PERFORMANCE-AWARE CONGESTION CONTROL IN WIRELESS SENSOR NETWORKS USING RESOURCE CONTROL

Charalambos Sergiou

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Cyprus

Recommended for Acceptance

by the Department of Computer Science

October, 2012

2012

APPROVAL PAGE

Doctor of Philosophy Dissertation

PERFORMANCE-AWARE CONGESTION CONTROL IN WIRELESS SENSOR NETWORKS USING RESOURCE CONTROL

Presented by

Charalambos Sergiou

Research Supervisor ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Dr. Vasos Vassiliou

Committee Member ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Dr. Andreas Pitsillides

Committee Member ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Dr. Chryssis Georgiou

Committee Member ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Dr. Georgios Ellinas

Committee Member ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ
Dr. Azzedine Boukerche

University of Cyprus

October, 2012

PERFORMANCE-AWARE CONGESTION CONTROL IN WIRELESS SENSOR NETWORKS USING RESOURCE CONTROL

Charalambos Sergiou

University of Cyprus, 2012

Recent advances in Wireless Sensor Networks (WSNs) lead to applications with increased traffic demands. Performance in terms of throughput, latency, and power consumption is now a dominating factor that leads research in WSNs. There are many cases in the fields of automation, health, and disaster response that demand WSNs with strict performance assurances. Congestion occurrence is a key factor that can negatively affect the performance of WSNs. The overwhelming majority of approaches that deal with congestion control in WSNs attempt to control congestion, by reducing the rate with which sources inject packets in the network. This method is called *traffic control*. Although traffic control seems to be an effective method for controlling congestion, it presents a number of drawbacks which are not easy to ignore. The most important drawback steams from the fact that higher traffic load occurs when the monitored event takes place. At this instance there is a higher probability of congestion occurrence in the network. By controlling the rate with which packets are injected in the network, the amount of information that reaches the data sinks reduces. This fact can jeopardize the purpose of the network. Moreover, network connectivity issues arise since in most cases, this approach utilizes the shortest path from source to sink. Thus, in case of heavy data load, this path of nodes can easily become power exhausted. This leads to the creation of routing "holes" in the network.

In this thesis we approach congestion control and avoidance in WSNs with a different perspective. In particular, when congestion occurs, instead of reducing the rate with which packets

are injected in the network we program nodes to route a number packets through alternative paths in order to avoid the congested areas. This method is called *resource control*. To achieve this, we take advantage of the fact that wireless sensor nodes are frequently redundantly and/or densely deployed.

Thus, initially we prove with mathematical analysis using a traffic flow model, that the traffic control method can be proven inefficient in specific scenarios where performance assurances are needed. Then, we prove that the inefficiencies of this method can be handled effectively by the resource control method. Then, we argue on the importance of topology control algorithms and study how tree- forming schemes can assist the purpose of congestion control with specific performance bounds. After this study, we propose an algorithm, called HTAP (Hierarchical Tree Alternative Path) which is an efficient algorithm for congestion control and avoidance, in terms of throughput and power consumption. Then, we propose an alternative algorithm, called DAlPaS (Dynamic Alternative Path Selection), which keeps the advantages of HTAP algorithm but presents higher and more stable performance in terms of time delay, power consumption and throughput. HTAP and DAlPaS algorithms are also evaluated under different placements. Finally, we compare the performance of HTAP against different types of congestion control algorithms in WSNs.

ACKNOWLEDGEMENTS

I would like to express my gratitude to everyone who helped and supported me throughout my Ph.D. experience.

First and foremost, I would like to thank my advisor, Assistant Professor Dr. Vasos Vassiliou. It has been a great honor for me to be his first Ph.D. student. Dr. Vassiliou has taught me how to perform scientific research and how to take decisions on critical issues. I appreciate all his contributions of time, ideas, and patience to make my Ph.D. experience productive and stimulating. The joy and enthusiasm he has for his research was contagious and motivational for me, even during tough times in the Ph.D. pursuit.

Moreover, i would like to thank all people of Networks Research Laboratory (NetRL) of University of Cyprus and especially the Director and founder of the group, Professor Andreas Pitsillides who was the person that guided me to the world of Wireless Communications, while i was studying for my MSc Degree.

I also wish to express my very special thanks to my family and especially to my parents that supported me in all the ways during this process.

Lastly, and most importantly, I wish to thank my beloved wife Christiana and my little daughters Eleni and Georgia, for their patience, love and support during this long effort. To them, I dedicate this thesis with deepest gratitude.

CREDITS

In this section, we provide a complete list of publications and submissions stemming from the work in this thesis.

Book Chapters

C. Sergiou and V. Vassiliou, "Efficient Node Placement for Congestion Control in Wireless Sensor Networks," *InTech- Wireless Sensor Networks - Technology and Applications*, vol. ISBN: 978-953-51-0676-0, 2012.

Journals

C. Sergiou, V. Vassiliou, and A. Paphitis, "Hierarchical Tree Alternative Path (HTAP) Algorithm for Congestion Control in Wireless Sensor Networks," *Elsevier Ad Hoc Networks*, vol. 11, Issue 1, pp.257-272, 2013.

C. Sergiou and V. Vassiliou, "Estimating Maximum Traffic Volume in Wireless Sensor Networks Using Fluid Dynamics Principles," *accepted in IEEE Communications Letters*.

C. Sergiou, P. Antoniou, and V. Vassiliou, "Congestion Control and Avoidance in Wireless Sensor Networks: A Survey," *submitted in IEEE Surveys and Tutorials (under 2nd review)*.

Conferences

C. Sergiou and V. Vassiliou, "Source-based Routing Trees for Efficient Congestion Control in Wireless Sensor Networks," in *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, May 2012, pp. 378–383 [1].

C. Sergiou and V. Vassiliou, "Study of Lifetime Extension in Wireless Sensor Networks through Congestion Control Algorithms," in *IEEE Symposium on Computers and Communications (ISCC)* July 2011, pp. 283 –286 [2].

C. Sergiou and V. Vassiliou, "Performance Evaluation of the DAlPaS Congestion Control Algorithm in Wireless Sensor Networks," in *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, June 2011, pp. 1–7 [3].

C. Sergiou and V. Vassiliou, "DAlPaS: A Performance Aware Congestion Control Algorithm in Wireless Sensor Networks," in *18th International Conference on Telecommunications (ICT)*, May 2011, pp. 167 –173 [4].

M. Koutroullos, **C. Sergiou**, and V. Vassiliou, "Mobile-CC: Introducing Mobility to WSNs for Congestion Mitigation in Heavily Congested Areas," in *18th International Conference on Telecommunications (ICT)*, May 2011, pp. 400 –405 [5].

C. Sergiou and V. Vassiliou, "Energy Utilization of HTAP under Specific Node Placements in Wireless Sensor Networks," in *European Wireless Conference (EW),* April 2010, pp. 482 –487 [6].

V. Vassiliou and C**. Sergiou**, "Performance Study of Node Placement for Congestion Control in Wireless Sensor Networks," in *3rd International Conference on New Technologies, Mobility and Security (NTMS)*, Dec. 2009, pp. 1 –8 [7].

C. Sergiou, V. Vassiliou, and A. Pitsillides, "Reliable Data Transmission in Event-Based Sensor Networks During Overload Situation," in *WICON '07: Proceedings of the 3rd International Conference on Wireless Internet*, Austin, Texas, October 2007, pp. 1–8 [8].

Poster Abstracts

C. Sergiou, M. Koutroullos, and V. Vassiliou, "Poster Abstract: A Congestion Mitigation Approach using Mobile Nodes in Wireless Sensor Networks," in *10th International Conference on Information Processing in Sensor Networks (IPSN)*, April 2011, pp. 161 –162 [9].

C. Sergiou and V. Vassiliou, "Poster Abstract: Energy Hole Prevention in Wireless Sensor Networks," in *9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2010, pp. 398–399 [10].

C. Sergiou and V. Vassiliou, "Poster Abstract: Alternative Path Creation vs Data Rate Reduction for Congestion Mitigation in Wireless Sensor Networks (IPSN)," in *9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, April 2010, pp. 394–395 [11].

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) are wireless networks consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations [12,13]. They comprise of small, cooperative devices which are constrained in terms of computation power, memory space, communication bandwidth, and energy supply. These devices, usually called "nodes", integrate sensors, radio communications, and digital electronics into a single integrated circuit (IC) package. A typical architecture of a wireless sensor node is presented in Fig.1.

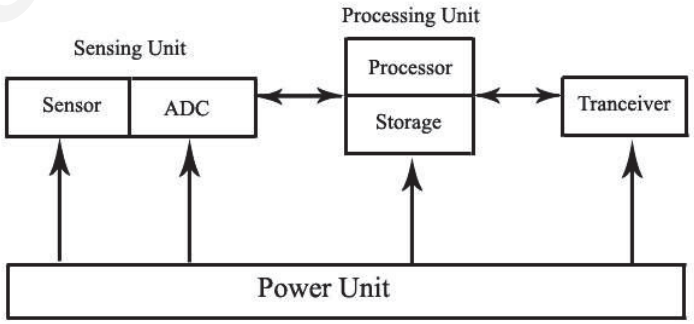

Figure 1: Wireless Sensor Node Architecture.

Lately, applications that employ wireless sensors increase and can be roughly categorized as follow:

- Military area monitoring
- Environmental monitoring
- Industrial monitoring
- Water/wastewater monitoring
- Structural monitoring
- Passive localization and tracking
- Smart home monitoring

Currently, a wide range of WSNs has also been deployed in civilian areas for habitat observation [14, 15], health monitoring [16], and object tracking [17, 18].

Moreover, intense study has been carried out concerning many aspects of WSNs, especially in the physical [19, 20], MAC [21–23], and network layers [24–27].

Research studies in the area of sensor networks have mainly focused on fundamental networking problems, e.g. physical layer, medium access control (MAC), topology, routing, energy efficiency, and have largely ignored network performance assurances. Lately, with the emergence of mission-critical applications (e.g., health monitoring, plant monitoring, etc.), there has been an increased interest in the performance of control mechanisms, so as to avoid congestion caused by the uncontrolled use of the scarce network resources. Due to these resource limitations and to the error prone nature of wireless environment, transport control protocols used in wired-line or even in infrastructure based wireless networks seem inadequate to apply in WSNs.

Typically, WSNs operate under light load and 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 [28]. This high generation rate of data packets is usually uncontrolled and often leads to congestion. Congestion can appear either due to collisions in the medium, since they create retransmissions and delays, or due to overflow of the buffers of nodes. [22, 27]. Buffer overflow results in random drops of data packets and increased delay. Dropped packets are a major handicap for WSNs, since they result in severe energy consumption. If no countermeasures are taken, the power of congested nodes can be quickly exhausted leading to the creation of "dead paths" (holes) in the network.

Currently, many schemes and algorithms have been proposed for congestion control and avoidance, as well as for reliable data transmission in WSNs. Concerning congestion control and avoidance, most algorithms use cross layer approaches between MAC and network layer. In the great majority of congestion control algorithms, authors employ traffic control. Using this method, they choose to control the rate with which sources inject packets in the networks. On the other hand, reliable data transmission approaches usually run at the transport layer. Some of them are end-to-end and employ retransmissions using ACK or NACKs [29].

In this thesis we focus on congestion control and avoidance using the *resource control* approach. During this research, using analysis based on fluid dynamics principles, we prove the possible shortcomings of the traffic control method for congestion control and avoidance in WSNs. Then, we argue on the importance of topology control algorithms by studying tree forming schemes. Using the results of the mathematical analysis and the topology control study we propose two congestion control and avoidance algorithms that employ resource control which, as we prove, is a very effective method under specific scenarios. The first algorithm is called HTAP (Hierarchical

Tree Alternative Path) while the second is called DAlPaS (Dynamic Alternative Path Selection). We evaluate the performance of each algorithm under many scenarios, while we also study the energy utilization of each algorithm. Moreover, we study how the placement of nodes is possible to affect the performance of a network that employs resource control as a method to control congestion.

1.1 Motivation and PhD Thesis Objectives

The performance of wireless sensor networks is a domain that attracts a lot of attention lately. Research mainly focuses in controlled environments (e.g. big plants, factories etc.) where wireless sensors tend to replace old fashioned wired devices that are being used to monitor specific parameters like temperature, pressure, vibration etc. In such environments, wireless sensors periodically send their measurements, hop-by-hop to the sink, in a control center. There, the automated control system monitors the readings and if the readings exceed certain limits, appropriate actions are taken. The advantage of deployment in a controlled environment lies on the fact that the network topology and the data load is, in a way, predetermined since the functionality of the network lies mostly on periodic reports. Thus, design consideration mostly focuses on metrics like delay and reliability, which of course are not in any way trivial to achieve.

Our work moves a step ahead and attempts to develop controlled conditions in an *uncontrolled* environment. Uncontrolled environments are called the environments where nodes are spread in order to monitor specific events and report these events continuously when they happen. The effort of developing a wireless sensor network in such conditions imposes another significant requirement that has to be taken into account (besides delay and reliability). Since event-based WSNs produce a very big number of data packets when the monitored event takes place, congestion is

inevitable if no actions are taken. Therefore, congestion control which sometimes is also referred as *overload control*, is also needed.

In this thesis we focus on the design and development of specific algorithms which, if applied to an event-based wireless sensor network, will be able to guarantee the performance of the network without imposing any risk to the application. Risk to the application is considered the case where a sink node(s) receives a limited number of packets, either due to congestion because of packet drops, or due to buffer overflows, or due to overload (packet collisions in the medium), or due to source data rate reduction (a method used for congestion alleviation). Risk to the application may also occur when data packets arrive with delay out of bounds, according to the application demands. Another risk may be the case where due to massive power exhaustion of wireless nodes in specific parts of the network, connectivity holes are created and the network becomes fragmented and disconnected.

In this thesis we employ a cross layer design, combining the MAC and the Network layers. We intentionally avoid to operate at the transport layer since we prove that, especially in these networks, end-to-end approaches or approaches that demand heavy re-transmissions in case of lost packets cannot be proven functional.

Thus, based on these observations we design and implement a set of novel algorithms capable to control the performance of event-based WSNs using alternative path routing. By default we reject traffic control approaches since, as it has already been shown in [30], reducing the rate with which sources inject packets in the network can destroy the purpose of the network.

Thus, the algorithms we design are based on the resource control method. Employing this method for congestion control and avoidance has not been widely adopted. The reason lies on the limitations of this method, as the high possibility routing holes creation, the increased energy consumption, and the increased delay.

Hence, our target is to study extensively this method and design specific algorithms that are not only capable to overcome the limitations of the resource control method, but also to present stable and robust performance.

These algorithms are being tested under different conditions in order to evaluate their performance. Then, we also pay attention to the deployment of nodes in the network. We study and show that planned deployment can be more effective and efficient compared to random deployment. We also argue on the importance of topology control algorithms in the efficient operation of resource control algorithms in WSNs and we extract conclusions. Finally, we compare the traffic and resource control methods and at the end we study the lifetime extension that resource, congestion control algorithms can provide to the network.

1.2 Work Accomplished and Contributions

In this section we briefly present the work accomplished in order to achieve the objectives of this Ph.D. dissertation. Our research work is performed in two parts. In the first part we present a thorough literature survey identifying the past and current approaches in this area. Out of this survey we draw the design guidelines for our algorithms. Then, through a mathematical modeling, we present the inefficiencies of the traffic control method for congestion control and avoidance in WSNs, as well as the performance problems that existing resource control algorithms appear.

In the second part we initially propose a resource control algorithm called, HTAP (Hierarchical Tree Alternative Path) and we evaluate its performance. Then, we propose a second algorithm called, DAlPaS (Dynamic Alternative Path Selection) that also employs the resource control method and succeeds to provide guaranteed and stable performance. Following, we compare the performance of these algorithms under different placements. Finally, we compare the performance the performance of HTAP, against different types of congestion control algorithms in WSNs.

1.2.1 Part 1 of Research: Problem Identification

In the first part of the research a detailed survey of past and current approaches on the subject is being performed. Through this survey a classification of congestion related algorithms is carried out, based on the way they detect, notify, and face congestion. We survey a significant number of congestion control and avoidance protocols and we conclude on their inefficiencies on a number of critical parameters for the performance of WSNs. Through this revision we identify that the majority of the algorithms that we review, employ traffic control for congestion mitigation and avoidance while the few resource control algorithms that exist, are not able to guarantee stable and high performance, especially in terms of power consumption and time delay. Then, through a traffic model based on fluid dynamics analysis we show the problems that arise using the traffic control method for congestion control. Using this analysis, we claim that the resource control method, may provide a robust solution to congestion control issues in WSNs. Finally, we argue on the importance of topology control algorithms by comparing tree-forming schemes. Specifically, we compare source- based with sink- based trees and we extract conclusions.

1.2.2 Part 2 of Research: Proposed Solutions

In the second part of the research, after we clearly identify the inefficiencies of current approaches, we design and implement an algorithm capable of overcoming these inefficiencies. The algorithm is called HTAP (Hierarchical Tree Alternative Path). The HTAP algorithm takes advantage of the redundant deployment of nodes in WSNs and using the resources of nodes which are in “sleep” state, builds alternative paths in order to avoid congested areas. The HTAP algorithm initially builds a virtual tree from every source node to the sink. Therefore, in case of congestion, each node is immediately able to identify an alternative node and to use it in order to built an alternative path for congestion avoidance.

Then, we design and implement a second algorithm called DAlPaS (Dynamic Alternative Path Selection). The DAlPaS algorithm, when compared to HTAP, is even more lightweight in terms of computation complexity and energy, and attempts to provide a guaranteed high performance during the whole lifetime of the network. Thus, initially a spanning tree that initiates from the sink is being built. Then, before any congestion occurs, DAlPaS employs a "soft stage" algorithm with which any flows that merge on a single node are advised to seek for alternative paths. Using this "soft stage" algorithm, congestion is dealt with and avoided early, especially when there is no high load in the network. In the DAlPaS algorithm, a node indicates its unavailability in the network through a flag decision algorithm. Each node, using this simple algorithm, it declares itself available or not, for receiving traffic. If a node declares itself as unavailable, then a "hard stage" algorithm runs on its neighbor nodes and these nodes are forced to route packets through different paths. According to simulation results DAlPaS algorithm appears to be an effective, lightweight algorithm for congestion control and avoidance in WSNs.

After implementing and evaluating the performance of HTAP and DAlPaS we study how the placement of nodes on the field affect the performance of congestion control algorithms. Simulation results show that resource control algorithms are favored in terms of power consumption and time delay when nodes are densely deployed near sources and sinks.

Finally, we also compare the performance of HTAP and DAlPaS against, other congestion control in WSNs, but of different types.

1.2.3 Thesis Overview

The thesis is organized as follows. In Chapter 2 we present the background information on the topic and the related work. Specifically, after the introduction, in section 2.2 we describe the problem of congestion in WSNs, providing information on how and where congestion occurs.

Section 2.3 provides insights on the topics of congestion avoidance, congestion mitigation, and reliable transmission and classifies the examined algorithms based on the control scheme employed. Section 2.4 presents an additional classification, based on the mechanisms used for detection, notification, mitigation, and avoidance and section 2.5 identifies common performance evaluation metrics used in the literature. Finally, in section 2.7 we summarize the inefficiencies and advantages of currently proposed algorithms and we draw the guidelines for the design of novel, more effective and efficient congestion control and avoidance algorithms.

In Chapter 3 we present a mathematical analysis based on a macroscopic fluid dynamic model, in order to show the inefficiencies of the traffic control method, for congestion control in WSNs. Specifically, we propose a macroscopic fluid dynamic model dealing with the flow of information in a wireless sensor network. Using the three fundamental traffic variables velocity (v), flow (f) and density (ρ), we prove that a "conservation of information law" can stand in WSNs. Based on this law we can estimate the speed with which fluctuations propagate in the network when a node for some reason stops forwarding packets (or reduces the rate with which it forwards packets).

In Chapter 4 we perform a comparison between two basic topology control schemes. The first scheme concerns source based trees which are routing trees that are created from each source to sink and the second scheme concerns trees which are created from sinks to sources. We study how these schemes affect the performance of resource, congestion control algorithms.

In Chapter 5 we present and analyze the Hierarchical Tree Alternative Path (HTAP) algorithm. Initially, we present the network model that we employ. Then we present HTAP algorithm. Specifically, we describe its topology control scheme, then we present the Hierarchical Tree and Alternative Path Creation schemes, as well as how this algorithm handles the powerless or failed nodes. Finally, we evaluate the performance of HTAP algorithm though extensive simulations.

In Chapter 6 we present, analyze and evaluate the performance of the Dynamic Alternative Path Creation Algorithm (DAlPaS). Specifically, we present the three main schemes of this algorithm. The setup phase, the soft stage scheme and the hard stage scheme. Finally, we also evaluate the performance of DAlPaS algorithm though extensive simulations

In Chapter 7 we evaluate the performance of HTAP and DAlPaS under different node placements.

In Chapter 8 we we compare the performance of HTAP and DAlPaS against other congestion algorithms in WSNs of different types.

Finally, in Chapter 9 we close with the conclusions and future work.

Chapter 2

Background Information and Related Work

In this Chapter, a thorough literature review is presented. Through this review, we attempt to reveal the advantages and disadvantages of a significant number of existing congestion control and avoidance algorithms in WSNs. Based on this review we draw the design guidelines for the algorithms that we propose in the thesis.

2.1 Introduction

In recent years the problem of congestion control and avoidance attracted a lot of attention. Currently, some notable efforts exists that survey the subject. Three of them, focus directly on congestion control approaches for wireless sensor networks [29, 31, 32], while two others deal with congestion control as part of transport protocols [28, 33]. These three surveys that focus on congestion control approaches, are quite limited in content covering only a very small subset of research works which, in some cases [31,32], are outdated. Furthermore, the three aforementioned survey papers do not provide a critical evaluation of any of the presented approaches, avoiding to address and discuss the strengths and weaknesses of each approach. However, it is worth noting that one of the three papers, [32], provides a very basic classification of the presented approaches.

The two survey papers describing the basic design criteria and challenges of transport protocols for WSNs provide guidelines towards controlling (avoiding or mitigating) congestion in WSNs. Also, the papers emphasize on quality of service and reliability. In [33], a quite limited number of existing congestion control approaches are mentioned. On the other hand, the work in [28] covers a larger set of congestion control approaches, while providing differentiation based on congestion detection, congestion notification, and congestion mitigation mechanisms. However, the last two papers are considered outdated since a considerable amount of congestion control approaches have been proposed over the last few years.

In this Chapter we present a state of the art survey, review and classification of a significant number of algorithms.

2.2 Congestion in WSNs

A node in a wireless sensor network (WSN) is a small embedded computing device that interfaces with sensors/actuators and communicates using short-range wireless transmitters. Such nodes act autonomously but cooperatively to form a logical network in which data packets are routed hop-by-hop towards management nodes, typically called sinks or base stations. A WSN comprises a potentially large set of nodes that may be distributed over a wide geographical area, indoor or outdoor. Wireless sensor networks enable numerous sensing and monitoring services in areas of vital importance such as efficient industry production, safety and security at home, and in traffic and environmental monitoring. Traffic patterns in sensor networks can be derived from the physical processes that they sense. Sensor networks typically operate under light load and 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. This high generation rate of data packets is usually uncontrolled and often leads to congestion. In this state, collisions occur in the medium or in case of existence of an effective MAC protocol, the node buffers overflow [22, 27], resulting in random drops of data packets and increased delay. Dropped packets are a major handicap for these networks since they result in severe energy consumption. In the case that no countermeasures are taken, the power of congested nodes can be exhausted leading to the creation of routing "holes" in the network.

Congestion in WSNs can be classified in two major categories concerning how packets are lost and where in the network congestion takes place (Fig. 2).

2.2.1 How Packets Are Lost

1. **Packet collisions in the medium:** In a particular area, many nodes within range of one another attempt to transmit simultaneously, resulting in losses due to interference and thereby reducing throughput of all nodes in the area. We note that explicit local synchronization among neighboring nodes can reduce this type of loss, but cannot eliminate it completely, because non-neighboring nodes can still interfere with transmission.

2. **Packet drops due to buffer overflow:** Within a particular node, the queue or buffer, used to hold packets to be transmitted, overflows. This is the conventional definition of congestion, widely used in wired networks. In this case, nodes receive packets with higher rate that they can transmit.

2.2.2 Where Packets Are Lost

1. **Hotspot near source - Source Congestion:** Densely deployed sensors generating data events during a crisis state will create hotspots very close to the sources (e.g. within one or

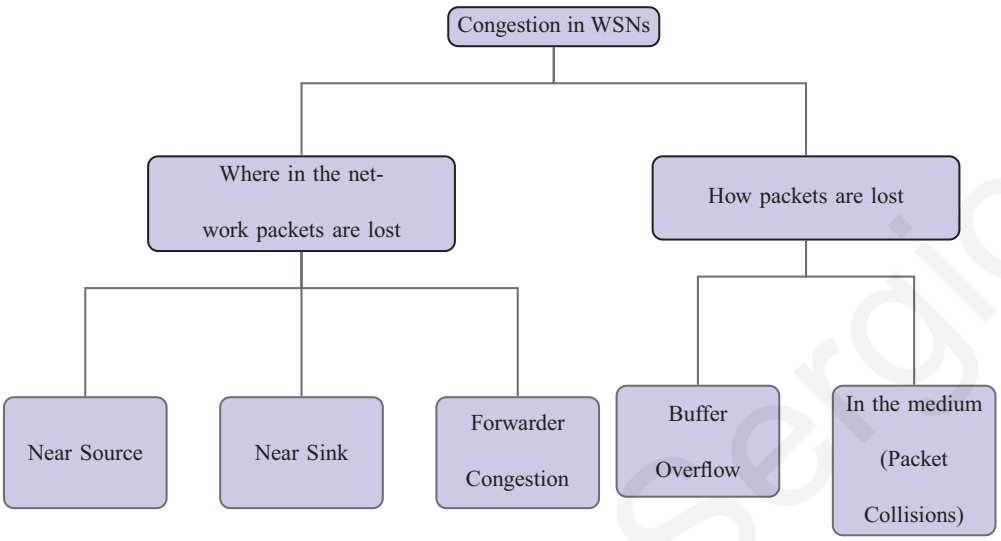

Figure 2: Congestion in WSN.

two hops). In this case, localized, fast time-scale mechanisms capable of providing back-pressure messages from the points of congestion back to the sources would be effective for immediate traffic control until the congestion is alleviated by other means. Also local de-synchronization of sources and resource provisioning techniques (resource control) would be effective too.

2. **Hotspot near the sink -Sink Congestion:** Even sparsely deployed sensors that generate data at low data rates can create hotspots in the sensor field, but likely farther from the sources, near the sink. Fast time-scale resolution of localized hotspots using a combination of localized back-pressure and packet dropping techniques would be more effective in this case. Source nodes may not be involved in the backpressure because of the transient nature of the problem in this situation. Also an effective way of alleviating sink congestion is to deploy multiple sinks that are uniformly scattered across the sensor field and therefore, balance the traffic between these sinks.

3. **Forwarder Congestion:** A sensor network will have more than one flow (sink-source pair), and these flows will intersect with one another. The area around the intersection will likely become a hot spot. In a tree-like communication paradigm, every intermediate node in the tree can suffer from forwarder congestion. Compared to the other congestion locations, forwarder congestion is far more challenging, because it is very difficult to predict the intersection points due to the network dynamics. In this case, even sparsely deployed sensors generating data will create both transient and persistent hotspots distributed throughout the sensor field. A combination of fast time scale actions to resolve localized transient hotspots, and closed loop rate regulation of all sources that contribute toward creating persistent hot spots seems to be effective. Resource control techniques could be used when traffic control methods cannot meet application's requirements.

2.3 Control Schemes

Generally, algorithms that deal with congestion in WSNs can be initially classified in three major categories: Congestion Control, Congestion Avoidance, and Reliable Data Transmission (Fig. 3). Although there are no clear and explicit boundaries between these three categories, we attempt a first classification of algorithms based on this set. In this thesis, *congestion mitigation* algorithms are considered the algorithms that take reactive actions when congestion arises in the network and their target is to control it. These algorithms normally involve MAC and network layer operations, and in some cases they also use transport layer actions.

Congestion avoidance algorithms are considered as the algorithms that take actions in order to prevent congestion from happening. These algorithms normally involve MAC and network layer operations.

On the other hand, *reliable data transmission* algorithms are the algorithms that beside their effort to control congestion in a network, they also attempt to recover all or part of the lost information. These algorithms normally apply when all information is critical for the application and usually involve transport layer mechanisms.

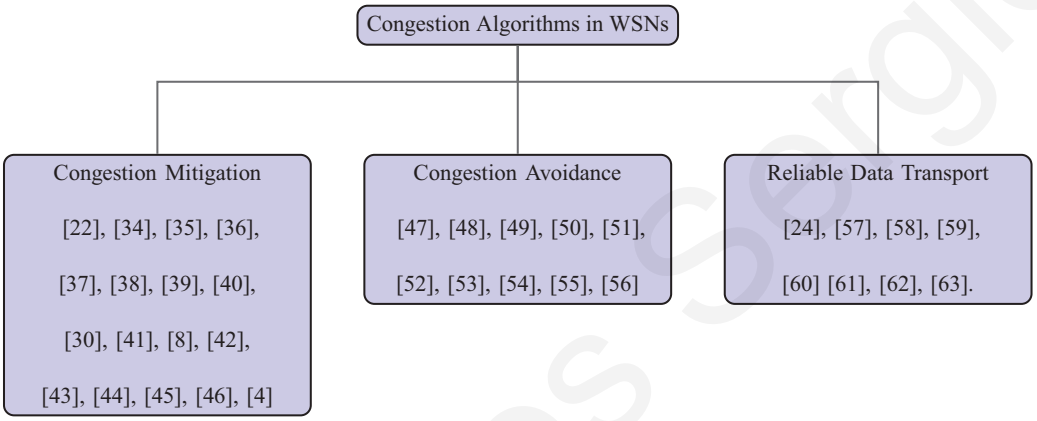

Figure 3: Initial Classification of Congestion Algorithms in WSN.

In general, the presence of congestion means that the load is (temporarily) greater than the network resources can handle in such a way that resources become depleted. In such a case the following control schemes may be used: control the load (traffic control), increase the resources (resource control), or employ MAC layer enhancements. MAC layer enhancements could help more in the direction of interference-based congestion (packets collision in the medium). If the packet generation rate is sufficiently small, simultaneous transmission of packets becomes independent of the rate. Rather, it depends on the time at which each node generates the packet. A good way to reduce this type of congestion is to perform phase shifting, an observation made by authors in [22]. Small amounts of phase shifting can be performed by introducing slight jitters at the data-link layer. In [22], the application layer itself also introduces phase shifts. While jittering at the data-link layer aims to cause small transmission variations between neighboring nodes, we think, that phase shifting at a higher layer can be achieved on a larger time scale. To handle buffer

based congestion (packet drops due to buffer overflow) one may employ the other two methods, a) traffic control or b) resource control as these would help in emptying the buffers of intermediates sensor nodes. It is possible to have more than one types of congestion occurring at the same time.

2.4 Classification of Algorithms

In this section we further classify the three major categories of congestion algorithms in WSNs as presented in Fig. 3 into further attributes.

2.4.1 Congestion Mitigation Algorithms

Algorithms that deal with congestion mitigation can be classified on the way they detect congestion, the way they notify the other nodes for this incident, as well as the way they face congestion (counteractions mechanisms).

1. **Congestion Detection:** Currently, there are four ways that algorithms use to detect congestion (Fig. 4).

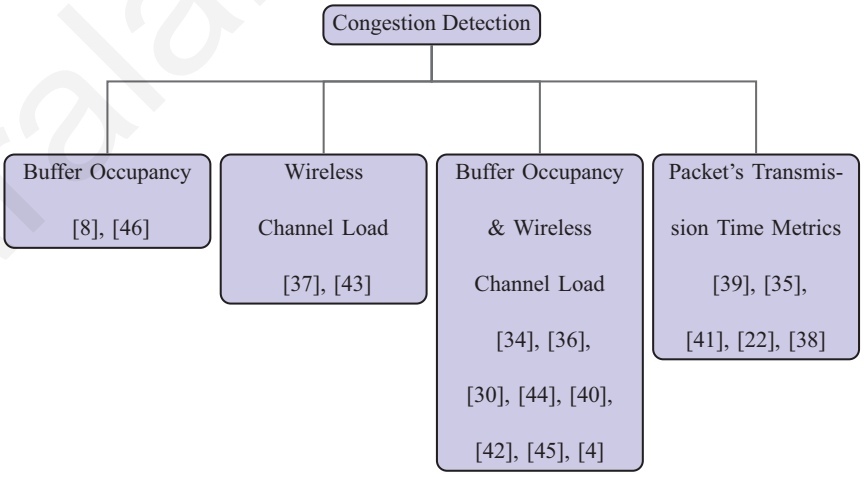

Figure 4: Congestion Mitigation Algorithms: Congestion Detection Mechanisms.

These are the following:

- **Buffer occupancy:** The algorithms that use this way for detecting congestion assume that there is an effective MAC protocol able to avoid packet collisions in the medium, or they assume that more than one nodes that are not in the range of each other are sending with a small time shift to a receiving node. In this case, congestion is measured through the increment of queue length in nodes. Algorithms that employ this method like [8,46] act proactively by inferring congestion when the buffer occupancy exceeds a certain percentage, while the sender's rate is lower that the receiver's rate.
- **Wireless Channel Load:** Algorithms using this method for congestion detection only count the medium load and take actions when the time for the transmission of a single packets exceeds some predefined thresholds.
- **Buffer Occupancy and Wireless Channel Load:** With this method, congestion is detected either at the medium or the buffer. Concerning buffer occupancy, the same tactic as above is maintained. In the second case (channel loading) if a node senses that channel loading reaches a certain fraction of channel capacity it infers that a new packet transmission will probably lead to packet collisions (congestion in the channel). In this category we also insert algorithms that form clusters and measure congestion through traffic intensity.
- **Packet Transmission Time Metrics:** Algorithms that employ this method to detect congestion use packet service time and packet inter-arrival time (or a combination of them) to detect congestion. Specifically, algorithms like [39] [41] count the packet service time and packet interarrival time and if it is beyond a limit they infer that congestion is imminent.

Commenting on the above mentioned methods we can safely state that each one presents advantages and disadvantages. Buffer occupancy is a very simple method, it can be easily implemented while it does not require much resources from nodes. The disadvantage of this method, relies on its dependance from the MAC protocol. If the MAC protocol is not efficient enough, it is possible to have collisions in the medium (if more than one nodes are sending concurrently packets to specific nodes) and buffers to receive limited number of packets. In this case the network is not possible to detect the hotspots and inevitably problems will be created in the network. Wireless channel load can efficiently tackle the problem of medium collisions but it can not react when buffers are fully occupied and drop packets. Packet transmission time metrics, although it is the most efficient method it is heavily depended on the attitude of the wireless medium. This means that it is possible to have packets drops due to other reasons e.g environment or physical causes and the network to trigger congestion situation. Buffer occupancy and wireless channel load is according to our opinion the most efficient way for congestion detection. It captures congestion either on the medium or the buffers. It is easy to be implement and it relatively consumes less power in comparison with the other methods. It is the method that it has been adopted by the most congestion mitigation protocols.

2. **Congestion Notification:** A second classification can be based on the way that algorithms use, to notify the rest of the network for congestion events. This is either explicit or implicit (Fig. 5).

    - **Explicit:** Using explicit congestion notification, control packets are sent by congested nodes to the rest of the nodes to inform them about congestion. This method has been used by the first congestion algorithms like [34] and [38]. Since then, it has been

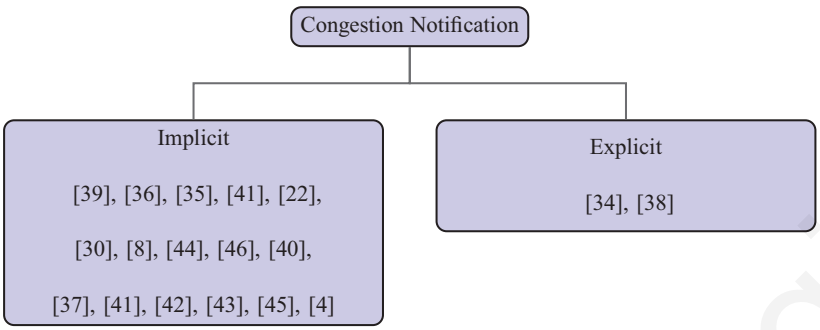

Figure 5: Congestion Mitigation Algorithms: Congestion Notification Methods.

proven that sending control packets when congestion has occurred, adds significant load to the already congested environment. Therefore, explicit congestion signaling has not been adopted by subsequent congestion control protocols.

- **Implicit:** Implicit congestion notification is the method that is being used, primarily, by the latest congestion control protocols. Specifically, congestion information is propagated to the rest network by overhearing data packets which are on fly. If congestion is detected, a notification bit is piggybacked in data packet's header or in ACK packets (when used). Implicit congestion notification avoids the addition of extra packets to the network when it is already congested.

It is clear that explicit congestion mitigation, although it is more accurate than implicit method, has been abandoned due to the fact that is power consuming. Thus, after the first efforts [34, 38] all subsequent algorithms adopted implicit congestion notification method.

3. **Congestion Counteraction Mechanisms:** Finally, a classification can be performed concerning the way that algorithms react to congestion in order to mitigate it. Congestion is mitigated either by rate reduction (traffic control) or by the creation of alternative paths

from the source(s) to the sink(s) for forwarding the excess data packets (resource control) (Fig. 6).

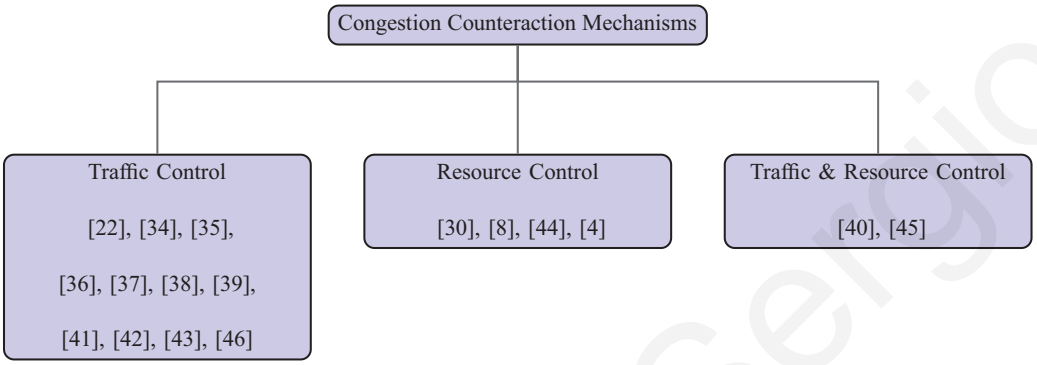

Figure 6: Congestion Mitigation Algorithms: Counteraction Mechanisms.

- **Traffic Control:** Traffic control mechanisms control the data load that is injected in the network when traffic demand is near or exceeds the capacity of network resources, leading to congestion. Traffic control can be seen as a means of taking traffic reducing steps, such as reducing the amount of packets injected into the network, to alleviate congestion. Traffic control may follow a window-based, or a rate-based approach.

  In the windows-based approach, a sender probes for the available network bandwidth by slowly increasing a congestion window (used to control how much data is outstanding in the network); when congestion is detected (indicated by the loss of one or more packets), the protocol reduces the congestion window greatly. The rapid reduction of the window size in response to congestion is essential to avoid network collapse. One of the most popular window-based approaches for adjusting sending rates is the additive increase multiplicative decrease (AIMD) policy. A number of congestion control approaches in wireless sensor network [34,36,40], are based on the AIMD policy. The advantage of this policy is that an AIMD-based protocol is agnostic to the underlying

link layer, requiring no prior knowledge of the available capacity. However, the AIMD policy is shown to provide unsatisfactory performance in wireless environments where high packet loss rates are often attributed to the time-varying conditions of the wireless channel, e.g., interference, multi-path fading, etc. Therefore, the resulting saw-tooth rate behavior may violate the QoS requirements (e.g. fidelity of the reported events).

Rate-based approaches attempt to estimate the available network bandwidth explicitly. This can be achieved, for example, by using a throughput formula, or empirically derived directives. In [46], a deterministic population balance equation inspired from biological systems is used as a throughput formula to adjust the sending rate of sensor nodes. In [64], a fuzzy logic based approach was proposed to combine a set of equations and rules to evaluate the traffic rates at source nodes. Furthermore, rate adjustments can be performed on the basis of empirically derived regions of operation [57].

One of the advantages of traffic control is that the burden of congestion alleviation lies, in most of the cases, to only one node, the source node. Also, when traffic reduction is applied congestion can be alleviated relatively quickly since the load in the network decreases. On the other hand, traffic control is not efficient for event-based networks, where the network becomes active when sensor nodes are triggered by an event. This is because traffic reduction can jeopardize the network's mission since all data packets carry valuable information about the event. In addition, traffic control can not be efficiently used during transient congestion phenomena caused by aperiodic and short term packet bursts due to the slowness of traffic control mechanisms (e.g. rate reduction) to react.

- **Resource Control:** To eliminate the disadvantage of the traffic control method, an alternative method, called resource-control has been proposed [8, 30]. In this case, when the network is congested (either in the medium or in the buffers), data packets follow alternative paths, which are not congested, in order to be forwarded to sink. This method has the advantage that traffic control is avoided and all data packets have a great potential to reach the sink. On the other hand, special care needs to be taken in order to meet the performance requirement like packet travel time, avoidance of loops, etc.
- **Traffic and Resource Control:** Some algorithms employ both methods in their effort to face congestion. This way is actually a hybrid method that attempts to trade on the advantages of both methods.

Choosing the best counteraction method for congestion is not a trivial effort. According to our opinion, the choice is application depended. Traffic control better applies to transient congestion situations, or in applications where reducing the rate with which sources are injecting data in the network is acceptable. Resource control better applies in applications where all data need to be transferred to the sink and in cases where permanent congestion situation is expected, provided that the network is dense enough, in order to provide availability of alternative paths.

4. **Further Attributes:** Congestion Control algorithms can also be classified in other parameters like the following:
   - Traffic Direction, if it is upstream or downstream.
   - Transport of packets, if it is Hop-by-Hop or End to End.
   - Whether it supports fairness among the nodes.

- Whether it supports multiple classes (e.g. high priority or low priority packets).
- Whether it states clearly or it is evident from any experimentation results that it conserves energy.

2.4.2 Congestion Avoidance

Congestion avoidance algorithms can be classified on the way they detect that congestion is going to happen and on the mechanism they use to avoid congestion.

1. **Congestion Detection:** Similar to congestion mitigation algorithms, congestion avoidance algorithms employ respective methods in order to detect congestion, with the difference that act in a preventive instead of reactive way. These are buffer occupancy, wireless channel load, or both of them as well as "per node load collection technique" appeared in [49].

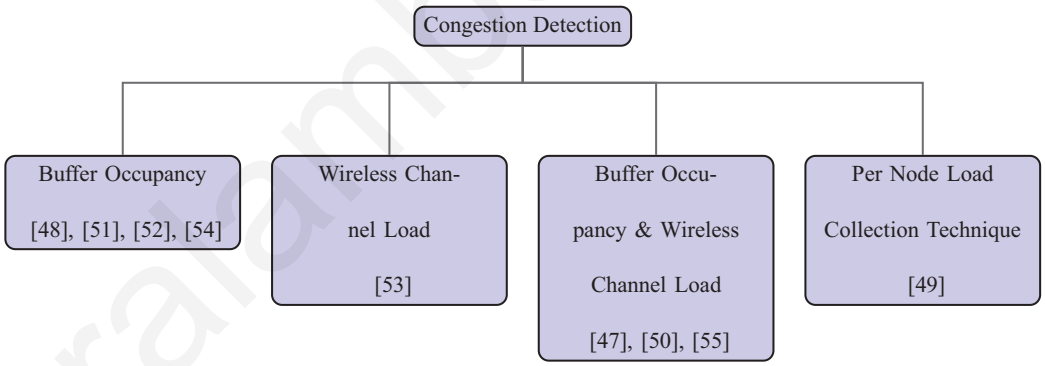

Figure 7: Congestion Avoidance Algorithms: Congestion Detection Mechanisms.

2. **Congestion Avoidance Mechanisms:** Congestion can be avoided using similar techniques as with congestion mitigation (traffic, resource control or both) with the difference that also in this case congestion avoidance algorithms act in a preventive way instead of reactive way. Other techniques have also been also been introduced. All methods are presented in Fig. 8.

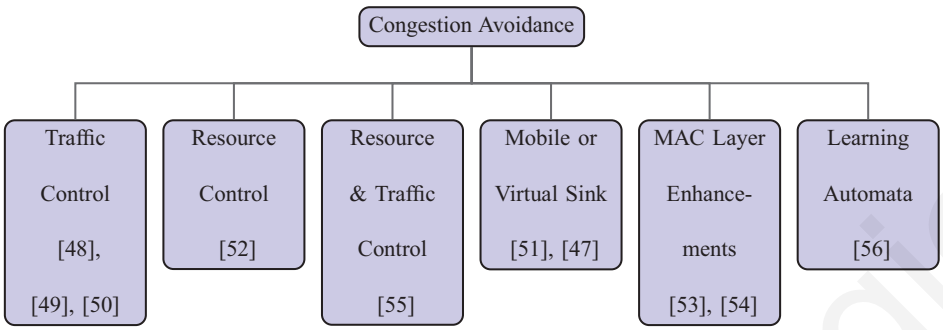

Figure 8: Congestion Avoidance Algorithms: Congestion Avoidance Mechanisms.

- **Mobile or Virtual Sink:** In these cases the sink is placed near the regions that show signs of high load. In the case of mobile sink [51] the network is split into clusters and there, an in-network storage model is introduced. Cluster heads collect all load from their nodes and transmit them to the mobile sink when it passes near the cluster. On the other hand, [47], introduced the concept of mini-sinks where some nodes with longer communication range, tunnel the traffic event from regions that are going to become congested.
- **MAC Layer Enhancements:** MAC layer enhancements are used in the MAC layer in order to avoid collisions in the medium. Implementations span from priorities in heavy loaded nodes [35, 54] to transmission phase shifting [22].
- **Learning automata:** In this case, code programmed to take intelligent actions (called automata) is developed at each of the network's nodes that are capable of controlling data flow rate at the intermediate nodes, based on, probabilistically, how many packets are likely to get dropped, if a particular flow rate is maintained.

Concerning congestion avoidance mechanisms the same comments stand, as for the congestion counteraction mechanisms for congestion mitigation algorithms. Concerning the rest

of the methods we believe that mobile or virtual sink is a promising idea but more effort is needed in order to be implemented in real scenarios. MAC layer enhancement is a very efficient and effective method for collision avoidance. Finally, concerning learning automata we can not be so fair in our judgement since we have seen it in just one paper [56].

2.4.3 Reliable Data Transport

Reliable data transport can be considered as a different category of transport layer protocol that focus in the reliable data transmission of information packets. But since they also provide congestion control they can be also considered as part of this work.

Reliable data transport protocols can be divided based on three basic attributes. These are traffic direction (Fig. 9), if they employ end-to-end of hop-by-hop reliability (Fig. 10), as well as where the reliability focus on (Fig. 11).

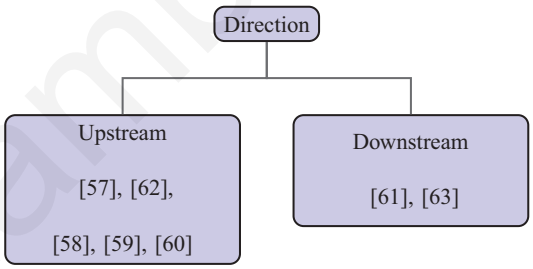

Figure 9: Reliable Data Transport Protocols: Direction.

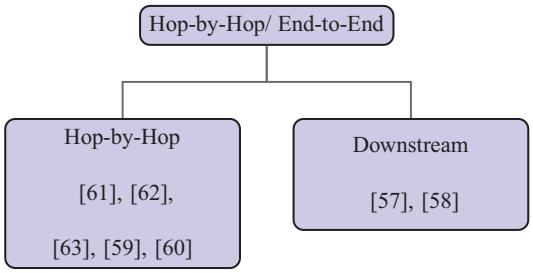

Figure 10: Reliable Data Transport Protocols: Hop-by-Hop/ End-to-End.

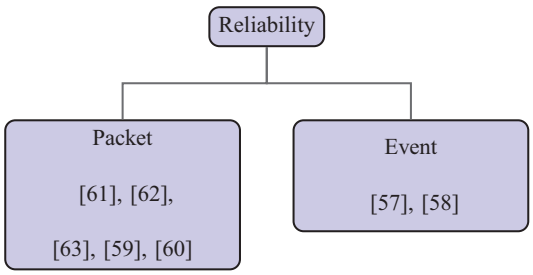

Figure 11: Reliable Data Transport Protocols: Reliability.

2.5 Common Congestion Control Metrics

In this section we present the most common metrics that the majority of algorithms use, in order to evaluate the performance of their efforts.

- **Packet delivery ratio:** This metric is used in order to measure the efficiency of the algorithm concerning the delivery of packets to sink. Packet drops are normally measured as the percentage of the total packets that are received by sinks divided by the number of packets that are produced by sources. The more closer to 100% the result is, the more efficient the algorithm.

- **Throughput:** Throughput is defined as the number of packets per unit time that is received by the sink. The higher the value of throughput is, the more efficient is the algorithm.

- **End-to- End delay:** This metric is used in order to measure the required time for a packet to reach the sink. This metric is an indication of the efficiency of the algorithm to quickly mitigate or avoid congestion. Small delay denotes increased performance, since high delay is incurred in congested hotspots due to retransmissions or due to long routes (in the cases a resource control algorithm is used)

- **Hop-by-hop delay:** Hop-by-hop delay is also a metric that measures the efficiency of the algorithm is terms congestion and overhead, since when congestion is avoided high queuing delays are also avoided.

- **Network Lifetime:** This metric reflects the long-term energy efficiency of the network. When the power of the nodes is uniformly exhausted then this value increases. This metric is usually high in resource control algorithms.

- **Average node energy consumption:** This metric indicates the energy consumption of nodes. The value of this metric should be kept low in order to indicate an efficient congestion control algorithm.

2.6 Extended Summary of Selected Algorithms

In this section we present an extended summary of the algorithms that we employ in this thesis, for comparison with the algorithms that we propose (HTAP and DAlPaS). These algorithms are ESRT [57], Directed Diffusion [24], SenTCP [39] and TARA [30].

ESRT

Sankarasubramaniam et al. proposed Event to Sink Reliable Transport (ESRT) [57]. ESRT considers reliability at the application level and provides stochastically reliable delivery of packets from sensors to the sink. It is an end-to-end protocol trying to guarantee a desired reliability through regulation of sensor report frequency. It provides reliability for applications, not for each single packet. The sink uses congestion feedback from sensor nodes to broadcast a notification to adjust the reporting rate with two goals: i) to receive a sufficient number of packets from the sink,

and ii) to receive only as many packets as necessary in order to avoid congestion and save energy (Fig. 12).

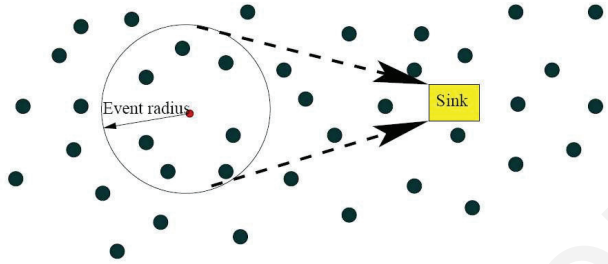

Figure 12: Event Detection in ESRT.

ESRT algorithm runs on the sink, while minimal functionality is required at resource constrained sensor nodes. ESRT operation is determined by the current network state, based on the reliability achieved and congestion condition in the network. Firstly, it needs to periodically compute the factual reliability r based on successfully received packets in a time interval. Secondly, ESRT deduces the required sensor report frequency f from r. Thirdly and finally, ESRT informs all sensors about f through an assumed channel with high power. ESRT identifies five characteristic operation regions: i) No Congestion, Low reliability, ii) No Congestion, High reliability, iii) Congestion, High Reliability, iv) Congestion, Low Reliability and v) Optimal Operating Region - which essentially translates to No Congestion, Medium-High Reliability. The target is to identify its current state and bring the network into OOR (Optimal Operating Region), according to Fig. 13.

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

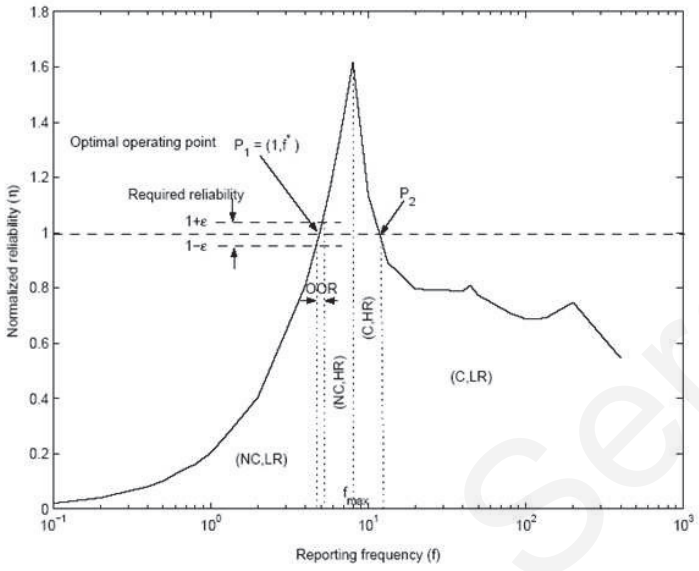

Figure 13: Five Characteristics Regions of ESRT.

renders it robust in random and dynamic topologies of WSNs. An additional benefit resulting from ESRT is energy-conservation since it can control the sensor reporting frequency.

A disadvantage of this algorithm is the fact that all nodes are treated equally. Therefore, in case of congestion in one region of the network, all nodes are forced to reduce their data rate, thus affecting negatively the network's throughput. Moreover ESRT does not handle different types of events requiring different levels of reliability. Summarizing ESRT is an end-to-end congestion control protocol that focuses on reliability. It provides fairness among the nodes since data rate reduction applies in all nodes in the network, even if congestion appears in a specific area in the network. ESRT has been implemented in ns-2 [65] simulation tool and provided results concern reliability.

Directed Diffusion

Intanagonwiwat et al. proposed Directed Diffusion [24]. Directed Diffusion is a data centric protocol because all communication is for named data. All nodes in a directed diffusion-based network are application-aware. This enables diffusion, to achieve energy savings by selecting empirically good paths (small delay) and by caching and processing in-network data (e.g., data aggregation). Directed diffusion consists of four basic elements: interests, data messages, gradients, and reinforcements. An interest message is a query from a sink node to the network, which indicates what the application wants. It carries a description of a sensing task that is supported by a sensor network. Data in sensor networks is the collected or processed information of an event (e.g. physical phenomenon), is named (addressed) using attribute-value pairs and a sensing task is diffused throughout the sensor network as an interest for named data. This dissemination sets up gradients within the network designed to "draw" events (i.e., data matching the interest). A gradient is direction state created in each node that receives an interest. This direction is set toward the neighboring node from which the interest was received. Events start flowing towards the sinks of interests along multiple gradient paths. To improve performance and reliability, the empirically "good paths" (e.g small delay) are reinforced by the sink and their data rate increases. On the other hand unreliable paths (e.g high delay) are negatively reinforced and pruned off. Directed Diffusion has been implemented in ns-2 [65] simulation tool. Results are provided in comparison with Omniscient Multicast and flooding and concern average dissipated energy, average delay and event delivery ratio.

SenTCP

Wang et al. proposed SenTCP [39]. SenTCP is an open-loop hop-by-hop congestion control protocol with two special features. Firstly, it jointly uses average local packet service time and average local packet inter-arrival time in order to detect congestion, by estimating the current

local congestion degree in each intermediate sensor node. The use of packet arrival time and service time not only precisely calculates congestion degree, but effectively helps to differentiate the reason of packet loss occurrence in wireless environments, since arrival time (or service time) may become small (or large) if congestion occurs.

Secondly, SenTCP uses hop-by-hop congestion control. In SenTCP each intermediate sensor node will issue feedback signal backward and hop-by-hop. The feedback signal, which carries local congestion degree and the buffer occupancy ratio, is used for the neighboring sensor nodes to adjust their sending rate in the transport layer. The use of hop-by-hop feedback control can remove congestion quickly and reduce packet dropping, which in turn conserves energy. SenTCP was tested on a simulated simple linear-like topology of 20 source nodes and was compared to TCP. Simulation results and comparison with TCP showed that SenTCP can reduce packet dropping, resulted from buffer overflow and in-turn energy would be conserved. SenTCP also effectively overcame the problem of differentiating congestion and packet error loss. The throughput of SenTCP is almost not influenced by packet error probability.

TARA

Kang et al. proposed the Topology Aware Resource Adaptation (TARA) protocol [30]. TARA focuses on the adaptation of the network's extra recourses in case of congestion, alleviating intersection hot spots. A graph-coloring problem is used to determine the needed topology for the resource adaptation strategy. TARA measures not only the buffer occupancy but also the channel loading in order to detect congestion. As soon as the congestion level of a node hits the upper watermark, it declares congestion and becomes a hot spot node. At this point, the hot spot node needs to quickly locate two important nodes: the distributor (node G) and the merger (node J), according to Fig. 14.

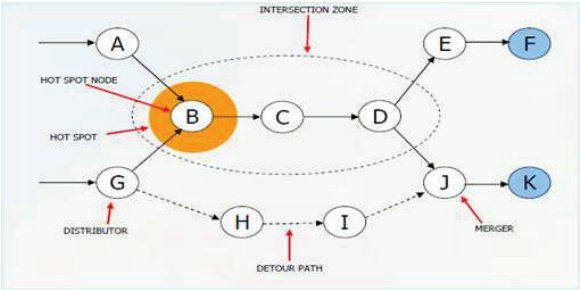

Figure 14: Hotspot in TARA Algorithm.

Then, a detour path can be established, starting at the distributor and ending at the merger. As suggested by their names, the distributor distributes the incoming traffic between the original path and the detour path, whereas the merger node, merges these two flows. Thus, in the case of congestion and the creation of hot-spot, traffic is deflected from the hot-spot through the distributor node along the detour and reaches the merge node, where the flows are merged. As soon as congestion has been alleviated the network stops using the detour path. For quick adaptation, the distributor node keeps in its memory which neighbor is on the original path. The performance of TARA was evaluated on a random topology of 81 nodes simulated in ns-2. Detailed simulation results have shown that TARA can energy efficiently absorb incoming traffic load. The results have also demonstrated that TARA performs very close to an ideal off-line resource control algorithm in terms of both fidelity satisfaction and energy conservation. TARA was compared against five strategies for congestion control which do not correspond to protocols or mechanisms found in nature. Results showed that TARA outperformed all these strategies. However, it is worth pointing out that TARA requires knowledge about the whole network topology, which makes the protocol impractical for large scale networks.

2.7 Summary and Design Guidelines

Summarizing on the survey of current solutions on congestion control and avoidance in WSNs, we notice that the majority of these algorithms employs traffic control for congestion mitigation in WSNs. As we state in Chapter 3, traffic control can be proven insufficient in many cases since a single bottleneck in the network is able to reduce the traffic from sources. This fact is not acceptable in many critical applications. Hence, we focus on the design of algorithms that employ resource control. Resource control can also assist in the uniform energy utilization of the network which leads to the extend of network's lifetime. The problem of existing resource control algorithms, is the fact that they do not provide strict performance bounds. Thus, in our effort, performance assurances comprise a critical factor for the design of congestion control and avoidance algorithms.

Concerning congestion notification, almost all algorithms, besides few initial efforts [34, 38] employ implicit ways. We agree with this approach since explicit notifications add unnecessary load to the already heavy loaded network. Finally, concerning the way that congestion is detected we prefer to choose the combination between wireless channel load and buffer occupancy. These techniques are efficient enough to indicate a congestion event in WSNs.

Thus, the rough guidelines for the design of our congestion control and avoidance algorithms are the following:

- Resource control for congestion mitigation.
- Implicit notification.
- Congestion detection based on buffer occupancy and wireless channel load.
- Guaranteed high and stable performance.

A short description of the algorithms referred in this Chapter, as well as relevant comparative tables can be found in Appendix A.

Chapter 3

Analysis Through a Fluid Dynamic Model

In order to enhance the design guidelines that we have extracted from the literature review, presented in the previous Chapter, we attempt to analyze and model traffic flows in WSNs. Specifically, we attempt to model free flowing data flows and examine how interruptions in data flow affect the performance of WSNs.

To perform this analysis, we employ a macroscopic fluid dynamic model. We intentionally choose a macroscopic model since it is well known that the behavior of WSNs is a complex and difficult task because the effects of significant network parameters are frequently unpredictable. This, along with the fact that in most network deployments, wireless sensor nodes are densely and randomly deployed, renders the individual study of the behavior of each sensor node impractical.

In this chapter we attempt to analyze, model, and estimate the maximum volume of traffic than can be carried out from the sources to the sink(s) of a WSN, without the use of any congestion control algorithms. To perform our analysis we employ a macroscopic fluid dynamic model. Using this model and three fundamental traffic variables, packet density, packet flow, and spatial packet rate, we calculate the limits of the network flows, in terms of capacity, in the absence of congestion control. Calculating these limits helps us prove a relation between incoming and outgoing flow

in the bottleneck nodes that can specify the optimal point at which the network should operate without the need of congestion control algorithms.

Studying the behavior of wireless networks is not a trivial task since a lot of parameters, usually with unpredictable behavior, are involved in this process. The situation is even more complex for WSNs, where the placement of nodes is frequently dense and random, while a big number of nodes transmit data concurrently in the case of an event. Thus, a number of authors [66–68], instead of studying the individual attitude of each node in terms of their microscopic parameters, choose to study the data flow from sources to sinks macroscopically, using fluid or other models. The most related (and to the best of our knowledge, the only) works in the literature employing fluid dynamics models are briefly described here.

One of the very first efforts that introduced deterministic fluid models for modeling the traffic of wireless communications networks appeared in 1994 by Leung et al. [66]. In this work the authors focus on wireless telephony and provide mathematical models to help understand system dynamics and analyze the performance of these networks. Specifically, they consider a highway with multiple entrances and exists, while vehicles can be in a calling or non-calling state. In order to perform their study, they introduce a deterministic fluid model and two stochastic traffic models for wireless networks. The deterministic model ignores the behavior of individual vehicles and treats them as a continuous fluid.

Gribaudo et al. [67] claim that the behavior of large-scale WSNs is complex and difficult to analyze, thus they develop an analytical model of the behavior of WSNs, based on a fluid approach. Actually, they represent WSNs by a continuous fluid entity distributed on the network area.

Toumpis et al. [68] investigate the spatial distribution of wireless nodes that can transport a given volume of traffic in a sensor network, while requiring the minimum number of wireless nodes. In that work they show that under specific assumptions the optimal distribution of

nodes induces a traffic flow identical to the electrostatic field that would exist if the sources and sinks of traffic were substituted with an appropriate distribution of electric charges. The authors introduce three macroscopic quantities: the information density function, the node density function, and the traffic flow function. Furthermore, they suggest a relation between the node density and the traffic flow which is based on the fundamental assumption that a location (x, y), where the node density is $d(x, y)$, can support any traffic flow vector with a magnitude less or equal to a bound $|\boldsymbol{T}(x, y)|_{\max}$ which is proportional to the square root of the density, i.e. $|\boldsymbol{T}(x, y)| \leq |\boldsymbol{T}(x, y)|_{\max} = K\sqrt{d(x, y)}$.

Concerning the estimation of the maximum volume of traffic that can be carried out from the sources to the sink, this depends on the physical and MAC layers.

Specifically, Toumpis et al. [68] studied the case when a simple time division MAC protocol that consists of only three slots is employed. In this scenario they considered a grid of $m \times m$ nodes with a source data rate of W bps. Using this placement they assumed that each node can listen to the transmission of its four nearest neighbors (or if it is located at the edge, to the two or three nearest nodes). Using these data, they calculated that the maximum traffic that is able to transverse this network is equal to $\frac{mW}{3}$.

Under a similar scenario Silvester et al. [69] proved that if, instead of a time division MAC protocol, a slotted Aloha is employed, the maximum possible traffic between the nodes is $k \times W \times m$ where k is a constant less than $\frac{1}{3}$.

Franceschetti et al. [70] considered a contention-based environment. They proved that under the best conditions the maximum possible traffic that can be carried out in this network from source to sink is $\Theta(W\sqrt{m})$. This can be achieved when the flows are completely disjoint and each flow consists of $\Theta(\sqrt{m})$ wireless nodes. Each node is able to carry $k_1 W$ bps, where constant k_1 captures the effects of a node having to share the channel with competing nodes.

In this Chapter we attempt to analyze, estimate, and model the maximum traffic volume that can exist in a WSN, in order to operate properly, without the need of congestion or overload control algorithms, focusing at the packet level and not at the node level as [67] and [68] do.

Specifically, we initially present a macroscopic version of a conservation of information law and we prove that this law can stand in WSNs when the maximum capacity of the network is not exceeded. In this case the data flow in the network can be represented by a continuity equation as in fluid dynamics. To the best of our knowledge this is the first time that this equation is being used in the context of Wireless Sensor Networks. Then, using the continuity equation we can prove a relation between the incoming and outgoing flow in the bottleneck nodes, which can set the limits in data flows. Using these limits we can estimate the maximum traffic volume of the network and we calculate the speed with which traffic interruption propagates to the backwards nodes, when no congestion control algorithm applies, causing queues formation.

3.1 System Model

Initially, we consider a number of disjoint data flows with space variable $x \in \mathbb{R}$ and time variable $t \geq 0$ (Fig. 15) moving in one direction (e.g. from source to sink). At this point we introduce three macroscopic traffic variables: (i) packet density function $\rho(x,t)$, (ii) spatial packet rate function (packet velocity) $v(x,t)$, and (iii) flow $f(x,t)$. As packet density $\rho(x,t)$, we consider the density of packets at time t at point x measured in packets per distance unit. The spatial packet rate $v(x,t)$ or packet velocity is the rate of packets at time t at point x, measured in length over time unit. Instead of keeping track of the spatial rate of each packet in the network, we assign to each point in the network a spatial packet rate field $v(x,t)$. Thus, flow is the product of $\rho(x,t) \times v(x,t)$, which is the number of packets that pass from a specific point in the network at time t and it is measured in packets per time unit.

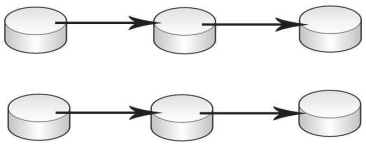

Figure 15: Disjoint Data Flows.

We also assume that the following statements hold:

- Packets are aware of their destination nodes.
- Lost packets are retransmitted.

The fact that lost packets are retransmitted until they reach the next hop is crucial, since if this statement holds, we can safely claim that at the macroscopic level the number of packets is "conserved".

3.1.1 Conservation of Information (Packets)

In this subsection we prove that a macroscopic version of a conservation of information law holds in WSNs, bearing in mind the assumptions stated above. Specifically, as we stated before, flow $f(x,t)$ is the product of $\rho(x,t) \times v(x,t)$.

Thus, the initial traffic density at time $t = 0$ at point x is $\rho(x,0)$. The movement of each packet can then be specified by calculating the derivative of point x with respect to t. This movement satisfies the following first order differential equation:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v(x,t)\ with\ x(0) = x_0. \tag{1}$$

The solution of (1) determines the position of each packet as the flow is evolving.

Let us consider a specific route segment between nodes n_j and n_k which is specified by $x = a$ and $x = b$ as it appears in Fig. 16. Now, we would like to calculate the number of packets in this segment, if we know the spatial packet rate.

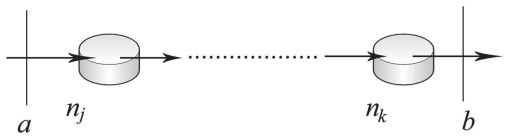

Figure 16: Packet Flow Between a Part of Route.

For this specific network instance (Fig. 16) it is possible to calculate the number of packets P, by integrating the traffic density:

$$P(t) = \int_a^b \rho(x,t)dx. \tag{2}$$

Assuming that no packets are created or destroyed in this segment (lost packets are retransmitted), the number of packets between $x = a$ and $x = b$ can only change due to the number of packets that enter $x = a$ and the number of packets that leave from n_k at $x = b$. Therefore the traffic flow $f(a,t)$ and $f(b,t)$ is not the same as the time is evolving.

The rate of change of packets with respect to time, dP/dt, is equal to the number of packets per time unit entering the network segment $[a,b]$ at $x = a$ minus the number of packets per time unit exiting the segment $[a,b]$ at $x = b$. In this case the packets are always moving upstream (from source to sink) and the rate of change of the number of packets per unit time, is the traffic flow at position a minus the flow at position b, both at time t, according to (3).

$$\frac{dP}{dt} = f(a,t) - f(b,t). \tag{3}$$

Taking the derivative of both sides of (2) with respect to time gives the following:

$$\frac{dP}{dt} = \frac{d}{dt}\int_a^b \rho(x,t)dx. \tag{4}$$

Combining (3) and (4) we have:

$$\frac{d}{dt}\int_a^b \rho(x,t)dx = f(a,t) - f(b,t). \tag{5}$$

$f(a,t) - f(b,t)$ can be substituted by taking the partial derivative of the right hand side of (5) with respect to x. Also taking the integral from $x = a$ to $x = b$, and assuming that $f(x,t)$ is smooth as a function of x and $\rho(x,t)$ is smooth as a function of t, we can apply the fundamental theorem of calculus which gives the following equation:

$$\int_a^b \frac{\partial}{\partial t}\rho(x,t)dx = \int_a^b -\frac{\partial f(x,t)}{\partial x}dx, \tag{6}$$

which can also be written as:

$$\int_a^b \left[\frac{\partial}{\partial t}\rho(x,t) + \frac{\partial f(x,t)}{\partial x}\right] dx = 0. \tag{7}$$

If we assume that the integrand is piecewise continuous, this can only hold for all route segments between by $x = a$ and $x = b$, if the integrand itself is zero. This gives us:

$$\frac{\mathrm{d}\rho(x,t)}{\mathrm{d}t} + \frac{\partial f(x,t)}{\partial x} = 0. \tag{8}$$

From (8) we get

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \frac{\partial f}{\partial x} = 0. \tag{9}$$

Equation (9) is a partial differential equation expressing the "conservation of information". This continuity equation states that the information (number of packets) in a wireless sensor network can be conserved, if the number of packets that enter the network do not exceed its capacity. Using this equation we can estimate the maximum traffic volume that can exist in a network without the need of congestion control.

3.2 Analytical Results

In this section we initially present a relation between packet density and traffic flow and then we study the case where flows merge in the network and create traffic bottlenecks.

3.2.1 Relation between packet density and traffic flow

At this point it is important to identify how the traffic flow rate varies in an established data flow when the packet density changes. In its turn, the packet density changes when the source reporting rate varies. To specify this relation we assume that the flow is a function of density and if $\rho \in [0, 1]$ then

$$f(\rho) = M \times \rho(1 - \frac{1}{k}\rho), \tag{10}$$

where M is a normalizing coefficient measured in bps that accounts for the MAC protocol characteristics, and k is a coefficient that captures the effects of a node having to share the channel with competing nodes. The interpretation of this assumption is that, if there are no other packets in the network, then a packet can travel with no delay in the intermediate node queues, using the shortest route. This fact maximizes the data flow. On the other hand, if the packet density is at its maximum, meaning that all buffers are full, then we can safely state that the data flow is approaching zero. The accuracy of this assumption depends on the physical layer and the MAC protocol used by the network.

This assumption follows a similar observation made in the *Event to Sink Reliable Transport (ESRT)* algorithm [57]. In this paper, the authors plot the reporting frequency of source nodes vs the reliability, which is defined as the number of packets received by the sink. Note that when the reporting frequency of the nodes increases, the number of packets that are injected in the network also increases. Fig. 17 shows that the number of packets received by the sink (reliability) rises to a peak point and then it decreases as the reporting frequency becomes larger.

Assuming that this relation provided in the ESRT algorithm holds for any node in the network and not just the sink, we can state that as the density of packets around a specific node increases (and this node has to forward these packets), then the flow (the number of packets that must

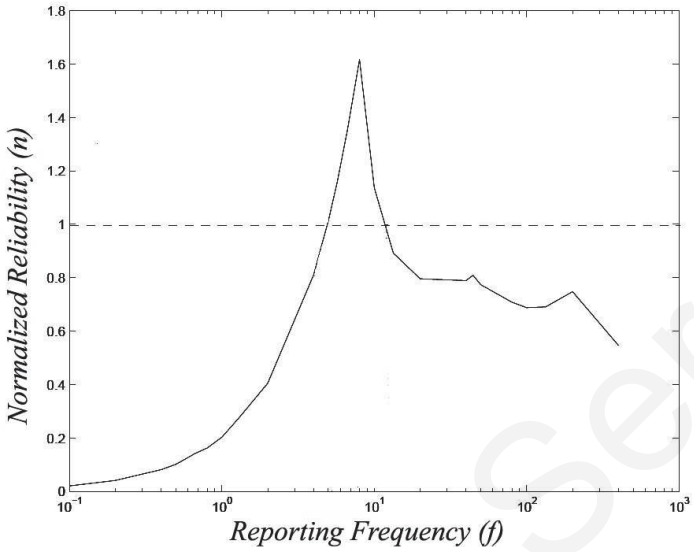

Figure 17: Normalized Reliability vs. Reporting Frequency.

be forwarded through this node in a specific time segment) reaches a maximum point and then decreases.

3.2.2 Traffic Bottlenecks

We consider two space variables $x, y \in \mathbb{R}$, while we study a contention-based scenario. Specifically, we use the results provided by [70] and we study the case when separate flows merge at a specific node in the network (Fig. 18).

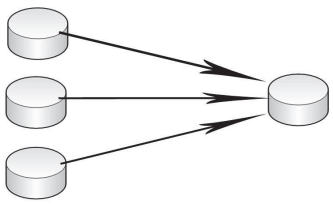

Figure 18: Separate Flows Merge at a Single Node.

According to [70], each node is able to carry kW bps, where constant k captures the effects of a node having to share the channel with competing nodes.

Initially we consider n active flows, which come from an equal number of nodes, and which cross on a specific node (the bottleneck). All other nodes in the network are in a sleep state and do not interfere with the active nodes of the network. All of the active nodes share the access medium and each one is, ideally, able to transmit $\frac{W}{n}$ bps, meaning that k is equal to $\frac{1}{n}$. It is clear that if the n flows do not take this restriction into account and continue to transmit packets at the full rate (W bps), then buffer-based congestion is going to appear at the nodes before the common node. These nodes will receive n times the number of packets they are able to forward, unless the value of ρ is calculated and used in reducing the source sending rate.

To calculate this value we must prove a relation between incoming and outgoing flows so as to avoid queue formation in the bottleneck node and subsequently to all backward nodes. Thus, we attempt to define the proper density of packets that the initial flow should have, in order to avoid the formation of queues at the bottleneck node.

We start by defining the initial conditions of the flows before (f_1) and after (f_2) the bottleneck node. Specifically, before the bottleneck node, we assume that according to (10) with coefficients $M = 1$ and $k = 1$ ($n = 1$), the flow function of each active flow is:

$$f_1(\rho_{1,0}) = \rho_{1,0}(1 - \rho_{1,0}), \rho_{1,0} \in [0, 1], \tag{11}$$

while after the bottleneck node (in the congested part) the flow function is

$$f_2(\rho_{2,0}) = \rho_{2,0}(1 - \frac{1}{k}\rho_{2,0}), \rho_{2,0} \in [0, 1] \text{ and } k = \frac{1}{n}. \tag{12}$$

As a general example we provide an analysis when the number of flows $n = 2$, i.e. two different flows merge at the common node. Plotting (11) and (12), for $k = \frac{1}{n} = \frac{1}{2}$, we get Fig. 19.

Since the flow is restricted after the bottleneck, we must calculate the maximum flow that is able to pass through the bottleneck. Studying Fig. 19, it is clear that it is enough to calculate the maximum point of the curve that represents the restricted flow. Thus, we calculate the points where the first derivative of the function $f_2(\rho_{2,0})$ turns to zero, as follows:

$$f_2'(\rho_{2,0}) = 0 \Rightarrow [\rho_{2,0}(1 - \frac{1}{k}\rho_{2,0})]' = 0 \Rightarrow \rho_{2,0} = \frac{k}{2}. \tag{13}$$

The implication for the case analyzed here is that in order to avoid queue formation, the flow entering the bottleneck node should be limited to $f(\rho) = \rho(1 - \frac{1}{k}\rho)$ where $\rho = \frac{k}{2}$. The result is that $f_1(\rho)$ should not be greater than $\frac{k}{4}$.

To calculate the point where queue formation begins (for the nodes before the bottleneck) we solve the following equation with respect to the density ρ.

$$f(\rho) = \rho(1 - \rho) = \frac{k}{4} \Leftrightarrow \bar{\rho} = \frac{1 - \sqrt{1 - k}}{2}. \tag{14}$$

Therefore, if we do not want any queue formation, the boundary condition on the initial flow should be $f_1(\rho_1, 0) < f(\bar{\rho})$.

Finding this point in Fig. 19, for $k = 1/2$, we notice that queue formation starts very early (when ρ is 0.146 and f is 0.125) which is well before the density value which provides the maximum flow at the bottleneck node.

3.3 Characteristic Speed and Queue Formation Rate

3.3.1 Characteristic Speed

Using the conservation law (9) and (10) we have:

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \frac{\partial f(\rho)}{\partial x} = 0 \tag{15}$$

where $f(\rho) = \rho v(\rho)$ is the flow and the problem is now solvable for ρ.

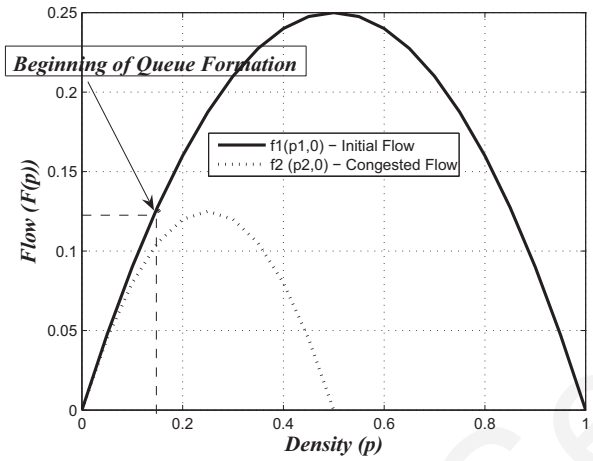

Figure 19: Beginning of Queue Formation.

If both function $f(\rho)$ and the solution $\rho(x,t)$ are differentiable, this equation can be written by developing the spacial derivative to give continuity in equation:

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + s(\rho)\frac{\partial f(\rho)}{\partial x} = 0 \tag{16}$$

where $s(\rho)$ represents the "characteristic speed". "Characteristic speed" is, therefore, the speed with which fluctuations propagate in the traffic flow, given a density ρ.

3.3.2 Queue Formation Rate

Using (10) we initially study the case, in which a node is unable to forward the traffic that it receives and queues are forming in its buffer.

We consider as $x = 0$ the point where the queue begins. Using 10 we have

$$\rho_0(x) = \begin{cases} \rho_l & \text{if} \quad x < 0 \\ 1 & \text{if} \quad x > 0 \end{cases} \tag{17}$$

with $0 < \rho_l < 1$. Based on (17), if $x > 0$ the flow is congested and nodes are unable to forward a single packet, while packets still arrive from sources to the queue ($x < 0$). It is definite that in

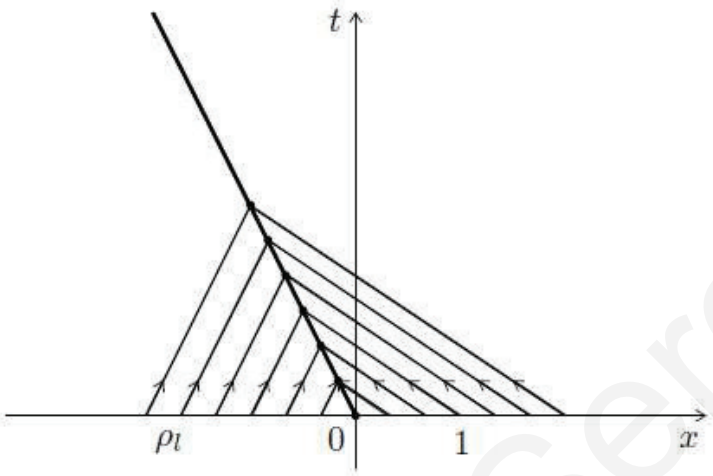

Figure 20: Characteristics.

such a case, if no congestion control actions are taken, queues are going to built up and since node buffers are not infinite, packet drops will happen.

To solve this partial differential equation we have to use the "method of characteristics" [71]. As we notice a discontinuity appears in this equation when $x = 0$, which means that we must use the initial conditions (17) in order to provide a "weak solution".

Thus, for $f(\rho) = \rho(1-\rho)$ we have $f'(\rho) = 1 - 2\rho$,, and the equation of characteristics is

$$\begin{cases} x - (1 - 2\rho_l)t = x_0 & \text{if} \quad x_0 < 0 \\ x - t = x_0 & \text{if} \quad x_0 > 0 \end{cases} \tag{18}$$

Plotting the above equations in Fig. 20, we notice that they intersect along a straight line issuing from $x_0 = 0$. This line is actually the location of the discontinuity at successive time $t > 0$ and has an equation of the form $x - st = 0$. In this equation $s < 0$ is the slope.

In order to calculate s we can use the well known Rankine-Hugoniot (Chapter 5 [71]) relation, in order to show that this discontinuity between the two different traffic densities ρ_l and ρ_0 travels

with speed equal to:

$$s = \frac{f(\rho_l) - f(\rho_0)}{\rho_l - \rho_0} \tag{19}$$

In this wireless sensor networks problem, s is the speed with which the queue of packets increases (i.e. propagates backwards), as new packets join the queue. Equation (19) actually proves that the speed with which node queues grow or shrink depends on the differences of the incoming and outgoing flows through this specific node at time t. Thus, buffer queues grow when the flow of incoming packets is higher than the flow of transmitted packets, while the actual difference in the value of the flow, defines the time at which the buffer is going to fill up. This means that any congestion control mechanism that employs a traffic control method (e.g. reduces the source data rate in order to control congestion) should be able to calculate this speed, in order to suppress the rate of incoming flow and avoid buffer fill-up. Using this value we can calculate the expected time that a buffer of B capacity is going to fill up and start dropping packets. The time is

$$t = B/s, \tag{20}$$

if the buffer is completely empty. By using this speed s the network is able to understand very early a possible congestion situation.

3.4 Numerical Results

To evaluate the theoretical results of the previous sections, a series of simulation have been performed using the Prowler [72] simulator, a probabilistic wireless network simulator. Prowler provides a radio fading model with packet collisions, static and dynamic asymmetric links, and a CSMA MAC layer serving as a contention based MAC protocol.

To perform the simulations we have used the radio propagation model provided by Prowler. The transmission model is given by:

$$P_{rec,ideal}(d) \leftarrow P_{transmit}\frac{1}{1+d^{\gamma}} \tag{21}$$

where, $2 \leq \gamma \leq 4$. Equation (21) presents an ideal transmission function with no errors. In order to provide realistic conditions to our simulations we add fading effects to the radio propagation model according to (22):

$$P_{rec}(i,j) \leftarrow P_{rec,ideal}(d_{i,j})(1+a(i,j))(1+\beta(t)) \tag{22}$$

where $P_{transmit}$ is the signal strength at the transmitter and $P_{rec,ideal}(d)$ is the ideal received signal strength at distance d. Variables a and β are random variables with normal distributions $N(0,\sigma_a)$ and $N(0,\sigma_\beta)$, respectively. A node j can receive packets from node i if $P_{rec}(i,j) > \Delta$ where Δ is the threshold.

3.4.1 Simulator Setup

The average radio range of transmission for each node is a radius of 10m. In our simulations we employed the default values of Prowler simulator, a choice made by several authors [73–76]. Specifically, we set $\sigma_a = 0.45$, $\sigma_\beta = 0.02$ and the reception threshold is set to $\Delta = 0.1$. Also we set $p_{error} = 0.05$. This parameter (p_{error}), models the probability of a transmission error caused for any other reason. These values add fading effects to the ideal transmission function. In particular, they model an imperfect circle. Varying the values of these parameters will affect the shape of the transmission radio range. Fig. 21a presents a snapshot of Prowler Simulator when the default parameters are employed, while Fig. 21b presents the results when we double the values of these parameters. Studying the related graph on the snapshots, that represents the signal power in relation to the distance, we notice that there is some distortion at low distance. But since this

variation is related with the signal transmission at the physical layer, it cannot affect the overall conclusions that we extract in this thesis, since are work is on MAC and Network layers. Thus, we use the same default values for these parameters throughout the thesis.

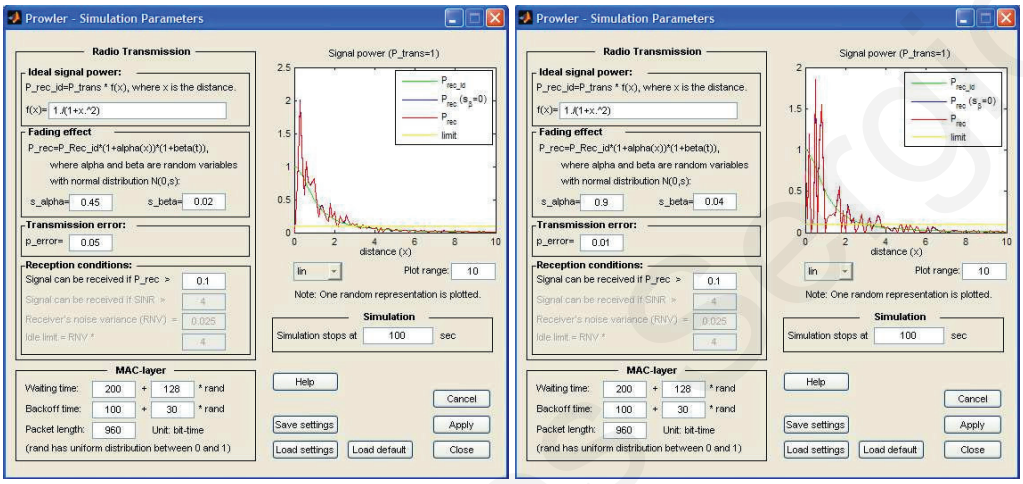

Figure 21: (a)Default Simulator Parameters (b) Variable Simulator Parameters.

The rest of the parameters we employ represent the Mica-Z node and the most important of them are presented in Table 1.

Table 1: Simulation Parameters.

Max Data Rate (kbps)	250
Transmission Power (dbm)	2
Receive Threshold (dbm)	-74
Transmission Current (mA)	17.4
Receive Current (mA)	19.7
Packet Size (bit)	1024
Buffer Size(Bytes)	512K
MAC layer	CSMA/CA

To perform the experiments we deployed a network with 11x11 nodes, which are placed strictly on the crossed lines of the grid, according to Fig. 22. We consider that there are two event sources at the corners of the topology on the left hand side of the grid and their data converge to the node which is at the center of the grid before being forwarded to the sink. To achieve this, we set as active nodes the nodes which are shaded in Fig. 22, while the rest of the nodes are in sleep state. Also a simple multicast routing protocol is employed, which is able to deliver packets to all nodes that are one hop away from the sending node.

Packets are transmitted in a hop-by-hop fashion, starting from each source node and using the upstream nodes that are one hop away, eventually reach the three nodes before the converging node at the center of the grid (C.N). This topology provides the implementation of the scenarios discussed in the previous sections and especially in Fig.18. As we stated above we use CSMA/CA MAC layer, and the three nodes before the "congested node" compete for access to this node.

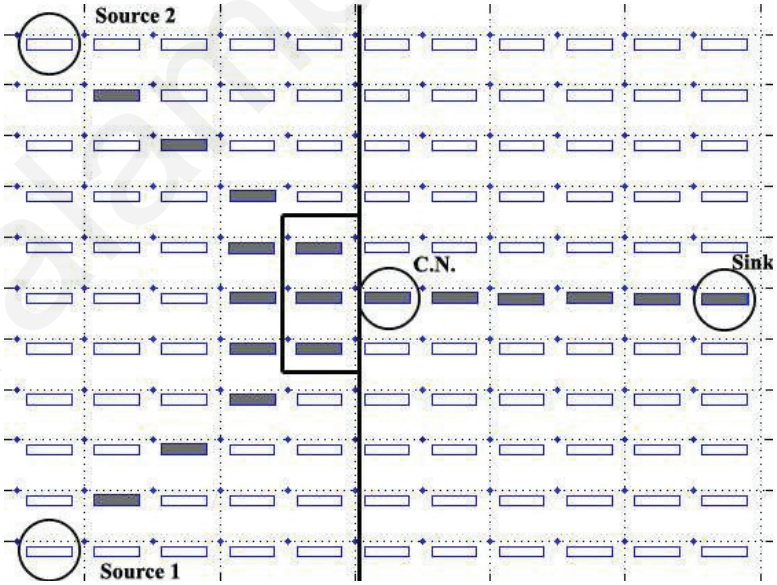

Figure 22: Network Topology in Prowler Simulator.

3.4.2 Scenarios and Results

A total of 50 runs have been conducted for each simulation point and the results presented below are the average of all simulation runs.

The first parameter we examine is the time until the buffer of the "congested node" fills up. For this specific simulation series we set the data rate at 60 packets/s, while according to Fig. 22, the "congested node" is receiving packets from three nodes.

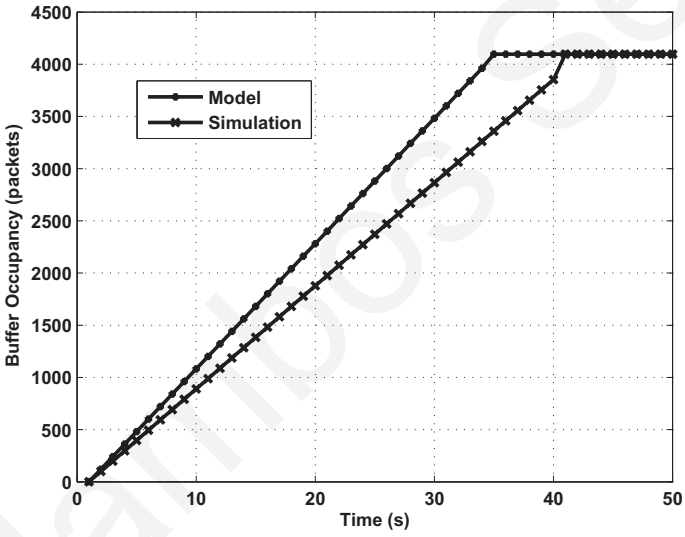

Figure 23: Analytical and Simulation Results until "Congested Node" Buffer Fills Up.

Analytical results, according to (19), indicate than the buffer of the "congested node" is going to fill up in 34-35 seconds, while the speed s for this process is calculated to 120 packets/s. Simulated results show that the buffer fills in 40 seconds with an average of 102 packets/s. Results show, that the analytical model is able to capture well the attitude of the network. The difference between analytical and simulated model, can be attributed to the MAC layer, which is not employed in the analytical model. MAC layer retransmissions are able to delay the transmission of packets to the next hop nodes.

The next parameter we study is throughput (Fig. 24). Throughput describes the number of packets per second which are received by the sink. In this simulation series we attempt to study the results provided by (14). Analytical results prove that if a node is receiving data from three nodes concurrently, then the total flow (data rate) of these three nodes, must be reduced to $\bar{\rho} = 0.09$, which according to Fig. 19 corresponds to flow equal to 0.08. Bearing in mind that the maximum flow (180 packets/s) is achieved when flow corresponds to the value of 0.25 (see Fig. 19) this means that the total incoming flow must be reduced to $(0.08/0.25) \times 180$ which equals to an average of 57.2 packets/s.

To simulate this phenomenon we implement a typical congestion control mechanism capable to suppress data rate (traffic control method), in order to avoid buffer packet drops. In this specific scenario, since we want to discover the time of the start of queue formation, we program this algorithm to suppress immediately the data rate, as soon as the first packet is enqueued, and to use an additive increase multiplicative decrease (AIMD) method to find the exact point where the maximum throughput is achieved with no packet loss.

In this graph, as we stated, the maximum throughput is achieved when the incoming data rate is at an average of 57.2 packets/sec. Simulation results indicate that the maximum throughput is achieved when data rate is at an average of 52 packets/sec. The results prove that the theoretical model is able to capture the attitude of the network. The small variation in the value can be justified due to the effect of the MAC and physical layer.

3.5 Concluding Remarks

Through this analysis we prove that in order to avoid any packet drops in WSNs, each data flow should avoid bottlenecks or other obstacles (e.g. routing holes) that obstructs free data flow

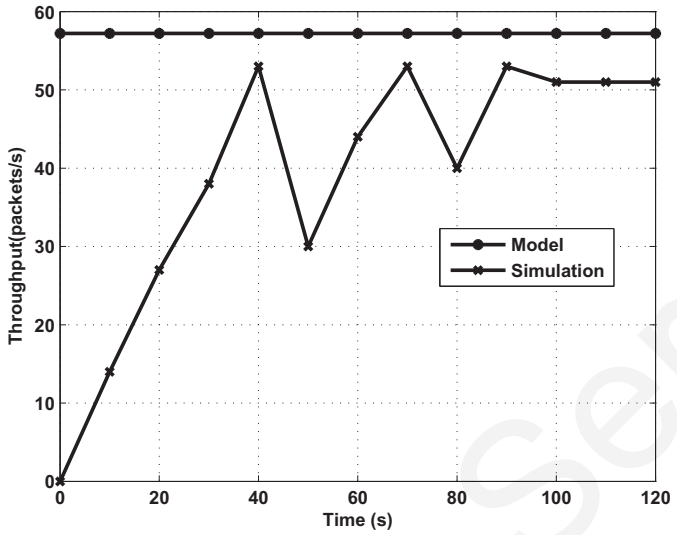

Figure 24: Analytical and Simulated Throughput.

in the network. Since this condition is infeasible in real networks, congestion control mechanisms should be added.

It is clear that bottlenecks create significant problems to the performance of WSNs. We prove that, in order to avoid queue formation the data rate of the incoming flow must be severely reduced. In case that this action is omitted, buffer fill-up is going to happen, while the time that passes until the nodes' buffers fill-up depends on the difference between incoming and outgoing flows. This issue becomes more worrisome for the performance of the network, if we count that just one bottleneck in the network is capable to reduce the throughput of all the flows passing though it, as it happens in the example we present in this work. Source data reduction, or traffic control, is the type of congestion control method that is being used by the majority of congestion control algorithms in WSNs [22,34–36,39,57]. Although this method has been proven effective for transient congestion situations and it is considered as a generally lightweight method, the disadvantage we presented

above can be proven fatal, especially for critical applications or applications that demand all sensed data to be received by the sink.

Thus, we believe that in such cases congestion control algorithms that employ a resource control method are more effective than algorithms that employ a traffic control method. For instance, in the example we present in this Chapter, if a resource control algorithm applies when the queue formation begins, then each one of the three nodes that merge on the "congested" node will search for an alternative path and the throughput will not be reduced. Resource control is the method that we adopt in this thesis.

Chapter 4

Tree-Forming Schemes for Topology Control in Wireless Sensor Networks

4.1 Introduction

As we discussed in the previous Chapters, massive and random placement of sensor nodes on a monitored field renders communication between nodes, a difficult task to be achieved. Interference, congestion, and routing problems are possible to arise at any point in such networks. Routing challenges in WSNs stem from the unique characteristics of these networks, such as limited energy supply, limited computing power, and limited bandwidth on the wireless links, which impose severe restrictions on the design of efficient routing protocols. According to [77], a number of routing challenges and design issues like, among others, node placement and energy consumption, can affect routing process in WSNs. Thus, topology control, in conjunction with routing challenges, becomes an important issue that has to be carefully considered in order to achieve proper network operation.

Permitting all nodes to transmit concurrently without any control will result in high interference, high energy consumption, and reduced network lifetime. Topology control algorithms focus

in lowering the initial network topology, by reducing active nodes and links, thus saving resources and increasing network lifetime. Employing tree forming schemes as topology control algorithms in WSNs has been widely adopted, since it is an efficient and robust solution. The most common type of tree topologies, is the shared, core-based tree, which has the sink node as the root. Networks that employ this specific type of topology control scheme, create a spanning tree that initiates from the sink and as a result the initial placement and redundant connections between nodes are severely reduced. Shared, core-based trees, also known as sink-based trees in WSNs, are proven to be an efficient and effective topology control solution. Sink-based trees is the term that we will use for the rest of this thesis.

In this Chapter, we study whether specific tree-forming schemes can improve the performance of congestion control algorithms when the network faces an overload condition. In particular, we focus on source-based trees. Source-based trees are created by each node that becomes source node (its analog part senses an event). This type of routing trees has not been widely adopted in WSNs since it incurs excessive overhead to the network. In particular, we study whether and how source-based trees can improve, under specific circumstances, the performance of congestion control algorithms.

4.2 Analysis

A sink-based tree is an efficient topology control solution in WSNs. Usually, such trees are being build as spanning trees using the sink as the root and all nodes forward their data using this structure. Using the sink node as the root is optimum, since the sink in WSNs is, most of the times, a robust node that does not suffer from power limitation issues. Thus, this situation defeats the main disadvantage of sink-based tree, which is the fact that core nodes can become single points of failure. The creation of a sink-based tree is normally a simple procedure that begins after the

network discovery phase. Nodes become part of this tree at the initialization phase of the network and maintain their position in it until topology changes happen.

In this work we consider that the following assumptions exist:

- Nodes communicate only with nodes that are one hop away.
- Dense placement of nodes achieves the coverage of the network.
- As we describe in the next subsection, nodes route packets only to the nodes that are one level higher than themselves.

Many wireless sensor network applications rely on the availability of a collection service to route data packets towards a sink node. A typical collection protocol provides for the construction and maintenance of one or more routing trees, having each a sink node as their root. A sink can store the received packets or forward them to an external network, typically through a reliable and possibly wired communication link. Within the network, nodes forward packets through the routing tree up to (at least) one of the sinks. To this end, each node selects one of its neighboring nodes as its parent. Nodes acting as parents are responsible of handling the packets they receive from their children and forwarding them towards the sink. To construct and maintain a routing tree, the collection protocol must initially define a specific metric to be used by each node, in order to select its parent. The distance in hops to the sink or the quality of the local communication link (or a function thereof) can for instance be used as metrics for parent selection. In either cases, nodes need to collect information about their neighboring nodes in order to compute the parent selection metric. To this end, nodes regularly exchange corresponding messages, usually called beacons, that contain information about, e.g., the (estimated) distance in hops of the node to the sink or its residual energy.

4.2.1 Sink-Based Tree Creation

The first phase of network establishment is network discovery. In this phase, the major task of every node is to discover its neighbor nodes in the network. A common method for network discovery is some sort of flooding algorithm, initiated and controlled by the sink. After the end of this phase each node maintains a routing table in which the IDs of the neighbors nodes are kept, as well as any other information requested by the algorithm. An example of the end of initial phase is presented in Fig. 25 in which the nodes are aware of all of their one-hop neighbors.

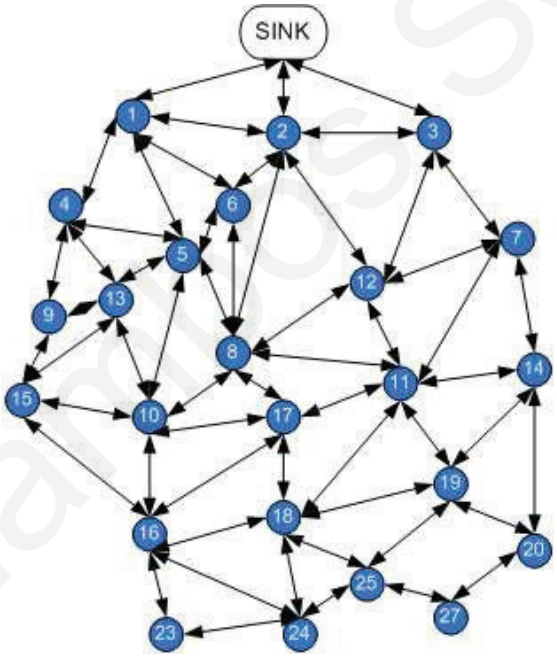

Figure 25: Initial Network Connectivity (Before Topology Control Algorithm Applies).

It is clear that the initial topology shown in Fig. 25 must be reduced, otherwise interference and energy issues are possible to arise. For example, if an event appears and is captured by nodes 10, 11, and 20 and all three nodes start transmitting packets to all the nodes they are connected with, then it is certain that after some time, interference and buffer based congestion will arise. Congestion will lead to quick power exhaustion and the network will not be able to accomplish

its task. Thus, the network initial topology must be reduced in order to take into account the redundancy in the connectivity.

Phase 2 of the tree creation begins from the sink and the target is to reduce the initial topology and place nodes in levels from the sink to the edge nodes. The process initiates with the sink broadcasting a "Hello" message with the $level$ field set to 0. Nodes that receive this message update their $level$ field to 1 and re-broadcast this message. Each node that receives a "Hello" message, updates its $level$ field and broadcasts this message accordingly. If a node receives more than one "Hello" messages, it updates its $level$ field based on the lower level. Thus, if its level changes to a lower value, it informs the nodes around it and the process iterates until all nodes are set in levels. Hence, a spanning tree is created.

To explain this concept better let us consider again Fig. 25. The sink broadcasts a "Hello" message with $level$ set to 0. Nodes $1, 2$ and 3 receive this message, update their $level$ field to 1, and re-broadcast this message. Nodes that receive the message from level 1 nodes, also update their level and the procedure iterates until all nodes connect at the lowest level possible. Fig. 26 presents the results after this procedure is applied to the network.

As shown in Fig. 26, creating a level-based, shared tree, leads to a "relaxed" topology (with a lower degree of connectivity). Nodes communicate with a lesser number of nodes, a fact that renders the network more capable to avoid and control congestion situations (either in the medium or in the buffers) while the source-to-sink packet time is reduced to the minimum, since a spanning tree is created.

4.2.2 Source-Based Tree Creation

Source-based trees introduce a completely different routing concept in comparison with sink-based tree. The biggest difference lies on the fact that source-based trees, abandon the shared,

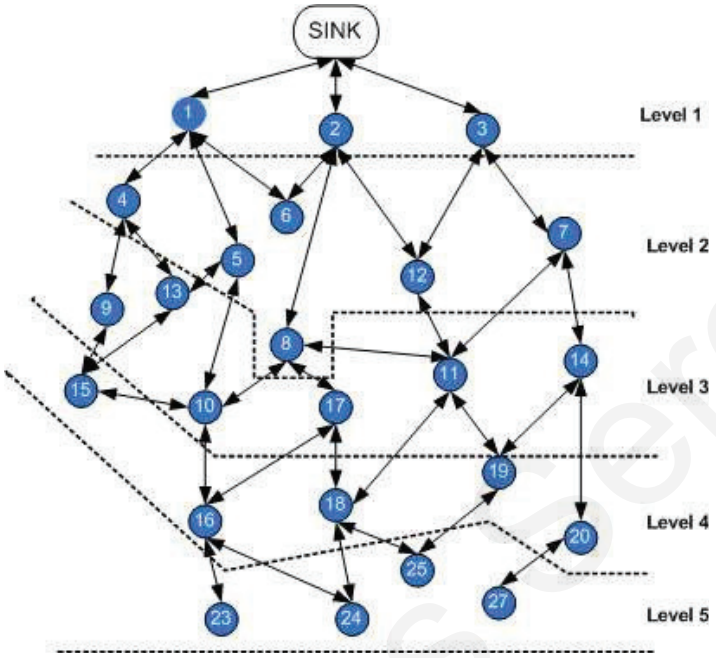

Figure 26: Phase 2 of Network Connectivity (After Topology Control Algorithm Applies).

core-based, philosophy and move to distributed solutions. Source-based trees are constructed on demand (when a node becomes a source node) and continue their operation until nodes stop being sources. Hence, when a node senses that is becoming a source node (receives analog data) it starts building its routing tree from itself (source) to the sink.

Building a proper and well tuned routing tree from a source to the sink is not a straightforward procedure as with sink-based trees. The reason lies on the fact that source-based trees do not "reduce" the topology as sink-based trees do. Contrary, they incur significant overhead since every node is possible to create its own tree, due to the fact that every node can become a source.

A "naive" approach, is to construct a source-based tree just based on level information (similar to the sink-based solution). For example, if we use the topology of Fig. 25 this will result in the tree of Fig. 27. This tree is considered "naive", since it presents several drawbacks that impose severe restrictions to the proper operation of the network.

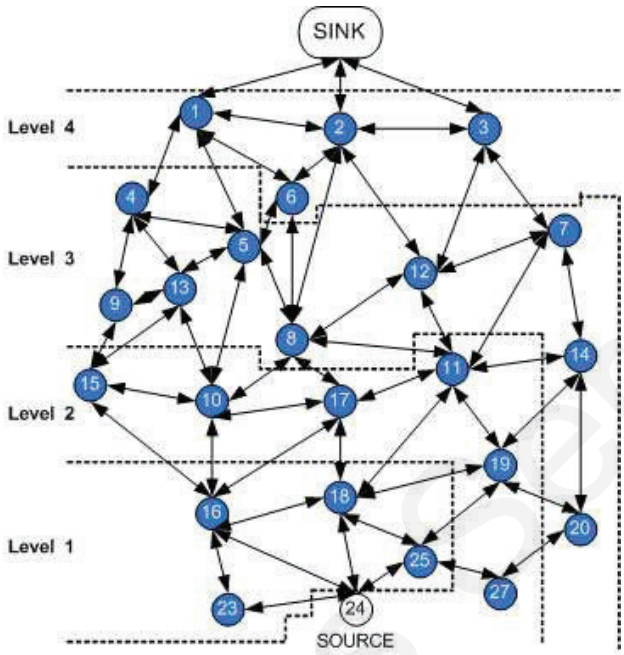

Figure 27: A “Naive” Source Based Tree.

One of the drawbacks is that it does not create unique trees using the shortest path to the sink. We also notice that it is possible for a packet to be routed to a node that does not have any higher level node (closer to the sink). For example, if a packet is forwarded to node 14 then this node is not possible to know to which node to forward this packet to. Node 14 is connected to node 7 and 20, which are at the same level and node 11 and 19 which are at a lower level. Thus, if packet is forwarded to node 20 (which is at the same level as node 14) then node 20 will come to the same situation as node 14. Routing loops is possible to be created between these nodes.

Thus, source-based trees presents several disadvantages that have to be addressed before they get used in topology control and routing.

4.3 Source-Based Trees in Congestion Control Algorithms

A sensible question that arises, is why to employ source-based trees for any reason, since sink-based trees provide an efficient topology control solution, while source-based trees present several disadvantages. The reason lies on the fact that source-based trees can provide numerous alternative paths from source to sink in comparison with sink-based trees. Since in this thesis, we choose to employ the resource control method for the development of congestion control algorithms, we study how this type of topology control trees is possible to affect the performance of algorithms that employ this method and study whether it provides any additional benefits to their performance.

To construct source-based trees which can fulfill their purpose we consider a small set of "critical" parameters:

Location Awareness: Nodes must be aware of their position in relation to the sink. Localization and positioning in WSNs is a subject that attracted a lot of research work [78] and solutions already exist in the literature. Hence, if a node is aware of its position then it selects as neighbors only nodes that are closer to sink than itself. For example in Fig. 27, node 27 will not be selected by node 25, due to its position which is further away from the sink, even though it is a level 2 node.

Higher Level Connection Availability: A node is kept on the tree only if it is connected to at least one node at a higher level than itself. For example in Fig. 27, node 14 will remain out of the tree since it is not connected to a node at a higher level. Concerning this parameter, an enhancement could be the connection with nodes at the same level, but closer to the sink. In this Chapter we examine just the influence of critical parameters, and we avoid to involve any enhancements.

Number of nodes kept in neighbor table: In dense placements this number could be very long and will create significant overhead problems to the network. Thus, a maximum number of neighbors must be set. According to [79] a value of six neighbors seems to be optimal. Also a minimum number of two must exist. If a node is connected to less than two higher level nodes, it must add in its neighbor table nodes which are at the same level, but nearest to the sink. If such nodes do not exist, then this node should not be considered as a good neighbor for data forwarding.

Employing these parameters in Fig. 27, the initial source-based tree transforms to the one in Fig. 28.

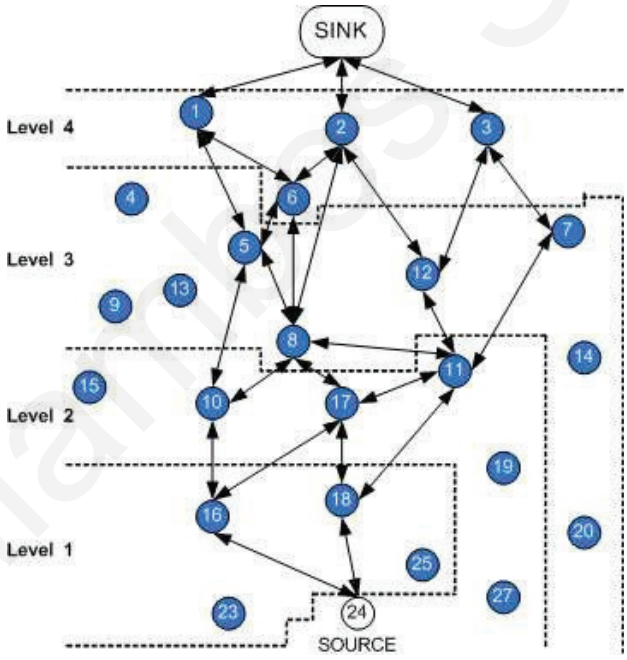

Figure 28: A Properly Constructed Source Based Tree.

The resulting topology is significantly reduced in comparison with Fig. 27 while all nodes are connected to upstream nodes and each node provides at least two alternative paths. Nodes that do not comply with the parameters discussed above, are not part of this routing tree. Of course,

due to the dynamic nature of WSNs, it is possible that a node which is not at the routing table the present moment, to become part of it, in case of topology changes.

Also, although Fig. 28 seems even simpler than sink-based tree topology (Fig. 26), sink-based tree topology is a global, shared topology while source-based tree (Fig. 28) concerns just one source node, out of possible, all nodes in the network. This means that if many sources exist in the network, there could also exist many different trees that share the same nodes.

4.4 Added Signaling Overhead Requirements

Signaling overhead could become an issue for source-based trees if not carefully tuned. For example, if we consider the initial topology (Fig. 25) we calculate that 114 messages need to be exchanged between the nodes for the creation of a sink-based tree. This will create the network of Fig. 26 where the active connections between nodes will be reduced to 36 in comparison with 56 in the initial topology (Fig. 25). On the other hand, if we consider the source-based tree (Fig. 28) we notice that from the farthest possible source (node 24), 51 messages need to exchanged in order to end-up with a topology that provides 26 active connections.

Of course this result is just for one source out of possibly every node in the network. For this reason source-based trees, even though they are being built on demand (when a node becomes a source node), should be able to maintain their connectivity and re-use it in order to avoid added signaling overhead. For this reason, it is clear that the "critical" parameters that we discussed in the previous section should be maintained, otherwise the creation of a robust source-based tree is not feasible. Especially the third parameter, "Number of nodes kept in neighbor table", is very important since a small but sufficient number of nodes can significantly reduce signaling overhead. If we consider again Fig. 28 we notice that if node 18 becomes a source it may use the tree that it has already been built by node 24, thus avoiding any extra overhead.

4.4.1 Complexity Analysis

At this point, we perform complexity analysis of the two topology control schemes. We consider a random topology that consists of n nodes and we assume that through a plain flooding algorithm, each node has discovered its position in the network.

Source based Tree: When a node becomes a source node, it will try to establish its routing tree by means of broadcasting a $hello$ message. Each node that receives this message creates its Neighbor Table, N and retransmits the $hello$ message. Since each node retransmits this message just one time, this will result to an exchange of messages equal to n [1] .

Furthermore, the algorithm requires bi-directional links, which are practically achieved using a 2-way handshake. This creates the additional message cost of $2 \times N.length$. The worst case in this topology is the case where all nodes become source nodes. This will result to an exchange of messages equal to n^2 since all nodes will initiate the same process to create a source-based tree, resulting in the total of messages:

$$n^2 + 2 \times N.length \tag{23}$$

Thus, the message complexity of source-based trees is equal to $O(n^2)$. Due to this result, algorithms that employ resource control should employ specific techniques in order to reduce this possible high number of message exchange. Two techniques that can be used here are: (a) to reduce the actual number of nodes in the neighbor table, and (b) to enable source node to re-use already known source-based trees.

Sink-based Tree: Unlike source-based trees, sink-based trees create a single shared tree. Thus, after the discovery phase, the sink broadcasts an initial *hello* message which also contains the level

[1] The actual messages could be slightly higher than n in case that a node receives a message from a node with a lower level in comparison with the previous one. But since this number of messages cannot change the final message complexity of the algorithm, we omit it.

of the node in respect to the sink. Each node receiving the initial *hello* message adds one to the level field of the message and retransmits it. The receiving node marks as a parent the sending node and updates its Neighboring Table. This induces a retransmission of the initial $hello$ message of n-times. Thus, the network creation is of message complexity n, since nodes exchange at maximum n messages between them [2] .

$$O(n) \tag{24}$$

4.5 Topology Alteration and Trees Re-Construction

Topology can easily change in WSNs due to hardware failures and power exhaustion of the wireless sensor nodes. Thus, in many cases, there is demand for changes in the current topology since it is no longer optimal. This process is called topology maintenance. Topology maintenance can be triggered for several reasons according to [80]:

- Time-based: In this case topology maintenance is triggered when a timer expires.
- Energy-based: It is triggered when the energy level of a number of nodes goes below a predefined threshold.
- Random: It is triggered randomly based on a variable like time.
- Failure-Based: It is triggered when a number of nodes has failed. It requires the existence of failure detection and notification mechanism.
- Density-Based: It is based on the density of the network and specifically the degree of the nodes (above a specific number).

[2] The same comment as with source-based tree stands also for sink-based tree. The actual messages could be slightly higher than n in case that a node receives a message from a node with a lower level in comparison with the previous one. But since this number of messages cannot change the final message complexity of the algorithm, we omit it.

- Combination: A combination of the reasons above.

Concerning the tree-forming schemes that we discuss in this section, topology maintenance decisions should be based on a combination of reasons. Specifically, since these algorithms are going to be used with congestion control algorithms, the topology maintenance decisions must be based on energy and failure reasons.

Specifically, concerning sink-based trees, topology maintenance can be required when energy depletion and failure of nodes reaches a level where nodes are not able to find any available nodes at a higher level in order to forward their packets. Topology maintenance can be used to provide the necessary alternative paths for this purpose. In this work, since we consider heavy loaded environments, we employ topology maintenance at a local level. In particular, when the nodes discover each other after their initial deployment, each node maintains information in its neighbor table about all nodes that can physically communicate with.

In the case of sink-based trees, when a node cannot forward packets to its usual next-hop neighbor it forwards its packets through other appropriate neighbor nodes it discovered during the initial deployment. The same happens when a node gets power exhausted or fails. Thereby the problem is solved locally.

Source-based trees behave differently. In source-based trees, trees are being built on demand and only when a node becomes source node. As we stated in the previous section, in source-based trees the number of neighbor nodes must be reduced in order to avoid severe interference issues. However, when the number of these nodes becomes smaller than two, topology maintenance can then be triggered in order to search for alternative paths to these nodes. The selection of the moment to trigger topology maintenance is critical. Triggering this process very early and for a limited number of sources could cost more energy than the energy that is supposed to be saved by topology maintenance. Triggering the process too late, could cost severe energy consumption

and throughput reduction due to packet drops. This tradeoff must be be determined taking into account the specific application requirements.

The next figures present an example of the employment of topology maintenance in source and sink-based trees under resource and traffic control cases. To extract these results we employ the simulation parameters and network model of section 4.6. Moreover we consider that the network performs topology maintenance each time its overall energy becomes equal to the 50% of the total energy that the network had, before the topology maintenance. Thus, the first time that it performs network maintenance is when the total energy of the network falls to 50% of the initial, the second time is equal to the half of the 50% minus the required energy for the network maintenance, while the third is equal to the half of the second time minus the required energy for the third network maintenance. The third network maintenance is the last that it can be performed, since there is not much remaining energy for a fourth one.

Studying figures 29 and 30 we notice that in both control methods, when source-based trees are employed, network maintenance happens much earlier, compared with sink-based trees. Especially in the traffic control method, we notice that the difference is higher, since in this case sink-based trees present less energy consumption.

Fig. 31 presents the percentage of network's remaining energy when the network is unable to forward a single packet from source to sink, when no network maintenance is applied. Studying this figure we notice that in the traffic control method, sink-based trees stall, when the remaining energy of the network is near 20%.

If we compare this result with the fact that when network maintenance is applied in the traffic control scenario, a sink-based tree is able to perform network maintenance three times. This means that in this case the network maintenance extents the lifetime of the network. Concerning source-based trees the conclusions are different. In this case, since each source builds its own tree,

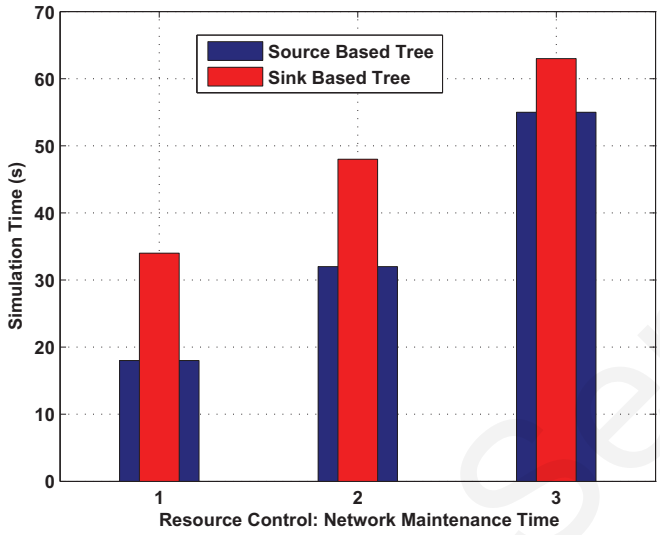

Figure 29: Topology Maintenance - Resource Control Method.

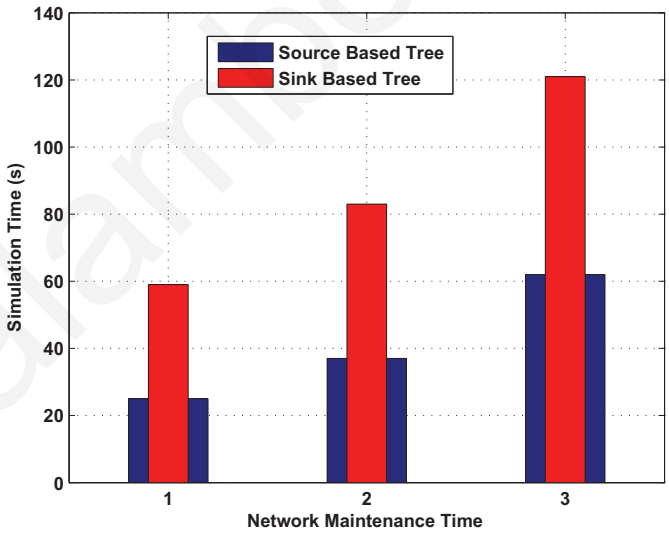

Figure 30: Topology Maintenance - Traffic Control Method.

performing network maintenance several times is not so efficient. Building a tree from each source is an energy consuming process and its benefits should be counted as a part of the application of the network.

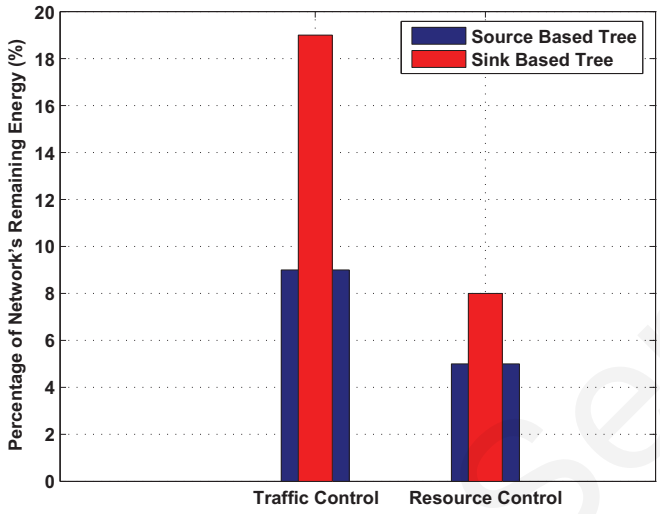

Figure 31: Network Lifetime without Maintenance.

4.6 Evaluation

To compare the performance of source-based against sink-based trees concerning resource and traffic control algorithms for congestion control in WSNs, we performed a series of simulations using Prowler [72] simulator under the same simulator parameters as of section 3.4 of Chapter 3.

4.6.1 Network Model

We assume that there is a WSN, where nodes are initially deployed uniformly. One sink is located at a specific point in the network. Each node is considered that is aware of its position in relation to the sink. In our network deployment we consider that all nodes (except the sink) are identical and "carrier sense multiple access with collision avoidance" (CSMA/CA) is employed as MAC protocol.

Moreover, in order to compare the performance of source and sink-based trees when the network faces congestion, we employed as a resource control reaction mechanism, the alternative

path creation scheme of Hierarchical Tree Alternative Path (HTAP) algorithm. This scheme is analytically described in next Chapter.

4.6.2 Scenarios and Results

For each of the performance metrics presented below, the results are the average of 20 runs for 10s for each measurement point (except for the cases that it is otherwise stated). Nodes are uniformly placed on a square grid. All nodes communicate only with the nodes that are one hop away. We employed 12 sources and every source is programmed to transmit at maximum 128 packets/s, while we increase the number of nodes on the grid.

The first metric we examine is sink throughput. Sink throughput indicates the ability of any congestion control algorithm, to control congestion and transmit a maximum number of packets to sink.

Fig. 32 presents the results when the resource control method applies.

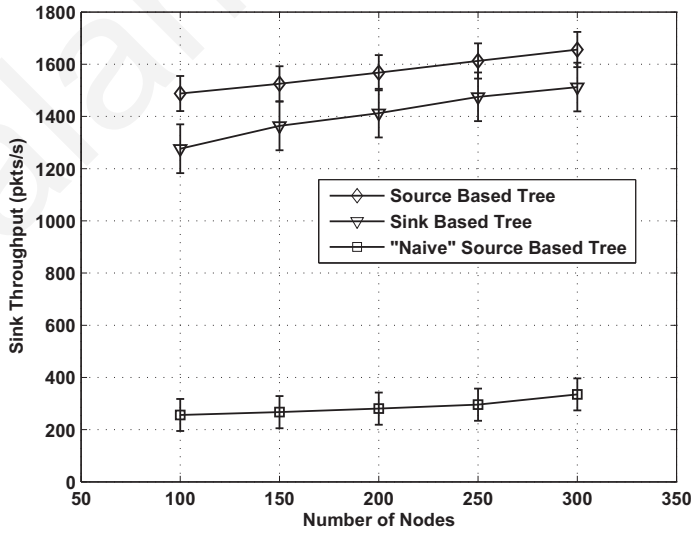

Figure 32: Resource Control Method: Sink Throughput.

Results show that, when resource control method is applied along with a source-based tree, sink throughput increases. This result indicates the ability of source-based trees to provide more alternative paths, in comparison to sink-based trees. Thus, algorithms that employ resource control are favored by the employment of source-based trees and as a result they enhance their performance in terms of sink throughput. Sink-based trees cannot provide as many alternative paths as source-based trees and normally throughput is reduced. A "naive" tree is the worst option, since routing circles and uncontrolled number of alternative paths enhance congestion problem.

Also, we notice that as the number of nodes in the network increases, throughput also increases since more alternative paths can be provided.

When the traffic control method is employed, the situation is different (Fig. 33).

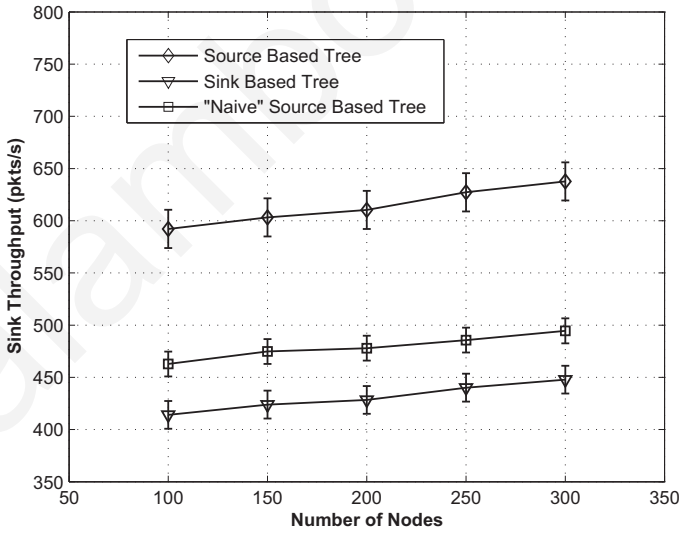

Figure 33: Traffic Control Method: Sink Throughput.

In this case, sink throughput is significantly reduced in comparison with the case that the resource control method applies. When the traffic control method applies, the network attempts to adjust traffic to network resources. Thus, since algorithms that employ traffic control method,

do not alter their routing path and always use the shortest path, sink throughput is affected by the number of flows that cross each other. It is notable that in this scenario, sink-based tree provides the worst results in comparison with the other two trees. This is as expected, since when a sink-based tree is employed, data flows from each source share the same topology. On the other hand, when source and "naive" trees are employed each source node creates its own routing tree from the beginning. Hence, more nodes are employed and less number of data flows cross between them. "Naive" tree gives acceptable results concerning sink throughput, since after its creation each source finds the shortest path to the sink. The results are worse than source-based trees, since a naive tree creates longer paths to sink and flows cross in more nodes in comparison with source-based trees.

The next parameter we study is the average delay of successfully received packets from the sources to the sink.

Fig. 34 presents the results for the resource control method.

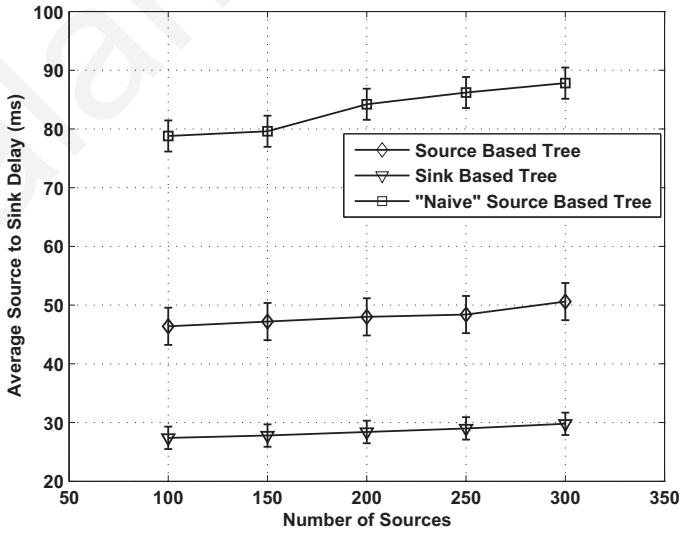

Figure 34: Resource Control Method: Average Delay from Sources to Sink.

Sink-based tree, provides less delay in comparison with the other trees. This is as expected, since sink-based tree is a spanning tree and the position of each node from the sink is the minimum. Between the other two trees, the average delay when source-based tree is employed is less than the "naive tree". Source-based tree provides more routing paths in case of congestion while "naive" tree provides longer routes and in some cases routing circles that increase the delay.

Fig. 35 presents the results when traffic control is employed.

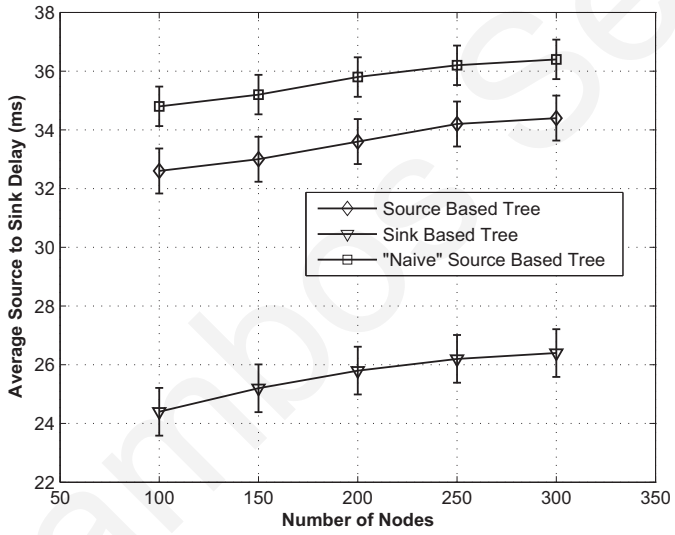

Figure 35: Traffic Control Method: Average Delay from Sources to Sink.

In this case the overall delay is less, in all three topologies, in comparison with the case that the resource control method applies. This result is as expected, since algorithms that employ traffic control always use the shortest path from source to sink and do not re-route any packets. Between the three topologies, sink-based tree presents much better results in comparison with the other two trees. The reason lies in the fact that sink-based tree creates a spanning tree using a shared topology. On the other hand, source-based trees present an increased initial overhead in order to

create each tree from each source, while the naive tree provides longer routes which results in increased delay.

Finally, we study the average energy consumption in units. Each unit is equal to 1000mW in 1 second.

Fig. 36 presents the results when the resource control method applies. As it is expected "naive" tree presents the worst results while sink-based tree, the best. This is normal, since sink-based tree

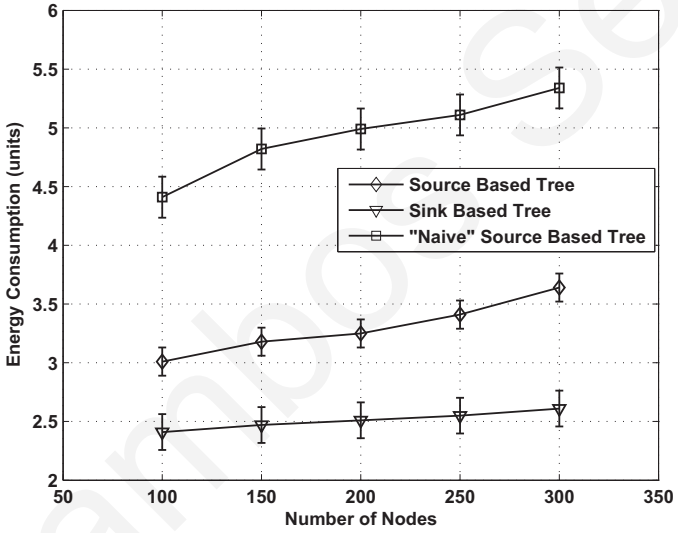

Figure 36: Resource Control Method: Average Energy Consumption.

creates a shared topology and nodes follow shortest paths in order to reach sink. On the other hand source-based trees create a big number of paths and in some cases nodes follow longer routes in order to reach sink.

Concerning the traffic control algorithm, the situation is as follows. In this case, also sink-based trees provide the best results, but the reason is different. According to Fig.33, when traffic control is used with sink-based trees the result is that fewer packets are delivered to the sink, since

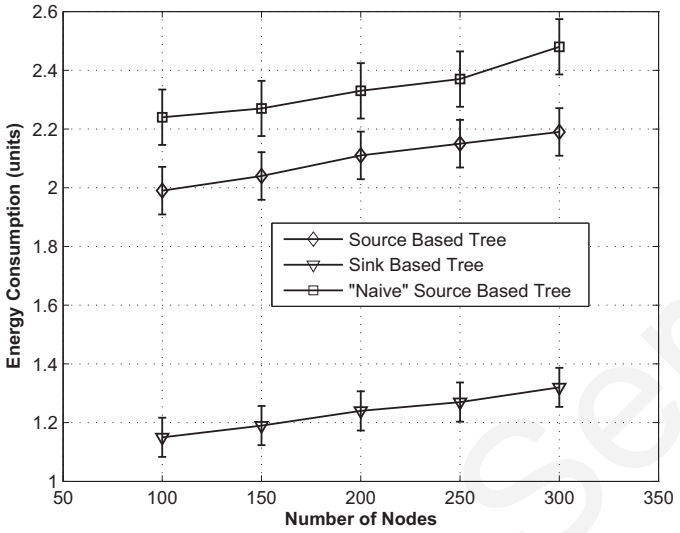

Figure 37: Traffic Control Method: Average Energy Consumption.

the sources' data rates are reduced. Thus, less packets lead to fewer transmission and less energy consumption. Moreover, the average delay is also smaller which translates into fewer hop counts.

4.7 Concluding Remarks

In this Chapter we study specific tree forming schemes for topology control in wireless sensor networks. In particular, we consider whether source-based trees can provide an efficient topology control solution similar to sink-based trees.

Concerning sink-based trees, results show that can be combined effectively with resource control algorithms as with traffic control algorithms. Sink-based trees provide very good results especially in terms of delay and energy consumption with both types of congestion control algorithms.

Concerning source-based trees, results show that can also provide a proper topology control solution in WSNs when overload condition exists. Source-based trees can provide better results than sink-based trees, in terms of sink throughput when either traffic or resource congestion control

algorithms apply. The drawback they present, is the longer delay and the higher energy consumption, in comparison with sink-based trees. Thus, we consider that source-based trees can provide a significant topology control solution when the application demands the vast majority of packets to reach the sink. In this case, source-based trees can be combined with resource, congestion control algorithms. In any case, source-based trees must be carefully tuned since the creation of a "naive" tree, can become fatal for network operation.

Chapter 5

Hierarchical Tree Alternative Path Algorithm

Following the results and conclusions of Chapters 3 and 4, we present in detail a novel algorithm, called Hierarchical Tree Alternative Path (HTAP). HTAP is an algorithm designed to face congestion in WSNs by employing alternative paths for the avoidance of congested areas (or nodes) in the network. The bigger advantage of HTAP algorithm is the fact that it is a simple algorithm which actually does not add any significant overhead to the already heavily loaded network, in case of congestion. HTAP employs the resource control method for congestion control, while it utilizes source-based trees. Furthermore, the HTAP algorithm introduces a novel adaptive method for inferring congestion in the network. This adaptive method uses buffer occupancy as a first indication of congestion occurrence and then it employs the ratio of out/in data rate in order to trigger the alternative path creation.

To evaluate its performance, HTAP algorithm has been extensively tested and compared to an established algorithm that also employs a resource control method for the transmission of packets from source to sink, called Topology-Aware Resource Adaptation (TARA) [30]. Simulation results show that the HTAP algorithm exhibits a better performance attitude, in the range of 2-4%, in

all examined parameters in comparison with TARA, while it is evidently much easier to be implemented and introduces significantly less communication overhead than TARA. For completeness, we also consider the case where no congestion control is employed. We believe that this work presents a significant advancement in the area of congestion control in WSNs since it provides a simple, robust, effective, and efficient solution to this problem.

5.1 Network Model and Problem Description

We assume that there is a densely deployed WSN, where nodes are initially deployed randomly, but uniformly in space. One sink is located at a specific point in the network topology, while the number of source nodes is variable, since an event is possible to be random and be captured by more than one nodes. We also consider that each node knows its position and the position of the sink node(s) in the grid. Localization and positioning in WSNs is a subject that attracted a lot of research work [78] and it is beyond the scope of this work. In our network deployment we consider that all nodes beside sink are identical and CSMA/CA is employed as the MAC protocol.

The problem that we attempt to counter in this work is to enable the network to deliver to the sink(s) all (or almost all) of the data packets which have been created, without intervening on the rate with which sensor nodes are injecting these packets in the network. This means that this network should be able to detect possible congestion nodes (hot spots) and to avoid them. To achieve this target we need to employ nodes which are not in the initial route from the source to the sink.

5.2 Hierarchical Tree Alternative Path (HTAP) Algorithm

HTAP is a dynamic congestion control algorithm that bases its path switching decision on local information, such as the congestion state of its neighbors.

HTAP consists of four different schemes:

- Topology control.
- Hierarchical Tree Creation.
- Alterative Path Creation.
- Handling of Powerless and/or Failed Nodes.

We now describe each scheme in detail while in Fig. 38 we present the operation of HTAP algorithm through a flowchart.

5.2.1 Topology Control

Topology control is crucial in WSNs since it can handle issues arising from the redundant number of nodes and their dense deployment. Problems like interference, maximum number of possible routes, use of maximum power to communicate to distant nodes directly, etc., are possible to arise. Since HTAP is an algorithm that attempts to employ the network's extra resources (unused nodes), it is obvious that guaranteeing a redundant number of paths is essential. In order to maintain the performance characteristics of the network in case of congestion, these paths must be carefully selected. Topology control is, therefore, the first scheme applied in the HTAP algorithm.

An effective topology control algorithm should be able to preserve connectivity with the use of minimal power, while maintaining an optimum number of nodes as neighbors to each node. In this work we employ, with a variation, the Local Minimum Spanning Tree algorithm (LMST) [79] as the initial topology control that runs on the network. LMST is an algorithm capable of preserving the network connectivity using minimal power, while the degree of any node in the resulting topology is restricted to six nodes. As it is analytically explained in [79], this feature (6

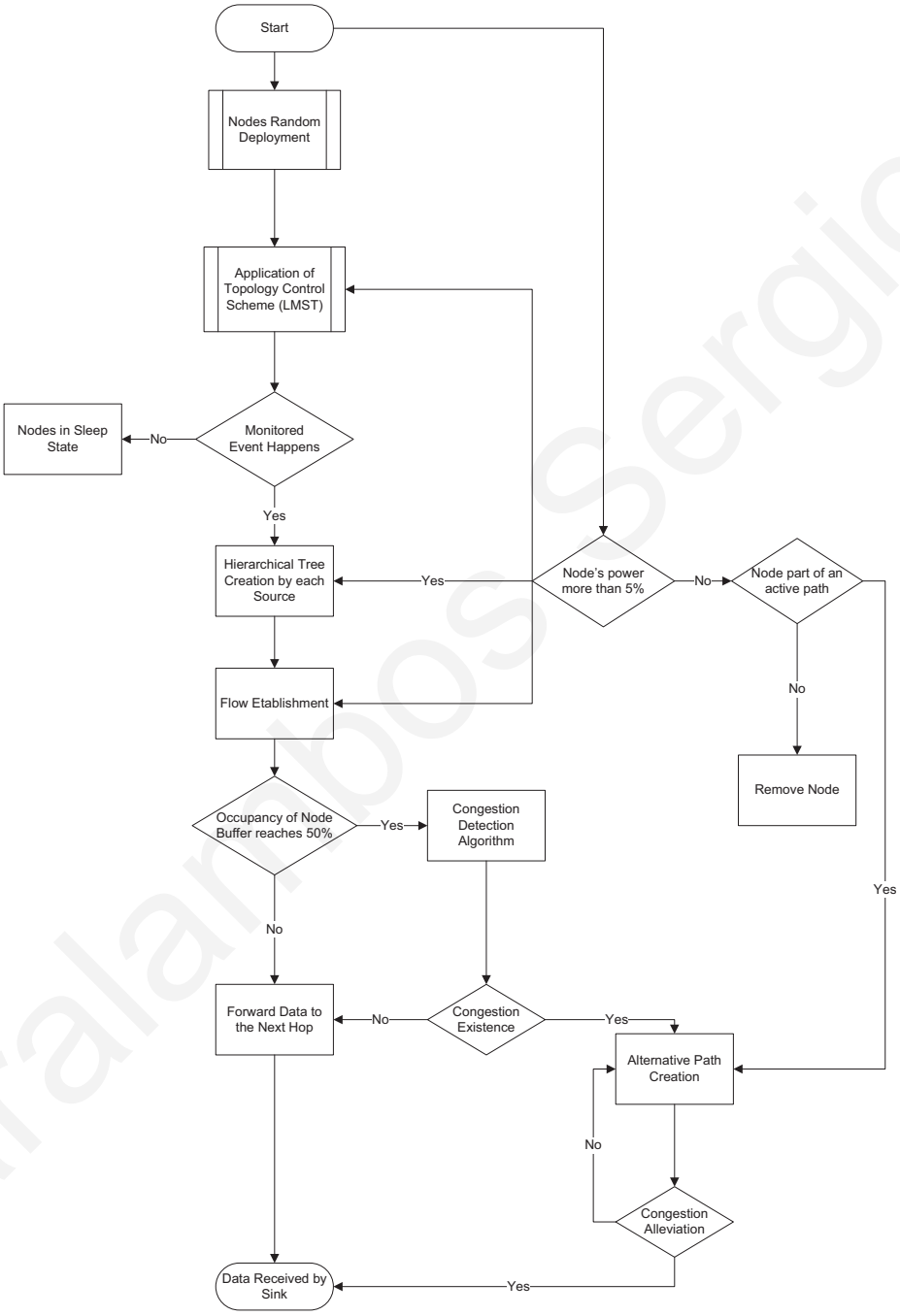

Figure 38: Flowchart for HTAP Algorithm.

neighbors per node) is desirable because a small node degree reduces the MAC-level contention and interference. In LMST each node builds its local minimum spanning tree independently, using Prim's algorithm [81] and keeps on the tree only those neighboring nodes which are one hop away. In addition, the resulting topology is possible to use only bi-directional links, a matter which, as we will explain later, is valuable for the successful operation of HTAP.

Details of the LMST algorithm: At this point we provide a brief description of the operation of the LMST algorithm.

In LMST, authors consider the network topology constructed under the common maximum transmission range d_{max} as an undirected simple graph $G = (V, E)$ in the plane, where V is the set of nodes and G is the edge set of G. Also $NV_u(G)$ is defined as the "visible neighborhood" set, which is the set of nodes, that node u can reach by using its maximum transmission power.

LMST consists of the following phases:

- Information collection.
- Topology construction.
- Determination of transmission power.
- An optional optimization phase which is the construction of topology with only bidirectional. edges

We detail them below:

Information collection: This is the initial phase where each node attempts to discover the other nodes in the network. Thus, each node u needs to contact and exchange information will all nodes in its visible neighborhood $NV_u(G)$. To do this, each node broadcasts periodically a"Hello" message using its maximum transmission power. In this "Hello" message each node piggybacks its ID and its location information.

Topology Construction: During this phase, each node, after obtaining the information concerning its visible neighborhood $NV_u(G)$ (i.e. the nodes it can communicate with), applies Prim's algorithm [81] independently in order to obtain its local minimum spanning tree. Specifically, in order to build a power efficient minimum spanning tree, nodes use their Euclidean distance, since power consumption is, in general, a strictly increasing function of the Euclidean distance. Also the tree must be unique. Since Prim's algorithm does not guarantee unique trees, LMST uses a custom weight function between the edges of node u in order to construct a power efficient tree which is unique. Then, node u, using this unique, power-efficient tree determines six one-hop neighbors.

Determination of transmission power: After the construction of the minimum spanning tree, each node measures the receiving power of the "Hello" messages. By performing this process, each node can determine the specific power levels it needs to reach each of its neighbors. This attribute is used in case that a node needs to broadcast a message (e.g., that it is congested). Then, it adjusts its transmission power to a power level that can reach the farthest neighbor.

Construction of topology with only bi-directional edges: This is an optimization already provisioned in LMST. If this optimization applies, then the network consists only bi-directional links. This is achieved either by enforcing all the uni-directional links to become bi-directional or to delete all the uni-directional links. In this work, due to the dense placement and the fact that a plethora of bi-directional links exists, we choose to delete all the uni-directional links.

Technical details, specifying exactly how these phases are implemented can be found in Section III-B of [79].

As we stated above, the variation introduced to LMST by HTAP, relates to the selection of the one hop neighbor list. In this neighbor list, each node keeps records of the IDs of the one-hop nodes that is able to transmit data, as well as their congestion state. Thus, instead of keeping

records of all nodes that fulfil the criteria as one-hop neighbor, the modified LMST used by HTAP, keeps records only about the neighbor nodes that reside closer to the sink than itself, since these are the nodes that will provide the alternative paths, if congestion occurs. In order to discover which nodes are closer to the sink than itself, each node compares its location information with the location of its neighbors and the location of the sink. The location of its neighbors, as we stated above, is communicated to the nodes during the "Information Collection" phase, through the initial "Hello" message while the position of the sink, is assumed to be known to all nodes.

Fig. 39(a), presents the initial network connectivity, while Fig. 39(b) presents the network after LMST topology control.

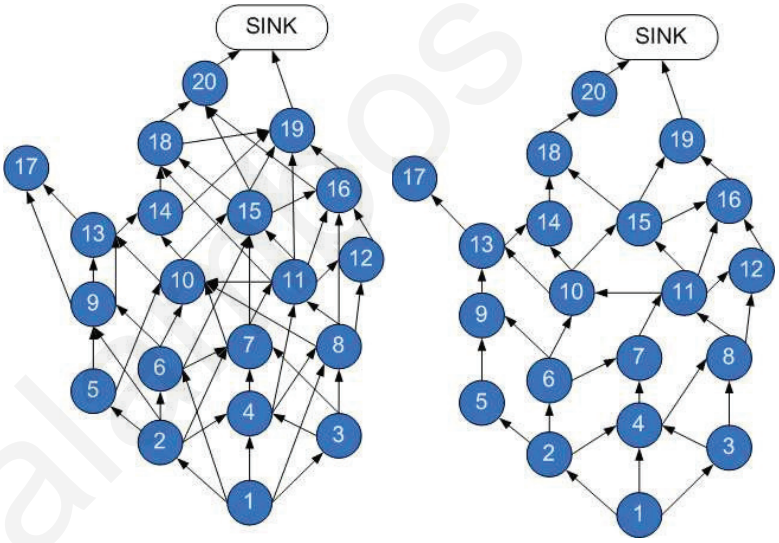

Figure 39: (a)Initial Network Connectivity (b) Network Connectivity after Topology Control.

5.2.2 Hierarchical Tree Creation

The hierarchical tree creation algorithm runs over the topology control algorithm and only at after a node becomes a source (starts sensing a phenomenon). This algorithm consists of two main steps, described below:

1. **Path Creation:**In this step a hierarchical tree is created beginning at the source node. After the end of the topology control phase, each node is able to be connected to at most six nodes which are only one hop away from itself. During this phase each node that is becoming a source node is self-assigned as level 0 and sends a $level\_discovery$ message to the neighbors selected during topology control phase. Nodes that receive this packet are considered as children to the source node and are set as level 1. Each of these nodes broadcast again the $level\_discovery$ packet, and the pattern continues with the level 2 nodes. This procedure iterates until all nodes are assigned a level and stops when the $level\_discovery$ packets reach the sink.

   When the procedure finishes it is possible that the sink receives more that one $level\_discovery$ packets from different nodes and each packet may have a different level value. This is an indication that disjoint paths are reaching the sink.

   The hierarchical tree algorithm is also able to identify and rectify some issues that are possible to arise. Specifically, it is possible for a node to be the last one that receives the $level\_discovery$ packet when there are no other nodes upstream able to forward that packet. This node responds by broadcasting a "negative ACK" packet (NACK) indicating that it cannot route any packets. When the upstream nodes receive the NACK they become aware of the situation.

   An example of the operation of hierarchical tree algorithm, and placement of nodes in levels, is illustrated in Fig. 40(a) and 40(b).

   After the LMST topology control algorithm runs in the network the level placement procedure takes as input the resulting topology and attempts to place nodes in levels from each source to sink. Continuing on the same network example as before, node 1 is considered as

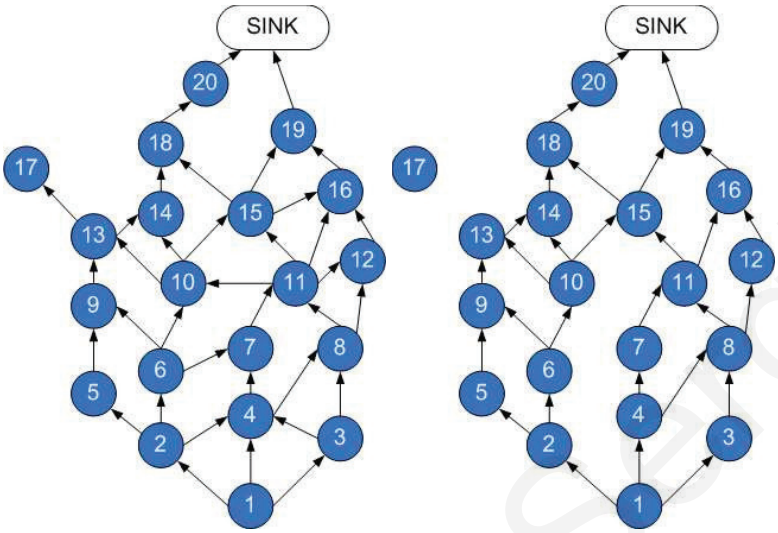

Figure 40: (a) Network Connectivity after Topology Control (b) Level Placement Procedure.

the source and it broadcasts a message to the nodes that it can connect to. In this case, nodes 2, 3, and 4 receive the $level\_discovery$ message and are assigned as level 1 nodes for node 1. Then these nodes (2, 3, and 4), transmit this message to the nodes they are connected to, and these nodes are assigned as level 2 nodes (5, 6, 7 and 8). This procedure iterates until the packets reach the sink.

During this procedure, if a node receives a packet from more than one nodes it retains the information of only the node that places it at a lower level (i.e. closer to the source). For example, node 7 is connected with nodes 6 and 4. During the level placement procedure it will receive a packet from node 6 placing it at level 3, and a packet from node 4 making it a level 2 node. Node 7 will choose node 4 as its upstream parent, become a level 2 node, and will ask from its neighbor nodes to become level 3 nodes.

Finally, in Fig. 40(b), node 17 represents the case where a node does not have any upstream neighbor to route packets through. In this situation the level placement algorithm removes this node from the table of upstream nodes of node 13.

2. **Flow Establishment:**A connection is established between each transmitter and receiver pair using a two-way handshake. Through this packet exchange, the congestion state of each receiver is communicated to the transmitter. Let us consider again Fig. 40(b), where node 1 is the source and nodes 2, 3, and 4 are receivers. Firstly, node 1 sends a packet to node 2. When node 2 receives this packet, it sends an ack packet back to 1. In this ack packet node 2 piggybacks its current congestion state. This exchange makes the source node aware of the congestion state of all the neighbors it can overhear. When the congestion state of a child (downstream neighbor) reaches a pre-specified limit, the node updates its upstream nodes of the congestion state using a dedicated control packet.

5.2.3 Alternative Path Creation

This algorithm runs when congestion is about to occur at a specific node in the network. The ability of the employed topology control algorithm (LMST) to preserve network connectivity with the use of minimal possible power, as well as the fact that the node degree in the topology derived under the algorithm is small, assists in limiting collisions in the medium. Also, it helps to mitigate the well known hidden and exposed terminal problems as there will not be so many nodes that have to be silenced to ensure error-free communication. However, congestion is still possible to happen when a node receives packets with a higher rate than it can transmit (buffer-based congestion). In a wireless sensor network where all nodes, except the sink, are exactly the same, this can happen if a node is receiving packets from at least two flows, or if the nodes to which it has to transmit packets to cannot accept any more packets.

When the buffer of a node starts filling, this node has to take action. Each node is programmed to run locally a lightweight congestion detection (CD) algorithm. When the buffer reaches a buffer-level threshold value, the CD algorithm starts counting the rate with which packets are

reaching the node. Since each packet is identified in its packets header by the NodeID, the CD algorithm is aware of all the nodes (k) that transmit packets through this node, as well as their data rate.

$$\sum_{i=1}^{k} Rx_i \geq Tx_{max}. \tag{25}$$

By using (25), the CD algorithm is able to calculate the total receiving rate ($\sum_{i=1}^{k} Rx_i$) and compare it with its maximum transmission rate (Tx_{max}). When this ratio is large (and above a certain percentage) the node sends a backpressure message to the nodes that transmit packets through it to search for an alternative path. The selection of these nodes is performed firstly from those that transmit with the lower rate. The purpose of this tactic is to maintain the performance characteristics of the network (keep the throughput of nodes at the maximum possible level without packet drops) and minimize the impact of the change to the network.

When an upstream node is informed to stop transmitting packets through a specific downstream node, it searches in its neighbor table and finds the next downstream available node with the same level (in comparison with the congested node) and starts transmitting packets through it. Concurrently, all upstream nodes that reside in a level lower than the congested node update their neighbor tables with the fact that this downstream node is congested and avoid transmitting any data through this node. Similarly, when this downstream node becomes available again, it informs the upstream nodes accordingly.

An advantage of HTAP in comparison with similar schemes like [30] and [45] is the fact that it does not employ specific nodes as distributors and mergers. Through the advantage that is offered from the use of the topology control algorithm and the source based hierarchical tree, each node is able to inhibit the transmission of packets through itself and also it is able to join the first available shortest path, after path alternation.

Congestion Threshold: As mentioned above, a key point to the operation of the HTAP algorithm is the value of the congestion threshold. Nodes, using (25) calculate the total receiving rate and compare it with the maximum transmission rate they have at the moment. The issue that arises is that both parameters (receiving and sending rate) are heavily dependent on the current network situation. Thus, it is possible for a node to receive a large number of packets in a short period of time and then to keep receiving packets at a lower (normal) rate. A node is then congested (in terms of occupied buffer space) but actually it is not experiencing any problems that force it to control this overload situation. Therefore, the parameter that needs to be tuned is not just the value of the threshold, but also the duration of the excess transmission (burst period). If the duration is set too low, then the "alternative path creation" algorithm will be triggered often. In cases when the situation is transient, the creation of alternative paths would add unnecessary overhead to the network (delays and power consumption). On the other hand, if the burst period is very big, buffer overflows will occur and nodes will react overdue to this situation.

HTAP handles this issue by using an adaptive method. Initially, buffer monitoring begins when the buffer occupancy of each node reaches 50% of the total. At this instance the affected node counts the number of nodes from which it is receiving packets. Then it assumes that each node is transmitting with the maximum data rate and calculates the time until the buffer occupancy will reach the 85% limit. When this time elapses it checks again the occupancy of the buffer. If it is between 80 and 85% it considers that, indeed, it is receiving packets with a higher rate than it can transmit and it triggers the "alternative path" algorithm to avoid congestion. If the buffer occupancy is less than 80% it re-calculates the remaining buffer and adjusts accordingly the time, which obviously is greatly reduced. If at the next measuring epoch the buffer occupancy is still below 80%, the first threshold is adjusted from 50% to 70%. Such behavior can happen in areas

where a permanent event is taking place. When a permanent event is affecting the network, is expected that by setting the buffer to just 50% will contribute to the overhead.

Let us proceed with an example. Consider a network that consists of nodes with a buffer size (B) of 128KBytes (or 1024 packets if we consider that a packet equals to 1024 bits) and a maximum data rate (r) of 128Kbps (or 128 packets per second). Also consider that a node is receiving data from five different nodes (n). When the 50% of the buffer (64KBytes or 512 packets) is full the node begins the process.

Initially, it counts the number of nodes from which it is receiving data (in this case n=5) and considers that they transmit with the maximum data rate (r=128Kbps). Afterwards, it calculates the time t which is $t = B/(n \times r) = 64KBytes/(5 \times 128kbps) = 0.8s$. This means that in 0.8s the CD algorithm will check again the buffer occupancy. If the buffer occupancy is between 80 and 85% (820-870 packets) it will trigger the "alternative path" algorithm. If it is less, e.g. 65% (665 packets), then it will measure again the number of nodes that transmit packets and will adjust the time accordingly. Thus, if we consider that the node is now receiving packets from four nodes instead of five, it will calculate the remaining buffer up to 85 % (870 packets) which is now 20% (205 packets) as well as the maximum data rate of 4 nodes. This will provide a new timeout value and the algorithm will check buffer occupancy after this time elapses. If the buffer remains below 70% (717 packets) but higher than 50% (512 packets) it will set the first threshold to 70% (717 packets) and will stop monitoring this node until buffer occupancy exceeds this threshold (70% or 717 packets).

By employing this extended but lightweight congestion detection scheme the HTAP algorithm is able to face both permanent and transient congestion situations successfully.

5.2.4 Handling of Powerless or Failed Nodes

Special care is taken in the HTAP algorithm concerning the nodes which are power exhausted or failed. These nodes cause major problems to the network in case they act as sources or relay nodes. Thus, when a node is about to get power exhausted or it is failed, it should immediately be extracted from the network and the tables of its neighbor nodes should be updated. This procedure should be as simple as possible due to the fact that this can happen when the network is in a crisis state.

Concerning the powerless nodes, when the power of a node reaches the "power extinction" limit, the node immediately broadcasts this fact to the nodes around it. Receiving nodes remove the related NodeID from their neighbor list. If the power-exhausted node is a part of an active path (a path that is relaying packets to the sink), the upstream nodes will apply the "alternative path" algorithm and find another path to forward packets to the sink.

On the other hand, each node that does not acknowledge a packet two consecutive times, is considered as failed and the node that initially recognizes this fact, informs all other nodes in its neighbor table, in order to also remove this node from their neighbor tables.

5.3 Performance Evaluation

To evaluate the performance of HTAP we choose to use a simulation tool. Although real experiments provide more realistic and accurate results in comparison with simulations, the fact that we need to evaluate congestion control algorithms that employ resource control renders the implementation and operation of a real testbed impractical. Resource control algorithms demand a significant number of nodes and variable scenarios in order to comprehensively study their behavior. Thus we evaluate the performance of HTAP and the other algorithms through extensive

simulations and try to implement the simulation conditions as closer to the real conditions as possible (Section 5.3.1).

Through simulations, we evaluate the performance of HTAP in comparison with two other algorithms. The first one is "no congestion control" (No CC) algorithm which represents the case where no congestion control is employed in the network while the second is TARA [30]. Also we present results in comparison with the initial version of HTAP the HTAP v1 [8] and a version called "HTAP No HT". HTAP v1 is a version of HTAP where the topology control scheme is not implemented, while "HTAP No HT" is a version of HTAP where the Hierarchical Tree is not implemented. In "HTAP No HT", routing is initially performed based on the shortest path from each source to sink and when congestion occurs alternative path creation is based only on the congestion level of the neighbor nodes.

5.3.1 Simulation Environment and Setup

To perform our simulations we have used the Prowler simulator [72], and the simulation settings and parameters of section 3.4 of Chapter 3.

5.3.2 Performance Evaluation of HTAP

In the following scenarios 100 nodes were uniformly (randomly) deployed on a 500m x 500m grid. The sink was set in the upper right corner of the grid. Events were generated at the bottom left corner of the grid and all nodes that sensed an event were becoming sources. Thus, the number of sources in our simulations is dynamic. Each simulation run has been performed 20 times and average results have been extracted. In the results we present the standard deviation limits of each point. Also, as we stated above, in this series of experiments we also present the performance of

the initial version HTAP presented in [8] for comparison purposes. We denote this initial version of HTAP as "HTAP v1".

Percentage of Successfully Received Packets: The first performance metric we have examined is the percentage of successfully received packets according to (26). This metric is particularly important for critical/emergency applications, where every packet has to be received by the sink, since it illustrates the ability of algorithms to control congestion and provide reliable communication.

$$ReceivedPktsRatio(\%) = \frac{SuccessfullyReceivedPkts}{TotalPktsSent} \tag{26}$$

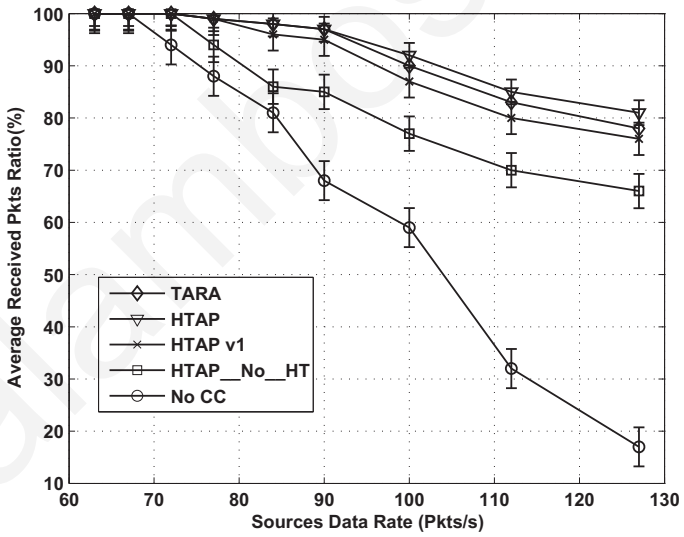

Figure 41: Percentage of Successfully Received Packets.

Analyzing the results of Fig. 41 we notice that if no congestion control algorithm is applied, the number of successfully received packets decreases dramatically as the load increases in the network. This fact is expected since the network becomes overloaded and packets are dropped either due to collisions or due to full buffers. Concerning resource control algorithms we notice

that HTAP and TARA start reducing their performance after sources data rate exceed 90 packet/s. Probably the reason they start presenting this performance is due to the fact that all resources in the network are fully utilized. This leads to improper operation of resource control algorithms due to lack of resources. In this specific simulation series we note the performance of HTAP algorithm is at an average of 2.5% better than TARA. Concerning the two versions of HTAP, without topology control (HTAP v1) and without Hierarchical Tree ("HTAP No HT") we notice that their performance is degraded in comparison with HTAP. The non implementation of topology control leads to the connection of each node to uncontrolled number of nodes, fact that prolongers the routes and incurs to increased interference between nodes. On the other hand, when no Hierarchical Tree is implemented, the results are even worse since alternative path creation is based only on the congestion level, fact that can easily lead to routing circles.

Network Throughput: To further study this attitude we count the actual number of packets that are received by the sink (throughput). This metric is a strong indication of the ability of the algorithms to transmit a satisfactory number of packets to the sink. Results are presented in Fig. 42.

This figure indicates that although the percentage of received packets is decreasing for HTAP and TARA, the actual number of packets that is being received by the sink is increasing. This fact is a strong indication that resource control algorithms can provide the sink with a large number of data packets, even when the load in the network is very high. The same happens with HTAP v1 and "HTAP No HT".

Increasing Resources: To check how the number of resources affect the performance of resource control algorithms we keep the node data rate at 128 packets/second and in each run we increase the number of nodes. Results are presented in Fig. 43 and Fig. 44.

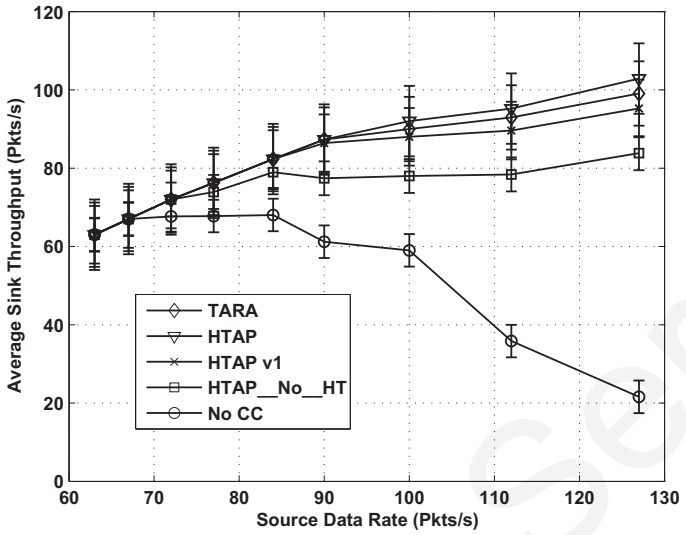

Figure 42: Throughput.

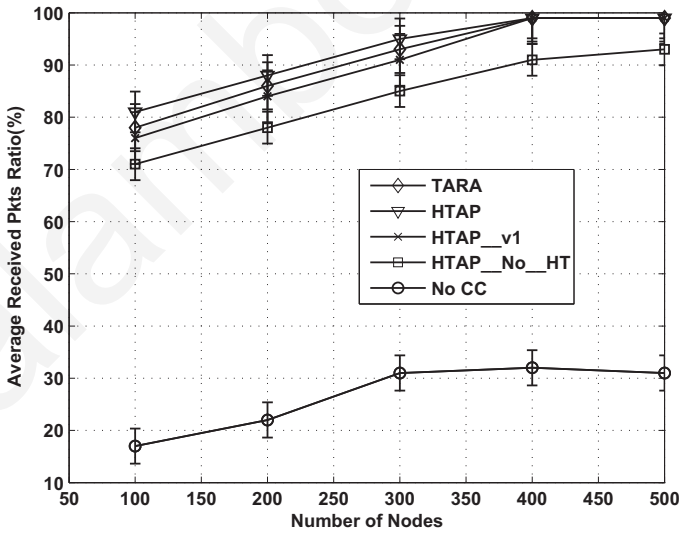

Figure 43: Average Received Packets Ratio (%) with Increasing Number of Nodes.

Fig. 43 and Fig. 44 indicate that as the number of nodes in the network increases, resource control algorithms improve their performance and are able to deliver a larger amount of packets to the sink. We also notice that as resources are increasing, HTAP and TARA present similar

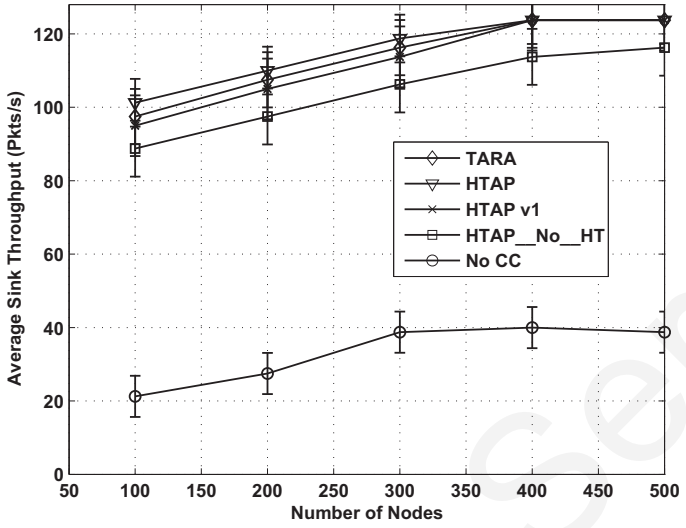

Figure 44: Average Throughput with Increasing Number of Nodes.

performance, which is an indication that their resource control mechanisms can operate properly. The "no congestion control" case is also slightly improved but again its performance is very low. Concerning HTAP v1, we notice that the absence of topology control algorithm is not so important when there are too many resources in the network in comparison with the load. This means that Hierarchical Tree can operate properly and can deliver to the sink all data, since the placement of nodes in levels operates properly. On the other hand, if no Hierarchical Tree is implemented the situation is different. In this case, routing of packets just based on the congestion level can easily lead to routing circles since in such case the majority of nodes do not receive any packets at all. The reason that this version of HTAP is able to deliver packets to the sink lies on ability of topology control algorithm to keep in its neighbor table only nodes that reside upstream to sink. Otherwise the situation would become much worse.

Average Hop-by-Hop Delay: The next metric that we evaluate is the average delay. This metric is an indication of how algorithms handle inter-path interferences, link layer retransmissions, and the overhead introduced by control packet exchanges.

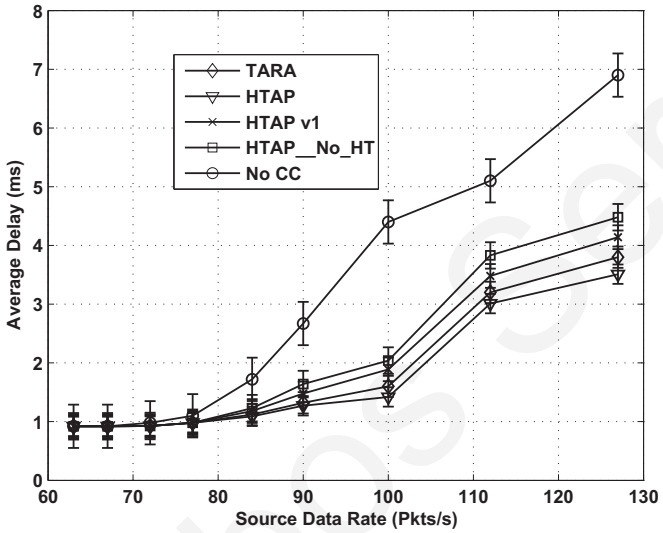

Figure 45: Average Hop-by-Hop Delay.

Results in Fig. 45 depict that, HTAP and TARA present increased delay after the source data rate reaches 100 packet/s, since the load in the network is increasing and an attempt is made so that all packets reach the sink. Packets drops that happen at this phase (also according to Fig. 41) justify this attitude. Between HTAP and TARA we notice that HTAP presents less delay in the range of 2-3%. This happens because in HTAP, nodes exchange less control messages in comparison with TARA and the algorithm introduces less overhead. The performance of HTAP is also much better than HTAP v1 and "HTAP No HT". This means that the topology control algorithm in combination with the Hierarchical Tree, are critical for the successful operation of HTAP.

Energy Consumption: Then we study total energy consumption of nodes (Fig. 46). This parameter is an indication of the energy efficiency of the algorithms. The energy consumption is the sum of used energy of all the nodes in the network, where the used energy of a node is the sum of the energy used for communication, including transmitting (Pt), receiving (Pr), idling (Pi) and sleep state (Ps). Thus, the total energy that a node consumes is

$$E_{node} = T_t.P_t.L + T_r.P_r + T_iP_i + T_s.P_s \tag{27}$$

where T_t, T_r, T_i, T_s is the total time that a node transmits, receives or it is in idle or sleep state and L is the transmission power.

In a Mica Z node, the energy used for transmission, reception and idling is more or less the same (Tx: Rx: Idle= 17.4 mW: 19.4 mW: 17 mW) while the energy used in sleep state is negligible in comparison with the other states. Thus we consider it as zero.

We measure the power consumed in **units** where each **unit** equals to 1000mW in 1s (W x s).

In this metric we actually sum the consumed power of all nodes after the end of simulation according to the following equation:

$$E_{total} = \sum_{i=1}^{N}(E_{node}) \tag{28}$$

where N is the number of nodes.

When there is no congestion control the consumed energy increases quickly and then stabilizes, since the network has reached its maximum capacity. On the other hand, TARA and HTAP present increased energy consumption since many more nodes are now employed for the transmission of packets from the source to the sink. As we can see from Fig. 42 this energy consumption is rather negligible in comparison with the achieved throughput. Again, between the two resource

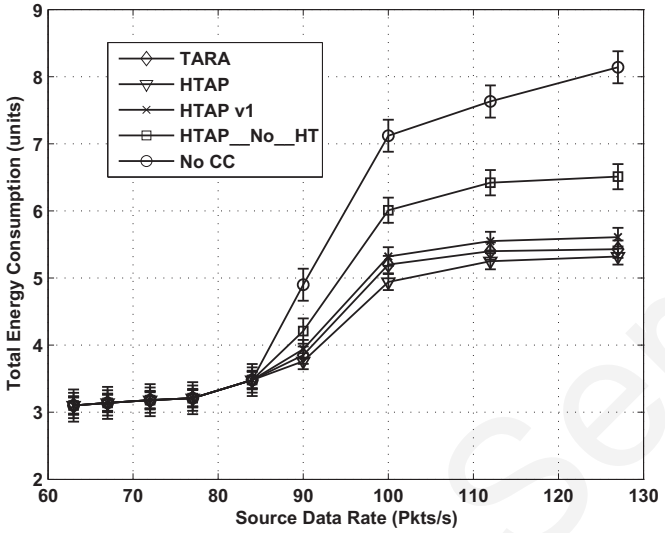

Figure 46: Total Energy Consumption.

control algorithms HTAP is 2-3% better than TARA and better than HTAP v1 and "HTAP No HT".

Network Remaining Energy: Finally, we study the percentage of the network's remaining energy (network lifetime) after the moment that the network is not able to transmit a single packet from the source to the sink, due to a lack of available paths. This metric is an indication of the effectiveness of algorithms to utilize uniformly the resources of the network. In this simulation series each algorithm run over the same topology for different numbers of nodes (100, 200, 300). In each set, the simulations were running until the network became disconnected.

It is clear, from Fig. 47, that resource control algorithms (HTAP and TARA) can uniformly utilize network resources in comparison with the case that no congestion control algorithm exists.

If we zoom at the two resource control algorithms (Fig. 48) we recognize that their performance increases as the number of nodes is increasing. This fact is expected since more nodes provide more resources. Moreover, we notice that the performance of HTAP is better than TARA.

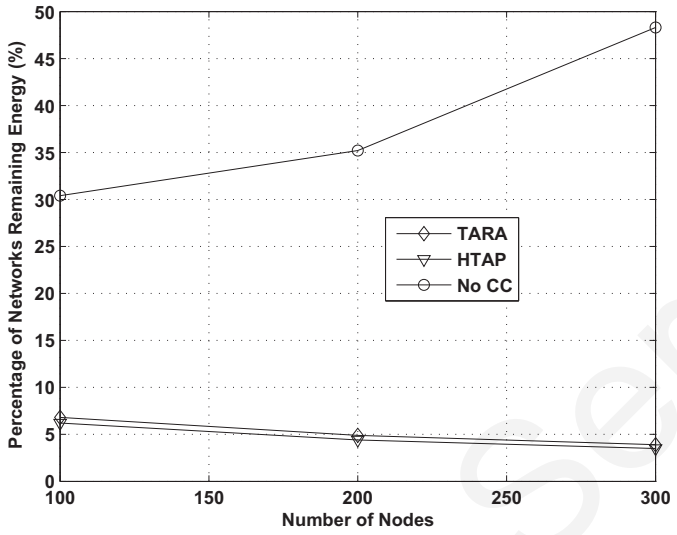

Figure 47: Network's Remaining Energy (%).

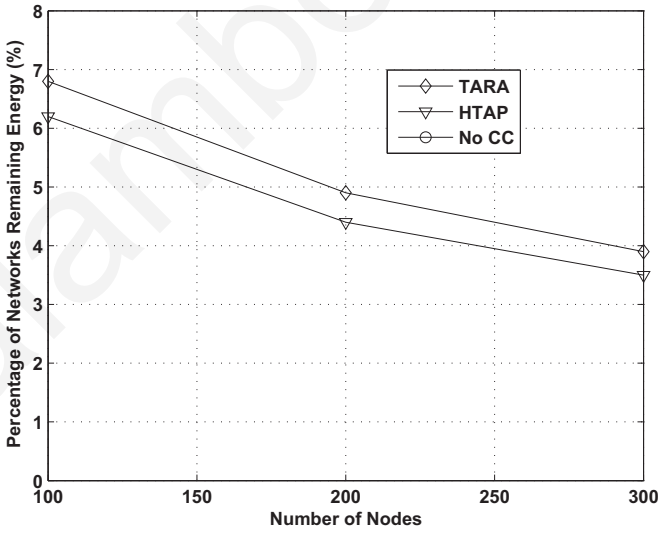

Figure 48: Network's Remaining Energy (%) (Zoom on the Lower Part of Figure 47.)

This is an indication that HTAP can utilize the network resources more efficiently and can provide increased lifetime.

5.3.3 Evaluation of Congestion Threshold Adaptive Method

As we stated in section 5.2.3 the adaptive method that is employed in HTAP algorithm is important for the improved performance of the algorithm. In order to validate this statement we have designed and implement specific scenarios. Specifically, we kept all the simulation parameters the same as in the previous section and we varied the congestion threshold. We have set four values for this congestion threshold. The first value is static but low (30%), the second is static but high (85%), while the third is static again and set to 50%. Finally the last one is the adaptive method we have used.

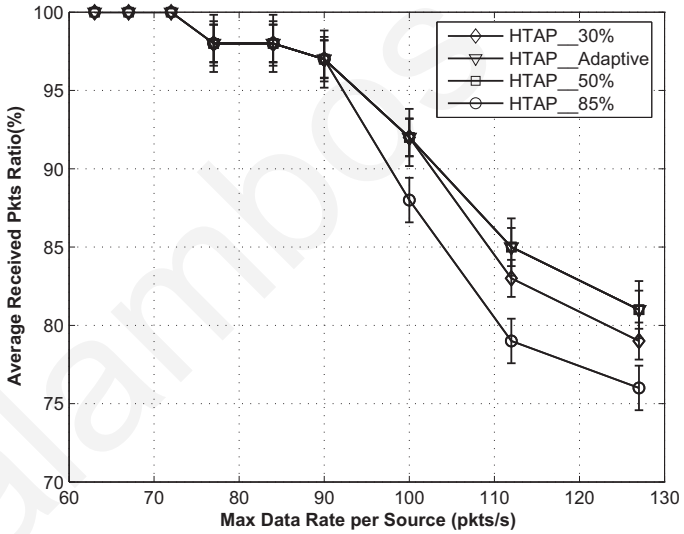

Figure 49: Percentage of Successfully Received Packets.

The first parameter we have examined is the percentage of successfully received packets. As we notice in Fig. 49 when the threshold is set to a low percentage of buffer occupancy (30% and 50%) the network presents almost the same performance as when adaptive method is being used. This means that when the nodes trigger the alternative path mechanism very early, the network avoids congestion and the only reason that packets are lost is because the capacity of the network

is exceeded, as it happens when the source data rate is high. A small variation exists when the threshold is set to a very low value (30%), when the max data rate per source is over 100 packets per second. This can be explained by the fact that when the data rate is very high, a big number of nodes in the network have their buffer 30% filled and some packets are lost in the procedure of finding alternative paths.

On the other hand when the congestion threshold is set to a static high value (85%) the network starts dropping packets when the data rate exceeds 90 packets/s per source. This is an indication that when the alternative path creation is triggered, congested nodes cannot "absorb" the packets which are "in transit". Therefore, until the backpressure message reaches the sending nodes their buffer is already full and packets are getting lost.

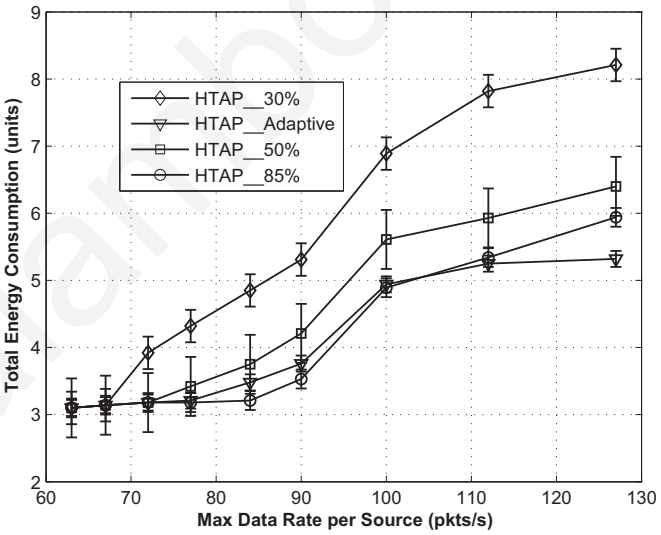

Figure 50: Total Energy Consumption.

To reinforce the findings of the previous figure, we study the Total Energy that is consumed during a congestion epoch. Fig. 50 indicates that when the threshold is set to 30% and 50% the energy that is consumed during a congestion epoch is high. We also notice that the energy

consumption starts even when the load in the network is not very high (67 packets/sec per source for "HTAP 30%" and 72 packets/sec per source for "HTAP 50%"). Since, according Fig. 49, packet drops are limited at these values, the energy that is consumed is due to the alternative path creation. Thus, these values affect negatively the performance of the algorithm. When the value is set to 85% we notice that the energy consumption is similar to the "adaptive" case and it just increases when the load in the network is very high (128 packets/sec per source). Similar findings exist for the average hop-by-hop delay (Fig. 51).

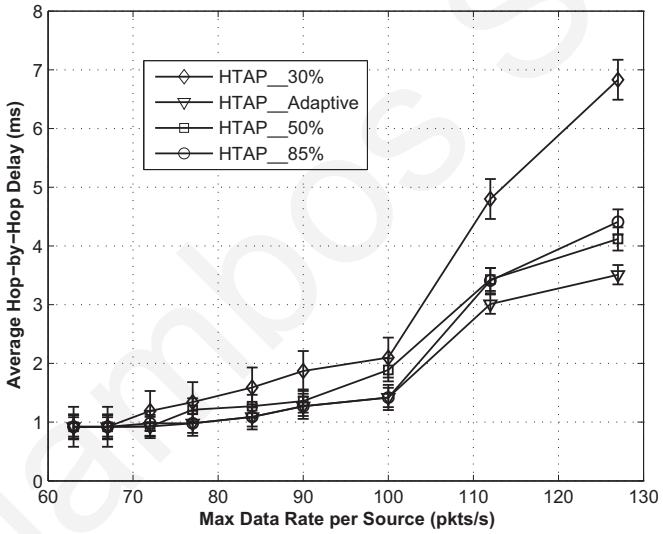

Figure 51: Average Hop-by-Hop Delay.

5.3.4 Summary

As we conclude the performance evaluation section, we can state that the HTAP algorithm provides an increased performance in the range of 2-3% in comparison with TARA, which is a comparable scheme. The topology control algorithm based on Local Minimum Spanning Tree

(LMST) that was added in this version of HTAP has been proven crucial for the successful operation of it. Specifically, this enhancement improved the average hop-by-hop delay, while the avoidance of "deadlocks" improved packet drops and retransmissions.

5.4 Concluding Remarks

In this chapter we presented the Hierarchical Tree Alternative Path (HTAP) algorithm. HTAP is an algorithm that manages to control congestion in WSNs using a resource control method. When congestion appears in the network the HTAP scheme transmits the excess packets to the sink through alternative routes, employing nodes which are not in the initial path from the source(s) to the sink. HTAP's successful and efficient functionality relies on a topology control scheme that creates the initial connectivity in the network and on a hierarchical tree scheme which discovers all possible upstream routes from the sources to the sinks when an event occurs. HTAP employs an adaptive "congestion threshold" that renders it capable to avoid transient congestion situations, while it is efficient enough to create alternative paths to the sink in order to control persistent congestion situations. The major advantage of the HTAP algorithm is its simplicity, which adds minimal overhead to the already heavy loaded networks that is intended to operate onto. HTAP has been evaluated and its performance was compared with another resource control algorithm (TARA) and the "no congestion control" case. Simulation results show that HTAP is an efficient and simple solution for facing overload situations in densely deployed WSNs.

Chapter 6

Dynamic Alternative Path Selection Algorithm

Studying the performance of HTAP algorithm we noticed that, in some cases, the average delay is slightly higher than the expected. Moreover, HTAP is an algorithm that bases its alternative path creation decision on buffer occupancy, a fact that requires an underlying efficient MAC protocol.

In this Chapter we present the Dynamic Alternative Path Selection (DAlPaS) algorithm. DAlPaS is a congestion control and avoidance algorithm that attempts to choose an alternate path in case of congestion, taking into account a number of basic performance parameters.

Complementary to Energy Aware Protocols [82, 83] that find the lowest energy route or energy sufficient paths to forward data and base their path alternation decision on these conditions, DAlPaS also takes into consideration the node's congestion situation (both in terms of buffer occupancy and channel interference). On the other hand, while congestion control and reliable data transmission protocols like [8,30,84] base their "alternate path" decision on a congestion threshold or the path's cost, DAlPaS also counts the node's remaining power.

DAlPaS differs from HTAP for three main reasons:

- DAlPaS routes packets through a sink-based tree that starts from sink, opposed to HTAP where each source node builds its own tree.

- DAlPaS bases its alternate path decision on a combination of reasons that is handled by a hard stage mechanism. On the other hand, HTAP alternates paths based on the buffer occupancy of nodes.

- DAlPaS does not depend its operation on the performance of the underlying MAC protocol.

DAlPaS is a dynamic and distributed algorithm. Besides the "Setup Phase", where the network is initially discovered, all subsequent decisions are based only on the condition of the node in the current data forwarding "epoch".

6.1 Network Model and Problem Description

We consider that the same network problem as with Chapter 5 exists. Specifically, we assume that there is a densely deployed WSN, where nodes are initially deployed randomly, but uniformly in space. One sink is located at a specific point in the network topology, while the number of source nodes is variable, since an event is possible to be random and be captured by more than one nodes. We also consider that each node knows its position and the position of the sink node(s) in the grid. In our network deployment we consider that all nodes beside sink are identical and CSMA/CA is employed as the MAC protocol.

The problem that we attempt to counter in this Chapter is, as with Chapter 5, to enable the network to deliver to the sink(s) all (or almost all) of the data packets which have been created, without intervening on the rate with which sensor nodes are injecting these packets in the network. Also, in this Chapter we attempt to improve the performance of the HTAP algorithm, by introducing a new algorithm which should be simple and efficient.

6.2 DAlPaS Operation

DAlPaS consists of four different schemes:

- Setup Phase
- Soft stage algorithm.
- Hard stage algorithm.
- Flag decision algorithm.

Soft, hard and flag decision algorithms are part of DAlPaS mechanism.

We now describe each scheme in detail, while in Fig. 52 we present the operation of DAlPaS through a flowchart.

6.2.1 Setup Phase

At the beginning of the Setup Phase the sink broadcasts a “Hello” message marked as Level 0. Nodes that are in the radio range of the sink receive this packet, mark it as Level 1 and set themselves as Level 1 nodes. Level 1 nodes re-broadcast this “Hello” message transmitting in full power. Upon receiving a “Hello” message for the first time, a node adds one to its level and broadcasts it again. A node may receive more than one “Hello” messages (from different neighbors). In such a case it rebroadcasts the message only if it has changed (lowered) its current level information. In case there are many nodes at the immediately lower level, it keeps all in its neighbor table, along with all other connectivity information it overhears. With this procedure, nodes discover each other, build and initialize their neighbor tables, and at the same time record their minimum hop distance to the sink.

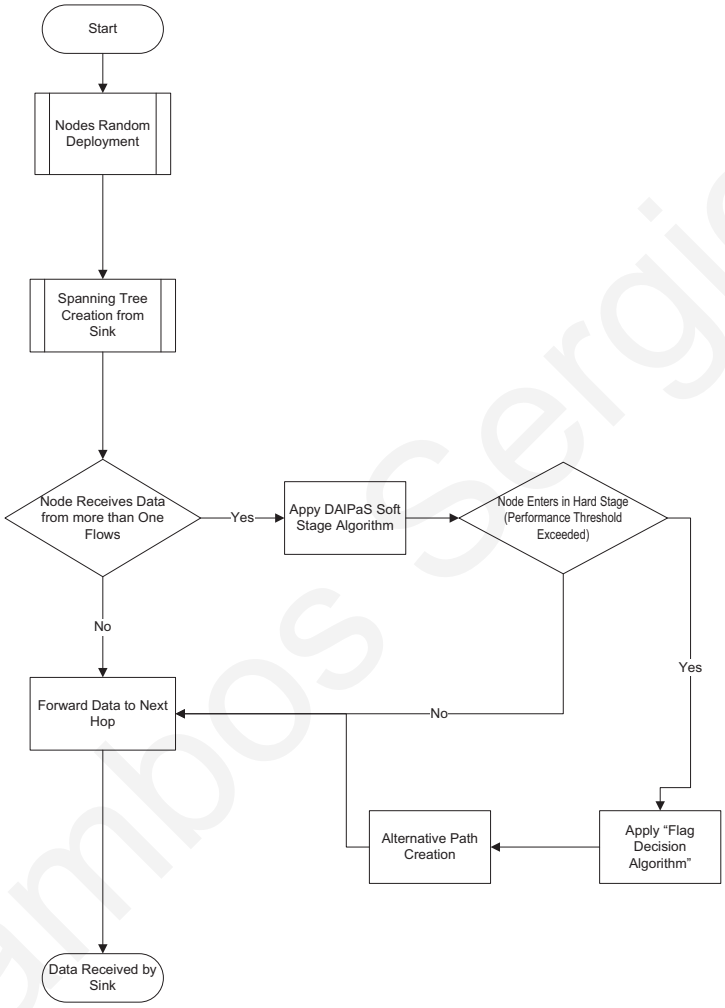

Figure 52: Flowchart for DAlPaS Algorithm.

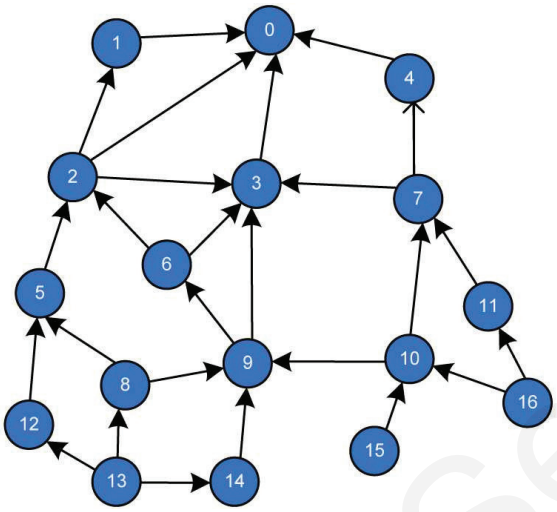

Figure 53: Initial Network Connectivity.

To explain this concept better let us consider the network in Fig. 53. In this scenario the sink (node 0) broadcasts a “Hello” message as Level 0 and nodes 1, 2, 3 and 4 receive this message and set themselves as Level 1 nodes. Then, these nodes broadcast the message downstream. Nodes 5, 6, 7 and 9 receive the “Hello” message and mark themselves as Level 2 nodes. The procedure iterates until the whole network is discovered. Fig. 53 actually illustrates the understanding of each node about all possible upstream paths from itself towards the sink through nodes at the same or lower levels.

When node 8 broadcasts its “Hello” message (as Level 3), node 9 will also receive it. Node 9 will compare the level of this message with the level it already possesses (Level 2) and will ignore it. If for any reason node 9 receives the message from node 3, after the message from node 8, it will update its level from Level 3 to Level 2 and will re-broadcast the updated level. Fig. 54 shows the connectivity resulting from using only the lower level nodes.

An example of neighbor table is presented in Table 2. The neighbor table maintains records for the ID of its neighbors, their buffer occupancy, their remaining power, their number of hops

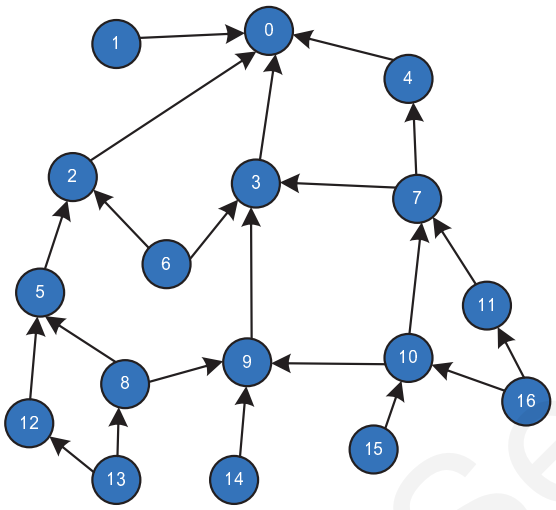

Figure 54: Placement of Nodes in Levels.

to sink, as well as a field called “Flag”, which indicates the node’s availability at the current moment. The Flag mechanism is explained later in this section. It is clear that the neighbor table holds information for all neighboring nodes and not only for the nodes that are one level closer to the sink. Using this “custom” flooding mechanism all possible routes, which can be used to forward packets upstream, are discovered.

Table 2: Example of Neighbor Table for Node 2.

Node ID	Buffer Occupancy	Remaining Power	Number of Hops	Flag
0	0	1J	0	True
1	0.90	0.95J	1	True
3	0.92	0.96J	1	True
6	0.85	0.96J	2	True
5	0.90	0.93J	2	True

6.2.2 DAlPaS Mechanisms

After the setup phase where all possible routes (paths) from source to sink have been discovered, nodes connected to the source begin to forward data. Initially, nodes forward their data packets through the node that provides the shortest route to the sink. Each data packet header contains the sequence number of the packet, the sending node ID and well as the destination (receiving) node ID. When a data packet is received by a node the packet must be acknowledged. If the node has successfully received the packet, it broadcasts an ACK packet which its headers contains a set of fields as described in Table 3. All nodes receiving this ACK packet, modify their neighbor table with the updated values.

DAlPaS scheme employ two stages. A soft and hard stage.

Soft Stage: Soft stage is introduced as a proactive way to face (or avoid) transient congestion situation as well as to balance and spread the traffic between as much nodes as possible. Each node is entered in DAlPaS soft stage's alert condition when it receives packets from more than one flows. In such a case each node that it faces this situation is a candidate congested node (in case that its receiving rate exceed the transmitting rate). In order to prevent this situation, DAlPaS attempts at first to keep each node receiving data from one flow. To achieve this, the node sets the "Next Packet Sequence Number" field in the ACK packet header to "False" for the specific node ID that would like to "inhibit" its transmission. When this node receives this ACK packet it is informed that it should check for next suitable path starting from a another node at the same level as before. The sending node is able to understand that the reason of this inhibition is another flow, since the Flag field in the ACK packet remains "True". Note, that in soft stage receiving nodes just advice the sending nodes to find another path, thus they keep receiving and forwarding their data until forwarding nodes decide to stop sending packets. In soft stage, receiving nodes

always try to keep the flows with the bigger sending data rate. By employing this tactic besides the obvious benefit of buffer based congestion avoidance, the network utilizes its resources uniformly and routing holes are avoided.

Hard Stage: In case of high traffic load or in case where the performance requirements of the application especially in terms of delay are not satisfied through a new routing path, it is possible that an "inhibited" node to decide to keep sending packets to the same node although it is aware that it is possible to cause congestion. In this case if receiving node exceeds the "performance threshold" is entering in hard stage.

Hard stage is a situation where the network **forces** the flows to change routing paths since "performance threshold" has been exceeded. Responsible to monitor and apply performance thresholds is a "Flag Decision" algorithm which is described below.

Flag Decision Algorithm: The flag decision algorithm runs when a node enters in hard stage. In this stage a node becomes temporarily or permanently unable to accept any more packets from any flows. A node may become unable to receive data for the following reasons:

- **Buffer Occupancy is reaching its upper limit**: If a node receives packets at a higher rate than it can transmit, it will soon have its buffer overflowed. For example, in Fig.55 this can happen if nodes 10 and 11 send data to node 7 and their aggregation data rate is more than the data rate that node 7 can forward. In this case the buffer of node 7 soon overflows.

  In a scenario like this, when the receiving node figures out that its buffer is about to overflow it sets its Flag field to in the next ACK packet that it broadcasts to "False". All nodes in the area that receive this packet alter immediately the flag entry for this node in their "Neighbor Table" to "False" and look in their table for the next hop that is available to forward their

data. When the "failed" node has recovered and is able again to receive packets, it broadcasts a "Hello" packet stating this event and neighbor nodes alter the flag to "True" in their neighbor table.

- **Low Remaining Power**: The "Flag Decision Algorithm" also applies in the case that a node is getting power exhausted. Each node is programmed to set its flag to "False" whenever its remaining power falls below a certain percentage of the total.

  When this happens, whether during a data session or when a node runs in idle, the node alters its Flag to "False" and informs its neighbors for this event through an ACK or a "Hello" packet. The nodes that receive this message apply the same procedure as for the buffer occupancy case respectively. The remaining power threshold depends on the application. In time critical applications it is best to be kept low to make sure that we use the shortest available path for the longest time. In periodic applications it may be set to a higher level in order to maintain a more energy-uniform utilization of the network.

- **Higher level node unavailability**: Another case in which the "Flag Decision Algorithm" applies is when, although a node is available, concerning the buffer and power availability, there is no other node available at a level higher that itself to transmit data to. In this case it is also forced to advertise a "False" flag. If the nodes at the higher level are not available due to power extinction, buffer occupancy or because they may have been physically removed from the network, this functionality protects the network from forwarding packets to network "routing holes".

Alternative Path Creation: Since the algorithm is dynamic, the number of hops to the sink for a node is possible to change when the state of nodes at a level closer to sink changes. The choice of the next node to forward data, after avoiding the congested node, depends firstly on its

Table 3: ACK Packet Header.

node ID	Next Packet Sequence Number	Buffer Occupancy	Remaining Power	Number of Hops	Flag

availability (Flag) and the number of hops to the sink. Using this tactic each node can, with an easy and simple way, to find the next node with minimum computation. It just sorts the number of available nodes in ascending order with respect to their number of hops to the sink and forwards the packets to the first node in the list. If the first node becomes unavailable in any way (e.g soft or hard stage algorithms apply) the sender immediately chooses the next node in the list. In case that more than one nodes are in the same level (same number of hops to sink), the table is sorted based on the remaining power (above or below some specific thresholds). Finally, in case that more than one nodes are above these thresholds they are sorted based on their remaining buffer occupancy. In the extreme case where even this value is the same for more than one nodes, the algorithm chooses the node with the smaller node ID to forward the packet.

Using this method the algorithm gives priority to the maintenance of performance metrics like the mean time for the transmission of packets from source to sink, as well as to the network's uniform energy utilization, thus avoiding the creation of energy and routing "holes".

For example let us consider Fig. 55 and Fig. 56. In this example node 14 forwards packets to the sink through nodes 9 and 3. If for any of the reasons explained above node 3 fails, node 9 will have to search in its neighbor table to find a node to replace node 3 since it cannot reach directly the sink. We note (from Fig. 54) that node 9 does not have any other "Level 1" nodes to connect to. Therefore, it must search for "Level 2" nodes. In this case it finds node 6, from which it will forward the rest of the data. In this case since node 6 is "Level 2" node, node 9,

immediately becomes “Level 3” node. Concurrently all nodes connected to it (8, 10, 14) will update their tables.

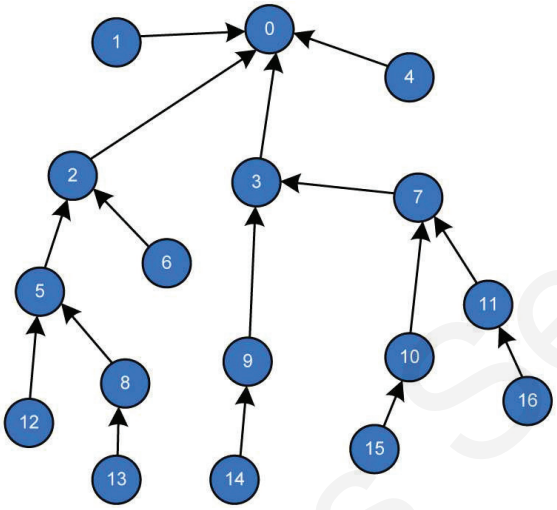

Figure 55: Shortest Paths from Source to Sink before Node Failure.

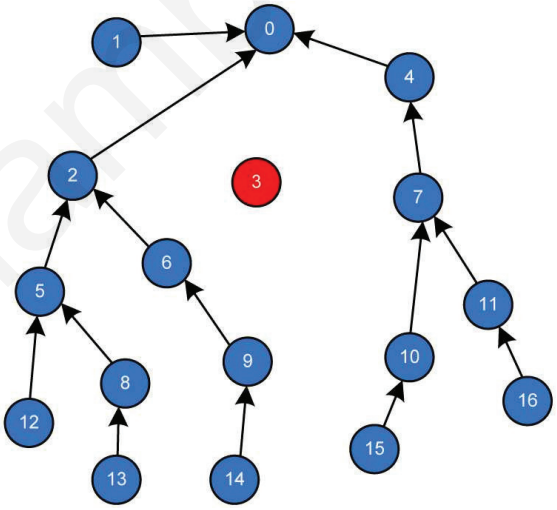

Figure 56: Paths from Source to Sink after Node Failure.

Depending on the application a **weighted function** could be implemented, concerning the choice of the alternative path. As we stated above, four parameters define the choice of the alternative path: *Availability (flag)*, *number of hops to sink*, *remaining energy*, and *buffer occupancy*.

For example, if there is demand for low delay, the *number of hops to sink* should have the highest weight. If the network lifetime is more important, then *remaining energy* should take the highest weight. In this chapter we consider that all parameters are equal, giving priority to *hop count*.

An example of a weighted function is presented here. We consider again the topology of Fig. 54 and the fact that node 3 has failed. In this case we consider that there is an application that considers as primary factor for the choice of the alternative path, the remaining *buffer occupancy*. Thus, it applies a weighted function, capable to capture this issue. This function is presented in (29)

$$Weight = [B \times 6) + (P \times 2) - (H \times 2)] \times F \tag{29}$$

where B equals to remaining *buffer occupancy*, P equals to the *remaining node energy*, H equals to *hop count* and F to the *Flag*. The values of *Flag* could be either 0 or 1. If it is equal to 0, it means that the node is unavailable and weight becomes equal to 0, value which actual declares that this node is indeed unavailable. Also, in this function we subtract the number of *hops to sink*, since the more they are, the less desirable this node becomes.

Table 4: Example of Neighbor Table for Weighted Function.

Node ID	Buffer Occupancy	Remaining Power	Number of Hops	Flag	Weight
10	0.8	0.4J	3	0	0
7	0.8	0.8J	3	1	5,8
6	0.4	0.2J	2	1	2,4
8	0.9	0.3J	2	1	5,2

Thus, studying the results of Table 4, we notice that node 9 will now select node 7 as an alternative node, instead of node 6 (choice that was just based on the *number of hops to sink*).

6.3 Algorithm Evaluation

The DAlPaS algorithm has been evaluated through simulations, and its performance is presented in this section. The evaluation has been performed using Prowler network simulator [72] using the simulation settings and parameters of section 3.4 of Chapter 3. The performance of DAlPaS has been compared against TARA, HTAP and No Congestion Control (No CC) algorithms, as well as against a version of DAlPaS algorithm where no soft stage is implemented, called "DAlPaS HARD".

For each of the performance metrics presented below, the results is the average of 20 runs for 10s for each measurement point (except for the cases that it is otherwise stated). Nodes are placed uniformly on a square grid over a 500x500m area. The initial number of deployed nodes is 100. The event happens randomly in the left bottom quadrant of the grid, while the sink is situated on the upper right edge of the grid. Nodes have a sensing range of 25 meters and a communication range of 50 meters. All nodes that sense the event are becoming source nodes using the rest as relay nodes to the sink.

The first performance metric we have examined is the percentage of successfully received packets (Chapter 5, Eq. (26)). This metric is particularly important for critical/emergency applications, where every packet has to be received by the sink and indicates the ability of algorithms to maintain a robust network. Results for this particular metric are presented in Fig. 57.

It is clear that if no congestion control algorithm is applied in a WSN while the amount of data injected in the network is increasing, this will result in the severe degradation of the network's performance, since an important percentage of the transmitted packets will not reach the sink. On the other hand, when congestion control algorithms are applied the results are significantly improved. Focusing on congestion control algorithms we notice that DAlPaS presents better performance in

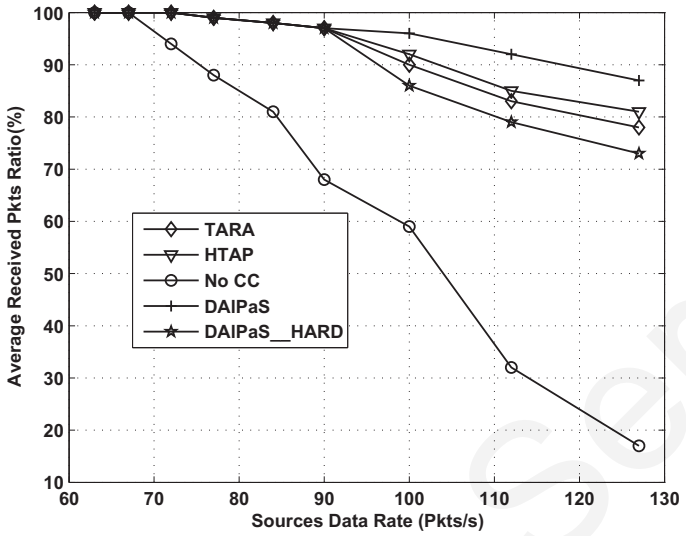

Figure 57: Percentage of Successfully Received Packets.

comparison with the other algorithms. This result indicates that the mechanisms of DAlPaS are efficient and can balance the traffic load between nodes efficiently, in order to avoid packet drops. This conclusion is reinforced by the fact that if we remove the soft stage mechanism from DAlPaS, then "DAlPaS HARD" presents worse results in comparison with all three algorithms. This fact indicates how important is to balance the traffic between nodes at early stage before congestion occurs.

The advantage of DAlPaS is more evident if we consider the total number of packets that manage to reach the sink (Fig. 58). DAlPaS manages to deliver 20-25% more packets to the sink, compared to HTAP and TARA respectively, and even more when the soft stage mechanism is not implemented ("DAlPaS HARD"). This benefit is clearly attributed to the employment of a "soft stage topology control" where nodes are being "advised", in the first place, to avoid using as next nodes those that already serve another flows. This fact helps the network to utilize its resources more uniformly.

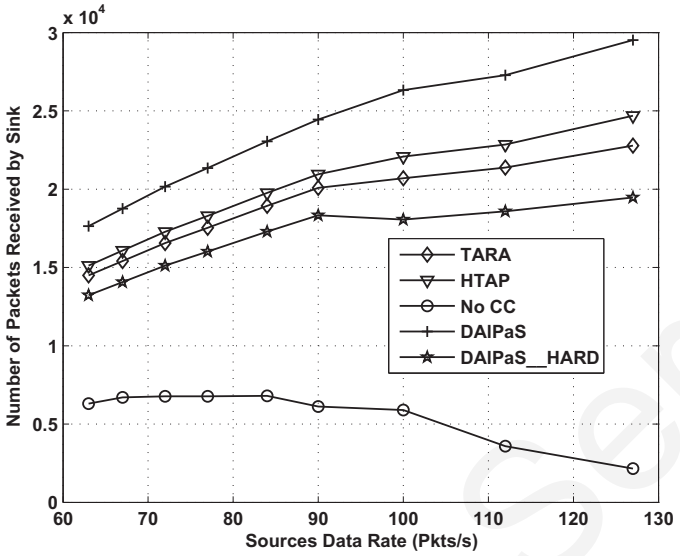

Figure 58: Number of Packets Received by the Sink.

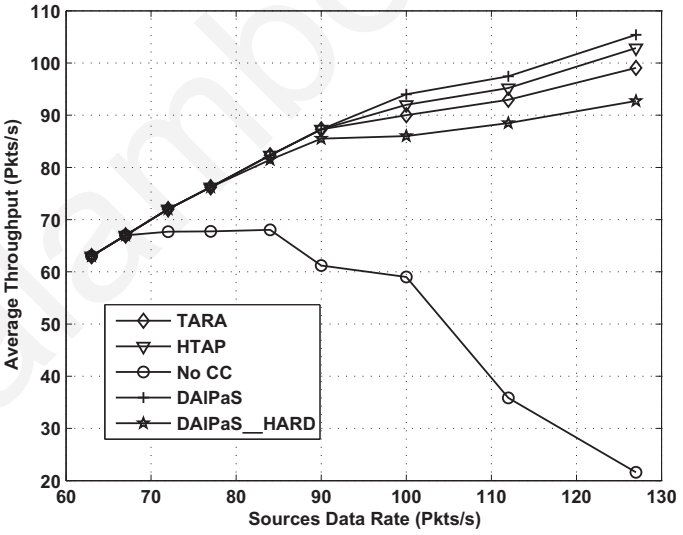

Figure 59: Average Throughput with Increasing Load.

The next parameter we have analyzed is the rate of data that reach the sink (throughput). The graph in Fig. 59 should be considered in conjunction with Fig. 57 in order to compare the actual

number of packets that reach the sink when each algorithm is applied. We recognize that the DAlPaS algorithm exhibits the best performance.

The next metric we have evaluated is the "average hop-by-hop delay". This metric is an indication of how algorithms handle inter-path interferences, link layer retransmissions and overhead introduced by control packet exchanges.

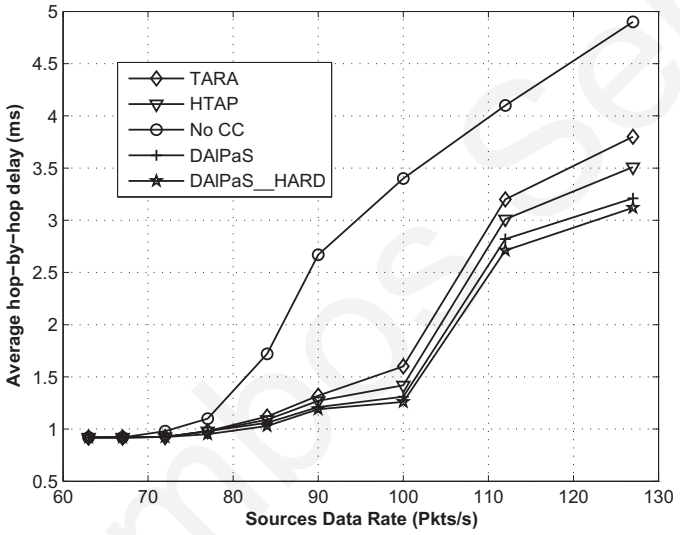

Figure 60: Average Hop-by-Hop Delay.

Studying the results of Fig. 60 we notice that "DAlPaS HARD" presents the least delay between the other algorithms. This is normal because DAlPaS algorithm without the soft stage mechanism implemented is completely lightweight and only runs when a node enters in hard stage. But bearing in mind the previous we notice that DAlPaS with soft stage implemented is much more efficient than without.

Finally, we evaluate the percentage of the network's remaining energy. In this simulation series opposed to the other metrics, simulations run until the network is not able to deliver any more packets from the source to the sink, due to several power exhausted nodes. Calculating the

remaining power of "alive" nodes we get an indication of which algorithm manages to utilize more efficiently the network resources and increase the network's lifetime. In this series of simulations we used 300 nodes, with a source data rate of 100 packets/s.

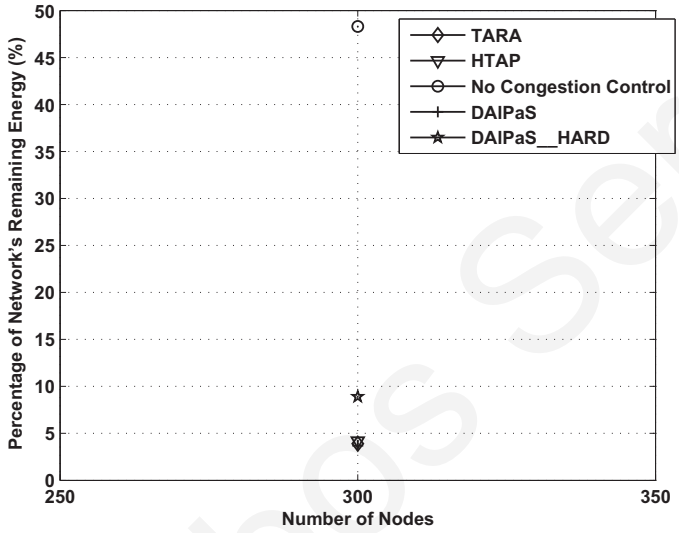

Figure 61: Percentage of Networks Remaining Energy.

Fig. 61 indicates that when "No Congestion Control" is employed in the network, the network "stalls" when the total remaining power of nodes is near 50%. In other words, when the network utilizes 50% of its resources it stops performing. On the other hand resource control algorithms manage to utilize almost all network resources. This is a strong indication that nodes are exhausting their power uniformly and the network's lifetime extends considerably.

If we focus on the four resource control algorithms (Fig. 62) we notice that, again, the DAlPaS algorithm presents better results than HTAP and TARA and even better that "DAlPaS Hard". This result is another indication of how important is the soft stage mechanism of DAlPaS algorithm in its operation.

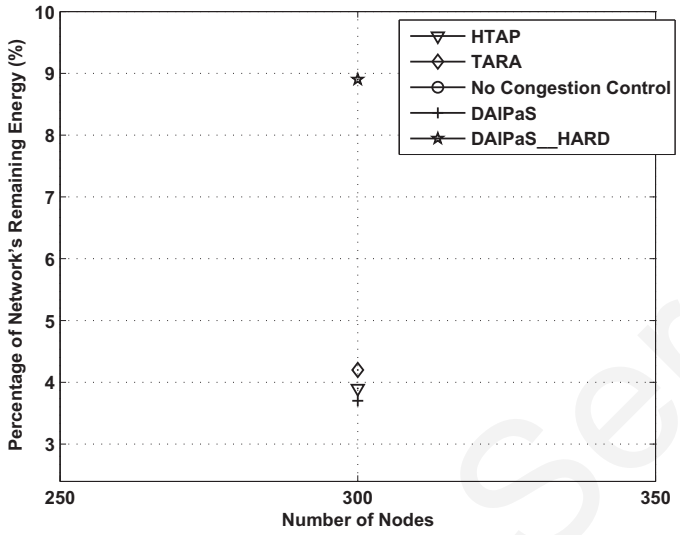

Figure 62: Percentage of Networks Remaining Energy (Zoom of Lower Part of Fig. 61).

6.4 Discussion on the Complexity of DAlPaS and HTAP Algorithm

In this section we discuss the complexity of DAlPaS and HTAP algorithms in comparison with the TARA algorithm.

Beginning from HTAP, as it is shown by the results, it is a lightweight and simple algorithm. The simplicity of the HTAP algorithm lies on the fact that when congestion occurs in the network, the computations that the HTAP performs are minimum. Thus, when the load in the network is high, nodes than run the HTAP algorithm will just perform minor computations.

If we study the operation of HTAP algorithm, step-by-step we notice that:

- The topology control algorithm runs only once, at the beginning of network operation.
- The Hierarchical Tree algorithm runs when a node becomes a source node and before any congestion occurrence in the network.

- Alternative Path Creation runs when the buffer occupancy of the node exceeds 85% and the monitoring begins according to the "Adaptive threshold algorithm".

- Congestion notification is implicit.

The signaling overhead in HTAP is higher when the topology control algorithm runs and especially when source based trees are created, but it is kept at minimum levels when congestion occurs and alternative path creation needs to be applied.

On the other hand, the DAlPaS algorithm is simpler. DAlPaS, after the setup phase, attempts in the first place to avoid congestion through the soft stage mechanism. If the load is high, the soft stage algorithm incurs signalling overhead, since messages are exchanged between the nodes in order to find a new route. However, when congestion occurs, the hard stage mechanism has a small signalling overhead.

If we compare the operation of DAlPaS and HTAP with TARA algorithm, we notice that TARA algorithm is much more complicated, since each time congestion occurs, nodes should find a proper topology, through a graph coloring scheme, in order to route the excess data. This "heavy process" in HTAP and DAlPaS, runs only once (at the beginning). Also in TARA, a distributor and a merger node should be specified in each congestion session, while in HTAP and DAlPaS routing is performed on the tree that has already been created, before congestion occurrence.

6.4.1 Message Complexity Analysis

At this point we present a message complexity analysis of HTAP, DAlPaS and TARA.

1. **HTAP:**

   Initial Phase: HTAP's operation is based on a source based tree. As we presented in Chapter 4 the required messages for the creation of a source based tree results to messages

exchange equal to $n^2 + 2 \times N.length$, which means that the complexity is $O(n^2)$. HTAP using the LMST algorithm attempts to reduce this message exchange. For this reason, every node in HTAP is connected to at maximum six (6) nodes. Thus, each node exchanges messages with at maximum 6 nodes, one hop way. This results to message exchange, at the worst case, equal to $6 \times n$. This means that the complexity for the creation of the initial tree is equal to $O(n)$.

Data Flow in normal conditions: During normal network conditions (no congestion exists), the messages are generated at sources and are forwarded to the sink via the network. Each node is acting as a relay node, forwarding incoming messages according to its routing table. Each received data message is acknowledged. We denote the distance d as the hop distance from source to sink. Consequently each node in the source - sink path transmits one data message and one acknowledgment for each message generated at the source. This results to a total of $2 \times d$. In the worst case, where all nodes participate in the transmission this would rise to the maximum of $2 \times n$, which is of order of $O(n)$.

Data flow under congested conditions: Upon congestion detection, the congested node piggybacks its congested state to the acknowledged packet and the receiving node acts appropriately. When congestion occurs each node adapts itself by using the information from the neighboring table which is updated at every acknowledgment reception. Therefore, the overall message exchange is not burdened and the message complexity remains low, as in normal conditions ($O(n)$).

2. **DAlPaS:**

Initial Phase: Unlike HTAP, DAlPaS constructs a sink-based tree. This is achieved by flooding the network from the sink with an initial hello message, which also contains the

level of the node in respect to the sink. Each node that receives the initial hello message, adds one to the level field of the message and retransmits it. The receiving node marks as a parent the sending node and updates its Neighboring Table. This induces a retransmission of the initial hello message n-times. Thus, the network creation is of message complexity ($O(n)$).

Data Flow in normal and congested conditions: Similar to HTAP, nodes in DAlPaS algorithm only update their neighboring tables from information contained in the acknowledgments. On congested situations each node adapts accordingly, without the need of additional messages that would overload the wireless channel. Each message generated at a source, requires $2 \times d$ messages to reach the destination sink. Thus, in the worst case the message complexity is ($O(n)$).

3. **TARA:**

Data Flow in normal conditions: TARA authors do not provide the necessary details for the network construction, thus we assume a sink-based tree initialization as in DAlPaS. The same applies to normal network conditions. ACKs are used to acknowledge the reception of packets, therefore each message generated at a source requires $2 \times d$ messages to reach the destination sink.

During congestion detection, TARA performs the following tasks in order to alleviate congestion:

- Locate Traffic Distributor
- Locate Traffic Merger
- Establish Detour Path

Traffic Distributor: To locate the Traffic Distributor node, the congested node transmits an upstream packet towards the source. The first node, with low congestion state, that receives the packet and belongs to the source-sink path (the notion *stream* is adopted) is elected as the Traffic Distributor. This operation might be neglected as it requires a few acknowledgment retransmissions from congested node towards the source node and this would be equal to a fraction of *d*, which is the hop-distance from source to sink.

Traffic Merger: To locate the Traffic Merger, the Distributor transmits a downstream control packet towards the sink with the ID of the node that the traffic is destined for. The messages needed to complete this action are also roughly evaluated to a fraction of distance *d*.

Establishing the Detour Path: After the merger is selected, the Distributor must be informed about the new detour path. To achieve this, the Merger floods locally a *REQ* packet including a time-to-live (TTL) field, towards the Distributor. This local flooding could result to a retransmission of $N.length \times (TTL - 1)$ times of the *REQ* packet until it reaches the Distributor, where *N* is the table of neighboring nodes of each node, and TTL the time-to-live field value. To have a brief understanding of this number we could assume a neighboring table of length equal to 9 and a TTL value of 6, which would result in more than 50 retransmissions. In a network of 100 nodes the order of magnitude of the generated messages is $O(n)$ and is proportional to the density of the network and the chosen TTL value. We should notice that, according to TARA authors, the merger could be located one hop away from the sink in some cases, which would lead to a non negligible value of TTL. If for example the hotspot is near the source, the TTL could easily be greater than $\frac{d}{2}$, thus generating the sum of $N.length \times \frac{d}{2}$ messages. It is also important to note that is possible for more than one nodes to be elected as mergers. It is the Distributor's choice which alternative path

is of greater capacity. According to TARA authors, *REQ* packets are retransmitted based on some criteria. To include this in our analysis we can introduce a function $f(t)$ which represents the number of the candidate mergers, depends on the topology of the network and may include an heuristic to calculate the possible retransmissions. Therefore, the resulting message sum is *N.length* $\times (TTL - 1) \times f(t)$.

6.4.2 Time Complexity Analysis

At this point we present Time Complexity Analysis oh HTAP, DAlPaS and TARA.

1. **HTAP:**

   In order to achieve its operation, HTAP needs the following data stored in each node:

   - Node Id
   - Number of Hops to Sink
   - Data Rate (when *Congestion Detection* algorithm runs locally)

   Upon reception of a message, each node forwards this message to a node at a higher level (a node with less hops to sink- its *parent*). This node is firstly chosen during the initialization phase. Upon congestion detection, based on buffer occupancy, the congested node runs a *Congestion Detection* algorithm locally. This algorithm sums the total incoming packets from all the *transmitting* nodes and when the sum exceeds a threshold, the congested node sends a backpressure message to specific nodes through the acknowledgments. Nodes that transmit at lower rates are selected first. During the operation of the *Congestion Detection* algorithm, each message received, requires an update to the neighbor table, of order $O(N.length)$, a sum of all data rates of the same order and, a comparison to the threshold, resulting to $2 \times N.length + 1$ operations which is of order O(*N.length*).

When the threshold is exceeded, the congested node informs through the acknowledgment messages the *transmitting* nodes about its condition. When the *transmitting* node receives the acknowledgment, it is aware that the *parent* node is close to congestion. Thus, it simply chooses another node from its Neighbor Table, with hop count *less* or equal to its own. This operation requires a traversal of the Neighbors Table, that requires time O(*N.length*) for every *transmitting* node.

2. **DAlPaS:**

In order to achieve its operation, DAlPaS needs the following data stored in each node's Neighbors Table:

- NodeID
- Buffer Occupancy
- Remaining Power
- Number of Hops
- Flag

The size of the neighboring table is proportional to the density of the network and depends on the communication range of the nodes. During congestion, the selection of a node depends firstly on the *flag* value, secondly on its *Number of Hops* from sink, then on its *Buffer Occupancy* and finally, on its *Remaining Power*. The table is created during network initialization but it is constantly updated at every acknowledgment reception. Choosing a simple array structure to save this information will result to the following time complexities:

- add Element: O(1)
- find/update Element: O(*N.length*)

- array sort: O(*N.length*)

For each message delivery, a fraction of the network's total nodes will have to perform an O(*N.length*) time complexity operation. If we consider the whole network as a distributed system, a message generated at source will induce an operation of $d \times N.length$ time complexity all over the network.

Upon congestion detection, the congested node transmits a negative acknowledgment, which results to a O(*N.length*) operation in *N.length* nodes, that will receive the packet. Afterwards, the network operation continues as in normal conditions; each node adapts by itself according to its neighboring table. A node selection after congestion detection can take at most $4 \times N.length$ time, which is of the order $O(N.length)$.

3. **TARA:**

As we already stated, TARA does not provide details on network construction. It is stated though that each node maintains in its neighboring table the incoming traffic volume from each of its neighbors. Thus, every message reception will induce an update in the neighboring table, an operation requiring O(*N.length*) time.

On congestion detection, the congested node transmits a control packet downstream, which may be retransmitted limited number of times m, in order to locate the traffic distributor. This will induce $m \times N.length$ total calculations, one in each node.

Afterwards, to locate the traffic merger, the distributor will send upstream a control packet, which will be retransmitted a number of l times (where $l < d$), until it reaches a node capable to become the merger. Consequently this will induce $l \times N.length$ total calculations, one in each node until the merger.

When the Merger is elected, it tries to establish the detour Path by locally flooding an *REQ* packet including a Time-To-Live (TTL) field towards the distributor. A node may receive multiple *REQ* packets from a merger due to: a) the nature of flooding and, b) many nodes have the potential to become mergers. According to the specified criteria by the authors of TARA, any node receiving a *REQ* packet may either choose to retransmit it with an updated TTL value or drop it.

If we assume that m_c candidate mergers exist, these noses will flood the network with equal number of *REQ* packets. Each *REQ* packet may be retransmitted by $N.length \times TTL$ at equal number of nodes. Following our analysis about the establishment of the detour path, we can assume a sum of

$$N.length \times (TTL \cdot f(t) + m_c + l) \tag{30}$$

calculations with time complexity $O(n)$ each and overall complexity $O(n^2)$.

6.5 Concluding Remarks

In this chapter we present and study the performance of the DAlPaS algorithm, an algorithm designed for congestion control and avoidance in Wireless Sensor Networks. DAlPaS is a resource control congestion control algorithm and it has been evaluated against TARA and HTAP which are also resource control algorithms, as well as against "DAlPaS HARD" which is the version of DAlPaS without the implementation of soft stage mechanism. Also it has been evaluated against "No CC", which represents the case where no congestion algorithm is applied in the network. Simulation results show that DAlPaS algorithm outperforms the other two resource control algorithms (TARA and HTAP) as well as the case when "DAlPaS HARD" is employed. The only

case that “DAlPaS HARD” presents better results than HTAP is concerning the hop-by-hop delay. Finally, we perform complexity analysis between HTAP, DAlPaS and TARA algorithms.

Chapter 7

Performance Evaluation of HTAP and DAlPaS through Different Node Placements

In this Chapter we study how different node placements can affect the performance of the HTAP and DAlPaS algorithms. Different node placements are possible to provide a variable number of paths, which can improve or reduce the performance of HTAP and DAlPaS.

7.1 Node Placements

The placement of nodes in a network can be divided into three major categories. In this work we choose to place nodes in four different placements that cover all categories: a deterministic placement (Grid), a semi-deterministic (Biased Random), and two non-deterministic (Simple Diffusion and Random).

7.1.1 Deterministic Node Placement

In deterministic node placement, nodes are placed on exact pre-defined points on a grid or in specific parts of the grid. Usually, deterministic or controlled node placement is specified by the type of nodes, the environment in which the nodes will deploy, and the application. Therefore,

in applications like indoor surveillance systems or building monitoring, nodes must be placed manually [85] (either by hand or by robots).

Grid Placement: In this placement nodes are placed strictly on the lines of a Grid (Fig. 63).

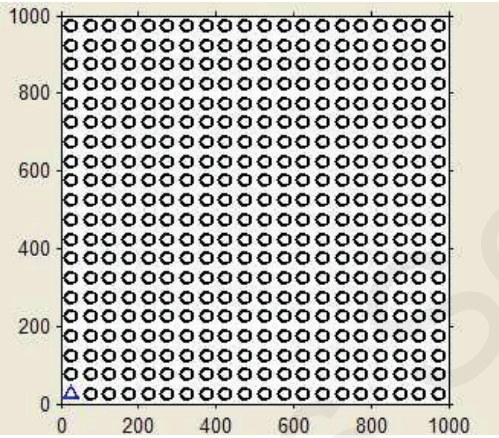

Figure 63: Grid Placement.

7.1.2 Semi-Deterministic Node Placement

Semi-deterministic placement is the placement where, although individual nodes are placed in a non-deterministic way on the grid (e.g random), the areas where nodes are going to be spread are pre-determined. This means that in a microscopic way the placement of nodes is non-deterministic, while in a macroscopic way the placement is deterministic. In this Chapter we employ biased-random placement, where nodes are placed in two specific areas (near source and near sink). Note that the actual node placement is performed in a random way in these areas (Fig. 64).

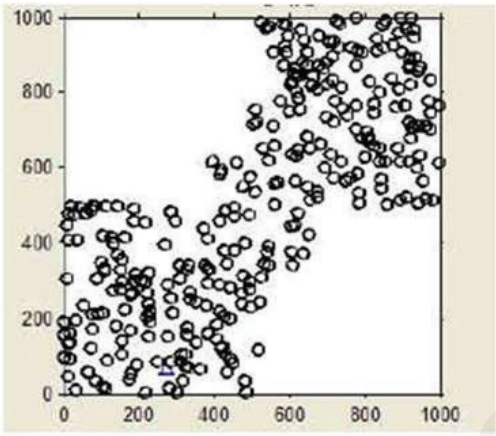

Figure 64: Biased Random Placement.

7.1.3 Non-Deterministic Node Placement

Deterministic placement is not so realistic when many sensor nodes are placed in a large area. In such situations, stochastic placement is needed. In this Chapter we employ two stochastic placements: Simple Diffusion and Random placement.

Simple Diffusion: This node placement emulates the distribution of nodes when they are scattered from air, i.e. from airplane (Fig. 65). Simple diffusion placement is analytically explained in [86].

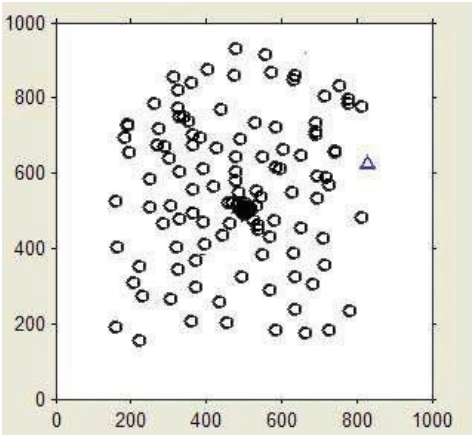

Figure 65: Simple Diffusion Placement.

Random Placement: This is a commonly used topology and sensor nodes are placed so that their density is uniform (Fig. 66). This is the topology that has been used for the simulations in this thesis as well as in many WSN-related research.

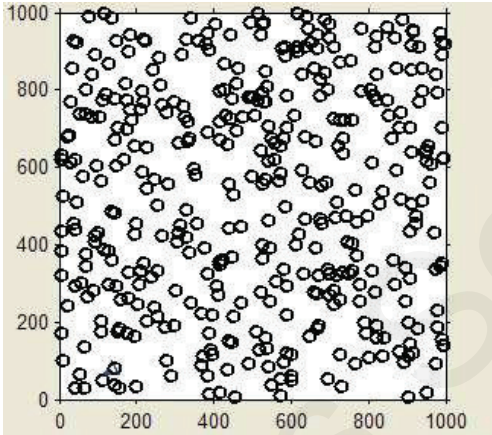

Figure 66: Random Placement.

7.2 Performance Evaluation of HTAP Algorithm under Different Placements

For the evaluation of HTAP under different placements we employ the same simulation parameters and as of section 3.4 of Chapter 3. Again, 100 nodes were deployed in 500m x 500m grid under the placements we described in the previous section.

7.2.1 Results

The first metric we study is the percentage of successfully received packets.

As we notice in Fig. 41, where the placement is random, at the maximum data rate (128 packets/s) the percentage of received packets is near 80%. With exactly the same simulation parameters and different placements we notice that this percentage changes. When nodes are placed under a Biased-Random placement, the percentage is much higher. Even when Simple Diffusion applies,

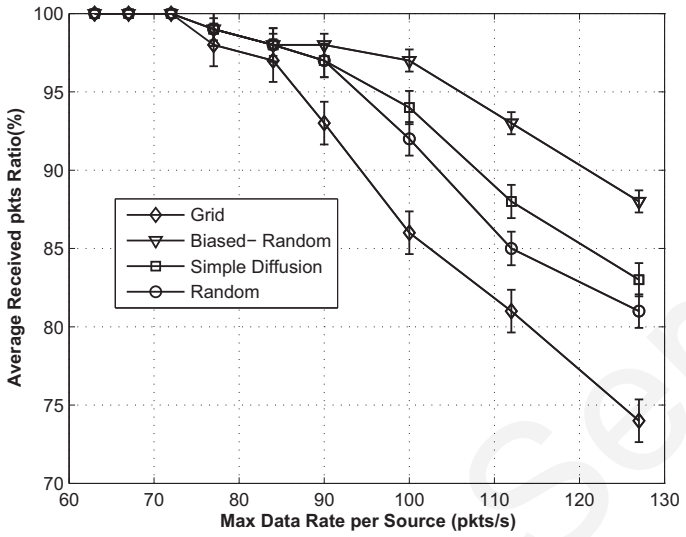

Figure 67: Percentage of Successfully Received Packets.

the percentage is higher than Random placement, while Grid placement presents the worst results. The reason lies on the number of paths which are created when diverse placements are employed. When the Biased-Random placement is employed, nodes are placed near the sources and the sink which are the areas with the highest probability of congestion occurrence. Subsequently, more paths are available in order to extract and transmit data to sink. On the other hand, when the Grid placement is used, the number of paths is limited. This means that network capacity is smaller and the results are the worst.

The next metric we consider is the actual number of packets that is being received by sink (throughput). As we notice in Fig. 68, the Biased Random placement presents the best results, and the number of packets that are being received by the sink increase. Simple Diffusion also presents better results than Random placement and Grid placement presents the worst results.

Next, we study the Average Delay (Fig. 69) and the Total Energy Consumption (Fig. 70).

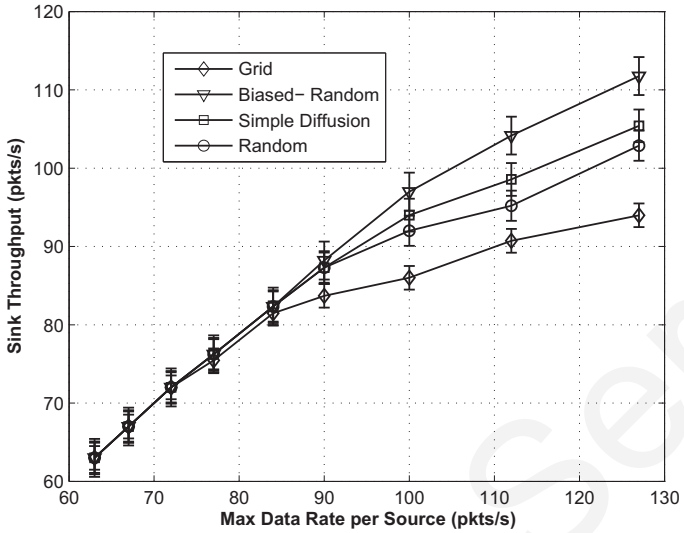

Figure 68: HTAP: Sink Throughput.

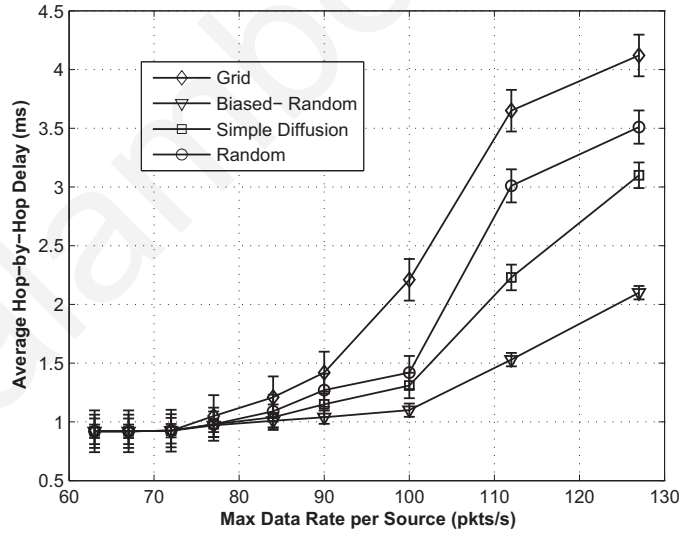

Figure 69: HTAP: Average Delay.

Also in this case the results are in correlation with the results in Fig. 67 and 68. Hop-by-Hop delay and Total Energy Consumption are strictly related with packet drops and link layer retransmission and the results again vary under different placements.

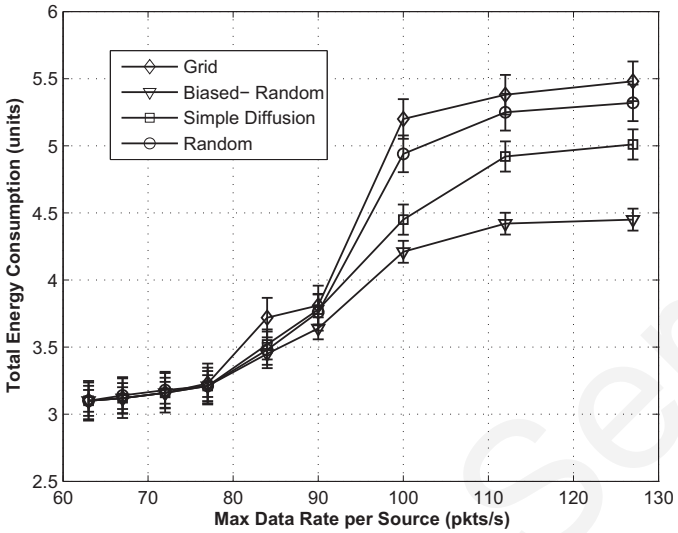

Figure 70: HTAP: Total Energy Consumption.

Finally, we study the percentage of network's remaining energy (Fig. 71) at the point where the network is unable to transfer a single packet from a source to the sink.

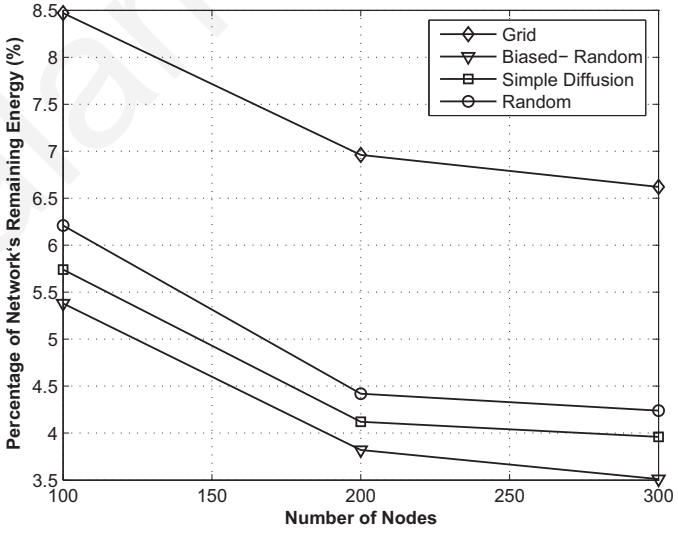

Figure 71: HTAP: Network's Remaining Energy (%).

Analyzing these results we notice that when nodes are deployed under Biased-Random placement, the percentage of the network's remaining energy is the best, indicating that under this placement, the utilization of network resources is more uniform in comparison with the other placements.

7.3 Performance Evaluation of DAlPaS Algorithm under Different Placements

Concerning the evaluation of DAlPaS algorithm under different placements, we employ the same simulation parameters and we just alter the employed placements.

7.3.1 Results

The first result is about the percentage of successfully received packets (Fig. 72).

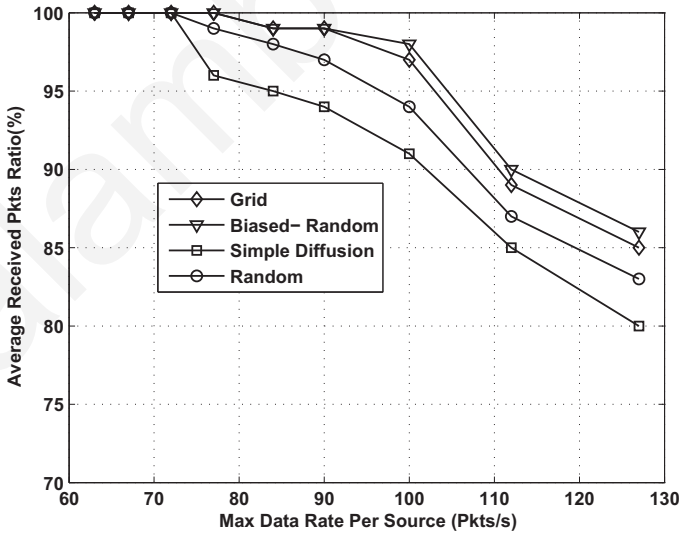

Figure 72: DAlPaS: Percentage of Successfully Received Packets.

Studying this result, we notice a different attitude of DAlPaS algorithm in comparison with HTAP. Specifically, we notice that DAlPaS algorithm is not significantly affected by different

node placements, as HTAP. Moreover, we notice that placements like Grid, where HTAP does not presents high performance, DAlPaS presents very good performance. The reason for this attitude lies on the operation of DAlPaS algorithm. DAlPaS maintains records of all nodes that can connect with and employs these nodes in order to create alternative paths to the sink. Contrary, HTAP algorithm maintains information only about the six nodes that can connect with, according to the results of the topology control algorithm. For this reason, Grid placement which is a structured and dense placement, can provide to DAlPaS several alternative paths, while concerning HTAP can provide only three (there are only just three nodes, one hop away from each source).

These results are reinforced by the next figure (Fig. 73).

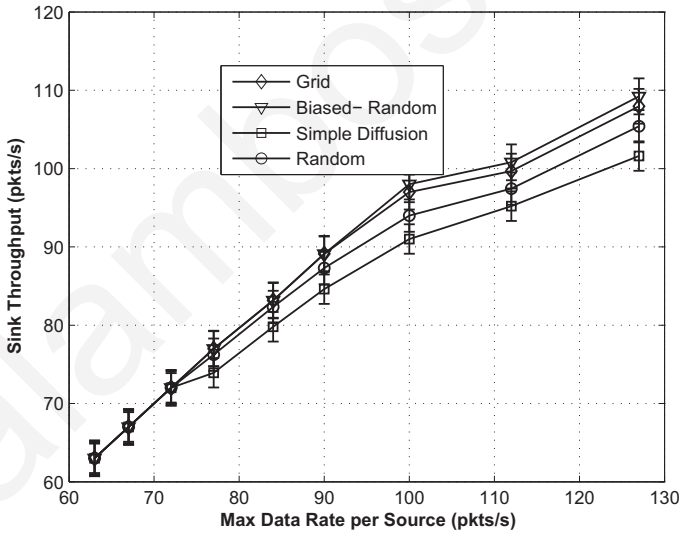

Figure 73: DAlPaS: Sink Throughput.

In this figure we notice that throughput is not significantly affected by different node placements as it happens with HTAP algorithm.

If we focus only on DAlPaS, we notice that Biased-Random and Grid placements presents better results in comparison with Simple Diffusion and Random placements. The reason lies on the

fact that Biased- Random placement provides redundancy of nodes at the places where congestion is more likely to happen (sources and sinks), while Grid placement is dense and structured all over the network. On the other hand, Simple Diffusion provides redundancy only near sink.

The next results concern average delay and total energy consumption.

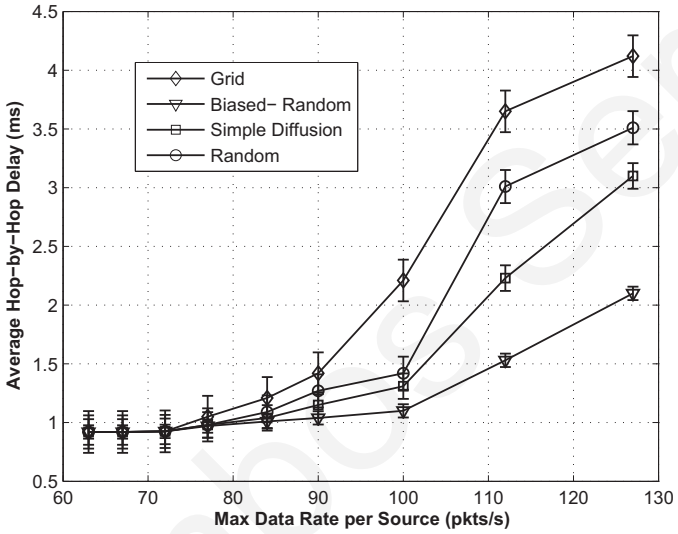

Figure 74: DAlPaS: Average Delay.

In these figures we notice that the attitude of DAlPaS algorithm is similar to HTAP. Although DAlPaS algorithm, under Grid placement can provide higher throughout in comparison with HTAP, the fact that it creates alternative paths using nodes that are distance away, forces these nodes to transmit with higher power. Due to this fact, their power is consumed faster, while the interference that it is caused, contributes to more energy consumption. As a result the delay also increases. On the other hand, biased- random placement can provide shorter paths near hot spots. For this reason, under this placement, it presents better results in terms of energy consumption and delay. Grid placement provides even worst results than Simple Diffusion and Random

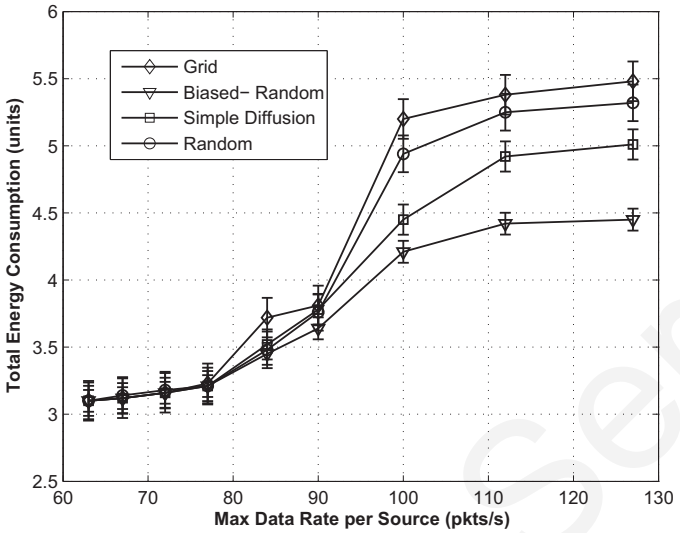

Figure 75: DAlPaS: Total Energy Consumption.

placements. These placements may not be able to provide dense and consistent placements but they can provide shorter routes.

The results are reinforced by the last figure, that represents the percentage of network's remaining energy after the network stalls. It is clear that DAlPaS algorithm under Biased Random presents the best performance and manages to utilize network's resources more efficiently, in comparison with the other placements and especially under Grid Placement.

7.4 Concluding Remarks

In this Chapter we study the performance of HTAP and DAlPaS algorithm under different placements. For this purpose we employ a deterministic placement (Grid), a semi-deterministic (Biased Random), and two non-deterministic (Simple Diffusion and Random). Simulation results show that HTAP is favored in all parameters when it runs under Biased Random placement. This placement can provide multiple alternative paths near source and near sink which can facilitate the

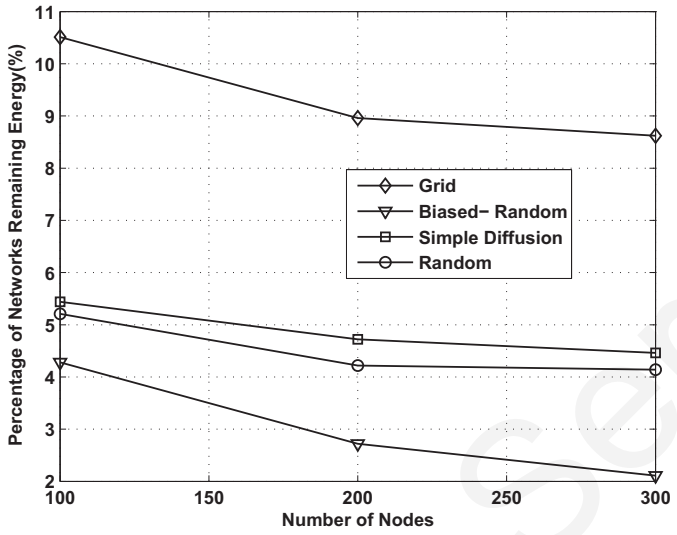

Figure 76: DAlPaS: Network's Remaining Energy (%).

creation of alternative paths. On the other hand, it presents the worst results under Grid Placement. This placement provides only three possible paths which are not enough for the efficient operation of HTAP.

Concerning the DAlPaS algorithm the situation is different. DAlPaS is favored in terms of throughput, when it runs under dense and consistent placements like Biased Random and Grid. Nodes in DAlPaS maintain in their neighbor table all nodes that can connect with. Thus, in case of congestion or even when there is need for the application of DAlPaS soft stage mechanism, they can seek in this table and find alternative nodes. However, this fact is possible to create longer routes in placements like Grid. Such placements are not able to provide short paths near hot-spots. For this reason, placements like Grid present higher delay and higher energy consumption.

Chapter 8

Comparison of HTAP and DAlPaS against Different Types of Congestion Control Algorithms in WSNs

In this Chapter we perform a comparison of HTAP and DAlPaS, against algorithms that represent different types of congestion control and avoidance algorithms in WSNs. In particular, we compare HTAP and DAlPaS against SenTCP [39] which is a congestion control algorithm that employs a traffic control method, against ESRT [57] which is a Reliable Data Transport algorithm that also employs a traffic control method, as well as against Directed Diffusion [24]. Directed Diffusion is not a dedicated congestion control algorithm but it employs multiple paths for the transmission of data from sources to sinks, fact that facilitates the avoidance of hotspots in the network. Finally, and for comparison purposes, we also employ two generic schemes. The plain flooding and the case where no congestion control exists in the network.

This Chapter is divided in two parts. In the first part we present an energy efficiency analysis focusing on the comparison between algorithms that employ resource and algorithms that employ traffic control. In the same part we also present results that concern the performance of the algorithms, in terms of energy conservation and the lifetime of the network, while in the second

part we present results related to to the successful operation of the algorithms (i.e. throughput, successfully received packets e.t.c).

8.1 Energy Efficiency Analysis and Simulation Results

We consider an event-driven hierarchical network where all nodes forward their packets upstream to the sink. Nodes are deployed on the crossed lines of a grid and each node transmits packets only one hop away. In this network there is only one sink. Each node has a sensing range of R meters and a communication range of 2R meters. We also consider the occurrence of a permanent event in the network which is captured by the nodes which are in R meters distance from the event. These nodes are becoming source nodes and attempt to forward their packets to the sink.

It is known from [87] that the energy consumed when sending a packet of m bits over a one hop wireless link of distance d, can be expressed as:

$$E_h(m,d) = \{E_T(m,d) + P_T T_{st} + E_{encode}\} + \{E_R(m) + P_R T_{st} + E_{decode}\} \quad (31)$$

where

E_T = energy used by the transmitter circuitry and power amplifier

E_R = energy used by the receiver circuitry

P_T = power consumption of the transmitter circuitry

P_R = power consumption of the receiver circuitry

T_{st} = startup time of the transceiver

E_{encode} = energy used to encode

E_{decode} = energy used to decode

If we consider that there is a path of n nodes between a source node and the sink, the energy consumed for transmitting this packet over this path is

$$E_{path} = E_T(m,d) + \left\{ \sum_{i=2}^{n} E_T(m,d) + E_R(m) \right\} + \beta, \tag{32}$$

where β is the sum of P_T, P_R, T_{st}, E_{encode} and E_{decode} for each node. Using these equations we can claim that each node that is not going to become a source node during its lifetime in WSNs, is able to relay a finite maximum number of packets. The number of packets is strictly related to its initial energy and the distance d of its next-hop neighbor.

At this time we define two more variables. Sensor Lifetime SL is the total time that a node remains alive in the network and it depends on its energy consumption per unit time. Thus,

$$SL = \frac{E_0}{E_t}, \tag{33}$$

where E_0 is the initial energy of the node and E_t the energy consumption per time unit. Energy consumption per time unit is strictly depended on the rate of packets that it receives and transmits and the distance over which it transmits them. On the other hand, Network Lifetime NL is the total time that elapses until the network is not able to transmit packets from the source(s) to the sink(s) anymore.

Without loss of generality we consider a grid placement where all nodes are equally spaced and each node transmits upstream (to sink direction) only one hop away. Since the distance d is constant and the size of packet is also the same (m), we can safely denote that the only major variable factor that affects the lifetime of a relay node, is the number of packets that it transmits and receives.

For our analysis we consider that the network, upon its configuration, is facing an event that creates heavy data load. If no congestion control algorithm is applied we can calculate from (31) and (32) that the network's lifetime is very limited. If we consider that each node has a buffer capacity of C packets of m bits each, and each node is receiving packets with a higher rate than it is capable of transmitting, then this buffer will soon become full. The furthest away from the source this node is, the more energy is going to be wasted (network-wide) when a packet drop occurs, according to (32).

Therefore, it is evident that congestion control algorithms need to be involved. As we discussed before, currently there are two types of congestion control algorithms: Traffic Control and Resource Control. Studying these two methods, from the energy consumption perspective, we can deduce the following: Algorithms that employ the traffic control method, although they reduce the load in the network in order to avoid congestion they do not alter the routing path and keep transmitting packets from the same path. In such case, the energy consumption per time unit (E_t) of these nodes is increasing and the sensor node lifetime SL is reducing. Thus, if we consider that each node in this path is relaying p packets/s, then combining (31) and (33) we can derive that each node in this path consumes $p \times E_h(m, d)$ Joule/s. As time is increasing, E_t is increasing as well and according to (33) SL reduces. Since this situation is the same for the whole path, it is expected that in finite time t, which is heavily depended on the duration that this path is continually used, the node will be power exhausted. When nodes are power exhausted, the network's lifetime is decreasing even if the energy of the nodes in the rest of the network remains unaltered.

When the resource control method is employed the situation is different. In this case, when a node becomes a hotspot (buffer or medium congested) the excess packets are routed through other nodes. In addition, since algorithms that employ this method always start using the shortest path, it means that in the case of congestion, extra packets are routed through longer routes. The equation

which is actually affected in this method is (32). In this case, parameter n, which is the number of nodes that form the path, is increasing, leading to higher energy consumption. If we consider that algorithms that employ this method, always employ a topology control scheme [8,30], the number of alternative paths increase but the nodes that implement these paths are limited in order to avoid long routes. This means that several paths can be used, leading to balanced energy consumption. Therefore, the network is exhausting its energy uniformly and its Network Lifetime increase.

If we compare the two methods after an intensive congestion situation of duration τ it is possible to have the following relationship between traffic control (tc) and resource control (rc) concerning their total remaining energy :

$$\sum_{i=1}^{m} E_{tc} \geq \sum_{i=1}^{m} E_{rc} \text{ and } NL_{tc} < NL_{rc} \tag{34}$$

where m is the initial number of nodes in the network. This means that it is possible the sum of nodes remaining power to be bigger when traffic control is employed compared to the resource control method, but at the same time the network's lifetime to be smaller. It is clear that the maximum lifetime can be achieved when nodes are exhausting their power uniformly.

8.1.1 Scenario Analysis and Results

To study the performance of the algorithms in terms of network lifetime and energy utilization a series of simulations has been conducted using Prowler network simulator [72] with the simulation settings and parameters of section 3.4 of Chapter 3.

The first parameter that we examine in order to evaluate the energy utilization of the network is NL, the Network Lifetime. What we actually count is the number of power exhausted nodes at the moment that the network is not able to transmit even a single packet from the source to the sink. This metric is a strong indication of whether the network has uniformly utilized its resources or if

it has exhausted a number of them in specific parts creating “holes” in the network. Results have been taken from two different simulation series for each algorithm (200 nodes and 300 nodes) and they are presented in Fig. 77.

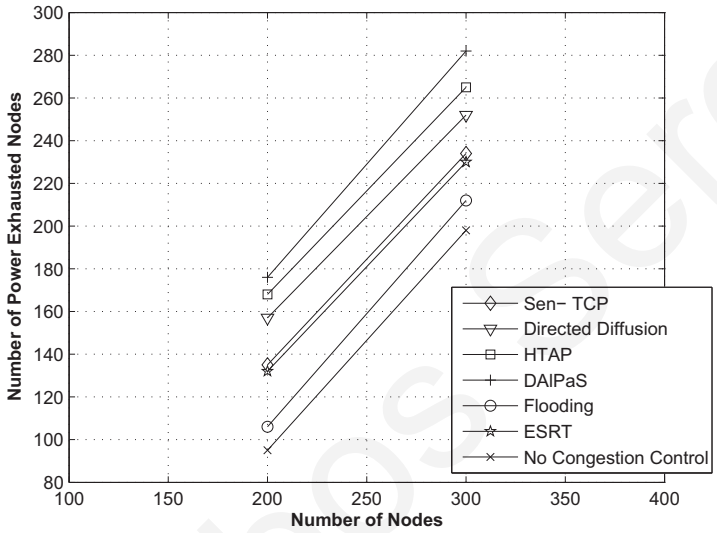

Figure 77: Number of Power Exhausted Nodes.

It is clear that in case that nodes do not run any congestion control algorithm (No Congestion Control), the network is going to exhaust all nodes around the source and soon will not be able to establish a path between the source and the sink. In the case of 200 nodes, although more than half of the nodes are still alive, the network is inoperable. In the case of 300 nodes when two thirds of the nodes are exhausted the network is again inoperable. Similar results to the No Congestion Control are depicted when Flooding is applied. In this case although the network’s resources are more uniformly utilized, since every packet is replicated and send all over the network, the congestion problem again exhausts all nodes around the source and the same problem as before arises. On the other hand, Sen-TCP and ESRT present similar performance between them and a much better performance compared to no congestion control and flooding. These solutions are

not without problems either. These algorithms always try to find the shortest path from the source to the sink. In the case that congestion is persistent, all direct paths from the source to the sink will still be exhausted and "network holes" will be created in the network. It is notable that in sum, almost 25% of network's power remains unused while the network is stalled. In Fig. 77 we observe that although these algorithms present (as expected) much better performance in comparison with plain flooding and no congestion control they still have an important number of nodes unused. Finally, DAlPaS, HTAP, and Directed Diffusion, algorithms that employ multiple paths to transmit data to the sink in case of congestion, present the best attitude concerning the number of alive nodes. Especially HTAP and DAlPaS, which are dedicated resource congestion control algorithms, manage to utilize uniformly almost all network nodes. This is a strong indication that using resource control can significantly increase the lifetime of WSNs.

To reinforce the indications we received from Fig. 77, we also studied the percentage of Network's Remaining Energy after the network stalls. This metric, indicates the average remaining energy of all alive nodes. The results are presented in Fig. 78.

Furthermore, we consider the duration that the network remains operational. Studying Fig. 79, we notice that when resource control is employed (DAlPaS and HTAP) the network remains more time operational, until it becomes disconnected. This fact, confirms the results of the previous figures, which indicate that the network lifetime extents when resource congestion control are involved. On the other hand, we notice that SenTCP and ESRT present similar performance, which is worse than the performance of resource control algorithms. Flooding and no congestion control, as it is expected, present the worst performance.

Finally, we also count the average energy consumption per delivered packet. Actually, in this figure we count the number of successfully delivered packets by the sink, and we divide this

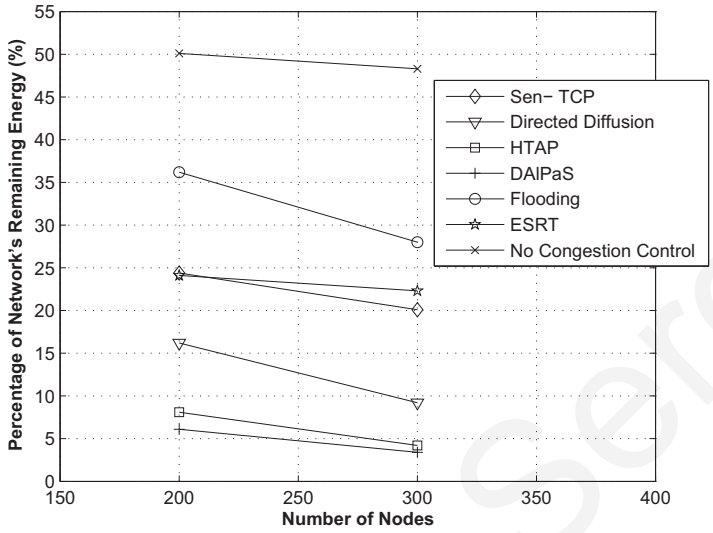

Figure 78: Percentage of Network's Remaining Energy.

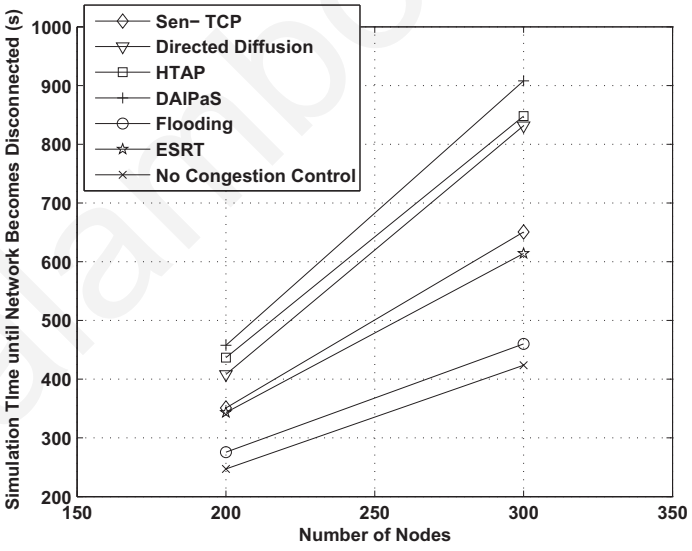

Figure 79: Simulation Time until Network Becomes Disconnected.

number by the total power exhausted by the network. This figure gives us a strong indication about the ability of each algorithm to manage efficiently its power resources.

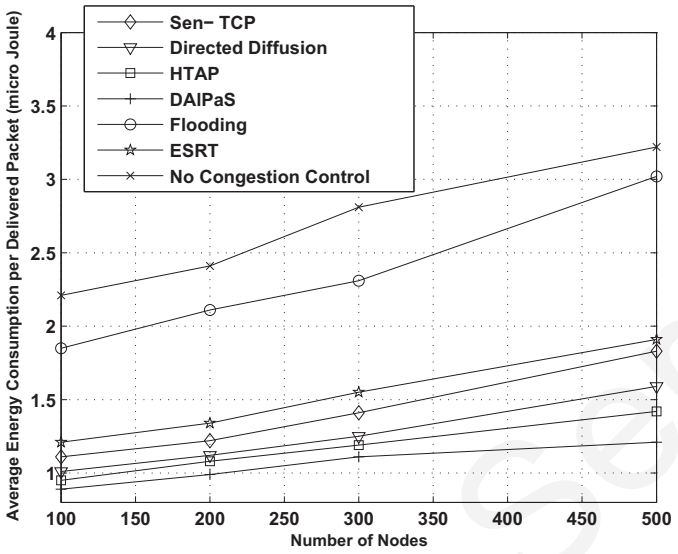

Figure 80: Average Energy Consumption per Delivered Packet.

As Fig. 80 indicates, HTAP and DAIPaS present the best performance. This result shows, that although the nodes, in algorithms that employ a resource control method, spend more energy for the transmission of a single packet from source to sink, the overall number of packets that they deliver to sink during their whole lifetime, "cost" less than in comparison with traffic control algorithms. This results is another strong indication, of the efficient operation of resource control algorithms in cases where high data load exists.

8.2 Results that Concern the Successful Operation of the Algorithms

In this section we present results that concern the successful operation of the algorithms. The first metric that we evaluate is "average packet drops" (Fig. 81). Packet drops is one of the most significant events in terms of congestion control and it clearly indicates a problem in the network, while it consumes significant amount power.

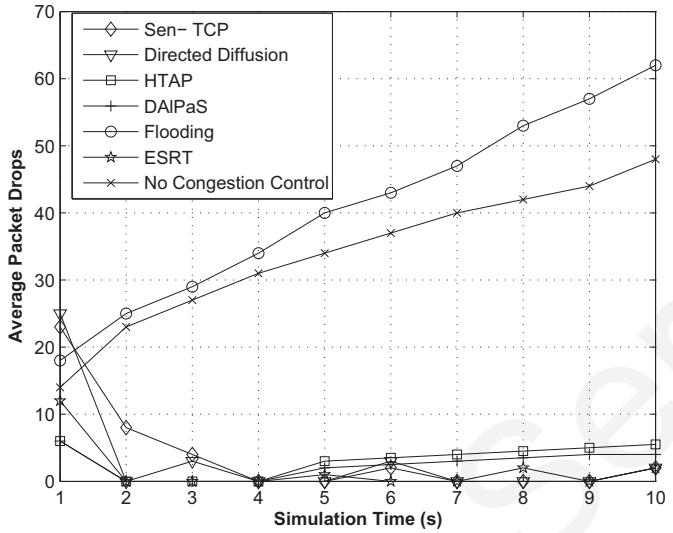

Figure 81: Average Packet Drops.

The results indicate that all five congestion control algorithms are able to handle congestion and minimize packet drops. This is a strong indication that the power they consume is charged to their routing attitude and not to packets drops. This fact is another clue that resource control algorithms are able to handle efficiently the nodes' limited power and increase significantly the network's lifetime. On the other hand, algorithms that do not apply congestion control techniques lose packets exponentially and soon the network is getting power exhausted.

The next parameter that we examine, is the actual number of packets that is being received by the sinks when we vary the peak data rate of the sources.

Studying Fig. 82 we notice that resource control algorithms (HTAP and DAlPaS) are able to deliver much higher number of packets to the sink, in comparison with traffic control algorithms, especially when the data rate of the sources is increasing.

This result is reinforced by the next figure, that represents the sink throughput under the same conditions.

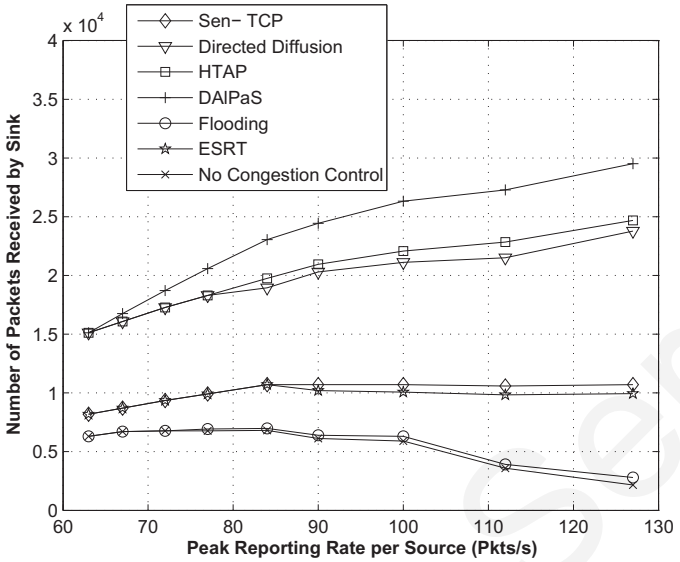

Figure 82: Number of Packets Received by the Sink.

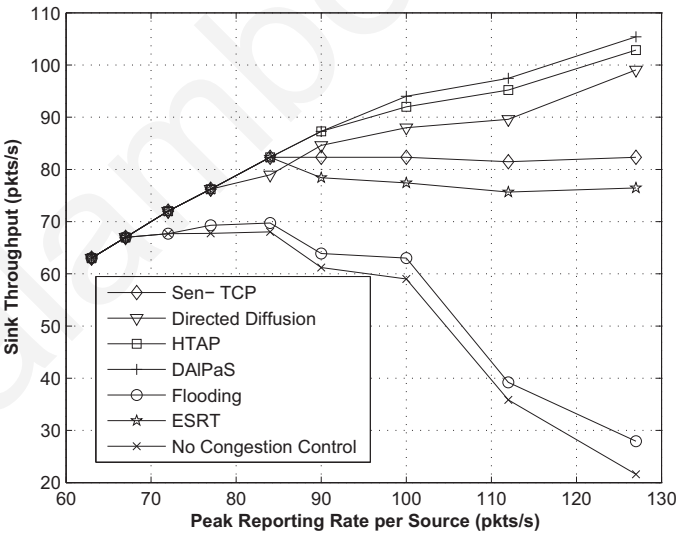

Figure 83: Sink Throughput.

According to Fig. 83, resource control algorithms maintain higher sink throughput which increase as sources data rate increase. On the other hand traffic control algorithms, after a specific point where the source data rate is enough to cause congestion, these algorithms reduce source

data rate and maintain it at a point where they can keep their operation without congestion. As a result, sink throughput remains stable and does not increase as it happens with traffic control algorithms.

Finally, we also present the average delay for the end-to-end transmission of a packet from source to sink.

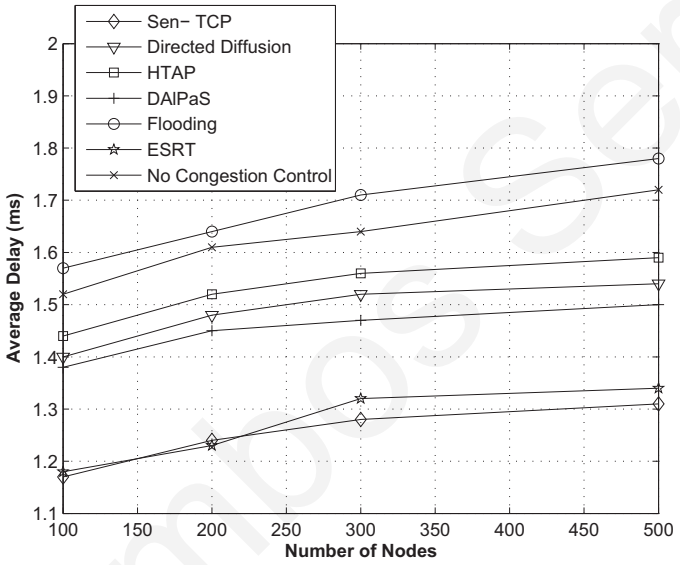

Figure 84: Average Delay.

As it is expected, resource control algorithms experience bigger delays in comparison with traffic control algorithms. The reason lies on the fact, that a packet traverses between a bigger number of hops to reach the sink, and delay increases. But this graph also confirms the fact that due to multipath routing the network is uniformly utilized since many hops take part in the path from source to sink.

8.3 Combination of Resource and Traffic Control

An interesting discussion at this point would be the combination of traffic and resource control methods in a hybrid algorithm that would combine the benefits of the two methods, while trying to alleviate their disadvantages.

As we already mention in Chapter 2 there already exists an effort in literature, with promising results, that attempts to do so. This is CADA (Congestion Avoidance Detection and Alleviation) [45]. In this work, authors suggest a congestion mitigation approach that employs both resource control and traffic control method depending on the case. If congestion takes place in an intersection hotspot, then resource control applies, while if congestion takes place in a convergence hotspot, traffic control applies.

Although this combination of traffic and resource control is promising, we believe that it will also be interesting to see whether these methods could combine in a different way. In particular, as we show in this thesis, resource control is very efficient for applications where permanent high load is expected, while traffic control is efficient for transient cases. The challenge here is the development of a lightweight and robust algorithm which can efficiently decide when to apply the resource and when the traffic control method. This point is critical, since wrong judgement can lead to fatal results. Specifically, this algorithm should be able, using an adaptive method that would combine the data load in the network and the duration, to decide which is the proper timing to shift between the resource and traffic control method. Also, it should be able to understand the cases where the data load exceeds the capacity of the network. By calculating this parameter, it will be able to apply traffic control, in the cases where the network, under the resource control method, starts losing packets due to the unavailability of redundant paths.

8.4 Concluding Remarks

In this chapter we compare the performance of HTAP and DAlPaS with a number of Congestion Control algorithms that employ different method than resource control. Two of them (ESRT and Sen-TCP) mitigate congestion through the reduction of the source's data rate (traffic control), while the other one (Directed Diffusion) attempts to balance the network load by splitting the traffic among several nodes which are not in the initial shortest path from source to sink, creating multiple paths. Also, we study the case when no congestion control algorithm applies in the network or when a generic routing algorithm (flooding) applies. The simulation results depicted that resource control algorithms can significantly increase the network lifetime since they uniformly utilize network resources. This fact leads to balanced energy consumption by almost all nodes in the network and creation of routing "holes" is avoided. Balanced energy consumption is very important for low powered sensor nodes. On the other hand the case of not using a congestion control algorithm in the network has been permanently excluded, since, according to our results, it will certainly become catastrophic for the network.

The strong conclusion that we extract from this research work is that in cases where a heavy injection of data packets is expected (e.g event-based networks) or when congestion is persistent, algorithms that employ resource control outbalance algorithms that employ traffic control and their use will significantly assist the network to increase its lifetime due to uniform energy utilization. On the other hand in periodic reported events, where a huge amount of data is never expected, algorithms that employ traffic control could be more efficient since they maintain good reporting delays.

Chapter 9

Conclusions

In this thesis we study the congestion control problem in WSNs, focusing on the development of performance-aware congestion control algorithms. The main scope of this study is to research whether, all data packets, created by sources during crisis state can be forwarded to sinks without reducing the data rate of the sources and whether under these conditions the performance of network in terms of delay, energy consumption and throughput can be maintained.

For this purpose, two novel congestion control algorithms have been proposed. The first is Hierarchical Tree Alternative Path (HTAP) algorithm and the second is Dynamic Alternative Path Selection (DAlPaS) algorithm.

Initially, in this thesis we have presented a thorough survey of the great majority of congestion control and avoidance as well as reliable transport algorithms in WSNs. Then, we performed a mathematical analysis stating the issues that traffic bottlenecks can create in WSNs and how traffic and resource control algorithms react to this problem. Then, we studied the importance of topology control algorithms in WSNs, and presented two different topology control algorithms. Source based trees and sink based trees. Afterwards, we provided a detailed description of the two proposed algorithms, HTAP and DAlPaS providing extensive simulation results.

In what follows, we first give an overview of the contributions and findings of this thesis, then discuss possible future research directions, and finally give a closing statement

9.1 Contributions and Findings of this Thesis

Initially in this thesis we prove, using fluid dynamics analysis, that traffic bottlenecks is a major issue in WSNs and that queue formation starts very early when bottlenecks appear in the network. Thus, in order to avoid queue formation the data rate of the incoming flow must be severely reduced. In case that this action is omitted, buffer fill-up is going to happen, while the time that passes until the nodes' buffers fill-up depends on the difference between incoming and outgoing flows. In order to solve this issue congestion control algorithms needs to be applied. These algorithms can be based either on traffic or resource control. According to the results of this thesis, the traffic control method is an effective method for transient congestion occurrences but can be proven unappropriate when application needs all data to be transferred to sink. For this reason we base our proposed algorithms to the resource control method, a method that has not attracted a lot of interest due to the overhead that it creates. In this thesis we addressed all possible problems that resource control methods may face and presented two novel algorithms: HTAP and DAlPaS.

Before presenting the two algorithms, we study a number of issues that were raised in relation with topology control algorithms. Specifically, we studied whether source-based trees can provide an effective topology control solution as sink-based trees do. The results show that source based can provide a proper solution under specific circumstances and for specific applications.

Source based trees constituted the base for the first novel algorithm that we presented. Hierarchical Tree Alternative Path (HTAP) algorithm, is an algorithm that attempts to route the excess

packets in case of congestion through alternative paths. HTAP is based on a source based hierarchical tree, which is created when congestion occurs. HTAP infers congestion through a novel adaptive method and dynamically takes decision in order to alternate path. HTAP algorithm has been extensively tested and simulation results showed that it presents increased performance to the range of 2-4% in comparison with TARA [30] which is an established resource control algorithm.

The only minor issue that we noticed with HTAP was the fact that under specific scenarios it presents some delay. In order to correct this issue we presented a second algorithm, called DAlPaS (Dynamic Alternative Path Selection). DAlPaS algorithm is based on a sink-tree and presents a novel feature. DAlPaS acts probatively and attempts to balance the traffic between all nodes on the tree using soft stage and hard stage along with flag decision algorithm. Simulation results showed that DAlPaS is even more effective algorithm than HTAP and TARA.

9.2 Future Work

As a future work for this thesis, it would be interesting to study the feasibility of the implementation of a hybrid algorithm that can combine the advantages and avoid the disadvantages of traffic and resource control method. A discussion that drives the guidelines for such an algorithm is performed in Chapter 8.

Moreover we could study whether other parameters besides hop count can serve as the basis for the topology control algorithms and how this issue can affect the performance of HTAP and DAlPaS. For example, nodes can be placed in levels from source to sink or vice versa, based on a combination of hop count and RSSI value, finding a proper balance point between them.

Concerning the algorithms themselves it would be interesting to study whether mobile nodes can appear to the point where alternative paths are needed or even better mobile sinks. Mobile nodes solution could be become very effective in congestion control problems

Finally, it would be notable to study whether these algorithms can apply in other context of WSNs. For example, it would be interesting to study whether instead of congestion, security problems can be avoided using these algorithms. In this case the metric for alternative path creation could be the security threat.

9.3 Closing Remark

In this thesis we show that resource control method can be proven as a very effective method for facing congestion in WSNs. The resource control method is a method that has been neglected in comparison with the traffic control method due to its complexity. In this thesis we show that if carefully tuned, it can provide very good results in terms of throughput while it maintains important performance parameters. HTAP and DAlPaS are two good examples of algorithms that employ resource control.

Bibliography

[1] C. Sergiou and V. Vassiliou, "Source-based Routing Trees for Efficient Congestion Control in Wireless Sensor Networks," in *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), (PWSN Workshop)*, May 2012, pp. 378– 383.

[2] ——, "Study of Lifetime Extension in Wireless Sensor Networks through Congestion Control Algorithms," in *IEEE Symposium on Computers and Communications (ISCC)*, July 2011, pp. 283 –286.

[3] ——, "Performance Evaluation of the DAlPaS Congestion Control Algorithm in Wireless Sensor Networks," in *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, June 2011, pp. 1 –7.

[4] ——, "DAlPaS: A Performance Aware Congestion Control Algorithm in Wireless Sensor Networks," in *18th International Conference on Telecommunications (ICT)*, May 2011, pp. 167 –173.

[5] M. Koutroullos, C. Sergiou, and V. Vassiliou, "Mobile-CC: Introducing Mobility to WSNs for Congestion Mitigation in Heavily Congested Areas," in *Telecommunications (ICT), 2011 18th International Conference on*, May 2011, pp. 400 –405.

[6] C. Sergiou and V. Vassiliou, "Energy Utilization of HTAP under Specific Node Placements in Wireless Sensor Networks," in *European Wireless Conference (EW)*, April 2010, pp. 482 –487.

[7] V. Vassiliou and C. Sergiou, "Performance Study of Node Placement for Congestion Control in Wireless Sensor Networks," in *3rd International Conference on New Technologies, Mobility and Security (NTMS)*, Dec. 2009, pp. 1 –8.

[8] C. Sergiou, V. Vassiliou, and A. Pitsillides, "Reliable Data Transmission in Event-Based Sensor Networks During Overload Situation," in *WICON '07: Proceedings of the 3rd International Conference on Wireless Internet*, Oct. 2007, pp. 1–8.

[9] C. Sergiou, M. Koutroullos, and V. Vassiliou, "Poster Abstract: A Congestion Mitigation Approach using Mobile Nodes in Wireless Sensor Networks," in *10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2011, pp. 161 –162.

[10] C. Sergiou and V. Vassiliou, "Poster Abstract: Energy Hole Prevention in Wireless Sensor Networks," in *9th ACM/IEEE International Conference on Information Processing in Sensor Networks (Poster Session)*, 2010, pp. 398–399.

[11] ——, "Poster Abstract: Alternative Path Creation vs Data Rate Reduction for Congestion Mitigation in Wireless Sensor Networks," in *9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010, pp. 394–395.

[12] K. Romer and F. Mattern, "The Design Space of Wireless Sensor Networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, Dec. 2004.

[13] T. Haenselmann, *Sensornetworks*. GFDL Wireless Sensor Network Textbook, April 2005.

[14] A. Cerpa, J. Elson, M. Hamilton, J. Zhao, D. Estrin, and L. Girod, "Habitat Monitoring: Application Driver for Wireless Communications Technology," in *SIGCOMM '01: Workshop on Data Communication in Latin America and the Caribbean*, 2001, pp. 20–41.

[15] E. Biagioni and K. Bridges, "The Applications of Remote Sensor Technology to Assist the Recovery of Rare and Endangered Species," *International Journal of High Performance Computing Applications, Special Issue on Distributed Sensor Networks*, vol. 16, no. 3, pp. 315–324, Aug. 2002.

[16] L. Schwiebert, S. K. Gupta, and J. Weinmann, "Research Challenges in Wireless Networks of Biomedical Sensors," in *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001, pp. 151–165.

[17] H. T. Kung and D. Vlah, "Efficient Location Tracking Using Sensor Networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, March 2003, pp. 1954–1962.

[18] R. Brooks, P. Ramanathan, and A. M. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1163–1171, 2003.

[19] C. Chien, I. Elgorriaga, and C. McConaghy, "Low-Power Direct-Sequence Spread-Spectrum Modem Architecture for Distributed Wireless Sensor Networks," in *International Symposium on Low Power Electronics and Design*, 2001, pp. 251–254.

[20] R. J. Cramer, M. Z. Win, and R. A. Scholtz, "Impulse Radio Multipath Characteristics and Diversity Reception," in *IEEE International Conference on Communications (ICC)*, vol. 3, 1998, pp. 1650– 1654.

[21] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Networks," in *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001, pp. 272–287.

[22] A. Woo and D. E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," in *Proceedings of the 7th annual International Conference on Mobile Computing and Networking (MobiCom'01)*, 2001, pp. 221–235.

[23] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," in *International Conference on Distributed Computing Systems*. IEEE Computer Society, 2003, pp. 46–55.

[24] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed Diffusion for Wireless Sensor Networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[25] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom)*, 1999, pp. 174–185. [Online]. Available: http://dx.doi.org/10.1145/313451.313529

[26] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestmant, "A Survey of Gossiping and Broadcasting in Communication Networks," *Networks: An International Journal*, vol. 18, pp. 319–349, 1988.

[27] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks," Charlottesville, VA,

USA, Tech. Rep., 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=900537

[28] C. Wang, M. Daneshmand, B. Li, and K. Sohraby, "A Survey of Transport Protocols for Wireless Sensor Networks," *IEEE Network, May/June*, vol. 20, pp. 34–40, 2006.

[29] G. Srinivasan and S. Murugappan, "A Survey of Congestion Control Techniques in Wireless Sensor Networks," *International Journal of Information Technology and Knowledge Management*, vol. 4, no. 2, pp. 413–415, July/December 2011.

[30] J. Kang, Y. Zhang, and B. Nath, "TARA: Topology-Aware Resource Adaptation to Alleviate Congestion in Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 919–931, 2007.

[31] J. Zhao, L. Wang, S. Li, X. Liu, Z. Yuan, and Z. Gao, "A Survey of Congestion Control Mechanisms in Wireless Sensor Networks," in *Proceedings of the 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, ser. IIH-MSP '10, 2010, pp. 719–722.

[32] R. Chakravarthi, C. Gomathy, S. Sebastian, K. Pushparaj, and V. B. Mon, "A Survey on Congestion Control in Wireless Sensor Networks," *International Journal of Computer Science and Communication*, vol. 1, no. 1, pp. 161–164, January-June 2010.

[33] C. Wang, K. Sohraby, B. Li, and W. Tang, "Issues of Transport Control Protocols for Wireless Sensor Networks," in *International Conference on Communications, Circuits and Systems (ICCCAS)*, May 2005, pp. 422– 426.

[34] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," in *SenSys '03: Proceedings of the 1st international*

Conference on Embedded Networked Sensor Systems, 2003, pp. 266–279. [Online]. Available: http://portal.acm.org/citation.cfm?id=958523

[35] C. T. Ee and R. Bajcsy, "Congestion Control and Fairness for Many-to-One Routing in Sensor Networks," in *SenSys '04: Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems*, 2004, pp. 148–161.

[36] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks," in *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004, pp. 134–147.

[37] R. Vedantham, R. Sivakumar, and S.-J. Park, "Sink-to-sensors congestion control," *Ad Hoc Networks*, vol. 5, no. 4, pp. 462 – 485, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B7576-4JM0T76-1/2/14742f9f889d550dc3a041349294e3d5

[38] K. Karenos, V. Kalogeraki, and S. V. Krishnamurthy, "Cluster-Based Congestion Control for Supporting Multiple Classes of Traffic in Sensor Networks," in *EmNets '05: Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, 2005, pp. 107–114.

[39] C. Wang, K. Sohraby, and B. Li, "SenTCP: A Hop-by-Hop Congestion Control Protocol for Wireless Sensor Networks," *IEEE INFOCOM (Poster Paper)*, March 2005.

[40] L. Popa, C. Raiciu, I. Stoica, and D. S. Rosenblum, "Reducing Congestion Effects in Wireless Networks by Multipath Routing," in *ICNP*. IEEE Computer Society, 2006, pp. 96–105. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICNP.2006.320202

[41] C. Wang, K. Sohraby, V. Lawrence, B. Li, and Y. Hu, "Priority-based congestion control in wireless sensor networks," *International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 1, pp. 22–31, 2006.

[42] G. Wang and K. Liu, "Upstream Hop-by-Hop Congestion Control in Wireless Sensor Networks," in *IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2009, pp. 1406 –1410.

[43] X. Yin, X. Zhou, R. Huang, Y. Fang, and S. Li, "A Fairness-Aware Congestion Control Scheme in Wireless Sensor Networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 5225 –5234, November 2009.

[44] P. Antoniou, A. Pitsillides, T. Blackwell, and A. Engelbrecht, "Employing the Flocking Behavior of Birds for Controlling Congestion in Autonomous Decentralized Networks," in *2009 IEEE Congress on Evolutionary Computation*, A. Tyrrell, Ed., Trondheim, Norway, May 2009, pp. 1753 – 1761.

[45] W.-w. Fang, J.-m. Chen, L. Shu, T.-s. Chu, and D.-p. Qian, "Congestion Avoidance, Detection and Alleviation in Wireless Sensor Networks," *Journal of Zhejiang University - Science C*, vol. 11, pp. 63–73, 2010, 10.1631/jzus.C0910204. [Online]. Available: http://dx.doi.org/10.1631/jzus.C0910204

[46] P. Antoniou and A. Pitsillides, "A Bio-Inspired Approach for Streaming Applications in Wireless Sensor Networks Based on the Lotka-Volterra Competition Model," *Comput. Commun.*, vol. 33, pp. 2039–2047, November 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2010.07.020

[47] C.-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft, "Siphon: Overload Traffic Management using Multi-Radio Virtual Sinks in Sensor Networks," in *SenSys '05: Proceedings of the 3rd international Conference on Embedded Networked Sensor Systems*, 2005, pp. 116–129.

[48] S. Chen and N. Yang, "Congestion Avoidance Based on Lightweight Buffer Management in Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 9, pp. 934–946, 2006.

[49] K. Karenos and V. Kalogeraki, "Facilitating Congestion Avoidance in Sensor Networks with a Mobile Sink," in *IEEE Real Time Systems Symposium (RTSS)*, 2007, pp. 321–332.

[50] M. M. Alam and C. S. Hong, "Buffer and Rate Control Based Congestion Avoidance in Wireless Sensor Networks," in *KIPS*, May 2007, pp. 1291–1293.

[51] M. I. Khan, W. N. Gansterer, and G. Haring, "Congestion Avoidance and Energy Efficient Routing Protocol for Wireless Sensor Networks with a Mobile Sink," *JNW*, vol. 2, no. 6, pp. 42–49, 2007.

[52] T. He, F. Ren, C. Lin, and S. Das, "Alleviating Congestion Using Traffic-Aware Dynamic Routing in Wireless Sensor Networks," in *SECON*. IEEE, 2008, pp. 233–241.

[53] Y.-P. Hsu and K.-T. Feng, "Cross-layer Routing for Congestion Control in Wireless Sensor Networks," in *Radio and Wireless Symposium, 2008 IEEE*, 2008, pp. 783 –786.

[54] G. Rajsekar, M. Mathew, N. Dineshraj, S. Barath, M. Sudha, and M. Valarmathi, "Collision Avoidance Scheme in Energy Constrained Wireless Sensor Networks using MAC Protocol," in *International Conference on Computing, Communication and Networking (ICCCN)*, 2008, pp. 1 –4.

[55] J.-M. Huang, C.-Y. Li, and K.-H. Chen, "TALONet: A Power-Efficient Grid-Based Congestion Avoidance Scheme Using Multi-Detouring Technique in Wireless Sensor Networks," in *Wireless Telecommunications Symposium*, 2009, pp. 1 –6.

[56] S. Misra, V. Tiwari, and M. Obaidat, "LACAS: Learning Automata-Based Congestion Avoidance Scheme for Healthcare Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 4, pp. 466 –479, May 2009.

[57] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks," in *4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, 2003, pp. 177–188.

[58] Y. Iyer, S. Gandham, and S. Venkatesan, "STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks," in *14th International Conference on Computer Communications and Networks (ICCCN)*, 2005, pp. 449 – 454.

[59] J. Paek and R. Govindan, "RCRT: Rate-Controlled Reliable Transport for Wireless Sensor Networks," in *SenSys '07: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2007, pp. 305–319. [Online]. Available: http://dx.doi.org/10.1145/1322263.1322293

[60] Y.-M. Liu and X.-H. Jiang, "An Extended DCCP Congestion Control in Wireless Sensor Networks," in *International Workshop on Intelligent Systems and Applications*, 2009, pp. 1 –4.

[61] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "P.S.F.Q: A Reliable Transport Protocol for Wireless Sensor Networks," in *WSNA '02: Proceedings of the 1st ACM international*

workshop on Wireless Sensor Networks and Applications. New York, NY, USA: ACM Press, 2002, pp. 1–11. [Online]. Available: http://dx.doi.org/10.1145/570738.570740

[62] F. Stann and J. Heidemann, "RMST: Reliable Data Transport in Sensor Networks," in *Proceedings of the First International Workshop on Sensor Net Protocols and Applications*. Anchorage, Alaska, USA: IEEE, April 2003, pp. 102–112. [Online]. Available: http://www.isi.edu/ johnh/PAPERS/Stann03a.html

[63] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz, "A Scalable Approach for Reliable Downstream Data Delivery in Wireless Sensor Networks," in *MobiHoc '04: Proceedings of the 5th ACM International Symposium on Mobile Ad-Hoc Networking and Computing*, 2004, pp. 78–89.

[64] M. Ghalehnoie, N. Yazdani, and F. Salmasi, "Fuzzy Rate Control in Wireless Sensor Networks for Mitigating Congestion," in *Internatioal Symposium on Telecommunications*, 2008, pp. 312–317.

[65] "The Network Simulator ns-2 (v2.1b8a)," October 2001.

[66] K. Leung, W. Massey, and W. Whitt, "Traffic Models for Wireless Communication Networks," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 8, pp. 1353 –1364, Oct. 1994.

[67] M. Gribaudo, C. Chiasserini, R. Gaeta, M. Garetto, D. Manini, and M. Sereno, "A Spatial Fluid-Based Framework to Analyze Large-Scale Wireless Sensor Networks," in *International Conference on Dependable Systems and Networks*, 2005, pp. 694 – 703.

[68] S. Toumpis and L. Tassiulas, "Packetostatics: Deployment of Massively Dense Densor Networks as an Electrostatics Problem," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 4, 2005, pp. 2290 – 2301.

[69] J. Silvester and L. Kleinrock, "On the Capacity of Multihop Slotted ALOHA Networks with Regular Structure," *IEEE Transactions on Communications*, vol. 31, no. 8, pp. 974 – 982, Aug. 1983.

[70] M. Franceschetti, O. Dousse, D. Tse, and P. Tiran, "Closing the Gap in the Capacity of Random Wireless Networks," in *International Symposium on Information Theory, ISIT*, 2004, p. 438.

[71] M. Garavello and B. Piccoli, *Traffic Flow on Networks — Conservation Laws Models*. AIMS Series on Applied Mathematics, ISBN-13: 978-1-60133-000-0, 2006, vol. 1.

[72] Prowler: Probabilistic Wireless Network Simulator. [Online]. Available: http://www.isis.vanderbilt.edu/Projects/nest/prowler/

[73] M. Bala and L. Awasthi, "Performance Tradeoff with Routing Protocols for Radio Models in Wireless Sensor Networks," *Wireless Engineering and Technology*, vol. 2, no. 2, pp. 53–59, 2011.

[74] D. Gupta and A. K. Sharma, "On Performance Evaluation of WSN Routing Protocols for MICA and MICAz using Different Radio Models," *International Journal of Energy, Information and Communications*, vol. 2, no. 4, pp. 181–194, 2011.

[75] L. Xin, H. Qingfeng, and Z. Ying, "Combs, Needles, Haystacks: Balancing Push and Pull for Discovery in Large-Scale Sensor Networks," in *Proceedings of the 2nd International*

Conference on Embedded Networked Sensor Systems (SenSys '04). ACM, 2004, pp. 122–133. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031510

[76] Y. Zhang and Q. Huang, "A learning-based adaptive routing tree for wireless sensor networks," *Journal of Communications*, vol. 1, no. 2, pp. 12–21, 2006. [Online]. Available: http://ojs.academypublisher.com/index.php/jcm/article/view/01021221

[77] J. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," *IEEE Wireless Communications*, vol. 11, pp. 6–28, 2004.

[78] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *Computer*, vol. 34, pp. 57–66, August 2001. [Online]. Available: http://dx.doi.org/10.1109/2.940014

[79] N. Li, J. Hou, and L. Sha, "Design and Analysis of an MST-based Topology Control Algorithm," in *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM)*, vol. 3, March-April 2003, pp. 1702 – 1712.

[80] P. Wightman and M. Labrador, "Topology Maintenance: Extending the Lifetime of Wireless Sensor Networks," in *IEEE Latin-American Conference on Communications (LATINCOM)*, Sept. 2009, pp. 1 –6.

[81] R. C. Prim, "Shortest Connection Networks and Some Generalizations," *Bell System Technology Journal*, vol. 36, pp. 1389–1401, 1957.

[82] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 4, pp. 11–25, 2001.

[83] R. P. Mann, K. R. Namuduri, and R. Pendse, "Energy-Aware Routing Protocol for Ad Hoc Wireless Sensor Networks," *EURASIP Journal Wireless Communications Networks*, vol. 2005, no. 5, pp. 635–644, 2005.

[84] F. Ye, G. Zhong, S. Lu, and L. Zhang, "GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks," *Wireless Networks*, vol. 11, no. 3, pp. 285–298, 2005.

[85] M. Younis and K. Akkaya, "Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621–655, 2008.

[86] M. Ishizuka and M. Aida, "Performance Study of Node Placement in Sensor Networks," *ICDCSW '04: Proceedings of the 24th International Conference on Distributed Computing Systems Workshopsd*, pp. 598–603, March 2004.

[87] H. Karvonen, Z. Shelby, and C. Pomalaza-Raez, "Coding for Energy Efficient Wireless Embedded Networks," in *International Workshop in Wireless Ad-Hoc Networks,*, May- June 2004, pp. 300 – 304.

[88] C. Wang, B. Li, K. Sohraby, M. Daneshmand, and Y. Hu, "Upstream Congestion Control in Wireless Sensor Networks through Cross-Layer Optimization," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 4, pp. 786 –795, may 2007.

[89] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," in *SenSys: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, 2003, pp. 126–137. [Online]. Available: http://doi.acm.org/10.1145/958491.958506

Appendix A

Short Description of Algorithms Presented in Chapter 2

In Appendix A we present a short description of algorithms cited in Chapter 2, besides those algorithms already presented in Chapter 2, in section 2.6 (SenTCP [39], TARA [30], ESRT [57] and Directed Diffusion [24]).

A.1 Short Review of Algorithms

There are several algorithms that have been proposed in the literature, which attempt to resolve the congestion problem in WSns. In this section we present and review a representative number of them. The algorithms are classified into congestion control, congestion avoidance, and reliable data transmission algorithms. In this effort we try to match each algorithm to the category that fits better, since many algorithms fit in more than one categories. Algorithms are sorted within each category, based on the year of publication. Through this classification we attempt to present the evolution of this research field through time.

A.1.1 Congestion Control Algorithms

The first actual effort for controlling the traffic in WSNs is the ”Adaptive Rate Control” [22] algorithm. Although this algorithm does not refer directly to congestion control in can be considered as so, since it controls the network's data rate in order to guarantee fairness and subsequently to avoid overload situations.

ARC

Woo et al. proposed the Adaptive Rate Control (ARC) scheme in 2001 [22]. ARC does not involve any congestion detection or notification mechanisms. ARC uses an AIMD-like traffic control scheme to mitigate congestion, which works as follows: an intermediate node increases its sending rate by a constant a if it overhears a successful packet forwarding by its parent node. Otherwise, the intermediate node multiplies its sending rate by a factor b, where $0 < b < 1$. ARC maintains two independent sets of a and b, for source traffic and transit traffic respectively, in order to guarantee fairness. ARC was tested on a 11-node star topology, a 11-node tree topology and a real testbed. Simulations have shown that ARC is effective in achieving fairness while maintaining good aggregate bandwidth with reasonable energy efficiency, especially low traffic situations which are the common case in sensor networks. However, ARC was not compared to other related approaches.

CODA

One of the first algorithms in literature that refer directly to congestion control and avoidance in WSNs is the Congestion Detection and Avoidance algorithm (CODA) [34]. This algorithm constitutes the base for this research field and it is one of the most cited algorithms. CODA attempts to face congestion by implementing three mechanisms: Congestion detection, open-loop hop-by hop backpressure notification, and closed-loop multi-source AIMD-like traffic control to

mitigate congestion. Congestion detection is of prime importance in CODA. It is used to detect whether there is congestion in an area in the network in order to activate the rest of the mechanisms. Congestion detection is performed using the present and past channel loading conditions, along with the current buffer occupancy. Due to the high energy consumption of persistent channel listening, CODA employs a sampling scheme that activates channel monitoring when it is needed. Once congestion is detected, the nodes notify explicitly their upstream neighbor nodes via a backpressure mechanism. Backpressure signals are propagated towards the source and the nodes that receive the signals must take decisions based on their local congestion policy, in order to reduce the traffic in the network. Decisions concern rate reduction, packet drops etc. Also each node that receives a backpressure message decides whether to further propagate it, based on its congestion condition. Finally, in order to control congestion when multiple sources are sending to a single sink, CODA implements a closed-loop, multi-source AIMD-like traffic regulation. In this case when a source's event rate is higher than a pre-specified threshold, which is always a fraction of the maximum theoretical throughput of the channel, it requires an ACK packet from the sink in order to maintain its rate. In cases of lost ACK packets, the source reduces its sending rate. CODA was tested both in ns-2 [65] and in a real testbed. Simulations in ns-2 took into account both randomly generated topologies ranging from 30 to 120 nodes, while a tree-based topology was used in the real testbed. Results showed that CODA significantly improves the performance of data dissemination applications such as directed diffusion by mitigating hotspots, and reducing the energy tax with low fidelity penalty on sensing applications. However, the AIMD-like traffic regulation may not be very effective in WSNs because it results in a saw-tooth rate behavior which may violate the QoS requirements (e.g. fidelity of the reported events). Furthermore, the end-to-end nature of the closed-loop mechanism may result in reduced responsiveness causing

increased latency and high error rates, especially during long periods of congestion. Also back-pressure signals and ACK control messages consume additional energy and bandwidth. CODA was not compared against other related approaches.

Following CODA, the notion of fairness began to gain momentum. Moreover, algorithms shifted from explicit congestion notification to implicit. Also cross layer techniques appeared in literature.

CCF

Ee at al. proposed a distributed and scalable mechanism for congestion control and fairness (CCF) [35] for many-to-one routing in WSNs. CCF provides congestion detection on the basis of packets service time, and congestion mitigation through traffic control. CCF controls congestion in a hop-by-hop manner and each node uses exact rate adjustment based on its available service rate and child node number. In particular, CCF assumes a tree routing structure having the sink acting as a root and all data sources as leaves. Each sensor receives and forwards packets from its upstream neighbors; each upstream neighbor is the root of an upstream sub-tree. The sensor learns the number of data sources in each of those upstream sub-trees, measures its own down-stream forwarding rate, computes per-source fair rate, which is propagated upstream such that the data sources do not send packets beyond the rate. There are two alternative scenarios that this algorithm could be applied: (a) all nodes are generating data and routing them to the sink and (b) most nodes in the network are silent, only the nodes that detect an event generate data. Such rout-ing structures often result in the sensors closer to the base station experiencing congestion, which inevitably cause packets originating from sensors further away from the base station to have a higher probability of being dropped. The basic concept for controlling congestion consists of the following steps that repeatedly run at each sensor node: (a) Measure the average rate r at which

packets can be sent from this mote, b) divide the rate r among the number of children motes downstream n, to give the per-node data packet generation rate $rdata = r/n$, adjust the rate if queues are overflowing or about to overflow, and (c) compare the rate $rdata$ with the rate rdata.parent sent from the parent and use and propagate the smaller rate downstream. CCF was tested using both simulations (a randomly generated topology of 116 nodes, with a per-node maximum degree of 5 and a maximum network depth of 6) and actual implementation in UC Berkeley's sensor motes (a network of 10 motes). CCF was shown to achieve simple fairness, where each node was able to receive almost the same throughput. However, in applications where different sensors (e.g. geographically deployed in different places) need to gain different throughput (i.e. priority-dependent throughput), CCF will not be applicable. Also, the rate adjustment in CCF relies only on packet service time, something which could lead to low utilization when some nodes do not have enough traffic or the packet error rate is high. CODA was not compared against other related approaches.

Fusion

Hull et al. [36] proposed a scheme called 'Fusion' for mitigating congestion control in WSNs. In general, Fusion detects congestion by monitoring the queue size of each node and performing channel sampling at fixed intervals. In the presence of congestion, Fusion provides implicit congestion notification to all nodes in a radio neighborhood by setting a congestion bit in the header of every outgoing packet. Congestion is mitigated on the basis of traffic control. More specifically, Fusion combines three congestion control techniques that operate at different layers: (a) hop-by-hop flow control, (b) source rate limiting scheme, and (c) prioritized MAC. In hop-by-hop flow control, each sensor sets a congestion bit in the header of every transmitted packet. Using the broadcast characteristic of the wireless medium, every packet provides congestion feedback to all nodes in a radio neighborhood with every transmission. So there is no need for explicit control messages, which waste a large portion of the already limited bandwidth. Hop-by-hop flow control

consists of two components: congestion detection and congestion mitigation. Congestion detection is done based on the sensor's queue size. If the sensor's queue space gets beyond a specified limit, a congestion bit is set. Otherwise the congestion bit is removed. Congestion mitigation is a way to control the nodes' transmission rate in order to prevent queues at their next-hop node from overflowing. When a sensor overhear to a packet with the congestion bit set, it stops forwarding data. Otherwise, the congestion would grow bigger, and eventually the whole network will collapse. Rate limiting is a way to limit the sending rate of a sensor. Each sensor listens to the traffic its parent forwards, to estimate N, the total number of unique sources routing through the parent. A token bucket scheme is used to regulate each sensor's send rate. A sensor accumulates one token every time it hears its parent forward N packets, up to a maximum number of tokens. The sensor is allowed to send only when its token count is above zero and each transmission costs one token. In the prioritized MAC mechanism, the MAC layer provides assistance to sensors in order to react fast to congestion. A standard CSMA MAC layer is used, with a modification that implements a prioritization scheme. According to this scheme, congested nodes have higher priority compared to the other nodes. Specifically, if a sensor is congested its back-off window is the one-fourth the size of a non-congested sensor's back off window, allowing queues to drain more quickly and increasing the likelihood congestion control information will propagate throughout a sensor's neighborhood. The performance of Fusion was evaluated in an indoor testbed of 55 Crossbow Mica2 nodes using both event-based and periodic data traffic. Fusion claims to achieve good throughput and fairness at high offered loads. However, the rate adjustment used in Fusion is not smooth, something which may affect link utilization and fairness. Also the frequent use of the wireless radio for channel probing leads to energy wastage. Fusion was not compared against other related approaches.

COMUT

Karenos et al.proposed COMUT [38], a cluster-based congestion control mechanism for supporting multiple classes of traffic in WSNs. In COMUT, each cluster node detects congestion through traffic intensity estimation. This estimation is broadcasted to the cluster head which evaluates the congestion level. Congestion is mitigated on the basis of an AIMD-like traffic regulation. More specifically, COMUT consists of three different parts. Cluster formation, traffic intensity estimation, and rate regulation. In the cluster formation procedure sensors are organized into clusters. In each cluster, a cluster head called sentinel is elected. COMUT employs ZRP (Zone Routing Protocol) to assist in the formation of clusters. After the cluster is formed the level of congestion of each cluster must be estimated. To perform this estimation COMUT calculates the traffic intensity within and across multiple clusters. Due to the fact that traffic intensity is highly affected by the number of incoming and existing flows, COMUT involves a queuing network where each sensor is modeled as a queue. Once traffic intensity is calculated, congestion is controlled through AIMD-like source rate adjustment. The congested cluster through its sentinel node informs the other sentinels, and through them the source, for its condition. COMUT controls congestion through rate reduction and takes into account multiple classes of traffic. Thus, the sending rate of the low importance flow is dropped to a minimum if packets with higher importance exist along the congested path. The performance of COMUT was evaluated through ns-2 [65] simulations using randomly generated topologies of $60 - 140$ nodes. COMUT was shown be highly successful in abating congestion and in reducing wasteful packet drops, achieving energy savings. Packets from flows of high importance were delivered with extremely high fidelity. Researchers showed that congestion is controlled and all important flows can be admitted and delivered with minimum drops, achieving energy savings. COMUT was not compared against other related approaches.

Another evolution to the subject is the shift to network layer using routing techniques in order to mitigate congestion.

BGR

Popa et al. proposed Biased Geographical Routing (BGR) [40] protocol to reactively split traffic when congestion is detected. BGR detects congestion based on buffer occupancy and wireless usage, exponentially averaged to eliminate noise. Wireless usage is measured by periodically sampling wireless medium. Congestion notification is performed implicitly, on the basis of a single congestion bit added to each packet. Thus, each node that promiscuously listens to the packets sent by its neighbors, is able to detect their congested status. Congestion is mitigated using two algorithms, namely: In-Network Packet Scatter (IPS) and End-to-End Packet Scatter (EPS). IPS alleviates transient congestion by splitting traffic immediately before the congested areas. In contrast, EPS alleviates long term congestion by splitting the flow at the source, and performing rate control on the basis of the AIMD strategy. EPS selects the paths dynamically, and uses a less aggressive congestion control mechanism on non-greedy paths to improve energy efficiency. The 'bias' used in BGR determines how far the trajectory of splitting traffic will deviate from greedy route (which is always the shortest path). BGR was tested on a random topology of 400 nodes simulated in ns-2. Results showed that BGR works well for flows where the distance between the source and the destination is large enough to allow the use of non-interfering multiple paths. For short-range flows, where multiple paths could not be used, the throughput obtained by BGR is smaller with at most 14%, as the short-range flows interfere with split flows of long-range communications. However, by increasing long-range flows throughput fairness among the different flows was improved. On the other hand, it is worth noting that because the bias is randomly chosen, BGR likely makes congestion worse under some situations. In addition, BGR needs node location information provided by either GPS or other coordinate system. This overhead is non negligible.

Also, the AIMD strategy is not very effective in WSNs because it provokes a saw-tooth rate behavior which may violate the QoS requirements. In addition, AIMD-like mechanisms take a long time for data rates to converge in low-rate wireless links.

Later on, the early efforts to introduce resource control in order to mitigate congestion [30], [8] appeared. Also cross layer optimizations and fairness continue to gain momentum. TARA algorithm has already been presented in Chapter 2, in section 2.6

PCCP

Wang et al. proposed a hop-by-hop node priority-based upstream congestion control protocol for WSNs [88], [41]. PCCP refutes the congestion control protocols that argue in favor of providing equal fairness to each sensor node in a multi-hop WSN (e.g. CCF [35]) by attaching a weighted fairness to each sensor node. PCCP offers different degrees of priority indexes such that a sensor node with a higher priority index enjoys a higher bandwidth and also sensor nodes that inject more traffic get more bandwidth. PCCP further defines the priority index for both self generating traffic and transit traffic, based on which the queue length for source and the transit traffic is allocated. PCCP infers the degree of congestion through packet inter-arrival time and packet service time and then imposes hop-by-hop congestion control depending on the measured congestion degree and the priority index. PCCP uses implicit congestion notification by piggybacking the congestion information in the header of data packets, thus avoiding additional control packets. PCCP allows the application layer to dynamically override the priority index of any sensor node(s) of any particular region. This feature might be required by many applications of WSN. PCCP was tested on a small tree-based topology of 7 nodes (using both single-path and multi-path hardwired routing) and a linear topology of 10 to 40 nodes. Simulation neglected the details of MAC protocols, but assumed that MAC protocols provide even access opportunities for each neighboring node. Simulation results showed that: (1) PCCP achieves high link utilization and flexible fairness; (2)

PCCP achieves small buffer size; therefore it can avoid/reduce packet loss and therefore improve energy-efficiency, and provide lower delay. PCCP was compared against the CCF [35] for the case of single path routing. Results showed that CCF achieves lower throughput than PCCP in the interval when a node does not generate sufficient traffic. Researchers claimed that this is because CCF cannot effectively allocate the remaining system capacity and use a work-conservation scheduling algorithm. Also in the presence of packet losses, PCCP was shown to achieve much higher throughput than CCF. The reason was attributed to the fact that CCF only uses packet service time to detect congestion therefore it cannot detect either underutilized links or nodes. On the other hand, PCCP also uses packet inter-arrival time to detect congestion, thus it is able to detect underutilized links and nodes.

CONSISE

Vedantham et al. proposed Congestion Control from Sink to Sensor (CONSISE) [37]. CONSISE approaches congestion problem in a different way compared to the other algorithms. From Sink to Sensors instead from Sensor to Sink. In this paper authors state that the factors that can contribute to sink-to-sensor congestion can be the reverse path contention and broadcast storms (due to packets which are broadcast from sinks to sensors). The actual functionality of CONSISE lies on the fact that sensor nodes are able to determine and adjust their sending rate based on the congestion level around their location at the end of each epoch. Upstream nodes are informed about the downstream nodes' sending rate through an explicit feedback and based on that they also adjust their sending rates. Then, downstream nodes select their preferred upstream node and notify it that it can send data in a higher data rate, while concurrently the rest of the nodes, which are not selected, reduce their data rate.

The latest efforts in WSN congestion control utilize more the cross layer concept, while they also focus on performance. Moreover, much effort is taking place in resource control for congestion mitigation.

UHCC

Wang et al. proposed Upstream Hop-by-Hop Congestion Control Protocol (UHCC) [88]. UHCC is a protocol based on a cross layer design that tries to reduce packet losses while guaranteeing priority-based fairness with lower control overhead. It consists of two components: Congestion Detection and Rate adjustment. An index called congestion index is responsible for providing the congestion level of each node. The congestion index takes as inputs the unoccupied buffer size and the traffic rate at the MAC layer. Based on the congestion index every upstream traffic rate is adjusted with its node priority to mitigate congestion hop-by-hop. UHCC protocol is simulated on a tree based topology and it is compares against CCF [36] and PCCP [41] protocols. Simulation results show that UHCC achieves higher throughput, better priority-based fairness and lower packet loss ratio.

FACC

Yin et al. in [43] deal beside congestion control, also with the fairness issue. They propose a ”Fairness-Aware Congestion Control Scheme” (FACC). In this algorithm intermediate nodes are categorized into ”near-sink nodes” and ”near-source nodes”. Near-source nodes maintain a per-flow state and allocate an approximately fair rate to each passing flow by comparing the incoming rate of each flow and the fair bandwidth share. This means that, if for example congestion occurs at an intermediate sensor, the generating rates of source nodes are forced to slow down, in accordance with this nodes' available bandwidth. Eventually, the whole network adapts toward

the maximum congestion-free throughput. Furthermore, the lower generating rates will alleviate wireless interference and contention. ”Near-Source node” process comes with the following functions:

- Estimation of the Available Bandwidth.
- Computation of the Flow Arrival Rate.
- Estimation of the Number of Active Flows.
- Transmission Control on Near-Source Nodes.

On the other hand, near-sink nodes do not need to maintain a per-flow state and use a lightweight probabilistic dropping algorithm based on queue occupancy and hit frequency. They actually implement three mechanisms. A Stateless Fair Queue Management Mechanism, a Hop-by-Hop Backpressure mechanism and Fairness of the Stateless Queue-Management mechanism. The first mechanism in order to achieve fairness attempts to give more chances to those flows with lower occupancy. Thus, arriving packets that belong to higher occupancy flows have higher dropping probabilities. By using the Hop-by-Hop backpressure mechanism, FACC informs the ”near-source node” for a drop packet in its flow in order for the node to adjust its sending rate.

FACC has been implemented in ns-2 simulation tool [65] and results show that FACC improves the number of dropped packets, throughput and energy consumption compare to the backpressure mechanism of CODA [34] and ”no congestion control algorithm”.

CADA

Fang et al. proposed CADA [45], an approach for Congestion Avoidance Detection and Alleviation in WSNs. In this algorithm, the congestion level of a node is measured by an aggregation of

buffer occupancy and channel utilization. CADA actually counts the growing rate of the buffer's occupancy and when it exceeds a certain limit, the node is considered congested. On the other hand if the packet delivery ratio decreases drastically, while the local channel loading reaches the maximum achievable channel utilization, it infers that there is channel congestion. For congestion mitigation CADA employs both resource control and traffic control depending on the case. If congestion takes place in an intersection hotspot, then resource control applies, while if congestion takes place in a convergence hotspot, traffic control applies. The performance of CADA was evaluated using random topologies of $500 - 5000$ nodes and a number of congestion control scenarios in the ns-2 [65] simulator. CADA was compared against TARA and a no congestion control strategy. Simulation results prove that CADA present better results concerning throughput, energy consumption, end to end delay, and average per hop delay in comparison with TARA [30] and the no congestion control strategy.

A.1.2 Congestion Avoidance Algorithms

Siphon

Wan et al. proposed Siphon [47]. Siphon is a source-to-sink congestion control protocol that aims at maintaining application fidelity, congestion detection, and congestion avoidance by introducing some virtual sinks (VS) with a longer range (IEEE 802.11 Wi-Fi) multi-radio (such as Stargate) within the sensor network. VSs can be distributed dynamically so that they can tunnel traffic events from regions of the sensor field that are beginning to show signs of a high traffic load. At the point of congestion, these VSs divert the extra traffic through them to maintain the required throughput at the base s tation.The siphon algorithm mainly aims at addressing the VS discovery, operating scope control, congestion detection, traffic redirection, and congestion avoidance. The

VS discovery works as follows: the physical sink sends out a control packet periodically with a signature byte embedded in it. The signature byte contains the hop count of the sensor nodes that should use any particular VS. Each ordinary sensor node maintains a list of neighbors through which it can reach its parent VS. Finally each VS maintains a list of its neighbor VSs. Each VS has a dual radio interface: a long range one to communicate with other VSs or with a physical sink (if applicable), and a regular low-power radio to communicate with the regular sensor nodes. In the case of congestion, a sensor node enables the redirection bit in its header and forwards the packet to its nearest VS. When the VS finds the redirection bit enabled, it routes the packets using its own long range communication network toward the physical sink, bypassing the underlying sensor network routing protocols. Siphon uses a combination of hop-by-hop and end-to-end congestion control depending on the location of congestion. If there is no congestion, it uses hop-by-hop data delivery model. In case of congestion, it uses hop-by-hop data delivery model between source nodes and the VS at point of congestion and an end-to-end approach between the VS handling the congestion and the physical sink.

Summarizing, Siphon is a set of fully distributed algorithms that employs CODA [34] mechanisms for congestion detection (buffer occupancy and wireless channel load) as well as a PostFacto Congestion Detection mechanism in which physical sink extract conclusions on possible network overload. The novelty of Siphon lies on the fact that it employs a set of virtual sinks for congestion mitigation through a traffic redirection way. Simulation are implemented in ns-2 [65] simulator on a random topology. Also Siphon has been evaluated using experimental implementation through a testbed and is being compared to CODA [34] algorithm.

Light Weight Buffer Management

Chen at al. proposed light weight buffer management technique for congestion avoidance [48]. This technique is based on the fact that a sensor y sends a packet to another sensor x only when x has the buffer space to hold the packet. Taking in account that the remaining buffer of a sensor node changes whenever it receives or forwards a packet to a neighbor node, the node incorporates in the packet header its buffer state. This is done by using one bit to indicate that its buffer is full or several bits to indicate the exact remaining buffer. Thus, neighboring nodes receive or overhear the buffer state of their neighbor and they cache its condition. According to the buffer state, the neighbor nodes decide whether to transmit or not new packets. This scheme avoids packet drops due to buffer overflow. Each node can adapt not only its own data rate, but also the data rate of its connected neighbors, since when the upstream nodes are congested the other nodes are forced to reduce its data rate. This procedure is iterative and finally leads to a maximum congestion-free throughput. This approach is different from other traffic control approaches due to the fact that it never drop packets. This algorithm is a hop-by-hop congestion avoidance. The proposed scheme is compared against global rate control, CODA's [34] backpressure mechanism and no congestion control. All simulation are performed in random topology. Finally, the authors do not present specific energy related results.

CoSMoS

Karenos et al. proposed "COngestion avoidance for Sensors with a MObile Sink" (CoSMoS) [49]. In this work the additional challenges introduced by a mobile sink are addressed. Firstly, the rate of path reconfigurations needs to be increased in order to achieve reliable data delivery. Secondly, effective load estimation techniques need to be implemented since path reconfiguration can result in sudden load changes along the paths and thirdly, transient periods of reduced path quality must be proactively prevented. CoSMoS is a scheme which is based on a

joint routing and congestion control approach. Cosmos scheme consists of two parts: A low cost, low complexity routing scheme that effectively considers the paths dynamic reliability variations during sink mobility and a regional load collection technique to estimate the maximum sustainable load of each node within a region and along a path. CoSMoS algorithm has been implemented in Mica-2 motes. Experimental results present that it manages to balance congestion and reliability to achieve higher delivery ratios without hurting throughput. No energy results are provided by authors.

Buffer and Rate Control Based Congestion Avoidance

Alam et al. proposed a "Buffer and Rate Control Based Congestion Avoidance" protocol [50]. This protocol consists of three schemes. These are: the Upstream Source Count, the Buffer Occupancy based rate control, and the Snoop based MAC level ACK. Using the first two schemes, the protocol controls the rate of upstream nodes. This fact provides two advantages. The first is that congestion, due to media access contention, is reduced as the upstream nodes proactively decrease their rate, while the second is the fact that congestion due to buffer overflow is avoided as the upstream nodes defer transmission of packets whenever their downstream nodes buffer is full. In the third scheme (Snoop based MAC level ACK) explicit ACK are avoided. Instead, each node may overhear its own transmitted packet while forwarded by its downstream node. To accomplish this, the upstream node MAC address and a sequence number are appended into the MAC frame. Simulation results are provided by authors comparing this protocol with Shortest Path Routing with no congestion control, snoop based with implicit acknowledgement and Source count and buffer occupancy based rate. Results present that this protocol can reduce collision drop Rate, increase delivery ratio and improve the network's energy efficiency.

CAEE

Khan et al. proposed CAEE protocol designed for ”Congestion Avoidance and Energy Efficiency in WSNs” [51]. The distinguished features of this protocol lie on the fact that it introduces the concept on Mobile Sinks. Specifically, in this case the network is divided into clusters called mini-sinks. The cluster head is called data collector node. The main responsibility of a data collector node is to receive and store the collected data from the sensor field to the mini-sink. The mobile sink periodically visits each mini-sink in the sensor field for data retrieval. Simulation results (against the case where just a static sink is used) prove that the CAEE protocol can increase the network's lifetime since packets travel on a few hops (until mini-sinks) and the collected by mobile sink. Concurrently congestion hot spots can be alleviated since mini-sinks represent multiple collection points. The performance of CAEE is evaluated using OMNET++ simulation tool. A uniform but random topology is selected. CAEE is compared with the case when a static sink is employed. Simulation results present that CAEE protocol outperforms the static sink scenarios concerning packet count and energy.

TADR

He et al. proposed a traffic-aware dynamic routing (TADR) algorithm [52], to route packets around the congestion areas and scatter the excessive packets along multiple paths consisting of idle and under-loaded nodes. Enlightened by the concept of potential in common physics, the 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.

Simulation have been performed in Tossim [89] simulation tool. A random topology have been used. Results show that TADR achieves its objectives and improves the overall throughput

by around 370% as compared to a benchmark routing protocol. Furthermore, TADR has low overhead suitable for large scale dense sensor networks.

ANAR

Hsu et al. proposed an Adaptive NAV-Assisted Routing(ANAR) [53] protocol to alleviate the network congestion. ANAR protocol is based on the cross-layer information and employs the existing information (the Network Allocation Vector (NAV)) from the Request-To-Send (RTS) and the Clear-to-Send (CTS) packets within the MAC scheme. Through the NAV vectors a congestion free probability is computed. This probability is carried within the route discovery process and determine the feasible route for packet delivery. The protocol is dynamic since it is able to adaptively switch between the selected paths while the level of network congestion has been changed. ANAR protocol has been implemented in ns-2 simulation tool [65] and has been compared to AODV and LBAR protocol, on a random topology. Results present better performance in terms of packet delivery ratio, end-to-end delay and power consumption.

Priority Based Medium Access Protocol

Rajsekar et al. proposed a Priority Based Medium Access Protocol for Congestion Avoidance [54]. This MAC protocol gives proportional access based on source count value. For example, a node that carries a higher amount of traffic gets more access time than others. Each node then calculates its contention window on a provided equation. Simulation series have been performed in MATLAB and claim that an optimal contention window size can minimize collision in the MAC layer and effectively help transmit all packets without delays. This protocol in not compared to any other protocol since only the concept of the algorithm is implemented in MATLAB.

TALONet

Huang et al. proposed TALONet as a power-efficient grid-based congestion avoidance scheme using multi-detouring technique [55]. TALONet implements three schemes: different transmission power levels in order to alleviate congestion in data link layer, buffer management for avoiding buffer level congestion, and a multi-path detouring technique in order to increase resources for congested traffic flows. The operation of TALONet consists of three phases. These are the network formation phase, the data dissemination phase, and the framework updating phase. In the network formation phase each node, after receiving a control packet from sink containing its location and information about the side length of each square grid, imaginarily builds a virtual grid framework G and figures out the coordinates of all virtual grid points in G. In this case nodes can be normal or TALON. TALON nodes are considered the nodes which are close to grid's cross points. During the data dissemination phase the TALON nodes are responsible for collecting and relaying the sensing data. During this phase, normal nodes through the help of a grid-based routing protocol forward their data to their closer TALON. Then, this TALON forwards these data to its closer TALON until data reaches the sink. Finally, during the framework updating phase, in order to save power, the sink broadcasts control packets including offsets for all nodes. Then the network enters again into the network formation phase. TALONet is implemented in ns-2 [65] simulation tool on a grid topology. Results present that TALONet performs better in terms of power consumption and packet drops, in comparison with TARA [30], "no congestion control", and backpressure algorithms. Results are also provided for the energy performance of the algorithm.

LACAS

Misra et al. proposed an adaptive learning solution for congestion avoidance in WSNs named "Learning Automata-Based Congestion Avoidance Algorithm in Sensor Networks" (LACAS) [56]. The target of this work is to control the data rate of intermediate nodes in order to avoid congestion before this reaches the sink. To achieve this, code capable of taking intelligent actions (called automata) is developed at each of the network's nodes that are capable of controlling the rate of flow of data at the intermediate nodes based on probabilistically how many packets are likely to get dropped if a particular flow rate is maintained. In this case an automaton "learns" from past behaviors and chooses a better data rate in order to avoid congestion. Simulation results under metrics like energy consumption, throughput and collisions prove that LACAS is able to control congestion in an efficient way.

Flock-CC Antoniou et al. proposed the Flock-based Congestion Control (Flock-CC) protocol [44]. This approach focuses on designing a robust and self-adaptable congestion control protocol for WSNs. Flock-CC adopts a Swarm Intelligence paradigm inspired by the collective behavior of bird flocks having global self- properties achieved collectively without explicitly programming them into individual nodes. The main idea is to 'guide' packets (birds) to form flocks and flow towards the sink (global attractor), whilst trying to avoid congestion regions (obstacles). The direction of motion of a packet flock is influenced by repulsion and attraction forces between packets, as well as the field of view and the artificial magnetic field in the direction of the artificial magnetic pole (sink). In particular, packets are 'flying' through the network while being attracted to nodes with low wireless channel loading, and being repelled from nodes with high buffer occupancy. Thus, in Flock-CC congestion is inferred using both buffer occupancy and wireless channel loading. Congestion notification is implicitly performed having each node broadcasting (using a small control packet) its buffer occupancy and the wireless channel loading in its vicinity to all nodes within the node's transmission range. Each packet synthesizes the attraction and repulsion

forces to and from neighboring packets as well as the global magnetic force towards the sink and moves in an oriented manner through the network whilst avoiding congestion regions. Flock-CC is simple to implement at the individual node (each node follows a small set of rules), and involves minimal information exchange. Flock-CC was tested on both lattice and random topologies of 300 nodes using a number of scenarios for different network and traffic conditions. Performance evaluations showed the effectiveness of the Flock-CC protocol in balancing the offered load by exploiting available network resources. Flock-CC was shown to provide graceful performance degradation in terms of packet delivery ratio, packet loss, delay and energy tax under low, high and extreme traffic loads. In addition, the proposed approach achieved robustness against failing nodes, scalability in different network sizes and outperformed typical conventional approaches.

LVCC

Antoniou et al. proposed the Lotka-Volterra based Congestion Control (LVCC) [46] protocol. LVCC focuses on streaming applications in wireless sensor networks and on how congestion can be prevented by regulating the rate of each traffic flow based on the Lotka-Volterra population model. LVCC detects congestion on the basis of buffer occupancy, while congestion avoidance is performed by means of traffic control. The traffic flows initiated by each node play the role of competing species and the buffer (queue) capacity of the parent node can be seen as the limiting resource. LVCC provides hop-by-hop rate adaptation by regulating the traffic flow rate at each node. Each node is in charge of self-regulating and self-adapting the rate of its traffic flow i.e., the rate at which it generates or forwards packets. The traffic flows compete for available buffer capacity at their one-hop-away receiving node involved in the path leading to the sink. Each sending node is expected to regulate its traffic flow rate in a way that limiting buffer capacities at all receiving nodes along the network path towards the sink are able to accommodate all received packets. The sending rate evolution of each flow will be driven by variations in buffer occupancies

of nodes along the network path towards the sink. Due to the decentralized nature of the LVCC protocol, and in order to satisfy the need for low communication overhead, each node regulates its traffic flow rate using local information (i.e. from one-hop away neighbors). LVCC involves minimal exchange of information and computation burden and is simple to implement at the individual node. Performance evaluations revealed that LVCC achieves adaptability to changing traffic loads, scalability and fairness among flows, while providing graceful performance degradation as the offered load increases. LVCC was not compared against related congestion avoidance approaches found in WSN literature.

A.1.3 Reliable Data Transport Algorithms

GARUDA

Park et al. proposed GARUDA [63]. GARUDA provides reliable point-to-multipoint data delivery from the sink to the sensors. GARUDA consists of the following elements.

- an efficient pulsing based solution for reliable short-message delivery
- a virtual infrastructure called the core that approximates a near optimal assignment of local designated servers, which itself is instantaneously constructed during the course of a single packet flood
- a two-stage NACK based recovery process that effectively minimizes the overheads of the retransmission process, and performs out-of-sequence forwarding to leverage the significant spatial re-use possible in a WSN
- a simple candidacy based solution to effectively support the different notions of reliability that might be required in a WSN.

GARUDA has been implemented if ns-2 [65] simulation tool and results focus on reliability and energy.

STCP

Iyer et al. proposed STCP [58]. STCP is a generic, scalable and reliable transport layer protocol where the majority of functionalities is implemented in the sink. STCP supports networks with multiple applications and provides additional functionalities such as controlled variable reliability and congestion detection and avoidance. In STCP, before transmitting packets, sensor nodes inform the sink through a ”Session Initiation Packet”. Through this packet, the sink is informed about the number of flows initiated from a source, the type of data, the transmission rate, and the required reliability. As soon as the sink receives this packet it sends an ACK packet to the source node, and the source node can start sending packets. STCP packet headers consists of the sequence number, a clock field, flow id, and congestion notification bit. Since the sink knows the rate of transmission from the source, the expected arrival time for the next packet can be found. The sink maintains a timer and sends a negative acknowledgement (NACK) if it does not receive a packet within the expected time. Also, sensor nodes specify the required reliability for each flow in the session initiation packet. Reliability is measured as the fraction of packets successfully received. Concerning congestion control, nodes inform the sink whether they are experiencing buffer overflow, by setting their congestion notification bit, while the sink informs the source of a congested path by setting the congestion bit on the ACK packet. In this case, a source node may alter its routing path or decrease the sending rate to mitigate congestion.

RCRT

Paek at al. proposed RCRT [59], a protocol that focuses on reliable delivery of sensor data form source to sink, while avoiding congestion collapse. RCRT focuses on the transport layer and its traffic management functionality resides on the sink. RCRT attempts to guarantee 100% reliable data delivery based on a NACK scheme. Thus, in case of packet losses, the sink requests the missing packets from the source by sending a NACK with the missing packet numbers. RCRT implements at sink three basic components: the congestion detection component which detects congestion through round trip time, rate adaptation, and rate allocation which decreases flow rates to control congestion; congestion detection performed using as congestion indicator the ”time to recover loss”. This means that as long as the network is able to repair quickly enough the packet losses (e.g around on Round Trip Time) the network in not congested. In other case it figures out that there are congested spots in network. In case of congestion RCRT applies a rate adaptation mechanism to control it. Also RCRT applies a rate allocation mechanism on which specific rates are allocated to each flows when the application differs (e.g video transmission etc). STCP has been implemented in TOSSIM simulation tool and results are provided also for energy spent and packet's latency.

Extended DCCP

Liu at al. proposed an extension to Datagram Congestion Control Protocol with a new congestion control component [60]. DCCP is a transport layer protocol designed for providing a standard way to introduce congestion control and congestion control negotiations into multimedia applications. Extended DCCP comes with the following added functions:

- Buffering of received packets at the receivers
- retransmission of lost or corrupted packets by the senders

- detection and deletion of duplicated packets at the receivers
- in-order delivery of received packets to the application program at the receivers

In this case the sender has four states: Normal State, Congestion State, Failure State (route change or link failure), and Error State (transmission error). Overall, extended DCCP has the ability to provide a reliable operation, with a good aggregate throughput.

A.2 Comparative Tables for Algorithms Presented in Chapter 2

Table 5: Congestion Control Algorithms.

Protocol/ Mechanism	Congestion Detection	Congestion Notification	Congestion Mitigation	Traffic Direction	Fairness	Energy Conservation	Year of Publication
ARC [22]	The event if the packets are successfully forwarded or not	Implicit	Traffic Control	Source to Sink	Yes	Good	2001
CODA [34]	Buffer occupancy and wireless channel load	Explicit	Traffic Control	Source to Sink	No	Good	2003
CCF [35]	Packet Service Time	Implicit	Traffic Control	Source to Sink	Yes	Good	2004
Fussion [36]	Buffer Occupancy and Wireless Channel load	Implicit	Traffic Control	Source to Sink	No	Good	2004
CONSISE [37]	Wireless Channel Load	Implicit	Traffic Control	Sink to Sensor	Yes	Good	2005
COMUT [38]	Cluster/ Traffic Intensity Estimation	Explicit	Priority Based Traffic Control	Source to Sink (Cluster-by-Cluster)	No	Good	2005
Sen- TCP [39]	Buffer Occupancy/ Packet inter-arrival Time	Implicit	Traffic Control	Source to Sink	No	Good	2005
BGR [40]	Buffer Occupancy and Wireless Channel Load	Implicit	Resource and Traffic Control	Source to Sink	Yes	N/A	2006
TARA [30]	Buffer Occupancy and Wireless Channel load	Explicit	Recourse Control	Source to Sink	Yes	Good	2007
PCCP [41]	Packet Interarrival Time/ Packet Service Time	Implicit	Priority Based Traffic Control	Source to Sink	Yes	N/A	2007
RCRT [59]	Sink decides based on ”time to recover loss”	Implicit	Sink Based Traffic Control	Sink to Source	No	N/A	2007
HTAP [8]	Buffer Occupancy	Implicit	Recourse Control	Source to Sink	No	Good	2007
CONSISE [37]	Wireless Channel load	Implicit	Traffic Control	Sink to Source	No	Good	2007
UHCC [88]	Buffer Occupancy and Wireless Channel load	Implicit	Traffic Control	Source to Sink	Yes	Good	2007
FACC [43]	Wireless Channel load	Implicit	Traffic Control	Source to Sink	Yes	Good	2009
CADA [45]	Buffer Occupancy and Wireless Channel load	Implicit	Recourse and Traffic Control	Source to Sink	No	Good	2009
DAIPaS [4]	Buffer Occupancy and Wireless Channel load	Implicit	Resource Control	Source to Sink	Yes	Good	2011

Table 6: Congestion Avoidance Algorithms.

Protocol/ Mechanism	Congestion Detection	Congestion Avoidance Mechanism	Year of Publication
Siphon [47]	Buffer Occupancy, Wireless Channel load and Sink Decides	Traffic redirection through Virtual Sinks	2005
Light- Weight Buffer Management [48]	Buffer occupancy	Traffic Control	2006
CoSMoS [49]	Per node load collection technique	Rate-based control	2007
Buffer and Rate Control Based Congestion Avoidance [50]	Buffer Occupancy and Wireless Channel load	Traffic Control	2007
CAEE [51]	Buffer Occupancy	Load Collection through a Mobile Sink	2007
TADR [52]	Buffer Occupancy	Resource Control	2008
ANAR [53]	MAC layer load	Traffic redirection through alternative paths	2008
Priority Based Medium Access Protocol [54]	Buffer Occupancy	Most Loaded Nodes get higher priority to Medium	2008
TALONet [55]	Buffer Occupancy and Wireless Channel load	Resource and Traffic Control	2009
LACAS [56]	N/A	Learning Automata adjust flows Rate	2009
FCC [44]	Buffer Occupancy and Wireless Channel load	Resource Control	2009
LVCC [46]	Buffer Occupancy	Traffic Control	2009

Table 7: Reliable Data Transport Mechanisms.

Protocol/ Mechanism	Hop- by- Hop /End-to -End	Traffic Direction	Reliability	Year of Publication
PSFQ [61]	Hop-by-Hop	Downstream	Packet	2002
ESRT [57]	End-to-End	Upstream	Event	2003
Directed Diffusion [24]	Hop-by-Hop	Upstream	Event	2003
RMST [62]	Hop-by-Hop	Upstream	Packet	2003
GARUDA [63]	Hop-by-Hop	Downstream	Packet and Destination Related	2004
STCP [58]	End-to-End	Upstream	Event and Packet	2005
RCRT [59]	Hop-by-Hop	Upstream	Packet	2007
EDCCP [60]	Hop-by-Hop	Upstream	Packet	2009