ABSTRACT

One of wireless sensor networks' challenging issues is the congestion control problem. While a lot of attention has been given to rate reduction (Traffic Control) methods to avoid congestion, a new approach emerged. Resource Control mitigates congestion using idle and low load nodes, offers higher throughput and utilizes sources uniformly. In this Thesis we focus on three resource control schemes: TARA, a highly cited algorithm, HTAP, and DALPaS, simpler but also efficient proposals, used for event-based applications.We implemented the three algorithms using the Prowler/Rmase simulation environment and then studied their behavior and performance in different conditions. Finally, we optimize each algorithm using their intrinsinc parameters. Simulation results show that the simplicity of HTAP and DALPaS can outperform TARA in a constantly congested scenario. Also, the performance of each algorithm can be boosted-up when studying thoroughly its intrinsic parameters. This Thesis proposes a variety of small and simple changes for each algorithm that can increase its performance dramatically.

Aristodemos Pafitis – University of Cyprus, 2011

EVALUATION OF RESOURCE CONTROL ALGORITHMS FOR CONGESTION CONTROL IN WIRELESS SENSOR NETWORKS

Aristodemos Pafitis

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

at the

University of Cyprus

Recommended for Acceptance

by the Department of Computer Science

May, 2011

APPROVAL PAGE

Master of Science Thesis

EVALUATION OF RESOURCE CONTROL ALGORITHMS FOR CONGESTION CONTROL IN WIRELESS SENSOR NETWORKS

Presented by

Aristodemos Pafitis

Research Supervisor

Dr Vasos Vasileiou

Committee Member

Dr Chryssis Georgiou

Committee Member

Dr Nikolas Stylianides

University of Cyprus

May, 2011

ACKNOWLEDGEMENTS

I would deeply like to thank my advisor, Dr. Vasos Vassiliou, for his valuable guidance and support through my study and research. Also I would like to thank my friend and colleague Charalampos Sergiou for his valuable help, comments and technical support. At last I am grateful to my parents Marios and Areti for their moral support and help, during the research and writing of this MSc Thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to one or more main locations[19] [7]. Initially, the development of WSNs was motivated for military use, but as sensor networks develop rapidly, more and more remote sensing applications are being enabled. A wide range of them is now deployed in civilian areas for industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, environment and habitat monitoring [4], healthcare applications [21], home automation, traffic control, object tracking[12][3], transportation and urban monitoring [6].

In recent years multiple and diverse studies have been carried out concerning many different aspects of wireless sensor networks mainly, in physical layer, MAC layer, network layer, energy efficiency, sensor's operating systems, and applications. Having addressed many of the fundamental problems of sensor networks, lately the issue of congestion control and avoidance has arisen and attracted a lot of attention. Congestion in WSNs is more complicated in comparison to conventional networks, since it may appear in different types and at different places.

Concerning the type of congestion, we can discriminate it in two categories. The first category of congestion is due to interference among several densely deployed sensors. Many nodes try to transmit simultaneously resulting in packet collisions and therefore wasted energy consumption. The second category is happening within a particular node. Usually because of high-rate input traffic, the queue (buffer) that holds the packets to be transmitted overflows [14]. This is the conventional definition of congestion and is also the main cause of packet losses. This type of congestion assumes that a reliable MAC protocol is used to avoid packet collisions from simultaneously transmitting nodes. A variety of solutions has been proposed to solve this problem. One popular strategy which is gaining ground is congestion mitigation. This is done either by rate reduction (traffic control) or by resource increment (resource control). This thesis focuses on the later class of algorithms, who aim to mitigate congestion by creating alternative paths from the source(s) to the sink(s).

1.2 WSNs Special Characteristics

Sensor nodes can be imagined as small computers, extremely basic in terms of their interfaces and their components. They usually consist of a processing unit with limited computational power and limited memory, sensors, a communication device (usually radio transceivers), and a power source usually in the form of a battery. Wireless sensor networks are ad-hoc networks, which means they lack infrastructure, therefore are self configured. These are the main characteristics of WSN's but they have some more distinctive features:

- Their network topology is unique. Actually is constantly dynamic due to the time varying link and node variation. Sometimes is also the case where some nodes are mobile in the network.

- They are addressed to a variety of diverse applications from habitat monitoring, target tracking, to security surveillance and more. Different applications have different reliability and Quality of Service requirements.
- WSNs have peculiar traffic characteristics. The primary traffic is upstream flow from source to sink, but occasionally the sink may generate downstream traffic for query or control. Usually the upstream communication is one to many and dependent on specific applications the delivery can be event-based, continuous or query-based.
- Sensor nodes have limited resources including low computation capability, small memory, low communication bandwidth and limited battery.
- Messages in sensor networks usually have small size compared to existing networks. As a result there is no concept of segmenting a message.
- The sensing range in general is much smaller than the radio range, and thus the density required for sensing coverage results in a dense network.
- Existence of asymmetric links. The communication links in a sensor network are not reversible in general.

1.3 Thesis Objectives

In this Thesis we present the major aspects of congestion control in wireless sensor networks and focus on a few resource control mechanisms on alleviating congestion. Our work is particularly concentrated on event-based networks. Although it has been stated by some researchers that all wireless sensor networks are event-based [20], we would like to distinguish them from networks that periodically transmit data. In event-based networks, data packets are produced only upon the observation of specific events that satisfy pre-specified conditions. Our research involves two major parts. Firstly, we have implemented three Resource Control Schemes using a novel, for our University, wireless sensor simulator. Secondly, by the means of the simulator we have studied and analyze each algorithm and carried out a performance evaluation of the three. In the next chapter related work is discussed, in Chapter 3 the algorithms under investigation are described, next the platform on which the simulations took place will be presented. Finally we discuss the results and conclusions derived from simulations.

Chapter 2

Congestion in WSNs

2.1 Introduction

In this research we concentrate on event-based networks. The distinguishing feature of event-based networks is the production and transmission of a big number of data packets to the sinks when the network comes in a crisis state. Crisis state is the network's state, when an event satisfies a pre-specified condition. When a specific condition is satisfied, there may be a huge transmission of data packets, which, if not controlled, can easily lead to congestion. Congestion, especially in these networks, is highly undesirable, not just because of the obvious power consumption, but also due to the fact that it is able to destroy the networks' mission. The data packets which are produced when an event- based network satisfies the pre- specified condition are critical for the success of the applications.

A typical example is the case of fire in a forest. Sensors are programmed to send data packets when the temperature in the forest exceeds a pre-specified value. In the case of a fire, sensor nodes which are next to the event, sense the fire and begin transmitting a large amount of data packets simultaneously. The target of this network is the continuous provision to sinks (i.e. local fire stations) with data packets, in order to keep them updated, for the fire's frontline. If the high load of data packets is left unattended (no congestion control) the network will become congested and the packets will not be forwarded to sinks. The result will be the failure of the network's mission. Due to the high importance of these packets, and the strict time limitations, the reduction of data rate is prohibited (i.e. in a forest which is getting burned, almost every data packet provides new information for the spreading of fire).

2.2 Types of Congestion

Traditionally, congestion control in a network is done in the Transport Layer. In wired networks, transport protocols are used to eliminate or mitigate congestion, reduce packet losses, and guarantee endto-end reliability. However, protocols such as UDP or TCP are not effective in WSNs and cannot be applied directly to them. This is one major reason why congestion control in WSNs has drawn a lot of attention. In order to present the related work on the subject, it is useful first to classify congestion according to the following two characteristics: how packets are lost and where.

1. How Packets are lost

   Mainly there are two causes for congestion in WSNs. The first is due to packet arrival rate exceeding packet service rate. This takes place within a particular node when the queue used to hold packets to be transmitted, overflows. This is also the conventional definition of congestion. The second is influenced by the link level performance. It has to do with contention, bit-error, and interference at the physical and data link layers. This happens in a particular area in the network, usually when two or more nodes within range attempt to transmit simultaneously, resulting in losses due to packet collision.

2. Where in network packets are lost

- Hotspot near the source. Densely deployed sensors generating data events during crisis state will create hotspots very close to sources (with in two hops distance).
- Hotspot near the sink. Even sparsely deployed sensors that generate data at low rates create hotspots potentially anywhere in the sensor field but most likely farther than the sources, near the sink.
- Forwarder Congestion. A sensor network may have more than one flow (sink source pair), and these flows will probably intersect. The area around the intersection will likely become a hot spot. In a tree-like communication propagation, every intermediate node in the tree can suffer from forwarder congestion. This scenario appears to be more challenging than the previous two because it is vary difficult to predict the intersection zones due to the network dynamics. In such case, even a sparsely deployed network will create both transient and persistent hotspots distributed in the sensor field.

Congestion in WSNs has a direct impact on energy-efficiency and application's QoS. First, congestion can cause buffer overflow and lead to longer queuing time and therefore more packet loss. And not only reliability and QoS are affected with packet losses, but also the limited available energy is wasted. Second, it can still degrade link utilization. Third, link-level congestion usually results in transmission collisions. This in turn will increase packet service time and waste additional energy. Therefore congestion in WSNs must be efficiently controlled by either avoidance or mitigation. In the following paragraphs we will discuss the mechanisms used to deal with this problem.

2.3 Control Schemes

Generally algorithms that deal with Congestion in WSNs can be classified in three major categories. These are Congestion Control, Congestion Avoidance and Reliable Data Transmission, although there are not clear and explicit boundaries among them.

We will consider as congestion control algorithms those that take reactive actions when congestion arises in the network and their target is to mitigate it. These algorithms involve the MAC and network layer and in some cases the transport layer. Congestion avoidance algorithms are considered those that take actions in order to prevent congestion from happening. These usually involve MAC and network layer. On the other hand, reliable data transmission algorithms are those that beside their effort to control congestion in the network they also attempt to recover all or part of lost information. These algorithms normally apply when all information is critical for the application and normally involve transport layer.

Basically, the presence of congestion shows us that the load is (temporarily) greater than resources (in a part of the network) can handle. In this case three are the optional control schemes to be used: Decrease the load, increase the resources or employ MAC layer enhancements. The latter choice should help the most in the direction of interference based congestion (frequent packet collisions). 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. An effective way to reduce this type of congestion is to perform a phase shifting [28]. Small amounts of phase shifting can be performed by introducing slight jitters at the data-link layer. To handle buffer based congestion (packet drops due to buffer overflow) one may employ the other two methods: a) decreasing the load or b) increasing the resources as these could help empty the buffers of intermediate nodes. At this point we should also note that it is possible to have more than one type of congestion occurring at the same time.

2.4 Classification of Algorithms

Furthermore, congestion control and avoidance algorithms can be characterized based on some attributes. These are the way congestion is detected, how the congestion event is been notified to the rest of the network, what mechanisms are used to mitigate congestion and how is congestion prevented.

1. Congestion Detection:

   Currently there are four major ways that algorithms use to detect congestion. These are:

   - 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. Some algorithms detect congestion when buffer drops arise while other act proactively assuming congestion when the buffer occupancy exceeds a certain percentage while the sending rate is lower than the receiving rate.
   - Buffer occupancy and Channel Load: With this method congestion is detected either on the medium or the buffer. Concerning the buffer occupancy the tactic used is the same as mentioned before. For the channel load case, if a node senses that channel

loading reaches a certain fraction of channel capacity it infers that new packet transmission will probably lead to packet collision. This category also includes algorithms that form clusters and measure congestion through traffic intensity.

- Packet's transmission time metrics: Algorithms that employ this method to detect congestion use packet service time and packet inter-arrival time to detect congestion, or a combination of them. If these times exceed a time limit it is inferred that congestion has risen.
- Sink decides: In this case, decision for congestion state resides on the sink. Usually a *time to recover loss* is used[16]. In case of packet loss, sink issues a NACK asking for retransmission of this packet. If the packet is not successfully received within a RTT, sink assumes that congestion has occurred.

2. Congestion Notification:

   A second classification can be considered the way that algorithms use to notify the rest of the network that congestion has occurred. This notification is either implicit or explicit.

   - Implicit: This method is the most widely used. Specifically congestion is notified to the rest network by overhearing data packets which are on the fly. If congestion has been detected, notification is piggybacked in data packet headers and some times in ACK packets. So no more extra packets are added to the network when is already congested.
   - Explicit: Using explicit notification, control packets are sent by congested nodes to the rest nodes to inform them about congestion. This method was used by the first congestion algorithms. It is avoided in later designs, since it has been proven that

sending control packets when congestion has occurred adds significant load to the already congested environment.

3. Congestion Mitigation:

   There are two ways to mitigate congestion. Either by rate reduction or by the creation of alternative paths from source to sink for forwarding the excess data packets.

   - Traffic Control (Rate Reduction): When the network signals, either explicitly or implicitly, that congestion has occurred, source nodes reduce their data rate until congestion has alleviated. Rate reduction can be performed using different ways like AIMD (Additive Increase Multiplicative Decrease) etc. The advantage of this method is that the burden of congestion alleviation lies almost solely to source and this is happening relatively quickly since the load in the network decreases. The disadvantage is that, especially in event-based networks where traffic appears when the monitored event takes place, reduction of data rate can jeopardize the network's mission since all data packets carry valuable information about the event. Also, 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 (interference, multi-path fading etc). It is trivial that this method is inapropriate for event - based environments, as it has been stated in section 2.1

   - Resource Control (Alternate path creation): To eliminate the disadvantage of rate reduction strategy, alternative path creation has been proposed. In this case, when the network is congested, data packets follow alternative paths, which are not congested, in order to be forwarded to sink. This method has the advantage that source rate reduction 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.

4. Congestion Prevention:

   This attribute is mainly oriented to congestion avoidance algorithms. The problem can be avoided using similar techniques as with congestion mitigation (rate reduction and alternative path creation) with the difference that congestion avoidance algorithms act preventive instead of reactive. Also, some other techniques have been proposed and are presented below.

   - Mobile or Virtual Sink: In this scheme the sink is 'moved' near the regions of fields that show signs of high load. Network is split into clusters and there, an in network storage model is introduced. Cluster head collects all load from its nodes and transmit them to the mobile sink when it passes near the cluster. An other proposal was the concept of mini-sinks where some nodes with longer range. Tunnel the traffic event from regions that are going to be congested.
   - MAC Layer Enhancements: They are used in order to avoid collisions in the medium. Implementation spans from priorities in heavy loaded nodes or by phase shifting.
   - Learning automata: In this case 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 in maintained.

5. Further Attributes:

Other classifications can also be made according to the following parameters: Traffic direction (upstream or downstream), packet transport (hop-bu-hop or end-to-end), whether fairness is supported, if multiple priority classes exist and more.

2.5 Comparison of Congestion Control Methods

By employing the traffic control method (rate reduction), affected nodes inform the nodes that forward packets to them to reduce their sending rate since soon they will be overloaded. In such case, affected nodes use backpressure messages and this message is relayed backwards to sources until the data rate is controlled. While this method can be effective in cases where total load in the network is heavy (rare event since WSNs are usually redundantly deployed) or in specific scenarios it cannot be effective in the forwarder congestion scenario. In this case it is possible that a number of flows with low data rate pass through a specific node and congest it. If the flows are committed to reduce the already low data rate while the remaining network could be underloaded. An other significant disadvantage of this method is the fact that in WSNs heavy load in produced when an event is taking place. Reducing the data rate could negatively affect the mission of the network. We should note that traffic control is the preferred method by the majority of congestion control algorithms.

Resource control method operates in a different way in comparison with traffic control method. In this case due to redundant deployment of nodes in the network, affected sources attempt to employ nodes which are not involved in the current flow for sending packets from source to sink in order not to reduce the data rate. Especially concerning the forwarded congestion scenario the advantage is obvious since flows that are merging to a specific node are able to use other paths to reach the sink. Thus data rate is not affected and final throughput is increased. Of course, using this method, is not possible for each node to arbitrarily decide which flows to reject and for rejected flows to randomly choose which node to join. A number of parameters has to be considered in mind according to the type of congestion and to application requirements. This method so far, has not attracted a lot of attention. The most representative work that employs resource control is TARA [11].

In this Thesis I concentrate only in resource control algorithms. My effort is to analytically study some representative algorithms and compare them under different situations(various topologies and different data loads). Finally I try to optimize each one by thorough investigation of their parameters.

2.6 Related Work on Resource Control Schemes

2.6.1 HTAP

Sergiou et al. [23] proposed Hierarchical Tree Alternative Path (HTAP). HTAP is a scalable and distributed framework for minimizing congestion in event based networks. It does not employ rate limiting actions, but tries to maintain high level of packet rate while minimizing packet losses. It is based on the creation of alternative paths from source to sink, using the plethora of a networks unused nodes. The use of these nodes leads to a balanced energy consumption, avoiding the creation of holes in the network and prolonging network lifetime. The algorithm was first described and presented in [22]

2.6.2 TARA

Kang et al. [11] proposed Topology Aware Resource Adaptation (TARA) protocol. TARA focuses on the adaptation of networks extra resources in case of congestion, alleviating intersection hot spots. TARA uses buffer occupancy and channel load for congestion detection. Congestion alleviation is performed with the assistance of two important nodes. The Distributor and the

Merger. Between them a detour path is established starting from the Distributor who divides its output load between his original path and the new detour path. Respectively, the Merger merges the two flows. As long 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.

2.6.3 DALPaS

Sergiou et al. also proposed DALPaS, a simple Dynamic Alternative Path Selection scheme that attempts to face congestion by increasing capacity while maintaining performance requirements [24]. Each node in the network informs its neighbors implicitly using the ACK messages, for his congestion state, remaining power and an availability flag. Congestion Detection is based on nodes Buffer occupancy. DALPaS employs two states, soft and hard. During soft state, a proactive scheme is used to avoid transient congestion. DALPaS algorithm attempts at first to keep each node receiving data from one flow. In case where the performance requirements of the application are not satisfied, it is possible for a node to enter in hard state. In this case it informs his neighbors using his availability flag, that the node is unable to accept any more packets from any flows. In any case, each node can in an easy and simple way, find the next node with minimum computation, by sorting its neighbor list. DALPaS can efficiently and adaptively choose an alternative path to avoid congestion by taking into account a number of critical parameters that affect the performance of the network.

2.6.4 BGR

L. Popa et al. proposed Biased Geographical Routing (BGR) protocol to reactively split traffic when congestion is detected [17]. Two congestion control algorithms namely, In-Network Packet

Scatter (IPS) and End-to-End Packet Scatter (EPS), are used to lenerage BGR to avoid the congested areas of the network. IPS alleviates transient congestion by splitting traffic immediately before the congested areas. Additionally, EPS alleviates long term congestion by splitting the flow at the source, while 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). Because the bias is randomly chosen, BGR makes congestion worse under some situations. In addition BGR node location information provided by GPS or other coordinate system is needed. Also, the AIMD strategy is not very effective in WSNs because it provokes a saw-tooth rate behaviour which may violate QoS requirements. In addition, AIMD-like mechanisms take a long time for data rates to converge in low-rate wireless links.

2.6.5 Flock - CC

Antoniou et al. proposed Flock - based Congestion Control (Flock CC) mechanism [2]. It focuses on designing a robust and self adaptable congestion control mechanism in Autonomous Decentralized Networks with emphasis on WSNs. It is inspired by the collective behavior of bird flocks having global self-properties achieved collectively without explicitly programming them into individual nodes. The 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 the packet flocks is influenced by (a) repulsive and attractive forces between packets and (b) the gravitational force of the sink. The Flock-CC approach provides sink direction discovery, congestion detection (based on nodes' buffer load and channel load) and traffic management. Evaluations proved that the proposed congestion control mechanism is effective in load balancing and provides graceful performance dergradation under low, medium and high traffic loads compared to conventional congestion control schemes.

2.6.6 Siphon

Wan et al. proposed 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 multi-radio (IEEE 802.11) capabilities. VSs are placed within the network. They 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 station. The Siphon algorithm aims to address the VS discovery, operating scope control, congestion detection, traffic redirection and congestion avoidance. The VS discovery woks 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. Finall 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 the physical sink 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 towards 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. Siphon has been evaluated using packet-level simulation and experimental implementation through a testbed. The optimality of Siphon is dependent on the optimality of the number of VSs.

2.6.7 TADR

He et al. Proposed a traffic-aware dynamic routing (TADR) algorithm, to route packets around the congestion areas and scatter the excessive packets along multiple paths consisting of idle and under-loaded nodes [18]. Enlightened by the concept of potential in classic physics, the TADR algorithm is applied 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. The simulation results show that TADR achieves its objectives and improves the overall throughput as much as 3 times as compared to a benchmark routing protocol. Additionally, TADR has low overhead suitable for large scale dense sensor networks.

2.6.8 TALONet

Huang et al. Proposed TALONet as a Power-Efficient Grid-Based Congestion Avoidance Scheme Using Multi-Detouring Technique [8]. TALONet implements three schemes. Different transmission power level in order to alleviate congestion in data link layer, buffer management for avoiding buffer level congestion and a multipath 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 framework updating phase. In 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 (cp) 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 Data Dissemination phase TALON nodes are responsible for collecting and relaying the sensing data. During this phase normal nodes through the help of grid-Based routing protocol forward their data to their closest TALON. Then this TALON forwards this data to its closest TALON and the process repeats until data reaches sink. Finally during framework updating phase , in order to save power, sink broadcasts control packets including offsets for all nodes. Then the network enters again to network formation phase.

2.6.9 CADA

Fang et al. proposed CADA, which stands for Congestion Avoidance Detection and Alleviation in WSNs [5]. In this algorithm nodes congestion level is measured by an aggregation of buffer occupancy and channel utilization. It actually counts the growing rate of buffers occupancy and when exceeds certain limit , node is considered congested. On the other hand if 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 rate 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. 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 and a no congestion control algorithm.

2.6.10 Comments

In this thesis we study and analyze three of the Resource Control Methods proposed by the academic community, HTAP, DALPAS and TARA. The reasons we focused only in these three algorithms is that they don't require any special nodes (like Siphon) nor an underlying coordinate system using GPS or similar technologies (like BGR and TALONet) and they use the same MAC protocol (CSMA/CA). In these works, all nodes are considered equal and of same technology. TARA is considered a state of the art algorithm, while HTAP and DALPaS are two very simple, although reliable and efficient schemes, inspired and created in the University of Cyprus.

The use of plain nodes and of the same technology, is of great value since it keeps the routing mechanisms simple and the cost of the network is lower. Additionally using the same nodes, offers flexibility during the deployment of the network. For example, one may chose to scatter the nodes over a region by dropping them from a flying machine [9]. Moreover, a network employing a homogenous set of sensors is more robust to hardware failures and is easier to maintain.

Chapter 3

Description of Evaluated Algorithms

This section presents in greater detail the three algorithms chosen for evaluation under the scope of this thesis..

3.1 Hierarchical Tree Alternative Path

The hierarchical tree alternate path (HTAP) algorithm bases its functionality on the creation of alternative paths from source to sink in order to prevent congestion from happening. HTAP uses the node's current state for congestion detection, i.e. its buffer occupancy. HTAP consists of two algorithms: the Alternative Path Creation algorithm (APC) and the Hierarchical Tree Creation algorithm (HTC). The philosophy behind these two algorithms is similar. Both of them are based on the creation of alternative paths from the source(s) to the sink when congestion is taking place. For the creation of these paths the nodes which are in dormant state are used. APC uses these nodes in a general random way, compared to HTC where these nodes are placed in a hierarchical tree from source to sink. The combination of the two comprises HTAP. The algorithm is completed with the use of a Topology Control scheme and a method of handling powerless nodes. Each algorithm and their combination is explained hereafter.

Topology Control

A topology control scheme first applies in the HTAP algorithm. Topology control is crucial in WSNs since the redundant number of nodes and their dense deployment is possible to impose several restrictions to the correct operation of the network. 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 attempts to employ network's extra resources to forward the excess packets to sink, a redundant number of paths is needed. But in order to maintain the performance characteristics of the network in case of congestion, these paths must be carefully selected. An effective topology control algorithm should be able to preserve connectivity with the use of minimal power while it keeps an optimum number of nodes as neighbor to each other. An appropriate mechanism is the Local Minimum Spanning Tree algorithm (LMST). In LMST each node builds its local minimum spanning tree independently using Prim's algorithm and keeps only on tree, nodes that are one hop away as neighbors. Our implementation of the algorithm in the simulator was simpler. To achieve a small neighbor list with minimum hops, we restricted the power of the initial level discovery message. In this way it would be received by nodes within a small radius, thus emulating the LMST algorithm. The topology control is directly connected with the hierarchical tree creation which will be explained in the next section.

Hierarchical Tree Creation

1. Path Creation

   In this first step, a hierarchical tree is formed beginning at the source node. During this phase, the source node is self-assigned as level 0 and broadcasts a level-discovery message. Sensors that receive this packet are handled as children to the source node and are set as level 1. Each of these nodes rebroadcasts the level discovery message and the pattern continues with level 2 nodes and so on. The process iterates until all nodes are assigned a level and the level discovery msg reaches sink. If a node is set as a higher level than the received msg, it updates its level and further broadcasts the packet. Otherwise the packet is ignored. When the procedure finishes, it is possible that the sink receives more than one level-discovery packets, from different nodes and with different level. This is an indication that disjoint paths are reaching the sink. Also, the algorithm takes provisions for nodes that cannot propagate their level anymore. If for example a node is level 4, all its neighbors are level 3 and higher level nodes are out if reach, this node cannot propagate any received packets. For this reason he removes itself from the next-hop lists of its neighbors by broadcasting a negative ack packet, indicating that it cannot forward any packets to the sink. In the case that a node cannot connect to a higher level node, it is possible to connect to a node of the same level.

2. Flow Establishment

   Connection is established between each transmitter and receiver using a 2-way handshake. Packets are exchanged between each transmitter and receiver in the network, in order to get connected. Through this packet exchange, the congestion state of each receiver is communicated to the transmitter. Upon each reception of a message, an ack message is sent back in which the current congestion state is piggybacked. In this way a source node is aware of the congestion state of all its children. When the congestion state of a node reaches a pre-specified limit it notifies its parent immediately.

Alternative Path Creation

This algorithm runs when congestion is possible to occur in a specific node in the network. Since the topology control algorithm overcomes collisions in the medium by choosing the smallest transmission power to maintaining network connectivity, congestion is possible to happen when a node is receiving packets with a higher rate than it is transmitting (buffer based congestion). In a wireless sensor network where all nodes beside the sink are exactly the same, this can happen if a node is receiving packets from at least two flows or if nodes to which is sending packets cannot accept them anymore. Receiving packets from two or more flows is possible to happen since a node could be the physical and logical (tree) neighbor of more than one nodes and in either case it could be the node with the shortest path to sink. So in cases where more than one sources are transmitting packets through this node with a total rate bigger than its transmitting rate, congestion is possible to happen. When the buffer of a node starts filling with an increasing rate, the node takes action. When the receiving rate reaches a threshold, the node sends a backpressure message to specific nodes that are transmitting packets through it (its parent nodes). The selection of these nodes is performed firstly from the nodes that transmit with the lower data rate. The purpose of this tactic is to maintain the throughput of nodes to the max possible level without packet drops in order to maintain the performance characteristics of the network, since initially all nodes use the shortest path to the sink. So when a node is informed to stop transmitting through a specific node, it searches in its neighbor table and finds the next available node with the next smaller or even same level and starts transmitting through it. If this node is in the transmitting range of the congested node it is already aware of its condition so there is no problem of routing packets through it again. With this way the congestion hotspot is avoided.

Handling of Powerless nodes

In HTAP special care is taken concerning the nodes whose power is exhausted. These nodes cause major problems to the network in case they are source or relay nodes. Thus, when a node is going to lose its power, it should immediately be extracted from the network and the tables of its neighbors should be updated. This procedure should be simple since it can arise during a crisis state. In the case when the power of a node reaches the power extinction limit, it immediately broadcast this fact to the nodes around it. The nodes receiving this msg update their neighbor tables by removing it from them. If the powerless node is in an active path, the involved nodes apply the alternative path algorithm, when they receive the power extinction message and find an other route to forward packets to sink.

3.2 Topology Aware Resource Adaptation

Topology Aware Resource Adaptation (TARA) is a congestion control algorithm that employs resource control. TARA is based and developed upon a capacity analysis tool using a graph-theoretic approach. TARA is focused on intersection hotspots in permanent congestion situations. Using its model, TARA attempts each time to form a new topology that has just enough capacity to handle traffic during congestion. Although this tool is out of the scope of my work, I will present the three important findings that were extracted using this capacity analysis model.

Firstly, minimizing the path length in a string topology (All nodes are serially placed with a source on one end and a sink on the other) does not increase the capacity if the resulting topology has a path length of more than two nodes.

Second, if the node whose incoming traffic volume is less than the Channel Capacity experiences congestion due to interference with other flows, congestion can be eliminated by rerouting the incoming traffic onto the non-interfered path.

Thirdly, the capacity of a merging topology can be increased by moving the merging point within two hops away from the sink. Having these findings in mind, the designers of TARA propose the following.

As soon as the hot spot node detects that its congestion level is above a threxhold, it needs to quickly locate two important nodes: the Distributor and the Merger. Then, a detour path can be established, starting at the distributor and ending at the merger, as shown in Figure 1. As suggested by the names, the Distributor distributes the incoming traffic between the original path and the detour path with the loads of the two paths being topology-aware.

Congestion detection in the framework of TARA is done by both buffer occupancy and channel load as proposed in [27]. A nodes congestion state is an aggregation of these two metrics. As soon as the congestion level hits the threshold, it declares congestion and becomes a hot spot node.

Traffic Distributor

In order to select the traffic distributor, every sensor node keeps track of the incoming traffic volume from each of its neighbors. This can be achieved by adding one more column in the neighbor table, a data structure that is already maintained by each node in most sensor networks. Having available this information, the hot spot node can select the upstream neighbor that injects the most packets and send an upstream control to that neighbor. If that neighbor node is also congested, it repeats the process until it reaches a node with a low congestion level. That node will then become the traffic distributor.

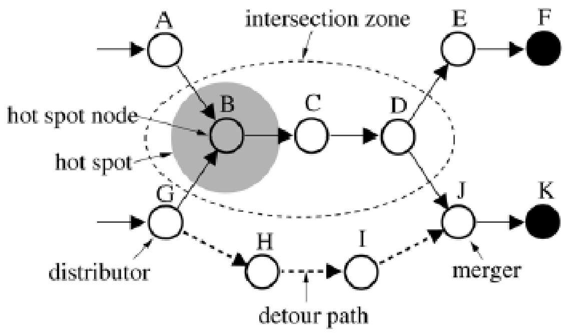


Figure 1: Illustrating the basic concepts of TARA, picture taken from [11]

Traffic Merger

To locate the traffic merger, the distributor traces downstream by sending a downstream control packet to the sink that the most traffic is designed for. Therefore, the merger should be located on the routing path from the distributor towards this sink, with a low congestion level. It is pointed out that congestion level should not be the sole criteria for choosing a merger. The choice of the merger is dependent on the topology of the intersection zone, which includes all the nodes the two intersecting flows have in common. If its the case of merging zones, i.e. two flows heading for the same sink, the location of the merger is still of great importance because it bears a great impact on the resulting capacity, as was noted in the third finding. In order to apply this, each node on the routing path keeps track of the hop distance between itself and the sink. Therefore, as the downstream control packet reaches a node, which contains the desired capacity level, the node can decide whether or not to become a merger based on its distance to the sink.

Detour Path

The merger tries to establish the detour path by locally flooding an REQ packet including the time-to-live (TTL) field, to toward the distributor. A node may then receive multiple REQ packets from a merger due to nature of flooding. To limit the flooding overhead the following optimizations are adopted:

1. A node discards all the REQ msgs if its local congestion is above a threshold.
2. A node discards all REQ msgs if its already on the original routing path.
3. A node decrements the TTL field of the packet before forwarding. It drops the REQ msgs whose value falls below 0.
4. A node keeps track of the largest TTL value it has seen. It drops subsequent REQ msgs with lower TTL values.

Traffc Distribution

To alleviate congestion, the distributor should split the outgoing traffic between the original path and the detour path. TARA adopts a strategy that requires the distributor to check each packets destination sink before routing it. Each detour path has a corresponding sink, and if a packets destination sink does not match the detour path, distributor will send it to through the original path. Among packets that have the matching sink, a weighted fair-share scheduling policy is used to split the traffic between two routes.

To converge into an equilibrium state, TARA needs to ensure that the duration of each resource adaptation is finite and frequent adaptations are prevented. To achieve this, the timer in each node silently expires, except for the distributor node. If TARA fails to create a detour path, the distributor's timer expires and notifies the hot spot node of this failure. Upon receiving this message, the hot spot falls back to the traditional traffic control approach by sending a backpressure message to its upstream neighbors.

3.3 Dynamic Alternative Path Selection

Dynamic Alternative Path Selection scheme is another congestion control and avoidance algorithm that uses resource control to mitigate congestion. DALPaS takes into consideration not only nodes congestion state (both in terms of buffer occupancy and channel load) but is also energy aware. DALPaS is a completely 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. The details of the algorithm are presented below.

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 now they know that they are one hop away from sink. These Level 1 nodes re broadcast this “hello” msg, transmitting in full power. Upon receiving a “hello” msg for the first time, a node adds one to its level and broadcasts it again. Nodes drop following “hello” msgs except if they change to a lower level. I this case they rebroadcast the “hello” msg with its new value. Every time a node keeps in its neighbor table all the connectivity information it overhears. With this procedure, nodes discover each other, build and initialize their neighbor tables and at the same time record the minimum hop distance to the sink.

The neighbor table maintains records for the ID of its neighbors, their buffer occupancy, their remaining power, their hop distance from sink and a Flag field which indicates nodes availability at the current moment. The Flag mechanism is explained in the next section. It is obvious that the neighbor table holds information for all neighboring nodes and not only those which are one hop away. Using this flooding mechanism all possible routes, which can be used to forward packets upstream, are discovered.

DALPaS Mechanisms

After the setup phase where all possible paths from source to sink are 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 sending node ID, as well as the destination (receiving) node ID. When a data packet is received it must be acknowledged. If a node has successfully received a packet, it broadcasts an ACK packet whose header contains the following information:

- Node ID
- Next Packet Sequence Number
- Buffer Occupancy

- Remaining Power
- Number of Hops to Sink
- Availability Flag

All nodes receiving this ACK packet update their neighbor tables with the new values. DALPaS operation scheme involves two stages, soft and hard stage.

1. Soft Stage

   Soft stage is a proactive way of congestion avoidance which also offers balancing of the traffic among many nodes. A node enters DALPaS soft stage condition if it receives packets from more than one flows. In such case, the node is a candidate congested node, in case its receiving rate exceeds its transmitting rate. In order to prevent this situation DALPaS algorithm attempts at first to keep each node receiving data from one flow. To achieve this, the candidate congested 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 is informed that it should check for next suitable path starting from another node at the same level as before. The sending node understands that this inhibition was caused by another flow, since the Flag field in the ACK packet remains "true". In soft stage, receiving nodes just advice the sending nodes to find another path. They keep forwarding the data they receive. In soft stage, receiving nodes prefer the flows with the higher data rate

2. 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 for an inhibited node to decide to keep sending packets to the same node. In this case, if the "performance threshold" is exceeded the node enters in hard stage. Hard stage is a situation where the network forces the flows to change routing paths since "performance threshold" is exceeded. Responsible to monitor and apply performance thresholds id the Flag Decision Algorithm, which is described below.

3. Flag Decision Algorithm

   Flag Decision Algorithm runs when a node enters in hard stage. In this case 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 reseans:

   - Buffer Occupancy is reaching its upper limit. In this scenario, when the receiving node senses that its buffer will soon overflow, it sets its Flag field in the next ACK packet that will broadcast, to "False". The nodes in the area that receive this packet, alter immediately the flag field for the concerned node and look in their neighbor table for the next available hop to pass their data. When the failed node recovers, it broadcasts a hello message stating this event and neighboring nodes change its flag entry back to "True".
   - Low Remaining Power. Each node is programmed to set its flag to false every time its remaining power falls below a certain percentage if the total. Again, as before, the node changes its availability flag to false and informs its neighbors via an ACK message and the same procedure applies as in the buffer occupancy case. The limit of power can be chosen according to applications requirements and objectives.
   - Higher level Node unavailability. In this case, although a node is available with respect to buffer and power, there is no other node available to forward the data to. In this case,

the node informs the neighboring nodes using its availability flag again. This function protects the network from forwarding packets in black holes.

4. 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 availability (Flag) and the number of hops to the sink. Using this tactic each node can, in an easy simple way, find the next node with minimum computation. It just sorts the number of available nodes in ascending order with respect to their distance from sink, and forwards the packets through the first node in the list. If the first node becomes unavailable for any reason the sender chooses the next node from the list. In case that more than one nodes are in the same level, the algorithms decides based on the nodes available power. Finally, if there still exists a tie, the algorithm picks up the node with least 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. Using this method, the algorithm gives priority to the maintenance of performance metrics like the average delay, as well as to the networks uniform energy utilization, thus preventing the creation of holes.

Chapter 4

Implementation and Evaluation

4.1 Description of the Simulator

To evaluate the performance and compare the three employed algorithms we used the Prowler simulator and the RMASE application. In this section we will describe these tools and how we used them to build the algorithms under study.

4.1.1 Prowler

Prowler is a probabilistic wireless network simulator, developed in MATLAB [1]. Although Prowler provides a generic simulation environment, its current target platform is the Berkeley MICA mote running TinyOS which is widely used in the current field of research (Routing Protocols development for WSNs). It supports an event-based structure similar to TinyOS/NesC that makes it easy to port an algorithm back to the hardware platform. The probabilistic wireless network simulator (Prowler) is an event-driven simulator that can be set to operate in either deterministic mode (to produce replicable results while testing the application) or in probabilistic mode (to simulate the nondeterministic nature of the communication channel and the low-level communication protocol of the motes). It can incorporate arbitrary number of motes, on arbitrary (possibly dynamic) topology, and it was designed so that it can easily be embedded into optimization algorithms. The simulator runs under MATLAB, thus it provides a fast and easy way to prototype applications, and has nice visualization capabilities. The network simulator models the important aspects of all levels of the communication channel and the application. The nondeterministic nature of the radio propagation is characterized by a probabilistic radio channel model. A simplified, but accurate model is used to describe the operation of the Medium Access Control (MAC) layer. The applications interact with the MAC layer through a set of events and actions [25].

The Target System

Prowler was designed to simulate a very successfull, low-cost prototype field node (mote) family that was developed at Berkeley. The used variant (MICA) includes an 8-bit, 4 MHz Atmel ATMEGA103 microcontroller, 128k Program Memory, 4kB RAM, and an RFM TR1000 radio chip capable of providinh 50kbps transmission rate at 916.5MHz. The motes can also accomodate a set of interchangeable sensors (temperature, light, magneto, sound, etc).

Radio Propagation Models

The radio propagation model determines the strength of a transmitted signal at a particular point of the space for all transmitters in the system. Based on this information the signal reception conditions for the receivers can be evaluated and collisions can be detected. The signal strength from the transmitter to a receiver is determined by a deterministic propagation function (modeling the decay of signal strength with distance), and by random disturbances (modeling the fading effect, the time-varying nature of the signal strength, and other miscellaneous transmission errors).

$$P_{rec.ideal}(d) = P_{transmit}\frac{1}{1+d^{\gamma}} \tag{1}$$

where $P_{rec.ideal}$ is the ideal reception signal strength, $P_{transmit}$ is the transmission signal power, d is the distance between the transmitter and the receiver, and γ is a decay parameter with typical values of $2 \leq \gamma \leq 4$. Real signals, however, behave in a much different manner. The signal strength can vary very heavily as distance changes. Also in time the signal strength can change even if the distance between the transmitter and receiver is constant. This fading effect is modeled by random disturbances in the simulator. The received signal strength from node j to node i is calculated from the propagation function (1) by modulating it with random functions:

$$P_{rec}(i,j) = P_{rec.ideal}(d_{i,j}) \cdot [(1 + a(d_{i,j})) \cdot (1 + \beta(t))] \tag{2}$$

Parameters a and β are random variables with normal distributions $N(0, \sigma_a)$ and $N(0, \sigma_\beta)$ and adjustable parameters σ_a and σ_β. An additional parameter p_{error} models the probability of transmission errors caused by any unmodeled effects (e.g. external disturbances, unreliable hardware, etc).

For signal reception and collisions three models exist. In the first one the signal is received if the signal strength is greater than a reception limit parameter. The channel is sensed idle if there is no signal that could be received. There is a collision if two transmissions overlap in time and both could be received. In the second mode each receiver has a noise variance parameter. Signal to Interference Ratio is measured and the signal is received if the SINR at the receiver is greater than the reception limit during the whole length of the transmission. The channel is sensed idle if the total signal strength is smaller than an idle limit, which depends on the noise variance of the receiver. There is a collision if the SINR at the receiver becomes smaller than the reception limit at any time during the reception. The third model is similar to the second one with the difference that is using the Rayleigh function for the fading effect. The first model is simple and fast in calculations, while the two others are more accurate. In our simulations we chose the first model mainly for computation purposes.

MAC-layer model

The MAC layer communication is modeled by a simplified event channel. It is a simple CSMA MAC layer. When the application emits the *SendPacket* event, after a random *WaitingTime* interval the MAC layer checks if the channel is idle. If not, it continues the idle checking until the channel is found idle, before each idle check waiting for random intervals characterized by *BackoffTime*. When the channel is idle the transmission begins and after *TransmissionTime* the application receives the *PacketSent* event. After the reception of a packet on the receivers side, the application receives a *PacketReceived* or *CollidedPacketReceived* event, depending on the success of the transmission. The *WaitingTime* and *BackoffTime* parameters are random uniformly distributed variables in predefined intervals, while *TransmissionTime* is constant (i.e. all messages have the same length).

The application level

The applications are event-based, similarly to the real TinyOS framework. In the simulator the following events can be caught: *InitApplication*, *PacketSent*, *PacketReceived*, *CollidedPacketReceived*, *ClockTick*. The application can activate the *SetClock* and *SendPacket* actions which cause further events. A few debugging and visualization commands are also available, e.g. switch on/off the LEDs on the motes, draw lines and arrows and print messages.

4.1.2 Rmase

Rmase has been developed for dealing with the challenge of comparing different routing algorithms for sensor networks [26]. Rmase consists of a network topology model, an application model, and a performance model. Rmase also supports a layered routing architecture for plug-and-play common routing components. Many applications have been modeled in Rmase, and an increasing list of routing algorithms and components has been developed. Rmase has been used to develop new routing algorithms, to analyze performance trade-offs, and to select the best routing algorithm for a given application and its performance requirements. Rmase is implemented as an application in Prowler.

A component handles events (e.g., PacketReceived, PacketSent, ClockTick) and executes commands (e.g., SendPacket). By default events are passed up and commands are passed down the layers. However, one can stop the flow by setting variable pass to 0 (false) for a command or event.

The minimum configuration of layers for Rmase includes the MAC layer, a routing layer, and the application layer, with the MAC layer at the bottom and the application layer at the top. The application layer is for generating the application scenarios and commands SendPacket, the routing layer will forward packets received for other nodes or passes the received packet up if it is the destination of that packet, and the MAC layer will actually send the packet out. A packet shall arrive at the application layer if and only if the node is a destination. The MAC layer simulates the TinyOS MAC layer functionality. A packet can be sent to the next hop address (data.address = nodeID) or broadcast (data.address = 0). One can also set the communication mode to promiscuous, so that all neighbors can hear a packet, even if the packet is sent to the next hop node. The MAC layer also simulates energy consumption with a parameterized energy consumption model for transmitting, receiving, and idling states. A node can also switch between sleep and idle states and no energy is consumed in sleep state.

The statistics layer is added on top of the application layer for collecting routing statistics (e.g., the total number of sent packets, the total number of received packets, the time delays of received packets, etc.), for calculating the performance metrics.

Furthermore, one can have a set of control layers between the routing layer and the application layer, and a set of support layers between the MAC layer and the routing layer. The control layers are for network initialization or maintenance, e.g., building and maintaining a routing table or a spanning tree. The support layers are for adding features such as transmitting and receiving queues, data aggregation/fragmentation, neighborhood management, confirming/delaying transmissions, and checking duplications, etc. It is the algorithm designer's choice to put individual functions at different layers so that common functions can be shared by different algorithms.

Common Routing Componentsl

This layered architecture allows for the sharing of common components for different algorithms. For example, many algorithms need neighborhood management and transmission queues, single path routing may need confirmation for reliable transmission, and flooding-based routing may add transmission delays to reduce collision. Thus, for a given application scenario, one shall select not only the routing algorithm, but also the support and control components from the component repository. Table 1 shows a list of support and control components implemented in Rmase.

For the implementation of the three algorithms in Rmase I took advantage of some support and control components that already exist in the application. Beside this the routing mechanisms, congestion detection and resource control were created from scratch by me, according to each algorithm. All the source code with comments is available at Appendix A.

4.1.3 Reasoning on the choice of the Simulator

Prowler/Rmase is a simple to use, event-driven simulator built in MATLAB. It satisfies all non-fuctional and functional requirements for simulators as described in [10]. Although NS-2 simulator is the most commonly used simulator in the bibliography, we chose not to use it for a number of reasons.

Name	Type	Definition	Parameters	Affected data fields
Probabilistic Faulty Model	support	model the probabilities from alive to dead and from dead to alive	FailProb, WakeupProb	
Transmission Queue	support	maintain a transmission queue		
Aggregation Queue	support	pack a maximum of M packets from the queue into one packet to transmit. On the receiving side, a packed packet will be unpacked into the original packets	AggregatePackets, AggregateTimeout	
Maximum Hops	support	bound the maximum number of hops in transmission	maxhops	
Neighborhood Management	support	maintain a neighbor list	from	
Transmission Confirmation	support	trigger a timeout event when no confirmation is heard within a certain time	TransTimeout	
Duplication Check	support	mark duplicated packets		SeqID, source, duplicated
Transmission Delay	support	delay a packet transmission according to the timeDelay field of data, set by upper layers, and transmit a forwarding packet probabilistically according to the number of times the packet has been heard	randOff, minDelay	SeqID, source, duplicated
Hello Initialization	control	broadcast a couple of "hello" packets at a certain interval from each node		msgID<0
Backward Initialization	control	flood from the destination to set up hop counts or to build an initial spanning tree, which are used by many routing algorithms		msgID<0

Table 1: Common Routing Components in Rmase application

While NS-2 [15] is a powerful network simulator, some researchers claimed it has some drawbacks for simulating wireless sensor networks [13, 29]. An object-oriented design in ns-2 may introduce some unnecessary interdependence between modules which may lead to difficulties in a new protocol addition [29]. Additionally, ns-2 has been developing for about 10 years; it has quite a large and expanding infrastructure which leads to a steep learning curve for a novice user.

Furthermore, Prowler is oriented to then MICA mote platform, widely used for real implementations in the field of study on WSNs. It's event-based architecture makes it easy to port a routing scheme directly on real hardware. Also, the library of common routing components in Rmase, helped our work in part and provided some useful ideas.

Finally, although more simulators exist, like GloMoSim [29] or TOSSIM[13], I decided on Prowler/Rmase because of previous experience with the MATLAB environment. The reader is directed to [30] for the obvious efficiency of a high - level simulator like Prowler/Rmase.

Inspite of the mentioned above advantages, we should also note the most important drawback of Prowler/Rmase and this is the lack of documentation and bibliography.

4.1.4 Implementation of Algorithms in Prowler-Rmase

In this section we will briefly describe the steps we followed to embody the algorithms in Rmase application. As was already mentioned Rmase offers a packet of common components to its repository. Two of these components were used 'as is' in all three algorithms. These are the neighbor layer and the transmit queue layer. The first one maintains a list of the neighboring nodes and it can be extended so we can store some information about them and use it in our routing scheme. The second keeps a queue of outgoing messages and was used to simulate the buffer of each node.

For TARA we did not use any other component offered by Rmase. Instead, we created an other layer, named TARA layer, and all the routing procedure is inside. It consists of a switch command which guides the node's behavior on each event occurrence. All the control packets required by the algorithm to locate the distributor, the merger, and for route establishment are created within the TARA layer. The channel loading is also calculated in this layer and combined with the queue transmit layer the congestion detection metric can be derived. Finally, the setup phase is included in this layer. The parameters that can be set in TARA are the buffer size and the upper watermark (min limit to react in a congestion event). For the sensing of channel load, we used the values as in [27] but in a different way. Since Rmase keeps track of all the transmission and reception times, instead of sampling in a time period of $100msec$, we measure the number of received packet for a particular node, in the last $100msec$, or 4000 if we convert them to simulation time. Here it is noted that nodes receive all packets that are transmitted within their sensing range.

Concerning the other two algorithms we found useful an other control layer offered by Rmase.

This was the ack retransmit layer. Although we did not use it as is (since these algorithms do not use any retransmission mechanism) it helped me create the ack layer used to transmit the ACK messages required by HTAP and DALPAS. For DALPAS, the rest of the calculations are done in the DALPAS layer. These include the setup phase, Soft and Hard Stage states and the Flag Decision Algorithm function.

Regarding HTAP, an other control layer from Rmase was used. This is the init backward layer and it was used for the initialization phase. Along with the transmit queue layer, the ack layer, and the neighbor layer, HTAP algorithm is completed with the HTAP layer. All the functions of the algorithm are coordinated and controlled in this layer. Finally, all three algorithms were simulated over the same MAC protocol, which is a simplified CSMA/CA layer as it was described earlier.

4.2 Simulation Results

4.2.1 Metrics

Various performance metrics are used for comparing different routing strategies in sensor networks. Prowler provides the following and these were used for optimization and comparison purposes:

Latency: Time to send a message from source to destination. For any destination, if n packets have arrived, latency for that destination is given by $\sum \frac{d_i}{n}$, where d_i is the latency of the ith packet. The latency of the network is then averaged by the number of destinations.

Throughput: Number of messages per second received at destination. The throughput of the network is the sum of the throughputs from all destinations.

Success Rate: The total number of packets received at all the destinations vs. the total number of packets sent from all the sources. It measures the overall success of the network

Loss Rate: Number of lost packets vs. the total expected number of packets for that destination. It is useful to measure quality (e.g., more dropped messages result in lower sensing resolution).

Energy Consumption: 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) and idling (Pi). Assume the transmission power level be L, and the time for transmitting, receiving and idling be Tt, Tr and Ti, respectively, the total energy consumption of a node is: $PtTtL + PrTr + PiTi$.

Energy Efficiency: Ratio between the total number of packets received at the destinations vs. the total energy consumption in the network.

LifeTime Prediction: $E - (\bar{u} + \sigma)$, where E is the total initial energy at each node (full battery charge), $\bar{u} = \sum_i \frac{u_i}{N}$ s the average used energy, N is the total number of nodes in the network, and $\sigma^2 = \frac{(u_i - \bar{u})^2}{N}$.

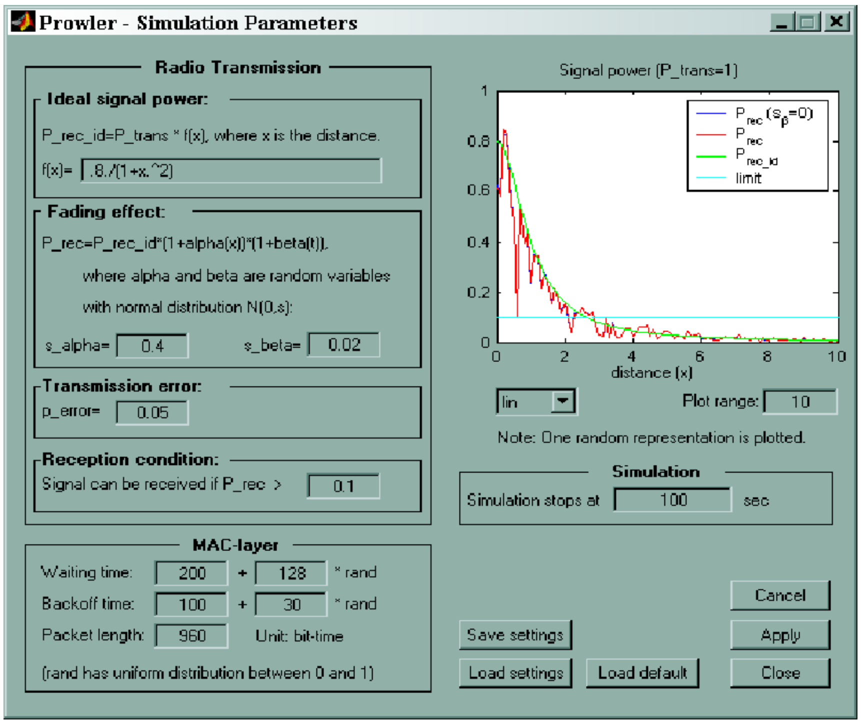
4.2.2 Simulation Environment

The following data apply in the simulating environment.

Radio Propagation

The signal propagation function is $\frac{1}{1+x^2}$. The reception limit is 0.1. The probability of transmission errors is 0.05. The topology variance of the radio transmission signal is 0 and the random variance of the transmission signal is 0.02. In this way the fading effect is nullified, which reduces the radomness of the radio propagtion. This minimizes the required repetitions of a simulation run and provides clearer results for the sake of comparison. MAC Layer minimum waiting time is 200, additional maximum random waiting time is 128, minimum back off time is 100 and maximum random backoff time is 30 (all values in simulation time). The length of a packet is 200 bits, which corresponds to 25 bytes, and the bit-time in seconds is $\frac{1}{40000}$.

Topology


Prowler - Simulation Parameters

Radio Transmission

Ideal signal power:
P_rec_id=P_trans * f(x), where x is the distance.
f(x)= .8./(1+x.^2)

Fading effect:
P_rec=P_rec_id*(1+alpha(x))*(1+beta(t)),
where alpha and beta are random variables
with normal distribution N(0,s):
s_alpha= 0.4 s_beta= 0.02

Transmission error:
p_error= 0.05

Reception condition:
Signal can be received if P_rec > 0.1

MAC-layer
Waiting time: 200 + 128 * rand
Backoff time: 100 + 30 * rand
Packet length: 960 Unit: bit-time
(rand has uniform distribution between 0 and 1)

Note: One random representation is plotted.

Simulation
Simulation stops at 100 sec

Figure 2: Prowler Radio Parameters

The topology used for the simulation is a 10 by 10 grid with 0.1 random offset values in both axes, to introduce a small ammount of nondeterminism. The transmit signal strength is set to 0.7, in Prowler parameters, and the maximum radio range is about *2d*, where *d* is the standard distance between two neighbor nodes in the grid. This is shown in Figure2. Sources are placed in the left down corner while the sink is placed at the top right corner in order to create convergence conditions. Each node's buffer is capable of holding 15 packets maximum. In all simulations

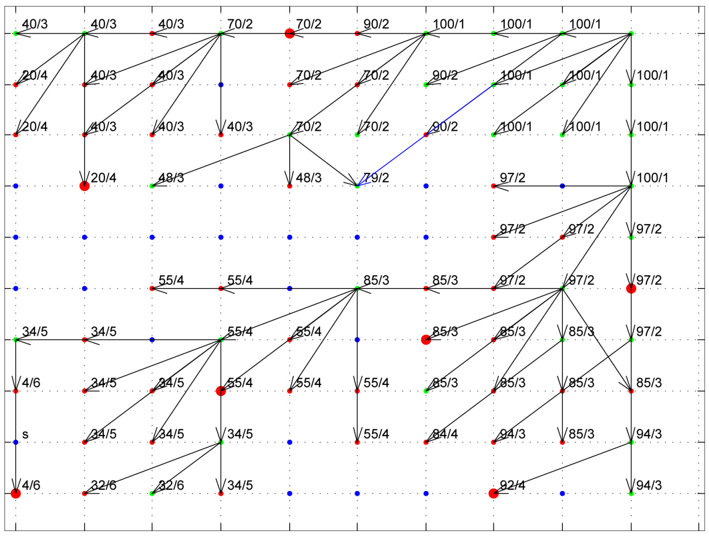


Figure 3: Instance of initialization phase

applies the following traffic pattern: At the beginning sources do not transmit any packets but the initialization phase takes place. After approximately 1 second all sources begin to transmit their data with the assigned data rate. The whole simulation session corresponds to 10 seconds. Each simulation run has been performed 20 times and average results have been extracted.

Note that parameters in Prowler (like Signal Strength) and metrics in Rmase (like Energy Consumtion) are given in plain numbers and not in their real measurement units (i.e. dB and Watt or Joule). This is due to the fact that these tools were built for algorithm routing schemes optimization and comparison purposes. Moreover, *sensing range* is not defined in Prowler since sensing events are not simulated. Data Sources are chosen deterministically or using a probabilistic function in a prespecified area in the network.

Figure 3 shows an instance of the connectivity during the initialization phase. In Figures 4, 5, 6 and 7, snapshots of the three employed algorithms are shown. In figure 4 we can see the connectivity of the network after the initialization phase. Sources are placed at the down-left

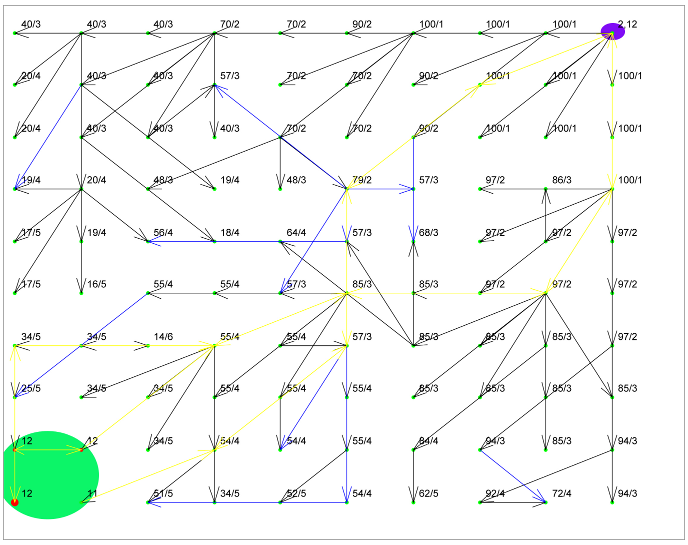


Figure 4: Sources and Sink placement

corner and the sink at the upper-right corner. Yellow arrows represent paths on which real data travels from source to sink.

In figure 5 we can see the alternative path creation in DALPaS algorithm. In this figure, 3 nodes are marked. The green node informs the orange node that is receiving data from another flow simultaneously, so the orange node decides to change its next-hop node with the blue node. Figure 6 show the alternative path creation for HTAP algorithm. The blue node has reached it's Buffer Threshold and informs neighboring nodes with an ACK message (red arrows). Therefore, the green node choses the grey node as its next-hop. The last example, shown in figure 7, shows the behavior of TARA. In this case, the blue node is the *Distributor*, which splits traffic between the original path (yellow arrows towards sink), the established *detour path* is the green path and the *Merger* is the Sink.

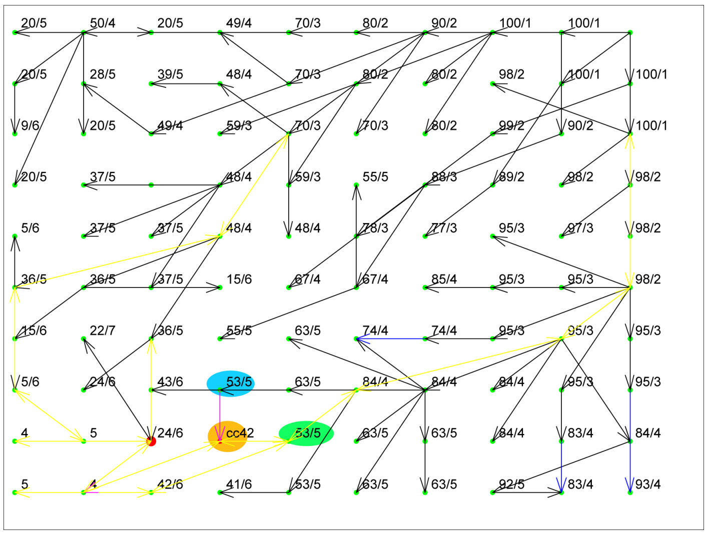


Figure 5: DALPaS alternative path creation

4.2.3 Results

In the first series of simulation tests, we examine the behavior of each algorithm under different source rates. Figure 8 represents the results. As the Data Rate of the generated packets is increased we notice that the *success rate* starts to fall. This is an indication that the network becomes overloaded and packets are dropped either due to collisions or due to full buffers. This also means that the available resources of the network are fully utilized. Next figure (Fig.9) presents the throughput of the network as data rate increases. Although the percentage of received packets is decreasing the actual number of packets that are received by the sink is increasing. This is a strong indication that resource control algorithms can provide sink with a big number of data packets even in overloaded situations. In order to check how the number of resources affect the performance of the algorithms we keep the networks data rate constant and in each run we increase the number of nodes. Figure 10 indicates that as the number of nodes in the network is increasing, Resource Control algorithms improve their performance and ara able to deliver bigger nuber of packets to sink.

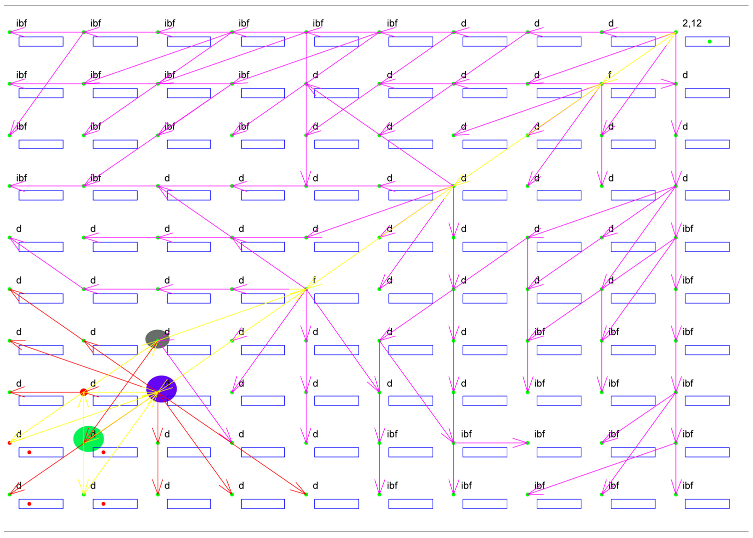


Figure 6: HTAP alternative path creation

Finally, comparing the three algorithms together in a highly congested environment, we obtain the following results, shown in Figure 11. During this simulation we increased the source nodes to five and we scattered them probabilistically in an area of radius *4d* around point *(0, 0)*, which is the bottom left corner. This setting results in a total generated packet rate of 25 packets per second.

The first graph in Figure 11 shows the delay with respect to simulation time. We can see that DALPaS and HTAP demonstrate almost a constant delay during the simulation run. This smooth behavior is an indication of robustness and is due to the fact that the next hop selection is done using minimum computation at each node.

On the other hand, TARA demonstrates a more complex pattern. During the first stages of the simulation run and while buffer occupancy is low, TARA achieves better results in terms of delay. As time passes and buffers fill near to repletion, TARAs resource control mechanism is initiated and a dramatic rise in delay is observed. This indicates the much higher overhead introduced by TARAs method trying to locate a Distributor and a Merger.

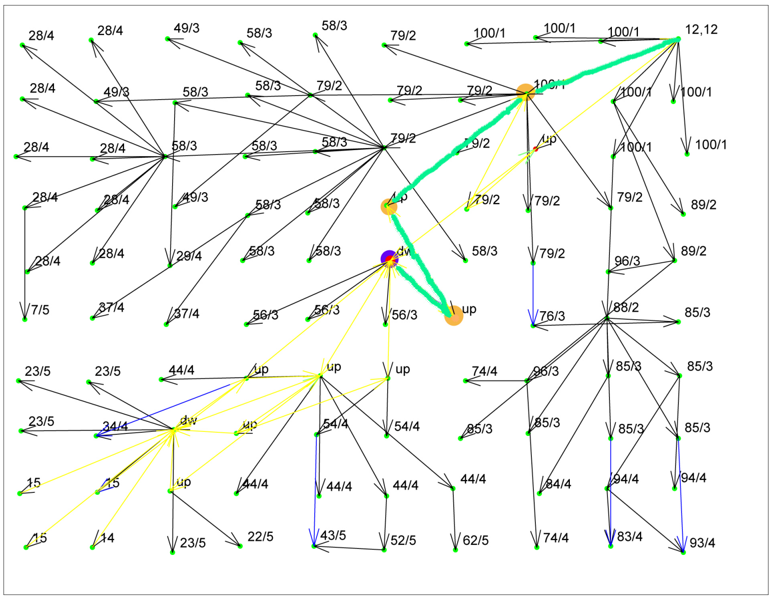


Figure 7: TARA alternative path creation

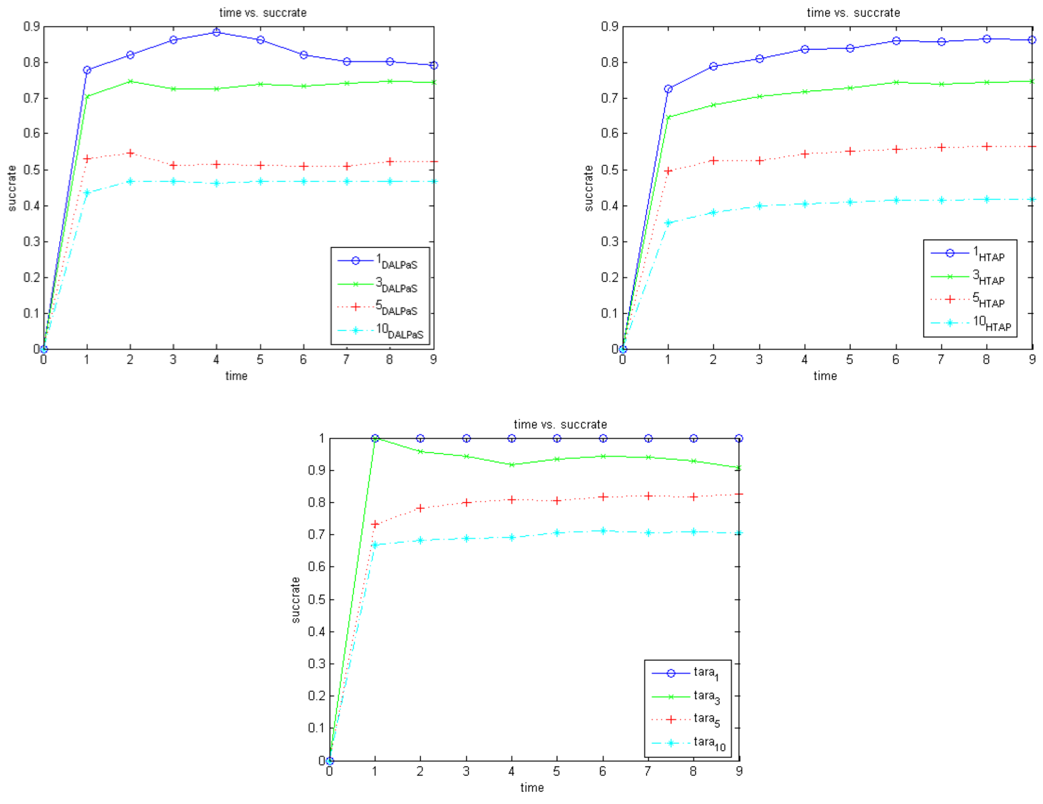


Figure 8: Success Rate with increasing Data Rate

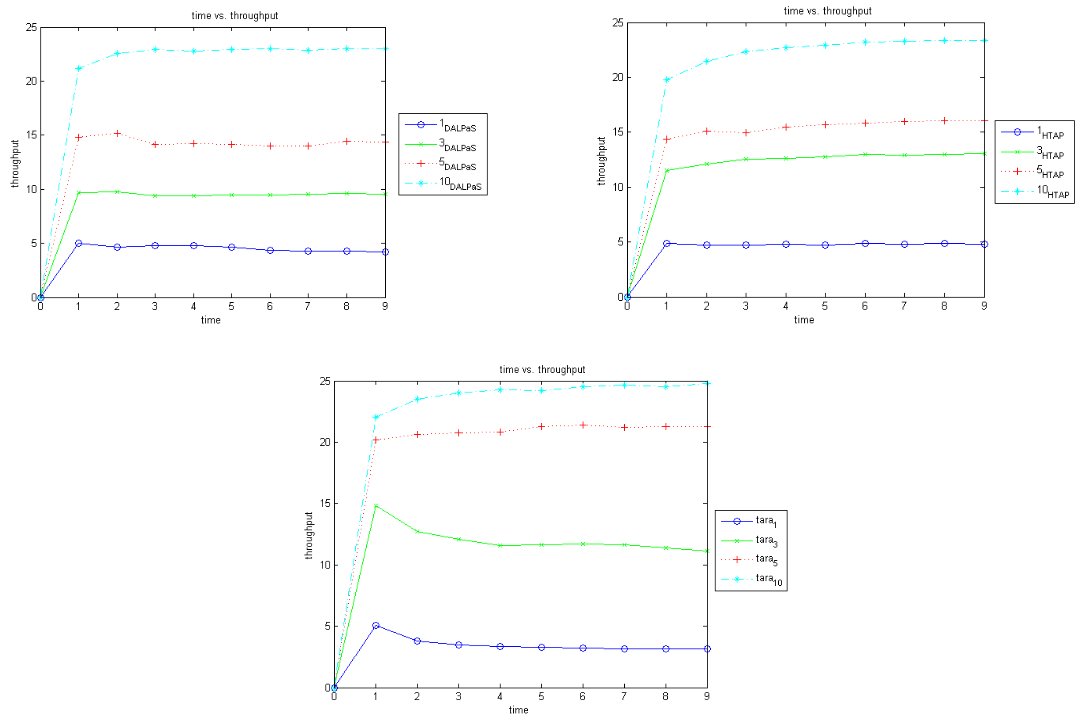


Figure 9: Throughput with increasing Data Rate

TARAs method has also a noticeable impact in success rate. We conjecture that this is a direct consequence of the extra control packets injected into the, to locate and establish an alternative path towards the sink. And all these during the crisis state when the medium is highly loaded with traffic and buffers are close to saturation. This tactic definitely leads to additional collisions and packet drops.

Consequently, since throughput measures the received packets at destination per second, the throughput figure is analogous to success rate. Energy efficiency is also defined as a product of success rate and logically follows a similar pattern.

Additionally, it is clear that TARA consumes more Energy in total and this is also due to the algorithms additional overhead. HTAP has the lowest energy consumption. DALPaS lies between the two mentioned above algorithms. DALPaS sends an acknowledgement message above each

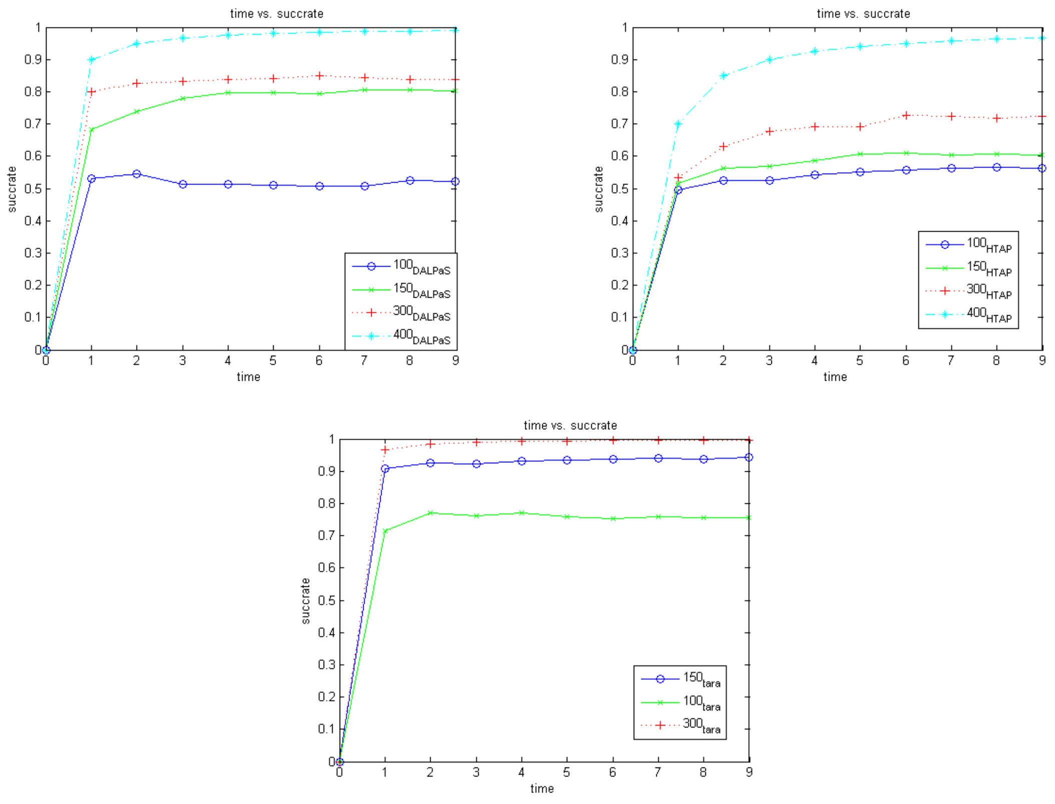


Figure 10: Success Rate with increasing Number of Nodes

reception, while HTAP uses a 2-way handshake for flow establishment. This is the reason why initially HTAP seems to consume more energy but as time passes DALPaS energy consumption rises higher.

What is very interesting is the Life-time prediction which is close related to energy consumption. As it was explained earlier, this metric measures the effect of energy efficiency and load balancing on the routing application. It can be obtained at any time to predict the lifetime of the network. From this metric, one can see that if two algorithms (A and B) use the same amount of energy, but B uses energy more evenly across the network, then B has a longer lifetime. In our case, DALPaS consumes more energy than HTAP but their lifetime prediction figure is the same which leads us to the conclusion that DALPaS uses energy more evenly in the network.

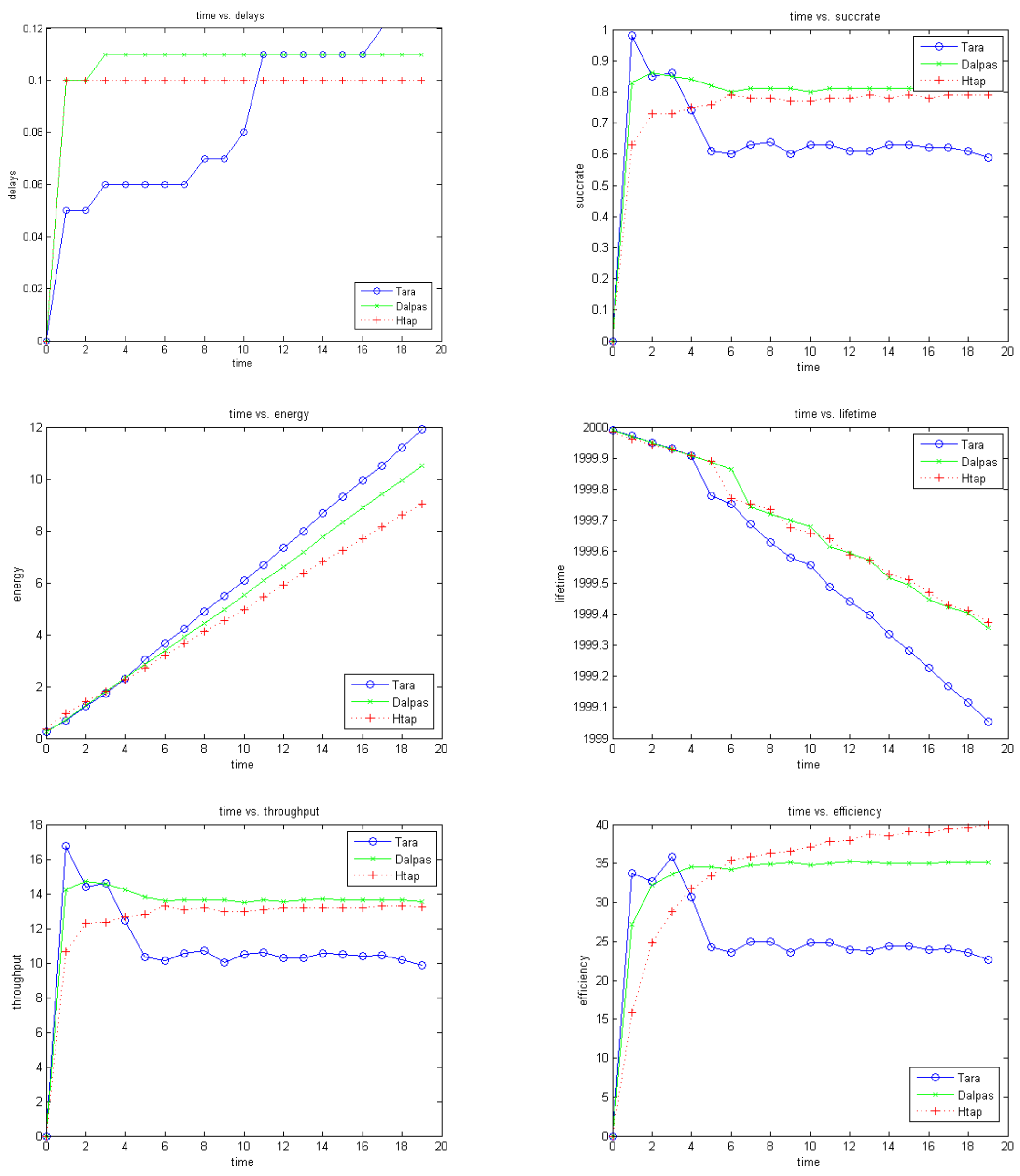


Figure 11: Comparison of DALPaS, HTAP and TARA in a higly congested environment

TARA performs poor it this metric and this is because of higher total energy consumption as it was explained before. Also, HTAP and DALPaS perform better load balancing in the network.

It is obvious that the simplicity of DALPaS and HTAP achieve much better results against TARA. The method of TARA, trying to locate a Distributor and a Merger during a crisis state, has a negative impact on its performance. Although during lower congested states TARA can achieve better performance (Figure 8), during a higher congestion state is not sufficient enough.

Chapter 5

Optimizations

In [11], the authors try to figure out an adaptation to their algorithm to make it resilient to transient congestion. To achieve this they tried to set the *upper watermark* value which is the threshold that activates the mechanisms to allieviate congestion. In their work Kang et al. [11] proposed a transient threshold that adapts to transient congestion. Since our simulation scenario involves a different congestion situation we tried to optimize TARA's reaction in this case. For this reason we examined the behaviour of the algorithm with different *upper watermark* values. The results are shown in Figure 12.

It is obvious that in the congestion scenario we examine, higher value of the *upper watermark* produces the best results in terms of throughput and energy consumption. This is a consequence of the fact that the Congestion Detection mechanism of TARA is very sensitive in high traffic loads and initiates the congestion mitigation mechanisms even when the buffer occupancy of the nodes is at low levels. Due to the fact that traffic is constantly high in the scenarios we are interested in, this results in continuous attempts of creating alternative paths from the majority of the nodes in the network and not only from those who are truly affected by congestion. Thus, setting a higher

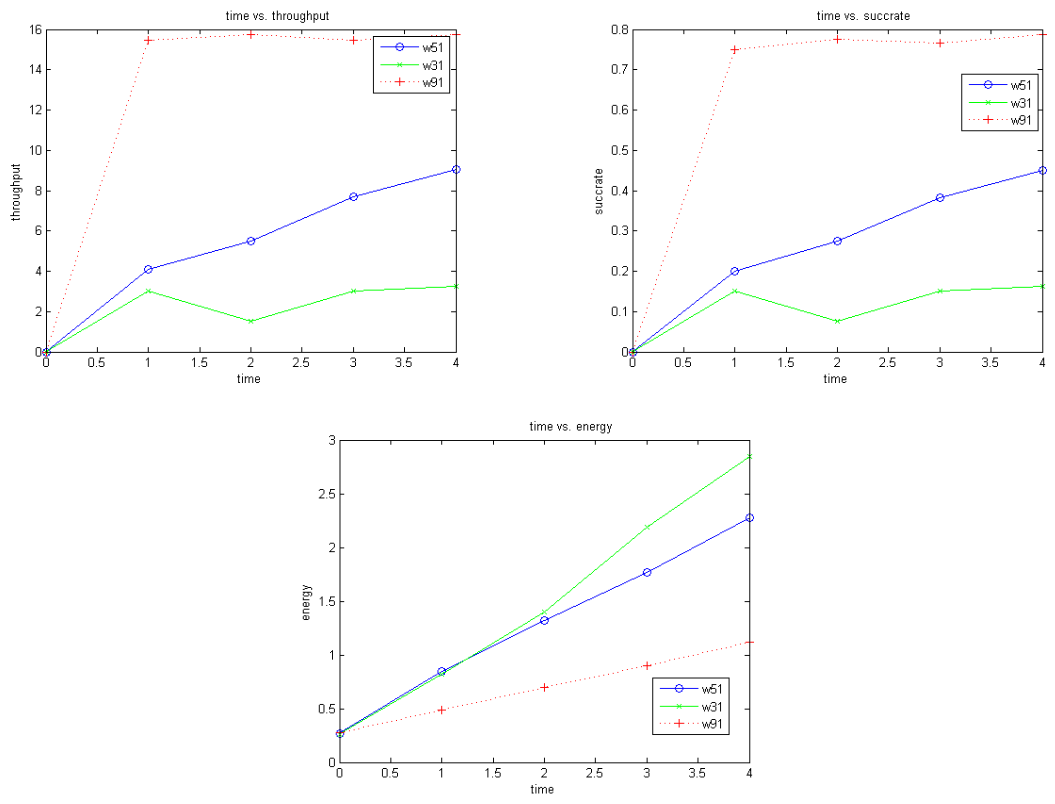


Figure 12: TARA: Throughput, Success Rate and Energy consumption for different values of UpperWatermark (0.31, 0.51 and 0.91)

threshold, only the truly affected nodes are involved in the alternative path creation and better results are achieved.

Having this in mind, we tried to optimize HTAP on the same basis. Hence, we studied its behaviour under different *Buffer Thresholds*. Results are shown in Figure 13.

With Buffer Size equal to 15, if we place the Threshold to initiate congestion control at 8, we can achieve optimum results. If we set the limit higher, congestion is detected too late to achieve proper mitigation. On the other hand, lowering the threshold initiates the congestion control mechanism too early, thus causing needless use of additional nodes. Throughput and Success Rate are maximized and, as expected, Energy Consumtpion is higher. Nevertheless, we can see that Energy Efficiency is better when *Buffer Threshold* is lower.

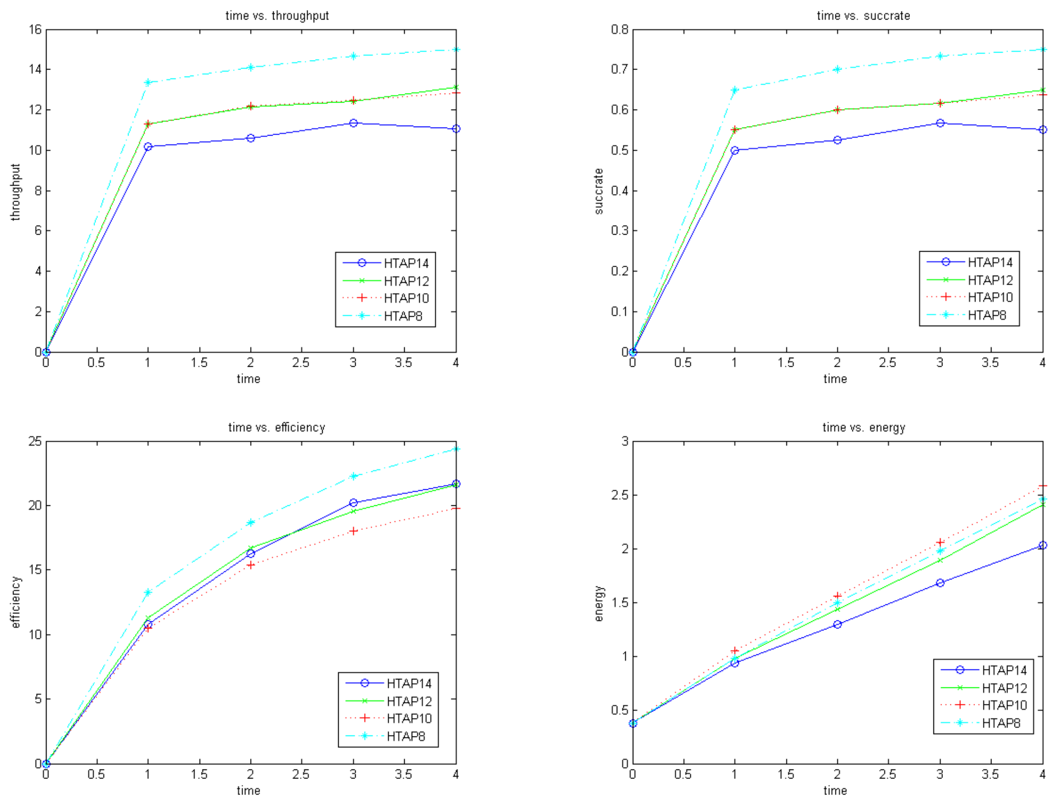


Figure 13: HTAP: Throughput, Success Rate, Energy Efficiency and Energy Consumtpion for Different values of Buffer Threshold (8, 10, 12 and 14)

Concerning DALPaS, similar results are expected for the *Buffer Threshold* value. Instead of repeating the simulation test for DALPaS's Buffer Threshold, we tried something different: To mimic HTAP's Topology Control Algorithm without changing the simplicity of DALPaS. For this purpose, we reduced the transmitted signal strength, during only the initiallization phase. Less transmitted power will result in better quality links, thus aiding the overall use of the congestion mechanisms in the network. Results are dislpayed in Figure 14.

It is obvious that reducing the transmitted signal strength during the initiallization phase to 0.8 of maximum power, produces better results for Throughpout and Success Rate. Concerning Energy Consumption, less transmitt power can result in more hops from source to sink, thus more energy utilization but also in this case we can see that Energy Efficiency is better.

For DALPaS, it is also possible, and very easy to apply, a second term in the Performance Threshold parameter. Instead of using only Buffer Occupancy as a performance metric, we added also a Delay parameter (in Prowler, it is possible for each node to have knowledge of the total delay in the network). When a node chooses to change it's path towards the sink because of congestion, this can result in a long path that introduces high delay in the overall perfomance. Using the proposed adaptation a node may chose to change it's alternative path again until the Delay Performance parameter is also sattisfied. The results are shown in Figure 15.

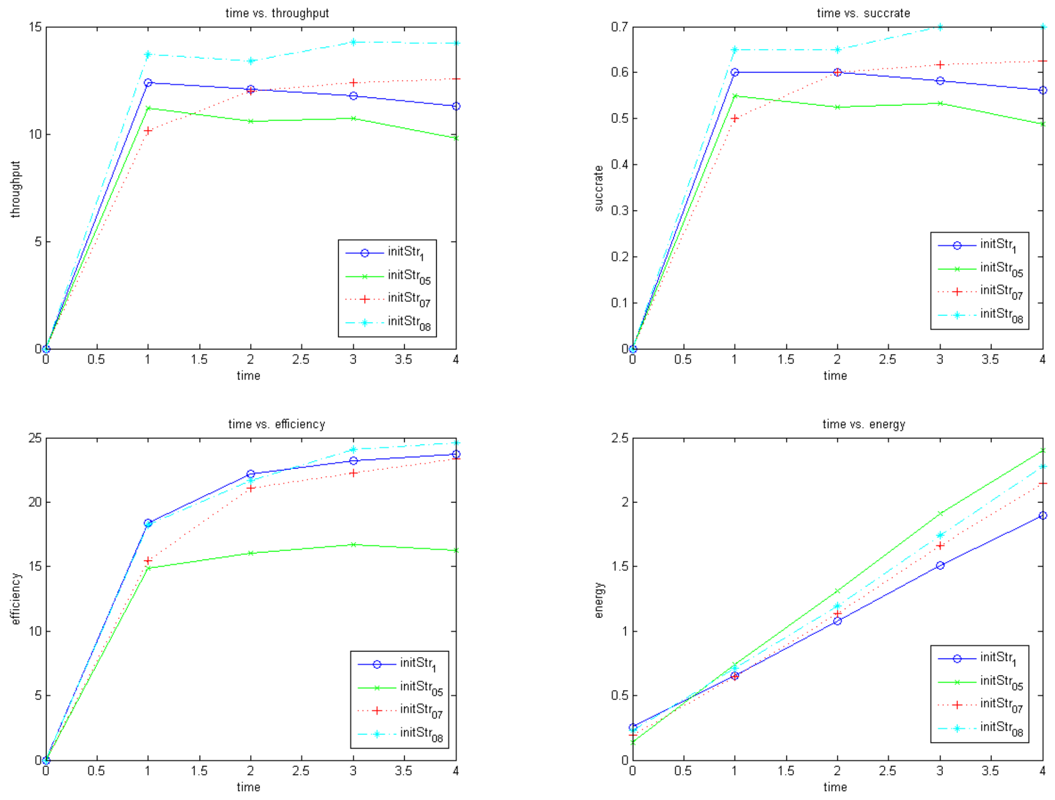


Figure 14: DALPaS: Throughput, Success Rate, Energy Efficiency and Energy Consumtpion for Different Init hello Tx Power

We can see (Figure 15) that not only the delay is reduced, but also throughput and subsequently energy efficiency are affected positively with the combined performance metric. Even without knowing which are the best paths to follow toward the sink(s), by using this metric as a threshold nodes are forced to adapt and satisfy the application's demands and Quality of Service.

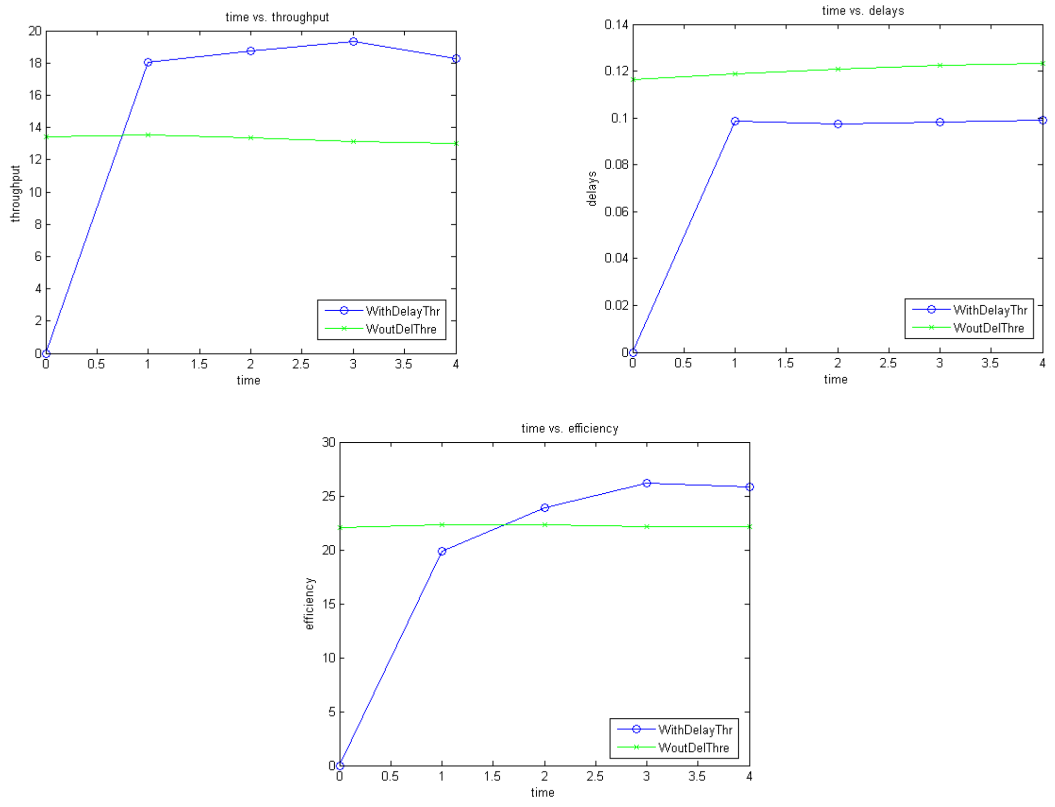


Figure 15: DALPaS: Throughput and Energy Efficiency using a combined Performance Metric that includes Delay

Chapter 6

Conclusions and Future Work

In this thesis we investigated about congestion control in wireless sensor networks and methods against it. The special nature of sensor networks provides a very challenging field of research on how to deal with congestion. In this work we have focused on schemes that counteract congestion using Resource Control methods. Contrary with Traffic Control schemes, which try to reduce the load in order to mitigate congestion, resource control methods can satisfy the application fidelity requirements using alternative paths towards the sink. For this purpose, we have studied and presented three algorithms, proposed in the literature. Topology Aware Resource Adaptation (TARA), which is considered a resource control algorithm, Hierarchical-Tree Alternate Path (HTAP), and Dynamic Alternative Path Selection (DalPas). They were implemented using a simulator in MATLAB, Prowler and Rmase, which was specifically designed to study routing algorithms for sensor networks. Using the data available from the bibliography, we evaluated the three different designs, studied their behavior, and compared them under parameters such as traffic load, network density and topologies. Each algorithm has a unique design, thus they act differently in different frameworks. Using the results and with the experience we had gained while studying the algorithms and from their implementation in the simulator, we achieved to tune the algorthms to an optimized level. We also noticed that small and simple changes in the intrinsic parameters of each algorithm can have a great effect on its behaviour.

In the future, further study could be performed, including different topologies in the simulations and more algorithms from those proposed in the bibliography. It would also be very challenging to apply these algorithms in real sensors and study them assiduously. The architecture of the simulator is very close to TinyOS/NesC and this makes it very helpful to port the source code onto real hardware.

The current work can be extended in various ways, and one may try tuning different parameters for various implementations, using the Prowler-Rmase simulation environment and my source code for the abovementioned algorithms. This study can also act as a guideline for future designs for congestion control and avoidance algorithms, since it offers a thorough view in the structure and behavior of three resource control schemes.

Bibliography

[1] Prowler: Probabilistic wireless network simulator. http://www.isis.vanderbilt.edu/ Projects/nest/prowler/, 2010.

[2] Pavlos Antoniou, Andreas Pitsillides, Andries Petrus Engelbrecht, Tim Blackwell, and Loizos Michael. Congestion control in wireless sensor networks based on the bird flocking behavior. 5918:220–225, 2009.

[3] Richard R. Brooks, Parameswaran Ramanathan, and Akbar M. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, 2003.

[4] Alberto Cerpa, Jeremy Elson, Michael Hamilton, Jerry Zhao, Deborah Estrin, and Lewis Girod. Habitat monitoring: Application driver for wireless communications technology. In *SIGCOMM LA '01: Workshop on Data communication in Latin America and the Caribbean*, pages 20–41, New York, NY, USA, 2001. ACM.

[5] Wei-wei Fang, Ji-ming Chen, Lei Shu, Tian-shu Chu, and De-pei Qian. Congestion Avoidance, Detection and Alleviation in Wireless Sensor Networks. *Journal of Zhejiang University - Science C*, 11:63–73, 2010. 10.1631/jzus.C0910204.

[6] Romano Fantacci and Francesco Chiti. Urban microclimate and traffic monitoring with mobile wireless sensor networks. Wireless Sensor Networks: Application-Centric Design, Geoff V Merrett and Yen Kheng Tan (Ed.), ISBN: 978-953-307-321-7, InTech, Available at: http://www.intechopen.com/articles/show/title/urban-microclimate-and-traffic-monitoring-with-mobile-wireless-sensor-networks, 2010.

[7] Thomas Haenselmann. *Sensornetworks*. GFDL Wireless Sensor Network textbook, April 2005.

[8] Jyh-Ming Huang, Chun-Yi Li, and Kuong-Ho Chen. Talonet: A power-efficient grid-based congestion avoidance scheme using multi-detouring technique in wireless sensor networks. In *Wireless Telecommunications Symposium, 2009. WTS 2009*, pages 1 –6, 22-24 2009.

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

[10] Alan Miller Ittipong Khemapech and Ishbel Duncan. Simulating wireless sensor networks.

[11] Jaewon Kang, Yanyong Zhang, and Badri Nath. TARA: Topology-Aware Resource Adaptation to Alleviate Congestion in Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(7):919–931, 2007.

[12] Hsiang-Tsung Kung and Dario Vlah. Efficient location tracking using sensor networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1954–1962, March 2003.

[13] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 126–137, New York, NY, USA, 2003. ACM.

[14] Chenyang Lu, Brian M. Blum, Tarek F. Abdelzaher, John A. Stankovic, and Tian He. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks. Technical report, Charlottesville, VA, USA, 2002.

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

[16] Jeongyeup Paek and Ramesh Govindan. RCRT: rate-controlled reliable transport for wireless sensor networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 305–319, New York, NY, USA, 2007. ACM.

[17] Lucian Popa, Costin Raiciu, Ion Stoica, and David S. Rosenblum. Reducing congestion effects in wireless networks by multipath routing. In *ICNP*, pages 96–105. IEEE Computer Society, 2006.

[18] Fengyuan Ren, Tao He, Sajal K. Das, and Chuang Lin. Traffic-aware dynamic routing to alleviate congestion in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 99(RapidPosts), 2011.

[19] Kay Romer and Friedemann Mattern. The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications*, 11(6):54–61, December 2004.

[20] Yogesh Sankarasubramaniam, Ozgur B. Akan, and Ian F. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 177–188, New York, NY, USA, 2003. ACM.

[21] Loren Schwiebert, Sandeep K. S. Gupta, and Jennifer Weinmann. Research challenges in wireless networks of biomedical sensors. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 151–165, New York, NY, USA, 2001. ACM Press.

[22] Charalambos Sergiou. Congestion control in wireless sensor networks. Master's thesis, University of Cyprus, June 2007.

[23] Charalambos Sergiou, Vasos Vassiliou, and Andreas Pitsillides. Reliable Data Transmission in Event-Based Sensor Networks During Overload Situation. In *WICON '07: Proceedings of the 3rd International Conference on Wireless Internet*, pages 1–8, Austin, Texas, October 2007.

[24] Charalampos Sergiou and Vasos Vassiliou. DAIPaS: A Performance Aware Congestion Control Algorithm in Wireless Sensor Networks. In *18th International Conference on Telecommunications (ICT 2011)*, 8-11, May 2011.

[25] Ajay K. Sharma and Deepti Gupta. Article: Performance evaluation of routing protocols for wsns based on energy-aware routing with different radio models. *International Journal of Computer Applications*, 3(12):6–14, July 2010. Published By Foundation of Computer Science.

[26] Gyula Simon, Peter Volgyesi, Miklos Maroti, and Akos Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In *Proceedings of IEEE Aerospace Conference*, volume 3, pages 1339–1346, Big Sky, MT, USA, 2003.

[27] Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell. CODA: Congestion Detection and Avoidance in Sensor Networks. In *SenSys '03: Proceedings of the 1st international Conference on Embedded Networked Sensor Systems*, pages 266–279, New York, NY, USA, 2003. ACM Press.

[28] Alec Woo and David 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)*, pages 221–235, New York, NY, USA, 2001. ACM Press.

[29] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS'98)*, pages 154–161. IEEE Computer Society, May 1998.

[30] Ying Zhang and Gyula Simon. High-level sensor network simulations for routing performance evaluations.

Appendix A

MATLAB source code