ENHANCING QUALITY OF SERVICE IN LOW DUTY-CYCLE WIRELESS SENSOR NETWORKS

A PROJECT REPORT

Submitted by

SHERIN GEORGE

in partial fulfilment for the requirement of award of the degree

of

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE 641 049
(An Autonomous Institution Affiliated to Anna University, Chennai)

APRIL 2013

BONAFIDE CERTIFICATE

Certified that this project work titled **"ENHANCING QUALITY OF SERVICE IN LOW DUTY-CYCLE WIRELESS SENSOR NETWORKS"** is the bonafide work of Ms. SHERIN GEORGE (1120108017), who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other students.

Prof. N. JAYAPATHI, M.Tech.,	Mr.K.SIVAN ARUL SELVAN, M.E., (Ph.D)
HEAD OF THE DEPARTMENT	SUPERVISOR
Professor	Associate Professor
Dept. of Computer Science and Engineering	Dept. of Computer Science and Engineering
Kumaraguru College of Technology	Kumaraguru College of Technology
Coimbatore- 641 049	Coimbatore- 641 049

Submitted for the Project Viva-Voce examination held on _____.

---------- ----------

Internal Examiner **External Examiner**

ABSTRACT

Recent technologies in wireless communications have enabled the development of low-cost Wireless Sensor Networks (WSNs). Wireless Sensor Networks usually have limited energy and transmission capability and hence turn active only when they perform sensing tasks or communications and remain dormant during idle periods. Broadcasting is one of the essential services in Wireless Sensor Networks (WSN) and is used to propagate messages from a node or a source to all other nodes in the network. The control messages have to be broadcasted from source to other nodes during network configuration. Also, to query the nodes about an event, message has to be broadcasted to all the nodes. The broadcasting is also used to propagate routes to the nodes. Hence implementing an effective broadcast service, which is simple, reliable and energy-efficient with less overhead, is critical for the effective functioning of Wireless Sensor Networks. In this project, the Quality of Service of the broadcasting is enhanced by reducing the message cost and the time cost for low duty-cycle WSNs. This project provides two solutions, namely, centralized dynamic and distributed solution which improves the Quality of Service of broadcasting.

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project. I express my profound gratitude to **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E, Chairman**, **Dr.B.K. Krishnaraj Vanavarayar**, **Co-Chairman, Mr. M. Balasubramaniam, M.Com, M.B.A, Correspondent, Mr.Sankar Vanavarayar**, **M.B.A., PGDIEM, Joint Correspondent** and **Dr.S.Ramachandran, Ph.D., Principal** for providing the necessary facilities to complete my project.

I take this opportunity to thank **Prof.N.Jayapathi, M.Tech**., Head of the Department, Department of Computer Science and Engineering, for his support and motivation. Special thanks to my Project Coordinator **Dr.V.Vanitha, M.E., Ph.D.**, Senior Associate Professor, Department of Computer Science and Engineering for arranging brain storming project review sessions.

I register my sincere thanks to my guide **Mr. K. Sivan Arul Selvan, M.E., (Ph.D.),** Associate Professor, Department of Computer Science and Engineering, my project supervisor. I am grateful for his support, encouragement and ideas. I would like to convey my honest thanks to all **Teaching** and **Non Teaching Staff members** of the department and my classmates for their support.

I dedicate this project work to my **Parents** for no reasons but feeling from bottom of my heart that without their love this work wouldn't be possible.

-SHERIN GEORGE

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACK	-	Acknowledgment
ADB	-	Asynchronous Duty-Cycle Broadcast
ADC	-	Analog to Digital Converter
AEA	-	Adaptive Election Algorithm
CDMA	-	Code Division Multiple Access
DSF	-	Dynamic Switch Forwarding
MEMS	-	Micro Electro Mechanical Systems
PBBF	-	Probability Based Broadcast Forwarding
QoS	-	Quality of Service
RI-MAC	-	Receiver Initiated Medium Access Control
SEP	-	Schedule Exchange Protocol
TRAMA	-	TRaffic Adaptive Medium Access
WSN	-	Wireless Sensor Network

CHAPTER 1

INTRODUCTION

The Wireless Sensor Networks (WSNs) are highly distributed networks of small, light-weight wireless nodes, deployed in large numbers to monitor the environment or system. Monitoring the system includes the measurement of physical parameters such as temperature, pressure, or relative humidity and co-operatively passes their data to the main location (sink). The advancements in Micro-Electro Mechanical Systems (MEMS) have made building sensors possible.

The sensor networks are used in a variety of applications. The military application include battlefield surveillance and monitoring, guidance systems of intelligent missiles and detection of attacks. The WSNs are also used for forest fire and flood detection, habitat exploration of animals, patient diagnosis and monitoring. The WSNs are also making their way into a host of commercial applications at home and in industries.

1.1 SENSOR NODES

The Wireless Sensor Networks consists of sensor nodes ranging from few hundred or even thousands depending on the application. Each sensor node may be connected to one or more other sensor nodes. Each node of the sensor networks consists of four units: the sensor unit, the processing unit, the transceiver unit and the power unit. In addition to the above units, a wireless sensor node may include a number of application-specific components, for example a location detection system or mobilizer; for this reason, many

commercial sensor node products include expansion slots and support serial wired communication.

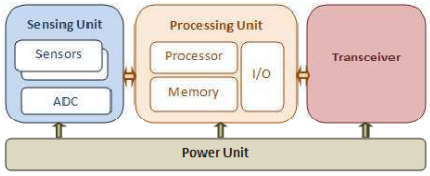

Fig 1.1 Components of Sensor Node

Sensing Unit: The main functionality of the sensing unit is to sense or measure physical data from the target area. The analog voltage or signal is generated by the sensor corresponding to the observed phenomenon. The continual waveform is digitized by an Analog-to-Digital Converter (ADC) and then delivered to the processing unit for further analysis. The sensing unit is a current technology bottleneck because the sensing technologies are much slower than those of the semi-conductors.

Processing Unit: The processing unit which is generally associated with a small storage unit manages the procedures that make the sensor nodes collaborate with the other nodes to carry out the assigned sensing tasks.

Transceiver Unit: There are three deploying communication schemes in sensors including optical communication (Laser), Infrared, and Radio-Frequency (RF). Laser consumes less energy than radio and provides high security, but requires line of sight and is sensitive to atmospheric conditions. InfraRed uses diffuse

light or directed light but are limited in its broadcasting capacity. RF is the most easy to use but requires antenna. Various transmission strategies have been developed such as modulation, filtering, and demodulation. Amplitude modulation, which assigns different amplitudes to the binary values (0, 1) and frequency modulation, which assigns frequencies to the binary values (0, 1) are standard mechanisms. Amplitude modulation is simple but susceptible to noise.

Power Unit: One of the most important components of a sensor node is the power unit. Every sensor node is equipped with a battery that supplies power to remain in active mode. Power consumption is a major weakness of sensor networks. Any energy preservation schemes can help to extend sensor's lifetime. Batteries used in sensors can be categorized into two groups; rechargeable and non-rechargeable. Often in harsh environments, it is impossible to recharge or change a battery. While individual sensors have limited sensing region, processing power and energy, networking a large number of sensors give rise to a robust, reliable, and accurate sensor network covering a wider region. The network is fault-tolerant because many nodes sense the same events.

1.2 WIRELESS SENSOR NETWORKS ARCHITECTURE

The design of Wireless Sensor Networks is influenced by the factors such as scalability, fault tolerance and power consumption. The basic kinds of WSN architecture are layered and clustered.

1.2.1 Layered Architecture

A layered architecture has a single powerful base station (BS), and the layers of sensor nodes around it correspond to the nodes that have same hop-count to the BS. The Fig 1.2 describes the layered architecture.

Applications of layered architecture:

- In-building wireless backbones
- Military sensor-based infrastructure

Unified Network Protocol Framework (UNPF)

UNPF is a set of protocols for the implementation of a layered architecture for WSNs. UNPF integrates three main operations in its protocol structure: Network initialisation and maintenance Protocol, MAC (Medium Access Control) protocol and routing protocols.

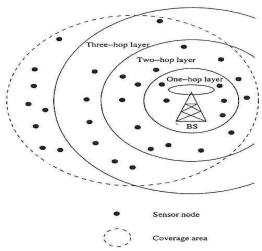

Fig 1.2. Layered Architecture

Network initialisation and maintenance Protocol

- Organises the sensor nodes into different layers.
- BS broadcasts their identifier (ID) using a known CDMA (Code Division Multiple Access) common control channel.

- All nodes which hear this broadcast record the BS ID.
- Nodes send a beacon signal with their own IDs.
- Nodes at single-hop distance form layer one.
- BS now broadcasts control packet with layer one node IDs.
- Layer one nodes inform the BS of layer two nodes.
- BS again broadcast its ID to layer two nodes.
- Layered structure build by successive rounds of beacons.

MAC protocol

- Each node is assigned a reception channel by BS channel allocation.
- Channel reuse is such that collisions are avoided.
- Nodes schedule the transmission lots for all neighbours and broadcasts the schedules- channel scheduling.

Routing Protocol

- Downlink from BS is by direct broadcast on the control channel.
- Enables multi-hop data forwarding from nodes to the BS.
- Node to which a packet is to be forwarded is selected considering the remaining energy of the nodes.

1.2.2 Clustered Architecture

The clustered architecture organises the sensor nodes into clusters, each governed by a clustered-head. The nodes in each cluster are involved in message exchanges with their respective cluster-heads, and these heads send messages to a BS, which is usually an access point connected to a wired network.

Clustered architecture is especially useful for sensor networks because of its inheritance suitability for data fusion. The data gathered by all members of the cluster can be fused at the cluster-head, and only the resulting information need to be communicated to the BS. Sensor networks should be self-organising, hence the cluster formation and election of cluster-heads must be an autonomous, distributed process. This is achieved through network layer protocols such as the Low-Energy Adaptive Clustering Hierarchy (LEACH).

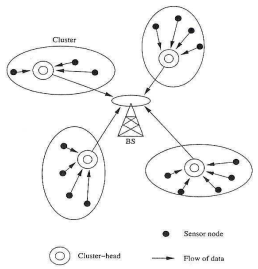

Fig 1.3 Clustered architecture

Low-Energy Adaptive Clustering Hierarchy (LEACH)

- LEACH is an example for clustered architecture.
- Minimizes energy dissipation in sensor networks.
- Randomly selects nodes as cluster-heads and performs periodic re-election.
- The operation of LEACH is divided into rounds.
- Each round is divided into two phases in LEACH: set-up and steady phase.

Set-up phase

- A cluster head advertises its neighbours using a CSMA MAC (Carrier Sense Multiple Access Medium Access Control).
- Surrounding nodes decide which cluster to join based on the signal strength of these messages.
- Cluster heads assign a TDMA (Time Division Multiple Access) schedule for their members.

Steady phase

- All source nodes send their data to their cluster heads.
- Cluster heads perform data aggregation/fusion through local transmission.
- Cluster heads send them back to the BS using a single direct transmission.
- After a certain period of time, cluster heads are selected again through the set-up phase.

1.3 ISSUES AND CHALLENGES IN DESIGNING A WSN

The WSN has many design issues and challenges. A few are discussed below.

1. **Random Deployment**: Sensor nodes are randomly deployed and hence do not fit into any regular topology. The setup and maintenance of the network should be entirely autonomous.
2. **Lack of Infrastructure**: Therefore, all routing and maintenance algorithms need to be distributed.
3. **Limited Energy**: As the sensors rely on their battery for power, which cannot be replaced or recharged, the available energy should be used efficiently.

4. **Hardware Design and Energy efficiency**: The micro-controller, operating system and application software should be designed to conserve power.
5. **Synchronization**: Sensor nodes should be able to synchronize with each other in a completely distributed manner so that TDMA schedules can be imposed and ordering of the events can be performed without ambiguity.
6. **Adaptability**: A WSN should be capable of adapting to changing connectivity due to the failure of nodes or new nodes powering up.
7. **Real-Time communication**: WSN must be supported through provision of guarantees on maximum bandwidth or other QoS parameters.
8. **Secure Communication**: The communication of WSNs in military applications should be made secure.

1.4 DATA GATHERING

The objective of the data-gathering is to transmit the sensed data from each sensor node to a BS. One round is defined as the BS collecting data from all sensor nodes once. The goal of algorithms which implement data gathering is to maximize the number of rounds of communication before the nodes die and the network becomes inoperable. This means minimum energy should be consumed and the transmission should occur with minimum delay, which is a conflicting requirements. A few data gathering algorithms are discussed below.

1.4.1 Direct Transmission

All sensor nodes transmit their data directly to the BS. This is expensive is BS is far away from some nodes. Also care should be taken in order to enable nodes to transmit during their turn to avoid collisions. But this in turn imposes access delay.

1.4.2 Power-Efficient Gathering for Sensor Information Systems

- The topology information is available to all nodes.
- Any node which has one-hop distance to the BS is selected as the leader.
- A chain of nodes starting from the farthest node is constructed.
- At every node data fusion is carried out.
- The leader finally transmits one message to the BS.
- The leadership is carried out in a sequential order.

1.4.3 Binary Scheme

- Chain-based scheme, which classifies nodes into different levels.
- All nodes which receive message rise to the next.
- The number of nodes is halved from one level to the next.
- Possible when CDMA, so that transmissions of each level can take place simultaneously.

1.4.4 Chain-based Three-level Scheme

- For non-CDMA sensor nodes, chain is constructed as in PEGASIS.
- The chain is divided into a number of groups to enable simultaneous transmissions with minimum interference.
- Within group nodes transmit one at a time.
- One node from each group aggregates data from all group members and rises to the next level.
- In second level, all nodes are divided into two groups.
- In third level, message exchange between one node from each group of second level.
- Finally leader transmits a single message to the BS.

1.5 DATA DISSEMINATION (BROADCASTING)

Data dissemination or broadcasting is a process by which queries or data are routed in the WSN. A node (sink) which is interested in an event seeks information about it. An interest is a descriptor for a particular kind of data or event that a node is interested in, such as temperature, intrusion, or presence of bio-agents. For every event that a sink is interested in, it broadcasts its interest to its neighbours and periodically refreshes its interest. The interest is propagated across the network and every node maintains a cache of events to be reported. The intermediate nodes maintain a data cache and can aggregate the data or modify the rate of reporting the data. The paths used for data propagation are modified by preferring the shortest paths and deselecting the weaker or longer paths. The basic idea of broadcasting is made efficient and intelligent by different algorithms for interest and data routing.

1.5.1 Need for Broadcasting

- Broadcasting is the way the sink or base station can propagate information to all nodes.
- During network configuration control information has to propagate throughout the network.
- Upon observing an event, queries have to be propagated across the network.

1.5.2 Basic Techniques in Broadcasting

Two basic techniques of broadcasting are flooding and gossiping.

Flooding

In flooding, each node which receives a packet broadcasts it if the maximum hop-count of the packet is not reached and the node itself is not the

destination of the packet. This technique does not require complex topology maintenance or route discovery algorithms. The flooding has some disadvantages such as propagation of duplicate message to the same node, overlapping regions of coverage and lack of resource consideration.

Gossiping

Gossiping is a modified version of flooding, where the nodes do not broadcast a packet, but sent it to a randomly selected neighbour. This avoids the problem of implosion but it takes a long time for message to propagate throughout the network. Though gossiping has considerably lower overhead than flooding, it does not guarantee that all nodes of the network will receive the same message. It relies on the random neighbour selection to eventually propagate the message throughout the network.

1.6 LITERATURE SURVEY

In this section, the papers related to broadcasting in Wireless Sensor Networks are discussed. From that, the QoS in broadcasting is focused and studied.

1.6.1 Smart Gossip

Kyasanur et al (2006) presented a broadcasting technique called Smart Gossip which enumerates the “importance” of each node to propagating the messages. The importance of a node increases if other nodes depend on it for receiving a message. Hence the node which is more important has to broadcast with higher probability. Other nodes which are less important propagate with lower probability. There is no unique subset for dissemination and hence the responsibility of broadcasting is distributed among multiple subsets which in turn balance the load. This also leads to better fault tolerance properties. In Smart Gossip, when a node overhears a message, it tries to infer the sender of message as a parent, child or sibling by applying simple rules. Based on the deduced relationships, Smart Gossip assigns different gossip probabilities to each node. Each node computes a gossip probability its parent should use and announces this probability to its parent. The parent uses this probability to broadcast the forth-coming messages. The gossip probabilities assigned to different nodes are revised periodically considering the changes in topology.

Gossip Probability

Nodes extract information from overheard gossip messages and attempt to deduce whether the sender of the message is a parent, child, or a sibling. The header of each gossip message contains a parent identifier field (pid), and a required gossip probability field ($p_{required}$). When a node forwards a gossip message, it sets the pid field to the identifier of the node from which it received

the gossip message. Initially, when dependencies are not known, each node gossips with probability one. Over time, as nodes learn about their dependencies, the gossip probabilities are refined. Each node computes a gossip probability its parents should use, and sets the$p_{required}$ field to this value. Over time, a node remembers the latest $p^i_{required}$ value announced by each child i. The gossip probability used by a node is given by, p_{gossip}= max($p^i_{required}$). If a node has no children, it still gossips with a low probability to ensure its parents are aware of the presence of the node.

A node may designate the sender of a gossip message as its parent. However, this is not sufficient because when a child forwards a packet; its parent may overhear the packet and incorrectly identify the child as a parent. To avoid this, add an additional check where a node Y, on receiving a packet from node X, checks if X's parent (specified in pid field) is either Y or one among Y's parents. If pid field is set to Y, then Y adds X as its child. If pid field is set to one of Y's parents, then Y adds X as its sibling (i.e., sibling(Y) = {X}). Failing both these conditions, Y adds X as its parent.

Each node maintains four sets - Neighbourset, ParentSet, SiblingSet, ChildSet. During initialization, a node permanently inserts itself into Neighbourset and SiblingSet, while the other two sets are empty. Whenever the node receives a message from any node X with pid field set to some Y, it uses the following concise rules:

• Add X to Neighbourset
• If Y is not in Neighbourset, add X to ParentSet
• else if Y is in ParentSet, add X to SiblingSet
• else if Y is in SiblingSet, add X to ChildSet

The application that utilizes smart gossip can specify its reliability requirement as an average reception percentage, τarp. For example, τarp = 90% implies that the application at the gossip source expects each node in the network to receive at least 90 out of 100 packets sent out from the source, with high probability.

Advantages

- The Smart Gossip keeps track of the previous broadcasts and adaptively adjusts the gossip probabilities.
- It is adaptable to different topologies. As the gossip probabilities are revised often it has the advantage of being distributed and light-weight.

Disadvantages

- This protocol assumes only one message originator at an instant.
- Also it requires every node to be awake during dissemination.

1.6.2 Broadcast Storm Problem

Broadcasting refers to sending messages to other hosts in a network. The broadcast is spontaneous which makes the rebroadcasting result redundant messages. As the rebroadcasting nodes are close to each other this leads to heavy contention. Since the timing of the rebroadcasts is highly correlated it ends in collision. Collectively the problems associated with flooding are known as broadcast storm problem. An approach used to alleviate the broadcast storm problem is to inhibit the hosts from rebroadcasting to reduce redundancy. The contention and collision can be solved by differentiating the timing of

rebroadcasting. A node does not rebroadcast if the expected additional coverage is low. Based on this observation, Ni et al (1999) described five schemes suggested to lessen the effect of broadcast storm problem. The schemes are probability-based scheme, counter-based scheme, distance-based scheme, location-based scheme and cluster-based scheme.

Probabilistic Scheme

In probabilistic scheme, when a node receives a message for the first time, it rebroadcasts with a probability P. This probability decreases when the same message is heard multiple times. To solve the problems of contention and collision insert a small delay before rebroadcasting the message. So the timing can be differentiated.

Counter-based scheme

In counter-based scheme, a counter is used to keep track of the previous messages. When the same message is heard multiple times, the expected additional coverage decreases. The rebroadcasting is inhibited if the counter value is greater than a predetermined threshold value. It facilitates the message to be propagated only when there is reasonable number of nodes hearing it for the first time.

Distance-based scheme

In distance-based scheme, the distance between the sender and the host is computed. When the computed distance is small, there is only little additional coverage for the host. When the distance is large, the additional coverage is larger. A minimum distance threshold is computed so that rebroadcasting is allowed only if the computed distance is smaller than the threshold distance, rebroadcasting is cancelled. For instance, suppose host H heard a broadcast

message from S for the first time. If the distance, say d, between H and S is very small, there is little additional coverage H's rebroadcast can provide. If d is larger, the additional coverage will be larger. In the extreme case, if d = 0, the additional coverage is 0 too. The relationship between the distance d and the additional coverage is πr^2 - INTC(d). So this can be used as a metric by H to determine whether to rebroadcast or not. Now, suppose that before a rebroadcast message is actually sent, host H has heard the same message several times. Let d_{min} be the distance to the nearest host from which the same message is heard. Then H's rebroadcast will provide additional coverage no more than πr^2 - INTC(d_{min}). In the distance-based scheme, d_{min} is used as the metric to evaluate whether to rebroadcast or not. If d_{min} is smaller than some distance threshold D, the rebroadcast transmission of H is cancelled.

Location-based scheme

The location-based scheme, the location of the broadcasting hosts are acquired with the use of GPS (Global Positioning System) with respect to longitude, latitude and altitude. The additional area that can be covered if the host rebroadcasts the message is calculated. The additional coverage is compared with a predefined coverage threshold to determine whether the receiving host should rebroadcast or not. This information is used to estimate the additional coverage more accurately.

Cluster-based scheme

The cluster-based scheme allows only the gateway nodes to propagate the message. In a cluster, the head's rebroadcast can cover all other hosts in that cluster if its transmission experiences no collision. Apparently, to propagate the broadcast message to hosts in other clusters, gateway hosts should take the responsibility. But there is no need for a non-gateway member to rebroadcast the message. Hence, the non-gateway nodes are inhibited from rebroadcasting.

Advantages

- The broadcast storm problem alleviates the effect of rebroadcasting and hence contention and collision.

Disadvantages

- It suffers from the need that all nodes to be awake during broadcasting.
- Among the above methods, location based scheme is used more commonly. But it requires all nodes to be equipped with the GPS devices with the appropriate accuracy.

1.6.3 Trickle

The first step towards sensor network reprogramming is an efficient algorithm for determining when motes should propagate code, which can be used to trigger the actual code transfer. Levis et al (2004) proposed Trickle, an algorithm for code propagation and maintenance in wireless sensor networks. Trickle uses the concept of polite gossip. A node periodically broadcasts its metadata. The metadata describes the code each node has. Trickle sends the metadata to local broadcast address.

There are two options when a node receives the metadata; it either finds that the code is up to date or it finds that it has an older version of the code. When a node hears a metadata that is identical to its metadata, it stays quiet. When a node finds that any other node has old metadata, it generates a code update so that the node which has old metadata can be made up to date. Each mote maintains a counter c, a threshold k, and a timer t in the range [0,τ]. k is a small, fixed integer (e.g., 1 or 2) and τ is a time constant. When a node hears a metadata identical to its own, it increments c. At time t, the mote broadcasts its metadata if c < k. When the interval of sizeτ completes, c is reset to zero and t is

reset to a new random value in the range [0, τ]. If a mote with code φ_x hears a summary forφ_{x-y}, it broadcasts the code necessary to bring φ_{x-y} up toφ_x. If it hears a summary for φ_{x+y}, it broadcasts its own summary, triggering the mote with φ_{x+y} to send updates.

Using the Trickle algorithm, each mote broadcasts a summary of its data at most once per period τ. If a mote hears k motes with the same program before it transmits, it suppresses its own transmission. In perfect network conditions – a lossless, single-hop topology – there will be k transmissions everyτ. If there are n motes and m non-interfering single-hop networks, there will be km transmissions, which is independent of n. Instead of fixing the per-mote send rate, Trickle dynamically regulates its send rate to the network density to meet a communication rate, requiring no a priori assumptions on the topology. In each interval τ, the sum of receptions and sends of each mote is k. The random selection of t uniformly distributes the choice of who broadcasts in a given interval. This evenly spreads the transmission energy load across the network. If a mote with n neighbours needs an update, the expected latency to discover this from the beginning of the interval is $\frac{\tau}{n+1}$. Detection happens either because the mote transmits its summary, which will cause others to send updates, or because another mote transmits a newer summary. A large τ has a lower energy overhead (in terms of packet send rate), but also has a higher discovery latency. Conversely, a small τ sends more messages but discovers updates more quickly. This km transmission count depends on three assumptions: no packet loss, perfect interval synchronization, and a single-hop network. Trickle's maintenance algorithm can be easily adapted to also rapidly propagate code while imposing a minimal overhead. Trickle assumes that motes can succinctly describe their code with metadata, and by comparing two different pieces of metadata can determine which mote needs an update.

Advantages

Trickle has been designed for three important benefits.

- It imposes low maintenance overhead.
- It can propagate codes quickly.
- It can scale to thousand-fold changes in network density.
- In addition, it is capable of handling network repopulations, and is healthy to manage network transience, loss and disconnections.

Disadvantages

- It does not consider the active/dormant schedules of the WSN and assumes nodes that a awake receive the periodic updates.
- Also the transmission scalability suffers under the CSMA protocol as utilization increases.

1.6.4 RI-MAC (Receiver Initiated Medium Access Control)

Sun et al (2008) proposed RI-MAC, an asynchronous duty-cycle protocol, which does not require synchronization among the nodes in the network. The sender of the RI-MAC stays silent until it receives an explicit signal from the receiver announcing when to start data transmission. Every node periodically wakes up based on its active/dormant schedule and checks if there are any incoming data for this node. If the medium is idle, it immediately broadcasts a beacon announcing it is awake and is ready to receive data frame. A node which has data to send transmits the data as soon as it receives the beacon from the intended receiver. The data transmission will be acknowledged by another beacon. The beacon serves as an acknowledgement and also invites new data

transmission to same receiver. If there is no data for the receiver, the node goes to sleep. The working of RI-MAC is shown in Fig 1.4.

In RI-MAC, each node periodically wakes up based on its own schedule to check if there are any incoming DATA frames intended for this node. After turning on its radio, a node immediately broadcasts a beacon if the medium is idle, announcing that it is awake and ready to receive a DATA frame. A node with pending DATA to send, node *S* in this figure, stays active silently while waiting for the beacon from the intended receiver *R*. Upon receiving the beacon from *R*, node *S* starts its DATA transmission immediately, which will be acknowledged by *R* with another beacon. The ACK beacon's role is twofold: first, it acknowledges the correct receipt of the sent DATA frame, and second, it invites a new DATA frame transmission to the same receiver. If there is no incoming DATA after broadcasting a beacon, the node goes to sleep. RI-MAC significantly reduces the amount of time a pair of nodes occupy the medium before they reach a rendezvous time for data exchange. This short medium occupation time enables more contending nodes to exchange DATA frames with their intended receivers, which helps to increases capacity of the network and thus potential throughput. More importantly, this increase is adaptive, by letting a beacon serve both as an acknowledgment to previously received DATA and as a request for the initiation of the next DATA transmission. In RI-MAC, medium access control among senders that want to transmit DATA frames to the same receiver is mainly controlled by the receiver.

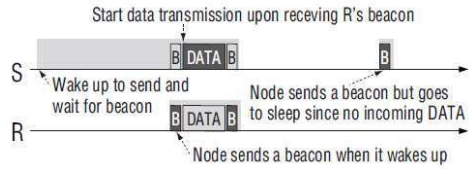

Fig 1.4: Overview of RI-MAC

A beacon frame in RI-MAC always contains a Src field, which is the address of the source transmitting node of the beacon. A beacon can also include two optional fields, depending on the roles the beacon serves: Dst, for destination address, and BW, for backoff window size. A node that receives a beacon can determine which fields are present in the beacon by looking at the size of the beacon. A beacon in RI-MAC can play two simultaneous roles: as an acknowledgment to previously received DATA, and as a request for the initiation of the next DATA transmission. After node *R* wakes up and senses clear medium, *R* transmits a base beacon. If the medium is busy, *R* does a backoff and attempts to transmit the beacon later. After receipt of the first DATA frame from *S*, in the following beacon transmission by *R*, the Dst field is set to *S* to indicate that this beacon also serves as the acknowledgment for the DATA received from *S*. Similar to ACK transmission, transmission of this acknowledgment beacon starts after SIFS delay, regardless of medium status. Nodes other than *S* ignore the Dst field in the beacon and treat it as a request for the initiation of a new data transmission.

Advantages

- RI-MAC support broadcasting either by unicast data transmission or by repeated transmission of data back-to-back for a time equal to sleep

interval. The RI-MAC reduces the time a sender and its intended receiver occupies the medium.

- As the medium access is controlled by the receiver, the RI-MAC is more capable in detecting collisions and recovering lost data frames.
- RI-MAC also reduces overhearing, as a receiver expects incoming data only within a small window after beacon transmission.
- It has lower cost for detecting collisions and recovering lost DATA frames, and higher power efficiency, especially when the network load increases.

Disadvantages

- The end-to-end latency is high in RI-MAC.
- The RI-MAC suffers for power efficiency under light load.

1.6.5 PBBF (Probability-Based Broacasting Forwarding)

Miller et al (2005) described PBBF, which is a MAC layer approach which can be incorporated into any sleep scheduling protocols. PB BF can be integrated into MAC protocols via two parameters: (1) p, which is the probability that a node rebroadcast immediately without ensuring that any of its neighbours are awake, and (2) q, which is the probability that at a given instant, a given node which is expected to be asleep stays awake due to its active/dormant schedule and is the receiver of the immediate broadcast. PBBF can be described using Fig 1.5. Node 1 has to broadcast the message which is to be sent after AW1.Utilizing parameter p, Node 1 broadcasts message immediately without waiting for AW2 to announce it. Node 3 which stays awake due to the parameter q, receives the immediate broadcast. Node rebroadcasts the message via normal broadcast and hence waits for AW2 to announce it, so as to

ensure that each of the neighbouring nodes receive the message. Node 2 receives this message and rebroadcasts it to its neighbours.

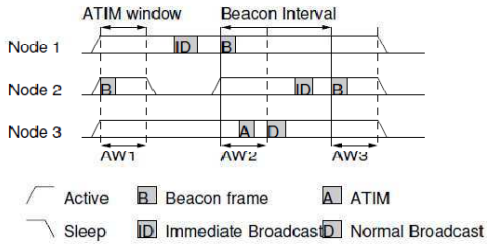

Fig 1.5 Broadcast in PBBF

The pseudo-code of changes to any sleep scheduling protocol required for PBBF is given below.

```
Sleep-Decision-Handler ()
/* Called at the end of active time */
/* If stayOn is true, remain on; otherwise sleep*/
stayOn   false
if DataToSend = true or DataToRecv = true
 then
stayOn   true
 else if Uniform-Rand(0, 1) < q
 then stayOn   true
Receive-Broadcast (pkt)
/* Called when broadcast packet pkt is received */
 if Uniform-Rand(0, 1) < p
```



```
 then Send(pkt)
else Enqueue(nextPktQueue, pkt)
```

The original sleep scheduling protocol is a special case of PBBF with p = 0 and q = 0. The always-on mode (i.e., no active-sleep cycles) can be approximated by setting p = 1and q = 1. PBBF is still slightly different than always-on in this case because it still has the byte overhead (e.g., sending synchronization beacons) and temporal overhead (i.e., PBBF cannot send data packets during the ATIM window) of active-sleep cycles. Through the use of two parameters, p and q, PBBF protocol provides a trade-off between energy, latency, and reliability. While p presents a trade-off between latency and reliability (i.e., the fraction of nodes receiving a broadcast), q presents a trade-off in terms of energy and reliability. As p increases, latency decreases while the fraction of nodes not receiving a broadcast increases (unless q = 1). As q increases, energy consumption increases, but the fraction of nodes receiving a broadcast increases (unless p = 0). Whenever a node decides to rebroadcast a message immediately, all the neighbours that are currently awake receives the message. But if there are no neighbours that are currently awake there will be no receivers for immediate broadcast. The parameter q allows the nodes to stay awake regardless of their active/dormant schedules and hence become the receivers of immediate broadcast.

Advantages

- PBBF ensures that each node receives at least one copy of the broadcast message with high probability.
- It also reduces the latency due to sleeping.
- The PBBF investigates the trade-offs between reliability, latency and energy consumption.

Disadvantages

- It is difficult to set threshold values for parameters 'p' and 'q'.
- It has the disadvantage that duty-cycles are subject to changes in network traffic.

1.6.6 ADB (Asynchronous Duty-Cycle Broadcasting)

Sun et al (2009) proposed ADB which utilises asynchronous duty-cycle and optimizes the level of transmission to each neighbour individually. As the neighbours wake up at different schedules, ADB makes use of unicast to propagate broadcast message to neighbours so that it can learn which of the neighbours have received the broadcast. ARQ (Automatic Repeat Request) is used as the part of unicast to improve reliability. Each receiver is updated with the information on the progress of the broadcast using footer in the data frame. This avoids redundant transmissions. ADB also allows delegating the transmission from some neighbour to another neighbour which has better link quality. This avoids transmission over poor links and hence allows the nodes to sleep as early as possible. This, in turn reduces the energy consumption and delivery latency. The working of ADB is explained in Fig 1.6.

In this example, the network consists of three nodes, nodes *S*, *R1*, and *R2*, all within transmission ranges of each other. Node *S* wants to broadcast a DATA packet to all nodes. When *R1* wakes up, node *S* transmits the packet upon receiving *R1*'s beacon in the same way as for unicast in RI-MAC. However, ADB includes a new "footer" in DATA frames and acknowledgment beacons (ACKs), indicating the progress of the broadcast, including some transmissions that are about to happen. A receiving node uses this information to avoid

unnecessary transmissions and to decide whether it should forward the packet to a neighbour that has not received it. In this example, the ADB footer in the DATA frame from *S* informs *R1* that *R2* has not been reached yet by the broadcast and that the quality of the link (*S*, *R*2) is poor.

Suppose the quality of link (*R*1,*R2*) is good (e.g., because of the short distance). Node *R1* decides to deliver the packet to *R2* and indicates the good quality of (*R*1,*R2*) in the footer of the ACK to *R1*. Upon receiving this ACK, *S* learns that it is better for *R1* to transmit the packet to *R2*, so *S* delegates handling of *R2* to *R1*. As *S* has no other neighbour to be reached, *S* then goes to sleep immediately. When *R2* wakes up, *R1* unicasts the DATA frame to *R2* in the same way, except that the ADB footer in the DATA frame indicates that *S* has received the DATA frame, allowing *R2* to sleep immediately because all neighbours of *R2* have been reached. When a node wakes up and receives a broadcast DATA packet, the node must decide whether or not to transmit it to each of its neighbours. To facilitate this decision, ADB propagates information on the progress of the broadcast and on link qualities by embedding this information into DATA packets and acknowledgment beacons. In order to efficiently embed this information, each node v includes the status of each of its neighbours in the footer of DATA and ACK frames. Node v assigns one of the following values as the status of each neighbour w: REACHED, if w has received the packet; DELEGATED, if some other node is going to deliver the packet to w; or P(v,w), an integer representing Q(v,w), otherwise. The P(v,w) is referred as the priority of this link. If node w's status is REACHED or DELEGATED, v does not attempt to transmit the packet to w. Otherwise, v attempts to transmit the packet to w, and the quality of link (v,w) is indicated by priority P(v,w). ADB includes the status of all direct neighbours in the footer of a frame to a node, rather than the status of a subset of neighbours that the receiver node might be interested in.

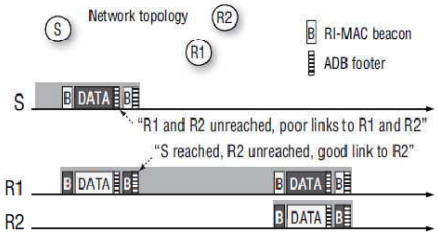

Fig 1.6. Overview of ADB

ADB distributes the status of neighbours using a bitmap that is constructed based on an append-only neighbour list: once a node v detects a new neighbour, it appends the neighbour to the end of its neighbour list N(v). A node v lists the status of neighbours using a bitmap with segments of equal length, with each segment corresponding to a node in N(v), the set of neighbours of node v. In order to refer to a node by its position in N(v), N(v) is organized as an array, with the segments arranged in the same order as the corresponding node in N(v). In order for a recipient node to be able to decode this bitmap, node v distributes the neighbour list to direct neighbours. Let Nw(v) denote w's local view of v's neighbour list. Due to packet losses caused by collisions or dynamics of wireless channels, Nw(v) could be stale and different from N(v). ADB ensures that Nw(v) is a prefix of N(v). With this property, even if a node w does not have a current copy of node v's neighbour list, w can still decode the beginning portion of a received bitmap without ambiguity. In a more dynamic network such as with mobility, a version number is assigned to each neighbour list to avoid ambiguity, but the neighbour list is used to efficiently handle the common case where sensor nodes are essentially stationary. Also, a node v will not remove any existing neighbour, say w, from its neighbour list N(v) even if node w has moved away or has failed. Instead, a node v will use the value zero for P(v,w) in its bitmap to tell its neighbours it does not currently have a valid

link to node w. When a node receives a broadcast data packet, it decides whether to transmit it to its neighbours or not, utilising the information on the progress of the broadcast and link qualities.

Advantages

- ADB coordinates transmissions to a node from its neighbours by efficiently distributing information on the progress of a broadcast together with DATA transmissions.
- Such information also indicates quality of the wireless links from the neighbours to the node, helping ADB avoid transmission attempts over poor links.
- ADB reduces redundant transmissions, collisions and energy consumption.

Disadvantages

- ADB has higher message cost and transmission energy.
- ADB lacks efficiency in large scale networks and while delivering large chunks of data.

1.6.7 Opportunistic Flooding

Guo et al (2009) described Opportunistic Flooding, which is a flooding method for low duty-cycle Wireless Sensor Networks. The main aim of Opportunistic Flooding is to allow packets travels in multiple paths. The Opportunistic Flooding makes the forwarding decision, so that the packet is forwarded with higher probability, if the packet reaches opportunistically earlier. When a node receives a packet it forwards it to its next-hop node if and only if the packet arrives opportunistically earlier than the packet delivered via an energy optimal tree. The Opportunistic Flooding calculates probability mass

function (pmf) of the delay of each node via energy optimal tree to aid the decision making process. A packet is forwarded opportunistically outside the energy optimal tree only if this forwarding can reduce the delay significantly. For a low-duty-cycle network in which two neighbours seldom wake up at the same time, a broadcasting packet cannot be received by many nodes simultaneously. In addition, the delivery ratio of traditional flooding methods becomes even worse when unreliable links and collisions are taken into account. Based on the network model, a flooding packet can only be forwarded from nodes with smaller hop counts to those with larger ones. As a result, the flooding structure of the network is a directed acyclic graph (DAG) of N vertices. The weights of the directed edges are the corresponding link quality. Based on the DAG, a tree structure can be constructed by assigning each node an incoming link with the best link quality among all incoming links. It can be easily proved that this tree structure is the energy-optimal one for flooding among all tree structures generated from the DAG. In other words, if a flooding packet is forwarded based on this tree, (i.e., a node only receives a flooding packet from its parent), the expected total number of transmissions is minimized. However, the flooding via the energy-optimal tree may have a long flooding delay, since a node's parent may not receive the flooding packet as early as its other neighbours, due the opportunistic nature of wireless communication. Based on this observation, the key idea of opportunistic flooding is to utilize opportunistic links outside an energy-optimal tree if the transmissions via these links have a high chance of making the receiving node receive the packet "statistically earlier" than its parent. Clearly, the flooding structure of the design is dynamically changing, where a node decides to forward its received flooding packet to a next-hop node if and only if this transmission is expected to deliver a new packet to that node, instead of an old/redundant one. In other words, the packet to be forwarded opportunistically shall be statistically earlier than the packet that is otherwise delivered via the energy-optimal tree. In order to

forward opportunistically early packets while avoiding late ones, opportunistic flooding consists of three major steps:

1. **The *pmf* Computation**

   Due to unreliable links, the delay of a flooding packet arriving at each node from its parent through the energy-optimal tree is a random variable. The probability mass function (pmf) of this delay is first derived for each node to guide the decision making process. From the pmf, each node computes its p-quantile delay Dp as the statistically significant threshold and shares this with all its pervious-hop nodes.

2. **Decision Making Process**

   A packet is forwarded opportunistically via the links outside of the energy-optimal tree only if this forwarding can significantly reduce the delay . Specifically, a node makes its forwarding decision locally based on three inputs: (i) the receiving time of the flooding packet, (ii) the link quality between itself and the next-hop node, and (iii) the p-quantile. The structure of decision making is dynamically changed for different flooding packets.

3. **Decision Conflict Resolution**

   Since each node makes its forwarding decision in a purely distributed manner, it would be the case that multiple nodes decide to forward the same packet to a common neighbour, which is called *decision conflict*. Two conflict resolution techniques are designed to avoid collisions and save energy further. With dynamic decisions per packet, the design permits a packet to travel along an opportunistically-fast route instead of a

fixed one via the energy-optimal tree. At the same time, late packets are not forwarded to reduce redundancy and save energy.

Advantages

- The opportunistic flooding reduces the flooding delay and redundancy in transmission.
- To improve performance further, a forwarder selection method is used to alleviate the hidden terminal problem and a link-quality based backoff method to resolve simultaneous forwarding operations.

Disadvantages

- The forwarding decision is made based on the receiving time of the packet, link quality and computed delay.
- The working schedules of the nodes are fixed and decided by the network.

1.6.8 DSF (Dynamic Switch-Based Forwarding)

Gu et al (2007) proposed DSF, which is designed for networks with unreliable links and programmed node communication schedules. The delivery latency is not only due to the fixed schedules but also due to the communication links. DSF uses multiple potential forwarding nodes at each hop. Each node retains a sequence of wake-up schedules of forwarding nodes. When the link quality is perfect, the end-to-end delay is the sum of two types of delays: (1) the total transmission delay, which is the product of number of hops and t, and (2) The sleep latency, which is the time spent on waiting for the receivers to wake up at each hop. However, the unreliable radio links between low-power sensor

devices suggests that the packet transmission between a sender and a receiver would not always be 100% successful. As a result, the waiting time at each hop is highly impacted not only by the node working schedule but also by the link quality.

When a node has a packet to be sent it looks up in the wake-up schedule and wakes up at the earliest time schedule when any of the forwarding nodes wakes up. The node attempts to send the packet to the forwarding node, which is awake. If the node receives the packet, forwarding is done. Else the node has to wake up in the next earlier schedule of any forwarding nodes. This continues until the transmission is successful. Fig 1.7 demonstrates the packet transmission process between one sender and n nodes in its forwarding sequence. In Fig 1.7, node A has a packet with forwarding sequence $S_n^A = (B1, B2, \cdots, Bn)$. First, node A wakes up at time $t1$ and tries to transmit the packet to the node $B1$. If the data delivery is successful, node A ends the current packet forwarding session. However, if the transmission fails, the node A wakes up again at time t_2 and tries to send the packet to the node B_2. This retransmission process continues with node A repeatedly trying to send the packet to the node in the sequence S_n^A. If the transmission fails at the last node B_n, node A drops the packet. For a given sink, each node maintains a sequence of forwarding nodes sorted in the order of the wake-up time associated with them.

To start sending a packet, a node looks up the time associated with the first node in the sequence, wakes up at that time interval, and tries to send the packet. If the transmission is successful, forwarding is done. Otherwise, the node fetches the next wake-up time from the sequence and tries to send the packet again. This retransmission process over a single hop continues until the sending node confirms that the packet has been successfully received by one of forwarding nodes or the sending node reaches the end of the sequence and drops the packet. DSF reduces the time spent on transmitting a packet. It also

optimizes the end-to-end delay, source-to-sink delivery ratio and energy consumption. The maximum time bound for a sender to retransmit a particular packet is set as *T*. Consequently, at node *e*, with known neighbouring nodes and their corresponding working schedule G, a full sequence of potential forwarding nodes that wake up before *T is available*. Because the length of the potential forwarding sequence of a node is finitely subject to the maximum retransmission time interval *T*, under the reality of unreliable link quality among pairs of wireless sensor devices, packets sent by a source node may not all arrive the destination sink node. Therefore, when reliable transmission has the highest priority for a sensor network application, the optimization of the expected data delivery ratio is critical.

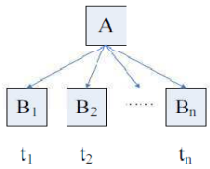

Fig 1.7. Example of Dynamic Scheduling

Advantages

- The major advantage of dynamic switching is the use of a forwarding sequence to reduce the time spent on transmitting a packet successfully at each hop rather than waiting for a particular forwarding node to wake up again after failure.

Disadvantages

- DSF is inadequate for low duty-cycle flooding which is an important function for dissemination
- In low duty-cycle sensor networks, a sender should not endlessly retransmit a packet because it would consume significant energy at the sending nodes.

1.6.9 TRAMA (TRaffic Adaptive Medium Access)

Rajendran et al (2003) proposed TRAMA, which uses a distributed election method based on the traffic information to determine which nodes can send during a particular time slot. TRAMA assigns clots to the nodes which has traffic to send and avoids slots for nodes with no traffic. This makes nodes go to sleep when there is no traffic. TRAMA consists of three components; (1) Neighbour Protocol (NP) (2) Schedule Exchange Protocol (SEP) and (3) Adaptive Election Algorithm (AEA) [9].There are two types of slots namely signalling slots and transmission slots. NP is used to propagate one-hop neighbour information to all nodes during signalling slots. As all nodes tries to propagate their information, the signalling slot is subject to collisions. SEP is used for exchanging schedules or traffic information during transmission slots. The schedules contain traffic information which informs about the set of receivers of a particular node. Every node has to announce its schedule using SEP before starting actual transmission. AEA is used select the transmitter and receiver for a particular slot. The selected transmitter can transmit data without any collision during the transmission slot to the selected receiver.

TRAMA selects receivers based on schedules announced by transmitters. Nodes using TRAMA exchange their two-hop neighbourhood information and the transmission schedules specifying which nodes are the intended receivers of their traffic in chronological order, and then select the nodes that should transmit and receive during each time slot. Accordingly, TRAMA consists of three components: the Neighbour Protocol (NP) and the Schedule Exchange Protocol (SEP), which allow nodes to exchange two-hop neighbour information and their schedules; and the Adaptive Election Algorithm (AEA), which uses neighbourhood and schedule information to select the transmitters and receivers for the current time slot, leaving all other nodes in liberty to switch to low-power mode. TRAMA assumes a single, time-slotted channel for both data and signalling transmissions. Fig 1.8 shows the overall time-slot organization of the protocol. Time is organized as sections of random- and scheduled-access periods. The random-access slots are referred as signalling slots and scheduled-access slots as transmission slots. Because the data rates of a sensor network are relatively low, the duration of time slots is much larger than typical clock drifts NP propagates one-hop neighbour information among neighbouring nodes during the random access period using the signalling slots, to obtain consistent two-hop topology information across all nodes. As the name suggests, during the random access period, nodes perform contention-based channel acquisition and thus signalling packets are prone to collisions.

Transmission slots are used for collision-free data exchange and also for schedule propagation. Nodes use SEP to exchange traffic-based information, or schedules, with neighbours. Essentially, schedules contain current information on traffic coming from a node, i.e., the set of receivers for the traffic originating at the node. A node has to announce its schedule using SEP before starting actual transmissions.

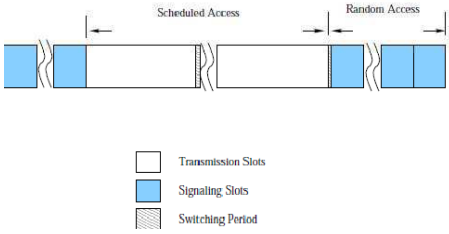

Fig 1.8: Time slot Organisation

SEP maintains consistent schedule information across neighbours and updates the schedules periodically. AEA selects transmitters and receivers to achieve collision-free transmission using the information obtained from NP and SEP. This is the case, because electing both the transmitter and the receiver(s) for a particular time slot is a necessity to achieve energy efficiency in a collision-free transmission schedule. AEA uses traffic information (i.e., which sender has traffic for which receivers) to improve channel utilization. The length of a transmission slot is fixed based on the channel bandwidth and data size. Signalling packets are usually smaller than data packets and thus transmission slots are typically set as a multiple of signalling slots to allow for easy synchronization.

TRAMA starts in random access mode where each node transmits by selecting a slot randomly. Nodes can only join the network during random access periods. The duty cycle of random- versus scheduled access depends on the type of network. In the case of sensor networks, there is very little or no mobility, depending on the type of application. Hence, the main function of random access periods is to permit node additions and deletions. Time synchronization could be done during this period.

During random access periods, all nodes must be in either transmit or receive state, so they can send out their neighbourhood updates and receive updates from neighbours. Hence, the duration of the random access period plays a significant role in energy consumption. During random access periods, signalling packets may be lost due to collisions, which can lead to inconsistent neighbourhood information across nodes. To guarantee consistent neighbourhood information with some degree of confidence, the length of the random access period and the number of retransmissions of signalling packets are set accordingly. NP gathers neighbourhood information by exchanging small signalling packets during the random access period. Signalling packets carry incremental neighbourhood updates and if there are no updates, signalling packets are sent as keep-alive" beacons. Each node sends incremental updates about its one-hop neighbourhood as a set of added and deleted neighbours. These signalling packets are also used to maintain connectivity between the neighbours. A node times out a neighbour if it does not hear from that neighbour for a certain period of time. The updates are retransmitted such that 0:99 probability of success is ensured. Because a node knows the one-hop neighbours of its one-hop neighbours, eventually consistent two-hop neighbourhood information makes its way across the network.

SEP establishes and maintains traffic-based schedule information required by the transmitter and receiver selection. A node's schedule captures a window of traffic to be transmitted by the node. This information is periodically broadcast to the node's one-hop neighbours during scheduled access. Nodes announce their schedule via schedule packets. Because nodes have two-hop topology information obtained through NP, there is no need to send receiver addresses in the schedule packet. Instead, nodes convey intended receiver information using a bitmap whose length is equal to the number of one-hop neighbours. Each bit in the bitmap corresponds to one particular receiver ordered by their identities.

The total number of receivers supported by this scheme depends on the size of the data slot and the number of slots for which receivers are announced. To broadcast a packet, all bitmap bits are set to 1, indicating that all one-hop neighbours are intended receivers of the packet. If the packet needs to be multicast to just 14 and 4, then only these bits are set in the bitmap. A node forms the bitmap for the winning slots based on the current traffic information for its queue. If the node's queue size is smaller than the number of bitmaps contained in the schedule, some of the winning slots will go unused. For these vacant slots, the node announces a zero bitmap. Slots with zero bitmaps could potentially be used by some other node in the two-hop neighbourhood.

The slot after which all the winning slots go unused is called ChangeOver slot. All unused slots happen contiguously toward the end before the last winning slot, which is reserved for announcing the next schedule. This maximizes the length of sleep periods. Nodes maintain schedule information for all their one-hop neighbours. The schedule information is consulted whenever a node has the highest two-hop priority to decide if the node will actually transmit (i.e., it has data to send and thus will use the slot) or will give up the slot to another node in the neighbourhood. Based on this decision, the schedule information for the node is updated either using the short summary from the data packet (if the node is receiving), or assuming transmissions (if the node is sleeping since it is not the intended receiver of transmitter).

Advantages

- Avoids the assignment of time slots to nodes with no traffic to send
- Allows nodes to determine when they can become idle

- Self adaptive to changes in traffic, node state, or connectivity
- Prolongs the battery life of each node.

Disadvantages

- Signaling slots consume significant energy.
- Latency gets higher as the load gets higher in the network.
- Transmission slots are set to be several times longer than the random-access period

1.6.10 Reliable Broadcast

Feng and Jiangchuan (2012) described the broadcast problem in low duty-cycle WSNs. The solution provided, together with their application/deployment-specific duty cycles, renders the all-node-active assumption impractical. This in turn introduces a series of new challenges toward implementing network wide broadcast. From a local viewpoint, since the neighbours of a node are not active simultaneously, a node would have to forward a message multiple times at different instances; from a global viewpoint, since the topology is time varying with no persistent connectivity, if not well planned, the latency for a message to reach all nodes can be significantly prolonged. Hence the authors solve this problem in two respective namely centralised dynamic solution and distributed solution.

In the centralised solution, there are two kinds of edges in the graph, referred to as time edges and forwarding edges, respectively. A time edge connects two neighbouring vertices along a row, from the earlier to the later. It corresponds to the case that no node among the receivers R will forward the message at a time t, and the same coverage state is, thus, inherited by the next

time slot. A forwarding edge corresponds to forwarding events. Specifically, a forwarding edge from $v_{R,t}$to $v_{R',t'}$ means that, at time t, one or more active nodes in R will forward the message, which leads to a new coverage status R'. The R ⊆R0, and R' − R is the set of nodes that newly receive the message in this round of forwarding. The time-coverage graph can be naturally related to the duty-cycle-aware broadcast problem: each forwarding sequence corresponds to a path from $v_{\{s\},t}$ to a vertex in the last row, and vice versa. The objective function $f(S, t_m - t_0) = \alpha|S| + \beta|t_m - t_0|$ assigns weight β to each time edge since a delay of one time unit is incurred, and weight $\alpha p + \beta(t' - t)$ to a forwarding edge from$v_{R,t}$to $v_{R',t'}$, where p is the number of nodes in R that forward the message at time t.

The distributed solution focuses on optimal forwarding sequence covering nodes within two hops, the 1-hop neighbours and 2-hop neighbours. Three reasons to choose two hops are: (1) it minimizes the computation overhead, and yet keeps reasonable accuracy; (2) since every node must maintain information about its direct neighbours, the topology and active/dormant information for 1- and 2-hop neighbours can be obtained through a simple beacon protocol, without any extra broad-scope protocols for information dissemination; (3) such information is sufficient to avoid most of message forwarding contentions.

For any node w, a Covering set, or CovSet is defined as the set of 1- and 2-hop neighbours that are known (by w) being covered by at least one forwarding. A CovSet is created when a new broadcast message is received, and is updated when node w forwards a broadcast message or a broadcast message is received or overheard. Specifically, when node w forward a broadcast message, based on the active/dormant patterns of its neighbours, it will find out those neighbours that are currently active and thus covered by this message, and then

add them to its CovSet. And similarly, when a broadcast message is received or overheard, node w will also find out the currently active neighbours of the message's sender and add them to its CovSet. An issue in distributed implementation is that the sequence calculation at different nodes is not necessarily synchronized, and is not consistent. This in turn makes the forwarding sequence calculated by node w differ from another node sequences. To solve the inconsistency, when the CovSet is changed (updated), node w will check if this change follows its current forwarding sequence. If the CovSet is changed due to an overheard message, node w then checks if this message is forwarded by the sender as indicated in its current forwarding sequence. If not, node w will recompute the forwarding sequence by incorporating the updated CovSet. Since the CovSet expands over time, the first row will become closer to the last row in each recomputation, implying that the computation cost reduces over time.

Advantages

- The distributed solurion and centralized dynamic solution is applicable to diverse schedules.
- Provides a generic tool for cross-layer optimization
- Can be easily extended to broadcast a series of messages or broadcast messages from multiple sources.
- A balance between efficiency and latency with coverage guarantees.

Disadvantages

- Higher computation cost due to maintenance of 2-hop information.
- Difficulty in obtaining the global connectivity and active/dormant patterns.

- QoS is not considered.

CHAPTER 2

IMPLEMENTATION OF BROADCASTING

2.1 PROBLEM DEFINITION

Broadcasting is one of the essential services in Wireless Sensor Networks (WSN). Broadcasting is used to propagate messages from a node or source to all other nodes in the network. The broadcasting involves propagation of data and control packets. Any node which wishes to query the network about an event has a query message that is to be broadcasted to all other nodes. The control messages have to be broadcasted from sink to other nodes during network configuration. Hence a reliable broadcast service is very important in the effective functioning of WSNs. Two basic approaches of broadcasting are flooding and gossiping. Their basic forms are inefficient as they assume all nodes are active. If all nodes are active during the broadcast process every node can receive or forward the message. This process of assuming the nodes to be active is referred as all-node-assumption. The all-node-active assumption fails to detain the distinguishing character of energy constraint WSNs. The energy constraint sensor nodes swap between dormant and active states. During the active state, the nodes execute sensing tasks and communications and thereby dispose of considerably excessive energy. But during the dormant state the nodes remain idle consuming less energy. In this context, the term duty cycle is defined as ratio between active period and full active and dormant period.

A low duty cycle WSN minimize the time a node spends in overhearing an unnecessary activity by placing the node in the dormant state. Hence, a low duty cycle WSN, the nodes have longer existence in the place where they are deployed for operation. In a low duty cycle WSN, where the number of nodes is small the broadcast can be enabled by waking up all the nodes through global synchronization. But it is not possible in large networks as it is difficult to provide prior knowledge about local timing information and schedules throughout the entire network. Also, the duty cycles are optimized based on the application or deployment and hence the broadcast service accepting the schedules must be a cross-layer optimization of the system. As the nodes in a network wake up during different time intervals, a node will have to send the message to its neighbouring nodes several times at different chances. This, in turn, prolongs the time necessary for a message to reach all the nodes. The performance degradation also occurs during broadcast in low duty cycle WSN as it fails to cover the entire network within the acceptable time. This problem can be overcome by providing two solutions for enhancing the quality of broadcast service in low duty cycle WSNs namely centralized dynamic and distributed solution. The centralized dynamic solution is acquired from the tree formed during the broadcast process. This is applicable to diverse duty-cycle aware strategies. The distributed solution relies only on local information and operations for reliable and scalable broadcast service.

2.2 QUALITY OF BROADCAST SERVICE

In a low duty cycle network, a node can forward the message to its neighbour only if the neighbour is awake. In addition a node that has already received the message can only forward it. Also the broadcast message should be

reached to all nodes in the network. Consider a low duty cycle network as shown in the Fig 2.1. The sink forwards the message to the nodes 1 to 3 only if the three nodes are awake. Else the sink has to send the message to the three nodes at different instances depending on the wake-up schedules of the nodes. If there is no overlapping of the active periods of the nodes (1 to 3), the sink will have to send the message three times at dissimilar instances. In case of multiple hops for example the message to reach node 5, if the node 2 is not awake for long time the message will take longer route through node 1.

The quality of the broadcast mainly depends on message cost and time cost. The message cost which is defined as the number of times the message is sent can be minimized if there are overlapping active periods of the nodes in the tree through which the message is propagated. The time cost which is defined as the time taken for the message to cover the entire network can also be minimized by forwarding through the active nodes irrespective of the shortest path.

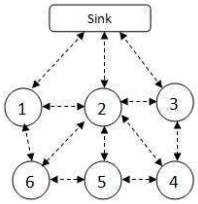

Fig.2.1. An example for duty-cycle-aware broadcast

If the propagation of message is denoted as $(n_i t_i)$, where the node n_i propagate the message at time t_i, then the propagation schedule can be denoted as described in equation 2.1.

$$S = (n_1, t_1), (n_2, t_2), \dots, (n_m, t_m) \quad (2.1)$$

The message cost is the calculated as $|S|$ and the time cost can be calculated from $(t_0 - t_m)$, where t_0 is the starting time of propagation from node s. The combination of message cost and time cost, $f = |S| + (t_0 - t_m)$, is the focus of this paper. This can be extended to a wide range of applications by assigning different weights (w_a, w_b). For the applications that need a message to be broadcasted immediately can use small w_a with a large w_b . For the applications that use large message which does not require immediate propagation can use large w_a with a small w_b which helps in saving the message cost and the energy. The propagation schedules actually depend on the ratio of w_a/w_b and also influence the message cost and the time cost.

2.3 CENTRALIZED DYNAMIC SOLUTION

The centralized solution is constructed on the basis of time and coverage. Consider a vertex $v_{R,t}$, where R represents the sensor nodes that have received the broadcast message at time t, i.e., the nodes in R have been covered. The set of nodes in R starts from {sink} and increase until {n}, where n is the total number of nodes. Each set in R denotes a connected sub-tree of the network from sink. The sink can be either the sink or any of the nodes in the network that acts as the source for the message. Only a few set of Rs among the 2^n sub-trees are active due to the duty cycles of the nodes. The vertex $v_{R,t}$ consists of two kinds of edges namely time edges and the propagation edges. The time edge is concerned with the case that no node in the set R is active and hence the

propagation for coverage is carried out in next time slot. The propagation edge from $v_{R,t}$ to $v_{R',t'}$ correspond to the case that one or more active nodes have propagated the message and the resulting new coverage at time t' is denoted by R'. This time-space coverage vertex corresponds to the propagation sequence discussed above. In the function $f = |S| + (t_0 - t_m)$,, for the time edge a weight w_b and a weight $w_a p + (t_0 - t_m)$ is assigned to each propagation edge from $v_{R,t}$ to $v_{R',t'}$, where p is the total number of nodes in the set R that propagate the message at time t.

2.4 DISTRIBUTED SOLUTION

Using the centralized dynamic solution, the lower bounds of message cost and time cost can be calculated. It can also be used for assessing different broadcast strategies. Practically, it very well suited for small networks with centralized entity and also for large broadcast messages with are low frequent. For large networks the centralized dynamic solution results in higher computational cost and also the complexity in obtaining the global connectivity and the active/dormant schedules. To solve these issues the distributed solution is addressed in this section.

The distributed solution focuses on the one-hop and two-hop neighbours. This reduces the computational overhead but still maintains reasonable accuracy. The global information about two-hop neighbours can be obtained by sending a simple beacon signal and this also reduces the message forwarding contentions. For a node w, a Covering Set is defined, which is set of nodes that can be covered by w in one or more propagations. When a new broadcast message is received, the Covering Set is created and it is updated when the node w broadcast the message. For the node w to forward a message, the node will find out which of the neighbours are active based on the active and dormant

schedules and these neighbours will be added to the Covering Set. Also when any broadcast message is received or overheard, the currently active neighbours of the message's sender is also added to the Covering Set. The Covering Set of a node gives the node's perception about its neighbours on the broadcast coverage. The centralized dynamic algorithm is modified accordingly to calculate the propagation schedule based on the Covering Set. Whenever the Covering Set is updated, the node w checks if it follows the propagation schedule. Since the Covering Set gets updated and expanded, the computational cost is lowered over time.

The Receiving Set for each node w is introduced to enhance strict coverage. The Receiving Set is defined as the set of 1-hop and 2-hop neighbours of node w that have already received the message. When a new broadcast message is received by w, the Receiving Set is created and appends the sender of this message to it. Later when the same message is received or overheard from some other neighbours, the node w appends sender into the Set, if it is not already in it. If all 1-hop neighbours are included in the Receiving Set, which ensures that all 1-hop neighbours have received this message, the node w can stop its propagation. In addition, each node piggy backs its Receiving Set along with the message. The receiving nodes updates their Receiving Set based on the piggy backed Receiving Set. A timeout is used to prevent the over-expanding of the Covering Set. The Covering Set is periodically reset to Receiving Set. The distributed solution is summarized in Fig 2.2. When a node w wakes-up, it checks if there is any message arrived for it. If so it checks the message type. If it is a new broadcast message, the node w creates the Covering Set and Receiving Set and appends the sender of the message and the nodes in Receiving Set piggy backed with this message. Also the node w adds the neighbours that are presently active and are covered by the set into the Covering Set. An ACK is scheduled, if the received message targets particularly on the node w. If the

received message is an ACK, the node w adds the sender of the message into Covering Set and Receiving Set. Now the node w will check its Receiving Set to know if all of its neighbours have received the message. If all neighbours are included in the Set then the node w require no further forwarding and hence can safely stop releasing the memory used for Covering Set and Receiving Set. Else the node w checks if its Covering Set follows the current propagation sequence. If not, node w re-compute the propagation schedule further and the message will be send until the timeout occurs.

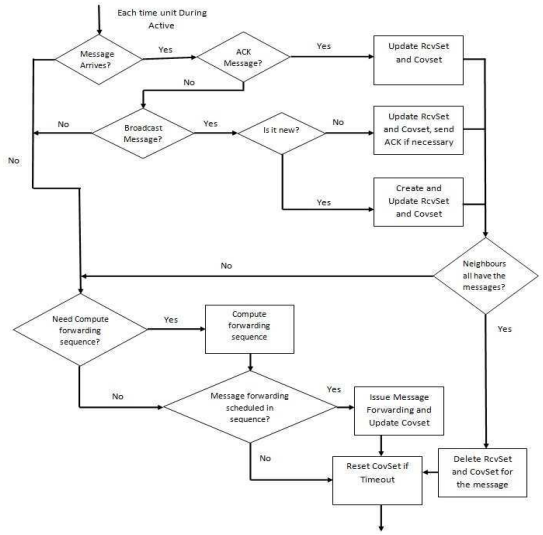

Fig.2.2. Operations of a Distributed Solution

CHAPTER 3

RESULTS

3.1 SYSTEM SPECIFICATION

3.1.1 Hardware Requirements

Processor	: Pentium Dual Core and above
Clock speed	: 1 GHz
Hard Disk	: 160 GB
RAM	: 1 GB or above
Cache Memory	: 512KB
Monitor	: Color Monitor
Keyboard	: 104Keys
Mouse	: 3Buttons

3.1.2 Software Requirements

Operating System	: Fedora 13
Language	: Network Simulator 2.32

3.1.3 Software Specification

The proposed work is implemented using Network Simulator NS2. For the evaluation purpose, 100 sensor nodes are deployed randomly. The sensing values for each of the sensor node at each time of sampling are varied randomly in the range from 0 to 10 joules. The nodes are given a random dormant and active cycles. A node with a broadcast packet sends the message to other nodes considering their active state and the remaining energy of the receiving node.

The nodes use the beacon signal to announce their neighbour tables. The performance of the solution proposed is examined through various simulations. The metrics used for evaluation are message cost, delay, interval, overhead involved, throughput and average energy. The various factors that affect the performance of the solution have been scrutinized. The sensing field is set to a square of 200m and the range of wireless communication is set to 10m. The number of nodes is varied between 800 and 2000. A number of topologies have been generated for each of the settings. During the set-up phase, the active and dormant schedules of the nodes are developed and exchanged between neighbours.

3.2 SNAP SHOTS

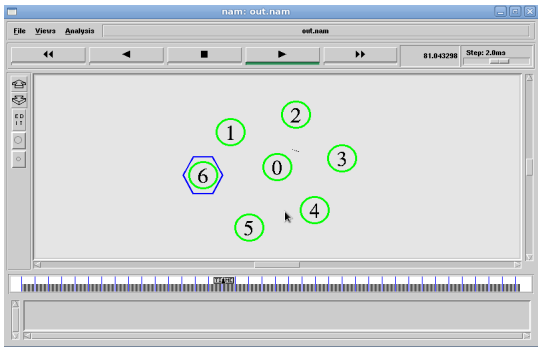

Fig 3.1 Broadcasting in centralised solution

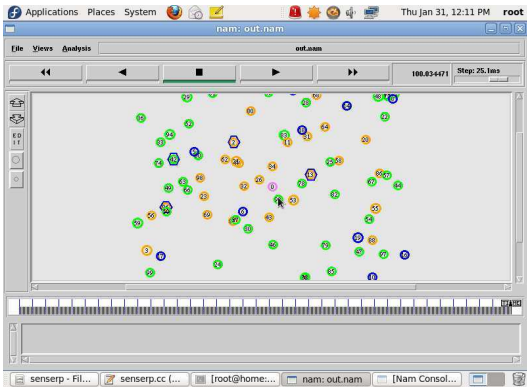

Fig 3.2 Low Energy Nodes

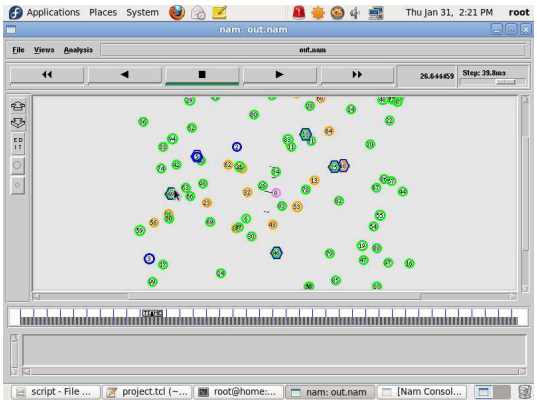

Fig 3.3 Broadcasting in Distributed Solution

3.3 ANALYSIS

3.3.1 Throughput

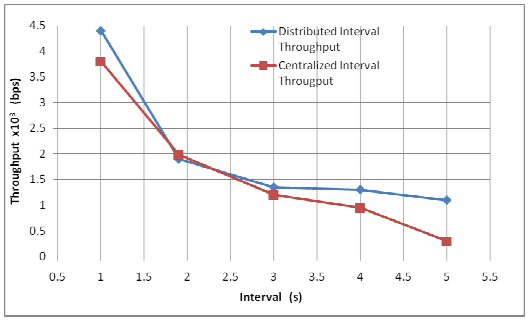

Fig 3.4 Throughput Curve

The throughput of the centralized and the distributed solutions are considered with the packet interval and the results are shown in the Fig 3.4. The result shows that a better throughput is achieved when the packet interval is less. It concludes that the centralized and distributed solution can be applied even when there is need for frequent transmission of packets.

3.3.2 Energy Efficiency

In sensor networks, consumed energy is the primary performance measure. Since the transmission of the packets is the dominant factor in consuming energy, the interval used as efficiency metric. Both centralised dynamic and distributed solution has higher energy consumption when the

interval of packet transmission is less. But when the energy consumption slows down as the interval of packet transmission is more. The results for energy efficiency are shown in Fig 3.5. The analysis shows a drip in the energy consumption when the packet transmission is less.

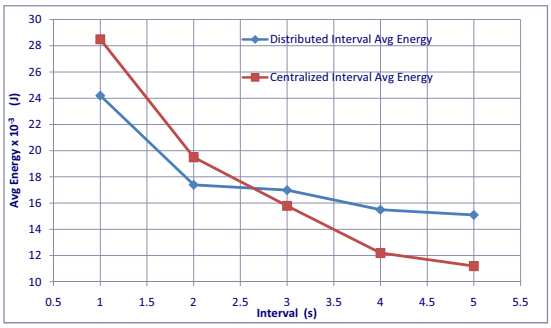

Fig 3.5 Energy Efficiency Curve

3.3.3 Message Cost

The message cost is the number of transmissions required to cover the entire network. The message cost is considered with the interval of transmissions. The results are tabulated in Fig 3.6. In centralised dynamic solution the message cost is same irrespective of the frequency of the messages send. This is due the smaller size of the network used in centralised dynamic solution where the entire network can be covered with limited number of messages. But the distributed solution higher message cost and message cost

decreases as the interval increases. This is due the easy propagation of messages in a less traffic environment.

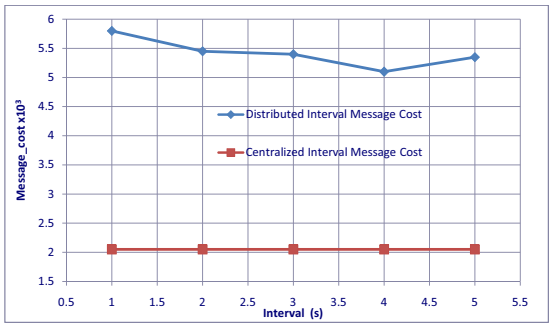

Fig 3.6 Message Cost Curve

3.3.4 Time Cost

The time cost is the time taken for the message to cover the entire network. The time cost is considered with the interval of transmissions. The results are tabulated in Fig 3.7. In centralised dynamic solution the time cost is same irrespective of the frequency of the messages send. This is due the smaller size of the network used in centralised dynamic solution where the entire network can be covered with limited number of messages. But the distributed solution higher time cost and time cost decreases as the interval increases. This is due the easy propagation of messages in a less traffic environment.

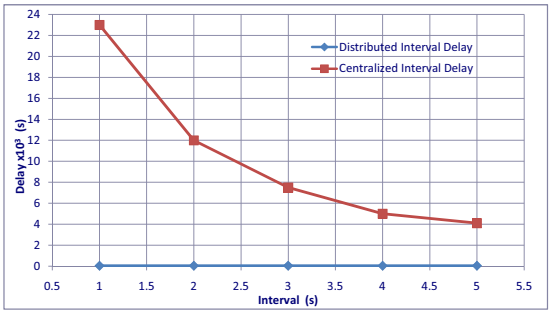

Fig 3.7 Time Cost Curve

3.3.5 Duty Cycle

The duty cycle is an important constraint in the low duty cycle WSN. The duty cycle consists of active and dormant periods. A low duty cycle WSN has more dormant periods. The dormant time is considered with the overheads required for transmission and the results are tabulated in Fig 3.8. In centralised dynamic solution the overhead required is same in spite of the dormant time. In distributed solution the overhead required increases as the dormant time increases.

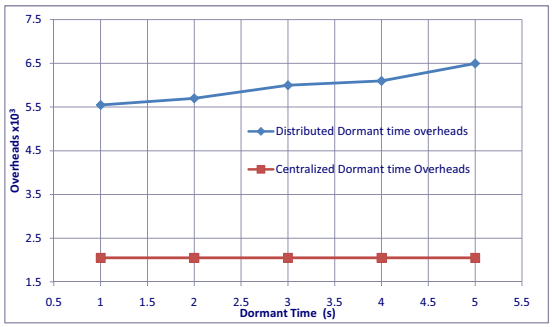

Fig 3.8 Duty-Cycle Curve

3.4 CONCLUSION AND FUTURE WORK

In this project, the quality of broadcast service has been analysed. The solutions proposed can be used for broadcasting, as the traditional approaches fail due to their all-node-active assumption. The centralized dynamic solution is used for small networks and also for assessing other approaches. The distributed solution which relies on local information and operations has also implemented as an extension of centralised solution. The performance of both solutions has been examined under various network configurations and also compared the solutions with each other. The results obtained shows that the distributed solution works better than the centralized solution.

In future, more aspects of quality such as bandwidth and jitter can be considered for low duty cycle WSN. The solution can also be implemented in the real world sensor networks to carry out experiments to investigate the quality of its performance. The solution can also be extended to delay tolerant networks.

APPENDIX

SOURCE CODE

```
#include <senserp/senserp.h>
#include <random.h>
#include <cmu-trace.h>
#include <energy-model.h>
#define max(a,b)        ( (a) > (b) ? (a) : (b) )
#define CURRENT_TIME    Scheduler::instance().clock()
int hdr_senserp::offset_;
static class SENSERPHeaderClass : public PacketHeaderClass {
public:
SENSERPHeaderClass() :
PacketHeaderClass("PacketHeader/SENSERP", sizeof(hdr_all_senserp)) {
bind_offset(&hdr_senserp::offset_);  }
} class_rtProtoSENSERP_hdr;
static class SENSERPclass : public TclClass {
public:
SENSERPclass() : TclClass("Agent/SENSERP") {}
TclObject* create(int argc, const char*const* argv) {
assert(argc == 5);
return (new SENSERP((nsaddr_t)
Address::instance().str2addr(argv[4])));  }
} class_rtProtoSENSERP;
int SENSERP::command(int argc, const char *const * argv) {
if(argc == 2) {
Tcl& tcl = Tcl::instance();
if(strncasecmp(argv[1], "id", 2) == 0) {
```



```
tcl.resultf("%d", index); return TCL_OK;     }
if(strncasecmp(argv[1], "rt_entry", 2) == 0) {
if(scheme==1) sendMsg();
return TCL_OK; }
if(strcmp(argv[1], "nbr_table") == 0) {
if(scheme==1) nb_print(); return TCL_OK;}
if(strcmp(argv[1], "Routing") == 0) {
if(scheme==1) Routing(); return TCL_OK;}
if(strcmp(argv[1], "Routing_table") == 0) {
if(scheme==1) Routing_tables(); return TCL_OK;}
if(strcmp(argv[1], "tree_table") == 0) {
if(scheme==1) tree_print(); return TCL_OK; }
if(strcmp(argv[1], "nbr_table_announce") == 0) {
if(scheme==1) sendNbrTableAnn(); return TCL_OK; }
if(strncasecmp(argv[1], "start", 5) == 0) {
ntimer.handle((Event*) 0);htimer.handle((Event*) 0);if(index != 0)
ptimer.handle((Event*) 0);if(scheme == 2) { dtimer.handle((Event*) 0);
ctimer.handle((Event*) 0);} if(scheme==1) bwtimer.handle((Event*) 0);
return TCL_OK;} if(strncasecmp(argv[1], "sink", 4) == 0) {if(scheme==1)
send_announce();
printf("N (%.6f): sink node is set to %d, start announceing", CURRENT_TIME,
index);
return TCL_OK; }
if(strncasecmp(argv[1], "send_announce", 4) == 0)
{ if(scheme==1) send_announce();
return TCL_OK;}} else if(argc == 3) {
if(strcmp(argv[1], "index") == 0) {
index = atoi(argv[2]);return TCL_OK;}
else if(strcmp(argv[1],"log-target") ==0 ||strcmp(argv[1], "tracetarget") == 0) {
logtarget = (Trace*) TclObject::lookup(argv[2]);if(logtarget == 0)
```

```
return TCL_ERROR; return TCL_OK;}
else if(strcmp(argv[1], "drop-target") == 0)
{return TCL_OK;}else if(strcmp(argv[1], "if-queue") == 0) {
ifqueue = (PriQueue*) TclObject::lookup(argv[2]);
if(ifqueue == 0) return TCL_ERROR;
return TCL_OK;}
else if (strcmp(argv[1], "port-dmux") == 0) {
dmux_ = (PortClassifier*)TclObject::lookup(argv[2]);
if (dmux_ == 0){
fprintf (stderr, "%s: %s lookup of %s failed\n", __FILE__, argv[1], argv[2]);
return TCL_ERROR;}return TCL_OK; }}
return Agent::command(argc, argv);}
SENSERP::SENSERP(nsaddr_t        id)        :Agent(PT_SENSERP),htimer(this),
dtimer(this), ntimer(this), bwtimer(this),ctimer(this), ptimer(this), rqueue() {
printf("N (%.6f): Routing agent is initialized for node %d \n",
CURRENT_TIME, id);
index=id; logtarget = 0;
ifqueue=0;seqno=0;LIST_INIT(&nbhead);
LIST_INIT(&nbnbhead);bid = 0;bind("MAC_BW",&MAC_BW);
bandwidth=MAC_BW;bind("sleep_time",&sleep_time);
bind("dormant_time",&dormant_time);bind("scheme",&scheme);
TS=2.0;TW=5.0;} void       SENSERP::sendHello(){
Packet *p = Packet::alloc();
struct hdr_cmn *ch = HDR_CMN(p);
struct hdr_ip *ih = HDR_IP(p);
struct hdr_senserp_announce *am = HDR_SENSERP_ANNOUNCE(p);
am->bw=bandwidth;
am->x=node_->X();
am->y=node_->Y();
am->pkt_type = SENSERP_HELLO;
```



```
am->hop_count=1;
cout<<"Node: "<<index<<" send hello at "<<CURRENT_TIME<<endl;
if(scheme == 2){
nb_set*         n_s = new nb_set();
SENSERP_Neighbour *nb = nbhead.lh_first;
int i=0;
for(; nb; nb = nb->nb_link.le_next) {
n_s->nbrs[i] = nb->nb_addr;
i++;}
n_s->count = i;am->nb_set_=n_s;
am->seqno=++bid; am->hop_count=2;}
ch->ptype() = PT_SENSERP;
ch->size() = IP_HDR_LEN + am->size();
ch->addr_type() = NS_AF_NONE;
ch->prev_hop_ = index;
ih->ttl_ = 1;
Scheduler::instance().schedule(target_, p, 0.0);}
void    SENSERP::recvHello(Packet *p){
struct hdr_cmn *ch = HDR_CMN(p);
struct hdr_ip *ih = HDR_IP(p);
struct hdr_senserp_announce *am = HDR_SENSERP_ANNOUNCE(p);
if(scheme == 2){if(ih->saddr() == index){
Packet::free(p);return;}}
double d= getDistance(node_->X(),node_->Y(),am->x,am->y);
if(scheme == 2){
cout<<"Node: "<<index<<" recv hello from "<<ih->saddr()<<" at
"<<CURRENT_TIME<<endl;
if(am->hop_count == 2)
for(int i=0;i<am->nb_set_->count;i++){
```



```
set_changed = (set_changed || cov_set.add(am->nb_set_-
>nbrs[i],CURRENT_TIME + (2 * Hello_interval) ));}
set_changed = (set_changed || cov_set.add(ih->saddr(),CURRENT_TIME + (2 *
Hello_interval) ));
set_changed = (set_changed || recv_set.add(ih->saddr(),CURRENT_TIME + (2
* Hello_interval) ));
if(!(--am->hop_count <= 0)){
if(!(my_seq_list.get_seqno(ih->saddr()) >= am->seqno)) {
my_seq_list.add(ih->saddr(),am->seqno);
Packet *np = p->copy();
struct hdr_cmn *ch1 = HDR_CMN(np);
ch1->direction() = hdr_cmn::DOWN;
Scheduler::instance().schedule(target_, np, 0.1 * Random::uniform());}}}
double wt=0;
wt=getWeight(ch->size(),am->bw,d);
nb_insert(ih->saddr(),wt);}
else {
nb->weight=wt;
nb->nb_expire = CURRENT_TIME +(1.5* Hello_Loss * Hello_interval);}
Packet::free(p);}
int      SENSERP::check_all_nbrs_active(){
int f=1;
SENSERP_Neighbour *nb = nbhead.lh_first;
for(; nb; nb = nb->nb_link.le_next){
MobileNode *n1;
n1=(MobileNode*)(Node::get_node_by_address(nb->nb_addr));
if(n1->energy_model()->sleep_mode_==1)
f=0;}
return f;}
void    SENSERP::distributed(){
```



```
if(msg_recv_flag == 0){
check_for_forwarding_seq();}
else { message_arrival_proc();}}
void    SENSERP::check_for_forwarding_seq(){
if(need_to_compute_seq() == 1){
Compute_forwarding_seq();
if(check_scheduling_forwarding_seq() == 1){
issue_message_forwarding();
if(check_coverset_timeout() == 1){
reset_coverset();}}else{
if(check_coverset_timeout() == 1){
reset_coverset();}}}else {
if(check_scheduling_forwarding_seq() == 1){
issue_message_forwarding();
if(check_coverset_timeout() == 1){
reset_coverset();}} else {
if(check_coverset_timeout() == 1){
reset_coverset();}}}}}

int  SENSERP::need_to_compute_seq(){
for(int i=0;i<dest_list_.count;i++){
if(rt_table_.check(dest_list_.nodeid[i]) == -1)
return 1;} return 0;}
void    SENSERP::Compute_forwarding_seq(){
for(int i=0;i<nb_recvset_.count;i++){
rt_table_.add(nb_recvset_.nodeid[i],nb_recvset_.nexthop[i]);}
for(int i=0;i<nb_coverset_.count;i++){
rt_table_.add(nb_coverset_.nodeid[i],nb_coverset_.nexthop[i]);}}
int SENSERP::check_scheduling_forwarding_seq() {
for(int i=0;i<dest_list_count;i++){
```

```
if(rt_table_check(dest_list_nodeid[i]) == -1)
return 1; } return 0;}
void    SENSERP::issue_message_forwarding(){
double delay=0;
for(int i=0;i<dest_list_.count;i++){
if(rt_table_.check(dest_list_.nodeid[i]) == -1){
issue_message_forwarding(dest_list_.nodeid[i],delay);
delay += ARP_DELAY;}}}
void    SENSERP::reset_coverset(){{
nsaddr_t c_nid[100];
int count=0;
for(int i=0;i<nb_recvset_.count;i++){
if(nb_recvset_.exp[i] < CURRENT_TIME){
c_nid[count++] = nb_recvset_.nodeid[i];}}
for(int i=0;i<count;i++){
nb_recvset_.remove(c_nid[i]);}}
int SENSERP::is_it_new(nsaddr_t nid,int s){
 if(my_seq_list.get_seqno(nid) < s)
return 1;  return 0; }
void    SENSERP::create_update_coverset(set_ *s) {
for(int i=0;i<s->c_count;i++){
nb_recvset_.add(s->cov_set[i],s->sender,Hello_interval);}}
void    SENSERP::create_update_recvset(set_ *s){
for(int i=0;i<s->c_count;i++){
nb_coverset_.add(s->recv_set[i],s->sender,Hello_interval);}}
int      SENSERP::update_recvset_coverset(set_ *s){
int flag=0;
for(int i=0;i<s->c_count;i++){
flag = (flag || nb_recvset_.add(s->cov_set[i], s->sender, Hello_interval));}
for(int i=0;i<s->c_count;i++){
```



```
flag = (flag || nb_coverset_.add(s->recv_set[i],s->sender,Hello_interval));}
return flag;}
void    SENSERP::send_Ack(Packet *new_p){
struct hdr_ip *ih1 = HDR_IP(new_p);
Packet *p = Packet::alloc();
struct hdr_cmn *ch = HDR_CMN(p);
struct hdr_ip *ih = HDR_IP(p);
struct hdr_senserp_announce *am = HDR_SENSERP_ANNOUNCE(p);
am->pkt_type = SENSERP_ACK;
set_ *set = new set_();
set->sender = index;
for(int i=0;i<recv_set.count;i++) {
set->recv_set[i] = recv_set.nodeid[i]; }
set->r_count = recv_set.count;
for(int i=0;i<cov_set.count;i++){
set->cov_set[i] = cov_set.nodeid[i];}
set->c_count = cov_set.count;
am->set = set;
ch->ptype() = PT_SENSERP;
ch->size() = IP_HDR_LEN + am->size();
ch->addr_type() = NS_AF_INET;
ch->prev_hop_ = index;
 ch->next_hop_ = ih1->saddr();
 ih->saddr() = index;
ih->daddr() = ih1->saddr();
 ih->sport() = RT_PORT;
ih->dport() = RT_PORT;
ih->ttl_ = 1;
Scheduler::instance().schedule(target_, p, 0.0);}
void    SENSERP::check_nbr_hav_msg(){
```



```
int nbr_msg = check_neighbours_all_hav_msg();
if(nbr_msg == 1){
hdr_senserp_announce *rh = HDR_SENSERP_ANNOUNCE(new_msg);
delete_rcvset_coverset(rh->set);
if(check_coverset_timeout() == 1){
reset_coverset();}}else{
check_for_forwarding_seq(); }}
int SENSERP::check_neighbours_all_hav_msg() {
if(new_msg != NULL){
hdr_senserp_announce *am = HDR_SENSERP_ANNOUNCE(new_msg);
if(am->hop_count != 1)
return 1; }return 0;}
void    SENSERP::delete_rcvset_coverset(set_ *s){
for(int i=0;i<s->c_count;i++){
nb_coverset_.rt_delete(s->cov_set[i],s->sender);}
for(int i=0;i<s->r_count;i++){
nb_recvset_.rt_delete(s->recv_set[i],s->sender);}}}
```


REFERENCES

1. Feng Wang and Jiangchuan Liu “On Reliable Broadcast in Low Duty-Cycle Wireless Sensor Networks”, IEEE Transactions on Mobile Computing 2012.
2. Guo, S., Gu, Y., Jiang, B., and He, T. “Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links,” Proc. ACM MobiCom, 2009.
3. Gu, Y. and He, T. “Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links,” Proc. ACM Int’l Conf. Embedded Networked Sensor Systems (SenSys), 2007.
4. Kyasanur, P., Choudhury, R.R. and Gupta, I. “Smart Gossip: An Adaptive Gossip-based Broadcasting Service for Sensor Networks,” Proc. IEEE Int’l Conf. Mobile Adhoc and Sensor Systems(MASS), 2006.
5. Levis, P., Patel, N., Culler, D. and Shenker, S. “Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks,” Proc. First Conf. Symp. Networked Systems Design and Implementation (NSDI), 2004.
6. Miller, M., Sengul, C. and Gupta, I. “Exploring the Energy-Latency Tradeoff for Broadcasts in Energy-Saving Sensor Networks,” Proc. IEEE Int’l Conf. Distributed Computing Systems (ICDCS), 2005.
7. Ni, S.-Y., Tseng, Y.-C., Chen, Y.-S. and Sheu, J.-P. “The Broadcast Storm Problem in a Mobile Ad Hoc Network,” Proc. ACM MobiCom, 1999.
8. Rajendran, V., Obraczka, K., and Garcia-Luna-Aceves, J. “Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks,” Proc. ACM Int’l Conf. Embedded Networked Sensor Systems (SenSys), 2003.
9. Sun, Y., Gurewitz, O., Du, S., Tang, L., and Johnson, D. B. “ADB: An Efficient Multihop Broadcast Protocol Based on Asynchronous Duty-Cycling in Wireless Sensor Networks,” Proc. ACM Conf.Embedded Networked Sensor Systems (SenSys), 2009.
10. Sun, Y., Gurewitz, O., and Johnson, D.B. “RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks,” Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys), 2008.

LIST OF PUBLICATIONS

1. Sherin George and Sivan Arul Selvan, K. “Enhancing the Quality of Broadcast Service in Low Duty-Cycle WSN” International Conference on Global Innovations in Technology and Sciences, Saintgits College of Engineering, Kottayam, 4th April 2013.
2. Sherin George and Sivan Arul Selvan, K., “Enhancing Quality of Service in Duty Cycle Aware Wireless Sensor Networks” National Conference on Information Technology, Communications and Signal Processing, Government Engineering College, Wayanad, 2nd March 2013.
3. Sherin George and Sivan Arul Selvan, K., “A Consistent Data Propagation in Low Duty-Cycle Wireless Sensor Networks” Fifth National Conference on Computing, Communication and Information Systems, Sri Krishna College of Engineering and Technology, Coimbatore, 9th February 2013.