provide multi-route paths in the case when the first routing path failed. After the spider-net zone topology has been initialised, each spider-net zone is similar to a big triangle and each cluster head will know its role in this network.

C. Spider-Net Data Dissemination Mechanism

In this section, we introduce a simple routing path setup and data dissemination mechanism for spider-nets in WSN. After the network topology has been initialized, the routing paths will be set up before the sinks reach the spider-net network. In our network topology, each small square of the spider-net is a trapezium shape, similar to the square shapes found in TTDD [6]. When an event happens in the spider-net, the response is similar to that of TTDD, with transmission of the event message only to the vicinity ICHs. This event message will use the routing path from the ICHs to the CCH group. Then the cluster heads in the CCH group belonging to the event message zone will save this event message and keep it memory resident until the sinks hello message arrives. These sinks can broadcast hello messages to α-hop-away cluster heads around the network. These vicinity cluster heads we call mole nodes. The mole nodes will forward the hello message through the ICHs to the CCH group and redirect the event message backflow to the sinks. For providing reliable communication in the Fault tolerant spider-net zone routing protocol (FTSNZ), we provide Quality of Service (QoS) patterns to adapt the data dissemination. Then we introduce our routing mechanisms in FTSNZ.

IV. FAULT TOLERANT SPIDER-NET ZONE (FTSNZ) ROUTING MANAGEMENT

A. Quality of Service Pattern

We apply a pattern framework for analysis of QoS time from [10] to our work. Due to this framework being focused on spatial-temporal QoS pattern analysis, the statistical model can be set up for every cluster head to analyse regional network conditions. This provides more practical measurement of QoS that has benefits for evaluating network conditions in dynamic network topologies. In intra-domain topologies, we can use the $Q_{net\_traffic}$, to measure network traffic on the routing paths in our network. In inter-domain topologies, we can use the Q_{end_delay}, to measure end-to-end delay on the routing paths in our network. These statistical results can apply thresholds to decide the network link quality. The time series data ΔQ_{end_delay} describes the QoS of end-to-end delay in connection C_i for some interval Δ.

$$Q_{end\_delay} = \Delta(Q_{C_1} + Q_{C_2} + \cdots + Q_{C_n}) \qquad (1)$$

B. Intra-Spider-Net Zone Routing Algorithm & Maintenance

As we mentioned in the description of ICH formation, the CCH group will start to organize the network similarly to the chain protocol [12]. They will use greedy broadcast announcement messages. These announcement messages work like the spider building the radial lines of the web, which will transverse through every ICH in the network until arriving at the GCHs. After ICHs and GCHs received these announcement messages, they will add the routing paths into the aggregation messages, from the transverse previous cluster heads and next cluster heads. In addition, as we discussed above, the CCH will broadcast announcement messages within α hop distance to notify the border cluster heads. This longitudinal routing will provide every ICH on this path with the routing information to its CCH and GCH. These announcement messages will allow the construction of a small routing table in each ICH, which records the neighboring cluster head information.

The announcement messages include a time tag. Therefore, both the CCH group and GCHs can summarize total transmission time and select the best routing paths to each other as shown as Fig. 1. After the best routing paths have been selected, the GCHs will use the best routing path to return registration messages to the CCH. Each registration message includes the gateway cluster head's (GCH's) ID, transmission time and routing path information. Once the registration packets have been received by the CCH, intra-region routing paths are set up.

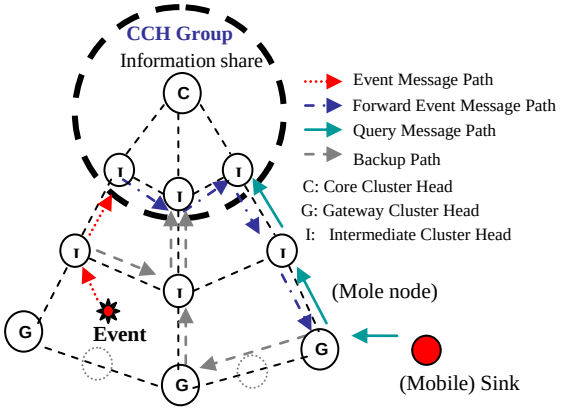

Figure 1. Intra-Spider-Net Zone Diagram.

We now consider how the event message transmits to the sinks. When a sensor in the network sends an event message to the vicinity GCH (or ICH), GCH use the greedy forwarding routing protocol to forward event messages to the CCH group. This event message will be saved in the CCH group until the sink's hello message arrives. When the sink sends a hello message, it will only be broadcast to α-hop vicinity mole nodes. These mole nodes will forward the hello message through middle ICHs to the CCH group. When the CCH group receives this message, they will check any aggregated event message that can send to sinks nodes. If the events over the defined threshold, these event messages can be send back to the sink using the same routing path. In addition, the event message also includes a time tag. If the time tag has expired, the CCH group will remove this event from its memory.

For the routing update and maintenance problem, we use QoS patterns and control packets to provide more practical measurement network conditions to find better paths. These QoS patterns ($Q_{pattern}$) have defined levels in each cluster head. The QoS patterns can provide consequences of conducting the next hop routing path. When CCHs receive this packet, they maintain or update routing paths in the region in accordance with the QoS levels of this path. This QoS pattern composes different gravity (Gty) of QoS levels of this routing path. These gravity parameters are used to provide self-adjustment and optimize priority tasks in each cluster head. In our algorithm, we list the QoS end-to-end delay's gravity (Q_{Gty}), Power remains' gravity (P_{Gty}) and Network lifetime's gravity (N_{Gty}).

$$Q_{pattern} = \Delta(Q_{end\_delay} \times Q_{Gty} + P_{remain} \times P_{Gty} + N_{lifetime} \times N_{Gty}) \quad (2)$$

, where P_{remain} is the remaining power, and $N_{lifetime}$ is the network lifetime.

If the QoS pattern is available, we apply the same algorithm from [11] which provide routing path with the probability P_{path} to CCH as follows.

$$P_{path} = \frac{(Q_{pattern})^{\beta}}{\sum_{j \in N_d^i} (Q_{pattern})^{\beta}}, \beta \geq 1, \quad (3)$$

where N_d^i is the set of neighbours of i over which a path to d is known, and β is a parameter value which can lower the exploratory behaviour of the proactive control packets. This proactive control packet can provide an adaptable routing path by evaluating the QoS pattern value in this path. In this way, routing failures can be solved when the single path suffers a failure.

C. Inter-Spider-Net Zone Routing Algorithm & Management

In this section, we discuss how to setup the routing paths between different regions. As we mentioned above, after a GCHs receives the announcement message from the CCH, the GCH will return the registration message to the CCH. When the CCH receives this registration message during the long waiting timer (LWT), then both CCH and GCH will have the routing information to each other. Otherwise, after the LWT expires and the CCH still does not receive a registration message from the GCH, the GCH will not belong to this CCH. Once the registration message return to the CCH, inter spider-net zone routing algorithm is initialised.

The query message includes a query ID, query item and Time-To-Live (TTL) value. When the CCH group receives a query message, it will check these two contexts. If these two parameters are not in this web or not limited in this spider-net web, then the CCH group will forward the message to the GCHs inside the web region. When a query message arrives at a regional GCH, we provide a simple "match function" for the query message in each GCH node. This match function can be executed as a simple answer function, which provides 'yes', 'no' and 'don't know'. There are three different answers can deal with query message forwarding. If the GCHs have the query answer, it means "yes", it will send back the source event to the sinks. Otherwise, the GCHs only have "no" or default value "don't know" to deal with the query message. When GCHs match the query message with "no" or default value "don't know", they forward the query message to others in the CCH group until they match the function or the TTL expires. These GCHs provide stability of routing to the spider-net zone.

For the routing updates and the routing maintenance, we use the same maintenance concept as the QoS pattern. Each cluster head will periodically send out control packet during the course of data session. In this case, if the GCH fails, the vicinity cluster head will send out control packets to search the cluster heads around this region to locally replace the GCH.

D. Data collection and redirect tunnel

Due to mobile sinks random movement, this causes broken link between mobile sinks and the mole nodes. Therefore, spider-net provides a redirect runnel, which provides a redirect link to the new mole node locations. One of the best ways is to build a link between new mole nodes with the previous mole nodes before the old link fail.

When mobile sinks move, they will periodically broadcast hello message to mole nodes. These mole nodes receive this message and determine the signal strength from the mobile sink. When communication signal with the vicinity cluster heads rises over the threshold, these cluster heads can become the new mole nodes. The new mole node will broadcast update link messages including IP address to the old mole nodes. On the other hand, when the old mole nodes receive such a broadcast message from a new mole node, it informs the previous cluster heads to redirect messages to the new mole nodes as shown in Fig. 2.

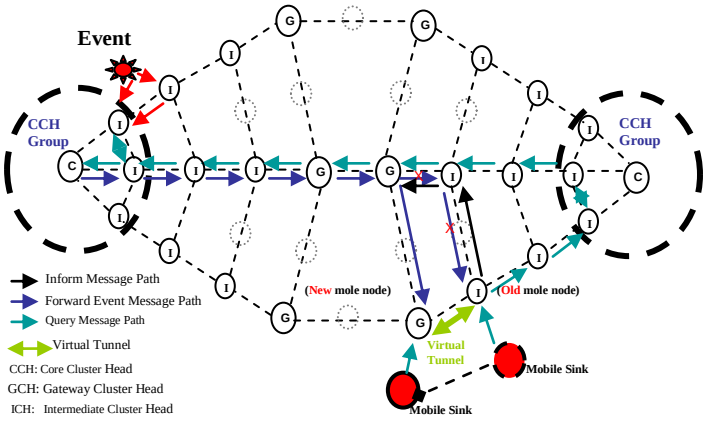

Figure 2. Regional Radiation Topology Algorithms Diagram.

V. FAULT TOLERANCE SPIDER-NET ZONE (SNZ) ROUTING ALGORITHM

Consensus-based Fault Tolerance Algorithm: Generally sensor nodes are very prone to failure. In dynamic and hostile environment the rate of failure increases because of environmental issues. This failure of nodes may cause malicious faults [6]. A malicious fault occurs when a faulty node delivers inconsistent data to non–faulty nodes thus results in so called Byzantine Faults [6]. Byzantine faults are described as a node in a network not only behaves erroneously, but also fails to behave consistently when interacting with other nodes. As described in [6], Byzantine faults can be reasoned from Byzantine Generals Problem which expressed in terms of generals deciding on a war mission of attack or retreat. The generals can communicate with one another only by messengers. After observing the enemy, they must decide upon a common plan of action. Generally it is very difficult to overcome the Byzantine faults and most existing solutions address only some specific Byzantine failure.

To achieve high performance, and to overcome Byzantine faults in MSWSN, we suggest a fault tolerance algorithm that combines two consensus-based fault tolerance schemes. The first part of our algorithm is to adapt a simple error-detection scheme called Consensus-Checking. This consensus-checking scheme has been implemented in parallel programming systems in ByzwATCh [8]. In [8], the cluster heads implementing consensus-checking are called Initiators. The initiator implements challenge-response, consensus-checking to vicinity cluster heads. These vicinity healthy nodes would respond by sending the checking results back to the initiator. Once initiator receives the reply message, it will have a list of cluster heads software and hardware health statuses. According to this checking list, the initiator can use these health statuses and mark a grade parameter 'κ' to each cluster head. The node that has higher mark κ means it has better health situation. This scheme can be used to prevent the faulty nodes of reporting inaccurate data by hardware and / or software checking algorithm. This scheme can assist our consensus-based decision scheme to find the "healthy" nodes in the networks.

The second part of our consensus decision algorithm comes from the Consensus Theory [9]. This consensus theory involves general procedures, which summarize estimates from multiple experts decisions based on Bayesian decision theory assumption. This theory has a combination formula obtained by the consensus rules. Several consensus rules have been proposed. Among them, the most commonly used consensus rule is the Linear Opinion Pool (LOP) which has the following (group probability) form for the user specified information (land cover) class if data sources are used:

$$C_j(Z) = \sum_{i=1}^{n} \lambda_i P(w_j \mid x_j) \qquad (6)$$

,where C_j is consensus rules, j is indicate information classes, $Z = [Z_1, Z_2, \cdots, Z_n]$ is an input vector, $P(w_j \mid x_j)$ is a source-specific posterior probability and $\lambda_i's$ $(i = 1,2,\cdots,n)$ are source-specific weights.

The main contribution of our fault tolerance algorithm is to replace the λ_i by the κ parameters which we mentioned in the first part. The healthy parameter κ is to express quantitatively the goodness of source data, which are controlled by the sources.

To clarify our algorithm we provide the following example with the diagrams shown in Fig. 3 (a), (b) to show how it is working with our proposed routing protocol.

(a.) After deployment the network will start to organize as a virtual spider net zone as described in [1]. Each Nine CHs will be organized as a small ICHs for consensus checking. Each one of these nodes will be the initiator for a certain period of time and then its role changes periodically. The first node can be

chosen according to its ID so the first initiator is the node with the lowest ID then the next one, and so on. The initiator is responsible to challenge the other eight neighboring nodes to collect their health statuses. This is shown in Fig. 3(a) where CH with ID = 5 becomes an initiator.

(b.) The initiator CH will generate challenge data and broadcast it to the vicinity cluster heads. The neighboring nodes, upon receiving this challenge message, will execute consensus-checking, which uses the challenge list as an input to a computational checking algorithm that performs a series of checks to generate an output message. This checking list can have hardware and / or software checking items. These checking items can be designed according to user requirements.

(c.) Each vicinity ICH node performs a consensus-checking to assess if any of the nodes returned a result that differs from the expected result. After these cluster heads complete this consensus-checking, they will respond back to the initiator by sending a response message. The initiator will aggregate the response messages and register the node health status results in its memory, as shown in Fig. 3(b).

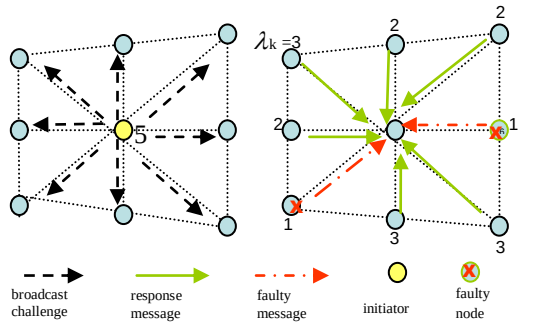

Figure 3(a). Challenge Message.

Figure 3(b). Response Message.

(d.) Based upon the results of this test, the initiator can select healthy cluster heads accordingly. The results can be quantified as parameter κ for each CH. The higher value means better healthy condition and vice versa. Once the healthy cluster heads have been listed, the initiator can run the consensus-based decision scheme using equation (2) after replacing λ_i for each cluster head by its κ, then the LOP will be executed to get the results $C_j(Z)$. The initiator will send back these results to each cluster head. In this algorithm, each node will have a threshold value T; this T can be defined by user or sensor application. If the cluster head's result λ_k is higher than *T*, it will have high priority to forward the data or event through the network. On the other hand, if the result is lower than T, it will become a standby node or only assist in message forwarding.

(e.) To maintain this small region spider-net zone network operation, our consensus-based algorithm also has a simple replacement scheme to reselect a new cluster head. When a cluster head fails or becomes standby node, its neighboring cluster head will be aware of that in the short term, thus it will broadcast spider-net zone construction message to the area. The first CH node replies to this message will become the new cluster head to replace the failed one. The diagram shown in Fig. 4 shows our fault tolerance algorithm.

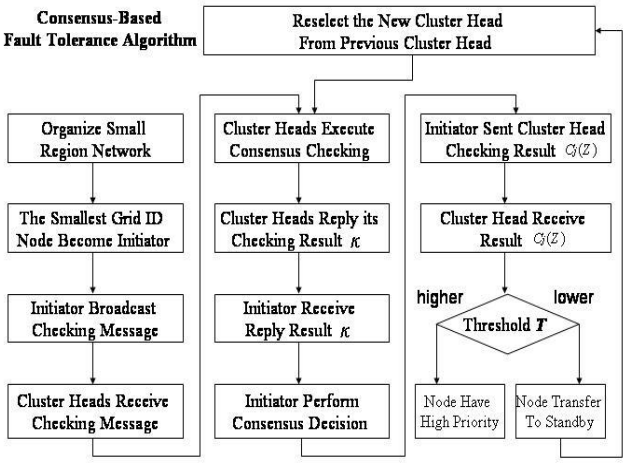

Figure 4. Consensus-based Fault Tolerance Algorithm Diagram.

Our consensus-based fault tolerance algorithm aims to support data source accuracy and reliability by applying both consensus-checking and consensus-detection schemes. It will help to quick discovery and replacement of any faulty CH node. As this detection and replacement of failure nodes is performed locally, the algorithm achieves energy-efficiency while not affecting the network communication. However, it increases communication reliability when it works with our magnetic coordinate routing protocol that aims to support data communication reliability and energy efficiency by using both parallel lines and magnetic polarity message forwarding schemes. An example to show how the two schemes interoperate is shown in Fig. 3(b) above. When the initiator has the reply messages from the CHs, it will be aware which cluster head is prone to failure or already in fault status, thus will replace it quickly. During that time, our magnetic-coordination routing will be using the redundant parallel routing capabilities, therefore, one side failure node will not influent network operation. When the two schemes work together, the network will achieve high reliable and efficient data dissemination in wireless sensor networks.

VI. PERFORMANCE EVALUATION

A. Performance Evaluation Setup

We evaluated FTSNZ using extensive simulation on the Georgia Tech Network Simulator (GTNetS) [13]. GTNetS has provided sensor network module, which is a scalable simulation tool, designed specifically to support large-scale sensor networks simulations. The design of the simulator closely matches the design of real network protocol stacks and hardware. This simulator applies C++ as its programming language and has an object-oriented design, which eases extensibility of existing simulation models. Here, we describe the simulation setup and the metrics

examined for the performance evaluation. In order to get the average behavior, we implemented FTSNZ in the GTNetS simulator sensor network module, our simulation consists of ($500 \leq N \leq 550$) sensor nodes including cluster heads in a 1200m × 1200m area or 4-5 nodes per 100m × 100m. The networks consist of ($100 \leq N < 150$) cluster heads which any of them could be a mole node. The Fig. 5 shows our FTSNZ simulation environment.

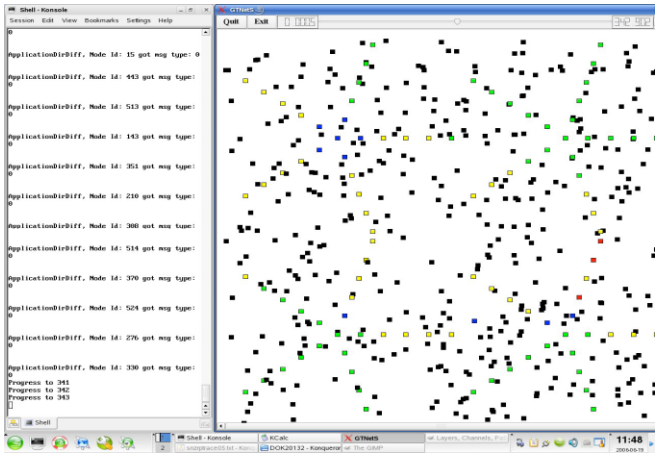

Figure 5. Simulation Environment.

The simulation time is 500 seconds. Each node has a radio range of 150m. We use few sources and one mobile sink randomly selected from 525 nodes in our simulation. The source sends data every 10 seconds, and the query is periodically sent every 20 seconds. The energy consumptions of transmitting and receiving are $0.66W$ and $0.395W$. To evaluate the performance of FTSNZ, we compare it to ODDD scheme. We use three main metrics to evaluate the performance of FTSNZ; namely, energy consumption, transmit successful rate and average end-to-end delay. Due to GTNetS has default one sink limitations, we only use one sink to compare with their algorithm. Energy consumption includes that of moles node competition, data dissemination, and the sink mobility. Our main concern is how to reduce the mole nodes competition for energy saving by considering the time limited to decide the mole nodes. Therefore, we focus on energy consuming for data dissemination and sink mobility management.

- Average Energy Consumption

In this experiment, we investigate the average energy consumption and set the number of sensor nodes is varied from 500 to 550 and one sink move by different speeds (0, 5, 10, 15, 20 *m/sec*). Both average sink refresh rate and average source update rate are set to 10 seconds. The node density has little influence on the energy per node in FTSNZ although more neighbors overhear data from a sender at high density. This is because there are more chances that better energy cost paths can be found in a higher density network. Fig. 6, FTSNZ shows the better performance in energy consumption. The reason that FTSNZ achieves less energy consumption is that the source can direct transmission event from CCHs to mobile sink. This eliminates event route through longer distance, which consumes energy in sensor nodes and cluster heads in the network.

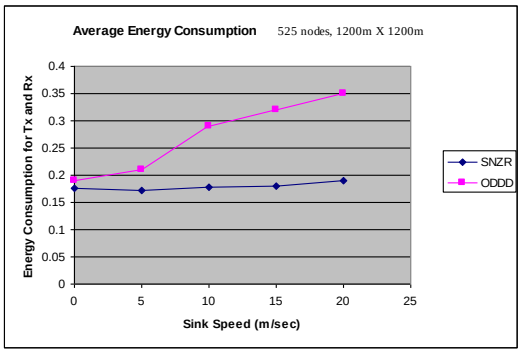

Figure 6. Average Energy Consumption.

- Average End-To-End Delay

In the following experiment, we investigate the average end-to-end delay as a function of the number of mobile sinks and their speed. Fig. 7 shows that end-to-end delay increases when mobile sink movement speed increase. In this experiment, the number of sensor nodes including cluster heads is varied from 80 to 90. Mobile sinks speed has influence on the end average delay per node in FTSNZ although the dynamic route changes from a sender send different route to mobile sinks when they moving at different speeds.

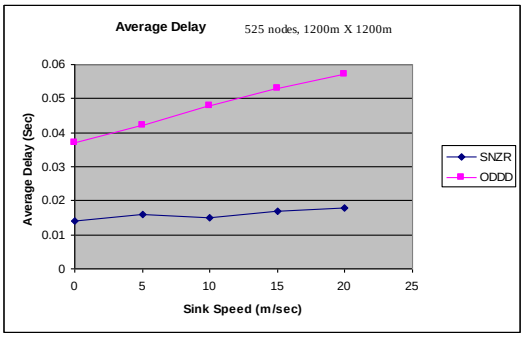

Figure 7. Average End-To-End Delay.

In this experiment, the number of sensor nodes including cluster heads is varied from 500 to 550. Speed of mobile sink move by different speeds (0, 5, 10, 15, 20 *m/sec*). As shown in Fig. 7, FTSNZ has a shorter delay than the ODDD. FTSNZ achieves lower average delay than the ODDD approach because of shortest routing paths and also consume less energy than the ODDD approaches.

- Average Packet Success Ratio

The success ratio is the ratio of the number of successfully delivered data messages that have been received by the sink. Our third experiment is to measure average success ratio for the different speed of mobile sink movement. As the default simulation setup, we have different speed of the mobile sink from 0, 5, 10, 15, 20 *m/sec.* All other simulation parameters are as specified as the default simulation scenario. As shown in Fig. 8, we observe that FTSNZ maintain high success ratio of above 90%. The average success ratio has a little bit decrease when mobile sink movement speed increase. Our result shows that

our scheme achieves better success rate than the ODDD scheme and obtains comparable success ratio with much less energy cost than the ODDD approaches.

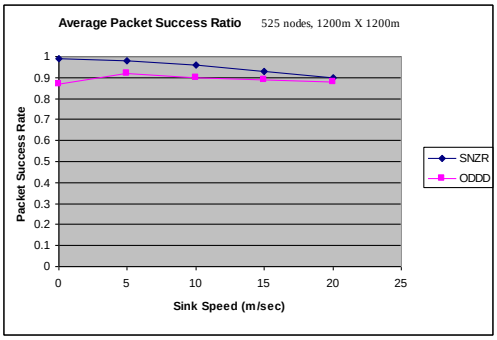

Figure 8. Average Packet Success Ratio.

B. Fault-Tolerance Evaluation

In this fault tolerance experiments, we deploy 525 nodes randomly over the region and one sink move by different speeds (0, 5, 10, 15, 20 *m/sec*). In the following paragraphs, we consider the effect of faulty nodes on the performance of our networks. Once we set up the network environment, then we vary the number of cluster heads failure from 5 to 20. These faulty cluster heads are randomly distributed in these 125 cluster heads that alternate the node failure rate fixed from 0.040 to 0.16. Fig. 9 ~ Fig. 12 depicts the numbers of cluster heads failure and mobile sink speed effect on packet success ratio of FTSNZ algorithm. The success ratio of approach is around 90% with original FTSNZ network fault tolerance algorithm. As the cluster heads failure rate continues to increase, the success ratio starts to fall down. However, comparing to the same environment without fault tolerance algorithm it increase up 54% packet success ratio with 20 cluster heads failure as shown in Fig. 9. Different from other routing algorithms for sensor networks, FTSNZ provide redundant links event dissemination for communication link fault tolerance in sensor network. This feature not only increases our average packet success ratio but also increase the communication overheads for event transmission.

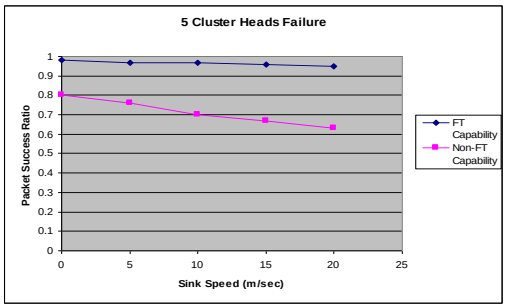

Figure 9. (Non) Fault Tolerance with 5 Faulty Cluster Heads.

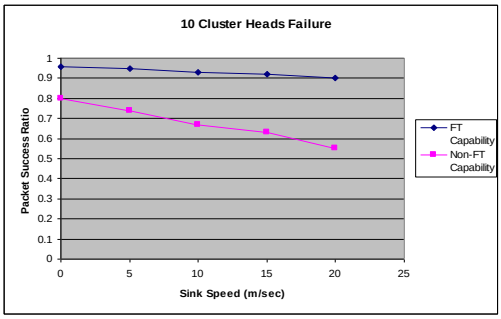

Figure 10. (Non) Fault Tolerance with 10 Faulty Cluster Heads.

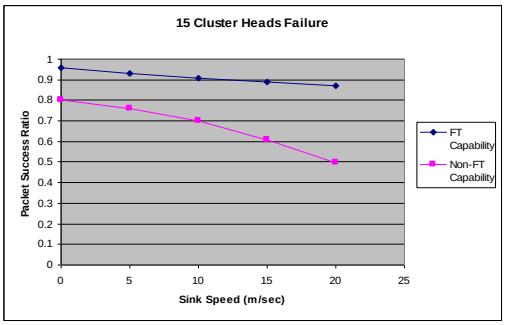

Figure 11. (Non) Fault Tolerance with 15 Faulty Cluster Heads.

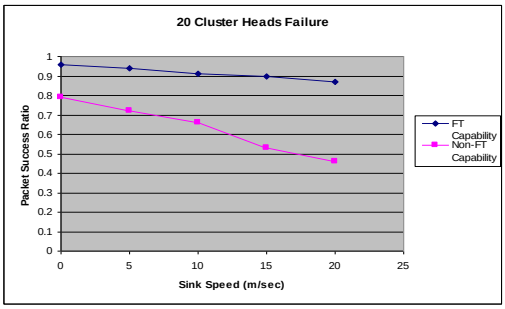

Figure 12. (Non) Fault Tolerance with 20 Faulty Cluster Heads.

VII. CONCLUSION

In this paper, we introduce a Fault Tolerance Spider-Net Zone (FTSNZ) routing protocol that combines spider-net zone routing and consensus-based fault tolerance to provides high performance, energy-efficiency and reliable communications in MSWSN (Mobile Sinks Wireless Sensor Networks). First FTSNZ has the potential as a significant efficient routing mechanism that can provide mobile or static sinks gathering sensing data with high-speed movement. FTSNZ uses a spider-net network topology with QoS pattern control to provide efficient routing, better guaranteed scheduling and data collision avoidance in wireless sensor networks. The simulation results show that our scheme achieves less energy consumption, less average delay and better packet success ratio than other related protocols. For our future work we will focus on other fault tolerance issues to adopt extremely reliable communication in wireless sensor networks.

REFERENCES

[1] S. Chang, M. Merabti, and H. Mokhtar: Spider-Net Zone Routing Protocol for Mobile Sink Wireless Sensor Networks, In Proceedings of the 2007 International Conference on Wireless Networks (ICWN), pp. 209-215, June 2007.

[2] J. Tateson and I. Marshall, "A Novel Mechanism for Routing in Highly Mobile Ad Hoc Sensor Networks", H. Karl, A. Willig Karl and A. Wolisz (Eds): Wireless Sensor Networks, LNCS 2920, pp. 204-217, 2004.

[3] S. Chang, M. Merabti, and H. Mokhtar, "EQEN - Efficient Quality of Service Framework for Mobile Sinks Wireless Sensor Networks", In Proceedings of the PGNET 2005, pp 26-31, Liverpool, UK, June 2005.

[4] H. Luo et al., "TTDD: Two-tier Data Dissemination in Largescale Wireless Sensor Networks", ACM/Kluwer Mobile Networks and Applications, pp.148-159, Atlanta, Georgia, USA, September, 2002.

[5] R. Kalva, J. Youn and C. Won, "ODDD: On-Demand Data Dissemination in Large Wireless Sensor Networks", 2005 IEEE 62nd Vehicular Technology Conference (VTC2005-Fall), Dallas, Texas, USA, September 2005.

[6] S. Kim, S. H. Son, J. Stankovic, S. Li, and Y. Choi. "Safe: A data dissemination protocol for periodic updates in sensor networks." In Workshop on Data Distribution for Real-Time Systems (DDRTS), May 2003.

[7] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A scalable and robust communication paradigm for sensor networks", In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, Boston, MA, pp. 56-67, August 2000.

[8] S. Lindsey, C. Raghavendra, and K. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," IEEE Trans. Parallel Distrib. System, vol. 13, no. 9, pp. 924--935, 2002.

[9] M. Gerla, G. Pei, X. Hong, and T. Chen, "Fisheye State Routing Protocol (FSR) for Ad Hoc Networks", November 2000, Internet Draft: draft-ietf-manet-fsr-01. txt, expired.

[10] Q Wen, Z Zhao and R Li, H Zhang, "Spatial-temporal compressed sensing based traffic prediction in cellular networks", In Proceedings of the 1st IEEE International Conference on Communication in China Workshops (ICCC), pp.119-124, August 2012.

[11] G. Caro, F. Ducatelle, and L. Gambardella. "AntHocNet: An adaptive natureinspired algorithm for routing in mobile ad hoc networks", European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking, vol. 16, no. 5, pp.443-455, June 2005.

[12] R. Rajagopalan, P. Varshney, K. Mehrotra and C. Mohan, "Fault tolerant mobile agent routing in sensor networks: A multi-objective optimization approach" Proc. of the 2nd IEEE Upstate New York workshop on Communications and Networking, Rochester, New York, November 2005.

[13] G. Riley, "Large-scale network simulations with GTNetS," In Proceedings of 2003 Winter Simulation Conference, pp. 676-684, 2003.

Shih-Hao Chang is a research fellow with National Taiwan University, Taiwan since 2011. His research interests include Wireless Sensor Networks, Packet Analysis and Malware Detection Algorithms, (M2M) Machine to Machine Communications. He received Ph.D. from John Moores University, Liverpool, UK in 2009. Prior to his Ph.D., he worked in industry for several years as an Engineer, Project Manager and IT Manager and has been directly involved in the development of several national and international projects for Lucent Technologies, Chunghwa Telecom. He is a member of IEEE. He His contact information is: Dr. Shih-Hao Chang, Research Fellow, Intel-NTU Connected Context Computing Center, National Taiwan University, Room F, 7F, Barry Lam Hall, No. 1, Sec. 4, Roosevelt Road, Taipei, 10617 Taiwan, Email: *shihhaochang@ntu.edu.tw*.

Ping-Tsai Chung is an Associate Professor with Department of Computer Science, Long Island University, Brooklyn Campus, where he joined since 2000. He has been serving as Chair of Department of Computer Science at LIU-Brooklyn since June 2004. He received Ph.D. in Computer Science from Polytechnic Institute of New York University (NYU-Poly) on January 1998. From 1997 to 2000, he has worked with AT&T and Lucent Technologies/Bell Labs for developing High Speed Network Management Systems. Previously, he has worked with Telecommunications Labs (TL) for a Broadband ISDN development project in Taiwan. His research interests are Network Computing, Intelligent Systems, Web Services and Biomedical Informatics. He has contributed several papers in above areas to the International Journals and Conferences. He is an Associate Editor for the Journal of Selected Areas in Bioinformatics (JBIO), Cyber Journals: Multidisciplinary Journals in Science and Technology (ISSN: 1925-2676). He is a senior member of IEEE Society, a member of ACM Society. His contact information is: Prof. Ping-Tsai Chung, Department of Computer Science, Associate Professor and Chair, Long Island University, 1 University Plaza, Brooklyn, New York 11201, USA, E-mail: pchung@liu.edu.