*J. Basic. Appl. Sci. Res.*, 2(2)1972-1980, 2012
© 2012, TextRoad Publication
ISSN 2090-4304
**Journal of Basic and Applied Scientific Research**
*www.textroad.com*

# A New Energy-Efficient Reliable Data Transfer Protocol for Data Transfer in WSN

**Shabnam Zabihi [a,*]; Saeed Hadidi [b]; Maedeh Khani [c]**

[a] Young Researchers Club, Langaroud Branch, Islamic Azad University, Langaroud, Iran
[b] Department Of Electronic, Ramsar Branch, Islamic Azad University, Ramsar, Iran
[c] Department Of Electronic, Khomein Branch, Islamic Azad University, Khomein, Iran

## ABSTRACT

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions. Wireless Sensor Networks (WSNs) are generally energy and resource constrained. As the traffic pattern in most WSN applications is from sensor-to-sink, in-network data aggregation methods are employed for effective utilization of available resources. In order to reliably transfer the aggregated data packets, there arises a need for data transport protocols that provide reliability at the packet level. Existing protocols that provide reliable data transfer for sensor-to-sink traffic either provide reliability at the event level or are not energy efficient. To provide energy-efficiency while enhancing the packet level reliability, we propose an energy-efficient reliable data transfer protocol. This protocol provides packet level reliability by extending the concept of monitors and improves the energy-efficiency by employing duty cycles. The performance of the new protocol is tested. Experimental results show that the new protocol has significant improvement in packet delivery ratio and energy savings.
**KEY WORDS:** Data delivery, Wireless sensor networks (WSN), Data Transfer, Reliable.

## 1- INTRODUCTION

Wireless sensor networks (WSN) have generated tremendous interest among researchers these years because of their potential usage in a wide variety of applications. Sensor nodes are inexpensive portable devices with limited processing power and energy resources. A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding [1, 2].

WSN are generally deployed in harsh environments for habitat monitoring, military surveillance, and emergency response. Nodes in WSN are generally organized in a multi-hop topology that is either flat or hierarchical, and generally consists of one or more Base Stations (or sinks) and a very large number of sensor nodes scattered in physical space. The sensor nodes sense physical information, processes it and send the data to the Base Station. The Base Station in turn queries the sensor nodes for information and is responsible for collecting the data and relaying it to other networks. The primary traffic pattern in most WSN applications is sensor-to-sink, although the sink occasionally sends control packets to the sensor nodes. The upstream traffic pattern may be continuous delivery, event driven, query driven, or hybrid delivery. These types of traffic patterns from sensor nodes to Base Station limit the network scalability as the nodes closer to the Base Station encounter heavy traffic and deplete their energy rapidly. Heavy data traffic from the sensors results in congestion at the nodes close to the sink, causing bottlenecks for the network traffic. For effective utilization of resources and to support intermittent heavy traffic, techniques such as in-network data aggregation have been proposed. In these techniques, depending on the content, packets are forwarded, and, the packets containing correlated information are aggregated at intermediate nodes. As these aggregated data packets contain the reduced correlated data, a significant amount of data is lost if even a single data packet is lost and the data may not be reliably sent to the sink. Many transport protocols for data reliability in

***Corresponding Author:** Shabnam Zabihi, YoungResearchers Club, Langaroud Branch, Islamic Azad University, Langaroud, Iran.
Email: shabnam_zabihi@yahoo.com

WSN have been proposed. However, most current protocols provide reliability only at event level from sensor to Base Station and cannot provide reliable data delivery in case of in-network data aggregation.

To reduce energy consumption and provide reliable data delivery in applications that use in-network data aggregation, data transport protocols that are energy efficient and provide packet-level reliability must be designed. A sensor-to-sink data transport technique [3-5] that provides packet-level reliability and supporting in-network data aggregation was proposed in which the inactive sensors were dynamically initiated as monitors. By utilizing the information provided by monitors, more reliable loss detection can be achieved in case of sudden node failures. However, this technique is not energy efficient due to the idle listening behavior of the sensor nodes, which deplete energy frequently. Sensor nodes spend a considerable amount of time in monitoring the environment while only some spend a small portion of time to report sporadic events. The energy consumption in the idle listening state is very high compared to the transmission or receiving state. By employing duty cycles energy efficiency can be improved but it degrades the network performance in terms of latency. The network infrastructure has to be designed to support duty cycles and provide energy efficiency while providing packet-level reliability [6-9].

Our design extends the concept of monitors and is adaptable to employ duty cycles to provide energy-efficiency while improving the packet level reliability. As in [10], active and inactive nodes are chosen and from the subset of inactive nodes, subset of nodes called monitors is chosen. By choosing only a subset of nodes, energy consumption can be reduced. To further reduce energy consumption, duty cycles are employed for all the active nodes that take part in data transfer and the monitor nodes that support the reliable data transfer between active nodes.

The rest of this paper is organized as follows. Section 2 explains the motivation and objective of this paper and describes our Protocol Design. Section 3 presents the performance evaluation of the presented protocol. Finally, we conclude in section 4.

## 2- MATERIALS AND METHODS

In WSNs, resource constraints and wireless errors pose a major challenge in achieving reliable data delivery of packets. The data flows are from both sensor nodes to sink (upstream) and sink-to-sensor nodes (downstream). The predominant traffic pattern is from sensor nodes to sink in which the sensor nodes forward the sensed data to the sink. As most networks use the traditional hop-by-hop mechanism to forward the data, only those nodes that forward the data are responsible for maintaining reliable transport of the packets. Depending on the application requirements, reliability is ensured at hop level or from end-to-end. Since in WSNs, many sensors may detect the same event and try to forward the data to other nodes, data may be redundant, which degrades the performance of the network by increasing collisions, delay, and energy consumption. To reduce redundancy in forwarded packets, techniques such as Data Aggregation are used wherein packets are collected at intermediate nodes and the correlated data is forwarded from one node to another. The packets thus forwarded must be reliably delivered as it contains the correlated data and loss of a single packet would result in a huge amount of data loss. This necessitates the protocols that were designed for these kinds of applications to provide reliability at the packet level.

**Assumptions:** Considering networks that use in-network data aggregation, the following assumptions are made in the proposed protocol:

• The network is densely deployed to report any event to the base station.

• The sub setting of the network is already done and nodes that are part of data forwarding are identified and divided into active and inactive sets of nodes.

• All the nodes know the status of their one-hop neighborhood node (active/inactive) by local broadcast mechanisms.

• For simplifying the explanation, the network deployment does not have any physical holes and the outer boundary is identified.

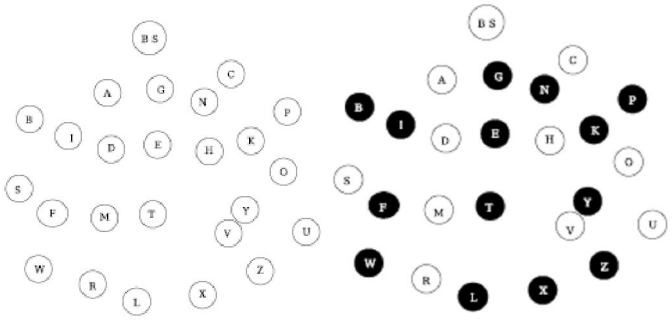
### 2-1- Configuration of Reliable Data Transfer

Before sensor nodes report any events to the base station using our reliable data transfer protocol, all the sensor nodes must follow some initial setup in order to make themselves send the data and make sure the data is reliably delivered to the base station. To provide packet-level reliability and identify hop levels for employing duty cycles, a data-gathering tree is constructed. The initial setup process consists of the following stages:

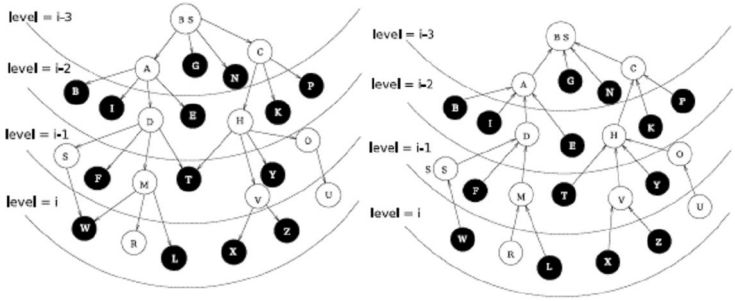
- Choosing a subset of nodes and identifying active/inactive nodes
- Constructing a data-gathering tree
- Establishing active data paths with the base station

Due to redundancy in the deployment of sensor nodes, carefully choosing only a subset of nodes from the entire network for data forwarding reduces the energy consumed by the redundant nodes. This helps in energy savings.

Also, as the number of nodes are reduced, collisions during data traffic are contained and the network performance is improved in terms of throughput. Congestion in the network will also be reduced as the network contains fewer nodes for data transfer, which reduces the average latency in forwarding the data to the Base Station. For the nodes that are part of this subset and actively participating in data forwarding, the status is set as active; and, for the remaining nodes, status is set as inactive. Each node broadcasts its status (active/inactive) message, *NODE-STATUS-MESG*, and all the nodes that receive this message record this node status. This way, all the nodes maintain the nodes that are active/inactive in its neighborhood and will use this



a) Wireless sensor network b) subsets of nodes



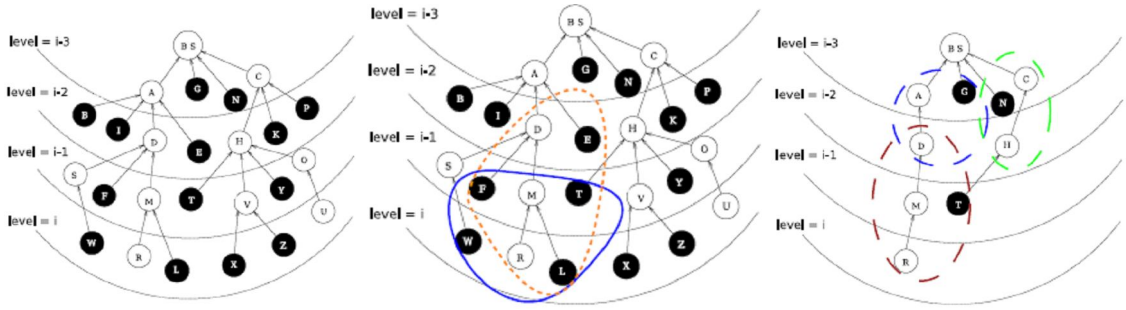

c) Data gathering three setup d) data path

Figure 1: Data Gathering Tree

Information in *monitor* configuration. All active nodes take part in Data forwarding and the remaining nodes named as inactive nodes do not take part in data forwarding.

To better illustrate the initial setup consider the Figure 1. The WSN is represented by Figure 1.a, which consists of entire sensor nodes in the network. Node BS represents the Base Station. Inactive nodes are represented in dark shade and active nodes are in light shade in the Figure 1.b. Nodes *{A,C,D,H, S,M, V,O,R, U}* form an Active set and *{G,N,B, I,E,K, P, F, T, Y,W, L,X,Z}* form an Inactive set.

To form the data path for all the active nodes, the Base Station initiates and broadcasts a *FORWARDER-REQ-MESG* and all the nodes that receives this message set their *forwarder* as Base Station and set their corresponding hop distance from the Base Station. Figure 1.c shows the construction of the data gathering tree; the arrows indicate the direction messages are sent. All the active nodes that receive the *FORWARDER-REQ-MESG* relay the message to nodes that are away from Base Station. Inactive nodes on the other hand just set their hop distance from the Base Station and configure the sender of *FORWARDER-REQ-MESG* as its *forwarder*. This mechanism continues until all the nodes in the network are configured with a *forwarder*. If a node already has a *forwarder* configured, the node that is closest to the Base Station is chosen as *forwarder* and its hop distance is configured accordingly. This way, all the active nodes establish an active data path with the Base Station. Inactive sets of nodes do not take part in relaying *FORWARDER-REQ-MESG* down the network, and thus do not take part in the active data path setup. With this method of setting up the data path, all nodes, irrespective of their exact locations, virtually reshuffle themselves depending on hop distance and a data-gathering tree with multiple hops is constructed. The rings in the Figure 1.c represent the hop level from the Base Station.




a) Data path b) Common inactive neighbor c) Configured monitors

Figure 2: Monitor Configuration

**Monitor Configuration:** After the data paths are setup as in Figure 2.a, neighboring inactive sets of nodes are considered to configure *monitors* for the nodes in the active data path. Every link between two active nodes will have an inactive node that acts a *monitor*. To reduce the collisions during *monitor* configuration and effectively configure a minimal set of *monitors*, this process is done in two stages. In the first stage, all the nodes that are at even hop distance from the Base Station are configured with *monitors*. For nodes that are at odd hop distance from the Base Station, *monitors* are configured in the second stage. As the process of configuring *monitors* is done only for the nodes that do not have a monitor already, the *monitor* configuration can be done in either order mentioned above and it results in the same number of *monitors* configured. This can be interchangeable as it would not alter the behavior of the network while forwarding data. Each of the above mentioned stages will follow the following steps for configuring *monitors*. This process can be better explained with an example. Consider the active data path $\{R \rightarrow M \rightarrow D \rightarrow A \rightarrow BS\}$.

Initially, nodes at even hops, *D* and *R*, will configure the *monitor* and, later nodes at odd hops, *A* and *M*, will configure the monitor only if a monitor is not configured already. Node *R* requests its *forwarder M* to share its one-hop inactive neighbors along with its *forwarder* information by sending a *NEIGHBOR-REQUESTMESG*. The *forwarder* node *M* replies back to *R* with the neighbor inactive node set *{E, F, T, L}* and its configured *forwarder* as *D* in the *NEIGHBOR-REPLY-MESG*. The neighborhood of node *M* is shown in dotted region in the Figure 2.b. Node *R*, after receiving the reply from *M*, would compare *M*'s inactive set of neighbors with its inactive set of neighbors *{F, T,W,L}*, shown in solid line region in Figure 2.b. Common inactive set of nodes are chosen, which are *{F, T, L}*. If multiple inactive nodes are common, node *R* randomly chooses one of them as a candidate *Monitor* node to monitor the link between *R* and *M*.

Let *T* be the randomly chosen *monitor*-candidate from the set *{F, T, L}*. Node *R* sends a *REQUEST-FOR-MONITOR* message to node *T* with the information o its *forwarder M* and *M*'s *forwarder* as *D*. It is possible that node *T* could receive multiple messages from other sets of nodes in its one-hop neighborhood. To give a fair chance for all the nodes, node *T* waits for a certain amount of time to receive all the *REQUEST-FOR-MONITOR* messages from its neighborhood. After this wait time, node *T* checks the possibility of acting as *monitor* for the requesting node and its *forwarder*. In this case for monitoring the link between nodes *R* and *M*, if node *T* receives multiple requests, after the elapsed wait time, it randomly chooses one node from the set of requested nodes and checks the possibility to act as *monitor* for that node and its *forwarder*. As node *T* has the information of the entire link $R \rightarrow M \rightarrow D$, it checks the possibility of acting as *monitor* for both the links $R \rightarrow M$ and $M \rightarrow D$. If *D* is not in the one-hop neighborhood of *T*, it acts as *monitor* for link $R \rightarrow M$ only. In the example, as *D* is in neighborhood of *T*, links $R \rightarrow M$ and $M \rightarrow D$ will be monitored by node *T*. Also, while agreeing to act as a *monitor* for a link, the candidate-*monitor* node checks if the link it has to monitor contains its own *forwarder*. In such cases, the candidate-*monitor* node will not agree to monitor that link. In other words, node *T* will not agree to monitor the link between $V \rightarrow H$, as node *H* is the *forwarder* for node *T*.

To reduce the number of nodes that act as *monitors*, each *monitor* is allowed to monitor a maximum of two links. Due to the trade-off of overloading a *monitor* node, monitoring multiple links to reduce the number of *monitor* nodes, a maximum of a two link monitoring capacity is set for the *monitor* nodes. Depending on the available links it can monitor, once the node *T* chooses to monitor the link $R \rightarrow M$ and $M \rightarrow D$, an *ACCEPT-TO-MONITOR* message is sent back to the requesting node *R*, stating the number of links the node *T* is willing to monitor. Node *R*,

on receiving this *ACCEPT-TO-MONITOR* message, configures node *T* as the *monitor* for the link *R ! M* and broadcasts a *MONITOR-NOTIFICATION-MESSAGE* message to all its one-hop neighbors about its newly configured *monitor T* and the links it can monitor $R \rightarrow V\ M$ and $M \rightarrow D$.

Inactive neighbors of *R*, except node *T* that receive this *MONITOR-NOTIFICATIONMESSAGE* message ignore the message. Active nodes that receive this message check for the possibility of node *T* acting as a *monitor* to them. Node *M*, upon receiving this message, sees that link $M \rightarrow D$ will also be monitored by node *T*, and configures *T* as its *monitor*. As node *M* is at an odd hop level that is scheduled to configure the *monitor* later, it skips the *monitor*-configuration process as it is already configured with a *monitor T*. The inactive node *T*, on receiving the *MONITOR-NOTIFICATION-MESSAGE*, checks if the message contains information on acting as *monitor*. If so, node *T* registers the nodes that it has to monitor and sends an *ACK-MESG* to node *R* to inform it about its confirmation as *monitor*. The originating node, on receiving the *ACK-MESG* from the *monitor* node, registers the *monitor* for the link between itself and its *forwarder*. Similarly, for the link between $D \rightarrow A$, node *G* is configured as *monitor*. The process will continue for all the active nodes in network, which results in all the links between active nodes being configured with a *monitor* node.

During this entire process of *monitor* configuration, it is possible for messages to fail due to congestion, collisions, and etc. To overcome these, each message is sent three times with a small random delay in between the messages. Configured *monitors* are shown in Figure 2.c. The link between $R \rightarrow M$ and $M \rightarrow D$ are monitored by node *T*. The link between $D \rightarrow A$ is monitored by node *G* and the link between $K \rightarrow C$ is monitored by node *N*. Since all the *monitor* nodes have a configured *forwarder*, in case of packet losses, *monitor* nodes would re-route the packet to its *forwarder* which is in a different data path.

**Packet Loss Detection:** Whenever a node detects an event, the node forwards the data to its *forwarder*. During this data forwarding, a packet might be dropped for various reasons. Broadly classifying, packet drops occur in the following scenarios:

- Collisions.
- Congestion at the receiver node (queue overflow).
- Link failures.
- Sudden Node failures.

*Monitor* nodes help in identifying all these losses and successfully deliver the packets to the Base Station through some other data path. In order to identify the packet losses, *monitor* nodes passively overhear the radio-data communication between the links that are being monitored.

**Channel Access:** Along with the aforementioned packet losses, due to the inability of the node to gain channel access, packets may be dropped at the sender nodes itself and not transmitted at all. *Monitor* nodes assist in reliable packet delivery only for the packets that were received by it. However, as the sender itself did not transmit the message, in such cases the monitor nodes cannot identify packet losses. To identify the packet drops caused because of the inability of a node to gain channel access and deliver them successfully, our protocol benefits from the powerful cross-layered architecture involving Medium Access Control (MAC) and Transport Layer. Native 802.11 MAC is used: a node sends a Request-to-Send (RTS) frame to its *forwarder* and upon receiving a Clear-to-Send (CTS) frame from the *forwarder*, the node sends the data packet, for which the *forwarder* acknowledges by sending an ACK message, which completes the data transfer between two nodes. After the *monitor* node identifies packet losses, it initiates a packet-loss recovery mechanism for reliably delivering the packets.

**Energy Efficiency:** Sensor nodes in WSN mostly use battery power, which makes them energy constrained. As the nodes deplete their energy, they eventually die and this hinders the performance of the network. As most of the nodes in WSN are in idle listening mode, the sensor nodes consume a lot of energy compared to the energy required to receive data. This waste of energy directly impacts the overall network performance. In order to extend the lifetime of the entire network, energy-efficient techniques must be incorporated in WSN.

Topology-control techniques and duty-cycle mechanisms reduce the energy consumed by the sensor network. In the proposed protocol, by choosing only a subset of nodes for data forwarding and from the set of inactive nodes, a minimal subset of nodes to act as *monitors* reduces the energy consumed by the network. However, as all the nodes in the network are awake all the time, they spend a considerable amount of time idle listening. To reduce energy consumption, duty cycles, wherein nodes go to sleep and wake up periodically for data

transmissions/receptions, are employed. By such active and sleep cycles, nodes save energy that would have been wasted in idle listening. However, when duty cycles are employed, the nodes must be awake for receiving data.

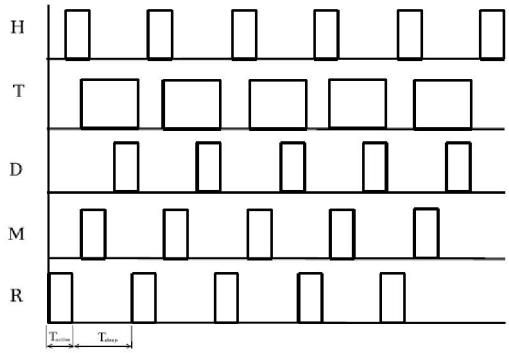



Figure 3: Staggered and Synchronized Duty Cycles

In staggered duty cycles, all the nodes that are of same hop level wake-up or go to sleep. Though all the active nodes in the network are assigned with the same duty cycle, the offset for each of the hop levels is varied. The offset difference between two hop levels, $i$ and $i - 1$ is the same as the amount of time required for transmitting one data packet. Because of this, all the nodes in a data path wake up sequentially as a chain reaction. Figure 3 shows that the nodes $R$, $M$, $D$, and $H$ which are at $i$, $i-1$, $i-2$, and $i-3$ hop levels wake up sequentially. Though this way of sequential wake-up is effective, this requires tight synchronization of wake-up and sleep cycles among the nodes of different hop levels. The data-gathering tree provides the ability to employ the staggered and synchronized duty cycles for reducing the latency, while maintaining the reliable packet delivery with the help of monitors. Our protocol employs staggered and synchronized wake-up cycles for all the nodes in the network.

This mechanism also helps in reducing the latency, except for that of set-up latency caused at the last level of nodes.

**Back-Off Timer Synchronization:** As discussed earlier, our protocol uses Native 802.11 MAC. In 802.11, whenever a RTS frame is sent and CTS is not received, MAC maintains a congestion window before retransmitting the frame. As the retry counts increase, the congestion window is increased exponentially. Depending on the value of the congestion window, a timer called back-off timer is fired for the corresponding amount of time, and after it expires, the RTS frame is retransmitted. Because of the congestion window being increased exponentially, the time between retransmitting the RTS frame is not uniform within the maximum retry counts.

### 2-2- Bounds Estimation

In order to estimate the performance of the network using our reliable data transfer protocol, the cost of energy consumption and latency were analyzed.

**Energy Consumption:** Let the number of active nodes be $x$ and the total number of links possible be $x - 1$. To monitor $x - 1$ links, the maximum number of *monitors* required is $(x - 1)$ (worst case) and the minimum number of *monitors* required is $(x - 1)/2$.

Let $E_r$ and $E_t$ represent the Reception and Transmission energy respectively. Let total transmit energy $E_T$ for $x$ nodes without *monitors* be $E_{nM}$. The total transmit energy $E_{TM}$ for $x$ nodes with *monitors* lies in the range

$$E_t \frac{3x - 1}{2} \cdot E_{TM} \cdot E_t(2x - 1)$$

Similarly, total reception energy *ERM* for $x$ nodes with *monitors* lies in the range

$$\frac{E_r(3x - 1)}{2} \cdot E_{RM} \cdot E_r(2x - 1)$$

From the above equations, (3.1) and (3.2), when duty cycles are not employed, the energy consumption in the case of *monitors* is always more compared to the case when there are no *monitors*.

**Average Delay:** Let the packets delivered to the Base Station without *monitor* support be $P_n$ and let the packets delivered to the Base Station with *monitor* support be $P_m$. We assume $P_m \geq P_n$, since *monitors* provide extra infrastructure in recovering the dropped packets. If *i* is the percentage increase in packet-delivery ratio, then

$$P_m = P_n + (iP_n) = P_n(1 + i)$$

Let the average hop length without *monitors* be *l*, and the average hop length with *monitors* be 2*l* (worst case). Let *t* be the transmission time for forwarding a packet from one node to another.

Then, the time taken to forward a packet to base station without *monitors* = ($lt$). For $P_n$ number of packets, time taken $T_n$ is,

$$T_n = (lt)P_n$$

The time taken to forward a packet to the Base Station with monitors = (2*lt*). For *Pm* number of packets, time taken *Tm* is,

$$T_m = (2lt)P_m$$

Using (3.3) and (3.4), we have

$$T_m = 2T_n(1 + i)$$

Equation (3.6) shows that average delay in case of *monitors* is twice that of the case without *monitors*.

## 3- RESULTS AND DISCUSSION

Extensive experiments were conducted in order to test the performance of the *monitor*-based approach.

### 3-1- Simulation parameters

The simulations were run with the simulation parameters as mentioned in Table 1. A subset of 30 nodes was considered to be active nodes and the remaining nodes were considered as inactive nodes. For data packets, Constant Bit Rate (CBR) traffic is generated. To evaluate the performance under a transient congestion scenario, the numbers of source nodes are varied for a constant packet interval rate. In all the experiments, each data point taken is an average of 20 independent runs.

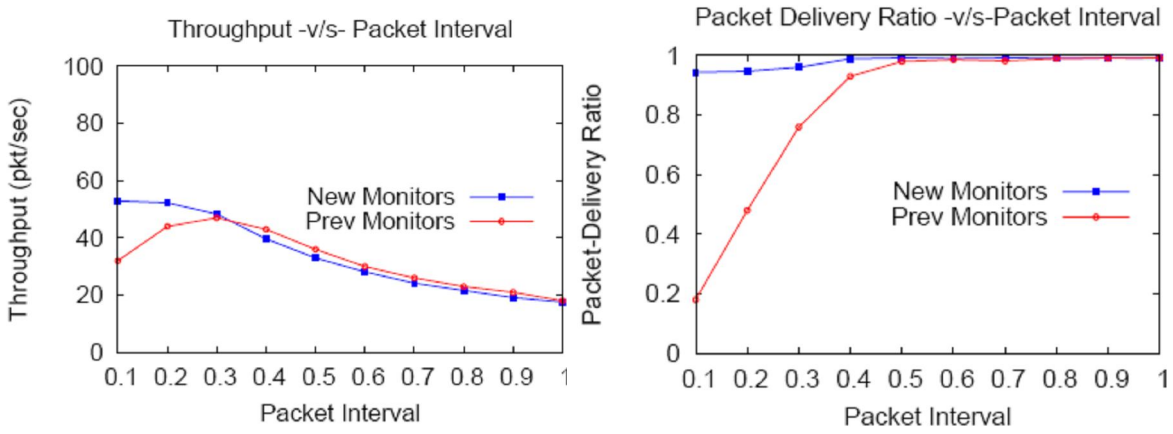
Table 1: Simulation Parameters

| Parameter | Value |
|---|---|
| **Area** | 1000*m* x 1000*m* |
| **Transmission radius** | 300*m* |
| **Total number of nodes** | 250 |
| **Number of sources** | 30 |
| **Data Packet size (bytes)** | 518 |
| **Packet interval rate** | 0.1(5*KB/s*) to 1(0.5*KB/s*) |
| **Transmit Power (W)** | 0.01490 |
| **Receive Power (W)** | 0.01280 |
| **Idle Power (W)** | 0.01240 |
| **Sleep Power (W)** | 0.000027 |
| | |

### 3-2- Comparative study

For a meaningful comparison of the proposed protocol with [13] in persistent and transient congestion scenarios, metrics such as throughput and packet-delivery ratio were considered.

**Persistent Congestion:** The proposed protocol is compared with the previous *monitor*-based approach [13] to evaluate the network performance under persistent congestion. Figure 4 (a) and Figure 4 (b) show the performance comparison of our protocol in terms of throughput and packet-delivery ratio. As the packet interval increases, throughput decreases due to fewer number of packets flowing through the network. In the previous *monitor*-based approach, congestion is observed at a packet interval of 0.3 seconds whereas in the proposed protocol, congestion is not observed even at higher packet frequencies (packet interval of 0.1 sec). Our protocol shows a significant improvement in throughput at higher frequencies where persistent congestion is observed. Also, throughput is calculated only for the original packets delivered to the Base Station

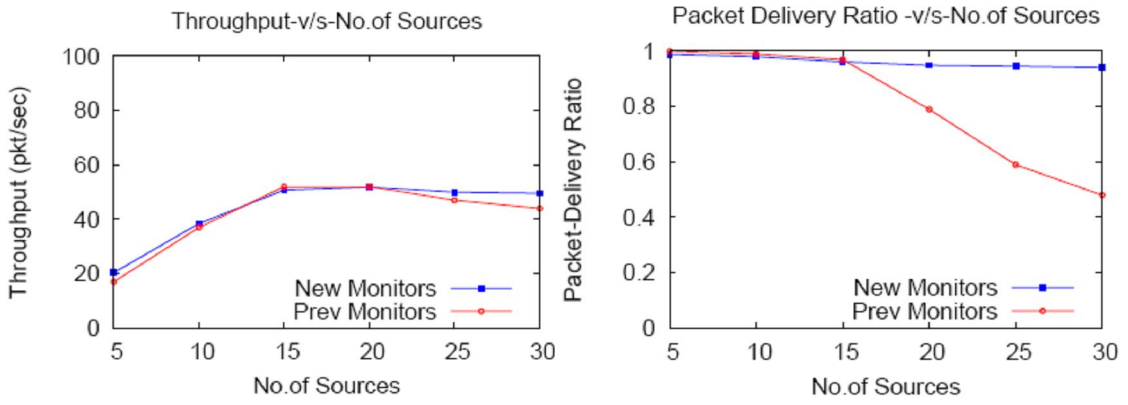



(a) Throughput v/s Packet Interval (b) Packet Delivery ratio v/s Packet Interval

**Figure 4: Comparison with original *monitors*-based approach in the case of persistent Congestion**

In the new *monitor*-based approach, where as the previous approach includes the duplicate packets that are delivered to Base Station. Due to this, though more actual packets are being delivered in the new *monitor*-based approach, the throughput graph appears fall a little bit when the network is not congested.

This improvement is because of the fact that in our protocol design, instead of choosing the entire set of nodes, only a subset of nodes is considered active. This decreases the congestion that is observed at higher frequencies. The packet-delivery ratio in Figure 4 (b) shows a significant improvement in our protocol compared to the previous *monitor*-based approach, even after subjecting the network to congestion below the packet interval rate of 0.3. Both protocols show a 100% packet-delivery ratio when not congested. Though both protocols benefit from *monitors*, when the network is saturated, our protocol alters the basic behavior of the network by shifting the knee point, which improves the packet-delivery ratio. As mentioned earlier, when the nodes cannot gain channel access for sending the data, the cross-layered design choices of our protocol helps the nodes to forward the data through *monitors*, Congestion which results in the shift of the knee point. Notice that our extended *monitor*-based protocol performs better than the previous *monitor*-based approach when subjected to persistent congestion.




(a) Throughput v/s Number of Events (b) Packet Delivery ratio v/s Number of Events

**Figure 5: Comparison with original monitors-based approach in the case of transient**

## 4- Conclusion

The main contribution of this paper is developing a reliable data transfer protocol by configuring inactive nodes as *monitors*. For upstream data traffic, many protocols have been proposed that provide reliability only at the event level. [11] was one of protocols that provide packet-level reliability from source to sink (Base Station). However, in this protocol, energy efficiency was not addressed. Unlike the other proposed methods, by utilizing a

cross-layered architecture using transport layer and MAC, our protocol enhances the packet-level reliability while reducing energy consumption. Our simulation results show that this technique improves energy efficiency and the packet delivery ratio even under congested scenarios. However, latency tends to increase under congested scenarios because of the additional support structure that provides packet-level reliability. In general, an increase in latency would affect the performance of the network. In the future, we would like to extend the *monitor*-based approach to select the *monitors* based on location and the coverage density. This way congestion can be contained and latency will be reduced. We also plan to investigate on very low duty cycles to further explore the potential to improve energy efficiency.

## REFERENCES

[1] E. Biagioni and S. H. Chen. A reliability layer for ad-hoc wireless sensor network routing. January 2004.

[2] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM MobiCom*, pages 85–96, 2001.

[3] Haiming Chen, Li Cui, and Victor O.K. Li. A joint design of opportunistic forwarding and energy-efficient mac protocol in wireless sensor networks. *IEEE GLOBECOM*, 2009.

[4] C.Intanagonwiwat, R.Govindan, and D.Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. *Mobile Computing and Networking*, 2000.

[5] C.Y.Wan, A.T.Campbell, and L. Krishnamurthy. Psfq: A reliable transport protocol for wireless sensor networks. *WSNA*, 2002.

[6] R. C. Doss and D. Chandra. Reliable event transfer in wireless sensor networks deployed for emergency response. In *Parallel and Distributed Computing Systems*, November 2005.

[7] A. Dunkels, J. Alonso, and T. Voigt. Making TCP/IP viable for wireless sensor networks. citeseer.ist.psu.edu/659578.html, January 2004.

[8] F.Stann and J.Heidemann. Rmst: Reliable data transport in sensor networks. *SNPA*, 2003.

[9] NS-2 network simulator. http://www.isi.edu/nsnam/ns.

[10] S. Park, R. Vedantham, R. Sivakumar, and I. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *MobiHoc'04*, 2004.

[11] Y. Zhou and M. R. Lyu. Port: A price-oriented reliable transport protocol for wireless sensor networks. In *IEEE International Symposium on Software Reliability Engineering (ISSRE '05)*, pages 117–126, 2005.