

REAL-TIME WIRELESS SENSOR NETWORKS

A Dissertation Presented to the
Faculty of the School of Engineering and Applied Science
University of Virginia



In Partial Fulfillment of the Requirements for the Degree
of
Doctor of Philosophy (Computer Science)

by

KUMAR SHASHI PRABH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF VIRGINIA

· MAY 2007 ·

© Copyright by
K. Shashi Prabh
All rights reserved
May 2007

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy (Computer Science)

K. Shashi Prabh

This dissertation has been read and approved by the examining committee:

Tarek Abdelzaher (Advisor)

John Stankovic (Chairman)

David Evans (Member)

Sang Son (Member)

Gang Tao (Minor Representative)

Accepted for the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

May 2007

ABSTRACT

This dissertation studies real-time application support in wireless ad-hoc and sensor networks. Real-time applications are performance critical applications that require bounded service latency. In multi-hop wireless ad-hoc and sensor networks, communication delays are dominant over processing delays. Therefore, to enable real-time applications in such networks, the communication latency must be bounded. The shared nature of the communication medium makes the delay characteristics of the medium access control (MAC) protocol in use very important. Furthermore, it is desirable that the MAC protocols for such networks be distributed and be able to spatially reuse the communication channel for scalability and efficiency.

In this dissertation, we derive expressions of real-time capacity that characterize the ability of a network to deliver data on time as well as develop network protocols that achieve this capacity. We introduce a hexagonal network topology based architecture for wireless ad-hoc and sensor networks for real-time applications. We present addressing and constant time routing protocols for the hexagonal network. We develop two distributed spatial-reuse time domain multiplexed MAC protocols with provable real-time properties for convergecast traffic. One protocol constructs network-wide transmission schedule and gives equal bandwidth to all the nodes. This protocol has zero scheduling message overhead and is optimal in the sense that the base-station does not idle. The other protocol supports a more general (unequal bandwidth to nodes) mixture of real and non-real time traffic. Clock synchronization is achieved implicitly. Real-time capacity expressions are obtained and analyzed for the earliest deadline first, deadline monotonic and these two scheduling algorithms.

Data gathering (many-to-one) and data dissemination (one-to-many) are two of the three canonical traffic patterns in WSN. Unlike data gathering at base-stations, transmission scheduling for data dissemination is an easy problem and hence is not as much an issue as is the minimization of number of transmissions for interference suppression and energy conservation. We present and prove the properties of an optimal multicast tree in WSN, which is cast as a generalized Steiner Tree Problem. We present a distributed heuristic that constructs and maintains near-optimal multicast tree on-line.

Advisor: Tarek Abdelzaher

ACKNOWLEDGMENTS

From the application for admission, to courses, exams, research, and finally with this dissertation, a six year long marathon ends. The credit for making this possible goes to several people. First of all, I am very grateful to my advisor Prof. Tarek Abdelzaher for giving me the opportunity to do research on problems that interest me. Without his help and support, this dissertation would not have been possible. The committee members Prof. John Stankovic, Prof. Sang Son, Prof. David Evans and Prof. Gang Tao sifted through the dissertation manuscript. Their effort has added to the clarity of presentation significantly. I thank them for their time and patience.

For the last five years I have been a part of the Control Research Group. I am grateful to the members of the group, especially my classmates, for their support. My special thanks to Ms. Brenda Perkins who helped me with official matters promptly and beyond the call of duty.

In the past six years, I had my fair share of ups and downs. I am, and shall forever remain, indebted to my parents, family and friends, who shared my sorrows and let me share their joys. According to the Indian philosophy, wife and husband are the two parts of the same (one) being. Therefore, I skip thanking her!

CONTENTS

List of Figures	xi
List of Tables	xiv
Chapter 1. Introduction	1
1.1. Real-Time Wireless Sensor Networks	1
1.2. Real-Time Capacity Expressions	2
1.3. Data Dissemination in WSN	3
1.4. Related Work	4
1.5. Thesis Statement and Dissertation Contributions	9
1.6. Dissertation Outline	10
Chapter 2. EDF Scheduling in a Pipeline	12
2.1. Introduction	12
2.2. Problem Formulation and Assumptions	13
2.3. Analytical Framework	14
2.4. Constant Time Utilization Bounds	29
2.5. Evaluation	33
2.6. Summary	36
Chapter 3. Real-Time Capacity Limits	38
3.1. Introduction	38
3.2. Problem Formulation	39
3.3. Assumptions	40

Contents	ix
3.4. Feasibility Conditions	40
3.5. Capacity Limits	44
3.6. Evaluation	50
3.7. Limitations	54
3.8. Summary	54
Chapter 4. Hexagonal Wireless Ad-Hoc Networks	55
4.1. Introduction	55
4.2. Related Work On Hexagonal Topology Networks	57
4.3. System Model	58
4.4. Assumptions	61
4.5. Scheduling for Convergecast	61
4.6. Evaluation	72
4.7. Discussion and Scheduling Implications	76
4.8. Limitations	77
4.9. Summary	78
Chapter 5. Scheduling General Traffic in Hexagonal Networks	79
5.1. Introduction	79
5.2. Problem Formulation	82
5.3. Assumptions	83
5.4. Transmission Scheduling Algorithm	83
5.5. Node Bandwidth Allocation and Schedulability Analysis	89
5.6. Real-Time Capacity	92
5.7. Evaluation	93
5.8. Limitations	96
5.9. Summary	100

Contents	x
Chapter 6. Data Dissemination in Wireless Sensor Networks	101
6.1. Introduction	101
6.2. Problem Formulation and Service Model	102
6.3. Assumptions	104
6.4. Properties of Minimum Steiner Data Caching Tree	107
6.5. Dynamic Cache Placement Heuristic	117
6.6. Evaluation	122
6.7. Steiner Minimal Trees and MSDCT	132
6.8. Real-time Guarantees of SDCT	133
6.9. Limitations	134
6.10. Summary	135
Chapter 7. Summary and Outlook	136
7.1. Summary	136
7.2. Future Work	138
7.3. Conclusions	139
Appendix A: Steiner Point in Cartesian Coordinates	140
Bibliography	143

LIST OF FIGURES

2.1	Waiting time	16
2.2	Maximizing preemption delay	18
2.3	Preemption delay	19
2.4	A busy period utilization curve	26
2.5	Load for J_n and J_{n-1}	27
2.6	Non-optimality of EDF for pipelines.	34
2.7	System utilization vs. task utilization	35
2.8	System utilization vs. pipeline size	36
3.1	Real-time capacity for load balanced traffic	45
3.2	Convergecast traffic	46
3.3	Real-time capacity for convergecast	48
3.4	Effect of radio radius, 800 nodes	51
3.5	Effect of radio radius, 1600 nodes	51
3.6	Effect of the number of sinks, 800 nodes	52
3.7	Effect of the number of sinks, 1600 nodes	52
3.8	Miss rate as a function of real-time capacity demand	53
4.1	Interference in hexagonal networks	59
4.2	Hextants	60
4.3	Oblique coordinate system for hexagons	62

List of Figures		xii
4.4	Routing in hexagonal mesh	64
4.5	Example: Addressing, routing and partitions of a 3-hop network	67
4.6	Cumulative distribution of transmission separation	73
4.7	Percentage packet collision in the presence of radio irregularity	74
4.8	Convergence of clocks	75
4.9	Exponential convergence of clocks	75
5.1	An example bandwidth allocation of the partitions	85
5.2	Token rotation order	87
5.3	State diagram of tokens	88
5.4	State diagram of nodes	88
5.5	Schedulable utilization when the deadlines are the same as periods	94
5.6	Schedulable utilization when the deadlines are larger than the periods	95
5.7	Schedulable utilization when the deadlines are smaller than the periods	96
5.8	Real-time capacity when the deadlines are the same as periods	97
5.9	Real-time capacity when the deadlines are larger than the periods	98
5.10	Real-time capacity when the deadlines are smaller than the periods	99
5.11	Real-time capacity as function of network size	100
6.1	Middleware Architecture	106
6.2	Refresh rates	107
6.3	Construction to prove internal angle symmetry	110
6.4	Construction to obtain angle expressions	113
6.5	Construction to prove binariness of MSDCT	116

List of Figures		xiii
6.6	Adding a new subscriber	118
6.7	Pseudo-code for processing join() requests	120
6.8	Pseudo-code for processing join() replies	121
6.9	Performance ratio of the SDCT heuristic vs. number of subscribers	125
6.10	Histograms of performance ratio of the SDCT heuristic	125
6.11	Message cost vs. number of subscribers	127
6.12	Message cost vs. clustering	128
6.13	Message cost vs. dynamics	129
6.14	Message cost vs. placement irregularity	131
6.15	Message cost vs. placement irregularity	132
7.1	Expressing the Steiner point in Cartesian coordinates	140

LIST OF TABLES

4.1	Transformation rules	63
6.1	Only local shiftings are needed to maintain near-optimality	122
6.2	Partitioning of the network	130

CHAPTER 1

Introduction

Wireless Sensor Networks (WSN) are self-organized ad-hoc networks whose nodes are capable of sensing, gathering, processing and communicating data, especially the data pertaining to the physical medium in which they are embedded. It is envisioned that a typical WSN will consist of a large number of nodes [51, 63, 6]. This enables sensing and actuation at a fine grained level, both spatially and temporally. Though significant on their own, wireless sensor networks play a central role in achieving the goal of truly ubiquitous computing and smart environments. By interfacing the data collection and dissemination infrastructure of WSN with external networks and devices, control and automation of physical entities like houses, factories, farms and so on can be achieved at a level that was not possible before.

1.1 REAL-TIME WIRELESS SENSOR NETWORKS

The performance-critical applications that require bounded delay latency are referred to as real-time applications. Those wireless sensor networks that are capable of providing bounded delay guarantees on packet delivery are referred to as real-time wireless sensor networks. A vast majority of WSN applications are real-time. Design and analysis of real-time WSN are the focus of this dissertation. In the following, we describe a few representative applications.

Consider a wireless sensor network whose nodes report to a base-station whenever the presence or movement of humans, animals, vehicles or some other kind of object is detected. In the case of moving object, it is also desired to monitor the movements of the object. Upon receiving the data from the network, the base-station takes some actions. The actions may be to turn on a camera, alert guards or something else

pre-specified. If the base-station gets detection messages after the object has moved out of the area where the reports originated, the camera won't be able to take the picture of the monitored object. In the case of intrusion, failing of timely tracking will let the intruder escape. Application of WSN to monitor the health of senior citizens and patients is an actively pursued area of research. Typically, the nodes of the network report to a collector node, which is connected to the health-care provider through an external network. The health-care provider's server monitors the WSN data for alarms, and alerts health care personnel upon alarm detection. The network would be of little use if upon emergencies, doctors do not get alerted soon enough to save patients' lives.

A multitude of problems needs to be solved to achieve the goal of supporting real-time applications in WSN. As mentioned earlier, the network uses shared (wireless) medium for communication. Therefore, one needs a distributed medium access control (MAC) protocol that is capable of providing guaranteed bandwidth over multiple hops. Bounded latency guarantee of end-to-end packet delivery is a necessary first step. The real-time support must be achieved at low power and low message overhead to limit interference and conserve energy. Since the network is supposed to consist of a large number of nodes, scalability of the solutions is important.

1.2 REAL-TIME CAPACITY EXPRESSIONS

Traditionally, *network capacity* has referred to the measure of data carrying capability of a network (see [32, 27, 4, 30]). We define *real-time capacity* of a network to be its information carrying ability for given deadlines, where only those information bits that arrive at their destination within the specified deadline count. For WSN the unit of data is a packet rather than bits. If a packet does not reach its destination by the given deadline, then its contribution to the real-time capacity is 0. This is in

contrast to the capacity defined in [32] and other research cited above where every transfer of bit counts.

Analytic expressions for real-time capacity facilitate the process of designing a network that is guaranteed to meet specified throughput and delay requirements. The expressions describe values of a set of variables that will enable the network to meet anticipated real-time requirements. In other words, they define the feasibility region in the space of such variables. In the event of dynamically changing network, which is expected of WSN, besides planning and designing, the feasibility region allows optimization of the operation of the network.

1.3 DATA DISSEMINATION IN WSN

Data gathering and data dissemination are the two key functionalities required of WSNs. For data dissemination, capacity is not as important an issue as interference and energy consumption. Due to the broadcast nature of the wireless medium, data dissemination is trivial in uni-hop wireless networks. This, however, is not the case with multi-hop wireless networks. Employing flooding-like algorithms is wasteful in terms of energy consumption. Furthermore, since the execution pattern and termination of such protocols are not deterministic, their co-existence with real-time applications without affecting the latter is not possible. The problem of data dissemination over multiple hops in energy-efficient manner is not straightforward.

Multicast is an appropriate method to disseminate data to a set of subscribers. In multicast, multiple transmissions of the same packet along a given path is suppressed, although packets are transmitted to each subscriber at the same rate. Asynchronous multicast improves the energy efficiency of multicast by disseminating packets at the desired rate of the individual subscribers. When the choice of inclusion of additional intermediate nodes to construct a minimum cost tree exists, the problem of constructing such a tree becomes the Steiner tree problem. Since there

is a considerable freedom in selection of nodes to construct the asynchronous multicast tree, an optimal solution can be obtained by constructing a Steiner tree. The construction of minimum Steiner tree, however, belongs to the class of NP-Hard problems. Therefore, a distributed *heuristic* that constructs near-optimal asynchronous multicast tree is useful for energy-efficient data dissemination.

1.4 RELATED WORK

1.4.1 Performance Guarantees

Enabling performance guarantees in WSN has been a very active area of research. Providing latency bounds for many-to-one (or, convergecast) traffic topology is the most difficult. It has been observed that even under very light load, a significant percentage of packet loss occurs within first few hops from the base station if CSMA based MAC protocols are used [5]. This is referred to as the funneling effect. A number of MAC protocols have been developed that attempt to decrease the latency of packet deliveries [54, 25, 56]. TRMA [54] is energy efficient CSMA and TDMA based MAC protocol. It uses special protocols to (Neighbor and Schedule Exchange Protocols) exchange two hop neighborhood and their schedule information among nodes. These message exchanges take place in CSMA mode. Using these messages, a somewhat complex algorithm (Adaptive Election Algorithm) constructs the schedule. A distributed link scheduling algorithm is presented in [25]. Z-MAC [56] operates in CSMA mode under low load conditions and switches to TDMA mode under high load conditions. The cost of time synchronization in Z-MAC is high. Motivated by the scalability problems of the existing TDMA MAC protocols and designed for many-to-one traffic, FUNNELING-MAC [5] employs a hybrid CSMA and TDMA scheme, where the TDMA scheduling is done in a small neighborhood of the sink only. None of the above mentioned solutions give latency guarantees.

Caccamo and Zhang have proposed Implicit-EDF for conflict free prioritized transmission in WSN [14]. The algorithm arbitrates access at the MAC layer. The authors propose to divide the sensor network in cellular network like (honeycomb) regions, where each cell operates at a different frequency than all of its six neighbors. Within each cell, nodes coordinate to achieve a contention free schedule. The solution provides real-time guarantees but the communication is assumed to be uni-hop. Unfortunately, the solution is not scalable over multiple hops, and hence has limited applicability for WSN which are inherently multi-hop.

1.4.2 Capacity

The first results for the capacity of ad-hoc wireless networks were obtained by Gupta and Kumar [32]. The authors showed that in a network of n nodes spread over unit area, where each node is capable of transmitting W bits per second, the per node *throughput capacity* scales as $\Theta(W/\sqrt{n \log n})$ bits/sec for randomly chosen source and destination, and as $\Theta(W/\sqrt{n})$ bits/sec for optimally assigned traffic pattern. The authors also showed that under optimal conditions, the network's *transport capacity* scales as $\Theta(W\sqrt{An})$ bit-meters per second, where A is the area of the network. Grossglauser and Tse incorporated mobility of the nodes to the network model of [32], and showed that at the cost of unbounded delay, the throughput capacity of the network with mobile nodes can be $\Theta(n)$ bits/sec [30]. This is achieved by one-hop relaying of the packets. Li *et al.* showed that the throughput capacity of the network can be $\Theta(n)$ bits/sec, if the traffic is *local* [45].

Gastpar and Vetterli [27] obtain capacity expressions using *network coding* [4], where all the other nodes (than the sender-receiver pair) cooperate in transmission. It is assumed that only one pair of nodes transmit/receive at any given time. Using this model, the authors obtain a throughput capacity of $O(\log n)$ bits/second. Jain *et al.* present a linear programming based formulation of the capacity problem [36].

Kodialam and Nandagopal present a polynomial time algorithm that achieves better than 67% of the optimal throughput for a wireless network of specified configuration [42]. The algorithm can compute whether a desired throughput rate between a given pair of end nodes is feasible. For feasible flows, it computes routes and schedules. The model does not account for interference due to transmissions from neighboring nodes. Kozat and Tassiulas [44] and Liu *et al.* [46] independently propose addition of limited infra-structural support to the ad-hoc network to improve capacity.

All of the above mentioned capacity results do not take the delays incurred by individual data bits or packets into account. Our problem is different in that our goal is to compute the capacity of the network for specified deadlines – arrival of packets after deadlines do not contribute to the capacity. The above mentioned work relates to our problem as a limiting case when the deadlines are so large that the packets never miss their deadlines.

1.4.3 Real-Time Capacity

Abdelzaher *et al.* presented the first results on the real-time capacity limits of wireless sensor networks [1]. Expressions for the real-time capacity limits are obtained using sufficient conditions on schedulability. A class of time independent fixed priority scheduling algorithms are considered. Load-balanced and converge-cast – the two extreme traffic topologies are considered. In this dissertation, we present the results and extend them for the EDF scheduling algorithm.

Yan *et al.* propose an energy-efficient service for providing differentiated surveillance by varying the degree of sensing coverage [67]. Cao *et al.* present a stochastic performance analysis of WSN for surveillance applications based on network parameters [15]. Ting Yan presented schedulability analysis of periodic flows in

WSN [66]. Using the results presented in the dissertation, design time analysis can be performed if the flows are known in advance.

Schmitt and Roedig use and extend the Network Calculus [13] to formulate the tradeoff between the buffer size requirements and power consumption with the worst case delay [59]. A suitable arrival curve for WSN is presented. Koubaa *et al.* extend the Sensor Network Calculus formulation of Schmitt and Roedig for a cluster based tree topology and IEEE 802.15.4/ZigBee MAC protocol [43]. The authors conclude that the bandwidth requirement grows exponentially with the depth of the tree. Their conclusion is based on the incorrect assumption that number of clusters grow exponentially with tree depth. The growth is in-fact quadratic in tree depth. To see this, the number of clusters at depth n is approximated by the ratio of the area contained within the ring of radii nR and $(n - 1)R$ and the area contained within one cluster πR^2 . Thus, the number of clusters at depth n is approximately $2n$. Therefore, the total number of clusters in the tree up-to depth n is $\sum_{i=1}^n 2i = n^2 + n$. Furthermore, the delay bound obtained is based on the Concatenation Theorem of Network Calculus. While the theorem has been proved for wired networks, it does not hold in its current form for wireless networks. This phenomenon arises due to interference of transmission on neighboring links. A Concatenation Theorem for links with mutual exclusion constraints needs to be furnished.

1.4.4 Data Dissemination

This dissertation presents and proves the properties of a minimum Steiner tree for data caching, namely binary structure and symmetrical internal angles, which to the best of our knowledge, are new results. Bhattacharya *et al.* present a heuristic for setting up a multicast tree in WSN [12]. The heuristic assumes the cost metric of an edge to be refresh rate * distance², and does not try to approximate an optimal Steiner Tree.

Several query-response paradigm data dissemination solutions exist. Directed Diffusion is a data-centric approach for communication [35]. In Directed Diffusion, intermediate nodes cache data. A transport layer for Directed Diffusion to provide guaranteed delivery and re-assembly has been proposed [64]. Ratnasamay *et al.* propose data-centric storage in WSN [61, 55]. Data is stored as (key, value) pair, and is replicated to avoid overloading. The keys are mapped to geographic locations for caching. Both of these rely on GPSR [40] for routing to a geographic coordinate or its closest neighbor. These efforts do not focus on minimum-energy multicast.

Kim *et al.* present SAFE, a protocol for data dissemination in WSN [41]. The SAFE protocol caches the data on *all* intermediate nodes en-route to the destination. Luo *et al.* propose data dissemination in a two-tier hierarchy supporting mobility of the sinks [68]. A grid structure is constructed pro-actively for forwarding data. The subscribers flood their local grid cell with their query. Nodes forming the grid forward the query upstream until data is found, which is then forwarded downstream to the subscriber. Finding a good grid size is crucial to good performance of this scheme. Regardless of the grid size, routing on a grid tends to increase route length and hence power consumption.

Cheng *et al.* present a protocol that builds a strongly connected minimum energy topology for WSN by adjusting the transmit power levels of nodes [19]. The metric optimized is the sum of transmit powers of individual nodes. The topology generated by the heuristics presented in the paper can be used for energy efficient routing.

Bauer and Varma present two heuristics to setup a multicast tree in a (virtual) circuit switched network for a given set of multicast members [10]. The K-SPH heuristic builds the tree by first creating forests, one for each member, and then by merging these forests pair-wise until a single tree is obtained. In the other heuristic, the tree starts to grow at one multicast member, and terminates when all members

are included. Unlike our heuristic, that manages the multicast tree *dynamically*, these heuristics *pre-compute* the multicast tree before starting data dissemination.

1.5 THESIS STATEMENT AND DISSERTATION CONTRIBUTIONS

Thesis Statement

It is possible to develop a systematic theory of real-time capacity of wireless sensor networks, and to construct low overhead protocols for real-time data gathering and data dissemination applications in wireless sensor networks.

Dissertation Contributions

Real-time wireless sensor networks: This research advances the design and analysis of wireless sensor networks for real-time applications. It presents analysis of real-time capacity limits of WSN for Deadline Monotonic (DM) and Earliest Deadline First (EDF) scheduling [Chapter 3]. It introduces an architecture based on a two-tier hexagonal topology for real-time application support in multi-hop wireless ad-hoc and sensor networks [Chapter 4]. It presents scalable time domain multiplexed MAC protocols that provide guarantees on bandwidth allocation. One protocol constructs network-wide transmission schedule and gives equal bandwidth to all the nodes [Chapter 4]. This protocol has zero scheduling message overhead and is optimal in the sense that the base-station does not idle. The other protocol supports a more general (unequal bandwidth to nodes) mixture of real and non-real time traffic [Chapter 5]. It presents distributed routing and implicit clock-synchronization protocols for this architecture.

Energy efficient data dissemination protocols: This research contributes to the area of energy efficient data dissemination [Chapter 6]. It extends the research on the Minimum Steiner Tree problem by defining and analyzing a more general construct called Steiner Data Caching Tree (SDCT). The tree cost metric of

SDCT is function of traffic rate in addition to the Euclidean edge lengths. This research presents and proves the optimality properties of this variant of the Steiner tree. Using the analytical properties of minimum SDCT, a middleware for data dissemination in WSN is presented. The middleware uses distributed protocol to construct near-optimal asynchronous multicast tree.

EDF schedulability on pipelines: The research presented here also makes contribution to the theory of real-time scheduling by presenting schedulability conditions for aperiodic tasks on a sequential resource pipeline [Chapter 2].

1.6 DISSERTATION OUTLINE

Chapter 2 presents the theory of EDF schedulability of aperiodic jobs in pipelines. It presents schedulability conditions for various load distribution scenarios. It furnishes the theorem on schedulability of arbitrary jobs that is used in the next chapter to derive real-time capacity limits in WSN. Chapter 3 presents a treatment of the real-time capacity limits. The limits are derived for two extreme traffic topologies – namely, the load balanced topology and the convergecast (i.e., many-to-one) topology. It considers DM and EDF scheduling algorithms, and discusses the implications of the capacity limit expressions. Chapter 4 presents an architecture for real-time support in wireless ad-hoc and sensor networks. The architecture is based on a cluster based hexagonal topology. The chapter presents addressing, routing, clock synchronization and scheduling algorithm for equal bandwidth MAC protocol. It presents the proofs of the optimality and real-time guarantees of the equal bandwidth transmission scheduling algorithm. Chapter 5 extends the previous chapter's research for a more general traffic. It considers a mixture of real-time as well as non-real time traffic, and the nodes are assumed to have a variable real-time bandwidth demand. Chapter 6 presents the solution of energy conserving data dissemination problem. It presents the analysis of optimal asynchronous multicast tree in WSN. Based upon the properties of minimum SDCT, it presents a distributed algorithm

that constructs and maintains near-optimal data dissemination tree dynamically. Chapter 7 summarizes the research presented in this dissertation, and points out the avenues for further research.

CHAPTER 2

EDF Scheduling in a Pipeline

2.1 INTRODUCTION

This chapter presents schedulability analysis of aperiodic jobs on pipelines. A pipeline refers to a system of processors and resources, arranged in tandem. In the next chapter, we present an analysis of real-time capacity of wireless sensor networks, where the packets are transmitted in priority order. We consider two prioritization algorithms, namely Earliest Deadline First and Deadline Monotonic. To derive the real-time capacity expressions, we model the network as a pipeline. For the derivation of the capacity expression for DM, we use the Stage-Delay theorem [2], and that for EDF, we use Theorem 2.14 presented in this chapter.

Each individual unit of the pipeline is called a *stage*. For our purpose of schedulability analysis, each stage is abstracted as a processor containing a dedicated job queue. Pipelines are especially suited when a job's overall execution consists of linearly partitioned sub-executions. In general, for the purpose of increasing throughput, tasks may be partitioned into multiple linear sub-units and hence can be made suitable for execution on a pipeline. Pipelines are a viable alternative to multiprocessors for increasing throughput.

In this chapter, we consider schedulability of hard real-time aperiodic jobs on a pipeline. We derive *sufficient* feasibility conditions for a pipeline using end-to-end deadlines and preemptive Earliest Deadline First (EDF) scheduling policy. These feasibility conditions determine schedulability for the pipeline as a whole, as opposed to individual stages of the pipeline. While many utilization bounds have

been derived for multiprocessors for various kinds of task sets ([29, 9, 47]), no similar bound exists for pipelines.

The rest of the chapter is organized as follows. We present the problem formulation in Section 2.2, followed by the analytical framework in Section 2.3. We derive schedulability bounds for various kinds of task sets in Section 2.4. We present the simulation results that compare the performance of the bounds derived in Section 2.5. We present the chapter summary in Section 2.6.

2.2 PROBLEM FORMULATION AND ASSUMPTIONS

In this section, we describe the system and our assumptions. We obtain bounds on the utilization of the entire pipeline. The individual stages are not assumed to maintain their individual state of utilization. We assume that the pipelines are sequential, and do not contain any loop or branches. Most scheduler implementations insert jobs in a prioritized queue, and schedule jobs at the head of the queue. Therefore, we do not assume that a lower priority job may skip a stage if its execution requirement is zero at that stage. We consider schedulability of aperiodic task sets. The inter-arrival time between any two instances of an aperiodic task is not assumed to be bounded from below. Thus more than one instance of a task may be released concurrently. Instances of a task are called *jobs*. Admitted jobs are scheduled to run according to the preemptive EDF scheduling policy. To enforce the correct execution ordering in such cases, and to avoid excessive preemption delays among jobs of the same priority, ties are assumed to be broken in FIFO order. We do not consider resource blocking. We assume that the system is work conserving. No clock synchronization among the stages of the pipeline is assumed.

In the next section, we present few basic results upon which we build the rest of this chapter.

2.3 ANALYTICAL FRAMEWORK

In this section, we first introduce some necessary notations. We follow these up with obtaining expressions for the worst case waiting time of the jobs.

We denote the number of stages of the pipeline by m . We denote jobs by J_i , where the indices reflect their priority (i.e., if $i < j$, then J_i is a higher priority job than J_j). We denote a job J_i 's overall worst-case execution time by C_i , and C_i 's stage-wise decomposition by C_{ik} , $1 \leq k \leq m$. We denote a job J_i 's release time by r_i , its relative deadline by D_i , and its absolute deadline, ($= r_i + D_i$), by d_i . If at time t , a job J_i has executed for $e_i(t)$ time units since its arrival, then we define its *outstanding execution* as $C_i^*(t) = C_i - e_i(t)$. The per stage outstanding execution times, $C_{ik}^*(t)$, are defined likewise.

2.3.1 Waiting time

In this sub-section, we consider the time that a job spends waiting in pipeline queues before it finishes its execution.

Definition 2.1 (Waiting time) Waiting time of a job is defined as the amount of time during which the job waits in the ready queues for execution, while higher priority jobs execute. The waiting time of a job J_i , that is released at r_i and finishes execution at f_i , is given by $W_i = f_i - r_i - C_i$, where C_i is the end-to-end execution time of the job.

To analyze the waiting time, the following theorem that considers execution order of the jobs in a pipeline is useful. Recall that in sequential pipeline, jobs exit from stages in priority order – a job can not skip a stage if execution requirement at that stage is zero. We call scheduling algorithms *job-level fixed priority* if they preserve the priority order among jobs. All (task-level) fixed priority scheduling algorithms are also *job-level fixed priority*. Since EDF determines priority using absolute deadline of the jobs, the EDF assigned priority of a job does not change over time. This

keeps the relative priority order between any pair of jobs fixed. Thus, although EDF is a task-level dynamic priority scheduling algorithm, it is a job-level fixed priority scheduling algorithm. However, Least-Laxity First is not a job-level fixed priority scheduling algorithm.

Theorem 2.2 *In a sequential pipeline, a job can preempt some lower priority job at-most once if a job-level fixed priority algorithm determines the priority order.*

Proof Let us consider two jobs J_i and J_n , where the latter is a lower priority job than J_i . The time of arrival of J_i at stage k is $r_{i,k}$. For the sake of clarity of presentation, let us denote the time of departure of J_i from stage k ($= r_{i,k+1}$) by $f_{i,k}$. If $f_{n,k} \geq r_{i,k}$ for all k , then the theorem is trivially true.

Let there exist a stage l such that $f_{n,l} < r_{n,l}$. At this stage J_i preempts J_n if the latter was executing at $t = r_{n,l}$, or J_i is scheduled to execute prior to J_n if the latter was waiting in queue at $t = r_{n,l}$. Since J_i is always ahead of J_n in stage l queue in the event of possible future preemptions, J_n starts to execute at stage l only after J_i exits this stage. Thus the number of preemptions of J_n by J_i at stage k is at-most 1, and $f_{i,l} \leq f_{n,l}$. At all subsequent stages, J_n starts to execute only after J_i exits there. Therefore, at no subsequent stage than l , J_n can be preempted by J_i . Hence the proof. \square

Now we consider the stage waiting time of a job J_n at some stage k . After J_n arrives at stage k , it can't start execution until all the higher priority jobs that were present at that stage have finished their executions. Furthermore, once J_n starts executing, it may get preempted by higher priority jobs that arrive at stage k afterwards.

At the first stage J_n finishes execution after all higher priority jobs in the worst case. Therefore, the first stage waiting time is bounded by simply the sum of the

first stage outstanding execution time of all higher priority jobs.

$$W_{n1} \leq \sum_{i=1}^{n-1} C_{i,1}^*(\text{MAX}(r_n, r_i)) \quad (2.1)$$

We found it convenient to calculate the stage waiting time of J_n at subsequent stages by breaking the delay into two parts. We calculate the waiting time of J_n at stage k due to those jobs that do not preempt J_n at stage k first, and then bound the additional delay that is caused by those jobs that preempt J_n at stage k . Since the jobs that arrive later than J_n can preempt the latter at some arbitrary stage, the last job to execute at stage k before J_n finishes execution at that stage is not necessarily J_{n-1} . Let us denote the index of the job that finishes immediately before J_n at stage k by the predicate $P(n, k, 1)$. Similarly, we denote the i^{th} job to finish before J_n at stage k by $P(n, k, i)$. In the following W_{nk} denotes the waiting time of J_n at stage k due to higher priority jobs than J_n that do not preempt J_n at stage k .

The expression for W_{nk} is equal to the sum of the execution and waiting times of its immediate predecessor at stage k , $J_{P(n,k,1)}$, minus the execution time of J_n at stage $k - 1$. A timing diagram depicting the relationship is shown in Fig. 2.1. In the following x^+ denotes $\text{MAX}(x, 0)$. The following treatment is with reference to

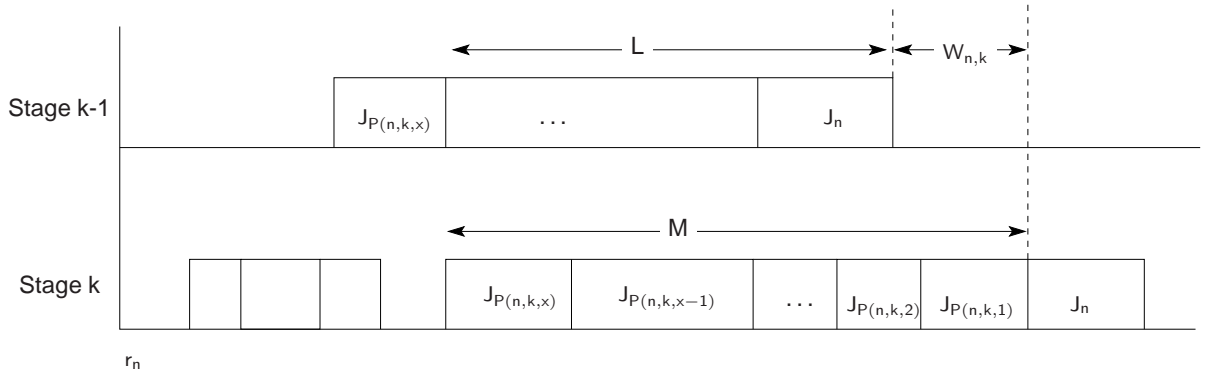


Figure 2.1 Execution at stage k from $t = r_n$ till $t = f_{n,k}$

J_n , and hence all outstanding execution times are with reference to $t = r_n$. We shall abbreviate $C_{i,k}^*(\text{MAX}(r_n, r_i))$ as $C_{i,k}^*$. Let us consider execution at stage k from $t = r_n$

until the time when J_n exits the stage k (Fig. 2.1). Let $J_{P(n,k,x)}$ be the first job that did not wait at stage k and $J_{P(n,k,x-1)}$ and all subsequent jobs up to J_n experience a non-zero waiting time at stage k . Following our assumption about $J_{P(n,k,x)}$, $W_{n,k}$ is given by $M - L$ in (Fig. 2.1). Therefore, we have:

$$\begin{aligned}
W_{nk} &= \left(\sum_{j=1}^x C_{P(n,k,j),k}^* - \sum_{j=1}^{x-1} C_{P(n,k,j),k-1}^* - C_{n,k-1} \right)^+ \\
&= \left(C_{P(n,k,x),k}^* - C_{n,k-1} + \sum_{j=1}^{x-1} C_{P(n,k,j),k}^* - C_{P(n,k,j),k-1}^* \right)^+ \\
&\leq \left((C_{P(n,k,x),k}^* - C_{n,k-1})^+ + \sum_{j=1}^{x-1} (C_{P(n,k,j),k}^* - C_{P(n,k,j),k-1}^*)^+ \right)^+ \\
&= (C_{P(n,k,x),k}^* - C_{n,k-1})^+ + \sum_{j=1}^{x-1} (C_{P(n,k,j),k}^* - C_{P(n,k,j),k-1}^*)^+. \tag{2.2}
\end{aligned}$$

Computation of the second term of (2.2) requires knowledge of the state of the pipeline when J_n arrives at each of the stages and the history of execution at that stage. This conflicts with our goal to obtain a fast schedulability test. Therefore, we further bound (2.2) as follows: $C_{P(n,k,x),k}^*$ is less than or equal to the maximum execution time of all n jobs at stage k , i.e., $C_{P(n,k,x),k}^* \leq \text{MAX}_{\{i|i \leq n\}} C_{i,k}^*$. Furthermore, the sum $\sum_{j=1}^{x-1} (C_{P(n,k,j),k}^* - C_{P(n,k,j),k-1}^*)^+ \leq \sum_{i=1}^{n-1} (C_{i,k}^* - C_{i,k-1}^*)^+$, since on the right side the sum is extended over all $n - 1$ higher priority jobs. Thus we have,

$$W_{nk} \leq \left(\text{MAX}_{\{i|i \leq n\}} C_{i,k}^* - C_{n,k-1} \right)^+ + \sum_{i=1}^{n-1} (C_{i,k}^* - C_{i,k-1}^*)^+ \tag{2.3}$$

Now, we consider the k^{th} stage waiting time of J_n due to jobs that preempt J_n at stage k . Let the remaining execution time of J_n at the time of preemption be λ (Fig. 2.2), where $\lambda < C_{i,k}$. Let $g_{n,k+1}(i)$ denote the time interval between the arrival time of J_n at stage k and its finishing time at stage $k + 1$ when it gets preempted at stage

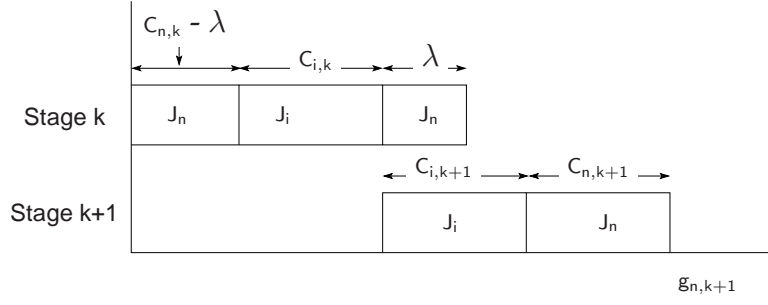


Figure 2.2 J_i preempts J_n when the latter needs an additional execution time of λ

k by some job J_i . Then, $g_{n,k+1}(i)$ is,

$$g_{n,k+1}(i) = C_{n,k} - \lambda + C_{i,k} + C_{i,k+1} + C_{n,k+1}.$$

Clearly, $g_{n,k+1}(i)$ is maximized when $\lambda \rightarrow 0$. Now, we execution the corresponding g of J_n had J_i arrived at stage k prior to J_n (Fig. 2.3). The time interval between the arrival time of J_n at stage k and its finishing time at stage $k + 1$ in this scenario is (written as $g'_{n,k+1}(i)$) given by:

$$g'_{n,k+1}(i) = C_{i,k} + \text{MAX}(C_{n,k}, C_{i,k+1}) + C_{n,k+1}.$$

The additional delay introduced by the late arrival of J_i is given by:

$$\begin{aligned} g_{n,k+1} - g'_{n,k+1} &= C_{n,k} - \lambda + C_{i,k+1} - \text{MAX}(C_{n,k}, C_{i,k+1}) \\ &\leq \text{MIN}(C_{n,k}, C_{i,k+1}) \end{aligned}$$

This additional delay is bounded over all stages as (written as $\Delta g_n(i)$):

$$\begin{aligned} \Rightarrow \Delta g_n(i) &\leq \text{MAX}_{\{k|k \leq m\}} (\text{MIN}(C_{n,k}, C_{i,k+1})) \\ &\leq \text{MAX}_{\{k|k \leq m\}} (C_{n,k}, C_{i,k}). \end{aligned}$$

Thus a later arriving job (than J_n) can decrease the parallel execution of a pipeline

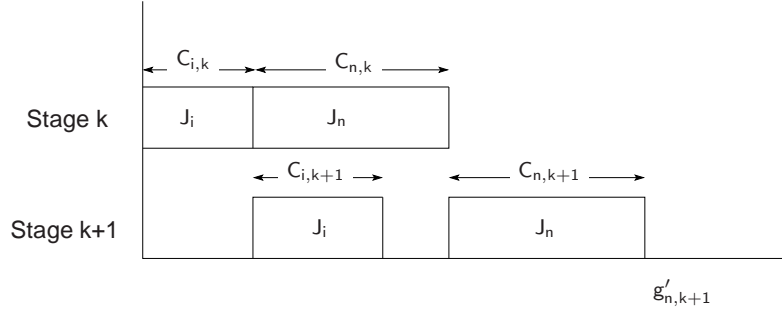


Figure 2.3 J_i preempts J_n and the latter needs large execution time at stage k

by a quantity that is the larger of the preempting and the preempted job's maximum stage execution time. Since we proved earlier (Theorem 2.2) that a job can preempt another job at-most once, for a given J_i , $\Delta g_n(i)$ is non-zero for only one of the stages. Therefore, the (end-to-end) waiting time of J_n due to all higher priority tasks is bounded by the sum of $W_{n,k}$ at all stages and one $\Delta g_n(i)$ delay by every higher priority job that arrived afterwards.

In other words,

$$W_n \leq \sum_{k=1}^m W_{nk} + \sum_{l \in \{\text{Higher priority jobs arriving after } J_n\}} \text{MAX}_{\{k|k \leq m\}} (C_{n,k}, C_{l,k}). \quad (2.4)$$

After obtaining the bound on the stage-wise waiting time of J_n , we now proceed to obtain the bound on the end-to-end waiting time of J_n in the pipeline.

Substituting the expressions for W_{n1} (2.1) and W_{nk} (2.3) in (2.4), we get:

$$W_n = \sum_{i=1}^{n-1} C_{i1}^* + \sum_{k=2}^m W_{nk} + \sum_{l \in \{\text{Higher priority jobs arriving after } J_n\}} \text{MAX}_{\{k\}} (C_{n,k}, C_{l,k})$$

$$\begin{aligned}
\Rightarrow W_n &\leq \sum_{i=1}^{n-1} \left(C_{i1}^* + \sum_{k=2}^m (C_{ik}^* - C_{i,k-1}^*)^+ \right) + \\
&\quad (m-1) \left(\text{MAX}_{\{i,k|1 \leq i \leq n-1\}} C_{ik}^* - C_{n,k-1} \right)^+ \\
&\quad + \sum_{l \in \{\text{Higher priority jobs arriving after } J_n\}} \text{MAX}_{\{k|k \leq m\}} (C_{n,k}, C_{l,k}) \\
\Rightarrow W_n &\leq \sum_{i=1}^{n-1} \left(C_{i1}^* + \sum_{k=2}^m (C_{ik}^* - C_{i,k-1}^*)^+ \right) \\
&\quad + (m-1) \left(\text{MAX}_{\{i,k|1 \leq i \leq n-1\}} C_{ik}^* - C_{n,k-1} \right)^+ + p_n \text{MAX}_{\{i,k|1 \leq i \leq n\}} C_{ik}^*, \quad (2.5)
\end{aligned}$$

where p_n is the number of higher priority jobs arriving after J_n .

The first sum in (2.5) is:

$$\begin{aligned}
&\sum_{i=1}^{n-1} \left(C_{i1}^* + \sum_{k=2}^m (C_{ik}^* - C_{i,k-1}^*)^+ \right) \\
&= \sum_{i=1}^{n-1} \left(\sum_{k=1}^m C_{ik}^* + \sum_{k=2}^m (C_{ik}^* - C_{i,k-1}^*)^+ - \sum_{k=2}^m C_{ik}^* \right) \\
&= \sum_{i=1}^{n-1} \left(\sum_{k=1}^m C_{ik}^* - \sum_{k=2}^m \text{MIN}(C_{ik}^*, C_{i,k-1}^*) \right) \\
&= \sum_{i=1}^{n-1} \text{MAX}_{\{k\}} C_{i,k},
\end{aligned}$$

since subtracting $k-1$ of the smallest terms from k terms leaves the largest term.

Therefore, (2.5) can be written as:

$$W_n \leq \sum_{i=1}^{n-1} \text{MAX}_{\{k\}} C_{i,k}^* + (m-1) \left(\text{MAX}_{\{i,k|1 \leq i \leq n-1\}} C_{ik}^* - C_{n,k-1} \right)^+ + p_n \text{MAX}_{\{i,k|1 \leq i \leq n\}} C_{ik}^*, \quad (2.6)$$

2.3.2 Schedulable load

We now seek a metric to measure the amount of computation that the system must perform during the time interval in which a certain job is current in order to

make the job meet its deadline. For the sake of convenience of argument, let us consider a job J_x , and assume that it arrives some time in the middle of the current busy period. When it arrives, there are some jobs already executing at various stages of their execution time. Furthermore, after the arrival of J_x and before its completion, more jobs are admitted, some of which have earlier deadlines than J_x . Since we concern ourselves here with EDF, all the jobs that have a deadline earlier than J_x are also of higher priority than x . Out of all jobs described, let us isolate the jobs that are of higher priority than J_x , and let the index of J_x in this set be i . According to our convention, we denote J_x by J_i . J_i uniquely determines this set, and we refer to this set by \mathbf{J}_i . The execution demand due to jobs that arrived earlier than J_i (including itself, is $\sum_{j=1}^i C_j^*(t = r_i)$, whereas the execution demand due to jobs that arrive after J_i does is $\sum_{j=1}^i C_j = \sum_{j=1}^i C_j^*(t = r_j)$. Therefore, the total execution demand on the system due to J_i and its higher priority jobs can be written as $\sum_{j=1}^i C_j^*(t = \max(r_i, r_j))$. This much computation must be done in the time interval $[r_i, d_i]$, the magnitude of which is $d_i - r_i = D_i$.

Definition 2.3 (Load of a job) The load experienced by J_i , denoted by L_i , is defined as $L_i = \sum_{j=1}^i C_j^*(t = \max(r_i, r_j))/D_i$.

We now derive an expression for load in terms of the execution demand, the wait time in job queues and deadline that makes a job, J_n , critically schedulable. A critically schedulable job is defined as follows:

Definition 2.4 (Critically schedulable job) A job J_n is defined as critically schedulable if, were the execution requirement of J_n increased infinitesimally while keeping its deadline fixed, J_n would have missed its deadline.

Since J_n is critically schedulable, its waiting time, W_n , equals its laxity. Hence,

$$W_n = D_n - C_n$$

Using the inequality for the waiting time (2.6), we get,

$$D_n - C_n \geq \sum_{i=1}^{n-1} \text{MAX}_{\{k\}} C_{i,k}^* + (m-1) \left(\text{MAX}_{\{i,k|1 \leq i \leq n-1\}} C_{ik}^* - C_{n,k-1} \right)^+ + p_n \text{MAX}_{\{i,k|1 \leq i \leq n\}} C_{ik}^*$$

$$\Rightarrow \sum_{i=1}^{n-1} \text{MAX}_{\{k\}} C_{i,k}^* \leq D_n - C_n - (m-1) \left(\text{MAX}_{\{i,k|1 \leq i \leq n-1\}} C_{ik}^* - C_{n,k-1} \right)^+ - p_n \text{MAX}_{\{i,k|1 \leq i \leq n\}} C_{ik}^*$$

But, $\sum_{i=1}^{n-1} C_i^* \leq m \sum_{i=1}^{n-1} \text{MAX}_{\{k\}} C_{i,k}^*$. Therefore,

$$\sum_{i=1}^{n-1} C_i^* \leq m(D_n - C_n) - m(m-1) \left(\text{MAX}_{\{i,k|1 \leq i \leq n-1\}} C_{ik}^* - C_{n,k-1} \right)^+ - mp_n \text{MAX}_{\{i,k|1 \leq i \leq n\}} C_{ik}^* \quad (2.7)$$

From definition, the load L_n is given by,

$$L_n = \sum_{i=1}^n C_i^* / D_n$$

$$= \sum_{i=1}^{n-1} C_i^* / D_n + C_n / D_n$$

$$= 1 + 1/D_n \left(\sum_{i=1}^{n-1} C_i^* - W_n \right).$$

But the waiting time can not exceed the total execution time of all higher priority jobs. Therefore, we write

$$L_n = 1 + 1/D_n \left(\sum_{i=1}^{n-1} C_i^* - W_n \right)^+.$$

Equivalently, for critically schedulable J_n ,

$$L_n = 1 + 1/D_n \left(\sum_{i=1}^{n-1} C_i^* - D_n + C_n \right)^+. \quad (2.8)$$

Plugging for $\sum_{i=1}^{n-1} C_i^*$ from (2.7) in (2.8), we get

$$L_n \leq 1 + 1/D_n \left\{ (m-1)(D_n - C_n) - m(m-1) \left(\text{MAX}_{\{i,k|1 \leq i \leq n-1\}} C_{ik}^* - C_{n,k-1} \right)^+ \right. \\ \left. - mp_n \text{MAX}_{\{i,k|1 \leq i \leq n\}} C_{ik}^* \right\}^+. \quad (2.9)$$

2.3.3 Bound on schedulable load

After obtaining the expression for load that makes a *given* job critically schedulable, we now consider the load that will make *every* job of a given task set schedulable.

As we saw previously, the waiting time of a job J_n depends on the future arrival of higher priority jobs (while J_n has not finished execution) – in addition to those that arrived earlier. Let τ be the given task set.

Definition 2.5 (Schedulable system load) The schedulable system load, denoted by \hat{L} , is defined as *sufficient condition* on the least upper bound of the schedulable load for any job in τ .

To construct the bound on schedulable load, we *minimize* the schedulable load obtained in (2.9) over τ . Let $C_{ik_{max}}$ and $C_{ik_{min}}$ denote the upper and lower bounds on C_{ik} . Let p denote the largest of the number of higher priority jobs that arrive while any job in τ is current, the maximization is done over all jobs. Let $u_{i_{max}} = \text{MAX}_{\{i\}} u_i$. Minimization of (2.9) gives:

$$\hat{L} = 1 + ((m-1)(1 - u_{i_{max}}) - m(m-1)(C_{ik_{max}} - C_{ik_{min}})/D_{min} \\ - mpC_{ik_{max}}/D_{min})^+. \quad (2.10)$$

The small number of variables of \hat{L} allows to keep the number of states to be maintained by the admission controller small, and hence act fast. In the next sections, we shall obtain non-trivial bounds on the maximum admissible load as a function of the properties of the tasks in τ , which we denote as \hat{L} . The bound \hat{L}

has the property that if the load experienced by any job J_i that is an instance of some task in τ does not exceed \hat{L} then J_i will not miss its deadline.

2.3.4 Utilization based acceptance test

In the previous section, we obtained a bound on the schedulable load of a pipeline. As the last step in the treatment of our schedulability analysis, we take up the subject of the construction of a small time complexity acceptance test. As pipeline executes the jobs, the outstanding computation times of some or all of the jobs change. Therefore, load changes continuously with time. We now seek a metric that changes only for a constant number of times during the execution of a job so that using this metric, constant time complexity utilization tests can be constructed. Few definitions are in order here.

Definition 2.6 (Current job) A job J_i is defined to remain *current* in the time interval $[r_i, d_i]$ where r_i is the release time of J_i , and d_i is its absolute deadline.

Definition 2.7 (Job synthetic utilization) The synthetic utilization of an aperiodic job J_i is defined as $u_i = C_i/D_i$ when J_i is current.

Please note that we shall use the terms *synthetic utilization* and *utilization* in the following interchangeably. Whenever, we shall discuss real utilization, we shall make a note of the fact.

Definition 2.8 (Busy period) A busy period for a pipeline is defined as the interval of time during which at-least one of the stages of the pipeline has one or more jobs to execute.

Definition 2.9 ((Synthetic) Utilization of a system) Let $B(t)$ be a boolean function of the state of the pipeline which is defined to be 1 during a busy period, and 0 otherwise. Then, the system utilization, $U(t)$, of a pipeline is defined as equal to $U(t) = B(t) \sum_{i \in \{\text{Current jobs who arrived since the last idle time}\}} u_i$.

In other words, the system utilization is defined as equal to the sum of utilizations demands of all current jobs in the system who arrived since the last idle time, and 0 whenever the system has no outstanding execution (i.e., when the system is idle). This definition implies that for the purpose of schedulability tests, we need to consider the latest busy period only. $U(t)$ changes only when a job is admitted, when a job expires and whenever the system gets idle. Hence updating $U(t)$ requires at-most 2 updates (of a constant number of variables) per admitted job. This makes obtaining a bound on $U(t)$ suitable for the construction of fast admission control tests. In the following theorem, we prove that enforcing $U(t) \leq \hat{L}$ at the time of admission of the jobs ensures schedulability.

Theorem 2.10 *If the system utilization, $U(t)$, is kept below \hat{L} whenever a job is admitted, then all jobs meet their deadline.*

Proof The proof is in two parts. Let us consider some job J_n , that is current in the time interval $[r_n, d_n]$.

In the first part we consider the case when all higher priority jobs that are current in $[r_n, d_n]$ arrive *after* r_n . And in the second part, we consider the case when some of the higher priority jobs arrive *before* r_n .

Part-I of the proof:

Let $U_n(t)$ denote the pipeline utilization due to J_n and all other higher priority jobs in $(r_n, d_n]$. Since we are considering EDF priority, all such jobs must have a deadline no greater than d_i .

Consider a plot of $U_n(t)$ (Fig. 2.4). Since there was no contributing job to $U_n(t)$ prior to $t = r_n$, and the utilization is non-zero only until the deadline of jobs, the curve $U_n(t)$ is 0 everywhere outside $[r_n, d_n]$. Furthermore, the utilization demand u_j of a job J_j is nonzero only in the interval $[r_j, d_j]$. Hence, $\int_{r_n}^{d_n} u_j(t) dt = \int_{r_j}^{d_j} u_j(t) dt =$

$C_j/D_j \int_{r_j}^{d_j} dt = C_j/D_j \times D_j = C_j$. The integral $\int_{r_n}^{d_n} \mathbf{U}_n(t) dt$ can then be written as:

$$\begin{aligned} \int_{r_n}^{d_n} \mathbf{U}_n(t) dt &= \int_{r_n}^{d_n} \sum_{j=1}^n u_j(t) dt \\ &= \sum_{j=1}^n u_j \int_{r_n}^{d_n} dt \\ &= \sum_{j=1}^n C_j. \end{aligned} \quad (2.11)$$

Let us denote by $\hat{\mathbf{U}}_n$ the largest value of $\mathbf{U}_n(t)$. Then using (2.11), we can write the

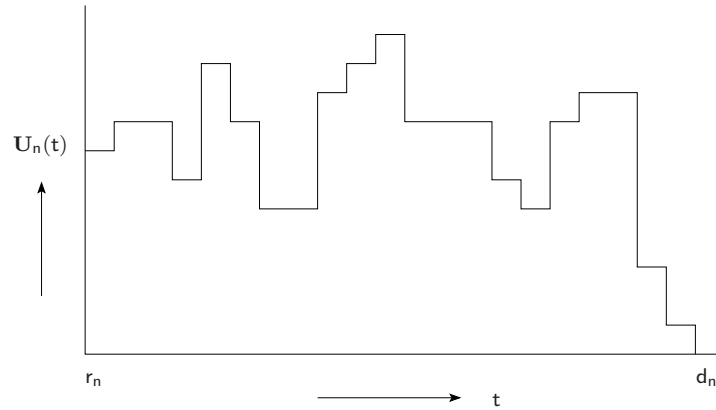


Figure 2.4 A busy period utilization curve

following:

$$\begin{aligned} L_n &= \sum_{j=1}^n C_j/d_n = \int_{r_n}^{d_n} \mathbf{U}_n(t) dt / D_n \\ &\leq \int_{r_n}^{d_n} \hat{\mathbf{U}}_n dt / D_n \\ &\leq \hat{\mathbf{U}}_n \end{aligned} \quad (2.12)$$

But $\mathbf{U}(t) \leq \hat{L}$ at all times in $[0, d_n]$ by design. Therefore from (2.12) $L_n \leq \hat{L}$ at all times, including the time when $\mathbf{U}_n(t)$ attains its maximum. This completes the first part of the proof.

Part-II of the proof:

Proof by induction. WLOG, we assume that the pipeline is fully utilized, for it follows from the definition of load that a fully utilized pipeline would never decrease the load of some job obtained while pipeline is partially occupied. Therefore, if the load for some job were to exceed \hat{L} with a partially occupied pipeline, it would still exceed \hat{L} with more jobs executing. We consider fully utilized pipeline because if the pipeline is fully utilized, $U(t)$, touches the curve of \hat{L} for at-least one time instant during the time interval when the job J_n was current. Hence, showing $L_n > U(t)$ immediately implies $L_n > \hat{L}$. In the event of partially used pipeline, the implication of $L_n > U(t)$ remains inconclusive.

Base case: consider the highest priority job that arrived since the beginning of the busy period.

$$L_1 = C_1/D_1 = u_1$$

Clearly, $u_1 \leq \hat{U}_1$. Furthermore, \hat{U}_1 is equal or smaller than the largest value of $U(t)$ in $[r_1, d_1]$. Since $U(t) \leq \hat{L}$ at all times by design, $L_1 \leq \hat{U}_1 \leq \hat{L}$. General case:

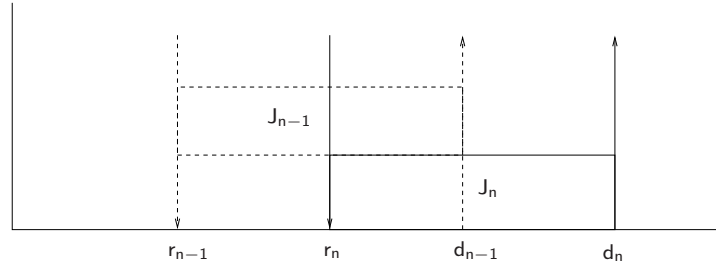


Figure 2.5 Load for J_n and J_{n-1}

Let $L_i \leq \hat{U}_i \leq \hat{L}$ for $i = 1, \dots, n - 1$. Recall that J_{n-1} is the job that has immediate higher priority than J_n , that is, after the expiration of the deadline of J_{n-1} , the next job whose deadline expires is J_n , since we are using EDF. In the following, we write the total execution in an interval $[x, y]$ as $\mathcal{W}(x, y)$. The load for job J_n (Fig.(2.5)) can be written as:

$$L_n = \frac{L_{n-1}D_{n-1} + C_n - \mathcal{W}(r_{n-1}, r_n)}{D_n}.$$

By induction hypothesis,

$$\begin{aligned} L_n &\leq \frac{\hat{U}_{n-1}D_{n-1} + C_n - \mathcal{W}(r_{n-1}, r_n)}{D_n} \\ &= \frac{\int_{r_{n-1}}^{d_{n-1}} \hat{U}_{n-1} dt + \int_{r_n}^{d_n} u_n dt - \mathcal{W}(r_{n-1}, r_n)}{D_n} \\ &\leq \frac{\int_{r_n}^{d_n} \hat{U}_n dt}{D_n} - \left(\mathcal{W}(r_{n-1}, r_n) - \int_{r_{n-1}}^{r_n} \hat{U}_{n-1} dt \right) / D_n \\ &= \hat{U}_n - \left(\mathcal{W}(r_{n-1}, r_n) - \int_{r_{n-1}}^{r_n} \hat{U}_{n-1} dt \right) / D_n \end{aligned} \quad (2.13)$$

Since the pipeline is executing at its full capacity, $\mathcal{W}(r_{n-1}, r_n) \geq \int_{r_{n-1}}^{r_n} \hat{L} dt$, but $\hat{L} \geq \hat{U}_{n-1}$ by design. Therefore, $\mathcal{W}(r_{n-1}, r_n) \geq \int_{r_{n-1}}^{r_n} \hat{U}_{n-1} dt$. Application of this inequality to (2.13) gives,

$$L_n \leq \hat{U}_n \leq \hat{L}.$$

This completes the proof. □

Now we are ready to spell out sufficient conditions for schedulability that have time complexity that is constant in the number of current jobs. For reader's convenience, we present a list of frequently used symbols before proceeding on to the derivation of utilization bounds.

List of frequent symbols

m	Number of stages of the pipeline
J_i	Jobs
C_i	The overall execution time of J_i
C_{ik}	The execution time of J_i at stage k
r_i	Release time of J_i
D_i	Relative deadline of J_i
d_i	Absolute deadline ($= r_i + D_i$) of J_i
$e_i(t)$	The amount of time that J_i has executed at time t , since its release
$C_i^*(t)$	The outstanding overall execution time of J_i at time t , ($= C_i - e_i(t)$)
W_i	Waiting time of J_i
α_{ik}	The fraction C_{ik}/C_i
α_{ik}^*	The fraction C_{ik}^*/C_i
L_i	The load for J_i ($= \sum_{j=1}^i C_j^*(t = r_i)/D_i$)
u_i	Utilization of job J_i ($= C_i/D_i$)
U	Utilization of the pipeline

2.4 CONSTANT TIME UTILIZATION BOUNDS

In the following sections, we present constant time utilization bounds for three types of task sets, namely the set that contains tasks of arbitrary, but known decomposition of execution time across the stages of the pipeline, the set that contains tasks of a fixed decomposition of execution time at any given stage, and the set that contains load balanced tasks, that is, the task instances execute for equal time at every stage.

We also show that in a pipeline, the uniprocessor bound is the best that we can guarantee if the decomposition of execution time of the tasks is unknown. This includes those tasks that use only one of the stages of the pipeline. As we shall see in

Section 2.4.4, such tasks are responsible for bringing the schedulable system utilization down to that of a uniprocessor. In practice, however, the tasks will have their execution requirements distributed over multiple, possibly all, stages of the pipeline (when pipelines are used to scale up computation, the sub-unit size distribution is programmer controllable indeed). Such task sets are of interest for our system. We show that the guaranteeable system utilization for such task sets is considerably higher. For load-balanced task sets in an m stage pipeline, $O(m)$ system utilization is achievable (see Section 2.4.3 for the exact relation).

2.4.1 Known Decomposition of Execution Time

We now state the bound on pipeline utilization when the decomposition of execution times across the stages is known.

Theorem 2.11 (U_{Bound_1}) *Total system utilization $U = \sum_{i=1}^n U_i \leq 1 + ((m - 1)(1 - u_{i_{max}}) - m(m - 1)(C_{ik_{max}} - C_{ik_{min}}) / D_{min} - mpC_{ik_{max}} / D_{min})^+$ is a sufficient condition for schedulability of jobs J_1, \dots, J_n in a pipeline using EDF.*

Proof of Theorem 2.11 The theorem follows immediately from Theorem 2.10 and the bound on load \hat{L} (2.10). \square

To construct an on-line admission controller from this utilization bound, we need to assume that the task set is periodic. This is due to the fact that for aperiodic job arrivals, p , the largest number of higher priority jobs that can preempt some given job can be large enough to render this bound too pessimistic and hence useless. Using game theoretic arguments, Dertouzos and Mok show that optimal scheduling on multiprocessors is impossible if start times, deadlines or the computation times of the jobs are not known apriori [23]. Algorithm 1 presents a constant time admission control for periodic task sets based on the bound of Theorem 2.11. The time

complexity of computing p is $O(|\tau|)$, and that of the rest of the steps is $O(1)$ for every task; where $|\tau|$ is the size of the task-set. Therefore, the time complexity of this algorithm is $O(|\tau|^2)$. Observe that the number of tasks is in general much smaller than the number of jobs running on the pipeline.

```

INIT:  $U \leftarrow 0$  { $U$  is the pipeline utilization}
 $U_{Bound_1} \leftarrow \infty$ 
 $\tau \leftarrow \phi$ 
loop
5:  Do upon arrival of every task  $T_x$ 
     $U'_{Bound_1} \leftarrow U_{Bound_1}$  for  $\tau \cup T_x$ 
    if  $U + C_x/D_x \leq U'_{Bound_1}$  then
      Admit  $T_x$ 
       $U += C_x/D_x$ 
10:   $U_{Bound_1} \leftarrow U'_{Bound_1}$ 
       $\tau \leftarrow \tau \cup T_x$ 
    else
      Reject  $T_x$ 
    end if
15: end loop

```

Admission control algorithm 1: Using U_{Bound_1}

2.4.2 Fixed Decomposition of Execution Time

In the following, we derive bounds for further restricted task sets where every job executes for the same fraction of its end-to-end execution time at any given stage of the pipeline. In other words, $\alpha_{ik} = \alpha_k$ for all i , where α_{ik} is the fraction of the end-to-end execution time of a job J_i at stage k , that is $\alpha_{ik} = C_{i,k}/C_i$. The jobs are however free to have arbitrary total execution time and deadline. Since the α 's are fixed, (2.10) can be written as:

$$\hat{L} = 1 + ((m-1)(1-u_{i_{max}}) - m(m-1)(\alpha_{max} - \alpha_{min})C_{max}/D_{min} - mp\alpha_{max}C_{max}/D_{min})^+. \quad (2.14)$$

Let \mathcal{U}_{max} denote C_{max}/D_{min} . Then, (2.14) is:

$$\hat{L} = 1 + ((m-1)(1-u_{i_{max}}) - m(m-1)(\alpha_{max} - \alpha_{min})\mathcal{U}_{max} - mp\alpha_{max}\mathcal{U}_{max})^+. \quad (2.15)$$

Therefore, we have the following theorem:

Theorem 2.12 (\mathbf{U}_{Bound_2} : Sufficiency condition when $C_i^k = \alpha_k C_i$, $\sum_k \alpha_k = 1$) $\mathbf{U} = \sum_{i=1}^n u_i \leq 1 + ((m-1)(1-u_{i_{max}}) - m(m-1)(\alpha_{max} - \alpha_{min})\mathcal{U}_{max} - mp\alpha_{max}\mathcal{U}_{max})^+$ is a sufficient condition for schedulability of jobs J_1, \dots, J_n in a pipeline using EDF, if the fractions of execution time are fixed and the same for all jobs.

Admission control algorithms corresponding to the \mathbf{U}_{Bound_2} , and the following bounds can be written in a similar manner as Algorithm 1, and are omitted for the sake of space.

2.4.3 Load Balanced Execution Time

From (2.15), the bound that we derived in the previous theorem (2.12) attains its maximum when the jobs spend equal fraction of their (unrestricted) execution time at every stage of a pipeline, i.e., $\alpha_{ik} = 1/m$ for all k .

Theorem 2.13 (\mathbf{U}_{Bound_3} : Sufficiency condition when $C_i^k = C_i/m$, $\forall k$) $\mathbf{U} = \sum_{i=1}^n u_i \leq 1 + ((m-1)(1-u_{i_{max}}) - p\mathcal{U}_{max})^+$ is a sufficient condition for schedulability of jobs J_1, \dots, J_n in a pipeline using EDF, if the fractions of execution time that the jobs execute at all the stages are equal.

2.4.4 Unknown Decomposition of Execution Time

We now consider the task set where stage-wise decomposition of the overall execution time is unknown.

Theorem 2.14 (U_{Bound_4}) $U = \sum_{i=1}^n u_i \leq 1$, that is $U_{Bound_4} = 1$, is a sufficient condition for schedulability of jobs J_1, \dots, J_n in a pipeline using EDF if the decomposition of the execution times of the jobs is unknown.

Proof of Theorem 2.14 The expression (2.10) tends to 1 from above as the second term under $()^+$ sign tends to 0. Without knowledge of the decomposition of execution time, the second term is 0, and $\hat{L} = 1$. Application of Theorem 2.10 to $\hat{L} = 1$ completes the proof. \square

The sufficiency condition of Theorem 2.14 is also the utilization bound for uniprocessors. This is a tight lower bound for a general task set when the decomposition of C is unknown. To show, consider the following example (Fig. 2.6): two jobs J_1 and J_2 are to be scheduled on an m -stage pipeline. Suppose that J_1 has a very small execution time requirement, $0 < \epsilon < 1$, at the first stage, and 0 execution time at all subsequent stages. J_2 also needs ϵ execution time at the first stage, but it also needs 1 execution time at some subsequent stage. Finally, let the relative deadlines of J_1 and J_2 be 1 and $1 + \epsilon$, respectively, and let them be released simultaneously. EDF will schedule J_1 before J_2 , and hence J_2 will miss its deadline. The system utilization of this example of unschedulable set of jobs is $1 + \epsilon$. By choosing small enough ϵ , the system utilization can approach arbitrarily close to 1. This establishes the bound's tightness.

This example also serves the purpose of showing that EDF is not optimal for pipelines. An optimal scheduling algorithm would have scheduled J_2 before J_1 at the first stage, thereby enabling both jobs to meet their deadline.

2.5 EVALUATION

We performed simulations to evaluate the actual utilization of a pipeline when an admission control based on the bounds derived in this chapter is used. We implemented a simulator that generated a number of periodic tasks whose deadline

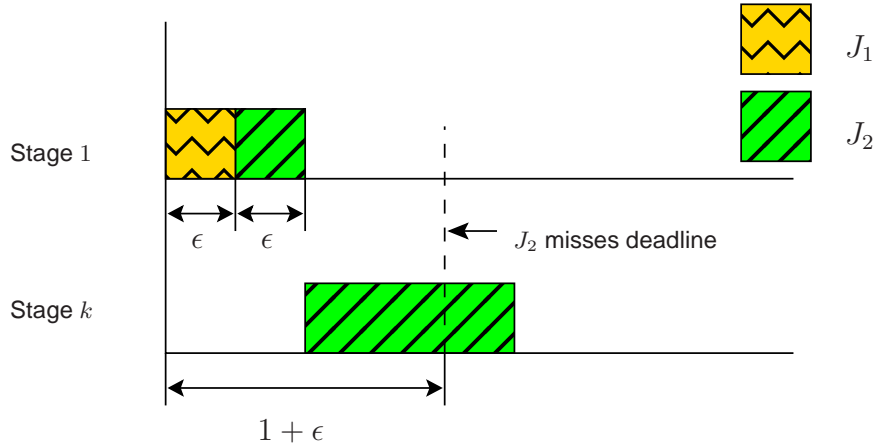


Figure 2.6 Non-optimality of EDF for pipelines.

was set equal to period. Therefore, the task utilization also represented the actual utilization of the pipeline. We studied the actual pipeline utilization as a function of the size of the pipeline and the utilization of individual tasks. The reported actual system utilization is averaged over all the stages of the pipeline. Each point corresponds to the average of 1000 runs.

In all simulations, the pipeline was run at the maximum load allowed by the bound. The stage computation times of the pipeline of fixed α (cf. section 2.4.2) was varied by a normally distributed random number, where 99.99% of the values varied between $\pm 15\%$ of the mean. The per stage execution times of arbitrary jobs, and individual end-to-end execution times of load balanced task set was varied similarly.

We measured utilization by creating task sets where every member of the set had a given utilization u . Figure 2.7 shows the system utilization averaged over all stages as a function of task sets of individual utilizations u . The performance ratio tends to get more pessimistic for the jobs of larger utilization because in this case there is less slack available to allow for execution parallelism. In the limit of $u \rightarrow 1$, all bounds tend to the uniprocessor bound. The curves converge to 0.1 at $u = 1$ because the simulated pipeline had 10 stages.

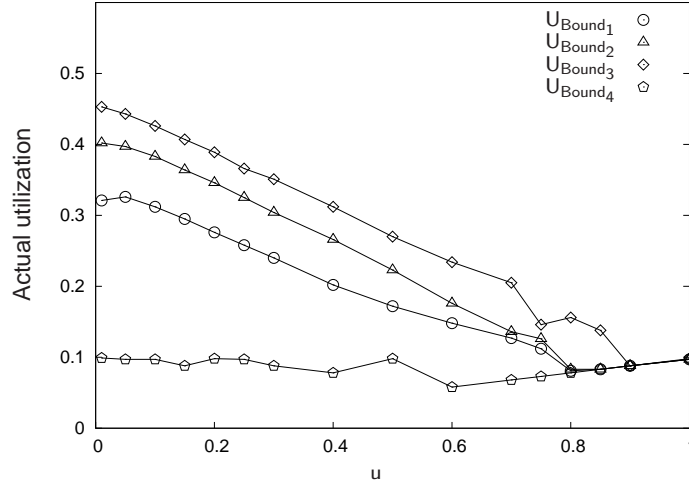


Figure 2.7 Actual system utilization vs. task utilization. In the legend, U_{Bound_i} refer to the utilization bounds derived earlier.

Figures 2.8a–d show the variation of utilization as a function of the number of stages of a pipeline. The figures are parameterized by u . For load balanced task sets of smaller individual task utilizations, the bound scales well with the number of stages of the pipeline. Using the result for load balanced task set, U_{Bound_3} (2.13), it can be shown that in the limit of large number of stages and in the presence of small fluctuations in C_i and D_i for a load-balanced periodic task set, the per stage utilization tends to $(1 - u)/2$. In the simulations, actual per stage utilization for load balanced task set nearly achieves this limit for a pipeline of 15 or more stages (Fig. 2.8). For task sets of larger utilizations, the bounds for load balanced task sets and that for task sets of fixed per-stage execution ratio achieve 20-25% average utilization and do not become pessimistic as the number of stages of the pipeline is increased. We conclude that u is a more dominant parameter than pipeline size, and that the task sets that issue heavy or mixed jobs tend to make the schedulable system utilization of pipelines pessimistic.

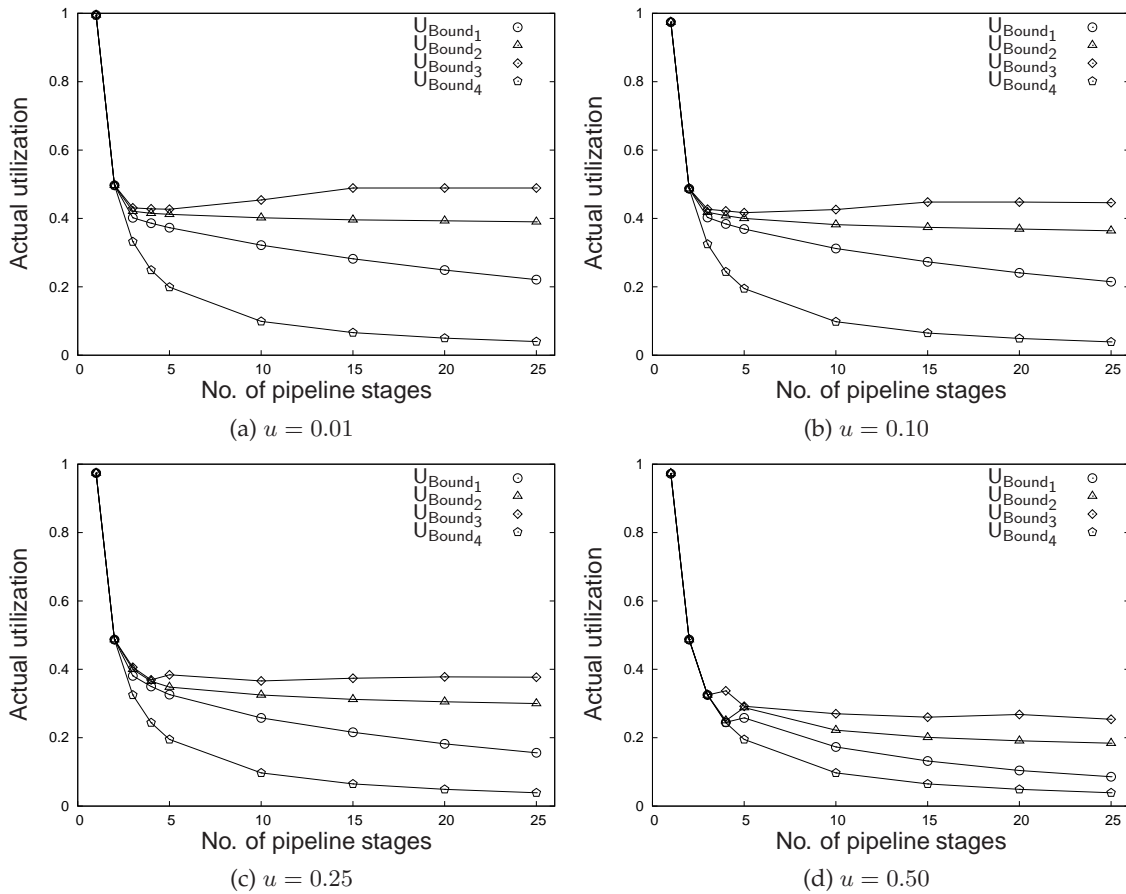


Figure 2.8 Actual system utilization vs. number of pipeline stages

2.6 SUMMARY

We shall use Theorem 2.14 to derive the real-time capacity expressions in the next chapter. We also note that the utilization bound for the load balanced case has somewhat direct but rather limited applicability to predict schedulability of streams in WSN. For the case of isolated streams, the nodes on the path of the stream are equally loaded, and therefore, one may apply Theorem 2.13 to determine the EDF schedulability of aperiodic or periodic flows (periodic flows are special case of aperiodic flows) in WSN by modeling links as the stages. An appropriate interference model can easily be taken into account by scaling down the schedulable utilization. For example, let the schedulable utilization obtained is x . If the interference model tells that the links connected by an edge from get blocked, then, only one in three

stages of the pipeline can operate in the worst case. Therefore, if the schedulable utilization incorporating the interference is $x/3$.

In regards to the theory of real-time scheduling in pipelines, we derived bounds on the schedulable system utilization using EDF for a general task set and for several interesting special cases. We saw that the bounds on schedulable utilization increased as we moved towards more specific task sets. We also saw that under load-balanced conditions, the utilization bound for a pipeline can be of the order of number of stages in the pipeline. This is the first time that we have a utilization bound for pipelined architectures that guarantees utilization of the order of number of stages of the pipeline. We showed non-optimality of EDF in pipelines.

CHAPTER 3

Real-Time Capacity Limits

3.1 INTRODUCTION

Capacity is a measure of information carrying ability of a network. Real-time capacity of a network is defined to be its information carrying ability for given deadlines. Only those information bits that arrive their destination within the specified deadline count. This chapter obtains closed-form analytic expressions for real-time transport capacity of multi-hop wireless sensor networks. Our focus will be primarily on large networks. The expressions are obtained using sufficient schedulability conditions, and therefore provide a conservative estimate. Using simulation studies, we found that the level of pessimism of the expressions varies in the range of 10-40% in most cases. The analytic expressions for real-time capacity facilitate the process of designing a network that is guaranteed to meet specified real-time requirements. The feasibility region defined by the capacity expressions can be used for optimization of the operation of the network in the event of dynamically changing network, which is expected of WSN.

The treatment of real-time capacity limits for Deadline Monotonic scheduling algorithm (DM) uses earlier results of Abdelzaher *et al.* on schedulability of aperiodic jobs in pipelines [2]. In DM, the priority assignment depends upon the relative deadline. Jobs with smaller relative deadline are assigned higher priority. The treatment for EDF scheduling algorithm uses the schedulability results presented in the previous chapter.

The rest of this chapter is organized as follows: we present the problem formulation in the next section, followed by our assumptions in Section 3.3. We present the

path feasibility conditions in Section 3.4, followed by derivation of real-time capacity expressions in Section 3.5. We present simulation results in Section 3.6. After that we present a discussion of the limitations of the theoretical results of this chapter. Finally, we present the chapter summary in Section 3.8.

3.2 PROBLEM FORMULATION

We consider a multi-hop wireless sensor network where the number of nodes is denoted by n . The set of nodes that can receive packets from node j is denoted by $neighborhood(j)$. The packets are denoted by P_i . The arrival time of a packet, A_i is defined as the time when the packet arrives at the transmit queue of the originating node. Each packet has a relative deadline D_i associated with it. The packet must be delivered to its destination by the time $A_i + D_i$. We associate a per-hop transmission time, C_i , with each packet. This transmission time is a function of raw bandwidth and characteristics of the physical medium. In terms of the effective bandwidth, W , C_i is the ratio of the packet size and W .

The per-hop utilization is defined as C_i/D_i . A packet contributes this much to the overall utilization of every node in its path from time A_i to $A_i + D_i$. Therefore, if $S_j(t)$ is the set of packets that are present in the network at some given time t , whose deadline has not expired yet and whose path includes node j , then the utilization of node j , denoted by U_j is defined as

$$U_j(t) = \sum_{P_i \in S_j(t)} C_i/D_i \quad (3.1)$$

Since WC_i is the packet size, $WU_j(t)$ gives the real-time capacity demand at node j . Therefore, the real-time capacity demand for the entire network is given by $W \sum_j U_j(t)$. If we obtain an upper bound on U_j such that if the node's utilization does not exceed U_j then all packets are guaranteed to meet their deadline, then no deadline miss would occur if the real-time capacity demand does not exceed

$W \sum_j U_j(t)$. In other words,

$$RTC = W \sum_j U_j(t), \quad (3.2)$$

where RTC stands for real-time capacity.

In the following, we shall denote the real-time capacity expressions for fixed priority and EDF scheduling of packets by RTC_{FP} and RTC_{EDF} respectively.

3.3 ASSUMPTIONS

For channel access, we assume an ideal MAC protocol, that is the MAC protocol transmits packets in priority order. The propagation speed in wireless medium is speed of light, and the distance scale is of the order of a few kilometers. Therefore, the propagation delay of the packets is negligible as compared to queuing and transmission delays. Accordingly, we ignore the propagation delay of packets in the following analysis. We do not assume circular radio range in the theoretical derivations. We, however, assume that the number of nodes that may be blocked due to a transmission is upper bounded, and that the bound is known.

We assume that the per-hop transmission time of a packet remains constant throughout the route of the packet. We use the assumption of large number of nodes to perform approximations of fixed priority real-time capacity limit expressions.

3.4 FEASIBILITY CONDITIONS

We consider transmission of a packet through a sequence of nodes, which we call the packet's path. We now derive a path-specific condition on meeting end-to-end deadlines as a bound on a function of utilization values. If the value of this function does not exceed the bound, then the packets transmitted along this path are guaranteed to meet their deadline.

In the following analysis, we use theorems for schedulability of tasks contending for a given (set of) resource(s). In the context of WSN, the resource is the channel

at the receiver node $j + 1$. Let $neighborhood(j)$ denote the set of nodes who contend for the transmission of node j . The broadcast nature of wireless medium allows only one node of this to transmit at a time. Therefore, all packets scheduled for transmission in $neighborhood(j)$ form one single virtual queue whose total utilization is equal to

$$V_j = \sum_{i \in neighborhood(j)} U_j, \quad (3.3)$$

where V_j is the neighborhood utilization.

Let m be upper bound on the size of $neighborhood(j)$ for all j . Then, $\sum_j V_j \leq m \sum_j U_j$. Therefore, (3.2) can be written as

$$RTC \geq W/m \sum_j V_j, \quad (3.4)$$

3.4.1 Corrections for priority inversion

In wireless networks, since no concurrent transmission can occur in the neighborhood of a *receiver*, the transmission of a packet can block another transmission up to two hops away. From the point of view of prioritized transmissions, this can lead to priority inversions of a unique kind [1]. For example, consider the case of a receiver receiving a high priority packet. No node in the neighborhood in the receiver can transmit simultaneously. If there is some lower priority packet sender at the periphery of the neighborhood that has packets for a receiver in the nearby neighborhood, then it gets blocked. If this latter transmission were the highest priority for the latter receiver, from the perspective of the receiver, this is a priority inversion. Since this phenomenon arises not due to waiting for a lower priority transmission, it is referred to as *pseudo priority inversion*.

The effect of delays due to pseudo priority inversion decreases the real-time capacity. We quantify the the effect of pseudo priority inversion as an increase in

neighborhood utilization by a factor $1 \leq \beta \leq 2$. Applying this correction to the expression for the real-time capacity (3.5) gives,

$$RTC \geq \frac{W}{m\beta} \sum_j V_j, \quad (3.5)$$

3.4.2 Fixed Priority Scheduling

Let us consider some packet P_n ¹. Let L_j denote the time between the arrival of the last bit of P_n at node j and departure of its last bit from the same node. In other words, L_j is the packet's delay at node j .

The following theorem proved by Abdelzaher *et al.* gives the bound on the delay of a task as a function of utilization for time-independent fixed priority scheduling algorithms:

Theorem 3.1 (The Stage Delay Theorem [2]) *If task T spends time L_j at resource j , and u_j is a lower bound on the maximum utilization at that hop, then:*

$$L_j = \frac{u_j(1 - u_j/2)}{1 - u_j} D_{max} \quad (3.6)$$

where D_{max} is the maximum end-to-end deadline of all tasks of higher priority than T .

This theorem applies to a very general class of resources, where the access to the resource is granted to only one task.

Correspondingly for wireless sensor networks, from the stage delay theorem,

$$L_j = \frac{V_j(1 - V_j/2)}{1 - V_j} D_{max}. \quad (3.7)$$

In other words, if the neighborhood utilization of node j does not exceed V_j , then the packet delay at that node does not exceed L_j . The end-to-end bound on the delay is obtained by simply summing over all nodes:

¹This section is based on [1]

$$\sum_{j=1}^N L_j = \sum_{j=1}^N \frac{V_j(1 - V_j/2)}{1 - V_j} D_{max}, \quad (3.8)$$

where N is the number of hops on the packet's path. For the packet to meet its deadline, the end-to-end delay must not exceed its relative deadline, D_n . Therefore,

$$\sum_{j=1}^N \frac{V_j(1 - V_j/2)}{1 - V_j} D_{max} \leq D_n \quad (3.9)$$

Rearranging the terms, we get:

$$\sum_{j=1}^N \frac{V_j(1 - V_j/2)}{1 - V_j} \leq D_n/D_{max} \quad (3.10)$$

To obtain a sufficient bound, the ratio of the deadline of packet P_n to that of a higher priority packet that delays its transmission, D_n/D_{max} must be minimized over all possible cases. Let us denote the minimum of this ratio for a given fixed priority scheduling policy as α . Deadline monotonic algorithm assigns priority in the order of decreasing relative deadlines. Thus, by definition, $\alpha = 1$ for deadline monotonic scheduling algorithm. Clearly, this ratio can not be larger than 1. Therefore, deadline monotonic scheduling algorithm gives the maximal bound:

$$\sum_{j=1}^N \frac{V_j(1 - V_j/2)}{1 - V_j} \leq 1 \quad (3.11)$$

3.4.3 EDF Scheduling

We use Theorem 2.14 to derive schedulability bound on neighborhood utilization for EDF. Similar to the derivation for fixed priority scheduling algorithms presented in the previous section, we consider transmission of a packet along some path whose nodes are labelled $1 \dots N$. Since V_j is the neighborhood utilization of each node j on this path, the total utilization on the path is $\sum_{j=1}^N V_j$. From Theorem 2.14, all packets on this path are schedulable provided

$$\sum_{j=1}^N V_j \leq 1. \quad (3.12)$$

3.5 CAPACITY LIMITS

In this section, we present the analysis of real-time capacity limits for Deadline Monotonic and Earliest Deadline First scheduling algorithms.

3.5.1 Load balanced traffic

Load balanced traffic refers to the traffic pattern where every node has the same node utilization. Equivalently, every neighborhood utilization becomes the same. More precisely, every neighborhood utilization equals the lowest bound of all neighborhood utilizations, determined by the upper bound on neighborhood size m . We denote this common bound on neighborhood utilization by V . Let N be the upper bound on the path length. Therefore, from (3.11),

$$\begin{aligned} \sum_{j=1}^N \frac{V(1 - V/2)}{1 - V} &\leq 1 \\ \Rightarrow \frac{V(1 - V/2)}{1 - V} &\leq 1/N \end{aligned} \quad (3.13)$$

Solving for V gets:

$$V = 1/N + 1 - \sqrt{1/N^2 + 1} \quad (3.14)$$

From (3.5),

$$RTC_{FP} = \frac{nW}{m\beta} (1/N + 1 - \sqrt{1/N^2 + 1}), \quad (3.15)$$

where n is number of nodes in the network.

Now we obtain the real-time capacity for EDF. From (3.11),

$$\sum_{j=1}^N V \leq 1$$

$$\Rightarrow V \leq 1/N \quad (3.16)$$

From (3.5),

$$RTC_{EDF} = \frac{nW}{mN\beta} \quad (3.17)$$

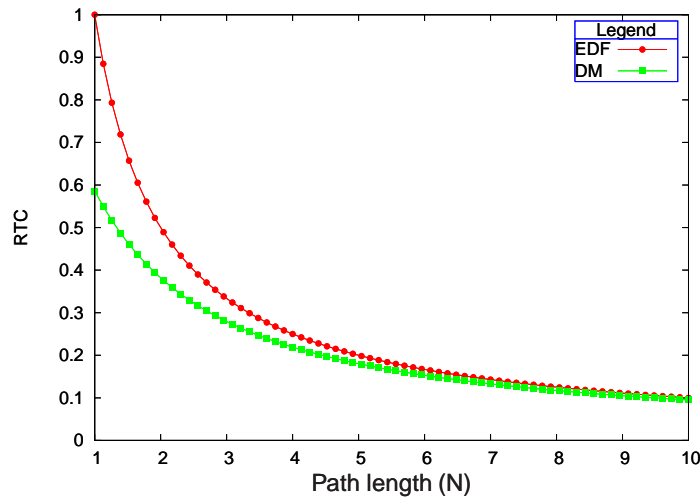


Figure 3.1 Real-time capacity (Normalized in the units of nW/m)

Figure 3.1 shows a comparison of the real-time capacity limit expressions for DM and EDF. The figure shows that the real-time capacity limit for DM approaches that for EDF as the path length increases. This behavior of the curves can be explained as follows: as paths get larger, $1/N^2 \rightarrow 0$. Therefore, $RTC_{FP} \rightarrow \frac{nW}{mN\beta} = RTC_{EDF}$.

3.5.2 Convergecast traffic

Convergecast refers to the traffic pattern in which nodes transmit data to a common data aggregation point or base-station (Figure 3.2). There may be a number of data aggregation points distributed in the network. In such a case, the traffic is destined to the nearest data aggregation point. The data aggregation points form

the bottleneck. The nodes in the neighborhood of the data aggregation points are loaded the most. In the following, we derive an approximate expression for real-time capacity limit for convergecast.

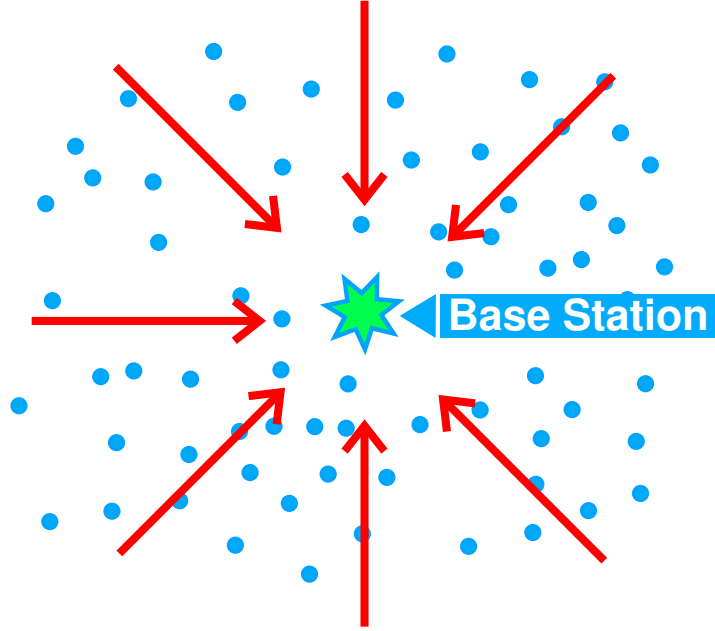


Figure 3.2 Convergecast traffic

Let K be the number of data aggregation points in the network. We consider the traffic sent to an arbitrary data aggregation point k . Let N_k be largest path length of the packets destined to k . Since all traffic ends up at the data aggregation point k , at steady state, the total amount of traffic generated for k must equal the total amount that can be delivered to k . Let $m/\pi R^2$ be the average density of nodes in the area of one radio range R . The number of nodes located j hops away from the data aggregation point is approximated by the product of the area of the ring $\pi j^2 R^2 - \pi(j-1)^2 R^2$ and the node density $m/\pi R^2$ which is equal to $(2j-1)m$ nodes. Therefore, the average node utilization, and hence neighborhood utilization, decreases with distance from the aggregation point. If we denote the utilization at the data aggregation point to be V , then the utilization of the j^{th} hop nodes is $V_j = V/(2j-1)m$.

Therefore, from the feasibility condition for DM (3.11),

$$\begin{aligned}
& \sum_{j=1}^{N_k} \frac{V_j(1 - V_j/2)}{1 - V_j} \leq 1 \\
\Rightarrow & \sum_{j=1}^{N_k} \frac{\frac{V}{(2j-1)m} \left(1 - \frac{V}{(2j-1)2m}\right)}{1 - \frac{V}{(2j-1)m}} \leq 1 \\
\Rightarrow & \sum_{j=1}^{N_k} \left\{ \frac{V}{(2j-1)m} \frac{2(2j-1)m - V}{2(2j-1)m - V} \right\} \leq 1 \\
\Rightarrow & \sum_{j=1}^{N_k} \left\{ \frac{V}{2(2j-1)m} \left(1 + \frac{(2j-1)m}{(2j-1)m - V}\right) \right\} \leq 1 \tag{3.18}
\end{aligned}$$

Inequality (3.18) is nonlinear in V . We present a closed form expression that approximates the exact solution below. From (3.5),

$$RTC_{FP} = KWN_k V/\beta, \tag{3.19}$$

where V is solution of (3.18).

Now we obtain the real-time capacity for EDF. From (3.11),

$$\begin{aligned}
& \sum_{j=1}^{N_k} V_j \leq 1 \\
\Rightarrow & V/m \sum_{j=1}^{N_k} \frac{1}{2j-1} \leq 1 \tag{3.20}
\end{aligned}$$

To obtain the sum $\sum_{j=1}^{N_k} \frac{1}{2j-1}$, we note that the Euler's constant $\gamma (\cong 0.577)$ is given by

$$\gamma = 1 + 1/2 + 1/3 + \dots + 1/x - \log x, \tag{3.21}$$

where the series approximation gets better as $x \rightarrow \infty$. Let $S = \sum_{j=1}^{N_k} \frac{1}{2^{j-1}}$, and let $S' = \sum_{j=1}^{N_k} \frac{1}{2^j}$. From (3.21), $S' = 1/2(\gamma + \log N_k)$. Furthermore,

$$\begin{aligned} S + S' &= 1 + 1/2 + 1/3 + \dots + 1/2N_k \\ &= \gamma + \log 2N_k \end{aligned} \quad (3.22)$$

Therefore,

$$\begin{aligned} S &= \gamma/2 + \log 2N_k - 1/2 \log N_k \\ &= \gamma/2 + \log 2 + 1/2 \log N_k \\ &\cong 1 + 0.5 \log N_k \end{aligned} \quad (3.23)$$

We get V by plugging (3.23) into (3.20) as:

$$V = \frac{m}{1 + 0.5 \log N_k} \quad (3.24)$$

From (3.5),

$$RTC_{EDF} = \frac{KW N_k}{\beta(1 + 0.5 \log N_k)} \quad (3.25)$$

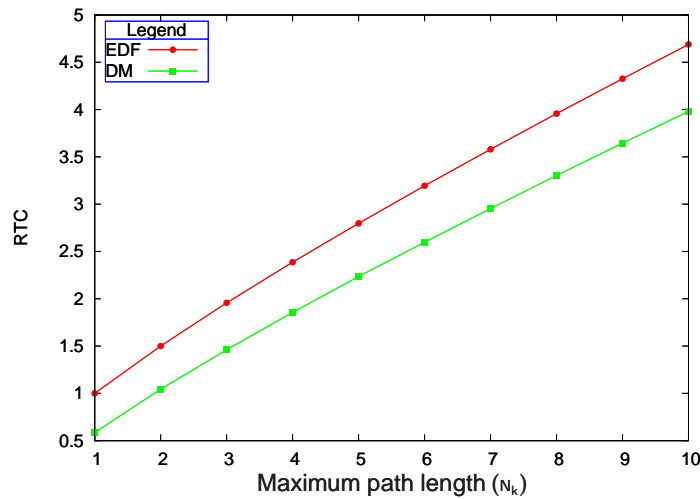


Figure 3.3 Real-time capacity for convergecast traffic (normalized in the units of KW)

Figure 3.3 presents a graph of the real-time capacity limit for DM and EDF scheduling algorithms. The values were obtained using numerical solution of the capacity expressions derived above. The plot shows that the relative difference in the capacity expressions get diminished as N_k increases. This can be reasoned by noting that in (3.18), the second term can be approximated by unity, given that $V < 1$. Upon this approximation, (3.18) becomes same as the bound on V for EDF (3.20).

From (3.5),

$$RTC_{FP} \approx \frac{KWN_k}{\beta(1 + 0.5 \log N_k)} \quad (3.26)$$

3.5.3 Convergecast vs. Load-balanced Traffic

In the following, we compare the real-time capacity limits obtained for load balanced traffic and convergecast traffic. For the purpose of comparison, we use the expressions for EDF since the approximate closed form expressions for DM resemble that for EDF. Let RTC^{LB} and RTC^{CC} be the real-time capacity limits for the load balanced and convergecast cases respectively. Then, for the similar path lengths, namely $N = N_K$, from (3.17) and (3.25), we have,

$$\frac{RTC^{LB}}{RTC^{CC}} = \frac{n(1 + 0.5 \log N)}{KmN^2}. \quad (3.27)$$

Since mN^2 is approximately the number of nodes inside one aggregation point domain, $mN^2 \cong n/K$. Therefore,

$$\frac{RTC^{LB}}{RTC^{CC}} \cong 1 + 0.5 \log N. \quad (3.28)$$

This ratio becomes 1 when $N = 1$. This is to be expected since each transmission excludes $m - 1$ nodes of the neighborhood in both traffic topologies. For larger values of N , the capacity limit for load balanced case is larger than that for the convergecast since the absence of bottleneck in the earlier case allows the reception of larger amount of traffic at the destination per unit time. The gain, however, is only

logarithmic. The reason for this phenomenon is that for the load balanced traffic, the intersection of different flows, and hence interference grows with the path size. But, for the convergecast traffic, the traffic flow is radial, and hence the interference is not much of an issue at larger hop distances from data aggregation points. As we shall see in the next chapter, the streamlined flow of packets in the convergecast case can be leveraged to increase the parallelism of data transmission, and accordingly much higher real-time capacity can be obtained.

3.6 EVALUATION

We implemented a simulator to evaluate the pessimism in the capacity expressions. The simulator constructs a network of sensor nodes of a user-specified size in a perturbed grid structure. The radio layer is implemented as a simplified disk model of a specified radius (range). The sinks are distributed uniformly across the network. We generated traffic at each non-sink node such that each packet was assigned a deadline at random from a preselected set. All packets were sent to their nearest sink. Packet contention was resolved in priority order. Only those nodes were allowed to transmit who were not within the radio range of another node that was already scheduled to receive a transmission. Ties between simultaneously arriving same priority packets were broken at random. We implemented a shortest path routing scheme in which the neighboring node nearest to the sink was chosen as the next hop. If this node was blocked due to another transmission, the packet was not scheduled until that transmission was over. The MAC layer implements deadline monotonic prioritization for medium arbitration.

All packets were checked for deadline misses. If there was a miss, the actual capacity consumption of all in-transit traffic was computed by multiplying each in-transit packet by the traversed hop count and normalizing by the end-to-end deadline. Each run was repeated 50 times with different randomized workloads. The

minimum capacity consumption at which a deadline miss occurred was recorded, and is shown on the plots as *critical capacity*.

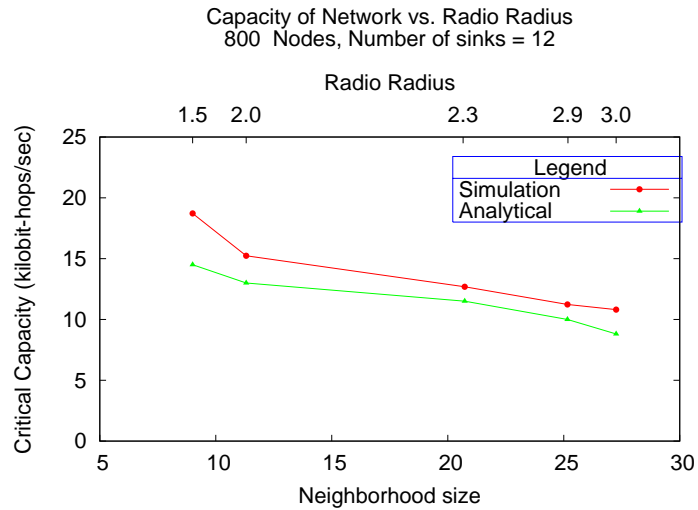


Figure 3.4 Effect of radio radius, 800 nodes

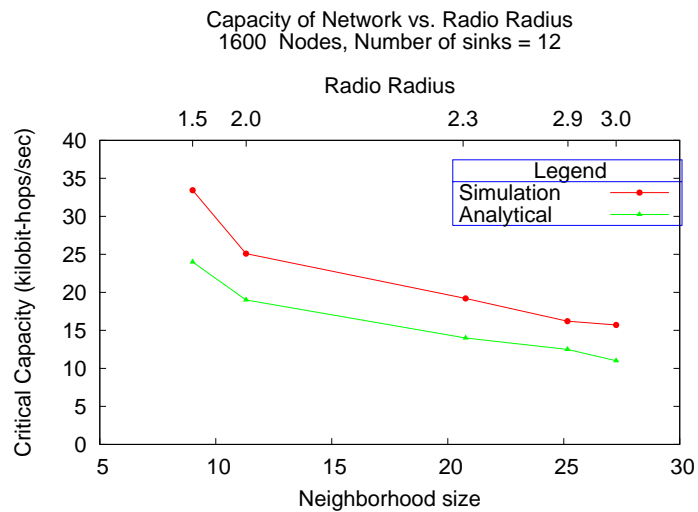


Figure 3.5 Effect of radio radius, 1600 nodes

Figures 3.4 and 3.5 show the effect of increasing the radio radius, shown on the top horizontal axis, on real-time capacity in a network of 800 nodes and 1600 nodes respectively. Observe that increasing the radio radius also increases the neighborhood size (i.e., the number of nodes within the radio range), shown on the bottom

horizontal axis. The number of sinks was kept at 12. The lower curve in both figures is the analytic capacity bound computed from Equation 3.26. This equation accounts for priority inversion. Parameters α and W are set to 1. The top curve shows the minimum consumed capacity at which deadline misses were observed in simulations. Note the very close match between simulation and analytic prediction even at very large network sizes. As expected, capacity decreases with increasing radio radius because the latter increases the effective density.

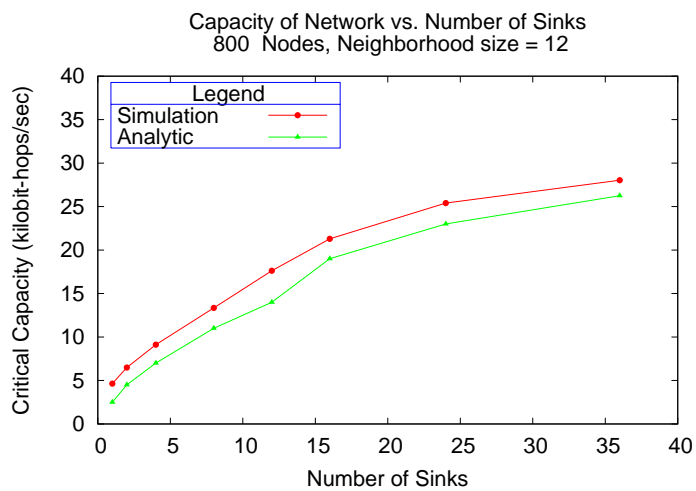


Figure 3.6 Effect of the number of sinks, 800 nodes

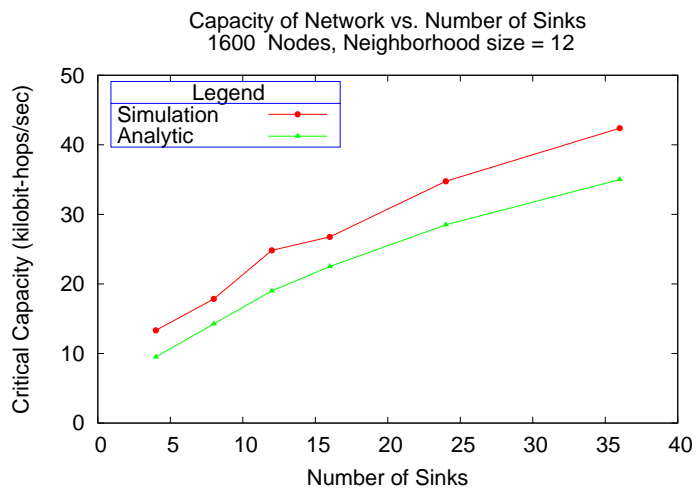


Figure 3.7 Effect of the number of sinks, 1600 nodes

Figures 3.6 and 3.7 repeat the experiments for networks of 800 and 1600 nodes respectively, this time varying the number of sinks. The radio range is kept constant at a neighborhood size of 12 nodes. As before, a very close match is observed between simulation and analysis. Capacity grows with the number of sinks because data collection bottlenecks are alleviated.

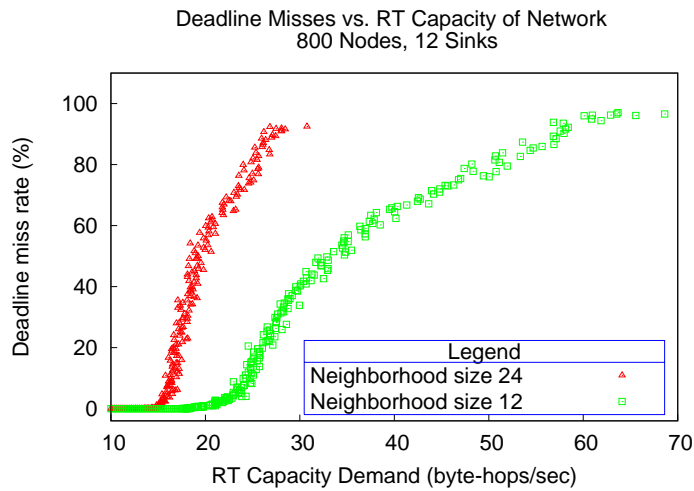


Figure 3.8 Miss rate as a function of real-time capacity demand

Finally, Figure 3.8 shows the sharp increase in the miss ratio in a network of 800 nodes that occurs when capacity is exceeded. In this curve, the network workload is increased past the capacity bound. The miss ratio is then plotted against the capacity requirements of the workload shown on the horizontal axis. Each point in the figure corresponds to a single experiment. Two sets of data points are shown for two different radio ranges that correspond to neighborhoods of 12 nodes and 24 nodes respectively. From Figure 3.4, we can see that the capacity bounds for these two cases are around 13 and 12 respectively. The miss ratio becomes non-zero shortly after these bounds are exceeded and increases sharply soon thereafter.

3.7 LIMITATIONS

Irregularities in node placement will affect the accuracy of the results derived in this chapter. The capacity expressions depend on the upper bound on the neighborhood size, that is the upper bound on the number of nodes that may be blocked when a transmission takes place. The results obtained here will get pessimistic if the upper bound on the neighborhood size deviates excessively from the average.

Another source of pessimism results from irregularity of radio range. If the radio range is so irregular that the maximum path length deviates too much from the average, then also, the expressions will not represent the true real-time capacity. We make no assumptions about in-network data aggregation. The results presented here remain valid even if in-network data aggregation is performed, except that the capacity expressions refer to the aggregated data packets.

3.8 SUMMARY

In this chapter, we presented real-time capacity limit expressions under the sufficient schedulability condition. We derived limiting expressions for large networks. We saw that under such limiting circumstances, the real-time capacity limits for deadline monotonic and earliest deadline first scheduling algorithms tend to agree on a common value. We showed that for convergecast, the real-time capacity increases as the square root of the number of nodes. This implies existence of diminishing return and hence favors multiple smaller networks as opposed to a large one. We find that the real-time capacity obtained analytically agrees with that obtained using simulation within 30-40% in general. The agreement gets better as the effective density increases.

CHAPTER 4

Hexagonal Wireless Ad-Hoc Networks

4.1 INTRODUCTION

We considered the real-time capacity limit of WSN previously. For analysis, we assumed the existence of MAC layers that transmit packets in EDF or DM priority order. In this chapter, we present TDMA based MAC protocol that gives deterministic bandwidth guarantees, assuming that no fault occurs. We show that the real-time capacity of the protocol is higher than that for EDF and DM. More importantly, the MAC protocol presented here has zero scheduling overhead.

The problem of packet collisions in shared medium is alleviated by either sensing the medium for possible ongoing transmissions and avoiding collisions, or by carefully timing (scheduling) the transmissions so that no collision occurs. The TDMA based collision avoidance algorithms that attempt to *schedule* transmissions are referred to as *transmission scheduling algorithms*. While contention based collision avoiding algorithms offer a simpler and more dynamic solution to the medium access problem, schedule based algorithms can provide deterministic service delay bounds. It is crucial for real-time applications that the transmission delays be known and bounded.

However, Arikan proved that the problem of scheduling in single frequency multi-hop wireless networks to meet specified origin-destination packet rate is NP-Hard [7]. Coffman *et al.* showed that the problem of scheduling file transfers in a distributed network such that the transfer is completed in minimum amount of time is NP-Complete [39]. These results imply that developing a decentralized low overhead TDMA based MAC protocol for arbitrary topology WSN is infeasible (unless

P=NP) for either specified bandwidth requirement or minimum end-to-end transmission time. Therefore, we focus on developing TDMA based MAC protocol for regular topology WSN, namely hexagonal WSN.

As an added benefit, imposing a regular communication topology on networks affords simple but efficient network protocols. Among the well known regular topologies (simple paths, trees, hypercubes etc.), hexagons offer two very desirable properties in a distributed communication system – namely, constant node degree and good bisection width. In sufficiently dense wireless ad-hoc networks, it is possible to adjust the transmission power of radio units to achieve certain connectivity patterns. There exist techniques known as *topology control* that attempt to achieve specified connectivity objective with minimum energy consumption (see [58] for a survey).

The use of hexagonal meshes has already been reported in previous literature. The HARTS system employs processors connected using a hexagonal *torus* network topology [62]. As the consequence of regularity afforded by the hexagonal torus topology, the addressing and resulting routing and broadcasting algorithms are not only simple but efficient as well [18]. An example of simplification of routing algorithms gained due to hexagonal topology in cellular networks can be found in [49]. Afforded by the techniques of ad-hoc networks topology control, we consider wireless ad-hoc networks of hexagonal topology, and show that for such networks, conflict free transmission schedules can be constructed without any message overhead for this purpose. We chose hexagonal topology because it allows optimal spatial reuse in a natural manner (see the section on scheduling). The packet scheduling algorithm presented here can be used on top of a Carrier Sense Multiple Access/ Collision Avoidance (CSMA/CA) MAC protocol to *emulate* a TDMA protocol; or it can be assimilated in TDMA or hybrid CSMA/TDMA MAC protocols.

To enable conflict-free transmission schedule in distributed systems, clock synchronization is essential. Relative synchronization of local clocks suffice for the purpose of transmission scheduling. Using the information contained in data packets, we present a clock synchronization algorithm that has zero message overhead. The protocol relies solely on overhearing neighbors' data transmissions to synchronize clocks and is shown to converge quickly to a common time. We published the results of this chapter in [52].

The rest of this chapter is organized as follows. We present related work in Section 4.2, followed by the description of the system model in Section 4.3, and our assumptions in Section 4.4. We present our addressing, routing, scheduling, and clock synchronization algorithms in Section 4.5, and present the real-time capacity of our transmission scheduling algorithm in Section 4.5.5. We present simulation results in Section 4.6. We present the real-time implications of the work presented in this chapter in Section 4.7, and the limitations of the results in Section 4.8. Finally, we present the chapter summary in Section 4.9.

4.2 RELATED WORK ON HEXAGONAL TOPOLOGY NETWORKS

Chen *et al.* present addressing and routing algorithms for (wired) multi-processors connected in hexagonal torus topology [18]. Nocetti *et al.* adapt and simplify this addressing scheme for hexagonal cellular networks [49]. The addressing schemes proposed by Nocetti *et al.* and Chen *et al.* use three axes to determine coordinates (and hence addresses) of nodes in a plane. Consequently, the addressing scheme of Nocetti *et al.* [49] has the downside of inability to assign a unique address to the nodes (Chen *et al.* [18] avoid this problem by imposing an additional ordering constraint). Carle *et al.* [16] and Decayeux *et al.* [22] assign unique coordinates to the nodes of a planar hexagonal network using only two axes. Some topological properties of hexagonal networks are presented in all the above mentioned papers. In this chapter, we present yet another addressing scheme based on the radial symmetry

of the many-to-one (convergecast) communication topology that yields simple algorithm for scheduling. We also outline a transformation to map the new addresses to coordinates in a frame of two axes aligned at 120 degrees as in the work of Carle *et al.* [16]. Elson *et al.* present clock synchronization algorithms in wireless sensor networks in [24]. A survey of such algorithms can be found in [65].

4.3 SYSTEM MODEL

We consider multi-hop transmission of data packets in wireless ad-hoc networks over single frequency shared radio channel. We consider convergecast traffic. Convergecast refers to the communication topology where nodes transmit packets to a common sink node, also called base-station or aggregation point. For densely deployed wireless sensor networks, we assume a two-tiered cluster-based network topology. At the first tier, all active (nodes that are not in power-save mode) nodes including the cluster heads sense data which is collected by the cluster heads. At the second tier, cluster heads send and route data packets to their destination. This cluster-based two-tier topology is similar to the original version [8] but with the exception that the cluster heads also assume the role of gateway nodes. Similar to the LEACH [34] architecture, we assume that nodes belonging to one cluster use CDMA to transmit data to the cluster head, and use appropriate radio power level to do so. Cluster heads maintain code distribution and transmission schedules of the active sensor nodes in their cluster. Thus, transmissions to different cluster heads can occur simultaneously. For inter-cluster communication however, we do not assume CDMA transmissions since this approach is not scalable for multi-hop transmissions. We shall focus on constructing schedules for conflict-free inter-cluster communication in the rest of this chapter, and shall not dwell on intra-cluster transmissions anymore.

We consider nodes (cluster heads) of the network to be connected in regular hexagonal pattern. Only those nodes that are connected by an edge can hear some

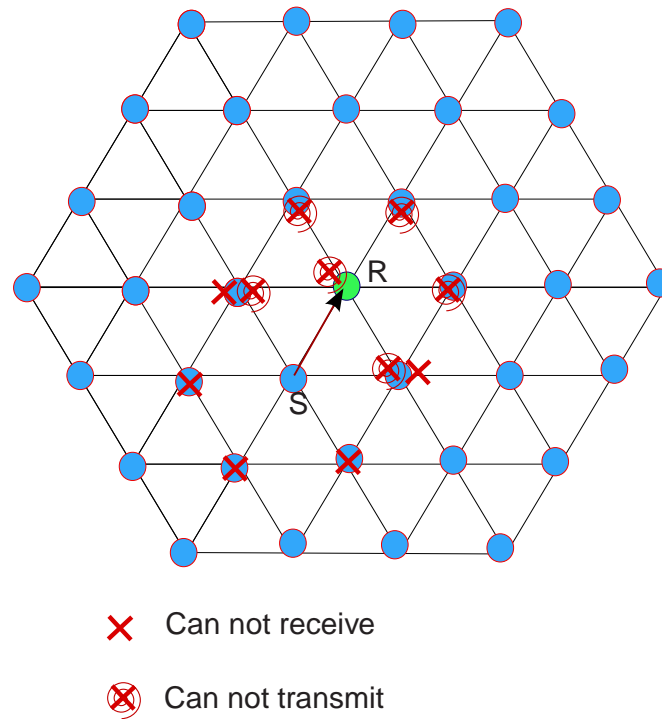


Figure 4.1 Interference in hexagonal networks

node's transmission. Appropriate node placement at deployment time and topology control algorithms at run-time can ensure the abstraction of a hexagonal network topology. While this logical topology might remain a slight over-simplification of the underlying actual connectivity, we show in the evaluation section that our algorithm is surprisingly robust to deviations of the actual topology from the hexagonal mesh.

Figure 4.1 illustrates the interference model used in this chapter. Let us consider a transmission from node S to R shown in the figure. In the neighborhood of these two nodes, the one-hop neighbors of S can not receive any transmission, and that of R can not transmit any packet successfully while the transmission $S \rightarrow R$ is taking place. Such nodes are indicated with an \times and an \times with a spiral in the figure.

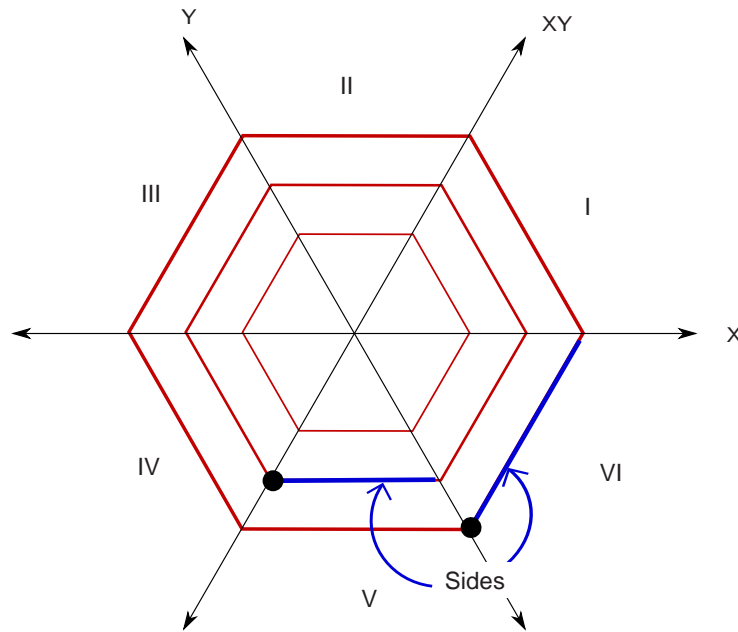


Figure 4.2 Hexants

Hexants and Sides

Of the three principal diagonals, we select a pair inclined at 120 degrees to be the axes, labelled as X and Y (Figure 4.2.) Let XY denote the line bisecting the X and Y axes. These three lines drawn through the origin divide the plane into six regions. We refer to these six regions as *hexants*. Each of the six hexants are marked with roman numerals in the figure.

Consider a set of concentric hexagons. A hexagon ring at distance h edges from the origin contains $6h$ nodes. We define the set of h consecutive nodes contained within any hexant, inclusive of one node on one of the diagonals, as the nodes on one *side* at distance h . In this chapter, we adopt the convention of including the first diagonal in the anti clockwise sense to the definition of a side.

4.4 ASSUMPTIONS

In the following we list our assumptions. Many of the assumptions listed below have been mentioned in earlier sections. We re-list them here for the reader's convenience.

We assume that the system does not have faults. The real-time guarantees hold when packet losses do not occur. We assume a cluster based topology. All active nodes in a cluster report sensed data to the cluster heads. The cluster heads process this data (compress, eliminate redundancy etc.) and transmit to the base-station. We assume fixed packet size and constant bandwidth. We assume that all packets sent to the base-station are routed by the cluster heads. We assume sufficient node density to form hexagonal topology. For interference, we assume that those and only those nodes that are connected to the sender by an edge can hear its transmissions.

4.5 SCHEDULING FOR CONVERGECAST

In this section, we present addressing, routing, clock synchronization and distributed transmission scheduling algorithms for wireless ad-hoc networks of hexagonal network topology. The scheduling algorithm creates bounded latency TDMA schedules with spatial reuse. We consider convergecast traffic. The aggregation point is thus the bottleneck for this communication topology.

4.5.1 Addressing

For convergecast, the arrangement of nodes can be seen as that of concentric hexagons centered at the aggregation point, where the neighboring hexagons are separated by one hop. Due to this symmetry, we choose the aggregation point to be the origin. In such an arrangement, all nodes on any given concentric hexagon are equidistant from the origin. We assign addresses of the form $[h, i]$ to the nodes, where h is the shortest hop-count of the node from the origin and i denotes the

index of a node located on the hop- h hexagon. The index starts at the x -axis and increases in the counter-clockwise direction. Hence the first hop nodes are addressed as $[1, 0], [1, 1] \dots [1, 5]$. For brevity, we use $[h, i]$ to refer to a node as well as its address. Observe that nodes of the form $[h, \cdot]$ are all located on the same hexagonal ring at distance h from the origin (· denotes wildcard). Since the number of nodes on h^{th} hop hexagon is $6 \times h$, the node addresses range from $[h, 0]$ to $[h, 6h - 1]$ (see Figure 4.5 for an example).

Oblique Coordinates and Transformations

Now, we present a transformation from addresses of the form $[h, i]$ to the coordinates of the nodes in oblique Cartesian system. As we shall see, the use of Cartesian coordinates makes calculation of distance between any pair of nodes simple.

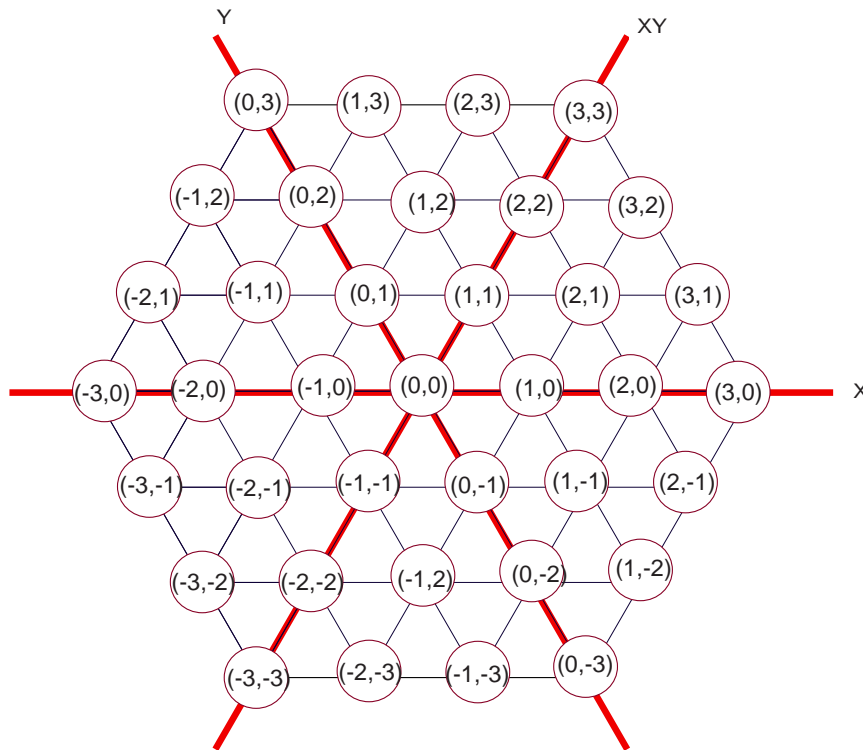


Figure 4.3 Oblique coordinate system for hexagons

We choose one of the principal diagonals as the X axis and the other inclined at 120 degrees to be the Y axis. An example of assignment of coordinates in this system

is shown in Figure 4.3. To transform $[h, i]$ to (x, y) , we use the equations for hextants, Q (4.2) and K (4.5), which are reproduced here for the reader's convenience.

$$Q = \left\lfloor \frac{i}{h} \right\rfloor, K = i - \left\lfloor \frac{i}{h} \right\rfloor h.$$

The transformation rules are as follows:

Q	Transformation
0	$[h, i] \Rightarrow (h, i)$
1	$[h, i] \Rightarrow (h - K, h)$
2	$[h, i] \Rightarrow (-K, h - K)$
3	$[h, i] \Rightarrow (-h, -K)$
4	$[h, i] \Rightarrow (K - h, -h)$
5	$[h, i] \Rightarrow (K, K - h)$

Table 4.1 Transformation rules

Inverse transformation can be obtained similarly.

Theorem 4.1 *The distance between nodes (x_1, y_1) and (x_2, y_2) is given by $\text{MAX}\{|x_1 - x_2|, |y_1 - y_2|, |x_1 - x_2 - y_1 + y_2|\}$.*

Proof of this theorem is similar to that of Theorem 2 of [22].

4.5.2 routing

For routing, we use the following algorithm. All the nodes falling on the X , Y or XY axes route packets along the straight line joining them to the origin (Figure 4.4). All other nodes route packets as follows: in hextants I and IV, packets are routed parallel to the XY -axis towards the origin, in hextants II and V, packets are routed parallel to the Y -axis towards the origin, in hextants III and VI, packets are routed parallel to the X -axis towards the origin (see Figure 4.5 for an example). Once the packet reaches one of the diagonals, it is routed along the diagonal to the base-station. In other words, all non-diagonal nodes on a given side route towards the diagonal at 60 degrees. This algorithm keeps the traffic flow in all regions of the network nearly balanced, and routes the packets along a shortest path.

VAR: $q = \lceil i/h \rceil$
 Route: $[h, i] \Rightarrow [h - 1, i - q]$

Algorithm 2: Routing

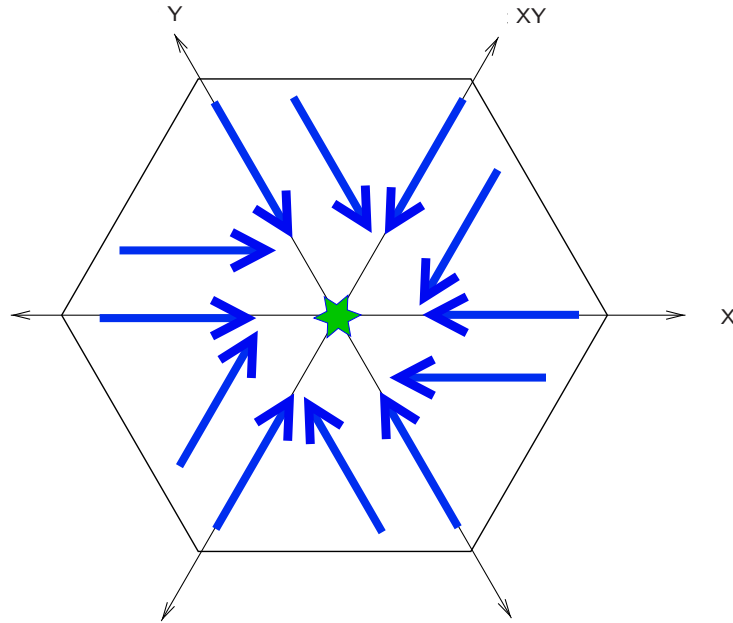


Figure 4.4 Routing in hexagonal mesh

4.5.3 Scheduling

In the following, we derive closed form expressions for construction of cyclic TDMA schedule with spatial re-use such that:

- ◆ during each cycle, every node can send one locally originated packet to the sink
- ◆ by the end of each cycle, all such packets are received at the sink

Let H be the radius (the maximum shortest distance hop-count of any node to the aggregation point) of the hexagonal network. Since the number of nodes on the h^{th} hop hexagon is $6h$, the total number of nodes (excluding the sink node) is given by $\sum_{h=1}^H 6h = 3H^2 + 3H$. Therefore, the total number of packets that must be received by the sink every cycle is $3H^2 + 3H$. Since the sink node can receive only one packet at a time, a minimum of $3H^2 + 3H$ time slots are needed to transmit all of these

packets to the sink, where each time slot is assumed to be large enough to transmit one packet over one hop (packets are assumed to be of the same size).

The scheduling algorithm employs spatial reuse – during every time slot, up to H transmissions are scheduled. Additionally, the transmissions are scheduled in such a way that packets flow continuously to the sink node, and by the end of $3H^2 + 3H$ time slots, the sink receives all packets.

4.5.3.1) *Intuition into Scheduling*

Of the six hexants, no two odd hexants have any edge in common. Similarly, all three even hexants do not have any edge in common. Therefore, except for the last hop transmissions to the sink node, any transmission in hexant I can not interfere with any other transmission in hexants III and IV. Furthermore, transmission(s) on a given side do not interfere with transmission(s) on the sides that are 3 hops or more apart – even if in the same hexant. We accomplish the goal of conflict free scheduling with spatial reuse into two steps. First, we divide the nodes into six disjoint sets where, except for the neighboring nodes on a side, no two nodes have an edge in common. And in the second step, we allocate time slots to the nodes such that the neighboring nodes of a given side transmit in separate time slots.

Formally speaking, we partition¹ the nodes of the network into six subsets such that all nodes of the form $[h, \cdot]$ in any of the subsets do not interfere with any other node $[h', \cdot]$ of the same subset where $h \neq h'$. For illustration, let us consider a three hop network (Figure 4.5). Nodes $[1, 0]$, $[2, 4]$, $[2, 5]$ and $[3, 12]$, $[3, 13]$, $[3, 14]$ belong to the three sides of odd hexants (which implies that the three diagonal nodes on the sides are mutually separated by 120 degrees.) In this arrangement, none of the second hop nodes $[2, 4]$, $[2, 5]$ interfere with any of the first or the third hop nodes. To generalize, starting at any of the first hop node $[1, i]$, we choose all nodes on the side

¹Partition of a set is defined as the disjoint set of its subsets such that the union of the subsets is the set.

at 120 degrees of the next concentric hexagon, where the angle is measured between the diagonals. This partitioning procedure ensures that any two set of nodes located at two different hops but in the same hexant are at-least 3 hops apart, and thus do not interfere. For example, in a 4-hop network, the set of nodes of the same partition as $[1, 0]$ and falling into the same hexant are $[4, 0]$, $[4, 1]$, $[4, 2]$, $[4, 3]$. The nearest node to $[1, 0]$ is $[4, 0]$ which is 3 hops away.

However, being neighbors, not all nodes on any given side may transmit simultaneously. For example, in Figure 4.5 not all of $[2, 4]$, $[2, 5]$ or $[3, 12]$, $[3, 13]$, $[3, 14]$ can transmit simultaneously without conflict. We structure TDMA schedule cycles as a series of sub-cycles where each sub-cycle consists of six time slots. We schedule the neighboring nodes in successive sub-cycles, hence in different time slots. For example, the set of nodes corresponding to a valid schedule for this partition is $\{\{[1, 0], [2, 4], [3, 12]\}, \{[1, 0], [2, 5], [3, 13]\}, \{[1, 0], [2, 4], [3, 14]\} \dots \{[1, 0], [2, 4]\} \dots \{[1, 0]\}$. This example is a 3-hop network. Hence it is sufficient to allocate only one time slot to all nodes at the third hop since each originate one packet and route none. It suffices to allocate two slots to $[2, 5]$ since it originates one and routes one. Similarly, it suffices to allocate three slots to $[2, 4]$ and fifteen time slots to the nodes at first hop respectively. Schedules for the rest five partitions can be constructed symmetrically. All the six subsets can then be interleaved to form a set of sub-cycles. Since there are $3H^2 + 3H$ time slots in one cycle, it contains $(H^2 + H)/2$ sub-cycles.

4.5.3.2) *Closed form expressions for scheduling*

We now derive a set of expressions that determine time slots for transmission as a function of the tuple $[h, i]$. First, we obtain an expression to determine the partition of the nodes and then we obtain expressions for the time slots when a node may transmit.

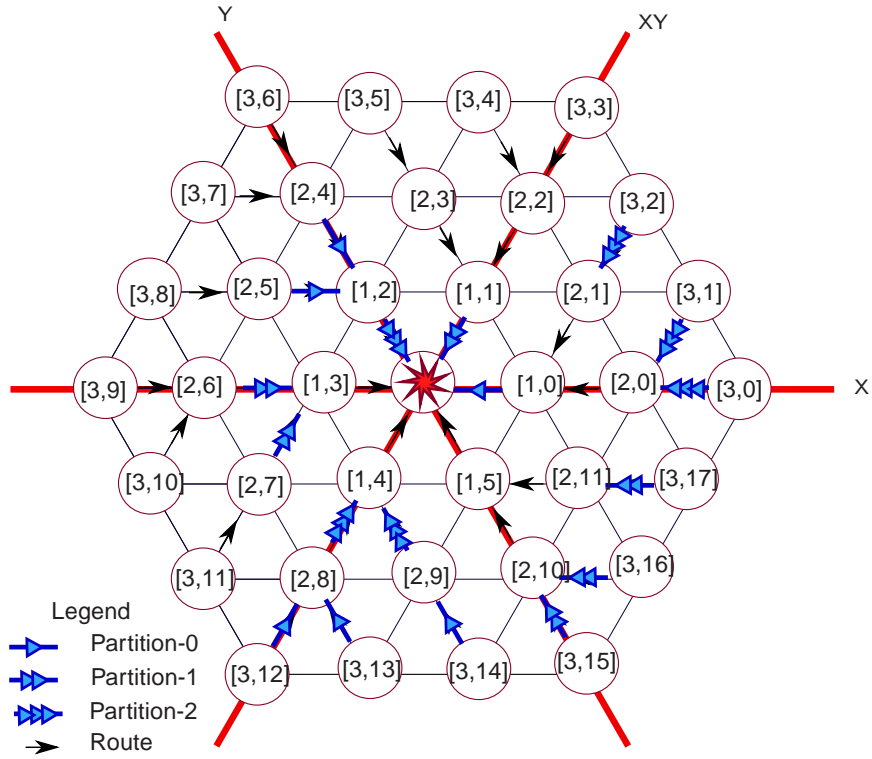


Figure 4.5 Example: Addressing, routing and partitions of a 3-hop network

Let R be defined as

$$R([h, i]) \triangleq (h - 1) \pmod{3}. \quad (4.1)$$

The variable R indicates spatial reusability. Nodes of different h but the same R are at-least 3 hops away. Let Q be defined as

$$Q([h, i]) \triangleq \left\lfloor \frac{i}{h} \right\rfloor. \quad (4.2)$$

Observe that $Q + 1$ gives the hextant of a node. Then the partition $P([h, i])$ of $[h, i]$ is given by

$$P = (Q - 2R) \pmod{6},^2 \quad (4.3)$$

where the arguments are omitted for the sake of brevity.

² $0 \leq P \leq 5$. According to modulus algebra conventions, if the residue is negative, P is made positive by adding 6.

The complimentary expression $P = (2R - Q) \pmod 6$ gives another partition where the partitions are in the clockwise order.

Lemma 4.2 *The expression $P = (Q - 2R) \pmod 6$ partitions h consecutive nodes of one side and no other node of an h^{th} -hop hexagon in the same subset.*

Proof We shall show that $P = (Q - 2R) \pmod 6$ assigns the consecutive nodes $\{[h, kh], \dots, [h, (k+1)h - 1]\}$ to one partition, where $0 \leq k < 6$ is an integer. Let us denote this set by $\mathcal{N} \triangleq \{[h, kh + j] | 0 \leq j < h\}$. From (4.1), $R(\mathcal{N})$ is the same for all nodes of the form $[h, \cdot]$, and hence is the same for all nodes in \mathcal{N} . From (4.2), $Q(\mathcal{N}) = \lfloor \frac{kh+j}{h} \rfloor = k$. Hence Q is also the same for all nodes in \mathcal{N} . Furthermore, $Q(\mathcal{N}) \neq Q(\mathcal{N}')$ if $k \neq k'$. Since k takes 6 unique values, this lemma follows. \square

Lemma 4.3 *If a partition p includes nodes of some side in hextant q , then, p contains nodes from the hextants $(q \pm 2) \pmod 6$. In other words, the diagonal nodes from neighboring hops in p are 120 degrees apart.*

Proof We first show that if some diagonal node $N = [h, kh]$, where $0 \leq k < 6$ is some integer, is in partition p , then another diagonal node $N' = [h+1, (k+2)(h+1) \pmod{6(h+1)}]$ is also in the same partition.

$$R(N) = (h - 1) \pmod 3$$

$$R(N') = h \pmod 3$$

$$Q(N) = k$$

$$Q(N') = (k + 2) \pmod 6 \tag{4.4}$$

Therefore, $P(N') - P(N) = 2 \pmod 6 - 2 \pmod 3 = 0$, or N and N' belong to the same partition. Similarly, $N'' = [h - 1, (k - 2)(h - 1) \pmod{6(h - 1)}]$ also belongs to the same partition as N . From (4.4), if N is in hextant q , then the other two are in

hexants $(q \pm 2) \pmod 6$. But our choice of N was arbitrary. Hence using Lemma 1, this lemma follows. \square

Theorem 4.4 *The expression $P = (Q - 2R) \pmod 6$ partitions the network such that nodes on any two different sides in the same partition do not interfere.*

Proof From Lemma 2, nodes that belong to the same partition are either in all odd or all even hexants. Since the diagonal nodes on each neighboring-hop sides are 120 degrees apart, it also implies that if two sides are the same hexant and in the same partition, they must be $3n$ hops apart, where $n > 0$ is an integer. Hence the proof. \square

The nodes falling on the principal diagonals have address of the form $[h, kh]$, where integer $0 \leq k \leq 5$ and that of the rest of the nodes is of the form $[h, kh + j]$ where integer $1 \leq j < h$. It is easily seen that packet accumulation on the nodes on the six principal diagonals, $[h, kh]$, is given by $(H - h + 1)(H - h + 2)/2$ and that on the other nodes $[h, i \neq kh]$ by $H - h + 1$. The nodes farther from the sink thus need smaller number of slots than those that are nearer to the sink. Recall that, in any given partition, there are $h - 1$ consecutive nodes located on any one side at hop-distance h . We schedule all nodes on any given side alternately until the diagonal and non-diagonal nodes are scheduled for $H - h + 1$ time slots, and then schedule the diagonal nodes for an additional $\{(H - h + 1)(H - h + 2)/2\} - \{(H - h + 1)\} = (H - h + 1)(H - h)/2$ time slots.

Let

$$K = i - \left\lfloor \frac{i}{h} \right\rfloor h. \quad (4.5)$$

Then the series of nodes having index $\{kh, \dots, (k + 1)h - 1\}$ get $K = 0, 1, \dots, h - 1$.

Proof Let

$$i = kh + j. \quad (4.6)$$

Then we want to show that $j = K$.

$$\left\lfloor \frac{i}{h} \right\rfloor = k + \left\lfloor \frac{j}{h} \right\rfloor = k, \quad (4.7)$$

since $j = 0, \dots, h - 1$. Therefore, on plugging (4.6) and (4.7) into (4.5), we get $K = kh + j - kh = j$. \square

Since P uniquely identifies one partition, we start scheduling nodes of partition P starting at time slot P . Assuming that the time slots are sequenced as $0, 1, 2, \dots$, nodes in partition $0 \leq P \leq 5$ are scheduled in time slots of the form $P + 6n$, where n is a positive integer. Since the neighboring nodes of a side have consecutive K values, assigning them the time slots of the form $P + 6K$ allocates slots to neighboring nodes in consecutive sub-cycles.

Since there are $6h$ nodes on an h^{th} -hop hexagon, the adjacent slots allocated to a given node are separated by $6h$. Nodes $[h, i]$ are scheduled during the time slots:

$$t = P + 6K + 6nh, \text{ where } n = 0, 1, \dots, (H - h). \quad (4.8)$$

The diagonal nodes $[h, kh]$ are scheduled during the additional time slots:

$$t = P + 6(H - h + 1)h + 6m, \quad (4.9)$$

where $m = 0, 1, \dots, (H - h)(H - h + 1)/2 - 1$. Nodes can optimize energy consumption by going to "sleep" state after transmitting all packets.

During every time slot, one packet is transmitted to the sink and one packet to a node at one hop from the sink (except for the last six time slots). Due to the symmetry of traffic, at the end of every sub-cycle, all the first hop nodes contain exactly the same number of packets. Hence follows the continuity of packet flow to the sink node.

Theorem 4.5 *The transmission schedule obtained by the algorithm presented above is optimal in the sense that it achieves the minimum schedule length.*

Proof The algorithm schedules every node for u time slots if the number of packets routed by the node is $u - 1$. One slot is allocated for the packet originated at the node. From (4.8-4.9), during any given cycle, the last node to be scheduled has $P = 5$, $h = 1$, and $m = (H - h)(H - h + 1)/2 - 1$. Thus it is scheduled in the time slot $t = 3H(H + 1) - 1$. Thus, from (4.9), the length of one cycle is $3H(H + 1)$, which is equal to the total number of packets received at the sink. The sink node can receive only one packet at a time. Therefore, $3H(H + 1)$ slots are necessary for it to receive all the packets. Since, the algorithm schedules all the packets in the minimum amount of slots, the optimality follows. \square

4.5.4 Clock Synchronization

Since the order of transmissions is pre-determined by the scheduling algorithm presented in the previous section, over-hearing of transmissions can be used to deduce neighboring nodes' own schedules and also to synchronize their clock. The algorithm for implicit clock synchronization is presented in Algorithm 3. Observe that the proposed algorithm does not ensure monotonicity of time as adjustments can set the clock into the past. While this is, in general, a deprecated practice, in the special case of data collection sensor networks, where the primary network task is data transmission to a base-station, no serious adverse consequences ensue.

```

VAR:  $t$  {local time}
loop
  Do upon listening/receiving from node  $[h', i']$ 
   $t \leftarrow (t + t')/2$ 
end loop

```

Algorithm 3: Clock synchronization

4.5.5 Real-Time Capacity

During one cycle, all the $6h$ nodes at every h^{th} hop hexagon send one locally originated packet to the base-station. Thus each such packet contributes sh byte-hops to the real-time information transmission, where s is the packet size. Thus, the total amount of information transmission by all nodes in one cycle is

$$\sum_{h=1}^H 6sh^2 = sH(H+1)(2H+1). \quad (4.10)$$

The algorithm presented in this chapter transmits $3H^2 + 3H$ packets in the same number of time slots. Each time slot is s/W seconds, where W is the bandwidth. Thus, the real-time capacity using the transmission scheduling algorithm presented here is:

$$\begin{aligned} RTC &= \frac{sH(H+1)(2H+1)}{s/W(3H^2+3H)} \\ &= \frac{W(2H+1)}{3} \text{ byte-hops/sec.} \end{aligned} \quad (4.11)$$

Therefore, at the cost of linear increase in latency (due to the schedule cycle size), the real time capacity grows sub-linearly with the size of the network.

4.6 EVALUATION

We implemented a simulator to evaluate the robustness of our transmission scheduling algorithm in the presence of topological irregularities, namely long links, and to evaluate the rate of convergence of the implicit clock synchronization algorithm. Long links refer to those that arise when the interference range of a node is larger causing it to interfere with neighbors that it would not normally reach in a hexagonal topology. Observe that another form of irregularity is when the transmission range is too short, precluding connectivity along some of the hexagonal edges. This latter type, however, can be solved by deploying the nodes closer together until only

long links remain. The above argument justifies considering only long links in this evaluation.

Figure 4.6 shows the cumulative distribution of separation between receiver node(s) and the other sender nodes for three networks of size $H = 5$ (90 nodes), $H = 10$ (330 nodes) and $H = 20$ (1260 nodes). That there are no nodes of 2 or less hop separation can be seen as an experimental validation of our scheduling algorithm. The point of inflexion of the curves fall around $H/2$, meaning that the largest fraction of transmissions are separated by $H/2$ hops. Notably, as the network size increases, the curves get flatter, indicating that even larger fraction of transmissions happen at farther separations. Thus, our formalism should be fairly resilient in the presence of random long links.

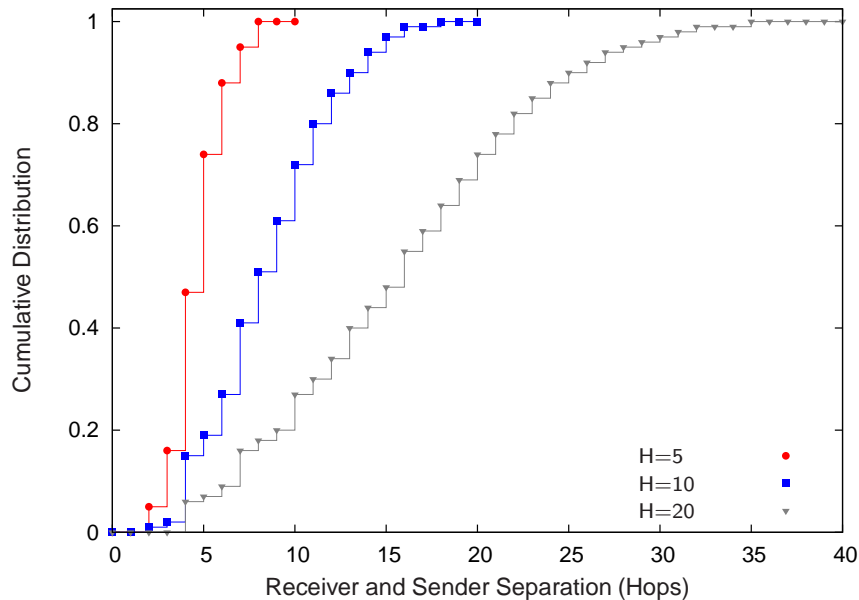


Figure 4.6 Cumulative distribution of separation between receiver and simultaneous senders (to different other receivers)

To verify our assumption, we simulated networks of irregular topology. We introduced long links where the excess length (> 1) of the links, x , was distributed according to Poisson probability distribution function ($p(x) = \lambda \exp^{-\lambda x}$). In Figure 4.7,

the percentage of collisions is reported as a function of the mean of the Poisson distribution, λ . Consistent with Figure 4.6, as the size of the network increases, the effect of irregularity diminishes. Collisions show up after $\lambda = 0.2$, indicating that the scheduling algorithm is robust in the presence of occasional long links. The performance degradation is graceful with the increase in the size of irregularity.

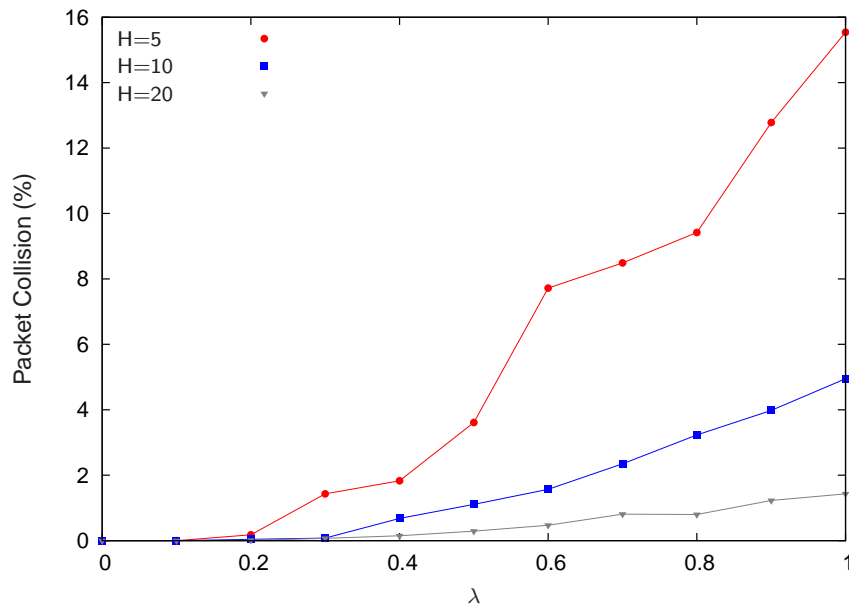


Figure 4.7 Percentage packet collision in the presence of radio irregularity

We simulated the our clock synchronization algorithm to evaluate its convergence rate. We started schedule cycles with nodes having uniformly randomly distributed clock times. We obtained the standard deviation of local times of all nodes as a function of simulation time where time granularity was the slot size. First we kept the initial range of random errors, E , in $[0, 100]$ and varied the network size. Shown in Figure 4.8, we observed a sharp exponential decline in the coefficient of variation initially, followed by a slow convergence to finer synchronization. Our clock synchronization algorithm achieves standard deviation < 1 within one schedule cycle. Next, to evaluate the effect of initial range of clock asynchrony on the convergence, we varied the error range exponentially, keeping the network size fixed

to $H = 10$. In Figure 4.9, we observe that even though the initial range of errors are varied exponentially, the rate of convergence is only linear. This is a cumulative consequence of time averaging.

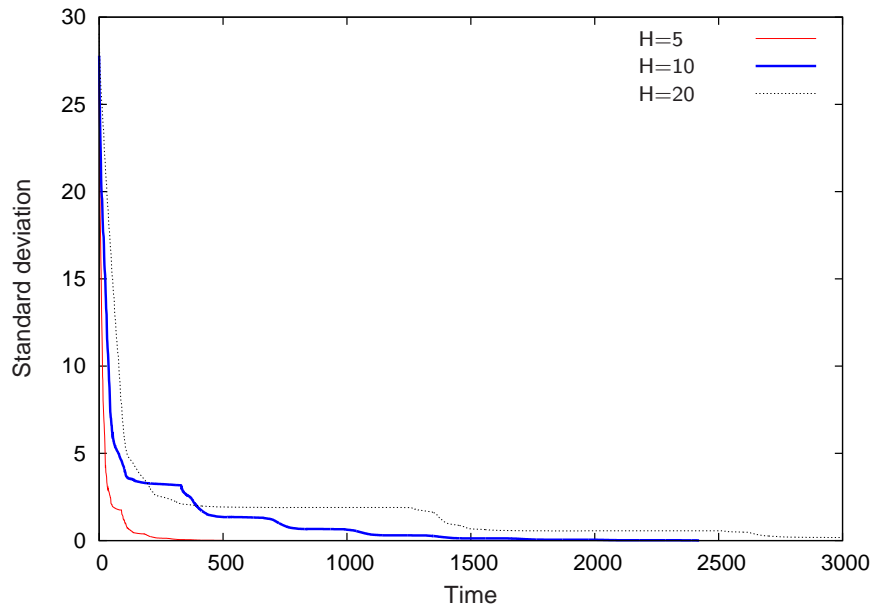


Figure 4.8 Convergence of clocks. $E = 100$ for all plots

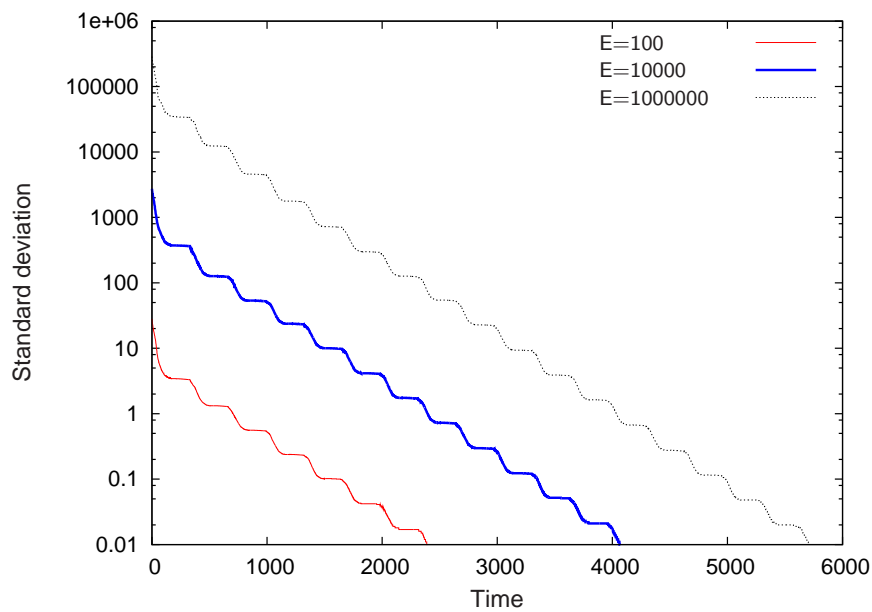


Figure 4.9 Convergence of clocks parameterized by initial error size. $H=10$ for all plots.

4.7 DISCUSSION AND SCHEDULING IMPLICATIONS

The scheme presented and evaluated in this chapter has significant implications on the ability of sensor networks to provide real-time guarantees. We have shown that given proper (i.e., hexagonal) node placement and topology control algorithms (that create a hexagonal mesh), it is possible to develop trivial addressing, routing, and clock synchronization algorithms that guarantee each node a fixed portion of network bandwidth and guarantee a bounded latency to the sink node. In particular, each node will be able to send for one time unit every $3H^2 + 3H$ time units. Its communication will reach the sink node in no more than $3H^2 + 3H$ time units. Since our algorithm transmits all packets in minimum time, in the case where all packets have the same deadline, our scheduling algorithm is optimal.

Previously proposed schedulability algorithms for real-time broadcast LANs (where each node can send for x out of y seconds on the LAN) can now be applied to schedule real-time traffic on the multi-hop sensor network with the added observation that a transmitted packet will incur a latency of no more than $3H^2 + 3H$ time units before it is delivered. Our simple addressing, routing, and clock synchronization algorithms provide a practical solution to ensure predictability of the network, making a wealth of prior schedulability analysis techniques applicable. Moreover, the sink node is 100% utilized, which is optimal. The protocol is self-synchronizing and easy to implement.

Hexagonal WSN associated with the equal bandwidth MAC protocol may be used for gathering periodic data from the network where the base-station is ensured of data collection from all parts of the network. It can be used for monitoring applications. Since all parts of the network get equal bandwidth, multiple objects can be detected and reported simultaneously. The protocol is especially useful in overloaded networks, since there is no scheduling overhead. It may be noted that under overloaded conditions, the CSMA based protocols perform very poorly. A suitable

prioritization scheme may be used by the cluster heads to select important data for transmission to the base-station, and hence enable prioritized transmissions locally.

4.8 LIMITATIONS

In the previous derivations, we assumed that faults do not occur. Making the protocol fault-tolerant is an interesting problem for future work.

Lost packets are not handled. Replicating packet routes is one well-known method to deal with this problem. Alternatively, since every node gets a known number of packets in a fixed order every cycle, lost packets can be detected, and requested for retransmission. Depending upon the packet loss rate, an appropriate number of time slots may be set aside at the end of every cycle to retrieve lost packets. The number of extra time slots need to be upper bounded or fixed to maintain the deterministic bandwidth guarantees.

Topological irregularities also affect the real-time guarantees of the system. We evaluated the protocol in the presence of long links, and found that it is robust in the presence of occasional long links. The reason for this robustness is that majority of transmissions occur at a separation of more than 2 hops, and the median of the separation length grows with the size of the network. In general, if the six first hop nodes (from the base-station) are free from topological irregularities, then the effects of topological irregularities will remain localized.

Missing or interrupted links create problems as well. Since the routing algorithm does not use all the links to route packets, a random link failure is not likely to disrupt the performance in other parts of the network. The farther the link failure occurs from the base-station, the smaller its impact will be. In other words, the impact of the faults grow as the links get closer to the base-station. This type of fault can be addressed in the protocol by either activating the topology control and adding a new node to the hexagon or by routing packets through neighboring nodes. The later solution, however, requires adding extra time slots to the cycles. Similar to the

missing link fault is the case of heterogeneous nodes. The heterogeneity of bandwidth will not affect the overall performance if it occurs at farther nodes. However, if it occurs nearer to the base-station, the number of collided packets will increase.

If multiple sinks are present inside one hexagon, then the protocol will still be fully functional, but will no longer be optimal. We considered a non 3-D network. As we mentioned in the related work section, addressing and routing in 3-D hexagonal networks have been studied. The most straightforward way to extend the protocol for 3-D networks appears to be a 3-D stacking of planar hexagons.

4.9 SUMMARY

In this chapter we presented a novel way to design wireless ad-hoc networks, namely constructing a hexagonal network topology. We presented addressing and constant time routing algorithms for multi-hop hexagonal networks. We presented closed form expressions that creates network-wide conflict free TDMA schedule with zero message overhead. The resulting MAC protocol gives equal bandwidth to every node, and is optimal in the sense that the base-station does not idle. It affords real-time guarantees on packet delivery. Using simulation, we showed that our algorithms are robust. The effect of faults tend to be localized if the faults happen farther from the base-station. The six first hop nodes are the most important nodes for robustness of the protocol.

CHAPTER 5

Scheduling General Traffic in Hexagonal Networks

5.1 INTRODUCTION

In the previous chapter, we introduced the hexagonal wireless ad-hoc networks. We presented algorithms for the basic functionalities of the network – addressing, routing, clock synchronization and transmission scheduling. We presented a spatial-reuse time domain multiplexed (STDM) medium access control (MAC) protocol. The MAC protocol was designed to give equal bandwidth to all nodes. Such a solution is useful in cases like overloaded network where data from all sources are equally important or constant rate sampling of data from every part of the network. However, a more general case of traffic involves a variable bandwidth demand at the nodes of the network. Since the hexagonal WSN is cluster based, it may be the case that some clusters have larger memberships than others, and accordingly, need larger bandwidth.

Collecting data of physical entities like temperature or humidity are usually regarded as time-triggered applications. Periodic task model has been very successful in the analysis of such applications. The same WSN can also be used for event-triggered applications. Clearly, the data generated by such applications are aperiodic in nature. An intrusion detection application will need to transmit data to the base-station only when it detects intrusions. Providing periodic time slots to the nodes for such applications is clearly wasteful. A more useful solution is to provide

time slots for periodic packets on per-node basis, and to provide a budgeted amount of time slots for aperiodic packets on per-group-of-nodes basis every cycle.

In the following, we develop a MAC protocol that supports both periodic and aperiodic traffic. The set of nodes that share the budgeted aperiodic time slots is formed by all nodes on a side of the hexagon. All such packets are scheduled such that they reach the base-station within one cycle time. A node may have none, any or both of the above mentioned kinds of traffic requirement. The MAC protocol uses available bandwidth slack to transmit additional aperiodic packets.

To be able to provide real-time service guarantees, the duration between two successive opportunities for transmission of the nodes must be bounded. The timed token protocol [31] has been used extensively to support distributed real-time applications in wired networks and multi-processor architectures. In the timed token protocol, a special bit pattern called *token* circulates among the nodes. The time sensitive packets are termed as *synchronous* messages, and the non real-time packets are termed as *asynchronous* messages. Upon reception of the token, every node is allowed to transmit synchronous messages for a fixed duration. If the node senses availability of extra bandwidth by early arrival of the token, it is allowed to transmit an additional amount of asynchronous messages up to the detected bandwidth slack.

The expected time interval between any two successive visits of the token to the same node, called target token rotation time (TTRT), is an important parameter of the protocol from performance perspective [37, 33]. It has been proven by Sevcik and Johnson [38, 60] that the duration between successive visits of the token in the timed token protocol [31] is bounded by $2.TTRT$. Due to this bounded latency property, the use of the timed token protocol for medium access control in real-time distributed systems has been a focus of considerable research, and has resulted in adoption of the protocol to IEEE 802.4 token bus, fiber distributed data interface

(FDDI) and survivable adaptable fiber optic embedded network (SAFENET) standards.

Using the boundedness property of the consecutive token visit intervals, Agarwal et al. proved the real-time properties of the timed token protocol such as the bandwidth available to a node in an extended period of multiple token rotations [3]. In addition to the inter-token arrival latency, a bandwidth allocation scheme for synchronous traffic is necessary to provide real-time guarantees. A number of synchronous bandwidth allocation (SBA) schemes exist [3, 70, 69]. Most of these SBA schemes have guaranteed worst case utilization in the range of 0 - 0.33.

The STDM transmission scheduling of packets presented in the previous chapter can be thought of as timed token protocol where the tokens are implicit (determined by the expressions (4.8, 4.9)). In this chapter, we generalize the scheduling algorithm of the previous chapter (Section 4.5.3) to accommodate variable amount of real-time packets originating at the nodes. Our analysis is fundamentally different from that for the wired networks reported in the literature since —

- ◆ We consider end-to-end scheduling of packets. Previously reported analyses for wired networks consider the time of putting the last bit on the wire by the source node to determine the schedulability. The deadline is considered to be met if the bit was placed no later than the packet's deadline. We consider meeting the deadline only if the packet was delivered to the *base-station* no later than the deadline.
- ◆ In token ring networks, every packet traverses the same (logical) path. Packets of our system, in general, take different paths to the base-station.
- ◆ We employ spatial reuse, which initiates multiple transmissions simultaneously. Multiple tokens exist in the network, H to be exact. Hence we also need to consider distributed synchronization of the tokens such that collisions do not occur.

The rest of this chapter is organized as follows: we present the formal problem statement in the next section, followed by the assumptions in Section 5.3. We present the synchronous bandwidth allocation and transmission scheduling algorithms in Section 5.4. We present an analysis of schedulable utilization and real-time capacity of the system in Section 5.5, followed by the real-time capacity expression of the system in Section 5.6. We present simulation results in Section 5.7, and the limitations of the results in Section 5.8. We present the chapter summary in Section 5.9.

5.2 PROBLEM FORMULATION

We consider a hexagonal topology wireless ad-hoc network where $[h, i]$ denotes node i located on the hexagonal ring that is h hops away from the base station. In the following, we use the time required to send one packet over single hop and token rotation overhead as the unit of time. Each node $[h, i]$ needs to send n_i packets every P_i time units, where each of the n_i packets originated at the node has deadline D_i (For the simplicity of notation, we suppress writing h in the subscripts.) Thus, the packets $rn_i+1, \dots, (r+1)n_i$ must reach the base station by time rP_i+D_i . The deadline D_i is *not* necessarily the same as P_i . Each node may also have aperiodic packets for transmission to the base-station.

Wireless ad-hoc network nodes generate as well as route packets. In reference to a given node, we distinguish the packets generated at the node by referring them as *local packets*. Similar to the transmission scheduling algorithm presented in the previous chapter, our schedule consists of cycles. We use \mathcal{T} to denote the number of time slots contained in one cycle. This number remains fixed unless a new schedule is constructed, which may happen due to changes in the synchronous bandwidth demand of nodes.

5.3 ASSUMPTIONS

The real-time guarantees are given on the assumption of the absence of the faults that are described in the previous chapter (Section 4.4). In addition to the absence of those faults, we assume that the nodes conform to the bandwidth allocation. In other words, we assume that the nodes do not transmit for more than allocated time slots.

During any given cycle, the amount of periodic traffic that can be originated at the nodes is determined by assuming that the nodes will always consume the allocated synchronous bandwidth. If this synchronous bandwidth is not used by either the periodic or aperiodic traffic, the cycle length is not affected. This is done to avoid per cycle overhead needed for the (distributed) synchronization among the tokens.

5.4 TRANSMISSION SCHEDULING ALGORITHM

We describe the transmission scheduling algorithm for the general load in this section. We start with bandwidth allocation to the six partitions of the network. We follow this up with a discussion of scheduling of the aperiodic packets to utilize any slack in the schedule, as well as allocating bandwidth budgets when the slack in the schedule is insufficient. We present the token passing algorithm at the end.

Let l_i denote the bandwidth allocated to node $[h, i]$ for local packets and f_i be that for forwarding other packets. During one cycle, a node may transmit periodic packets for up to b_i time units, where the bandwidth $b_i = l_i + f_i$. From Algorithm 2 for routing packets, $[h, i] \Rightarrow [h - 1, i - Q]$. Therefore,

$$f_i = \sum_j b_j |[h, j] \Rightarrow [h - 1, i]. \quad (5.1)$$

We derive expressions to determine the bandwidth and cycle time in the next section. For the purpose of present discussion of the scheduling algorithm, we assume

that the bandwidths l_i are given. To partition the nodes for spatial reuse, we apply Theorem 4.4. An example is presented in Figure 5.1, where the sides that contain the nodes of the same partition are shown in the same color. The numbers on the sides of the hexagon represent the total bandwidth given to the nodes that belong to that particular side of the hexagon. Each side covers one hextant at any given hop. Let this total bandwidth on a side at hop count h from the base station in hextant j be denoted by $A(h, j)$.

$$A(h, j) \triangleq \lceil \sum_{i \in [h, \cdot]} b_i | [h, i] \in \text{Hextant}(j) \rceil. \quad (5.2)$$

Observe that $A(h, j)$ is monotonically non-increasing in h , i.e., $A(h, j) \geq A(h + l, j) \forall l \leq H - h$.

Bandwidth of a partition

Let us define the synchronous bandwidth of partition i as

$$B_i \triangleq \max(A(h, j) | j \in \text{Partition}(i)). \quad (5.3)$$

Due to the monotonicity of A , we need to consider only the first three hops to determine B_i s. Let us consider the example of Figure 5.1. The bandwidth $A(\cdot, 1)$ associated with the first three sides of the first partition are 7, 15 and 7 (shown in navy blue color). Hence, from (5.3), the bandwidth allocated to the first partition, B_1 , is 15. Similarly, $B_2 = 7$, $B_3 = 20$ and so on.

If the nodes in partition i are given B_i time slots in a cycle, then every periodic packet can be transmitted to the next hop in the same cycle. Hence, if the transmissions be interleaved among the six partitions as we did in the last chapter, with the exception that bandwidth allocations are respected, then, all synchronous packets can be transmitted to the base station in $\sum_i B_i + K$ time units, where K is the initial “warm-up time,” needed for one packet from every hextant to reach the six first hop

nodes. The exact value of K depends on a particular load distribution, but its upper bound is $6H$.

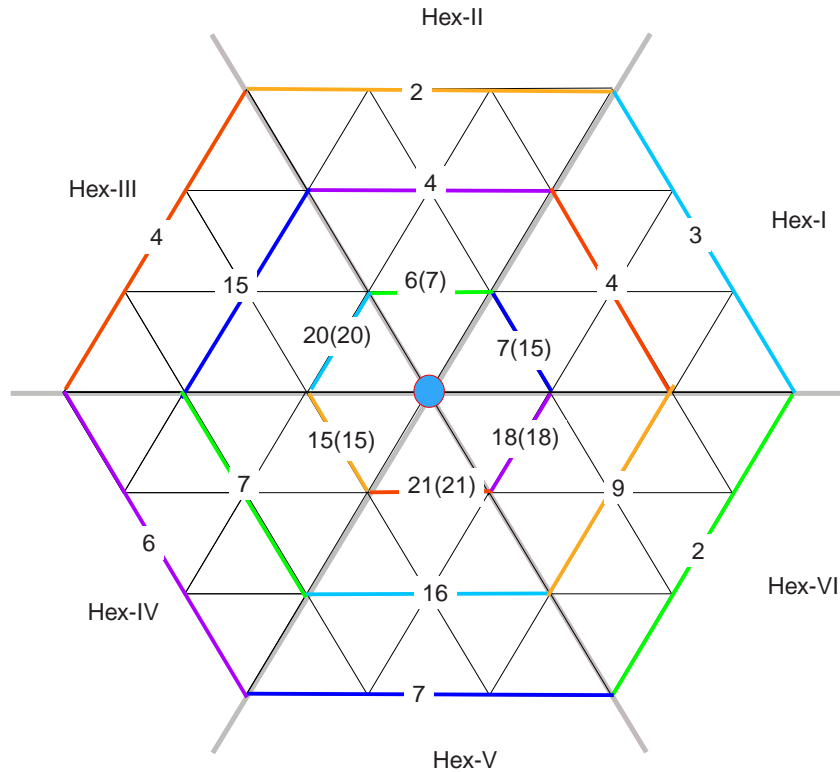


Figure 5.1 An example bandwidth allocation of the partitions. Only the first three hops are shown.

Bandwidth allocation for aperiodic traffic

Once again, let us consider the (hypothetical) example shown in Figure 5.1. The bandwidth allocated to the first partition is 15. However, the maximum bandwidth demand of periodic tasks in the first hexant is 7. Hence an additional 8 units of the bandwidth can be used to schedule aperiodic packets in this hexant. Similarly one unit of extra bandwidth is available to the nodes of the second hexant. However, there is no extra bandwidth available in the rest four hexants. Thus, lack of load balance among the nodes of the network results in bandwidth slack available in some other parts of the network. This slack may be used to transmit asynchronous packets. To allocate a bandwidth budget of β_i to aperiodic packets in hexant i , B_i

needs to be incremented by the difference of β_i and the slack. Therefore, to allocate an aperiodic budget of 5 units to the first hexant, no adjustment is needed since a slack of 8 units is already available. However to allocate the same budget to the third hexant, the bandwidth of the third partition is adjusted to $B'_3 = B_3 + 5 = 25$. The order in which this adjustment is done is important. Since adjustment in one hexant can increase the slack in another, the bandwidth adjustment must start with the hexant with the smallest slack and proceed to the other hexants in the order to increasing slack. This minimizes the cycle length increase.

Variable bandwidth medium access control protocol

Recall that in the equal bandwidth MAC protocol presented in the previous chapter, the first transmissions start at the first nodes (determined by anti-clockwise order) of the sides of the first partition, shown with the start mark in Figure 5.2. After these nodes have sent one packet, the second set of transmissions take place at the first nodes of the sides of the second partition, and so on. For the variable bandwidth MAC, we use the same transmission order. For the synchronization of tokens, we take the following approach. At the initiation time, the partition bandwidth and asynchronous budgets of the sides of all partitions are stored in the first nodes of every side of the partition. For example, in Figure 5.2, nodes $[1, 0]$, $[2, 4]$ and $[3, 12]$ are the first nodes (in transmission order) of the first three sides of the first partition. The base-station initiates transmissions by sending H tokens to the first nodes of the first partition with a fixed time for transmission initiation. The tokens can be in two states, namely, SEND-TOKEN and PASS-TOKEN. If a node gets the token in the SEND-TOKEN state, it can transmit one asynchronous packet if the remaining budget is non zero, or one synchronous packet if it has not used its allocated synchronous bandwidth during the current cycle. If the node does transmit a packet, it subtracts one from the remaining bandwidth allocated to the side and

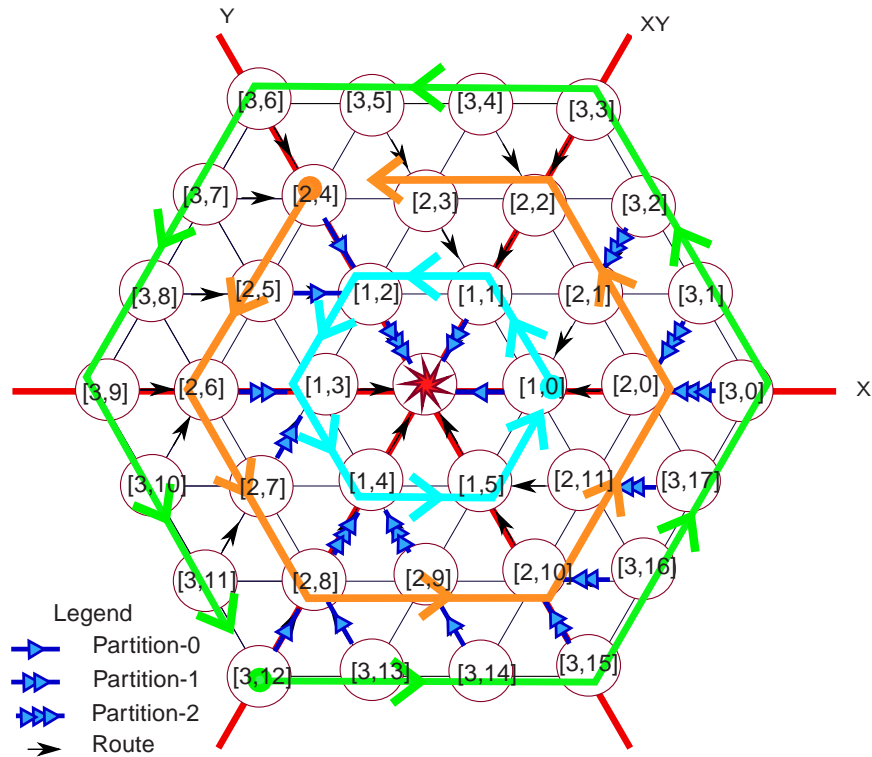


Figure 5.2 Token rotation order

from its own remaining allocation; alters the state of the token to PASS-TOKEN and forwards the token to its anti-clockwise neighbor (arrows in Figure 5.2). Once the PASS-TOKEN token reaches the last nodes of the partition (which may also be the result of the transition from SEND-TOKEN to PASS-TOKEN locally), the nodes alter the state of the token to SEND-TOKEN, decrement the remaining partition bandwidth, and send the token to their anti-clockwise neighbor in the adjacent hextant. If the remaining synchronous bandwidth allocation at a node becomes zero, then the send token and the remaining total bandwidth allocated to the side is sent to the next neighbor in the anti-clockwise order. If a non-zero remaining partition bandwidth exists at the last node of any side, then upon the receipt of SEND-TOKEN, the remaining partition bandwidth is decremented by one, the token is held for one time unit and then released to the neighbor in the adjacent hextant. If the remaining partition bandwidth reaches zero, the token zips through the hextant(s). At the end of the

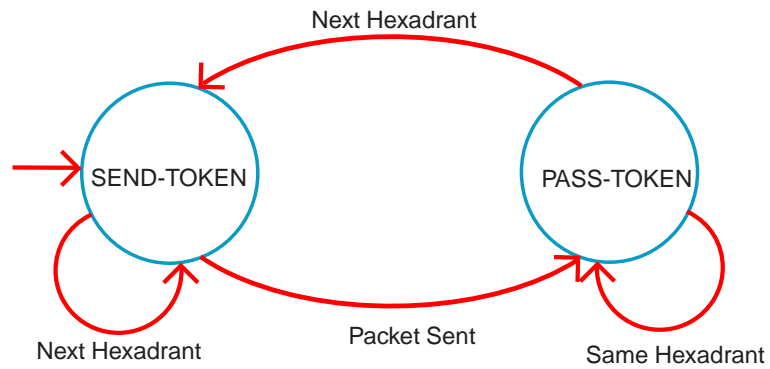


Figure 5.3 State diagram of tokens

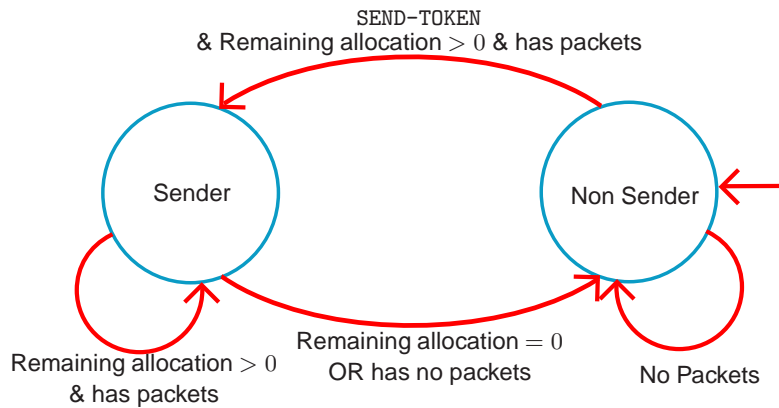


Figure 5.4 State diagram of nodes

cycle the remaining side bandwidths at the first nodes of the partitions, and the remaining partition bandwidth at the last nodes of the partitions are restored to their initial values. Figures 5.3- 5.4 present the state diagrams of tokens and nodes.

Loss of token is deduced from no reception of the token after a time interval of one token rotation time. Upon the token loss detection, the first node of the first partition generates a new token. Delayed tokens may also be interpreted as lost tokens. If the designated node receives a delayed token after it has generated a new token, it discards the delayed token.

5.5 NODE BANDWIDTH ALLOCATION AND SCHEDULABILITY ANALYSIS

In order to avoid packet buildup and hence instability over long term, the number of real-time local packets injected into the network in \mathcal{T} units of time must be smaller than or equal to the number of such packets delivered at the base station in \mathcal{T} units of time. Hence

$$6H + \sum_i B_i \leq \text{Maximum packet latency} \leq \mathcal{T}. \quad (5.4)$$

Nodes can transmit periodic packets for up to l_i time units every cycle \mathcal{T} . Since nodes must get a chance to transmit at-least once every D_i time units, the cycle time \mathcal{T} can not be larger than D_{Min} .

Therefore, the feasibility conditions are:

$$6H + \sum_i B_i \leq \text{Maximum packet latency} \leq \mathcal{T} \leq D_{\text{Min}}. \quad (5.5)$$

The number of cycles contained in time D_i is given by $\lfloor D_i/\mathcal{T} \rfloor$. During these $\lfloor D_i/\mathcal{T} \rfloor$ cycles, n_i packets must be transmitted to the base station. Therefore, the necessary condition for meeting the deadlines is given by:

$$l_i = \frac{n_i}{\lfloor D_i/\mathcal{T} \rfloor} \quad (5.6)$$

We are considering arbitrary deadlines, which may be larger than periods. For stability (in time), it is necessary that the rate of the number of packet generation does not exceed the rate of packet consumption. Therefore, for this equilibrium of packets, the number of packets generated within a deadline interval must not be larger than the number of packets that is schedulable at every node. In other words,

$$\begin{aligned} l_i \lfloor D_i/\mathcal{T} \rfloor &\geq \lceil D_i/P_i \rceil n_i \\ \text{or, } l_i &\geq \frac{\lceil D_i/P_i \rceil}{\lfloor D_i/\mathcal{T} \rfloor} n_i \end{aligned} \quad (5.7)$$

Since there is no value in allocating larger bandwidth to real-time packets than the necessary requirement,

$$l_i = \frac{\lceil D_i/P_i \rceil}{\lceil D_i/T \rceil} n_i \quad (5.8)$$

Observe that (5.8) implies (5.6).

We now obtain the expression for the schedulable load in terms of utilization.

$$\begin{aligned} \sum_i l_i &= \sum_i \left\{ \frac{\lceil D_i/P_i \rceil}{\lceil D_i/T \rceil} n_i \right\} \\ &= \sum_i \left\{ \frac{\lceil D_i/P_i \rceil P_i}{\lceil D_i/T \rceil T} \frac{n_i}{P_i} T \right\} \\ &\geq \gamma T \sum_i \rho_i, \end{aligned} \quad (5.9)$$

where

$$\gamma = \min \left(\frac{\lceil D_i/P_i \rceil P_i}{\lceil D_i/T \rceil T} \right), \quad (5.10)$$

and,

$$\rho_i = \frac{n_i}{P_i} \quad (5.11)$$

From (5.3) and the partition bandwidth allocation algorithm, we have

$$\sum_i l_i \leq \sum_i B_i \leq \sum_i l_i + \sum_{h,k} a_{h,k}, \quad (5.12)$$

where $a_{h,k}$ is the bandwidth allocated for aperiodic traffic.

From(5.4),

$$\begin{aligned}
\mathcal{T} &\geq \sum_i B_i + 6H \\
&\geq \sum_i l_i + 6H \text{ From (5.12)} \\
&\geq \sum_i \rho_i + 6H \text{ From (5.9)} \\
\Rightarrow \sum_i \rho_i &\leq \frac{1}{\gamma} - \frac{6H}{\gamma\mathcal{T}}. \tag{5.13}
\end{aligned}$$

But $\sum_i \rho_i = \sum_i n_i/P_i$, and hence, is equal to the utilization of periodic packets, which we denote as U_{RT} . Therefore, from (5.13),

$$U_{RT} \leq \frac{1}{\gamma} - \frac{6H}{\gamma\mathcal{T}}. \tag{5.14}$$

Hence, the schedulable utilization of periodic packet flows is an increasing function of cycle size \mathcal{T} . Thus, when selecting the cycle size, the largest feasible value should be chosen.

Now we consider the aperiodic packets. Let us define the utilization of aperiodic packets as the ratio of maximum aperiodic budget summed over all nodes of the network and the cycle size. In other words, U_{NRT} , the utilization of aperiodic packets is given by:

$$U_{NRT} \triangleq \sum_{h,k} a_{h,k}. \tag{5.15}$$

Let

$$\Gamma = \max \left(\frac{\lceil D_i/P_i \rceil P_i}{\lfloor D_i/\mathcal{T} \rfloor \mathcal{T}} \right). \tag{5.16}$$

Then, from (5.8),

$$\sum_i l_i \leq \Gamma \mathcal{T} \sum_i \rho_i. \tag{5.17}$$

From (5.12), we have

$$\begin{aligned}
\mathcal{T} &\leq \sum_i l_i + \sum_{h,k} a_{h,k} \\
&\leq \Gamma \mathcal{T} \sum_i \rho_i + \sum_{h,k} a_{h,k} \\
\Rightarrow \sum_{h,k} a_{h,k} &\geq \mathcal{T} (1 - \Gamma \sum_i \rho_i) \\
\Rightarrow U_{NRT} &\geq 1 - \Gamma U_{RT}
\end{aligned} \tag{5.18}$$

The inequality (5.18) gives the lower bound on the utilization of the aperiodic traffic (per cycle).

5.6 REAL-TIME CAPACITY

During one cycle, $\sum_i l_i$ periodic packets are transmitted. The byte-hop product of the data transmission is $\sum_i s l_i h_i$, where s is the size of the packet. Every cycle consists of \mathcal{T} time slots, where each slot is large enough to send one packet over one hop and allow for token rotation time. Hence the slot size equals $(s + 6\epsilon H)/W$ seconds, where W is the bandwidth and ϵ is the token size. Thus, the real-time capacity is:

$$\begin{aligned}
RTC &= \frac{s \sum_i l_i h_i}{\frac{s+6\epsilon H}{W} \mathcal{T}} \\
&\geq \frac{W \gamma \mathcal{T} \sum_i \rho_i h_i}{(1 + \frac{6\epsilon H}{s}) \mathcal{T}} \text{ From (5.8)} \\
&= \frac{W \gamma \sum_i \rho_i h_i}{(1 + \frac{6\epsilon H}{s})} \\
&= \frac{W \gamma}{(1 + \frac{6\epsilon H}{s})} \sum_i \frac{n_i h_i}{P_i} \text{ byte-hops/sec.}
\end{aligned} \tag{5.19}$$

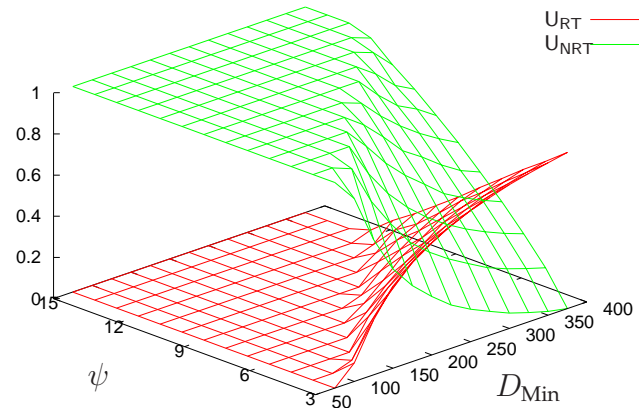
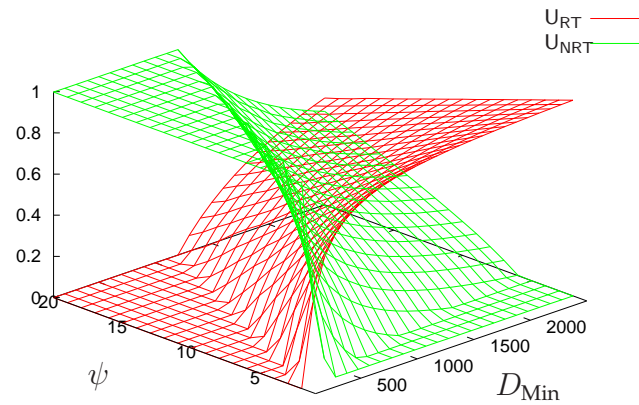
Thus, the real-time capacity is an increasing function of the ratio D_i/P_i for a given feasible task set.

5.7 EVALUATION

We implemented a simulator to study the behavior of the analytical expressions of utilization bounds and real time capacity. The simulator takes the network size, minimum deadline, D_{Min} and the ratio $D_{\text{Min}}/\tau \triangleq \psi$ as input parameters. It generates periodic tasks of deadlines whose deviations from D_{Min} are exponentially distributed. We generated three kinds of tasks, one for each of the three cases of $P_i \geq, =, \leq D_i$; and two network sizes – 5 and 10-hop hexagons (corresponding to 90 and 330 nodes respectively). Each data point is averaged over 1000 runs, where loads were generated afresh in each run. In the following, we denote the real-time capacity of the MAC protocol presented in this chapter by RTC_{VB} (variable bandwidth) and that of the MAC protocol presented in the previous chapter by RTC_{EB} (equal bandwidth).

The first three set of figures (Figure 5.5 - Figure 5.7) show the utilizations of real-time packet flows that are schedulable by the variable bandwidth MAC. Since $\tau = D_{\text{Min}}/\psi$, larger values of ψ result in small cycle times. At small cycle times, the warm-up and token-passing overheads become significant and hence the real-time utilization decreases.

The cycle time depends on D_{Min} only. The period enters into the utilization expressions only as the ratio $\lceil D_i/P_i \rceil$, and since D_i are exponentially distributed, the real-time utilization bound, U_{RT} does not change noticeably for the cases of the periods being larger or smaller than the deadlines. In the special case of deadlines equal to the periods, the utilization bound becomes independent of individual deadlines and periods, and γ gets minimized. Therefore, the utilization bound is maximum for this case. Among Figures 5.5–5.7 the volume under the real-time utilization U_{RT} curves is maximum in Figures 5(a)–5(b) which corresponds to the case of deadlines equal to periods.

(a) $N = 90$ ($H = 5$)(b) $N = 330$ ($H = 10$)**Figure 5.5** Schedulable utilization when the deadlines are the same as periods ($D_i = P_i$)

The three set of figures (Figure 5.8 - Figure 5.10) are the plots of real-time capacity of the two MAC protocols. The utilization bounds of the variable bandwidth MAC protocol are also shown. Observe that the real-time utilization bound of equal bandwidth MAC protocol is 1 everywhere except for the region $D_{\text{Min}} < 3H(H + 1)$ where it is 0. These figures highlight the trade-off between real-time capacity and latency. While equal bandwidth MAC protocol offers higher real-time utilization, it can't accommodate flows of deadline $< 3H(H + 1)$. On the other hand, variable bandwidth

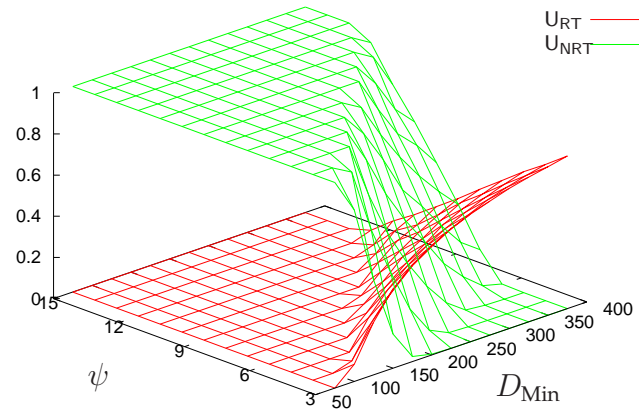
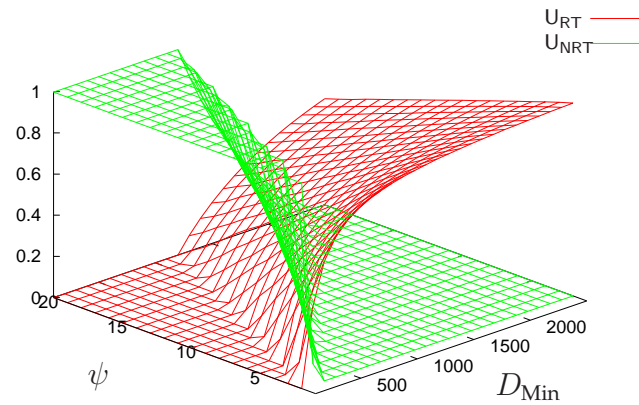
(a) $N = 90$ ($H = 5$)(b) $N = 330$ ($H = 10$)

Figure 5.6 Schedulable utilization when the deadlines are larger than the periods ($D_i \geq P_i$)

MAC can accommodate almost every feasible flow, but it offers smaller real-time capacity. On the loads that the equal bandwidth MAC protocol can't schedule, the offered real-time capacity of variable bandwidth MAC is small.

Figure 5.11 shows the real-time capacity of the variable bandwidth MAC protocol as a function of network size. Similar to the equal bandwidth MAC, the real-time capacity grows sub-linearly with the number of nodes in the network, but the rate of increase is slower.

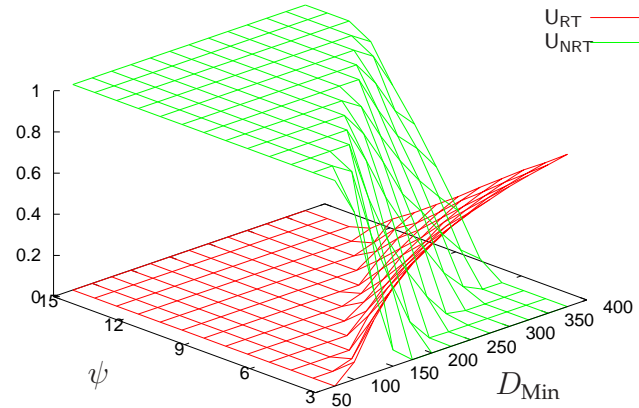
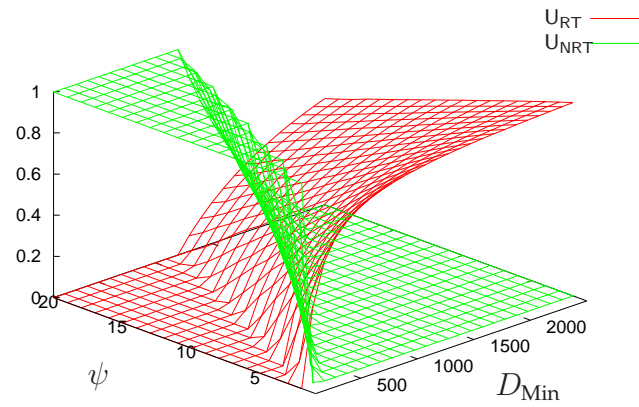
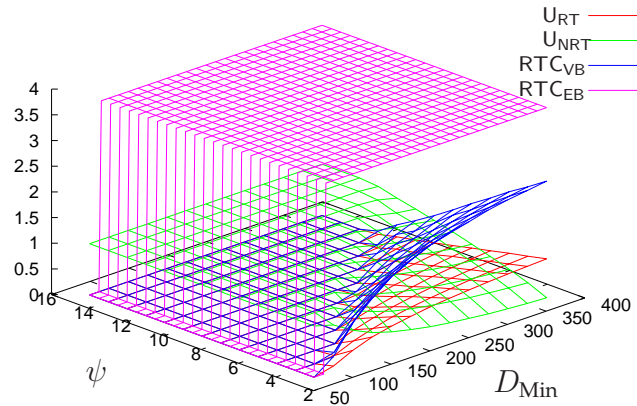
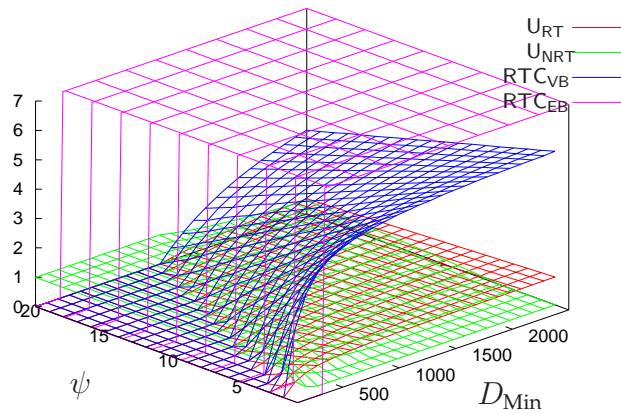
(a) $N = 90$ ($H = 5$)(b) $N = 330$ ($H = 10$)

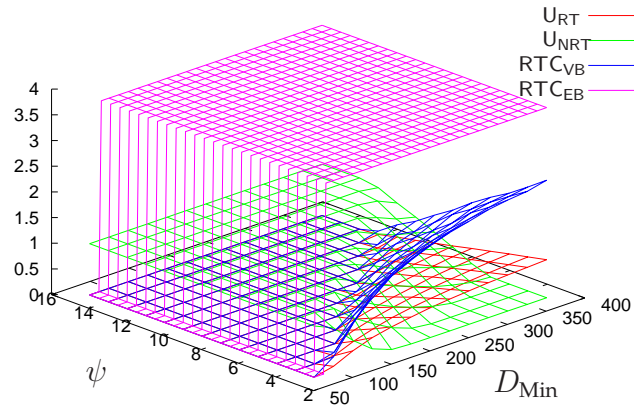
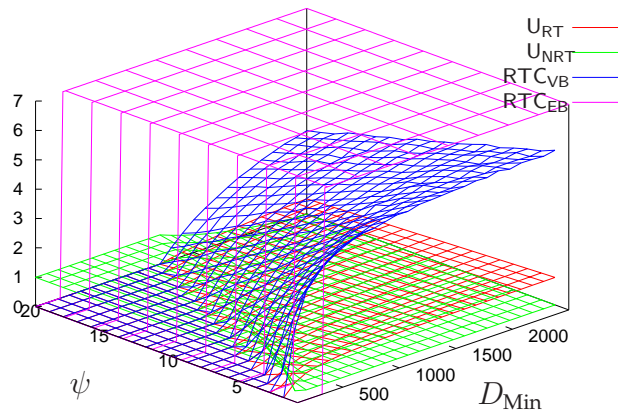
Figure 5.7 Schedulable utilization when the deadlines are smaller than the periods ($D_i \leq P_i$)

5.8 LIMITATIONS

Because of the token rotation overhead, the network utilization decreases with the token rotation time, and becomes zero when the token rotation time gets so small that no time is left for packet transmissions. As mentioned earlier, smaller deadlines require smaller token rotation times, and thereby reduce the network utilization. Therefore, as the evaluations show, the admissible load decreases significantly if the minimum deadline is too small, that is, $D_{\text{Min}}/N < 1$.

(a) $N = 90$ ($H = 5$)(b) $N = 330$ ($H = 10$)**Figure 5.8** Real-time capacity when the deadlines are the same as periods ($D_i = P_i$)

The faults discussed in the previous chapter, can adversely affect the real-time properties of this protocol as well. However, the faults due to heterogeneity in the network pose less of a problem for this protocol than for the equal bandwidth MAC protocol. This is due to the presence of explicit tokens. Lower bit-rate nodes will simply hold the token for longer time. The token rotation time however may

(a) $N = 90$ ($H = 5$)(b) $N = 330$ ($H = 10$)**Figure 5.9** Real-time capacity when the deadlines are larger than the periods ($D_i \geq P_i$)

cause significant overhead if the largest packet transmission time(s) vary significantly from the average. This problem may be avoided by not using such nodes for cluster heads.

This MAC protocol relies heavily on the nodes to conform to bandwidth allocations. A malfunctioning or malicious node can block transmissions from all other nodes of any given side. Such node however, won't be able to affect transmissions in other hexants. It also won't be able to affect transmissions of real-time traffic

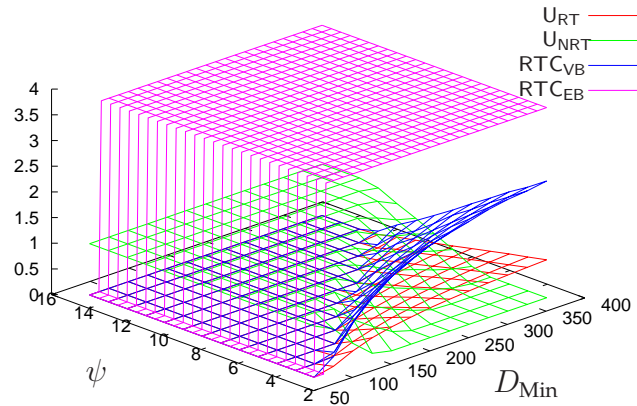
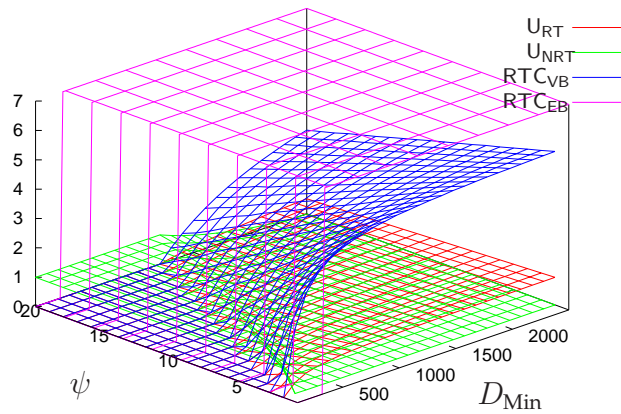
(a) $N = 90$ ($H = 5$)(b) $N = 330$ ($H = 10$)

Figure 5.10 Real-time capacity when the deadlines are smaller than the periods ($D_i \leq P_i$)

from those sides that are closer to the base-station but are in the same hexant as the malfunctioning node. As we have discussed earlier, the six first hop nodes are the most significant for proper functioning of the protocols. If the malfunctioning node happened to be one of the six first hop nodes, it can potentially disrupt the entire network.

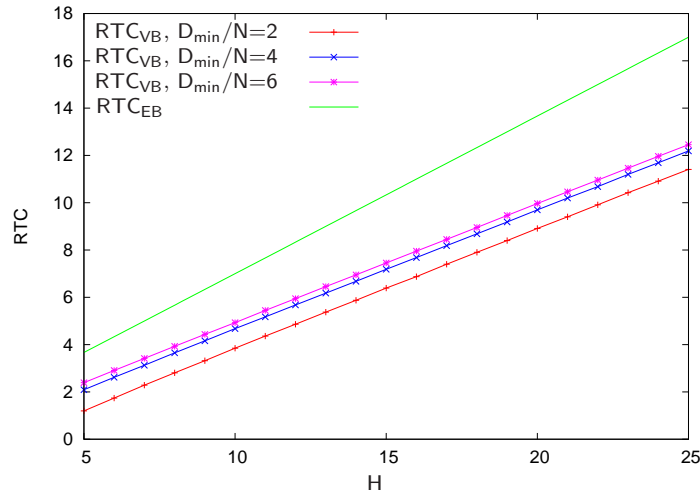


Figure 5.11 Real-time capacity as function of network size, parameterized by the ratio D_{Min}/N , where N is the number of nodes in the system.

5.9 SUMMARY

In this chapter we presented a distributed spatial-reuse time domain multiplexed MAC protocol for transmission of both real-time and non real-time traffic in hexagonal wireless ad-hoc networks. We presented an expression for local bandwidth allocation algorithm for periodic real-time traffic. We derived expressions for the upper bound of schedulable utilization of the periodic traffic and the lower bound on the utilization of aperiodic traffic. We showed that the schedulable real-time utilization is an increasing function of cycle time. We derived the real-time capacity of the transmission scheduling algorithm presented in this chapter. The utilization and real-time capacity expressions can be used at design time to select network parameters such as minimum relative deadline, token rotation time, and packet size.

CHAPTER 6

Data Dissemination in Wireless Sensor Networks

6.1 INTRODUCTION

Data gathering and data dissemination are the two key functionalities required of WSN. In disaster management applications, access of relevant data to the personnel facilitates coordination of the relief efforts. In a fire-fighting situation, firemen and affected people may want to know the condition of the area in real-time. Each may use a PDA to get data from the network to know whether the fire is being extinguished and to make strategic decisions accordingly. In the last two chapters, we focussed on the traffic for data gathering in WSN. For such traffic, scheduling becomes important for giving real-time guarantees. Epidemic algorithms are often used for data dissemination. Such algorithms not only consume energy excessively, but also the termination of such protocols are often not deterministic. It is desirable that the network be able to support both functionalities without causing too much stress on the limited resources, especially the energy supply. Although using broadcast solves the data dissemination problem in uni-hop wireless networks (e.g., cellular network) trivially, the solution for multi-hop WSN in energy-efficient manner is not straightforward.

When multiple nodes receive data from some given data sender, it is useful to reduce multiplicity of transmissions. This is the problem of efficient multicast. Multicast allows suppression of redundant transmissions along same path, and thus saves both energy and bandwidth. Asynchronous multicast improves the energy

efficiency of multicast by disseminating packets at the desired rate of the individual subscribers. Organizing data dissemination paths as a tree with some nodes acting as data caches leads to a more efficient use of bandwidth and energy supply than unicast. Consequently, the network can support a larger number of subscribers for longer durations. We call such a tree a Steiner Data Caching Tree (SDCT). In the following, we study the properties and construction of SDCTs in WSNs. For example, in Section 6.4, we prove that a minimum SDCT (**MSDCT**) is a *binary* tree. We published the results that appear in this chapter in [53].

The remainder of this chapter is organized as follows: in the next section, we present the data cache placement problem formulation and the service model, followed by a list of assumptions in Section 6.3. We prove properties of minimum data caching trees in Section 6.4. Section 6.5 presents the dynamic data cache placement heuristic, followed by evaluations of the heuristic in Section 6.6. In Section 6.7, we present a brief discussion on adaptation of the existing rich collection of polynomial time Euclidean Steiner Tree algorithms to SDCT. In Section 6.8, we consider the real-time issues of data dissemination, followed by a discussion on limitations. We present conclusions in Section 6.10.

6.2 PROBLEM FORMULATION AND SERVICE MODEL

In the following, we consider the problem of efficient dissemination and retrieval of data pertaining to some event of interest to multiple subscribers. In WSN applications, such as monitoring, events of interest occur at certain places. These events are then sensed by local sensor nodes. One or more of these nodes send the processed information to the subscribers of the data, also called base stations, or sinks. The subscribers are more powerful computers (e.g., a laptop or a PDA). The node that communicates data about the event to the subscribers can be chosen using some distributed leader election algorithm such as [48]. This scenario can be described as a

publish-subscribe model, in which the leader node is the publisher of data, and the subscribers belong to the multicast group.

The frequency of data sent to the subscribers depends upon the frequency of data generated by the event, and the requested data refresh rate of the subscribers. **Data Refresh Rate** on a link is the rate at which data pertaining to the event being monitored by the WSN should be delivered to the receiver. This is different from the data generation rate by individual sensors, which is related to the rate of change of event state itself. The data refresh rate, for example, may be set by the receiver in accordance with its maximum tolerated data staleness and it might be lower than the rate of change of data at the source. To optimize energy consumption, update traffic is *asynchronously* multicast from focus locales (where events of interest occur) to subscribers. In contrast to *multicast*, in which the rate at which data is sent to a set of subscribers is the same, in *asynchronous multicast*, the rate at which data is sent to each subscriber is not necessarily the same. Hence, data may need to be buffered at intermediate nodes of the tree and forwarded asynchronously at a different rate from the one at which it was received. We call nodes that buffer and asynchronously forward data, *data caches*.

Given a publisher (with a corresponding data generation rate) and some subscribers (with their respective desired refresh rates), we want to reduce the traffic in the network. As noted earlier, organizing data dissemination paths as a tree leads to a more efficient use of bandwidth and energy supply than unicast. The nodes of the tree are used as caches for buffering and dissemination of data. In a wireless sensor network, since a large number of nodes are scattered in an area of interest, there is considerable freedom in the selection of caches' locations. We pick those nodes for caching data that minimize the cost of the tree. This problem is a variant of the **Steiner Problem** which is defined as follows: Given a finite set of points, P , in a plane (or in some other metric space), find a network that connects all the points of the set

with minimal length [20]. The solution of this problem is a tree that is called Steiner Minimal Tree (SMT). The tree may contain points other than those contained in the set P ; such points are called *Steiner points*. [26] showed that the Steiner Minimum Tree problem is NP-hard.

For our problem, to find the locations of caches, we consider Steiner trees whose edges are weighted by the traffic rate *in addition to* their Euclidean length. We refer to these trees as Steiner Data Caching Trees (SDCT). The SDCT that has the minimum cost is called the Minimum Steiner Data Caching Tree (MSDCT). MSDCT reduces to Steiner Minimal Tree (SMT) when the data refresh rates are equal on all edges of the tree. In the absence of channel irregularities, packets are forwarded from one cache to another on a path with a number of hops that is roughly proportional to the Euclidean distance between the two cache locations. Observe that this proportionality assumption does not imply that packets are routed on a minimum-hop path or even on a straight-line path. Individual hop distances along a path between two caches might be much smaller than the radio range (to eliminate poor channels). Hops might not be perfectly aligned in the same direction. However, given a long enough path, it is intuitive that the number of hops traversed will generally increase with increasing distance to destination. Hence, we take Euclidean length as a metric of path cost.

6.3 ASSUMPTIONS

Construction of SDCT requires nodes other than the publisher and subscribers for data caching. It is assumed that the network has enough additional nodes to construct a binary tree. For the sake of simplicity of mathematical analysis, this study assumes uniform distribution of nodes spread over a two-dimensional surface. One of our simulation studies, Section 6.6.6, shows that the energy conserving properties of the SDCT is quite robust in the presence of node distribution irregularities.

In the analytical treatment, we assume that the noise and packet loss characteristics of the network are uniform across the network. This allows us to use the same cost metric throughout the network. We do not consider the effects of congestion. Localized radio and terrain irregularities may weaken the energy saving properties of our approach. We assume that the subscriber nodes connect to the multicast tree for a long enough duration so that the cost of cache migration remains negligible.

After a brief description of the data-caching service model, we shall continue our discussion of SDCTs.

6.3.1 Service Model

Consider a data caching tree where nodes represent cache locations (as well as the locations of the source and final destinations) and edges connect communicating nodes. While each edge is itself a multi-hop path, we find it useful to abstract it by a single path-cost-value that is proportional to the product of its length and rate of data transmission. Observe that all nodes other than the source and final destinations are Steiner points (caches) added to the tree to reduce cost. It is desired to find the most appropriate locations for hosting these caches (i.e., to find the minimum Steiner data caching tree, MSDCT).

In the following, we prove that an MSDCT is binary and at-least two of the three internal angles at any given internal node (Steiner point) are equal. Since in binary trees each node has three neighbors, we derive expressions that determine the exact location of the Steiner point for three nodes. Each data cache uses this result to locate itself optimally with respect to its three neighbors on the binary tree, hence optimizing local cost. Collectively, the overall result is a Steiner tree structure that reduces the cost of asynchronous multicast from the source to all destinations for the given data generation rate and requested refresh rates.

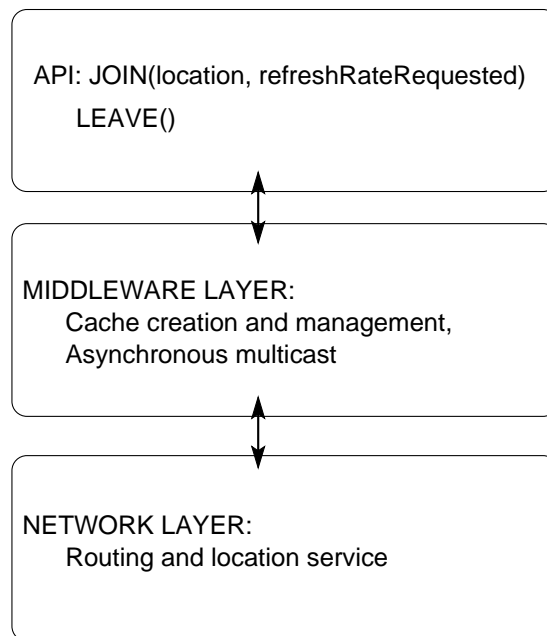


Figure 6.1 Middleware Architecture

We use these results to develop an application-layer service (Section 6.5) that conserves energy by caching data at near-optimal locations, and sending asynchronous multicast to the subscribers from dedicated caches. As mentioned above, the publisher, the intermediate cache nodes, and the subscribers form the asynchronous multicast tree. Data caches located at the nodes of the multicast tree are updated in a lazy fashion. Data being communicated is non-accumulative, and hence only its most recent copy is kept in the caches. The architecture of the data-caching middleware is shown in Figure 6.1. It provides these main functionalities:

1. It reconstructs the data caching tree dynamically and in a distributed fashion upon joining and leaving of subscribers or changes in their data refresh rate requests
2. It sends data to the subscribers from dedicated caches instead of the original source(s)
3. It finds near-optimal locations for creating caches that minimize the energy expended in data dissemination

4. It sends data to the subscribers at optimal rates

6.4 PROPERTIES OF MINIMUM STEINER DATA CACHING TREE

In this section, we study the properties of a minimum Steiner data caching tree (MSDCT) for asynchronous multicast. First, we shall formulate rules to determine the optimum refresh rates on the edges of the tree. As we shall see in the next subsection, an optimal assignment of data refresh rates on the branches of the SDCT leads to elegant symmetries in the tree structure; specifically, two of the three internal angles formed at any node of the tree are always equal, and these angles can be expressed as a function of the refresh rates. We shall then derive an exact expression for the location of a Steiner point for a set of three nodes. At the end of the section, we prove that the MSDCT is binary. In the next section, this result will be used to formulate a heuristic for dynamically constructing and managing a near-optimal SDCT.

6.4.1 Refresh Rate Rules

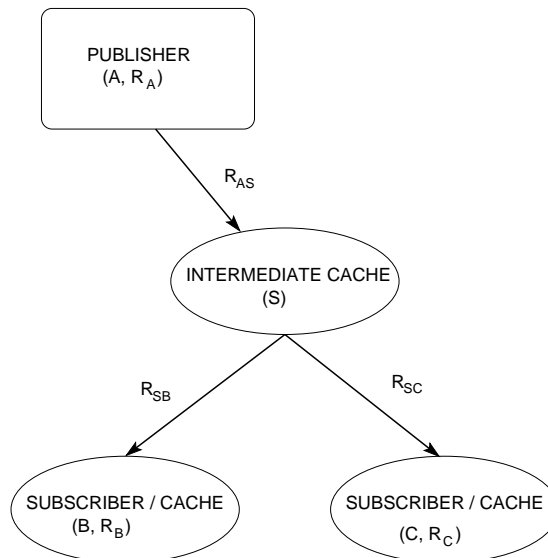


Figure 6.2 A cache S serving two nodes B and C . The cache receives data from A .

As shown in Figure 6.2, let A represent the sensor node sending data to S . S is the cache that serves B and C , where B and C are either caches or subscribers. Let R_A , R_B , and R_C be the requested data refresh rates of A , B and C respectively.¹

The optimal data refresh rates on the edges are determined on the basis of the following two observations:

- ◆ The maximum refresh rate on any of the branches can not exceed the rate at which data is generated by the publisher
- ◆ Data need not be sent at a higher rate than the subscriber's requested rate

The following equations reflect these observations. We base the rest of this paper on these equations, and refer them as the **Refresh Rate Rules**.

$$\textbf{Rule-1:}$$
 If $R_A \geq R_B \geq R_C$ then $R_{AS} = R_B = R_{SB}$; $R_{SC} = R_C$ (6.1)

$$\textbf{Rule-2:}$$
 If $R_B \geq R_A \geq R_C$ then $R_{AS} = R_A = R_{SB}$; $R_{SC} = R_C$ (6.2)

$$\textbf{Rule-3:}$$
 If $R_B \geq R_C \geq R_A$ then $R_{AS} = R_A = R_{SB} = R_{SC}$ (6.3)

Similarly, for the case when $R_C \geq R_B$. That is, at least two edges always have the same refresh rate. As we shall see in the following theorems, applying these rather simple rules to the data caching problem yields elegant symmetries. Next, we describe the cost metric of SDCTs.

6.4.2 Cost Metric of SDCT

We set the weight w_{AB} of the edge connecting two nodes A and B as:

$$w_{AB} = d_{AB} * R_{AB} \quad (6.4)$$

¹We use the following notation in this paper:

R_N = Requested data refresh rate of node N .

$R_{N_i N_j}$ = Actual refresh rate on the edge connecting N_i and N_j .

where d_{AB} is the Euclidean distance between the nodes, and R_{AB} is the data refresh rate on the edge. This equation expresses the cost metric of SDCTs in a network of uniformly distributed nodes, free from congestion or other irregularities. The metric states that the number of transmissions (per unit time) along a path grows with both path length and refresh rate. It does not make explicit assumptions, however, on individual hop length, routing policy, or path shape.

As mentioned in the previous section, congestion and non-ideal radio channels may call for a modification of this metric. In practice, it is good to avoid long hops, since they are usually less reliable and entail more retransmissions. In the following treatment, we assume that long-hop neighbors are black-listed and non-reliable links are eliminated from routing tables. Hence, all remaining links are reliable.

6.4.3 Internal Angles and the Steiner Point

In this section we prove that at-least two of the three internal angles formed at the Steiner point by a set of three nodes are equal. We follow the proof by deriving the expression for the internal angles in terms of the refresh rates, and use the result to find the Steiner point.

Theorem 6.1 (Internal Angle Theorem) *For a given set of three vertices $\{A, B, C\}$ and their corresponding Steiner point S , if the cost metric of an edge is defined as the product of its Euclidean length and data traffic rate, and the data traffic rates are determined by the Refresh Rate Rules, then at-least two of the internal angles formed at the Steiner point S are equal.*

Proof Let $c_{ABC}(S)$ be the cost of the tree T formed by $\{A, B, C, S\}$, as shown in Figure 6.3. Then $c_{ABC}(S) = AS * R_{AS} + SB * R_{SB} + SC * R_{SC}$. The three internal angles at S are: $\angle ASB$, $\angle ASC$, and $\angle BSC$. The three refresh rates R_{AS} , R_{SB} , and R_{SC} on the branches are determined using the Refresh Rate Rules (6.1 - 6.3). WLOG, let R_C be the smaller of $\{R_B, R_C\}$. First, we consider the case $R_A \geq R_B \geq R_C$.

Application of the rule 1 implies $R_{AS} = R_B = R_{SB}$, $R_{SC} = R_C$. We normalize the rates $R_{AS} = R_{SB}$ to 1. Therefore, $R_{SC} = R_C/R_{SB}$. Next, if $R_B \geq R_A \geq R_C$, then the rule 2 implies $R_{AS} = R_A = R_{SB}$, $R_{SC} = R_C$. After normalizing the rates $R_{AS} = R_{SB}$ to 1, once again we get $R_{SC} = R_C/R_{SB}$. Finally, if $R_B \geq R_C \geq R_A$, all three refresh rates on the edges are equal, and hence upon normalizing, they all are 1. We denote R_{SC} by r . Clearly, $r \leq 1$.

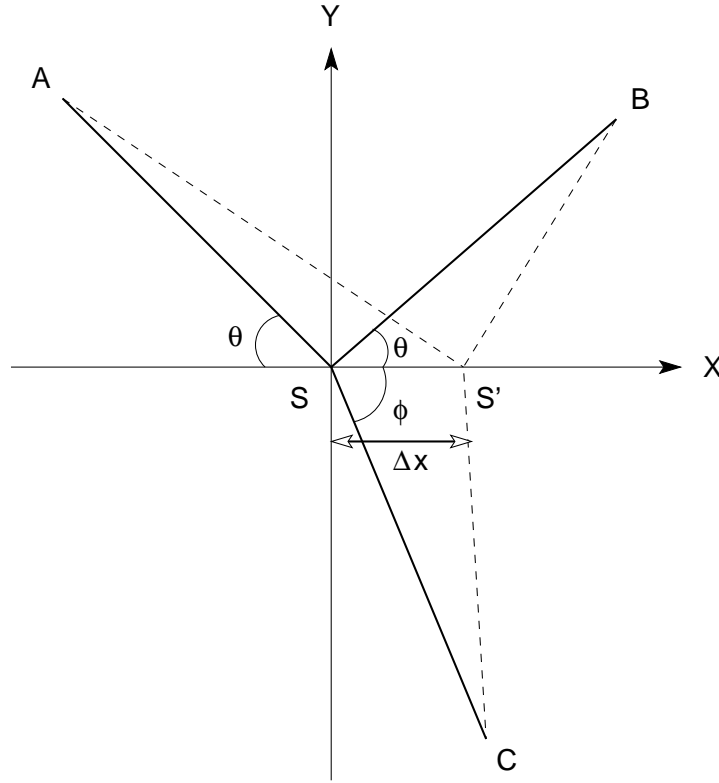


Figure 6.3 Construction to obtain $\partial c_{ABC}(S)/\partial x$. We prove that $\phi = \pi/2$.

Let us draw a line through S that bisects the $\angle ASB$, as shown in the figure. We begin with assumption that $\angle S'SC = \phi$, where ϕ is some arbitrary angle. We shall prove in the following that $\angle S'SC = \pi/2$, that is C lies on the bisector of $\angle ASB$.

The cost of T in terms of normalized rates is: $c_{ABC}(S) = AS + SB + rSC$. For the sake of brevity, we shall denote $c_{ABC}(S)$ by $c(S)$ in the following. Since S is

the Steiner point, $c(S)$ is also the minimum. Hence $\delta c(S) = 0$ for infinitesimal displacements around S . Any such displacement can be decomposed into its X and Y components yielding $\frac{\partial c(S)}{\partial x} = 0 = \frac{\partial c(S)}{\partial y}$. For simplifying the proof, we shall set the bisector of the $\angle ASB$ to be the Y -axis. We now consider $\frac{\partial c(S)}{\partial x}$. Let S' be a point on the X -axis such that $SS' = \Delta x$. Let $\angle BSS' = \theta$.

From the cosine law of triangles,

$$\begin{aligned}
 AS'^2 &= AS^2 + \Delta x^2 - 2 AS \Delta x \cos(\pi - \theta) \\
 &= AS^2 + \Delta x^2 + 2 AS \Delta x \cos \theta \\
 \text{or, } AS' &= AS \left(1 + \frac{2 \cos \theta}{AS} \Delta x + \frac{1}{AS^2} \Delta x^2\right)^{1/2} \\
 &= AS \left(1 + \frac{\cos \theta}{AS} \Delta x + O(\Delta x^2)\right) \text{ using binomial expansion} \\
 O(\Delta x^2) &\text{ denotes a series whose terms contain } \Delta x^n, \text{ where } n \geq 2 \\
 &= AS + \Delta x \cos \theta + O(\Delta x^2) \tag{6.5}
 \end{aligned}$$

Similarly,

$$BS' = BS - \Delta x \cos \theta + O(\Delta x^2) \tag{6.6}$$

$$CS' = CS - \Delta x \cos \phi + O(\Delta x^2) \tag{6.7}$$

Therefore,

$$\begin{aligned}
 c(S') &= AS' + BS' + rCS' \\
 &= AS + BS + rCS - r\Delta x \cos \phi + O(\Delta x^2) \\
 &= c(S) - r\Delta x \cos \phi + O(\Delta x^2)
 \end{aligned}$$

$$\begin{aligned}
\Rightarrow \frac{\partial c}{\partial x} &= \lim_{\Delta x \rightarrow 0} \frac{c(S') - c(S)}{\Delta x} \\
&= \lim_{\Delta x \rightarrow 0} [-r \cos \phi + O(\Delta x)] \\
&= -r \cos \phi
\end{aligned} \tag{6.8}$$

Therefore, $\partial c / \partial x = 0$ if $\phi = (2n + 1)\pi/2$, $n \in \mathbb{Z}$. By construction, $\phi \in [0, \pi]$. Hence, $n = 0$, and $\phi = \pi/2$ is the only admissible solution. Moreover, since ϕ , the root of $\partial c / \partial x = 0$, is univalued, $\phi = \pi/2$ is also the *global minimum*. Therefore, C lies on the Y -axis, or, $\angle ASC = \angle BSC$. This completes the proof. \square

6.4.4 Expressions for Internal angles and the Steiner Point

We now derive expressions for internal angles in terms of the normalized refresh rate ratio r and the exact location of the Steiner point, for a set of three nodes. Please refer to Figure 6.4 for the following derivations. From the Internal Angle Theorem, $\angle ASC = \angle BSC$. Consider a point S'' along the Y -axis at distance Δy from S . Since $c(S) = AS + SB + rSC$ is the minimum, $\partial c(S) / \partial y = 0$.

Similar to Equation 6.5,

$$AS'' = AS - \cos \alpha \Delta y + O(\Delta y^2) \tag{6.9}$$

$$BS'' = BS - \cos \alpha \Delta y + O(\Delta y^2) \tag{6.10}$$

$$CS'' = CS - \Delta y \tag{6.11}$$

Therefore,

$$\begin{aligned}
c(S'') &= AS'' + BS'' + rCS'' \\
&= AS + BS + rCS - 2 \cos \alpha \Delta y - r \Delta y + O(\Delta y^2) \\
&= c(S) - 2 \cos \alpha \Delta y - r \Delta y + O(\Delta y^2)
\end{aligned} \tag{6.12}$$

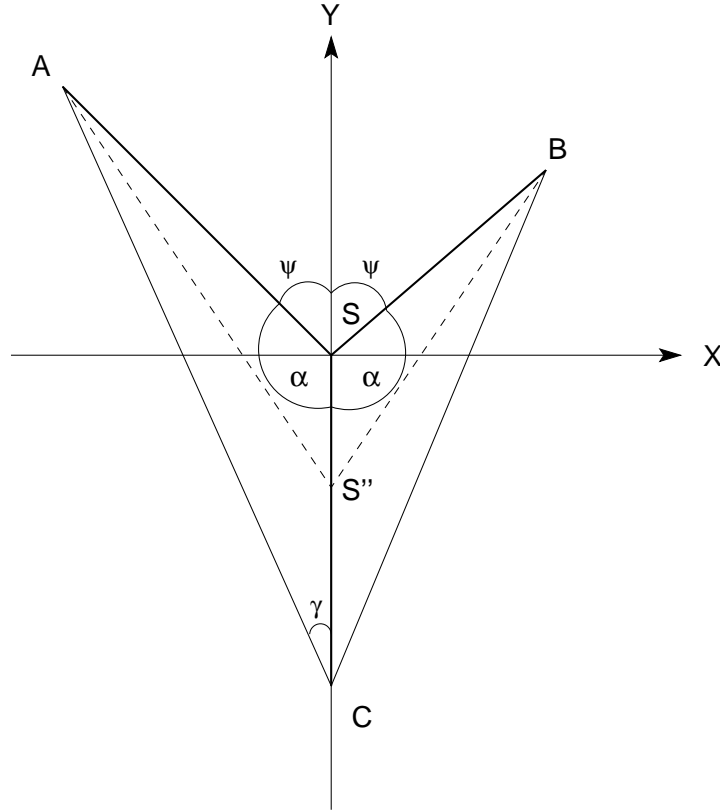


Figure 6.4 Finding expression for the internal angles α , and ψ

$$\begin{aligned} \frac{\partial c}{\partial y} &= \lim_{\Delta y \rightarrow 0} \frac{c(S'') - c(S)}{\Delta y} \\ &= \lim_{\Delta y \rightarrow 0} [-2 \cos \alpha - r + O(\Delta y)] \\ &= 2 \cos(\pi - \alpha) - r \end{aligned}$$

$$\frac{\partial c}{\partial y} = 0 \Rightarrow \alpha = \pi - \arccos(r/2) \quad (6.13)$$

$$\text{or, } \psi = \arccos(r/2) \quad (6.14)$$

Since $0 \leq r \leq 1$, and S lies inside the triangle ABC

$$\pi/2 \leq \alpha \leq \text{MIN}(2\pi/3, \pi - \angle ACB/2) \quad (6.15)$$

This result is the most distinguishing feature of MSDCTs. In contrast to MSDCTs, the Euclidean SMTs have exactly one value for all the three internal angles. Note that when $r = 1$, $\alpha = 2\pi/3$; and hence all the three internal angles at S are equal to $2\pi/3$. This corresponds to the Cavalieri's (1647) famous result for the Torricelli points² in which he proved that each of the internal angles at a Torricelli point equals $2\pi/3$.

We now find expression for the Steiner point S :

Consider the triangle ABC .

$$\angle BAC = \arccos\left(\frac{b^2 + c^2 - a^2}{2bc}\right) = \theta_A \text{ (say)}, \quad (6.16)$$

where $a = BC$, $b = AC$, and $c = AB$.

Let $\angle SCA = \gamma$, $\angle SBA = \delta$, and $\angle CAS = \beta$. Then,

$$\begin{aligned} \angle SAB + \delta &= 2\alpha - \pi \\ \beta + \gamma &= \pi - \alpha \\ \Rightarrow \delta &= \alpha - \gamma - \theta_A \end{aligned} \quad (6.17)$$

From $\triangle ASB$ and $\triangle ASC$, using Sine rule:

$$\begin{aligned} c / \sin 2\psi &= AS / \sin \delta \\ b / \sin \alpha &= AS / \sin \gamma \\ \text{or, } \sin \delta &= (2b/c) \sin \gamma \cos \psi \\ \Rightarrow \sin(\alpha - \gamma - \theta_A) &= (2b/c) \sin \gamma \cos \psi \end{aligned} \quad (6.18)$$

α and γ together locate S . A fully-worked solution in the Cartesian coordinates can be found in the appendix. In our service, described in Section 6.5, the optimal coordinates of the caches are determined using these expressions, given the locations

²A Torricelli point is same as an Steiner point when the number of nodes is 3.

and refresh rates of its neighbors. In practice, the sensor node nearest to the optimal location becomes the cache.

6.4.5 Maximum Degree of an MSDCT Node

In this section, we shall show that the maximum degree of a Minimum Steiner Data Caching Tree (MSDCT) node is equal to 3, that is, an optimal SDCT is binary.

Maximum Degree Theorem The maximum degree of a Minimum Steiner Data Caching Tree is 3.

Proof Clearly, the degree of a Steiner node in SDCT is a number greater than 2. Let us consider the case of a node of degree 4. In the following treatment, we shall show that such a node can be split into two nodes, each of degree 3 and lower cost. First, we note that at least one of the angles formed at the node is acute.³ As shown in Figure 6.5, suppose that the nodes A and B form an acute angle at S . WLOG, let $R_A \geq R_B$, and $R_C \geq R_D$. We shall use the notation a for AS , a' for AS' , b for BS and so on. We now choose another point S' along SA . For simplicity of the proof, we choose $SS' \ll AS$, which allows us to expand a' and b' in terms of SS' .

$$\begin{aligned}
 c(S) &= aR_A + bR_B + cR_C + dR_D \\
 c(S', S) &= a'R_A + b'R_B + SS'R_{SS'} + cR_C + dR_D \\
 \text{or, } c(S', S) &= aR_A - SS'R_A + bR_B - SS'R_B \cos \theta \\
 &\quad + O(SS'^2) + SS'R_{SS'} + cR_C + dR_D \\
 \text{or, } c(S', S) - c(S) &= SS' * [-R_A - R_B \cos \theta + R_{SS'} + O(SS')] \tag{6.19}
 \end{aligned}$$

³We do not consider the rectilinear Steiner trees.

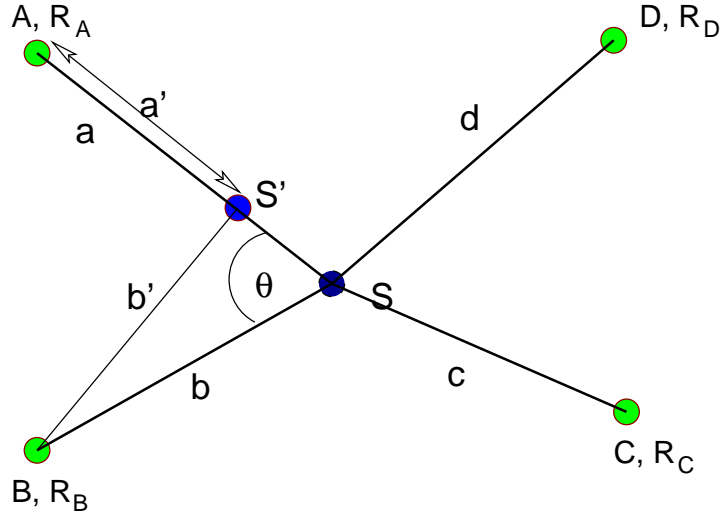


Figure 6.5 Construction to prove binariness of MSDCT. $\{A, B, S', S, C, D\}$ has lower cost than $\{A, B, S, C, D\}$

$R_{SS'} = R_A$ or R_C depending on whether S feeds S' or vice versa. Thus we have two cases:

Case(i) : $R_{SS'} = R_A$

$$c(S', S) - c(S) = SS' * [-R_A - R_B \cos \theta + R_A + O(SS')]$$

$$\Rightarrow c(S', S) - c(S) = SS' * [-R_B \cos \theta + O(SS')] \quad (6.20)$$

since $0 \leq \theta < \pi/2$, $-R_B \cos \theta < 0$

$$c(S', S) < c(S) \quad (6.21)$$

Case(ii) : $R_{SS'} = R_C$

Since data is now flowing from S' to S ,

$$R_C \leq R_A \text{ (Section 6.4.1)}$$

$$c(S', S) - c(S) = SS' * [-R_A - R_A \cos \theta + R_C + O(SS')]$$

$$\Rightarrow c(S', S) - c(S) \leq SS' * [-R_A \cos \theta + O(SS')] \quad (6.22)$$

once again, since $0 \leq \theta < \pi/2$, $-R_A \cos \theta < 0$

$$c(S', S) < c(S) \quad (6.23)$$

Equations (6.21) and (6.23) imply that a Steiner node of degree four can always be reduced to a pair of nodes of degree three and a smaller cost tree. Using similar reasoning, it can be easily shown that a node of degree n , where $n > 4$, can be reduced to a node of degree 3 and another node of degree $n - 1$, and smaller cost. Thus, from the principle of induction, the maximum degree of an MSDCT node is 3 (i.e., the minimum Steiner Data Caching Tree is binary). Therefore, an optimal multicast tree for n_B subscribers consists of exactly $n_B - 2$ caches. \square

We shall use the refresh rate rules, the expression for Steiner points, and the Maximum Degree Theorem - all from this section - to present a heuristic for dynamically constructing and managing a near-optimal SDCT next.

6.5 DYNAMIC CACHE PLACEMENT HEURISTIC

This section describes a distributed cache placement heuristic that constructs and maintains near-optimal SDCT dynamically. We use the refresh rate rules described in Section 6.4.1 to determine the refresh rates at the edges of the tree, and the expressions derived in Section 6.4.4 to compute the Steiner points. We shall construct *binary* SDCTs in accordance with its optimality property (Section 6.4.5).

Consider that a new subscriber B (Figure 6(a)) requests data from the source. We want to determine the location of a node S that can be used as a dedicated cache to serve B . Since the new cache will receive data from the existing multicast tree T , selection of the node of T from which the new cache S receives data is influenced by the requested refresh rate and the location of B . Attaching a high refresh rate cache can result in increase of traffic on multiple edges of a branch of T . To illustrate the point, suppose that the node to which S is attached is N , and that S needs to be refreshed at a (partially or fully satisfiable) rate higher than that of N . Then, attaching S to N results in increase of traffic on the branch of T from N upwards to the root, up-to the first cache that is already being refreshed at a rate equal to or

higher than that needed by S . Therefore, the best N is not necessarily geographically the nearest. However, the cost does depend on the length of edges as well. Thus, the new cost of $T = \text{cost of attaching } B \text{ to } T + \text{increase in the cost of existing } T$. A good heuristic must address both factors.

In the following, we describe a heuristic to place caches in a wireless sensor network that addresses these issues. The heuristic first finds a cache N that offers the smallest increase in the cost of the multicast tree T , as if the requesting subscriber B were to be attached to N directly. A new cache S is then created that serves the requesting node, B , and one of the children of N , the cache to which S attaches itself, thus preserving the binary structure of the tree (Figure 6(b)). Each node on the multicast tree rooted at the publisher node maintains location information of its parent, as well as locations and refresh rates of each of its two children. These location tables route data from the source to the subscribers.

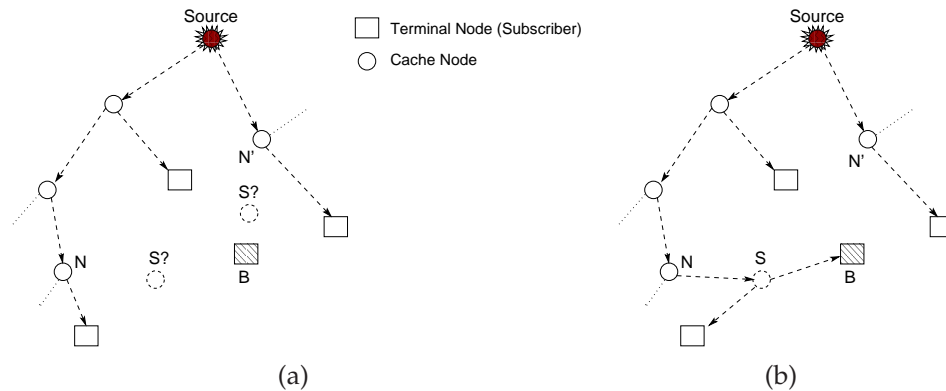


Figure 6.6 Adding a new subscriber. (a) B requests data from the source. N' is nearer to B than N , but N' has a much lower refresh rate. “ $S?$ ’s” indicate two possible choices. Edges of the multicast tree are drawn with dashed lines to represent that multiple intermediate nodes route the packets from one node of the tree to another. (b) The multicast tree after B is attached to it. The new cache S serves B and one of the children of N .

6.5.1 *Joining the Multicast Tree*

A subscriber joins the multicast tree by sending a **join()** message to the the source. The message may be either sent explicitly to the source node if known, or to the region of interest where the elected leader eventually gets the message. The message contains the location of the subscriber and its desired data refresh rate. Figures 6.7 and 6.8 show the pseudo-codes of two routines, SEND-JOIN-REPLY and PROCESS-JOIN-REPLIES, that send and process messages as a result of join requests. The following description of the heuristic contains references to the lines of the pseudo-code marked by curly braces.

{6.7.1 - 6.7.13} Upon receipt of a join() message, the root computes the cost of attaching the new subscriber to itself directly. If none or one of its children are caches, then it sends its node address, cost, and the maximum possible refresh rate to the subscriber. Else, it forwards the message to its children (or, child) *caches*. They compute the increase in cost along the branch connecting them to their parent, and the cost of connecting the subscriber to themselves directly. If this new total cost is larger than the cost received in the parent's message, then the children update cost increase due to refresh rate, and forward the parent's message to their children; otherwise, if the new total cost is smaller, they send their node address, new total cost and the maximum possible refresh rate to their children. {6.7.14 - 6.7.23} If both children of a cache are subscribers, then the process of join message forwarding terminates, and the final message from that branch is sent to the subscriber. {6.8.4 - 6.8.10} Upon receipt of all join reply messages, the subscriber picks the message that contains the least cost, and {6.8.11 - 6.8.14} computes the location of the Steiner point where a new cache will be created. If no node is present at that exact location, which is often the case, the nearest free node is selected.

{6.8.15 - 6.8.22} These steps keep the nodes of the tree locally optimal. For example, in Figure 6(b), after addition of S to the tree, the Steiner coordinates for N

SEND-JOIN-REPLY(*subs-node*, *subs-rr*, *best-node*, *min-cost*, *rr-cost*)

- ▷ The pseudo-code is explained in sections 6.5 and 6.5.1.
- ▷ *subs-node* is the subscriber that requests data from the source
- ▷ *subs-rr* is the refresh rate of the subscriber. All “rr”s mean refresh rates.
- ▷ *best-node* is the least cost option if the subscriber were to connect to this node of the multicast tree directly
- ▷ *min-cost* is the cost of connecting the subscriber node to the *best-node*
- ▷ *rr-cost* is the part of the cost that comes from difference in refresh rate requirement

```

1  if this node is the source node
2      then min-cost ← ∞
3      rr-cost ← NIL
4      message-count ← 1
5      if (source-rr > subs-rr)
6          then subs-rr ← subs-rr
7          else subs-rr ← source-rr
8  child-is-cache ← FALSE
9  connect-cost = DISTANCE(this-node, subs-node) * MAX(subs-rr, source-rr)
10 Increment rr-cost by DISTANCE(this-node, parent-node) * MAX(subs-rr - this-cache-rr, 0)
11 if connect-cost + rr-cost ≤ min-cost
12     then min-cost ← connect-cost + rr-cost
13     best-node ← this-node
14 if both children of this node are caches
15     then Increment message-count by 1 ▷ One more join reply message to the subscriber
16 if the left child of this node is a cache
17     then child-is-cache ← TRUE
18 forward the join request with the updated cost and the node information to the left child
19     ▷ Only the terminal cache sends the accumulated information
20     of the branch to the requesting subscriber
19 if the right child of this node is a cache
20     then child-is-cache ← TRUE
21 forward the join request with updated cost and the node information to the right child
22 if child-is-cache is FALSE ▷ The last non-subscriber node on this branch
23     then Send the message-count, adjusted subs-rr, min-cost, and best-node
    information to the requesting subscriber

```

Figure 6.7 Pseudo-code for processing join() requests

based on its new neighbor and the other two existing neighbors might be different than its current value. If the network is dense enough, the nearest sensor node of the new location for N might actually be a different node. One can easily see that migrating N to this new node will yield a tree of smaller cost. As shown in the pseudo-code, these steps are repeated for the neighboring nodes until the nodes get shifted to their locally optimal location. Since the heuristic constructs a *near*-optimal tree, these steps keep the tree’s departure from optimality over time under control.

```

PROCESS-JOIN-REPLIES(message-count, subs-rr, best-node, min-cost)
  ▷ The pseudo-code is explained in section 6.5.1.
  INIT:
  1 max-messages-to-expect  $\leftarrow$  1
  2 curr-message-count  $\leftarrow$  0
  3 curr-min-cost  $\leftarrow$   $\infty$ 
  ▷ END OF INIT
  4 while max-messages-to-expect  $\neq$  curr-message-count
  5 do Increment curr-message-count by 1
  6   max-messages-to-expect  $\leftarrow$  MAX(max-messages-to-expect, message-count)
  7   Adjust the requested refresh rate to subs-rr
  8   if curr-min-cost  $>$  min-cost
  9     then curr-min-cost  $\leftarrow$  min-cost
  10    curr-best-node  $\leftarrow$  best-node
  ▷ FIND-STEINER-POINT locates the Steiner point based on the results of section 6.4.4
  ▷ A fully-worked calculation in Cartesian coordinates can be found in the appendix 7.3
  11 S1  $\leftarrow$  FIND-STEINER-POINT(this-subs-node, curr-best-node, curr-best-node.left-child)
  12 S2  $\leftarrow$  FIND-STEINER-POINT(this-subs-node, curr-best-node, curr-best-node.right-child)
  ▷ Out of S1 and S2, create cache at the location that has smaller local cost.
  13 new-cache  $\leftarrow$  MIN-COST-NODE(S1, S2)
  14 create cache at new-cache and inform the neighbors about the new location
  15 mark{new-cache.parent, new-cache}
  16 while change in the cost of the multicast tree is not within specified tolerance
  17 do for all nodes v that are caches and are marked
  18   v'  $\leftarrow$  FIND-STEINER-POINT(v.parent, v.left-child, v.right-child)
  19   if v  $\neq$  v'
  20     then mark{v'.parent, v', v'.left-child, v'.right-child}
  21     migrate the cache at v to v', and inform the neighbors about the new location
  22     unmark{v}

```

Figure 6.8 Pseudo-code for processing join() replies

The Table 6.1 shows the savings in the cost of some trees (for the same setting as described in Section 6.6) as a result of these migrations.

As the concluding step of the join procedure, the subscriber sends messages to the new cache, the old cache, and as applicable - to all other affected nodes about any changes in the location pointers and refresh rates of their parents and children.

{6.7.14 - 6.7.15, 6.8.6} To determine the number of messages that the subscriber should expect to receive, a counter is included in the messages that get propagated from the source in response to the join request. This counter is incremented each time a message reaches a node whose both children are caches. The subscriber keeps track of the maximum of the expected number of responses reported. When the

Radius	Tree A	Tree B	Tree C
0	98.4138	74.35	70.8751
1	94.7760	74.1424	70.5885
≥ 2	94.7760	74.1424	70.4812

Table 6.1 The cost of a multicast tree as neighboring caches within a given radius are shifted to new locally optimal locations upon joining of a new subscriber. The radius within which the caches are shifted (shown in the column1) is in terms of the number of edges from the newly created cache.

number of replies equals the maximum expected, it executes the process of computing the Steiner point, cache creation, and tree updating.

6.5.2 *Leaving the Multicast Tree*

When a subscriber leaves a multicast tree, unless the tree's binary structure is restored, the tree is left with one cache that has only one child. A subscriber leaves the multicast tree by sending a **leave()** message to its parent. The parent then executes the process of computing the new Steiner point for its other child, its parent, and its sibling. It then executes the process of cache creation, restoration of local optimality, and tree updating.

6.5.3 *Change in Refresh Rate*

Since the locations of the nodes of the data caching tree depend on the data refresh rates on the branches, the tree needs to be restructured if the refresh rate requirements of a subscriber change. When a subscriber wants to change its refresh rate, it first sends a **leave()** message to its parent, and then it sends a **join()** message to the source.

6.6 EVALUATION

This section consists of two parts. In the first part, Section 6.6.1, we evaluate the performance of our SDCT heuristic (Section 6.5) with respect to MSDCT obtained

by exhaustive search. In the second part, Section 6.6.2 - Section 6.6.6, we evaluate our data caching service using the GloMoSim [28] network simulator. We compared the performance of SDCTs constructed using the SDCT heuristic against unicast, in which the publisher sends updates to the subscribers directly without any caching along the route. Since unicast is the simplest method to disseminate data asynchronously, any other protocol must outperform unicast to merit consideration. We also compared SDCTs against binary greedy data caching trees, in which caches are created greedily instead of at the Steiner points. This greedy protocol is similar to the SDCT heuristic except for the steps {6.8.11 - 6.8.12} of the PROCESS-JOIN-REPLIES. Instead of computing the Steiner point, the greedy protocol locates a node that lies one hop away from the best node towards the subscriber on the line joining the two. If the two are closer than one hop, a neighbor of the best node is selected. We used three routing protocols for evaluation - Ad hoc On Demand Distance Vector (AODV) [50], Greedy Perimeter Stateless Routing (GPSR) [40], and Geographic Forwarding (GF) [40]. GF computes a neighbor table, and updates it periodically. GF forwards the packets to nodes that are geographically nearest to their destination. GPSR improves upon GF by adding capability to route around voids in a network. We used the 802.11 MAC protocol throughout our simulations. The size of the simulated network was $5000\text{m} \times 5000\text{m}$, and it had 784 nodes. The nodes were distributed uniformly in the network.

As noted earlier, we carried out the experiments using the GPSR routing algorithm in addition to GF and AODV. In those experiments when the network contained no voids, the graphs for GPSR almost overlapped those for GF. Hence, for the sake of clarity of presentation, we have omitted those GPSR plots. Since GF and GPSR send beaconing packets to neighbor nodes periodically to determine their alive neighbors, the number of messages sent was non-zero even when there were

no subscribers. We used the following data for MICA-II motes published by its manufacturer Crossbow [21] to calculate the power consumption by the radio unit:

- ◆ Current drawn while transmitting: 16mA
- ◆ Current drawn while receiving: 8mA
- ◆ Bandwidth: 38,400 baud
- ◆ Packet size: 56 byte

We measured the number of packets at the radio layer, and used the average number of messages sent as the main cost metric. Since, in most cases, variations in the proportionality constant between the number of packets received by the nodes of the network and the number of packets sent was small, and we also report approximate values of average power consumption on the mirror Y -axis. The [12] paper evaluates a heuristic very similar to the near-optimal SDCT heuristic presented in this paper on MICA-II motes.

6.6.1 Performance Ratio of the SDCT Heuristic

In this section, we compare the cost of SDCT produced using the heuristic presented earlier with the cost of the minimum SDCT (or, MSDCT). It can be easily seen from the pseudo-code of the heuristic that the topology of the SDCT that it constructs depends on the order of arrival and departure of the subscribers. For a set of n subscribers, the total number of arrival orderings is equal to $n!$. Figure 6.9 shows the performance ratio (defined as the ratio of the cost of the tree constructed by the heuristic to the cost of a minimum tree) of the SDCT heuristic as a function of the number of the subscribers. All subscribers were located within a narrow circular band from the source. We collected data for a maximum of 2000 orderings for each number of subscribers. Thus the maximum points do not represent the absolute worst cases. The next two plots of Figure 6.10 show the distribution of data points for the cases where the numbers of subscribers were equal to 10 and 20 respectively.

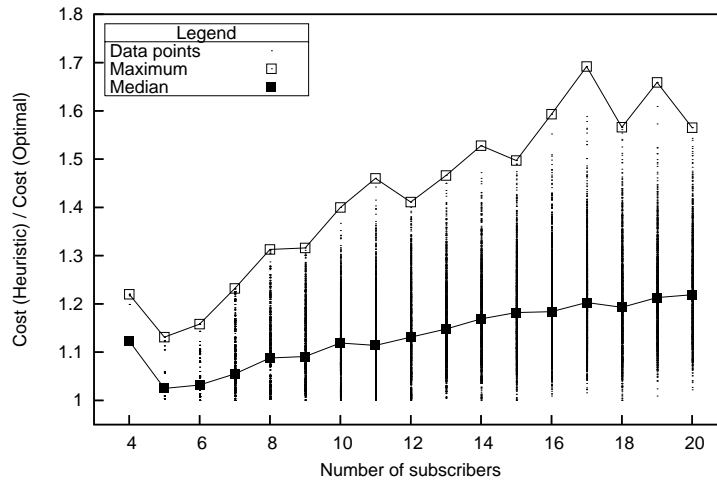


Figure 6.9 Performance ratio of the SDCT heuristic vs. number of subscribers

As the histograms show, large performance ratios are distributed exponentially. Thus, although it is possible that the cost of a tree generated by the heuristic can have a higher performance ratio than the maximum shown in Figure 6.9, the probability of occurrence of such cases is extremely small. The median points of the figure 6.9, and the frequency distribution of the histograms assure that in most cases, the cost of the tree generated by the heuristic remains within 2% of the minimum cost *per subscriber*.

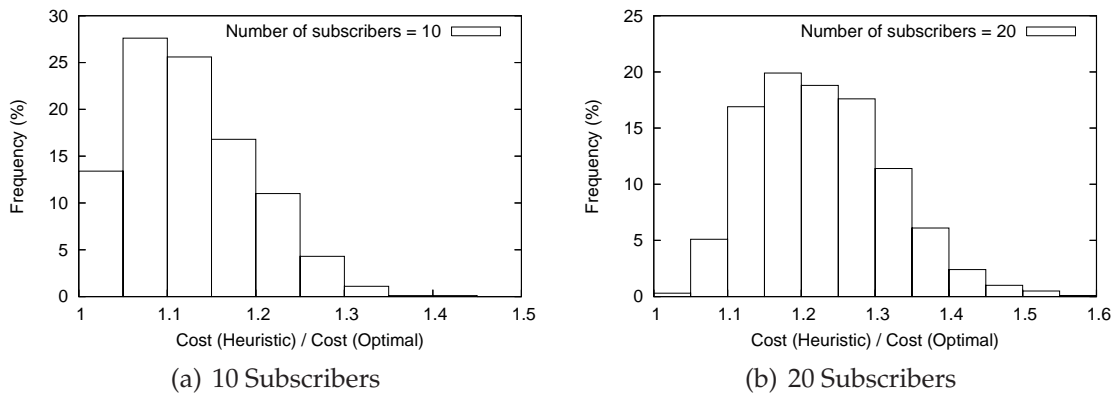


Figure 6.10 Histograms of performance ratio of the SDCT heuristic

6.6.2 Number of Subscribers

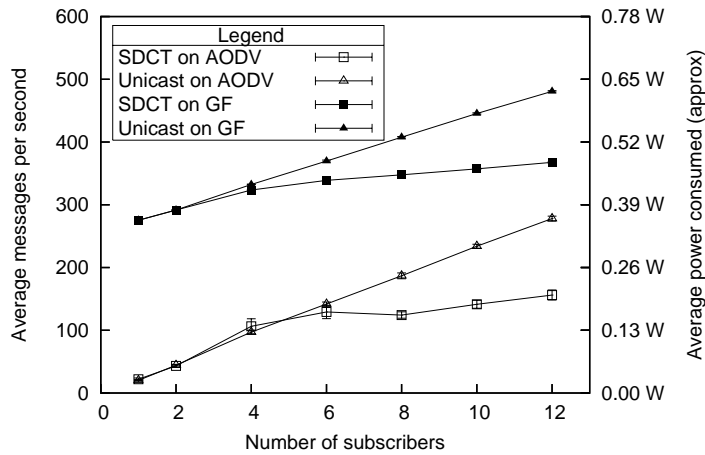
This first simulation experiment studies the performance of the middleware as a function of the number of subscribers. The subscribers were located at approximately the same distance from the source, and were spread out in a region large enough (a region subtending 180° at the source) to not cause excessive collisions. Figure 6.11 shows the average number of messages sent by the nodes per second as the number of subscribers varied. The error bars in this and all subsequent figures represent 95% confidence interval for the metric on the main Y -axis.

The SDCT and greedy heuristics create a cache when the number of subscribers exceeds 2. Therefore, the plots overlap each other from 1 to 2 subscribers. The gap between AODV and GF curves is in-part due to the periodic beaconings sent out by the GF routing protocol. The (extrapolated) gap at 0 subscribers is purely due to the GF beaconings.

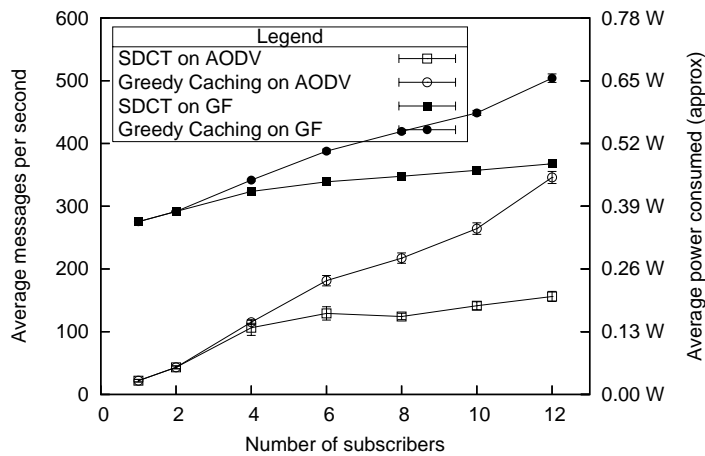
As the plots suggest, the benefit of data caching is small when the number of subscribers is small, but the cost saving gets more and more pronounced as the number of subscribers increases. Compared to unicast and greedy caching trees, SDCTs saved about 50% of the cost of data dissemination when the number of subscribers was 10. The graphs suggest further increase in the energy saving as the number of subscribers increases.

6.6.3 Spatial Distribution of the Subscribers

Spatial distribution of traffic in the wireless sensor network affects energy consumption. Traffic spread over larger area means larger SDCT, and hence increased cost. In this experiment, we simulated the network with varying degree of spatial distribution of the subscribers. The cost of the multicast tree as a function of the spatial distribution of the subscribers is shown in Figure 6.12. In the figure, x° clustering means that all the subscribers were located in a sector that subtended an angle x°



(a) SDCT vs. unicast



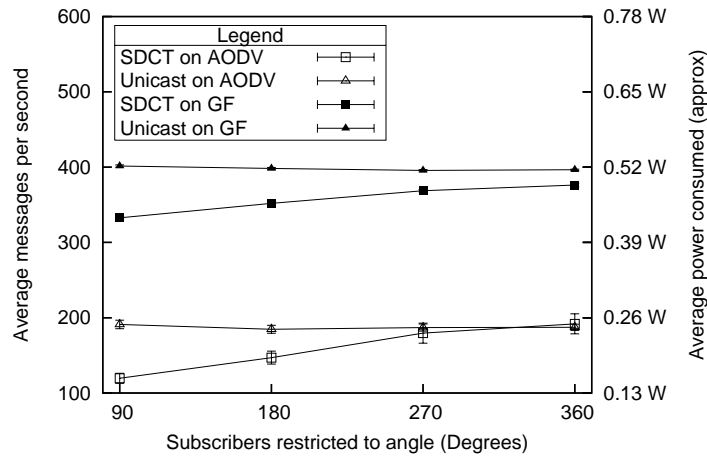
(b) SDCT vs. greedy protocol

Figure 6.11 Average number of messages sent per unit time vs. number of subscribers

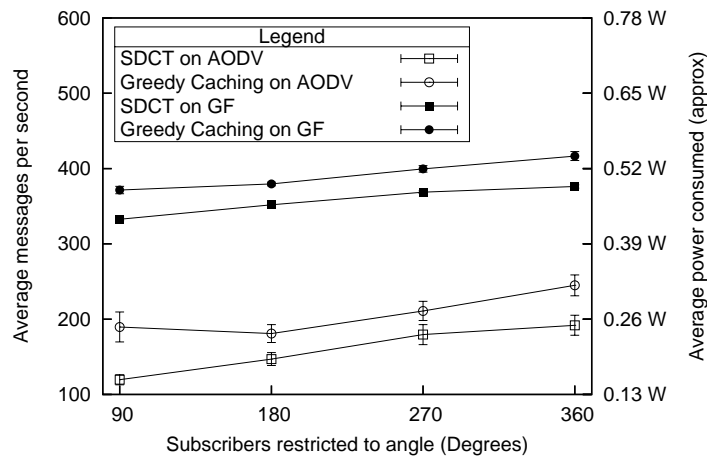
at the source node. Since the number of subscribers was small (10), at higher clustering angles, the multicast tree had similar cost as the unicast tree. Hence, we see the curves for unicast and data caching trees converge as the bases get spread over a larger area.

6.6.4 Effect of Subscriber Dynamics

Each time a subscriber joins or leaves the network, the SDCT needs to be restructured. In this experiment, we evaluate the overhead of restructuring. Figure 6.13 shows the effect of frequency of tree restructuring on the average number of



(a) SDCT vs. unicast

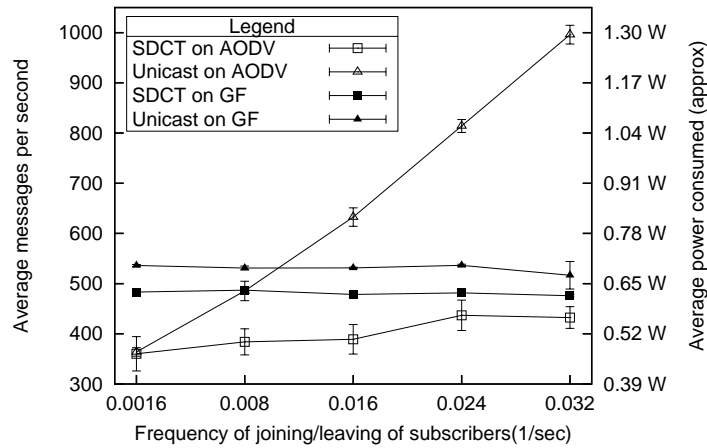


(b) SDCT vs. greedy protocol

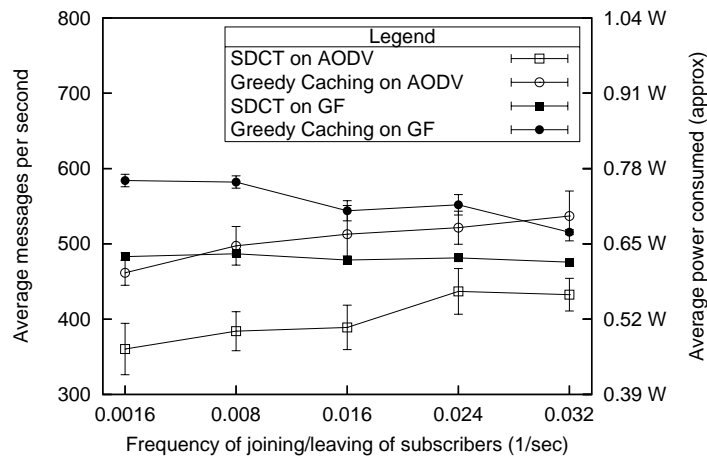
Figure 6.12 Average number of messages sent per unit time vs. clustering of the subscribers messages sent by the nodes per second. Leaving of a subscriber was followed by joining of another in the same neighborhood to keep the cost of the tree unchanged. The curves show that the restructuring overhead of SDCTs is small, and that SDCTs perform better than greedy caching trees. The data for GF is almost immune to the dynamics of the multicast group because it uses static routing tables.

6.6.5 Partitioning of the Network

Table 6.2 shows the performance comparison of SDCT and unicast with respect to the life-time of source nodes and partition-free network. The source nodes were



(a) SDCT vs. unicast



(b) SDCT vs. greedy protocol

Figure 6.13 Average number of messages sent per unit time vs. frequency of joining and leaving of the subscribers

among the first ones to run out of battery. As the source nodes died, we kept replenishing their energy reserve to allow the simulations to continue. In a real deployment, this corresponds to a new leader getting elected at the event location. As expected, the simulations that used GF for routing showed that it takes longer to partition when the traffic is spread over a larger region. However, the partitioning time for unicast was independent of clustering, since in the case of unicast, almost all partitionings arose from the depletion of the energy reserve of the nodes neighboring to the source.

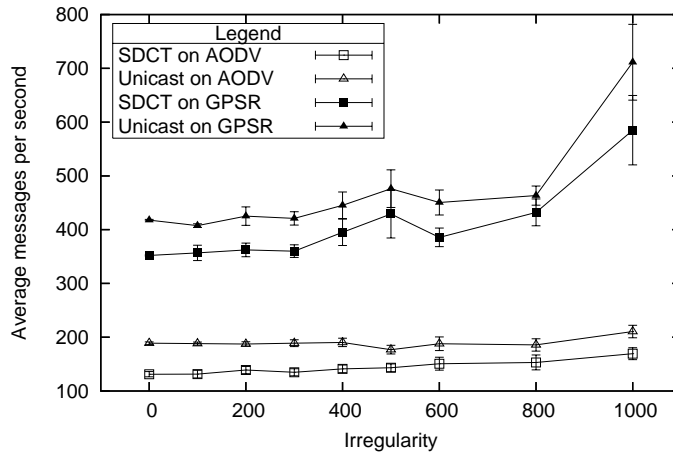
Clustering	90°				180°			
Routing	AODV		GF		AODV		GF	
SDCT	Yes	No	Yes	No	Yes	No	Yes	No
T_{SR}	3384±744	633±9	2586±39	613±9	3269±618	644±12	2420±153	620±9
T_P	> 5000	1045±24	1603±138	883±60	> 5000	1023±33	1833±129	987±27

Table 6.2 Partitioning of the network: T_{SR} is the time at which the source node died. T_P is the time when the network was partitioned. The number ‘> 5000’ indicates that the network remained unpartitioned till the end of simulation (that ran for 5000sec). The errors represent 6σ ($\approx 99.7\%$) confidence interval.

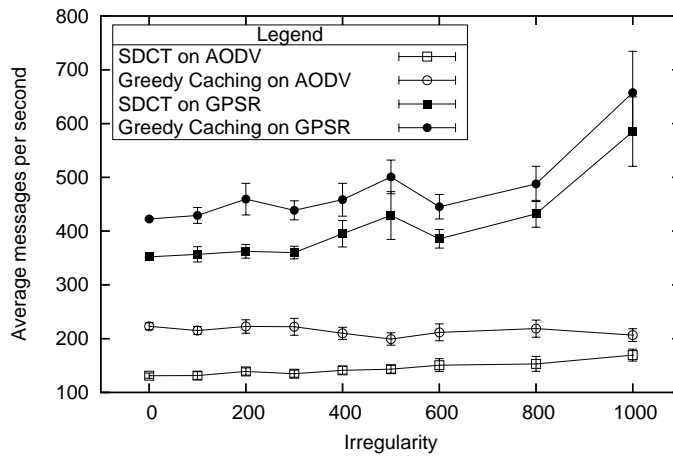
6.6.6 Irregularities in Node Placement

We presented and evaluated the optimality properties of the SDCT heuristic in irregularity free, random, uniformly distributed networks. In reality, as time passes, nodes start to “die” or malfunction. As a result, the distribution of nodes in the network becomes more and more non-uniform. In the regions of sparse or non-uniform node distribution, the probability of finding a node for caching data within a small neighborhood of the Steiner point becomes smaller with increase in the irregularities. Figure 6.15 shows the performance of our SDCT heuristic in the presence of non-uniform node placement in the network. The nodes were given a random deviation around an ideal grid position in both $\pm x$ and $\pm y$ directions. The numbers on the X -axes of the figures (6.14, 6.15) represent the maximum deviation about the ideal position in percentage, for example 100% means that the node would be placed randomly in an area formed by four neighboring squares of the grid centered at the ideal location of the node.

At smaller irregularities, the (routing) cost of the tree edges vary more or less similarly for all the three protocols with the irregularities introduced. Since these costs dominate over change in cost due to deviations from ideal locations in the placement of internal nodes of SDCT or the greedy tree, the difference in the cost remains more-or-less the same at smaller irregularities. As the amount of irregularity increases, performance of all the three protocols become indistinguishable. The



(a) SDCT vs. unicast



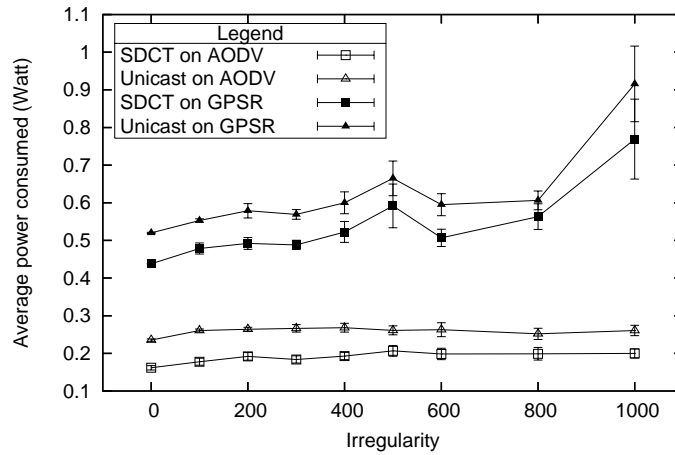
(b) SDCT vs. greedy protocol

Figure 6.14 Average number of messages sent per unit time vs. irregularity in the placement of nodes. The numbers on the x-axis show the maximum deviation from perfect grid in percentage of grid size.

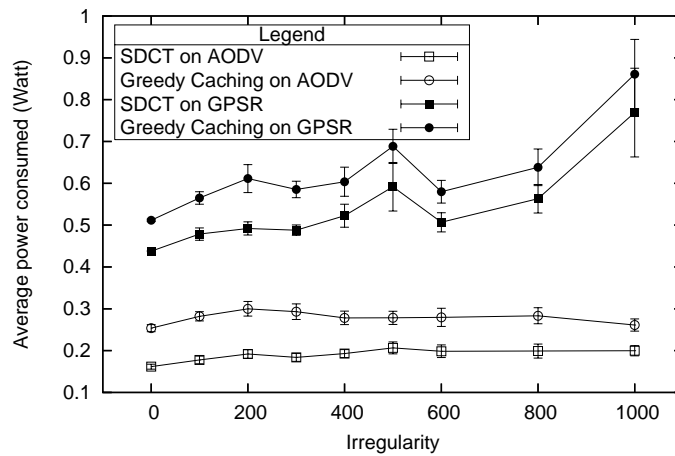
plots in Figures 6.14 and 6.15 show that SDCT is quite robust up-to a very high level of irregularity introduced in the network. Its performance was consistently better than the unicast and greedy multicast trees in irregular networks.

The following list summarizes the factors that influence the energy efficiency of SDCT (as compared to unicast):

- ◆ Number of subscribers : Larger \Rightarrow More effective
- ◆ Data refresh rates : Larger \Rightarrow More effective



(a) SDCT vs. unicast



(b) SDCT vs. greedy protocol

Figure 6.15 Power consumption vs. irregularity in the placement of nodes

- ◆ Distance between source and subscribers : Few hops only \Rightarrow Not effective
- ◆ Density of subscribers : Larger \Rightarrow More effective

6.7 STEINER MINIMAL TREES AND MSDCT

We proved in the Section 6.4 that the maximum number of degrees of an MSDCT node is three. This is also true for SMT's. We also showed that at least two of the internal angles formed at an MSDCT node are equal, whereas all the three internal angles of an SMT node are equal, and hence are *exactly* equal to 120° . On the other hand, the internal angles of an MSDCT node have a *range*, given by Equation (6.15).

Therefore, MSDCT is a more general case of SMT. As we mentioned earlier, MSDCT reduces to Steiner Minimal Tree (SMT) when the data refresh rates are equal on all edges of the tree. As mentioned before, the problem of finding the SMT has been shown to be NP-hard. There has been much activity in the direction of finding approximate polynomial solutions to many variants of the Steiner tree problem [17, 57, 11]. The problem of finding Steiner points for data caching in the wireless sensor networks needs a different approach since the subscribers are not known a priori and global knowledge of the locations of all nodes of the network is also not available (at some centralized location). The locations and timings of requests of the subscribers depend on events, and can't be assumed to be known a priori. In general, we want a distributed solution that manages the tree dynamically.

However, in some applications, like monitoring temperature inside a building or a smart nursing home, where networks do not change much over time, it is possible to put the nodes at pre-determined locations. If the refresh rates of the subscribers are more or less the same, one can use some of the polynomial time SMT heuristics, for example the K-SPH heuristic [10], to pre-compute the MSDCTs before starting to disseminate data.

6.8 REAL-TIME GUARANTEES OF SDCT

As we mentioned earlier, unlike data gathering at base-stations, transmission scheduling for data dissemination is an easy problem and hence is not as much an issue as is the minimization of number of transmissions for interference suppression and energy conservation. We presented the solution of energy efficient data dissemination problem in the form of SDCT, and a distributed heuristic that constructs and maintains near-optimal SDCT on-line. The middleware that uses this heuristic to provide the data dissemination service runs on top of the transport layer. Hence the real-time properties of the middleware depend upon the underlying network layers.

The real-time capacity can be expressed in two units of measurement: namely byte-hops/sec and byte-meters/sec. A suitable measure of hop length relates the two units (and vice-versa). The cost of SDCT is defined as the product of traffic rate (bytes/sec) and the Euclidean length (meter) of the edges. Thus, the cost of the tree equals the real-time capacity demand of the tree. Therefore, to determine the real-time feasibility of the SDCT, one needs to compare the real-time capacity of the network and the cost of the tree. If the real-time capacity of the network meets or exceeds the cost of the tree, the data dissemination protocol will be able to give guarantees to the subscribers in terms of fixed data rates. The guarantee on data rate enables bounded latency of transmissions, making the communication real-time.

6.9 LIMITATIONS

In this chapter, we considered the case where the edge weights are proportional to the Euclidean length and transmission rate uniformly across the network. In the presence of congestion or lossy terrains, packets may need multiple re-transmissions or may get routed along a path that deviates significantly from proportionality to distance (e.g., to route around a hot-spot). In these circumstances, the edges of the SDCTs need to be weighted in a location dependent manner to account for the congestion cost. For example, on lossy paths, the edge weights can be expressed as a function of the expected number of retransmissions in addition to the Euclidean length and transmission rate. In the case of networks of highly non-uniform channel noise characteristics, the results of this chapter will not be useful.

Another kind of irregularity stems from non-uniformity of node placement. Through simulation, we showed that the data dissemination protocol is very robust in the presence of this irregularity.

Cache migration cost is not taken into account explicitly in the derivations. Caches keep only the most recent copy of the data. In general, cache migration cost is not

much of a concern since it involves few messages – messages to the parent and children and a data packet containing the most recent copy of the data. In those cases where the cache migration cost is significant, incorporation of this variable in the dissemination protocol is straightforward, but doing so in the theoretical analysis may not be so trivial. Finally, race conditions will arise if two or more join requests originate at nearby locations simultaneously. Only one of the requests may succeed and the unsuccessful requests will need to be retransmitted.

6.10 SUMMARY

Motivated by the importance of the need to save the energy spent on data dissemination in wireless sensor networks, we presented optimality properties of Steiner data caching trees, and an energy-conserving application-layer service for data caching and asynchronous multicast. We showed that the optimal multicast tree is binary. We also showed that the asynchronous multicast protocol presented in this chapter constructs and maintains near-optimal multicast tree, and helps in prolonging the life of the network. The simulation results that we presented in this paper verify the importance of data caching. If the real-time capacity of the network meets or exceeds the cost of the asynchronous multicast tree, the subscribers can get guarantees on the data rate, and hence real-time guarantees on the communication.

CHAPTER 7

Summary and Outlook

In this last chapter, we first revisit the thesis statement and its support from the research presented in this dissertation. Next, we present a discussion of the future directions based on this research. Finally, we present our conclusions.

7.1 SUMMARY

We began with the following thesis statement:

It is possible to develop a systematic theory of real-time capacity of wireless sensor networks, and to construct low overhead protocols for real-time data gathering and data dissemination applications in wireless sensor networks.

I supported the thesis statement by deriving real-time capacity expressions for two well known scheduling algorithms, EDF and DM, one each from the classes of fixed and dynamic priorities; and for the two scheduling algorithms for hexagonal WSN. Real-time capacity depends heavily on the MAC protocol. Only those MAC protocols that support bounded communication latency can provide deterministic real-time guarantees. Prior to this work, there existed some MAC protocols that construct TDMA schedules, but none gave any multi-hop deterministic delay guarantee.

The problem of optimal scheduling in general multi-hop radio networks is NP-Hard. I showed that by using the hexagonal topology, it is possible to construct optimal schedule at no message overhead. The use of hexagonal topology brings

many other important benefits, especially from the perspective of energy consumption. The addressing, routing and clock synchronization algorithms become implicit, thereby reducing the considerable overhead of these necessary management functions. Furthermore, the use of cluster based topology enables the support of dynamic network – node addition, node rotation, arbitrary sleep schedules etc. at almost no cost.

I provided a set of closed form expressions that determine the transmission schedule of a node locally. The case of scheduling convergecast traffic is harder due to the formation of bottleneck at the data aggregation node. But this case also offers less interference, and hence certain parallelism, at farther distances because of the streamlined nature of the traffic, if shortest path routing is used. I used this parallelism to construct the optimal schedule.

Data dissemination is the dual problem of data collection. Steiner tree problem is suitable to model the energy conserving data dissemination problem in wireless sensor networks. Prior to this work, there was no research on the Steiner tree problem where edges are weighted by the traffic rate and Euclidean length. I introduced the Steiner Data Caching Tree (SDCT), a generalized Steiner tree problem, where the tree cost is the function of Euclidean length and the traffic rate. I presented the analysis of the optimal SDCT. I proved that the optimal SDCT is binary and the tree exhibits symmetry of internal angles at its nodes. I also showed that in the limiting case of equal traffic rate on the branches, the optimal SDCT becomes the optimal (Euclidean) Steiner tree. Since the problem is NP-hard, I presented a distributed algorithm that constructs and maintains near-optimal SDCT dynamically. In other words, data subscribers can join and leave the dissemination tree at will. The algorithm maintains the near-optimal structure in such events by local actions. I showed that asynchronous multicast in combination with near-optimal SDCT saves energy and prolongs network lifetime.

7.2 FUTURE WORK

To construct a regular topology WSN, we considered hexagonal topology since it gave elegant formulation to maximize simultaneous transmissions. It, however, requires maintaining six neighbors, in general. Constant time addressing and routing can be done for all simple regular topologies (square, n-ary tree and others.) It would be interesting to see how complex the scheduling algorithms get for such topologies. One may investigate the tradeoff of maintaining a less rigid topology than a hexagon versus the low overhead and optimality of scheduling in hexagonal meshes.

An important future work is to extend the hexagonal WSN for fault-tolerance. An immediate extension to the MAC protocols of Chapters 4 and 5 is the efficient handling of node (cluster head) failures, and mobility of cluster heads. This problem is closely related to the issue of obstacles. Both create “loss” of links. One solution appears to be using neighboring nodes as proxy. This however, may create undesired long links. Modification of the routing and TDMA scheduling algorithms to address the problem of lost messages is another important future direction. Extension of the MAC protocols for the multiple sink case is another problem to investigate. The possibilities include the construction of individual hexagons around each base station or modification of routing and scheduling while maintaining the optimality of the schedule.

Another research direction is to extend the hexagonal WSN for three-dimensional deployment (e.g., in a multi-story building.) Three-dimensional hexagonal WSN can be achieved by either stacking planar ones on top of each other, or by constructing a hexagonal sphere. It would be interesting to find out the strengths and weaknesses of the two approaches.

For the data dissemination problem, it would be interesting to investigate the optimality properties of SDCT when the cache migration cost is taken into account.

Investigation of SDCT where the cost of the tree has localized fluctuations is important to extend the predictability of the energy saving properties of the protocol in non-uniform terrains, where the non-uniformity is with respect to noise, packet loss, hop length etc.

We showed that EDF was not optimal for pipelines. It would be very useful to find scheduling algorithms that can provide near-optimal utilizations. Periodic tasks are a special case of aperiodic tasks. Often, analyzing schedulability for periodic tasks lead to simpler and more elegant analysis and better utilization bounds than that for aperiodic tasks — although at the cost of restricted applicability. The application of periodic task model are numerous, however, to merit an investigation of preemptive as well as non-preemptive EDF scheduling of periodic tasks on pipelines. For the preemptive analysis, it would be interesting to incorporate resource blocking into the analysis.

7.3 CONCLUSIONS

The excessive overhead associated with the packet scheduling problem in multi-hop wireless networks is a result of the intractability of the problem. The overhead can be eliminated or significantly reduced if the topology of the network is made regular. Hexagonal topology offers optimal scheduling. The real-time capacity for data gathering applications scales as the radius of the network (or, data collection sections, if multiple base-stations exist.) This indicates that the law of diminishing returns operates here, and hence makes a case for multiple sections each with a base-station, rather than a large one with a single base station. For data-dissemination, asynchronous multicast has significant advantages in terms of energy conservation.

In a nutshell, regular network topology, localized communication and asynchronous multicast on Steiner like tree are worth considering when designing a wireless sensor network for real-time applications.

APPENDIX A: STEINER POINT IN CARTESIAN COORDINATES

We solve the Equations (6.14), and (6.18) of Section 6.4.4 to obtain expressions for the coordinates of the Steiner point in the Cartesian coordinate system for a set of three given nodes. We obtain the solution by obtaining expressions for the straight lines CS and AS (Figure 7.1), and then their point of intersection.

$$\theta_A = \arccos\left(\frac{b^2 + c^2 - a^2}{2bc}\right) \quad (7.1)$$

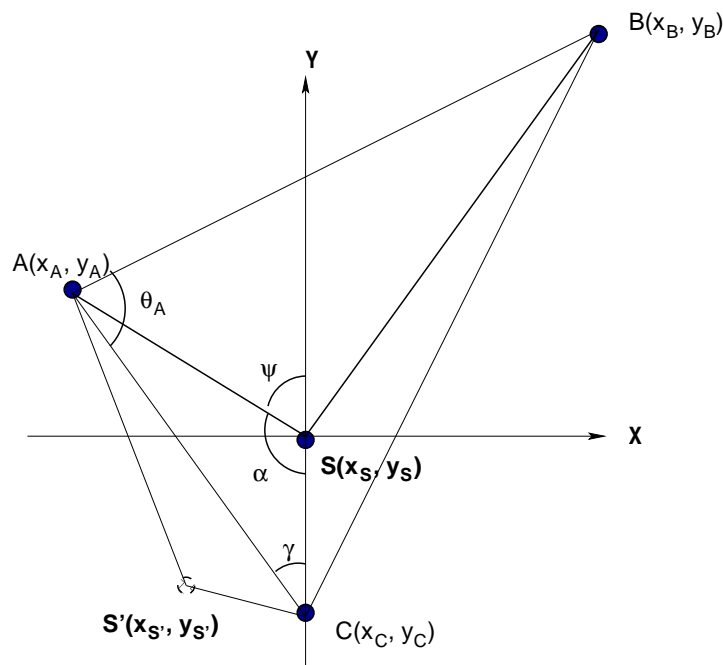


Figure 7.1 Calculating the coordinates of Steiner point in the Cartesian coordinate system.

From equation 6.14:

$$\psi = \arccos(r/2)$$

$$\alpha = \pi - \psi$$

From equation 6.18:

$$\gamma = \operatorname{arccot}\left(\frac{\cos(\alpha - \theta_A) + (2b/c) \cos \psi}{\sin(\alpha - \theta_A)}\right) \quad (7.2)$$

Equation of line AC :

$$y - y_C = m_{AC}(x - x_C), \text{ where}$$

$$m_{AC} = \frac{y_A - y_C}{x_A - x_C}$$

Equation of line CS :

$$y - y_C = m_{CS}^{\pm}(x - x_C), \text{ where} \quad (7.3)$$

$$m_{CS}^{\pm} = \frac{m_{AC} \pm \tan \gamma}{1 \mp m_{AC} \tan \gamma}(x - x_C)$$

Equation of line AS :

$$y - y_C = m_{AS}^{\pm}(x - x_A), \text{ where} \quad (7.4)$$

$$\begin{aligned} m_{AS}^{\pm} &= \frac{m_{AC} \pm \tan(\pi - \alpha - \gamma)}{1 \mp m_{AC} \tan(\pi - \alpha - \gamma)}(x - x_A) \\ &= \frac{m_{AC} \pm \tan(\psi - \gamma)}{1 \mp m_{AC} \tan(\psi - \gamma)}(x - x_A) \end{aligned}$$

Let us define:

$$c_{CS}^{\pm} = y_C - m_{CS}^{\pm} x_C$$

$$c_{AS}^{\pm} = y_A - m_{AS}^{\pm} x_A$$

Since S lies at the intersection of CS and AS , we solve Equations (7.3) and (7.4) to get (x_S, y_S) . Out of the four solutions corresponding to each combination of the two pairs of m^s , only two lie in the region of interest:

$$S^{(1)}(x_S^{(1)}, y_S^{(1)}) : x_S^{(1)} = \frac{c_{CS}^+ - c_{AS}^-}{m_{AS}^- - m_{CS}^+}, y_S^{(1)} = \frac{c_{CS}^+ m_{AS}^- - c_{AS}^- m_{CS}^+}{m_{AS}^- - m_{CS}^+} \quad (7.5)$$

$$S^{(2)}(x_S^{(2)}, y_S^{(2)}) : x_S^{(2)} = \frac{c_{CS}^- - c_{AS}^+}{m_{AS}^+ - m_{CS}^-}, y_S^{(2)} = \frac{c_{CS}^- m_{AS}^+ - c_{AS}^+ m_{CS}^-}{m_{AS}^+ - m_{CS}^-} \quad (7.6)$$

Between $S^{(1)}$ and $S^{(2)}$, one that is closer to B is S , and the other is S' .

BIBLIOGRAPHY

- [1] Tarek Abdelzaher, K. Shashi Prabh, and Raghu Kiran. On real-time capacity limits of multihop wireless sensor networks. In *Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS'04)*. IEEE Computer Society Press, Los Alamitos, CA, 2004.
- [2] Tarek Abdelzaher, Gautam Thaker, and Patrick Lardieri. A feasible region for meeting aperiodic end-to-end deadlines in resource pipelines. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 436–445. IEEE Computer Society Press, Los Alamitos, CA, 2004.
- [3] G. Agrawal, B. Chen, W. Zhao, and S. Davari. Guaranteeing synchronous message deadlines with the timed token medium access control protocol. *IEEE Trans. Comput.*, 43(3):327–339, 1994.
- [4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.
- [5] Gahng-Seop Ahn, Se Gi Hong, Emiliano Miluzzo, Andrew T. Campbell, and Francesca Cuomo. Funneling-MAC: a localized, sink-oriented MAC for boosting fidelity in sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 293–306, New York, NY, USA, 2006. ACM Press.
- [6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):391–552, 2002.
- [7] Erdal Arıkan. Some complexity results about packet radio networks. *Information Theory, IEEE Transactions on*, 30(4):681–685, Jul 1984.

- [8] Dennis J. Baker. Distributed control of broadcast radio networks with changing topologies. In *INFOCOM*, pages 49–55, 1983.
- [9] T. P. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. In *Proc. of the 24th IEEE Real-Time Systems Symposium (RTSS '03)*, pages 120–129. IEEE Computer Society Press, Los Alamitos, CA, 2003.
- [10] Fred Bauer and Anujan Varma. Distributed algorithms for multicast path setup in data networks. *IEEE/ACM Trans. Netw.*, 4(2):181–191, 1996.
- [11] Piotr Berman and Viswanathan Ramaiyer. Improved approximations for the steiner tree problem. In *Selected papers from the third annual ACM-SIAM symposium on Discrete algorithms*, pages 381–408, New York, NY, USA, 1994. Academic Press, Inc.
- [12] S. Bhattacharya, H. Kim, K. Shashi Prabh, and T. F. Abdelzaher. Energy-conserving data placement and asynchronous multicast in wireless sensor networks. In *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, New York, NY, 2003. USENIX.
- [13] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus: a theory of deterministic queuing systems for the Internet*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.
- [14] Marco Caccamo and Lynn Y. Zhang. The capacity of implicit EDF in wireless sensor networks. In *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on*, pages 267 – 275. IEEE Computer Society Press, Los Alamitos, CA, July 2003.
- [15] Qing Cao, Ting Yan, John Stankovic, and Tarek Abdelzaher. Analysis of target detection performance for wireless sensor networks. In *Distributed Computing in Sensor Systems, First IEEE International Conference, Proc. of (DCOSS 2005)*, pages 276–292. Springer, 2005.

- [16] Jean Carle and Jean-Frederic Myoupo. Topological properties and optimal routing algorithms for three dimensional hexagonal networks. *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on*, 01:116, 2000.
- [17] Moses Charikar, Chandra Chekuri, To yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 192–200, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.
- [18] Ming-Syan Chen, Kang G. Shin, and Dilip D. Kandlur. Addressing, routing, and broadcasting in hexagonal mesh multiprocessors. *IEEE Trans. Comput.*, 39(1):10–18, 1990.
- [19] Xiuzhen Cheng, Bhagirath Narahari, Rahul Simha, Maggie Xiaoyan Cheng, and Dan Liu. Strong minimum energy topology in wireless sensor networks: Np-completeness and heuristics. *IEEE Transactions on Mobile Computing*, 2(3):248–256, 2003.
- [20] Dietmar Cieslik. *Steiner Minimal Trees*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.
- [21] Crossbow Technology, Inc. <http://www.xbow.com/>, 2004.
- [22] Catherine Decayeux and David Seme. 3D hexagonal network: Modeling, topological properties, addressing scheme, and optimal routing algorithm. *IEEE Trans. Parallel Distrib. Syst.*, 16(9):875–884, 2005.
- [23] M. L. Dertouzos and A. K. Mok. Multiprocessor online scheduling of hard-real-time tasks. *IEEE Trans. Softw. Eng.*, 15(12):1497–1506, 1989.

- [24] Jeremy Elson and Deborah Estrin. Time synchronization for wireless sensor networks. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 186, Washington, DC, USA, 2001. IEEE Computer Society.
- [25] S. Gandham, M. Dawande, and R. Prakash. Link scheduling in sensor networks: distributed edge coloring revisited. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, pages 2492–2501, 2005.
- [26] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal on Applied Math.*, 32:835–859, 1977.
- [27] M. Gastpar and M. Vetterli. On the capacity of wireless networks: The relay case. In *Proc IEEE Infocom 2002*, pages 1577 –1586. IEEE Computer Society Press, Los Alamitos, CA, June 2002.
- [28] Mario Gerla, Lokesh Bajaj, Mineo Takai, Rajat Ahuja, and Rajive Bagrodia. Glomosim: A scalable network simulation environment. Technical Report 990027, UCLA, Computer Science Department, May 1999.
- [29] Joël Goossens, Shelby Funk, and Sanjoy Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-Time Systems*, 25(2-3):187–205, 2003.
- [30] Matthias Grossglauser and David N. C. Tse. Mobility increases the capacity of ad-hoc wireless networks. In *Proc IEEE Infocom 2001*, pages 1360–1369. IEEE Computer Society Press, Los Alamitos, CA, 2001.
- [31] R. M. Grow. A timed-token protocol for local area networks. In *Proceedings of the Electro '82*, 1982.
- [32] P. Gupta and P. R. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404, March 2000.

- [33] Moncef Hamdaoui and Parameswaran Ramanathan. Selection of timed token protocol parameters to guarantee message deadlines. *IEEE/ACM Trans. Netw.*, 3(3):340–351, 1995.
- [34] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, page 8020, Washington, DC, USA, 2000. IEEE Computer Society.
- [35] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 56–67, New York, NY, USA, 2000. ACM Press.
- [36] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu. Impact of interference on multi-hop wireless network performance. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 66–80. ACM, 2003.
- [37] R. Jain. Performance analysis of FDDI token ring networks: effect of parameters and guidelines for setting TTRT. In *SIGCOMM '90: Proceedings of the ACM symposium on Communications architectures & protocols*, pages 264–274, New York, NY, USA, 1990. ACM Press.
- [38] M. J. Johnson. Proof that timing requirements of the FDDI token ring protocol are satisfied. *IEEE Transactions on Communications*, 35:620–625, June 1987.
- [39] E. G. Coffman Jr., M. R. Garey, D. S. Johnson, and A. S. LaPaugh. Scheduling file transfers in a distributed network. In *PODC '83: Proceedings of the 2nd annual ACM symposium on Principles of Distributed Computing*, pages 254–266. ACM, 1983.

- [40] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 243–254, New York, NY, USA, 2000. ACM Press.
- [41] Sooyeon Kim, Sang H. Son, John A. Stankovic, Shuoqi Li, and Yanghee Choi. SAFE: A data dissemination protocol for periodic updates in sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 228, New York, NY, USA, 2003. IEEE Computer Society.
- [42] Murali Kodialam and Thyaga Nandagopal. Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 42–54. ACM, 2003.
- [43] Anis Koubaa, Mario Alves, and Eduardo Tovar. Modeling and worst-case dimensioning of cluster-tree wireless sensor networks. In *Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS'06)*, pages 412–421, Los Alamitos, CA, USA, 2006. IEEE Computer Society.
- [44] Ulas C. Kozat and Leandros Tassiulas. Throughput capacity of random ad hoc networks with infrastructure support. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 55–65. ACM, 2003.
- [45] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 61–69. ACM, 2001.
- [46] Benyuan Liu, Zhen Liu, and Donald F. Towsley. On the capacity of hybrid wireless networks. In *22nd Annual Joint Conference of the IEEE Computer and*

- Communications Societies (INFOCOM 2003)*, pages 1543 – 1552. IEEE Computer Society Press, Los Alamitos, CA, April 2003.
- [47] J. M. Lopez, J. L. Diaz, and D. F. Garcia. Utilization bounds for EDF scheduling on real-time multiprocessor systems. *Real-Time Systems*, 28(1):39–68, 2004.
- [48] Navneet Malpani, Jennifer L. Welch, and Nitin Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 96–103, New York, NY, USA, 2000. ACM Press.
- [49] Fabian Garcia Nocetti, Ivan Stojmenovic, and Jingyuan Zhang. Addressing and routing in hexagonal networks with applications for tracking mobile users and connection rerouting in cellular networks. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):963–971, 2002.
- [50] Charles E. Perkins and Elizabeth M. Royer. Ad hoc on demand distance vector (aodv) routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New York, NY, USA, 2 1999. IEEE Computer Society.
- [51] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, 2000.
- [52] K. Shashi Prabh and Tarek Abdelzaher. On scheduling and real-time capacity of hexagonal wireless sensor networks. In *ECRTS 2007: Proceedings of the 19th Euromicro Conference on Real-Time Systems*, page To appear.
- [53] K. Shashi Prabh and Tarek F. Abdelzaher. Energy conserving data cache placement in sensor networks. *The ACM Transactions on Sensor Networks*, 1(2):178–203, November 2005.
- [54] Venkatesh Rajendran, Katia Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In

- SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 181–192, New York, NY, USA, 2003. ACM Press.
- [55] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: A geographic hash table for data-centric storage in sensor networks. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, New York, NY, USA, 2002. ACM Press.
- [56] Injong Rhee, Ajit Warrier, Mahesh Aia, and Jeongki Min. Z-MAC: a hybrid mac for wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 90–101, New York, NY, USA, 2005. ACM Press.
- [57] Gabriel Robins and Alexander Zelikovsky. Improved steiner tree approximation in graphs. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 770–779, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.
- [58] Paolo Santi. Topology control in wireless ad-hoc and sensor networks. *ACM Comput. Surv.*, 37(2):164–194, 2005.
- [59] Jens B. Schmitt and Utz Roedig. Sensor network calculus - a framework for worst case analysis. In *DCOSS*, pages 141–154, 2005.
- [60] Kenneth Sevcik and Marjory J. Johnson. Cycle time properties of the FDDI token ring protocol. In *SIGMETRICS '86/PERFORMANCE '86: Proceedings of the 1986 ACM SIGMETRICS joint international conference on Computer performance modelling, measurement and evaluation*, pages 109–110, New York, NY, USA, 1986. ACM Press.
- [61] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensor networks. In *First ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets 2002)*, New York, NY, USA, 2002. ACM Press.

- [62] Kang G. Shin. HARTS: A distributed real-time architecture. *Computer*, 24(5):25–35, 1991.
- [63] J. A. Stankovic, Tarek F. Abdelzaher, Chenyang Lu, Lui Sha, and J. C. Hou. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7):1002–1022, 2003.
- [64] Fred Stann and John Heidemann. RMST: Reliable data transport in sensor networks. In *Proceedings of the First International Workshop on Sensor Net Protocols and Applications*, pages 102–112, New York, NY, USA, 4 2003. IEEE.
- [65] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, 2005.
- [66] Ting Yan. *Analysis Approaches for Predicting Performance of Wireless Sensor Networks*. PhD thesis, University of Virginia, 2006.
- [67] Ting Yan, Tian He, and John A. Stankovic. Differentiated surveillance for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 51–62, New York, NY, USA, 2003. ACM Press.
- [68] Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 148–159, New York, NY, USA, 2002. ACM Press.
- [69] Sijing Zhang, Alan Burns, and Tee-Hiang Cheng. Cycle-time properties of the timed token medium access control protocol. *IEEE Trans. Comput.*, 51(11):1362–1367, 2002.
- [70] Qin Zheng and Kang G. Shin. Synchronous bandwidth allocation in FDDI networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(12):1332–1338, 1995.