Optimal Schedules for Data Gathering in Wireless Sensor Networks

Mahesh Bakshi

A Thesis
In the Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Computer Science) at
Concordia University
Montréal, Québec, Canada

June 2017

Concordia University
School of Graduate Studies

This is to certify that the report prepared

By: **Mahesh Bakshi**

Entitled: **Optimal Schedules for Data Gathering in Wireless Sensor Networks**

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (COMPUTER SCIENCE)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. Sudhir Mudur

_____ Examiner

Dr. Ali Akgunduz

_____ Examiner

Dr. Jaroslav Opatrny

_____ Examiner

Dr. Thomas Fevens

_____ Supervisor

Dr. Brigitte Jaumard

_____ Supervisor

Dr. Lata Narayanan

Approved _____

Dr. Volker Haarslev, Graduate Program Director

June 8, 2017

_____

Dr. Amir Asif, Dean

Faculty of Engineering and Computer Science

Abstract

Optimal Schedules for Data Gathering in Wireless Sensor Networks

Mahesh Bakshi, **Ph.D.**

Concordia University, 2017

Wireless Sensor Networks (WSNs) are widely used for target monitoring: sensors monitor a set of targets, and forward the collected or aggregated data using multi-hop routing to the same location, called the sink. The resulting communication scheme is called ConvergeCast or Aggregated ConvergeCast.

Several researchers studied the ConvergeCast and the Aggregated ConvergeCast, as to produce the shortest possible schedule that conveys all the packets or a packet aggregation to the sink. Nearly all proposed methods proceed in two steps, first the routing, and then the scheduling of the packets along the routes defined in the first step.

The thesis is organized around four contributions. The first one is an improvement of the previous mathematical models that outputs (minimum-sized) multi-set of transmission configurations (TCs), in which a transmission configuration is defined as a set of links that can transmit concurrently. Our model allows the transmission of several packets per target, in both single-path and multi-path settings; we give two new heuristics for generating new improved transmission configurations.While such models go beyond the routing step, they do not specify an ordering over time of the configurations. Consequently, the second contribution consists of several algorithms, one exact and several heuristics, for ordering the configurations. Our results show that the approach of scheduling when restricted to a tree generated by the first contribution significantly outperforms the ordering of configurations of TC-approach for single-rate, single packet per sensor traffic patterns, but the TC approach gives better results for multi-rate traffic and when there are a large number of packets per sensor.

In the last two contributions, we propose an exact mathematical model that takes care, in a single phase, of the routing and the scheduling, for the ConvergeCast and the aggregated ConvergeCast problem. They both correspond to decomposition models in which not only we generate transmission configurations, but an ordering of them.

We performed extensive simulations on networks with up to 70 sensors for both ConvergeCast and Aggregated ConvergeCast, and compared our one phase results with one of the best heuristics in the literature.

Acknowledgments

I wish to express my gratitude to my supervisors, Prof. Jaumard and Prof. Narayanan. Their encouragement, guidance, and deep support during my study enabled me to develop an understanding of the subject. Their wide knowledge, detailed and constructive comments have been a great value for me. Their timely financial support throughout my PhD helped me to concentrate on my work.

I am grateful to all my student colleagues for their help and support. I would like to specially thank Prof. Kaddour for allowing and helping me to extend his work.

Lastly, and most importantly, I wish to thank my spouse, for her understanding, support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A wireless sensor network (WSN) is a connected network of spatially distributed autonomous sensors that monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and forward the collected data through the network to a central sink node in the network. The development of wireless sensor networks was initially motivated by military applications such as battlefield surveillance. Today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on. Applications of WSN [34] can be broadly classified into three groups, namely, Environmental (Noise [68], RiverFloodDetection [23], etc.,), Condition Monitoring (WindTurbine [74], Pipelines [73], etc.,) and Process Automation (WaterConsumption [50], ProductionAutomation [52], etc.). WSN solutions should be scalable, reliable, low latency and power-efficient. Indeed, the combination of requirements is hard to meet.

In this thesis, we focus on a broad class of data-collection applications called *target coverage*. We give two sample target coverage applications to motivate our work, that differ in whether they require all the data collected by sensors or only a summary of the data.

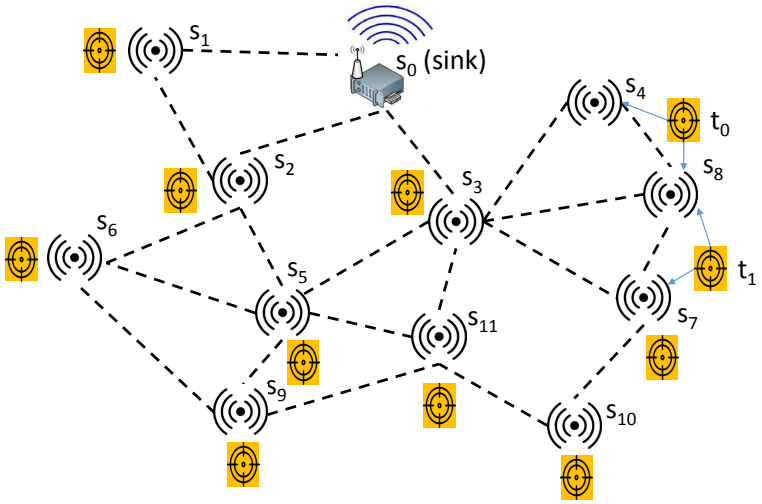
Application 1: Volcano Monitoring. The first application is active volcano monitoring, such as the network of 80 Waspmote sensors deployed in the Masaya volcano in August 2016 that connected Nicaragua's most active volcano to the internet. The sensors measure atmospheric pressure, humidity, temperature, various types of gases like sulphur dioxide, hydrogen sulphide and carbon dioxide. They also collect seismic data, gravity data with gravimeters in different places around and inside the volcano. Many other sensor network testbeds that have been deployed for a volcano to further research are described in [80, 55, 38, 46].

Application 2: Building monitoring. Another example is the deployment of wireless

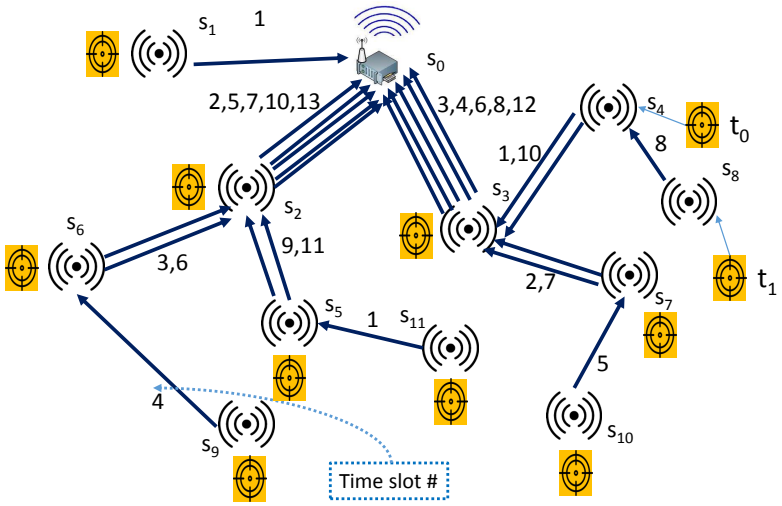
sensors in a smart building; sensors can sense the presence or absence of someone in an office, and a corresponding actuator can turn the lights on or off. The *building management system* collects and records information about energy usage in the building. In particular, it may be desirable to record the total or average number of hours the ventilation system or lights were on in different rooms. To enable data collection, sensors in each room may collect and send information about the amount of time the lights were on to a central sink node, which can in turn report the information to the building management system.

The communication pattern used to send the data from the sensors to the sink is called *ConvergeCast*. ConvergeCast is often done using a spanning tree of the network, with the sink as the root of the tree. Figure 1(a) shows a sensor network, and the routes taken by packets to achieve ConvergeCast are shown in Figure 1(b). While in Application 1, it may be of interest to send every item of data collected by sensors to the sink, Application 2 exemplifies the fact that in many situations, what is of interest is not to collect *every* item of data but a *function* of the data, such as the minimum or maximum or average reading. In such cases, tremendous energy savings can be obtained by requiring every intermediate sensor nodes to *aggregate* the data it receives before forwarding to the sink, thereby drastically reducing the number of packet transmissions required, and consequently both the time needed for the sink to receive the information it needs, and the energy used. For example, if the sensors are monitoring the temperature at each target, and what is required is for the sink to know the maximum temperature over all targets, each sensor needs to forward only the maximum of its own data and those received from its children. Such a ConvergeCast operation is called an *Aggregated ConvergeCast*. Figure 1(c) illustrates Aggregated ConvergeCast. Notice that the ConvergeCast operation requires a total of 23 packet transmissions, while the Aggregated ConvergeCast operation requires only 11 packet transmissions. In this thesis, we study both ConvergeCast and Aggregated ConvergeCast.

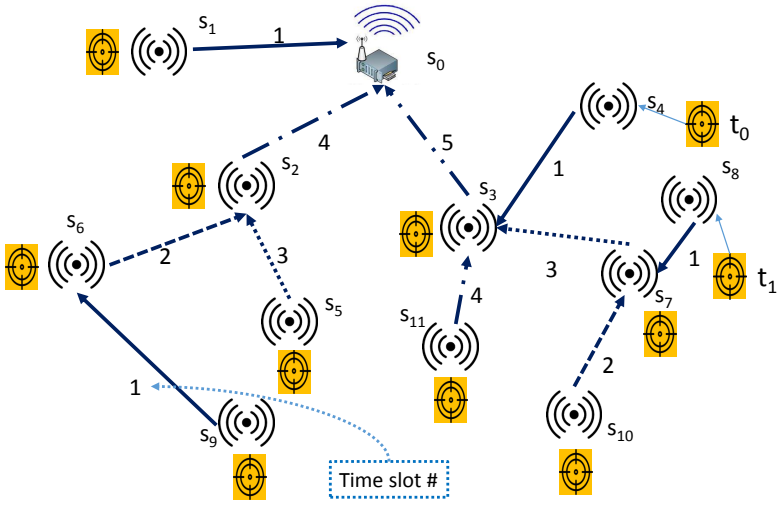
To improve reliability and to provide redundancy, it is often required that multiple sensors report on the same volcanic area or room in the building. This requirement is called q-coverage, *viz*, q sensors report on the same target (area within the volcanic region, room within the building). In this thesis, we consider q=1, 2, 3 in our experiments.



(a) Input



(b) ConvergeCast Communication Pattern



(c) Aggregate ConvergeCast Communication Pattern

Figure 1: ConvergeCast Problem Instance (q=1)

1.2 Medium access control

Medium access control (MAC) is one of the critical issues in the design of wireless sensor networks. As in any wireless network, the wireless transmission medium is a shared resource, and a *collision* is said to occur when two nodes send data at the same time over the same channel. To address collisions, a sensor network must employ a MAC protocol to arbitrate access to the shared medium and at the same time to fairly and efficiently share the bandwidth resources in a network. MAC protocols can be broadly classified into *contention-based* (e.g., IEEE 802.11) and *schedule-based* (e.g., FDMA, TDMA) protocols.

Although most of the state of the art wireless sensor networks (WSNs) are formed by nodes that are designed to operate in an autonomous, distributed fashion and thus use contention-based protocols, some recent work in the literature such as [39] indicates that centrally-coordinated network and MAC layer protocols are at least as efficient as distributed protocols in numerous settings, while bringing several advantages such as code simplicity, ease of management or observability. In particular, many WSN applications exhibit a regular traffic pattern by periodically collecting sensor measurements at a centralized sink. In this context, TDMA (Time Division Multiple Access) offers a convenient multiple access scheme at the MAC layer since it guarantees high bandwidth utilization and low energy consumption. In TDMA, the time is divided into frames, each containing a certain number of fixed size slots. Typically, a central entity is responsible to define a frame schedule assigning each node a fixed number of slots for transmitting and receiving data. Moreover, several transmission links can be scheduled in the same slot if no harmful level of interference occurs among them.

To a great extent, interference is captured in theoretical studies using either the *protocol* model or the *physical/SINR-based* model. In the *protocol* model, a transmission from a node s to a node t in time slot τ is successful if t lies within the transmission range of s and outside the transmission ranges of all other nodes transmitting in the same time slot τ. Under the protocol model, assuming that all nodes have the same transmission range T_{max}, the sensor network can be represented by a unit disk graph, where two nodes are connected if the distance between them is at most T_{max}. Then the criterion for successful transmission from s to t requires that no other neighbor of t is transmitting in the same time slot.

In reality, one has to take into account not just neighboring transmitters, but also environmental noise, as well as interference from nodes that are not immediate neighbors. In addition, node t *may* be able to receive s's transmission even if another neighboring node s' is transmitting simultaneously, provided the signal of s overwhelms that of s'. These factors are captured by the *physical interference* model: a transmission is successful if and only if the signal-to-interference-and-noise-ratio ($SINR$) at the intended receiver exceeds a certain

threshold so that the transmitted signal can be decoded with an acceptable bit error rate. This model is widely considered as a more accurate representation of the behavior of real systems [72]. We will use the *physical* interference model in this thesis.

Each sensor has a certain maximum transmission power specification. We consider sensors that have the ability to dynamically adjust their transmission power. Both the widely used MicaZ and TelosB sensors allows tuning of transmission power from 0.001mW to 1mW. Note that transmitting at the maximum power enables the sensor to directly reach sensors that may not be reachable with lower power. However, transmitting at high power levels increases interference to other sensors. Sensors are assumed to transmit using the modulation and coding schemes [54] of IEEE 802.15; according to this specification, a transmission with data rate θ can be decoded successfully if the *SINR* measured at the receiver is above a corresponding threshold β_θ.

1.3 Problem Statement

In this thesis we look at target coverage applications where sensors need to send raw readings or summarized data to the sink. Sensor nodes generate readings/data about the targets they are monitoring, and forward the data to other sensors or the sink. Other sensor nodes may not monitor targets but simply act as forwarding nodes in the network. We will study two kinds of ConvergeCast problems in this thesis: ConvergeCast Scheduling (CC) and the Aggregated ConvergeCast Scheduling (ACC).

1.3.1 ConvergeCast problem

The **ConvergeCast** problem uses the location of a set of sensors, a set of targets and the sink, as well as the desired coverage level q as input, and finds a minimum length TDMA frame that achieves the ConvergeCast operation, that is,

1. Each target is monitored by exactly q sensors.
2. Each sensor forwards all the data it receives along a path to the sink.
3. The sink node gets q readings about every target.

The solution involves deciding which sensors cover which targets, finding paths to send the sensor readings from the covering sensors to the sink node, as well as scheduling transmission slots for the sensors on these paths that avoid excessive interference. The goal is to find a transmission schedule of minimum length, that is, a schedule with the minimum number of

transmission slots. Figure 1 demonstrates an instance of ConvergeCast. Figure 1(a) shows the input for the problem: the location of the set of targets (e.g., t_0, t_1), sensors$\{s_1, \ldots, s_{11}\}$, and the sink node s_0. The desired coverage level is $q = 1$, i.e., each target needs to be covered by one sensors. Figure 1(b) gives a ConvergeCast solution for this input. Observe that each target is monitored by one sensor, that is, q=1. Target t_0 can be monitored by sensor s_4 or by sensor s_8, selection of the sensor monitoring a target is also part of solution. Each of the monitoring sensors needs to send the information concerning the target(s) it is monitoring along a path to the sink, as shown in the figure. Since sensor s_9 is monitoring one target, it needs to send one packet to the sink. Finally the links along all paths have to be scheduled while respecting interference constraints. One possible schedule is as follows. In time slot 1, schedule the links (s_1, s_0), (s_4, s_3) and (s_{11}, s_5) simultaneously, then in slot 2, the links (s_2, s_0) and (s_7, s_3). In the next eleven slots, the sets $\{(s_3, s_0), (s_6, s_2)\}$, $\{(s_3, s_0), (s_9, s_6)\}$, $\{(s_2, s_0), (s_{10}, s_7)\}$, $\{(s_3, s_0), (s_6, s_2)\}$, $\{(s_2, s_0), (s_7, s_3)\}$, $\{(s_3, s_0), (s_8, s_4)\}$, $\{(s_5, s_2)\}$, $\{(s_2, s_0), (s_4, s_3)\}$, $\{(s_5, s_2)\}$, $\{(s_3, s_0)\}$, $\{(s_2, s_0)\}$ can be scheduled in turn, and it can be verified that all data reaches the sink.

1.3.2 Aggregated ConvergeCast problem

Given a set of sensor locations, and a sink node, we consider the problem of finding a minimum-length schedule for Aggregated ConvergeCast. In particular, a valid schedule satisfies the following constraints assuming each sensor is monitoring a target:

1. Each sensor sends exactly one packet.
2. A sensor cannot receive a packet during or after the time slot when it transmits.
3. The sink node receives all the aggregated data.

Figure 1 demonstrates an instance of Aggregated ConvergeCast and a possible solution. Figure 1(a) shows the input for the problem: the location of the set of sensors $\{s_1, \ldots, s_{11}\}$, and the sink node s_0. Figure 1(c) gives an Aggregated ConvergeCast tree. We assume that every sensor is monitoring a target and has an item of data to send to the sink. Observe that without aggregation using a defined interference model, we need 23 packet transmissions using this tree, and any schedule would be of length at least 12 slots. However, if each sensor waits to receive information from its children, and aggregates its own data with that received from its children, the operation can be achieved using 11 packet transmissions, and there is a schedule with 5 slots.

Each of the monitoring sensors needs to aggregate and send the information concerning the target it is monitoring along a path to the sink, as shown in the figure. Since sensor s_3 is

monitoring its own target and receives a packet from s_4, s_7 and s_{11}, it needs to aggregate four packets and send one aggregated packet towards the sink. Each sensor has a path to the sink as given in Figure 1(c). It uses 5 slots $\{(s_9, s_6), (s_1, s_0), (s_8, s_7), (s_4, s_3)\}$, $\{(s_6, s_2), (s_{10}, s_7)\}$, $\{(s_7, s_3), (s_5, s_2)\}$, $\{(s_2, s_0), (s_4, s_3)\}$, $\{(s_5, s_2)\}$, $\{(s_2, s_0)\}$, $\{(s_3, s_0)\}$ in slot 1 to slot 5 respectively. It can be verified that aggregated data from all the sensors reaches the sink. The information acquired at the sink is commonly the aggregated information like maximum or average, so that we can accumulate data rapidly and reduce consumption of transmission power. At the same time, interference from simultaneous transmissions is also reduced as we use fewer transmissions.

A complete Aggregated ConvergeCast solution consists of a tree, and a schedule for links in the tree that avoid interference. A subtle issue is that interference is caused not just by tree links, but also by non-tree links. For instance in Figure 1(c), the tree links (s_5, s_2) and (s_{11}, s_3) cannot be scheduled in the same time slot, even though the receivers of the two links are different, because of the existence of the non-tree link (s_5, s_3) which causes interference at s_3.

In this thesis, we study algorithms to schedule ConvergeCast both with and without aggregation using TDMA. Both of these problems are NP-hard as shown in [10] and [18] respectively.

1.4 Literature review

In this section we will survey the most recent work on ConvergeCast (Section 1.4.1) and Aggregated ConvergeCast (Section 1.4.2).

1.4.1 ConvergeCast

We classify the research on ConvergeCast into two main categories: scheduling algorithms with mathematical programming models and scheduling algorithms without mathematical programming models.

ConvergeCast: Mathematical Programming Models

The problem of generating a minimum number of transmission configuration occurrences to solve communication instances has been extensively studied in the literature for TDMA wireless networks. There are two classes of algorithms that have been proposed in the literature.

The first class considers mathematical programming tools, i.e., column generation models that allow the decoupling of the TFMP$^+$(TDMA Frame Minimization Problem) problem into two subproblems solved alternately until an optimality condition is satisfied. Earlier work studied the problem in the context of WiMax networks without considering the transmission power, see, e.g., [25], [15] and then later with the integration of the power control constraints [26], [14], [62]. In [27], ElBatt and Ephremides solved the problem via two alternating phases that define a set of admissible links along with their transmission power in the context of ad-hoc networks. The main contribution was to eliminate the need of computationally expensive algorithms by splitting the problem and executing the power control in a distributed fashion. Kaddour [44], [43] adapted the previous column generation models for the design of wireless sensor networks subject to *SINR* constraints, as well as q-coverage, power control and rate adaptation considerations. However, due to the computational complexity of generating transmission configurations under such constraints, his model lacks scalability.

ConvergeCast: Non-Mathematical Programming Models

The second class of algorithms deals with heuristics, see, e.g., [11], [48]. For joint link scheduling and power control with the use of heuristics, see, e.g., [75] who consider the objective of throughput improvement while considering fairness through a new introduced factor called *demand satisfaction factor*. After the original model was formulated as a Mixed Integer Linear Program (MILP) (not a column generation model), the key idea was to iteratively use the solutions obtained from a Linear Program (LP)- a relaxed version of the problem - as guidelines for channel scheduling.

Similarly, for joint scheduling and routing, many heuristics have been proposed; see for example [48], [31], [40], [12], [70], [17], [58], [61], [13], [53], [45] [36], [47] and [32]. Of these, [48], [40], [12], [70], [36], [47] and [32] use the *SINR* model; the remaining papers use the protocol model. The authors of [48], [36], and [47] proposed approximation algorithms for multi-hop networks assuming unlimited transmission power, constant power and limited transmission power respectively. But they deal with arbitrary traffic patterns, not converge cast.

The authors of [40, 12, 70, 32] deal with ConvegeCast in the *SINR* model. [40] provides heuristics for ConvergeCast based on trees using the *SINR* model. They claim to show how to use multiple frequencies to eliminate interference. Two ConvergeCast heuristics using Dijkstra and graph coloring are provided in [12]. In [70], nodes are divided into clusters and a non-linear optimization model is given to get a ConvergeCast solution using the *SINR* model. Similarly, Gong and Yang first identify a ConvergeCast tree, then construct a weight-based heuristic [32] for scheduling on the tee. The weight of a link is related to its capacity to

cause interference to other links. None of the algorithms in [40, 12, 70, 32] consider multi-rate sensors or q-coverage, and do not provide any bounds on the accuracy of their solutions.

1.4.2 Aggregated ConvergeCast Scheduling

Most of the existing work for Aggregated ConvergeCast Scheduling uses the protocol, i.e., graph-based interference model and a minimum spanning tree rooted at the sink node, also commonly called Shortest Path Tree in the literature. We start with a brief description of this work, and then describe related work on the SINR interference model.

1.4.3 Aggregated ConvergeCast Using Protocol Interference Model

Aggregated ConvergeCast Scheduling for unit disk graphs using the protocol interference model is studied in Kesselman *et al.* [49], Gandhi *et al.*[30], Wan *et al.* [78], Xu *et al.* [81], Gagnon *et al.* [29], Guo *et al.* [35], Pan *et al.* [66], Jakob *et al.* [41], Yousefi *et al.* [82]. Guo *et al.* [35] gave an Aggregated ConvergeCast schedule of length $O(D + \delta)$, where D is the diameter of the input graph and δ is the maximal degree. As every Aggregated ConvergeCast schedule is of length at least D, it gives $O(\delta)$-approximation ratio (δ can be $\Theta(n)$). Gandhi *et al.*[30] gave a randomized approximation algorithm ratio of $\sqrt{\tilde{d}n}$, where $\tilde{d}$ is the average degree. Kesselman *et al.* [49] showed that aggregation can then be achieved in $O(\log n)$ assuming the Collision Detection protocol is available at each sensor. Pan *et al.* [66] construct a scheduling tree using a weight function based on receiver's depth and number of children and propose a scheduling algorithm based on neighbours' degree. Jakob *et al.* [41] uses top-down approach and produce a heuristic schedule without any tree construction. Yousefi *et al.* [82] provided another heuristic based on a distributed algorithm. Erzin *et al.* [28] proved that for a given Aggregated ConvergeCast tree, the problem of finding optimal schedule is still NP-hard using protocol interference model.

1.4.4 Aggregated ConvergeCast Using *SINR* Interference Model

Moscibroda *et al.* [64], Li *et al.* [59], Li *et al.* [57], Halldorsson *et al.*[37], and Wang *et al.* [77] study the problem using the SINR interference model and propose heuristics. Assuming discrete power levels, Moscibroda *et al.* [64] proposed a polylogarithmic bound of $O(\log^4 n)$ slots for their scheduling algorithm using an SINR model, where n is the number of sensors. For uniform or linear power levels, their algorithm needs $O(n^2 \log n)$ slots. Halldorsson *et al.*[37] relax the SINR interference model by using unlimited transmission power, ignoring noise, and $\alpha > 2$, where α is the path loss exponent. They then provide an algorithm that

connects an arbitrary point set in $O(\log n)$ slots, improving on the results of Moscibroda *et al.* [64].

Li *et al.*[59] provided a heuristic using the dominating Set for the Aggregated Converge-Cast. Li *et al.* [57] provide a $O(\log^3 n)$ heuristic. This last heuristic uses a round-based approach; in each round it gives preference to the smaller links and selects set of links satisfying a simplified SINR condition. Data is transmitted on the selected links, whose source sensors are subsequently removed from consideration. This process is repeated until all sensors forward the aggregated data to the sink. The major drawback of this approach is that it assumes the network is connected even after removing some links. Wang *et al.* [77] propose an algorithm with a lower bound of $O(d \log^2 n))$, where d is the depth of the tree. This algorithm also consists of several in rounds; in each round we schedule all the links in the highest layer first and repeat this procedure in each round for all the links in different layers. Indeed, we will compare our algorithms with this last heuristic as it appears to be the most efficient one using a SINR interference model.

Ebrahimi *et al.* [24] give a schedule using a mathematical model for a related problem with several aggregated trees.

1.5 Thesis Contributions

The contributions of this thesis have been published in four papers. We give a brief description of each below:

Chapter 2[8]: We propose an improvement to the TFMP (Time Frame Minimization Problem) model of [44] to derive a near-optimal set of configurations for ConvergeCast. Our model allows the transmission of several packets per target, in both single-path and multi-path settings. We give two new heuristics for solving the pricing problem, i.e., for generating new improving transmission configurations. Our results show that significant gains in scalability can be obtained, thanks to our enhanced solution scheme.

Chapter 3[9]: We compare two common approaches to computing a minimum length schedule for the TDMA frame. In the first approach, called the TC-approach, an optimal (minimum-sized) multi-set of transmission configurations (TCs) that are interference-free and that cover the ConvergeCast traffic is computed. It is generally left unspecified in what order and how many times to actually schedule these TCs. In the second approach, called the two-phase approach, first a routing tree or subgraph is computed, and next, sets of non-interfering links are scheduled in rounds, based on which links have available traffic in each round.

In this paper, we start by giving a new column generation approach called TFM-Tree, to build an optimal set of TCs when the scheduling is restricted to a tree. Our model takes into account variable power assignment, q-coverage, and multi-rate sensors. Next, for any given set of TCs that comprise a feasible solution, we give algorithms to schedule the TCs. In particular, we give an ILP model that computes an optimal schedule using the given set of TCs, as well as several new and efficient scheduling heuristics. For the two-phase approach, we observe that the TFM-Tree model gives as a by-product a possible routing tree. We give a new scheduling algorithm for the second phase, called the Round-Optimal-Scheduling (ROS) algorithm, and significantly modify the scheduling algorithm in [32]. We performed extensive experimental evaluations of both approaches. Our results show that the two-phase approach using the TFM tree significantly outperforms the TC-approach for single-rate, single packet per sensor traffic patterns, but the TC approach gives better results for multi-rate traffic and when there are a large number of packets per sensor.

All the existing mathematical approaches just output set of configurations but do not order them. Similarly, a multiset of transmission configurations obtained by phase one of TFMP does not constitute a schedule for ConvergeCast. We investigated set of algorithms to schedule these transmission configurations and also proposed a mechanism to obtain the optimal schedule.

Chapter 4[5]: In this paper, we give for the first time, a mathematical programming formulation for a complete and optimal solution, i.e., an ordered sequence of transmission configurations that achieves ConvergeCast. This solution provides much better results than those of the previous best available mathematical programming or heuristic approaches in the literature.

Chapter 5[3]: We consider the scheduling problem for Aggregated ConvergeCast in wireless sensor networks with the physical model for interference. Previous work on the problem has provided either heuristics without performance guarantees, or approximation algorithms which do not perform well in practice. We propose here a first mathematical model that outputs an optimal Aggregated ConvergeCast schedule. Since the resulting Integer Linear Program (ILP) model is computationally hard to solve, we use large scale optimization techniques, namely a Dantzig-Wolfe decomposition algorithm, to solve it. We performed extensive simulations on networks with upto 70 sensors, and compared our results with one of the best heuristics in the literature [77]. Our results show that our ε-optimal schedule is significantly better than the previous best schedule, i.e., it produces TDMA frames that are about 50% shorter.

All numerical experiments in this thesis were implemented in Java using the CPLEX Concert Technology (version 12.6). Data sets were the same for Chapter 4 and Chapter 5, but different for Chapter 2 and Chapter 3.

Chapter 2

Efficient Minimization of TDMA Frame Length in Wireless Sensor Networks

M. Bakshi, M. Kaddour, B. Jaumard, and L. Narayanan. An efficient method to minimize TDMA frame length in wireless sensor networks. submitted for publication, 2017. An extended abstract of this paper has been published in IEEE Wireless Communications and Networking Conference (WCNC), 2015 [7].

2.1 Introduction

Although most of the state-of-the-art Wireless Sensor Networks (WSNs) are formed by nodes that are designed to operate in an autonomous and distributed fashion, some recent work in the literature such as [39] indicates that centrally-coordinated network and Media Access Control (MAC) layer protocols are at least as efficient as distributed protocols in numerous settings, while bringing several advantages such as code simplicity, ease of management or observability. In particular, many WSN applications exhibit a regular traffic pattern resulting from a periodic collection of sensor measurements at a centralized entity called the sink. In this context, Time Division Multiple Access (TDMA) offers a convenient multiple access scheme at the MAC layer since it guarantees high bandwidth utilization and low energy consumption. In TDMA, the time is divided into *frames*, where each frame contains a certain number of fixed size *time slots*. All sensor measurements are to be transmitted to the sink in a single frame. Typically, a central entity is responsible for defining a frame *schedule*, by assigning each node a fixed number of slots for transmitting and receiving data.

Moreover, several transmission links can be scheduled in the same time slot if no harmful level of interference occurs among them. This variant is sometimes called Self-Organized Time Division Multiple Access (STDMA).

Interference is generally captured in theoretical studies or simulation using either the protocol model or the physical interference model [71]. Under the protocol model, a successful transmission occurs when the intended receiving node falls inside the transmission range of its transmitting node and falls outside the interference ranges of all other (non-intended) transmitters. On the other hand, under the physical model, a transmission is successful if and only if the Signal-to-Interference-and-Noise-Ratio (SINR) at the intended receiver exceeds a certain threshold so that the transmitted signal can be decoded with an acceptable bit error rate. The latter model is widely considered a much more accurate representation of the behavior of real systems [72]; we adopt this model of interference in our work.

The general problem of determining a minimum-length frame, and consequent schedule, that satisfies given traffic demands as well as SINR constraints is NP-hard as shown in [10]. In this paper, we study a target coverage application, in which n sensors collectively monitor a set of m targets, so that each target is monitored by q sensors, and all sensor measurements are sent to a designated sink node. As explained above, a TDMA frame must specify a set of transmissions that meet SINR requirements for every time slot in the frame. We define a *transmission configuration* to be such a set of links, with associated data rates, that can transmit concurrently during one time slot subject to the SINR requirements. We study the TDM Frame Minimization Problem (TFMP), which requires the generation of the smallest possible multiset of *transmission configurations* that achieves the transmission of all sensor measurements to the sink node. It can be easily seen that the TFMP problem is equivalent to the problem of maximizing the network throughput

2.1.1 Related work

The problem of generating a minimum number of transmission configuration occurrences to solve communication instances has been extensively studied in the literature for TDMA wireless networks. There are two classes of algorithms that have been proposed in the literature.

The first class considers mathematical programming tools, i.e., column generation models that allow the decoupling of the TFMP$^+$ problem into two subproblems solved alternately until an optimality condition is satisfied. Earlier work studied the problem in the context of WiMax networks without considering the transmission power, see, e.g., [25], [15] and then later with the integration of the power control constraints [26], [14], [62]. In [27], ElBatt and

Ephremides solved the problem via two alternating phases that define a set of admissible links along with their transmission power in the context of ad-hoc networks. The main contribution was to eliminate the need of computationally expensive algorithms by splitting the problem and executing the power control in a distributed fashion. Kaddour [44] adapted the previous column generation models for the design of wireless sensor networks subject to SINR constraints, as well as power control and rate adaptation considerations. However, due to the computational complexity of generating transmission configurations under such constraints, his model lacks scalability.

The second class of algorithms deals with heuristics, see, e.g., [11],[48]. For joint link scheduling and power control with the use of heuristics, see, e.g., [75] who consider the objective of throughput improvement while considering fairness through a new introduced factor called *demand satisfaction factor*. After the original model was formulated as a Mixed Integer Linear Program (MILP) (not a column generation model), the key idea was to iteratively use the solutions obtained from a Linear Program (LP)- a relaxed version of the problem - as guidelines to schedule some channel. Similarly, for joint scheduling and routing, many heuristics have been proposed; see for example [48], [31], [40], [12], [70], [17], [58], [61], [13], [53], [45] [36], [47] and [32]. Of these, [48], [40], [12], [70], [36], [47] and [32] use the SINR model; the remaining papers use the protocol model. The authors of [48], [36], and [47] proposed approximation algorithms for multi-hop networks assuming unlimited transmission power, constant power and limited transmission power respectively. But they deal with arbitrary traffic patterns, not converge cast. The authors of [40, 12, 70, 32] deal with convegercast in the SINR model. [40] provides heuristics for ConvergeCast based on trees using the SINR model. They claim to show how to use multiple frequencies to eliminate interference. Two ConvergeCast heuristics using Dijkstra and graph coloring are provided in [12]. In [70], nodes are divided into clusters and a non-linear optimization model is given to get a Convergecast solution using the SINR model. Similarly, Gong and Yang first identify a ConvergeCast tree, then construct a weight-based heuristic [32] for scheduling on the tee. The weight of a link is related to its capacity to cause interference to other links. None of the algorithms in [40, 12, 70, 32] consider multi-rate sensors or q-coverage, and do not provide any bounds on the accuracy of their solutions. As such, they are not comparable to our work.

2.1.2 Our results

In this paper, we present a scalable optimization model to minimize the TDMA frame length that enables the gathering of sensor observations at the sink. The solution consists of finding

subsets of transmission links which can be activated simultaneously, and an association between targets and sensors. Our model satisfies the given target coverage requirements and allows for multi-path routing, power control, rate adaptation. In particular, *(i)* the interference model is SINR-based; *(ii)* sensors can dynamically adjust their transmission power to reduce interference or hop-distance from the sink; *(iii)* transmission links can admit different data rates depending on the SINR threshold at the receiver; *(iv)* different packets between the same source and destination can take different paths; *(v)* a target can be associated with multiple sensors providing redundancy and reliability to sensing data.

The adopted approach, formulated initially as an integer linear program, relies on a column generation decomposition formulation to decouple traffic and bandwidth management, which are solved in the so-called *restricted master problem*, from feasible configuration generation, the so-called *pricing problem*. Our model for traffic and bandwidth management problem is a generalization of the work in [44], allowing the transmission of several packets per target by the monitoring sensors. In contrast with [44], the difficult problem of generating feasible configurations is tackled by two heuristics, called Hybrid1 and Hybrid2, both of which efficiently generate transmission configurations subject to SINR, rate adaptation and power control constraints. When our heuristics fail to generate an improving transmission configuration (i.e., a configuration whose addition improves the value of the current mathematical model), the pricing problem (generation of new transmission configurations) is solved exactly.

2.1.3 Organization of the paper

The rest of this paper is organized as follows. We present our network model in Section 2.2. We describe in Section 2.3 the enhanced column-generation-based model to maximize network throughput. Section 2.3.3 and 2.3.4 are devoted to a detailed presentation of our two heuristic approaches to solve the pricing problem, i.e., to generate the improving transmission configurations. Section 2.4 presents our extensive computational experiments in order to assess the scalability of our approach, and the accuracy of the obtained results, as well as a comparison with the results of [44]. Finally, conclusions are drawn in Section 2.5.

2.2 Sensor Network Model

We consider a set of n sensors, denoted as $S = \{s_1, s_2, ..., s_n\}$ and a set of m targets, denoted as $T = \{t_1, t_2, ..., t_m\}$, deployed arbitrarily on a given area. Each target must be covered by q sensors, known as the q-coverage requirement. We assume that a target can be covered

by a given sensor if the Euclidean distance between them does not exceed the sensing range $R_{\max}$. Each sensor periodically generates a set of data packets for every target that it is monitoring; each such data packet is of size a multiple of σ bits. All the monitored data must be forwarded, possibly after passing through several hops, to a sink node, denoted by s_0.

We assume that each sensor is equipped with a single radio that can be tuned dynamically without a significant delay to transmit with some power level in the range $[0, P_{\max}]$. Let λ_{max} be the transmission range obtained with power P_{max}. Network connectivity is represented by a directed graph $G = (S \cup \{s_0\}, L)$ where

$$L = \{(s_i, s_j) : d_{ij} < \lambda_{\max}, i = 1, \ldots, n,\ j = 0, \ldots, n\}.$$

According to the given deployment of sensors and targets on the area, we assume that all sensors have a path toward the sink and are able to transmit according to a given modulation and coding scheme (MCS), as proposed in [54], where MCS_r ($r \in R$) generates date rate θ_r. Let $\theta_1 < \theta_2 < \ldots < \theta_{|R|}$.

According to the physical model (see, e.g., [33]), a transmission with rate θ_r can be decoded successfully if the SINR measured at the receiver is above a corresponding threshold β_r. More precisely, the transmission of a node s_i on the link (s_i, s_j) with data rate θ_r is successful if SINR_{ij}, measured at the intended receiver, given by

$$\text{SINR}_{ij} = \frac{p_i d_{ij}^{-\alpha}}{N_0 + \sum\limits_{i':i' \neq i} p_{i'} d_{i'j}^{-\alpha}} \geq \beta_r \tag{1}$$

where p_i is the transmission power of node s_i, d_{ij} is the distance between s_j and s_j, α is the propagation loss exponent, N_0 is the ambient noise, and the summation in the denominator is taken over all other nodes i' transmitting in the same time slot (even if to different intended receivers).

We assume a TDMA access scheme, where the channel is divided into time slots of fixed duration. A set of N contiguous time slots, where N is a system parameter such that the sink receives q packets from each target (q-coverage), forms a frame. Such a frame repeats cyclically over the time. In each frame, every sensor node is assigned a set of specific time slots. This last set constitutes the schedule according to which the sensor nodes operates in each frame. The duration of each slot is T_{SLOT}, which corresponds to the time required to send a data packet of σ bits using the lowest data rate θ_1.

2.3 TDMA Frame Minimization Problem Plus (TFMP$^+$)

In this section, we describe our solution to the TDMA Frame Minimization problem. We first give an ILP that we call TFMP$^+$ to solve the problem; this is an extension of the TFMP model given in [44] that allows for sensors to send multiple packets per target to the sink. We then enhance the column generation formulation for TFMP given in [44] by introducing two new heuristics to solve the time-consuming pricing problem. Our experiments show that our solution is much more scalable than the original TFMP model.

2.3.1 Optimization Model

We adopted a decomposition scheme as in [51],[25],[44] that is based on the concept of a *transmission configuration*, i.e., a subset of radio links that can be scheduled concurrently without violating the SINR requirement at each receiver. Our objective is to define a minimum-length set of TDMA configurations by determining jointly which sensors cover each target, the set of concurrent transmitting links during each slot, along with their used MCSs and power levels, and by establishing the routes toward the sink.

A configuration c lasts for one time slot and is formally defined as:

$$c = \left\{ (x^c_{ijr}, p^c_i) : (s_i, s_j) \in L, r \in R, \mathrm{SINR}_{ij} \geq \beta_r, \sum_{r \in R} x^c_{ijr} \leq 1 \right\} \quad (2)$$

where x^c_{ijr} is a binary parameter indicating if link (s_i, s_j) is scheduled in c with data rate r, and p^c_i is the transmission power used by s_i. The set of all potential configurations is denoted by C.

We denote by the integer variable λ_c the number of occurrences of configuration c, i.e., the number of time slots during which the configuration c is scheduled. Let y_{ij} be a binary variable indicating if target t_j is covered by sensor s_i ($i \neq 0$). In addition, let f_{ij} be an integer flow variable counting the number of data packets transmitted on link (s_i, s_j) during the whole TDMA frame. The TFMP$^+$ problem can be formulated by the following Integer Linear Program (ILP):

$$[\mathrm{TFMP}^+] \qquad \min \sum_{c \in C} \lambda_c \quad (3)$$

subject to:

$$\sum_{s_i \in S} y_{ij} = q \qquad t_j \in T \tag{4}$$

$$\sum_{s_j \in S} f_{ji} + \sum_{t_k \in T} y_{ik} \times pkts_k = \sum_{s_i \in S \cup \{s_0\}} f_{ij} \qquad s_i \in S \tag{5}$$

$$\sum_{s_i \in S} f_{i0} = q \times \sum_{t_k in T} pkts_k \tag{6}$$

$$\sum_{c \in C} \sum_{r \in R} T_{\text{SLOT}} \theta_r x^c_{ijr} \lambda_c - f_{ij} \hat{\sigma} \geq 0 \qquad (s_i, s_j) \in E \tag{7}$$

$$\lambda_c \geq 0 \text{ and integer}, \qquad c \in C \tag{8}$$

$$f_{ij} \geq 0 \text{ and integer}, \qquad (s_i, s_j) \in E \tag{9}$$

$$y_{ik} \in \{0, 1\} \qquad s_i \in S, t_k \in T. \tag{10}$$

Constraint (4) ensures that every target is covered by exactly q sensors. Constraint (5) represents a flow conservation rule that states that the sum of the incoming traffic into sensor s_i, i.e., the sum of traffic forwarded to s_i by the other sensors and of the local traffic generated by the monitoring of targets under the responsibility of s_i, is equal to the outgoing traffic. Constraint (6) guarantees that all data packets are gathered by the sink. The channel capacity constraint (7) ensures that the number of times each link (s_i, s_j) is included in all the scheduled configurations ($\lambda_c > 0$) is sufficient to forward the allocated traffic: $f_{ij}\ \hat{\sigma}$ represents the number of bits to be transmitted on link (s_i, s_j), while $\sum_{c \in C} \sum_{r \in R} T_{\text{SLOT}}\, \theta_r\, x^c_{ijr}\, \lambda_c$ is equal to the available channel capacity with the selected set of transmission configurations.

To allow for the possibility that each sensor generates multiple readings for every target that it monitors, which can be routed independently to the sink, we introduce a new integer variable "$pkts_k$" which denotes the number of packets collected from each target k. Note that the different packets collected from the same target can take different routes to the sink.

We also consider the case when a sensor reading does not fit into a time slot, and sensors need to *fragment* the reading. In this case we require that the entire reading is assembled at the next hop before any fragment can be routed further. Note that all fragments of a packet should now take the same route to the sink. This can be achieved in our model by simply adjusting the value of the parameter $\hat{\sigma}$ in Constraint 7. Recall that T_{SLOT} is the time required to send a packet of size σ using the lowest data rate. So using $\hat{\sigma} = \sigma$ implies no fragmentation, and $\hat{\sigma} = 2\sigma$ means that each packet should be fragmented into two fragments of size σ and so on.

Finally, the model also handles the situation when k readings are collected from each

sensor, but each reading needs to be fragmented. In this case, the multiple packet fragments corresponding to a reading take the same route (and in fact are reassembled at every node on the path), while packets corresponding to different readings may take different paths.

While clearly the model is not scalable if it is required to exhaustively enumerate all the candidate transmission configurations, we can use column generation techniques, as described in the next section, to solve it by taking advantage of the implicit enumeration of those configurations, thereby resulting in a scalable solution scheme.

2.3.2 Column Generation and Pricing Problem

Column generation (CG) is an exact and efficient algorithm for solving large-scale linear programs [56],[20]. A key observation is that for a mathematical program with m constraints and n variables, the optimal solution of the linear programming cannot contain more than m nonzero variables, even if $m << n$, and n is an exponential function, e.g., with respect to the number of links in a network. Consequently, most of the variables will be non-basic (equal to zero) in the optimal solution of the linear program. CG is a technique that allows to quickly identify the set of nonzero variables in the optimal solution without an explicit enumeration of all the variables (columns) of the optimization model. Indeed, the original problem (called the master problem) is decomposed into *the restricted master problem* (RMP) and the *pricing* problem. The RMP is the problem as described in (3) - (10) with a very small collection of configurations (or variables λ_c), and aims at selecting the best set of configurations among the generated/considered ones. The pricing problem corresponds to the configuration generator that generates only the variables (λ_c)/configurations which improve the optimal value of the objective of the current RMP. The solution process alternates between the solution of the RMP and of the pricing until the optimality condition is satisfied: the pricing problem can no longer generate a variable (column/configuration) such that, if added to the current RMP, the optimal value of the enhanced RMP is improved. This amounts to checking whether the optimal value of the objective function of the pricing problem, i.e., the reduced cost (see, e.g., [56], [20]) is negative (minimization case).

For our problem TFMP$^+$, the RMP is formulated as the linear relaxation of (3)-(10) with only a subset $C_0 \subseteq C$ of candidate configurations, in which the variables λ_c, f_{ij} and y_{ij} become non negative variables with $\lambda_c \geq 0$.

The pricing problem relies on finding a new transmission configuration with a negative reduced cost. While the search for an improving transmission configuration guarantees the best improvement (for the next re-optimization of the current RMP) with the addition of a single configuration at a given iteration of the column generation algorithm, it suffices

to generate a configuration with a negative reduced cost (not necessarily the most negative one) in order to iterate. Moreover, it usually leads to an overall faster solution of the linear relaxation, see, e.g., Chapter 1 in [22].

Let u_{ij} be the dual variables associated with (7- ij), the reduced cost of any configuration c can be written:

$$\text{RED\_COST}_c = 1 - \sum_{(s_i,s_j)\in L} \sum_{r\in R} \theta_r \, T_{\text{SLOT}} \, x^c_{ijr} \, u_{ij}. \tag{11}$$

If the minimum reduced cost has a negative value, the corresponding configuration (column) is added to C_0 and the RMP is solved again, otherwise the optimal solution has been reached. We call the pricing problem provided in [44] as PP_TFMP. We use TFMP$^+$ as the name of our enhanced optimization model given in Section 2.3.1. using the PP_TFMP pricing problem. The computationally time consuming part corresponds to the solution of the PP_TFMP. Consequently, we discuss in the next section how to solve it more efficiently, keeping in mind that it suffices to identify a transmission configuration with a negative reduced cost (no need to identify the transmission configuration with the most negative reduced cost) in order to iterate in the solution process of the column generation technique.

2.3.3 Algorithm Hybrid1 - Enhanced Pricing Problem1

We propose a hybrid algorithm, called Hybrid1, which attempts to generate an augmenting transmission configuration, i.e., a configuration with a negative reduced cost, thanks to a greedy heuristic called Greedy1, and which solves PP_TFMP exactly if Greedy1 fails in the generation of an augmenting transmission configuration.

Algorithm 1 Heuristic Greedy1

Require: *Dual values, set of data rates, link lengths*

Ensure: L^{SELECT}*: Transmission configuration (with power values and data rates)*

```
L^SELECT ← ∅ {Links already selected as radio links of the configuration under construction}
L^SORTED ← list of dual values sorted in decreasing order
for each link ℓ ∈ L^SORTED do
  okay-to-add ← TRUE.
  for each ℓ' ∈ L^SELECT do
    if (ℓ and ℓ' have a common endpoint) or (PP_TFMP+ (ℓ) is not feasible) then
      okay-to-add ← FALSE.
    end if
  end for
  if okay-to-add then
    L^SELECT ← L^SELECT ∪{ℓ}
  end if
end for
```

Greedy1 proceeds as follows. It adds in a greedy fashion the links that can contribute the most to the reduced cost, while causing the least interference. It starts from an empty transmission configuration whose links will later be stored in the L^{SELECT} set of links. We next try to add the link associated with the largest dual values (see the expression (11) of the reduced cost) in order to reach a negative reduced cost. Let ℓ be that link. Ties are broken using the length of the links, as a shorter link will cause less interference than a longer one. In order check whether ℓ can be added to the configuration under construction, we check the SINR conditions, and search if there exists a feasible power value. This can be done using the following restricted version PP_TFMP$^+(\ell)$ of PP_TFMP in which $\ell = (i, j)$ is checked against the links already selected for defining the next transmission configuration, in order to identify the rate θ_r of ℓ and the power P_r of its source i.

$$[\text{PP\_TFMP}^+(\ell)] \qquad \max \qquad \sum_{r \in R} \theta_r T_{\text{SLOT}} u_\ell x_r \tag{12}$$

subject to:

$$\sum_{r \in R} x_r \leq 1 \tag{13}$$

$$p_{i'} d_{\ell'}^{-\alpha} - \beta_{r(\ell')} \sum_{\ell''=(i'',j'') \in L^{\text{SELECT}},\, i' \neq i''} p_{i''} d_{i'',j'}^{-\alpha} \geq \beta_{r(\ell')} N_0 \qquad \ell' = (i',j') \in L^{\text{SELECT}} \setminus \{\ell\} \tag{14}$$

$$p_i d_{\ell}^{-\alpha} - \beta_r \sum_{\ell'=(i',j') \in L^{\text{SELECT}}, i' \neq i} p_{i'} d_{i',j}^{-\alpha} - L_1 x_r \geq \beta_r N_0 - L_1 \qquad r \in R \tag{15}$$

$$0 \leq p_{i'} \leq P_{\max} \qquad \ell' = (i',j') \in L^{\text{SELECT}} \tag{16}$$

$$x_r \leq L_2\, p_i \qquad r \in R \tag{17}$$

$$x_r \in \{0,1\} \qquad r \in R \tag{18}$$

We now stress the particularities of PP_TFMP$^+(\ell)$. The decision variables are: x_r, the rate of link $\ell = (s_i, s_j)$ and $p_{\ell'}$, the transmission powers of all the links ℓ' in the configuration under construction (including ℓ). Constant $\beta_{r_{\ell'}}$ is the SINR threshold associated with the links ℓ' (rate $r_{\ell'}$) in L^{SELECT}. L_1 and L_2 are again large constants.

PP_TFMP$^+(\ell)$ determines whether or not a candidate link ℓ with a particular dual value can be added to the configuration under construction. PP_TFMP$^+(\ell)$ is a particular case of PP_TFMP restricted to one link, under the assumption that the links of L^{SELECT} have been selected to be part of the configuration under construction. In order to be able to add ℓ to L^{SELECT}, we need to check the SINR requirements with constraints (14) and (15), for both ℓ with respect to L^{SELECT} and for any $\ell' \in L^{\text{SELECT}}$ with respect to $(L^{\text{SELECT}} \setminus \{\ell'\}) \cup \{\ell\}$. Once the rate of ℓ is selected, under the condition that a feasible solution exists, powers of the origins of the links of L^{SELECT} need to be recomputed. Lastly, we evaluate the contribution of ℓ in the reduced cost, see (11), of PP_TFMP$^+$.

Algorithm 2 Hybrid1

Require: *Dual values, set of data rates, link lengths*

Ensure: *Either LP_OPT =* TRUE*., that is, the linear relaxation of the TFMP problem has been solved optimally, or returns a transmission configuration with a negative reduced cost.*

```
LP_OPT ← FALSE.
Call Greedy1 ⇝ L^SELECT
if RED_COST(L^SELECT) ≥ 0 then
  Solve PP_TFMP exactly.
  if RED_COST(PP_TFMP) ≥ 0 then
    LP_OPT ← TRUE.
  else
    return configuration produced by PP_TFMP.
  end if
else
  return configuration L^SELECT produced by Greedy1.
end if
```

Algorithm 2 describes the complete Hybrid 1 algorithm. First it calls the Greedy1 heuristic given in Algorithm 1. If the returned configuration has a negative reduced cost, then we add the corresponding column to the RMP. Otherwise, we call the exact pricing, that is, PP_TFMP, for finding a new configuration before moving back to the RMP. This process of invoking *Hybrid1 algorithm* is repeated until the exact pricing returns a nonnegative reduced cost.

2.3.4 Algorithm Hybrid2 - Enhanced Pricing Problem2

While Hybrid1 improves the scalability of the TFMP$^+$ model, it still requires solving an MILP (that is, PP-TFMP$^+(\ell)$) in each iteration of the Greedy1 heuristic. We next propose a new hybrid algorithm, called Hybrid2, in which we attempt to eliminate the iterative solution of an MILP in the Greedy1 heuristic. In order to do so, we use the application of the Perron Frobenius (PF) theorem to the *Power Control Problem*, as formulated in [67], in order to check the feasibility of a given configuration. We first recall that theorem, and then explain how we use it in a greedy heuristic, called the Greedy2 Heuristic. Hybrid2 is similar to Hybrid1, except for replacing Greedy1 by the Greedy2 Heuristic.

PF Theorem

Consider n transmitters with powers $P_1, P_2, ...P_n > 0$, transmitting to n receivers. Let $G_{ij} > 0$ be the path gain of transmitter j to receiver i, and β_i be the SINR threshold for receiver i. Then, the signal power at receiver i is $S_i = G_{ii}P_i$ and interference power at receiver i is $I_i = \sum_{k \neq i} G_{ik}P_k$. Assuming a null noise ($N_i = 0$), we get:

$$\text{SINR} = \frac{S_i}{I_i} = \frac{G_{ii}P_i}{\sum_{k \neq i} G_{ik}P_k} = \frac{P_i}{\sum_{k \neq i} \tilde{G_{ik}}P_k} \geq \beta_i \Rightarrow \beta_i \sum_{k \neq i} \tilde{G_{ik}}P_k \leq P_i.$$

In matrix form, it becomes:

$$\underbrace{\begin{bmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & & & \\ \cdots & 0 & 0 & \beta_n \end{bmatrix} \begin{bmatrix} 0 & G_{1j} & \cdots & G_{1n} \\ G_{21} & 0 & \cdots & G_{2n} \\ G_{i1} & G_{i2} & \cdots & G_{in} \\ \cdots & & & \\ G_{n1} & G_{nj} & \cdots & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} P_1 \\ P_2 \\ \cdots \\ P_n \end{bmatrix}}_{\underline{P}} \leq \underbrace{1}_{\lambda} \times \underbrace{\begin{bmatrix} P_1 \\ P_2 \\ \cdots \\ P_n \end{bmatrix}}_{\underline{P}}. \quad (19)$$

It leads to $A\underline{P} \leq \lambda \underline{P}$ where λ is the eigenvalue and $\underline{P}$ is the eigenvector.

An eigenvalue/eigenvector pair of the matrix $A \in R^{n \times n}$ and $A \geq 0$ satisfies the equation $A\underline{x} = \lambda \underline{x}$ where λ and x represent the eigenvalue and the corresponding eigenvector, respectively. The PF theorem says, if a matrix $A \in \mathbb{R}^{n \times n}$ and $A \geq 0$ then there is an eigenvalue λ_{pf} of A that is real and nonnegative. For any other eigenvalue λ of A, we have $|\lambda| \leq \lambda_{pf}$. As shown in [67], a set of links is SINR-feasible if the eigenvalue λ satisfies $0 \leq \lambda \leq 1$.

Heuristic Greedy2

The heuristic Greedy2 is described in Algorithm 3. We first build a list of all links, L^{SORTED} sorted in decreasing order of dual values, with one copy of the link for each possible data rate. We observe that very often dual values have similar values. Consequently, in order not to always choose the same set of links, we introduce some randomness in the link selection among the links with identical dual values. Next, we start building the configuration L^{SELECT}. In each iteration, we select the next link from L^{SORTED} to add to the configuration and use the Perron-Frobenius theorem described above to check if the eigenvalue $0 < \lambda \leq 1$, which implies the feasibility of the configuration. When no more links can be added, the corresponding eigenvector gives the powers to be used by the links. If all the powers are less than the

maximum power then we can consider the configuration under construction to be SINR feasible.

Algorithm 3 Heuristic Greedy2

Require: *Set of dual values.*

Ensure: *Status:* FALSE/TRUE*; if* TRUE $\rightsquigarrow$ *a potential configuration with its set of links, and for each link: power value and data rate.*

```
N(ℓ) ← set of links that share an endpoint with ℓ
for all links ℓ and all data rates θ do
    key(ℓ^θ) = θ× dual value of ℓ
    L ← L ∪ {ℓ^θ}
end for
L^SORTED ← L sorted in decreasing order of key(ℓ^θ).
if MostDualsAreSame then
    L^SORTED ← randomly permute links of same dual value in L^SORTED
end if
ℓ ← first element of L
L^SELECT ← {ℓ}
L^SORTED ← L^SORTED \ (N(ℓ) ∪ {ℓ})
while |L^SORTED| > 0 do
    ℓ ← Next(L^SORTED)
    Determine whether isSINRFeasible (i.e., SINR ≥ β_r) is true/false using Perron-
Frobenius theorem on L^SELECT ∪ {ℓ}
    if isSINRFeasible then
        L^SELECT ← L^SELECT ∪ {ℓ}
        L^SORTED ← L^SORTED \ N(ℓ)
    end if
    L^SORTED ← L^SORTED \ {ℓ}
end while
if RED_COST(L^SELECT) < 0 then
    return TRUE
else
    return FALSE
end if
```

For completeness we give the pseudocode for Hybrid2 in Algorithm 4.

2.3.5 ILP Solution of the TFMP$^+$ Problem

The CG technique solves the linear relaxation of ILP (3)-(10), while we need to obtain integer values for variables λ_c and f_{ij}, and binary values for variables y_{ik}. Recall that the coverage variable y_{ik} are decision variables that indicates whether or not a target is covered by a given sensor (0 or 1), and the flow variable f_{ij} counts the number of data packets transmitted on

Algorithm 4 Hybrid2

Require: *Dual values, set of data rates, link lengths*
Ensure: *Either LP_OPT =* TRUE*., that is, the linear relaxation of the TFMP problem has been solved optimally, or returns a transmission configuration with a negative reduced cost.*
LP_OPT ← FALSE.
Call Greedy2 $\rightsquigarrow L^{\text{SELECT}}$
if RED_COST(L^{SELECT}) ≥ 0 **then**
  Solve PP_TFMP exactly.
  if RED_COST(PP_TFMP) ≥ 0 **then**
    LP_OPT ← TRUE.
  else
    return configuration produced by PP_TFMP.
  end if
else
  return configuration L^{SELECT} produced by Greedy2.
end if

link (s_i, s_j) during the TDMA frame. Indeed, a data packet should reach the sink as a whole entity without costly fragmentation.

In order to derive an ILP solution, we solve exactly the last generated RMP with integer requirements for variables λ_c, f_{ij} and y_{ik}. This results in an upper bound $\tilde{z}_{\text{ILP}}$ on the optimal ILP value of (3)-(10), denoted by $z^{\star}_{\text{ILP}}$. Let $z^{\star}_{\text{LP}}$ denote the optimal LP solution of the TFMP$^+$ problem. Note that $z^{\star}_{\text{LP}} \leq z^{\star}_{\text{ILP}} \leq \tilde{z}_{\text{ILP}}$, and that the accuracy ε of solution $\tilde{z}_{\text{ILP}}$ is given by:

$$\varepsilon = \frac{\tilde{z}_{\text{ILP}} - z^{\star}_{\text{LP}}}{z^{\star}_{\text{LP}}}.$$

In our optimization model we use three pricing algorithms namely PP_TFMP pricing, Hybrid1 pricing and Hybrid2 pricing. We name these three optimization models as TFMP$^+$, Hybrid1 and Hybrid2 optimization models respectively. All the three optimization models produce an optimal CG solution. In some cases, in our experiments, due to slow convergence, we needed to stop the solution process of the LP solution before the optimality condition was satisfied. In such a case, we estimated the solution accuracy with ε_1, computed as follows:

$$\varepsilon_1 = \frac{\tilde{z}_{\text{ILP}} - z^{\text{best}}_{\text{LP}}}{z^{\text{best}}_{\text{LP}}},$$

where $z^{\text{best}}_{\text{LP}}$ is the best estimate we obtained for $z^{\star}_{\text{LP}}$. Note that $z^{\star}_{\text{LP}} \leq z^{\text{best}}_{\text{LP}}$, and that ϵ_1 is not an upper bound on the solution accuracy, but merely an estimate.

2.4 Numerical Results

In this section, we discuss the numerical experiments we conducted in order to evaluate the performance of our Hybrid1 and Hybrid2 solution schemes. We compare their performance with the TFMP optimization model of [44] using PP_TFMP pricing, In Section 2.4.1, we describe the data instances. In Section 2.4.2, we compare the accuracy of the three schemes for different values of q (q-coverage). Here we restrict our experiments to single-rate sensors, and assume that each sensor generates a data packet each time it monitors a target. In Section 2.4.3, we analyze the performance of all three algorithms for multi-rate sensors, and in Section 2.4.4 we analyze the performance of the Hybrid1 algorithm when we scale our model to multiple packets from each target. Finally, in Section 2.4.5, we compare the power characteristics of the solutions provided by all three algorithms.

2.4.1 Data Instances

Sensors and targets were uniformly and independently deployed over a square area of 625 meters side length. We fixed the number of targets to 100 nodes and then varied the number of sensors from 40 to 110 nodes. We also considered three different coverage levels to provide reliability and redundancy. We considered a single sink in all experiments. During the generation of input data instances, we ensure that each target/sensor has at least one path to the sink node. We also ensure that all the input data instances have at least q sensors within the maximum transmission range of each target (q-coverage). This ensures that we can use the same data instance to compare the performance of different coverage levels. All presented results correspond to averages over 10 data instances.

The model was implemented in Java using the CPLEX Concert Technology (version 12.6). We ran all our experiments on a cluster with 12 GB of memory. Similar to the parameters used in the literature e.g., [44] and out-door sensor specifications, we set the maximum transmission power to 13 mW, the path loss exponent to $\alpha = 2$, and the noise to 10^{-6} W. The sensing range was fixed to 150 meters. On the basis of the modulation and coding schemes proposed in [54] as an extension to the 802.15.4 standard, the possible transmission rates are $b_k = \{250$ kb/s, 500 kb/s, 1 Mb/s, 2 Mb/s$\}$ and require the following SINR thresholds $\beta_k = \{1.3, 2.0, 4.0, 10.0\}$. Note that we considered the standardized case of a single available data rate, which then corresponds to 250 kb/s. We assume single packet size $\sigma = 1{,}000$-bytes as is standard in the literature. Recall that our model allows for larger-sized packets, by resetting the value of $\hat{\sigma}$ in Constraint 7, and for multiple packets per target by changing the parameter $pkts_k$; this will be used in the experiments in Section 2.4.4.

2.4.2 Comparison of the solution accuracies and computational times of TFMP, Hybrid1 and Hybrid2 algorithms

In this section, we compare the performance of the previously proposed TFMP algorithm [44] with the Hybrid1 and Hybrid2 algorithms given in Section 2.3.3 and 2.3.4. Recall that the objective of TFMP, Hybrid1 and Hybrid2 algorithms is to find the minimum frame length. We therefore compare how these algorithms perform under similar settings. Table 1 shows the frame length achieved when the algorithms are restricted to using a single data rate (250 *kbps*), while Table 2 reports the average computational times(in seconds).

Table 1: Average number of time slots with q-coverage (10 instances) - Single rate

# sensors	q=1								
	TFMP			Hybrid1			Hybrid2		
	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$
40	1,713.7	1,714.0	0.0	1,713.7	1,714.4	0.0	1,713.7	1,714.0	0.0
50	1,452.1	1,453.3	0.0	1,453.0	1,454.1	0.1	1,452.1	1,453.2	0.0
60	1,131.5	1,132.7	0.1	1,131.5	1,132.7	0.1	1,131.5	1,132.9	0.1
70	1,057.9	1,058.6	0.1	1,057.9	1,059.7	0.2	1,057.9	1,059.2	0.1
† 80	944.5	946.9	0.2	945.3	947.6	0.2	945.1	947.3	0.2
† 90	885.1	888.2	0.4	884.7	887.9	0.4	890.9	893.3	1.0
†100	1,077.9	1,081.3	23.5	875.6	878.8	0.4	874.2	877.9	0.3
†110	1,959.6	1,960.7	115.1	1,090.0	1,092.0	19.8	911.4	915.6	0.5
# sensors	q=2								
	TFMP			Hybrid1			Hybrid2		
	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$
40	3,588.8	3,589.2	0.0	3,588.8	3,589.2	0.0	3,588.8	3,589.2	0.0
50	3,064.6	3,065.5	0.0	3,064.6	3,065.4	0.0	3,064.6	3,065.4	0.0
60	2,376.9	2,378.1	0.0	2,376.9	2,378.2	0.1	2,376.9	2,378.1	0.0
70	2,252.9†	2,255.4	2.1	2,208.9	2,211.4	0.1	2,208.9	2,210.3	0.1
† 80	1,955.3	1,957.1	0.0	1,956.4	1,958.3	0.1	1,954.5	1,956.2	0.0
† 90	1,822.4	1,825.2	0.4	1,818.5	1,820.9	0.1	1,823.9	1,826.6	0.4
†100	1,935.3	1,938.1	7.4	1,804.0	1,807.6	0.2	1,807.1	1,810.9	0.4
†110	4,065.6	4,066.8	115.2	1,883.8	1,887.7	0.2	1,889.5	1,892.9	0.5
# sensors	q=3								
	TFMP			Hybrid1			Hybrid2		
	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$	$z^{\star}_{\text{LP}}/\dagger\, z^{\text{best}}_{\text{LP}}$	$\tilde{z}_{\text{ILP}}$	$\varepsilon/\dagger\,\varepsilon_1$
40	5,635.6	5,635.6	0.0	5,635.6	5,635.6	0.0	5,635.6	5,635.6	0.0
50	4,820.1	4,821.0	0.0	4,820.1	4,821.1	0.0	4,820.1	4,821.0	0.0
60	3,734.4	3,736.0	0.0	3,734.4	3,736.0	0.0	3732.9	3,734.7	0.0
70	3,434.9†	3,436.4	0.2	3,430.9	3,433.1	0.1	3,430.9	3,432.3	0.0
† 80	3,005.5	3,007.0	0.0	3,005.9	3,007.9	0.1	3,008.5	3010.0	0.1
† 90	2,951.5	2,954.4	5.6	2,797.9	2,800.7	0.1	2,809.6	2812.7	0.5
†100	2,787.4	2,790.6	1.9	2,739.5	2,742.5	0.1	2,783.3	2,786.2	1.7
†110	6,580.2	6,581.8	125.8	3,089.3	3,092.7	6.1	2,915.1	2,918.5	0.1
† LP solution of the RMP is stopped when the solution of PP takes more than 12h									

For smaller topologies, we consider accuracy ϵ using the lower bound as given by $z^{\star}_{\text{LP}}$.

For larger input instances, as discussed in Section 2.3.5, we consider accuracy ε_1 using best found lower bound, as given by $z_{\text{LP}}^{\text{best}}$, with a time limit of 12h. In Table 1 we provide the accuracies of our three solutions for (q-coverage) with $q = 1, 2, 3$. We can see that all these three algorithms provide very good accuracies or $\varepsilon/\varepsilon_1$ optimal solutions. To measure the scalability of our algorithms, we look at the results of the large input instances, as for smaller instances all three algorithms provide an identical lower bound. We can see that the two hybrid algorithms are comparable. However, they both converge faster and produce solutions with better accuracies for 1, 2, and 3-coverage. % q (q-coverage).

Table 2: Average computational times with q-coverage (seconds) - Single Rate

	TFMP			Hybrid1			Hybrid2		
# sensors	$q=1$	$q=2$	$q=3$	$q=1$	$q=2$	$q=3$	$q=1$	$q=2$	$q=3$
40	59.7	64.4	72.7	44.6	54.5	54.4	52.9	61.4	70.6
50	276.7	335.0	435.1	226.0	285.5	377.6	406.3	660.8	952.8
60	1,310.9	2,016.5	2,905.9	1,150.8	1,887.6	2,510.9	2,889.0	4,560.8	5,495.9
70	10,968.9	11,368.7	15,486.0	10,924.8	13,527.9	16,680.7	14,961.0	16,337.2	19,068.5
†80	26,679.2	29,521.7	42,074.2	25,650.0	29,833.4	37,843.9	29,811.5	31,064.9	39,426.3
† 90	35,078.2	43,905.7	38,762.4	33,139.2	42,718.1	42,615.2	32,543.3	43,858.3	45,480.8
† 100	42,763.9	89,937.3	44,158.3	41,160.8	41,520.6	41,517.3	37,926.4	42,893.7	43,858.3
† 110	44,443.1	44,088.7	45,348.8	44,411.5	45,643.1	46,021.3	46,713.7	46,875.7	49,010.4
† LP solution of the RMP is stopped when during the solution of PP, we reach the time limit (12h)									

Table 2 reports the average time of 10 problem instances for all three TFMP, Hybrid1 and Hybrid2 algorithms. The time taken by the hybrid algorithms matches with that of TFMP$^+$ for smaller instances. For larger instances ($n \geq 80$), the LP solution of the RMP is stopped when the PP takes more than 12 hours. So the time taken by all three algorithms is comparable for large instances. However, Hybrid1 and Hybrid2 converge to much better solutions in this time. We may observe that the time taken by Hybrid1 algorithm is less than that of Hybrid2 algorithm. Another interesting observation is that as the coverage level increases, cpu time only increases slightly, even though traffic increases by a factor of q.

Table 3: Analysis of the Number of Generated Configurations - Single Rate

# sensors	TFMP					
	q= 1		q= 2		q= 3	
	Overall #	# selected	Overall #	# selected	Overall #	# selected
40	34.7	17.3	37.5	19.6	42.3	22.7
50	65.7	22.6	76.6	25.5	86.9	28.2
60	131.3	29.1	156.2	33.1	171.3	37.0
70	230.0	38.9	200.8	38.8	237.4	43.3
† 80	258.0	42.3	265.0	44.2	284.6	47.9
† 90	256.9	39.4	260.7	43.4	237.7	44.4
† 100	149.4	34.4	140.1	34.0	158.6	36.3
† 110	50.7	32.1	58.7	42.4	62.0	48.8

# sensors	Hybrid1								
	q= 1			q= 2			q= 3		
	Overall #	# selected	# exact PP	Overall #	# selected	# exact PP	Overall #	# selected	# exact PP
40	44.9	18.9	17.2	50.0	21.7	21.5	51.5	23.5	23.0
50	87.4	23.6	43.6	94.2	27.4	52.4	108.8	29.9	62.0
60	167.3	30.5	101.9	203.9	33.0	123.0	208.7	37.1	131.6
70	284.2	38.1	189.3	296.4	42.8	200.2	303.5	44.4	214.8
† 80	316.9	40.5	206.4	335.3	45.2	235.6	348.0	46.8	237.7
† 90	346.4	39.7	204.4	359.4	43.9	219.3	365.0	45.2	204.7
† 100	290.3	35.7	133.0	262.4	37.8	125.2	285.6	38.6	137.0
† 110	180.0	33.3	73.6	211.0	33.0	85.8	173.3	35.6	61.9

# sensors	Hybrid2								
	q= 1			q= 2			q= 3		
	Overall #	# selected	# exact PP	Overall #	# selected	# exact PP	Overall #	# selected	# exact PP
40	50.3	19.1	18.5	61.4	22.2	22.0	63.1	24.6	24.0
50	112.3	24.2	48.0	111.8	25.8	54.9	128.8	30.3	67.1
60	175.2	28.9	102.7	206.1	33.8	135.4	218.4	37.4	142.6
70	295.5	38.4	202.6	286.9	40.7	202.5	313.1	45.3	228.2
† 80	325.8	40.8	231.3	337.9	44.0	239.6	352.1	48.0	253.4
† 90	329.0	39.1	224.1	335.2	44.3	231.3	309.5	44.8	213.0
† 100	263.4	34.3	150.1	247.9	36.2	146.8	255.7	36.8	146.1
† 110	191.2	31.6	91.1	155.4	34.9	75.6	133.9	38.9	63.6

† LP solution of the RMP is stopped when the solution of PP takes more than 12h

Table 3 gives the total number of iterations generated by TFMP and hybrid solutions, the number of selected columns, as well as the number of times the original pricing problem is called in the "# exact PP" column. We can see that particularly for larger data instances, the total number of generated columns is much smaller for TFMP than both the Hybrid1 and Hybrid2 algorithms; this explains the poorer quality of its solutions. The number of selected columns (among all the generated columns) in the final solution are shown in the 'selected' column. We observe that the number of selected columns is higher for both our algorithms compared to TFMP; this means they produce higher number of configurations compared to TFMP. Finally, a larger value in the "# exact PP" column implies a higher solution time, as the expensive exact pricing is being called more often.

2.4.3 Comparison of single and multi-rate networks

Table 4: Comparison of Single and Multi-rate average number of slots

	TFMP			Hybrid1			Hybrid2		
# sensors	Single	Multi	Gain	Single	Multi	Gain	Single	Multi	Gain
40	1,714.0	1,377.0	24.5	1,714.4	1,377.0	24.5	1,714.0	1,377.0	24.5
50	1,453.3	1,224.6	18.7	1,454.1	1,160.0	25.4	1,453.2	1,159.9	25.3
60	1,132.7	897.7	26.2	1,132.7	897.7	26.2	1,132.9	897.6	26.2
70	1,058.6	910.0†	16.3	1,059.7	818.3	29.5	1,059.2	818.0	29.5
† 80	946.9	1,286.0	-26.4	947.6	733.8	29.1	947.3	752.6	25.9
† 90	888.2	1,711.0	-48.1	887.9	697.8	27.2	893.3	694.4	28.6
†100	1,081.3	1,701.8	-36.5	878.8	700.9	25.4	877.9	690.6	27.1
† LP solution of the RMP is stopped when during the solution of PP, we reach the time limit (12h) Gain $= 100 \times (\overline{z}_{\text{ILP}}^{\text{single}} - \overline{z}_{\text{ILP}}^{\text{multi}})/\overline{z}_{\text{ILP}}^{\text{multi}}$									

Table 4 compares the time-slots required for solutions using single and multi-rate sensors for all three algorithms. For larger topologies we stop our computation after a certain time as explained earlier. As multi-rate solutions are associated with an increase in the combinatorial aspect of the TFMP model, the quality of the solution within the time constraint is not so good. Consequently, we do not see improvement over single rate solutions for the case of TFMP. However, both Hybrid1 and Hybrid2 obtain significant improvements using multi-rate sensors. In particular, we gain around 25 percent of time slots using multi-rate solutions over single rate. Thus, the scalability of our two Hybrid1 and Hybrid2 solutions is much better.

Table 5: Average computational times with q-coverage (seconds) - Multiple rates

Sensors	TFMP	Hybrid1	Hybrid2
40	394.7	287.8	383.6
50	1,846.8	1,542.8	2,084.2
60	10,627.8	9,119.0	12,539.0
70	35,119.2	36,970.7	35,747.6
† 80	48,232.0	42,717.8	41,559.2
† 90	57,568.3	48,694.1	51,393.7
†100	57,544.7	56,911.7	56,943.3
† LP solution of the RMP is stopped when during the solution of PP, we reach the time limit (12h)			

Table 5 shows the time taken by all three algorithms for multi-rate sensors. Comparing Tables 2 and 5, we see that multi-rate solutions take a longer time than single rate solutions. This is to be expected as the number of combinations to be considered is much more. This appears to be the cost to be paid for the better quality of multi-rate solutions.

2.4.4 TFMP$^+$: One vs. several packets per target

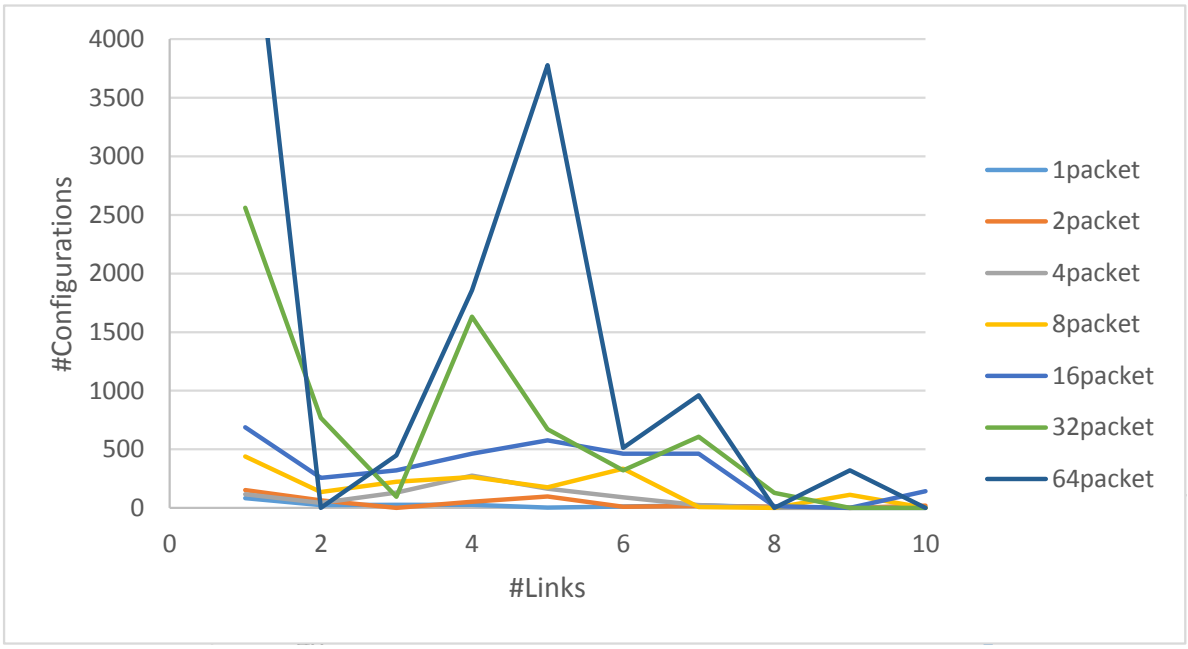
In this section, we report on two different experiments to investigate the effect of increasing the amount of traffic in the network on the number of links in the generated configurations. In both cases we used the Hybrid1 algorithm, and considered 40 single-rate sensors monitoring 100 targets.

In the first experiment, we assume that we have multiple readings at each target and that each reading is of size σ and can fit in one time slot. Different readings can therefore take different paths to the sink. Our results are summarized in Figures 2(a) and 3(a). We varied the number of readings/packets collected from each target by changing the parameter $pkts_k$ from 1 to 64 for each simulation.

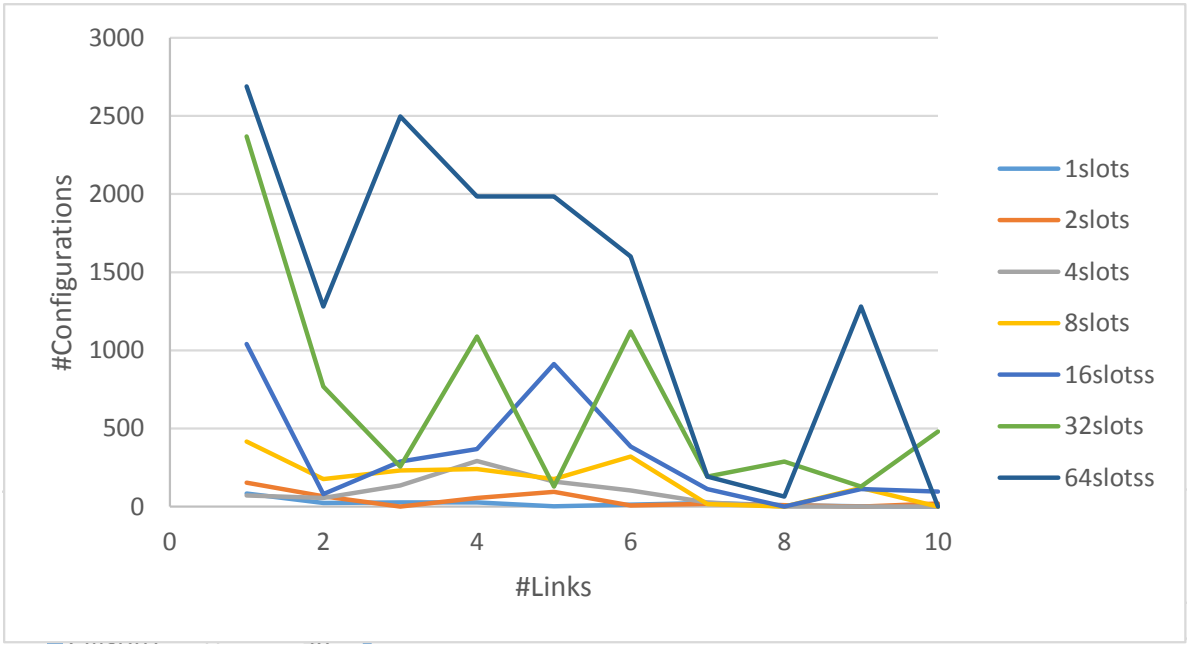
In the second experiment, we consider the scenario when there is one reading per sensor, but it needs to be fragmented and needs multiple time slots to be sent to the next hop, where it is reassembled before being forwarded to the next hop. Thus all packets corresponding to a sensor reading use the same path to the sink. For that purpose, we increased the parameter $\hat{\sigma}$, i.e., we use $\hat{\sigma} = \sigma, 2\sigma, 4\sigma, \ldots 64\sigma$ in (7). Our results are summarized in Figures 2(b) and 3(b). Observe that this corresponds to varying the number of slots from 1 to 64 for each reading to be forwarded to the next hop. .

In Figure 2, we depict the number of configurations on the vertical axis, and the size (i.e., number of transmitting links) of the configurations on the horizontal axis, while in Figure 3, the percentage of links in each configuration is on the vertical axis, and the configuration index is on the horizontal axis, where each block of vertical bars is associated with the number of slots it takes to transmit one packet.

In Figures 2 and 3, we can see that, as traffic increases, we can see a larger percentage of larger configurations, that is, configurations with larger number of links. This implies greater spatial reuse, as higher number of links can be scheduled simultaneously. We can therefore expect that with a dynamic traffic stream, we would see greater efficiency of reuse.

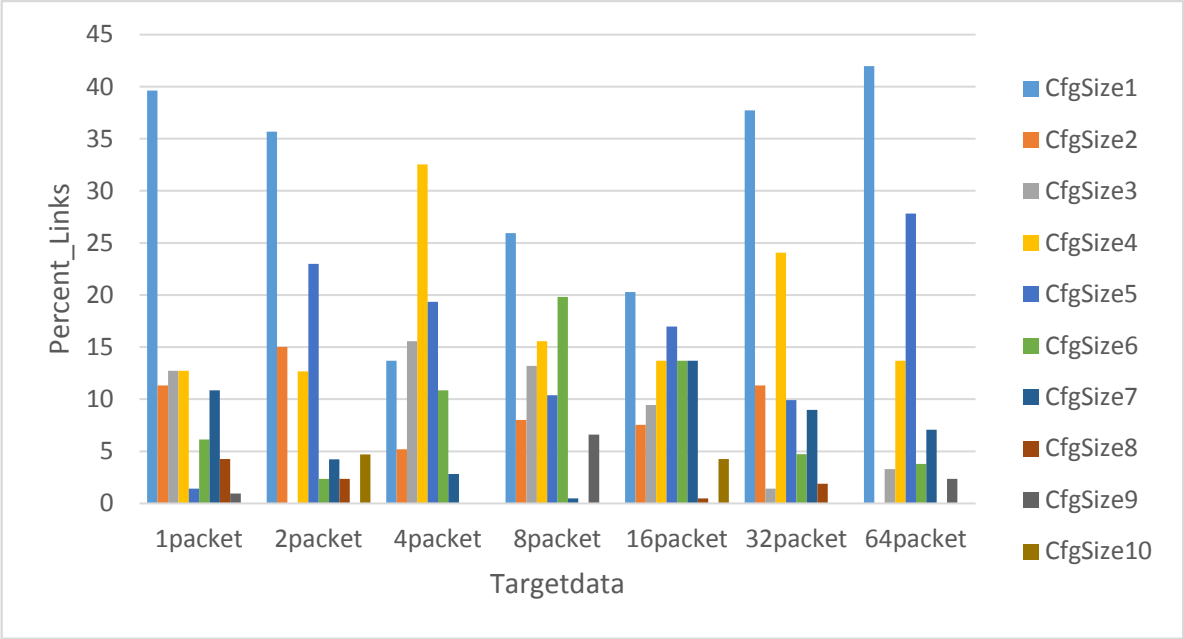


(a) Multiple readings from each target

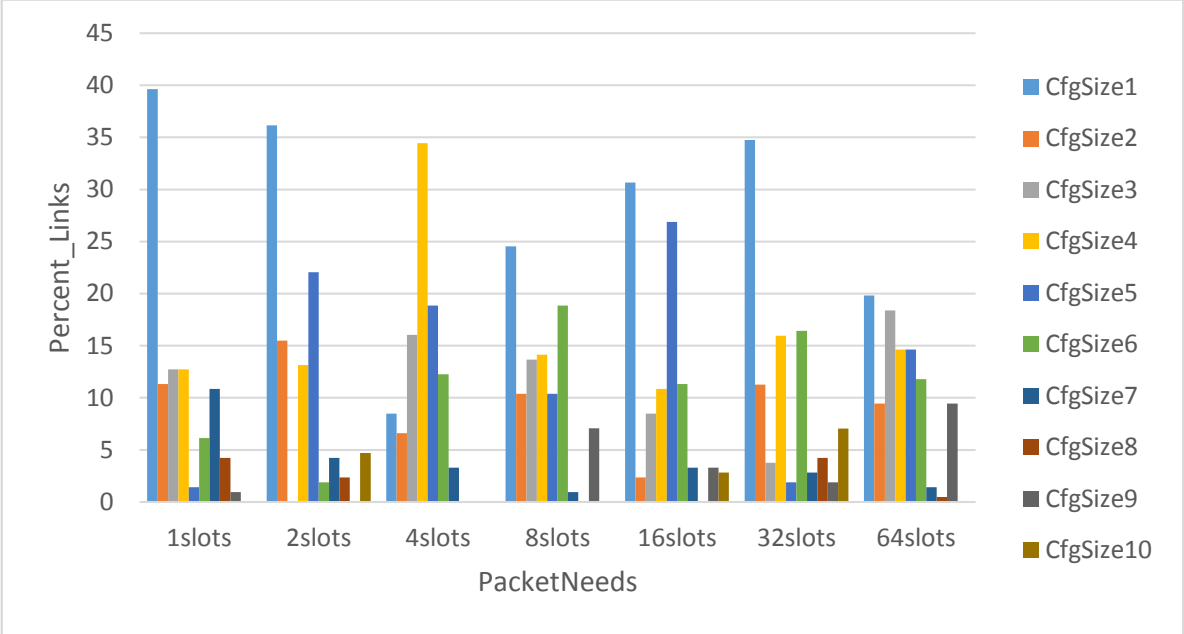


(b) Multiple time slots to send a reading

Figure 2: Distribution of the sizes of the configurations



(a) Multiple readings from each target



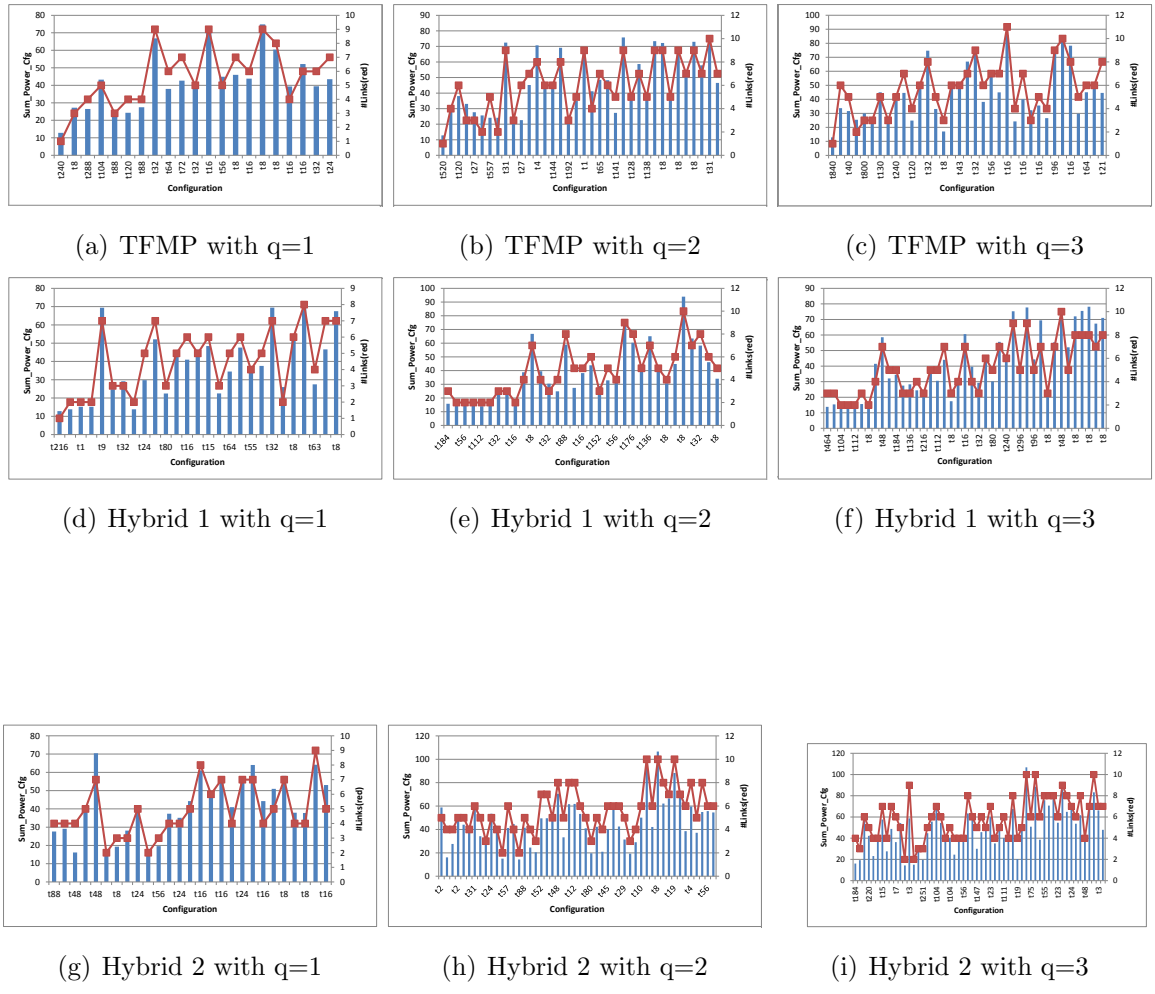
(b) Multiple time slots to send a reading

Figure 3: Distribution of the links among the configurations

2.4.5 Power Characteristics of the Transmission Configurations

In this section, we analyze various characteristics of the transmission configurations generated by the three algorithms: TFMP, Hybrid1, and Hybrid3. All results in this section are from

simulation runs for 70 single-rate sensors collecting data from 100 targets. Figure 4 compares the transmission power and number of links in each configuration for TFMP, Hybrid1 and Hybrid2. The x-axis shows specific configurations used, together with the number of times it is used in the solution. The red line corresponds to the number of links in that configuration. The blue bars represent the sum of powers in that configuration. As seen in Figure 4, the total power used by links in a configuration is, in general, proportional to the number of links in that configuration. We conclude that the power used by all links is similar on average, for all configurations, in all algorithms. This is a good indication of a good solution. Finally, we computed the total power used by all links used in all configurations (including multiple copies of every configuration). This represents the energy cost of the solution. Our results show that this cost is similar for the solutions produced by all three algorithms, for $q = 1, 2, 3$.



(a) TFMP with q=1 (b) TFMP with q=2 (c) TFMP with q=3

(d) Hybrid 1 with q=1 (e) Hybrid 1 with q=2 (f) Hybrid 1 with q=3

(g) Hybrid 2 with q=1 (h) Hybrid 2 with q=2 (i) Hybrid 2 with q=3

Figure 4: Total power per configuration (70 single-rate sensors, 100 targets)

Table 6: Number of configurations of a given length (70 single-rate sensors, 100 targets)

	q=1			q=2			q=3		
# Links	TFMP	Hybrid1	Hybrid2	TFMP	Hybrid1	Hybrid2	TFMP	Hybrid1	Hybrid2
1	240	216	0	520	0	0	840	0	0
2	0	33	120	32	336	181	136	328	226
3	96	487	160	403	976	184	1072	1704	551
4	512	118	320	112	432	422	312	448	970
5	192	265	184	1012	248	584	602	1048	769
6	128	48	104	379	256	291	646	272	548
7	104	160	144	154	152	154	355	248	200
8	8	9	16	9	296	130	69	40	217
9	56	0	32	76	24	35	128	48	11
10	0	0	0	31	8	0	8	48	51
11	0	0	0	0	0	0	16	0	0
Average Conf. Length	4.2	3.8	4.6	4.3	4.3	4.7	4.0	4.2	4.8

Table 6 gives the number of configurations of a given length. For $q = 1$, the average configuration length of Hybrid1 (3.8) is less than that of TFMP(4.2) and Hybrid2(4.6). Hybrid2 in general produces larger configurations. These three algorithms use different average configuration lengths, and yet produce similar same frame size for smaller networks.

2.5 Conclusion

In this paper we studied the TDMA Frame Minimization problem to achieve ConvergeCast in wireless sensor networks. We designed an optimization model to derive schedules with minimum length, hence maximizing network throughput. By leveraging advanced networking capabilities offered by current sensors, a wide range of network parameters are considered in the solution related to coverage, routing, power control and rate adaptation. Since the straightforward formulations of these problems are NP-hard, we introduced a computationally feasible column-generation-based method to compute near-optimal solutions. Furthermore, since the underlying pricing problem remains NP-hard, we proposed two algorithms that scale up to problems with larger sizes. Both our algorithms outperform the algorithm given in [44] for large topologies, and are therefore more scalable. Furthermore, we describe two realistic scenarios in which each sensor monitoring a target produces multiple packets relating to the target, and described how to modify our algorithms to support such scenarios. We performed a comprehensive analysis of the solutions produced by our algorithms.

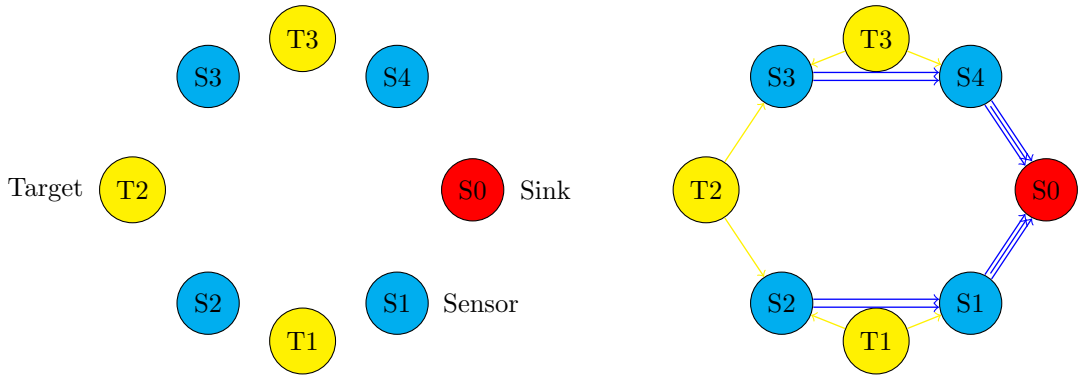
Chapter 3

TDMA Scheduling in Wireless Sensor Networks

M. Bakshi, M. Kaddour, B. Jaumard, and L. Narayanan. TDMA scheduling in wireless sensor networks. submitted for publication, 2017. An extended abstract of this paper has been published in IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). [2]

3.1 Introduction

A wireless sensor network (WSN) is a network of spatially distributed autonomous sensors that can monitor physical or environmental conditions, such as temperature, sound, or humidity, and can communicate with each other using wireless transmissions. An important application of sensor networks concerns *target coverage*, where the sensor network is tasked with periodically collecting data about a given set of targets and sending the data to a central *sink node* in the network (see Figure 5(a)). The resulting many-to-one communication pattern in which data from a set of sources is to be routed to a common sink, is often referred to as *convergecast* (see Figure 5(b)). Depending on the environment and application, only a subset of sensor nodes might be close enough to the targets to monitor them, and in some cases, in order to ensure reliability, we might require q-coverage, that is, every target is to be monitored by q sensors.



(a) Input (b) MLCS Output

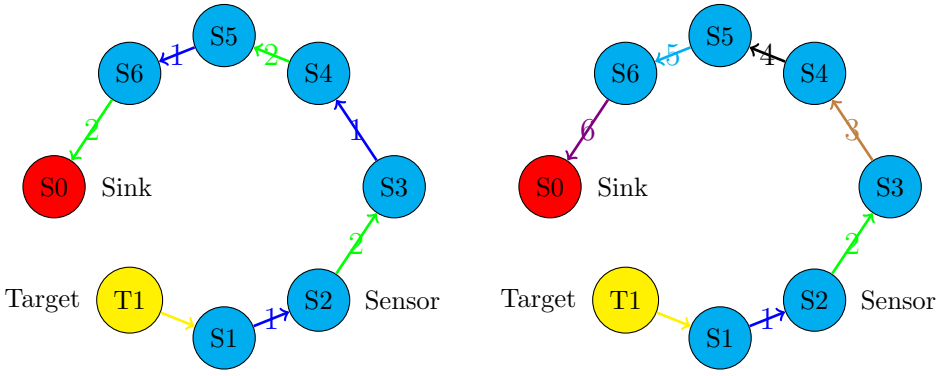
Figure 5: convergecast Problem Significance (q=2)

A central issue in the design of sensor networks for such applications is the question of medium access control (MAC). As in all wireless networks, the nodes share the access medium, and there needs to be an access control protocol that arbitrates access to the medium in a fair and efficient manner. MAC protocols can be broadly classified into *contention-based* (IEEE 802.11) and *schedule-based* (e.g., FDMA, TDMA) protocols. Although conventional wisdom holds that contention-based protocols are more suitable for WSN since the nodes are designed to work in an autonomous and distributed manner, some recent work in the literature such as [39] indicates that centrally-coordinated network and MAC layer protocols are at least as efficient as distributed protocols in numerous settings, while bringing several advantages such as code simplicity, ease of management or observability. In particular, many WSN applications, including target coverage applications, exhibit a regular traffic pattern by periodically collecting sensor measurements at a centralized sink. In this context, TDMA (Time Division Multiple Access) offers a convenient multiple access scheme at the MAC layer since it guarantees high bandwidth utilization and low energy consumption. In TDMA, the time is divided into frames each containing a certain number of fixed size slots. Typically, a central entity is responsible to define a frame schedule assigning each node a fixed number of slots for transmitting and receiving data. Moreover, several transmission links can be scheduled in the same slot if no harmful level of interference occurs among them.

In this paper, we consider the problem of finding an efficient TDMA schedule for a convergecast communication pattern in a given WSN. We use the *physical or SINR model* of interference: transmission is successful on a link if the signal to interference and noise ratio is above a certain threshold. Finding a schedule that minimizes the number of slots to achieve convergecast is NP-hard as shown in [10]. The previous work on this problem has taken one

of two approaches. The first approach uses mathematical programming to derive a multiset of *transmission configurations*, each of which consist of a set of links in the network that can transmit simultaneously without significant interference. Column generation is typically used as a technique to speed up the computation. A routing sub-graph is obtained as a by-product of the computation. The second approach is to divide the problem into that of finding a *convergecast tree*, and then specify a scheduling algorithm to route packets along the edges of the tree.

The advantage of deriving transmission configurations is that each transmission configuration can be scheduled without any further need to check for the possibility of interference. It is also the case that the multiset of transmission configurations suffices to cover all links necessary to achieve the convergecast. However, a serious deficiency in this approach is that a multiset of transmission configurations does not constitute a *schedule* for transmission, as there is no implied order between the configurations. Indeed, it may even be impossible to achieve convergecast using only the so-called optimal set of transmission configurations. For example, consider a sensor network with a path topology with n nodes from the sole target to the sink. The optimal set of transmission configurations may contain only two configurations (all even numbered nodes transmitting in the first configuration and all odd-numbered nodes in the second, see Figure 6(a)), but any schedule must use at least $n-1$ slots (see Figure 6(b)), as that is the distance from the source to the sink. We conclude that the number of transmission configurations is only a *lower bound* on the length of the schedule, and in fact a very weak lower bound in the worst case.



(a) CG Result (2 slots)

(b) Schedule (6 slots)

Figure 6: Scheduling Example

3.1.1 Our Results

In this paper, we investigate several scheduling questions. Given a set of transmission configurations that can potentially achieve convergecast, how do we schedule these configurations? How useful is it to use only the set of precomputed transmission configurations in finding an efficient schedule for convergecast, as compared to scheduling maximal sets of links on the fly based on the current traffic situation? While many convergecast algorithms in the literature restrict themselves to routing on a tree structure, is there any advantage to be gained by relaxing this restriction and routing instead on a subgraph of the original network? If restricting to routing on a tree, how much difference does the choice of tree make?

We study and compare two different approaches to the problem of convergecast. In the first approach, which we call the TC approach, we provide a new *column generation* model to obtain an optimal or near-optimal set of transmission configurations that restricts the routing graph to be a tree. This is an extension of the TFM model given in [44], where the routing graph is not necessarily a tree. Given a set of transmission configurations that constitute a feasible solution to convergecast, we give an ILP to find an optimal schedule using only the given set of TCs. We also give several scheduling heuristics that are non-optimal but very efficient in practice. Note that both the ILP and the heuristics can be used both for tree-based and routing subgraph-based solutions.

In the second approach, which we call the two-phase approach, we first build a routing tree or sub-graph, and then schedule transmissions along this tree/sub-graph, in each round choosing a set of non-interfering links among those that have available traffic. We use two previously defined trees [32], [29] and also give a new tree, called the TFM tree which takes into account both the SINR interference model and the possibility of varying power levels. We also consider the routing subgraph defined by the TFMP$^+$ model in [7]. We give a new scheduling algorithm called the ROS algorithm which uses an ILP to find the optimal set of links to schedule in every round. We also significantly modify and generalize the algorithm in [32] to obtain a new algorithm called EMWF that (a) schedules only links that have data (b) works with routing graphs rather than routing trees (c) deals with q-coverage and multi-rate sensors. In contrast, the algorithm of [32] does not consider whether or not a link has available data before scheduling it, only considers routing tees, single-rate sensors, and 1-coverage.

We ran extensive experiments to compare both approaches. Our results can be summarized as follows:

1. Two-phase approaches are better, i.e. produce shorter schedules, than TC-based approaches for single-rate sensors when there are one or very few packets to be routed

per sensor. For the TC-based approach, the best results are obtained by our w_3-based heuristic, which prioritizes the configuration with the highest amount of traffic left to forward. For the two-phase approach, both scheduling algorithms have similar performance in terms of number of slots, though the EMWF algorithm is more computationally efficienti. The TFM tree is the best data gathering tree for both our two-phase scheduling algorithms.

2. Schedules based on routing subgraphs are better than those restricted to trees, for both approaches. Two-phase approaches do even better on routing subgraphs relative to the TC-based heuristic.

3. For multi-rate sensors or when there are many packets to forward, the TC-based heuristic performs better than two-phase algorithms.

3.2 Literature Survey

The problem of determining a minimum-length schedule that satisfies given traffic demands as well as SINR constraints is NP-hard as shown in [10]. We categorize the existing work on the problem into approaches using mathematical programming and approaches that use heuristics, some of which have proven approximation ratios.

Several papers use mathematical programming to solve scheduling problems in wireless networks. Algorithms for routing and scheduling in wireless multi-hop networks considering SINR constraints were given by [51]. [76] studied the same problem considering multiple channels. [44] considers q coverage by wireless sensor networks, and scheduling specifically for the resulting convergecast operation, and takes advantage of multiple power levels; improvements to this approach were given in [7]. In all these papers, the final output of the algorithms is a multiset of transmission configurations. They do not describe in what order to schedule these configurations, and as mentioned in Section 1, sometimes it is impossible to achieve a schedule using only the output multiset of configurations.

Similarly, for joint scheduling and routing, many heuristics have been proposed; see for example [48], [31], [40], [12], [70], [17], [58], [61], [13], [53], [45] [36], [47] and [32]. Of these, [48], [40], [12], [70], [36], [47] and [32] use the SINR model; the remaining papers use the protocol model. The authors of [48], [36], and [47] proposed approximation algorithms for multi-hop networks assuming unlimited transmission power, constant power and limited transmission power respectively. But they deal with arbitrary traffic patterns, not converge cast. The authors of [40, 12, 70, 32] deal with convegercast in the SINR model. [40] provides heuristics for convergecast based on trees using the SINR model. They claim to show how to

use multiple frequencies to eliminate interference. Two convergecast heuristics using Dijkstra and graph coloring are provided in [12]. In [70], nodes are divided into clusters and a non-linear optimization model is given to get a Convergecast solution using the SINR model. None of the algorithms in [40, 12, 70, 32] consider multi-rate sensors or q-coverage. As well, they do not provide any bounds on the accuracy of their solutions. As a result, they are not directly comparable to our work.

We now describe in a bit more detail two papers which we use in our experimental evaluation. [32] use a two-phase approach in the SINR model. First they construct a tree named *Low Latency High Compatibility (LLHC)* in which they attempt to minimize the degree of nodes in the entire tree. Next, they use a scheduling algorithm called Maximum Weight First (MWF), in which first links are sorted in decreasing order according to a weight function which is equal to the remaining traffic load on the link, plus the number of interfering links. Subsequently links are considered in order and scheduled in the current slot as long as they do not conflict with already chosen links. The authors of [29] proposed a two phase algorithm for *aggregation convergecast*, in the first phase of which they propose a new tree called *Degree-Constrained Aggregation Tree (DCAT)*. The key idea is to choose a parent with minimum degree in the graph, rather than in the tree. Note that their work was for the protocol model of interference, and the problem studied was aggregation convergecast. We implement both the LLHC tree and the DCAT tree to compare them with our results. We also implemented the MWF scheduling function and compared its performance with our ROS algorithm.

3.3 Network Model

We consider a set of n sensors, denoted as $S = \{s_1, s_2, ..., s_n\}$ and a set of m targets, denoted as $T = \{t_1, t_2, ..., t_m\}$, deployed arbitrarily on a given area. Each target must be covered by q sensors for redundancy and reliability. We assume that a target can be covered by a given sensor if the Euclidean distance between them does not exceed the sensing range $R_{\max}$. Each sensor generates a data packet of size σ bits each time it monitors an associated target. All the monitored data must be forwarded, possibly after passing through several hops, to a sink node, denoted by s_0.

We assume that each sensor is equipped with a single radio that can be tuned dynamically without a significant delay to transmit with some power level in the range $[0, P_{\max}]$. Traffic flows inside the network can be represented by a directed connectivity graph $G = \{S \cup \{s_0\}, E\}$ with $E = \{(s_i, s_j) : d_{ij} \leq \text{ transmission range}, i = 1, \ldots, n, j = 0, \ldots, n\}$. There exists an arc from s_i $(i \neq 0)$ to s_j if the separating distance d_{ij} is less than the transmission

range attained with $P_{\max}$. According to the given deployment of sensors and targets on the area, we assume that all sensors have a path toward the sink, and that each target can be covered by at least q sensors. Furthermore, sensors are assumed to be able to transmit according to R modulation and coding schemes (MCS). Each MCS_r ($1 \leq r \leq R$) generates a certain date rate θ_r ($\theta_1 < \theta_2 < \ldots < \theta_R$). A transmission with rate θ_r can be decoded successfully if the signal-to-interference plus noise-ratio (SINR) measured at the receiver is above a corresponding threshold β_r. Clearly, the higher the data rate is, the higher the requested SINR is.

According to the physical model given in [33], in the presence of interference caused by concurrent links in E, a transmission link (s_i, s_j) with data rate θ_r would be successful if SINR_{ij} measured at the intended receiver is greater than or equal to β_r. It is given by:

$$\text{SINR}_{ij} = \frac{p_i d_{ij}^{-\alpha}}{N_0 + \sum\limits_{i',i' \neq i} p_{i'} d_{i'j}^{-\alpha}} \tag{20}$$

where p_i is the transmission power of node s_i, d_{ij} is the distance between s_j and s_j, α is the propagation loss exponent, and N_0 is the thermal noise power. We use $P_{\max} = 15\text{mW}$, $N_0 = 10^{-6}$, data rates $\{250, 500, 1000, 2000\}$ kbps and $\beta = \{1.3, 2.0, 4.0, 10.0\}$.

In addition, we assume a TDMA access scheme, where a central entity divides the time cyclically into slots of fixed duration, which are then grouped into frames. The duration of each slot is T_s, which corresponds to the time required to send a data packet of σ bits using the lowest data rate θ_1. A TDMA schedule defines the group of admitted transmission links within each time slot so that the SINR at each receiver is above the corresponding required threshold. We neglect the slots that might be needed for control and synchronization purposes as their number is generally constant. Also, we assume that each sensor generates one data packet for each monitored target during the TDMA frame.

3.4 TDMA Frame minimization restricted to trees

In this section, we give an optimization model to find an optimal set of transmission configurations that achieve convergecast while ensuring that the routing subgraph is a tree. Our model is a modification of the model given in [7] . Our objective is to define a set of minimum-length TDMA configurations by determining jointly which sensors cover each target, the scheduled concurrent links at each slot, along with their used data-rates and power levels, and the established routes toward the sink. We refer to this problem in the rest of the paper as TFM Problem.

A configuration c is characterized by:

$$(x^c_{ijr}, p^c_i) : (s_i, s_j) \in E, r \in R, \mathrm{SINR}_{ij} \geq \beta_r, \tag{21}$$

where x^c_{ijr} is a binary parameter indicating if the link (s_i, s_j) is scheduled in c using data rate r, and p^c_i is the transmission power used by s_i. The set of all feasible configurations is denoted as $\mathcal{C}$.

We denote by the integer decision variable λ_c the number of slots in which the configuration c is scheduled. Let y_{ij} be a binary variable indicating if target t_i is covered by sensor s_j $(i \neq 0)$. In addition, let f_{ij} be an integer flow variable counting the number of data packets transmitted on link (s_i, s_j) during the whole TDMA frame. The TFM problem can be formulated by the following ILP:

$$\text{[TFM-tree model]} \qquad \min \sum_{c \in \mathcal{C}} \lambda_c \tag{22}$$

subject to:

$$\sum_{s_j \in S} y_{ij} = q \qquad t_i \in T \tag{23}$$

$$\sum_{s_j \in S} f_{ji} + \sum_{t_k \in T} y_{ki} = \sum_{s_j \in S \cup \{s_0\}} f_{ij} \qquad s_i \in S \tag{24}$$

$$\sum_{s_i \in S} f_{i0} = mq \tag{25}$$

$$b_{ij} \times (mq + 1) - f_{ij} \geq 0 \qquad (s_i, s_j) \in E \tag{26}$$

$$\sum_{s_j \in S} b_{ij} \leq 1 \qquad s_i \in S \tag{27}$$

$$\sum_{c \in \mathcal{C}} \sum_{r \in R} T_s \theta_r x_{ijr}^c \lambda_c - b_{ij}\sigma \geq 0 \qquad (s_i, s_j) \in E \tag{28}$$

$$\lambda_c \geq 0 \text{ and integer,} \qquad c \in \mathcal{C} \tag{29}$$

$$f_{ij} \geq 0 \text{ and integer,} \qquad (s_i, s_j) \in E \tag{30}$$

$$y_{ik} \in \{0, 1\} \qquad s_i \in S, t_k \in T \tag{31}$$

$$b_{ij} \in \{0, 1\} \qquad (s_i, s_j) \in E. \tag{32}$$

Constraint (23) ensures that every target is covered by exactly q sensors. Constraint (24) and (25) guarantees that each data packets has a path from its target to the sink. Using constraint (26) we introduce a boolean b_{ij} to indicate true if there exists a flow from i to j. Constraint (27) makes sure we have a tree rooted at sink. This constraint also makes sure that each source sensor will send data to one destination sensor node and hence avoid cycle. The channel capacity constraint (28) along with constraints (24) and (25) ensures that the number of times each link (s_i, s_j) is included in all the scheduled configurations ($\lambda_c > 0$) is sufficient to forward the allocated traffic.

Now, this model is solvable if we can determine by some means the set $\mathcal{C}$. But enumerating all feasible configurations would be computationally very expensive. We present in the next section a technique to alleviate this issue.

3.4.1 Column Generation Applied to TFM Tree

In order to solve efficiently the TFM-tree model, we can use the column generation technique that only requires an *implicit* enumeration of the set of configurations in order to solve the linear programming relaxation of the TFM-tree model. It consists in solving the model (22)-(32) for a restricted set of variables or columns, leading to a so-called restricted master problem (RMP), which selects the best configurations among the set of already generated

configurations. The RMP is next iteratively enriched with "augmenting configurations", generated by a so-called pricing problem, until an optimality condition is met. A configuration is an augmented one if, when added to the current RMP, it allows improving (i.e., reduce) the current optimal value of the RMP. Each new configuration is generated by the pricing problem, and using the theory of linear programming, is an augmented one if its so-called reduced cost is negative (see, e.g., [20] if not familiar with the concept of reduced cost in the linear programming theory). The expression of the reduced cost of configuration c is written as follows:

$$\text{RCOST}_c = 1 - \sum_{(s_i,s_j)\in E} \sum_{r\in R} \theta_r T_s x^c_{ijr} u_{ij} \tag{33}$$

The solution process consists in solving alternatively the RMP and the pricing problem until the pricing problem cannot generate anymore a new configuration with a negative reduced cost, meaning that we have reached the optimal solution of the linear programming relaxation of (22)-(32).

The pricing problem can be stated as follows:

$$\text{[TFM-Pricing]} \qquad \min_{c\in\mathcal{C}} \text{RCOST}_c \tag{34}$$

subject to:

$$\sum_{s_j\in S}\sum_{r\in R} x_{ijr} + x_{jir} \leq 1 \qquad s_i \in S \tag{35}$$

$$\sum_{r\in R} x_{0ir} = 0 \qquad s_i \in S \tag{36}$$

$$p_i d_{ij}^{-\alpha} - \beta_r \sum_{(s_u,s_v)\in E, u\neq i} p_u d_{uj}^{-\alpha} - L_1 x_{ijr} \geq \beta_r N_0 - L_1 \qquad s_i \in S, s_j \in S\cup\{s_0\}, r \in R \tag{37}$$

$$p_i \leq P_{\max} \sum_{s_j\in S}\sum_{r\in R} x_{ijr} \qquad s_i \in S \tag{38}$$

$$x_{ijr} \leq L_2\, p_i \qquad s_i \in S, s_j \in S, r \in R \tag{39}$$

$$x_{ijr} \in \{0,1\} \qquad s_i \in S, s_j \in S, r \in R \tag{40}$$

$$p_i \geq 0 \qquad s_i \in S, \tag{41}$$

where x_{ijr} indicates a transmission link between nodes s_i and s_j using rate θ_r, p_i is the transmission power of node s_i, L_1 and L_2 are large positive constants. Constraint (35) states that a node cannot transmit and receive at the same time and restricts it to a single rate, whereas constraint (36) prevents the sink from transmitting. Constraint (37) is an

enforcement of (20) by ensuring that when a link (s_i, s_j) is active, the SINR should be above the threshold β_r. Otherwise, this constraint becomes redundant. Constraints (38) and (39) bind between every transmission link and its power. If a transmission link is not active, power would be set to zero, and vice-versa.

Once the optimality condition is satisfied (i.e., the pricing problem is no more able to produce configurations with a negative reduced cost), solving the last RMP with integrality requirements leads to an ILP solution, with an easy way to evaluate accuracy.

3.5 Scheduling Algorithms using the TC approach

The result of a column generation model such as the one given in the previous section or [51, 76, 44, 7] is a multi-set of transmission configurations. Each such configuration can be scheduled in a single time slot without any further need to check for interference. Additionally, we are assured that it is feasible to complete the convergecast operation only by using the transmission configurations. However, the solution of the column generation model does not suggest any order in which to use them, nor how many times each transmission configuration should be used (but does provide a lower bound on the number of times each is required). In this section, we start with an approach to find an optimal schedule (Section 3.5.1) and subsequently give a number of scheduling heuristics using only these transmission configurations (Section 3.5.2).

3.5.1 Optimal schedule given a set of TCs

We first provide an approach to find an optimal schedule using only the given set of configurations and a single datarate.

$$\text{OptimalSchedule} \qquad \min \sum_{\tau \in TS} z^{\tau} \tag{42}$$

subject to:

$$D_s^\tau - b_\ell^\tau = D_s^{\tau+1} \qquad \tau \in TS, \ell \in \{(s,s') \in E\} \tag{43}$$

$$D_s^\tau + b_\ell^\tau = D_s^{\tau+1} \qquad \tau \in TS, \ell \in \{(s',s) \in E\} \tag{44}$$

$$\sum_{s \in S} D_s^0 = D_0^{TS} \tag{45}$$

$$b_\ell^\tau \leq 1 - \sum_{c \in C} \lambda_c^\tau (1 - x_{\ell c}) \qquad \tau \in TS, \ell \in \{(s,s') \in E\} \tag{46}$$

$$\sum_{c \in C} \lambda_c^\tau = z_\tau \qquad \tau \in TS \tag{47}$$

$$z^\tau \in \{0,1\} \qquad \tau \in TS. \tag{48}$$

$$\lambda_c^\tau \in \{0,1\} \qquad \tau \in TS, c \in C \tag{49}$$

$$b_\ell^\tau \in \{0,1\} \qquad \tau \in TS, \ell \in \{(s,s') \in E\} \tag{50}$$

$$D_s^\tau \in Z^* \qquad \tau \in TS, s \in S \tag{51}$$

The input to this optimal schedule ILP model is multi-set of configurations. Here we represent D_s^τ as the number of packets available in sensor s in slot τ. The boolean variable z^τ says if slot τ is used or not. The boolean λ_c^τ says whether a configuration c is used in slot τ or not. Similarly b_ℓ^τ says weather a link ℓ is used in slot τ or not. Constraints (43) and (44) ensure the flow of data at each sensor. Constraint (45) says that in time slot TS, the sink node gets all the packets initially present at the sensor nodes monitoring the targets. Constraint (46) ensures that only the links in the configuration scheduled in time slot τ can send data in that time slot. Constraint (47) ensures that exactly one configuration is chosen in every slot.

3.5.2 TC-based scheduling heuristics

To provide a set of scheduling heuristics as shown in Algorithm 5, we next define the *weight* of a configuration, and always pick the configuration of maximum weight. Given a configuration c, denote by α_ℓ the boolean value that denotes whether or not link ℓ has data ready to transmit, x_ℓ the length of the queue for link ℓ, y_ℓ the number of packets still to be transmitted over link ℓ, and d_ℓ the hop-distance of ℓ to the sink node. We propose different weight functions for a configuration c:

$$w_1(c) : \sum_{\ell \in c} \alpha_\ell \ ; (\text{number of links with data})$$
$$w_2(c) : \sum_{\ell \in c} \alpha_\ell x_\ell \ ; (\text{sum of queues})$$
$$w_3(c) : \sum_{\ell \in c} \alpha_\ell y_\ell \ ; (\text{sum of remaining traffic})$$
$$w_4(c) : \min_{\ell \in c} \alpha_\ell d_\ell \ ; (\text{min distance})$$
$$w_5(c) : \max_{\ell \in c} \alpha_\ell d_\ell \ ; (\text{max distance})$$
$$w_6(c) : \max_{\ell \in c} \alpha_\ell y_\ell \ ; (\text{max of remaining traffic})$$
$$w_7(c) : \max_{\ell \in c} \alpha_\ell x_\ell ; (\text{max of queues}).$$

In Section 3.7, we discuss the performance evaluation of these heuristics, used either with the transmission configurations produced by the TFM-graph model in [7] or with the results

of the TFM-tree model proposed in Section 3.4.

Algorithm 5 Scheduling heuristic using weight function W on given set of transmission configurations C

Require: Weight function W, $load[\ell]$ = number of packets to sent on link ℓ, $data[s]$ =number of packets waiting to be forwarded from node s, C = set of configurations.

Ensure: Schedule. $NumSlots \leftarrow 0$

```
repeat
  for each configuration c in C do
    Compute c.weight according to weight function W
  end for
  max ← 0, max_c ← −1
  for each configuration c in C do
    if c.weight > max then
      max ← c.weight
      max_c ← c; %%max_c is the configuration with highest weight
    end if
  end for
  if max > 0 then
    for each link ℓ in max_c do
      if data[ℓ.src] > 0 then
        packets ← min(ℓ.rate/lowestRate, data[ℓ.src])
        Schedule[NumSlots] = Schedule[NumSlots] ∪ {ℓ}
        load[ℓ] = load[ℓ]−packets
        data[ℓ.src] = data[ℓ.src]−packets
        data[ℓ.dst] = data[ℓ.dst]+packets
      end if
    end for
    NumSlots ← NumSlots + 1
  end if
until sink gets all packets
```

3.6 Two-phase Scheduling Algorithms

Two-phase scheduling algorithms are two phase algorithms. In the first phase, a routing tree is defined to achieve convergecast. In the second phase, an algorithm to schedule the packets along the links of the tree defined in the first phase is described. In this paper, we propose a new algorithm for each phase. For the first phase, we propose a new TFM Tree for convergecast, namely, the tree implied by the result of the column generation algorithm described in Section 3.4. Clearly, this tree can be used with any algorithm for the second phase; we evaluate its performance with two different algorithms for the second phase in Section 3.7.

Algorithm 6 Round Optimal Schedule (ROS)

Require: Data gathering tree or graph $\hat{G}$, $data[s]$ =number of packets waiting to be forwarded from node s

Ensure: Schedule, Power to be used by each link in each slot **Uses:** $MILP(F, PowerVector, RateVector, NumSlots)$ *which outputs a set of links* $L \subseteq F$ *that can transmit simultaneously under SINR constraints, and assigns feasible powers and rates to all links for the current time slot.*

$NumSlots \leftarrow 0$ $load = Compute\text{-}Link\text{-}Load\text{-}Vector(\hat{G})$

repeat

$F \leftarrow \emptyset$

for each link ℓ with $load[\ell]0]$ **do**

if $data[\ell.src] > 0$ **then**

$F \leftarrow F \cup \ell$

end if

end for

$L \leftarrow MILP(F, PowerVector, RateVector, NumSlots)$ $Schedule[NumSlots] \leftarrow L$

for each link $\ell \in$ L **do**

packets $\leftarrow min(Rate[\ell]/lowestRate, data[\ell.src])$

$load[\ell] \leftarrow load[\ell]-$packets

$data[\ell.src] = data[\ell.src]-$packets

$data[\ell.dst] = data[\ell.dst]+$packets

end for

$NumSlots \leftarrow NumSlots + 1$

until *sink* gets all packets

For the second phase, we give a new algorithm, and also modify the algorithm in [32].

First, we propose a new algorithm (named Round-Optimal Schedule (ROS), see Algorithm 6) that schedules packets for convergecast along the links of a given tree. The algorithm proceeds in rounds. In each round, we examine the set of links in the routing tree/subgraph that have data available. We compute a maximum-sized subset of links that can be scheduled while not violating SINR conditions, and schedule them. We move the data to the respective destination nodes of the links, and proceed to the next round. We use an MILP to check the SINR condition and power level assignments. This MILP solves the optimization problem with the following objective function subject to constraints (35) - (41):

$$\text{[ROS]} \qquad \max \sum_{(s_i,s_j)\in \text{ links with data}} \sum_{r\in R} x_{ijr}. \tag{52}$$

Next we describe our modifications to the algorithm given in [32], whose scheduling algorithm is called Maximum Weight First (MWF). In their algorithm, first links are sorted in decreasing order according to a weight function which is equal to the remaining traffic load on the link, plus the number of interfering links. Subsequently links are considered in order and scheduled in the current slot as long as they do not conflict with already chosen links. MWF also uses a conflict graph $I = (V, E)$ for a scheduling tree T, such that each edge in T is represented as a vertex in I. If it is impossible to schedule two links simultaneously in T, then there is an edge in I between those two corresponding vertices. MWF uses the Perron-Frobenius theorem as given in [67] to assign powers to links. This algorithm has the same major problem as the previous TC-based models: links are scheduled even when they have no data. In other words, the algorithm as given does not give a true schedule. In this paper, we fix this problem by considering in line 10 only the links which have data ready to transmit. Secondly, the MWF algorithm in [32] assumes that every sensor initially has one data packet to transmit. We extend the algorithm by removing this assumption. Finally we extend the algorithm to deal with q-coverage, as well as multiple data rates. The requirement of q-Coverage is handled by the first phase, and is therefore does not need to be considered in this algorithm. For multi-rate, while greedily choosing links we assign maximum feasible rate that achieves feasible power assignment vector.

Algorithm 7 EMWF

Require: Data Gathering tree or graph $\hat{G}$; Transmitting power levels range $\{0,P_{max}\}$; $data[s]$ =number of packets waiting to be forwarded from node s, $IsTFM$=*true* only if it uses TFM based tree or subgraph, $load[\ell]$ = number of packets to send on link ℓ using TFM model

Ensure: Schedule, Power to be used by each link in each slot;

Generate conflict graph I for the links in T
$load = Compute\text{-}Link\text{-}Load\text{-}Vector(\hat{G})$
$NumSlots \leftarrow 1$
$TotalLoad \leftarrow \sum_{\ell \in T} load(\ell)$
repeat
  for each $\ell \in \hat{G}$ **do**
    $degree(\ell) \leftarrow$ degree of the node associated with ℓ in the conflict graph I
    $weight(\ell) \leftarrow load[\ell] + degree(\ell)$
    if $data[\ell.src] = 0$ **then**
      $weight(\ell) \leftarrow 0$
    end if
  end for
  Sort links in T in the descending order of their weight into T'
  for each ℓ in T' **do**
    for each possible rate r in decreasing order **do**
      if there exists feasible power assignment for $Schedule_{NumSlots} \cup \{\ell\}$ **then**
        $Schedule[NumSlots] \leftarrow Schedule[NumSlots] \cup \{\ell\}$
        Assign power vector for links in $Schedule[NumSlots]$ using PF-theorem
        $packets \leftarrow min(data[\ell.src], r/LowestRate)$
        $load(\ell) = load(\ell) - packets$
        $TotalLoad \leftarrow TotalLoad - packets$
        $data[\ell.src] \leftarrow data[\ell.src] - packets$
        $data[\ell.dst] \leftarrow data[\ell.dst] + packets$
        break
      end if
    end for
  end for
  Update conflict graph I, by removing vertices whose traffic load is satisfied in T
  $NumSlots = NumSlots + 1$
until $(TotalLoad > 0)$

The pseudocode for our extended algorithm, called Extended Maximum Weight First (EMWF) is presented in Algorithm 8.

Algorithm 8 Compute-Link-Load-Vector[1]

Require: Data Gathering Tree or subgraph $\hat{G}$; $data[s]$ =number of packets waiting to be forwarded from node s, $IsTFM$=*true* only if using TFM based tree or subgraph, in which case $TFMload[\ell] =$ is already computed

Ensure: $load(\ell)$ computed for each link $\ell \in \hat{G}$;

```
if IsTFM = true then
    return TFMload
else
    for each s ∈ S with data[s] > 0 do
        path = GetPathToSink(s)
        for each ℓ ∈ path do
            load[ℓ] ← load[ℓ] + data[s]
        end for
    end for
end if
```

[1] While there are more efficient ways to compute this vector for a routing tree, we present a unified method here that works for both routing trees and routing subgraphs

3.7 Experimental Results

We now present our evaluation of the two scheduling approaches. Each result presented here is an average over 10 simulations with randomly generated topologies. To have a more meaningful comparison of results, when looking at larger topologies, we embed the topologies used for smaller networks, and add extra nodes at random positions. Since the column generation-based methods take significantly longer on larger inputs, we use 2 hours as a stopping condition for all inputs. Note that on topologies with 80 nodes or more, this results in a sub-optimal column generation solution.

3.7.1 Scheduling on a Tree

Tables 7 and 8 show the lengths of schedules produced by the different algorithms. Among the different heuristics used for the TC method, we observe immediately that the *sum-based* heuristics work significantly better than the *max/min-based* heuristics. The best performer

is the w_3-based heuristic, that schedules in the next slot the configuration with the maximum remaining traffic.

Among the two-phase methods, the first question we seek to answer is, which tree is best? With both scheduling algorithms (ROS and EMWF), the best tree is consistently the TFM tree. The second question is: which scheduling algorithm is best? Both the EMWF and ROS scheduling algorithms exhibit similar results in terms of number of slots, for each of the three trees. However, the EMWF algorithm is more computationally efficient, and takes only a few seconds on topologies of 100 nodes, while the ROS scheduling takes two hours in some cases. We conclude that among the two-phase methods, while the additional computational cost of computing the TFM tree is worthwhile, the EMWF algorithm is more computationally efficient than ROS. That is, the best option would be to use the TFM tree and the EMWF scheduling algorithm.

Finally we compare the TC-based methods with the two-phase methods. Our observation is that the two-phase approaches significantly outperform the TC-based approaches, for 70, 80, and 90 nodes. For 100 nodes, the EMWF-TFM-tree combination still produces a better result, but the difference is less than for 90 nodes; this could be because the quality of the TFM tree used may be sub-optimal, owing to the stopping condition of 2 hours being imposed.

Table 7: Length of schedule produced by different algorithms

# Sensors	"Estimated" Lower Bound	Transmission Configuration (TC) Methods						
		sum-based			min-based	max-based		
schedule		w_1	w_2	w_3	w_4	w_5	w_6	w_7
40	105.0	147.3	142.8	124.1	229.1	213.6	239.0	197.0
50	125.1	171.3	167.2	146.9	306.6	283.8	316.8	258.3
60	115.0	167.9	163.8	138.6	344.2	300.9	358.4	270.4
70	132.9	190.7	189.6	161.3	399.6	368.3	418.5	314.8
80†	142.0	205.3	201.0	173.6	414.6	369.6	428.8	318.8
90†	172.8	253.6	226.2	202.4	390.0	345.6	404.2	325.4
100†	196.5	254.8	252.3	226.5	421.3	374.0	425.8	379.3
† LP solution of the RMP is stopped when the solution of PP takes more than 2h								

Table 8: Length of schedule produced by different algorithms

# Sensors	"Estimated"	two-phase Methods					
Tree schedule	Lower Bound	TFM EMWF	LLHC EMWF	DCAT EMWF	TFM ROS	LLHC ROS	DCAT ROS
40	105.0	123.7	140.9	140.8	125.0	141.1	141.8
50	125.1	144.3	163.2	165.9	144.9	164.2	165.2
60	115.0	137.1	169.2	174.2	137.8	169.8	172.9
70	132.9	151.2	203.7	198.1	153.9	194.4	196.6
80†	142.0	149.4	204.8	215.9	150.4	204.1	213.1
90†	172.8	172.6	218.6	226.6	176.4	218.2	227.4
100†	196.5	220.7	231.7	235.7	220.3	231.3	231.7
† LP solution of the RMP is stopped when the solution of PP takes more than 2h							

3.7.2 Routing on Trees or Subgraphs

We examine here the impact of restricting the scheduling on a tree. We compare the lengths of the schedules when we impose such a restriction versus when we do not impose such a restriction. We run the column generation algorithm in [7] to derive a set of transmission configurations as well as the resulting routing sub-graph, called the TFM graph. When scheduling on a tree, since the TFM tree was observed to be the best out of all considered trees, we consider only the TFM tree here. Then we consider the ROS, the EMWF scheduling algorithms and the w_3-based heuristic and compare the results, shown in Table 9.

When routing on graphs, as with trees, the ROS and EMWF algorithms outperform the w_3-based heuristic. In addition, both ROS and EMWF produces schedules with fewer slots while using the TFM graph compared to the TFM tree, while for the w_3-based scheduling, there is no significant difference. In other words, the difference between ROS and EMWF and the w_3-based heuristic is even more when using routing subgraphs than when using routing trees.

Table 9: Scheduling - Subgraph vs Tree

	TFM Graph				TFM Tree				
Sen	LB†	w_3	ROS	EMWF	LB†	w_3	ROS	EMWF	
40	102.3	123.2	122.0	123.7	105.0	124.1	125.0	123.7	
50	117.7	149.7	138.4	142.3	125.1	146.9	144.9	144.3	
60	110.5	141.8	134.1	130.6	115.0	138.6	137.8	137.1	
70	118.1	152.5	138.8	138.0	132.9	161.3	153.9	151.2	
80†	123.7	173.0	142.0	144.2	142.0	173.6	150.4	149.4	
90†	152.1	198.2	160.6	168.2	172.8	202.4	176.4	172.6	
100†	200.0	223.0	211.0	212.7	196.5	226.5	220.3	220.7	

LB† = Estimated Lower Bound

† LP solution of the RMP is stopped when the solution of PP takes more than 2h

3.7.3 Scheduling for q-Coverage

In this section, we consider the situation when only a subset of the sensors are close enough to the targets to monitor them. However, every target is required to be monitored by q sensors, $q \in \{1, 2, 3\}$. The remaining sensors only play a role in forwarding data to the sink. We compare our three best algorithms using the TFM graph for 100 targets, but with different numbers of sensors. Table 10 compares the number of slots used for different coverage levels. We see that for all coverage levels, ROS and EMWF outperform the w_3-based heuristic. However, as q increases, the percentage difference between the schedule length computed by the w_3-based heuristic and EMWF decreases. This agrees with the intuition that the TC-based approach is really based on routing a stream of data rather than a single data packet as in the previous experiments. Since there are more packets to be routed in 2-coverage or 3-coverage, w_3 's performance relative to the EMWF and ROS algorithms improves. Another interesting observation is that for $q = 2$ and 3, the EMWF schedule starts to perform better compared to ROS.

Table 10: Scheduling q-coverage (100 targets, TFM graph)

	$Q = 1$			$Q = 2$			Q =3		
	w_3	ROS	EMWF	w_3	ROS	EMWF	w_3	ROS	EMWF
40	235.5	242.1	239.6	472.5	510.1	499.8	745.7	798.6	788.2
50	205.1	195.2	198.0	395.3	395.4	395.1	622.1	633.3	636.4
60	175.1	168.3	165.0	345.2	342.4	335.8	515.4	511.8	506.9
70	171.2	155.8	155.1	327.3	318.6	313.7	492.2	491.6	484.7
80†	158.6	143.0	138.7	310.3	305.9	295.6	464.5	460.5	446.2
90†	162.6	132.6	131.4	294.8	259.3	256.6	449.2	409.7	406.9
100†	168.1	130.3	130.2	321.6	281.6	279.3	490.8	423.1	418.2
110†	172.0	138.9	140.4	328.5	292.0	286.8	496.8	446.1	442.1
† LP solution of the RMP is stopped when the solution of PP takes more than 2h									

3.7.4 Effect of Having more Packets per Sensor

In all the experiments in Section 3.7.1, we assumed that one packet is sufficient to forward data to the next sensor. In this section, we consider the situation in which each sensor reading may be too large to fit in a single packet, thus, each sensor may need to send multiple packets per reading. This resembles more closely the situation when sensors are sending a stream of data. We used the TFM routing subgraph, and compared the results of the two best algorithms: w_3-based, and EMWF. We did not consider ROS as it would be computationally too expensive. The number of slots in the schedules computed by the two algorithms is shown in as shown in Table 11 for the number of packets ranging from 1 to 32. We see that as the number of packets increases, the w_3-based heuristic does better than EMWF. In fact, the percentage difference between the w_3-based heuristic and EMWF increases with the number of packets. This is consistent with the results of Section 3.7.3; as the number of packets sent by a sensor increases, the relative performance of the w_3-based heuristic improves, and in fact is the best of the three algorithms when there are more than 4 packets per reading. We used 50 and 60 sensors for the results shown here, but we found a similar pattern for other numbers of sensors as well.

Table 11: Scheduling - Effect of having more packets per sensor

		Schedule	Schedule	
packets per reading	LB	w_3	EMWF	(EMWF - w_3)/w_3 as percentage
50 Sensors				
1	114	138	142	2.9
2	229	278	283	1.8
4	456	493	538	9.1
8	912	946	1,098	16.1
16	1,823	1,916	2,175	13.5
32	3,624	3,757	4,250	13.1
60 Sensors				
1	121	166	155	-6.6
2	242	315	305	-3.2
4	484	534	571	6.9
8	976	1,072	1,156	7.8
16	1,935	2,065	2,286	10.7
32	3,869	4,078	4,525	11.0

3.7.5 Scheduling for Multi-rate

Table 12: Scheduling - Multi rate

		Schedule	Schedule	Schedule
Sensors	LB†	w_3	ROS	EMWF
40	834.0	883.0	980.0	1,044.0
40	720.0	761.0	730.0	906.0
40	664.0	742.0	747.0	878.0
40	880.0	906.0	965.0	957.0
40	752.0	813.0	1,057.0	1,164.0
40	776.0	810.0	874.0	895.0
40	1,000.0	1,062.0	1,045.0	1,010.0
40	808.0	861.0	928.0	922.0
40	898.0	971.0	896.0	896.0
40	842.0	965.0	988.0	951.0
AVG	817.4	877.4	921.0	962.3
50	916.0	972.0	1,063.0	1,156.0
50	760.0	826.0	831.0	853.0
50	648.0	687.0	748.0	774.0
50	1,128.0	1,206.0	1,181.0	1,325.0
50	1,032.0	1,239.0	1,262.0	1,256.0
50	803.0	859.0	900.0	962.0
50	1,144.0	1,178.0	1,201.0	1,243.0
50	966.0	1,041.0	1,193.0	1,192.0
50	665.0	704.0	723.0	752.0
50	1,064.0	1,260.0	1,263.0	1,290.0
AVG	912.6	997.2	1,036.5	1,080.3

In the experiments in the previous sections, we used a single data rate for all sensors, and this enabled sending a single packet over a link in a time step. In this section, we evaluate the performance of three of our best algorithms using multiple data rates. We use a reading size such that an entire reading can be forwarded in one slot using the *highest possible* data rate, while using the highest of the 4 possible data rates, we could send 8 packets (one reading) in a single time slot. Rate and power assignment are done to satisfy SINR constraints either when using the optimization model (for the w_3-based heuristic) or in the ROS/EMWF algorithms. Table 12 shows that the w_3-based heuristic provides the best result, followed by ROS. EMWF has the worst results. We remark that both the w_3-heuristic and ROS find optimal solutions for some part of the problem (optimal transmission configurations for w_3 and round-optimal schedule for ROS, while EMWF is very efficient but does not make any attempt at optimality. We conclude that it is more advantageous to restrict to the precomputed transmission configurations when using multiple rates.

3.8 Conclusion

We investigated two basic approaches to scheduling for a convergecast operation in a WSN. In the first approach, called the TC-approach, a multiset of transmission configurations that are interference-free and that cover the convergecast traffic is computed and the scheduling algorithm restricts itself to using these configurations. In the second approach, called the two-phase approach, as exemplified in [32], first a routing tree or subgraph is computed, and next, sets of non-interfering links are scheduled in rounds, based on which links have available traffic in each round. In this paper, for the TC-based approach, we provide a new column generation model that restricts the schedule to a tree. Given any set of TCs that cover the required traffic, we give an ILP model to schedule the TCs as well as several new and very efficient scheduling heuristics. For the two-phase approach, we give a new tree called TFM tree, which takes into account the physical interference model as well as power control, as well as two scheduling algorithms (ROS and EMWF) for the second phase. Our results show that for single-rate sensors, when each sensor has 1 or very few packets to send, the two-phase approach using our TFM tree significantly outperforms the TC-approach in terms of the length of the produced schedule. However, if each node has several packets to send in the same frame, a situation which more closely resembles a data stream, or if the sensors are multi-rate, then the TC-approach produces better results.

Chapter 4

Optimum ConvergeCast Scheduling in Wireless Sensor Networks (WSNs)

M. Bakshi, B. Jaumard, and L. Narayanan. True convergecast scheduling in wireless sensor networks. submitted for publication, 2017. An extended abstract of this paper has been published in International Conference on Computing, Networking and Communications (ICNC), 2017. [6]

4.1 Introduction

A Wireless Sensor Network (WSN) is a connected network of autonomous sensors, which can monitor or sense physical or environmental conditions, such as temperature, sound, or pressure, and that can communicate with each other using wireless transmissions. WSNs have many applications in, e.g., agriculture, environment monitoring, wildlife tracking, smart buildings and cities. Many of these are multimedia applications and need multi-rate support [79], [58], [21]. An important class of applications for WSNs is *target coverage*: given a set of targets, each target must be monitored by q sensors and all the sensor readings must be forwarded using multi-hop routing to a sink node. ConvergeCast refers to the many-to-one communication scheme, where data from a set of sources are routed toward a common sink. ConvergeCast is the communication pattern used in target coverage applications.

In this paper, we consider the problem of finding a transmission schedule to achieve ConvergeCast in a WSN using TDMA as the MAC-layer protocol. The solution involves deciding which sensors cover which targets, finding paths to send the sensor readings from the covering sensors to the sink node, as well as scheduling transmission slots for the sensors on these paths that avoid excessive interference. The goal is to find a transmission schedule of minimum length, that is, a schedule with the minimum number of transmission slots. In

addition, we will not assume that transmission is done with a tree transmission scheme, as it does not necessarily lead to an optimal schedule, as explained in the example that follows.

Figure 7 illustrates an instance of ConvergeCast. The input is described in Figure 7(a), i.e., the location of the set of targets $\{t_0, t_1\}$, sensors$\{s_1, \ldots, s_7\}$, and the sink node s_0. The desired coverage level is $q = 2$, i.e., each target needs to be covered by two sensors. All possible transmission links are depicted in Figure 7(a).

Under the assumption of a tree transmission scheme, the ConvergeCast solution is depicted in Figure 7(b). Observe that target t_0 is monitored by sensors s_3 and s_6, while target t_1 is monitored by sensors s_2 and s_5. Each of the monitoring sensors needs to send the information concerning the target(s) it is monitoring along a path to the sink. Links have to be scheduled in such a way that the combination of the paths defines a ConvergeCast tree (using sensors) while obeying interference constraints. One possible optimal schedule is as follows. In time slot 1, schedule links (s_3, s_4) and (s_2, s_1) simultaneously, then in the next six slots, the sets $\{(s_5, s_3), (s_1, s_0)\}$, $\{(s_5, s_3), (s_4, s_0)\}$, $\{(s_3, s_4)\}$, $\{(s_4, s_0)\}$, $\{(s_3, s_4)\}$, $\{(s_4, s_0)\}$ can be scheduled in turn.

However, if we do not impose to use a tree transmission scheme, as illustrated in Figure 7(c), it is possible to get a shorter optimal schedule with 6 slots. Observe that target t_0 is monitored by sensors s_3 and s_5, while target t_1 is monitored by sensors s_2 and s_5. Each of the monitoring sensors needs to send the information concerning the target(s) it is monitoring along a path to the sink. Since sensor s_5 is monitoring two targets, it needs to send two packets to the sink, possibly along different paths, as is the case in the solution given in Figure 7(c). Finally the links along all paths have to be scheduled while respecting interference constraints. One possible optimal schedule is as follows. In time slot 1, schedule the links (s_3, s_4) and (s_2, s_1) simultaneously, then in slot 2, the links (s_5, s_3) and (s_1, s_0). In the next four slots, the sets $\{(s_5, s_2), (s_4, s_0)\}$, $\{(s_3, s_4), (s_2, s_1)\}$, $\{(s_1, s_0)\}$, $\{(s_4, s_0)\}$ can be scheduled in turn, and it can be verified that all data reaches the sink.

Many researchers have approached ConvergeCast and related scheduling problems by using mathematical programming modelling and algorithms. In general, these solutions produce as output, a set of *transmission configurations (TC)*. Each TC is a set of links that can be scheduled simultaneously during the same time slot without causing significant interference. Put together, the TCs provide a set of paths from the targets to the sink that meet coverage requirements. However, they do not actually provide a *schedule* specifying the time slots in which a sensor should transmit. Indeed, as already mentioned in [42, 2], it is non-trivial to produce a *schedule* given a set of TCs. Indeed, the number of TCs that is output by the previously proposed mathematical models only give a lower bound on the length of the schedule. In some cases, e.g., in a line topology, this is a very weak lower

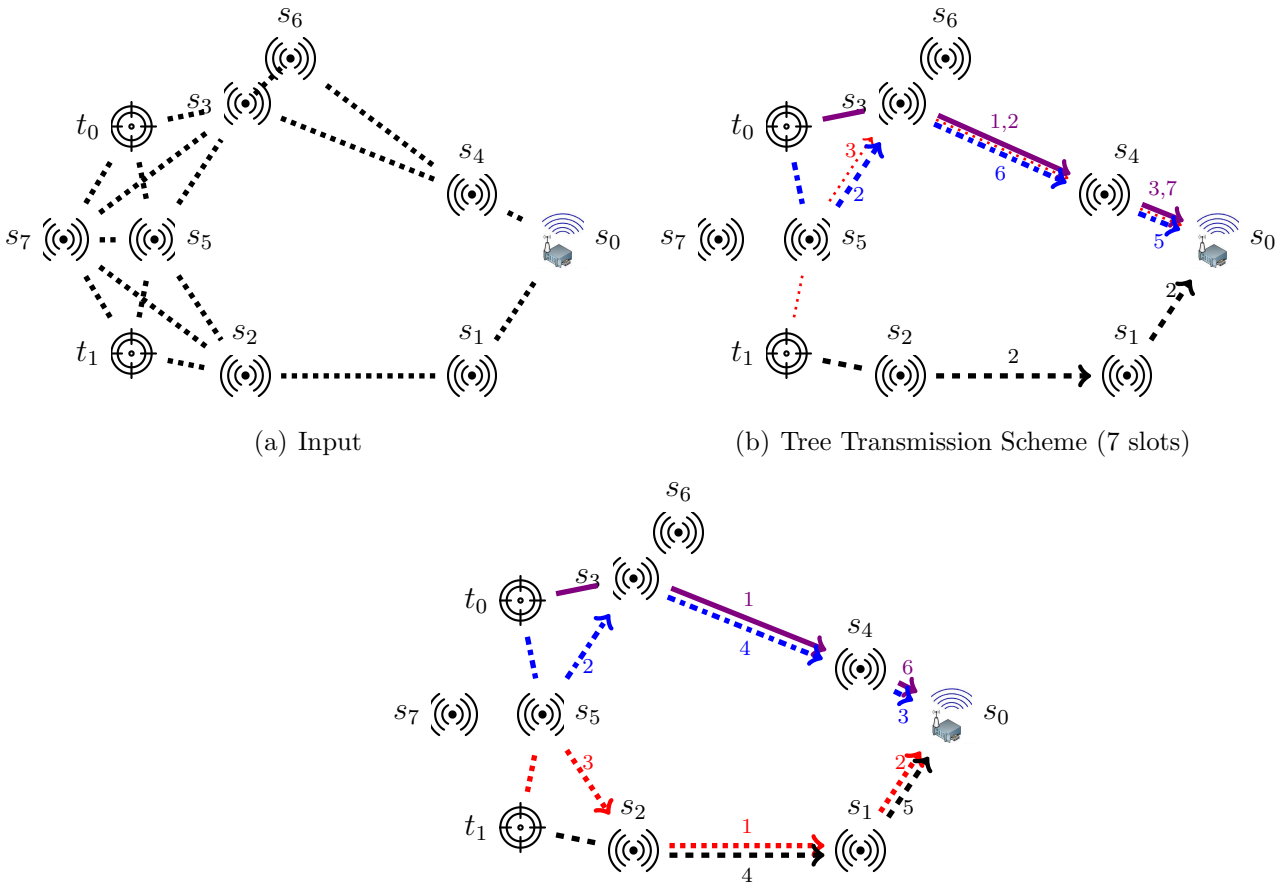
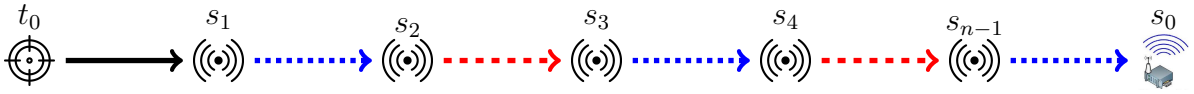
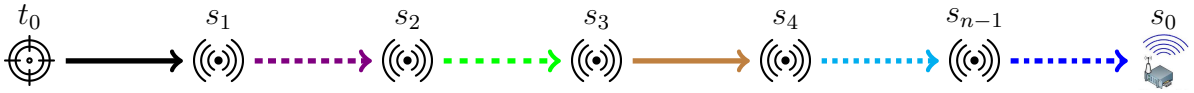


Figure 7: ConvergeCast Problem Instance (q=2)

bound, as shown in Figure 8. The optimal set of TCs is of size 2, but any schedule needs $n-1$ slots.



(a) Existing mathematical programming solutions output 2 transmission configurations, one with all "odd" links, another with all "even" links



(b) An optimal schedule requires $n-1$ slots: each link is scheduled in a different slot

Figure 8: Limitation of existing ConvergeCast solutions

4.1.1 Our Results

In this paper, we provide, for the first time, a mathematical programming formulation to find an optimal schedule for ConvergeCast in a TDMA-based WSN, using multi-rate transmissions, q-coverage and a physical interference (SINR) model.

We design a scalable solution process, called OSCC-1P (Optimal Schedule for ConvergeCast in One Phase or 1P) algorithm, which uses Dantzig-Wolfe decomposition techniques, to solve the resulting model. It outputs a proven ε-optimal solution for large ConvergeCast instances. We conducted extensive experiments to compare the performance of the OSCC-1P algorithm with the best previous algorithm of the literature. Accuracy is not only proven but significantly improved. The schedule output by OSCC-1P compares very favorably with the lower bound produced by the previously proposed TC-based approach [2]. For up to 70 sensor nodes, the OSCC-1P solution is proven to be within 4 % of the optimal solution. In terms of the upper bound, the OSCC-1P solution is upto 15 % better than the previously best solution.

In the next section, we provide the system model and a concise definition of the ConvergeCast problem. In Section 4.3, we present a brief overview of previous work. In Section 4.4, we provide a new single phase formulation for the ConvergeCast scheme. In Section 4.5, we propose a scalable OSCC-1P algorithm for solving the proposed mathematical programming model. Numerical results are presented in Section 4.6. Conclusions are drawn in the last section.

4.2 System Model and ConvergeCast Problem

In this section, we describe our system model, namely our interference model, the TDMA protocol and the ConvergeCast problem specification.

Let S be a set of n sensors and T a set of m targets. A target t can be covered by a sensor s if the Euclidean distance between t and s is less than the sensing range. Similarly, sensor s can forward data to another sensor s' only if the Euclidean distance between s and s' does not exceed transmission range ($R_{\max}$). Each sensor generates a data packet of size σ bits each time it monitors an associated target. We consider that the sensors and targets are deployed arbitrarily in a given area such that each target can be covered by at least q sensors, and each sensor has a path to the sink node s_0.

Communication between sensor nodes takes place using wireless transmissions, and therefore needs to account for the possibility of radio interference. Indeed, each link can cause interference to the other links transmitting at the same time. Among the different interference models which are used in the literature, the Signal to Interference plus Noise Ratio (SINR) [33] model is considered to the most realistic. SINR refers to the ratio of the signal received by the intended receiver of a link $\ell = (s, s')$ to the interference caused by the other parallel links plus noise, which is defined as:

$$\mathrm{SINR}_{s,s'} = \frac{p_s d_{ss'}^{-\alpha}}{N_0 + \sum\limits_{s'' \in S, s'' \neq s} p_{s''} d_{s''s'}^{-\alpha}}. \tag{53}$$

Therein, p_s is the transmission power of sensor s, $d_{ss'}$ is the distance between sensors s and s', α is the path loss exponent, N_0 is the thermal noise power, and S' is the set of sensors transmitting at the same time as s.

Sensors are assumed to transmit using the modulation and coding schemes [54] of IEEE 802.15; according to this specification, a transmission with data rate r can be decoded successfully if the SINR measured at the receiver is above a corresponding threshold β_r. That is, a transmission link ℓ with data rate r is successful only if:

$$\mathrm{SINR}_\ell \geq \beta_r, \tag{54}$$

where β_r is a threshold that depends on the data rate r.

We consider each sensor is equipped with a single radio that can be tuned dynamically without a significant delay to transmit with some power level in the range $[0, P_{\max}]$. We use out-door sensor specifications, i.e., $P_{\max} = 0.013$W, $N_0 = 10^{-6}$, set of data rates $R = \{250, 500\}$ kbps and $\beta_r = \{1.3, 2.0\}$.

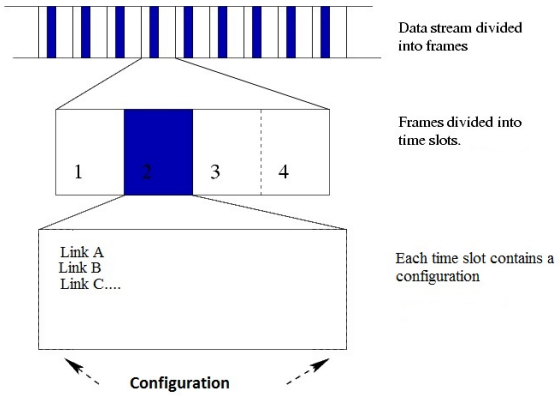


Figure 10: TDMA - Each frame achieves ConvergeCast

A **configuration** refers to a set of links that can be used simultaneously, while satisfying the SINR condition (54). We can formally define a configuration c:

$$c = \{(\ell, r_\ell, p_s) : \ell = (s, s') \in L, \text{SINR}_\ell \geq \beta_r\}, \tag{55}$$

i.e., c is a set of links $\ell = (s, s')$, using data rate r_ℓ on link ℓ, with p_s being the transmission power used by sensor s. Figure 9 gives an example of a configuration: links ℓ_{21} and ℓ_{34} can be scheduled simultaneously without violating the SINR constraints.The problem of finding a maximum set of links that can form a configuration even for the protocol interference model is NP-hard as it can be seen to be equivalent to the well-known Maximum Independent Set problem.

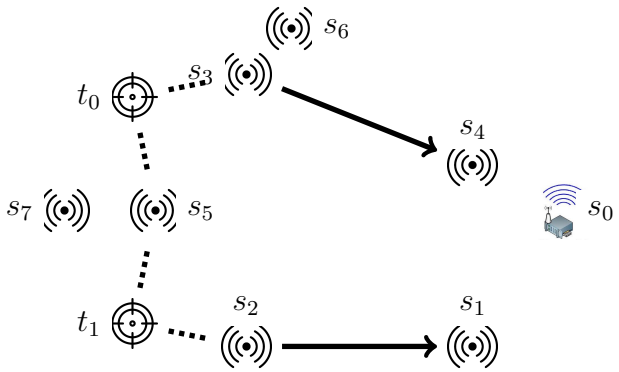


Figure 9: A configuration example (q=2): $c = \{(s_2, s_1), (s_3, s_4)\}$

As mentioned earlier, Time Division Multiple Access (TDMA) is considered as the MAC layer protocol for collision resolution in this paper. Many WSN applications, including the target coverage application that we consider here, require periodic collection of sensor measurements at a centralized sink node; TDMA is a suitable MAC layer protocol for such

purposes. We assume the slots needed for control and synchronization are negligible. In TDMA, as shown in Figure 10, time is divided into frames, each containing a certain number of fixed size time slots. Each slot holds one configuration, while the entire frame achieves the ConvergeCast operation, and can be repeatedly periodically as needed. In the context of the present study, we assume that the duration of each slot is T_s, which corresponds to the time required to send a data packet of σ bits using the highest data rate. Observe that the same configuration may be scheduled in multiple slots in the same frame. A frame can therefore be seen to be an ordered sequence of configurations.

The **ConvergeCast** problem uses the location of a set of sensors, a set of targets and the sink, as well as the desired coverage level q as input, and finds a minimum length TDMA frame that achieves the ConvergeCast operation, that is,

1. Each target must be monitored by exactly q sensors.
2. Each sensor forwards all the data it receives along a path to the sink.
3. The sink node gets all the data from the targets.

The general problem of determining a ConvergeCast solution using the SINR interference model and power control is NP-hard [10].

4.3 Existing Work

The scheduling problem of generating a minimum number of transmission configuration occurrences in a wireless TDMA network has been extensively studied in the literature. We can distinguish two classes of algorithms that we next discuss, heuristics and mathematical programming models/algorithms.

4.3.1 Heuristics

A first class of algorithms deals with heuristics, without using any mathematical models. They can be further classified into tree-based or unstructured sub-graph based on the one hand, and then into protocol and SINR interference based models.

Most of the heuristics are tree-based heuristics, see, e.g., [31], [40], [12], [17], and [32]. Among them only [40], [12] and [32] uses a SINR based interference model.
In [40] provides heuristics for ConvergeCast based on trees using the SINR model. They claim to show how to use multiple frequencies to eliminate interference. Two ConvergeCast heuristics using Dijkstra and graph coloring are provided in [12]. Similarly, Gong and Yang

first identify a convergecast tree, then construct a weight-based heuristic [32] for scheduling on the tee. The weight of a link is related to its capacity to cause interference to other links. For tree-based non-SINR models, [31] provides a variation of BFS algorithm, to derive a ConvergeCast algorithm, [17] is for ConvergeCast but assumes sink decides the location of sensors and each sensor is one hop reachable from sink node. Subgraph-based heuristics for ConvergeCast are namely [75], [70], [58], [61], [13]. Among them only [75] and [70], uses SINR model of interference. In [75] considered the objective of throughput improvement while considering fairness through a new introduced factor called *demand satisfaction factor*. After the original model was formulated as a Mixed Integer Linear Program (MILP), the key idea was to iteratively use the solutions obtained from a Linear Program (LP)- a relaxed version of the problem - as guidelines to schedule some channel. In [70], nodes are divided into clusters and a non-linear optimization model is given to get a Convergecast solution using the SINR model. Among non-SINR based models, [13] provided a Column Generation(CG) model for solving a network of 14 sensors. CG procedure does not guarantee lower bound as linear optimal is not always can be found. Similarly [61] proposes a geometric and Signomial formulation for routing and sleep sceduling in WSNs. They solved solutions for a network of 20 sensors. In [58] proposed a distributed heuristic based on random-walk algorithm for non-SINR ConvergeCast problem, they used multi-radio and multi-power to nullify interference.

None of the heuristic using SINR interference model in [40, 12, 70, 32] consider multi-rate sensors or q-coverage, and do not provide any bounds on the accuracy of their solutions. As such, they are not comparable to our work.

4.3.2 Mathematical Programming Approaches

The second class of studies considers mathematical programming tools (models and algorithms), and indeed, only provide a set of transmission configurations without providing a mechanism for ordering the configurations in order to produce a schedule.

Addis *et al.* [1] recently provided a survey of mathematical models and methods for energy-awareness into communication networks. This mainly covers non-SINR models for different network management problems. Very few studies used classical ILP formulations as they provide non scalable models. Tang *et al.* [75] considered the objective of throughput improvement while considering fairness through a new introduced factor called *demand satisfaction factor*. Initially, they formulated a MILP model, but as it was not scalable, they conducted their experiments using a heuristic. Li *et al.* [60] and Capone *et al.* [16] proposed MILP models for scheduling in wireless mesh networks.

Authors who consider decomposition models use column generation models that allow the

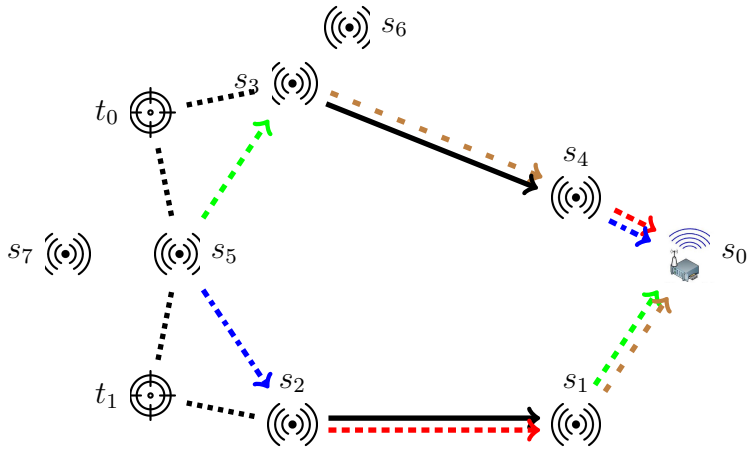
decoupling of the Time Frame Minimization Problem (TFMP) problem into two subproblems solved alternately until an optimality condition is satisfied. Earlier work studied the problem in the context of WiMax networks without considering the transmission power, see, e.g., [25], [15] and then later with the integration of the power control constraints [26], [62], [51], [76].

Kompella and Wieselthier [51], El-Najjar *et al.* [25] and several other studies provided column generation mathematical formulations for routing in the context of ad-hoc or WiMax networks and incomplete scheduling, i.e., no ordering of the transmission configurations output by their model. Uddin and Assi [76] who extended the formulation of Kompella and Wieselthier to work with multiple channels. Kaddour [44] adapted the previous column generation models for the design of wireless sensor networks subject to SINR constraints, as well as power control and rate adaptation considerations. However, due to the computational complexity of generating transmission configurations under such constraints, his model lacks scalability. Bakshi *et al.* [7] who improved the scalability of the mathematical formulation of Kaddour. Also, Bakshi *et al.* [2] highlighted the incompleteness of solutions provided by the previous mathematical formulations. The authors of [42], [2] were the first to provide a schedule based on the outputs (i.e., transmission configurations) of a mathematical programming formulation. None of the mathematical formulations provide direct ordering of transmission configurations and hence does not provide an actual schedule. This was not well advertised in the literature, as several papers wrongly claimed to produce a schedule, while they were only providing a set of unordered transmission configurations.

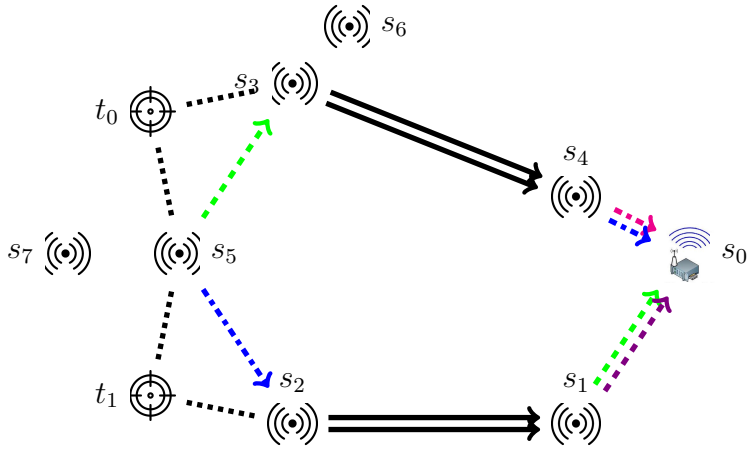
4.4 Mathematical Model for TDMA Frame Minimization (OSCC-1P)

In this section, we propose a single step ConvergeCast mathematical formulation that outputs a schedule of the transmission configurations that utilizes the minimum number of slots in order to forward all the data towards the sink. We start with a discussion of the incompleteness of previous approaches, and then proceed to give our solution.

As pointed out in [2], the mathematical formulations for wireless scheduling in the literature do not provide a complete ConvergeCast solution. There are two ways in which the provided solutions are incomplete. First, instead of computing an *ordered sequence* of configurations, they simply output a *multiset* of configurations; the ordering of this set is non-trivial to compute and is left unspecified. Second, and perhaps even more important, the provided multi-set is simply a *cover* of all the paths from sensors to sink, and *does not take into account whether or not data is available at a sensor before it transmits.*



(a) Previous TFMP formulation [2] outputs 5 configurations:$\{(s_5, s_2), (s_4, s_0)\}$, $\{(s_5, s_3), (s_1, s_0)\}$, $\{(s_3, s_4), (s_1, s_0)\}$, $\{(s_2, s_1), (s_4, s_0)\}$, $\{(s_3, s_4), (s_2, s_1)\}$, but there exists no feasible schedule with only 5 configurations.



(b) Complete ConvergeCast using 6 configurations (5 distinct ones). One feasible schedule: $\{(s_3, s_4), (s_2, s_1)\}$, $\{(s_5, s_3), (s_1, s_0)\}$, $\{(s_5, s_2), (s_4, s_0)\}$, $\{(s_3, s_4), (s_2, s_1)\}$, $\{(s_1, s_0)\}, \{(s_4, s_0)\}$

Figure 11: ConvergeCast using configurations (q=2)

To illustrate the latter issue, we use the example given in Figure 7(a). The models given in [44][7] produce as output a set of five configurations $\{(s_5, s_2), (s_4, s_0)\}$, $\{(s_5, s_3), (s_1, s_0)\}$, $\{(s_3, s_4), (s_1, s_0)\}$, $\{(s_2, s_1), (s_4, s_0)\}$, $\{(s_3, s_4), (s_2, s_1)\}$ as the solution to the ConvergeCast, using the routes shown in Figure 11(a). Each configuration satisfies the SINR constraints, and the number of times each link is present in all configurations suffices to carry the number of packets the link has to transmit. However, it is impossible to build a schedule of five slots,

with one configuration for each slot, to achieve ConvergeCast. For example, if we use the configuration $\{(s_5, s_2), (s_4, s_0)\}$ in the first slot, the link (s_4, s_0) does not yet have data, so the configuration is only partly used, and it has to be scheduled again. In fact, any schedule using the set of configurations above must use at least seven slots. This shows that the TFMP solutions in [44], [7] only produce a lower bound on the length of the schedule, and as already observed, in the case of a path, a very weak lower bound. Clearly then, using previous approaches, a second phase for obtaining a valid schedule for ConvergeCast is needed. Figure 11(b) shows a valid schedule for achieving ConvergeCast for the example of Figure 7(a) that uses six time slots.

We now propose a mathematical programming model that gives a valid and optimal transmission schedule to achieve ConvergeCast.

Let TS be the set of time slots, indexed by τ. We assume that |TS| is an upper bound on the number of required time slots. It can be calculated using any of the heuristics from the literature. We denote by L the overall set of potential transmission links, indexed by ℓ. The remaining notations that are needed for setting the model have been defined in Section 4.2.

The OSCC-1P model uses three set of variables. The first set of decision variables z_c^τ is such that each variable z_c^τ indicates if configuration c is selected, i.e., scheduled in time slot τ. The second set of variables correspond to binary variables y_{ts} indicating each if target t is covered by sensor s. The third set of variables are integer ones, such that each variables D_s^τ counts the number of data packets in the buffer of sensor s during timeslot τ. It works with a set of configurations as input for each timeslot τ, defined by $a_{\ell r}^c$. It says weather a link ℓ is used with a datarate r in a configuration c.

The proposed model, called OSCC-1P, can be written as the following one-phase ILP:

$$\text{ConvergeCast [OSCC-1P]} \qquad \min \sum_{\tau \in TS} \sum_{c \in \mathcal{C}^\tau} z_c^\tau \tag{56}$$

subject to:

$$\sum_{s\in S} y_{ts} = q \qquad t \in T \tag{57}$$

$$D_s^0 = 0 \qquad s \in S \cup \{s_0\} \tag{58}$$

$$D_s^0 + \sum_{t\in T} y_{ts}\sigma = D_s^1 \qquad s \in S \tag{59}$$

$$D_s^\tau - \sum_{c\in C^\tau} \sum_{\ell\in\omega^+(s)} \sum_{r\in R} rT_s a_{\ell r}^c z_c^\tau \qquad + \sum_{c\in C^\tau} \sum_{\ell\in\omega^-(s)} \sum_{r\in R} rT_s a_{\ell r}^c z_c^\tau = D_s^{\tau+1}$$

$$\tau \in TS \setminus \{0\}, s \in S \cup \{s_0\} \tag{60}$$

$$D_s^\tau \geq \sum_{c\in C^\tau} \sum_{\ell\in\omega^+(s)} \sum_{r\in R} rT_s a_{\ell r}^c z_c^\tau \qquad \tau \in TS \setminus \{0\}, s \in S \cup \{s_0\} \tag{61}$$

$$D_{s_0}^{|TS|} = mq\sigma \tag{62}$$

$$\sum_{c\in C} z_c^\tau \leq 1 \qquad \tau \in TS \tag{63}$$

$$y_{ts} \in \{0,1\} \qquad t \in T, s \in S \tag{64}$$

$$z_c^\tau \in \{0,1\} \qquad \tau \in TS, c \in C \tag{65}$$

$$D_s^\tau \in Z^+ \cup \{s_0\} \qquad \tau \in TS, s \in S \cup \{s_0\}. \tag{66}$$

Constraints (57) ensures that every target is covered by exactly q sensors. Constraints (58) initializes all sensors with zero readings in slot $\tau = 0$. We assume targets as a set of nodes which can be monitored by any sensor if they are within the given sensing-range. Constraints (59) ensures that the sensor directly monitoring targets will take initial σ bits of readings from time slot 1. Constraints (60) are flow constraints to ensure that the source of a link looses upto σ bits and the destination of a link gains those bits of a packet whenever we schedule a specific link. Constraints (61) define cutting-plane inequalities: although not necessary for the model, they are useful to obtain a better lower bound, see an illustrative example in Section 4.5.1. They ensure that sensor node s only sends data it has. Constraints (62) make sure that all packets are destined to the sink. Constraints (59), (60) and (62) guarantee that each data packet has a path from its target to the sink. Constraint (63) ensures that at most one configuration can be used in any timeslot. Constraints (57)-(66) ensure that the number of times each link is included in all the scheduled configurations using (z_c^τ) are sufficient to forward the allocated traffic to the sink.

The above OSCC-1P model works with a set of configurations for each timeslot τ. But how to generate this set? The set of all possible configurations is exponentially large. In the next section we explain how to re-interpret the model (56)-(66) as a decomposition model

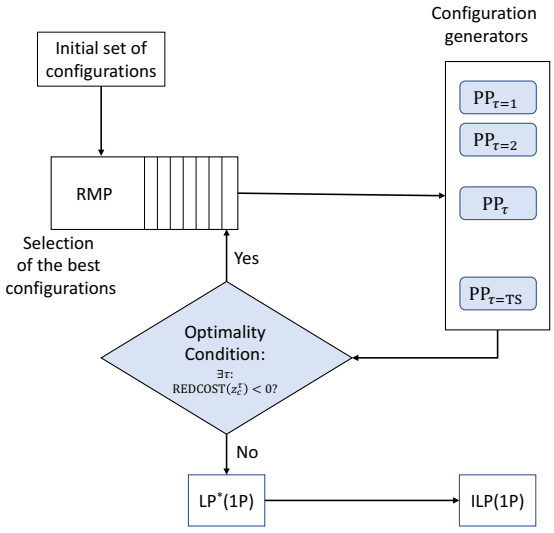


Figure 12: Solution flowchart

in order to be able to design a scalable solution process.

4.5 Solution Scheme of OSCC-1P

Column Generation (CG) is an efficient and exact algorithm for solving a set of large-scale linear programs. CG relies on the fact that most of the variables will be equal to zero in the optimal linear programming solution. Hence, only a subset of columns needs to be explicitly enumerated when solving a CG model. This property is also true for the ConvergeCast model (56)-(66) as it involves a very large number of variables due to the combination of different power levels, links, configurations etc, but only some of them will be considered in an optimal solution. This means we can use CG for solving the linear-relaxed OSCC-1P model to reach an optimal ILP solution (i.e., schedule of the selected transmission configurations). Then, we will use all the configurations generated in this process to solve the resulting OSCC-1P ILP model and obtain a near-optimal solution of the ConvergeCast problem.

The solution flow is summarized in the flowchart shown in Figure 12. The **selection of the best transmission configurations** is done via the so-called OSCC-1P Restricted Master Problem (or OSCC-1P-RMP for short), made of a restricted set of transmission configurations for each time slot, which we will discuss in Section 4.5.1. **Configuration Generators** use a set of so-called Pricing Problems in the mathematical programming literature. We denote them by (OSCC-1P-Pricing) as explained in Section 4.5.2, one for each time slot (τ). Each pricing problem generates a new configuration (c^{new}) that can improve the value of the objective function of OSCC-1P-RMP, if its objective (called reduced cost in

mathematical programming literature, is negative.

We use the initial feasible solution given by [2] as the **initial set of configurations** and then solve the OSCC-1P-RMP model. Using the dual values of 1P-RMP model output, we solve all OSCC-1P-Pricing problems in a round robin fashion. We add all the configurations with a negative reduced cost to OSCC-1P-RMP and re-solve OSCC-1P-RMP. The process is repeated until none of the pricing problems is anymore able to produce a negative reduced cost, and the optimal solution of the linear relaxation of OSCC-1P is reached ($z^{\star}_{\text{LP}}(\text{OSCC-1P})$).

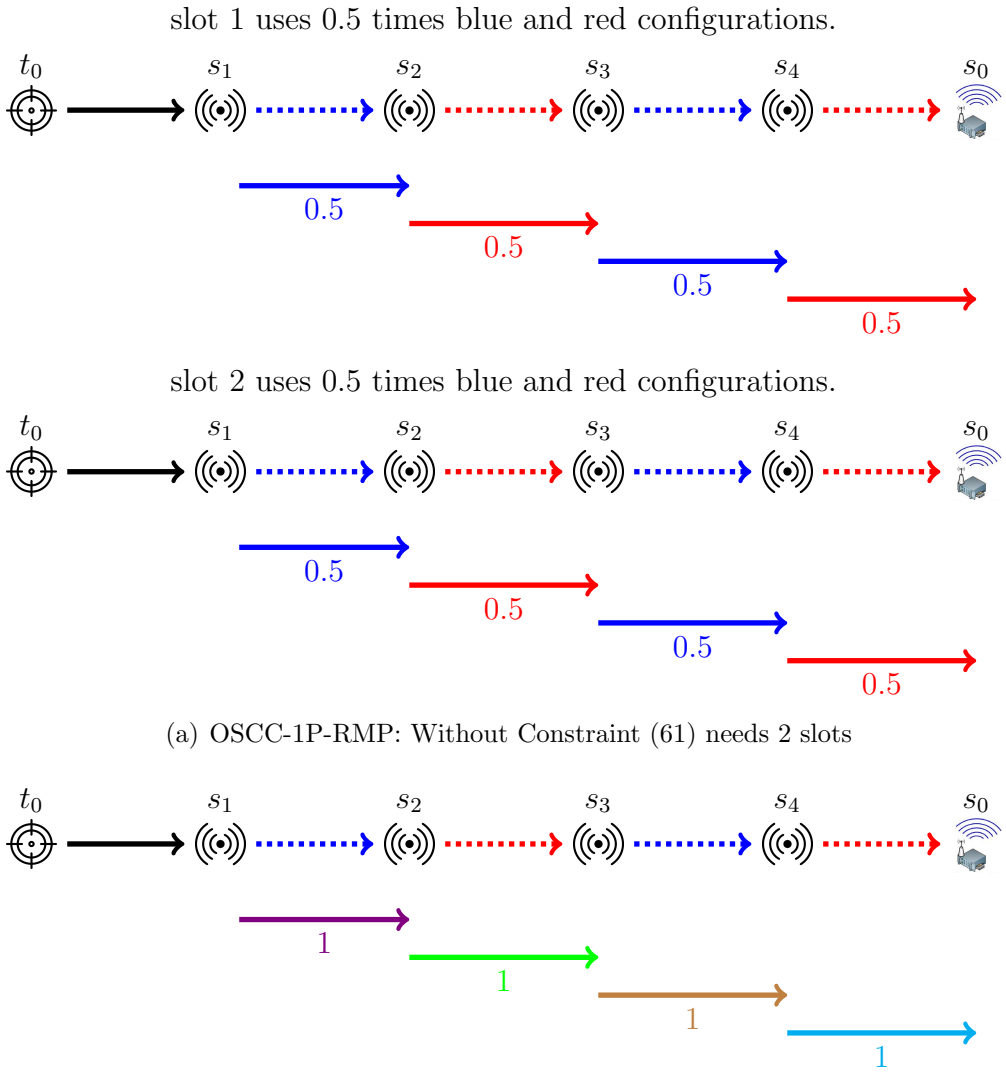
Once we reach the optimal solution of the linear relaxation of OSCC-1P, we use all the configurations generated in the process to produce an optimal solution of the linear relaxation of OSCC-1P and solve it exactly in order to get an ILP solution. That resulting ILP solution, $\tilde{z}_{\text{ILP}}(\text{OSCC-1P})$, defines a an ε-optimal solution of the ConvergeCast problem, where ε is defined as follows:

$$\varepsilon = \frac{\tilde{z}_{\text{ILP}}(\text{OSCC-1P}) - z^{\star}_{\text{LP}}(\text{OSCC-1P})}{z^{\star}_{\text{LP}}(\text{OSCC-1P})}.$$

4.5.1 Restricted Master Problem (OSCC-1P-RMP)

OSCC-1P-RMP is derived from (56)-(66) with a restricted set of variables/configurations. We next comment on why constraints (61) were instrumental in getting accurate lower bounds, and consequently a highly scalable solution scheme.

Constraint (61) ensures that in a continuous solution, node s will send data only when it has data. This constraint is not required for an ILP solution of OSCC-1P as it is taken care of by integrality constraints of decision variables D^{τ}_{s}, z^{τ}_{c}. Without Constraints (61) in OSCC-1P-RMP, it is possible to use some links where incoming data equals outgoing data even when a source link does not have available data. For instance, consider the example in Figure 13. If we consider a path of length 4, in which only sensor s_1 has one packet and needs to forward it to s_0, then as shown in Figure 13(a) we need 2 slots without using Constraints (61). A drawback of not having constraint (61) is that some sensors (like s_2) forward data even when they have no available data. After including constraint (61), we are in the situation of Figure 13(b). Four slots are required, and this greatly helps to get a ConvergeCast solution with a very high precision (ε) using OSCC-1P-RMP.



(a) OSCC-1P-RMP: Without Constraint (61) needs 2 slots

(b) OSCC-1P-RMP:With Constraint (61) needs 4 slots

Figure 13: Continuous ConvergeCast Solution: white node - target, gray node - sensor, lightgray node - sink.

4.5.2 OSCC-1P-Pricing: Configuration Generator

The objective of 1P-Pricing problem is the reduced cost associated with variable z_c^τ. It is computed using the dual values generated by 1P-RMP. Let $u_{s\tau}^{(60)}, u_{s\tau}^{(61)}, u_\tau^{(63)}$ be the values of the dual variables with respect to constraints (60), (61) and (63), respectively.

The pricing problem can be stated as follows:

$$\text{[OSCC-1P-Pricing]} \quad \min_{c \in \mathcal{C}} \quad \text{RED COST}(z_c^\tau)$$
$$= 1 - u_\tau^{(63)} + \sum_{s \in S} u_{s\tau}^{(61)} \sum_{r \in R} rT_s \sum_{\ell \in \omega^+(s)} a_{\ell r}^c$$
$$- \sum_{s \in S} u_{s\tau}^{(60)} \sum_{r \in R} rT_s \left(\sum_{\ell \in \omega^-(s)} a_{\ell r}^c - \sum_{\ell \in \omega^+(s)} a_{\ell r}^c \right) \quad (67)$$

subject to:

$$\sum_{r \in R} \Big(\sum_{\ell \in \omega^+(s)} a_{\ell r} + \sum_{\ell \in \omega^-(s)} a_{\ell r} \Big) \leq 1 \qquad s \in S \quad (68)$$

$$\sum_{r \in R} a_{\ell r} = 0 \qquad \ell \in \omega^+(s_0) \quad (69)$$

$$p_s d_\ell^{-\alpha} - L a_{\ell r} - \beta_r \sum_{s'' \in S, s'' \neq s} p_{s''} d_{s''s'}^{-\alpha} \geq \beta_r N_0 - L \qquad s \in S, \ell \in \omega^+(s), r \in R \quad (70)$$

$$p_s \leq P_{\max} \sum_{\ell \in \omega^+(s)} \sum_{r \in R} a_{\ell r} \qquad s \in S \quad (71)$$

$$a_{\ell r} \leq L' \, p_s \qquad s \in S, \ell \in \omega^+(s), r \in R \quad (72)$$

$$a_{\ell r} \in \{0, 1\} \qquad \ell \in L, r \in R \quad (73)$$

$$p_s \geq 0 \qquad s \in S \quad (74)$$

Variable $a_{\ell r}$ indicates a transmission link between sensors sensors s and s' using rate r and p_s denotes the transmission power of sensor s. L and L' are large positive constants. Constraint (68) states that a node cannot transmit and receive at the same time, whereas constraint (69) prevents the sink node s_0 from transmitting. Constraint (70) enforces the SINR condition (54). Constraints (71) and (72) binds every transmission link with its power. If a transmission link is not active, the power would be set to zero, and vice-versa.

4.6 Results

In this section, we present the performance evaluation of our new model for ConvergeCast and of the scalability of its solution scheme. We also provide comparison with our previous best algorithm described in [2], refered as CC-2P or 2P(ConvergeCast-two phase), as it was a two phase algorithm: firstly, the computation of the transmission configurations and secondly, their ordering in order to produce a feasible schedule.

We are interested in assessing:

1. the accuracies of the ε-solutions that we output. Accuracies are given by comparing the lower bound (LB) given by the optimal LP solution (called OSCC-1P lower bound) versus the upper bound (UB) given by the ILP (called the OSCC-1P solution).

2. the lower bound given by the TFMP solution in [44] (called the CC-2P lower bound) versus our lower bound and

3. the upper bound given by the TFMP_ROS schedule from [2] (called the CC-2P upper bound) versus our upper bound.

All the experimental results are computed as an average over 10 instances using randomly generated topologies in a square grid of side 625m. We ensure that each topology used in the experiments is connected, and allows for q-coverage of all targets; if the randomly generated topology does not meet these requirements, we simply discard it, and generate another topology. We consider the sensing range to be 150m, the maximum transmission range as 100m, derived using the data rate of 150kbps, and a maximum transmission power of 13mW. Finally, we considered coverage levels q from 1 to 3. We consider 40, 50, 60, and 70 sensors, and the number of targets is incremented from 10 to the number of sensors in each experiment, in steps of 10. All these parameters match with that of the parameters present in the literature [44] and outdoor sensor specifications [65].

4.6.1 1-Coverage: Solution Accuracies and Computational Times

In this first set of experiments, we investigate the quality of the schedules that our new model and solution process output for 1-coverage.

Results are summarized in Table 13. Observe that the OSCC-1P upper and lower bound are quite close in all experiments; in contrast, the CC-2P upper and lower bound can differ by as much as 55 %, especially for large number of sensors monitoring few targets. Secondly, the OSCC-1P lower bound is 4 % better than the CC-2P lower bound on average, and over 10 % better when the number of targets is small. Finally, the quality of our OSCC-1P solution (upper bound) is better than the CC-2P upper bound by 15 % on average. We can conclude that our method gives a solution that is very close to optimal and provides a big improvement on the state-of-the-art, particularly when the number of targets is low compared to the number of sensors.

Table 13 also compares the accuracy and bound improvement of OSCC-1P algorithm over CC-2P algorithm, where accuracy corresponds to the closeness of schedule with the LB, expressed as a percentage. The accuracy (ϵ) of OSCC-1P is seen to be much better

Table 13: ConvergeCast Analysis (q=1)

targets	LB			UB			ε_2	ε_1
	2P	1P	I^{LB}	1P	2P	I^{UB}		
40 sensors								
10	24.3	27.2	10.6	28.1	32.7	16.4	34.6	3.4
20	47.0	47.6	1.1	48.4	55.0	13.6	16.9	1.7
30	69.8	71.5	2.4	72.8	85.8	17.9	23.0	1.8
40	93.7	95.3	1.7	96.3	111.9	16.2	19.4	1.1
50 sensors								
10	17.8	19.0	5.9	19.7	23.6	19.8	32.2	3.9
20	33.6	35.2	4.7	36.6	41.1	12.4	22.4	3.7
30	42.4	46.0	7.8	46.4	49.6	6.9	17.0	0.9
40	63.2	65.5	3.5	66.1	72.6	9.8	14.9	0.9
50	76.1	81.4	6.5	81.7	90.1	10.3	18.4	0.4
60 sensors								
10	16.1	17.9	10.0	18.2	22.9	25.6	42.4	2.0
20	30.0	31.2	4.1	31.9	40.1	25.8	33.9	2.1
30	41.4	43.7	5.3	45.0	49.9	10.8	20.4	2.9
40	59.2	60.6	2.2	62.9	71.5	13.7	20.7	3.8
50	72.1	72.4	0.3	73.6	83.3	13.2	15.5	1.7
60	81.3	83.4	2.5	85.0	93.2	9.6	14.6	1.9
70 sensors								
10	13.1	15.1	13.2	16.7	20.3	22.0	55.3	10.5
20	25.7	26.4	2.5	28.8	34.5	19.8	34.2	9.2
30	38.6	38.9	0.8	40.9	49.3	20.5	27.8	5.2
40	51.9	52.2	0.6	55.6	65.1	17.0	25.5	6.6
50	64.6	64.6	0.0	69.9	79.0	13.1	22.2	8.1
60	74.9	75.6	1.0	81.2	87.0	7.2	16.2	7.4
70	91.1	91.9	0.8	95.3	101.0	6.0	10.8	3.7

$\varepsilon_2 = 100 \times (UB^{2P} - LB^{2P})/LB^{2P}$; $\varepsilon_1 = 100 \times (UB^{1P}$ - $LB^{1P})$ / LB^{1P}
$I^{UB} = 100 \times (2P^{UB} - 1P^{UB})$ / $1P^{UB}$; $I^{LB} = 100 \times (2P^{LB} - 1P^{LB})$ / $1P^{LB}$

than that of CC-2P, particularly as the number of targets decreases for a fixed number of sensors. Figure 14 demonstrates that the OSCC-1P solution is much better than that of CC-2P solution. Table 14 compares CPU time of OSCC-1P and CC-2P, we need more time for the OSCC-1P algorithm as it gives a complete solution.

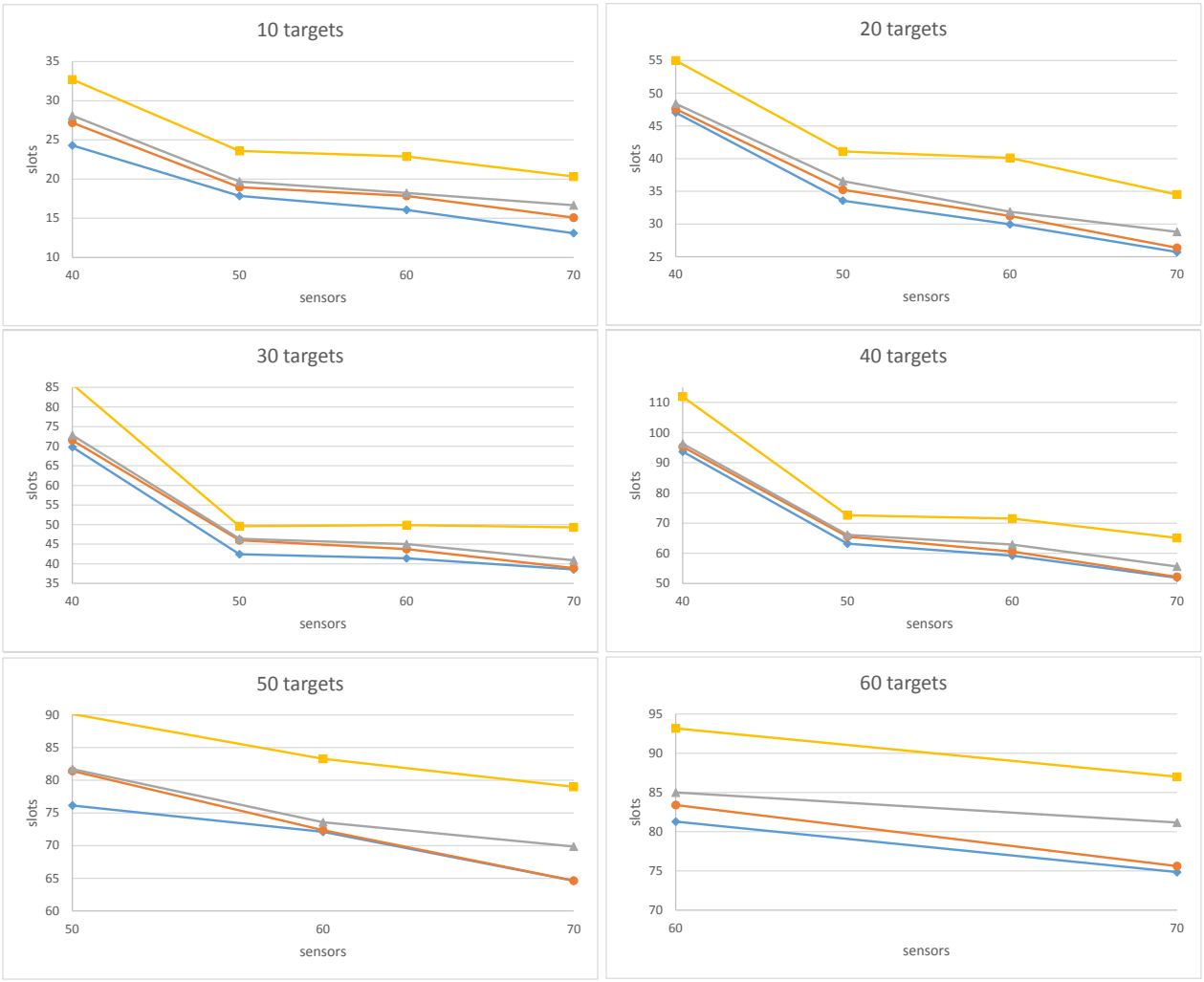


Figure 14: ConvergeCast lower bound and upper bound (q=1); $\diamond$= LB^{2P},o= LB^{1P},Δ= UB^{1P}, $\square$= UB^{2P}

Table 14: ConvergeCast CPU time in hours ($q = 1$)

$\|S\|$	2P		1P		1P LB		1P UB	
	LB	UB	UB	LB	σ	max	σ	max
10 targets								
40	< 0.1	< 0.1	0.8	1.9	0.3	1.5	0.9	3.3
50		0.1	3.8	3.8	3.1	9.2	3.1	9.2
60		0.5	4.7	1.9	7.1	22.8	6.6	22.8
70		4.1	33.4	24.3	21.5	70.0	21.9	70.0
20 targets								
40	<0.1	<0.1	0.6	1.4	0.5	1.5	1.2	3.7
50		0.1	6.5	6.3	3.6	10.2	4.0	10.3
60		5.2	5.7	2.3	6.7	6.5	6.4	6.6
70		2.9	55.1	54.3	21.5	86.6	22.2	90.2
30 targets								
40	<0.1	<0.1	0.8	1.8	0.7	2.0	1.9	5.5
50		0.1	2.6	2.2	3.7	11.3	5.4	17.4
60		0.5	5.4	5.4	5.5	16.1	5.5	16.2
70		3.5	45.3	42.3	23.9	66.7	26.4	79.0
40 targets								
40	<0.1	0.1	0.8	2.0	0.7	2.6	2.1	7.5
50		0.2	3.0	3.0	3.2	8.6	3.2	8.7
60		0.7	10.8	10.6	9.3	23.9	9.5	24.5
70		2.5	52.8	42.3	32.2	100.6	36.0	103.1
50 targets								
50	<0.1	0.2	0.9	0.9	0.6	2.4	0.6	2.4
60		0.6	17.6	14.2	11.5	35.5	18.3	55.9
70		2.8	58.0	42.1	24.2	67.4	37.8	101.1
60 targets								
60	<0.1	0.6	9.6	9.1	10.6	34.0	11.9	38.5
70		2.7	52.7	38.0	21.3	59.8	32.8	94.8
70 targets								
70	<0.1	2.6	54.2	33.8	39.9	101.8	50.2	145.9

σ = standard deviation, max = maximum time over 10 instances

Table 15: q-Cover ConvergeCast using 40 sensors

$\|T\|$	$q=2$					q =3				
	2P	1P	1P	2P	UB	2P	1P	1P	2P	UB
	LB	LB	UB	UB	I^{UB}	LB	LB	UB	UB	I^{UB}
10	50.3	50.9	52.4	63.4	21.0	78.5	79.1	80.7	94.3	16.9
20	97.5	97.9	98.7	114.6	16.1	152.2	152.3	153.7	175.8	14.2
30	144.2	145.3	146.5	170.2	16.2	225.1	225.3	227.1	267.1	17.6
40	194.6	197.2	198.4	228.7	15.3	302.7	303.1	305.1	338.4	10.9
$I^{UB} = 100 \times (2P^{UB} - 1P^{UB})/1P^{UB}$										

4.6.2 q-Coverage with $q \geq 2$: Solution Accuracies and Computational Times

We now look at the same type of comparisons in the context of q-coverage for $q = 2$ and $q = 3$.

Table 15 provides the LB and UB values of the OSCC-1P method for different coverage levels. We fixed the number of sensors as 40 and varied targets from 10 to 40. It is obvious that for a higher coverage level we need more slots, to forward more data. We can observe that the OSCC-1P method remains highly accurate, and also that we obtain 10-20% improvement in the length of the schedule compared to the earlier CC-2P method. Table 16 compares the LB and UB values of the OSCC-1P method for single and multiple data-rates.

When Using multiple rates, we are able to reduce the number of packet transmissions by sending more data in a slot. We can see that the lower bound is better than that of the single rate. However, the accuracy of the schedule deteriorates a bit.

Table 16: Single vs. Multi data rate

		single			multi		
$\|T\|$	$\|S\|$	LB	UB	ε	LB	UB	ε
10	10	29.0	29	0.0	27.0	27	0.0
20	20	61.3	63	2.8	59.0	63	6.0
30	30	76.0	77	1.3	63.5	77	21.3
40	40	87.3	89	1.9	74.0	83	12.2
$\varepsilon = 100 \times (UB - LB)/LB$							

4.7 Conclusion

For the NP-hard problem of ConvergeCast, we provided a first complete mathematical formulation to get an optimal "true" schedule, i.e., not only the minimum number of required configurations, but also the ordering of the selected configurations, all in one step.. The scalability of this formulation is enhanced by adding a set of valid inequalities in order to strengthen its linear relaxation. The enhanced resulting LB improves by 4 % the best one provided by a mathematical programming formulation. On the other hand, the quality and accuracy of the optimized ConvergeCast solution is improved by about 15 % when compared to the best previously available solution.

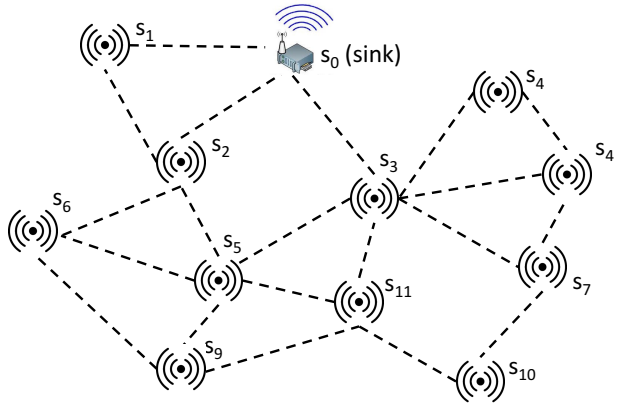
Chapter 5

Optimal Aggregated ConvergeCast Scheduling with an SINR Interference Model

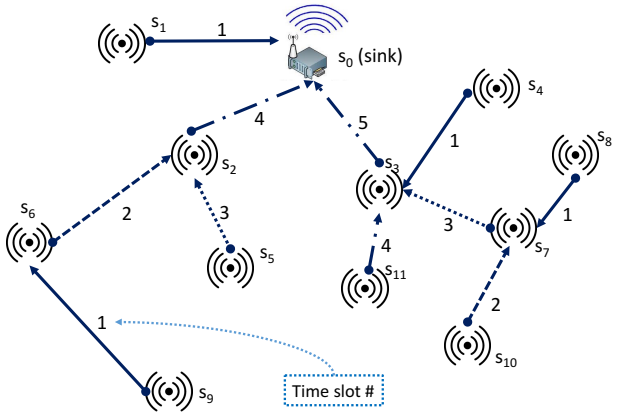
M. Bakshi, B. Jaumard, and L. Narayanan. Optimal aggregated convergecast scheduling. submitted for publication, 2017. An extended abstract of this paper has been submitted to IEEE International Conference on Wireless and Mobile Computing, Networking and Communications. (WiMob), 2017.

In target coverage applications, wireless sensors are required to periodically collect data about targets they monitor and send them to a central node called the sink. In many situations, what is of interest is not to collect *every* item of data but a *function* of the data, such as the minimum or maximum or average reading. In such cases, tremendous energy savings can be obtained by requiring every intermediate sensor nodes to *aggregate* the data it receives before forwarding the *function* value to the sink, thereby drastically reducing the number of required packet transmissions, and consequently both the time needed for the sink to receive the information, and the energy used. For example, if the sensors are monitoring the temperature at each target, and what is required for the sink is to know the maximum temperature over all targets, each sensor needs to forward only the maximum of its own data and those received from its predecessors. Such a ConvergeCast operation is called an *Aggregated ConvergeCast*.

Given a set of sensor locations, and a sink node, we consider the problem of finding a minimum-length schedule for Aggregated ConvergeCast. We assume a Time Division Multiple Access (TDMA) network, and a SINR (Signal to Interference plus Noise Ratio) model of interference. For many WSN applications, TDMA is considered a more efficient medium access scheme than random access schemes [39]. A SINR model of interference is



(a) Aggregated ConvergeCast Input



(b) Aggregated ConvergeCast Tree and Schedule

Figure 15: Aggregated ConvergeCast

a more realistic model of interference than the protocol model of interference: a receiver node receives a packet so long as the signal to interference plus noise ratio is above a certain threshold. As in most studies, we assume that every sensor is monitoring a target and has an item of data to send to the sink.

Figure 15 demonstrates an instance of Aggregated ConvergeCast and a possible solution. Figure 15(a) shows the input for the problem: the location of the set of sensors $\{s_1, \ldots, s_{11}\}$, and the sink node s_0. Figure 15(b) gives an Aggregated ConvergeCast tree. Observe that without aggregation, we need 22 packet transmissions using this tree, and for a given interference scheme, it can be shown that any schedule would be of length at least 12 slots. However, if each sensor waits to receive information from its children, and aggregates its own data with that received from its children, the operation can be achieved using 11 packet transmissions, with a schedule that requires only 5 slots.

Each of the monitoring sensors needs to aggregate and send the information concerning

the target it is monitoring along a path to the sink, as shown in Figure 15(b). Since sensor s_3 is monitoring its own target and receives a packet from s_4, s_7 and s_{11}, it needs to aggregate four packets and send an aggregated packet once towards the sink. Each sensor has a path to the sink as indicated in Figure 15(b). The information acquired at the sink is commonly the aggregated information like "maximum" or "average", so that we can accumulate data rapidly and reduce consumption of transmission power. At the same time, interference from simultaneous transmissions is also reduced as we use less communications.

Assuming that a given sensor aggregates only once the information it gets from its predecessors together with its own information, an easy observation is that a complete and exact Aggregated ConvergeCast solution consists of (i) a tree and (ii) a schedule (ordering) of the links in the tree that avoids interference. That observation was made in the context of the protocol interference model by [63], and remains valid for the SINR interference model. Observe that interference is caused not just by tree links, but also by non-tree links. For instance in Figure 15(b), the tree links (s_5, s_2) and (s_{11}, s_3) cannot be scheduled in the same time slot, even though the receivers of the two links are different, because of the existence of the non-tree link (s_5, s_3) which causes interference at s_3.

The Aggregated ConvergeCast problem is known to be NP-hard in both protocol and SINR models of interference [18], [57]. In fact, even given an aggregation tree, finding an optimal aggregation schedule is NP-hard [77]. The solutions given in the literature are either heuristics with no indication of how far the solution is from the optimal solution, or approximation algorithms that perform badly in practice.

Our Results. We propose the first mathematical model that outputs an optimal schedule for Aggregated ConvergeCast using the SINR interference model. Our solution is a one-phase method that simultaneously builds a tree and a schedule. To ensure the scalability of the solution process, we use a large scale optimization modelling and method to solve the linear relaxation of the proposed ILP (Integer Linear Programming) model. We are then able to solve problems of upto 70 sensors, and obtain schedules, which are about 50% better than the schedules output by the best previously proposed heuristic with a SINR model of interference [77]. We added cutting planes in order to speedup further the solution process.

The paper is organized as follows. We describe the background in Section 5.1, and related work in Section 5.2. We propose a mathematical decomposition model that achieves an optimal Aggregated ConvergeCast in Section 5.3. Finally, numerical results and concluding comments are given in Section 5.4 and 5.5, respectively.

5.1 Background

In this section, we define the notation and then the network model.

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of n wireless sensors and s_0 the sink node. Sensor s can forward data to another sensor s', if the Euclidean distance between them is less than the maximum transmission range. This leads to define a graph $G = (V, L)$ such that each node of $V \equiv S$ is associated with a sensor, and there is a directed link $(s, s') \in L$ if sensor s' is within the transmission range of s. Sensors communicate using wireless transmissions and need to address the possibility of radio interference. We use the Signal to Interference plus Noise Ratio (SINR) model of interference. At the intended receiver of a link $\ell = (s, s')$, SINR refers to the ratio of the signal received to that of the interference from the other transmission links plus noise. SINR is defined as:

$$\text{SINR}_{(s,s')} = \frac{p_s d_{(s,s')}^{-\alpha}}{N_0 + \sum\limits_{s'' \in S', s'' \neq s} p_{s''} d_{(s'',s')}^{-\alpha}} \tag{75}$$

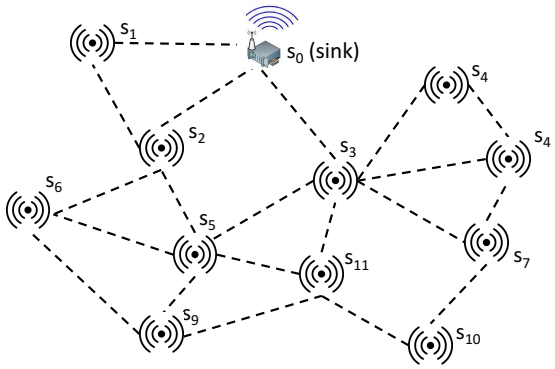
where p_s is the transmission power of sensor s, $d_{(s,s')}$ is the distance from s to s', α is the path loss exponent, N_0 is the thermal noise power and S' is the set of sensors that are transmitting at the same time as s. According to Modulation and Coding Schemes (MCS) [54] of IEEE 802.15.4, a signal can be decoded successfully if the SINR measured at the receiver is above a given threshold β, which depends on the transmission rate. We used parameter values based on the out-door sensor specifications [65], [44], i.e., we use power level $p = 0, 0.013$W, $N_0 = 10^{-6}$, a single data rate = 250kbps and $\beta = 1.3$.

A transmission **configuration**, or configuration for short, refers to a set of links that can be scheduled simultaneously while satisfying the SINR constraint above. Note that the problem of finding a configuration with maximum cardinality is already NP-hard for the protocol interference model [19], as it is equivalent to the maximum independent set problem.

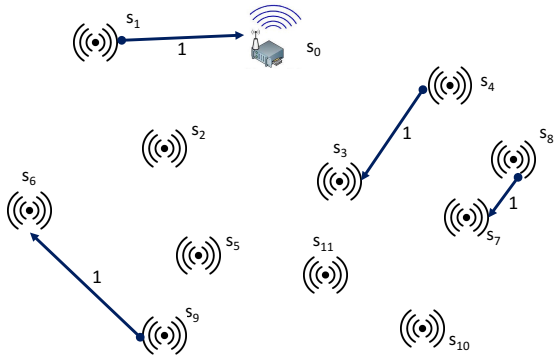
We assume the use of Time Division Multiple Access (TDMA) as the MAC layer protocol for collision resolution. Therein, time is divided into frames, each frame contains a fixed number of slots, and each slot holds one configuration.

We assume that each sensor monitors a target, and that initially each sensor s has a reading r_s. We assume the aggregation function is a function f that is defined on a set of readings. A solution to the Aggregated ConvergeCast problem needs to find a minimum length TDMA frame, i.e., an ordered sequence of configurations. Such a sequence achieves the Aggregate ConvergeCast operation, that is, each sensor transmits exactly once, and the sink can derive the value of $f(r_1, r_2, \ldots, r_n)$ from the data it receives. Figure 16 shows a valid schedule for

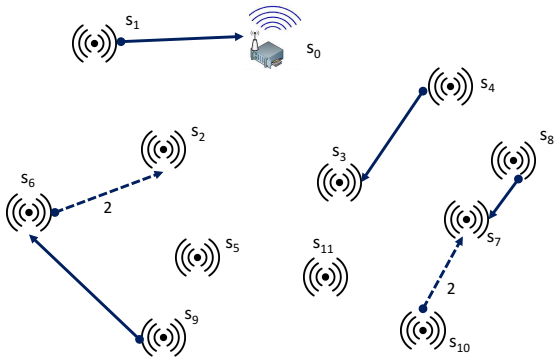
achieving Aggregated ConvergeCast that uses five time slots (five configurations).



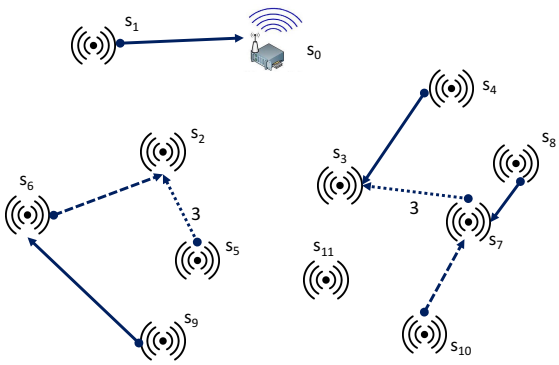
(a) Aggregated ConvergeCast Input



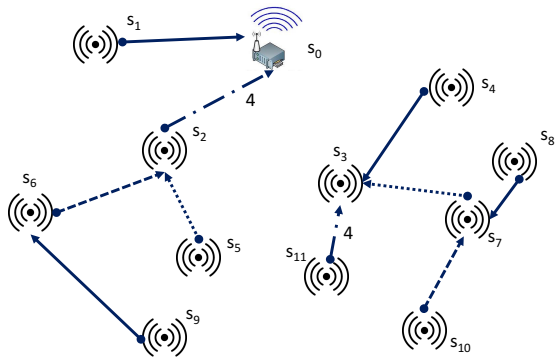
(b) Slot 1: $\{(s_4, s_3), (s_9, s_6), (s_8, s_7), (s_1, s_0)\}$



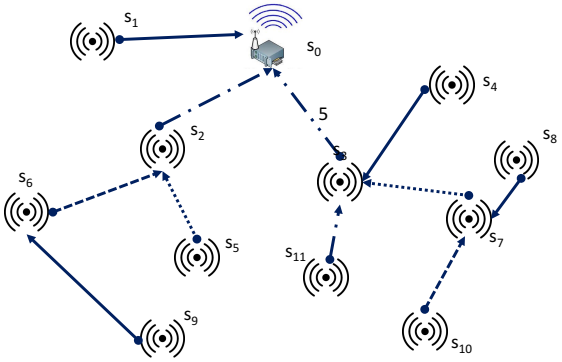
(c) Slot 2: $\{(s_{10}, s_7), (s_6, s_2)\}$



(d) Slot 3: $\{(s_7, s_3), (s_5, s_2)\}$



(e) Slot 4: $\{(s_2, s_0), (s_{11}, s_3)\}$



(f) Slot 5: $\{(s_3, s_0)\}$

Figure 16: Aggregated ConvergeCast using 5 Configurations

In particular, a valid schedule satisfies the following constraints:

1. Every sensor is monitoring a target and consequently, each sensor sends exactly one packet.

2. A sensor cannot receive a packet during or after the time slot when it transmits.

3. Links scheduled in the same time slot satisfy SINR requirements.

4. The sink node receives all the aggregated data.

Aggregated ConvergeCast is proved to be NP-hard in Chen *et al.* [18] even for the protocol interference model.

5.2 Related work

Most of the existing work for Aggregated ConvergeCast Scheduling uses the protocol, i.e., graph-based, interference model and a minimum spanning tree rooted at the sink node. We start with a brief description of these studies, and then describe related work on the SINR interference model.

5.2.1 Using Protocol Interference Model

Aggregated ConvergeCast Scheduling for unit disk graphs using the protocol interference model is studied in Kesselman *et al.* [49], Wan *et al.* [78], Xu *et al.* [81], Gagnon *et al.* [29], Gandhi *et al.*[30], Guo *et al.* [35], Pan *et al.* [66], Jakob *et al.* [41], Yousefi *et al.* [82]. Guo *et al.* [35] gave an Aggregated ConvergeCast schedule of length $O(D + \delta)$, where D is the diameter of the input graph and δ is the maximal degree. As every Aggregated ConvergeCast schedule is of length at least D, it gives $O(\delta)$-approximation ratio (δ can be $\Theta(n)$). Gandhi *et al.*[30] gave a randomized approximation algorithm ratio of $\sqrt{\tilde{d}n}$, where $\tilde{d}$ is the average degree. Kesselman *et al.* [49] showed that aggregation can then be achieved in $O(\log n)$ assuming the Collision Detection protocol is available at each sensor. Pan *et al.* [66] construct a scheduling tree using a weight function based on receiver's depth and number of children and propose a scheduling algorithm based on neighbours' degree. Jakob *et al.* [41] uses top-down approach and produce a heuristic schedule without any tree construction. Yousefi *et al.* [82] provided another heuristic based on a distributed algorithm. Erzin *et al.* [28] proved that for a given Aggregated ConvergeCast tree, the problem of finding an optimal schedule is still NP-hard using protocol interference model.

5.2.2 Using SINR Interference Model

Moscribroda *et al.* [64], Li *et al.* [59], Li *et al.* [57], Halldorsson *et al.*[37], and Wang *et al.* [77] study the problem using the SINR interference model and propose heuristics. Assuming discrete power levels, Moscribroda *et al.* [64] proposed a polylogarithmic bound of $O(\log^4 n)$ slots for their scheduling algorithm using an SINR model, where n is the number of sensors.

For uniform or linear power levels, their algorithm needs $O(n^2 \log n)$ slots. Halldorsson *et al.* [37] relax the SINR interference model by using unlimited transmission power, ignoring noise, and $\alpha > 2$, where α is the path loss exponent. They then provide an algorithm that connects an arbitrary point set in $O(\log n)$ slots, improving on the results of Moscribroda *et al.* [64].

Li *et al.* [59] provided a heuristic using dominating sets for the Aggregated ConvergeCast. Li *et al.* [57] suggested another $O(\log^3 n)$ heuristic. This last heuristic uses a round-based approach; in each round it gives preference to the smaller links and selects set of links satisfying a simplified SINR condition. Data is transmitted on the selected links, whose source sensors are subsequently removed from consideration. This process is repeated until all sensors forward the aggregated data to the sink. The major drawback of this approach is that it assumes the network is connected even after removing some links. Wang *et al.* [77] proposed a heuristic with a lower bound of $O(d \log^2 n))$, where d is the depth of the tree. The heuristic also goes in rounds; in each round they schedule all the links in the highest layer first and repeat such link scheduling in each round for all the links in different layers.

We will compare our algorithms with this last heuristic as it appears to be the most efficient one using a SINR interference model.

Ebrahimi *et al.* [24] give a schedule using a mathematical model for a related problem with several aggregated trees.

5.3 Optimal Model for Aggregated ConvergeCast

We now propose a first exact Aggregated ConvergeCast ILP model, called ACC, assuming each sensor has a target to monitor. After presenting the solution of the ACC model in Section 5.3.2, we propose two enhancements, called ACC-MP and ACC-PP, in order to improve the convergence speed of the solution process in Sections 5.3.3 and 5.3.4, respectively.

5.3.1 Basic ACC Model

We now describe the ACC model in order to solve the Aggregated ConvergeCast problem. We use two sets of decision variables. The first set corresponds to decision variables z_c^τ, with each variable equal to 1 indicating that a link transmission configuration c is scheduled in time slot τ, and 0 otherwise. The second set of variables is such that: $P_\ell^\tau = 1$ if link ℓ is used for transmission in slot τ, 0 otherwise.

In the constraints, coefficient a_ℓ, with $\ell = (s, s')$, indicates a transmission link from sensor s to s' with power p_s.

We denote by $\omega^-(s)$ the incoming links (data) to s, and by $\omega^+(s)$ the outgoing links (data) from s.

$$[\text{ACC}] \qquad \min \sum_{\tau \in TS} \sum_{c \in \mathcal{C}^\tau} z_c^\tau \tag{76}$$

subject to:

$$\sum_{\tau \in TS} \sum_{\ell \in \omega^+(s)} P_\ell^\tau = 1 \qquad s \in S \tag{77}$$

$$(1 - \sum_{\ell \in \omega^+(s)} P_\ell^\tau)(n+1) \geq \sum_{\tau' \in TS : \tau' > \tau} \sum_{\ell' \in \omega^-(s)} P_{\ell'}^{\tau'} \qquad s \in S, \tau \in TS \tag{78}$$

$$\sum_{c \in C} z_c^\tau \leq 1 \qquad \tau \in TS \tag{79}$$

$$\sum_{c \in C^\tau} a_\ell^c z_c^\tau = P_\ell^\tau \qquad \ell \in L, \tau \in TS \tag{80}$$

$$z_c^\tau \in \{0,1\} \qquad c \in C^\tau, \tau \in TS \tag{81}$$

$$P_\ell^\tau \in \{0,1\} \qquad \ell \in L, \tau \in \text{TS}. \tag{82}$$

Each sensor has a target to monitor, therefore each sensor has a packet to forward. Constraint (77) ensures every sensor node must transmit once. Constraints (78) ensure that: if s sent a packet to s' at time τ, then no one can send another packet to s after time τ; it takes care of aggregation. Constraints (79) make sure that at most one configuration can be used in a time slot. Constraints (80) ensure that if a link is used in any time slot, then it is included in one of the scheduled configurations.

As can be observed, the ACC model needs a set of configurations for each time slot τ. But how can we generate this set? The set of all possible configurations is exponentially large. In the next section, we explain how to solve it using a scalable column generation based algorithm.

5.3.2 Solution of the ACC Model

Column Generation and ILP Solution

The ACC model has an exponential number of variables. Therefore, in order to solve it, we use the Column Generation (CG) method for solving *exactly* the ACC using explicitly only a very small subset of variables z_c^τ, see, e.g., [20] if not familiar with the CG method. The Column Generation method allows an optimal solution of the linear relaxation of the ACC model. We used it, combined with a heuristic in order to generate an initial set of variables,

i.e., the heuristic of Wang *et al.* [77]. The implementation of the CG method requires the decomposition of the ACC model into the so-called Restricted Master Problem (RMP), i.e., the ACC model with a very restricted set of variables, and the so-called pricing problem, i.e., a generator of "improving" link transmission configurations. Such configurations, if added to the current RMP, lead to an improved value of the linear relaxation of the current RMP. The column generation method then consists in solving RMP and PP in rounds until we reach the optimal solution of the linear relaxation of the ACC model. Then, considering only the configurations generated in order to reach the optimal linear programming relaxation of the ACC model, we derive an integer solution of the Aggregated ConvergeCast problem.

By doing so, we get an ε-optimal solution for model ACC, with

$$\varepsilon = \frac{\tilde{z}_{\text{ILP}} - z^{\star}_{\text{LP}}}{z^{\star}_{\text{LP}}},$$

where $z^{\star}_{\text{LP}}$ denotes the optimum value of the linear relaxation of ACC and defines a lower bound on the value of the optimum ILP solution ($z^{\star}_{\text{ILP}}$), and $\tilde{z}_{\text{ILP}}$ an upper bound on $z^{\star}_{\text{ILP}}$, even if $\tilde{z}_{\text{ILP}}$ is the optimal ILP value of the last generated RMP.

Pricing Problem - Transmission Configuration Generator

As discussed in the previous paragraph, in the CG solution scheme, we use a configuration (column) generator, called `ConfigurationGenerator`, one for each τ. We now state it.

The objective of the `ConfigurationGenerator` corresponds to the minimization of the reduced cost $\text{REDCOST}^{\tau}_{c}$. Recall (see [20]) that a negative REDCOST indicates that the corresponding configuration (column) can contribute to the improvement of the objective of the ACC model. Let $u^{\tau(80)}_{\ell}$, $v^{\tau(79)}$ be the dual values of Constraints (80), (79) respectively. The expression of the reduced cost can be written:

$$\text{[ConfigGen]} \quad \min_{c \in C} \ \text{REDCOST}^{\tau}_{c} = 1 - \sum_{\ell \in L} u^{\tau(80)}_{\ell} a^{c}_{\ell} - v^{\tau(79)} \tag{83}$$

subject to:

$$\sum_{\ell \in \omega^+(s)} a_\ell + \sum_{\ell \in \omega^-(s)} a_\ell \leq 1 \qquad s \in S \tag{84}$$

$$a_\ell = 0 \qquad \ell \in \omega^+(s_0) \tag{85}$$

$$p_s d_\ell^{-\alpha} - \beta \sum_{s'' \in S' : s'' \neq s} p_{s''} d_{s''s'}^{-\alpha} - L_1 a_\ell \geq \beta N_0 \eta - L_1 \qquad \ell = (s, s') \in L \tag{86}$$

$$p_s \leq P_{\max} \sum_{\ell \in \omega^+(s)} a_\ell \qquad s \in S \tag{87}$$

$$a_\ell \leq L_2 \, p_s \qquad \ell = (s, s') \in L \tag{88}$$

$$a_\ell \in \{0, 1\} \qquad \ell = (s, s') \in L \tag{89}$$

$$p_s \geq 0 \qquad s \in S. \tag{90}$$

where a_ℓ indicates a transmission link between two sensors sensors s and s' using p_s, the transmission power of sensor s, L_1 and L_2 are large positive constants. Constraints (84) say that a node cannot transmit and receive at the same time, whereas constraints (85) prevent the sink from transmitting. Constraints (86) enforce the SINR condition, i.e., (75). Constraints (87) ensure that if there is an outgoing link then that the power of the source link is less the maximum power. Constraints (88) ensure that the transmission link is not active if the power is set to zero, and vice-versa. When none of the [ConfigGen] problems produces a configuration with a negative reduced cost in a linear programming iteration, then we can claim we have reached the optimal linear programming solution of ACC or ACC-MP.

5.3.3 A first improvement: ACC-MP Model

While conducting the numerical experiments with the solution scheme described in Section 5.3.2, we noticed that the lower bound provided by $z^\star_{\text{LP}}$, the optimal value of the linear relaxation of ACC was very weak, and then investigated how to tighten the set of constraints of the ACC model (76) - (82), and consequently improve both the accuracy of the solution and the convergence speed of the solution process. We therefore looked for so-called cuts or valid inequalities.

Indeed, we added new sets of constraints, which calculate explicitly (it was only implicitly done in the basic ACC model) paths from each sensor to the sink node. These last constraints correspond to flow constraints. We call the resulting model ACC-MP.

Observe that, with the addition of cutting-plane constraints, we can avoid the generation of configurations that cannot be part of a feasible Aggregated ConvergeCast schedule. Indeed, the newly added constraints put a priority on completing existing partial paths

(throughout the already generated configurations) from sensors (targets) to sink, whenever there is an opportunity for improving the optimal value of the current RMP while doing so.

The addition of the flow constraints require the introduction of a set of variables $f = (f_{ss'})_{\ell=(s,s')\in L}$ to ensure that each target has a path to the sink node. Model ACC-MP can be written as follows:

$$\text{[ACC-MP]} \qquad \min \sum_{\tau \in TS} \sum_{c \in \mathcal{C}^\tau} z_c^\tau \tag{91}$$

subject to:

$$\sum_{(s's)\in L} f_{s's} + 1 = \sum_{(ss')\in L} f_{ss'} \qquad s \in S \tag{92}$$

$$\sum_{s\in S} f_{ss_0} = n \tag{93}$$

$$\sum_{\tau \in TS} P_{ss'}^\tau \leq f_{ss'} \leq \sum_{\tau \in TS} P_{ss'}^\tau \times n \qquad (s,s') \in L \tag{94}$$

$$\text{Constraints: } (77) - (82)$$

$$f_{ss'} \geq 0 \text{ and integer} \qquad (s,s') \in L. \tag{95}$$

Constraints (92) and (93) ensures that each sensor has a path to the sink node. Constraints (92) guarantee that, at each sensor, the number of incoming paths plus one (i.e., the path from its own target) equals the number of outgoing paths. Constraint (93) makes sure that the sink is the destination of a path originating from each sensor. Constraint (94) ensures that we use link P_ℓ^τ only based on flow variables $f_{ss'}$. The left-hand inequality makes sure that we do not allow P_ℓ^τ to have a non-zero value when there is no flow. The right-hand inequality ensures that whenever there is a flow, P_ℓ^τ is true. Lastly, we use all the constraints of ACC.

5.3.4 A second improvement: ACC-PP Model

While testing the improvement described in the previous section, we found out that, while the lower bound (optimal value of the LP relaxation) was improved, there was still room for further improvement. We therefore next investigated the idea of strengthening the lower bound of ACC-MP model throughout the introduction of cutting-planes in the configuration generator ([ConfigGen-ACC-PP] model). Indeed, while in decomposition models, constraints are usually not repeated in the master and in the pricing problems, it sometimes help to do so. Consequently, we decided to duplicate Constraints (78) in the pricing problem, and this required two new sets of constraints (97) and (98).

The resulting Configuration Generator is called [ConfigGen-ACC-PP] and can be written as follows:

$$[\text{ConfigGen-ACC-PP}] \qquad \min_{c \in C} \ \text{REDCOST}_c^{\tau} \tag{96}$$

subject to:

$$(84) - (90) \text{ as well as}$$

$$(1 - a_\ell) L_1 \geq \sum_{\tau' \in TS, \tau' < \tau} \ \sum_{\ell' \in \omega^+(s)} P_{\ell'}^{\tau'} \qquad \ell = (s, s') \tag{97}$$

$$(1 - a_\ell) L_1 \geq \#C_s - \Big(\sum_{\tau' \in TS, \tau' < \tau} \ \sum_{\ell' \in \omega^+(\text{children}(s))} P_{\ell'}^{\tau'} \Big) \qquad \ell = (s, s') \tag{98}$$

Constraints (97) ensure that links that transmitted in earlier slots are discarded. Constraints (98) ensure that links are considered only when all their children have transmitted. All together, (97) and (98) are equivalent to (78).

We tested the embedding of [ConfigGen-ACC-PP] in both ACC and ACC-MP (restricted) master problems. We found out that qualities of the solutions were very similar, while computational times were higher with the embedding of [ConfigGen-ACC-PP] in ACC-MP. Consequently, we decided to go on only with the so-called ACC-PP model, that combines ACC for the master problem and [ConfigGen-ACC-PP] for the pricing problem.

5.4 Computational Experiments

We now present extensive computational experiments for comparing all three new proposed models ACC, ACC-MP, and ACC-PP for Aggregated ConvergeCast, as well as with the best previously proposed heuristic with an SINR interference model (Wang *et al* [77]).

5.4.1 Data Sets

All experimental results use randomly generated topologies in a square grid of side 625m. We ensure that each topology is connected; if the randomly generated topology does not meet these requirements, we discard it and generate another topology.

Wireless sensor parameters are those used in the literature [44] for outdoor sensor specifications [65]: sensing range is 150m, maximum transmission range is 100m, assuming a data rate of 150kbps, and a maximum transmission power of 13mW. We consider 40, 50, 60, and 70 sensors in each experiment.

Each value in the tables and the figures correspond to an average computed over 10 different randomly generated topologies.

5.4.2 Comparison of ACC, ACC-MP, and ACC-PP Models

Table 17: Schedule - Aggregated ConvergeCast (40, 50 sensors)

	ACC		ACC-MP		ACC-PP		
40sen	LP	ILP	**LP**	ILP	**LP**	ILP	Wang2012
0	**6.8**	18	**6.8**	18	**17.0**	18	30
1	**6.0**	23	**6.0**	23	**22.5**	24	32
2	**10.7**	18	**11.4**	18	**17.0**	19	28
3	**5.2**	20	**5.5**	20	**19.3**	21	29
4	**16.5**	24	**17.5**	24	**23.0**	24	37
5	**5.9**	18	**8.2**	18	**17.0**	19	27
6	**5.9**	18	**5.9**	18	**17.0**	18	30
7	**6.4**	19	**6.7**	19	**18.0**	19	26
8	**6.0**	23	**6.7**	23	**20.8**	22	35
9	**5.1**	19	**5.6**	19	**18.0**	19	32
avg	**7.5**	**20.0**	**8.0**	**20.0**	**19.0**	**20.3**	**30.6**
gap		**168.3**		**149.0**		**7.1**	
	ACC		ACC-MP		ACC-PP		
50sen	LP	ILP	**LP**	ILP	**LP**	ILP	Wang2012
0	**5.9**	22	**6.7**	22	**20.0**	22	39
1	**5.5**	17	**6.3**	17	**16.0**	17	31
2	**5.1**	20	**5.9**	20	**19.0**	20	34
3	**4.6**	20	**5.1**	20	**18.0**	20	35
4	**6.1**	20	**5.7**	20	**19.0**	20	40
5	**6.3**	19	**6.6**	19	**17.0**	19	31
6	**5.4**	20	**7.3**	20	**19.0**	20	36
7	**6.9**	21	**7.0**	21	**21.0**	21	33
8	**5.7**	16	**6.1**	16	**16.0**	16	32
9	**5.9**	25	**6.6**	25	**24.0**	25	39
avg	**5.7**	**20.0**	**6.3**	**20.0**	**18.9**	**20.0**	**35.0**
gap		**248.4**		**216.0**		**5.7**	
gap = 100*(ILP-LP)/LP			ILP=Schedule		LP=Lower Bound		

Table 18: Schedule - Aggregated ConvergeCast (60, 70 sensors)

ACC-PP 60sen				ACC-PP 70sen		
	LP	ILP	Wang2012	LP	ILP	Wang2012
0	**23.0**	25	42.0	**30.0**	32	47.0
1	**20.0**	22	36.0	**23.0**	24	42.0
2	**18.0**	19	40.0	**25.4**	26	43.0
3	**21.0**	22	39.0	**27.0**	27	43.0
4	**18.0**	19	34.0	**27.0**	27	40.0
5	**28.0**	28	40.0	**32.0**	33	42.0
6	**20.0**	21	40.0	**24.0**	24	49.0
7	**25.0**	25	38.0	**26.0**	27	38.0
8	**17.0**	18	36.0	**25.0**	25	38.0
9	**19.0**	21	41.0	**29.0**	30	44.0
avg	**20.9**	**22.0**	**38.6**	26.8	27.5	42.6
gap		**5.2**			2.5	
gap = 100*(ILP-LP)/LP		ILP=Schedule		LP=Lower Bound		

Table 17 and 18 give a comparison of the solutions provided by our ACC, ACC-MP, and ACC-PP models with those output by the algorithm of [77]. We can see that the schedules produced by all our three models are comparable, and all of them obtain substantial improvements (50% to 75%) over the schedule of [77]. However, the integrality gap is very large in ACC. Indeed ACC-MP provides a better lower bound than that of ACC but still is far from the ILP. This improvement can be attributed to the valid inequalities that were added in ACC-MP. Finally, ACC-PP substantially reduces the integrality gap to 7%. The reduction in the integrality gap is shown pictorially in Figure 17 for different network sizes. It can be seen that the gap is further reduced for ACC-PP as the network size increases.

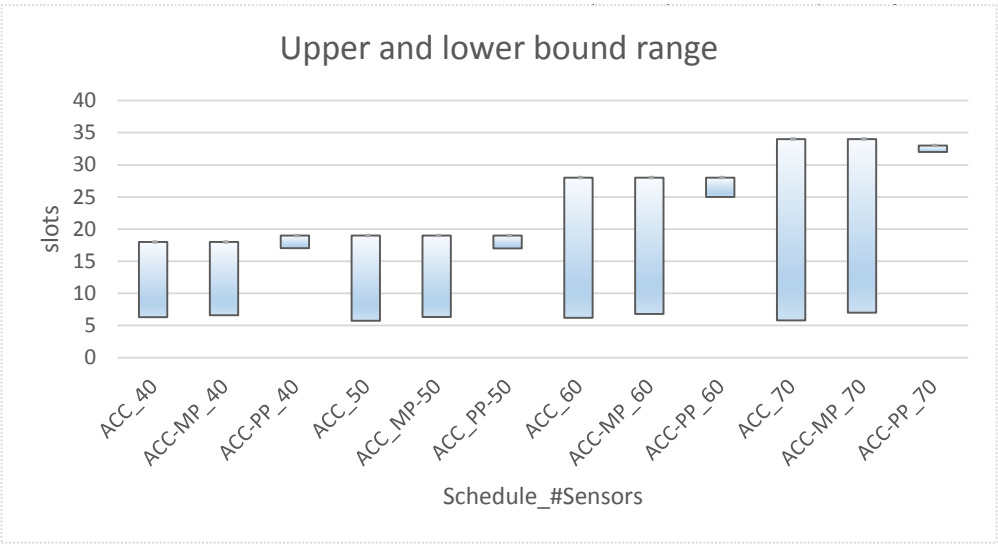


Figure 17: Bounds

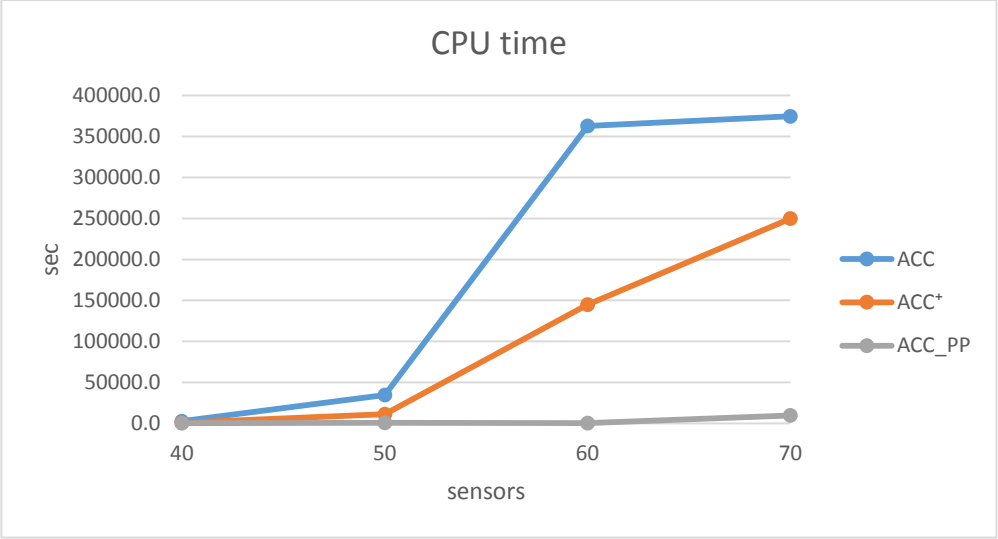


Figure 18: Computational times

In Figure 18, we compare the computational times of ACC, ACC-MP, and ACC-PP models, with the number of sensors between 40 and 70. Surprisingly, ACC-PP is much faster than the other two models, ACC-MP and ACC-PP, and its accuracy is also much better.

5.4.3 Comparison of ACC-PP Algorithm with the Heuristic of Wang *et al.* [77]

We do a detailed comparison of the schedules produced by ACC-PP and [77] on a topology of 70 nodes. The frame length achieved by ACC-PP is 24 slots and that by [77] is 43 slots for this topology. On average, for every 4 frames used by the algorithm of [77], we save one frame using ACC-PP. Note that both algorithms have to schedule exactly 70 links for this topology.

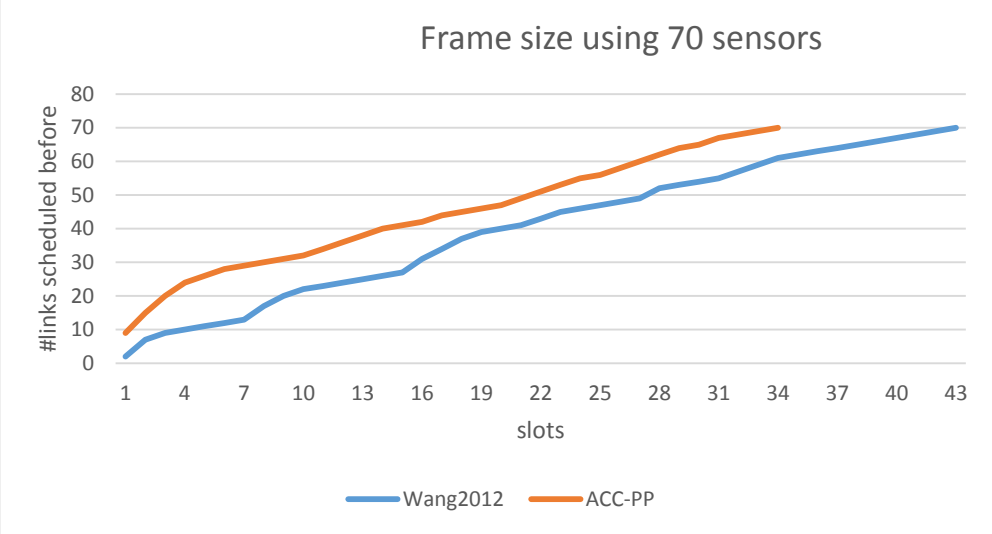
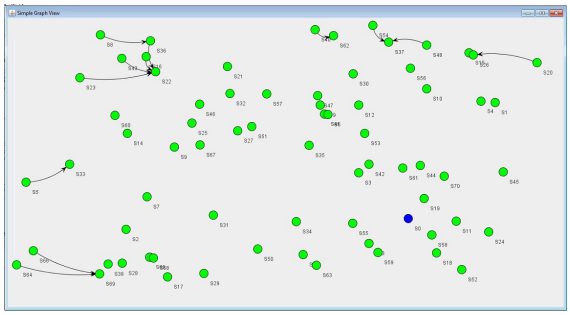
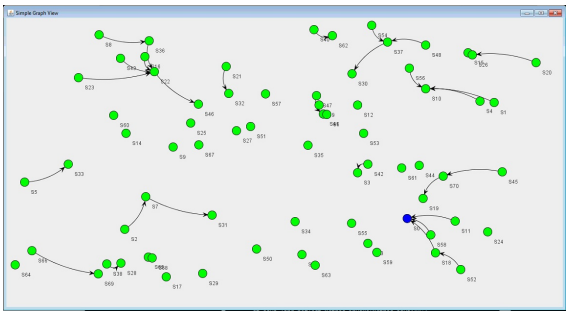


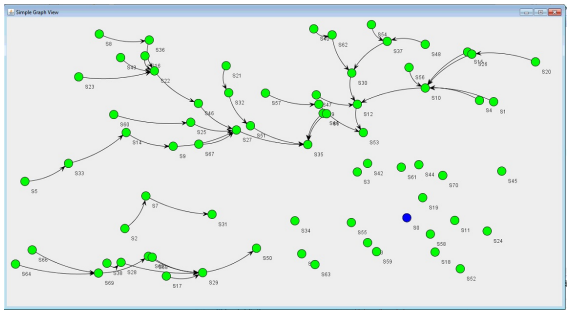
Figure 19: Frame Analysis



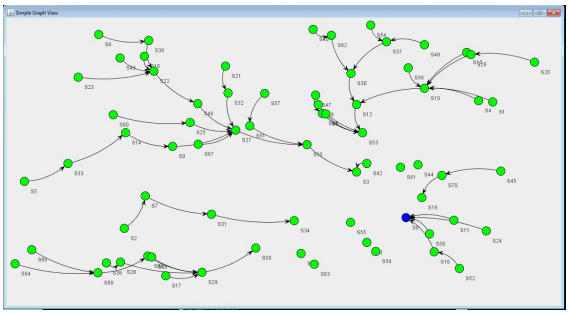
(a) Wang *et al.* (2012) - After 5 slots (12 scheduled links)



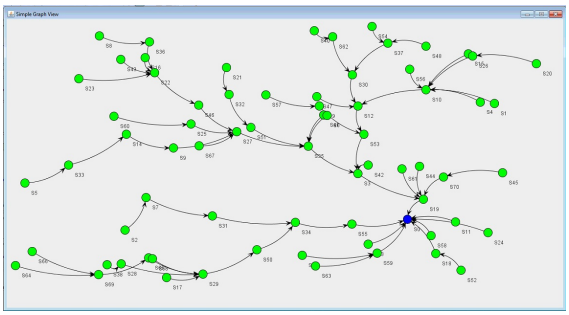
(b) ACC-PP - After 5 slots (28 scheduled links)



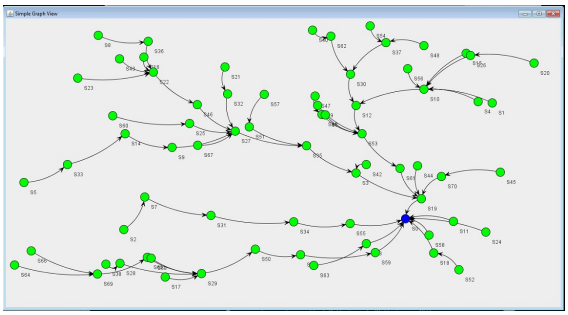
(c) Wang *et al.* (2012) - After 25 slots (48 scheduled links)



(d) ACC-PP - After 25 slots (58 scheduled links)



(e) Wang *et al.* (2012) - Final tree after 43 slots



(f) ACC-PP - Final tree after 34 slots

Figure 20: Aggregated ConvergeCast using Wang2012 and ACC-PP

Figure 20 shows the links that are scheduled using both methods, after 5 slots, and after 25 slots. It can be seen that ACC-PP schedules many more links in earlier slots than [77]. For example, in the first five slots, ACC-PP successfully schedules 28 links compared to 12 of [77]. Figure 19 compares the number of links scheduled before a given slot for both methods. It demonstrates that the number of links scheduled before a given time slot increases at a similar rate for both methods, but the method of Wang *et al* [77] gets a much slower start; this accounts for the much longer frame length.

5.5 Conclusion

We design a first exact model ACC with two enhancements (ACC-MP and ACC-PP models) to improve its scalability, in order to compute an optimal schedule for the NP-hard Aggregated ConvergeCast problem. With all three models, we can solve instances of up to 70 sensors within very reasonable computing times. All three models obtain similar schedules, which are about 50% shorter than the schedules produced by the best heuristic proposed so far with an SINR interference model. In addition, the most efficient model, ACC-MP, outputs ε-optimal solutions with an accuracy of 7%, which is already enough to produce much shorter schedules than the best algorithm of the literature.

Chapter 6

Conclusions and Future Work

In this thesis, we studied the problem of finding a minimum length TDMA frame or schedule to achieve ConvergeCast and Aggregated ConvergeCast. The results of the thesis have been published and/or submitted in [7, 8, 2, 9, 6, 5, 4, 3].

In [7, 8], we studied the problem of finding the minimum number of configurations to forward data to the sink in wireless sensor networks. We designed an optimization model consisting of master and pricing problems, to solve the problem. Since the underlying pricing problem remains NP-hard, we proposed two algorithms that scale up to problems with larger sizes. Both our algorithms outperform the algorithm given in [44] for large topologies, and are therefore more scalable. Furthermore, we describe two realistic scenarios in which each sensor monitoring a target produces multiple packets relating to the target, and described how to modify our algorithms to support such scenarios. We performed a comprehensive analysis of the solutions produced by our algorithms.

Since the solution produced in [44, 8] does not actually give a schedule, in [2, 9], we investigated two basic approaches to scheduling for a ConvergeCast operation in a WSN. In the first approach, called the TC-approach, a multiset of transmission configurations is computed and the scheduling algorithm restricts itself to using these configurations. In the second approach, called the two-phase approach, first a routing tree or subgraph is computed, and next, sets of non-interfering links are scheduled in rounds. For the TC-based approach, we provide new optimal solution using TCs and several new scheduling heuristics. For the two-phase approach, we give a new tree called TFM tree, as well as two scheduling algorithms (ROS and EMWF) for the second phase. Our results show that the tree-based approach using our TFM tree significantly outperforms the TC-approach, unless each node has several packets to send in the same frame, a situation which more closely resembles a data stream.

For the NP-hard problem of ConvergeCast, in [6, 5], we provided a first complete mathematical formulation to get an optimal "true" schedule, i.e., not only the minimum number of required configurations, but also the ordering of the selected configurations. The quality and accuracy of the optimized ConvergeCast solution is improved by about 15 % when compared to the best previously available solution.

In [4, 3], we design a mathematical model to achieve an optimal schedule for the NP-hard Aggregated ConvergeCast problem. We can solve instances of up to 70 sensors within very reasonable computing times. We obtain schedules, which are about 50% shorter than previously proposed schedules.

6.1 Future Work

Our optimal true schedule ConvergeCast formulation and the optimal schedule formulation of the Aggregated ConvergeCast needs to solve sequentially a set of pricing problems in each CG iteration. Each of these pricing problems are independent of each other and only need a set of dual values produced by the restricted master formulation as input. We can take advantage of this property of being independent and instead of sequential execution, we can have parallel execution. This can help us solve larger networks much more efficiently.

We considered a single sink throughout the thesis, as it is the generic scenario for the ConvergeCast operation. It would be interesting to modify the formulation to work with multiple sinks.

In all the experiments in this thesis, we assume the location of the sensors are given. It would be an another interesting problem if one has to also decide the initial location of the sensors along with the ConvergeCast named Optimal Sensor Location problem [69]. This is another interesting research area, that is much more challenging then the addressed NP-hard ConvergeCast problem.

Finally, ConvergeCast has the advantage of having all the readings at the sink, while that of Aggregate ConvergeCast optimizes energy and traffic by only sending aggregated data. If we could formulate a scalable model that has the advantages of both ConvergeCast and Aggregated ConvergeCast using compressed sensing [24] that would be another future direction to work on.

Bibliography

[1] B. Addis, A. Capone, G. Carello, L.G. Gianoli, and B. Sans. Energy management in communication networks: a journey through modeling and optimization glasses. *Computer Communications*, 9192:76 – 94, 2016.

[2] M. Bakshi, B. Jaumard, M. Kaddour, and L. Narayanan. TDMA scheduling in wireless sensor networks. In *IEEE Canadian Conference on Electrical and Computer Engineering - CCECE*, pages 1–6, 2016.

[3] M. Bakshi, B. Jaumard, and L. Narayanan. Optimal aggregated convergecast scheduling. *submitted for publication*, 2017.

[4] M. Bakshi, B. Jaumard, and L. Narayanan. Optimal aggregated convergecast scheduling. *submitted for publication*, 2017.

[5] M. Bakshi, B. Jaumard, and L. Narayanan. True convergecast scheduling in wireless sensor networks. *submitted for publication*, 2017.

[6] M. Bakshi, B. Jaumard, and L. Narayanan. True convergecast scheduling in wireless sensor networks. In *International Conference on Computing, Networking and Communications, ICNC*, pages 627–631, 2017.

[7] M. Bakshi, M. Kaddour, B. Jaumard, and L. Narayanan. An efficient method to minimize TDMA frame length in wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC)*, pages 825 – 830, 2015.

[8] M. Bakshi, M. Kaddour, B. Jaumard, and L. Narayanan. An efficient method to minimize TDMA frame length in wireless sensor networks. *submitted for publication*, 2017.

[9] M. Bakshi, M. Kaddour, B. Jaumard, and L. Narayanan. TDMA scheduling in wireless sensor networks. *submitted for publication*, 2017.

[10] S.A. Borbash and A. Ephremides. Wireless link scheduling with power control and SINR constraints. *IEEE Transactions on Information Theory*, 52:5106–5111, November 2006.

[11] G. Brar, D. M. Blough, and P. Santi. Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks. In *12th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 2–13, 2006.

[12] C. Buratti and R. Verdone. Joint routing and scheduling for centralised wireless sensor networks. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6, 2016.

[13] M. Cao, V. Raghunathan, S. Hanly, V. Sharma, and P. R. Kumar. Power control and transmission scheduling for network utility maximization in wireless networks. In *IEEE Conference on Decision and Control*, pages 5215–5221, 2007.

[14] A. Capone and G. Carello. Scheduling optimization in wireless mesh networks with power control and rate adaptation. In *IEEE International Conference on Sensing and Communication and Networking (SECON)*, pages 138–147, 2006.

[15] A. Capone, G. Carello, I. Filippini, S. Gualandi, and F. Malucelli. Routing, scheduling and channel assignment in wireless mesh networks: Optimization models and algorithms. *Ad Hoc Networks*, 8(6):545–563, 2010.

[16] A. Capone, M. Pioro, Y. Li, and D. Yuan. On packet transmission scheduling for min-max delay and energy consumption in wireless mesh sensor networks. *Electronic Notes in Discrete Mathematics*, 52:61 – 68, 2016. INOC 2015 7th International Network Optimization Conference.

[17] S. Chae, K. Kang, and Y. Cho. A scalable joint routing and scheduling scheme for large-scale wireless sensor networks. *Ad Hoc Networks*, pages 427–441, 2013.

[18] X. Chen, X. Hu, and J. Zhu. Minimum data aggregation time problem in wireless sensor networks. In *Mobile Ad-hoc and Sensor Networks*, volume 3794 of *Lecture Notes in Computer Science*, pages 133–142. 2005.

[19] I. Chlamtac and S. Kutten. A spatial reuse TDMA/FDMA for mobile multi-hop radio networks. In *IEEE INFOCOM 1985*, pages 389–394, 1985.

[20] V. Chvatal. *Linear Programming*. Freeman, 1983.

[21] D.G. Costa, I. Silva, L.A. Guedes, F. Vasques, and P. Portugal. Availability issues in wireless visual sensor networks. *Sensors*, 14(2):2795–2821, 2014.

[22] G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors. *Column Generation*. GERAD 25th Anniversary Series. Springer, 2005.

[23] S. Ravela E. A. Basha and D. Rus. Model-based monitoring for early warning flood detection. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, pages 295–308, New York, USA, 2008. ACM.

[24] D. Ebrahimi, S. Sebbah, and C. Assi. A column generation method for constructing and scheduling multiple forwarding trees in wireless sensor networks. *IEEE Transactions on Wireless Communications*, pages 6513–6523, 2016.

[25] J. El-Najjar, C. Assi, and B. Jaumard. Joint routing and scheduling in WiMAX-based mesh networks: A column generation approach. *IEEE Transactions in Wireless Communications*, 9:2371 – 2381, 2010.

[26] J. El-Najjar, C. Assi, and B. Jaumard. Cross-layer optimization for the coding-aware routing and scheduling problem in WiMAX-based mesh networks. *Wireless Communications and Mobile Computing*, 5:525–538, 2013.

[27] T. ElBatt and A. Ephremides. Joint scheduling and power control for wireless ad hoc networks. *IEEE Transactions on Wireless Communications*, 3:74–85, 2004.

[28] A. Erzin and A. Pyatkin. Convergecast scheduling problem in case of given aggregation tree: The complexity status and some special cases. In *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–6, July 2016.

[29] J. Gagnon and L. Narayanan. Efficient scheduling for minimum latency aggregation in wireless sensor networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1024–1029, 2015.

[30] R. Gandhi, M. Halldórsson., C. Konrad, G. Kortsarz, and H. Oh. *Radio Aggregation Scheduling*, pages 169–182. Springer International Publishing, Cham, 2015.

[31] R. Gatti, S. S. Kumar, Shivashankar, K. N. S. Kumar, and P. R. Prasad. Improvement of speed in data collection rate in tree based wireless sensor network. In *IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 720–723, 2016.

[32] D. Gong and Y. Yang. Low-latency SINR-based data gathering in wireless sensor networks. In *IEEE Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*, pages 1941–1949, April 2013.

[33] O. Goussevskaia, Y.-A. Pignolet, and R. Wattenhofer. Efficiency of wireless networks: Approximation algorithms for the physical interference model. *Foundations and Trends in Networking*, 4:313–420, March 2010.

[34] V. C. Gungor and G. P. Hancke. *Industrial Wireless Sensor Networks: Applications, Protocols, and Standards*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2013.

[35] L. Guo, Y. Li, and Z. Cai. Minimum-latency aggregation scheduling in wireless sensor network. *Journal of Combinatorial Optimization*, pages 279–310, 2016.

[36] M. Halldórsson and P. Mitra. Wireless capacity with oblivious power in general metrics. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1538–1548, 2011.

[37] M. Halldórsson and P. Mitra. Wireless connectivity and capacity. In *Symposium on Discrete Algorithms (SODA)*, pages 516–526, 2012.

[38] R. Huang, W. Z. Song, and M. Xu et al. Real-world sensor network for long-term volcano monitoring: Design and findings. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):321–329, 2012.

[39] U. Hunkeler, C. Lombriser, H. L. Truong, and B. Weiss. A case for centrally controlled wireless sensor networks. *Computer Networks*, 57:1425 – 1442, 2013.

[40] O. Durmaz Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi. Fast data collection in tree-based wireless sensor networks. *IEEE Transactions on Mobile Computing*, pages 86–99, 2012.

[41] M. Jakob and I. Nikolaidis. A top-down aggregation convergecast schedule construction. In *IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 17–24, July 2016.

[42] B. Jaumard, T. M. Murillo, and S. Sebbah. Scheduling vs. pseudo-scheduling models in IEEE 802.16j wireless relay networks. In *26th Biennial Symposium on Communications (QBSC)*, pages 101–106, 2012.

[43] M. Kaddour. A column generation approach to maximize capacity of multi-rate power controlled TDMA wireless sensor networks. In *Ad-hoc, Mobile, and Wireless Network*, volume 7960 of *Lecture Notes in Computer Science*, pages 233–244, 2013.

[44] M. Kaddour. Optimizing the throughput-lifetime tradeoff in wireless sensor networks with link scheduling, rate adaptation, and power control. *Wireless Communications and Mobile Computing*, 16(12):1510–1525, 2016.

[45] E. Karami and S. Glisic. Joint optimization of scheduling and routing in multicast wireless ad hoc networks using soft graph coloring and nonlinear cubic games. *IEEE Transactions on Vehicular Technology*, pages 3350–3360, 2011.

[46] T. Kellner. Sam likes it hot: This intrepid explorer just connected nicaraguas most active volcano to the internet. *GE Reports*, 2017. [Online; accessed 29-March-2017].

[47] T Kesselheim. A constant-factor approximation for wireless capacity maximization with power control in the SINR model. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 1549–1559, 2011.

[48] T. Kesselheim. Approximation algorithms for wireless link scheduling with flexible data rates. In *European Conference on Algorithms (ESA)*, pages 659–670, 2012.

[49] A. Kesselman and D. R. Kowalski. Fast distributed algorithm for convergecast in ad hoc geometric radio networks. *Parallel and Distributed Computing*, pages 578 – 585, 2006.

[50] Y. Kim, J. Kang, D. Kim, E. Kim, P. K. Chong, and S. Seo. Design of a fence surveillance system based on wireless sensor networks. In *Proceedings of the International Conference on Autonomic Computing and Communication Systems*, pages 4:1–4:7, Brussels, Belgium, 2008. ICST.

[51] S. Kompella, J.E. Wieselthier, A. Ephermides, H.D. Sherali, and G.D. Nguyen. On optimal SINR-based scheduling in multihop wireless networks. *IEEE/ACM Transactions on Networking*, 18(6):1713–1724, December 2010.

[52] H.-J. Korber, H. Wattar, and G. Scholl. Modular wireless real-time sensor/actuator network for factory automation applications. *Industrial Informatics, IEEE Transactions on*, 3(2):111–119, May 2007.

[53] V. S. Kumar and V. Sharma. Joint routing, scheduling and power control providing QoS for wireless multihop networks. In *National Conference on Communications (NCC)*, pages 1–6, 2015.

[54] S. Lanzisera, A.M. Mehta, and K.S.J. Pister. Reducing average power in wireless sensor networks through data rate adaptation. In *Proceedings of the IEEE International Conference on Communications*, pages 480–485, 2009.

[55] R. Lara, D. Bentez, and A. Caamao et al. On real-time performance evaluation of volcano-monitoring systems with wireless sensor networks. *IEEE Sensors Journal*, 15(6):3514–3523, 2015.

[56] L.S. Lasdon. *Optimization Theory for Large Systems*. MacMillan, New York, 1970.

[57] H. Li, Q. S. Hua, C. Wu, and F. C. M. Lau. Minimum-latency aggregation scheduling in wireless sensor networks under physical interference model. In *ACM Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, pages 360–367, 2010.

[58] J. Li, X. Guo, and L. Guo. Joint routing, scheduling and channel assignment in multi-power multi-radio wireless sensor networks. In *IEEE International Performance Computing and Communications Conference*, pages 1–8, 2011.

[59] X. Y. Li, X. Xu, S. Wang, S. Tang, D. GuoJun, Z. JiZhong, and Q. Yong. Efficient data aggregation in multi-hop wireless sensor networks under physical interference model. In *IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 353–362, 2009.

[60] Y. Li, A. Capone, and D. Yuan. On end-to-end delay minimization in wireless networks under the physical interference model. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 2020–2028, 2015.

[61] F. Liu, C. Y. Tsui, and Y. J. Zhang. Joint routing and sleep scheduling for lifetime maximization of wireless sensor networks. *IEEE Transactions on Wireless Communications*, pages 2258–2267, 2010.

[62] J. Luo, C. Rosenberg, and A. Girard. Engineering wireless mesh networks: Joint scheduling, routing, power control, and rate adaptation. *IEEE/ACM Transactions on Networking*, 18(5):1387 – 1400, October 2010.

[63] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. In *Symposium on Operating Systems Design and Implementation (OSD)*, pages 131–146, Boston, MA, USA, 2002.

[64] T. Moscibroda and R. Wattenhofer. The complexity of connectivity in wireless networks. In *INFOCOM*, pages 1–13, 2006.

[65] M. Mridula and RN. Shukla. Current wireless sensor nodes (Motes): Performance metrics and constraints. *International Journal of Advanced Research in Electronics and Communication Engineering*, pages 45–48, 2013.

[66] C. Pan and H. Zhang. A time efficient aggregation convergecast scheduling algorithm for wireless sensor networks. *Wireless Networks*, 22(7), 2016.

[67] S.U. Pillai, T. Suel, and S. Cha. The Perron-Frobenius theorem: some of its applications. *IEEE Signal Processing Magazine*, 22(2):62–75, March 2005.

[68] S. Santini, B. Ostermaier, and A. Vitaletti. First experiences using wireless sensor networks for noise pollution monitoring. In *Proceedings of the Workshop on Real-world Wireless Sensor Networks*, pages 61–65, New York, USA, 2008. ACM.

[69] P. Sen, K. Sen, and U. M. Diwekar. A multi-objective optimization approach to optimal sensor location problem in {IGCC} power plants. *Applied Energy*, 181:527 – 539, 2016.

[70] L. Shi and A. O. Fapojuwo. Tdma scheduling with optimized energy efficiency and minimum delay in clustered wireless sensor networks. *IEEE Transactions on Mobile Computing*, pages 927–940, 2010.

[71] Y. Shi, Y. T. Hou, J. Liu, and S. Kompella. Bridging the gap between protocol and physical models for wireless networks. *IEEE Transactions on Mobile Computing*, 12(7):1404–1416, 2013.

[72] Y. Shi, Y.T. Hou, J. Liu, and S. Kompella. Bridging the gap between protocol and physical models for wireless networks. *IEEE Transactions on Mobile Computing*, 12:1404–1416, 2013.

[73] F. Stajano, N. Hoult, I. Wassell, P. Bennett, C. Middleton, and K. Soga. Smart bridges, smart tunnels: Transforming wireless sensor networks from research prototypes into robust engineering infrastructure. *Ad Hoc Networks*, 8(8):872–888, November 2010.

[74] R. A. Swartz, J. P. Lynch, S. Zerbst, B. Sweetman, and R. Rolfes. Structural monitoring of wind turbines using wireless sensor networks. *Smart Structures and Systems*, pages 183–196, 2010.

[75] J. Tang, G. Xue, C. Chandler, and W. Zhang. Link scheduling with power control for throughput enhancement in multihop wireless networks. *IEEE Transactions on Vehicular Technology*, 55:733–742, May 2006.

[76] M.F. Uddin and C. Assi. Joint routing and scheduling in wmns with variable-width spectrum allocation. *IEEE Transactions on Mobile Computing*, 12:2178–2192, Nov 2013.

[77] G. Wang, Q. Hua, and Y. Wang. Minimum latency aggregation scheduling for arbitrary tree topologies under the sinr model. In *Ad-hoc, Mobile, and Wireless Networks*, pages 139–152, 2012.

[78] H. Wang and W. Jia. An optimized scheduling scheme in ofdma wimax networks. *International Journal of Communication Systems*, 23(1):23–39, 2010.

[79] W. Wang, B. Peng, H. Wang, and H. Sharif. Study of an energy efficient multi rate scheme for wireless sensor network mac protocol. In *ACM International Workshop on Quality of Service &Amp; Security for Wireless and Mobile Networks*, Q2SWinet '06, pages 51–54, New York, NY, USA, 2006. ACM.

[80] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the Symposium on Operating Systems Design and Implementation*, pages 381–396, Berkeley, CA, USA, 2006.

[81] X. Xu, S. Wang, X. Mao, S. Tang, and X. Y. Li. An improved approximation algorithm for data aggregation in multi-hop wireless sensor networks. In *ACM workshop on Foundations of wireless ad hoc and sensor networking and computing*, pages 47–56, 2009.

[82] H. Yousefi, M. Malekimajd, M. Ashouri, and A. Movaghar. Fast aggregation scheduling in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 14(6), June 2015.