Congestion Control In Wireless Sensor Network

by

Anand Kulkarni

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 09, 2010

Keywords: Wireless Sensor Network, Directed Diffusion, Congestion Control

Approved by

Alvin Lim, Chair, Associate Professor, Computer Science and Software Engineering
David Umphress, Associate Professor, Computer Science and Software Engineering
Drew Hamilton, Associate Professor, Computer Science and Software Engineering

Abstract

For this thesis we consider Directed Diffusion routing protocol. It manages paths in a data-centric way. A sink node publicizes its interest in some events. Such an interest will be disseminated within the network and received by nodes that have data on those events. During this process, gradients that record two-way information along all possible paths are established. Afterward, a mechanism called path reinforcement is used to select high-quality routes. Directed Diffusion also provides two sets of application programming interface (API). One set is network routing API, which enables upper-layer entities in sources and sinks to communicate through the network. This API takes the form of publish/subscribe paradigm. A sink node can subscribe to interesting events. Also, sources will publish their sensed events. The Directed Diffusion platform takes care of the underlying implementation. To allow processing in intermediate nodes, Directed Diffusion also offers another API called filters. Application-specific operations can be implemented as filters to manipulate data packets as they pass through the network. Filters are provided through an interface to specify interest in particular events. For each filter, a priority is associated. Although directed diffusion is a widely used in sensor networks, it has several weaknesses. It only considers delay as the routing metric to select best path. As a result, high throughput may not be achieved. Its failure to consider deadline during the route selection lowers the percentage of packets which can meet the deadline of real-time traffic. Its reliance on flooding gives rise to differences in interference levels during the exploratory data phase and the actually data transmission phase, which makes the route decision even more inaccurate. It does not consider path interference which worsens it performance during high rate data transmission. We propose a routing protocol to augment and extend directed diffusion to handle these issues. First, we propose an algorithm which considers throughput, delay and

intra-flow interference metric for a path. ETX (Expected Transmission Count) is used as the metric for measuring link quality. We improve on QoS-based routing by limiting interference in lossy links. We use an improved method for computing aggregate ETX for a path that increases end-to-end throughput. We compute multiple paths between source-sink pair. We choose best two paths based on the aggregate ETX metric and delay. The main goal of our protocol design is to use these routes to achieve high throughput and less delay. We monitor the data transmission over a period of time. We measure maximum queue length over the active path, we measure the rate of increase in queue length, and in addition we keep track of ETX which can give us an idea of interference over links on active path. We have setup inter-node queues, and intra node queues to handle data flowing through nodes. We maintain a threshold value for all the queues. We check the current queue state after an interval of δ. Depending on the increase in queue length we estimate if the queue might be over the threshold value in time frame δ. We also measure the ETX value to see if it shows sign of interference. If these metrics show sign of congestion on the active route, then sink will signal the source to switch data transfer to an alternate path. This switching of path is done by sending an explicit control message called SWITCH_PATH_MESSAGE to source across the network from sink to source. The control message is sent using an alternate path which should be used for data transfer. This makes our protocol end-to-end. If switching of path does not help to control congestion, we use multiple paths to cope with high data rates. As a last option, if even this does not mitigate congestion, then we limit source sending rate.

Acknowledgments

I would like to express my appreciation to Dr. Alvin Lim for the invaluable guidance he has provided throughout my study at Auburn University. I would like to express my gratitude to the advisory committee members, Dr. David Umphress and Dr. Drew Hamilton for taking time to review my work. I owe my deepest gratitude to Dr. David Umphress and Dr. Daniela Marghitu for funding my Master's education and being a constant source of support and encouragement.

My friend RaghuKisore Neelisetti has been very helpful and has made significant contribution to this research. I would also like to thank several fellow students including Santosh Kulkarni, Vasavi Chilamantula, Manish Kulkarni, Indraneil Gokhale, Salil Vaijapurkar, and my other lab mates for their support and friendship.

I owe my deepest gratitude to my parents who helped me in every step of my life and pursue my Master's education. I couldn't have made it without my sister Nilima, my brother Amol and his wife Sapna, and I thank them. I am grateful to Kirti for the very special person she is, and for the incredible amount of patience she had with me in the last 6 months. The knowledge that my family will always be there to pick up the pieces is what allows me to repeatedly risk getting shattered.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Auburn Auburn University

LoA List of Abbreviations

Chapter 1

Introduction

In September 1999, networked micro sensors technology was heralded as one of the 21 most important technologies for the 21st century by Business Week [1]. In recent years, due to advances in low-power circuit and radio technologies, Wireless sensor networks have emerged as a new information-gathering paradigm based on the collaborative effort of a large number of sensing nodes [2]. The recent advancements in MEMS technology, processor design, and wireless communication have enabled a wide range of monitoring applications using networks of sensor nodes. There is a considerable amount of research in the area of wireless sensor networks ranging from real-time tracking to ubiquitous computing, where users interact with potentially large numbers of embedded devices [3]. A typical sensor network is formed by a large amount of nodes. Each sensor consists of small individual microcontroller fitted with sensors and some means of communication such as radios. The main components of sensors consist of a sensing unit, a processing unit, a transceiver, and a power unit. A typical sensor is shown in Figure 1.1 [40]:

- Sensing Unit:

  The main functionality of the sensing unit is to sense or measure physical data from the target area. The analog voltage which is generated by the sensor corresponding to an event is then digitized by an analog-to digital converter (ADC) and then delivered to the processing unit for more analysis

- Processing Unit:

  The processing unit plays a major role in managing collaboration with other sensors to achieve the predefined tasks. There are currently several families of this unit including

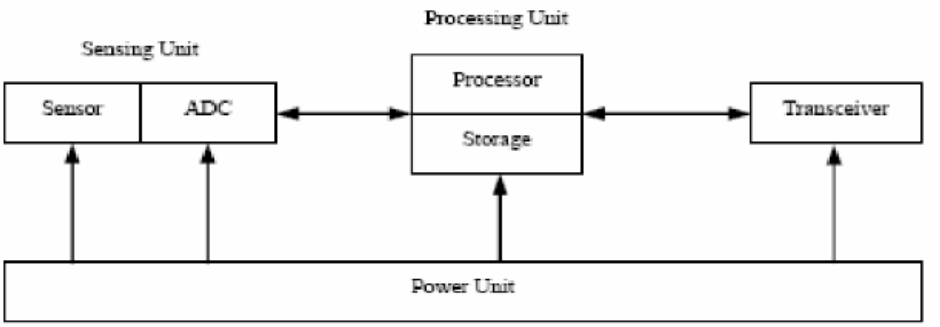

Figure 1.1: Sensor Structure

microcontrollers, microprocessors, and field-programmable gate arrays (FPGAs). The Non-volatile memory and interfaces such as ADCs can be integrated onto a single integrated circuit [41] [42]. The processing unit needs storage for tasking and to minimize the size of transmitted messages by local processing and data aggregation [43]. Flash memory is widely used due to its cost and storage capacity.

- Transceiver:

  There are three deploying communication schemes in sensors including optical communication (laser), infrared, and radiofrequency (RF). RF is the most easy to use but requires antenna.

- Power Unit:

  Power consumption is a major weakness (problem) of sensor networks.. 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.

- Transport Protocol in Sensor Networks:

  Depending on the type of application, each sensor node may be required to perform some local (in - network) computations and data aggregation. Applying TCP to wireless sensor networks is expensive because of its three-way handshake mechanisms and

packet header size. UDP is considered to be more suitable for sensors although it was designed to provide unreliable data transport.

Usually there is no pre-determined topology for a sensor network. Instead, these sensor nodes construct and dynamically maintain the structure of the network through wireless communication. They are expected to be deployed over wide areas and transmit gathered data to one or several central nodes, which is also called the base stations / sink. Since the distance between a node and a base station may exceed the range of the radio, relaying by intermediate nodes needs to be performed so that the data can ultimately reach the base station. Unlike routers in the Internet, the relaying devices are themselves sensor nodes that are simultaneously gathering and transmitting data. Hence, each node serves as a router and a host. They forward packets to other nodes if they are for different nodes. This makes it easy to achieve ubiquitous computing. As many nodes serve as routers, they will be forwarding packets to multiple nodes. So this means that they will be acting as junctions. This results in congestion at those nodes. Currently, there are many ways of efficiently getting at specific information from the network. [4] proposes data centric storage in order to efficiently retrieve, for instance, the highest temperature reading in the network. As the main task of WSN [14] is to gather information from the physical world, all the data flows go towards a common sink. In [8], certain attributes such as the total number of nodes in the networks can be processed within the network before reaching the base station / sink. However, certain applications of sensor networks require information to be gathered from all or a subset of nodes, information that cannot be aggregated within the network and that a minimum level of fidelity is required of.

Sensor networks come in a wide variety of forms, covering different geographical areas, being sparsely or densely deployed, using devices with a variety of energy constraints, and implementing an assortment of sensing applications. One application driving the development of sensor networks is the reporting of conditions within a region where the environment abruptly changes due to an observed event, such as in habitat monitoring, target detection,

earthquakes, floods, or fires. The research of sensor networks was initially driven by military applications like battlefield surveillance and enemy tracking. During the Cold War, the Sound Surveillance System (SOSUS), a system of acoustic sensors on the ocean bottom, was deployed at strategic locations to detect and track quiet Soviet submarines. Modern research on sensor networks started around 1980 with the Distributed Sensor Networks (DSN) program at the Defense Advanced Research Projects Agency (DARPA). Researchers at Carnegie Mellon University (CMU), Pittsburgh, PA, Massachusetts Institute of Technology (MIT), Cambridge, University of Massachusetts, Amherst were working on different sensor network test beds. To name just a few examples, one can deploy a sensor network to study the behavior of an endangered species, to report fires in a forest, to monitor the health of a building, or to collect the traffic information on a busy highway, etc. These networks will usually be left unattended, with each individual node sensing the physical environment, performing processing if necessary, and reporting the data to the sinks through multi-hop wireless communication. In order to conserve energy which is the rarest resource, a sensor network requires low reporting rates from the source nodes during the periods when the specific events are not observed. However, a much higher rate will be needed as soon as these events occur such that necessary actions can be taken promptly. For example, a temperature sensor as part of the fire detection application only needs to report 1 packet every 5 minutes when its reading is below 50 degree, but it must report more than 100 packets per minute once its readings exceed 100 degree [5]. As a result, sensor networks experience traffic alternating between periods with a very low traffic volume (referred to as dormant state) and periods with a high traffic volume - referred to as crisis state [6]. Once in a crisis state, the sudden traffic increase may lead to congestion in the network. When congestion occurs, the network will enter into an unstable state and packets will be randomly dropped. This is particularly undesirable because the data generated during the crisis state are of great importance, often critical, to the applications [7].

The Wireless Sensor Network is an event driven paradigm that relies on the collective effort of numerous micro sensor nodes. This has several advantages over traditional sensing including greater accuracy, larger coverage area and extraction of localized features. In order to realize these potential gains, it is imperative that desired event features are reliably communicated to the sink. To accomplish this, a reliable transport mechanism is required in addition to robust modulation and media access, link error control and fault tolerant routing. The WSN paradigm necessitates a collective event-to-sink reliability notion rather than the traditional end-to-end notion. Sensor networks are being increasingly deployed for surveillance and monitoring applications. These networks will suffer from severe congestion as soon as the target events occur. During congestion, important data packets may be dropped, which can essentially nullify the purpose of sensor networks. There is a critical need for new thinking regarding overload traffic management in sensor networks. It has now become clear that experimental sensor networks (e.g., mote networks) and their applications commonly experience periods of persistent congestion and high packet loss, and in some cases even congestion collapse [9]. This significantly impacts application fidelity measured at the physical sinks, even under light to moderate traffic loads, and is a direct product of the funneling effect; that is, the many-to-one multi-hop traffic pattern that characterizes sensor network communications.

Sensor networks typically operate under light load and then suddenly become active in response to a detected or monitored event. Depending on the application this can result in the generation of large, sudden, and correlated impulses of data that must be delivered to a small number of sinks without significantly disrupting the performance (i.e., fidelity) of the sensing application. Although a sensor network may spend only a small fraction of time dealing with impulses, it is during this time that the information it delivers is of greatest importance. The flow of events has similarities to the flow of people from a large arena after a sporting event completes. A major limitation in the design of existing sensor networks is that they are ill-equipped to deal the funneling of events and increasing traffic demands [9].

This leads to increased transit traffic intensity [10], congestion [6], and large packet loss [11] (which translates into wasted energy and bandwidth). As a result, the sensors nearest the sink will use energy at the fastest rate, shortening the operational lifetime of the network. Many packets may be accumulated at congested nodes and discarded finally. This will increase delay, high overhead of retransmission, low throughput in Adhoc network. If node is congested but the channel quality is high the node will still transmit the data packets at a high rate. Due to this congestion the results will be worse. Congestion in Wireless Sensor Networks (WSNs) has following negative impact on performance:

- A drastic decrease in throughput and
- Increased per-packet energy consumption.

The congestion problem in WSNs is quite different from traditional networks. All the data flows go towards a common sink. Due to this centralized traffic pattern, just bypassing the hot-spots is ineffective to eliminate the congestion because it will reappear near the sink. Most congestion control in WSNs has only focused on traffic control (including end-to-end and hop-by-hop). i.e. they basically try to throttle the incoming traffic into the network once congestion is detected. Although traffic control strategies are effective to alleviate congestion in traditional networks, and are also suggested in some wireless sensor network scenarios, they are restricted or even unsuitable for special purposes for the following reasons.

1. Reducing source traffic during a crisis state [6] is undesirable since it will significantly violate fidelity requirements. It may be a better option to increase capacity by turning on more resources to accommodate excessive incoming traffic during the crisis state. The wireless sensor network can provide elastic resource availability because of its dense deployment, unlike its wired or other wireless counterparts. This distinct advantage enables WSN to employ adaptive capacity planning schemes to avoid possible congestion and satisfy fidelity requirements at the same time.

2. It is highly likely that the congestion caused by burst traffic is often transient by nature. For example, the sensor nodes will generate transient bursts of traffic when the abnormal events occur. It could be inefficient to cope with these transient congestions by the traffic control based on feedback, and they may be alleviated through quickly adjusting the network resource provisioning. Conventional end-to-end reliability definitions and solutions are inapplicable in the WSN regime and would only lead to a waste of scarce sensor resources.

3. There is usually an abundance of resources in sensor networks (unlike its wired or other wireless counterparts) because these networks are usually densely deployed in order to achieve a reasonable network lifetime.

The transport of event impulses is likely to lead to varying degrees of congestion in sensor networks. There are many different types of routing protocols in wireless sensor networks. Routing protocols that require location information, such as LAR [17], GPSR [18], and DREAM [19], do not need to flood routing requests. Others, such as DSR [20], AODV [21], ZRP [22], and TORA [23], suffer from the effects of flooding, even with some optimizations, since nodes do not know their locations. As pointed out by Gupta and Kumar [15], the fundamental reason leading to the degradation of the performance as number of nodes increases is the fact that each node has to share the radio channel with its neighbors. In order to illustrate the congestion problem consider the following simple but realistic simulation scenario.

Figure 1.2 shows the impact of congestion on data dissemination in a sensor network for a moderate number of active sources with varying reporting rates. The ns-2 simulation [16] results are for the well-known directed diffusion [14] scheme operating in a moderately sized 30-node sensor network using a 2 Mbps IEEE 802.11 MAC with 6 active sources and 3 sinks. The 6 sources are randomly selected among the 30 nodes in the network and the 3 sinks are uniformly scattered across the sensor field. Each source generates data event packets at a common fixed rate while the sinks subscribe (i.e., broadcast corresponding

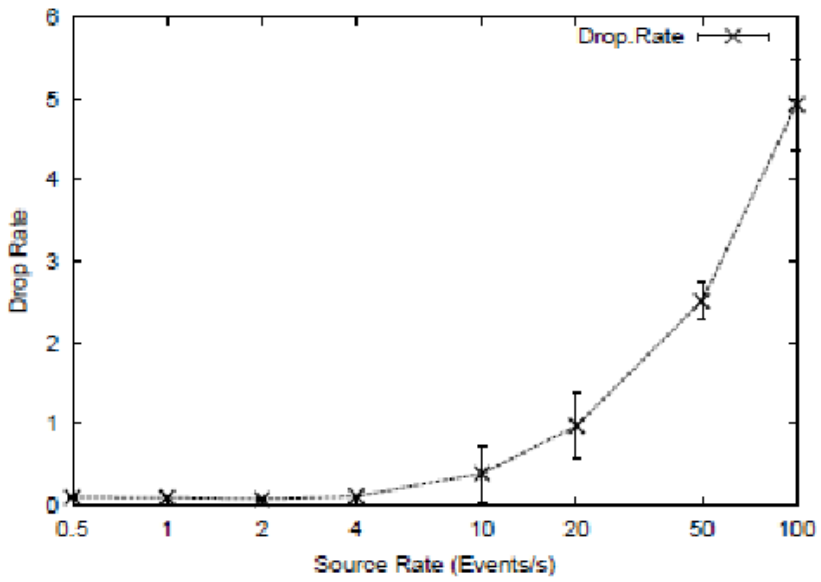

Figure 1.2: Congestion effect in WSN

interest) to different sources at random times within the first 20 seconds of the 50-second simulation scenario. Event and interest packets are 64 and 36 bytes in size, respectively. The plot illustrates that as the source rate increases beyond a certain network capacity threshold (10 events/sec in this network), congestion occurs more frequently and the total number of packets dropped per received data packet at the sink increases rapidly. The plot shows that even with low to moderate source event rates there is a large drop rate observed across the sensor network. For example, with a source event rate of 20 events/sec in the network, one packet is dropped across the sensor field for every data event packet received at the sink. Dropped packets can include MAC signaling, data event packets themselves, and the diffusion messaging packets. The drop rates shown in figure represent not only significant packet losses in the sensor network, but more importantly, the energy wasted by the sensing application. Directed diffusion, though regarded as an epidemic algorithm in [31] since it avoids broadcast storm, does not perform well with interest flooding. No matter what metrics are used in selecting a route (basic directed diffusion uses delay), the route which performs best during route discovery phase may not perform well during the actual

data transmission phase due to differences in interference levels caused by the different traffic patterns and changes in data rates at source. Interest flooding increases traffic in the network and causes maximum level of interference. Exploratory data are flooded to determine the best path, which follows gradients established in the interest propagation phase. Actual application traffic only flows through the reinforced path, which is not affected by inter-path traffic at all, assuming there is no other data transmission at that time. Every node has an interference range. Interference set [32] and conflict graph [33] are used to schedule network traffic or theoretically analyze the impact of interference on wireless networks. However, no routing protocol could have the prior knowledge about which path the actual data traffic will go through and what the traffic pattern will be like before the route is determined. Our goal is to design solutions which make more accurate routing decisions by reducing the interference level and hence packet loss, delay and congestion, during the route discovery phase and making it more similar to that during the actual data transmission phase. If due to changes in network topology and source data rate, there is congestion in the network, we try to overcome this congestion by choosing an alternate path which is unused and does not have congested nodes on it. This M.S. thesis aims to study the problem of congestion in WSNs and identify causes and symptoms strictly related with the philosophy and design of WSNs. The main objective is to examine the behavior of several network parameters and their impact to congestion in WSNs and address a framework for congestion control and avoidance that is different from existing traditional schemes that are based on rate control to alleviate congestion. The simulation scenarios were implemented and tested with the use of the NS-2 simulator.

The thesis is organized as follows. In Chapter 2 we give background information of congestion in computer Networks, and also describe pertinent WSNs features. We cover the congestion techniques used by other researchers in this chapter. In Chapter 3 we analyse congestion in WSNs and its design philosophy impact on congestion problem. We state objective of this thesis, and define the problem statement. We also discuss the techniques

we will be using to solve the problem. In Chapter 4 we get an overview of overall protocol design and the congestion control mechanism that we will implement in the protocol. In Chapter 5 we present all the implementation details of the protocol and scenarios which will give an overall idea of the working of protocol. In Chapter 6 we discuss the simulation setup and details of the parameters we use in simulation. We also go through the simulation results and see how the protocol performs under different circumstances. Finally in Chapter 7 we present the contributions of the thesis and conclusions that evolve from the Thesis Chapters and some suggestions are introduced for future work.

Chapter 2

Related Work

2.1 Routing and Congestion

The problem of congestion control stems from the fact that the route selected in route discovery phase is not the best one, or cannot support the application needs. In this thesis we trace back the congestion problem to routing techniques. Our aim is to select the best route which will support high throughput with lowest possible delays. As our congestion protocol is end to end, we try to minimize the delay for routes which will help us reduce the round trip time (RTT) for a given network topology. Routing metrics in wireless ad hoc networks are important considerations due to the unpredictability and heterogeneity of link qualities [57]. Existing wireless ad hoc routing protocols typically select routes using minimum hop count, e.g. DSR [20] and DSDV [58]. Directed diffusion [14] selects routes in sensor networks with the least delay. Recently, many new link quality metrics have been proposed. [59] compares the performance of the following three metrics. Adya et al. [59] measures the round trip delay of unicast probes between neighboring nodes and proposes Per-hop Round Trip Time (RTT). Per-hop Packet Pair Delay (PktPair) measures the delay between a pair of back-to-back probes to a neighbor node [59]. Expected Transmission Count (ETX) [57] measures the loss rate of broadcast packets between pairs of neighboring nodes and estimates the number of retransmissions required to send unicast packets. Weighted Cumulative Expected Transmission Time (WCETT) [60] is used for selecting channel-diverse paths and accounts for the loss rate and bandwidth of individual links. Park et al. [61] presented a new metric, Expected Data Rate (EDR), for accurately finding high-throughput paths in multi-hop ad hoc wireless networks based on a new model for transmission interference. Unfortunately, none of these metrics can be directly applied to wireless sensor network that simultaneously

take into account delay, throughput and interference. Furthermore, none of previous papers proposed a combined metric for sensor networks with all those considerations. In [62], ETX was incorporated into DSR and DSDV to improve throughput with little consideration of delay or interference. WCETT [60] is more suitable in multi radio wireless mesh networks. EDR [61], unlike ETX, cannot be computed dynamically. More space and computation are required by EDR when it is incorporated into DSR and AODV. Interference-aware protocols have recently been explored in multi-hop wireless networks. [67] studies routing problems in a multihop wireless network using directional antennas with dynamic traffic and presented new definitions of link and path interference. In their other paper [62], they present routing algorithms to compute interference-optimal cost-bounded paths and an optimal bandwidth allocation algorithm to allocate timeslots. We do not give detailed analysis, computation and implementation for interference because we try to make full use of the ETX information. [69] and [70] give the throughput bounds and capacity for interference-aware routing in wireless networks respectively. We may use them to test our protocol by observing the throughput performance. [71] derives an interference aware metric NAVC based on the information collected from 802.11 MAC. In [72], an interference aware routing scheme is designed to alleviate the near-far problem at the network level for cellular systems. EIBatt et. al. [73] address the problem of interference aware routing by coupling the lower three layers of the ISO Open Systems Interconnection (OSI) protocol stack. We only use ETX, the link layer indicator, to measure the link quality as well as interference to simplify the problem. Nguyen et. al. [74] considers radio interference and modifies OLSR routing protocol for bandwidth reservation and interferences. Our paper modifies directed diffusion, a routing protocol for wireless sensor networks, to take into account throughput, interference and delay. For our purposes, we need a routing protocol which achieves maximum throughput with lowest possible delay. This will help us to minimize retransmissions and packet losses in the network due to interference/collisions and link quality. To have this kind of metric we will be using

ETX which takes link quality/loss rate over a link along with a delay metric. This technique will be explained in detail in later sections.

RMST is a transport layer paradigm designed to complement directed diffusion [12] by adding a reliable data transport service on top of it. Its a NACK based protocol like PSFQ, which has primarily timer driven loss detection and repair mechanisms. It does not provide with any congestion control mechanism. TARA discusses the network hotspot problem and presents a topology aware resource adaptation strategy to alleviate congestion in sensor network.

2.2 MAC and Link Layer Reliability for congestion control

[45] proposes auto rate protocol at MAC layer. They use queue length to get estimate of congestion in the network. [45] assumes that the nodes total buffer size is Q and the current number of packets in the buffer is q. When the buffer is full, $i.e., Q = q$, the node is congested completely, the packets arrived at this node will be discarded. Conversely, when the buffer is not full, the input packet rate R_{in} and output packet rate R_{out} are monitored. R_{in} is the reciprocal of ΔT_{in}, i.e.

$$R_{in} = \frac{1}{\Delta T_{in}} \text{ , where } \Delta T_{in} \text{ represents the packet arrival interval.} \tag{2.1}$$

Packet arrival interval is defined as the time interval of two consecutive packets received at the node MAC layer. R_{out} is the reciprocal of ΔT_{out},

$$i.e., R_{out} = \frac{1}{\Delta T_{in}} \text{ , where, } \Delta T_{out} \text{ represents the packet service time.} \tag{2.2}$$

Packet service time is referred as the time interval between the time that a packet arrives at MAC layer and the time that it is transmitted successfully, ΔT_{out} is the sum of the time for queue, collision, back off and transmission [46]. The basic thought of [45] is

that the receiver measures its congestion condition and sends the congestion information back to the sender along with the transmission rate selected based on the channel condition and the sender transmits a limited number of back-to-back data packets at the selected rate according to the feedback information. If the receiver is less congested, the sender transmits as many back-to-back packets as possible at a higher rate while the channel quality is high. If the receiver is badly congested, the sender decreases the number of back-to-back packets and transmits them at an appropriate rate to alleviate the congestion. The drawback of this scheme is that the frame formats of RTS, CTS, ACK and DATA should be modified. Hence, it changes the 802.11 standard and cannot be generalized.

A traffic-aware dynamic routing (TADR) [47] algorithm is proposed to route packets around the congestion areas and scatter the excessive packets along multiple paths consisting of idle and under-loaded nodes. TADR algorithm is designed through constructing a mixed potential field using depth and normalized queue length to force the packets to steer clear of obstacles created by congestion and eventually move towards the sink. Traffic-aware dynamic routing (TADR) algorithm to route packets around the congestion areas and scatter the excessive packets along multiple paths where the idle or under-loaded nodes are sufficiently utilized in response to congestion under the fidelity requirements. The cornerstone of the TADR algorithm is to construct two independent potential fields using depth and queue length respectively. Normally, the depth field would find the shortest paths for packets. Once the queue length grows over a certain threshold, which always means congestion, the packets would flow along other suboptimal paths or just be cached in areas with more buffers.

The algorithms of ESRT [5] mainly run on the sink, with minimal functionality required at resource constrained sensor nodes. ESRT protocol operation is determined by the current network state based on the reliability achieved and congestion condition in the network. If the event-to-sink reliability is lower than required, ESRT adjusts the reporting frequency of source nodes aggressively in order to reach the target reliability level as soon as possible. If

the reliability is higher than required, then ESRT reduces the reporting frequency conservatively in order to conserve energy while still maintaining reliability. This self-configuring nature of ESRT makes it robust to random, dynamic topology in WSN. In order to determine the current network state S_i in ESRT, the sink must be able to detect congestion in the network. However the conventional ACK/NACK-based detection methods for end-to-end congestion control purposes cannot be applied here. The reason once again lies in the notion of event-to-sink reliability rather than end-to-end reliability. Only the sink, and not any of the sensor nodes, can determine the reliability indicator I and act accordingly. Moreover, end-to-end retransmissions and ACK/NACK overheads are a waste of limited sensor resources. Hence, ESRT uses a congestion detection mechanism based on local buffer level monitoring in sensor nodes. Any sensor node whose routing buffer overflows due to excessive incoming packets is said to be congested and it informs the sink of the same. At the same time ESRT makes the reasonable assumption that the sink is powerful enough to reach all source nodes by broadcast. Flows are not differentiated and sending rates of all flows are throttled when congestion is detected. On reception of packets with congestion notification bit high, sink node regulates the reporting rate by broadcasting a high energy control signal so that it could reach to all sources. This high powered congestion control signal may disrupt some other transmissions. Also the assumption of congestion notification by the sink node is very optimistic. In ESRT, the sink is required to periodically configure the source sending rate to avoid congestion. Once detecting congestion, all the data flows are throttled to a lower rate. In [5], the sink uses congestion feedback from sensor nodes to broadcast a notification to reduce reporting frequency. The effectiveness of this method is dependent on the persistence of congestion and the feedback latency. If congestion is transient, and feedback latency is significantly large, the notification arrival may arrive when congestion is no longer present. Feedback latency is in turn dependent on the diameter of the network, and thus the solution may not scale to huge sensor networks experiencing transient congestion.

Adaptive Resource Control Scheme to Alleviate Congestion in Sensor Networks [7] investigates a framework to alleviate congestion by turning on more resources as soon as the congestion is detected. In great detail it examines a family of strategies of managing the network resource during a crisis state. These strategies include the following features:

- Congestion alleviation:

  In this paper, they investigate the approach of using adaptive resource control strategies to alleviate congestion. [7] suggest that more routing paths need to be set up in order to share the load after the flow level congestion is detected. Sometimes, we need to wake up the sleeping nodes (these nodes were in sleep state to conserve energy) to form new routing paths. It is particularly important to control the resources at different temporal and spatial granularities.

- Energy awareness:

  While the long-term resource control scheme can reduce the congestion within the network, it also significantly increases the energy consumption. As a result, we would like to turn off the extra resources as soon as the source traffic decreases. Their scheme achieves this because we can detect the congestion change within the network timely.

This papers resource control scheme has two phases:

1. To increase resource provisioning as soon as congestion occurs; and

2. To reduce the resource budget as soon as congestion subsides.

Their scheme has managed to balance the need of reliably shipping data packets to the sink during a crisis state and the need of conserving energy consumption. The algorithm involves three steps:

1. Congestion Detection: The upstream nodes of a flow periodically notify their congestion level to downstream nodes by embedding their perceived congestion level into

the header of data packets. After detecting the flow-level congestion exceeds a predefined congestion threshold, the first node whose congestion level is below the hotspot proximity threshold will become the initiator and start creating multiplexing paths.

2. Alternative Path Creation: Building multiplexing paths is a challenging task. If the path is too close to the hot spot, it does not help, but it will further interfere with the already bad congestion. On the other hand, if the path is too far from the original one, then the packet delivery latency and the energy consumption will increase. In addition, building multiplexing path can be made even more complicated by other factors. For example, one may not be able to find a path from the initiator to the source; or there may exist another hot spot within the network. In this paper, we propose to solve the aforementioned dilemma by examining each candidate nodes congestion level.

3. Traffic Multiplexing: The node(s) where the multiplexing paths meet the original path are referred to as traffic dispatcher. The dispatcher will evenly distribute the traffic between multiple paths in a round robin fashion so that the congestion on the original path can be alleviated.

In [7], the Adaptive Rate Control (ARC) scheme monitors the injection of packets into the traffic stream as well as route-through traffic. At each mote, an estimation of the number of downstream motes is made, and the bandwidth split proportionally between locally generated and route through traffic, with preference given to the latter. The resulting bandwidth allocated to each mote is thus approximately fair. Reduction in transmission rate of route-through traffic has a backpressure effect on downstream motes, which can then reduce their generation rates.

In CODA [6], they present a detailed study on congestion avoidance in sensor networks. The basic idea is that as soon as congestion occurs, the source (or an intermediate nodes) sending rates must be reduced to quickly release the congestion. In the simple case, as soon as a node detects congestion, it broadcasts a backpressure message upstream. An upstream

node that receives the backpressure can decide to drop packets, preventing its queue from building up and thus controlling congestion. If multiple sources are sending packets to a sink, CODA also provides a method of asserting congestion control over these multiple sources by requiring constant feedback (ACKs) from the sinks. If a source does not receive the ACKs at predefined times, it will start throttling the sending rates.

[48] Proposed hop by hop rate adjustment algorithm and local link utilization approach. The rate adjustment algorithm is used by each sensor node. This algorithm continuously runs in a sensor to ensure the optimal queue occupancy and scheduling rate. The hop by hop rate adjustment based on the feedback information of immediate downstream node. The key consideration of this algorithm is to maintain optimal queue occupancy adjusting the scheduling rate. Scheduling rate is adjusted by an upper bound of maximal allowable scheduling rate. Thus it ensures that scheduling rate of child nodes must not be greater than scheduling rate of their parent. Still if any node determines that some of its parents child nodes are idle then it readjusts scheduling rate. We make use of the queue implementation in this paper.

Siphon - Overload Traffic Management using Multi-radio Virtual Sinks in Sensor Networks [9] proposes to exploit the availability of a small number of all wireless, multi-radio virtual sinks that can be randomly distributed or selectively placed across the sensor field. Virtual sinks are capable of siphoning of data events from regions of the sensor field that are beginning to show signs of high traffic load. Siphon is based on a Stargate implementation of virtual sinks that uses a separate longer-range radio network (based on IEEE 802.11 [49]) to siphon events to one or more physical sinks, and a short range mote radio to interact with the sensor field at siphon points. [9] proposes to randomly distribute or selectively place (as the case may be) a small number of all-wireless multi-radio virtual sinks (VSs) that are capable of offering overload traffic management services to the existing low-power sensor network. While such special nodes can be exploited to support a variety of application specific (e.g., aggregation, coding, low-delay transport) and common network functions

(e.g., storage, localized activation), we focus here on their ability to selectively siphon of data events from regions of the sensor field that are be- ginning to show signs of overload. In essence, virtual sinks operate as safety valves in the sensor field that can be used on-demand to divert selected packets from areas of high load, alleviating the funneling effect, in order to maintain the fidelity of the application signal at the physical sink.

Considering a sensor network consisting of such multi-purpose nodes, [50] proposes Prioritized Heterogeneous Traffic-oriented Congestion Control Protocol (PHTCCP) which ensures efficient rate control for prioritized heterogeneous traffic. Our protocol uses intra-queue and inter-queue priorities for ensuring feasible transmission rates of heterogeneous data. It also guarantees efficient link utilization by using dynamic transmission rate adjustment. PHTCCP uses hop-by-hop rate adjustment which ensures that heterogeneous data reach to the base station at their desired rates. The output rate of a node is controlled by adjusting the scheduling rate. In [50], the packet service ratio reflects the congestion level at each sensor node. When this ratio is equal to 1, the scheduling rate is equal to the forwarding rate (i.e., average packet service rate). When this ratio is greater than 1, the scheduling rate is less than the average packet service rate. Both of these cases indicate the decrease of the level of congestion. When it is less than 1, it causes the queuing up of packets at the queue. This also indicates link level collisions. Thus, the packet service ratio is an effective measure to detect both node level and link level congestion. This might be wrong because it might not reflect the network condition directly. There can be other reasons. Different types of data generated from such types of motes might have different transmission characteristics in terms of priority, transmission rate, required bandwidth, tolerable packet loss, delay demands etc.

STCP [51] provides a generic, scalable and reliable transport layer paradigm for sensor networks. Before transmitting packets, sensor nodes establish an association with the base station via a Session Initiation Packet. The session initiation packet informs the base station of the number of flows originating from the node, the type of data flow, transmission rate and

required reliability. When the base station receives the session initiation packet, it stores all the information, sets the timers and other parameters for each flow, and acknowledges this packet. It is important for the sensor node to wait for the ACK to ensure that the association is established. The nodes can now start transmitting data packets to the base station. In the reverse path, the base station transmits an ACK or NACK depending on the type of flow. Note that the source node will transmit packets associated with each flow independently, since the transmission characteristics may be different. Since the base station knows the rate of transmission from the source, the expected arrival time for the next packet can be found. The base station maintains a timer and sends a negative acknowledgement (NACK) if it does not receive a packet within the expected time. In event-driven flows, the base station cannot estimate arrival times of data packets. Thus, clock synchronization is not needed. Because of reliability requirement, positive acknowledgements (ACK) are used by source to know if a packet has reached the base station. STCP adopts method of explicit congestion notification with some modification. Each STCP data packet has a congestion notification bit in its header. Every sensor node maintains two thresholds in its buffer: t_{lower} and t_{higher}. When the buffer reaches t_{lower}, the congestion bit is set with a certain probability. When the buffer reaches t_{higher}, the node will set the congestion notification bit in every packet it forwards.

[52] is based transport layer of network stack. It mainly concentrates on many-to-one routing and builds trees to balance network load and avoid congestion.

In Congestion-Aware Routing Protocol for Mobile Ad Hoc Networks, they first propose a congestion-aware routing metric which employs the retransmission count weighted channel delay and buffer queuing delay, with preference of less congested high throughput links to improve channel utilization. Then, [53] propose the Congestion Aware Routing protocol for mobile ad hoc networks. CARM is an on-demand routing protocol that aims to create congestion-free routes by making use of information gathered from the MAC and physical layer. The CARM route discovery packet is similar to that in DSR [20] where every packet

carries the entire route node sequence. CARM employs the WCD metric in [20] to account for the congestion level. In addition, CARM adopts a route effective data-rate category. The combination of these two mechanisms enables CARM to ameliorate the effects of congestion in multi-rate networks. CARM uses the same route maintenance approach as that in DSR.

Reliable Event Detection and Congestion Avoidance in Wireless Sensor Networks [28] proposes source count based hierarchical medium access control (HMAC) that gives proportional access, i.e. a node carrying higher amount of traffic gets more access to the medium than others. Therefore, downstream nodes obtain higher access to the medium than the upstream nodes. This access pattern is controlled with local values and is made load adaptive to cope up with various application scenarios. To avoid congestion, before transmitting a packet each upstream node must be aware whether there is sufficient free buffer space at the downstream node. To implement this notion, we restrict an upstream node from delivering packets when its downstream node has not sufficient amount of free buffer space. This is achieved by our proposed source count based weighted round robin forwarding (WRRF). Sensing nodes must not transfer so high amount of data that can overwhelm the capacity of downstream nodes, particularly the nodes near to sink. Hence, we propose hierarchical medium access control (HMAC) that gives proportional access based on source count value, i.e. a node carrying higher amount of traffic gets more accesses than others.

Pump Slowly Fetch Quickly (PSFQ) [3] is a simple, scalable, and robust transport protocol that is customizable to meet the needs of emerging reliable data applications in sensor networks. PSFQ represents a simple approach because it makes minimum assumptions about the underlying routing infrastructure, it is scalable and energy efficient because it supports minimum signaling, thereby reducing the communication cost for data reliability, and importantly, it is robust because it is responsive to a wide range of operational error conditions found in sensor network, allowing for the successful operation of the protocol even under highly error-prone conditions. The key idea that underpins the design of PSFQ is to distribute data from a source node by pacing data at a relatively slow speed (pump slowly),

but allowing nodes that experience data loss to fetch (i.e., recover) any missing segments from their local immediate neighbors aggressively (fetch quickly). PSFQ comprises three protocol functions: message relaying (pump operation), relay-initiated error recovery (fetch operation), and selective status reporting (report operation). A user injects messages into the network and intermediate nodes buffer and relay messages with the proper schedule to achieve loose delay bounds. A relay node maintains a data cache and uses cached information to detect data loss, initiating error recovery operations if necessary. It is important for the user to obtain statistics about the dissemination status in the network as a basis for subsequent decision-making, such as the correct time to switch over to the new task in the case of programming sensors over-the-air. Therefore, it is necessary to incorporate a feedback and reporting mechanism into PSFQ that is flexible (i.e., adaptive to the environment) and scalable (i.e., minimizes the overhead). PSFQ [3] is scalable and reliable transport protocol that deals with strict data delivery guarantees rather than desired event reliability as it is done in ESRT. However, this approach involves highly specialized parameter tuning and accurate timing configuration that makes it unsuitable for many applications. Also PSFQ has several disadvantages (i)It cannot detect single packet loss since they use only NACK. (ii)It uses statically and slowly pump that result in large delay and finally, (iii)It requires more buffer as hop-by-hop mechanism is used.

"End-to-end Reliability in wireless sensor networks: Survey and Research Challenges" [55] and "A Survey of Transport Protocols for Wireless Sensor Networks" [26] funded by AT&T Labs Research are research paper which cover the challenges available in wireless sensor networks. They talk about various limiting factors that should be dealt with and provide new ideas for upcoming research.

Mitigating Congestion in Wireless Sensor Networks [56] was research done in MIT to merge multiple congestion control schemes into one protocol. Their congestion control scheme, which they call Fusion, integrates three techniques: hop-by-hop, flow control, rate limiting, and a prioritized MAC. Hop-by-hop flow control is designed to prevent nodes from

transmitting if their packets are only destined to be dropped due to insufficient space in output queues at downstream nodes. Rate limiting meters traffic being admitted into the network to prevent unfairness toward sources far from a sink. A prioritized MAC ensures that congested nodes receive prioritized access to the channel, allowing output queues to drain.

In [56] implementation, each sensor sets a congestion bit in the header of every outgoing packet. By taking advantage of the broadcast nature of the wireless medium, their implementation provides congestion feedback to all nodes in a radio neighborhood with every transmission. As a result, this implicit feedback obviates the need for explicit control messages that could use a large fraction of available bandwidth. Hop-by-hop flow control has two components: congestion detection and congestion mitigation.

A High-Throughput Path Metric for Multi-Hop Wireless Routing [57] defines a metric called ETX. We will look at it in greater detail in the later part of this thesis.

In interference-minimized multipath routing with congestion control in wireless sensor network for multimedia streaming [63], the source initiates path discovery to final destination. The three steps of path discovery for I2MR are:

1. Primary path discovery: Constructs shortest possible path from source to primary destination that minimizes intra-path interference.

2. Interference-zone marking: Marks one- and two-hop neighbors of intermediate nodes along primary path as interference-zone of path because interference range is assumed to be at most twice communication range.

3. Secondary and backup path discovery: Selects secondary and backup destinations and constructs shortest paths back to the source. Paths constructed are outside the interference zone of the primary path and minimize intra-path interference.

After successful path-discovery, the source uses the primary-secondary path-pair for load balancing. When an intermediate node along the active paths detects long term congestions,

it informs the source and the source attempts to reduce its loading rate to alleviate the congestions. The source eventually settles at the highest possible loading rate that the active paths can support.

Load Repartition for Congestion Control in Multimedia Wireless Sensor Networks with Multipath Routing [64] defines three load repartition strategies for congestion control, from mode 1 to mode 3. For the purpose of comparison, they define mode 0 which refers to the no load repartition scenario in which a source uses the same path without any congestion control concerns.

- Mode 1: The source uses all the available paths to a sink from the beginning of the transmission. The traffic is then uniformly load-balanced on these paths. In modes 2 and 3, explicit congestion notifications are used. At every intermediary node, when the reception queue occupancy is greater than a given threshold (80% of total buffer space for instance) or when the collision rate is above a given threshold, a Congestion Notification (CN) message is sent back to the sources for each path id known by the node.

- Mode 2: The source starts initially with one path. For each CN(nid, pid) message received, the source adds a new path (the first available path different from pid that is non active) until all available paths are marked as active. The load is uniformly distributed on the number of active path. It is therefore an incremental approach.

- Mode 3: The source starts initially with one path. Upon reception of a CN(nid, pid) message the source will uniformly balance the traffic of path pid on all available paths (including path pid). Therefore depending on the number of CNs received for each path, the transmission rate is not the same on all the active paths as opposed to mode 2.

In the paper, Reducing Congestion Effects in Wireless Networks by Multipath Routing [65] they propose Biased Geographical Routing (BGR), a lightweight, stateless, geographical

forwarding algorithm, as a cost-effective complement to greedy routing. BGR routes packets on curved trajectories, by forwarding packets along curves, instead of along the shortest path, towards the destination. To further mitigate congestion, [65] designed two congestion control mechanisms that leverage BGR: In-Network Packet Scatter (IPS) is a lightweight mechanism that aims to relieve transient congestion by locally splitting the traffic along multiple paths to avoid congested hotspots. End-to-End Packet Scatter (EPS) is an end-to-end mechanism that aims to alleviate longer term congestion, when IPS fails. EPS works by splitting the flow at the source, and performing independent rate control along each path in response to congestion.

There is also some research done for sensor network deployment in coal mines. The Research and Design of High Reliability Routing Protocol of Wireless Sensor Network in Coal Mine [66] address this issue. In these types of routing protocols, the main focus is on conveying the event to sink. Hence, there should be a path from sink to source which will route events reliably. It has been suggested in [75] that the rate adjustment be that of Additive Increase Multiplicative Decrease (AIMD) [75], which has the disadvantage of being more biased towards sources closer to the sink (in terms of latency), resulting in these nodes having greater sending rates. This will cause the base station to receive an even more disproportionate number of packets from nodes close by. In general, open-loop control is more appropriate for transient congestion, whereas closed-loop control is better for persistent congestion [39]. All of these schemes explained here have some drawbacks. None of the schemes perform equally well in all the scenarios. ESRT [5] allocates transmission rate to sensors such that an application-defined number of sensor readings are received at a base station, while ensuring the network is not congested. On reception of packets with congestion notification bit high, sink node regulates the reporting rate by broadcasting a high energy control signal so that it could reach to all sources. This high powered congestion control signal may disrupt some other transmissions. Also the assumption of congestion notification by the sink node is very optimistic. CODA [6] uses a combination of the present

and past channel loading conditions and the current buffer occupancy, to infer accurate detection of congestion at each receiver with low cost. As long as a node detects congestion, it sends backpressure messages to upstream nodes for controlling sending rate hop-by-hop. It is also capable of asserting congestion control over multiple sources from a single sink in the event of persistent congestion. Even though it overcomes some of the limitations of ESRT [5], it doesnt consider the event fairness and packet reliability at all. PSFQ [3] is scalable and reliable transport protocol that deals with strict data delivery guarantees rather than desired event reliability as it is done in ESRT. However, this approach involves highly specialized parameter tuning and accurate timing configuration that makes it unsuitable for many applications. Also PSFQ has several disadvantages

- It cannot detect single packet loss since they use only NACK,
- It uses statically and slowly pump that result in large delay and finally
- It requires more buffer as hop-by-hop mechanism is used.

As defined in Many-to-One Routing, event fairness is achieved when an equal number of packets are received from each node. In this proposal, individual nodes divide its effective available bandwidth equally amongst all upstream nodes. This, in turn ensures fairness. It several disadvantages including:

- It provides no reliability guarantee.
- The effective throughput may decrease due to implementation of ACK in transport layer.

2.3 Hardware Support for sensor networks

The latest series of Telos B motes [34], the ZigBee motes [35] with improved abilities, or PC104 [36] may be used for applications in WSNs which require intensive memory and

bandwidth. Telos is an ultra low power wireless sensor module ("mote") for research and experimentation [30]. The latest in a line of motes developed by UC Berkeley to enable wireless sensor network (WSN) research is called Telos. It is a new mote design built from scratch based on experiences with previous mote generations. Its new design consists of three major goals to enable experimentation: minimal power consumption, easy to use, and increased software and hardware robustness. ZigBee [35] is an important standardized approach to the control of sensor and actuator networks in home/building/industrial automation applications that use low rate wireless PAN network technology. ZigBee stack includes Network and Application Layer proposed by ZigBee Alliance and IEEE 802.15.4 MAC and Physical Layer for both mesh and tree based communication. PC104 [36] is a popular standardized form-factor for small computing modules typically used in industrial control systems or vehicles. It gets the name from the popular desktop personal computers initially designed by IBM called the PC, and from the number of pins used to connect the cards together (104). PC104 cards are much smaller than ISA-bus cards found in PC's and stack together which eliminates the need for a motherboard, backplane, and/or card cage. Power requirements and signal drive are reduced to meet the needs of an embedded system. Because PC104 is essentially a PC with a different form factor, most of the program development tools used for PC's can be used for a PC104 system. This reduces the cost of purchasing new tools and also greatly reduces the learning curve for programmers and hardware designers. Most of the sensors used in research for audio/video streaming are found to use embedded microprocessors which have higher computing abilities [34].

Chapter 3

Objective and Motivation

Accurate and efficient congestion detection plays an important role in congestion control of sensor networks. When congestion is detected in the network, we need to take some measures to handle this congestion and reroute, mitigate traffic so as to maintain network condition. Congestion taking place at a single node may diffuse to the whole network and degrade its performance drastically.

3.1 Motivation

3.1.1 The Congestion Problem

In computer networks, mismatch of incoming and outgoing data rates results in congestion. For wired networks like INTERNET, there are mixed links with different bandwidths. The node with the lowest bandwidth along a path from the source to the destination is called the bottleneck. Usually, congestion occurs in the bottleneck since it receives more data than it is capable of sending out. In this situation, packets will be queued and sometimes get dropped. As a consequence, response time will increase and throughput will also degrade.

Figure 3.1 illustrates network performance as a function of the load. When the load is light, throughput is linearly proportional to the load and response time is almost unchanged. After the load reaches the network capacity (the knee point), throughput won't increase much with the load. Instead, packets will be queued and the response time will become longer in this period. The throughput may suddenly drop if packets get discarded due to buffer overflow, which is called the cliff point as shown in Figure 3.1. Congestion can be realized in many ways, but in simple terms one may say that, if, for any time interval, the total sum of

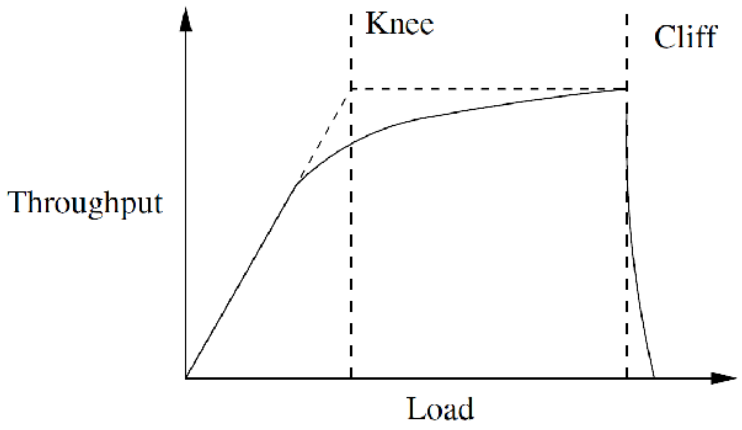

Figure 3.1: Effect of congestion on throughput

demands on a source is more than its available capacity, the source is said to be congested for that interval. Mathematically speaking:

$$\sum Demand > \text{Available Resources} \tag{3.1}$$

Congestion in wireless networks is different from that of wired networks. Due to the memory restrictions of the sensor nodes and limited capacity of shared wireless medium, network congestion may be experienced during the network operation. In wireless networks congestion happens due to contention caused by concurrent transmissions, buffer overflows and dynamically time varying wireless channel condition [24][25][26].

3.1.2 Effects of Congestion

As WSN is a multi-hop network, congestion taking place at a single node may diffuse to the whole network and degrade its performance drastically [27]. Congestion causes many folds of drawbacks:

1. Increases energy dissipation rates of sensor nodes,

2. Causes a lot of packet loss, which in turn diminish the network throughput and

3. Hinders fair event detections and reliable data transmission.

4. Large queuing delays are experienced as the packet arrival rate nears the link capacity.

5. The sender must perform retransmissions in order to compensate for dropped packets due to buffer overflow.

6. Unneeded retransmissions by the sender in the face of large delays may cause a router to use its link bandwidth to forward unneeded copies of a packet.

7. When a packet is dropped along the path to destination, the transmission capacity that was used at each of the upstream links to forward that packet to the point that it is dropped end up having been wasted.

Therefore, congestion control or congestion avoidance has become very crucial to achieve reliable event detection for the practical realization of WSN based envisioned applications. We find that one of the key reasons of congestion in WSN is allowing sensing nodes to transfer as many packets as they can. This is due to the use of opportunistic media access control. The high amount of data transferred by sensing nodes can overwhelm the capacity of downstream nodes, particularly the nodes near to sink [28]. Congestion due to buffer overflow is also not insignificant [29] [30].

3.2 Objective

Our objective in this thesis is to provide efficient data rate throughout the network operation, handling varying traffic rates. Congestion is main factor in degrading network throughput. To satisfy this objective we propose techniques to detect congestion, and take action to maintain a constant traffic rate. Our aim is to try and maintain network in an ideal state in which it will deliver maximum packets allowed by network bandwidth consistently. Initial step towards achieving this goal is to categorize packet loss and congestion types in wireless networks.

3.2.1 Packet Loss Categorization

In this section we list the various ways in which packets can be lost in a multi-hop wireless network. According to literature [39] packet losses are classified into four types:

Type I

If the transmitting mote is far from the receiving mote, the signal will attenuate significantly by the time it reaches the receiver. The signal attenuation is difficult to model since the radio signal strength is not uniform at the same distance from the mote in all directions.

Type II

If more than one sensor mote in the sensor network is transmitting simultaneously, interference will occur at the listening mote that is within range of the transmitting motes. In general, motes can be too far away to be considered neighbors, but still be close enough to interfere with reception. This type of interference is difficult to model also due to the same reason given for Type I loss.

Type III

The third cause of loss is due to self-interference, that is, a mote's transmission interferes with itself at the receiver, due to multi-path-effects, Rayleigh fading etc.

Type VI

The type of loss occurs when a packet is successfully received by a mote but has to be dropped due to queue overflow.

Of the causes listed, types I and III are dependent on the exact location and environment in which the motes are deployed, as well as the radio technology implemented in the motes. For instance, sensor motes placed less than 3 feet apart on a wall may not be able to hear each other due to reactions of the wall. We therefore cannot assume to have any control

over these losses. Type IV losses are due to congestion within the network. Clearly, the correct implementation of congestion control will minimize this type of losses. In this thesis we focus on type II and IV losses.

3.2.2 Types Of Congestion

We group congestion into two types according the reasoning of the problem appearance:

Type A

In a particular area, many motes within range of one another attempt to transmit simultaneously, resulting in type II losses and thereby reducing throughput of all motes in the area. This definition of congestion has been used in [25] as well. We note that explicit local synchronization among neighboring motes can reduce type II loss in this regard, but cannot eliminate it completely because non-neighboring motes can still interfere with transmission.

Type B

Within a particular mote, the queue, or buffer used to hold packets to be transmitted, overflows. This is the conventional definition of congestion, widely used in wired networks. This is also the cause of type IV losses. It is possible to have both types of congestion occurring at the same time. Also there is different approach for naming congestion types. According to the place in the network where congestion happens and the kind of sensor reporting traffic, three type of congestion are defined in [6] & [44].

3.2.3 The Ideal Case

After the load reaches the network capacity (the knee point), the number of packets delivered will not show a significant increase with the increase in offered load. Congestion control is necessary in avoiding congestion and/or improving performance after congestion.

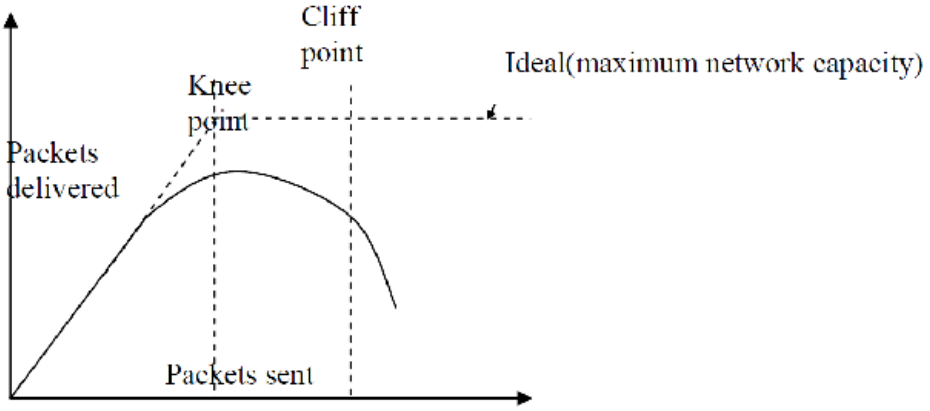

Figure 3.2: Network condition with congestion control

It aims to make the network operate around the knee point. Hence it tries to keep the throughput to maximum without affecting reliability. The ideal case is shown in Figure 3.2.

Congestion control is necessary in avoiding congestion and/or improving performance after congestion. It aims to make the network operate around the knee point in the above figure. These techniques are explained in detail in next sections.

3.3 Problem Statement

Assumptions and Goals

We begin by listing the assumptions we made about the networks.

- All nodes in the network are stationary.
- Each node is equipped with one 802.11b radio.

For this thesis we consider Directed Diffusion routing protocol. It manages paths in a data-centric way. A sink node publicizes its interest in some events. Such an interest will be disseminated within the network and received by nodes that have data on those events. During this process, gradients that record two-way information along all possible paths are established. Afterwards, a mechanism called path reinforcement is used to select high-quality

routes. Directed Diffusion also provides two sets of application programming interface (API). One set is network routing API, which enables upper-layer entities in sources and sinks to communicate through the network. This API takes the form of publish/subscribe paradigm. A sink node can subscribe to interesting events. Also, sources will publish their sensed events. The Directed Diffusion platform takes care of the underlying implementation. To allow processing in intermediate nodes, Directed Diffusion also offers another API called filters. Application-specific operations can be implemented as filters to manipulate data packets as they pass through the network. Filters are provided through an interface to specify interest in particular events. For each filter, a priority is associated. Although directed diffusion is a widely used in sensor networks, it has several weaknesses. It only considers delay as the routing metric to select best path. As a result, high throughput may not be achieved. Its failure to consider deadline during the route selection lowers the percentage of packets which can meet the deadline of real-time traffic. Its reliance on flooding gives rise to differences in interference levels during the exploratory data phase and the actually data transmission phase, which makes the route decision even more inaccurate. It does not consider path interference which worsens it performance during high rate data transmission. We propose a routing protocol to augment and extend directed diffusion to handle these issues. There are two phases in our protocol. The first phase consists of setting up efficient paths between the source - sink pair. The second phase consists of using these paths to efficiently transfer data. Data transmission considers multiple factors like the load on node, priority of applications running on those nodes, and congestion level on those nodes.

First, we propose an algorithm which considers throughput, delay and intra-flow interference metric for a path. ETX (Expected Transmission Count) is used as the metric for measuring link quality. We improve on QoS-based routing by limiting interference in lossy links. We use an improved method for computing aggregate ETX for a path that increases end-to-end throughput. We compute multiple paths between source-sink pair. We choose best two paths based on the aggregate ETX metric and delay. The main goal of our protocol

design is to use these routes to achieve high throughput and less delay. We monitor the data transmission over a period of time. We measure maximum queue length over the active path, we measure the rate of increase in queue length, and in addition we keep track of ETX which can give us an idea of interference over links on active path. We have setup inter-node queues, and intra node queues to handle data flowing through nodes. We maintain a threshold value for all the queues. We check the current queue state after an interval of δ. Depending on the increase in queue length we estimate if the queue might be over the threshold value in time frame δ. We also measure the ETX value to see if it shows sign of interference. If these metrics show sign of congestion on the active route, then we signal the source to switch data transfer to an alternate path. If switching of path does not help to control congestion, we use multiple paths to cope with high data rates. As a last option, if even this does not mitigate congestion, then we limit source sending rate.

3.4 Proposed Techniques

As mentioned earlier Directed diffusion path metrics are inefficient. Our first problem is to find paths which provide good throughput and high quality of service in the long run. Also, in adhoc networks, finding one path from source to sink is inefficient. The topology and node positions might change, some nodes might go down, and some of the links might fail due to environmental conditions. So having multiple paths, which can act as backup in case of failure is required. To overcome these main problems, we use the routing algorithm which considers metrics of ETX as well as delay. Using a combination of these two metrics, we try to find routes which have low interference and can support higher throughput. ETX is the measure of Signal to Noise Ratio (SNR) over a wireless link. If the value of ETX is lower, that means that the path has low error rate, interference, and hence can support higher throughput. In addition to that we also account for delay, which will help us in selecting shortest of available paths, which has high capacity.

We use this path to transmit data from source to sink. In wireless medium the data rate is affected due to lot of factors, which keep changing over time. This means that a path that we have selected during route discovery phase might degrade and provide lower throughput. This results in packet drops and packet retransmission. All these also affect the congestion in that region. In our proposed algorithm, we keep track of ETX metric during data transmission phase. If the value of ETX starts increasing, it means that the link quality is degrading. However this parameter alone is not a very reliable indicator. ETX gives us the link state - meaning how the link is performing at a given instant. In addition to the link state we also have to get a measure of how node is performing during that instant. It is very easy to misinterpret these parameters in wireless ad hoc network. So to get a more accurate estimate of congestion we consider queue length in making the decision.

Queue length is another parameter which is used to get an estimate of congestion. We use a congestion detection mechanism based on local buffer level monitoring in sensor nodes. Any sensor node whose packet buffer overflows due to excessive incoming packets is said to be congested and it informs the sink of the same. We use the technique of piggy-backing to infrom the sink of current buffer level and channel quality. Sink keeps monitoring the buffer levels, and it keeps track of average increase of the queue length. If sink feels that the average increase might cause the queue length to go over threshold value, then it sends a message to source to send data using alternate path. In addition to switching paths, we make use of priority queues which will give preference to higher priority applications. Each application will have a priority and deadline associated with it. In some applications like video or voice transmission, we have to stream data real time. For such applications, jitter control is more important than reliability. In other applications where we transmit highly important data, we have to achieve maximum possible reliability and throughput. We have to select the priority and deadline in such a way that it satisfies the application requirements. Deadlines will buy more waiting time for the packets in buffer before getting dropped. This will in turn make sure that packet drop rate for such application is as low as possible. At the

same time priority of the application will help it get a preference in queues. It is left to the user to determine the application priority and deadline based on the specific requirements.

Chapter 4

Design of Congestion Control Mechanism

Congestion Control Component Mechanisms are composed from 3 sub-mechanisms:

1. Congestion Detection

2. Congestion Feedback and

3. Control Mechanism.

Congestion Detection is the mechanism used to safely detect if the problem has occurred or is going to happen. The Feedback Mechanism, with which the sensor node or the sink gives a feedback to the network to take some action according to the problem and Control Scheme which is the final action taken.

4.1 Congestion Detection

Accurate and efficient congestion detection plays an important role in congestion control of sensor networks. To detect congestion, the level of congestion should be quantified to provide a fine-grained congestion control. We define two types of congestion levels in sensor networks as follows:

- Per-node congestion level

- Per-flow congestion level

The per-node congestion depicts the local congestion level each individual node perceives. To measure the per-node congestion level, each node can investigate the statistics

on several metrics such as queue length, packet drop rate, channel loading, etc. For example, ESRT [5] uses buffer utilization alone. However, if the wireless channel is unreliable either by the unreliable MAC protocol (e.g. when the maximum number of retransmissions is small) or by the unstable channel condition, the queue length or packet drop rate cannot be a metric for congestion detection. This is because even though a node's incoming traffic volume exceeds the outgoing channel capacity, the queue length remains small since packets quickly leave the queue due to collision and no subsequent retransmission after the collision. Therefore, in CODA [6], each node measures its congestion level based on the perceived channel loading as well as its queue length. In general, the per-node congestion level measurement function f() returns a positive real number indicating the congestion level based on the metrics of $m_1, m_2, ..., m_n$ as follows.

$$L_i = f(m_1, m_2, ..., m_n), \text{ where } L_i \text{ indicates the node i's congestion level.} \tag{4.1}$$

The per-flow congestion level can be calculated in a distributed manner by each node of a flow. The source records its current per-node congestion level in the header of a data packet destined for the sink. Each subsequent node compares its per-node congestion level with the per-flow congestion level recorded in the header of the data packet forwarded from the previous hop. If its per-node congestion level is greater than the per-flow congestion level, then the node updates the per-flow congestion level in the header of the data packet with its per-node congestion level and forwards it to the next hop. If its per-node congestion is less than the per-flow congestion level, the node simply forwards the packet without modifying the per-flow congestion level. Therefore, the highest per-node congestion level is passed downstream towards the sink. The sink may enforce an end-to-end congestion control based on the per-flow congestion level. This approach could be used more efficiently in small wireless sensor networks.

4.1.1 Sensor Node Initiated

Two main techniques for detecting congestion by a sensor node are explained below. It determines the local congestion levels and it performs better under high load traffic scenarios = 50% of channel load, but it performs as well in the lower traffic load region. In this category the following congestion indicators are used:

Buffer Occupancy

The simplest method is to compare the instantaneous buffer occupancy against some threshold value. If the threshold is exceeded the congestion state is diagnosed. This kind of detection is used from [75] [6] [76] [77] [55] [78] [7] [79] approaches. An improved method, like that used for [5], takes the growth trend into account and eliminates the possibility of the threshold that make up for a large fraction of the buffer size, to detect congestion state too late. With this method the buffer size is regularly sampled and congestion is diagnosed when the instantaneous buffer level is above some threshold and additionally the buffer size has grown in the immediate past. As shown from [6] buffer occupancy alone is not a reliable congestion indicator because packets can be lost in the channel due to collisions or hidden terminal situations and have no chance to reach a buffer. Only the situations of a full buffer and an empty buffer are reasonable indicators. This is because without link ARQ, the clearing of the queue does not mean that congestion is alleviated since packets that leave the queue might fail to reach the next hop as a result of collisions. CSMA does not guarantee collision-free transmissions among neighboring nodes because of the detection delay. However the buffer occupancy seems not to provide an accurate indication of congestion even when the link ARQ is enabled. Following much of the prior work on congestion control, [68] uses an exponentially weighted moving average of the instantaneous queue length as a measure of congestion:

$$AVG_q = (1 - w_q) * AVG_q + w_q * inst_q \tag{4.2}$$

where AVG_q = average queue length of node

w_q = weighted queue length of node

$inst_q$= queue length of node at the current instant

The average queue length is updated whenever a packet is inserted into the queue. Thus, if AVG_q exceeds a certain upper threshold U, the node is said to be congested. The node remains in a congested state until AVG_q falls below a lower threshold L. In practice, a single threshold is too coarse-grained to effectively react to congestion.

Channel Utilization

The goal of channel sampling is to obtain an estimate of the current channel utilization U. In wireless networks, a community of nodes shares a single transmission medium. To avoid collision and better utilize the bandwidth, some kind of medium access control (MAC) protocol is needed. Carrier sensing multiple access (CSMA) is a random access protocol, which allows users to transmit data in a none predetermined way. CSMA schemes require a user to be sure the medium is idle before the transmission. This is called carrier sensing. If the medium is busy, the user has to back-off for a random period and then re-sense. The random period is to minimize collision since other users may also want to take the medium at the same time. Once the channel is idle, the user can start transmission. Figure 4.1 illustrates the mechanism of RTS/CTS. Node N1 sends a RTS frame to node N2 before the real data transmission. Node N0 also receives the RTS and is blocked by it. Upon receiving the RTS, node N2 broadcasts the CTS frame to its neighbors. Thus, node N3 is also blocked. Node N1 starts transmitting data once receiving the CTS frame from node N2. The RTS/CTS mechanism is to deal with hidden terminal problems. In Figure 4.1, node

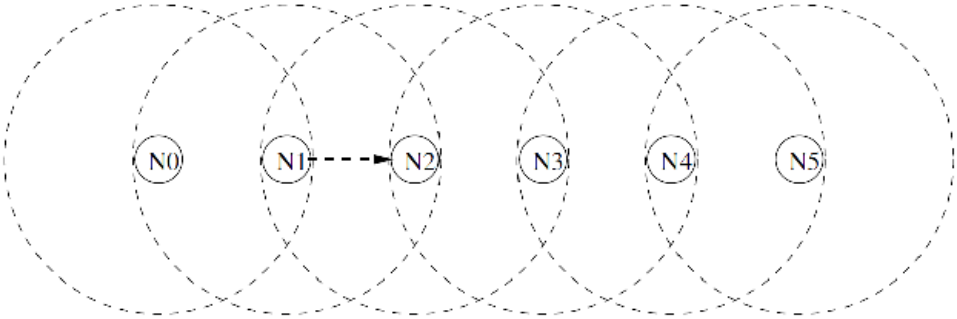

Figure 4.1: Interference cause in wireless sensor networks

N5 is a hidden node of the transmission from N1 to N2, since N5 is beyond the interference range of N2 (two hops). Node N5 cannot sense the data flow from N1 to N2 and will think the medium is idle. If there is no RTS/CTS, node N5 will directly start sending data packets to N4. In this case, the ACK frames from node N4 will be very likely to collide with the data received by N2. With the use of RTS/CTS, node N5 won't get the CTS from N4 and cause interference to N1 and N2 since N4 can detect the flow between N1 and N2.

In sensor networks, CSMA schemes are practically used, for example, IEEE 802.11 [49] and SMAC [86]. For CSMA, maximum utilization is $U_{max} < 1$, beyond which the rate of collisions increases and throughput decreases. Congestion is diagnosed when the channel utilization is within some neighborhood U_{max}.

In CSMA networks, it is straightforward for sensors to listen to the channel, trace the channel busy time and calculate the local channel loading conditions. Channel loading gives accurate information about how busy the surrounding network is, but it is inherently a local mitigation mechanism. It has limited effect, for example, in detecting large-scale congestion caused by data impulses from sparsely located sources that generate high-rate traffic. Listening to the channel consumes a significant portion of energy in a node. Therefore performing this operation all of the time is not practical in sensor networks. So sampling schemes activates local channel monitoring only [when queue is not empty] at the appropriate time to minimize the energy cost while forming an accurate estimate of conditions. This approach is also used in [55]. We use a similar approach to get the load condition of entire

path. We term this metric as ETX. Our scheme calculates ETX for a set of three consecutive links and sends it to sink. We use this metric at sink to get and estimation of channel load and interference. Some other schemes are described below:

1. Measuring the received signal strength RSSI on messages, to determine how busy the channel is from interference level. This is referenced as a possible congestion detection method in [80].

2. By counting the continuous occupied queues neighborhood queue) that may appear due to deadlock effects [81] because of CSMA MAC.

3. A combination of methods a and b seems to work very well. This kind of congestion detection is used in [6] [80] [9].The above methods showed to work very well on detecting congestion. The following are also proposed but seem to have mixed results.

4.2 Congestion Handling, Feedback and Control

This is a 2-step mechanism:

- Congestion Feedback
- Congestion control (action)

4.2.1 Congestion feedback

Based on congestion control algorithms classification done in [84] and according to control theory point of view, we divided congestion control solutions into two groups.

- Open Loop
- Closed Loop

Open Loop

These are the solutions in which control decisions of algorithms do not depend on any sort of feedback information from the congested spots in the network, and so they do not monitor the state of the network dynamically. The congestion control algorithm serves as a controller or control actuator, purely based on its own knowledge of local sensor node. They can be further classified as explicit control algorithms at the source or intermediate sensor nodes of the network. This is used by [9] where the congested node in the visibility of a Virtual Sink, redirect the traffic accordingly when working in the node initiated congestion detection mode. Authors of [82] use open loop control as well.

Closed Loop

In contrast, closed loop solutions are based on the concept of a feedback loop. This approach has three parts when applied to congestion control:

- Monitor the network to detect when and where congestion occurs.
- Pass this information to places where action can be taken.
- Adjust network operation to correct the problem.

The feedback can be either global or local; global means that the feedback information goes all the way from destination to source (end to end) where local means the feedback information comes only from immediate neighbors. With the provision of feedback these algorithms are able to monitor the network performance dynamically. The feedback may be explicit (sent explicitly as different or piggybacked messages) or implicit in other case. The explicit feedbacks can be further classified as persistent (if available all times) and responsive feedback (if only triggered under certain conditions). Furthermore many explicit closed loop algorithms are divided into 2 stages - anticipatory and reactive congestion control. In the anticipatory stage algorithms tend to achieve congestion avoidance whereas reactive strategy is a congestion recovery scheme that responds to conditions of network congestion.

In this thesis we use closed loop control mechanism. We monitor the network state on a regular basis. Each node will keep track of its queue (buffer) level. Also, it measures the SNR and calculates ETX value which will be conveyed to sink periodically. We keep track of these parameters, and get an estimate of current network state. If the parameters indicate that a network is being congested or is already congested, then we take a corrective action. This corrective action is inform source of congestion on current active path. Then depending on the state of source, it takes preventive action. The feedback mechanism is explained in next section.

4.2.2 Feedback Mechanisms

Congestion feedback mechanisms can be categorized in several ways. In one way, they are classified into explicit feedback and implicit feedback.

Explicit Feedback

Explicit feedback means that feedback is sent to the sender in an explicit form, like a dedicated bit, or a control packet.

- Explicit Periodic Feedback:

  This approach is used in [7] where the upstream nodes of a flow periodically notify their congestion level to downstream (toward the sink) nodes by embedding their perceived congestion level into the header of data packets. We make use of this approach to keep sink updated with current status of nodes on the active path. We include the queue length bit in packet header which will hold the maximum length of maximum queue on the path. There is also a count field which will be incremented by each node on path, whose queue length is more than the threshold value. This is an indicator of the number of nodes which are getting congested. We can use these metrics to take congestion control measures.

- Explicit Responsive Feedback:

  Explicit responsive feedback can be achieved using extra control messages. This approach is rarely used because extra packets consume sensors limited energy resources. This technique is followed by [6] where backpressure messages are used in case of congestion detection. We use this approach in even of congestion. To address the issue of congestion, we send the response packet to source using an alternate path.

Implicit Feedback

In implicit feedback, feedback information doesn't occupy any dedicated bits. It is realized by piggyback. The well-known example of implicit feedback is TCP, where 3-Acknowledgments imply congestion. From the aspect of information carried by feedback, they can be categorized into binary or non-binary feedback. Binary feedback can only tell if there is congestion or not. In contrast, non-binary feedback carries more information, which can indicate the congestion level. TCP is a binary feedback mechanism.

- Piggybacking (as implicit feedback): Different forms of piggybacking are used in research papers. In this case a congestion bit is inserted in the packet (message) header. Some of the techniques are listed below:
  - In inverse going traffic (sink to source). In [25] something alike happens when new transmission rate is included in the packet forwarded from root(sink) toward the leafs of a tree based sensor network
  - Taking advantage of synchronous link layer NACKS to inform sender if the receivers buffer exceeds threshold as happens in [77].
  - Overhearing outgoing packets from neighbor downstream nodes as happen in [6] [55][68].

– In RTS/CTS. In [79] proposal the authors suggest that congestion state is inserted in the RTS/CTS signals. The congestion information distributed by the MAC-signals may either include only those cost estimates (path, queue length) belonging to the final destination of the currently transmitted packet, or it may include much more than this, namely the cost estimates to every single node of the network.

In this thesis we keep track of ETX which indicates channel utilization. Also, we keep track of buffer occupancy which is an estimate of queue length - load on the node. We keep track of these two parameters, and measure the rise in values between two feedback's. By measuring these parameters we get an estimate of current network state. If the parameters indicate that the network is being congested or is already congested, then we take a corrective action. This corrective action is inform source of congestion on current active path. This feedback is an explicit responsive feedback. We have introduced two messages : SWITCH_PATH_MESSAGE and MULTI_PATH_MESSAGE, to implement the feedback mechanism. In our case the cost of the feedback is this one message. Then depending on the state of source, it takes preventive action. If the source receives a SWITCH_PATH_MESSAGE, it will switch the data transfer to an alternate path, making it active path. This scheme is possible because we change the routing algorithm for directed diffusion to find two disjoint paths for source - sink pair. Switching of paths will help us transmit data around the congested region. This has a twofold advantage. Firstly we will be using a path which does not have congestion - meaning better throughput. Secondly, we will be avoiding the congestion region, thus stopping the traffic from worsening the congestion and degrading the network performance. If the source has already switched path before and congestion has not resolved, then sink sends a MULTI_PATH_MESSAGE, which will indicate source to send data using multiple paths. This will distribute the network traffic, thus mitigating congestion. We explain the routing and congestion control in detail in next section.

Chapter 5

Implementation Details

5.1 Directed Diffusion

Directed diffusion uses a publish/subscribe communication model in which a sink node floods interests as requests for a named data. Directed diffusion consists of several elements: interests, data messages, gradients, and reinforcements [86]. An interest message is a query or an interrogation which specifies what a user wants. Each interest contains a description of a sensing task that is supported by a sensor network for acquiring data. Typically, data in sensor networks is the collected or processed information of a physical phenomenon. Such data can be an event which is a short description of the sensed phenomenon. In directed diffusion, data is named using attribute-value pairs. A sensing task (or a subtask thereof) is disseminated throughout the sensor network as an interest for named data. As the interest is propagated through the network, each intermediate node sets up a gradient with its neighbors and enables data that match the interest to be pulled towards the sink. Sensor nodes with data that match the interest will forward exploratory data propagated by intermediate nodes through established gradients to the sink. This is shown in Figure 5.1.

This dissemination sets up gradients within the network designed to draw events (i.e., data matching the interest). Specifically, a gradient is direction state created in each node that receives an interest. The gradient direction is set toward the neighboring node from which the interest is received. Events start flowing towards the originators of interests along multiple gradient paths. The sensor network reinforces one, or a small number of these paths. This process is depicted in the Figure 5.2.

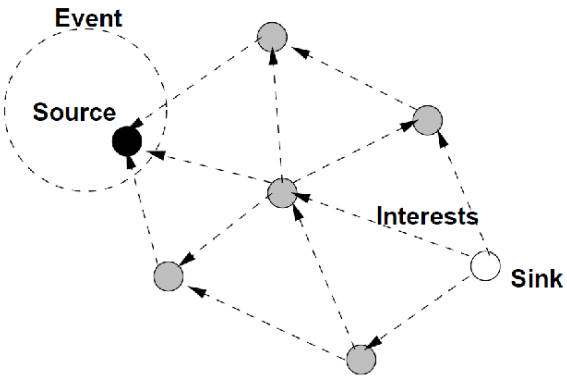

Figure 5.1: Interest Propogation in Directed Diffusion [12]

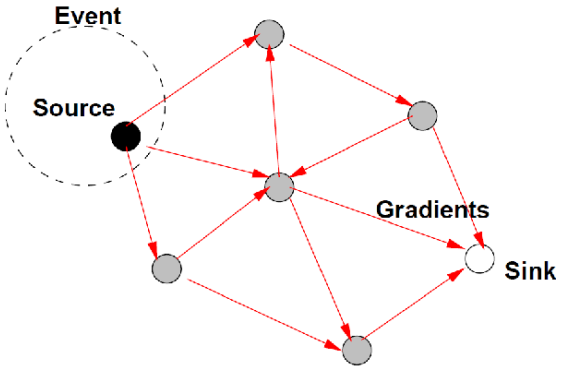

Figure 5.2: Gradient Establishment in Directed Diffusion [12]

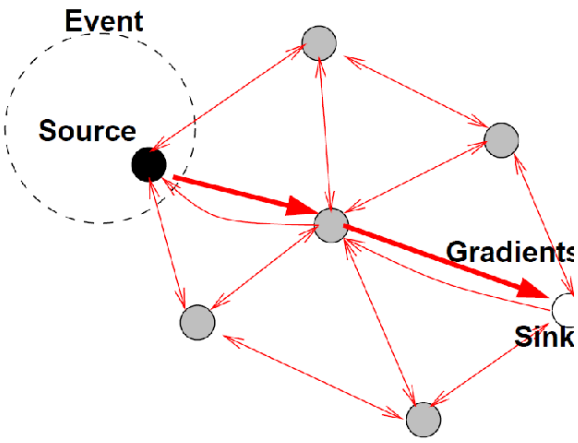

Figure 5.3: Reinforcement packets in Directed Diffusion [12]

The sink initiates a reinforcement message to the node that first forwarded the new data to it. Other nodes use the same rule to reinforce the upstream neighbor. This is shown in Figure 5.3.

The source node starts to send data through the reinforced path after it receives the reinforcement. Based on the above rule, basic diffusion generally selects route with the lowest delay. In the past few years, researchers have proposed a variety of single or hybrid metrics with the purpose of improving the performance, including throughput, delay, jitter, and deadline hit ratio, of wireless networks. Single metrics include RTT [59], PktPair [59], ETX [56], WCETT [60], and EDR [61]. Most of them can be implemented in directed diffusion. In order to consider both throughput and delay and take into account intra-path interference, a hybrid metric was proposed in [87]. Our routing algorithm highly relies on this metric to find efficient routes.

5.2 Changes to Path metric

In this section, we present a routing algorithm which finds routes which try to reduce channel interference, maximizes the throughput and minimizes the delay over lossy links in multi-hop WSNs [87]. In our current protocol, none of the specific multimedia QoS

requirements guide the routing decision process because we assume that every node has the same bandwidth and lack of global information prevents us from getting the interference area of a certain node. We only try to maximize throughput and minimize delay in order to meet the requirements. We assume that packets which are sent earlier by the source are added into the queue at each node earlier, which means they are processed earlier. Admittedly, packet scheduling helps achieve the hard deadline requirement. For sensor nodes, however, scheduling tasks may drain their energy. The ETX of a link is the predicted number of data transmissions required to send a packet over that link [56].

$$ETX = \frac{1}{d_f * d_r} \tag{5.1}$$

The forward delivery ratio, d_f , is the probability that a data packet successfully arrives at the recipient; the reverse delivery ratio, d_r, is the probability that the ACK packet is successfully received.

- Definition of ETX_p: The path ETX is the maximum of the sum of the ETX_{sum} of any three successive hops in a route. This computes the amount of bottleneck. N is the number of hops. ETX_j is the ETX value of the j^{th} hop. The number of bottleneck links may vary according to the network density.

$$ETX_p = \max_{i=0}^{N-3}(\sum_{j=i}^{i+2} ETX_j) \tag{5.2}$$

- Definition of $Delay_p$: The end-to-end delay of a packet in a network is the time it takes the packet to reach the sink from the time it leaves the source.

- Definition of $Cost_p$: The path cost is the combined metric of a route.

$$Cost_p = ETX_p * Delay_p \tag{5.3}$$

- Definition of decision interval (INTERVAL): We start an adaptive timer at each node (except the source) when the node receives the first exploratory packet. After an "INTERVAL" period, the timer expires and it selects the route with the lowest $Cost_p$.
- $EXPLORE_{delay}$: is a constant with the basic timeout value. ETX_i is the ETX value of the upstream link on which the first exploratory data arrive. Different "INTERVAL" may be computed at different nodes based on the following formula.

$$INTERVAL = ETX_i * EXPLORE_{delay} \tag{5.4}$$

5.3 Computing Path Metric

Our path metric is called $Cost_p$ which conforms to the following three goals.

1. The protocol should take into account both end-to-end delay and ETX of a route. Since the 802.11b MAC implements an ARQ (retransmission) mechanism, a links ETX can be computed. Delay can be calculated by using time-stamps from each packet.
2. The path metric should be independent of number of hops in a route, as number of hops is not an accurate measure of congestion or path quality.
3. The method for computing the path ETX must consider intra-flow interference.

First, it takes both end-to-end delay ($Delay_p$) and ETX of a route (ETX_p) into account. By adjusting the values of α and β , we are able to set different weights to each factor. If throughput is more important for an application, α should be greater than β and vice versa.

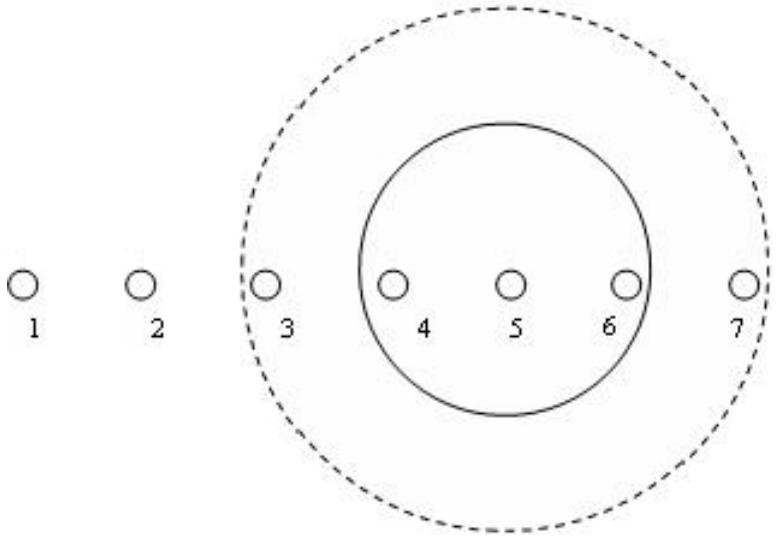

Figure 5.4: Transmission range and interference range for a chain of nodes. The solid line circle is a nodes transmission range while the dotted line circle shows the interference range. Nodes within 3 hops interfere with each other. [87]

The way we compute ETX for a path is based on the theoretical analysis and experimental demonstration in [63]. [87] employed the Packet Decoupling property to conclude that the first packet has outpaced the rest of the packets when the fourth packet is to be injected. [71] examined the capacity of a chain of nodes and they found that an ideal MAC protocol could achieve chain utilization as high as 1/3. The example below illustrates this principle for the node placement in Figure 5.4

We modify route selection process of directed diffusion by

1. Using $Cost_p$ as the metric instead of pure delay;

2. Reinforcing multiple links at the sink to obtain disjoint paths from the source.

These modifications will maximize the throughput and minimize the delay over lossy links in multi-hop WMSNs. We are not using specific multimedia QoS requirements such as bandwidth to guide the routing decision process or prioritized packet scheduling to avoid fast depletion of energy in sensor nodes. However we do consider the playout deadline since end-to-end delay constraint is one of the most important QoS requirement because data packets

Hop number n	ETX (0)	ETX (1)	ETX (2)	ETXp	T1

Figure 5.5: Extra fields added in packet header to compute ETX.

Last hop	ETX	Local timestamp	Costp	Cumulative SNR

Figure 5.6: Routing Table modified to include ETX metric.

arriving later than the deadline are simply useless, making them equivalent to data dropped. We discard paths which fail to meet the playout deadline. Throughput is influenced by ETX, which is estimated by the cumulative SNR (Signal to Noise Ratio). The use of historical SNRs offers more stable and accurate estimation of ETX. When interests are first flooded, a timestamp t_0 is inserted in the interest packets. Exploratory data packets are flooded after the source receives interests from the sink. At each intermediate node which received an exploratory data packet, SNR is read from the packet. Cross-layer design is probably needed here since SNR is a MAC layer metric. ETX information of the previous three upstream links, calculated from SNR, is inserted in the packet header in the ETX (n%3) fields [87]. This is shown in Figure 5.5

The ETX information of each link could be collected while the exploratory data are flooded. $Cost_P$ of each sub-path is kept at the intermediate nodes local table in ascending order. The format of the local table is shown in Figure 5.6

Only the one with the lowest $Cost_p$ is forwarded to the next hop. Another timestamp at the sink t_1 is recorded when the first exploratory packet reaches the sink. t_i is the timestamp for the ith exploratory packet to reach the sink. We only consider packets whose timestamp t_i satisfies the constraint

$$t_i - t_0 <= DL \tag{5.5}$$

where DL is set to be slightly less than the real playout deadline to take into account the time for flooding disjoint paths. If no path can satisfy the delay constraint, the sink adjusts the metric $Cost_P$ by giving more weight to β (for delay) and piggybacks the new value in new INTEREST messages. This routing algorithm will help us create a high throughput, low delay path. But for our purpose we need multiple paths which will act as back up paths in event of congestion.

5.4 Multiple path reinforcement

The sink stops putting exploratory data packets in the candidate pool when it received one that cannot meet the deadline or when the multi-path timer expires. It then sorts the candidate paths in ascending order of $Cost_P$ and selects the first ρ paths to reinforce, where $\rho > \Delta$ and Δ is the number of paths needed at the source. We need to find more than the required number of paths because some candidate paths may not be reinforced i disjoint nodes cannot be found or the delay exceeds the playout deadline. If two nodes try to reinforce a link that converge in the same node, the first one to reinforce would win.

In figure 5.7, A reinforces C first. Then, B tries to reinforce C and C will drop the packet because C regards it as an old message. C will then send B a NEG_RESPONSE so that B could delete the entry of C in its local candidate table and selects the next candidate, e.g. D. In addition, delay constraints must be satisfied by computing the difference between two local timestamps. We can guarantee that if we choose the node within the delay constraint in each step, the final route can also meet the delay constraint, no matter how many times we have to choose the next-best node. This technique not only guarantees disjoint nodes

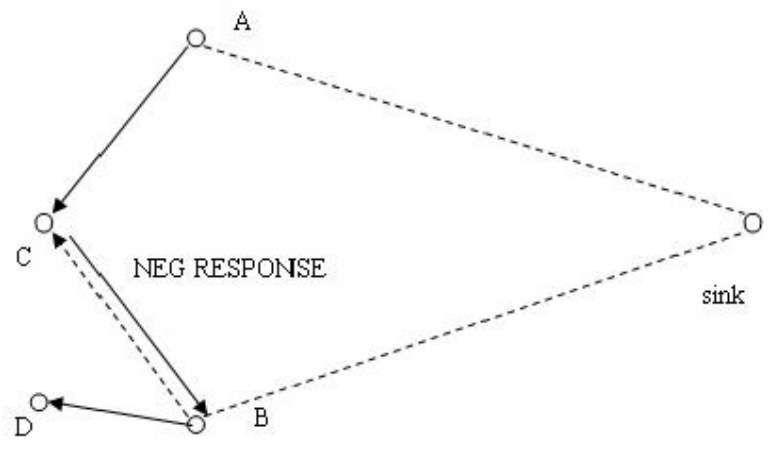

Figure 5.7: Scenario to reinforce multiple paths

(paths with disjoint nodes are definitely disjoint paths), but also ensures loop-free path since loops are broken when selecting the next candidate in the local table.

We will see an example of loops can be avoided. In figure 5.8, A has already reinforced B and C, D, E are reinforced sequentially.

When E tries reinforcing B, B drops the packet and sends NEG RESPONSE to E. So E is able to select the next candidate F to reinforce. Standard diffusion only reinforces one route and when there is a loop, the reinforcement packet is simply dropped and no packet is received at the source. Our algorithm guarantees the success of reinforcement packets reaching the source provided that the delay constraint could be satisfied.

5.5 Queue Priorities

We mentioned earlier that we make use of queues to buffer data. Queue length is one of the parameter which is used to get a estimate of congestion. Instead of using simple queues we make use of priority queues which will give preference to higher priority applications. In some applications like video or voice transmission, we have to stream data real time. In such applications, dropping some packets does not cause as much distortion as delay in trying to achieve maximum throughput. For such applications, jitter control is more important

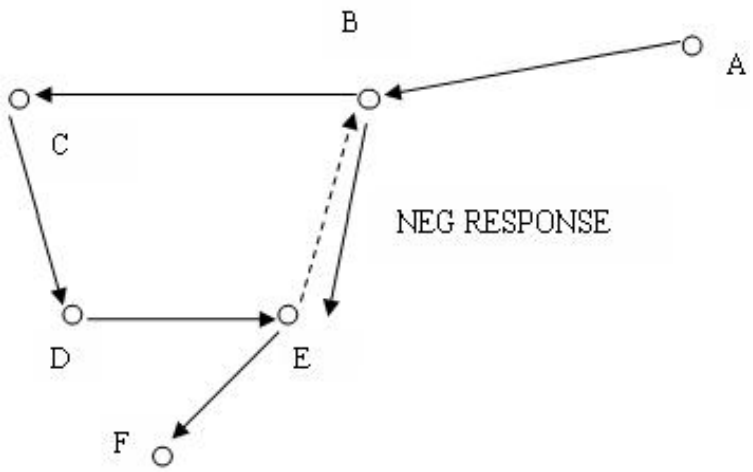

Figure 5.8: Avoid Loops using NEG_RESPONSE

than reliability. In other applications where we transmit highly important data, we have to achieve maximum possible reliability and throughput. For such applications, we can have less stringent deadlines but higher priorities, which will make the queues, deliver as much packets as possible. Deadlines will buy more waiting time for all the packets in buffer before getting dropped. This will in turn make sure that packet drop rate for such application is as low as possible. At the same time priority of the application will help it get a preference in queues. This will imply that the application packets will be transmitted as early as possible. These application packets will get more preference over the other applications; hence more deadlines will be met. Sometimes, due to high network load, we might not be able to meet deadlines of some packets. We will drop these packets as transmitting will use energy as well as network resources. It is left on user to determine the application priority and deadline based on the specific requirements. Both the queues explained below help us achieve the goals described in above paragraph.

5.5.1 Inter-Queue Priority

The application assigns the priorities for heterogeneous traffic. If there are multiple applications running on same nodes, each of them will have a application queue which will

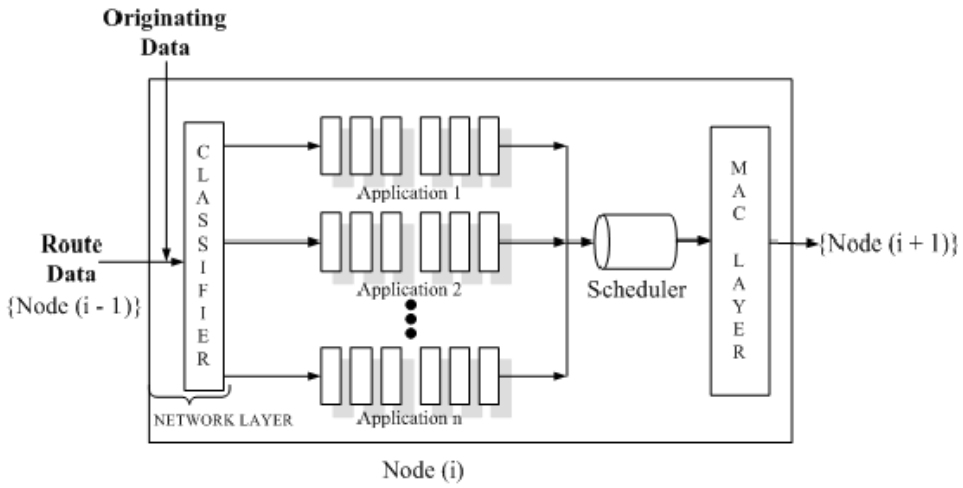

Figure 5.9: Inter Queue Priority

hold packets for that application. All the packets coming out of that queue will have same priority as the application. Therefore, we can say that each data queue shown in Figure 5.9 has its own priority. This is termed as Inter-Queue priority. The scheduler schedules the queues according to the Inter-Queue priority. It decides the service order of the data packets from the queues and manages the queues according to their priorities. This ensures the data with higher priority to get higher service rate.

Each application $A_1, A_2, A_3, ..., A_n$ will have a priority assigned to it say AP_1, $AP2$, $AP3, ..., APn$. This application priority will normally be in the range from 1 - 10, where 1 will be lowest priority and 10 will be highest priority. We are using a reverse convention of what is normally present in Linux systems. The reason for that is, it will help us calculate the data rate for a particular application depending upon this number. Hence to simplify the calculations, we make use of priorities in a different manner. Data rate of applications is determined by using the priority assigned to the application queue. Queue length is one of the parameter which is used to get a estimate of congestion. Instead of using simple queues we make use of priority queues which will give preference to higher priority applications. In some applications like video or voice transmission, we have to stream data real time. In such applications, dropping some packets does not cause as much distortion as delay in trying

to achieve maximum throughput. For such applications, jitter control is more important than reliability. In other applications where we transmit highly important data, we have to achieve maximum possible reliability and throughput. For such applications, we can have less stringent deadlines but higher priorities, which will make the queues, deliver as much packets as possible. Deadlines will buy more waiting time for all the packets in buffer before getting dropped. This will in turn make sure that packet drop rate for such application is as low as possible. At the same time priority of the application will help it get a preference in queues. This will imply that the application packets will be transmitted as early as possible. These application packets will get more preference over the other applications; hence more deadlines will be met. Sometimes, due to high network load, we might not be able to meet deadlines of some packets. We will drop these packets as transmitting will use energy as well as network resources. It is left on user to determine the applications priority and deadline based on the specific requirements. Depending on the assigned data rate and number of application, the available data-rate of these applications will vary. If a higher priority application does not have any data to transmit it will relinquish its slot and that will be used by other application which has a lower priority than it and has some data to transfer.

As sensor nodes have limited processing power and memory, we have to implement a simple scheduler which will not incur high power usage. Hence, we came up with a simple scheme where-in we add up the application priorities of all available applications. Each application is eligible for a share of data-rate, which is proportional to a fraction of its priority against the total application priorities. Hence, to represent it mathematically, the eligible share of each application i will be:

$$sum = \sum_{j=0}^{n} AP_j \tag{5.6}$$

$$F_I = \frac{AP_i}{sum} \tag{5.7}$$

The data rate available for this application will be this fraction of total data rate.

$$D_i = D * F_I \tag{5.8}$$

Each application will transmit data with a data rate of D_I. This data rate is allocated in round robin fashion. Each application A_I will get a time slot every n units. If the application queue has any data to transmit during that time slot it will use it transmit data. If it does not have any data it will pass it on it the next application in the queue. Consider a sensor node with 3 applications A_1, A_2, A_3. Each of these applications has a priority of 4, 7, 9 respectively. Depending on priority we can calculate the fraction F available for each of these applications as:

$$sum = \sum_{I=0}^{2} AP_I \tag{5.9}$$

$$sum = 4 + 7 + 9 = 20 \tag{5.10}$$

$$F_1 = \frac{4}{20} = \frac{1}{5} \tag{5.11}$$

$$F_2 = \frac{7}{20} = \frac{7}{20} \tag{5.12}$$

$$F_3 = \frac{9}{20} = \frac{9}{20} \tag{5.13}$$

Assume that the total data rate available for given sensor is 10 packets/sec. Using the above fractions we will determine the data rate available for each of the applications:

$$D_1 = \frac{4}{20} * 10 = \text{ 2 packets/sec} \tag{5.14}$$

$$D_2 = \frac{7}{20} * 10 = 3.5 \text{ packets/sec} \sim 3 \text{ packets/sec} \tag{5.15}$$

$$D_3 = \frac{9}{20} * 10 = 4.5 \text{ packets/sec} \sim 4 \text{ packets/sec} \tag{5.16}$$

The total data rate based on this is 9. But the allocated data rate is 10. To balance this we can allocate the remaining slots (1 in this case) to all the applications beginning from highest priority application. Hence the final data rate assigned to applications is:

$$AP_1 = 2 \; packets/sec \tag{5.17}$$

$$AP_2 = 3\ packets/sec \tag{5.18}$$

$$AP_3 = 5\ packets/sec \tag{5.19}$$

5.5.2 Intra Queue Priority

Apart from data produced by a node itself, there can be other sources of data. The same node can act as an intermediate node for other source - sink pair. In such case, there will be two sources of data. One will be data produced by itself. The other one will be data routed through it. We cannot ignore this source of data while designing priority queues. As directed diffusion is an ad-hoc network protocol, all the routing takes place at network layer. Hence, each packets next hop is newly decided by the current node. So, when the data to be routed comes in, it will be transferred to network layer, where the node will look up its gradient table and forward packet depending upon its interest. In our case, when node travels to network layer, we have a filter which is a priority queue. This means that when the packet comes in with a priority already assigned to it by its original application, it will be automatically put into one of the priority queues. All the queues shown in Figure 5.10 are priority queues.

Priority queues are used for giving the route data more priority than originating data. The reason behind this is that, as route data has already traversed some hop(s), their loss would cause more wastage of network resources than that of the originating (i.e., source) data. Hence, it would be better to forward those as soon as possible after receiving from the immediate downstream node. We term this type of priority as Intra-Queue priority. The classifier can assign the priority between the route data and originating data by examining

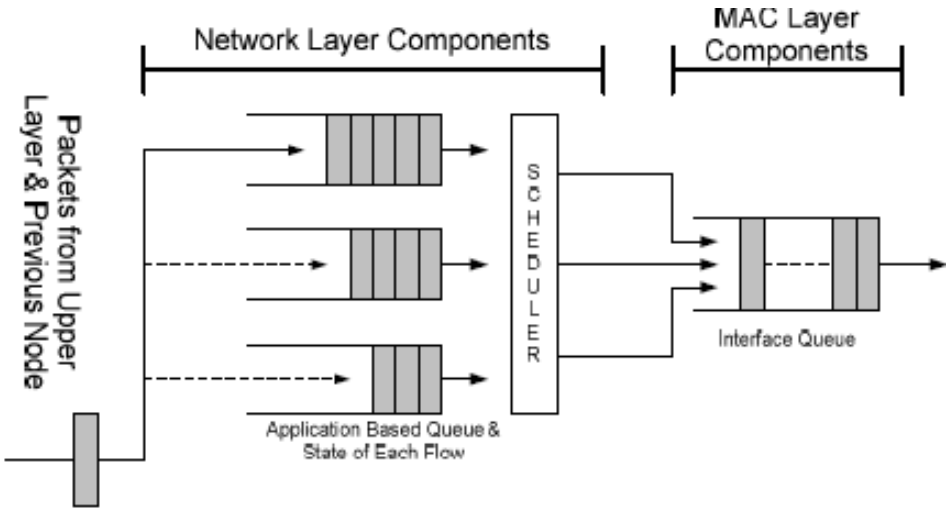

Figure 5.10: Intra Queue Priority

AP3	AP2	AP1	AP3	AP2	AP1	AP3	AP2	AP3	AP3	AP3	AP2	AP1	AP3	AP2	AP1	AP3	AP2	...	...	...

Figure 5.11: Time Slots assigned for Applications.

the source address in the packet header. In our case this classifier is deadline. Naturally, the deadline of originating data at the node is going to be less stringent than the deadline of packets being routed through the node. The packets being routed will have already elapsed some time in travelling. So we consider deadline as other of the metrics in deciding intra queue priority. As each packet header has a field which includes deadline for that packet, our job is simplified. Each of these queues will sort the packets in terms of deadlines. The packet having lowest deadline will be sent first. Hence we will try our best to route the packet in least time with minimum possible waiting delay. As mentioned above each of the application queues will get slots where they can transmit data. In the example mentioned above, we allocate three time slots to each of the applications. Figure 5.11 represents the data transmission of 3 applications.

Now, if in the duration of transmission, Application 3 does not have any packets to transmit in its given time slot, Application 2 can transmit data. There can also be a scenario

where-in a packet selected for transmission has missed its deadline. In such scenarios, we will have to drop that packet because it is already late. Delivering such packets to sink will result in resource utilization without increasing throughput or data quality.

5.6 Congestion Detection

In order to determine the current network state, the sink must be able to detect congestion in the network. However the conventional ACK/NACK-based detection methods for end-to-end congestion control purposes cannot be applied here. The reason once again lies in the notion of event-to-sink reliability rather than end-to-end reliability. Only the sink, and not any of the sensor nodes, can determine the reliability indicator i and act accordingly. Moreover, end-to-end retransmissions and ACK/NACK overheads are a waste of limited sensor resources.

Hence, we use a congestion detection mechanism based on local buffer level monitoring in sensor nodes. Any sensor node whose routing buffer overflows due to excessive incoming packets is said to be congested and it informs the sink of the same. The traffic generated during each reporting period, mainly depends on the reporting frequency f and the number of source nodes n. The reporting frequency f does not change within one reporting period. Assuming n does not significantly change within one reporting period, the traffic generated during the next reporting period will have negligible variation. Therefore the amount of incoming traffic to any sensor node in consecutive reporting intervals is assumed to stay constant. This, in turn, signifies that the increment in the buffer fullness level at the end of each reporting interval is expected to be constant.

Let,

b_k and b_{k-1} be the buffer fullness levels at the end of kth and (k-1) th reporting intervals respectively and

B be the Threshold value for the queue in Figure 5.12.

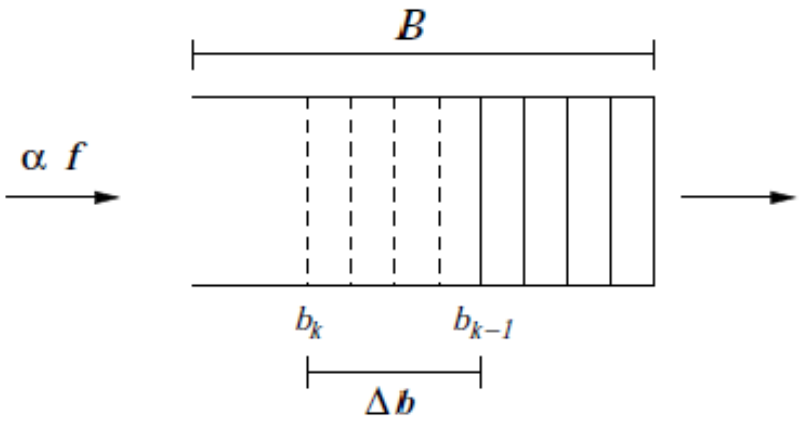

Figure 5.12: Queue Threshold

For a given sensor node, let Δ b be the buffer length increment observed at the end of last reporting period,

$$i.e., \Delta b = b_k - (b_{k-1}) \tag{5.20}$$

Thus if the sum of current buffer level at the end of kth reporting interval and the last experienced buffer length increment exceeds the buffer size,

$$i.e., b_k + \Delta b > B, \tag{5.21}$$

the sensor node infers that it is going to experience congestion in the next reporting interval. When node infers that its queue length is greater than threshold, it will put its queue length in the header field. Also, it will increment the Node Count Field in packet header. All the nodes lying on the path till sink will check their queue length against the threshold value and update the Node Count Header field accordingly. Nodes also check the Queue Length field, and if their queue length is more than queue length stored in packet header, they will update the queue length field in header of their queue length. As we can

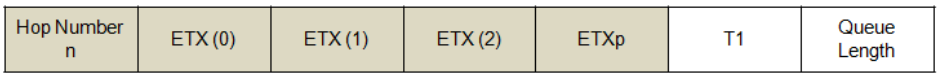

Figure 5.13: New fields added to keep track of congestion control

see in the Figure 5.10 , every node has multiple queues corresponding to each application. If node has many applications running on it, the individual queue for each queue length might not be high, but the cumulative queue length might be more than the node threshold. Hence while calculating the congestion state, node always takes cumulative queue length into account. Cumulative queue length is the actual number of packets node has to send or route.

When the sink receives this packet, it will have all the information about the nodes lying on its path. To include this information we have added following extra fields. The fields shown in gray have been added earlier. The new fields which have been added to take care of congestion are T_1, maximum queue length and Node Count.

5.7 Congestion States

If on a given route, one of the nodes has poor link quality, or is getting overloaded, the overall throughput of path degrades. If the queue length of node is over the threshold, it means it is getting overloaded. This will result in packets being dropped which will reduce the efficiency of route. We need to take some action before the condition degrades. The nodes will signal sink even before they get congested (cross the queue threshold). This gives time for sink to send a message back to source and take preventive action if the node is getting congested or corrective action if node is already congested. As we also consider ETX to compute the congestion condition, we will take into account network interference and packet retransmissions due to MAC layer losses. Sink will analyze packet headers and conclude if node(s) on the route is congested.

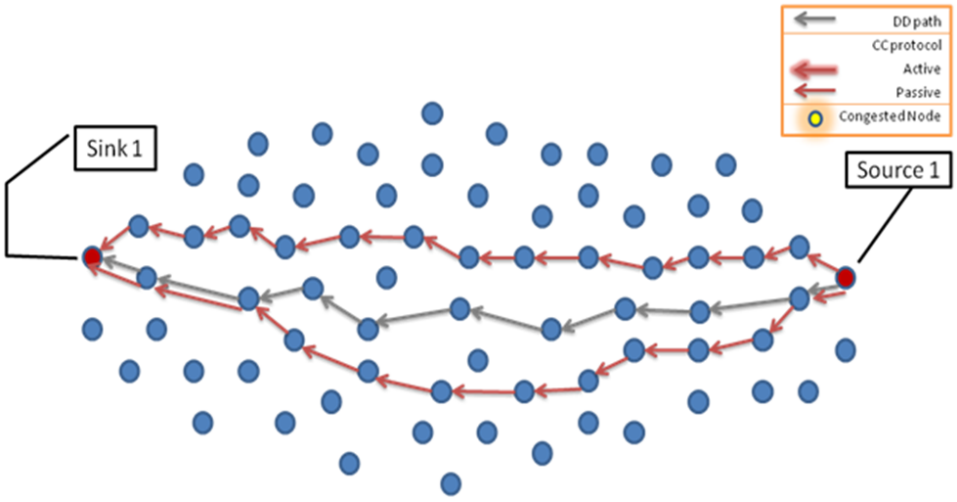

Figure 5.14: Scenario with a single source - single sink showing paths formed by Directed Diffusion and proposed Protocol

To consider a simple scenario, we can assume that there is a one source sink pair in the network. Both directed diffusion and Proposed Congestion Control protocol use interest packets, to establish paths. During the interest propagation, we take special care to keep track of two paths which are completely distinct than the other. We make use of NEG_RESPONSE to drop the nodes which have already taken part in interest propagation by some other path. Once the interest packets travel to sink, both the protocols use different metrics to decide upon the final path. While deciding upon the final path, directed diffusion finalizes only one path and sends a reinforcement message along that path. In case of DDCC we send positive reinforcement to multiple paths (in case of simulations we normally use two paths). Once the positive reinforcement reaches source, the gradients across the network will be set to denote these paths. The 5.14 shows the network state once the gradients are set. To differentiate between the protocols, we use color convention. The path selected by directed diffusion is shown in black, where as path selected by proposed protocol is shown in red. Also we observe that there are two paths in red. But in the initial state, we only use one of these paths as active path.

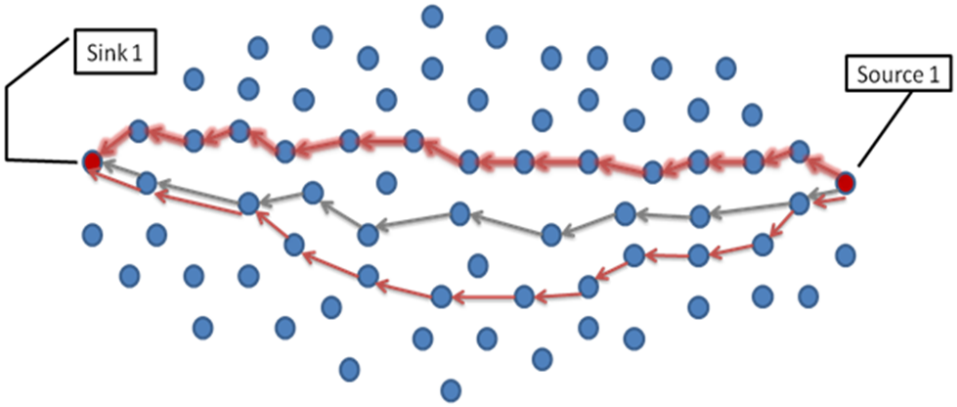

Figure 5.15: Scenario with a single source - single sink showing data flowing through active path.

We maintain the list of multiple paths in a data structure which is similar to circular queue. When the source starts sending data, it will select the first path available in the queue. This path will be most reliable, and will provide the best quality. For directed diffusion, no such data structure is required. Hence it will directly select the only path available and start sending data over that path. Of the two paths available, we show the active path by highlighting it with a red glow around it. The source sink pair will be using only one path initially. Hence only one path glows. The state of network once data transfer starts is shown in Figure 5.15:

If there is congestion over active path of proposed protocol, then we switch path to the lower one. But in case of directed diffusion, we will have to initiate route discovery and find a new path. Depending on the information received and the network state, various possible scenarios can occur. We explain some of the peculiar scenarios, which will take care of all the possible network conditions.

5.7.1 Initial Congestion State

We can say that network is in initial congested state if received queue length is not more than the threshold value and ETX value is unchanged (or not significant change). There can be many reasons which result in increase in queue length. They do not indicate congestion in

all cases. But, congestion can cause rise in queue length of congested node. Hence, we cannot predict that the cause for this flag to be raised is congestion. It can also be a momentary congestion caused due to burst of data at the source. In this case sink starts monitoring ETX and queue length for next few packets. For our simulation purpose we have assumed this number to be 7. If we find that ETX or queue length is increasing, we send a switch path message. But if the network condition remains stable and does not change over the period, we do not take any action. If the queue length of a node is greater than threshold, it will mean that the node is being overused. It might be because many sources can use it as an intermediate node. If this is the scenario, then due to interference of other paths, the ETX will increase and raise the congestion flag. But if the node itself is producing lot of data in addition to forwarding data, then its queue might get filled. In such scenarios it is not advisable to still use the same path because soon the node will start dropping packets. So to avoid such scenarios, we send a switch-path-message to source, and take up the alternate path. There can be scenarios where we predict that there is congestion, when there is not, and send a switch_path_message. This is a case of false alarm. But this will only cost us one extra packet over the network. Because, once sink sends switch_path_message to source, source will switch its path to the alternate path and continue sending data. The alternate path selected is still better than the default path selected by directed diffusion. This can be clearly seen in our simulation results.

5.7.2 Congested Network

Lets consider a scenario where network is in congested state, and the congested node raises the flag. A node will update the packet header with its queue length. Packet header will also include ETX, and timeout value for that packet. At sink, when it receives the information, it will check the Queue length field included in packet header. If $Queue\ Length > Threshold\ value$, we send switch_path_message due to possible losses explained above. If the ETX value of the path is greater than the ETX_MAX value, (which

will be the max value of ETX, which will still deliver the packets within timeout.) then it is indication that the network resources are being overused and we can be sure that there is congestion in the network. We can also be sure that there is lot of interference, which will cause packet loss at MAC layer, and queues are building up on the nodes. We can also check the delay parameter, which will be more because of possible retransmission in and around the congested region.

5.7.3 Prolonged congestion

A sensor can also suffer from prolonged congestion in some instances. This mainly occurs in scenarios where multiple sources, which do not have distinct paths to their respective sinks, are generating data at high rate. In this case, the node where two or more paths intersect will have to forward data packets from both the sources. This node will have high influx of data, and if this is more than the link capacity or node capacity, then it will not be able to handle it. This will cause in packet loss and eventually result in congestion. In such cases, if we switch path it will reduce congestion momentarily, but eventually mean shifting the point of congestion. This can also cause source-sink pairs to switch paths in round robin fashion, without improving network scenario. To avoid such cases, we go through the list of alternate paths only once. If we still face congestion over the new path, we conclude that the network load is more than what a single path can handle. This gives rise to the need of multiple paths. We send a multi_path message, if we traverse the set of routes once.

5.7.4 Path failure

Our algorithm is designed for avoiding congestion. But in some cases it can be used to recover route. Though we dont focus on this issue, we address this issue in this section. In ad-hoc network it is not possible for a sink to know if the source has data or not. So we introduce a control packet, End_Transmission, which is sent by source to sink when it has sent all the data. Once an intermediate node receives this message, they can go into dormant

state, unless they are being used by some other path. If during transmission, sink does not receive any message from source for a $time > 2 * delay$, (where delay is the time required by last packet to travel from source to sink), sink assumes that there has been a problem on the path from source to sink. It is not possible to know the cause of this delay. It can be node failure, link failure, or very high interference caused due to some external device. In this case, sink sends a switch_path_message to source and tells source to start sending data using alternate path. This feature can be used for path recovery. The advantage of using this method is that it will select and alternate path which is guaranteed to satisfy the application deadlines, throughput and QoS requirements.

We can have different source - sink pair combinations. Depending on the data rate and required throughput, the network will fall in one of the above described states. We cover some of the basic designs in which we can place source and sink in the following section.

5.8 Scenarios

In this section we try to cover all the possible source sink pair combinations which can exist in the network. For the ease of tracking events, measuring the metrics, and doing calculations, we limit ourselves to two source sink pairs for our simulation.

5.8.1 Overlapping paths

The first scenario which we describe here is where we have two source sink pair's. Both the sources viz. Source1, Source2 work independently using the two different protocols. If the network is running on directed diffusion, then it will find only one path between source - sink. This path is shown in black. There will one path between each of the respective source - sink pair. If the network is running proposed protocol, then it will find two paths. These paths are shown in red. We consider a scenario where one of the paths determined by one source - sink pair will partially overlap with one of the paths of other source - sink pair. The paths dont cross each other, but at some point, a part of the path overlaps with the

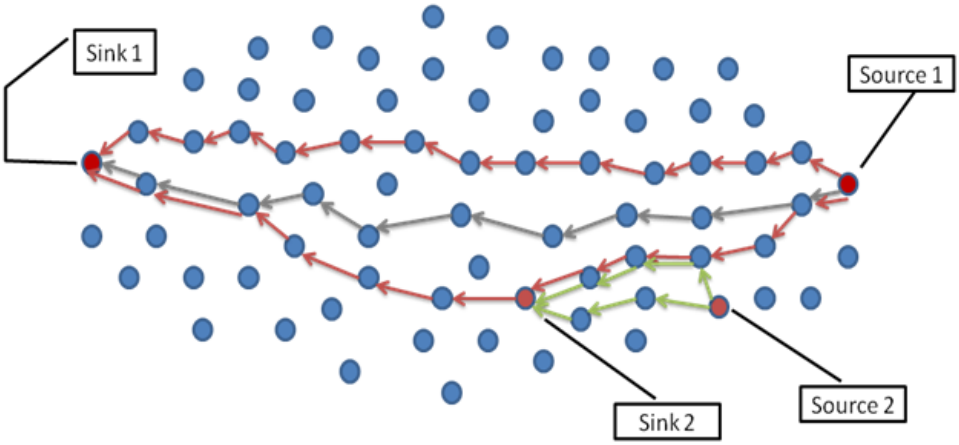

Figure 5.16: Scenario where one of the paths for source sink pair overlaps

other. When the source sends interest to corresponding sink, they are unaware of the other source-sink pair existing in the network. Intermediate nodes will receive interest packets from both the sources. When the intermediate nodes see these interest messages, they compare attributes of newly packet with the entries of their routing table. Hence, when the sources determine path, these paths are distinct to the other path of the same source. There is a possibility that one of the paths might overlap with the other source - sink pair. We will not be able to predict which path will coincide. In our scenario, we consider that the primary path of first source - sink pair will be the one which overlaps with the primary path of other source-sink pair. In the initial stage of the network, data is being transmitted on primary path. This primary path of both source sink pairs overlaps with each other. This is shown in Figure 5.16.

Once the path is set, the source starts sending data to sink. As we know in the initial phase of network, we just use one of the paths which will be called primary path. As we can see in figure 5.16, three nodes are shared by both paths. This means that only half of the total throughput of the link will be available for each of the paths. If the sum of data rate of both the sources is greater than the total throughput of the link, then it is natural that three nodes will be overloaded. Those three nodes will not be able to handle the resultant load of source1 and source2. That is shown in Figure 5.17 in yellow.

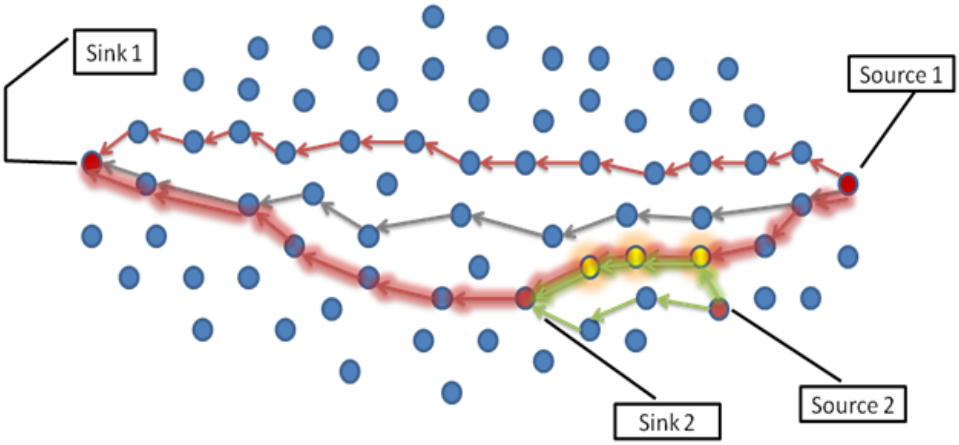

Figure 5.17: Overlaping path scenario showing data transfer over the common path

This overload of data will cause congestion over those three nodes. Due to high rate of incoming data, queues will start building up on those three nodes. Apart from building up of queues, there will be other effects of congestion. As the queues build up, waiting time of packets in those queues will increase. This will increase the delay for these packets. In addition to this, queues and high load over the channel will also cause interference around that area. We keep measuring these metrics at sink. Once we notice that these metrics are increasing, we send a switch_path_message. Now in this scenario, we cannot really determine which of the sinks will notice the network congestion. We assume that the sink closer to congestion region will detect degradation of these metrics, and send switch_path_message to its corresponding source. When the source receives switch_path_message it sends data via alternate path. That is depicted in Figure 5.18.

5.8.2 Partially crossing paths

One more scenario that we will be covering is where paths of source - sink pair partially cross each other. In this scenario, we have one source - sink pair going across the network. We have placed another source in the middle of two paths between $source_1 - sink_1$. In this scenario, both the paths between $source_2 - sink_2$ will cross one of the paths between $source_1 - sink_1$. Even in this case, when two sources - $source_1$ *and* $source_2$ broadcast interest

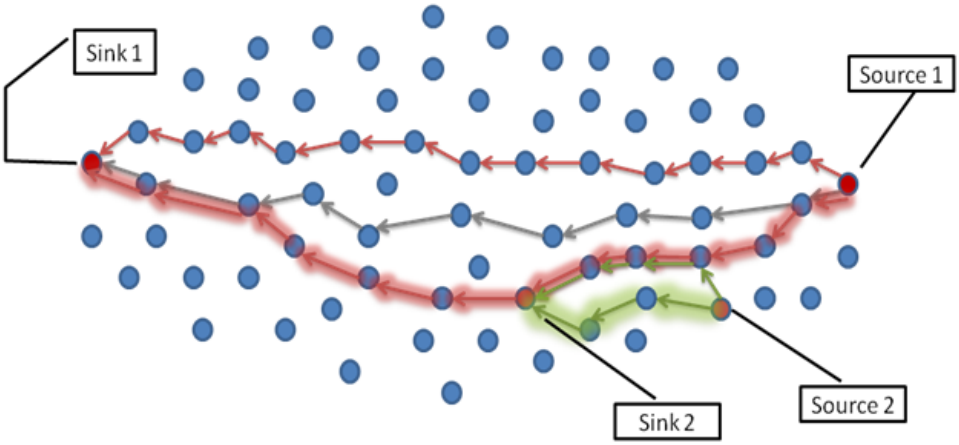

Figure 5.18: Overlapping path scenario where path switch occurs resulting in source-sink pairs taking different paths which resolve congestion.

messages to find path to sink, all the intermediate nodes consider the two interest packets as different packets, as they have two different attributes. For each of these source-sink pairs, there will be only one path that is active. The way these paths are formed is shown in Figure 5.19.

Once the sources start transmitting data, the data flows through one of the paths - the primary path. This path will be active path for that duration. We show the active path as glowing red. For the sake of explanation we consider the top path of $source_1 - sink_1$ pair to be active. Also, we consider the left path to be primary path for $source_2 - sink_2$ pair. As we can see in the figure, these two paths intersect each other at a node, which is shown in bright yellow. Also, we show the path taken by Directed Diffusion in black. As we know Directed Diffusion will find only one path from source to sink. The paths for these two source sink pairs will intersect each other at a node represented in dark yellow. This is the node where congestion occurs, if the source-sink pair runs on directed diffusion. The bright yellow node represents congestion spot for proposed protocol. We use the same metrics explained in previous scenario to get an estimate of congestion. If we assume that the pair $source_1 - sink_1$ detects congestion and sends a switch_path_message, then the congestion region is avoided. In addition to this the paths of source sink pairs are now distinct. This

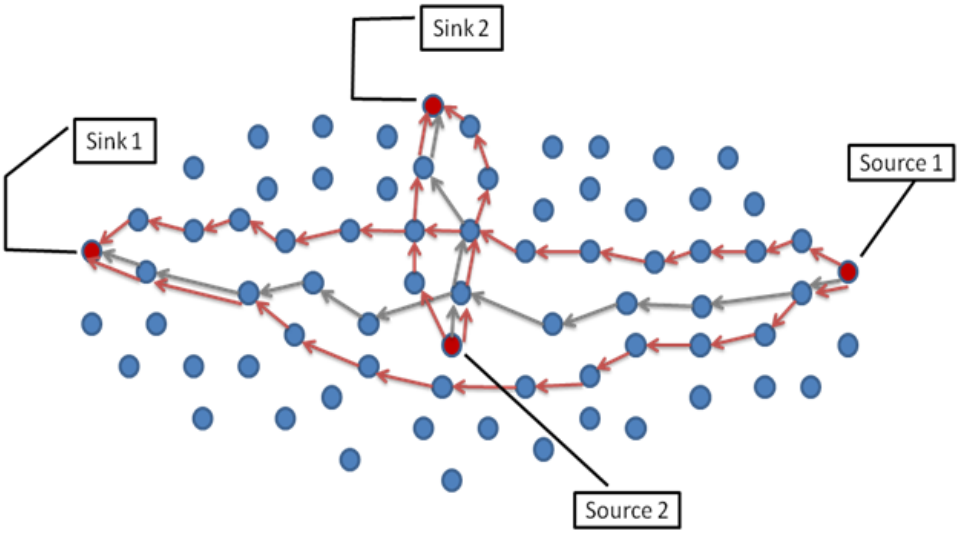

Figure 5.19: Partially crossing path scenario showing two set of paths discovered by each source-sink pair.

will ensure that congestion does not occur in the network due to interference of these two paths! The forming of congestion spot is shown in Figure 5.19.

Due to this congestion spot, if the other source-sink pair (pair 2) decides to switch paths and try to avoid congestion, it will send a switch_path_message. Now this switch_path_message will go to $source_2$. Once $source_2$ switches path, the alternate path will be taken. Now, none of the source, sink nodes are aware of other source sink pair existing in the network, or the path which goes across its intermediate nodes. The switching of path will result in $source_2 - sink_2$ taking an alternate path. But the alternate path also intersects the $source_1 - sink_1$ path in the same way as does the primary path (Figure 5.21). Hence, it might resolve congestion for a few milli/micro seconds. But due to the existing high data rate on few of the nodes of alternate path, it will result in congestion over the nodes common to alternate path of $source_2 - sink_2$ and primary path of $source_1 - sink_1$. We term this as *Congestion-Shift*. This shift takes place in Figure 5.22 .

Once Congestion Shift occurs, the performance of both paths going through congestion region will degrade. This will again trigger a congestion avoidance mechanism. Now, at this stage, if the other $source_1 - sink_1$ pair detects congestion, then it can send a

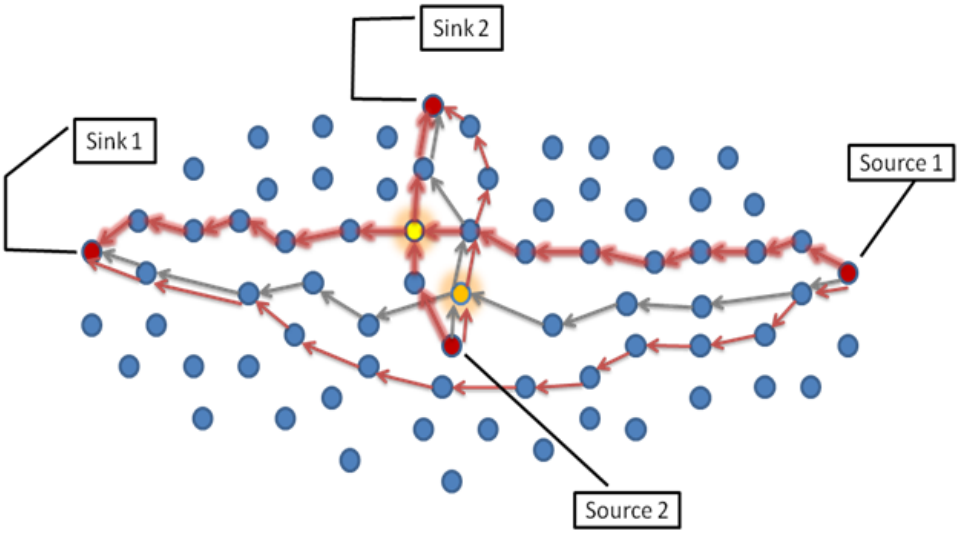

Figure 5.20: Partially crossing path scenario shows that congestion is building up over common nodes on primary path of both source - sink pair. The same happens for directed diffusion.

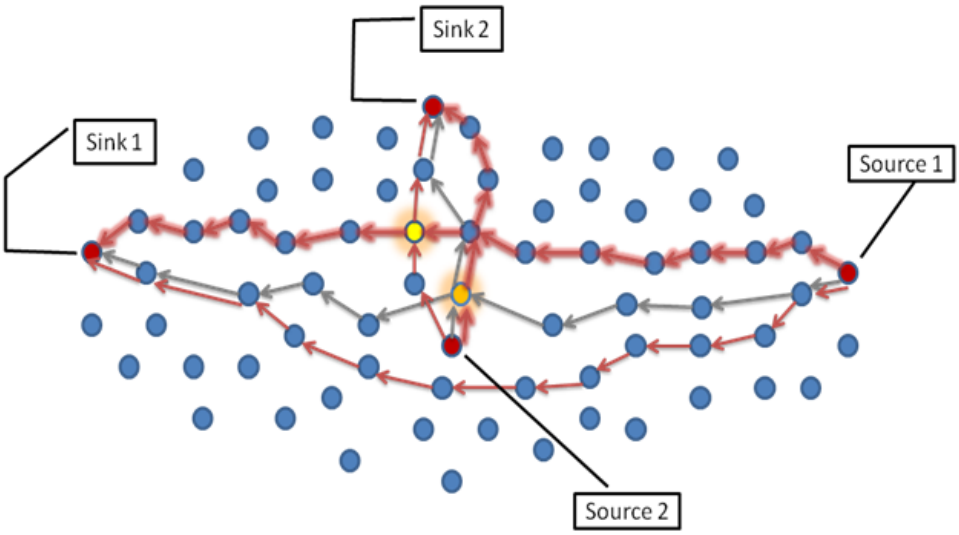

Figure 5.21: Partial crossing path scenario showing reduced congestion over common nodes on primary path, after alternate path is selected by one source - sink pair

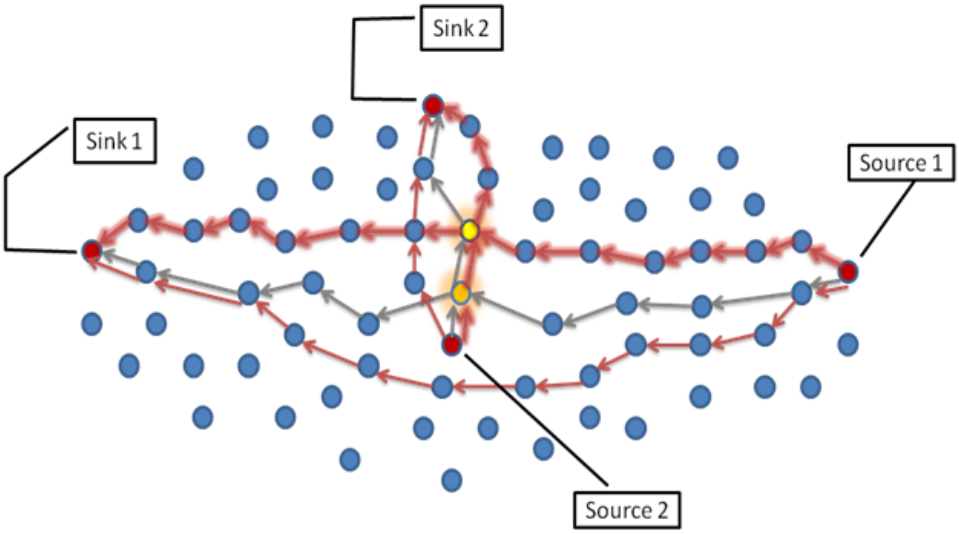

Figure 5.22: Partial crossing path scenario showing congestion building up on the intermediate nodes of alternate path. We term this as Congestion - Shift.

switch_path_message to its corresponding source. This switch path will completely avoid the congestion region and solve the problem of congestion. But, instead of $sink_1$ detecting congestion, if $sink_2$ detects congestion. It will take preventive measures to avoid congestion. In case of $sink_2$, it has already used both the paths - primary path and alternate path. If congestion still exist on the link, then it will deduce that the source sending is too high for one single path to handle. Hence it will decide to use multiple paths. Which means it will send multi_path_message to $source_2$. When source sees this message, it will start sending data over all the paths through which it receives multi_path_message. Sink sent the multi_path_message through two paths. Hence, $source_2$ will transmit using the two paths as shown in Figure 5.23

Even when the $source_2 - sink_2$ is using multiple paths to transmit data, there are a few common nodes resulting in intersecting paths. This might take care of overall load on the paths, but there might still be some interference due to essentially three paths going thorough that region. This might still affect the performance. This can cause $sink_2$ to ask $source_2$ to send data at slower rate. But even in this case if $sink_1$ detects congestion and switches path, then it will resolve the congestion problem. Once, this happens, $sink_1 - source_1$ will

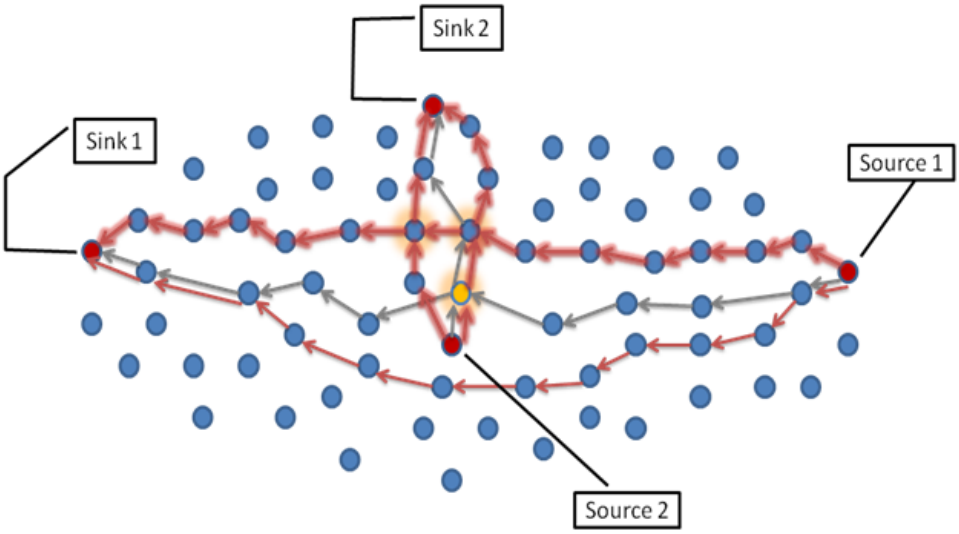

Figure 5.23: Partial overlapping path scenario showing use of multiple paths.

use alternate path, whereas $source_2 - sink_2$ will be using multiple paths. There might not be a need to use multiple paths once $source_1 - sink_1$ changes paths. But we dont have any mechanism to signal the $source_2$ to switch back to single path or switch back to primary path. Even in this case, as we know that the network state will be refreshed every few minutes. Hence, we can rely on that for source - sink pairs to make efficient use of resources.

5.8.3 Complete crossing paths

The last scenario we will consider is a network where we have two source - sink pairs whose paths cross each other completely. Which means, all the paths between source - sink pair (including alternate paths) will intersect the paths from other source-sink pair. We place source sink pair across the network, in such a way that it completely crosses the other source sink pair. Hence the source - sink pair essentially form a 'X'. The intersecting point of the two paths will be a set of nodes where the resultant load/throughput will be equal to the sum of throughput offered by both source-sink pair. This will result in congestion around the node where the paths intersect. The formation of paths is shown in Figure 5.24.

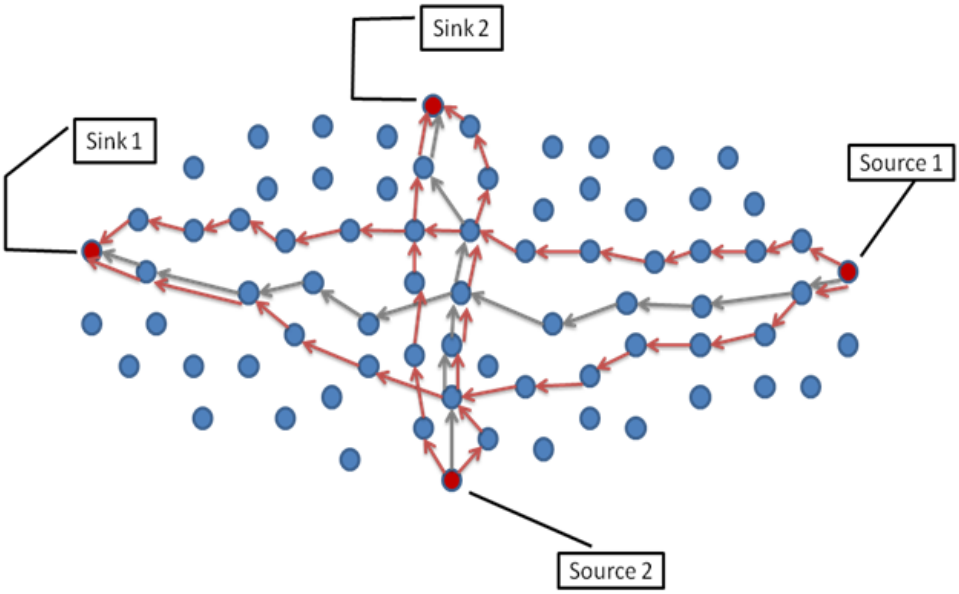

Figure 5.24: Scenario with source - sink pair having cross paths

Each of this sources has two paths going through network to the sink. As they lie across the network, they, both these alternate paths cross each path of other source-sink pair. When they sources want to transmit data, they will choose one of the paths which will be a primary path for that source. As primary paths of two sources cross each other, the node common to both paths will have to forward incoming data from both sources to respective nodes. Which means the common node will have effective data-rate equal to sum of data rate of two sources. If the resultant data rate is more than capacity of network, then the node will not be able to handle the load. Nodes queue will start to building up. It will start dropping packets. It will also result in congestion around that node. This is shown in Figure 5.25 .

When the throughput of two paths starts decreasing, one of the two sinks will detect it. The sink will send a switch_path_message so as to indicate the source to send data via a different path. But as we can see in Figure 5.25, switching paths will not really help as the alternate path intersects with active path of other source sink pair (all paths of a source-sink

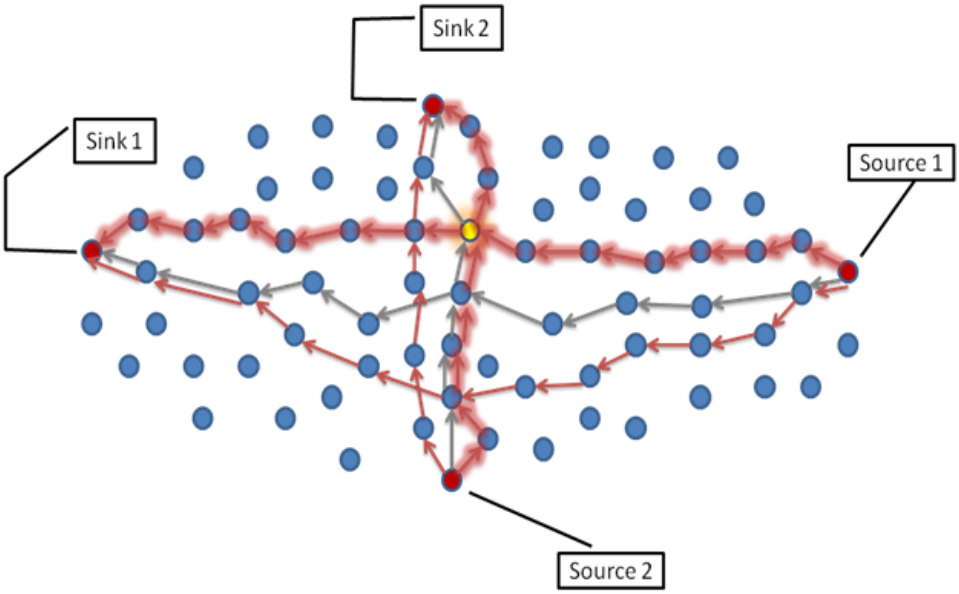

Figure 5.25: Cross path scenario showing active path being used for data transfer

pair intersect with all paths of other source-sink pair). The node which is congested is shown in yellow in Figure 5.25 .

When the path switch occurs, it will result in Congestion-Shift which explained in the previous scenario. When congestion shifts occurs, it will be again detected by one of the two sinks. The network can be in different states depending on which sink detects congestion. If the second sink detects congestion, it will be the first time it detects congestion. Hence it will send a switch_path_message to its source. This will cause the source-sink pair to use an alternate path. This will again result in congestion shift as in the previous case. The network will still remain in the same state even after sending one more switch_path_message. But instead of other source-sink detecting, the original sink detects congestion, then the network might end up in a different condition. The source-sink pair has already sent a switch_path_message and has used all its paths. So it will send a multi-path over multiple paths. Hence one of the source-sink pair's will start using multiple paths. If there is still congestion in network, then the other source-sink pair might also start using multiple paths. This will cause each of the source-sink pair's to use two paths. There can be a maximum of

4 nodes in common lying on these 4 paths. These paths will share data rate of each source, and eventually reduce the total load over common nodes. In some, cases congestion can still exist, if the load offered by sources is very high. In such scenario dividing load over multiple paths will not solve the congestion problem. It will just result in spreading the effect of congestion over different nodes. But it will not completely take care of congestion. We cannot really do anything if the load offered is too high for the network to handle. The only option that we have is to send a message to source to slow down so as to alleviate congestion. This part has not been covered in this thesis. In these cases we can call some other congestion control technique like CODA.

Chapter 6

Simulation And Results

6.1 Simulation Environment

The simulator we have used to simulate the directed diffusion ad-hoc routing protocols is the Network simulator 2 (ns) [88] from Berkeley. To simulate the mobile wireless radio environment we have used mobility extensions to ns that is developed by the CMU Monarch project at Carnegie Mellon university.

6.1.1 Network Simulator

Network simulator 2 is the result of an on-going effort of research and development that is administrated by researchers at Berkeley. It is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols. The simulator is written in C++ and a script language called OTcl. NS uses an OTcl interpreter towards the user. This means that the user writes an OTcl script that defines the network (number of nodes, links), the traffic in the network (sources, destinations, type of traffic) and which protocols it will use. This script is then used by ns during the simulations. The result of the simulations is an output trace file that can be used to do data processing (calculate delay, throughput etc) and to visualize the simulation with a program called Network Animator (NAM). An overview of how a simulation is done in ns is shown in Figure 6.1 .

The version 2.29 of the Network simulator supports simulation of mobile wireless environments. The Network simulator alone was initially only intended for stationary networks with wired links. Researchers at CMU Monarch group realized the need for mobility models in ns and therefore started to design and implement a mobility model that would extend the

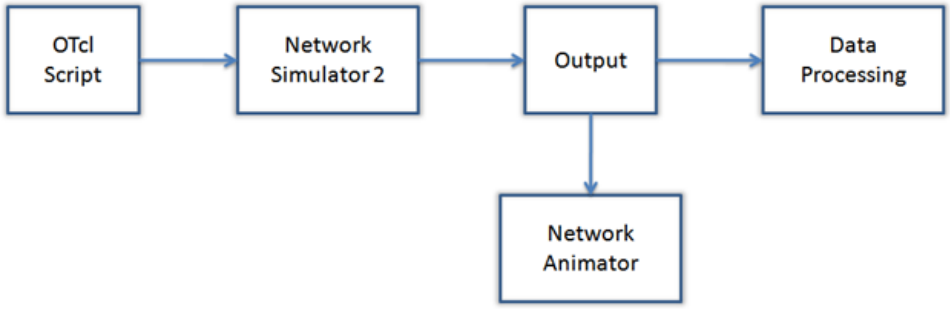

Figure 6.1: Top level overview of input/output in Network Simulator - 2

simulator. The CMU Monarch extensions to ns provide new elements at the physical, link, and routing layers of the simulation environment. Using these elements, it is possible to construct detailed and accurate simulations of wireless subnets, LANs, or multi-hop ad-hoc networks. Recent versions of ns contain further extensions to this model to allow combined simulation of wired and wireless networks. The following section provides an overview of the extensions added to NS.

6.1.2 Mobility Extensions

Node Mobility

Each mobile node is an independent entity that is responsible for computing its own position and velocity as a function of time. Nodes move around according to a movement pattern specified at the beginning of the simulation.

Realistic Physical Layers

Propagation models are used to decide how far packets can travel in air. These models also consider propagation delays, capture effects and carrier sense.

MAC 802.11

An implementation of the IEEE 802.11 Media Access Protocol (MAC) protocol was included in the extension. The MAC layer handles collision detection, fragmentation and acknowledgments. This protocol may also be used to detect transmission errors. 802.11 is a CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) protocol. It avoids collisions by checking the channel before using it. If the channel is free, it can start sending, if not, it must wait a random amount of time before checking again. For each retry an exponential back off algorithm will be used. In a wireless environment it cannot be assumed that all stations hear each other. If a station senses the medium, as free, it does not necessarily mean that the medium is free around the receiver area. This problem is known as the hidden terminal problem and to overcome these problems the collision avoidance mechanism together with a positive acknowledgment scheme is used. The positive acknowledgment scheme means that the receiver sends an acknowledgment when it receives a packet. The sender will try to retransmit this packet until it receives the acknowledgment or the number of retransmits exceeds the maximum number of retransmits. 802.11 also support power saving and security. Power saving allows packets to be buffered even if the system is asleep. Security is provided by an algorithm called Wired Equivalent Privacy (WEP). It supports authentication and encryption. WEP is a Pseudo Random Number Generator (PRNG) and is based on RSA RC4. One of the most important features of 802.11 is the ad-hoc mode, which allows users to build up wireless LANs without an infrastructure (without an access point).

Address Resolution Protocol

The Address Resolution Protocol, ARP is implemented. ARP translates IP-addresses to hardware MAC addresses. This takes place before the packets are sent down to the MAC layer.

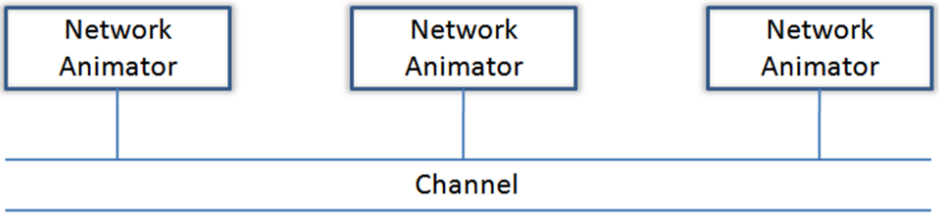

Figure 6.2: Diagram showing design of physical layer shared by NS2 simulator

Radio Network Interfaces

This is a model of the hardware that actually transmits the packet onto the channel with a certain power and modulation scheme.

Transmission Power

The radius of the transmitter with an omni-directional antenna is about 250 meters in this extension. Different antennas are available for simulations.

Shared Media

The wireless extensions in ns2 are based on a shared media model (Ethernet in the air). This means that all mobile nodes have one or more network interfaces that are connected to a channel. A channel represents a particular radio frequency with a particular modulation and coding scheme. Channels are orthogonal, meaning that packets sent on one channel do not interfere with the transmission and reception of packets on another channel. The basic operation is as follows, every packet that is sent (i.e. put on the channel) is received (i.e. copied to all mobile nodes) connected to the same channel. When a mobile node receives a packet, it first determines if it is possible for it to receive the packet. This is determined by the radio propagation model, based on the transmitter range, the distance that the packet has traveled and the amount of bit errors. The conceptual diagram is shown in Figure 6.2.

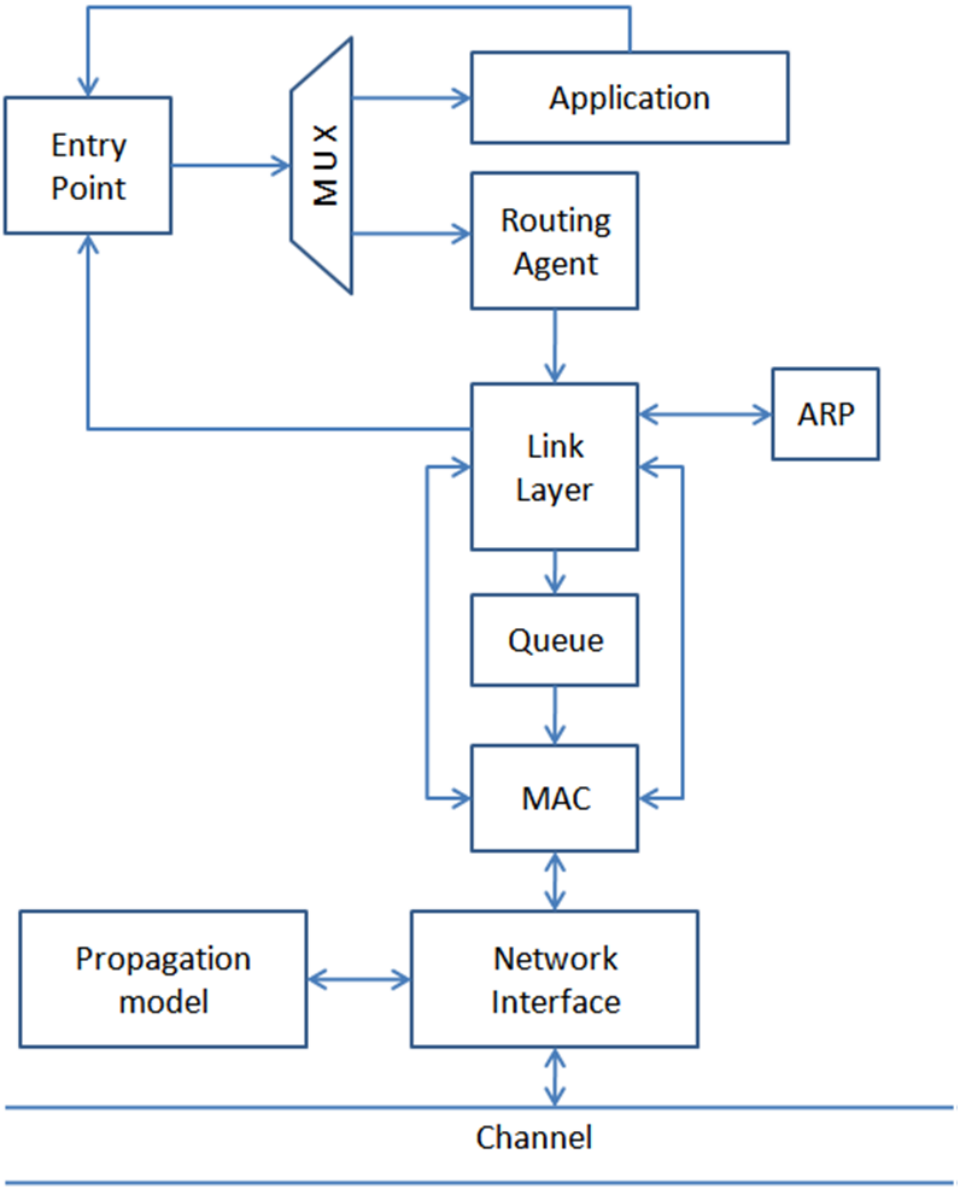

Figure 6.3: A model of mobile node in NS-2

6.1.3 Mobile Node

Each mobile node makes use of a routing agent for the purpose of calculating routes to other nodes in the ad-hoc network. A system model of mobile node is shown in Figure 6.3. Packets are sent from the application and are received by the routing agent. The agent decides a path that the packet must travel in order to reach its destination and stamps it with this information. It then sends the packet down to the link layer. The link layer level uses an Address Resolution Protocol (ARP) to decide the hardware addresses of neighboring nodes and map IP addresses to their correct interfaces. When this information is known, the packet is sent down to the interface queue and awaits a signal from the Multiple Access Control (MAC) protocol. When the MAC layer decides it is ok to send it onto the channel, it fetches the packet from the queue and hands it over to the network interface which in turn sends the packet onto the radio channel. This packet is copied and is delivered to all network interfaces at the time at which the first bit of the packet would begin arriving at the interface in a physical system. Each network interface stamps the packet with the receiving interfaces properties and then invokes the propagation model. The propagation model uses transmit and receive stamps to determine the power with which the interface will receive the packet. The receiving network interfaces then use their properties to determine if they actually successfully received the packet, and send it to the MAC layer if appropriate. If the MAC layer receives the packet error-free and collision-free, it passes the packet to the mobiles entry point. From there it reaches a multiplexer, which decides if the packet should be forwarded again, or if it has reached its destination node. If the destination node is reached, the packet is sent to a port de-multiplexer, which decides to what application the packet should be delivered. If the packet should be forwarded again the routing agent will be called and the procedure will be repeated.

6.1.4 Simulation Overview

A typical simulation with ns and the mobility extension is shown in Figure 6.4. Basically it consists of generating the following input files to ns:

1. A scenario file that describes the movement pattern of the nodes.

2. A communication file that describes the traffic in the network.

These files can be generated by drawing them by hand using the visualization tool or by generating completely randomized movement and communication patterns with a script. We used the scripts provided as part of ns2 since we found this method faster. These files are then used for the simulation and as a result from this, a trace file is generated as output. Prior to the simulation, the parameters that are going to be traced during the simulation must be selected. The trace file can then be scanned and analyzed for the various parameters that we want to measure. This can be used as data for plots with for instance Gnu plot. The trace file can also be used to visualize the simulation run with either Ad-hockey or Network animator.

We simulated Directed Diffusion with Congestion control over NS 2 version 2.29. NS-2 has implemented the basic directed diffusion. We compare our protocol with the basic directed diffusion to get an idea of how much better/worse our protocol performs in different scenarios. To get an idea of the performance in varying conditions, we run the simulation by varying parameters like network size, data rate and source-sink pair positions. We compare our protocol and basic diffusion over different network sizes of 100, 144, 196, 256, 324 and 400 nodes configured in the rectangular area. For simulations in which we keep network size constant, the distance between any two nodes is always 100m. The size of packets is 500 bytes, greater than double of the smallest video frame which is 187 bytes. We use a deadline of 200ms because the playout deadline varies from 50ms to 200ms. The deadline for the exploratory packets is 2000ms. We measure the performance of 12 simulation runs with randomly generated seed. The simulation time of each run is 1000s in ns2. We compare

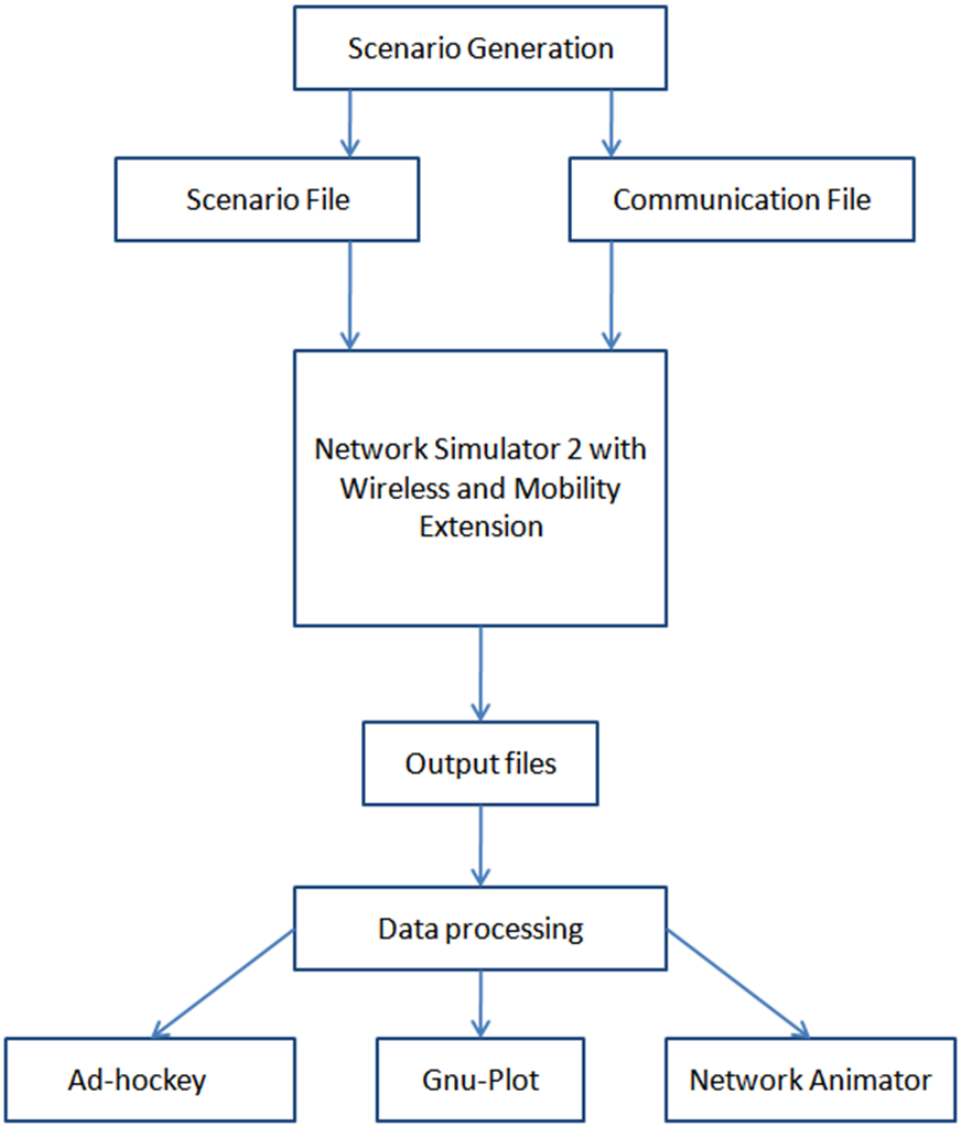

Figure 6.4: A system design of Network Simulator to get input and generate output.

throughput (packets per second), end-to-end delay (ms) and energy usage. We also show the way queues build up during simulation. The simulations were conducted on an Intel(R) PC with a Xeon(TM) processor at 1700 MHz, 512 Mbytes of RAM running Fedora Core 4.0.

Validation of simulation setup is an important part of simulation. In this thesis, we use NS-2 to implement protocol details. To verify if we have implemented the protocol correctly, and the code does not have memory leaks, or unreferenced pointers, we make use of DDD debugger. We have tried our best to eliminate any memory leaks and make sure that code is correctly written. We have used the NS-2 trace files to verify if the simulation setup has been correctly implemented. For a given scenario, we trace the exploratory packets to trace the routes which have been established during the routing phase. We then trace the routing of data packets to see if data is following the same route which was selected by the source. This will prove that we have found route correctly and are using one of the routes to transmit data. While checking exploratory packets, we also check for exploratory packets which are being sent to setup alternate routes. When data transmission starts, and sink detects sign of congestion, it will send a SWITCH_PATH_MESSAGE to source. We scan the trace files for this SWITCH_PATH_MESSAGE. We note down the time at which SWITCH_PATH_MESSAGE reaches source. This message tells source to start transmitting data using an alternate path. We then look for data flowing through new path. We check the timestamp of first data packet and compare it with timestamp of SWITCH_PATH_MESSAGE to see if the data transmission began immediately after receiving SWITCH_PATH_MESSAGE. This will prove that the data is flowing through new path, and was triggered by the control message. We also check nodes on old path to make sure that data is no longer transmitted using this path. This proves that SWITCH_PATH_MESSAGE mechanism has been implemented correctly. We also check for number of times SWITCH_PATH_MESSAGE is sent across the network and see if multiple SWITCH_PATH_MESSAGE's maintain the network state. We have verified the working of MULTI_PATH_MESSAGE in a similar way by scanning trace files for MULTI_PATH_MESSAGE and making sure that data is flowing through both the paths.

We check the sequence number on data packets to make sure that data is being divided on two paths and not just being duplicated. All these validation steps are used to ensure proper working of the implemented protocol before we begin to collect results.

Metrics

Before we get into actual simulation results, we will discuss the metrics we want to vary to get an estimate of the performance of our algorithm. We will measure throughput, delay, energy and queue length of the nodes on active paths.

Parameters

The metrics has to be measured against some parameter that describes the characteristic behavior of an ad-hoc network and can be varied in a controlled way. One of the advantages of simulation is that it allows us to do so. To get an idea of the protocol we will change following parameters in the network:

1. Network size: It is the number of nodes and the geographical size of the area that the nodes are moving within. The network size basically determines the connectivity. Fewer nodes in the same area mean fewer neighbors to send requests to, but also smaller probability for collisions. To see how our protocol performs in large networks, as against small networks, we test our protocol on various network sizes.

2. Data Rate: To check the load handling capacity, we check our protocol on varying loads.

3. Number of Source-Sink pairs: We also try to keep the data rate constant but increase number of source-sink pairs. We try to implement as many combinations of source sink pairs as possible from the above shown scenarios.

We can also change the packet size and get an idea of network load. In our case we do not change packet size.

6.2 Results

We simulated the basic Directed Diffusion over NS-2 simulation with different network sizes and throughput. We also made a few changes to basic directed diffusion which comes with NS-2 to implement our protocol. The changes we made to basic directed diffusion include: adding a new packet type to incorporate switch_path_message and multi_path_message, finding multiple routes, changing path metric. We added a filter which acts as queue between two_phase_pull and gradient filter, which will store incoming data packets and forward it to two_phase_pull for processing. This filter acts as a priority queue and creating multiple queues for multiple applications to store respective priorities of those applications. Priority queue is implemented using a pre-gradient filter and a post-gradient filter. Each variant essentially performs the three steps.

1. Add priority information to packets at sink.

2. Extract priority information from packets at source.

3. Compute priority and append it to outgoing data packets.

The priority is stored in the priority attribute by the source and used to prioritize the data packet at each intermediate hop. The source also timestamps the packet so that intermediate nodes can compute the elapsed time. Upon receiving a data packet, intermediate nodes extract the location of the sink and the timestamp. They use this information along with their location to update the priority again using equation:

$$V = \frac{||p_i - p_{dst}||}{T_{deadline} - T_i} \tag{6.1}$$

We also added two extra types in the message structure to include the control message types viz. switch_path_message and multi_path_message. Once we made all the changes

required to implement our proposed protocol in directed diffusion we have done a number of experiments to assess the reliability of our protocol. We have measured different metrics which according to us are a good measure robustness and reliability of the proposed protocol. These metrics give us an idea of how the protocol behaves in certain scenarios such as congestion. As explained above there are different scenarios where congestion occur. Each of these scenarios have varying magnitude and affect the overall performance in a different ways. In this chapter we will describe how the simulations were done.

We have done different types of simulations:

- Offered load simulations: we vary the load that we offer the network, to see how the protocols behave when for instance the load is high.

- Network size simulations: we vary the number of nodes in the network. The geographical area itself was fixed in order to make sure the density of nodes is varied.

In all our simulations we have used a grid topology, where each node will have 4 neighbors except for the nodes lying on the edge of the grid who will have 3 neighbors. To avoid these nodes to be considered in path formations, we have selected source - sink pairs in such a way so that they lie in the grid such that they do not have to rely on edge nodes to find path. We do not use randomized scenarios because we want to know where the source-sink are present for each run. For a given set of parameters we make 12 runs. We want our simulations to represent all the 3 scenarios we explained earlier. We have a fixed source - sink position which will guarantee us that the network is in one of the 3 explained scenarios. We make 4 runs on each of these 3 scenarios for given metrics. Then we take an average of these 12 runs and use those values as a measure of throughput or delay.

The reason we consider 12 runs for a given set of parameters is because we want to prove that our simulations are not seed specific, and results for each seed are consistent with the other. In case of our simulations, we also want our simulations to be able to represent all the 3 different scenarios explained above. If we select 12 random seeds, the path formation

for these seeds cannot be predicted. 12 random seeds may or may not cover all the scenarios in forming path structure. We try to simulate our protocol over 3 scenarios to prove the point that our protocol is not dependent on path formations, between source-sink pair. It performs equally well all the three scenarios, giving consistent results. The deviation of values for throughput, delay and average energy for all the 12 seeds is very little, and well within the range of acceptable deviation. Hence, taking an average of all these values, which fall within the same range, will give us an estimate of overall performance of network, instead of scenario specific performance. As all the parameters, except random seed are same for all the simulation runs, we can say that these 12 runs are similar to each other, and prove the same metric. Hence taking an average of throughput/delay/energy for these 12 runs will be a more appropriate way of representing the protocol behaviour. Each of the graphs shown below for proposed protocol (DDCC) have a 'I' which is a indicator of variations in values for a particula parameter over three different scenarios for 12 different seeds in NS-2.

The parameters that can be changed are:

- Maximum speed: Every time a speed is going to be randomized, it is randomized in the interval [0,maximum speed]. For most of our simulation we have not used the randomized feature. We manually set the packet rate of a node to transmit constant bit rate traffic. In our simulations, randomized traffic did not generate high amount of congestion and hence it was tough to test our protocol. Hence, for our convenience we have decided upon using a constant traffic rate so as to be sure that congestion occurs in network.

- Number of nodes: This was constant during one set of simulation runs. We used 100 nodes for all simulation except the size simulation where we varied the number of nodes. This parameter determines the density of mobile nodes in a given area of dimensions 1000 x 1000 meters.

Simulation Parameters	
Parameter	Value
Transmitter range	250.10 meters
Bandwidth	2 Mbps
Simulation time	900 s
Number of nodes	100
Environment size	1000m X 1000m
Traffic type	Constant Bit Rate
Packet rate	200, 100, 67, 50, 25, 10
Packet size	500 Bytes
Simulation Runs	12

Table 6.1: Simulation Parameters for Offered Load simulations

- Simulation time: The time for which the simulations will be run. We have used a simulation time of 900 seconds for all simulations. We calculated the metric considering the initial route discovery phase, as well as the intermediate route discovery phase where it would find new paths.

6.2.1 Simulation Parameters for Offered Load

Table 6.1 shows the simulation parameters and setup configuration for simulations in which we change offered load and observe the performance of the protocol.

6.2.2 Offered Load Simulation

We compare the proposed protocol with traditional directed diffusion in terms of throughput, delay and energy consumption. The first graph compares the throughput of directed diffusion to proposed protocol. Figure 6.5 shows throughput vs data-rate for a fixed network size of 100 nodes. This includes results from 12 runs: consisting of 4 runs each for each of the 3 different scenarios explained in previous section. We vary data rate through a range of 200, 167, 125, 100, 67, 40 packets/sec. The graph for directed diffusion shows a rise in throughput till data rate of 100 packets/sec, after which there is a fall in throughput. This point is called as knee point, and is a point after which effects of congestion are seen. This

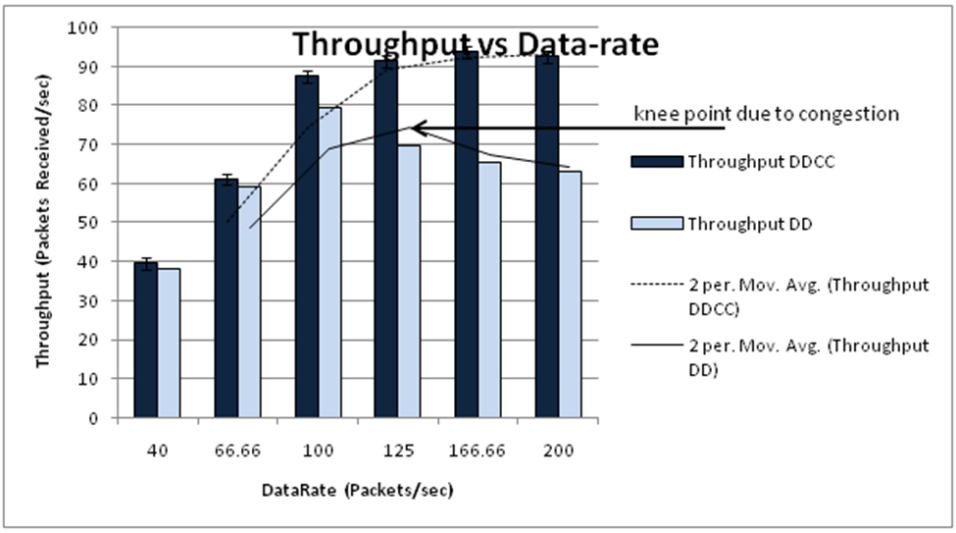

Figure 6.5: Throughput vs Datarate.

knee point indicates that network is congested, and packet drop rate has increased. This results in drop in throughput for higher data rates. As opposed to directed diffusion, the proposed protocol handles this increase in data rate pretty efficiently. The delivery ratio for the proposed protocol very stable as compared to directed diffusion. There is a very slight change in throughput of proposed protocol at very high data rates. But this change is very insignificant as compared to directed diffusion. There is a very little drop in throughput in proposed protocol at the data rate of 200 packets/sec as compared to throughput at 167 packets/sec.

Also, the knee for proposed protocol is very subtle, and it tries to keep the data rate at a constant 93 packets/sec. As we mentioned earlier, we did 4 simulation runs for each of the 3 different scenarios. Then we get average throughput for all the 3 different scenarios. In case of very high data rate, and a complete crossing path scenario, there is very high congestion, and proposed protocol needs to switch path multiple times before it decides upon a using multiple paths. This limits the maximum throughput of the system to less than 100 packets/sec. Delay is also a good measure of reliability and effectiveness of the protocol. Figure 6.6 shows a graph of Delay vs Data-rate for a fixed network size of 100

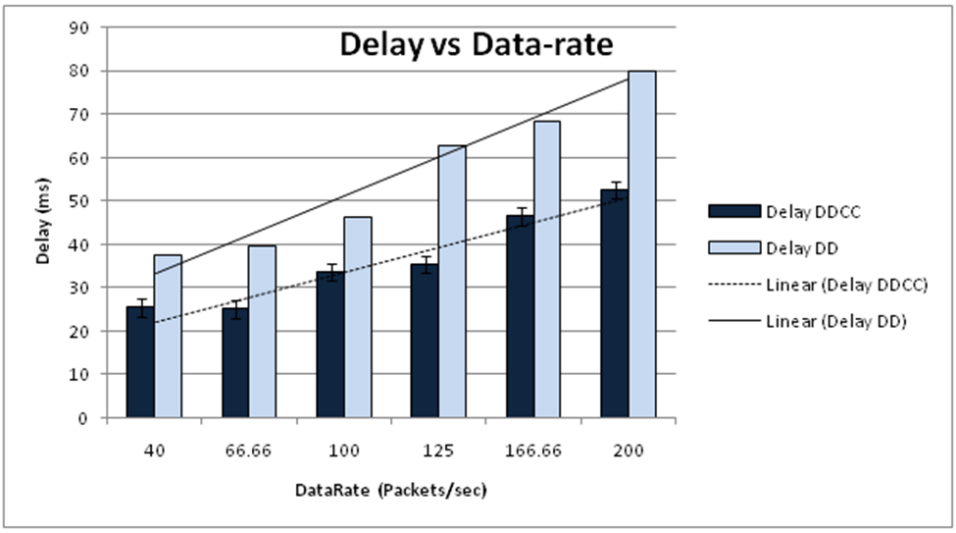

Figure 6.6: Delay vs Datarate.

nodes. This also includes results from 12 runs: consisting of 4 runs each for each of the 3 different scenarios explained in previous section. As we can see in Figure 6.6, delay gradually increases as data rate increases. As the data rate increases, the number of packets flowing through wireless medium increase. This might cause interference, as well as congestion on the network. This will result in causing more delay in packet delivery. Both the proposed protocol, and directed diffusion show a gradual increase in delay due to this reason. But the rate of increase in proposed protocol is less than that of directed diffusion. Also, the effective delay for directed diffusion is much more that that of proposed protocol. This can be caused due to many reasons.

The proposed protocol uses multiple metrics while setting up paths. These metrics are ETX, delay and average queue length. ETX helps us select a path which is more reliable, and has low SNR ratio. Delay metric also plays an important role of selecting a path with minimum delay, but it is combined with ETX. In addition to this queue length helps us select a path over nodes which have lower queue lengths. This might mean that those nodes do not participate in path formation with other source-sink pair. We can never be sure of this, because path formation also depends on the topology of network. All these three metrics

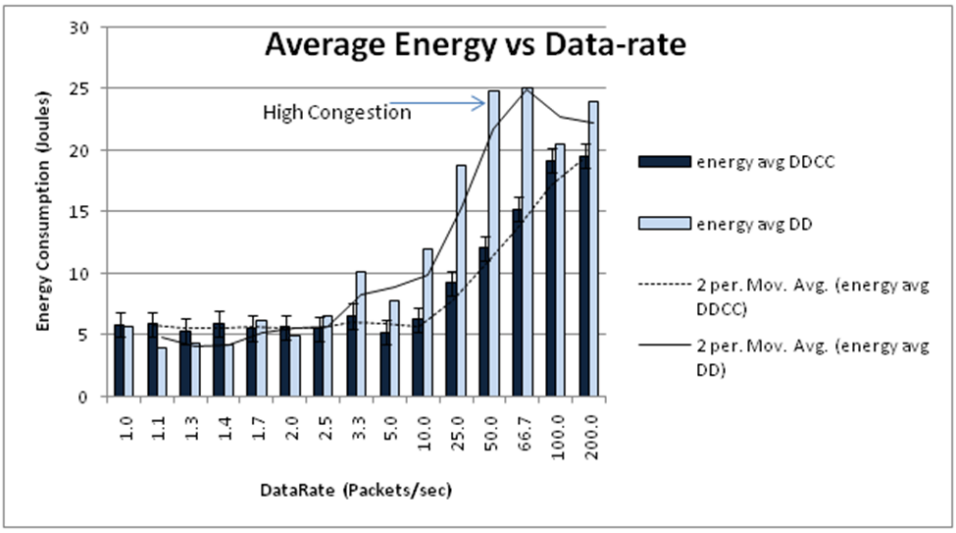

Figure 6.7: Average Energy Consumption vs Datarate.

help proposed protocol select a path which will provide low error rate, low delay and less interference. This means that the effective delay of packets transmitted over this path is lower. Hence, our protocol performs better than diffusion in terms of end-to-end delay.

Energy consumption during data transfer is an important metric for measuring congestion. When there is rise in network congestion, the performance of network starts degrading. In wired protocols, like TCP, packet drop is a sign of congestion. When a node (receiver) senses that the packets are being dropped, it will run a backoff algorithm, which will tell the sender to reduce the data sending rate. This strategy does not work in wireless networks. In wireless ad-hoc networks, there can be many reasons for packet loss like link quality, interference, and packet collision. Due to which ad-hoc network protocols cannot incorporate any such schemes to limit congestion. As a result even in case of network congestion, the sink will still keep sending data at a constant rate, thus worsening congestion condition. This will only result in packet retransmissions, interference, which in turn will result in more energy being used by a node for sending data. Hence energy consumption can give us an idea of how network copes in times of congestion. Also, in ad-hoc networks, energy is a limited resource

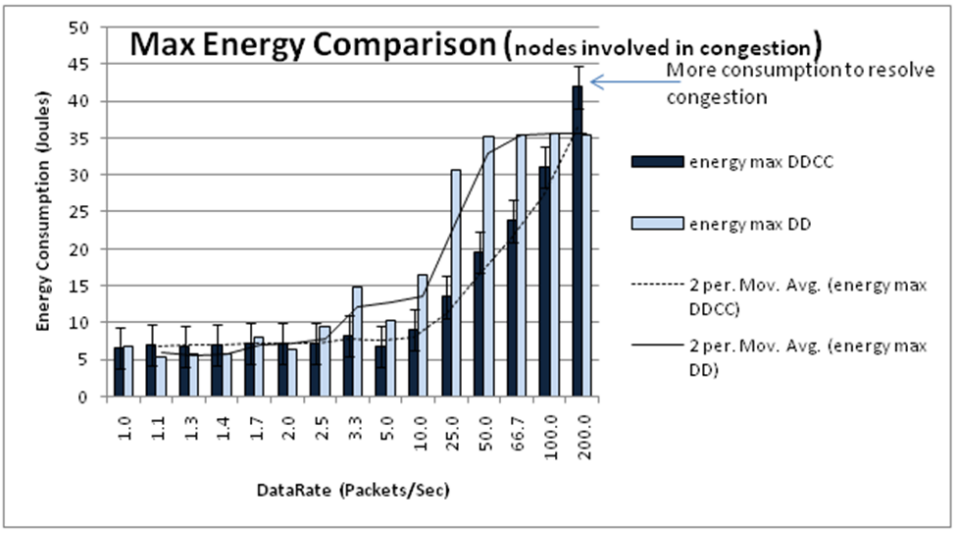

Figure 6.8: Maximum Energy Consumption vs Datarate.

and should be used efficiently. So, we measure energy used by our proposed protocol against directed diffusion.

Figure 6.7 shows average energy consumption of network for different data rates and uniform network size of 100. This includes results from 12 runs: consisting of 4 runs each for each of the 3 different scenarios explained in previous section. Average energy denotes the energy consumed by all the nodes throughout the network. It gives an estimate of energy used by nodes to forward interest, exploratory packets, as well as the energy consumed by nodes to transmit data. We vary the data rate of the source - sink pairs to have values ranging from 200, 100, 67, 50, 25, 10, 5 etc. We also calculate energy consumption for varying data rates below 5 packets/sec. We measure energy consumed in Joules. At very high traffic rates of 200 packets/sec, directed diffusion is not able to keep up with the generated traffic and just drops the incoming packets in the initial phase. But if we reduce the data rate to 100 or 66 packets/sec, then the data rate is just high enough to cause congestion. But in this case diffusion can barely try to transmit most of the packets. This will cause more energy consumption due to the effort of trying to achieve high throughput. But at the same time there is congestion in network, which will result in retransmissions, and drops. This will

increase delay and also reduce throughput as shown in the previous graphs. When the data rate decreases, the total energy consumed gradually decreases to a point of 2 packets/sec. If we further lower the data rate we can see that the total energy consumption for proposed protocol is more or less constant, and does not fall below a certain value. At this point the total energy consumption is more than that of basic directed diffusion. We can say that this energy value is the minimum energy required by our protocol.

In DDCC (proposed protocol), we find multiple paths and in addition keep track of some metrics throughout the data transmission phase. This will consume some minimum amount of energy. In Figure 6.7 , we can see that once data rate drops below 2 packets/sec, DDCC still consumes some minimum amount of energy each cycle. But directed diffusion energy consumption is lower than DDCC. This means our protocol benefits outrun those of directed diffusion until we hit a data rate of 2 packets/sec. If we drop below that, then directed diffusion consumes less energy. Once we start increasing the data rate, the energy consumed by nodes running DDCC gradually increases. But the total energy used is still less than the node which run on directed diffusion, which is to say that the rate of increase with DDCC protocol is comparatively less than directed diffusion. Also, at a very high data rate of 200 packets/sec, our protocol still performs well enough consuming less average energy than directed diffusion. Average energy is a representation of energy consumption for regular traffic flow.

In our case, we are more concerned about congestion in network, and the energy consumed to overcome that. We propose to measure maximum energy consumed by a sensor, in the network to get an estimate of that. This means, in case of the third scenario, where two source - sink pairs are across each other in the network, the node which lies at intersection of two paths will consume maximum energy and it is the point where congestion occurs. Figure 6.8 shows maximum energy consumption of network for different data rates and uniform network size of 100. To get an accurate estimate of maximum energy consumption, we have used only scenario 3 for simulation. To plot each point on the graph, we made 7

runs for a given data rate providing different random seed to the simulation. For each run have to determine the node which acts as intersecting node, and measure energy for that particular node. Normally there will be two nodes which are intersecting nodes for each of the two paths. To get the total energy consumed, we will have to take into account the energy of the node which is currently active. If both the nodes are active, then we will have to add up the energy consumption of two nodes for this interval to get the total energy consumed. This maximum energy can act as a measure of amount of energy used while trying to resolve congestion. When we consider Figure 6.8, we can see that max. energy consumed by one of the nodes at data rate of 200 packets/sec is very high. The justification for this is that, when the data rate is very high, as explained in scenario, queues will build up at the intersecting node. This will cause multiple switch - path messages to be sent across the network. In addition to that, we also use a multipath message, causing both the intersecting nodes to be used simultaneously. Hence, both nodes transmitting data will be using energy simultaneously. Due to this factor, the energy consumed by DDCC is more as it adds up the energy consumed by two nodes in that period of time. But as we reduce the data rate, the maximum energy consumed by DDCC is far less that that used by directed diffusion. Directed diffusion has only one path from source - sink, and once that gets congested, it is going to try to send data through the same path. This will result in high packet loss rate, as well as low throughput. This can be deduced from throughput graph shown in Figure 6.5 and the energy graph.

If we look closely at Figure 6.8, we can see that the range in which values lie for each parameter on the graph is larger than the range for other graphs explained above. The only reason for a large range of values for maximum energy consumption graph is that these are sepcifically designed for scenario 3. In this scenario, there are multiple nodes which are intersecting point for the two paths. The energy consumtption will be sum of nodes falling on intersection of two paths. Due to this the number of nodes simultaneously used for data transmission is more for scenario 3 than for any other scenario. Because of this the energy

Simulation Parameters	
Parameter	Value
Transmitter range	250.10 meters
Bandwidth	2 Mbps
Simulation time	900 s
Number of nodes	100, 144, 169, 256, 324, 400
Environment size	1000m X 1000m
Traffic type	Constant Bit Rate
Packet rate	100 packets/sec
Packet size	500 Bytes
Simulation Runs	12

Table 6.2: Simulation Parameters for Network Size simulations

consumption given by this will be greater than energy consumption by other 2 scenarios. Hence there is a variation in energy values. But we consider only scenario 3 for plotting this graph. Hence the values under consideration in this graph are fairly consistent and have a slight standard deviation. The I in this graphs is representation of range of values for all 3 scenarios.

6.2.3 Simulation Parameters for Network Size

Table 6.2 shows the simulation parameters and setup configuration for simulations in which we change network size and observe the performance of the protocol.

6.2.4 Network Size Simulation

In addition to compare the proposed protocol by varying load, we also compare the protocol by varying the network size. The purpose of comparing it with varying network size is that we can get an idea of how scalable our protocol is. Number of node in the network is a very important factor in adhoc networks. If number of nodes in a given area is increased, then node density increases. This means that the links between nodes are stronger because the signal strength between two nodes is going to be strong. At the same time, as a node will be surrounded by many other sensors, there will be interference that is caused by signal

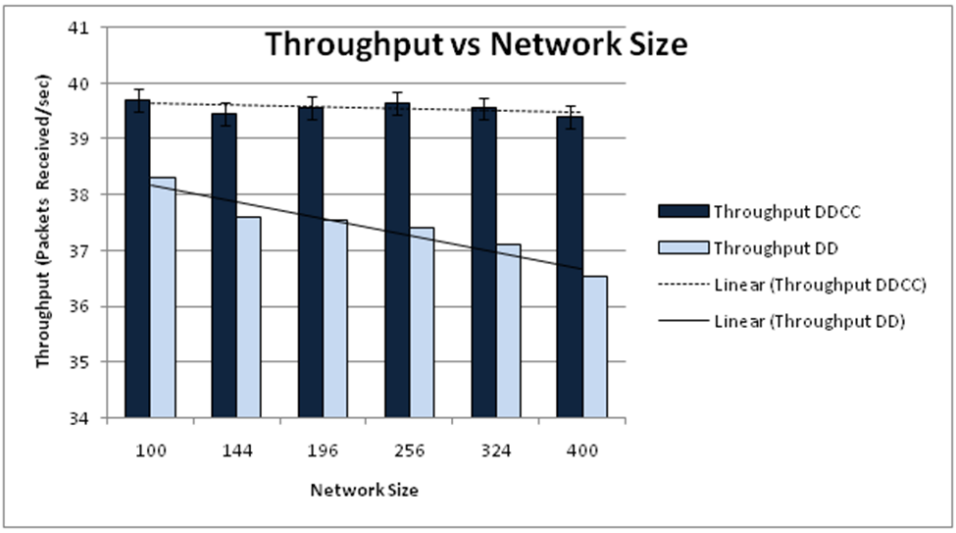

Figure 6.9: Throughput vs Network Size.

strength due to those sensors. Also, if for a given area, we decrease the number of nodes in that region, then connectivity between those nodes will be weak. The main reason for that is because nodes will be farther from each other. This will mean that the signal between 2 nodes will have to travel longer distance, resulting in weak signals. We need to test our protocol in both the cases where there is high density of nodes, with high interference, as well as in scenarios where there is low connectivity.

In our scenario, we run the simulation 12 times. We run the simulation 4 times, with same setup for each of the 3 different scenarios. Then we take an average of these readings to get an estimate of the network performance for a given scenario. The first graph compares throughput with the network size. We run the simulation with constant parameters of data rate and bandwidth. We change the network size and observe the variations in throughput. Figure 6.9 shows the effects of throughput on network size. For the proposed protocol, DDCC, the throughput more or less remains constant with network size. This means that the protocol handles the issue of scalability very nicely. There might be some delays associated with the end-to-end transmission time of control messages like switch_path_message and multi_path_message. But the simulations show that these delays are very less, mainly because

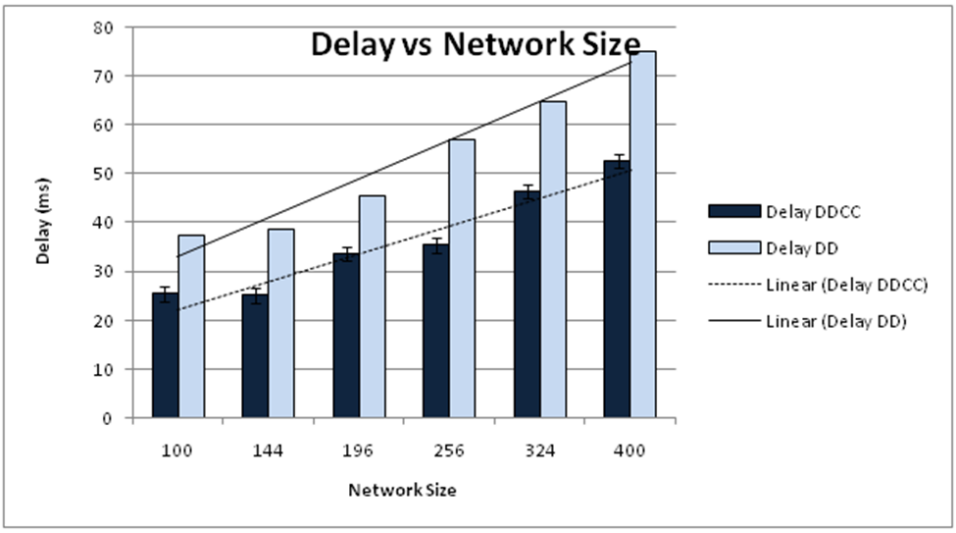

Figure 6.10: Delay vs Network Size.

the packets transmitted are in a priority queue and best effort is made to deliver those packets. Also, in case of congestion, most of the packets will be sent via alternate path once we detect congestion. Hence, we are able to keep throughput constant irrespective of network size.

The graph in Figure 6.10 shows simulation results for delay against network size using same parameters. We calculate average delay by finding the transmission time for each packet and getting an average of these to find effective delay throughout the transmission. As the number of nodes increases, the interference between nodes increases. This might cause the packets to be retransmitted. Also, as the number of nodes increase, the number of transmission over a path increases. This might cause the end-to-end delay to increase.

In addition to measuring throughput and delay, I have also measured the energy consumption against network size. We will first see the average energy variations with increasing network size. Figure 6.11 shows that average energy consumed by each node decreases with network size. When directed diffusion finds paths between source - sink, it just relies on delay. It does not consider the link quality while deciding upon path. This affects the performance of protocol when number of nodes in the network are less. As node density is less,

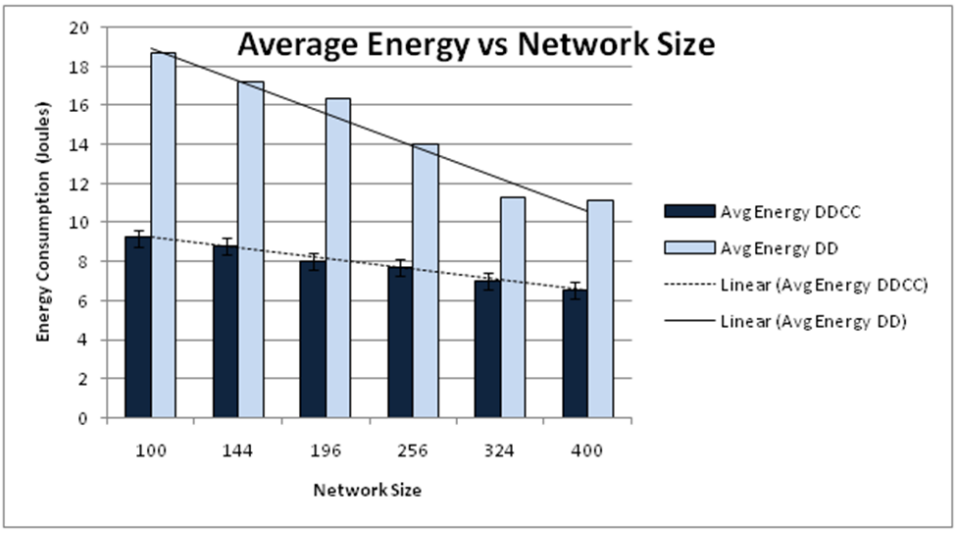

Figure 6.11: Average Energy Consumption vs Network Size.

all the links between nodes might not be able to transmit data with same reliability. Due to this, when the number of nodes is 200, the amount of energy used by directed diffusion is very high. The reason is that the paths which have been selected on delay metric are not the best ones, and do not perform well during data transmission. As the network density increases, number of nodes increase. This will shorten the wireless links between nodes. This also means that the path selected based on delay might perform well as the wireless between nodes is going to be strong. Hence, the transmissions are more reliable and will consume less energy per node. In case of the proposed protocol, DDCC, we try to select the best available path. So even if the node density in network is less, we select a path which is most reliable, and has least SNR. This ensures that there are less retransmission over the wireless link. Hence the energy used is less than that required by directed diffusion. As DDCC selects the best available path, the energy used in network remains relatively unchanged. There is a slight decrease in energy consumed when the network density is higher. This brings up a peculiar characteristics of wireless network. As the network density increases, end-to-end delay might increase, but the energy consumed for transmission decreases.

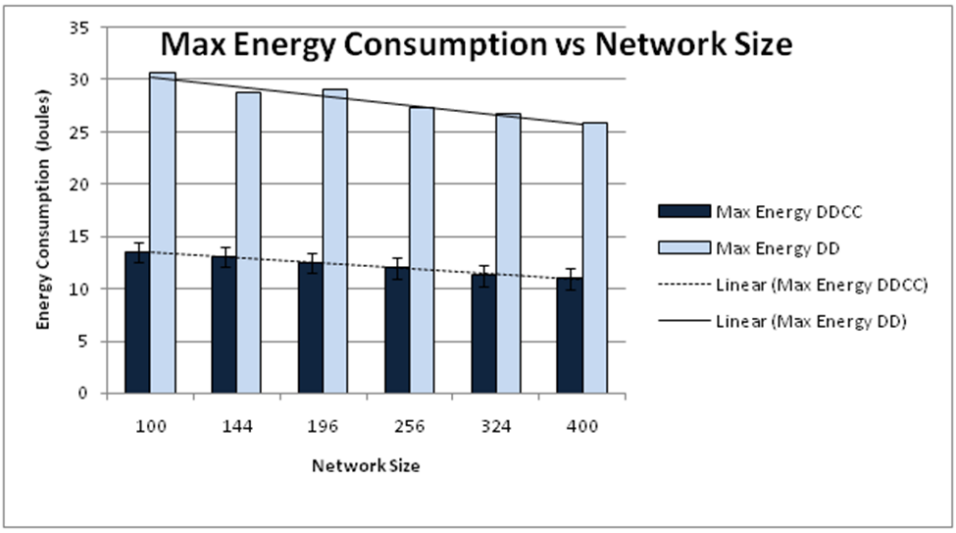

Figure 6.12: Maximum Energy Consumption vs Network Size.

The last graph, shown in Figure 6.12, shows the effect of maximum energy on network density. As explained earlier, the maximum energy consumption is mainly a measure of the congestion. For a packet rate of 100 packets/ second, the amount of congestion over the node decides the maximum energy consumption of a node. Congestion over the node at given data rate is not too high. We use the same technique we used earlier to calculate maximum energy consumption of a given scenario. We will use scenario 3 for simulation, and would calculate the energy consumption of only the nodes lying on intersecting paths. We can see that if the average energy consumption is around 8 Joules per minute, then maximum energy consumed is around double of it, which is 16 joules per minute. The maximum energy consumed remains more or less constant for DDCC. This is mainly because the congestion level over the nodes is not too high. Also, congestion resolution takes place on end-to-end basis. But there is a gradual decrease in the maximum energy consumed, which is due to the fact that due to more node density, the transmission energy consumed is slightly lower. Even with directed diffusion there is a slight decrease in maximum energy consumed, which can be attributed to the fact that the links are shorter, which might reduce the energy consumed during data transmission.

Chapter 7

Conclusion and Future Work

We have proposed Directed Diffusion with Congestion Control (DDCC) protocol,which tries to prevent congestion in wireless ad-hoc networks. In scenarios where the data rate is very high and congestion cannot be prevented, it tries to avoid congested region by sending data via alternate path. In contrast to directed diffusion which is severely affected by data rate and network size, DDCC is more consistent under varying topologies and configurations. Directed diffusion fails in cases of high load, and is very inefficient in terms of energy and end-to-end delay. DDCC incorporates a efficient path finding algorithm which helps to find paths which can handle high load than paths used by Directed Diffusion. We do not use source data rate limiting technique in event of congestion. This enables the source to transmit important data even in the state of congestion, with very low packet losses without increasing end-to-end delay. DDCC collects statistics over the active path and tries to determine whether the network path is in state of congestion. When it concludes that the path is getting congested, DDCC will have source send data through an alternate path. To implement this we use an end-to-end congestion detection and recovery technique. By selecting an alternate path, we achieve two important things. Firstly, we avoid congested region, due to which congestion situation will not worsen in the region. Secondly, we select an alternate path, which enables us to get transmit data with lower delay without affecting the data rate.

We have tried to overcome many drawbacks of directed diffusion. We have made the path finding algorithm more efficient enabling it to consider multiple metrics while selecting a path. We also find multiple paths, without using significant resources. As we broadcast packets throughout the network for path discovery, it is an expensive process, and running path discovery algorithm in event of congestion or route failure will worsen the network state

and performance. Instead we make use of multiple acknowledgement packets we receive at destination due to use of broadcast mechanism to find multiple, but disjoint paths. These paths act as backup routes in case of congestion or path failure. We collect path statistics on a regular interval. We use these statistics to see how route is performing, and whether there is congestion or bottleneck on any routes. Congestion state arises when the data flowing through network is more than it can handle. The network is in a critical state, because it is transmitting important data, but the resources which are available to do so are not sufficient. Any preventive/corrective actions in this case should have a primary objective of not incurring any extra cost or using excessive resources. We achieve that by having backup routes and activating them in such critical situations, with only one extra message. Multiple paths help us to switch paths in event of congestion incurring no extra route discovery cost. Also, the energy consumed in switching path is far less than energy saved by switching path to an efficient less congested route, which provides better data delivery. The advantage of switching path is we avoid congested region, and this enables us to achieve high data rates and lower delays.

We have proved the efficiency of our protocol using simulation. We implemented the protocol in NS-2 simulator. NS-2 has a implementation of basic directed diffusion. We incorporated the changes into the basic diffusion to implement our protocol. We created scenarios which represent all the states a network can be in, when there are two source - sink pairs. We made simulation runs for each of these scenarios to see how our protocol performs for varying path formation combinations. To get an estimate of scalability and stability of our protocol, we ran simulations for sufficiently long duration. We made multiple runs of simulation for same scenario with different random seed to the NS-2 simulator, to get statistics for varying conditions. We then took average of these values to get a more accurate representation of performance.

From the resulting graphs, we can see that there has been a significant rise in throughput. For our simulations, we have tested the two protocols for a data rate as high as 200

packets/sec. For this data rate the throughput achieved by DDCC is 46% more than that of directed diffusion. We measure end-to-end delay for various topologies varying network size and data rate. The delay achieved by DDCC can be as low as 40% than that achieved by directed diffusion. As we are concentrating on congestion, energy consumed during data transfer is an important factor to prove efficiency of our protocol. Our protocol can be 52% more energy efficient than directed diffusion in average scenarios. In case of few scenarios, we also compare the maximum energy consumed by a node during data transfer, which gives an estimate of energy consumed by a congested node. We can see that a congested node running on DDCC consumes 75% less energy than a node running on directed diffusion.

Thus, in this thesis, we have shown that DDCC performs better in all aspects than directed diffusion. There are many possibilities for future work in ad-hoc networks, and congestion control in wireless sensor networks. We have proved the efficiency by simulation, however verifying the same by actual experiment remains to be done. As I mentioned earlier, we can use the statistics collected on a route for route recovery. We have an algorithm in place which finds efficient routes which act as back up routes in case of congestion. As a part of future work we can come up with a mechanism to determine a node or link failure, and as a preventive action, we can switch to a back up route to transmit data. Also, we use an end-to-end congestion detection and recovery technique. End-to-End scheme has a few drawbacks in wireless ad-hoc network. Future work can also include implementing this technique using local recovery and path discovery scheme. Currently we have implemented the congestion control algorithm for wireless sensor networks. One of the future work scopes can be implement this scheme in SmartGrid networks. SmartGrid networks are highly dense networks with multiple links connecting every node. Congestion in such networks can adversely affect the network performance, and a light weight recovery technique in such networks will be very useful.

Bibliography

[1] 21 ideas for the 21st century. Business Week, pp. 78-167, August 1999.

[2] TIAN HE, BRIAN M. BLUM, JOHN A. STANKOVIC and TAREK ABDELZAHER Department of Computer Science, University of Virginia, AIDA: Adaptive Application-Independent Data Aggregation in Wireless Sensor Networks, ACM Transactions on Embedded Computing Systems, Vol. 3, No. 2, May 2004, Pages 426457.

[3] Chieh-Yih Wan, Andrew T. Campbell, Member, IEEE, and Lakshman Krishnamurthy, Pump-Slowly, Fetch-Quickly (PSFQ): A Reliable Transport Protocol for Sensor Networks, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 23, NO. 4, APRIL 2005.

[4] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govindan, Deborah Estrin, Data-Centric Storage in Sensornets, ACMS IGCOMM Workshop on Hot Topics in Networks (HotNets 2002), Princeton, NJ, October 2002.

[5] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In proceedings of the ACM MobiHoc Conference, 2003.

[6] C.Wan, S. Eisenman, and A. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks. In proceedings of the ACM SenSys 2003.

[7] Kang, J., Zhang, Y., Nath, B., Yu, S. (2004). Adaptive resource control scheme to alleviate congestion in sensor networks. In Proceedings of IEEE Workshop on Broadband Advanced Sensor Networks, 2004.

[8] Samuel Madden, Michael Franklin, Joseph Hellerstein, Wei Hong, TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks, OSDI 2002, December 2002,Boston MA.

[9] CHIEH-YIH WAN, SHANE B. EISENMAN, ANDREW T. CAMPBELL, JON CROWCROFT, Siphon: Overload Traffic Management for Sensor Networks, ACM Transactions on Sensor Networks, Vol. 3, No. 4, Article 18, Publication date: October 2007.

[10] C-Y. Wan, A. T. Campbell, and J. Crowcroft, A Case for All-Wireless Dual-Radio Virtual Sinks. In Proc. of the 2nd ACM Conf. on Embedded Networked Sensor Systems, pages 267 268. Baltimore, Nov 3-5 2004.

[11] J. Zhao and R. Govindan, Understanding Packet Delivery Performance in Dense Wireless Sensor Network, In Proc. Of the 1st ACM Conf. on Embedded Networked Sensor Systems, pages 1 13. Los Angeles, Nov 5-7 2003.

[12] Ramesh Govindan, Deborah Estrin, John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Directed diffusion for wireless sensor networking, IEEE/ACM Transactions on Networking (TON), Volume 11, Issue 1 (February 2003).

[13] Ramesh Govindan, Deborah Estrin, John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Directed diffusion: A scalable and robust communication paradigm for sensor networks.

[14] F. L. LEWIS, Introduction to Wireless Sensor Networks.

[15] P. Gupta and P. R. Kumar, The capacity of wireless networks, IEEE Transactions on Information Theory, vol. 46, no. 2, March 2000.

[16] Wu Xiuchao. Simulate 802.11b channel within ns2. Technical report, SOC, NUS.

[17] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. Wireless Network, 6(4):307-321, 2000.

[18] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking, pages 243-254, New York, NY, USA, 2000. ACM.

[19] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (dream). In MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, pages 76-84, New York, NY, USA, 1998. ACM.

[20] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks, Mobile Computing, 353, 1996.

[21] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on, pages 90-100, 25-26 Feb 1999.

[22] Zygmunt J. Haas and Marc R. Pearlman. The performance of query control schemes for the zone routing protocol. In SIGCOMM '98: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, pages 167-177, New York, NY, USA, 1998. ACM.

[23] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, page 1405, Washington, DC, USA, 1997. IEEE Computer Society.

[24] Wan, C.Y., Eisenman, S.B., Campbell, A.T.: CODA: Congestion Detection and Avoidance in Sensor Networks. In: the proceedings of ACM SenSys, Los Angeles, pp. 266279. ACM Press, New York (2003)

[25] Ee, C.T., Bajcsy, R.: Congestion Control and Fairness for Many-to-One Routing in Sensor Networks. In: the proceedings of ACM SenSys, Baltimore, pp. 148161. ACM Press, New York (2004)

[26] Wang, C., Sohraby, K., Li, B., Daneshmand, M., Hu, Y.: A survey of transport protocols for wireless sensor networks. IEEE Network Magazine 20(3), 3440 (2006)

[27] Sankarasubramaniam, Y., Ozgur, A., Akyildiz, I.: ESRT Event-to-Sink Reliable Transport in wireless sensor networks. In: the proceedings of ACM Mobihoc, pp. 177189. ACM Press, New York (2003)

[28] Md. Mamun-Or-Rashid, Muhammad Mahbub Alam, Md. Abdur Razzaque, and Choong Seon Hong, Reliable Event Detection and Congestion Avoidance in Wireless Sensor Networks.

[29] Chen, S., Yang, N.: Congestion Avoidance Based on Lightweight Buffer Management in Sensor Networks. IEEE Transaction on Parallel and Distributed Systems 17(9), 934946 (2006)

[30] Zhang, H., Arora, A., Choi, Y., Gouda, M.G.: Reliable Bursty Convergecast in wireless Sensor Networks. In: the proceedings of ACM MobiHoc, pp. 266276. ACM Press, New York (2005).

[31] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An empirical study of epidemic algorithms in large scale multihop wireless networks, 2002.

[32] K. Kar P. Chaporkar and S. Sarkar. Throughput guarantees through maximal scheduling in wireless networks. In Proc. 43rd Allerton Conf. on Commun., Control, and Comp., 2005.

[33] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu. Impact of interference on multi-hop wireless network performance. In MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking, pages 66-80, New York, NY, USA, 2003. ACM.

[34] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[35] Crossbow technology. http://www.xbow.com/. Accessed Jan. 2, 2010.

[36] Pc104.com. http://www.32.com/. Accessed Jan. 2, 2010.

[37] Dah-Ming Chiu and Raj Jain, Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. Computer Networks and ISDN Systems, 17, 1989.

[38] Ion Stoica, Scott Shenker, Hui Zhang, Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks, SIGCOMM 1998.

[39] C. T. Ee and R. Bajcsy, Congestion Control and Fairness for Many-to-One Routing in Sensor Networks, in proc. of ACM SenSys 2004, November 2004.

[40] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A Survey on Sensor Networks, IEEE Communication Magazine, August 2002.

[41] Jason Lester Hill, System Architecture for Wireless Sensor Network, PhD Dissertation, University of California, Berkeley, 2003.

[42] M.A.M. Vieira, C.N. Coelho. Jr., D.C. da Silva Jr., and J.M. da Mata, Survey on Wireless Sensor Network Devices, IEEE, 2003.

[43] J. Feng, F. Koushanfar, and M. Potkonjak, System-Architecture for Sensor Networks Issues, Alternatives, and Directions, ICCD02, 2002.

[44] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, S. Shenker, L. Yin, and F. Yu. Data-centric storage in Sensornets. In the first ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), 2002.

[45] Wei Wu, Zhongzao Zhang, Xeujha ,Autorate mac based congestion control protocol, Information Technology Journal 8 (8) : 1205-1212, 2009.

[46] Wang. C, B. Li, K. Sohraby, M daneshmand, and Y. Hu, 2007, Upstream congestion control in wireless sensor network through cross layer optimization. IEEE J. Select Areas Commun, 25: 768-795.

[47] Tao He, Fengyuan Ren, Chuang Lin, Sajal Das, Alleviating Congestion Using Traffic-Aware Dynamic Routing in Wireless Sensor Networks, IEEE SECON 2008.

[48] Md. Obaidur Rahman, Muhammad Mostafa Monowar, Byung Goo Choi, Choong Seon Hong, An Approach for Congestion Control in Sensor Network Using Priority Based Application.

[49] IEEE 802.11-2007, IEEE Standard for Information technology.

[50] Muhammad Mostafa Monowar, Md. Obaidur Rahman, Al-Sakib Khan Pathan, and Choong Seon Hong, Congestion Control Protocol for Wireless Sensor Networks Handling Prioritized Heterogeneous Traffic

[51] Yogesh G. Iyer, Shashidhar Gandham, S. Venkatesan, STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks.

[52] Cheng Tien Ee, Ruzena Bajcsy, Congestion Control and Fairness for any-to-One Routing in Sensor Networks.

[53] Xiaoqin Chen, Haley M. Jones, A .D .S. Jayalath, Congestion-Aware Routing Protocol for Mobile Ad Hoc Networks.

[54] Paulo Rogrio Pereira, Antnio Grilo, Francisco Rocha, Mrio Serafim Nunes, Augusto Casaca, Claude Chaudet, Peter Almstrm and Mikael Johansson, END-TO-END RELIABILITY IN WIRELESS SENSOR NETWORKS: SURVEY AND RESEARCH CHALLENGES

[55] Bret Hull, Kyle Jamieson, Hari Balakrishnan, MIT Computer Science and Artificial Intelligence Laboratory, Mitigating Congestion in Wireless Sensor Networks, SenSys'04, November, 2004.

[56] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris, A high throughput path metric for multi-hop wireless routing. Wirel. Netw., 11(4):419-434,2005.

[57] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris, Performance of multihop wireless networks: shortest path is not enough. SIGCOMM Computer Communication Rev., 33(1):83-88, 2003.

[58] Charles E. Perkins and Pravin Bhagwat, Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers, SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications, pages 234-244, New York, NY, USA, 1994. ACM.

[59] Richard Draves, Jitendra Padhye, and Brian Zill. Comparison of routing metrics for static multi-hop wireless networks, SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, pages 133-144, NY, USA, 2004. ACM.

[60] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multi-radio, multi-hop wireless mesh networks. In MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking, pages 114-128, New York, NY, USA, 2004. ACM.

[61] J. Park and S. Kasera. Expected data rate: an accurate high-throughput path metric for multi-hop wireless routing. In Proc. of second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005.

[62] Jian Tang, Guoliang Xue, and Christopher Chandler. Interference-aware routing and bandwidth allocation for qos provisioning in multihop wireless networks: Research articles. Wirel. Commun. Mob. Comput., 5(8):933-943, 2005.

[63] Jenn Yue Teo, Yajun Ha and Chen Khong Tham, Interference-minimized multipath routing with congestion control in wireless sensor network for multimedia streaming.

[64] Moufida Maimour, C. Pham, Julien Amelot, Load Repartition for Congestion Control in Multimedia Wireless Sensor Networks with Multipath Routing,

[65] Lucian Popal, Costin Raiciu, Ion Stoica', David S. Rosenblum, Reducing Congestion Effects in Wireless Networks by Multipath Routing.

[66] Dong Jiang,Qianping Wang, Yan Zhao,Ke Wang, The Research and Design of High Reliability Routing Protocol of Wireless Sensor Network in Coal Mine.

[67] C. Chandler J. Tang, G. Xue. Interference-aware routing in multihop wireless networks using directional antennas. In Proceedings of IEEE INFOCOM, 2005.

[68] Rangwala, R. Gummadi, R. Govindan, and K. Psounis, Interference-Aware Fair Rate Control in Wireless Sensor Networks, SIGCOMM 2006.

[69] Csaba D. Tth Yunhong Zhou Chiranjeeb Buragohain, Subhash Suri. Improved throughput bounds for interference-aware routing in wireless networks. In Proc. 13th Computing and Combinatorics Conference, 2007.

[70] G. Hiertz B. Walke E. Weiss, O. Klein. Capacity and interference aware ad hoc routing in multi-hop networks. In Proceedings of 12th European Wireless conference, 2006.

[71] Fengguang An Liran Ma, Qian Zhang and Xiuzhen Cheng. Diar: A dynamic interference aware routing protocol for ieee 802.11-based mobile ad hoc networks. In International Conference on Mobile Ad-hoc and Sensor Networks (MSN) 2005, LNCS, 2005.

[72] H. Mahmood and C. Cornaniciu. Interference aware routing for CDMA wireless ad hoc networks. Military Communications Conference, 2005. MILCOM 2005. IEEE, pages 1059-1063 Vol. 2, 17-20 Oct. 2005.

[73] Tamer ElBatt and Timothy Andersen. Cross-layer interference-aware routing for wireless multi-hop networks. In IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing, pages 153-158, New York, NY, USA, 2006. ACM.

[74] Dang-Quan Nguyen and Pascale Minet. Interference-aware qos olsr for mobile adhoc network routing. In SNPD-SAWN '05: Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks, pages 428-435, Washington, DC, USA, 2005. IEEE Computer Society.

[75] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir Das. Ad Hoc On Demand Distance Vector (AODV) Routing IETF RFC 3561.

[76] S. Chen, N. Yang Congestion Avoidance based on Light-Weight Buffer Management in Sensor Networks ,University of Florida, Gainesville, FL 32611, USA.

[77] Hull, B., Jamieson, K., and Balakrishnan, H, Bandwidth management in wireless sensor networks, Tech. Rep. 909, MIT Laboratory for Computer Science, July 2003.

[78] Kiran Yedavalli, Using Wireless Advantage for Congestion Control in Wireless Sensor Networks, USC Technical Report CENG-2005-13, December 05.

[79] Glauche, Ingmar,Krause, Wolfram,Sollacher, Rudolf,Martin, Distributive routing and congestion control in wireless multihop ad hoc communication networks, Elsevier ,2004.

[80] Victor Shnayder Application Aware Congestion Control in Wireless Sensor, Harvard University.

[81] Ray, S. ,Carruthers, J.B.Starobinski, D. RTS/CTS-induced congestion in ad hoc wireless LANs, Volume: 3, On page(s): 1516- 1521 vol.3, 16-20 March 2003.

[82] Laura Galluccio , Andrew Campel, Sergio Palazzo, Concert: aggregation-based congestion control for sEnsoR networks, Sensys05 p 274 - 275 , 2005

[83] Alec Woo and David Culler, A transmission control scheme for media access in sensor networks, in Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking, Rome, Italy, July 2001, pp. 221235, ACM.

[84] Cui Qing Yang and Alapati V.S Reddy, A Taxonomy for congestion Control Algorithms, in Packet Switching Networks, IEEE August 1995.

[85] W. Ye, J. Heidemann and D. Estrin, An Energy-Efficient MAC Protocol for Wireless Sensor Networks," INFOCOM 2002, vol. 3, pp. 1567-1576, June 2002.

[86] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva, Directed Diffusion for Wireless Sensor Networking, ACM Mobicom 2000.

[87] Shuang Li, Alvin Lim, Santosh Kulkarni, and Cong Liu. Edge: A routing algorithm for maximizing throughput and minimizing delay in wireless sensor networks, Military Communications Conference, 2007, MILCOM 2007, IEEE, pages 1-7, 29-31 Oct. 2007.

[88] The network simulator ns-2. http://www.isi.edu/nsnam/ns/ - last accessed 1/25/2010.