ENERGY-EFFICIENT RELIABLE SENSOR-TO-SINK DATA TRANSFER FOR WIRELESS SENSOR NETWORKS

by

Vamsi Krishna Venkata Naga Nandanavanam

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

May 2010

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Vamsi Krishna Venkata Naga Nandanavanam

Thesis Title: Energy-Efficient Reliable Sensor-to-Sink Data Transfer for Wireless Sensor Networks

Date of Final Oral Examination: 28 January 2010

The following individuals read and discussed the thesis submitted by student Vamsi Krishna Venkata Naga Nandanavanam, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Sirisha Medidi, Ph.D.	Chair, Supervisory Committee
Murali Medidi, Ph.D.	Member, Supervisory Committee
Tim Andersen, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Sirisha Medidi, Ph.D., Chair of the Supervisory Committee. The thesis was approved for the Graduate College by John R. Pelton, Ph.D., Dean of the Graduate College.

Dedicated to my Parents and Family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. Sirisha Medidi, whose guidance, motivation and invaluable support helped me in completing my thesis. I thank her for the encouragement, her trust in me, and for helping me to become who I am today.

My sincere thanks to Dr. Murali Medidi for being on my thesis committee. This thesis wouldn't have been possible without his constant guidance and support. His inspirational lectures and special talks moulded me into a better researcher.

I thank Dr. Tim Andersen for his valuable inputs, and for serving as committee member to evaluate my thesis.

I would like to thank all the members of the SWAN Lab, especially Ramya Ammu, for their encouragement and extending support when I needed them the most.

My special thanks to my parents for providing me with this opportunity to pursue my Masters and to my brother for his valuable advice and support.

ABSTRACT

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. As aggregated data packets contain correlated information, a significant amount of information is lost if a data packet is lost. 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. By employing duty cycles, energy-efficiency can be improved but it degrades the network performance in terms of latency. 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. To further reduce the energy consumption and congestion in the network, only a subset of nodes is chosen as active nodes to transfer the data. We implemented our protocol using the NS2 simulator for evaluating its performance. Results show that our protocol has significant improvement in packet delivery ratio and energy savings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Wireless Sensor Networks (WSNs) provide a distributed, sensing and computing platform for monitoring the environments, in which deploying conventional networks is impractical. By placing sensor nodes over a large area it is possible to get the fine-grained readings of any particular data of interest. 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 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 [10] 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.

Motivated primarily by the challenge of applications that need packet-level reliability and energy-constraint characteristics of WSN, we propose a cross-layered network infrastructure design. 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 [13], active and inactive nodes are chosen and from the subset of inactive nodes, subset of nodes called *monitors* are 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.

1.1 Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 reviews the related work. Chapter 3 explains the motivation and objective of this thesis and describes our Protocol Design. Chapter 4 presents the performance evaluation of the presented protocol. Finally, we conclude in Chapter 5.

CHAPTER 2

RELATED WORK

Nodes in WSNs are mainly battery operated and have limited energy and processing capabilities. Unlike the traditional networks, WSNs differs in several aspects. In order for large deployments to be cost-effective, nodes are resource-constrained in terms of energy capacity, radio transmissions, processing capabilities, and memory storage. Depending on the applications, the reliable data transfer and energy efficiency play a very important role in WSN research. Many protocols have been proposed to provide energy efficiency and reliable data transfer of packets in WSNs. These protocols are broadly classified as

1. Protocols that provide Reliability
2. Protocols that provide Energy Efficiency

2.1 Reliability

WSNs consists of a large number of sensor nodes densely deployed in the areas of interest. On detecting the event, sensor nodes generate packets forward them to the Base Station or sink with the help of neighboring nodes. At the Base Station, these packets are processed and necessary action taken. This clearly shows the need for a reliable data transport in WSNs that cater to the primary objective of reliable event detection. Due to the error prone nature of WSNs existing TCP/IP protocols used for reliable data delivery in wired networking are not suitable [5]. One solution is to tailor TCP/IP protocols to meet the needs of WSNs. Another approach to provide reliability in sensor networks is by providing a new transport-layer protocol. Many transport protocols have been proposed to provide reliable data delivery [1, 4, 5, 8, 10, 12, 13, 19, 20, 22, 26, 27, 30].

Reliable Data Transport In Sensor Networks (RMST) [10] is proposed to provide reliability at the transport layer. This scheme adds reliable data transfer to directed diffusion. RMST is designed for delivering large blocks of data in multiple segments from a source node to a sink node. For example, this is required when time series data has to be transmitted. Reliability in RMST refers to the eventual delivery to all subscribing sinks of any and all fragments related to a unique RMST entity. A unique RMST entity is a data set consisting of one or more fragments from the same source. RMST does not include any real-time guarantees. In RMST, receivers are responsible for detecting whether or not a fragment needs to be re-sent. In the cached mode, the sink node and all intermediate nodes on an enforced path cache segments and check the cache periodically for missing segments. When a node detects missing segments, it generates a NACK message, which travels back to the source along the reinforced path. The first node A, having missing segments in its cache, forwards them again towards the sink (and thus towards the requesting node). If A can retrieve all requested segments from its cache, then A drops the NACK packet; otherwise, it is forwarded further upstream. Both the segments and the NACK packets are represented in terms of attributes, to be compatible with directed diffusion. In the non-cached mode of RMST, only the sink node has such a cache, but not the intermediate nodes; therefore, NACKs travel back to the source node (which clearly also needs to cache the segments). Some of the services offered by RMST are loss detection and repair: losses are detected at each node using a watch-dog timer. The timer handler sends a NACK for the missing fragments that have aged too long. An other important feature of RMST is a back channel at each node, which is a channel towards the source. This is used for sending NACK to the sender for missing fragments. RMST combines a NACK-based transport-layer protocol with S-ARQ to achieve the best results. RMST does not guarantee reliability when nodes fail and does not address the congestion in sensor networks.

To achieve event-to-sink reliability, event-to-sink reliable transport in WSNs (ESRT) [22] has been proposed. ESRT seeks to achieve reliable event detection with minimum energy expenditure and congestion control. It has been tailored for use in sensor networks with adaptability to dynamic topology, collective identification, energy conservation, and biased implementation at the sink. Reliability is measured by the number of data packets received at the sink. To measure reliability, the concept of observed event-level reliability and desired event-level reliability have been introduced. Observed event reliability (r_i) is the number of packets received in a decision interval i at the sink. The desired event reliability R is the number of data packets required for reliable event detection. If the observed event reliability is greater than the desired reliability, the event is deemed to be reliably detected. Otherwise, appropriate actions have to be taken to achieve this reliability. The reporting rate of a sensor node is defined as the number of packets sent out per unit time by that node. ESRT configures the reporting frequency f such that the desired event detection accuracy is achieved with minimum expenditure. Five different characteristic regions have been identified based on reporting frequency f, r, and R. The five states are “No Congestion, Low Reliability”, “No Congestion, High Reliability”, “Congestion, High Reliability”, "Congestion, Low Reliability" and "Optimal Operating Region". The goal of ESRT is to maintain operation in the Optimal Operating Region. The network can reside in any one of these states. Depending on the current state (S_i) of the network, ESRT finds the updated reporting frequency ($F_i + 1$) and broadcasts it to all the source nodes. If the observed event reliability is less than the desired reliability, ESRT increases the reporting frequency of the nodes; otherwise, if the observed reliability is more than the desired level, the reporting frequency is decreased to avoid congestion and reduce energy consumption. To detect the current state of the network, the sink must be able to detect congestion in the network. The sensor nodes detect congestion using the buffer sizes and set the congestion notification bit. Once the sink receives a packet with this bit set, it knows that congestion will take place and will update the reporting frequency accordingly. ESRT does not support end-to-end reliable data delivery and it is impractical to vary the transmission rates of the nodes depending on the applications.

Pump Slowly, Fetch Quickly (PSFQ) [5] distributes data from a source node by pacing data at a slow speed but allows nodes that experience data loss to fetch any missing segments from intermediate nodes very quickly. PSFQ is designed to ensure that all data segments are delivered to all the intended receivers with minimum support from the underlying transport infrastructure. It minimizes the number of retransmissions for loss detection and recovery operations with minimal signaling. It operates correctly even in an environment in which the radio link quality is very poor. PSFQ has three operations: pump operation, relay-initiated error recovery, and report operation. A source injects messages into the network and the intermediate nodes buffer and relay messages with the proper schedule to achieve loose delay bounds. A relay node maintains a data cache and uses cached information to detect data loss, initiating error recovery if necessary. A feedback and reporting mechanism is incorporated in PSFQ to help the source know the time at which the receivers received the information. In the PUMP operation, a user node broadcasts a packet to its neighbors every T min until all the data fragments have been sent. Neighbors that receive this packet will check against their local data cache, discarding any duplicates. If this is a new message, PSFQ will buffer the packet. If there is no gap in the sequence number, then PSFQ sets a schedule to forward the message. A node goes into the FETCH mode once a sequence number gap is detected. A fetch operation is the proactive request of a retransmission from neighboring nodes once loss is detected at a receiving node. PSFQ uses the concept of loss aggregation whenever loss is detected; that is, it attempts to batch up all message losses in a single fetch operation whenever possible. In addition to the PUMP and FETCH operations, PSFQ supports a report operation designed specifically to give feedback on the data delivery status information to users in a simple and scalable manner.

Witness-Aided Routing Protocol (WAR) [12] is designed to support unicast routing over both unidirectional and bidirectional links in ad hoc networks while maintaining low-bandwidth utilization and reliable-packet delivery. WAR is based on the notion of a witness host, which plays a central role in the routing and recovery process. The neighborhood of a mobile host is a union of two sets: the incoming and the outgoing neighborhood. The incoming neighborhood of host A is the set of mobile hosts one hop away from A, whose transmissions it can hear. Similarly, the outgoing neighborhood of A is the set of mobile hosts on each hop away from A, which can hear its transmissions. The two neighborhoods are the same if the links are bidirectional. A witness is a host that can overhear a transmission that was not destined to it. Thus, all witness hosts of A are members of N *out(A)*. When a witness host is also a member of N *in(A)* and W belongs to N *out(A)*. WAR is a reactive protocol and has three major components: route discovery, packet forwarding, and route recovery. The discovered route is invoked by a source host S every time it needs to route to a destination host D, and it does not have one already cached. Host S broadcasts a route-request message and starts a local timer to decide when to re-send the request in case no response arrives. All hosts in N *out(s)* hear the request and they replicate it so the route-request message propagates until it reaches D. Since the route request is a broadcast, it is likely that it arrives at the destination on many different paths. D will choose one of these and will send a route reply message to S. WAR uses source routing in order to deliver packets from a source host to it, so at each intermediate host, the packet contains information about the next hop in the route. Before A delivers a packet to the next host B in the route, it removes its id from the list of remaining hosts. A packet is considered to be successfully delivered if A receives a passive acknowledgment from B or if A receives a positive acknowledgment from any of its witness hosts. Otherwise, A assumes that the route is broken and initiates a route discovery procedure. During the route-recovery process, WAR attempts to quickly and inexpensively bridge the gap created in the route. This would allow the packet to travel to its destination as opposed to delaying it until S finds another route.

Directed diffusion [4] aims to optimize robustness, scalability, and energy efficiency for data-centric network interests. Events that are to be recorded are propagated by sink nodes throughout the network via flooding. These initial requests for information stipulate that responses should be sent at a low data rate. Upon receiving responses from multiple sources, a sink can use positive and negative reinforcement to increase and decrease gradient (*i.e.,* paths) data rates, respectively. Reinforcement mechanisms can vary; this paper uses interest messages with modified interval values for this purpose. Rules governing the propagation of control messages are executed locally; that is, each node decides individually without a global state how it will react to control messages. The adoption of a hop-to-hop paradigm obviates the need for complex routers and universally unique identifiers. As well, it can result in highly adaptive networks, in which broken links can be avoided automatically according to local policies. In $NS2$ experiments, the authors found that directed diffusion dissipated less energy than both flooding (a watermark) and omniscient multicast (a representation of attainable performance with traditional end-to-end architectures). Additionally, it incurred comparable delay to omniscient multicast, and substantially out-performed flooding. Two particularly important performance-enhancing factors of directed diffusion are negative reinforcement, which allows for the pruning of superfluous gradients, and duplicate suppression, which takes advantage of the incorporation of application layer semantics in the communication protocol (made possible by directed diffusion) to avoid transmitting redundant messages.

Wang and Medidi proposed a sensor-to-sink transport protocol [13] that is suitable for data aggregation and provides reliable upstream packet delivery by configuring the inactive nodes as *monitors*. Monitors assist in reliable loss detection and recovery in case of sudden node failures and congestion. The aim is to have a minimum set of nodes acting as monitors to keep all the nodes in the data path monitored. The configuration of monitors is done using a distributed heuristic that requires only one hop information. This heuristic activates a minimal set of monitors when the flow starts and deactivates them when the flow stops. Each node is assigned a rank that is initialized to zero initially. The rank of the node is incremented when nodes on the data path forward a rank-increment message. Similarly, after a flow is finished, the nodes decrement their rank. In the monitor-configuration process, if a data packet is received at a node and there is no monitor configured for that node, the node will broadcast a monitor-request message to its one hop neighbor with the highest rank, which replies with a monitor-agree message. The node will record its monitor and will inform all its other neighbors of its monitor. The neighbor nodes, once they receive this message, will update their rank and monitor accordingly. In case of packet losses due to congestion, retransmission to the same forwarder might not be successful. Hence, the retransmitted packets will be sent to the monitor node. The monitor will then choose a new route altogether to forward the packet. This will ensure reliable packet delivery.

Chen *et al.* proposed a light-weight opportunistic forwarding scheme (LWOF) [3] to provide reliable data delivery for wireless sensor networks with asynchronous duty cycle. To exploit the non-deterministic characteristic of opportunistic forwarding, an energy-efficient MAC protocol was also proposed. LWOF scheme uses the preamble in Low-Power Listening MAC to dynamically select the forwarder during data transmission thus reducing the overhead of maintaining historical network information or contention process. A preamble that lasts at least as long as the sleep period of the receiver is transmitted by the sender. The receiver node, when awake, detects the preamble and stays awake to receive the data. The sequential detection of preamble and busy tone signal help in reliably forwarding the data dynamically. The LWOF scheme uses two channels with low and high data rates for transmitting the busy tone and sensor data, respectively. Although the proposed protocol achieves a higher packet delivery ratio in the networks without congestion, performance of the protocol under congested-network scenarios were not explained.

2.2 Energy Efficiency

As nodes in WSNs are battery operated, it is not practically possible to replace the batteries frequently. Energy consumption needs to be minimized in order to prolong the network lifetime and performance. There have been many protocols proposed for energy efficiency [2, 6, 7, 14–16, 24, 28, 29, 31, 32]. Depending on the target applications of WSNs topology control mechanisms, wake-up scheduling mechanisms involving Medium Access Control (MAC) and hybrid mechanisms were proposed. Due to severe bandwidth, power, and range constraints in sensor networks, MAC protocols of these networks needs to be tailored for efficient bandwidth utilization and energy efficiency unique to sensor networks.

Power-Efficient Gathering in Sensor Information Systems (PEGASIS) [14] is the protocol in which only one node is chosen as a head node that sends the fused data to the Base Station per round. PEGASIS protocol requires the formation of a chain, which is achieved in two steps. During the chain construction phase, the farthest node from the Base Station is considered first, and a greedy approach is followed to construct the chain. During the data gathering phase, a leader of each round is selected randomly. Randomly selecting the head node also provides a benefit as it is more likely for nodes to die at random locations, thus providing a robust network. When a node dies, the chain is reconstructed to bypass the dead node. After the leader is selected, it passes the token to initiate the data gathering process. Passing the token also requires energy consumption but the cost of passing the token is very small because the token size is very small. In PEGASIS, the transmitting distance is reduced for the sensor nodes. Since each node gets selected once, energy dissipation is also less, compared to LEACH. Experimental results show that PEGASIS provides improvement by a factor of two more than LEACH.

To reduce energy consumption without affecting the connectivity of the network, an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks (SPAN) [2] has been proposed. SPAN is based on the observation that a dense sensor network can work with only a part of its nodes being active. It is possible to prolong the network lifetime while maintaining its functionality by carefully choosing the active nodes. SPAN is a distributed, randomized algorithm in which nodes make local decisions on whether to sleep or to join a forwarding backbone as a coordinator. SPAN elects coordinators from all nodes in the network. SPAN coordinators stay awake continuously and perform multi-hop packet routing within the network while other nodes remain active in power saving mode and check if they should wake-up and become a coordinator. SPAN ensures that sufficient numbers of coordinators are chosen so that every node falls under the range of at least one coordinator. The coordinators keep changing to ensure that all nodes share the task of providing connectivity equally. It tries to minimize the number of nodes acting as coordinators to reduce the latency and increase the network lifetime. Also it elects coordinators using only local information in a decentralized manner. A non-coordinator node becomes a coordinator if it discovers, using only information gathered from local broadcast messages, that two of its neighbors cannot reach each other either directly or via one or two coordinators. The intent to become a coordinator is announced with a HELLO message. Span resolves contention by delaying coordinator announcements with a randomized back-off delay. In order to ensure fairness, after a node has been a coordinator for some time it withdraws if every pair of neighbor nodes can reach each other via their neighbors or other coordinators. This gives a fair chance to all nodes that are eligible for being coordinators.

Geographical Adaptive Fidelity (GAF) [28] is introduced to reduce energy consumption in ad hoc wireless networks. GAF identifies equivalent nodes for routing based on location information and turns off unnecessary nodes. In a wireless network, a lot of energy is spent in idle listening. Idle energy is almost equal to transmission energy or reception energy. Powering off the radio conserves energy both in over hearing due to data transfer and in idle state energy dissipation when no traffic exists. Hence nodes that power down their radios are explored. It is also observed that when there is significant redundancy in an ad hoc network, multiple paths exist between nodes. Hence, a few nodes can be powered off while still maintaining connectivity. Routing Fidelity is defined as uninterrupted connectivity between communicating nodes. Routing Fidelity can be maintained as long as any intermediate node is awake. Each GAF node uses location information to associate itself with a virtual grid, in which all nodes in a particular grid square are equivalent with respect to forwarding packets. Nodes in the same grid will then coordinate which nodes will sleep and for how long. In GAF, nodes are in one of these three states: sleeping, discovery or active. Initially nodes start out in a discovery state. In this state, nodes turn their radio on and exchange discovery messages to find other nodes in the same grid. When a node enters the discovery state, it sets a timer, and when the time expires it moves to the active state. When a node enters the active state, it sets a timer and moves to the discovery state when the timer expires. A node in the discovery or active state changes state to sleeping when it determines that some other equivalent node can handle routing. GAF employs a load-balancing strategy so that all nodes remain up and running together for as long as possible. This ensures that all nodes are given equal chance and no one node is penalized more than the others. After a node remains in the active state for some time, it changes its state to discovery to give a chance to other nodes in the grid to become active. When the active node changes its state to discovery, it is more likely that it has less remaining energy than its neighbor nodes. In the ideal scenario, there is one active node at any point of time in one grid. When the nodes move, there is every chance that we might have a grid with no active nodes at all. To avoid this problem, each node estimates the time it expects to leave the grid and includes this information in the discovery message. When other nodes enter the sleeping state, they decide how long to stay in the sleeping state based on this information.

A Sparse Topology and Energy Management (STEM) was prosposed in [24]. The main objective of this scheme is to reduce energy consumption in a monitoring state to a minimum while ensuring satisfactory latency for transitioning to the transfer state. For the majority of the time, the network is only sensing its environment. This is referred to as the monitoring state. Once an event occurs, data has to be forwarded to the sink and the network transitions to the transfer state. STEM reduces energy consumption of the nodes by putting them into a sleep state. In the monitoring state, instead of full sleep, a node goes into low-power listen mode. However, in return for this energy reduction, a certain amount of latency is introduced to wake-up the nodes. Nodes in this design have three states: sleep, active, or listening. The node that wants to communicate (initiator node) polls the node that it has to wake up (target node) continuously. As soon as the target node hears the poll, the link between the nodes is activated. Once the link is activated, data transfer takes place using a MAC protocol. To wake up a node, a wake-up message is sent to the node in the form of a beacon packet (STEM-B meaning beacon-based) or a simple tone (STEM-T meaning tone-based) resulting in two variants of STEM. The topology management in STEM is specifically geared toward those scenarios in which the network spends most of its time waiting for events to happen, without forwarding traffic. Simulation results show a considerable improvement over GAF in both the scenarios. Though this scheme has many advantages, it suffers from high energy consumption due to continuous polling and requires extra radio on the sensor nodes.

In [29], authors proposed "Pipeline tone wakeup" (PTW), a wake-up/sleep mechanism to maintain energy efficiency and an end-to-end delay between the nodes in the MAC layer. To achieve this, PTW uses an additional wake-up channel in comparison with the regular data channel and uses a wake-up tone to awake neighboring nodes. Hence, any node in the neighborhood of the source node will be awakened. In PTW, the burden for tone detection is shifted from receiver to sender. This means that the duration of the wake-up tone is long enough to be detected by the receiver that turns on its radio periodically. The rationale behind this solution is that the sender only sends a wake up tone when an event id is detected while receivers wake up periodically. In addition, the wake up procedure is pipelined with the packet transmission so as to reduce the wake-up latency and, hence, the overall message latency. Suppose that node A has to transmit a message to node D through nodes B and C. At time t_0, A starts the procedure by sending a tone on the wake-up channel. This tone awakens all A's neighbors. At time t_1, A sends a notification packet to B on the data channel to inform it that the next data packet will be sent to B. Upon receiving the notification messages, of all A's neighbors except B learn that the following message is not intended for them. Therefore, they turn off their data radio. Node B, realizes to be the destination of the next data message, and replies with a wake-up acknowledgment on the data channel. Then, A starts transmitting the data packet on the data channel. At the same time, B starts sending a tone on the wake-up channel to awake all its neighbors. The packet transmission from A to B on the data channel, and the B's tone transmission on the wake-up channel are done in parallel. As in STEM, the data transmission is regulated by the underlying MAC protocol. It is shown by simulation that, if the time spent by a sensor network in the monitoring state is greater than several minutes, PTW outperforms STEM significantly, both in terms of energy saving and message latency, especially when the bit rate of sensor nodes is low.

The Latency minimized Energy Efficient MAC protocol (LEEM) [6] is a hop ahead reservation scheme that minimizes latency and increases energy efficiency. It does this by reserving the channel of the next hop in advance. It is useful in time-critical applications in which sensed events are to be reported immediately to the sink to take remedial or defensive actions. LEEM assists in sending the packets with minimum delay by using dual-frequency radio set up. Since the channel of the next hop is reserved in advance, the intermediate nodes in the data path forward the packet as soon as it is received. In an event-driven sensor network, the nodes spend a lot of time sensing the event. In order to reduce this energy, the control channel radios are kept in a lower power sleep mode. The control channel radio is made active periodically to check for any data transmissions and activate the data channel. In LEEM, nodes are resynchronized every hour. The synchronization helps in making reservations and reduces the delay. In a synchronized network, since each node knows the time at which its next hop node is active, it need not send continuous wake-up signals. This results in lower energy consumption. The reservation scheme in LEEM helps in eliminating the set-up latency at the intermediate nodes. When a sensor node's control channel radio is in sleep mode and an event occurs, the sensor node waits for the next hop node to become active. It then requests the next hop node to activate its data channel radio by sending a request packet. The receiver agrees by sending back an ACK. This procedure continues throughout the data path until the packet reaches the data sink. Whenever the data transfer takes place, the receiver of the packet reserves the channel for K hops ahead. If the value of K is one, it is a 1 hop reservation scheme; otherwise, it is an N-hop reservation scheme. When the current transmission gets completed at the receiver, the next hop channel becomes ready. Hence, the delay in setting up the next channel is avoided except at the first hop. Although LEEM shows significant improvement over other protocols like STEM and PTW in terms of energy consumption and latency, it is not applicable for applications that have continuous occurrence of events.

DMAC [15] is an energy-efficient and low-latency MAC protocol that is designed and optimized for applications in which the major traffic pattern consists of data collected from several source nodes to a sink through a unidirectional tree. DMAC uses a staggered active/sleep schedule to solve the data forwarding interruption problem and enables continuous data forwarding on the multi-hop path. In DMAC, data prediction is used to enable active slot request when multiple children of a node have packets to send in the same sending slot, while a More-to-Send packet is used when nodes on the same level of the data-gathering tree with different parents compete for channel access. Nodes that are out of the hearing range of both the sender and receiver are unaware of the ongoing transmissions, and therefore go to sleep until the next cycle/interval. The data-forwarding process will then stop at the node whose next hop towards the sink is out of the overhearing range because it is in sleep mode. This is the data-forwarding interruption problem. In DMAC, the activity schedule of nodes on the multi-hop path are staggered to wake up sequentially. An interval is divided into receiving and sending the sleep periods. In the receiving state, a node is expected to receive a packet and send an ACK packet back to the sender. In the sending state, a node will try to send a packet to its next hop and receive an ACK packet. In sleep mode, nodes turn off the radio to save energy. The receiving and sending periods have the same length, which is enough for one transmission and one reception. Based on the depth of the node in the tree, the node skews its wake-up scheme. Since nodes on a data path wake up sequentially to forward a packet to the next hop, sleep delay is reduced. Every node periodically turns to receiving, sending, and sleep states. When a node has multiple packets to send at a sending slot, it needs to increase it own duty cycle and request other nodes on the multi-hop path to increase their duty cycles. A more-data flag is added in the MAC header to indicate the request for an additional active period. The receiver checks the more-data flag and if the flag is set, it also sets the more data flag. With this mechanism, DMAC can react quickly to traffic variations to be both energy efficient and maintain low latency. In order to avoid collision and interference, DMAC uses data-prediction schemes and More-To-Send packets.

Zhou and Medidi proposed Energy-Efficient Contention Resilient MAC (ECR-MAC) [31] to improve both energy efficiency and delay without requiring additional hardware support and synchronization. ECR-MAC can efficiently handle spatially-correlated contention and scales well with network density. Dynamic Forwarding Scheme (DFS) is employed in ECR-MAC to allow more flexibility for packet forwarding and to improve both the energy efficiency and latency. DFS assigns multiple potential forwarders for a sender and each forwarder employs independent wake-up schedules without synchronization to reduce the protocol overhead. Instead of waiting for a particular forwarder, each sender hurls packets as quickly as possible to any one of the nodes termed as potential forwarder that can help transmit packets. This effectively reduces the transmission-energy consumption, and eliminates the requirement of synchronization. In a dense WSN, there usually exists sufficient potential forwarders that can serve as a sender. Since a sender chooses its forwarder based on the routing agent, ECR-MAC is cross-layered. In order to increase the energy efficiency and reduce delay, ECR-MAC employs duty cycles. Each node will be periodically activated for T_{active} to detect any packet-forwarding requirements. Before transmitting packets, a sender will wait for enough time to detect on-going communication. If none is detected, the sender sends WAKEUP messages periodically to see if one of its potential forwarders is awake. Any potential forwarder of this sender that receives a WAKEUP message will send a REPLY message. Upon receiving the first reply message from a potential forwarder, the sender sends data to the potential forwarder from whom it received the message and the forwarder replies with an ACK message. ECR-MAC handles contention by allowing senders to deploy independent routes that detour the congested network area, thereby increasing network throughput.

To address "spatially-correlated contention, Zhou and Medidi proposed a distributed topology control [32] to schedule node wake-up slots and design a MAC protocol to benefit from this topology control for improving energy efficiency and reducing latency. Energy consumption in an idle listening state is as much as the transmission and reception energy. One way to save energy is to employ duty cycles. By employing duty cycles, only a subset of nodes remain active at any point of time. The remaining nodes turn off their radios and keep checking their eligibility to remain active periodically. However, a lower duty cycle can require each node to spend a longer set-up latency to wake up its forwarder, which increases the end-to-end delay. In order to have low delay while having low duty cycles and high energy efficiency, the following sleep-based topology control was designed. To address spatially-correlated contention, the topology is controlled such that each node has multiple potential forwarders. Allowing each node to have multiple forwarders along with staggered scheduling cannot only reduce congestion but also achieve shorter delay since it significantly reduces the first hop set-up latency and eliminates the latencies in further hops.

For Maximizing System Lifetime in WSNs, Dong [7] presented a formal analysis of a number of network lifetime-maximization problems under different energy consumption models. The first model is attributed to packet transmissions. Wireless transceivers are assumed to consume power only when transmitting packets and thus energy is consumed on a per packet basis. This model is referred to as a packet model. The second model is the time-based model. Energy consumption during overhearing and idle periods is as equal as transmission and reception energy. If the pair-wise distance between sensor nodes is small, packet transmissions between sensor nodes consumes less energy. On the other hand, each sensor node covers more sensor nodes in its transmission range and thus more energy will be consumed due to overhearing. When wireless transceivers stay idle and no communication occurs, energy is completely consumed. Hence, this model is referred as a time-based model. The third model is the mixed model. In cases where nodes communicate frequently, energy consumption can be divided into two parts. On one hand, sensor nodes consume as much power as each other on a per time-unit basis; on the other hand, they may consume significantly different amounts of energy on a per packet basis due to transmission and reception. Hence, this model is termed a mixed model. Network lifetime is defined for a number of packets that can be delivered by the network. In a time-based model, network lifetime is defined to be the time until no backbone is formed. A variety of network maximization problems have been analyzed in the time-based model and packet model. The definitions and complexity analysis of problems is presented.

To reduce the communication latency for periodic energy-efficient radio sleep cycles, authors [16] considered the single wake-up schedule case in which each sensor can choose exactly one of the k slots to wake up. A novel graph-theoretical abstraction of this problem is formulated and it is proved that this problem is NP-hard. The optimal solution in special cases (tree and ring topologies) have been derived and analyzed and several heuristics solutions for arbitrary topologies have been proposed. It has also been proved that by carefully choosing multiple wake-up slots for each sensor, significant delay savings can be obtained over the single wake-up schedule case with the same duty cycle. One of the approaches used is adaptive listening in which nodes that lie one or more steps ahead in the path of a transmission can be kept awake for an additional length of time. However, this approach provides some reduction in sleep latency at the cost of greater energy expense due to extended activation and overhearing but is not sufficient for long paths. The authors address the problem of how should the activity of sensor nodes be scheduled in arbitrary network communication topologies in order to minimize the sleep latency while providing energy efficiency through periodic sleep. A number of assumptions have been made. A parameter k is chosen, which captures the duty cycling requirements of an application. The authors assumed low traffic load in which there is no delay due to congestion interference or collision. To get the required duty cycle, a sensor should be kept awake an average of a $1/k$ fraction of time slots. In the single wake-up schedule in which the schedule is k slots, each sensor is assigned one of the k slots during which it activates its radio for reception, while it can transmit in any slot if there is a packet to be forwarded. If a node has to forward a packet to its neighbor, it can wake up in the active reception slot of its neighbor and transmit the packet. This conserves energy of both the transmitting and receiving node. In the multi-scheduling, sensors wake up at multiple time slots. The schedule is increased proportionately so that each sensor remains active for only a $1/k$ fraction of time slots on average.

Ruiz *et al.* proposed a cross-layer protocol (QUATTRO) [21] that achieves energy efficiency by eliminating collisions along with quality of service. In this protocol, MAC and routing protocols collaborate to discover routes, organize nodes into clusters, and schedule the access to the transmission medium in a coordinated manner. The various tasks performed by routing protocol include path discovery, resource reservation, cluster formation, and gathering of information. The MAC protocol uses this information and creates activity schedules for clusters and uses a collision-free protocol for communication within each cluster. Nodes turn off their transceivers outside their activity period, thereby saving energy. Quality of service is ensured by using a bandwidth-dependence-aware reservation procedure in which a node will affect and be affected by the transmissions of its one and two hop neighbors, even if the nodes do not belong to the commonly established routes. Though the authors claim to achieve quality of service and energy efficiency, experiment results are not provided to support their claim.

CHAPTER 3

ENERGY EFFICIENT RELIABLE DATA TRANSFER PROTOCOL

3.1 Motivation and Design Considerations

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.

Many protocols provide event-level reliability for the upstream traffic. A Protocol that provides packet-level reliable transport for the upstream data traffic was proposed by [13]. However,the protocol proposed in [13] does not alter the behavior of the network in spite of having an additional mechanism to provide packet-level reliability. Although the packet delivery ratio improves due to this additional support mechanism, the network performance degrades when subjected to congestion, which is not desirable. Since WSNs become heavily congested when multiple events occur, protocols designed for providing packet-level reliability are expected to improve the performance of the network even under congested scenarios. A shift in knee point for the packet delivery ratio is not observed in [13], indicating unaltered network behavior in spite of having additional support mechanism. Along with providing packet-level reliability if the protocols designed were not energy efficient, the lifetime of the network degrades, thus affecting the performance. However,providing energy efficiency has its own set of overheads. The existing packet-level reliability protocols for upstream data traffic do not ensure energy efficiency. To fulfill these requirements, there is a strong need for a protocol that is energy efficient and improves the packet-level reliability by fundamentally altering the network behavior while reducing the cost of overhead.

In this thesis, the base idea of *monitors* presented in [13] is used and a protocol that ensures packet-level reliability and improves energy efficiency is proposed. Unlike traditional ways of designing protocols that are specific to one particular level (i.e., reliability in transport layer, medium access in data-link layer), the proposed protocol benefits from a cross-layered architecture by providing medium access control with application layer information for effective routing of the packets. The proposed protocol improves the energy efficiency of the network by incorporating a duty-cycling technique for energy savings. With duty cycles, the nodes wake up periodically for sending the packets and go to sleep when there are no packets to send. As the energy consumed in idle listening is comparable to that of energy consumed for receiving the data packets, the major contribution of energy consumption comes from idle listening. Duty cycles ensures that the nodes do not spend time staying awake in idle listening, thus increasing the energy efficiency. Further, eliminating redundant nodes and choosing only a subset of the sensor nodes to be active in data forwarding reduces the energy consumption, thus improving network lifetime. By constructing a data-gathering tree, nodes can be organized in levels that help in setting up synchronized and staggered duty cycles to reduce latency. The data-gathering tree also helps in identifying the subset of nodes for energy savings.

To provide packet-level reliability, the proposed protocol detects packet losses and provides a recovery mechanism to forward the packets on a different data path. In WSNs, packet losses are due to link failures, resource constraints, congestion, and sudden node failures. The proposed protocol provides an alternate support structure to address these challenges and provides packet-level reliability. In the following sections, the design challenges are identified and how the protocol addresses these issues are described.

- Link Failures:

  There are several reasons for packet losses in wireless networks. Due to errors in links between two nodes, packets may not be delivered. These errors can occur due to signal attenuation. Attenuation refers to any reduction in the strength of a signal and is caused by signal transmission over long distances. As a result, packets will be corrupted by the time they reach the receiver. Packet losses could also occur when two nodes try to transmit data simultaneously. When two nodes try to send data at the same time, they may collide and packets from either of the nodes might get dropped. In order to provide packet-level reliability the designed protocols need to have an ability to recover packets in case of link failures.

- Node Failures:

  Due to a drop in energy levels or by any other unforeseen events, nodes in a sensor network are subject to failures. If a node fails while transmitting/receiving a packet, all the packets that are sent from or intended for that node will be dropped. In order to ensure packet-level reliability, the protocols should be designed in such a way that packets being dropped should be identified and retransmitted. This will help mitigate packet losses, thereby increasing reliability.

- Congestion:

  When the rate at which the events are generated is more than the rate at which nodes forward the data, congestion occurs in the network. The network will have more traffic flowing and the sensor nodes will start buffering the packets if they are not able to transmit them immediately after receiving them. However, since the buffer size of any node is limited, any packet that arrives at the time when the buffer is full will be dropped. Also, as the medium around the sensor nodes is congested, more packet transmissions result in collisions, thereby dropping the packets. The protocol design should provide infrastructure to handle the network in congested scenarios.

- Packet Loss Identification:

  Detecting packet loss can be done at various levels. Nodes sending data packets can detect packet loss by using ACK/NACK messages sent by receivers. Receivers can detect packets based on timers or by means of packet sequence number (*i.e.,* whenever packet receives out of sequence number, it assumes the expected sequence packet is lost). The protocol that provides reliability must identify the packet losses as it enables the protocol to recover the lost packets efficiently.

- Packet Loss Recovery:

  Because of the nature of error-prone wireless networks, packets will be dropped because of link errors, node failures, and etc. Unlike the efficient end-to-end packet loss recovery, as in wired networks, most of the WSNs use hop-by-hop lost packet recovery. However, to provide packet-level reliability, packet recovery needs to be done at the link level, thus ensuring all packets flowing through each of the links are reliably delivered.

- Energy Efficiency:

  As the sensor networks are energy constrained, in order to extend the network lifetime, it is very important to reduce energy consumed by the nodes. As most of the nodes deplete their energy by idle listening when there is no traffic, the energy consumed due to idle listening is comparatively high. Also, due to redundancies in the deployment of the sensor network, energy consumption increases. In designing an energy-efficient protocol, these energy wastages must be considered and reduced.

- Scalability:

  As sensor networks contain a very large number of sensor nodes, networks should be scalable enough to provide packet-level reliability. Protocols need to be distributed in nature in order to reduce the overhead caused in the case of very large networks.

Considering the above challenges, an energy efficient *monitor*-based protocol to provide packet-level reliability is proposed in this thesis. In order to measure the performance of the protocol, the following standard metrics were chosen.

- Packet Delivery Ratio:

  In order to measure packet-level reliability, the ratio of the total number of packets successfully delivered to the base station to the total number of packets generated is measured. This packet-delivery ratio gives the measure of reliable packet transfer.

- Energy Consumed:

  To identify the energy efficiency of the proposed protocol, the total energy consumed in the network is calculated. The lower the energy consumption value, the better the energy efficiency of the protocol.

- Throughput:

  The network performance in terms of throughput is critical. Throughput as a measure of network capability in delivering the data packets will be calculated over a period of time. A higher throughput implies better performance of the network.

- Delay:

  Another standard metric of network performance is Delay. Depending on the nature of applications of the sensor networks, delay in the network plays a crucial role. Average delay is measured to identify the latency in forwarding the packets to the base station. Depending on the mechanisms employed for providing energy efficiency and packet-level reliability, there exists an overhead and trade-off in terms of delay.

3.2 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 subsetting 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.

3.3 Monitor Configuration for Reliable Data Transfer

3.3.1 Initial Setup

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:

1. Choosing a subset of nodes and identifying active/inactive nodes
2. Constructing a data-gathering tree
3. 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

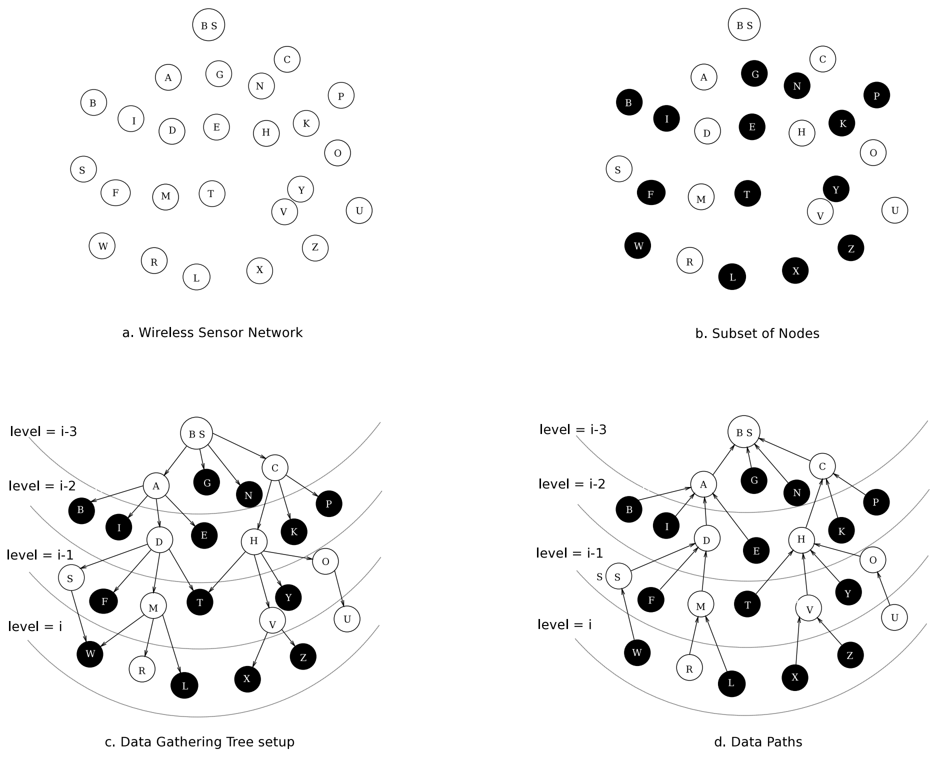


Figure 3.1: Data Gathering Tree

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 3.1. The WSN is represented by Figure 3.1.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 3.1.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 3.1.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 3.1.c represent the hop level from the Base Station.

To prevent collisions amongst these broadcasts, a node will randomly chose a time to broadcast the packet within a reasonable time frame. In order to restrict the broadcast messages from going in a loop, each node broadcasts these messages only three times. This is to make sure that even if some of the broadcast packets are dropped as a result of collisions another broadcast message would possibly make it to the receiving node. The final data path is represented in Figure 3.1.d; the arrows point to the *forwarder* of each node. For example, in Figure 3.1.d, active data paths are $\{R \rightarrow M \rightarrow D \rightarrow A \rightarrow BS\}$ and $\{V \rightarrow H \rightarrow C \rightarrow BS\}$, while the *forwarder* of inactive Node $G \rightarrow BS$, $I \rightarrow A$ and $K \rightarrow C$ and so on.

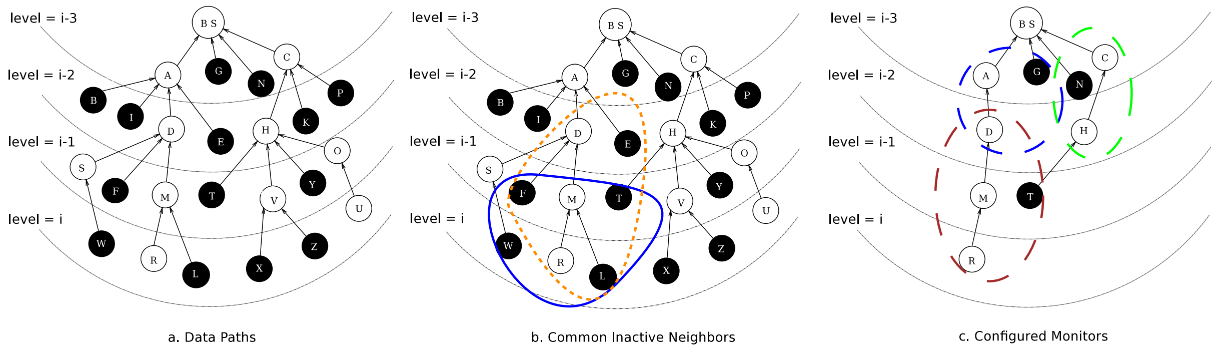


Figure 3.2: Monitor Configuration

3.3.2 Monitor Configuration

After the data paths are setup as in Figure 3.2.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-REQUEST-MESG*. 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 3.2.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 3.2.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 of 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 otherwords, 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 $R \rightarrow M$ 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 $R \rightarrow M$ and $M \rightarrow D$.

Inactive neighbors of R, except node T that receive this *MONITOR-NOTIFICATION-MESSAGE* 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 3.2.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.

Additional *monitors* are not configured to support the existing *monitors*. For example, links $T \rightarrow H$, $G \rightarrow BS$, and $N \rightarrow BS$ are not monitored. Providing an additional layer would overload the *monitor*. Our experiments confirm a marginal improvement in network performance when there is an additional layer of *monitors*. It also has to be noted that the nodes that are one hop away from the Base Station are also not configured with a *monitor* node. From the Figure 3.2.b, active nodes A and C are not configured with a monitor. Since these nodes have the Base Station as their *forwarder*, it is not necessary to have a *monitor*. This is based on the assumption that if the link to the Base Station fails, having a *monitor* that would again forward the data to the Base Station would also fail. After successful configuration of *monitors* for all nodes in the active data path, inactive nodes that are not *monitors* turn off their radios to save energy.

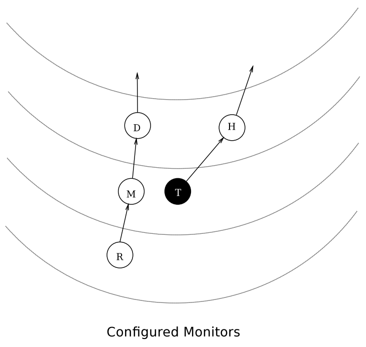


Figure 3.3: Monitor Aided Loss Detection/Recovery

3.3.3 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:

1. Collisions.
2. Congestion at the receiver node (queue overflow).
3. Link failures.
4. 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.

In Figure 3.3, the *monitor* node T overhears the communication between node $R \rightarrow M$, and $M \rightarrow D$. Whenever node R sends a data packet to node M, monitor T caches the packet. Node T will wait for enough time for node M to forward the data packet. If the same packet cached at T is transmitted from M, node T clears the packet from the cache, as this means that the data packet from R has been delivered to M. In the case where T monitors only the link R and M, the packet is deleted from the buffer of T. However, in the scenario in which node T also monitors the link between M and D, instead of deleting the packet from its buffer, the packet is labelled as sent from node M after node M transmits the message. Similarly, for the data transmission from node M to node D, the *monitor* node T overhears the communication, and if the node D transmits the same packet, node T deletes the packet from its buffer as the packet is now successfully delivered from M to D, resulting in a successful delivery of packets between links $R \rightarrow M$ and $M \rightarrow D$.

However, due to link losses, sudden node failures or collisions, if node M did not transmit the message that was sent from R, node T, after waiting for a reasonable amount of time assumes that the packet sent from node R is lost and did not reach node M. If the network is congested, it is possible that many packets are queued up at node M, which results in processing the data packets it received. Although the packet reaches node M, because of its inability to deliver the packet within the expected time, *monitor* node T assumes that the packet sent from M is lost and initiates the loss-recovery mechanism that would re-route the packet to its configured *forwarder* H in a different data path. After a certain amount of time node M transmits the same packet. This results in sending duplicate packets to the Base Station. Though this is an overhead in terms of extra packets in the network, at least one of these packets reach the Base Station. Since our primary objective is to deliver the packets reliably to the Base Station, *monitor* node T does not consider the packet sent from M after exceeding the estimated time as being successfully delivered between the link $R \rightarrow M$. Though these types of duplicates are not reduced, at each node a mechanism to eliminate sending duplicate packets is employed, which makes sure that the node does not resend already sent packets.

3.3.4 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.

In scenarios where the *forwarder* fails to send the CTS, the sender node retries for a maximum of 6 times (MAC specific) and drops the packet after 7 attempts. In Figure 3.3, node R sends the RTS to node M for sending a data packet. If node M fails to send a CTS, node R would have dropped the data packet after the maximum retry attempts. But due to the cross-layered architecture of our protocol, the MAC layer is modified to improve the packet-level reliability, which alters the behavior of the network. The packet drops due to an inability to gain Channel Access were not addressed in [5, 10, 12, 13, 22]. In our protocol, at the MAC layer of each active node, *monitor* information is shared because the node R considers the *monitor* node T as the alternate *forwarder* when it is unable to gain channel access with its *forwarder* node M. When node R is unable to receive CTS from node M after the maximum retry attempts, node R forwards the data packet to the *monitor* node T directly without dropping it. This results in a significant improvement in packets being delivered to the Base Station. This mechanism helps the *monitor* node to receive the packet sent by the sender directly without passive listening. The *monitor* node T, on receiving such data packets directly, forwards them to its *forwarder* node H that is in a different data path.

3.3.5 Packet-Loss Recovery

After the *monitor* node identifies packet losses, it initiates a packet-loss recovery mechanism for reliably delivering the packets. In Figure 3.3 *monitor* T caches the packet from R, and after detecting a packet loss, it re-routes the packet to its *forwarder* H which is in a different active data path. As the *forwarder* of the *monitor* node H, is not any of its monitored nodes, this helps avoid loops in data forwarding; This way monitor T, instead of attempting to resend the packet to M, which might again be lost, it chooses a data path that is different from M. This way, lost-packet recovery is handled with *monitors*.

3.3.6 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

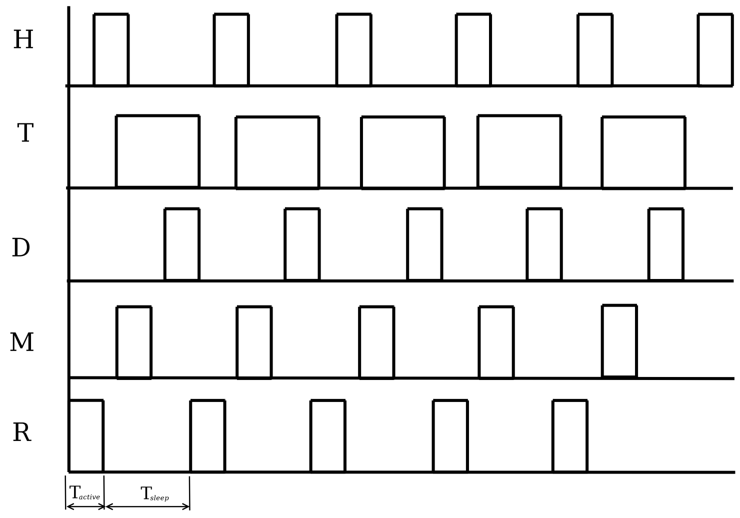


Figure 3.4: Staggered and Synchronized Duty Cycles

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.

From Figure 3.2.c, consider the nodes R and M, in which node R forwards the data to M. If duty cycles are introduced, node R and M go to wake-up/sleep cycles periodically. However, if these are not synchronized, node M might be sleeping when node R has data to send. In such cases node M must be awake when node R sends the data, rather the node R sends the data only when node M is awake. This can be achieved by having the same wake-up/sleep cycles for both R and M. But if an active data path is considered, $R \rightarrow M \rightarrow D \rightarrow A \rightarrow BS$, all the nodes R, M, D, and A are to be awake/sleeping at the same time. When a packet is being forwarded from R to Base Station via M, D, and A, due to the same wake-up/sleep cycles, node R has to awake unnecessarily for the transmissions of M, D and A. Similarly, node M has to be awake for the transmissions of D and A, and D has to be awake for transmissions of A. The amount of energy wasted during this unnecessary wake-up can be reduced by employing staggered 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 3.4 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.

When the nodes in the network have *monitor* support and duty cycles are in place, the duty cycles for the *monitor* nodes are to be chosen carefully. Consider the nodes as shown in Figure 3.3.a in which the links between $R \rightarrow M$ and $M \rightarrow D$ are monitored by the *monitor* T. Nodes R, M, and D can have normal wake-up/ sleep cycles. However, since the *monitor* node T has to overhear the transmissions on links $R \rightarrow M$ and $M \rightarrow D$, having normal duty cycles depending on hop level of the monitor node is not suitable. In such cases, assigning a duty cycle for node T, the same as that of node M (since both of them belong same hop level) would not allow T to listen the transmissions of node D, which would affect the normal operation of *monitors* and would in-turn affect the network performance. To avoid such a possibility, the duty cycle for the *monitor* nodes must be chosen independently of the normal duty cycles assigned for other nodes. Since *monitor* T has to listen to transmissions of node R, node M, and node D, the wake-up period of the *monitor* node needs to be long enough for it to listen to all three of the transmissions.

Figure 3.4 shows the wake-up period of node T, which is long enough to listen all the transmissions from R, M, and D. In order to synchronize with the *monitor's forwarder* H, node T's sleep cycle should be reduced, which means the *monitor* nodes have higher duty cycles compared to that of active nodes. As the *monitor* nodes can monitor links in at most one hop level above and below its hop level, duty cycles are to be assigned carefully to enable the *monitors* to overhear the data transmission and forward the data during link failures. As the *monitor* nodes monitor a maximum of two links, they are assigned a duty cycle that is one and half times that of an active node's duty cycle. This ensures that the *monitor* node is awake for the entire period of all the transmissions of the nodes it is monitoring, and thus would not affect the reliable packet delivery by the *monitor* nodes. In the case of *monitor* node

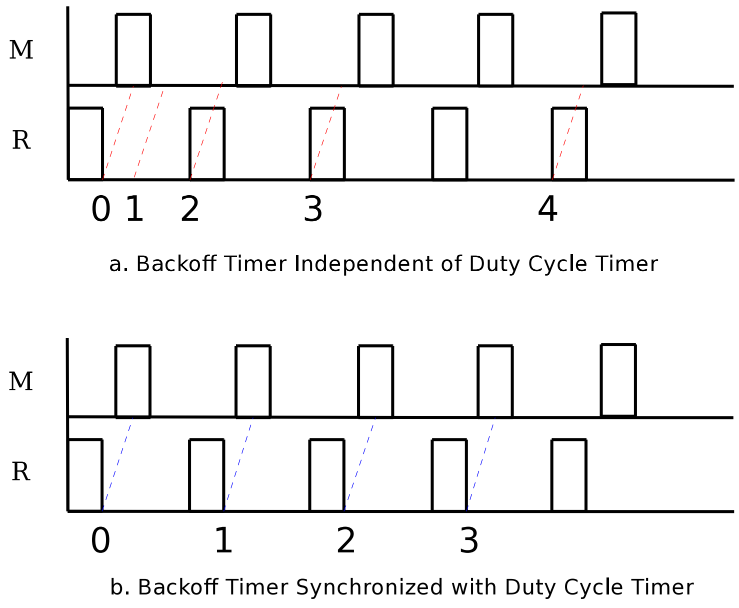

M
R
0 1 2 3 4
a. Backoff Timer Independent of Duty Cycle Timer
M
R
0 1 2 3
b. Backoff Timer Synchronized with Duty Cycle Timer

Figure 3.5: Back-Off Timer Synchronization

T monitoring only a single link, the duty cycle for the *monitor* should be equal to duty cycle of the receiver in the monitored link. In other words, if T is monitoring only link $R \rightarrow M$, the dutycycle of T should be the same as M and if T is monitoring only link $M \rightarrow D$, the dutycycle of T should be the same as that of D.

3.3.7 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.

Consider two nodes, R and M, as shown in Figure 3.5.a. The dotted line shows the retransmission of the RTS frame when the CTS is not received and the back-off timer has expired. When duty cycles are employed, and if the back-off timer is not synchronized with the active and sleep cycles, the retransmitted RTS frame may not reach the *forwarder* M. Then, node R retries the RTS frame when node M is sleeping. This leads to an increase in the congestion window, which in turn increases the delay and reduces the packet-delivery ratio. In order to overcome this, our protocol modifies the Native back-off timer behavior of the MAC to meet our goals. When the back-off timer is fired, node R checks if the f*forwarder* node M is awake or not. If node M is awake, node R retransmits the RTS frame. However, if node M is sleeping when the back-off timer is fired, instead of retransmitting the RTS frame, node R waits for the next wake-up slot and then transmits the RTS frame. Figure 3.5.b shows the behavior when the back-off timer is synchronized with wake-up/sleep cycles. This synchronization helps improve the packet-delivery ratio and reduce the latency.

3.4 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.

3.4.1 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 $E_t x$.
The total transmit energy E_{TM} for x nodes with *monitors* lies in the range

$$E_t(3x-1)/2 \leq E_{TM} \leq E_t(2x-1) \tag{3.1}$$

Similarly, total reception energy E_{RM} for x nodes with *monitors* lies in the range

$$E_r(3x-1)/2 \leq E_{RM} \leq E_r(2x-1) \tag{3.2}$$

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*.

3.4.2 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 $P_m > P_n$, since *monitors* provide extra infrastructure in recovering the dropped packets.
If i is the percentage increase in packet-delivery ratio, then

$$Pm = Pn + (iPn) = Pn(1+i) \tag{3.3}$$

Let the average hop length without *monitors* be l, and the average hop length with *monitors* be $2l$ (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,

$$T_n = (lt)Pn \tag{3.4}$$

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

$$T_m = (2lt)P_m \tag{3.5}$$

Using (3.3) and (3.4) , we have

$$T_m = 2T_n(1 + i) \tag{3.6}$$

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

CHAPTER 4

PERFORMANCE EVALUATION

To evaluate the performance, the proposed protocol is implemented in the *ns-2* simulator [18]. Extensive experiments were conducted in order to test the performance of the *monitor*-based approach. As the proposed protocol extends the concept of *monitors* presented in [13], first we provide a comparison of our approach with the previous *monitor*-based approach in [13] in terms of throughput and packet-delivery ratio. Second, we compare our approach to the one without the support of *monitors* in terms of standard metrics such as throughput, packet-delivery ratio, energy consumption, and average latency. Finally, as the proposed protocol employs staggered and synchronized duty cycles for energy efficiency, we evaluate how duty cycles and the *monitors* approach together impact the performance of the network.

4.1 Simulation Setup

The simulations were run with the simulation parameters as mentioned in Table 4.1. A subset of 30 nodes were 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 number 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 4.1: Simulation Parameters

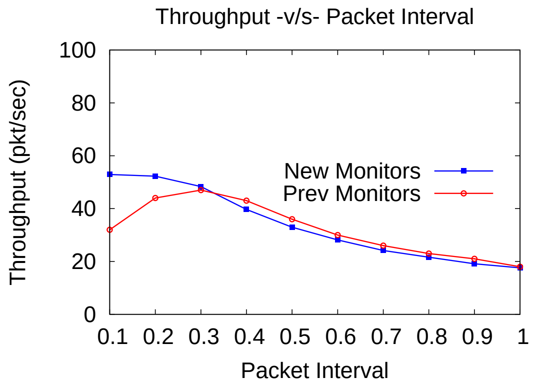
Parameter	Value
Area	$1000m$ x $1000m$
Transmission radius	$250m$
Total number of nodes	150
Number of sources	20
Data Packet size (bytes)	516
Packet interval rate	$0.1(5KB/s)$ to $1(0.5KB/s)$
Transmit Power (W)	0.01488
Receive Power (W)	0.01250
Idle Power (W)	0.01236
Sleep Power (W)	0.000016

4.2 Comparison with Previous *Monitor*-Based Approach

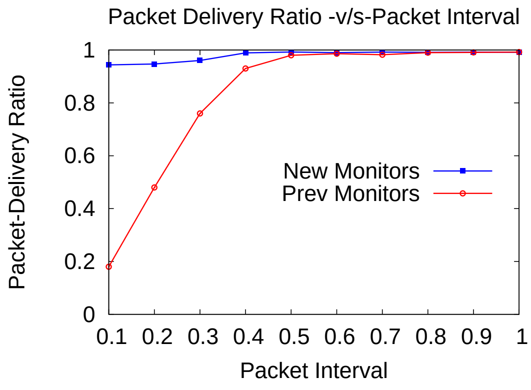
For a meaningful comparison of the proposed protocol with [13] in persistent and transient congestion scenarios, metrics such as throughput and packet-delivery ratio were considered.

4.2.1 Persistent Congestion

The proposed protocol is compared with the previous *monitor*-based approach [13] to evaluate the network performance under persistent congestion. Figure 4.1(a) and Figure 4.1(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

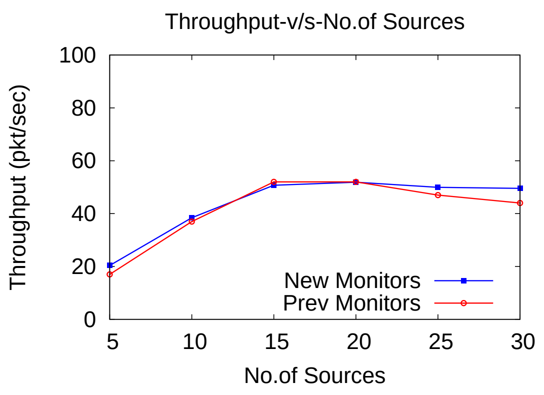


(b) Packet Delivery ratio v/s Packet Interval

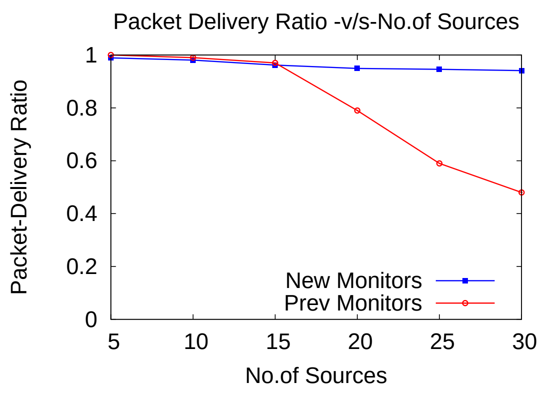
Figure 4.1: Comparison with original *monitors*-based approach in the case of persistent congestion

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 4.1(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*,



(a) Throughput v/s Number of Events



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

Figure 4.2: Comparison with original *monitors*-based approach in the case of transient congestion

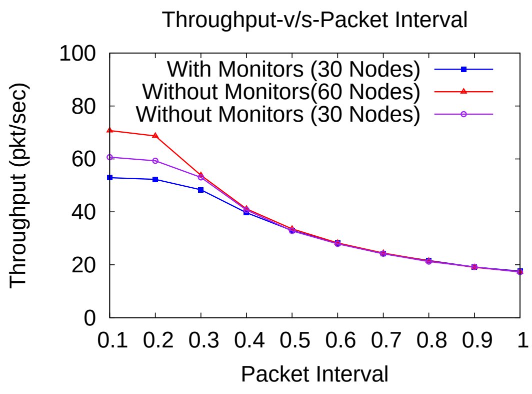
which results in the shift of the knee point. Notice that our extended *monitor*-based protocol performs better than the previous *monitor*-based approach when subjected to persistent congestion.

4.2.2 Transient Congestion

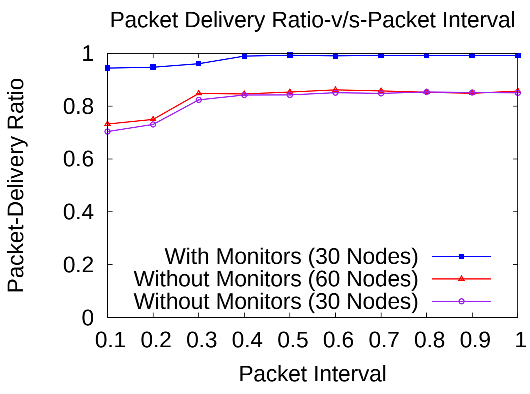
To evaluate our protocol performance under transient congestion situations, varying event centers were chosen ranging from 5 to 30 with a short term CBR traffic rate of $2.5KB/s$ and packet size of 516 bytes. Figure 4.2(a) and Figure 4.2(b) show the performance comparison of our protocol in terms of throughput and packet-delivery ratio. Throughput of our protocol peaks when the events are 20, and reduces slightly with more events due to congestion. Congestion is observed after reaching a peak of 15 events for the previous *monitors*-based approach, and throughput falls down after that. Our protocol achieves better throughput when subjected to congestion with higher numbers of events. Both protocols show about the same throughput when the even centers are less than 15.

As our protocol design benefits from subsetting of the nodes, congestion is reduced even with an increase in event centers. Our protocol is consistent in providing a higher packet-delivery ratio even with more event centers when compared to the previous *monitors*-based approach. By having *monitors* as alternate forwarders, in case of packet drops due to congestion, our protocol design is tolerant to highly congested network scenarios, thereby increasing packet-delivery ratio.

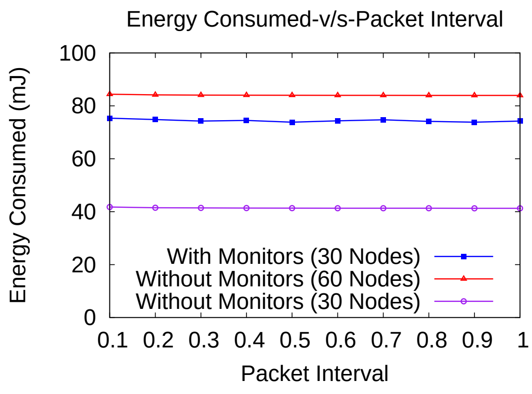
4.3 Comparison without Duty Cycles



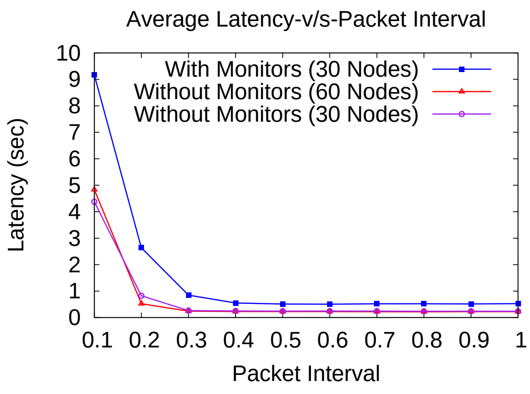
(a) Throughput v/s Packet Interval



(b) Packet Delivery ratio v/s Packet Interval



(c) Energy Consumed v/s Packet Interval



(d) Latency v/s Packet Interval

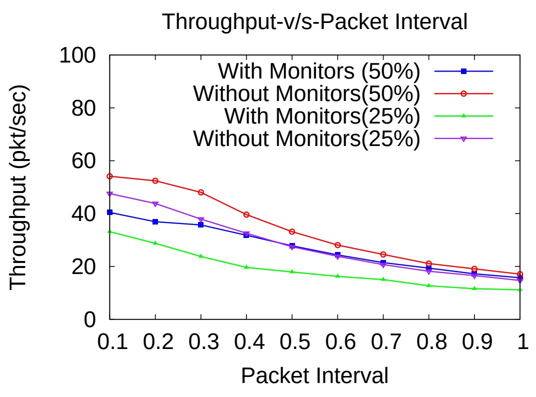
Figure 4.3: Comparison of *monitors*-based approach with and without *monitors*

To evaluate the performance of our protocol with *monitors*, for base-line comparison, we compare our protocol to the case of not having the *monitors* when duty cycles are not employed. In addition to metrics like throughput and packet-delivery ratio, we evaluate our protocol for energy-efficiency and latency. Our *monitor*-based approach consists of 30 active nodes, and *monitors* were chosen from the remaining inactive set of nodes. In order to show a meaningful comparison, we compare our *monitor*-based approach with 30 active nodes to the case without *monitors* but using 60 active nodes. This comparison helps in understanding the effect of subsetting.

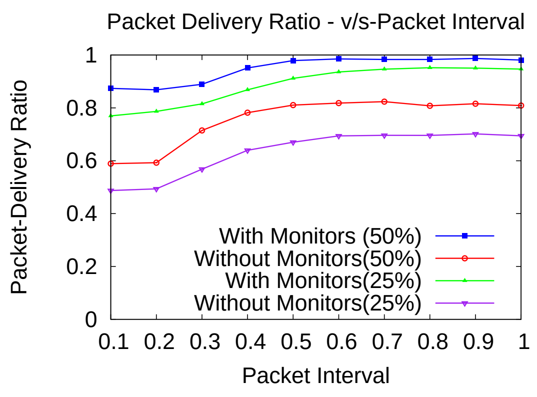
Figure 4.3(a) shows that the throughput is almost the same for both of the protocols during less congestion, while our protocol without monitors shows an improvement in throughput when subjected to congestion. This is because, in highly congested scenarios, our protocol uses monitors for reliable data delivery. As monitors choose alternate routes for forwarding the data, average hop count increases, which in turn increases the packet arrival time. Since throughput is calculated over a period of time, although more packets are reliably transferred, an increase in latency makes the throughput fall in congested scenarios. The packet-delivery ratio of our protocols with *monitors* has a significant improvement over the protocol without *monitors*, as shown in Figure 4.3(b). This is because *monitors* assist during link failures by forwarding the data through alternate paths. The *monitor*-based approach with 30 active nodes has better performance than the approach without monitors with 60 active nodes. With subsetting, the performance of 30 active nodes is close to that of the 60 active nodes approach. Figure 4.3(c) shows that our protocol without monitors has more energy savings when compared to our protocol with monitors. This is due to the additional nodes configured as *monitors*, which increase energy consumption. This increase in energy consumption due to *monitors* is less when compared to the performance of the network with more packets being reliably delivered to the Base Station. In Figure 4.3(d), the average latency of our protocol without *monitors* is slightly lower than the latency incurred by our protocol with *monitors.* As *monitors* assist in forwarding the packets during link failures, though the packets are delayed in reaching the Base Station they are reliably sent to the Base Station. Latency is calculated only for the packets that are delivered to the Base Station. Hence, the additional delivered packets contribute to the increase in latency.

4.4 Comparison with Duty Cycles

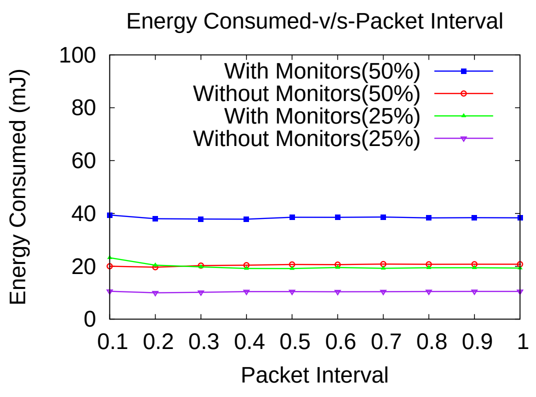
To evaluate the performance of our protocol for energy efficiency while maintaining reliable packet delivery, we evaluate our protocol with duty cycles for both with *monitors* and without *monitors* approaches. Simulation results were taken with different duty cycles of 50% and 25%. Figure 4.4(a) shows that our protocol without *monitors* has better throughput than our protocol with *monitors.* With the duty-cycling mechanism in place, the operational time of the network reduces drastically and hence the throughput, in the case of our protocol with *monitors*, is lower. This reduction in throughput is also because, though the packets are reliably delivered to the Base Station, due to latency incurred with duty cycles, the packet-arrival time increases, which affects the throughput. Figure 4.4(b) shows that during higher congestion, our protocol with duty cycles out performs our protocol without duty cycles. This is because of the reduction in collisions due to duty cycles when subjected to higher congestion. During periods of lower congestion, our protocol with duty cycles shows a comparable packet-delivery ratio while being active for 25% of the entire life time. The energy savings due to availability only a small amount of the time can be seen in Figure



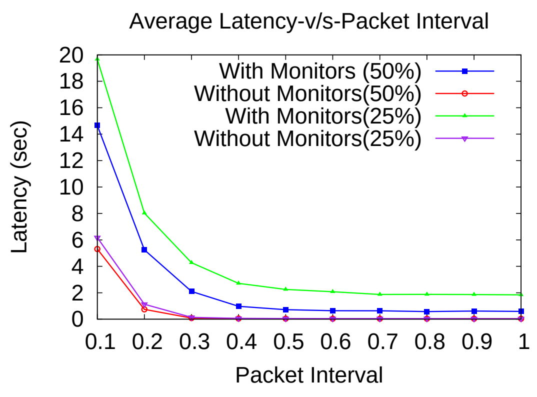
(a) Throughput v/s Packet Interval



(b) Packet Delivery ratio v/s Packet Interval



(c) Energy Consumed v/s Packet Interval



(d) Latency v/s Packet Interval

Figure 4.4: Comparison of *monitors*-based approach with and without Duty Cycles

4.4(c). Our protocol with duty cycles achieves an increase in energy savings when compared to our protocol without duty cycles. As expected, the latency increases with lower duty cycles which can be seen Figure 4.4(d). These graphs show that the *monitor*-based approach with duty cycles is energy efficient with little overhead in terms of delay. However, the packet-delivery ratio is close to the case in which duty cycles are not employed.

CHAPTER 5

CONCLUSIONS

Wireless Sensor Networks are mainly deployed for monitoring purposes in various fields. As many sensors detect the same event and try to forward the data to other nodes, data is redundant and degrades the performance of the network by increasing collisions, delay, and energy consumption. Data Aggregation techniques are used in some applications to reduce the redundancy in forwarded packets. In these techniques, packets are aggregated at intermediate nodes and the correlated data is forwarded from one node to another. As these aggregated packets contain the correlated data, loss of a single packet results in a large amount of data loss. Unless these aggregated packets are reliably transferred to the Base Station, their usefulness may be stifled. Hence, there arises a need for a packet level reliable data transport protocol. Also, as sensor nodes are energy constrained, energy efficiency is one of the primary concerns in designing protocols for these networks.

To enhance the packet-level reliability and reduce energy consumption, we developed 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. [13] 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.

BIBLIOGRAPHY

[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 oppurtunistic 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] M. Dhanaraj, B. S. Manoj, and C. S. R. Murthy. A new energy efficient protocol for minimizing multi-hop latency in wireless sensor networksd. *IEEE PerCom*, pages 117–126, 2005.

[7] Q. Dong. Maximizing system lifetime in wireless sensor networks. *IPSN*, pages 13–19, 2005.

[8] 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.

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

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

[11] C-F. Hsin and M. Liu. Network coverage using low duty-cycled sensors: random and coordinated sleep algorithms. *IPSN*, April 2004.

[12] I.Aron and S.Gupta. A witness-aided routing protocol for mobile ad-hoc networks with unidirectional links. *MDA*, 1999.

[13] J.Wang and S.Medidi. Topology control for reliable sensor-to-sink data transport in sensor networks. *IEEE Intl. Conference on Communications (ICC)*, 2008.

[14] Lindsey, S., Raghavendra, and C. S. Pegasis: Power-efficient gathering in sensor information systems. *IEEE Aerospace Conference Proceedings*, 3:1125–1130, 2002.

[15] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. *IEEE IPDPS*, pages 224–231, 2004.

[16] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay efficient sleep scheduling in wireless sensor networks. *IEEE INFOCOM*, pages 2470–2481, 2005.

[17] M. Medidi and Y. Zhou. Extending lifetime with differential duty cycles in wireless sensor networks. *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1033–1037, November 2007.

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

[19] 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.

[20] S. J. Park and R. Sivakumar. Sink-to-sensors reliability in sensor networks. In *ACM SIGMOBILE Mobile Computing and Communications Review*, volume 7, pages 27 – 28, July 2003.

[21] Joel Ruiz, Jose R.Gallardo, Luis Villasenor-Gonzalez, Dimitrios Makrakis, and Hussein T. Mouftah. Quattro: Qos-capable cross-layer mac protocol for wireless sensor networks. *IEEE GLOBECOM*, 2009.

[22] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. Esrt: Event-to-sink reliable transport in wireless sensor networks. *MobiHoc*, 2003.

[23] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Trans. on Mobile Computing*, pages 70–80, June 2002.

[24] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. pages 70–80, IEEE Transactions on Mobile Computing.

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

[26] T. Stathopoulos and D. Estrin. An information-driven reliability mechanism for wireless sensor networks. Technical Report 16, UCLA, June 2003.

[27] P. Volgyesi, A. Nadas, and A. Ledeczi. Reliable multihop bulk transfer service for wireless sensor networks. In *13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, March 2006.

[28] Y Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for adhoc routing. *ACM MobiCom*, pages 70–84, 2001.

[29] X. Yang and N. Vaidya. A wake-up scheme for sensor networks: Achieving balance between energy saving and end-to-end delay. *IEEE RTAS*, pages 19–26, 2004.

[30] 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.

[31] Y. Zhou and M. Medidi. Energy-efficient contention-resilient medium access for wireless sensor networks. *IEEE Intl. Conference on Communications (ICC)*, pages 3178–3183, June 2007.

[32] Y. Zhou and M. Medidi. Sleep-based topology control for wakeup scheduling in wireless sensor networks. *IEEE Intl. Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, pages 304–313, June 2007.