World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:6, No:8, 2012

Loop-free Local Path Repair Strategy for Directed Diffusion

Basma M. Mohammad El-Basioni, Sherine M. Abd El-kader, and Hussein S. Eissa

***Abstract*—**This paper proposes an implementation for the directed diffusion paradigm aids in studying this paradigm's operations and evaluates its behavior according to this implementation. The directed diffusion is evaluated with respect to the loss percentage, lifetime, end-to-end delay, and throughput. From these evaluations some suggestions and modifications are proposed to improve the directed diffusion behavior according to this implementation with respect to these metrics. The proposed modifications reflect the effect of local path repair by introducing a technique called Loop-free Local Path Repair (LLPR) which improves the directed diffusion behavior especially with respect to packet loss percentage by about 92.69%. Also LLPR improves the throughput and end-to-end delay by about 55.31% and 14.06% respectively, while the lifetime decreases by about 29.79%.

***Keywords*—**Attribute-value based naming scheme, data gathering, data-centric routing, energy-efficiency, locality, wireless sensor network.

I. Introduction

DIRECTED diffusion [1]-[6] is a data gathering and dissemination paradigm for Wireless Sensor Networks (WSNs) [7]-[10]. Directed diffusion is characterized by data-centric routing, application-aware processing, attribute-value based naming scheme, publish-and-subscribe information model, intermediate nodes might aggregate data, locality, robustness, and energy-efficiency. Directed diffusion uses a publish-and-subscribe information model in which an inquirer expresses an interest using attribute–value pairs. Sensor nodes, which can service the interest, reply with the corresponding data. The main elements of direct diffusion include interests, data messages, gradients, and reinforcements. An interest can be viewed as a query or an interrogation that specifies what the inquirer wants. A gradient can be thought of as a reply link pointing toward the neighbouring node from which the interest is received. The data message is the event description. Path reinforcement process is the process in which the sink can use multiple paths it established during the gradient setup phase to higher quality events by increasing its data rate [11]. The design space of directed diffusion is wide, fundamentally, directed diffusion is a general pattern for communications in WSN; it specifies the headlines for sensor nodes to communicate and achieve certain goals from its communication, but the details of this pattern, how these headlines are implemented, and how the resulted trade-offs among design goals are explored tolerate a lot of possibilities. This paper implements by simulation an instantiation from directed diffusion for tracking applications, so that, more than one sink can exist in the network in any place in the sensor field and each one can send a different request for data, the sensor nodes can sense more than one object from the same or different type of data. This instantiation is evaluated with respect to the loss percentage, lifetime, end-to-end delay, and throughput; then some modifications to this implementation are suggested to improve its performance, then the effects of these modifications on the paradigm are studied.

The rest of the paper is organized as follows; section II describes the details of directed diffusion instantiation implemented in this paper, section III evaluates this instantiation, section IV mentions some contemplations and recommendations for directed diffusion implementation, section V introduces enhancement to the used instance of directed diffusion and also it represents a comparison between two methods of path repair, repair by the sink and local repair, and finally section VI represents the conclusions and directions for future work.

II. Description Of Directed Diffusion Instantiation

This section describes the implementation of directed diffusion used in this paper, first it recites the types of messages transmitted during network operation and the tables needed by sinks and nodes to store the information required by the paradigm operation, and then it demonstrates the directed diffusion operation in some details.

A. The Messages Sent

The original interest message (Interest_Msg): represents the sink request for data and includes (sink ID, data type, area, Reporting Interval (RI), and Expiration time (Exp. time)).

The data message (Data_Msg): contains the source's sensed data and includes (data type, originator source ID,

B. M. Mohammad El-Basioni is assistant researcher at Electronics Research Institute (ERI), Computers and Systems Dept., Cairo, Egypt (phone: +2-010-303-557-3; fax: +202-333-516-31; e-mail: bbasioni@yahoo.com).

S. M. Abd El-kader is associate prof. at Electronics Research Institute (ERI), Computers and Systems Dept., Cairo, Egypt (e-mail: sherine@eri.sci.eg).

H. S. Eissa is associate prof. at Electronics Research Institute (ERI), Computers and Systems Dept., Cairo, Egypt (e-mail: hussein@eri.sci.eg).

originator source X-position, originator source Y-position, message sender ID, message sender X-position, message sender Y-position, timestamp, count of sensed targets, a specific value for each target of a certain property characterizes targets of this data type, and the distances of the originator source from each target).

The positive reinforcement message (Reinf_Interest_Msg): is the same as the sink original request for data (Interest_Msg) but with the reinforced values for the RI and Exp. time and with fields for the reinforced neighbor ID and the ID of the source reinforced by this message.

The negative reinforcement message (Neg_Reinf_Interest_Msg): is the same as the sink original data request (Interest_Msg) but with the negative reinforced values for the RI and Exp. time and with fields for the negative reinforced neighbor ID and the ID of the source negatively reinforced by this message.

B. Tables Held by Nodes

The interest cache (Interest_Cache): contains information about the data requests received at the node and it includes records for each interest include (the data type, area, the count of Gradients for this data request, which are the neighbors deliver the Interest_Msg to the node, the ID and the X-Y position of each Gradient, the maximum IR and Exp. time by which this data interest is requested by each Gradient, the count of data sources replied with the requested data through each Gradient, the ID of each source and the IR and Exp. time by which this data interest is requested from this source, the timestamp of the last data reply from this source to this data request, a flag to indicate whether this source is positively reinforced with respect to the IR and/or Exp. time through this Gradient for this data interest or not, the IDs of the sinks positively reinforced it if it is positively reinforced, and a flag to indicate whether this source is negatively reinforced with respect to the IR and/or Exp. time through this Gradient for this data interest or not).

The data cache (Data_Cache): it stores sources' data information and it includes records for each Data_Msg include (data type, area, the IDs of the source and sender of the Data_Msg, the X-Y position of the source, the timestamp and the arrival time of the Data_Msg, the sensed targets' information reported in the Data_Msg, and array (AllConveyNeighs) for the information of all the neighbors conveyed each Data_Msg in the time order they conveyed it; an awareness should be given for conserving the contents of this array of the first data message conveyed from each source if the Data_Cache is to be refreshed to reuse storage capacity).

The positively reinforced sources' table (Reinforced_Sources): the Reinforced_Sources table of a node saves the IDs of the data sources positively reinforced before through this node and the current and old index of the reinforced neighbor for each one of these sources in the AllConveyNeighs array of the first data message conveyed from each source.

C. Tables Held by Sinks

(Interests_Sent) table: this table stores the information of the Interest_Msgs which the sink sent, the source-neighbor pairs records which represent the sources replied to each interest and the sink's neighbors conveyed the first Data_Msg reply of each source, the timestamp of this Data_Msg, and a flag to indicate if this neighbor is positively reinforced for this source before.

(Sources_Replied) table: this table stores the information of the data sources sent data to the sink including ID, X-Y position, the timestamp and the arrival time of the latest Data_Msg received at the sink from each source, currently sensed objects of different data types and their corresponding distances from the source, the timestamp of the latest Data_Msg received at the sink from the source containing data about each object sensed by this source, and the time of the latest reinforcement message the sink sent to it for path repair.

(Sinistral_Sources) table: this table contains the information of the sinistral sources which didn't send any data to the sink for a time period greater than or equal to a specified period; this table includes fields for the ID, a flag to indicate if lastly a positive reinforcement was sent to this source for path repair, and if one sent, to which neighbor it was sent, flags to indicate if the source sent data in its last negative reinforcement interval, and if it sent, if the same reinforced neighbor for this source conveyed the source negative data. The first three fields are updated in each time of sinistral sources check, while the last two flags are updated during each interval of the negative data sending for each source.

(Sensed_Objects) tables: these are a set of tables, one for each object the sink was informed about it identified by its data type and an ID mapped to its unique value of the property used to characterize the same data type objects. Each table contains the timestamps in which the object data is received, the count of sources sensed the object in each timestamp (the maximum count reported is three even if the count of sources actually sensed the object in the timestamp is more), the arrival time of the third Data_Msg about this object received in each timestamp or the arrival time of the last Data_Msg if less than three Data_Msgs reported this object were received in the timestamp, and the information of these first three Data_Msgs including source ID, arrival time, convey neighbor ID, and the distance of the object from the source.

The sink may also store the information it received in different formats and in different tabular structures such that its representation becomes more useful and easier to be retrieved by the used applications whatever these applications are.

D. The Operation of Directed Diffusion Instantiation

The sink broadcasts Interest_Msg and constructs or updates the Interests_Sent table. If the sink received a Data_Msg, it constructs or updates the source-neighbor pairs records in the Interests_Sent table, the Sensed_Objects, and the Sources_Replied tables, then it checks whether the neighbor conveyed this Data_Msg is the first neighbor conveys data from this source, if yes, it sends a Reinf_Interest_Msg to it and marks it as positive reinforced for this source in the source-neighbor pairs records, if no, it sends a Neg_Reinf_Interest_Msg to it and marks it as negative reinforced for this source in the source-neighbor pairs records. The sink always does the check for cut paths (Sink Path Repair (SPR)) when it receives a Data_Msg; the sink does the check for cut paths to see if there is a problem in a path to a source, if this problem is a cut in the path so a path repair is required or it is just a breakdown in the path due to a certain condition and it may be a temporary breakdown and the path will return to be healthy and working after this condition passes, so a path repair isn't required; it does that by looping through the Sources_Replied table, if the difference between the incoming message timestamp and the timestamp of the latest Data_Msg of a recorded source is greater than a specified period measured in terms of the positively reinforced value for the RI and the time passed from the last Reinf_Interest_Msg sent to this source is also greater than a specified period set to be equal to the just mentioned period which measured in terms of the positively reinforced value for the RI plus another time period enough for the Reinf_Interest_Msg to reach the intended source, the source replies with the required data in the next sampling interval, and its reply reaches the sink, if this correct, it understands that the path is cut, so it adds this source to the Sinistral_Sources table, but also it doesn't send a Reinf_Interest_Msg for path repair until it found that after the source no longer sends data, at least one of its sensed targets is sensed by less than three sources to be sure that its reading is important for accurate specification for at least one target position, also, there is a probability that when the data from this source was stopped, it was sensing only one object and this object went out of its sensing range, i.e., there is no longer data to send about this object, so also before the sink sends the reinforcement message, it considers this probability, and sends the message if the count of targets lastly sensed by the source is greater than one, or equals one and this source is the only one senses this object from the object appearance time or it is not the only source but the last sources sensed this object sensed it from a specified period its pass indicates with a great probability that there is no information from any source about this object from a considerable period, so it is more probable that this object moved out the requested area or becomes inexistent for any reason, but also in this condition, if it found that the latest source still sends data but without this object data or it sends no data from that considerable time and the count of objects it sensed at that time equals one and the count of sources sense this object in this time is greater than three, it will not send the repair message; if it found this last source sensed this object from a considerable time, didn't send any data from that time and the count of objects it sensed at that time is greater than one, it will send the repair message to the neighbor which should be reinforced. Also the sink will not send the message if the last sources sensed this object sensed it not from a large period and the sensing circles corresponding to the sensing distances of these last sources sensed this object and the sensing circle corresponding to the maximum sensing range of any node and its centre is the sinistral source are separate, not coincident, and not contained in each other. If the sink sent a message, it marks this neighbor as positively reinforced for this source for all requests, and sets the time of last Reinf_Interest_Msg sent to this source to the current time, also it updates the Sinistral_Sources table to indicate that a Reinf_Interest_Msg for path repair was sent for this source and to which neighbor.

The decision of sending or not sending the Reinf_Interest_Msg for path repair at this time may be incorrect, but it is necessary to be taken at this time and it should be taken after a test to increase the probability of the decision correctness and accordingly decrease the probability of data loss of an important source, energy loss, and traffic increase for performing path repair for a path became useless, so the sending of data at negatively reinforced intervals is exploited to know the extent of the correctness of this decision by updating the Sinistral_Sources table during the negatively reinforced intervals for each source and making a test at the end of this interval if the sinistral source didn't send negative data through this interval it is deleted from the Sources_Replied table, if it sent and a reinforcement for repair was sent to it already, the sink checks if the same reinforced neighbor sent, it sets the neighbor to be reinforced if this source makes another problem to the same neighbor, if not, it increments the index of the neighbor to be reinforced, so that the next neighbor in the source-neighbor pairs records is sent the reinforcement message if this source makes another problem. If this source sent and the taken decision was not to send reinforcement for repair and from this test it was found that this decision is wrong, the sink will send the reinforcement immediately to the appropriate neighbor and update the tables should be updated.

The sink periodically checks for the occurrence of the case when all the links to all sources are cut and there is no longer any data received at the sink so the previous check for cut paths will not be performed because it is performed at the arrival of every Data_Msg to the sink and no reinforcement will be sent, so this case may continue for existence; the sink detects this case when it found that there is no data received from a specified period, so it repeats the previous check for cut paths but with respect to the current time not the

timestamp of an incoming message.

When the node receives an Interest_Msg, if it found that it is a new Interest, it adds it in the Interest_Cache with the information of the Gradient sent it, then, it broadcasts the Interest_Msg in a certain transmission range after updating the ID, X-position, and Y-position of the message sender with its own values; if the incoming Interest is recorded in the Interest_Cache, but the Gradient sent it is not recorded, it adds this Gradient with its information in the entry of this Interest and it broadcasts the Interest_Msg in a certain transmission range after updating the required fields if there is no another Gradient for this Interest has greater or equal value to the incoming Exp. time. If the Gradient sent the Interest is recorded, it updates the Exp. time value of this Gradient recorded in its existing entry in the Interest_Cache with its incoming value if the existing is smaller than the incoming, then it broadcasts the Interest_Msg in a certain transmission range after updating the required fields if there is no another Gradient for this Interest has greater or equal value to the incoming Exp. time. Fig. 1 and Fig. 2, represent the flow charts describe nodes' operation with respect to Data_Msg processing and Reinf_Interest_Msg processing in directed diffusion instantiation respectively.

III. Evaluation Of Directed Diffusion Instantiation

This section demonstrates the simulation setup including the employed general hypotheses, the used model for directed diffusion network evaluation, the performance metrics used in the evaluation, and the parameters' values.

A. The Scenario Assumptions

It is assumed that there is a number of sensor nodes are statically uniformly deployed in a two-dimensional square field, and the network is characterized by:

- There is one or more sinks are deployed inside the field.
- Sensor nodes are location-aware and non-rechargeable.
- The senor nodes can read and specify the value specific to each sensed object of a property of the objects, and all the sensors sense an object read the same value of its property, so that the objects of the same data type can be identified by their respective values of this property.
- The sensor nodes sense the object that enters their sensing ranges at the same time regardless of its distance from each node.
- For simplicity, it is assumed that the radio transmitter, radio receiver, and radio amplifier are the main energy consumers of a sensor node.
- The node can vary its transmission power depending on the distance to the receiver (as assumed in our previous work [12]).
- The Radio H.W. energy dissipation model [13] used is: to transmit a l-bit message over a distance d, the transmitter consumes

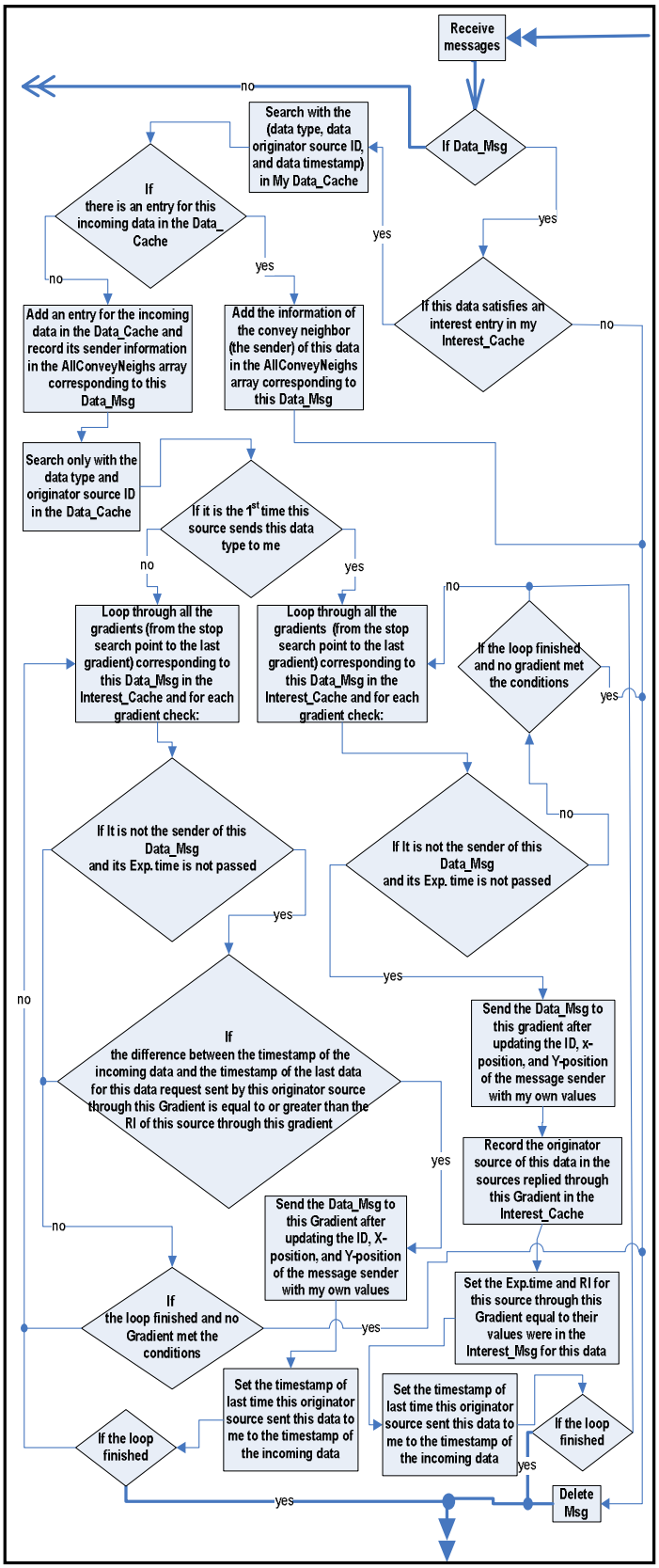

Fig. 1 Data_Msg processing by nodes in directed diffusion instantiation

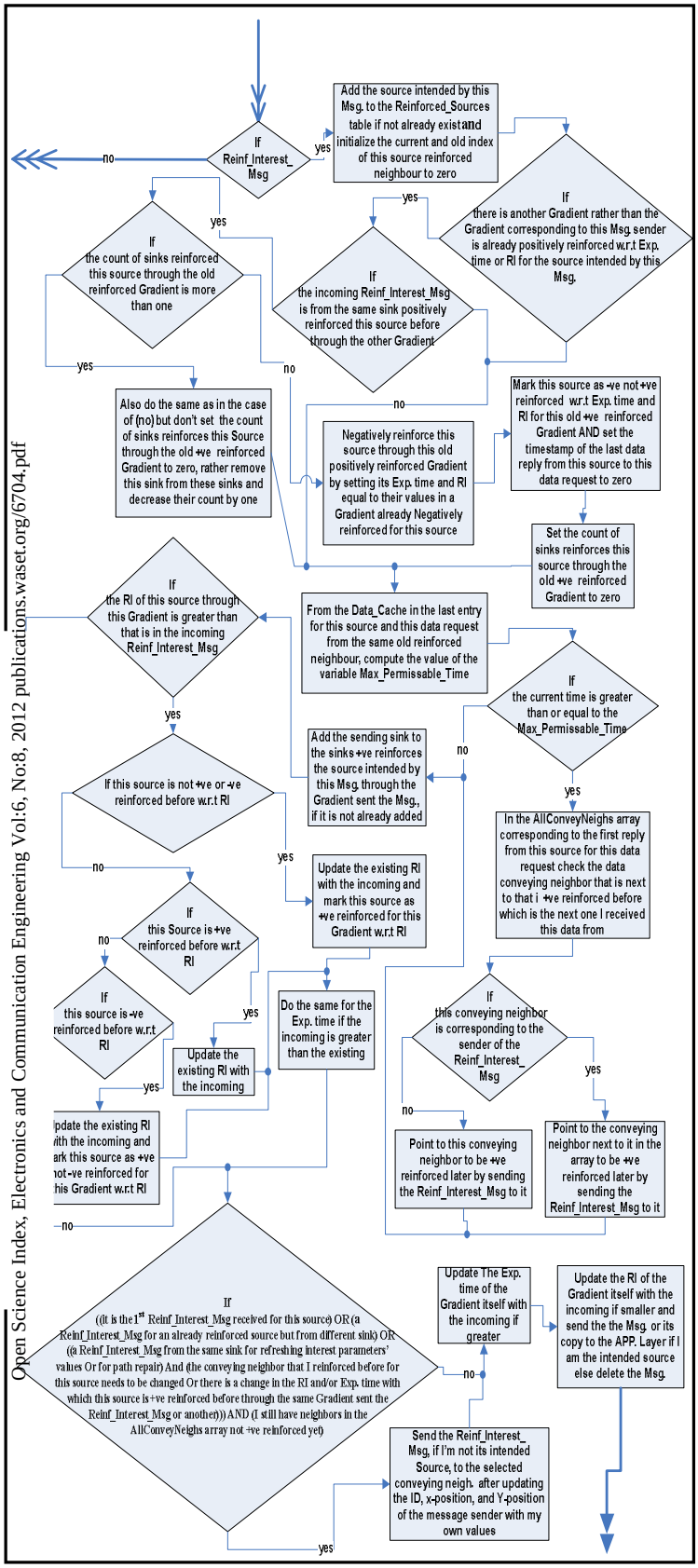

Fig. 2 Reinf_Interest_Msg processing by nodes in directed diffusion instantiation

$$E_{Tx} = \begin{cases} l \times E_{elec} + l \times e_{fs} \times d^2, d < d_o \\ l \times E_{elec} + l \times e_{amp} \times d^4, d \geq d_o \end{cases} \quad (1)$$

And to receive that *l*-bit message, the receiver consumes

$$E_{Rx} = l \times E_{elec} \quad (2)$$

Where e_{fs} and e_{amp} are the energies dissipated in the transmitter amplifier for either a free-space channel or a multi-path fading channel respectively. E_{elec} is the energy dissipated in transmitter and receiver electronics per bit.

- The signal propagation model used is the free space propagation model.
- All the sinks send the original Interest_Msgs with equal Initial RI but may be with different Exp. times.

B. The Network Model and Parameters' Values

The simulation runs were conducted using the discrete event simulator OMNeT++ [14] as the simulation platform to generate a network in 30 × 30 m^2 area in which sensor nodes are distributed statically and uniformly. A simple tracking application is used to study the protocol, the simulation model is suitable for monitoring and tracking different number of objects from the same type or different types, these objects may be mobile or static, there may be more than one static sink in the monitored area and each sink can send a request for data different than the others. For simplicity, in the simulation runs, it is assumed that there is only one sink node located at the point (27, 27) and it is assumed that it has infinite power and other resources. It sends an interest requests from the sensor nodes in the sub-monitored area [0, 10, 0, 10] the monitoring and tracking of four-legged-animals. At a certain point of time, an animal will enter this sub-monitored area followed by another animal, while the second animal remains in its place, the first animal moves with random steps in the sub-area until it exits it.

Each source node senses one or more objects and so each object is sensed by one or more sources. The sink receives the data from different sources, when it receives the first data from each source it reinforces the minimum delay path to this source by sending a positive reinforcement interest message to the first neighbor node conveys to it the data of this source, and accordingly this neighbor reinforces its neighbor node which first conveys to it the data from this source, and so on, until the positive reinforcement interest reaches the intended source (the processing used in simulation of the node's application layer for the received positive reinforcement interest is depicted in Fig. 3); the sink negatively reinforces its remaining neighbors which convey this source data later by sending to each of them a negative reinforcement interest message, each node receives this message resends it to all of its neighbors that conveyed this source data to it until the negative reinforcement interest reaches the intended source

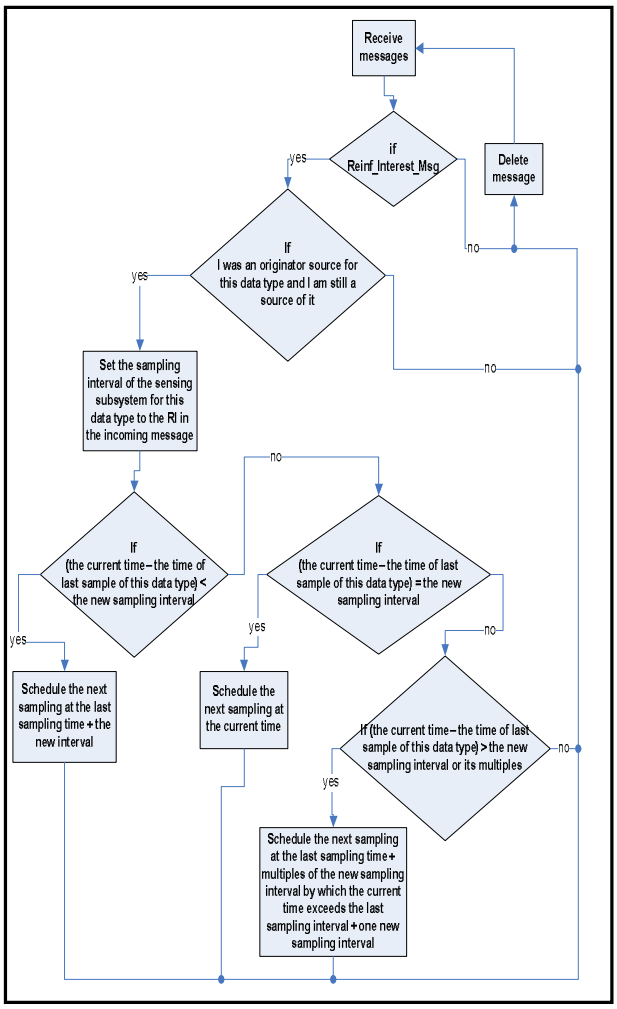

Fig. 3 Node's application layer processing of the Reinf_Interest_Msg in directed diffusion instantiation.

from its neighbors; the details of the implementation were more explained previously. It is assumed that the sink uses the data come from the different sources as soon as it reach in showing the place of the sensed objects in a screen; the more the sources sense the object, the more accurate the information about its place.

The source sends to the sink its ID, x-y position, the data type of the sensed object, a specific property of this object identifies it from any other objects of the same data type, the distance of the object from it, and the timestamp of this event, so if only one source senses the object, the information that the sink can extract from this data when it receives it will be that, the object was existing at a certain time in an unspecific point on the periphery of a circle its centre is this source and its radius is the distance reported by this source. If two sources sense the object, the information at the sink will be more accurate, the sink will understand that the object at this time was at one of two locations represented by the two points of intersection of the two sensing circles corresponding to these two sources reported distances from this object. If three sources sense the object, the sink can specify the location of the object at the reported time which is the point of intersection of the three sensing circles of the three sources, so the sink requires only the data of three sources at a certain time to specify accurately the location of the object. The parameters' values used in simulation are stated on Table I.

TABLE I
SIMULATION PARAMETERS

Parameters	Value
General Parameters	
Network Filed	(30, 30)
Nodes number	30~90
Broadcast range r	9 m
Sensing radius r_s	5 m
Sink position	(27, 27)
Initial energy	0.4 J
Data packet size	525 Bytes
Broadcast packet size	25 Bytes
E_{elec}	50 nJ/bit
e_{fs}	10 pJ/bit/m^2
e_{amp}	0.0013 pJ/bit/m^4
Threshold distance d_0	75 m
Original Interest parameters	
Data type	four-legged-animals
Area	[0, 10, 0, 10]
Reporting Interval	1s
Reporting Period	1800s
Positive Reinforcement Interest parameters	
Data type	four-legged-animals
Area	[0, 10, 0, 10]
Reporting Interval	0.5s
Reporting Period	1800s
Negative Reinforcement Interest parameters	
Data type	four-legged-animals
Area	[0, 10, 0, 10]
Reporting Interval	15s
Reporting Period	1800s

C. Performance Metrics

The directed diffusion implementation is evaluated with respect to the network lifetime, throughput, end-to-end delay, and loss percentage; these metrics are defined in simulation as follows:

1. Lifetime: the time (measured in seconds) from the network deployment until the network partitioning, i.e., the time at which the sink no longer receives data due to cutting of all paths to the sub-monitored area, cutting of all paths to the sources in this sub-area, or the death of these sources themselves.

2. Throughput: the data received at the sink through the network lifetime divided by the lifetime (measured in (bits/second)).

3. Packet loss percentage: the percentage of the packets lost from all the packets sent by all sources during the network lifetime, taking into account that the same packet may be received at the sink through more than one path, to the count of packets sent by all sources during the lifetime.

4. Average end-to-end delay: the average of the ripening time of each information at the sink in each timestamp, i.e., the average time of the arrival of the third source reading for each data sample of an object averaged all over the sensed objects through network lifetime, and this time is computed by subtracting the sample timestamp from the arrival time of the last source reading maximized to three readings if the sink received three or more different readings about the object in this timestamp. The time each packet containing information about a sensed object takes to travel from the source to the sink equals the count of the delays in the hops it crosses, and the delay of sending a packet from a node to its neighbor composed of the transmission delay, the propagation delay, and the queuing delay. The packet transmission delay equals the packet size divided on the data rate. The packet propagation delay equals the distance between the sender and the receiver divided on the speed of light, and the queuing delay of a packet equals the remaining time for the packet which is being transmitted at the entrance time of this packet in the transmission queue to finish transmission plus the count of the transmission delays of the packets waiting in the queue before this packet to be transmitted.

D. Results and Analysis

When a node or more in the positively reinforced path for a source dead, the positively reinforced data of this source ceases reception at the sink from this reinforced path, so the sink sends another Reinf_Interest_Msg for the same old reinforced neighbor, which represents the start of the cut path, if the time of negative reinforced data reception of this source didn't come yet or it came and it was found that this neighbor actually has healthy paths to the source and it delivers its data to the sink through them.

Each node receives this new reinforcement from the same sink from which it received the old reinforcement for this source and it finds that the remaining of the old reinforced path which represented with respect to it by its neighbor which previously reinforced for this source has a problem and didn't send data from a specified period, but it doesn't know if its neighbor is the dead node or another node in the remaining part of the path, so to compass this situation, because of the remaining part of the old path caused a problem the node changes the whole remaining path by assuming that its neighbor is the dead node, and it departs from this sinistral path by changing it by changing the neighbor to reinforce with the hope that the new selected path, which represented by each node the node reinforced reinforces it, warps around the problem infinitive. But it may be for all the nodes or it is usually for the sink neighbor nodes that the node's neighbor is not the dead node, so, it loses by choosing another neighbor not only one possible healthy path to the source through this neighbor, but more than one path through all the neighbor nodes of that neighbor, this causes the fast exhausting for the possible paths to the source available to each node so the node considered as a dead node doesn't deliver positive data for the source to the sink while there still exist healthy paths through them it delivers the source negative data to the sink, this effect manifests more when the nodal density is small, i.e., the number of neighbors for each node and for the sink also is small. But it may be and also it is usually that this new selected neighbor selects the node that caused the problem in the old path or another dead node because it is its first neighbor that conveyed the source data to it, and it can't determine if this neighbor is dead or alive because this is its first time to receive this Reinf_Interest_Msg and it didn't receive any positive data from this source before, this richly done when the interval during which the sink sends the Reinf_Interest_Msg to a neighbor exhausted all of its possible positively reinforced paths according to the used method for path repair increases, where the energy of some nodes during this interval is wasted due to wrongly sent Reinf_Interest_Msgs and lost Data_Msgs sent from the source.

When a neighbor to the sink exhausts all of its possible positively reinforced paths, the sink continues to send the Reinf_Interest_Msg to it when it stops to convey a source positive data even after the interval of this source negative data sending where the sink corrects the last decision taken for path repair because it found that the source still sends negative data through this neighbor, and this case may continue for a long period until all the paths through this neighbor are actually cut or this neighbor dies unless the source itself dies, then the sink corrects the decision by sending the Reinf_Interest_Msg to another neighbor when it found this neighbor no longer sends the source negative data. So we can deduce and say that:

- The smaller the nodal density, the smaller the number of nodes' neighbours, the smaller the count of a node's possible positively reinforced paths, the more the losses of data packets due to the fast exhausting of these paths, this causes the early surceasing of the source positive data although the source may be still alive.
- The larger the nodal density, the larger the traffic load, the larger the probability of early and larger nodes' death, the more the losses of data packets, especially that the number of sources in this case is larger and the death of a node may affect more than one source, i.e., loses more than one source data, so the loss percentage increases especially that the sources themselves die earlier, and although the count of a node's possible positively reinforced paths increases, but also the probability of a node to select a dead node in each new path increases.
- The larger the nodal density, the smaller the period in which the sink sends wrongly the Reinf_Interest_Msg for path repair to a neighbour exhausted all of its possible positively reinforced paths, the smaller the losses if the next neighbours the sink then reinforced them are alive, still have

possible positively reinforced paths, and the remaining paths they constitute has no dead nodes.

A mixture from these rational logic deductions makes the shape of the loss percentage curve to be as shown in Fig. 4, the curve with square markers corresponding to the legend "SPR", it approximately doesn't have a specific stable behavior, but the prevailing epithet of the curve is the increase with nodal density increase and also the difference between the curve points not considered to be large, so we can say the meant effect of the increasing in the count of possible positively reinforced paths for each node and the probable decreasing of the period in which the sink sends message to wrong neighbor when the nodal density increases is mitigated by a repugnant effect due to earlier nodes' death. Loop(s) may be formed in the reinforced path which aids in increasing loss percentage; when the node received the Reinf_Interest_Msg finds that the neighbor it will go to send the Reinf_Interest_Msg to it is itself the node sent it this Reinf_Interest_Msg, it doesn't select it and selects the next neighbor, but it can't detect if the selected neighbor is not the immediate sending node but it is an earliest sending node for this message in the anterior part of the path, so it sends to it again the message it sent causing the loop, the earliest sending node reacts to this by selecting another neighbour than it selected before which also greatly aids in the fast exhausting of the possible paths to the source available to each node. This loop may be repeated in one path to the same node or to others and this is more apparent in big nodal densities.

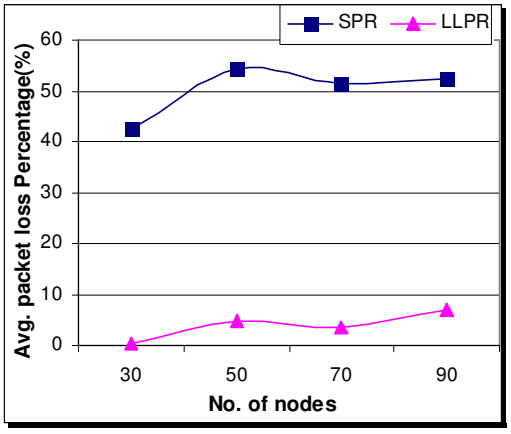

Fig. 4 Average packet loss percentage vs. number of nodes

The lifetime as shown in Fig. 5, the curve with square markers, decreases with nodal density increase because of the increase in the traffic load accordingly earlier sources' death and the earlier death of the critical nodes around the sink and around the sources. But it should be noted that, if the partitioning of the network in larger nodal densities is faster than in smaller densities in a working network, in smaller densities the network may be deployed partitioned, some nodes are separated from the others which may isolate the sink from targets' sources.

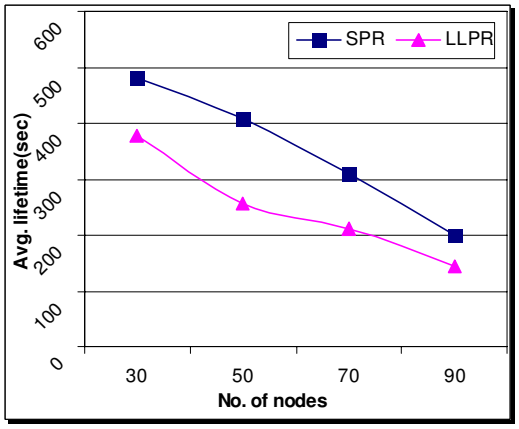

Fig. 5 Network lifetime vs. number of nodes

Due to the decreasing behavior of lifetime and the unstable behavior of loss percentage, the throughput also doesn't take a stable behavior but it can be said that it is approximately constant or tilts to declining as shown in Fig. 6, the curve with square markers, but it should be noted that, because the count of sources increases with nodal density increase the sink benefits from the smaller lifetime in the case of larger nodal density more than the larger lifetime in case of smaller nodal density; the average count of the precious specification of targets' position (three sources readings for the target distance in the same timestamp needed to accurately specify a target position) increases in larger nodal density.

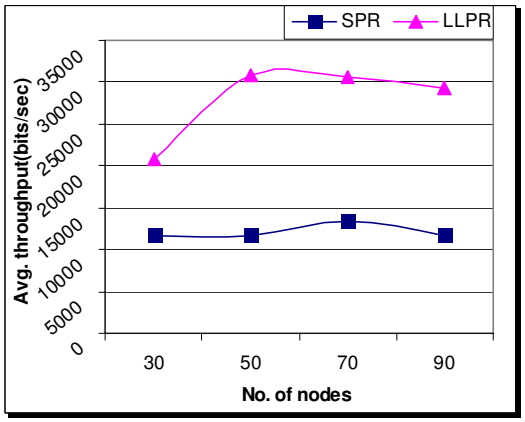

Fig. 6 Average throughput vs. number of nodes

As shown in Fig. 7, the curve with square markers, the average end-to-end delay increases with the nodal density increase. The sink positively reinforces a source by sending the Reinf_Interest_Msg to the first neighbor conveys its data and so does every node receives the Reinf_Interest_Msg, so, a low delay path has a small count of hops is firstly constituted between the sink and the source may approximately have the

same count of hops with nodal density increase, while in Reinf_Interest_Msg sending for path repair, the keenness of nodes for the new path only to move around the point(s) of problem in the old cut path makes them don't give a big concern to the length of the new path (the count of hops constitute the path), it is correct that the node selects the immediate next neighbor to the old one delivered to it the source data, but this results in longer path than the old one and the number of hops in the repair path increases with the increase in nodal density, especially that, not all the available neighbors to a node lead to healthy paths, so the healthy path finally it constituted may be very long.

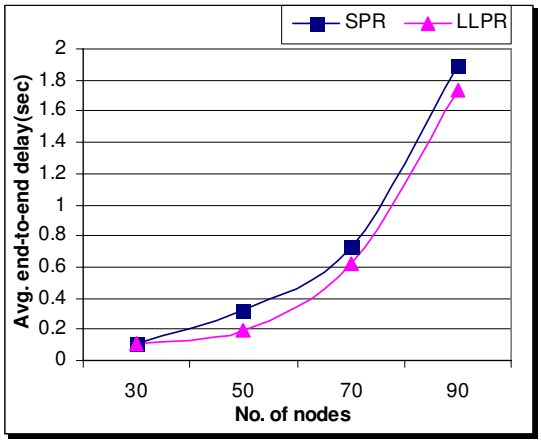

Fig. 7 Average end to end delay vs. number of nodes

IV. Contemplations, Notes, Comments, And Recommendations For Directed Diffusion Implementation

It is not necessary for the selection of the sink to the first neighbor conveyed reply from a specific source to a specific data request to be positively reinforced for this source, and accordingly the selection of every node in the reinforced path to the first neighbor conveyed this source's reply to this request, to result in the selection of the low delay path to be reinforced because the end-to-end delay doesn't depend only on the transmission delay and the propagation delay for the data packet through each hop, but it also depends on the queuing delay, so the physical layer begins to send the data message it received from the top layer only when it finishes the transmission of all the messages already reside in the transmission queue, i.e., the message is being transmitted and the messages wait in queue for transmission; the count of these messages depends on the types of messages the node is currently transmitting and the count of its neighbors to which it is transmitting unicast data messages for the first time it transmits this source data, this count will be changed when there will be a single path to the sink for this source, also the count of messages' types will be changed because in the most of the time the network operations turn to be only conveying data messages, so there will not be any control messages sent by nodes. Thus the sink may reinforce a path while there is another path conveys this source data faster, only all or some of the nodes in this faster path have a lot of neighbors, which increases their queuing delay in the first data conveying by which the sink and each node select the neighbor they will reinforce although this long delay will not exist if each of this nodes selects only one neighbor to send its data to it, i.e., if this faster path is reinforced rather than the already reinforced path. Even though, if we assume that the counts of neighbors of each node are approximately equal and as a result their queuing delays in the first data conveying are approximately equal, so the end-to-end delay depends only on the transmission delay and the propagation delay for the data packet through each hop, the reinforced path with the previous method also may not be the best with respect to the delay because when there is a reinforced path specific to each source for each sink, one or more of these reinforced paths may be criss-cross with others or a part or a whole of this path coincides on others, then the queuing delay will increase the delay of these paths upon which they were selected and it can not be determined whether there will be criss-cross or nearly coincident paths, i.e., it can not be determined whether there will be a queuing delay or not and if there will be a queuing delay what is its magnitude.

The sink sends a negative reinforcement message for a specific source to all of its neighbors that convey the first data message sent by this source after the first neighbor conveys it, starting from the second conveyer neighbor, but it should delay the start of negative reinforcement messages transmission (while it sends the positive reinforcement message to the first conveyer neighbor directly after its message reception) for a certain delay, may be set to be different and suitable for different initial nodal densities, to ensure that the flooding of the first data message is completely ended, so, each node received it from all of its neighbors convey this source first data message to it, thus when a node receives the negative reinforcement, it will send it to all of them (also it sends it to the neighbor it positively reinforced before only to broadcast it to its neighbors while remains positive reinforced); and also the sink delays the start of negative reinforcement messages transmission to ensure that the positive reinforcement for this source and other sources their first data received at the sink in the same time period sent and reached or is impending to reach them because the sending of negative reinforcements, especially the fact that the node sends the negative reinforcement to all of this source data conveyer neighbors as unicast messages, increases the queuing delay (especially at the sink, the sending of negative reinforcements for a source(s) may cause a big delay of positive reinforcement sending of others), so the reception of the positive reinforcement at the source will be delayed, thus the source remains sending data with the initial RI or with the negative RI if it received negative

reinforcements from all of its neighbors, which results in that the sink doesn't receive any additional data messages from this source for a considerable period of time until the source receives the positive reinforcement, replies in the appropriate time, and its reply received at the sink.

Because the sink sends for each source the positive reinforcement before the negative reinforcements and delays the start of negative reinforcements' transmission for a specific time, the positive reinforcement received at the source firstly (may be before the sink begins to send negative reinforcements or after it begins) and the source starts directly to send data according to the RI in the positive reinforcement and it doesn't wait for the negative reinforcement to be completely flooded through the network and received by all the nodes and accordingly to it by all of its neighbors, so the data message may be sent to more than one neighbor which haven't been yet negatively reinforced instead of only sending it to the reinforced neighbor, this data message was sent wrongly by these nodes, so it may not be received by the sink, but these nodes sent it and recorded the last timestamp of sending this data from this source of the Gradients it sent this data message to the timestamp of this data message, so, the start of the negative reinforcement reporting interval for specific source data (i.e., the time each node , whether it was the data source itself or an intermediate node received this source data and it should compare its timestamp with the last timestamp of this source data it forwarded to decide whether to forward it or not, sends this source data according to the RI which was in the negative reinforcement to its Gradients requested this data and this source sent it through them) may be different for the Gradients requested this data and this source sent it through them before. So, the sample of a source transmitted through multipath may be a sample with different timestamp than that transmitted through multipath of other sources even though they sent the initial sample with each other with the same timestamp, also the source itself may perform its multipath transmission with different samples with different timestamps (i.e., it sends a sample to the sink through some paths and another subsequent sample to be transmitted through the remaining paths), so the preparation period for the taken path repair decision correctness test for each source (which is equal to the period in which the sink receives the negative reinforcement data samples from a source for one negative reinforcement RI) should be set larger when the nodal density is larger because the count of sources increases and the traffic increases, so the variation in the start of the negative reinforcement reporting interval for different Gradients and different sources increases.

In the initial data propagation, if each node resends the same data message it received from its neighbors to the Gradients requested this data, this will represent wasted energy from the network, require large storage capacity from a node, and result in delay, so, each node resends the data message only once when it is received from the first neighbor conveyed it (this means that the sink receives the data message for a specific source only once from the neighbor conveyed it in each RI), but it records the IDs of all the neighbors conveyed this data message to it for the purpose of negative and positive reinforcements and path repair.

This method of benefiting from the negative reinforcement in checking the extent of correctness of the path repair decision taken lastly for a certain source may delay the sink in detecting the death of the sources and the neighbors that positively reinforced for them or in detecting the cut of all the paths lead to a source from the sink or from the neighbor already reinforced for this source to it, especially when one of these conditions such as source death done immediately after it sends its negative reinforced data sample. So the negative reinforcement RI should not be short to not burden the network in sending a lot of redundant data messages, and in the same time it should not be long to fasten the repair decision correction and also to fasten the reception of sources data at the sink if it didn't receive any data from them from a considerable period of time due to the cut of all the reinforced links between the sink and these sources and the failure of the path repair process to repair them.

The time period which passes without the sink or a node receives any message from a certain source, such that they can consider this source as a sinistral source and it should be checked if a path repair is required for it or not, is set based on the positive reinforcement RI, for example, it may be set to be equal to three positive RIs, but before a source receives any positive reinforcement, it remains for a period becomes larger with larger nodal density sending data with the original larger RI, in addition to the high traffic of sending large number of negative reinforcements and also positive reinforcements and data messages for different sources, a data sample with smaller timestamp of a certain source may be received at the sink lately and after the reception of a data sample with larger timestamp of another source, this causes the sink to wrongly consider this late source as a sinistral source and it sends to the same reinforced neighbor for this source another reinforcement for path repair (this reinforcement in highly traffic cases may be sent before the first one received by the source) and may accordingly cause intermediate nodes to wrongly choose other neighbors to reinforce rather than the still healthy old neighbors, which wastes the usages of some healthy paths in sending data by considering them cut off and as a result reduces the count of probable paths to sources, which may reduce even slightly the network lifetime. So it is preferable to make this time period which used to test for sinistral sources controllable by the sink and by the nodes or in other words adjustable to some specified different values according to an approximate prediction of the state of the network in terms of for example the time the network currently is working on or an estimate to the current traffic load, if that possible.

V. Some Enhancements To Directed Diffusion Implementation (The Proposed Technique (LLPR))

With the previous implementation, all the possible probabilities to the sink for a path between it and a source equal only the count of its neighbors' neighbors that conveyed this source data. These paths' possibilities may be increased by making each node follows the sink behavior in sending the repair positive reinforcement message to the same old reinforced neighbor, but this causes the message to be sent to the sinistral node from its neighbor node because it doesn't know about its death, so the same old sinistral path will be established. The previous problem will appear because of the absence of the knowledge of neighbor death from the node and the feeling of this node of losing a source data from a period compounds to the feeling of losing this source data of the other nodes which precede it in the old positively reinforced path from the precinct of the sink, in this case, each node as well as the sink remains to send the repair reinforcement to its dead neighbor unless it knows about its death, and it will not know about a neighbor death unless the neighbor informed it before its death. The local path repair for node death will help to solve this problem. The local repair strategy proposed in this paper (Loop-free Local Path Repair (LLPR)) is implemented by using two different types of control messages, Death_Alarming and Choose_Another_Reinforced_Neighbour messages; the node checks its energy level after receiving a Data_Msg, if it is below or equal to a threshold of energy suffices and slightly overflows its need for sending a Data_Msg and a Death_Alarming, it will begin firstly by broadcasting a Death_Alarming in its transmission range, if its energy level is above this threshold, it returns to check its energy level after each Data_Msg sent but against another energy threshold suffices and slightly overflows its need for sending two Data_Msgs and a Death_Alarming, then it follows the same just mentioned behavior. Also after the reception of a Reinf_Interest_Msg, the node will not forward this message to another node if it found its energy level is below a threshold abundantly enough at least for sending the Reinf_Interest_Msg and receiving two Data_Msgs to allow it changes its state in the reinforced path for the intended source from New to Old, if its energy is below, it broadcasts a Death_Alarming.

The loop in the path causes the cut of the path at the end point of the loop, so for solving this problem, the node in a reinforced path for a certain source takes two states New or Old, in the beginning all the nodes are in the state New, then it turns to the state Old if it received two Data_Msgs or more from a source just after it reinforced or re-reinforced for this source through a period smaller than the negative RI for this source; if the Old node accepts a new Reinf_Interest_Msg for the same source it again turns its state to New and so on (if the node found itself in an advance reinforced path not the afore-going reinforced path for a source, it should turn its state to New and delete this source from the Reinforced_Sources table). The Death_Alarming contains a flag to indicate the type of the death whether it is a death because of energy exhaustion (Reason 0) which is mentioned before or it is a death only with respect to some sources by exhausting all the neighbors that convey their data (Reason 1), in the case of Reason 0, the node clears the Reinforced_Sources table and the Death_Alarming contains the IDs of all the sources the node reinforced for and their count, in the case of Reason 1, the node clears the Reinforced_Sources table only from the sources it dead for and the Death_Alarming contains the IDs of them and their count (Death_Alarming is sent only once).

When a node receives a Death_Alarming it checks, if it is a Reason 0 alarm, it removes this alarm sender from the AllConveyNeighs arrays of all the sources it passes data for, then if it found that the alarm sender is a gradient for an Interest(s) it received, also it cancels this gradient by setting its Exp. time to the current time or zero or removing it at all; if the Death_Alarming is a Reason 1 alarm, it removes this alarm sender from the AllConveyNeighs arrays of only the sources recorded in the Death_Alarming which are the sources the Death_Alarming sender dead for, then if it found that the alarm sender is a gradient for an Interest(s) it received, it only makes the sources it reported in its alarm message expired for it. Then for each source recorded in the Death_Alarming, if the node is reinforced for this source, if the alarm sender is already its reinforced neighbor for this source and sends to it this source data from a small period and the reinforced Gradient for this source is not the alarm sender and the node is not the intended source itself and if the neighbor the node is going to reinforce it for this source is not the already reinforced Gradient for this source or the alarm sender, it sends a Reinf_Interest_Msg to the neighbor it is going to reinforce it which is indicated by the value of the current index in the Reinforced_Sources table which is at this time equal to the value of the old index of the neighbor to be reinforced for this source in the AllConveyNeighs array of the first conveyed data message from each source after removing the alarm sender from AllConveyNeighs arrays. If the neighbor the node is going to reinforce it for this source is the already reinforced Gradient for this source or the alarm sender, it increments its current index value in the Reinforced_Sources table, i.e., it sends the Reinf_Interest_Msg to the next neighbor for or instead of each sink reinforces this source for each interest corresponding to this source data with the parameters' values the source already reinforced by which. If after increasing the current index value, the AllConveyNeighs array becomes empty, the node broadcasts a Reason 1 Death_Alarming with all the sources it lost their data convey neighbors for them and it removes these sources from the Reinforced_Sources table.

The remaining part of solving the problem of loops

requires the Reinf_Interest_Msg to held two additional fields, one of them represents the count of hops from the sink of the Reinf_Interest_Msg initiator in the reinforced path for the intended source and the second represents the count of hops from the sink of the Reinf_Interest_Msg sender. The first field takes the value zero if the message is initiated by the sink for reinforcing the source or re- reinforcing it for path repair, and takes the value of the count of hops from the sink of any node if it initiates a path repair Reinf_Interest_Msg for this source. When a node receives a Reinf_Interest_Msg, it will not accept it to update the Interest entry and any other tables require updating and send the Reinf_Interest_Msg to the old or a new neighbor unless it found that it is the intended source of the message or it didn't reinforced for this source before or the incoming Reinf_Interest_Msg is for an old reinforced source but from different sink or it found that the source is already reinforced and the Reinf_Interest_Msg is from the same sink but it is in the New state and its count of hops from the sink is greater than the count of hops of the message sender or it is in the Old state and its count of hops from the sink is greater than the count of hops of the message initiator. Also a Reason 1 Death_Alarming is broadcasted if the node exhausted all the data convey neighbors for the intended source by the Reinf_Interest_Msg after it increments the current index value for the neighbor to be reinforced in the AllConveyNeighs array when the message is a reinforcement for path repair initiated by the sink and the already reinforced neighbor has a problem in sending the source data or the neighbor to be reinforced is the message sender. If the node didn't accept the Reinf_Interest_Msg, it sends a Choose_Another_Reinforced_Neighbour message to the Reinf_Interest_Msg sender to inform it that I am a member in the new reinforced path precedes you from the sink side so choose another node to reinforce for preventing loops formation. The Choose_Another_Reinforced_Neighbour is like in construction and dealing a Reason 1 Death_Alarming for the source intended by the Reinf_Interest_Msg, but it is sent only to the Reinf_Interest_Msg sender and it contains a new field to remind the sender about the count of hops from the sink of the message Initiator to use this count when resends the Reinf_Interest_Msg to another node.

The local repair is used to solve the problems resulted from nodes' death because it is faster, gives better results, little in consuming energy, and little with respect to traffic congestion especially that the path was working what prevented its working is the death of one or more nodes, so there is no need to repair the path except in the place of node death, by that at the time of repair, the ability to determine whether the reading of this source is important for better determination of at least one target at the current time at the sink is lost, so a path repair can be made for a source its data is not important for the sink at the current time, but the sink still needs this source data because it is important for it at a later time, although it is currently redundant, if some sources sense the object dead and the reading of this source became needed for better target specification, at this time the data of this source is available at the sink, the sink has no need to repair the old cut path of this source and waits its data to come especially if no local repair was made to the redundant source, the sink will have no information about this source death or living, so it will remain to send the reinforcement to it until the test it performs after each negative data sending, and also if the source path is cut without repair, the source will remain to send data through the connected part of the cut path losing its energy and loses other nodes' energy without any use. The interruption of a source data due to other factors such as node failure, node draft, and link cut due to bad weather conditions can't be made by local repair because it costs the node high energy consumption and high storage capacity and increases its operation complexity, so this task is still given to the sink with the same method it used except that if the sink decided to not send reinforcement for a source due to the moving of the object outside its range, it will not send it a reinforcement in subsequent tests even if it assess that the object is now in the source range again because according to the test performed, the intersection of the two sensing ranges doesn't mean that the tested source sense the object because the object my be in the other side of the other source that actually senses it. When the sink receives a Death_Alarming, it sends reinforcements for the reported sources in the message to other neighbors.

The behavior of the tested metrics for directed diffusion instantiation after using the LLPR strategy for node's death is shown in Fig. 4, Fig. 5, Fig. 6, and Fig. 7 where represented by the curve with triangle markers corresponding to the legend "LLPR", the loss percentage is decreased by about 92.69%, the throughput is increased by about 55.31%, the delay is decreased by about 14.06%, while the lifetime is decreased by about 29.79%.

VI. Conclusions And Future Work

This paper studies by simulation directed diffusion as a data gathering and dissemination paradigm for wireless sensor networks. Under some conditions and assumptions and based on the basics of directed diffusion explained in its paper, an instance from this paradigm is proposed to simulate the directed diffusion wireless sensor network for tracking applications for the purpose of understanding its operations, evaluating its performance, and auditing on some of the probable design mistakes which cause performance degradation or undesirable results. This paper also compares the directed diffusion performance in two cases, the first is the repair of the cut paths by the sink (SPR) and the second is the local repair of cut paths due to nodes' death (LLPR), in this case the nodes are given the responsibility of detecting, reporting, and dealing with the nodes' death, so that the point of cutting is specified accurately then the path repair is

performed at this point. Also LLPR works on preventing path loops formation. LLPR improves the Directed Diffusion behavior with respect to loss percentage, throughput, and end-to-end delay by about 92.69%, 55.31%, and 14.06% respectively, while the lifetime is decreased by about 29.79%.

The design space of directed diffusion is very wide; there may be a large number of its instantiations for each different application type by varying the methods used in positive and negative reinforcements, data aggregation, path repair, and interest propagation. So, its design can bear a lot of modifications and techniques to improve the performance and exploit its trade-offs, so it could be said that the directed diffusion performance can be improved not only to prevent or reduce the data messages loss during network lifetime and increase throughput for example, but to achieve design aims, it is aspired to reach them, and may be in the same protocol instance such as long increasing or constant lifetime and smaller decreasing or constant end-to-end delay with increasing nodal density to improve accuracy.

REFERENCES

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proc. 6th MobiCom*, Aug. 2000, pp. 56–67.

[2] K. E. Kannammal and T. Purusothaman, "New Interest Propagation Mechanism in Directed Diffusion Protocol for Mobile Sensor Networks," *European J. of Scientific Research*, vol 68, no. 1, pp. 36-42, 2012.

[3] A. A. Hady, S. M. Abd El-kader, H. S. Eissa, A. Salem, and H. M.A. Fahmy, " A Comparative Analysis of Hierarchical Routing Protocols in Wireless Sensor Networks," in: Jemal H. Abawajy, Mukaddim Pathan, Mustafizur Rahman, Al-Sakib Khan Pathan, Mustafa Mat Deris (Eds.), Internet and Distributed Computing Advancements: Theoretical Frameworks and Practical Applications, IGI Global, 2012, pp. 212-246.

[4] F. Dargahi, A. M. Rahmani, and R. Samadabadi, "A new clustered Directed Diffusion Algorithm based on credit of nodes for wireless sensor networks," Novel Algorithms and Techniques in Telecommunications and Networking, 2010, pp. 477-481.

[5] K.E. Kannammal and Dr. T. Purusothaman, "Evaluation of Directed Diffusion Protocol for Mobile Sensor Networks," International Journal of Engineering Science and Technology, vol 2, no. 6, pp. 2272-2277, 2010.

[6] N. Perwaiz and M.Y. Javed, "A study on distributed diffusion and its variants," in *Proc. 12th International Conference on Computers and Information Technology (ICCIT '09)*, 2009, pp. 44 – 49.

[7] I. F. Akyildiz, Wireless Sensor Networks, Series in Communications and Networking. John Wiley & Sons Ltd., 2010.

[8] S. Misra, I. Woungang, and S. C. Misra, Guide to Wireless Sensor Networks. Springer-Verlag London Limited, 2009.

[9] J. Zheng and A. Jamalipour, Wireless Sensor Networks: A Networking Perspective. John Wiley & Sons, Inc., 2009.

[10] A. Boukerche, Algorithms and Protocols for Wireless Sensor Networks. John Wiley and Sons, 2009.

[11] K. Sohraby, D. Minoli, and T. Znati, Wireless Sensor Networks: Technology, Protocols, and Applications. John Wiley and Sons, 2007.

[12] B. M. Mohammad El-Basioni, S. M. Abd El-kader, H. S. Eissa, and M. M. Zahra, "An Optimized Energy-aware Routing Protocol for Wireless Sensor Network," Egyptian Informatics Journal, vol 12, no. 2, pp. 61–72, 2011.

[13] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan, "An Application Specific Protocol Architecture for Wireless Microsensor Networks," IEEE Transactions on Wireless Communications, vol 1, no. 4, pp. 660 – 670, 2002.

[14] A. Varga, Omnet++ discrete event simulation system, the Technical University of Budapest, Department of Telecommunications (BME-HIT), retrieved from http://www.omnetpp.org/omnetpp/doc_details/2105-omnet-32-win32-binary-exe, 2005.