International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 3, March - 2013

SPIN: A Data Centric Protocol For Wireless Sensor Networks

[1]Amitoj Kaur, [2]Gursimrat Singh, [3]Jagroop Kaur
[1,2]*Student of M.Tech Computer Science, Punjabi University Patiala*
[3]*Assistant professor of Software Engineering, Punjabi University Patiala*

Abstract

Wireless sensor networks consist of large number of nodes that have sensing, computing and communication capabilities. It is basically an interconnection between large number of nodes that are deployed for system monitoring by measuring its parameters. Energy conservation is one of the main issues in WSN. Various protocols have been designed to overcome this obstacle. SPIN is one of those protocols. It is data centric protocol. In this paper we study traditional SPIN protocol, a modified SPIN protocol. Modified SPIN also known as M-SPIN is a energy efficient protocol. M-SPIN belongs to SPIN family protocol.

1. Introduction

Wireless Sensor Networks (WSNs) are composed of tiny, inexpensive sensor nodes with several distinguishing characteristics: they have very low processing power and radio ranges, permit very low energy consumption and perform limited and specific monitoring and sensing functions [1].

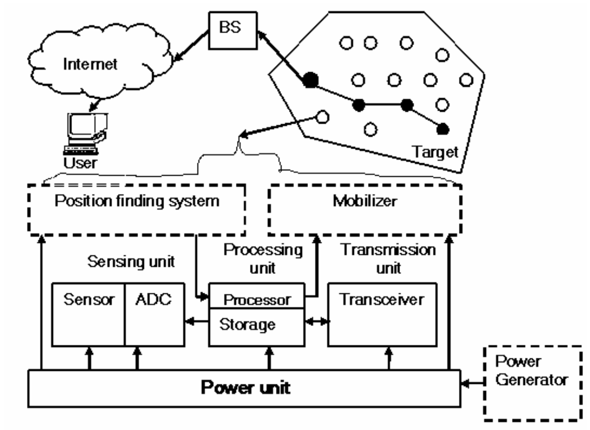

Fig.1 Components of sensor nodes and scattered sensor nodes

As shown in fig.1,each node consists of four components: power unit and central processing unit (CPU).sensor unit and communication unit. They are assigned with different tasks [3]. These nodes which are very small in size consist of sensing, data processing and communicating components. The position of these tiny nodes need not be absolute; this not only gives random placement but also means that protocols of sensor networks and its algorithms must possess self organizing abilities in inaccessible areas. However nodes are constrained in energy supply and bandwidth, one of the most important constraints on sensor nodes are the low power consumption requirements [2].

Sensor nodes not only carry limited but usually carry irreplaceable power sources and thus the main focus of sensor network protocol is primarily on power conservation. At the cost of lower throughput or higher transmission delay they must possess inbuilt trade-off mechanism that gives the end user the option of prolonging network lifetime [2].

Wireless sensor networks can also improve remote access to sensor data by providing *sink nodes* that connect them to other networks, such as the Internet, using widearea wireless links. If the sensors share their observations and process these observations so that meaningful and useful information is available at the sink nodes, users can retrieve information from the sink nodes to monitor and control the environment from afar [4].

We therefore envision a future in which collections of sensor nodes form ad hoc distributed processing networks that produce easily accessible and high-quality information about the physical environment. Each sensor node operates autonomously with no central point of control in the network, and each node bases its decisions on its mission, the information it currently has, and its knowledge of its computing, communication and energy resources. Compared to today's isolated sensors, tomorrow's networked sensors have the potential to perform their responsibilities with more accuracy, robustness and sophistication .

Several obstacles need to be overcome before this vision can become a reality. These obstacles arise from the limited energy, computational power, and communication resources available to the sensors in the network [4].

Energy: Because networked sensors can use up their limited supply of energy simply performing computations and transmitting information in a wireless environment, energy-conserving forms of communication and computation are essential .

Computation: sensors have limited computing power so may be unable to run some network protocols.

Communication: The bandwidth of the wireless links connecting sensor nodes is often limited, on the order of a few hundred Kbps, further constraining inter-sensor communication [4].

In this paper, we present *SPIN* (Sensor Protocols for Information via Negotiation), a family of negotiation-based information dissemination protocols suitable for wireless sensor networks. We focus on the efficient dissemination of individual sensor observations to all the sensors in a network, treating all sensors as potential sink nodes [4].

2. Data Centric Protocols

Data-centric protocols differ from traditional address-centric protocols in the manner that the data is sent from source sensors to the sink. In *address-centric* protocols, each source sensor that has the appropriate data responds by sending its data to the sink independently of all other sensors. However, in *data-centric* protocols, when the source sensors send their data to the sink, intermediate sensors can perform some form of aggregation on the data originating from multiple source sensors and send the aggregated data toward the sink. This process can result in energy savings because of less transmission required to send the data from the sources to the sink. In this section, we review some of the data-centric routing protocols for WSNs [5].

2.1. Flooding

In this we start with a source node sending its data to all of its neighbors. Upon receiving a piece of data, each node then stores and sends a copy of the data to all of *its* neighbors. This is therefore a straightforward protocol requiring no protocol state at any node, and it disseminates data quickly in a network where bandwidth is not scarce and links are not loss-prone.

It has three deficiencies which render in as an inadequate protocol. These are:

a) *Implosion:* In classic flooding, a node always sends data to its neighbors, regardless of whether or not the neighbor has already received the data from another source. This leads to the implosion problem, illustrated in Figure 2. Here, node A starts out by flooding data to its two neighbors, B and C. These nodes store the data from A and send a copy of it on to their neighbor D. The protocol thus wastes resources by sending two copies of the data to D [4]

b) *Overlap:* Sensor nodes often cover overlapping geographic areas, and nodes often gather overlapping pieces of sensor data. Figure 3 illustrates what happens when two nodes (A and B) gather such overlapping data and then flood the data to their common neighbor (C). Again, the algorithm wastes energy and bandwidth sending two copies of a piece of data to the same node. Overlap is a harder problem to solve than the implosion problem—implosion is a function only of network topology, whereas overlap is a function of both topology and the mapping of observed data to sensor nodes.

c) *Resource blindness:* In classic flooding, nodes do not modify their activities based on the amount of energy available to them at a given time. A network of embedded sensors can be "resource-aware" and adapt its communication and computation to the state of its energy resources [4].

2.2. Rumour Routing

Rumor routing is a logical compromise between query flooding and event flooding app schemes [6]. Rumor routing is an efficient protocol if the number of queries is between the two intersection points of the curve of rumor routing with those of query flooding and event flooding. Rumor routing is based on the concept of *agent*, which is a long-lived packet that traverses a network and informs each sensor it encounters about the events that it has learned during its network traverse. An agent will travel the network for a certain number of hops and then die. Each sensor, including the agent, maintains an event list that has event-distance pairs, where every entry in the list contains the event and the actual distance in the number of hops to that event from the currently visited sensor. Therefore, when the agent encounters a sensor on its path, it synchronizes its event list with that of the sensor it has encountered. Also, the sensors that hear the agent update their event lists according to that of the agent in order to maintain the shortest paths to the events that occur in the network [5].

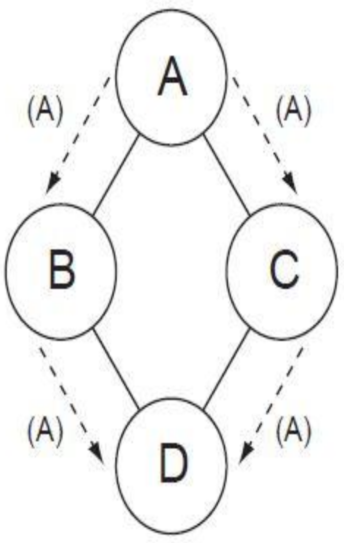

Fig.2 The implosion problem. In this graph, node A starts by flooding its data to all of its neighbors. Two copies of the data eventually arrive at node D. The system energy wastes energy and bandwidth in one unnecessary send and receive.

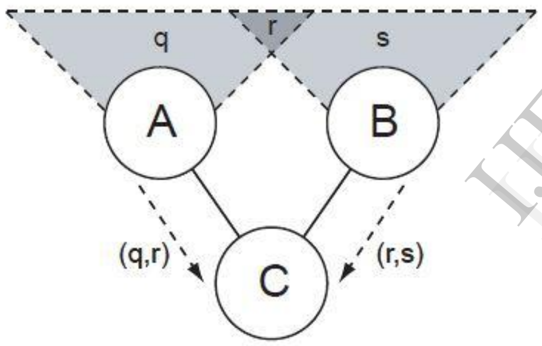

Fig.3. The overlap problem. Two sensors cover an overlapping geographic region. When these sensors flood their data to node C, C receives two copies of the data marked r.

2.3. SPIN

Sensor Protocols for Information via Negotiation was designed to improve classic flooding protocols. It fit under data delivery model in which the nodes sense data and disseminate the data throughout the network by means of negotiation. SPIN nodes use three types of messages for communication:

- ADV-When a node has new data to share; it can advertise this using ADV message containing Metadata.
- REQ-Node sends an REQ when it needs to receive actual data.
- DATA-DATA message contains actual data [3].

The SPIN family of protocols incorporates two key innovations that overcome deficiencies: *negotiation* and *resource-adaptation.* To overcome the problems of implosion and overlap, SPIN nodes negotiate with each other before transmitting data. Negotiation helps ensure that only useful information will be transferred. To negotiate successfully, however, nodes must be able to describe or name the data they observe. We refer to the descriptors used in SPIN negotiations as *meta-data* [4].

3. SPIN: Sensor Protocol for Information via Negotiation

The SPIN family of protocols rests upon two basic ideas. First, to operate efficiently and to conserve energy, sensor applications need to communicate with each other about the data that they already have and the data they still need to obtain. Exchanging sensor data may be an expensive network operation, but exchanging data *about* sensor data need not be. Second, nodes in a network must monitor and adapt to changes in their own energy resources to extend the operating lifetime of the system [4].

Heinzelman *et al.* in proposed a family of adaptive protocols called Sensor Protocols for Information via Negotiation (SPIN) that disseminate all the information at each node to every node in the network assuming that all nodes in the network are potential BSs [7],[8],[9]. This enables a user to query any node and get the required information immediately. These protocols make use of the property that nodes in close proximity have similar data, and hence there is a need to only distribute the data other nodes do not posses. SPIN is a negotiation-based information dissemination protocol suitable for WSN. It is based on the concept of metadata [1], [10].

3.1. Meta-Data

Sensors use meta-data to succinctly and completely describe the data that they collect. If x is the meta-data descriptor for sensor data X then the size of xin bytes must be shorter than the size of X, for SPIN to be beneficial. If two pieces of actual data are distinguishable, then their corresponding meta-data should be distinguishable. Likewise, two pieces of indistinguishable data should share the same meta-data representation. SPIN does not specify a format for meta-data; this format is application-specific. Sensors that cover disjoint geographic regions may simply use their own unique IDs as meta-data. The meta-data x would then stand for “all the data gathered by sensor x. A camera sensor, in contrast, might use (x,y,ϕ) as meta-data, where (x,y) is a geographic coordinate and ϕ is an orientation. Because each application’s meta-data format may be different, SPIN relies on each application to interpret and synthesize its own meta-data [4].

3.2. Spin Messages

SPIN nodes use three types of messages for communication:

- ADV-When a node has new data to share; it can advertise this using ADV message containing Metadata.
- REQ-Node sends an REQ when it needs to receive actual data.
- DATA- contains actual data [3]

Because ADV and REQ messages contain only metadata, they are smaller, and cheaper to send and receive, than their corresponding DATA messages [4].

4. SPIN Family

The SPIN family of protocol is made of four protocols, SPIN-PP, SPIN-BC, SPIN-RL, SPIN-EC and a modified SPIN (M-SPIN) [1].

4.1. SPIN-PP

It is a point to point protocol and is a simple handshake protocol for disseminating data through a lossless network. It works in three stages (ADV-REQ-DATA), with each stage corresponding to one of the messages described above. Also is known as SPIN-1protocol.

SPIN-PP, is optimized for a networks using point-to-point transmission media, where it is possible for nodes *A* and *B* to communicate exclusively with each other without interfering with other nodes. In such a point to point wireless network, the cost of communicating with *n* neighbors in terms of time and energy is *n* times the cost with the data of node A and send advertisements of the aggregated data to all of its neighbors(4). Second, nodes are not required to respond to every message in the protocol. In this example, one neighbor does not send an REQ packet back to node B (5). This would occur if that node already possessed the data being advertised. Although this protocol has been designed for lossless networks with symmetric communication links, it can easily be adapted to work in lossy or mobile networks. In lossy networks, nodes could compensate for lost ADV messages by readvertising these messages periodically, and nodes could compensate for lost REQ and DATA messages by re requesting data items that do not arrive within a fixed time period. Alternatively, the protocol might be augmented to use explicit acknowledgments. For example, whenever a node received an ADV message, it would send a request message (REQ) explicitly stating which advertised data it did and did not want to receive. In this way, the sender could differentiate lost ADV messages and ADV messages that had no corresponding requests for data, and thus re advertise only the lost ADV messages. Finally, for mobile networks, changes in the local topology can trigger updates to a node’s neighbor list. If a node notices that its neighbor list has changed, it can spontaneously re advertise all of its data [1]. In this protocol nodes make simple decisions and hence less energy is wasted resulting in energy consumption. Here each node needs to know only its single hop network neighbours. First, SPIN-PP can be run in a completely unconfigured network with a small startup cost to determine nearest neighbors. Second, if the topology of the network changes frequently, these changes only have to travel one hop before the nodes can continue running the algorithm [1].

4.2. SPIN-EC

This is a simple energy conservation protocol. SPIN-EC is an energy conserving version of SPIN-PP. When energy is plentiful, SPIN-EC nodes communicate using the same three-stage protocol as SPIN-PP nodes. When a SPIN-EC node observes that its energy is approaching a low-energy threshold, it adapts by reducing its participation in the protocol. In general, a node will only participate in a stage of the protocol if it believes that it can complete all the other stages of the protocol without going below the low-energy threshold. This conservative approach implies that if a node receives some new data, it only initiates the three-stage protocol if it believes it has enough energy to participate in the full protocol with all of its neighbors. Similarly, if a node receives an advertisement, it does not send out a request if it does not have enough energy to transmit the request and receive the corresponding data. This approach does not prevent a node from receiving, and therefore expending energy on, ADV or REQ messages below its low-energy threshold. It does, however, prevent the node from ever handling a DATA message below this threshold [1].

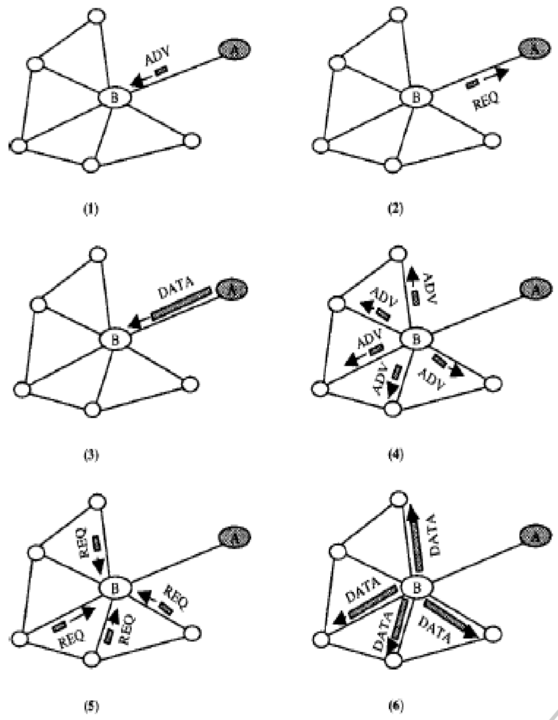

Fig. 4. The SPIN-PP protocol. Node A starts by advertising its data to node B (1). Node B responds by sending a request to node A (2). After receiving the requested data 3), node B then sends out advertisements to its neighbors (4), who in turn send requests back to B (5, 6).

4.3. SPIN-BC

This SPIN protocol is used in broadcast communication. SPIN-BC improves upon SPIN-PP for broadcast networks by exclusively using cheap, one-to-many communication. This means that all messages are sent to the broadcast address and thus processed by all nodes that are within transmission range of the sender [1]. During broadcast transmission the nodes communicate by a single, shared channel. In this when a node sends a message, it is received by every node in certain range of sender not bothering about the message destination. If the channel is sensed busy by the destination node then it waits until the channel becomes idle and then attempt to send the message. This channel has one disadvantage that when a node sends data all other nodes within the range of sender pay the price in terms of energy and time. However, there is one advantage of this that when a single node sends a message broadcast it is sent to all of its neighbours using one transmission. One-to-many network is 1/n times cheaper as compared to point to point network where n is number of neighbours of each node.

Like the SPIN-PP protocol, the SPIN-BC protocol has an ADV, REQ, and DATA stage, which serve the same purpose as they do in SPIN-PP. There are three central differences between SPIN-PP and SPIN-BC. First, as mentioned above, all SPIN-BC nodes send their messages to the broadcast address, so that all nodes within transmission range will receive the messages. Second, SPIN-BC nodes do not immediately send out requests when they hear advertisements for data they need. Upon receiving an ADV, each node checks to see whether it has already received or requested the advertised data. If not, it sets a random timer to expire. When the timer expires, the node sends an REQ message out to the broadcast address, specifying the original advertiser in the header of the message. When nodes other than the original advertiser receive the REQ, they cancel their own request timers, and prevent themselves from sending out redundant copies of the same request. The final difference between SPIN-PP and SPIN-BC is that a SPIN-BC node will send out the requested data to the broadcast address once and only once, as this is sufficient to get the data to all its neighbors. It will not respond to multiple requests for the same piece of data [1].Example of this protocol is shown in fig 5. Where node A advertise data and its neighbours checks whether they have received the advertised data or not (1).Three neighbours of A that are C, D, and E, do not have A’s data, and enter request suppression mode for different, random amounts of time. In our example C’s timer expires first, therefore C broadcasts a request for A’s data (2), which in turn suppresses the duplicate request from D. Though several nodes receive the request, only A,the originator of ADV packet, responds (3). After A sends out its data, E’s request is suppressed, and all send out advertisements for their new data (4).

4.4. SPIN-RL

SPIN-RL is a reliable version of SPIN-BC. It can disseminate data efficiently through a broadcast network, even if the network loses packets or communication is asymmetric. The SPIN-RL protocol incorporates two adjustments to SPIN-BC to achieve reliability. First, each SPIN-RL node keeps track of which advertisements it hears from which nodes, and if it does not receive the data within a reasonable period of time following a request, the node rerequests the data. It fills out the originating-advertiser field in the header of the REQ message with a destination, randomly picked from the list of neighbors that had advertised that specific piece of data. Second, SPIN-RL nodes limit the frequency with which they will resend data. If a SPIN-RL node sends out a DATA message corresponding to a specific piece of data, it will wait a predetermined amount of time before responding to any more requests for that piece of data[1].

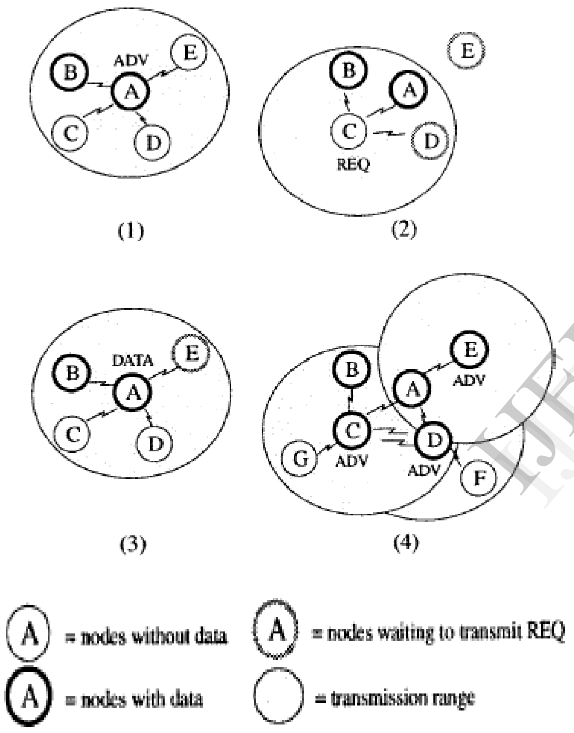

Fig 5. The SPIN-BC protocol.

5. Modified SPIN Protocol

One of the interesting fact is that consumption of energy does not only depends upon sensing data but also in transmitting or receiving the data to or from the neighbour nodes. So we can save a significant amount of energy if we can control the number of transmit and receipt of the data messages. Fig 6 shows a WSN network example. When an event occurs it divides the whole network into two regions A and B. Sensor nodes in region A are on one side and the sensor and sink nodes of region B lies to the other side. Upon receiving the data from the event node, sensor nodes of region A unnecessarily waste their energy in receiving or transmitting the data. The data will have to travel more hops in order to reach to the sink node, if they are sent via the nodes in region A. Thus, it is always desirable that when an event occurs the data is sent through the nodes in region B. This would save the energy that is spent for transmission of
a piece of data from an event node to the sink node. However, the existing SPIN protocols do not support such selective transmission. So a new protocol called modified SPIN or M-SPIN is proposed.

A few application such as alarm monitoring applications need quick and reliable responses. Suppose in forest fire warning system, quick response is needed before any disaster occurs. In this case, it is desirable that data must be disseminated towards the sink node very quickly. M-SPIN routing protocol is better approach for such type of applications than SPIN [11].

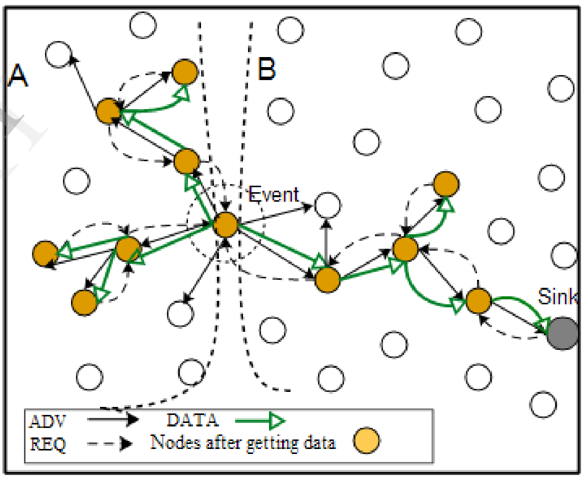

Fig 6. Data Transmission in WSN

M-SPIN has 3 phases:

- Distance discovery
- Negotiation
- Data transmission

The distance discovery phase is a new phase. In this we find the distance between the each node in the network to the sink nodes in terms of hops. Nodes having higher value of hops lie far away from the sink node. Negotiation is done on the basis of hop distance for sending the actual data. Then finally data is transmitted. We describe each of these phases in details in our next section.

5.1. Discovery Phase

Figure 7 shows the *Distance discovery* phase of M-SPIN. Hop distance is measured from sink nodes. Initially the sink node broadcasts *Startup* packet in the network with *type,nodeId* and *hop*. Here *type* means type of messages. The *nodeId* represents id of the sending node and *hop* represents hop distance from the sink node. Initial value of *hop* is set to 1. When a sensor node receives the *Startup* packet, it stores this hop value as its hop distance from the sink node in memory. After storing the value, the sensor node increases the hop value by 1 and then re-broadcast the *Startup* packet to its neighbour nodes with modified hop value. It may also be possible for a sensor node to receive multiple *Startup* packets from different intermediate nodes. Whenever a sensor node *b* receives *Startup* packets from its neighbors a_i, $1 \leq i \leq n$, it checks the hop distances and set the distance to the minimum, i.e.

$$\min \{ \forall \; h(a_i, b), i=1, n\}$$

Where $h(a_i, b)$ represents hop distances between the nodes a_i and b and n is the no. of neighbor nodes of node b from which it receives the startup packets.

This process is continued until all nodes in the network get the Startup packets at least once within the Distance discovery phase. After successful completion of this phase, next phase will be started for negotiation. Figure 8 shows some of the variables and structures used by the Distance discovery and Negotiation phase. StartupMsg structure contains three member variables. HopTable structure contains only one member called hop_t to store the hop value at each node. Figure 9 shows pseudo code for the Distance discovery phase [11].

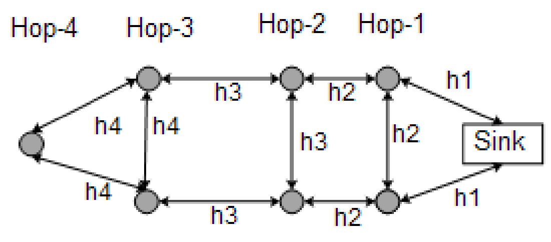

Fig 7. Distance discovery phase

5.2. Negotiation Phase

The source node sends an ADV message. Upon receiving an ADV message, each neighbor node verifies whether it has already received or requested the advertised data. Not only that, receiver node also verifies whether it is nearer to the sink node or not in comparison with the node that has sent the ADV message. If hop distance of the receiving node (own_hop) is less than the hop distance received by it as part of the ADV message (rcev_hop), i.e. *own_hop* < *rcev_hop*, then the receiving nodes send REQ message to the sending node for current data. The sending node then sends the actual data to the requesting node using DATA message.Fig 10 shows pseudo code for this phase.

As soon as a node gets data either from its own application or from other sensor nodes, it stores that data in its memory using the function *storepkt.* Also it uses *setCurrent* function to specify which data is presently residing in its memory.

When ADV message is received, then each receiving node first checks its record to ascertain whether it already has seen that data using the function *chkHistory*. Moreover, it calls *setDesired* to indicate which DATA packet it is waiting for.

The source nodes which receive the REQ use the function *getCurrent*. It helps to determine whether the received REQ is for the stored data specified by the *setCurrent* function for which the node has sent the ADV.

When a requesting node receives any data, it immediately checks whether the data is the same for which it has sent the request using *getDesired* function. The data packet contains the hop distance value along with the information about the event [11],[1].

```
Structures:

  typedef struct HopTable
  {
    uint16_t hop_t;
  };

 typedef struct StartupMsg
  {
    uint8_t type;
    uint16_t originNode;
    uint16_t counter;
  };

Along with these structures, M-SPIN also uses TinyOS
route message structure like TOS_Msg. It also uses
system variables like TOS_LOCAL_ADDRESS, and for
message broadcasting uses TOS_BCAST_ADDR address.
```

Fig 8. List of some structures and system variables used in Distance discovery and Negotiation phase.

```
Distance discovery() {
if (TOS_LOCAL_ADDRESS == 0) //node 0 is the sink node
 {
  ht.hop_t = 0;
  call Startup (1); // sink node sends Startup message
 }
if (Startup_TYPE == 3 && TOS_LOCAL_ADDRESS!= 0)
 {
  if (call chkHop (counter) == SUCCCESS)
   {
     call updHop (uint16_t hc) //increments the hop value
    }
  }
//sends the message to next neighbor nodes with updated hop value
call forwardHop (uint16_t hc)
}

Startup (uint16_t cn) {
StartupMsg *pSMsg =(StartupMsg *)&routeMsg.data [0];
uint8_t length = sizeof (StartupMsg);
pSMsg->msg_type = Startup_TYPE;
pSMsg ->node_id = TOS_LOCAL_ADDRESS;
pSMsg ->hop = cn;
fwdCount++;
if(fwdCount>=MAX_HOP){
  return;}
if (call SendMsg.send (TOS_BCAST_ADDR, length,
&routeMsg) == SUCCESS)
 {
   atomic sendRouteBusy = TRUE;}
}
```

Fig 9. Pseudo Code for Distance recovery

```
Negotiation ()
{
 // Advertisement for data broadcasts
 call ADV_Msg (origin, seq, sender, type);
 call waitREQ ();   //wait for REQ from neighbors
 if (received_packet_type = = ADV)
  {
   if(checkHistory (ADV.origin, ADV.seq) == SUCCESS)
    {
      if (own_hop < recv_hop)
       {
         call setDesired (ADV.origin, ADV.seq);
         // Request for desire data
         call REQ_Msg (origin, seq, sender, type);
       }
    }
  }
 if (received_packet_type = = REQ) {
   if (getCurrent (REQ.origin, REQ.seq) == SUCCESS)
    {
      if (fwd = = TRUE){
        mForward (storedPacket)
       }
      if (snd = = TRUE){
       mSend (storedPacket)
       }
    }
  }
}
```

Fig 10. Pseudo code for Negotiation

5.3. Data Transmission

Data transmission phase is same as SPIN-BC protocol. After request is received by the source node, data is immediately sent to the requesting node. If the requesting nodes are intermediate nodes other than the sink node then the *Negotiation* phase repeats. Thus, the intermediate sensor nodes broadcast ADV for the data with modified hop distance value. The sending nodes modify the hop distance field with its own hop distance value and add that in packet format of the ADV message. The process continues till data reaches the sink node [11], [1]. Fig 11 shows Negotiation and Data Transmission Phase.

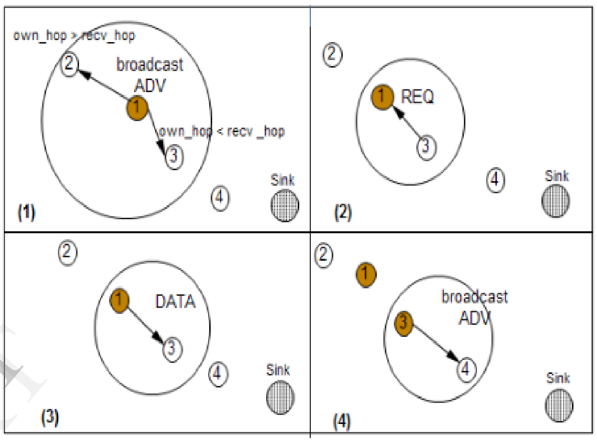

Fig 11. The M-SPIN protocol. (1) Node 1 starts advertising its data to all of its neighbors. (2) Node 3 responds by sending a request to node 1. (3) After receiving the request, node 1 sends the data. (4) Node 3 again sends advertisement out to its neighbors for the data that it received from node 1.

6. Conclusion

In this paper we have studied various data centric protocols used in Wireless Sensor Networks. The limitations of traditional protocols has been removed by the SPIN protocol family. SPIN is an energy efficient protocol for WSN. We also have another protocol M-SPIN which is modified version of the traditional SPIN protocol. M-SPIN helps to increase the enrgy efficiencies of the nodes. It is based on counting the number of hops between the sink and other nodes. It increases the energy efficiency by not sending the messages in the opposite direction of the sink node. However there are some limitations of this protocols. One is to calculate energy level at each node everytime makes the computation complex. Other major drawback of this is that some of the nodes may be used several times as compared to the other nodes which result in the energy dissipation of these nodes. The third major problem in M-SPIN is that the time needed to calculate the hop distance from sink node to all other nodes can be more than the time that can be tolerated. That is the nodes have to be quick enough to calculate the hop distance each time. So in future work can be done on these problems so that an energy efficient protocol is developed.

7. References

[1] Jyoti, Harkesk Sehrawat, Devendra Sharma, "Energy Efficient M-SPIN Protocol", *International Journal of Scientific & Engineering Research*, Volume 3, Issue 10, October-2012

[2] Neha Singh, Prof Rajeshwar Lal Dua, Vinita Mathur, "Wireless Sensor Networks: Architecture, Protocols, Simulator Tool", *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 2, Issue 5, May-2012

[3] Parul Tyagi, Ms. Surbhi Jain, " Comparative Study of Routing Protocols in Wireless Sensor Network", *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 2, issue 9, September-2012

[4] Joanna Kulik,Wendi Rabiner, and Hari Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks"

[5] Shio Kumar Singh, M P Singh , and D K Singh, "Routing Protocols in Wireless Sensor Networks –A Survey", *International Journal of Computer Science & Engineering Survey (IJCSES)* Vol.1, No.2, November 2010

[6] D. Braginsky and D. Estrin, "Rumor Routing Algorithm in Sensor Networks", *Proceedings ACM WSNA, in conjunction with ACM MobiCom'02,Atlanta,* GA, Sept. 2002, pp. 22-31.

[7] Jamal N. Al-Karaki, Ahmed E. Kamal**.** **"**Techniques in Wireless Sensor Networks: A Survey **"** *IEEE Wireless Communications*, December 2004, 1536-1 284/04.

[8] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *Proc. 5th ACM/IEEE Mobicom*, Seattle, WA, Aug. 1999. pp. 174–85.

[9] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Negotiation- Based Protocols for Disseminating Information in Wireless Sensor Networks," *Wireless Networks*, vol. 8, 2002, pp. 169–85.

[10] Kemal Akkaya, Mohamed Younis, **"**A Survey on Routing Protocols for Wireless Sensor Networks,**"** *Department of Computer Sciences and Electrical Engineering University of Maryland, Annual ACM/IEEE,* August 2000.

[11] Zeenat Rehena, Sarbani Roy, Nandini Mukherjee , "A Modified SPIN for Wireless Sensor Networks", *IEEE*, 978-1-4244-8953-4/11 2011.