RESOURCE MANAGEMENT FOR DISTRIBUTED REAL-TIME SYSTEMS

A Dissertation Presented

by

HUAN LI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2006

Department of Computer Science

RESOURCE MANAGEMENT FOR DISTRIBUTED REAL-TIME SYSTEMS

A Dissertation Presented

by

HUAN LI

Approved as to style and content by:

Prashant Shenoy, Co-chair

Krithi Ramamritham, Co-chair

James Kurose, Member

C. Mani Krishna, Member

Deepak K. Ganesan, Member

W. Bruce Croft, Department Chair
Department of Computer Science

To my parents — Mojie Li and Huifang Wu
To my husband and my lovely daughter — Shulin and Shufan.

ACKNOWLEDGMENTS

ABSTRACT

RESOURCE MANAGEMENT FOR DISTRIBUTED REAL-TIME SYSTEMS

SEPTEMBER 2006

HUAN LI

B.Sc., HUAZHONG NORMAL UNIVERSITY, WUHAN, CHINA

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Prashant Shenoy and Professor Krithi Ramamritham

Recent advances in embedded sensor systems and wireless technologies have made it possible to conceive a new type of real-time application — distributed real-time sensor systems. Examples of such systems include surveillance applications, distributed industrial control systems, disaster response systems and robotic applications, just to name a few. In addition to the requirements of providing temporal guarantees, these distributed embedded systems are normally associated with extreme resource constraints, both in computation and communication.

In this dissertation, we investigate some of the fundamental resource management problems that arise in the design and development of distributed real-time sensor systems. We consider a scenario in which a team of robots collaborating with each other to rescue people from a building on fire. The coordinated behavior is achieved

by assigning a set of control tasks, or strategies, to robots in a team. At each step, the application may have many functionally equivalent strategies, though some of them may not be feasible, given limited resource and time availability. In order to accomplish dynamic feasibility checking and improve system performance, we propose efficient resource allocation mechanisms. Our approaches not only minimize the communication cost, but also minimize and balance the workload of each processor, resulting in good performance with regards to system schedulability and feasibility.

With respect to real-time communication in sensor applications, each message will traverse multiple hops from the source to the destination with an end-to-end deadline. In order to provide timeliness guarantees, a key challenge is to bound the delay and prioritize per-hop transmission. However, existing wireless protocols such as 802.11 families may suffer from unpredictable delays, due to collision, back-off and false blocking problems. In this study, after showing that the problem of scheduling all wireless transmissions to meet deadlines is NP-hard, we derive effective deadline for each per-hop transmission and schedule the most urgent message. We develop novel algorithms to parallelize message transmissions. By carefully exploiting spatial reuse property and avoiding collisions, the deadline misses are minimized. Extensive simulations demonstrate the effectiveness of our approaches, especially under high channel contention environments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The rapid development of sensor devices, embedded systems and advanced wireless technologies have made it possible to conceive a new type of real-time applications — distributed real-time sensor systems. Examples of such systems include surveillance applications, distributed industrial control, earthquake response systems, robotic applications and so on. In addition to timing characteristics that require the system to complete its work and deliver its services on a timely basis, these distributed embedded systems are normally imposed with extremely resource constraints, both in computation and communication. In this chapter, we start by describing the problem motivation and the thesis goals. Then, the contributions and structure of the dissertation are summarized.

1.1 Motivation

The design and development of wireless sensor applications have received a lot of research attention recent years [23, 22, 6, 7, 48, 19, 107]. The advances in wireless communications and electronics have enable the inexpensive, low-power sensor nodes to be deployed throughout a physical space. These small sensor nodes, which consist of sensing, data processing and wireless capabilities, make it possible to provide dense sensing very close to phenomenon and process/communicate those information in react to the quick changing environment.

Potential sensor applications include environmental monitoring in remote areas, monitoring of ocean temperatures, searching and rescuing operations, disaster and

emergency response. A sensor application is typically composed of a collection of sensor nodes that continuously monitor the surrounding environment and a collection of sinks that aggregate, process, and react to the sensory data. Communication between the sensors and sinks requires a network; since the inherent nature of many sensor applications precludes the use of wired networks, wireless networks are commonly used in such applications.

Many wireless sensor network (WSN) applications require real-time communication in response to dynamic physical world. For example, a surveillance system needs to alert authorities of an intruder within a few seconds of detection [37]. Similarly, a fire-fighter may rely on timely temperature updates to remain aware of current fire conditions [58]. Another example of such applications is a team of robots searching for trapped people in a building on fire. Each robot is equipped with a set of sensors such as temperature and pressure monitors, video cameras, GPS, and infra-red monitors. Not all robots may have all of these sensors due to power and weight constraints or the consideration of design issues. For example, some robots may specialize in thermal imaging sensors for locating humans, while others may carry extra processing elements and fewer sensors. The robots pool the sensory data from all sensors and exchange information with one another to adjust their sensing strategies in a timely fashion in direct response to the evolving environment and determine where to move for the next step, both individually and as a group. Since the path for each robot needs to be determined to ensure timely mobility, the *transmission* and *processing* of sensory data imposes *timeliness constraints*. The distributed systems which provide timeliness guarantees for processing and transmitting sensor data are referred as *distributed real-time sensor systems*.

Compared with traditional distributed systems, the real-time guarantees for sensor-based systems is more challenging due to the resource constraints in power, processing capacity and wireless communication bandwidth. The distributed real-time sensor

systems exhibit performance challenges with regards to system schedulability, feasibility, and timeliness guarantees for wireless transmission. The key features for meeting these guarantees are as the follows.

- The system resources such as computation and communication should be utilized efficiently and cost-effectively to meet schedulability and feasibility requirements.
- The system should be able to optimize the communication, and the wireless capacity should be taken into use to the much extent.
- Different sensor data should be transmitted to the destinations on time. This means that the probability of transmission conflicts should be minimized and the system is able to bound or predict the delay.

In the example of robotic teams, the distributed control systems are built using sets of functionally equivalent controllers in the form of coordinated, adaptive control schemes. These controllers are distinguished by their use of resources including communication, processor and sensors. Although the control strategies are logically equivalent, some of them may not be feasible, with regards to the timeliness guarantees. The goal of finding a feasible strategy is, to assign tasks to robots/processors so that all tasks finish their work on time and the communication costs among tasks in different processors is minimum. One challenge of task assignment problem is that given limited resource and time availability, finding an optimal feasible resource allocation solution is known to be NP-hard. The other complicating aspect of this application domain is that the robot team is often moving and its size is not fixed. As robots enter or leave the team, the application must recompute the set of available strategies. Each time the team changes, the application must run an *on-line* algorithm to determine the task allocation and scheduling of a new feasible strategy. In the literature on real-time systems and on distributed systems, we can find nu-

merous approaches, e.g., simulated annealing [101], branch-and-bound [70], myopic heuristic [73], network flow, bin packing to different variants of this task assignment problem. However, most of these approaches are for off-line task assignment [56], and therefore, cannot be adopted into the robotic scenario.

In the applications of distributed sensory systems where message transmissions are performed over wireless networks, novel protocols and algorithms are needed to effectively tackle the unique resource constraints and application requirements to provide end-to-end temporal guarantees. Traditional CSMA-based schemes are deemed inappropriate since they all make the fundamental assumption of stochastically traffic distribution [6], which is not true for the sensor networks where traffic is variable but is highly *correlated* and dominantly *periodic*.

Many MAC protocols have been proposed for wireless sensor networks with the objective of the creation of self-organizing network, energy efficiency and fairly sharing communication resources between sensor nodes [90, 89, 105, 86, 108, 44, 104, 100]. The protocols for providing soft real-time guarantees are studied in SPEED [38, 39] and RAP [59], while little research has been done for temporal guarantees through collision avoidance and parallel transmission. Using traditional wireless protocols such as CSMA/CA-based 802.11 family of protocols, the time-sensitive sensor applications will have to meet the following challenges. First, CSMA/CA networks do not completely eliminate the possibility of collisions despite of the use of the collision avoidance techniques. Second, senders will back-off exponentially when they sense ongoing transmissions on the channel, which may cause unpredictable delays. Third, vanilla 802.11 networks suffer from the *false blocking*, as observed in [14]. And in the worst case, the blocking may propagate throughout the whole network [76].

The limited computation and communication resources in wireless sensor networks raise many challenges in the design of distributed real-time sensor systems. The motivation of this thesis is to design, analyze and evaluate efficient time-cognizant

resource management mechanisms, and provide insights to improve the performance objectives of distributed real-time sensor systems. In the next section, we examine some of the fundamental problems that need to be solved in order to build a feasible distributed real-time sensor systems. The following section summarize the primary contributions made by this dissertation. We conclude this chapter with the description of the structure of the dissertation.

1.2 Scope of Research

In this section, the scope of this dissertation are presented. The specific resource constraints, wireless channel properties and end-to-end temporal constraints in sensor data transmission are discussed.

1.2.1 Tasks, Resources And Temporal Constraints

In this dissertation, we call each unit of work that is scheduled and executed by the system a *task*. These tasks can provide some system function, e.g., computation of path plan for a robot team, or the transmission of a sensor data packet. We consider a system that consists of N sites, each having an identical *CPU*. The underlying network used to support interprocessor communication is wireless network. We model the wireless channel as a specific processor for the purpose of resource allocation and scheduling. Resources other than CPUs and wireless channel, which are called *processors*, are not considered in this dissertation. The cost of communication between a pair of tasks is usually significantly lower when they are on the same site than when they are on different sites, and is negligible in the following chapters. In this situation, we want to take into account the cost of communication between two tasks that are assigned on different sites, which depends on the volume of data exchanged and the bandwidth of the communication link. The goal of task assignment is to assign each

task to an appropriate site to feasibly schedule all tasks on it, and at the same time, the total communication cost among tasks on different sites is minimized.

The two characteristics that distinguish tasks in real-time systems from those in nonreal-time systems are the *deadlines* and *release times* [56]. The *release time* of a task is the instant of the time at which the task becomes available for execution. The *deadline* of a task is the instant of time by which its execution is required to be completed. In the dissertation, we assume the sensor data produced by sensor tasks are normally consumed by control tasks multi-hops away. And in this case, the deadline is also referred to as *end-to-end* deadline. In general, we call a constraint imposed on the timing behavior of a task a *temporal constraint* that can be specified in terms of release times and deadlines.

While considerable research has been devoted to the understanding of the performance of traditional distributed and multiprocessor real-time systems, very little work specifically geared toward resource management of distributed sensor systems has been done. The goal of the first part of this dissertation is to (1) characterize and understand the resource constraints of wireless sensing systems, and (2) explore novel *on-line* resource allocation and scheduling mechanisms to improve the system feasibility and efficiency.

1.2.2 Optimization of Wireless Communication

As discussed in Section 1.1, in most sensor applications, sensor data are transmitted over ad hoc wireless network and have timing constraints for the system in response to the fast changing physical world. Although one channel with one fixed frequency is considered in the dissertation, from spatial perspective, there may exist many *parallel processors (channels)* that can be used for data transmission, since some of the messages are geographically apart enough and there is no interference between those transmissions. The parallel data transmissions can be modeled as the trans-

mission tasks running on different processors simultaneously. However, the number of processors are not fixed or known in advance. This is because, given a set of data messages on different sites, any pair of parallel transmissions may affect the subset of all other transmissions, which leads to the solution space increase exponentially.

The goal of the second part of this dissertation is to (1) gain fundamental understanding of the parallelization problem in the wireless sensor data transmission with temporal constraints, (2) obtain insights on the problem complexity and its variations in practical use, and (3) develop and evaluate the performance of different parallelization-based algorithms for achieving minimum message transmission time.

1.2.3 Multi-hop Sensor Data Transmission

In distributed sensor systems, sensor updates must flow from the sensor tasks to the processing tasks, and from processing tasks to the actuators and outplay displays with bounded delay. The goals of communication protocols in real-time systems are different from those in traditional nonreal-time data communication systems [47]. The key performance in traditional system is system throughput, that is, how much data can be transferred over the network in one unit time from source to destination. In real-time system, the key measure is the probability of delivering a message by a certain deadline.

Protocols suitable for real-time transmission in wired networks have been studied in the last couple of decades [109, 110, 17, 30, 80, 106]. Recently, there has been increasing interest in the study of wireless communication and real-time protocols [45, 59, 38, 39, 16]. However, most of these protocols support some sort of *soft* real-time communications. For example, SPEED [38, 39] provides a delay guarantee per unit delivery distance(speed guarantee), and a predictable end-to-end communication delay under given spatial (distance) constraint can be obtained.

The goal of the third part of this dissertation is to (1) explore the wireless channel spatial properties and its impact on the real-time communication, and (2) design novel spatial reuse-based algorithms and evaluate the performance in terms of end-to-end temporal guarantees for varying network and data flow topologies.

1.3 Thesis Contributions

This dissertation presents a study of the resource management for distributed real-time sensor systems, to achieve the three goals outlined in Section 1.2. The first discusses the task allocation and scheduling in a resource-constrained distributed system (robot team). Utilization bound and precedence semantics are explored to improve the system scehdulability. The second part addresses the characteristics of wireless transmission in face of temporal constraints. A novel approach is proposed and analyzed for the purpose of minimizing the total communication time. The third part presents solutions to multi-hop message scheduling over a shared wireless channel. In particular, a novel scheduling algorithm that takes use of the spatial reuse property and temporal per-hop status are proposed and studied.

In this section, we elaborate on the contributions of the thesis. The following are problems and contributions presented in this dissertation.

- *Efficient Resource Allocation Techniques* In the applications of multi-robot systems where coordinated control schemes are built to achieve collaboration behaviors, the control strategies are distinguished by their use of communication, sensors and processors. Finding an optimal resource allocation with precedence constraints in a distributed environment is in general NP-hard [70], and even some of the simplest scheduling problems are NP-hard in the strong sense [26]. In this thesis, we focus on the problem of allocating control tasks in such distributed collaborative systems. We investigate the problem where communication is an important factor for each member to share information and achieve

collaboration behaviors. In order to improve schedulability, the system should be able to minimize communication overhead by allocating communicating tasks on the same platform. However, the more tasks on the same processor, there will have higher workload, which may decrease system performance and even lead to system infeasibility.

To solve the conflict aspects of minimization of communication and processor utilization, in this thesis, we propose efficient resource allocation algorithms that take into account the communication cost and utilization at the same time for each step. To minimize the communication cost, we define a novel concept, *communication cost ratio*, for counting the real communication weight as to the total execution times of a pair of communicating tasks. To bound and balance the workloads of processors, we further use a dynamic utilization *threshold* for the selection of processor at each step of task assignment. Our experimental studies show that our task allocation algorithm together with the precedence-encoded scheduling policy greatly improves performance of distribute real-time system in terms of schedulability and feasibility [52].

- *Optimization of Message Transmission* Allocation tasks to processors is the first step in enabling distributed real-time systems to meet the schedulability requirements. With these mechanisms in place, the next step is to design techniques for the optimization of message transmission. These techniques should take into account the wireless constraints, such as range constraint and interference constraint, while scheduling message transmissions. Also they should be able to avoid collisions in order to minimize the transmission delay.

  In this thesis, we study the problem of providing qualitatively-better QoS to transmit sensor data, using commodity 802.11 wireless networks. The first problem we investigate is the Optimal Parallel Communication Scheduling (OParCS)

problem, whose goal is to minimize the completion time of transmitting a set of sensor messages. We show this problem is NP-complete. Consequently, we propose several spatial channel reuse algorithms. These algorithms perform non-interfering transmissions in parallel, and explicitly avoid the transmission collisions. To enable comparisons, we also propose an A^*-based optimal solution. The evaluation results demonstrate that the spatial reuse algorithm results in good performance: the time to complete transmissions is within a factor of 8.5 of the optimal solution, and it outperforms the random method. More importantly, it is robust to the increases in communication density [51].

- *Scheduling Messages with Temporal Constraints in Multi-hop Sensor Networks* In the sensor applications where each message is associated with a deadline and need to traverse multiple hops from the source to the destination, the system should be able to provide the timeliness guarantees. However, if we use the best-effort techniques to transmit the messages, the deadline misses will be very high. An early work that studied the real-time communication in sensor networks is [59]. In their proposal of velocity monotonic scheduling mechanism, at each hop, packets are scheduled based on the highest velocity requirement, where the velocity is calculated by the deadline and the distance to travel of each packet. Although this approach takes the deadline into account, it still meet the problem of unbounded delay, due to collisions, back-offs or false blocking problems.

  In this thesis, we examine the problem of providing end-to-end temporal guarantees for multi-hop wireless transmissions. We argue that this problem is NP-hard, and consequently we sort to heuristics. To efficiently solve the problem with the goal of minimizing the deadline misses, we first derive effective message deadlines for each hop transmissions and use this information to choose the most urgent message for scheduling to meet the end-to-end deadline. Second,

we propose novel efficient algorithms which attempt to transmit messages in parallel to the much extent. We discuss the criteria upon which the messages can be sent at the same time and move the the next hop to the destination, to maintain the end-to-end deadline requirements. We compare the performance of our proposed algorithm CR-SLF with the PH-SLF, a simple CSMA-based algorithm. We evaluate the impact of sensor period, deadline and message size based on specific transmission topology. We also investigate how the different interference ranges may affect the properties of the algorithms. Our experiments show that CR-SLF performs much better than PH-SLF with the performance of deadline miss ratio, especially when utilization is high and/or the probability of collisions is high [49].

1.4 Structure of the Dissertation

In this introduction, we have provided the motivation and brief description for three problems arising in the design and development of distributed real-time systems. The rest of this dissertation is structures as follows. First, we discuss related work in Chapter 2. In Chapter 3, we describe the resource allocation techniques. In Chapter 4, we consider the problem of minimizing total transmission time and present our channel-reuse schemes. The problem of providing temporal guarantees for multi-hop wireless transmission is discussed in Chapter 5. We conclude with a research plan and a brief summary of our research contributions in Chapter 6.

CHAPTER 2

RELATED WORK

Distributed real-time sensor systems draw several related works, such as resource allocation, schedulability analysis and improvement, message transmission with temporal guarantees, etc.. In this chapter, we will highlight some of the most recent work in these areas.

2.1 Real-Time Task Assignment and Scheduling

The purpose of real-time computing is to execute, by the appropriate deadlines, its critical control tasks [47]. The allocation/scheduling problem can be stated as follows. Given a set of tasks, task precedence constraints, resource requirements, task characteristics, and deadlines, we are asked to derive feasible allocation/schedule on a given computer. There is an enormous literature on allocation/scheduling problems since 1970's. In the following, we present some techniques that relates to the meeting of deadlines in multiprocessor/distributed systems.

2.1.1 Task Assignment in Multiprocessor Environment

Except for trivial cases, the problem of finding an optimal assignment of tasks to processors is NP-hard in the strong sense [27]. Therefore, most methods use heuristics. Let us first consider a simplest form where a set of periodic independent preemptable tasks (task deadlines equal their periods) to be assigned to a multiprocessor system consisting of identical processors, and resources other than processors are not required by tasks. It is known that as long as the total utilization of all tasks assigned to the

same processor is no greater than 1, this task set is EDF-schedulable. It is easy to formulate this task assignment as the simple bin-packing problem [18], and the number of bins required to pack all items is the number of processors required to feasible schedule all tasks. The bin-packing is NP-complete, but there are many good heuristic algorithms to solve it. In the first-fit algorithm, tasks are assigned one by one in an arbitrary order, and the i_{th} task is assigned to the processor P_k if i) the total utilization of T_i and existing tasks already assigned to P_k is no greater than 1, but ii) assigning T_i to any other processors $P_1, ..., P_{k-1}$ (ordered by the increasing values of utilizations) would make the total utilization larger than 1. By measuring the ratio of the number of processors required by a heuristic algorithm to the number obtained by the optimal assignment, in the worst case, first-fit algorithm never achieves the ratio of more than 1.7 [18]. Simulation also show that when the utilizations of tasks are uniformly distributed in [0,1], the total utilization per processor achieved by first-bit is 0.93 on average. Above discussion is based on the assumption there is no limit on the number of processors. If the tasks are sorted in non-increasing order of their utilizations and are assigned in turn by that order, the ratio of the number of processors required by the first-fit decreasing algorithm to the optimal solution is only 1.22, in the limit as the number of processors approaches infinity. In the case where we are given m fixed number of processors, Oh and Baker [67] showed that on the fixed-priority basis, the first-fit (RMFF) can always find a feasible assignment if the total utilization of tasks is no greater than $m(2^{1/2} - 1) = 0.414m$. Lopez et al [57] later showed that RMFF can schedule any system of periodic tasks with utilization up to $(m + 1)(2^{1/(m+1)} - 1)$, a upper bound that appears to be tight.

Utilization-balancing algorithm attempts to balance processor utilization, and proceeds by allocating the tasks one by one to the least utilized processor at each step [47]. This algorithm also assigns r_i copies of task T_i to separate processors for fault-tolerance. It is shown that if $r_1 = ... = r_n = r$, and there are p processors in all and

$P \gg r$, the approximation result is 1.125 times of the optimal solution with regards to the sum of the squares of the processor utilizations.

2.1.2 Task Assignment in Distributed Real-time Systems

For task assignment in a distributed system, all resources including processors and communication media must be considered and then a complete end-to-end model of transactions passing through the many resources must be constructed [83]. Tindell *et al.* [101], Peng *et al.* [70], Abdelzaher *et al.* [1] and Ramamritham [74] studied the task allocation and scheduling problem. In their models, tasks can have precedence or communication constraints.

Tindell *et al.* describes an approach to solving the task allocation problem using a technique known as *simulated annealing*. In their work, simulated annealing is proved to be an effective approach to task allocation. However, it may not be directly used as an on-line algorithm, considering the speed of the algorithm. Besides, well-balanced allocations is shown in their paper to result in infeasible solutions, since token protocol is used as the message transmission model, and a high bus utilization gives a high token rotation time, resulting in less schedulable solutions. By using a branch-and-bound search algorithm [70], the optimal solution in the sense of minimizing maximum normalized task response time is found to the problem of allocating communicating periodic tasks to heterogeneous processing nodes. Though the heuristic guides the algorithm efficiently toward an optimal solution, the algorithm cannot be simply applied and extended to the applications that need to i) make on-line feasible guarantees, and ii) consider a non-preemptive schedule, which is NP-hard in the strong sense even without precedence constraints [27].In [1], a period-based method is proposed to the problem of load partitioning and assignment for large distributed real-time applications. Scalability is achieved by utilizing a recursive divide-and-conquer technique.[74] discussed a static algorithm for allocating and scheduling components

of periodic tasks across sites in distributed systems. How to allocating replicates is a major issue counted in the algorithm.

Despite the advances described above, feasibility analysis for parallel computation system is still not as well understood as for the single processor systems [83]. In the design of feasible distributed real-time sensor systems, a good task allocation should enable the system to provide on-line schedulability guarantees with regards to all kinds of physical and temporal constraints.

2.2 Real-Time Schedulability Analysis

Numerous research results have demonstrated the complexity of the design for real-time system [29, 75, 77, 78, 79], especially, with respect to the analysis and improvement of system schedulability [46, 68, 69, 103]. For tasks with temporal constraints, researchers have focused on generating task attributes, e.g., period, deadline and phase. For example, Gerber *et al.* [29, 79] proposed the period calibration technique to derive periods and related deadlines and release times from given end-to-end constraints. Techniques for deriving system-level constraints from performance requirements are proposed by Seto *et al.* [81, 82]. When end-to-end constraints are transformed into intermediate task constraints, most previous research results are based on the assumption that task allocation has been done *a priori*. However, *schedulability* is clearly affected by both the temporal characteristics and the allocation of tasks. A more comprehensive approach that takes into account the task temporal characteristics and allocations, in conjunction with schedulability analysis, is required.

For a set of *independent* periodic tasks, Liu and Layland [55] first developed the feasible workload condition for schedulability analysis under uniprocessor environments. Much later, Baruah *et al.* [13] presented necessary and sufficient conditions, namely, $U \leq n$ (n is the number of processors) based on *P-fairness* scheduling for

multiprocessors. Also, the upper bounds of workload specified for the given schedules, e.g., EDF and RMA, are derived for homogeneous or heterogeneous multiprocessor environments [8, 11, 24, 33, 32, 93, 10]. For example, Andersson et al. [8] shows that the utilization guarantee for any fixed-priority multiprocessor scheduling algorithm is equal or lower than $(m+1)/2$ for a $m-$processor system. Philips et al [72] studied the competitive performance of on-line multiprocessor scheduling algorithms. They showed that if a task set is feasible on m identical processors, then the same set is schedulable by preemptive RM scheduling algorithm on m processors that are faster by a factor of $(2-1/m)$. A number of studies have followed up on the schedulability analysis for both identical and "uniform" processors [12, 24, 31, 10].

All these techniques are for preemptive tasks and task or job migrations are assumed to be permitted without any penalty. If *precedence* and *communication* constraints exist, these results cannot be directly used.

2.3 Communication in Wireless and Sensor Networks

The goal of communication protocols in real-time systems is to provide real-time support for message delivery over wired or wireless networks. Numerously protocols have been proposed for providing real-time capabilities on wired networks [] in the past twenty years, though increasing work has been done in the design of real-time wireless communication recently. In this section, we present some of the major protocols for real-time support in various wireless communication scenarios.

2.3.1 Real-Time Communication for Wireless or Sensor Networks

Sensor networks can be used for many future applications, such as environmental monitoring, chemical or radiological detection and protection systems, and public surveillance systems etc.. Typically, each node in the sensor networks consists of sensors, wireless interface, and limited power and computational capabilities. Time-

dependent data being being routed and aggregated throughout the whole network must arrive at the destinations (monitors/actuators) on time to ensure the timeliness action to the real-world. Real-time issues that arise in various layers of the network stack in sensor networks are studied in [95, 94]. In terms of MAC layer, the authors pointed out that a key research challenge is to provide predictable delay and/or prioritization guarantees, while minimizing overhead packets and energy consumption. Other efforts that address real-time issues in sensor networks include the design of the communication stack and real-time routing protocols.

RAP [59] is a real-time communication architecture for large-scale wireless sensor networks that includes a novel packet scheduling policy called velocity monotonic scheduling. At each hop, packets are scheduled based on the highest velocity requirement, and the velocity is calculated by the deadline and the distance to travel of each packet. In this method, the requested velocity is mapped to a MAC-layer priority, which in turn reduces the deadline miss ratio.

SPEED [38, 39] is an adaptive location-based real-time routing protocol that uses feedback-based techniques to satisfy per-hop deadlines in face of unpredictable traffic. It supports soft real-time communication service with a desired delivery speed across the sensor network, so that the end-to-end delay is proportional to the distance between the source and destination. A neighborhood feedback loop on each node periodically computes the probability of forwarding a packet to a neighbor based on its measured speed in the last sampling period, and ensure that only those neighbors whose speed is higher than S — speed bound are eligible for receiving packets, and the neighbors with lower speeds get lower receiving probabilities. Since each node maintains an average delay for packet transmission, given a distance to travel in hops, whether a packet will meet its deadline can be determined. Simulation given in [38, 39] shown that SPEED achieve significant improvement in terms of miss ratio than geographic routing, DSR [43] and AVDV [71].

SWAN [4, 3] uses feedback information from the MAC layer to regulate the transmission rate of nonreal-time TCP traffic in order to sustain real-time UDP traffic. In particular, SWAN uses local rate control for best-effort traffic, and sender-based admission control for real-time UDP traffic. Explicit congestion notification is used to dynamically regulate admitted real-time traffic in the face of network dynamics such as mobility and temporary traffic overload. Simulation, analysis, and results from an experimental wireless testbed show that real-time applications experience low and stable delays under various multihop, traffic, and mobility conditions.

Kanodia et al. [45] proposed a service differentiation for delay-sensitive traffic by prioritizing 802.11. Woo and Culler [105] proposed an adaptive MAC layer rate control to achieve fairness among nodes with different distances to the base station. All of these algorithms work well by locally degrading a certain portion of the traffic, though they cannot handle long-term congestion.

Collision free real-time MAC protocol is proposed in [16], where a hexagonal cellular network architecture is used as the infrastructure for wireless transmission. Implicit earliest deadline first (EDF) scheduling is used inside each cell for intra-cell packet transmission; frequency division multiplexing (FDM) is used among adjacent cells to allow parallel transmissions in different cells. Each router is equipped with two transceivers so that they can transmit and receive in the same direction at the same time for inter-cell messages. The schedulability condition is given for a hybrid message set consisting of hard periodic and soft aperiodic messages.

Recently, there has been work on the relationship between the delay and capacity or throughput in wireless networks [28, 85, 25, 64, 65]. All of these efforts focus on computing asymptotic performance bounds. By modeling neighboring interference as conflict graph, lower and upper bounds on the maximum throughput for a given network and workload are computed in [41].

2.3.2 Scheduling-Based Transmission Protocols

In scheduling-based MAC protocols, the time at which a node can transmit is determined by a scheduling algorithm, so that multiple nodes can transmit simultaneously without interference on the wireless channel [95].

A large amount work has been focused on time division multiple access (TDMA) scheduling [66, 96, 54]. Most of them focus on the maximization of system throughput and fairness bandwidth occupation. Recent years, the distributed computation and assignment of schedules for a spatial reuse TDMA (STDMA) protocol for ad hoc networks is a topic of active research [36, 34, 88, 91]. The idea is to increase capacity by letting several radio terminals use the same time slot when possible. A time slot can be shared when the radio units are geographically separated such that small interference is obtained. Simulation results in [35] show that the average delay can be decreased considerably by taking the traffic load into consideration in a STDMA scheme and by modeling the total interference in the network. When directional antenna patterns are used, RA-MHA (the method uses minimum hop paths between sources and destinations as well as STDMA scheduling strategy) produces a substantial improvement in throughput and packet delay [88].

In the channel reuse scenario, the service area is divided into a number of regions called cells. In each cell is a base station that handles all the calls made within the cell. The total available bandwidth is divided permanently into a number of channels. Channels must then be allocated to cells and to calls made within cells without violating the channel reuse constraint. The minimum distance at which there is no interference is called the *channel reuse constraint*. Ramanathan [92] introduced a unified algorithm for efficient (T/F/C)DMA channel assignment to network nodes or to inter-nodal links in a (multihop) wireless networks. In [42], the authors established a relationship between the mutual exclusion problem and the distributed dynamic channel allocation problem.

CHAPTER 3

DYNAMIC RESOURCE ALLOCATION FOR ROBOTIC TEAMS

3.1 Introduction

Most of the current research in sensor networks puts their focus on fixed wireless networks where the nodes are deployed in the area with fixed locations, on the other hand, mobile sensor networks attract much less attentions. Consider a networked robot system, in which each small robot is implemented with a wireless network interface for communication, accurate odometry and compass for navigation, infrared and bump sensors for object detection [87]. Robots are moving around in order to achieve a common objective, e.g., search a goal and maximize (or minimize) some system characteristics (such as network connectivity). For example, a network that autonomously moves nodes to locations of low signal strength to improve throughput along a multi-hop transmission path [15]. One of the big difference between the fixed sensor network and the mobile sensor network is that, in mobile sensor system, the network topology is changing dynamically.

A particularly significant development in robotics within the past decade has been the move away from disembodied, traditional AI to real-time embedded decision making in physical environments [21]. For nearly two decades, the dominant paradigm in mobile robotics research involved the offline design of control algorithms based on deliberation. These planner-based algorithms relied on logic and models of the robots and their environments, but the systems were unresponsive and slow to adapt to dynamic environments. An alternative modern approach is to hybridize control

[9], where a planner and a receive system communicate through a third software layer designed explicitly for that purpose.

Different coordinated behaviors can be also achieved by assigning sets of control tasks, or strategies, to robots in a team [98]. These control tasks must be scheduled either locally on the robot or distributed across the team. An application may have many control strategies to dynamically choose from, although some may not be feasible, given limited resource and time availability. Thus, dynamic feasibility checking becomes important as the coordination between robots and the tasks that need to be performed evolves with time.

However, given limited resource and time availability, finding an optimal feasible resource allocation solution is known to be NP-hard. In this chapter, we address the problem of allocating control tasks to distributed processing entities (robots). We first provide the background and challenges arising in building such distributed collaboration system. With the purpose of improving system feasibility, we then present our *on-line* resource allocation algorithms that not only minimize the inter-robot communication overhead, but also minimize and balance the workload of each processor. Extensive experimental results have shown the effectiveness of communication-based least utilization algorithms, even under extreme resource-constrained environments. Since schedulability is an overlooked part of large multi-robot system design, we also demonstrated the application of our approach to real-world large robotic teams. By analyzing the structure of the processes and their tasks' timeliness constraints, we were able to calculate the upper bound on the size of feasible teams.

The remainder of this chapter is organized as follows. We begin this chapter by introducing the application background and the problems and challenges in Section 3.2. We next present the system and task model in Section 3.3. Then, in Section 3.4, we describe our task allocation algorithms and the scheduling methods in detail. A performance evaluation of the scheduling strategy and the allocation algorithms is

presented in Section 3.6. Finally, we conclude the chapter with a summary of our contributions in Section 3.8.

3.2 Application Background

There are many challenges in the development of distributed real-time sensor system. Typically, those systems need to provide temporal guarantees over highly resource-constrained environments. For the example of robotic searching applications, the computation and power resources are so limited in each robot, thus, the team members have to collaborate with one other in the timely fashion to respond to the changing environment.

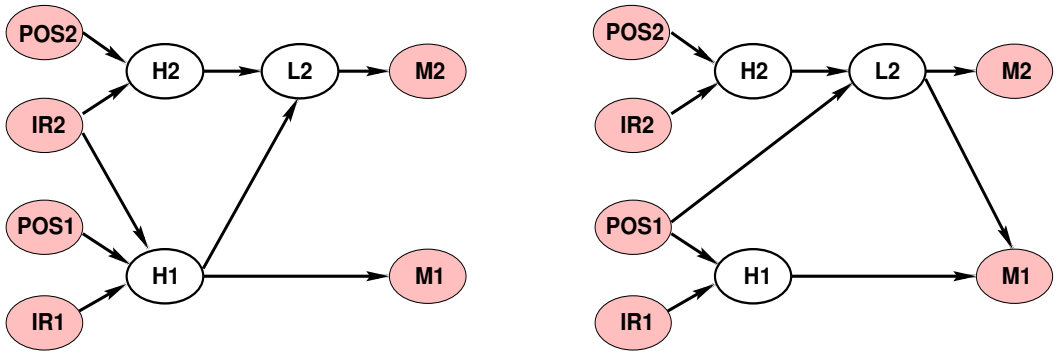
Collaborating with one another to accomplish a common goal, for example, searching a burning building for trapped people, is a promising application for a team of robots. Human operators may direct the search by teleoperation, but wireless communications in these situations can be unreliable. When a search robot ventures outside a reliable communication range, a second robot can autonomously create a network to preserve quality of service between the operator and the search robot. One instantiation of such technology constructs a series, kinematic chain of mobile robots where each of them actively preserves Line-Of-Sight (LOS) [98] and intra-network bandwidth. In the simplest case, two pairwise coordinated controllers, *push* and *pull* are developed for a team of two robots.

The *push* and *pull* controllers differ in the way in which the LOS region is computed and communicated between the pair. The application constructs a strategy by assigning *push* or *pull* controllers to the entire team. The task models for these two strategies, namely, the *push* and *pull* controllers themselves, are depicted in Figure 3.1. Here, each robot must run IR obstacle detection and odometric sensor processing tasks, denoted by IR_i and POS_i, respectively. In addition, both robots must run a command processing task, M_i, which takes desired heading and speed commands

and turns them into motor commands. All these tasks are preassigned to specific execution sites (robots). In the *push* configuration, the follower computes the LOS region in task H_1 (standing for the abbreviation of the path planner), and passes it to task L_2, which computes a new movement vector of the leader that maximizes the search area while keeping the leader within the specified LOS region. In the *pull* configuration, the leader robot does the search, and, concurrently, computes the LOS region in task L_2. The qualitative difference between the two configurations is that *pull* allows a leader robot to search for an area while "pulling" a following robot behind it; *push* allows a follower to specify the search area of the leader, in effect, "pushing" the leader along.

In many cases, different combinations of *push* and *pull* controllers, named *strategy*, will have the same coordinated search behavior. A discussion of how applications generate possible strategies is beyond the scope of this study. However, if there are n robots in the team, the potential strategies are 2^n, using *push/pull* as building blocks. Since the control tasks in a strategy, such as H_1, H_2 and L_2 in the *push/pull* model can be distributed among sites in a team, different task allocation may lead to different feasibility. Therefore, a strategy that is valid at the application level may not always be *feasible* at the system level. How to find feasible, schedulable strategies from a set of functionally equivalent strategies, given by the application, is one goal of this work.

One complicating aspect of this application domain is that the team is often moving and its size is not fixed. As robots enter or leave the team, the application must recompute the set of available strategies. Figure 3.2 shows a sequence from a simulation with five robots using *push* and *pull* controllers, where robot 0 is the leader searching for the goal — the square in the lower left of the map. Each time the team changes, the application must run an on-line algorithm to determine the task allocation and scheduling of a new feasible strategy.



(a) *push* strategy (b) *pull* strategy

Figure 3.1. Tasks in a **Leader/Follower** team.

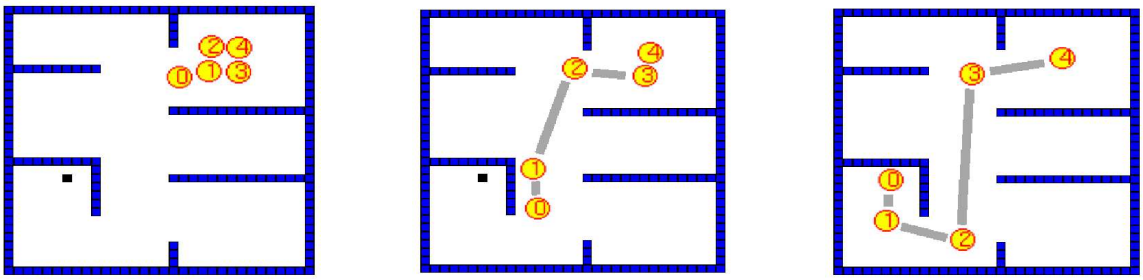


Figure 3.2. A sequence of active robots in a robot team.

If two communicating tasks have to be allocated on different robots, communication over the shared communication medium happens. To avoid the run time contention, the communication needs to be scheduled as well. Assigning tasks with precedence relationships in a distributed environment is in general an NP-hard problem [70], and even some of the simplest scheduling problems are NP-hard in the strong sense [26]. Given temporal and resource constraints, we propose two heuristic allocation algorithms. The purposes of those algorithms are: 1) minimizing communication overhead, 2) minimizing and balancing the processor workload, so that the overall schedulability is improved and optimal coordinated behavior is achieved.

3.3 System Model And Our Goal

In this section, the system and real-time task models are described. Also, the goal of this study to efficiently find a feasible strategy and the accomplishment of this goal are presented.

3.3.1 System and Task Model

A coordinated team consists of a set of sites (robots), each having an identical processor. In this chapter, we use *site* and *processor* interchangeably. Robots in a team share a communication medium that allows broadcast communication between robots. A *strategy*, which is specified at the application level, is denoted at the system level by an acyclic *Task Graph (TG)*, e.g., tasks graphs for *push/pull* strategies in Figure 3.1. To accomplish a common goal for the team, a set of functionally equivalent strategies are supplied by applications.

In a TG, nodes represent tasks (T_i), directed edges between tasks represent communication (*sender/receiver*) relationships or precedence (*producer/consumer*) constraints. The amount of communication is denoted as a communication cost attached to the edges. In our model, all tasks are periodic. Each task is characterized by a *period* P_i, *Worst Case Execution Time (WCET)* C_i, and relative *deadline* D_i, here, $D_i = P_i$. Periods can be *different* for different tasks. But if the sender and receiver run with arbitrary periods, task executions may get out of phase, which results in large latencies in communication [78]. Harmonicity constraints can simplify the reading/writing logic, reduce those latencies [77] and increase the feasible processor utilization bound [84]. To this end, we assume the period of a receiver task as a multiple of the related sender's period.

3.3.2 Our Goal

Given a set of sites and a set of functionally equivalent strategies, our goal is to find a feasible strategy. A strategy is *feasible* if and only if:

- within the LCM (Least Common Multiple) of task periods, each instance of a task is scheduled to run at its start time, and the completion time will not be later than its relative deadline;

- all constraints, such as *precedence*, are satisfied.

Based on the nature of the application, some tasks, e.g, sensor and motor systems, are required to run on designated processors, e.g., a specific robot platform. Other tasks, such as control or computation tasks, however, can be assigned to any site in a team. To find a feasible strategy, especially when the temporal and physical resources are tight, the system needs to:

1. assign unallocated tasks to appropriate processors so that the communication cost and the workload of each processor is minimized;

2. determine a feasible schedule for all task instances, including communication tasks.

The method for constructing communication tasks will be discussed in Section 3.5. In this chapter, we assume the *Leader* robot is responsible for computing the feasibility of all available strategies and deciding which strategy the team should implement. The decision and execution process works as follows. The initial team settings are supplied by the application. At run time, the *Leader* determines feasibility and chooses a strategy for the team to execute. The *Leader* broadcasts this result to the rest of the team and waits for confirmation responses. (How these messages are propagated over the network to the rest of the team is beyond the scope of this chapter.) After this broadcast phase is finished, the team members start the execution phase. We set a supervisory period, e.g., a multiple of the LCM, as the time interval during which the system runs under the current strategy. Once the supervisory period ends, the *Leader* checks if the team requires a new strategy due to a change in team

size or topology, recomputes a new strategy, broadcasts it to the team, and a new execution phase begins.

3.4 Allocation Algorithms

In this section, we will present two heuristic algorithms to allocate real-time tasks to distributed sites with the goal of minimizing communication cost and processor workload. Before we give the details of the allocation algorithms, let us first illustrate some of the attributes or concepts that will be used in this chapter.

T_i : Task ID

C_i : Worst Case Execution Time (WCET) of task T_i

P_i : Period of task T_i

D_i^n : Deadline of the n^{th} instance of T_i

E_i^n : Earliest start time of the n^{th} instance of T_i

S_x : Site (Processor) ID

u_x : Utilization of Processor S_x

u_x^i : Utilization of S_x that T_i is on

$T_i \rightarrow T_j$: Precedence constraint between T_j and T_i

$CCR_{i,j}$: Communication Cost Ratio for $T_i \rightarrow T_j$

If there are n tasks on processor S_x, the processor utilization is given by

$$u_x = \sum_{i=1}^{n} \frac{C_i}{P_i}$$

Here, CCR_{ij} is defined as:

$$CCR_{i,j} = \frac{communication\_cost\ (T_i \rightarrow T_j)}{C_i + C_j}$$

3.4.1 Greedy Algorithm

This algorithm takes into account the amount of communication and computation involved for *each pair* of communicating tasks. A decision is made as to whether these two tasks should be assigned to the same site, depending on the utilization condition; thereby, eliminating the communication cost. For schedulability purpose, a threshold t is used to minimize and balance the utilization of each processor. Initially, t is the maximum utilization value among all all processors which have been assigned preallocated tasks.

Greedy Allocation Algorithm
Input: a task graph $G = (E, V)$; P_i, C_i for each task T_i; communication costs. preallocated tasks with related processors; the number of processors m
Output: an assignment to all unallocated tasks such that the utilization of each processor (site) is less than 1.
Variables:
F: allocated task set
I: unallocated task set
R: set of $(CCR_{i,j}, S_x, T_j)$, s.t. $T_i \rightarrow T_j, T_i \in F, Site(T_i) = S_x, T_j \in I$
U: set of utilizations
t: utilization threshold
Algorithm 3.1:
1. Initialize $U = \{u_i | i = 1, 2, ..., m\}$, that is for each processor S_i:
   $u_i = \sum_j \frac{C_j}{P_j}, T_j \in F \wedge Site(T_j) = S_i$;
2. Let $t = max(u_i), u_i \in U$; /* initialize the utilization threshold */
3. **If** $(t > 1)$,**do**
4. exit without solution;
5. **For** all $T_i \in F$ /* initialize R */
6. Insert $(CCR_{i,j}, S_x, T_j)$ to R, s.t., $T_i \rightarrow T_j, Site(T_i) = S_x, T_j \in I$;
7. **While** (I is not empty) **do**
8. Let T_j be the task that has the largest value $CCR_{i,j}$ out of R;
9. Let $u_x^{'} = (u_x + \frac{C_j}{P_j})$; /* $T_i \rightarrow T_j, Site(T_i) = S_x$ */
10. **If** $((S_l = \textbf{thresholdUpdate}(u_x^{'}, S_x, T_j)) < 0)$, **do**;
11. exit without solution; /* cannot find an appropriate processor */
12. Update set F, I s.t., $F = F \cup \{T_j\}, I = I \setminus \{T_j\}$;
13. Delete $(CCR_{i,j}, S_x, T_j)$ from R;
14. **For** all T_k s.t., $T_j \rightarrow T_k$ and $T_k \in I$
15. Insert $(CCR_{j,k}, S_l, T_k)$ to R; /* update R, $Site(T_j) = S_l$ */

Table 3.1. ***Greedy*** allocation algorithm

At each step, among all unallocated tasks, the algorithm selects the one that has the *largest* communication cost ratio, and then attempts to assign it to the same processor as its sender. For instance, if $CCR_{i,j}$ has the largest value, where $T_i \rightarrow T_j$ and T_i is located on site S_x, the algorithm will attempt to allocate T_j to S_x, based upon whether or not the utilization of S_x becomes larger than the threshold t. For instance, let $t^{'}$ denote the "expected" utilization of S_x if T_j is assigned to S_x. If $t^{'} < t$, T_j is allocated to S_x and t keeps the value. Otherwise, the algorithm will find a site S_l that currently has the least utilization, and then attempts to assign T_j to S_l. Two cases need to be discussed under this situation.

Case 1: $t^{'} > 1$. In this case, if S_l is different from S_x, and the new utilization $u_l^{'}$ (after loading the selected task T_j) is less than 1, T_j is assigned to S_l and the threshold is updated to the max value of t and $u_l^{'}$. Otherwise, no processor can load the task and the algorithm fails.

Case 2: $t^{'} \leq 1$. In this case, we can simply assign the task to the processor with least utilization, and then update the threshold to $t^{'}$. Now the new threshold reflects the new workload demands.

Depending on the threshold update result, if an appropriate processor is found, the algorithm moves on to the next unallocated task that currently has the largest CCR, using the new threshold. The algorithm is deemed successful if no task is left unallocated. However, if no site can be found for the selected task because any of the processor's utilization will be greater than 1 after loading this task, the algorithm fails. The pseudo-code for the *Greedy* allocation algorithm and the function of threshold update is shown in Table 3.2 and Table 3.1, respectively.

Let N be the number of tasks, M be the number of processors. Generally there are M^N allocation ways, and finding an optimal feasible allocation so that tasks meet all physical and temporal constraints is known to be NP-hard. In our algorithm, the **While** loop runs in $O(N^2)$ time. Consequently, the algorithm runs in $O(N^2)$ time.

Function of Threshold Update

Processor **thresholdUpdate**(float $t^{'}$, Processor S_x, Task T_j)

/* return the allocation or -1 if fails, T_j and processor S_x is selected, $t^{'} = u_x^{'}$ */

1. **Case 1:** $t^{'} \leq t$, **do** /* $t^{'}$ is less than the threshold t */
2. Assign task T_j to processor S_x;
3. Update U with the new utilization $u_x = t^{'}$;
4. **Return** S_x;
5. **Case 2:** $t^{'} > t$, **do** /* $t^{'}$ is larger than the threshold t */
6. Find S_l that has the least utilization $u_l = min(u_i), u_i \in U$,
7. Let $u_l^{'} = u_l + \frac{C_j}{P_j}$;
8. **Case 2.1:** $t^{'} > 1$, **do** /* processor S_x cannot load T_j */
9. **If** $(S_l \neq S_x) \wedge (u_l^{'} \leq 1)$, **do**
10. Allocate task T_j to processor S_l;
11. Update U with $u_l = u_l^{'}$;
12. $t = max(t, u_l^{'})$;
13. **Return** S_l;
14. **Else**
15. **Return** -1; /* cannot find an assignment for T_j */
16. **Case 2.2:** $t^{'} \leq 1$, **do** /* $u_l^{'} \leq t^{'} \leq 1$ */
17. Allocate task T_j to processor S_l;
18. Update U with $u_l = u_l^{'}$;
19. $t = t^{'}$;
20. **Return** S_l;

Table 3.2. Threshold update

3.4.2 Aggressive Algorithm

Before we introduce a new algorithm, let us first look at an example that is depicted as a task graph in Figure 3.3, to illustrate the motivation. Here, T_1 and T_2 are preallocated on site S_1, T_3 is on site S_2; T_4, T_5 and T_6 are under consideration. The numbers attached to the arrows are communication costs. Table 3.3 shows the related parameters of each task, the communication ratios are given in Table 3.4. For S_1 and S_2, the initial utilization values are : $u_1 = 0.45, u_2 = 0.5$.

According to *Greedy*, since $CCR_{2,5}$ is the largest value among all communication ratios, T_5 is considered next. If assigning T_5 to S_1, the same site as T_2, the utilization will become $u_1 = 0.45 + 0.2 = 0.65 < 1$. Hence, T_5 is allocated to S_1. Now, let us consider T_4 and T_6. Because $CCR_{3,4}$ has the largest value among the remaining

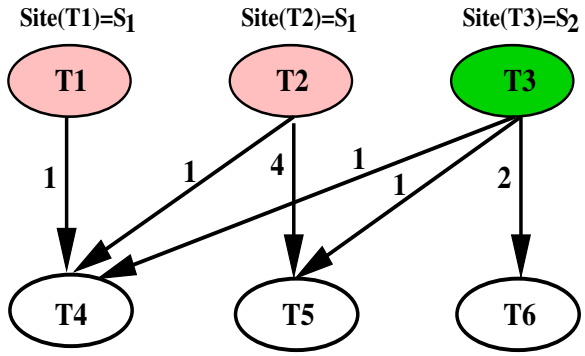


Figure 3.3. A simple task graph example

ratios, task T_4 will be assigned to site S_2, the same site as T_3, and u_2 becomes 0.7. At this moment, no site can load task T_6 — the utilization will be larger than 1 if loading T_6, therefore, the algorithm finally fails.

Task	T_1	T_2	T_3	T_4	T_5	T_6
WCET (C_i)	2	5	1	4	8	16
Period (P_i)	10	20	2	20	40	40

Table 3.3. Parameters for tasks in Figure 3.3

Communication	$T_1 \rightarrow T_4$	$T_2 \rightarrow T_4$	$T_3 \rightarrow T_4$	$T_2 \rightarrow T_5$	$T_3 \rightarrow T_5$	$T_3 \rightarrow T_6$
CCR	$\frac{1}{6}$	$\frac{1}{9}$	$\frac{1}{5}$	$\frac{4}{13}$	$\frac{1}{9}$	$\frac{2}{17}$

Table 3.4. Communication ratios for related communicating tasks in Figure 3.3

The second allocation algorithm we propose takes into account the total communication cost, and selects the task that has the largest accumulated CCR to do allocation. For T_4 in above example, since the accumulated communication cost from S_1 is greater than that from S_2, i.e., $(CCR_{1,4} + CCR_{2,4}) > CCR_{3,4}$, it is better to assign T_4 to S_1, other than S_2.

Because the utilization bound is still required for schedulability purpose, once the task is selected, the function of assignment and threshold update is the same as in

Greedy. To this end, for the *Aggressive* algorithm, R is the set of $(\sum_i CCR_{i,j}, S_x, T_j)$, where $\forall T_i, T_i \in F, T_i \rightarrow T_j, Site(T_i) = S_x$. Line 6 in Table 3.1 is changed to: Insert $(\sum_i CCR_{i,j}, S_x, T_j)$ to R; and line 13 is changed to: Delete $(\sum_i CCR_{i,j}, S_x, T_j)$ from R; line 15 is changed to: Insert $(\sum_i CCR_{i,k}, S_l, T_k)$ to R, where $\forall T_i, Site(T_i) = S_l, T_i \rightarrow T_k$. Following *Aggressive*, when T_4 is considered, it will be assigned to S_1 and thereafter $u_1 = 0.65+0.2 = 0.85$. Now, if we assign T_6 to site S_2, the utilization of S_2 becomes 0.9, and the algorithm succeeds. This slightly more complicated algorithm is shown to be more effective in the sense of higher schedulability.

3.5 Making Scheduling Decisions

After a successful task assignment is found, we need to find a feasible schedule for all instances of the tasks. Searching for a feasible schedule for real-time tasks subject to precedence constraints in a distributed environment is an intractable problem in the worst case, therefore, we propose to use heuristic methods. Before we discuss the approach, first, let us define some terminology.

- **Earliest start time** The earliest start time of an instance of a task is derived from the *precedence* constraints. Let L be the LCM of task periods. If task T_i has no predecessors, the first instance is ready to execute at time 0, denoted as $E_i^1 = 0$; and for the n^{th} instance of that task, $E_i^n = (n-1) \times P_i$, where $1 \leq n \leq L/P_i$. If T_i has predecessors, its first instance becomes enabled only when all its predecessors have completed execution. In order to achieve this condition, the tasks in the original task graph are topologically ordered. When a task T_i is processed, the lower bound of E_i^1 is set to $max(E_i^1, E_k^1 + C_k)$, where $\forall k, T_k \in Predecessors(T_i)$. Since we will model communication as a task if two communicating tasks are on different sites, and we have harmonicity constraints for all such pairs, initially, the lower bound of E_i^n is assigned to $(n-1) \times P_i + E_i^1$.

- **Communication task** If the pair of communicating tasks have been assigned to the same site, the communication cost is avoided; otherwise, communication needs to be scheduled. Consider $T_i \rightarrow T_j$ and $m = \frac{P_j}{P_i}$, we will construct m communication tasks; and the k^{th} such task T^k_{comm} has the following features.

  1. $P^k_{comm} = P_j$. T_j needs to process data sent from one instance of T_i only once during one period of T_j;
  2. $D^k_{comm} = P^k_{comm} - C_j$. This is an upper bound since the communication should finish its execution no later than the latest start time of T_j.
  3. $E^k_{Comm} = E^k_i + C_i$. This is a lower bound because the communication task should begin execution at least after the completion of the related instance of T_i;

  The data sent by the communication tasks for the same sender task are buffered at the destination site until the receiver task begin to process them. In this thesis, we assume the transmission is lossless once it is scheduled.

Given a task graph with location information, the first step is to build a set that includes all instances of each unallocated task, including the communication tasks. Then we use a search algorithm to find a feasible schedule. A scheduling decision determines the time at which a task can begin execution, in other words, the time when all its precedent tasks have completed execution before their respective deadlines. Within the LCM, each instance to be scheduled is treated as an individual entity.

Since it may take an exhaustive search to find a feasible schedule, which is intractable in the worst case, we adopt a heuristic approach. We take into account the most important properties of the real-time tasks and precedence constraints, namely, *deadline*, *earliest start time* and *laxity*, to actively direct the search towards a plausible path. The potential heuristic functions we use, $H(T)$ are as follows, where parameter

W is the weight factor used to adjust the effect of different temporal properties of the tasks.

- Earliest deadline first: $H(T) = Min\_D$;
- Minimum earliest-start-time first: $H(T) = Min\_E$;
- Minimum laxity first: $H(T) = Min\_L = min(D_i - (E_i + C_i))$;
- $H(T) = Min\_D + W \times Min\_E$;
- $H(T) = Min\_D + W \times Min\_L$;
- $H(T) = Min\_E + W \times Min\_L$.

The search attempts to determine a feasible schedule for a set of tasks in the following way. It starts with an empty partial schedule as the root and tries to extend the schedule with one more task by moving to one of the vertices at the next level in the search tree. It continues this process until a feasible schedule has been found. The heuristic function H is applied to each of the remaining unscheduled tasks at each level of the tree. The task with the smallest value is selected to extend the current partial schedule. Once a task is scheduled, the earliest start times of all its successors are updated accordingly. Because Min_E encodes the *precedence* constraints in the sense of their temporal relationship, it turns out that it outperforms other simple heuristics. The simulation studies also show that $(Min\_D + W \times Min\_E)$ has superior performance.

3.6 Evaluation

To study the features of the proposed algorithms, we conducted several experiments to evaluate the allocation algorithms with regards to schedulability. How these algorithms can be applied to an actual robotics application is discussed in Section 3.7. Tasks generated in a directed acyclic graph have the following characteristics:

- The computation time C_i of each task T_i is uniformly distributed between C_{min} and C_{max} set to 10 and 60 time units, respectively. The communication cost lies in the range $(CR \times C_{min},\ CR \times C_{max})$, where CR is the Communication Ratio used to assign communication costs. Experiments were conducted with CR values between 0.1 and 0.4.

- To address harmonicity relationships, we set a period range, $(minP_i^I, maxP_i^I)$, for each input task T_i (task without incoming edges), and $(1, maxP_j^O)$ for each output task T_j (task without outgoing edges), where $minP_i^I = Lower \times C_i$ and $maxP_i^I = Upper \times C_i$, $Lower = 1.1$ and $Upper = 4.0$. To ensure that the periods of output tasks are no less than those of input tasks, a parameter, $mult\_factor$ is used to set the upper bound of the period for output task T_j: $maxP_j^O = mult\_factor \times max(maxP_i^I)$, where T_i are input tasks and $mult\_factor$ is randomly chosen between 1 and 5. In order to make periods harmonic, first, we process input tasks and make their periods harmonic; then we tailor the techniques from [77] to process output tasks; finally, we use the GCD technique for intermediate tasks to achieve harmonicity constraints. The idea of computing GCD is to do a backward period assignment: a task T_k gets period P_k from all its successors so that $P_k = GCD\{P_l | P_l \in succ(T_k)\}$. Because of precedence constraints, periods of output tasks cannot be considered separately from those of input tasks, so P_1^O, which is the least period of output tasks, is calculated upon the largest period of input tasks (P_m^I), $P_1^O = \lfloor maxP_1^O / P_m^I \rfloor \times P_m^I$. Other output tasks' periods are computed upon P_1^O to achieve harmonicity.

- Parameter out_degree is used to set the precedence relationships in terms of data processed by multiple producers/consumers. For each task, except for output tasks, the out_degree is randomly chosen between 1 and 3. The total number of tasks in a task set is: $4 \times tasksetsize\_factor$, where $3 \leq tasksetsize\_factor \leq 8$,

and all the results shown here are for task sets with four input and four output tasks, though we have conducted experiments with different numbers.

All the simulation results shown in this section are obtained from the average value of 10 simulation runs. For each run, we generate 100 test sets, each set satisfying $\sum_{i=1}^{n}(C_i/P_i) \leq m$, where n is the number of tasks and m is the number of processors. For a given task set, if this condition is not held, at least one processor utilization will be larger than 1. The scheme used here is to remove the task sets that are definitely infeasible. Obviously, this does not eliminate all infeasible task sets because the presence of communication costs are not considered. However, since feasibility determination is intractable, if one heuristic scheme is able to determine a feasible schedule while another cannot, we can conclude that the former is superior. Therefore, the performance of the algorithms and parameter settings are compared using the *SuccessRatio* (SR):

$$SR = \frac{NT^{succ}}{NT}$$

NT^{succ} is the total number of schedulable task sets found by the algorithm, and NT is the total number of task sets tested. In this chapter, NT is 100 for each simulation run, and for each result point in the graphs, we have the average value of 10 runs. That is: $SR = (\sum_{i=1}^{10} SR_i)/10$, where $SR_i = NT_i^{succ}/100$.

The tests involved a system with 2 to 12 processors connected by a multiple-access network. Resources other than CPUs and the communication network are not considered.

3.6.1 Selecting a Scheduling Heuristic

In order to eliminate the bias from scheduling heuristic functions when study the performance of allocation algorithms, we first investigate the scheduling heuristic functions.

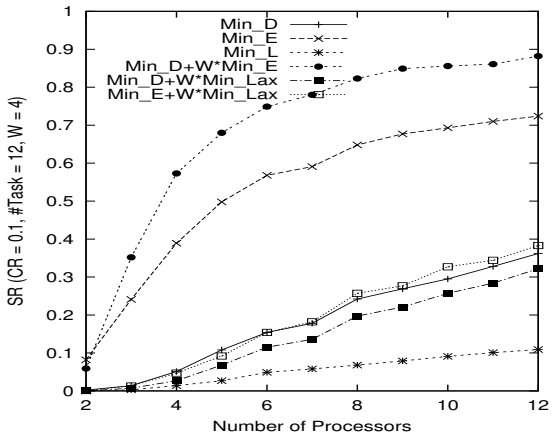


Figure 3.4. Effect of Scheduling Heuristics

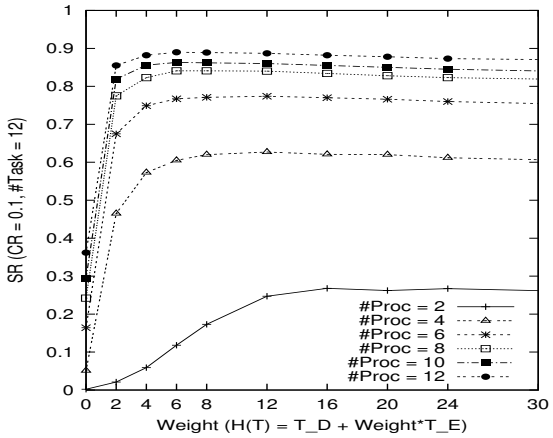


Figure 3.5. Effect of Weight

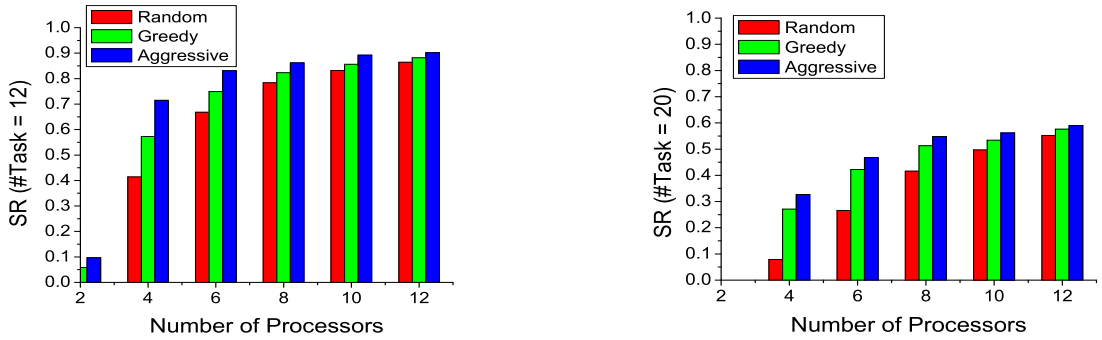


Figure 3.6. Performance of allocation algorithms ($CR = 0.1$).

For both *Greedy* and *Aggressive* algorithms, we find Min_E is the best simple scheduling heuristic, while $Min\_D + W * Min\_E$ has substantially better performance than other heuristics including Min_E. This is because *earliest start time* of each instance of a task encodes the basic *precedence* information, and besides *precedence* constraints, another important factor, *deadline*, is also taken into account in $Min\_D + W * Min\_E$. Figure 3.4 shows the effect of different scheduling heuristic functions when using *Greedy* algorithm. We have similar results for *aggressive* algorithm.

Since $Min\_D + W * Min\_E$ is a weighted combination of simple heuristics, we investigate its sensitivity to the weight (W) values for various number of processors. When $W = 0$, the heuristic becomes the simple heuristic Min_D, and does not perform well. When the weight changes from 0 to 4 (or to 12 if the number of processors is 2), we see a significant performance improvement. The algorithm is robust with respect to heuristics, because the performance is only slightly affected when the weight changes from 4 to 30 (or 12 to 30 if the number of processors is 2). To this end, we will choose $W = 4$ for following experiments. The results are shown in Figure 3.5

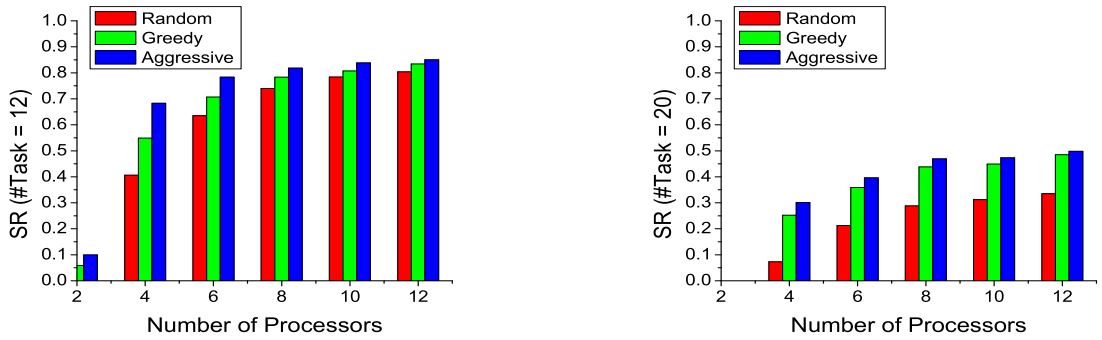


Figure 3.7. Performance of allocation algorithms ($CR = 0.4$).

3.6.2 Performance of Allocation Algorithms

In this section, we evaluate the performance of three allocation algorithms: *Greedy*, *Aggressive* and *Random*. The *Random* scheme will randomly assign an unallocated task to a processor as long as the utilization is less than 1. Figure 3.6 and Figure 3.7 illustrate the results when communication ratio is set to 0.1 and 0.4, respectively. As shown in the graphs, for each instance with different task set size, *Aggressive* outperforms *Greedy*, and *Greedy* outperforms *random*. The gains come from two factors: 1) the elimination of communication cost, 2) minimizing utilization for each processor. Since *Greedy* only considers the individual communication cost at each step, while *Aggressive* clusters and eliminates communication costs as many as possible, it is not surprising that *Aggressive* achieves better performance than *Greedy*.

The other observation is when the communication cost is heavier, the improvement in performance of *Greedy* or *Aggressive* is larger. Table 3.5 shows the difference in improvement of *Greedy* and *Aggressive* over *Random*. For both *Greedy* and *Aggressive*, in most cases, the improvements with $CR = 0.4$ are much greater than those with $CR = 0.1$. When $CR = 0.4$, the communication introduces more workload, and therefore, the system has more resource contention in terms of utilization boundary and deadline guarantee. So communication costs dictate the schedulability much more than the case when $CR = 0.1$. In contrast to random assignment, our approaches

exploit this important property to direct the allocation assignment, hence, they work better in the resource-tight environment.

#Processor	4	6	8	10	12
CR = 0.1	19.2	15.6	9.7	3.7	2.4
CR = 0.4	17.9	14.6	15.0	13.7	15

(a) *Greedy* over *Random*

#Processor	4	6	8	10	12
CR = 0.1	24.8	20.2	13.2	6.5	3.8
CR = 0.4	22.8	18.3	18.1	16.1	16.3

(b) *Aggressive* over *Random*

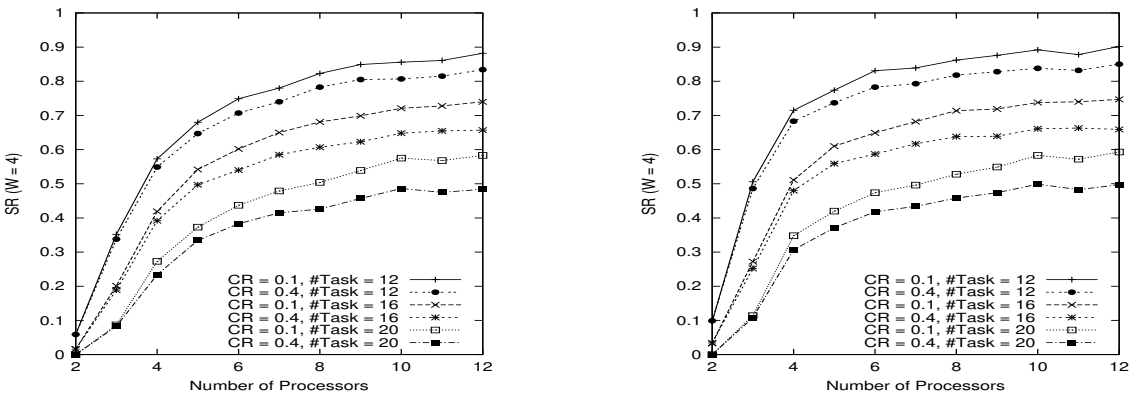
Table 3.5. Improvement of *Greedy* and *Aggressive* over *Random* (Percentage).

Finally, we find as the number of processors increases, the improvement for both *Greedy* and *Aggressive* tends to decrease for a given task set. This result further demonstrates the tighter the resource, the better our algorithms perform.

3.6.3 Effect of Communication

Using those allocation approaches minimizes the total communication cost by allocating sender and receiver tasks to the same site. However, in cases where such tasks have to be placed on different sites, the communication cost becomes a very important factor in overall performance. To investigate the effect of communication, we compare the results with $CR = 0.1$ and $CR = 0.4$ by varying the number of processors and the number of tasks. The results are illustrated in Figure 3.8.

Our results show that when the number of processors is very limited, e.g., 2 or 3, the performance is almost the same. This is because in such a situation, it is hard to find a feasible schedule for both cases. But as the number of processors increases, the performance for $CR = 0.1$ is better than that for $CR = 0.4$. This is because each



(a) Greedy (b) Aggressive

Figure 3.8. Effect of Communication

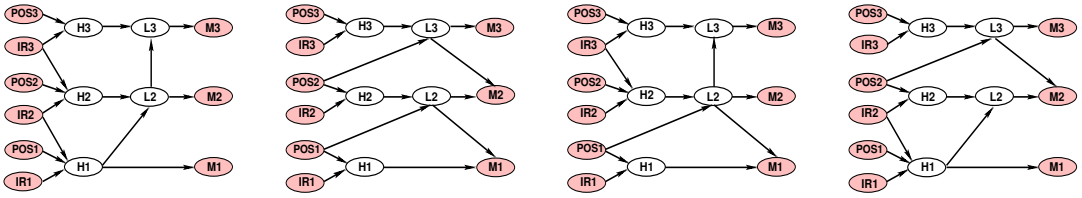
communication introduces extra workload in addition to the precedence constraints, if the communicating tasks have to be assigned to different processors. When the communication ratio (CR) is set to be larger, the communication costs are bigger, which have more impact on the earliest start time of related consumers and the overall system.

3.7 Application of Our Algorithms to Mobile Robotics

In this section, we return to the robotic problem discussed in Section 3.2, where two strategies, *push* and *pull*, are given for a team of two robots. In Table 3.6, the WCET of tasks are taken from an experimental implementation on a StrongARM 206MHz CPU; in Table 3.7, communication costs are based on the bytes transmitted using 802.11b wireless protocol with 11 Mbit/s transmission rate. Although 802.11b does not allow for real-time transmission guarantees, by prescheduling communications, medium contention is avoided. The periods are assigned with 220 ms for all sensor tasks and motor drivers by the application. Therefore, the periods of controller tasks are also designed to be 220 ms by the harmonic constraint. Though these figures

Task	$IR1(2)$	$Pos1(2)$	H_1	H_2	L_2	$M1(2)$
Push	20	120	35	25	5	20
Pull	20	120	25	25	18	20

Table 3.6. WCETs(ms) of tasks in Figure 3.1



(a) $\{Push, Push\}$ (b)$\{Pull, Pull\}$ (a) $\{Push, Pull\}$ (b) $\{Pull, Push\}$

Figure 3.9. Four possible strategies for a team of three robots using the ***push*** and ***pull*** controllers

are given based on tasks in Figure 3.1, they are compatible to tasks that occur with more robots. Let us consider the scenario when a third robot wants to join the team. Since the push and pull controllers are pairwise, there are four strategies, composed of *push* and *pull* controllers, that the application can use for a team of three robots: $\{Pull, Pull\}$, $\{Pull, Push\}$, $\{Push, Pull\}$, and $\{Push, Push\}$. The task graphs for these four strategies are shown in Figure 3.9.

First, let us use the *Aggressive* algorithm to analyze the task assignment in each strategy. In this example, since the accumulated communication cost is considered, the allocation is the same for all strategies: H_1 is assigned to S_1, H_2 to S_2, H_3 to S_3, L_2 to S_2 and L_3 to S_3.

	$IR_{1(2)} \rightarrow H_{1(2)}$	$Pos_{1(2)} \rightarrow H_{1(2)}, L_2$	$H_{1(2)} \rightarrow L_2$	$H_1 \rightarrow M_1$	$L_2 \rightarrow M_{2(1)}$
Push	0.02327	0.01236, 0	2.979	2.979	2.979(0)
Pull	0.02327	0.01236	0(2.979)	2.979	2.979(2.979)

Table 3.7. Communication costs for Figure 3.1

Strategy	$\{Ph, Ph\}$	$\{Pl, Pl\}$	$\{Ph, Pl\}$	$\{Pl, Ph\}$
Site 1	195	205.979	222.979	195
Site 2	202.979	208.958	220	205.979
Site 3	210.958	203	230.958	203

Table 3.8. The completion time for all strategies

Next, the algorithm will see which strategies are schedulable under the heuristic $Min\_D + W \times Min\_E$. To simplify the analysis, here W is set to 1. The completion times for tasks on each site are shown in Table 3.8. The algorithm finds that, except for $\{Push, Pull\}$, denoted as $\{ph, pl\}$, all other strategies are feasible, but with different completion times (including communication delay). Since multiple strategies are feasible, the application can use some criteria to rank the strategies. In this case, if the total laxity is used as the criterion, the application will choose the $\{Pull, Push\}$ strategy, because it has the maximum value.

The application can then use the feasible results when computing new sets of strategies. For example, if at some time a fourth robot joins the team, the application immediately knows that any strategy that contains $\{Push, Pull\}$ will not be feasible, since that strategy was already determined to be infeasible. Therefore, the application can use the feasibility analysis to prune infeasible strategies as the team's size scales. The idea of using the proposed algorithm to do the scalability analysis is shown in [99].

3.8 Conclusions

In this chapter, we consider the problem of task allocation on distributed robotic systems. The question on how to improve system schedulability while tasks maintain their physical constraints and meet their deadlines poses new challenges. In this study, issues involved in the design and development of real-time distributed robotic systems are addressed. The ideas of minimizing communication, balancing workload and using

utilization bound are developed well together to improve the system feasibility. The major contributions are:

- Taking pairwise communication costs into account for task assignment improves system performance, but not as much as when the accumulated communication costs from the same processor are considered.

- Using dynamically increased utilization bound not only minimizes total communication costs and utilization, but also balances the workload of each processors. This property can be further explored to other multiprocessor/distributed real-time applications, such as, finding minimum number of processors while meeting temporal constraints.

- When deadline and earliest start time are weighted combined in the scheduling policy, the heuristic performs significantly better than others, since it encodes both the precedence and the timeliness constraints.

CHAPTER 4

OPTIMIZATION OF SENDING DATA OVER WIRELESS TRANSMISSION

4.1 Introduction

In this thesis, we consider a class of wireless sensor applications that impose timeliness constraints on the transmission and processing of sensory data. We refer to such applications as real-time sensor applications.

The problem of scheduling *processing* tasks to meet physical and timeliness constraints in a real-time sensor application has been discussed in last chapter. In this chapter, we focus on the problem of transmitting sensor data to optimize the system performance. Due to their low cost and wide availability, we assume that the sensor applications employ commodity 802.11-based wireless networks. However, such wireless networks employ a best-effort technique based on CSMA/CA, where packet transmission intended for one neighbor may cause interference with other neighbors. Interference not only significantly reduces the channel utilization, but may also cause unbounded delay, and hence, they cannot be directly used in time-sensitive sensor applications, due to possible collisions, back-offs and false blocking problems.

In the CSMA/CA-base networks, a node must explicitly request permission for transmission via a "request-to-send" (RTS) packet and must receive a "clear-to-send" (CTS) acknowledgment before sending data. Once receive these messages, all nodes in the vicinity must avoid transmission for the required transmission duration and are thus *blocked*. Such a protocol can lead to *false blocking* and *blocking propagation*. As shown in Figure 4.1, node S_1 transmits data to node R_1 and node R_2 is blocked

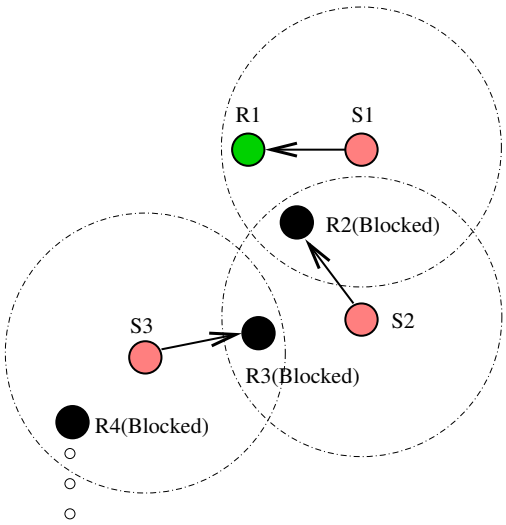


Figure 4.1. False blocking problem (blocking propagates from R_2 to R_3 to R_4)

since it within S_1's transmission range. While R_2 is blocked, node S_2 sends a *RTS* packet to R_2 and receives no response. However, node R_3 which is within S_2's range also receives the *RTS* and is blocked. If S_3 wishes to send data to R_3 and issues a *RTS*, it will not receive the *CTS* from R_3 and has to back off exponentially although, the transmissions $S_1 \rightarrow R_1$ and $S_3 \rightarrow R_3$ is, in fact, able to perform in parallel.

In this chapter, we develop techniques for optimizing the transmission of sensor data over wireless networks. We argue that the problem is NP-complete and present heuristics based on edge coloring to address this problem. These techniques should consider the wireless constraints, such as range and interference constraints, while scheduling message transmissions. In order to minimize the completion time of sending a set of sensor messages, we exploit the special characteristics of wireless transmission and design techniques to parallelizing the transmissions to the much extent, and in the meanwhile, the collisions and interference are *explicitly* avoided.

We also present an A^*-based optimal algorithm to enable comparisons with our heuristics. We conduct a detailed simulation study to evaluate our heuristics and find that the minimum weight color heuristic is robust to increases in communication

density and yields results that are close to the optimal solution. In conjunction with clustering and grouping techniques, we believe such communication scheduling techniques can adapt well to large scale sensor networks.

The rest of this chapter is structured as follows. Section 4.2 presents our system model and the problem formulation. Sections 4.3 and 4.4 present our edge coloring-based heuristics and the optimal solution, respectively. We discuss the implementation issues of schedule construction and decision propagation in Section 4.5. Section 4.6 presents simulation results. And, finally, Section 4.7 summarizes our work.

4.2 System Model and Problem Formulation

In this section, we present the system model, and then formulate the problem of scheduling communication in real-time sensor applications.

4.2.1 System Model

Consider a wireless sensor application with N nodes. We denote designated sender nodes by S and receiver nodes by R. Each node has a wireless network interface with a certain transmission range; depending on the exact wireless interface employed (e.g., 802.11b versus 802.11g), different nodes may have different transmission ranges. Each node can be a sender or a receiver or both, and no base-stations are assumed in this environment. A node should be within the transmission range of a sender to be an eligible receiver, and all communication is assumed to be unicast. The terms source and sender as well as sink and receiver are used interchangeably in this chapter.

The communication medium is assumed to be shared by all nodes in the system. If a receiver is within the range of multiple senders, then interference may happen if more than one sender attempts to transmit simultaneously. However, there will be no interference if two receivers are *mutually* outside the other sender's range. In Figure 4.2, R_1 is in the transmission range of sender S_1 and S_2; if both senders

attempt to simultaneously communicate with their receivers, interference may occur. On the other hand, S_3 can transmit messages in parallel with either of the other transmissions. In this work, we assume that the location of each node is known at all times, and thus the nature of the overlap can be determined for the purpose of scheduling the network transmissions. This is a reasonable assumption, since in the robotics example, robots carry GPS receivers and can additionally use localization algorithms [63] to precisely determine their locations.

Observe in Figure 4.2 that source S_2 sends data to two different receivers R_2 and R_3. Such a scenario might result from the need to transmit data from different sensors on robot S_2 to different robots. The unicast nature of the communication necessitates separate messages to each receiver.

To precisely state these assumptions, let $Range(S_i, R_x)$ denote that R_x is within the transmission range of S_i, and $S_i \rightarrow R_x$ denote a message transmission from S_i to R_x. We have following system constraints.

- *Network Interface Constraint:* At any instant, a node can be either a sender or a receiver, but not both.

- *Range Constraint:* A node can receive a message only if it is within the sender's range, i.e., $S_i \rightarrow R_x \implies Range(S_i, R_x)$.

- *Interference Constraint:* Two simultaneous transmissions will not interfere if and only if both receivers are mutually outside the other sender's range. That is, $\forall (i \neq j),\ (S_i \rightarrow R_x) \wedge (S_j \rightarrow R_y) \wedge \neg Interference(S_i \rightarrow R_x,\ S_j \rightarrow R_y) \iff \neg Range(S_i, R_y) \wedge \neg Range(S_j, R_x)$.

- *Unicast Constraint:* Each message can have only one recipient.

Given a set of senders, receivers, their locations and transmission ranges, the communication scheduler must determine a transmission schedule so that the above

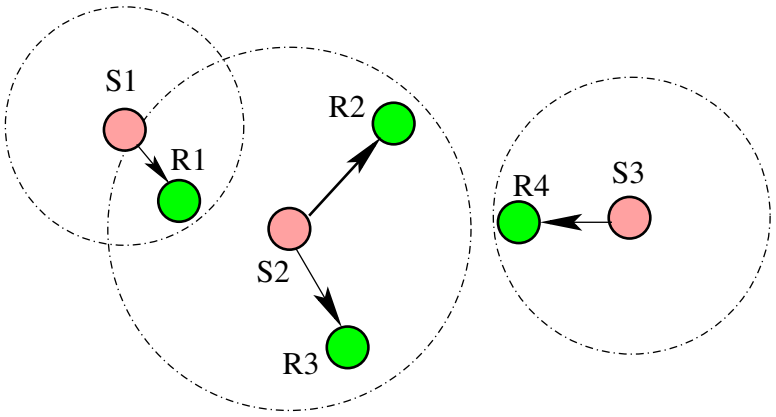


Figure 4.2. A communication example

constraints are satisfied and the time to complete all transmissions is minimized. We refer to this problem as the Optimal Parallel Communication Scheduling (OParCS) problem.

4.2.2 Graph-based Representation of the Problem

Consider a set of messages that are awaiting transmission at various nodes. The scheduling problem can be formulated using a weighted, directed graph $G = (V, E)$, in which each vertex denotes a node in the sensor application, and a directed edge from vertex v_i to v_j indicates that a message needs to be sent from v_i to v_j. The weight w of the edge denotes the transmission cost (time) and is a function of the message length (and the transmission rate). We refer to this graph as the *communication graph*. Each vertex is associated with a transmission range, therefore, the interference set, $I \subset E \times E$, includes all pairwise edges that will incur interference if messages are transmitted through those edges at the same time. If the range of each transmission is known in advance, we can obtain the interference set by checking the interference constraints in polynomial time. Given such a graph and associated interference set, the OParCS problem can be formulated as follows.

Input: Graph $G = (V, E)$, a weight function w that assigns a positive weight to each edge, and the interference constraint set $I \subseteq E \times E$.

Problem: Find a partition of E into disjoint sets $E_1, E_2, \ldots, E_n$ such that,

1. $\forall e_i, e_j \in E_i,\ (e_i, e_j) \notin I$,

2. $\forall e_i, e_j \in E_i,\ e_i, e_j$ do not share a common endpoint in G,

3. $\sum_{1 \leq i \leq n} \max_{e \in E_i}(w(e))$ is minimized.

The first condition avoids interference during message transmissions, while the second condition addresses the network interface and unicast constraints; the range constraint is implied in the input.

Proposition 1: The OParCS problem is NP-complete. The problem is NP-complete even if all weights are equal.

Proof:

Step 1: OParCS $\in NP$. since given the disjoint sets, validating them and compute the objective function $\sum_{1 \leq i \leq k} max_{e \in E_i}(w(e))$ can be done in polynomial time.

Step 2: OParCS is **NP**-hard. We prove that Minimum-Edge-Coloring (MEC) is polynomially reducible to OParCS problem, i.e., MEC $\leq_p$ OParCS. The Minimum-Edge-Coloring is an **NP**-complete problem [40].[1]

Given an instance of Minimum-Edge-Coloring with graph $G = (V, E)$, we can formulate an instance of the OParCS problem as follows:

- let the graph $G = (V, E)$ be $G = (V, E)$ in OParCS;
- $\forall e \in E$, let $w(e) = 1$;

[1] **Input:** Graph $G = (V, E)$.
Question: A coloring of E, i.e., a partition of E into disjoint sets $E_1, E_2, \ldots, E_n$ such that, for $1 \leq i \leq n$, no two edges in E_i share a common endpoint in G, and the cardinality of the coloring, i.e., the number of disjoint sets E_i is minimized.

- let $I = \emptyset$;

Suppose $E_1, E_2, \ldots, E_k$ is an optimal solution of OParCS. Now we establish that it, $E_1, E_2, \ldots, E_k$, is also a solution of the corresponding MEC problem. According to the constraints of the OParCS problem, $E_1, E_2, \ldots, E_k$ is a valid partition of E of MEC. Now we will show that $E_1, E_2, \ldots, E_k$ is an optimal solution of the MEC. We prove this by contradiction.

Assume that there exists another valid partition of E for MEC, $E_1^{'}, E_2^{'}, \ldots, E_{k'}^{'}$, and $k^{'} < k$. This partition, $E_1^{'}, E_2^{'}, \ldots, E_{k'}^{'}$ also satisfies the constraints of OParCS, and it has a lower value:

$$\sum_{1 \leq i \leq k'} max_{e \in E_i^{'}}(w(e)) = \sum_{1 \leq i \leq k'} 1 = k' < k = \sum_{1 \leq i \leq k} max_{e \in E_i}(w(e))$$

. Therefore, $E_1, E_2, \ldots, E_k$ is not an optimal partition, which contradicts the assumption.

This completes the proof.

4.3 Heuristic Communication Scheduling

Consider a simplified version of the problem where all messages are of equal length (edge weights are normalized to 1) and there is no interference between any of the messages. In this case, the only constraint is that all adjacent edges (sharing a common endpoint) cannot be scheduled simultaneously. Since an *edge coloring* of a graph is an assignment of colors to the edges such that adjacent edges receive distinct colors, we can color the edges so that edges with the same color can be scheduled simultaneously. In this simplified version, the number of colors in the edge coloring is equivalent to the time slots taken to schedule the transmission.

In this section, we propose polynomial time heuristics for the OParCS problem. Our heuristics use edge coloring as a building block — note that edge coloring can

not be used directly since it does not explicitly consider weights on edges, nor does it consider the interference constraint. Both of these factors should be taken into account when generating a transmission schedule for our problem.

In the following, we first present a color-based heuristic and then discuss three color selection strategies for coloring edges. The following are some of the attributes of a task graph that are useful to explain the algorithms.

e_i : edge ID

v_i : vertex ID

c_i : color ID

$e_{i,j}$: edge of $v_i \rightarrow v_j$

$c(e_i)$: the color of e_i

$e_i \sim e_j$: e_i is adjacent to e_j

$W(e_i)$: the weight, communication delay, of edge e_i

$W(c_i)$: the weight of the color c_i

$P(e_i)$: the palette associated with e_i

4.3.1 Edge Coloring Heuristic

The objective of the heuristic is to assign a color to each edge such that (i) no two edges sharing a common endpoint have the same color, (ii) no two edges with the same color interfere with one another, and (iii) the total time to complete transmission is minimized.

Given a communication graph and an interference set, we design a *palette* for each edge in the graph. Initially all palettes are identical and are assumed to contain a sufficiently large number of colors. How to make the palette large enough is not a focus of this study and is not addressed here. Also, each color in the palette is assigned

a weight. Initially, all colors have a weight of zero and as the heuristic progresses, the weight for a color will be set to the weight of the "heaviest" edge with that color.

The heuristic begins by picking the vertex with the maximum degree. If the degree of this vertex is d, then we need d distinct colors to color those incident edges. Once an edge has been assigned a color, that color is deleted from the palettes of all uncolored adjacent edges — two edges are said to be adjacent if they incident on a common vertex. From this point on, the heuristic repeats the following steps until all edges are colored.

1. Choose an edge e_i with the smallest palette (i.e., a palette with the least number of colors). This is because the smaller the palette is, the more constraint is on the possible colors. Ties are broken randomly.

2. Pick a color from the palette such that no other edges with that color interfere with this edge (we present three heuristics for this color selection step in the next section).

3. Delete the chosen color from the palettes of all uncolored edges that are adjacent to this edge, if the color is in those palettes.

4. Update the weight of the chosen color: $W(c_i) \leftarrow \max(W(c_i), W(e_i))$.

Once all edges have been colored, the transmission schedule involves scheduling all edges with the same color in parallel. For instance, all red edges are scheduled in parallel, then all the blue edges and so on. The total time to transmit messages of a given color depends on the edge with the maximum weight, which is also given by the weight of that color $W(c_i)$. Therefore, the time to transmit all messages is $\sum_k W(c_k)$, where k is the number of distinct colors that are needed to color the graph. The heuristic algorithm is depicted step by step in Algorithm 4.1.

Algorithm 4.1: Coloring-based Communication Scheduling

Input: A communication graph with message lengths and all constraints.

Output: The time to complete all transmissions without any conflict.

1. Find the vertex that has the maximum degree, and do:

   1.1 Color each incident edge e_x with a distinct color.

   1.2 For each neighbor e_n of the edge e_x, do:

   $P(e_n) \leftarrow P(e_n) \setminus \{c_y\}$, if $c(e_x) = c_y \wedge e_n \sim e_x$.

   1.3 Update the weights of the assigned colors, s.t.,

   $W(c_y) \leftarrow W(e_x)$, if $c(e_x) = c_y$.

2. Select the edge e_i that has the smallest palette. Break ties randomly.

3. Select the first appropriate color c_j based on the specific heuristic, and then test if there is an interference with existing same colored edges. If not, i.e., $\forall e_x,\ (c(e_x) = c_j) \wedge \neg Interference(e_i, e_x)$, assign c_j to e_i, s.t., $c(e_i) = c_j$; otherwise, choose the next appropriate color until no interference can happen.

4. If $W(e_i) > W(c_j)$, then $W(c_j) \leftarrow W(e_i)$.

5. $\forall e_k \sim e_i,\ P(e_k) \leftarrow P(e_k) \setminus \{c_j\}$, if $c_j \in P(e_k)$.

6. If there is at least one un-colored edge, goto step 2; otherwise calculate and output the final completion time: $\sum_i W(c_i)$.

4.3.2 Color Selection Policies

We propose three heuristics to choose a color from the palette for the selected edge. The first one is to consider the weight which is the function of communication length and attempts to cluster those messages that have close weights together. The second one is a random method and the third is the one that is commonly used in coloring problems.

- **Minimal Weight Color (MWC) Heuristic:** Observe that the total time to transmit messages of a certain color is governed by the longest message in the

set. If the palette of an edge contains a color that already indicates a *longer* message transmission time, i.e., the weight of the color is larger than that of the selected edge, then choosing that color will not result in any increase in the total time to transmit all messages (including the new one) of that color. This is the intuition behind this heuristic.

Suppose that the weight of the selected edge is $W(e_i)$. Consider only those colors from its palette that have a weight greater than $W(e_i)$. These are essentially colors that are associated with a message that is *longer* than the current message. If the palette has such colors, the MWC heuristic picks a color with the *least weight*. Note that the interference constraint must still be satisfied when picking a color.

If no color in the palette has a weight greater than the edge weight, then the heuristic simply picks the color with the maximum weight from the ones in the palette.

In summary, the heuristic attempts to assign messages with "similar" lengths with the same color and avoids increasing the weight of a color whenever possible. Doing so enables the heuristic to reduce the completion time for all messages.

- **Random Color Selection (RCS) Heuristic:** The random color selection heuristic picks a random color from the palette so long as the interference constraint is satisfied.

- **Least Used Color (LUC) Heuristic:** The least used color is a common heuristic for general coloring problems and we choose this heuristic to determine its effectiveness for our communication scheduling problem. This heuristic picks the least used color—the color with the least number of edges—such that the interference constraint is satisfied.

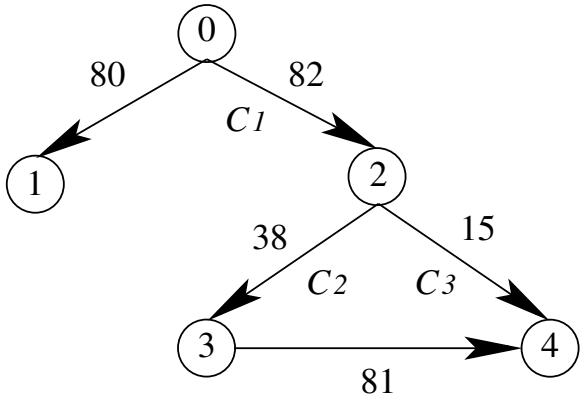


Figure 4.3. An example

4.3.3 An Example

Let us use the example depicted in Figure 4.3 to further explain the algorithm. The attached are the communication costs, denoted as transmission duration.

Suppose initially, each palette has 4 colors with weight of 0. Since vertex v_2 has the maximum degree, the heuristic begins by assigning distinct colors (c_i as shown in the figure) to all edges incident on v_2. After deleting the related color(s) from the palette, edge $e_{3,4}$ has the smallest palette (it has 2 colors, while $e_{0,1}$ has 3 colors), and is considered next.

If using MWC, because $W(c_1) = 82 > W(e_{3,4}) = 81$, color c_1 is selected for $e_{3,4}$; if using LUC, since c_4 is the currently least used color, c_4 is chosen and $W(c_4) \leftarrow 81$. Now, let us consider edge $e_{0,1}$. Any color except c_1 is a possible color. If we use MWC, since c_2 is the heaviest edge (the weights of c_2, c_3 and c_4 are all less than 80), c_2 is chosen and $W(c_2) \leftarrow 80$. For LUC, since all colors have been evenly used, any possible color can be chosen. In the above analysis, we assume that there is no interference in any step. Hence, the completion time for MWC is $W(c_1)+W(c_2)+W(c_3) = 82+80+15 = 177$, which is also the optimal solution; for LUC, before assigning edge $e_{0,1}$, the completion time is: $W(c_1)+W(c_2)+W(c_3)+W(c_4) = 82+38+15+81 = 216$. Note

here, we didn't show the process for RCS because it has multiple possible (random) choices at each step.

4.4 Optimal Communication Scheduling

In this section, we present an optimal solution to the OParCS problem—a solution that minimizes the completion time for all transmissions, subject to the constraints. Since the problem is NP-complete, the optimal solution has exponential complexity. Nevertheless, it is useful to consider such a solution to enable comparisons with our proposed heuristics.

Our technique uses directed search based on the A^* algorithm. A^* search has been shown to be *optimally efficient* in that no other search algorithm will expand fewer nodes in the search tree to locate the optimal solution [20]. The search process builds a search tree in which the root node represents the original communication graph. Expanding a node involves two steps. First, it finds *all* matchings for the current expanding node (a matching is a subset of edges such that no two edges share a vertex), subject to the interference constraints. Then, for each matching, the corresponding edges are deleted from the node (graph) and the resulting graph is added as a leaf node to the tree.

To find the next node to be expanded, an evaluation function $f(n) = g(n) + h(n)$ is needed in A^* so that the node with the lowest f value is selected. Here, $g(n)$ is the cost of the path from the root to node n, and $h(n)$ is the estimated cost of the cheapest path from n to the goal. To guarantee that the search algorithm is complete and optimal, A^* requires that h function should *never overestimate* the cost to reach the goal.

In our algorithm, the g function is defined as the sum of the costs of all matchings along the path from the root to the node. Let $W(e_x)$ denote the cost of edge e_x in the original communication graph, n_G denote a node representing a graph G in the

search tree, and $n'_{G'}$ denote a child of this node representing graph G', then $g(n'_{G'})$ is defined as:

$$\begin{aligned} g(n'_{G'}) &= g(n_G) + matching\_cost(M_i) \\ &= g(n_G) + max\left(W(e_x)\right),\ e_x \in M_i \end{aligned} \tag{4.1}$$

where, $G' = G - M_i$, M_i is a possible matching of G. Initially, $g(n_G) = 0$ for the root.

For a search node n_G and related graph G, if $W(v_i)$ denotes the weight of vertex v_i, then $h(n_G)$ is defined as:

$$\begin{aligned} h(n_G) &= max\left(W(v_i)\right) \\ &= max\left(\sum_j W(e^i_j)\right) \end{aligned} \tag{4.2}$$

where, e^i_j are all edges that are incident on vertex v_i. Observe that this h function *never overestimates* the cost to reach the goal. This is because the time to transmit the remaining messages is at least equal to the cost of the edges that can not be scheduled at the same time (i.e., are adjacent to each other).

4.5 Implementation of Communication Scheduler

In this work, we have presented a centralized algorithm and assume a centralized scheduler that has the perfect knowledge for the packet information at each node in the entire network flow graph. This is a reasonable assumption for applications such as coordinated robot team since a centralized path planner is used to determine the movement for each robot as well as for the whole group. And hence, the scheduler can run in conjunction with the planner to determine when each message should be transmitted. There are two design issues need to be discussed: i) how to build the schedule when a new flow message joins the network, and ii) how to propagate the schedule decision to the entire network once it is made at the leader node (robot).

The scheduler can be invoked periodically or on demand (like the scheduler studied in Spring System [73]). Upon each invocation, the scheduler must schedule all messages that have become ready for transmission since the previous invocation (thus, a newly arriving message must wait until the next scheduling instance before it can be scheduled for transmission). When invoked, the scheduler considers the current schedule (i.e., the messages that were scheduled in a prior invocation and are awaiting transmission) and all newly arrived messages at each node. Two approaches are possible for generating a new schedule. In the first approach, the unsent messages and the new messages are considered together to compute a new schedule. In this method, a new flow graph which includes all existent messages will be constructed, and a new schedule is calculated. The second approach is incremental—it determines a new schedule for the new messages and appends that schedule to the current schedule; the approach essentially assigns a transmission time to each new message while keeping the current schedule unchanged. In either case, the transmission times are then conveyed to the corresponding sender nodes.

Once a new schedule is constructed, the information needs to be broadcast to all nodes in the network with minimum time. Generally, in a network that has point-to-point links, the optimal solution of propagating information from one node to all the rest nodes in minimum time can be achieved by a shared minimum-height spanning tree, which can be obtained by breadth-first search algorithm or a generic Dijkstra's algorithm [61]. Note here, due to the potential collisions, the minimum-height tree may not be the optimal solution, and other method, e.g., constructing conflict-free shared tree may be helpful. Since this is not the focus of the study, we just assume here, a minimum-height tree is used in our scenario to convey the schedule decision.

4.6 Evaluation Results

We conduct a simulation study to evaluate the performance of our heuristics. Our study also compares the heuristics to the optimal solution computed by the A^* search algorithm.

We assume that the sensor network is represented by a 3-tuple (N, P, R), where N is the number of nodes, $P = \{(X_i, Y_i), 1 \leq i \leq N\}$ is the set of positions for the nodes in the area of $[100, 100]$ units. Each P_i is generated randomly in the area for its X and Y coordinates. R is the set of transmission ranges. We convert the communication network into a directed graph $G(V, E)$, so that $|V| = N$, and $(u \rightarrow v) \in E$ if and only if the Euclidean distance between u, v is less than or equal to R_u. The communication cost for each transmission W_i is randomly chosen over $[10, 100]$.

All results shown in this section are obtained as the mean of at least 1000 runs or with 95% confidence interval in $[-0.005, +0.005]$. In each run, one communication graph is generated with some specific settings. Since the algorithms attempt to minimize the total completion time for all transmissions in a graph, the performance is measured by comparing the completion times across different algorithms. For instance, if we have M graphs, and the comparison result for RCS to MWC per graph is:

$$Comp(RCS/MWC) = \frac{Time(RCS)}{Time(MWC)}$$

If we use *Completion Time Ratio (CTR)* to denote the mean, then for each result in the graph (e.g., RCS to MWC), we have:

$$CTR(RCS/MWC) = \sum_{M} Comp(RCS/MWC)/M$$

4.6.1 Comparison of Heuristics

In this section, we compare the three color selection heuristics by varying three system parameters, the number of nodes, the number of edges and the transmission

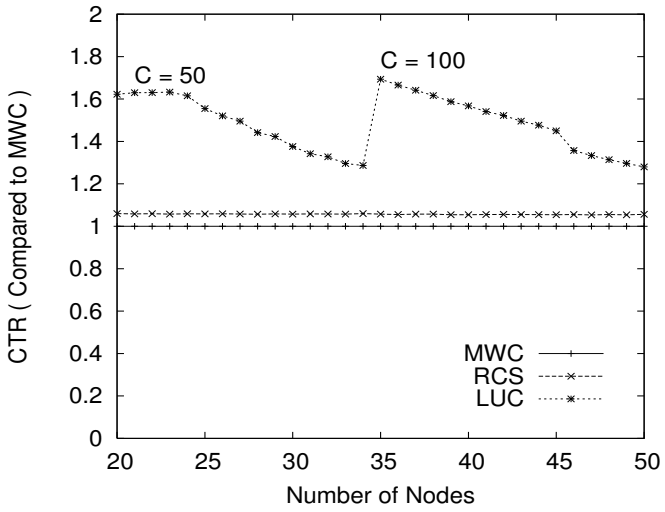


Figure 4.4. Effect of the number of nodes

range. Each time when the sensor network is constructed, the interference relationships are calculated based on their communication range, and the results are as the input for the further evaluation.

4.6.1.1 Effect of the Number of Nodes

To study the impact of the number of nodes (N), we systematically vary N from 20 to 50. For each N, we generate sufficient different communication graphs and determine the total time to schedule all messages (for each graph) using the three heuristics. Figure 4.4 shows the CTR of the three heuristics (we use the completion time of the MWC as the normalizing factor).

As can be seen, the minimum weight color (MWC) heuristic outperforms the other two heuristics. This is because it always put transmissions with approximate the same transmission durations in the same set and attempts to transmit them in parallel.

Random color selection (RCS) yields completion times that are within 8% of MWC, but the performance is not sensitive to the value of N. This is easy to understand. Since the first and the third steps of the *Edge Coloring Heuristic* have put effort to eliminate the constraints on the sharing of end points, if the networks are

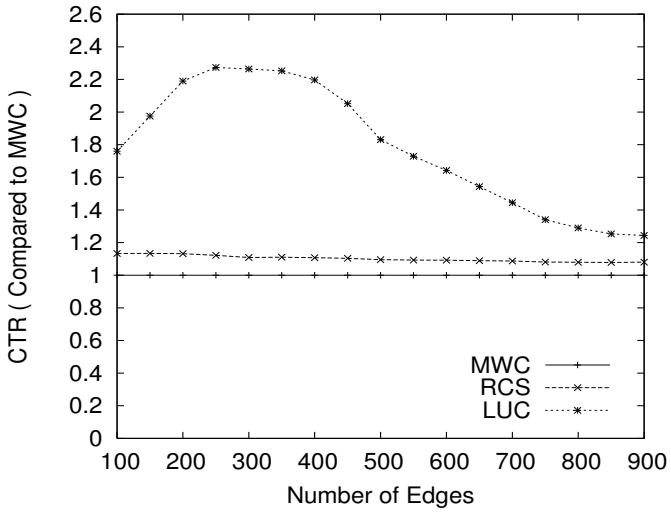


Figure 4.5. Effect of the number of edges

generated fully random, then we should expect to obtain a stable results for random algorithm.

The least used color (LUC) heuristic has the worst performance, though it is the most common heuristic used for normal coloring problems. This is because LUC always tries to find the color that is currently least used, which actually decreases the ability to schedule messages in parallel. The sudden increase in the LUC curve reflects the impact of the initial palette size. For a fixed palette size, when N increases, the CTR decreases. This indicates if the initial palette sizes are closer to the number of colors needed, the better are the LUC performance results. We choose more colors at $N = 35$ because the palette has not enough colors to cover all edges. (In the figure, $C = 50$ means that the initial number of colors in the palette is 50).

4.6.1.2 Effect of the Number of Edges

The number of edges reflects the communication density. Since the more edges are in the graph, the more communication constraints exist to achieve better performance. Here, we vary the number of edges in the communication graph systematically and study the performance of the three heuristics.

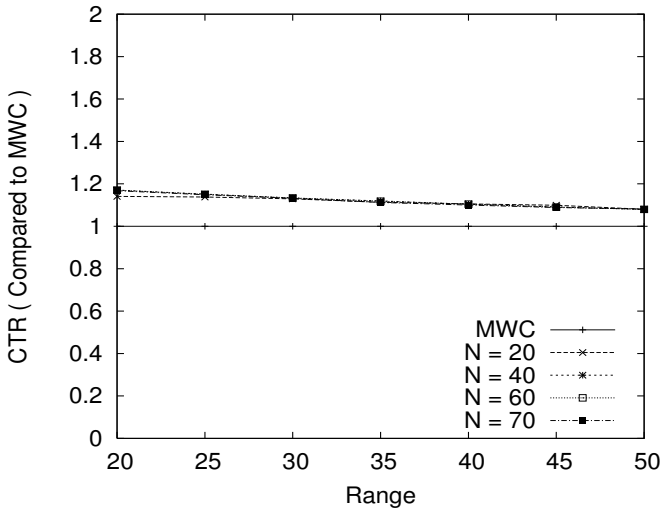


Figure 4.6. Effect of transmission range: RCS vs MWC

Figure 4.5 plots the completion time ratio for the three heuristics as the number of edges varies. Again, the MWC outperforms the other two heuristics. And the reason is about the same as explained above. Random color selection is about 10-18% worse and is not sensitive to different communication densities. For LUC, when the communication density is small, if the number of colors to be chosen from is much larger than necessary (and depends on the initial palette), the performance diverges from the optimal solutions. The performance of LUC improves as the number of edges increases, since the initial palette size tends to be suitable for the needs.

4.6.1.3 Effect of the Range

In this experiment, transmission range of a node is varied from 20 to 50. For a given range, we determine the completion times for the three heuristics for sensor networks containing 20, 40, 60 and 70 nodes. To avoid the impact of palette size, all initial palettes are set to have 100 colors.

Figure 4.6 compares the completion times of RCS and MWC, while Figure 4.7 compares the completion times of LUC and MWC. We find that MWC outperforms the other two heuristics across all ranges and network sizes. Like in the previous

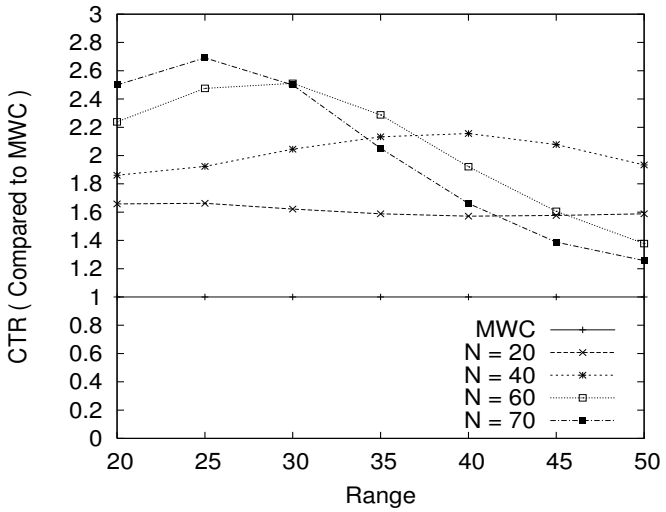


Figure 4.7. Effect of transmission range: LUC vs MWC

experiment, the performance of random color selection is within 10-20% of MWC, while that of LUC is significantly worse. For LUC, for a given N, initially the curve goes up to some peak, and then drops as the range increases. This is because, initially, the number of nodes has more impact on the communication density, and after some point, the *Range* dominates.

Overall, our results show that MWC heuristic yields the best performance among the three heuristics. This is not surprising since the heuristic takes the completion time of each color into account when assigning colors to edges (which in turn, helps minimize the total completion time). Next, we compare the performance of this heuristic to the optimal solution.

4.6.2 MWC versus the Optimal Solution

MWC is a time-based heuristic that has better performance than the other two heuristics in term of minimizing the communication completion time. We construct several examples to understand how far MWC is from the optimal schedule. Since the A^*-based optimal solution has exponential complexity as the problem scales, it is computationally feasible to compare MWC with the optimal solution only for small

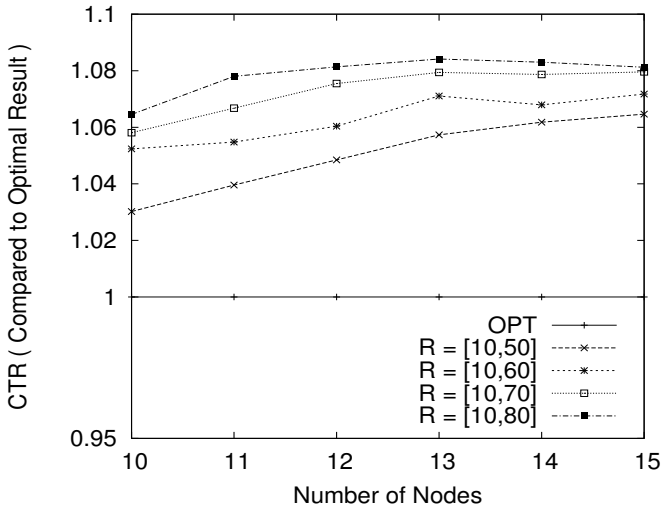


Figure 4.8. Completion Time of MWC to OPT

network sizes. Consequently, we restrict the input to no more than 20 nodes (and 20 edges) in our experiments (this still involves expanding $O(2^{20})$ search nodes in the search tree for one instance, and can take several hours to find a solution on a Pentium-4 workstation).

We conduct two experiments. In the first experiment, each node is assumed to have a different transmission range; we vary the number of nodes and compute the completion times of the schedules produced by MWC and A^*. In the second experiment, each node in the network has an identical transmission range; we vary the number of nodes and compute the completion times. Because of the space limit, we only show the results of the first experiment (the results of the second experiment have similar trend and are detailed in [50]). Figure 4.8 plots the CTR (normalized by the optimal solution); and Figure 4.9 plots the fraction of the cases where MWC yields a solution identical to the optimal solution.

We observe the following behavior:

1. The time to complete transmissions using MWC is within 8.5% of the optimal solution for sensor networks of up to 20 nodes (and 20 edges).

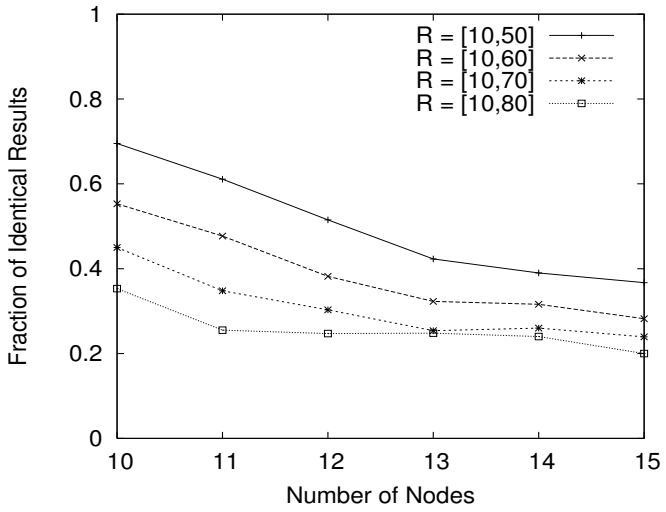


Figure 4.9. Fraction of Identical Results

2. While the solution yielded by MWC is different from the optimal solution in a large fraction of the cases, this sub-optimal schedule is only about at most 6-8.5% worse for a variety of transmission ranges.

3. When the transmission range becomes larger, e.g., in $R = [10, 80]$, or the number of nodes increases, the performance actually becomes stable, i.e., still within 8.5% of the optimal solution, which indicates that MWC is robust even when the complexity increases.

4.6.3 Summary of Experimental Results

In summary, our experiments obtain the following results:

- MWC is the best of the three heuristics across a wide range of system parameters. The better performance is a result of taking the communication duration of each transmission into account when generating a message transmission schedule. And the algorithm attempts to schedule the *similar* transmissions simultaneously as many as possible.

- RCS is a close second in terms of the completion times of its schedules. Further, RCS performs stable as the number of nodes increases or the range parameter changes. This is because the color selection is only one of the major steps with respect to meeting the constraints.

- LUC has worst performance and is very sensitive to the initial number of colors in the palette. If the number of colors is much more than the actual needs, it will in fact diverge from the optimal solution, which is different from the results of the common coloring problems.

- For small networks (up t0 20 nodes), the results of MWC are close to the optimal solution. Also, the performance of MWC can be considered to be stable, even with increasing range and number of nodes, because its performance is within 8.5% of the optimal solution.

4.7 Conclusions

In this chapter, we considered a class of wireless sensor applications that impose timeliness constraints. We assumed that these sensor applications are built using commodity 802.11 wireless networks and focused on the problem of providing qualitatively-better QoS during network transmission of sensor data. We formulate an optimization problem of message transmission over wireless channel. We prove that the problem is NP-Complete. We thus propose three heuristics based on edge coloring that are designed to explicitly avoid network collisions and minimize the completion time to transmit a set of sensor messages. Our simulation results showed that the minimum weight color heuristics yields the best performance across a range of systems parameters and is close to the optimal solution — the time to complete transmissions is within a factor of 8.5 of the optimal solution in the sensor networks tested.

CHAPTER 5

MESSAGE TRANSMISSION WITH TEMPORAL CONSTRAINTS IN MULTI-HOP SENSOR NETWORKS

5.1 Introduction

In the applications of real-time sensor networks, we have investigated the problem to minimize the completion time for sending a set of messages through wireless channel. In sensor applications where sensor data must traverse multiple hops across the network, the communication delay affects both the end-to-end sensor data delivery and the validity of sensor data. One such application is wireless sensor and actor networks (WSAN) [5] where a group of sensors and actors linked by wireless medium to perform distributed sensing and actuation tasks. The physical architecture of the WSAN is given in Figure 5.1. In such a network, sensors gather information about the physical world, while actors take decisions and then perform appropriate actions upon the environment, which allows a user to effectively sense and act at a distance. In order to provide effective sensing and perform right and timely acting, the collected and delivered sensor data must be *valid* at the time of acting.

In most of the real WSAN applications, integrated sensor/actor nodes may replace actor nodes and one example for an integrated sensor/actor node is robot. These robots can be resource-rich big robots or small autonomous robots [102, 97]. However, each of the small robot may not have a sufficient sensing capability to sense the entire area. Hence, in order to initiate more reliable actions, although each of them is not capable of performing complicated tasks, they can coordinate with one another by exploiting their wireless communication capabilities and then make the achievement.

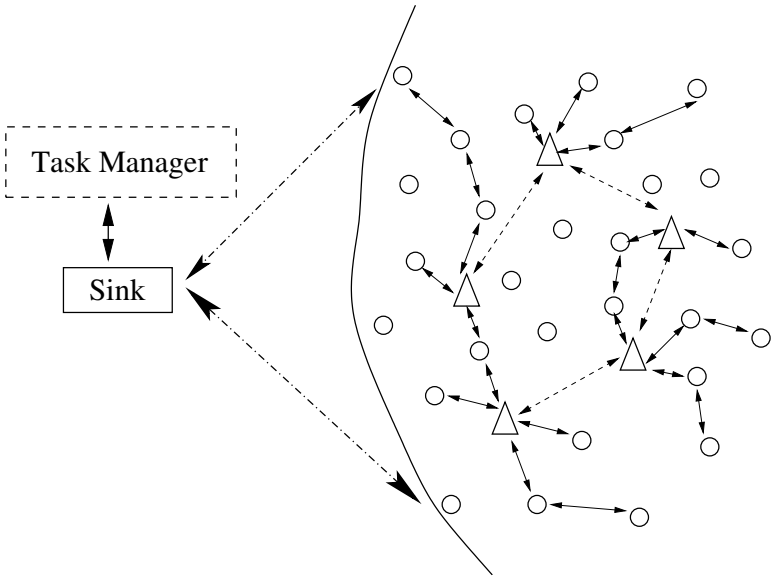


Figure 5.1. WSAN physical architecture.

As shown in Figure 5.1, sensor nodes are scattered in the *sensor/actor* field, and the sink monitors the overall network and communicates with *task manager* node and sensor/actor nodes (robots). Sensor data can be either sent to the related actors to initiate appropriate actions, or route back to the sink which then issue action command to actors. In either way, real-time communication is critical since actions are taken upon the most current information after the sensing occurs and hence the traffic is very delay sensitive.

As discussed before, traditional wireless protocols may cause unpredictable delay, due to collisions, exponential back-offs and blocking. To provide end-to-end timeliness guarantees, a key research challenge is to provide predictable delay and/or prioritization guarantees, while minimizing overhead packets and energy consumption [95]. This chapter is devoted to schedule messages to meet message deadlines over multi-hop wireless transmission. The problem we focus is to provide timeliness guarantees for multi-hop transmissions in a real-time sensor networks. In such applications, each sensor data that needs to be transmitted is associated with a validity and a latest start time of processing at the destination. Message deadlines are derived from the

validity of the accompanying sensor data and the start time of the consuming task at the destination. We show that the problem of meeting message deadlines is NP-hard even for single hop message transmissions. Consequently, we propose heuristics for on-line scheduling of messages with deadline constraints. Our technique (i) schedules messages based on their per-hop timeliness constraints, (ii) carefully exploits spatial reuse of the wireless channel and (iii) explicitly avoids collisions to reduce deadline misses. We evaluate our technique using simulations of various network topologies and examine the impact of various system and sensor parameters on meeting timeliness constraints. Our results show that: the channel-reuse-based algorithm outperforms the CSMA/CA-based algorithm for a wide variety of experimental settings, and especially when (i) the channel utilization (the fraction of bandwidth the wireless channel is busy over a time interval) is high, (ii) the interference range is large, or (iii) the probability of collisions is high.

Spatial channel reuse in ad-hoc networks has been studied from the perspective of improving channel utilization, throughput and fairness [60, 62]. However, to the best of our knowledge, this is the first instance where spatial reuse property has been used to address the problem of meeting real-time constraints for multi-hop message transmissions.

The remainder of this study is organized as follows. In Section 5.2, we present the data validity model, communication model and then formalize the problem. We discuss the algorithm design issues in Section 5.3 to address the intuition behind the heuristics. Then, our novel spatial-reuse scheduling algorithms are presented in Section 5.4. The simulation results are shown in Section 5.5. Finally, we summarize our study in Section 5.6.

5.2 System Model And Problem Formulation

In this section, the conception of data validity and deadline is defined. The communication model used in this chapter is presented. And finally, the problem is formalized.

5.2.1 Data Validity and Transmission Deadlines

In real-time sensor applications, the values of sensor data reflect the current state of the environment. Since the environment may be constantly changing, sensor readings have a temporal interval during which they are valid. For instance, the temperature readings from a few minutes back are no longer valid if the fire in a room burns out of control. We use the term *data validity* to define the time interval for which a data value produced by a sensor is valid.

In this work, we assume that sensors produce data periodically and each must reach the destination before the associated validity expires. Further, since the sensor update is consumed by the task at the destination, the effective deadline of a sensor message is the minimum value of expiration time and the start time of the consuming task. Formally, if a sensor value is produced at time t, and its validity is v time units, the *e*ffective *d*eadline (ed) of sensor message m is:

$$ed(m) = \min(t + v, \delta) \tag{5.1}$$

where δ is the start time of the consuming task that consumes m.

5.2.2 Communication Model

The communication medium used in the sensor application is assumed to be shared by all robots (nodes) in the system. Each node has a wireless network interface with a certain transmission range, depending on the exact wireless interface employed (e.g., 802.11b versus 802.11g). Each node can be a sender or a receiver or both, and no

base-stations exist in such environment. A node should be within the transmission range of a sender to be an eligible receiver, and all communication is assumed to be unicast. If a receiver is within the range of multiple senders, then interference may happen if more than one sender attempts to transmit simultaneously. However, there will be no interference if two receivers are *mutually* outside the other sender's transmission range.

Let $Range(S_i, R_x)$ denote that receiver R_x is within the transmission range of sender S_i, $S_i \rightarrow R_x$ denote a message transmission from S_i to R_x, we have following constraints.

- *Network Interface Constraint:* At any instant, a node can be either a sender or a receiver, but not both.
- *Range Constraint:* A node can receive a message only if it is within the sender's range, i.e., $S_i \rightarrow R_x \Longrightarrow Range(S_i, R_x)$.
- *Interference Constraint:* Two simultaneous transmissions will not interfere if and only if both receivers are mutually outside the other sender's range. That is, $\forall(i \neq j,\ x \neq y),\ (S_i \rightarrow R_x) \wedge (S_j \rightarrow R_y) \wedge \neg Interference(S_i \rightarrow R_x,\ S_j \rightarrow R_y) \Longleftrightarrow \neg Range(S_i, R_y) \wedge \neg Range(S_j, R_x)$.
- *Unicast Constraint:* At one instant, each message can have only one recipient.

5.2.3 Problem Formulation

In this section, we will formalize the problem faced in the design of communication plan and leave out the routing discovery and broadcast phase, since they have been thoroughly studied in networking community.

Each time when a new communication plan is built, the system has:

- A set of sensor messages, each is associated with a per-hop transmission duration, an end-to-end deadline and related routers.

- A set of mutually conflicting relationships among all transmissions.

Out goal is to schedule those wireless transmissions at each hop so that all end-to-end message deadlines are met.

Before presenting the algorithms to solve the multi-hop message transmission scheduling problem, we first consider a simplified version of the problem where all transmissions will traverse a single hop with unit transmission duration and have an overall deadline. We can formalize this problem using a contention graph $G = (V, E)$. In the graph, each vertex represents a message transmission and there exists an edge between two vertices iff the two transmissions (vertices) cannot be scheduled simultaneously (having mutual conflict). We can show the hardness of this problem by reducing the graph K-colorability problem [27] to it; here, K is the number of possible colors such that the connected vertices always have different color. If the contention graph is K-colorable, all transmissions can be completed within K time units.

5.3 Design Considerations

If the system has an overall deadline, the transmission scheduling problem can then transform to the problem of finding the minimum transmission completion time. This is because, if the minimum completion solution cannot make all transmissions meet the deadline, then at least on transmission will miss the deadline wherever we reorder the transmissions.

Although we have proposed and evaluated the effectiveness of the color-based heuristics, it is not sufficient enough to directly adopt to use those methods for multi-hop transmission with end-to-end deadlines. First, the previous problem doesn't have individual deadline for each message transmission, and transmission duration is the only property that needs to be considered. Second, there is only one hop transmissions in the previous version, so avoiding collisions is the most important factor that will affect the completion time.

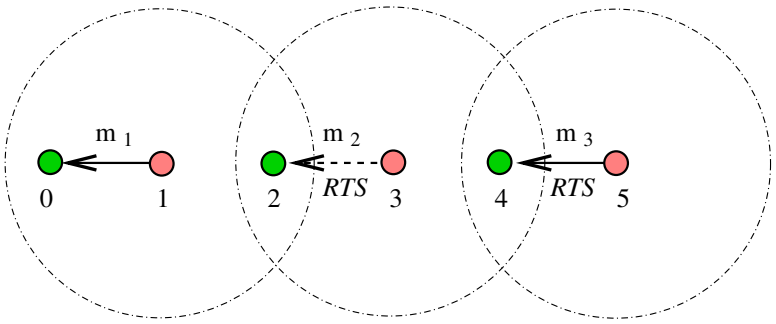


Figure 5.2. Example of false blocking and channel reuse

With regards to the multi-hop message transmission problem, a desirable scheduler should not only be cognizant of message deadlines when making scheduling decisions, but also effectively use the channel bandwidth by avoiding collisions and exploiting spatial reuse.

In this section, we first discuss the potential heuristics based on these two issues, and then propose our algorithms. Before we start our discussion, let us first define a terminology, LST, which is used to determine the urgent status of a message and is derived from the effective deadline.

Suppose a message m needs to traverse h hops from the source to the destination and let m^i denote m is at the i^{th} hop. Further, let $pd(m^i)$ denote the total propagation delay and transmission duration that will be incurred on the remaining hops from hop i to the destination. Then the latest transmission start time (LST) of the message at the i^{th} hop is:

$$l(m^i) = ed(m) - pd(m^i) \tag{5.2}$$

The LST denotes the *latest time* by which the i^{th} hop must start transmitting the message in order for it to reach the destination by its effective deadline. If there are multiple messages queued up at a robot awaiting transmissions, their $LSTs$ can be used to derive a transmission schedule.

5.3.1 Message Contention and Channel Reuse

Consider a simple one hop transmission example depicted in Figure 5.2. The location of each node and transmission range of each sender are shown in the figure. For simplicity, we assume the interference range is equal to the transmission range. As shown in Figure 5.2, three single hop messages m_1, m_2 and m_3 need to be scheduled for transmission. Table 5.1 depicts the arrival time (AT), the transmission duration (TT)(the difference between the instants when the first bit is sent out and the last bit is received by the receiver), and the effective deadline (ED), for each message. The latest start time (LST), calculated using Equation 5.2, is also shown in the table.

Since m_1 arrives at time 0 and is the only message in the system, node 1 transmits a RTS and receives a CTS acknowledgment from node 0. Since node 2 also receives this RTS, it is blocked during the transmission of m_1. When message m_2 arrives at time 1, node 3 sends a RTS to node 2 but does not receive a response. Node 4 receives this RTS message as well and is also blocked. Consequently, when node 5 sends a RTS to node 4 for message m_3, it does not receive a CTS and is unable to transmit m_3. Observe, from Figure 5.2, that m_1 and m_3 do not interfere with one another and it is possible to transmit them simultaneously. This is referred to as *false blocking*. In this example, false blocking causes the messages to be transmitted sequentially in the order m_1, m_2 and m_3, resulting in poor spatial reuse. More importantly, doing so causes m_3 to miss its deadline.

However, if the scheduler were to exploit spatial reuse and transmit m_3 in parallel with m_1, followed by m_2, then all messages are able to meet their deadlines (see Figure 5.3).

5.3.2 Why Simple Channel Reuse is Not Sufficient?

Although the previous example demonstrated the benefits of exploiting spatial channel reuse for meeting deadline constraints, surprisingly, parallelizing transmissions via spatial reuse can sometimes *increase* deadline misses. Consider the same

Message	AT	TT	ED	LST
m_1	0	2	6	4
m_2	1	5	8	3
m_3	1	2	8	6

Table 5.1. Message characteristics

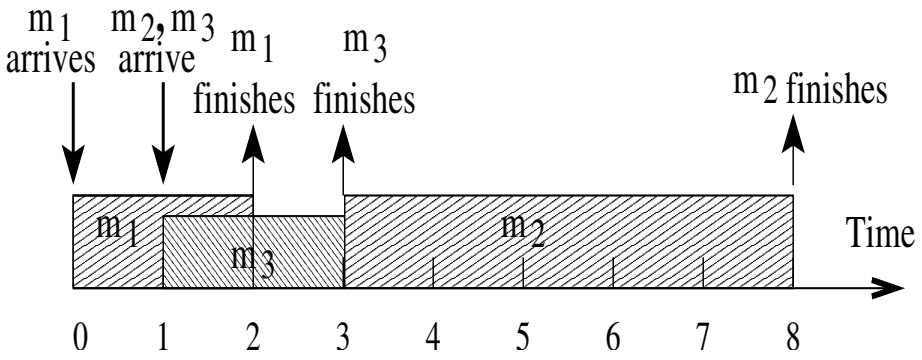


Figure 5.3. Parallel transmissions reduce deadline misses.

scenario depicted in Figure 5.2 but with the parameters listed in Table 5.2. Like before, the sensor message m_1 is transmitted first at time $t = 0$. Since m_1 and m_2 interfere with one another, m_2 is blocked. However, when m_3 arrives at time $t = 1$, it can be transmitted in parallel with m_1 since the two transmissions do not interfere. Assuming this is done, m_2 can not be transmitted until m_3 finishes at time $t = 3$. Since message m_2 requires a transmission duration of 6 time units, m_2 will finish only at $t = 9$, causing it to miss its deadline (see Figure 5.4). In this scenario, the only schedule that satisfies all deadlines is to transmit the messages sequentially: m_1, m_2 and m_3. The example shows that naively maximizing spatial reuse can sometime be detrimental in meeting deadline guarantees. Thus, the message scheduler should consider the potential impact of scheduling a message on future message transmissions.

The above examples illustrate the fundamental problem existing in order to achieve temporal guarantees in wireless sensor networks. As a result, the traffic delay has to be dealt with in both the time domain and the spatial domain.

Message	AT	TT	ED	LST
m_1	0	2	6	4
m_2	1	6	8	2
m_3	1	2	10	8

Table 5.2. Message characteristics

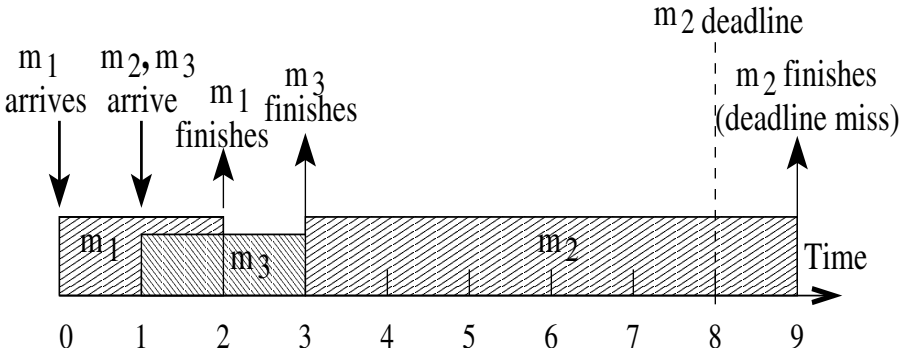


Figure 5.4. Deadline misses caused by parallel transmissions.

5.4 Scheduling Messages With Deadlines

The problem of scheduling parallel messages with deadlines over wireless channel can be shown to be NP-hard. The problem is NP-hard even when messages traverse a single hop, having unit transmission durations and identical effective deadlines. This can be proved by reducing the graph k-colorability problem [27] to it, where a contention graph $G = (V, E)$ is used to represent conflicts among transmissions. In the graph, each vertex represents a transmission, and an edge exists between two vertices iff the two transmissions (vertices) cannot be scheduled simultaneously. If the graph is k-colorable, all transmissions can be completed within k time units.

Due to the NP-hard nature, we must resort to heuristics to schedule multi-hop messages through the network. In this section, we present two such heuristics.

5.4.1 Per-Hop Smallest LST First (PH-SLF)

Per-hop Smallest LST First (PH-SLF) is a distributed scheduler, where each node makes local scheduling decisions independent of other nodes. In this approach, given a set of messages that are queued up at a node, the node schedules the message with the smallest LST for transmission. Observe that the latest start time (LST), as defined in Equation 5.2, is the deadline by which the node must start transmitting the message in order for it to meet its effective (end-to-end) deadline. The underlying MAC protocol is vanilla CSMA/CA. As a result, collisions and the resulting back-offs, and false blocking can not be eliminated in this approach. The advantage of this approach is that it can be used in conjunction with vanilla 802.11 networks, since PH-SLF can be implemented in software in the OS driver.

We use PH-SLF as our baseline algorithm. In the rest of this section, we present an approach that explicitly avoids collisions and maximizes spatial reuse.

5.4.2 Channel Reuse-based Smallest LST First (CR-SLF)

The goal of our Channel Reuse-based Smallest LST First (CR-SLF) approach is to be cognizant of message deadlines at each hop, while avoiding collisions and exploiting spatial reuse. Before presenting the approach, we define some terminology. The following list the key attributes for a message and the associated terminology.

m_x^i : message m_x at the i^{th} hop

$T(m_x^i)$: the transmission of m_x^i

$a(m_x^i)$: the arrival time of m_x at the i^{th} hop

$d(m_x^i)$: the deadline of m_x^i, $d(m_x^i) = ed(m_x)$

$l(m_x^i)$: the latest start time *(LST)* of $T(m_x^i)$, see Eq. 5.2

$s(m_x^i)$: the transmission start time of $T(m_x^i)$

$f(m_x^i)$: finish time of m_x^i—the time m_x reaches the next hop

$e(m_x)$: the execution time for m_x (same for all per-hop transmissions of m_x)

Since sensor updates are generated periodically, the arrival time of a message at the source (first hop) is the time at which the sensor data is produced. The arrival time at an intermediate node is the time the last bit of the message arrives at that hop enroute to the destination. Observe that the arrival time at an intermediate node depends on when the message is scheduled for transmission at the previous hop. The start time is the time when message is scheduled for transmission and the finish time is the time when the message is completely received by the next hop node (and the channel becomes idle again). The execution time is the time for which the channel is busy and is the sum of the transmission delay and the propagation delay (the difference between the instants when the first bit is sent out and the last bit is received by the next hop). Observe that,

$$a(m_x^{i+1}) = f(m_x^i) = s(m_x^i) + e(m_x^i)$$

In the rest of this paper, transmission refers to a per-hop transmission unless specified otherwise. With this background, we present the intuition behind our approach, followed by the details.

5.4.2.1 Overview

Given a set of nodes, their locations and transmission ranges as well as a set of messages queued up at these nodes, their destinations, effective deadlines, and the associated routes, our scheduler derives a schedule to meet these deadlines. The scheduler exploits the following characteristics in deriving this schedule:

- It maximizes spatial reuse by scheduling non-interfering message transmissions in parallel.

- It considers message transmissions in the order of their LSTs. The LST is the "local" deadline of a transmission at a hop, since it is the latest time by which the message must be scheduled to meet its end-to-end effective deadline.

The basic idea is to partition the set of message transmissions into *disjoint sets* such that transmissions within each set do not interfere with one another and can be executed in parallel. These sets are ordered sequentially, and all transmissions within a set must finish before transmissions in the next set can begin.

To construct these sets, the scheduler considers the transmissions in order of their LSTs. At each step, the transmission with the smallest LST is chosen and the scheduler checks if it is feasible to assign this transmission to an existing set (the feasibility will be discussed in the next section). If no existing set is feasible, a new set is created with that message transmission so long as the deadline is met. Once a message is scheduled at hop i, it can be considered for scheduling at hop $i+1$. Observe that a message needs to be transmitted hop by hop, since the arrival time at the next hop is not known until it is scheduled for transmission at the previous hop. The above process continues until all queued up messages are scheduled along all hops from their sources to their destinations (i.e., the message transmission on each hop is assigned to a feasible set). The constructed sets define the transmission schedule for these multi-hop messages.

5.4.2.2 Details of the Algorithm

Initially, the schedule $\mathcal{S}$ is empty: $\mathcal{S} = \phi$. The goal is to construct a set of sets $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ where elements are disjoint and message transmissions in each set are non-interfering. We define a *start time* and a *finish time* for each set S_j. The *start time* of set S_j, $s(S_j)$, denotes the instant where all its transmissions can start using the channel; the *finish time* of S_j, $f(S_j)$, denotes the instant where all transmissions have reached the their respective receivers. In general, $s(S_{j+1}) = f(S_j)$;

that is, transmissions in a set can start to transmit when those in the prior set have finished.

The algorithm proceeds in the following steps.

Step 1 : *Select a transmission to schedule.* From the list of yet to be scheduled message transmissions, the scheduler chooses the one with the *smallest* LST. This enables the scheduler to consider the most urgent transmission first. Other strategies are possible, such as choosing transmissions based on the earliest data validity first or earliest start time first, although we do not consider such alternatives in this paper.

Step 2: *Assign this message transmission to a set.* Suppose that n sets have been constructed in the partial schedule thus far: $S_1, S_2, ..., S_n$, and $T(m_x^i)$ has been selected. The scheduler attempts to assign this transmission to the *first feasible set* in the set list. If no existing set is feasible (or the schedule is empty), then a new set S_{n+1} is added to the list and $T(m_x^i)$ is inserted into this set so long as the deadline constraint is not violated.

A set S_j to be feasible for a message transmission $T(m_x^i)$ iff the following conditions are satisfied.

1. The finish time of the set $f(S_j)$ is later than the arrival time of the message $a(m_x^i)$. This indicates that a message transmission should be added to a set only if there is some temporal overlap with existing transmissions so that parallelism can be exploited.

2. The finish time of m_x^i is no later than its effective deadline, i.e., $f(m_x^i) = max(s(S_j), a(m_x^i)) + e(m_x^i) \leq d(m_x^i)$.

3. $T(m_x^i)$ does not interfere with any existing message transmissions in S_j.

4. The insertion of the transmission $T(m_x^i)$ into S_j does not violate deadlines of messages in subsequent sets $S_k, j < k \leq n$.

The first three conditions are easy to understand, so we elaborate on the fourth condition. If the current message m_x^i happens to be the longest message or its transmission finishes last in S_j, then the duration for which S_j occupies the wireless channel is increased. As a result, the transmission start times of messages in subsequent sets will need to be pushed forward. Since a later start time may violate their deadlines, the scheduler needs to verify that inserting this transmission of m_x^i in S_j does not impact the deadline meeting of subsequent message transmissions. To do so, we first need to compute the new finish time of S_j. If inserted in S_j, the start time of m_x^i is $s(m_x^i) = \max(s(S_j), a(m_x^i))$. Then, its finish time is $f(m_x^i) = s(m_x^i) + e(m_x^i)$. The new finish time of the set is the maximum finish time of all messages in the set: $f_{new}(S_j) = max_{\forall x}(f(m_x))$.

Next, we can compute the amount by which all subsequent transmissions are pushed forward. This is done by computing the new start time of each set S_k, $j < k \leq n$, which is simply the finish time of the previous set: $s(S_k)) = f(S_{k-1})$. The new start time of each message transmission $T(m_y)$ in the set is recomputed as $s(m_y) = \max(s(S_k), a(m_y))$. Then, the finish time is $f(m_y) = s(m_y) + e(m_y)$. The new finish time of the set is the maximum finish time of all messages in the set: $f_{new}(S_k) = max_{\forall y}(f(m_y))$, $T(m_y) \in S_k$. Given the new start and finish times of the affected messages, the scheduler needs to verify that the finish time of each message is not later than its deadline: $f(m_y) \leq d(m_y)$. If no deadlines are violated, then it is possible to insert the current selected message transmission, $T(m_x^i)$, in S_j.

The scheduler searches for the first set in the list $S_1, S_2, ..S_n$ that is feasible and inserts the selected transmission into that set. If no existing set is feasible and inserting the transmission into a new set S_{n+1} violates its deadline, then the algorithm can not meet the deadline for this message. In this scenario, the message is removed from consideration and all scheduled transmissions for hops 1 to $i-1$ are removed from

the corresponding sets. And then the start/finish times of those sets are adjusted accordingly.

Step 3 : *Update the finish time of the feasible set and insert a new transmission for the next hop.* If a feasible set S_j is found, then the transmission $T(m_x^i)$ is inserted into the set, and the new finish time is updated as discussed above. The transmission $T(m_x^i)$ is deleted from the list of yet to be scheduled transmissions, and the next hop transmission $T(m_x^{i+1})$ is inserted into the list, assuming that hop i is not already the destination for the message. All these steps are repeated until the list of unscheduled transmissions becomes empty.

5.4.2.3 An Example

In this section, we will use a simple one-hop transmission example to illustrate the basic procedure of the algorithm. Let us consider the example shown in Figure 5.2 with message characteristics in Table 5.1.

Because $T(m_1)$ arrives first, and at time 0, it is selected to be scheduled. Initially, the schedule set is empty, a new set S_1 is created for the transmission of m_1, and we have: $s(S_1) = 0,\ f(S_1) = f(m_1) = 2$. Then, $T(m_2)$ is considered since it has the smallest LST. Because $T(m_2)$ interferes with $T(m_1)$, set S_1 is not feasible and a new set is needed for $T(m_2)$. Thus, we will have two sets: $S_1 = \{T(m_1)\},\ S_2 = \{T(m_2)\}$, and $s(S_2) = f(S_1) = 2, f(S_2) = max(s(S_2), a(m_2)) + e(m_2) = 2 + 5 = 7$.

Finally, let us consider the last transmission of m_3. First, consider the set S_1 with the four conditions:

1. The finish time of S_1 is 2, which is later than $a(m_3)$;
2. $f(m_3) = max(s(S_1), a(m_3)) + e(m_3) = 1 + 2 = 3 < d(m_3)$;
3. Transmission of m_3 will not interfere with the existing transmission of m_1 in the set;

4. Since $f(m_3) = 3 > f(S_1)$, we have the new possible finish time for S_1: $f_{new}(S_1) = f(m_3) = 3$; so the new start time for S_2 is $s_{new}(S_2) = f_{new}(S_1) = 3$ and we have $f_{new}(m_2) = 3 + 5 = 8 \leq d(m_2) = 8$.

Therefore, S_1 is feasible for $T(m_3)$, transmission of m_3 is assigned to S_1. The final schedule is : $S_1 = \{T(m_1), T(m_3)\}$, $S_2 = \{T(m_2)\}$, which indicates that m_1 and m_3 are transmitted in parallel, followed by the transmission of m_2.

5.5 Evaluation

We have designed an event-driven simulator to simulate a team of robots that exchange multi-hop sensor messages and move in the environment. We compare the performance of our proposed algorithm CR-SLF with the PH-SLF scheduling using our simulator. We evaluate the impact of sensor period, deadline and message size based on specific transmission topology. We also investigate how the different interference ranges may affect the properties of the algorithms. Last, we study the impact of node mobility; observe that the movement of a team of robots is highly correlated, and consequently, mobility models such as the random waypoint are not suitable for our scenario.

The metrics used to measure performance is the *Deadline Miss Ratio* which is defined to be:

$$\frac{number\ of\ unsuccessful\ end{-}to{-}end\ transmissions}{total\ end{-}to{-}end\ transmissions}$$

A successful message transmission is one where the message is transmitted from the source to the destination before the effective deadline. Note here, for the CSMA/CA based algorithm PH-SLF, a message that cannot meet the deadline will still be transmitted from the source through the network; but for the proposed algorithm CR-SLF, once a message is found that it cannot meet the deadline, it will not be scheduled to

transmit.The deadline-missing in CR-SLF happens for two reasons: one is the design problem which means that those messages cannot meet the deadlines using any algorithms, the other reason is since the problem is NP-hard, the heuristic algorithm may not be able to find the optimal solution.

5.5.1 Experimental Settings

The wireless card on each robot and its radio parameter are based on the existing commercial product (e.g., Lucent Wavelan card) with a 2 Mbps data rate and a transmission range of 250 meters. Unless specified otherwise, the interference range is set to be equal to the transmission range, and each pair of nodes (robots) are separated by the distance of 200 meters which is likely to yield close to the maximum capacity possible [53].

Table 5.3 gives the default parameter settings used in our simulations. The effective deadlines are specified in relative terms, relative to the start times of the respective transmissions at the source.

Message Size	Sensor Period	Relative Deadline
512 byte	10ms	50ms

Table 5.3. Default Settings

5.5.2 Impact of the Sensor Period and Deadline

In this experiment, the robotic group is based on a chain topology with 8 robots, as illustrated in Figure 5.5. Two sensor messages periodically travel through intermediate nodes from two end robots to the opposite end on both directions. We use this scenario to demonstrate the effectiveness of our proposed algorithm, CR-SLF, for different sensor periods and deadlines, and compare it to PH-SLF scheduling.

The impact of varying the sensor periods is shown in Figure 5.6. In this simulation, the period to generate a new message is depicted on the x-axis. We can see that CR-

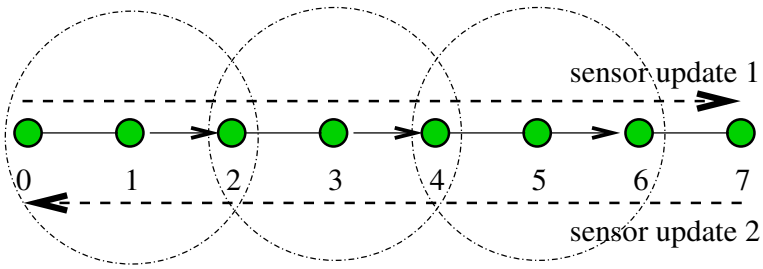


Figure 5.5. Scenario 1: transmissions over a robot chain

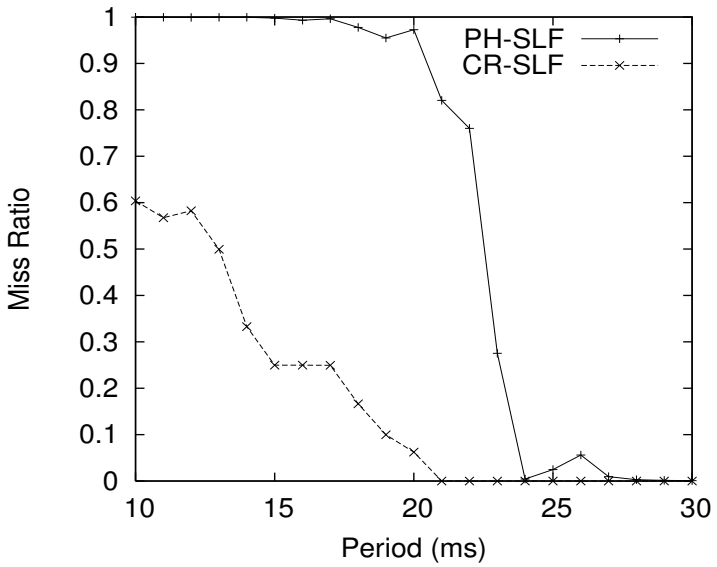


Figure 5.6. Impact of sensor period

SLF has fewer deadline misses than the PH-SLF scheduling, especially when the period is between 10ms and 20ms and where collisions are likely unless the scheduler is careful. Beyond 20ms, since the transmission duration per hop is approximately around 2ms. When the next update is generated, the previous one has reached the destination (without considering the update from the opposite direction). Hence, the probability of collisions is very low. Therefore, both algorithms are able to schedule messages with few collisions or deadline misses.

The impact of message deadlines is shown in Figure 5.7. Here, the sensor period is set to 19ms. As the deadline of each sensor update goes from 50ms to 150ms, we

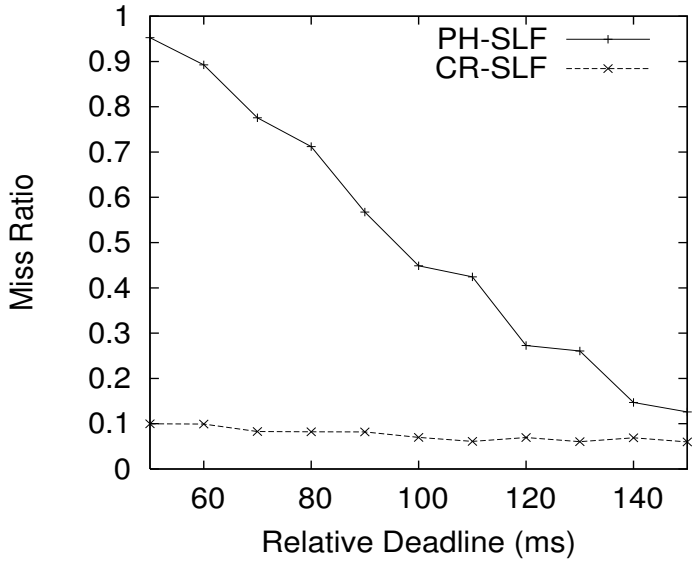


Figure 5.7. Impact of message deadline

have 40% improvement for CR-SLF and 87% improvement for PH-SLF. The reasons we have less improvement for CR-SLF are: 1) although the deadlines are different, the utilization and the probability of collisions are the same; CR-SLF has already explicitly taken these issues into account, and the deadline will only affect the order of transmissions considered, 2) for the PH-SLF scheduling, the larger deadline actually gives it more chances to back off and retransmit, increasing the number of eventual successful transmissions.

5.5.3 Impact of Distance

In the previous section, we evaluated the impact of the sensor period and message deadline with a node distance of 200 meters. In this section, we study how varying the distance between robots affects the results.

We assume the chain topology and assume that two sensor updates are periodically generated from two ends. The period is set to 19ms. We vary the distance between each pair of robots from 50 meters to 200 meters and plot the resulting miss ratio in Figure 5.8. We observe that CR-SLF performs much better than PH-SLF for all

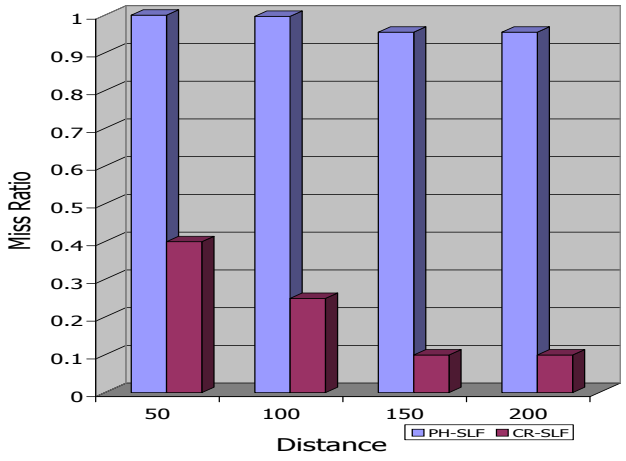


Figure 5.8. Impact of distance

settings. Specifically, when the distance between nodes is small, i.e., 50 meters, the interference due to one transmission from one node will impact at least 5 other nodes (transmissions on node 2 to node 5 will affect transmissions on all other nodes). Even with this high interference, our proposed algorithm, CR-SLF enables more than 60% of the messages to meet their deadlines, while no message meets its deadline when a CSMA/CA-based algorithm is used.

5.5.4 Impact of Message Size

In this example, we study the effect of varying the message size. We use a cross topology as shown in Figure 5.9. There are four periodic sensor updates traveling through the network. The period of each sensor is 30ms and the relative deadline is 100ms.

We vary the message size from 256 bytes to 1024 bytes for each respective transmission and measure the miss ratio. The results are shown in Table 5.4. As we know, the packet size reflects the transmission duration. Since small packet sizes have a smaller probability of collisions, both algorithms work very well. However, as the

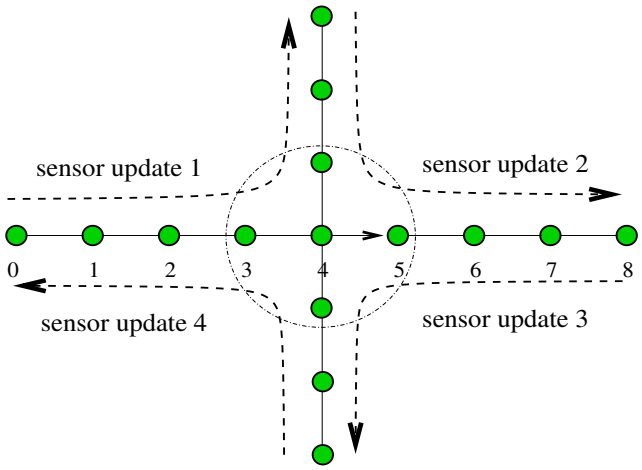


Figure 5.9. Scenario 2: transmissions over a cross topology

packet size increases, the per hop transmission duration increases. This increases the duration for which the channel is busy, since many transmissions may be traversing intermediate hops at the same time. Consequently, collisions cause exponential back-offs and increase the queue delay at each hop. As a result, the performance of PH-SLF scheduling degrades a lot. On the other hand, since CR-SLF explicitly avoids collisions, it is able to meet the deadlines for larger message sizes, so long as the messages are feasible.

Size (byte)	PH-SLF MissRatio	CR-SLF MissRatio
256	0	0
512	0.725	0.124
1024	0.996	0.394

Table 5.4. Impact of message size

5.5.5 Impact of Interference Range

Note that one node can interfere with message reception at another node even when they are too far apart for successful transmission [53]. In this simulation, we use a tree topology, shown in Figure 5.10, to study the impact of the interference range.

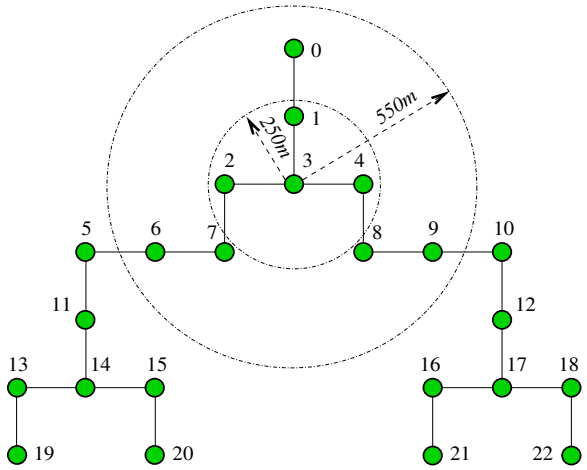


Figure 5.10. Scenario 3: tree topology with various interference range

Observe that in a tree, the interference increases for nodes closer with the root, due to the larger node density. We use the parameters in [53] to vary the interference range. We consider two different interference ranges, namely 250 meters and 550 meters, and two different sensor periods, namely 20ms and 25ms. Sensor messages from two leaf nodes, node 19 and 22, are assumed to be sent out periodically to the root (node 0). The third sensor update is sent from a leaf, node 20, to another leaf on the other side, node 21. The relative deadline for each message update is set to 200ms.

We measure the miss ratio for all four combinations of the interference ranges and the sensor periods. Figure 5.11 shows the results and we have several observations. First, CR-SLF performs better than the PH-SLF scheduling for all different parameters. Second, for a fixed sensor period, when the interference range increases, the miss ratio increases for both algorithms, but the difference between the two algorithms gets smaller. The reason is that if the interference range is small, the probability of collisions is dominated by the sensor period. However, when the period is larger, the occurrence of collisions is dominated by both of the sensor period and the interference range.

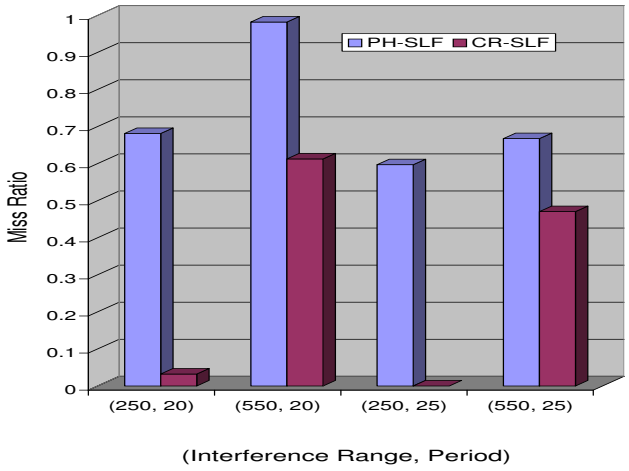


Figure 5.11. Impact of interference range

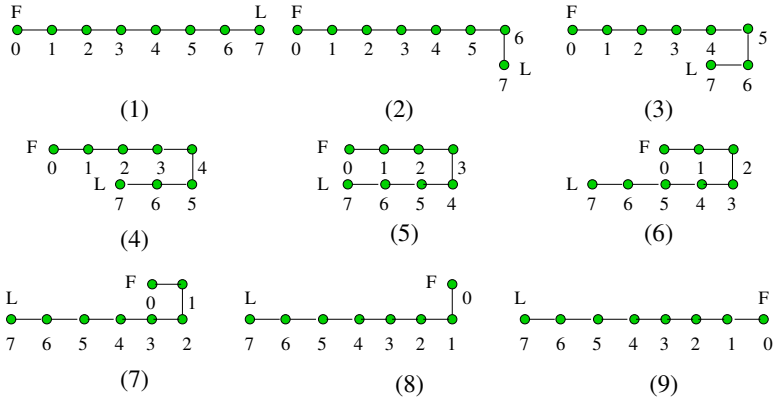


Figure 5.12. Scenario 4: movement of a robot team

5.5.6 Impact of Routing and Node Mobility

In this section, we consider the impact of routing and node mobility. Figure 5.12 depicts the sequence of the moves in a robot chain, where the leader (node 7) leads the team to make a U-turn. Two sensor messages traverse the chain, one from node 0 to 7 and vice versa, with sensor period of 15ms. Two scenarios are considered here: (i) overlay routing, where each update traverses through all intermediate nodes of the overlay chain, and (ii) shortest path routing, where transmissions use the shortest path from the source to destination.

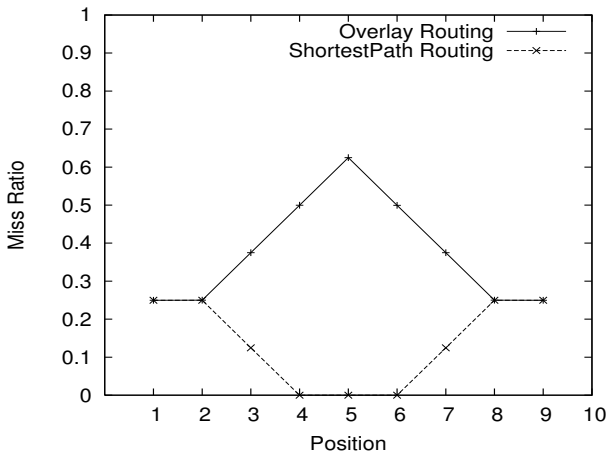


Figure 5.13. Impact of mobility

The results of CR-SLF are shown in Figure 5.13. For the overlay routing, since the update has to travel through every intermediate node, as the team moves from position (1) to (5), the interference increases due to the increase of the node density. Hence, the miss ratio increases. But when the team moves from (5) to (9), the team becomes a chain again, the miss ratio decreases since the interference decreases.

For the shortest path routing, the algorithm will always find the shortest path for message transmissions. For instance, in position (5), messages are directly sent out from node 0 to 7 without passing through any other nodes. So when the team moves from position (1) to (5), the number of intermediate routers involved actually decreases. This causes the miss ratio to decrease since the probability of collisions also decreases.

5.6 Conclusions

In this chapter, we study the problem to schedule messages with temporal constraints. Consider the application where a team of robots equipped with sensors that collaborate with one another to achieve a common goal. Sensors on robots produce periodic updates that must be transmitted to other robots and processed in real-time to enable such collaboration. Since the robots communicate with one another over

an ad-hoc wireless network, we address the problem of providing timeliness guarantees for multi-hop message transmissions in such a network. We derive the effective deadline and the latest start time for per-hop message transmissions from the validity intervals of the sensor data and the constraints imposed by the consuming task at the destination. Our technique schedules messages by carefully exploiting spatial channel reuse for each per-hop transmission to avoid MAC layer collisions, so that deadline misses are minimized. Extensive simulations show the effectiveness of our channel reuse-based SLF (smallest latest-start-time first) technique when compared to a simple per-hop SLF technique, especially at moderate to high channel utilization or when the probability of collisions is high. The major reason is as follows:

- CR-SLF explicitly avoids collisions while PH-SLF incurs exponential back-off at each collision.
- CR-SLF doesn't inject infeasible packets into the network once it finds out the packet cannot meet the deadline; hence, the infeasible packet will not affect other feasible packets.

CHAPTER 6

SUMMARY AND FUTURE RESEARCH

In this chapter, I summarize the contributions of the thesis and then discuss future research directions.

6.1 Summary and Contributions

In this thesis, I identify several fundamental problems of the design and development of distributed real-time sensor systems. Two key features of such applications make the problems challenging. First, the applications exhibit highly dynamic resource sharing requirement with many computation and communication variations. Second, they are built through wireless infrastructure, and transmitting sensor data over traditional wireless networks may suffer from unbounded delay. My research goal is to investigate the inherent characteristics of such resource-constrained environments, and to identify and provide the solutions to fundamental problems that arise in the design of such distributed embedded systems, so that all physical and temporal constraints are guaranteed. I will briefly describe my contributions in the following sections, and then I will discuss my future research directions.

6.1.1 Summary

My dissertation focuses on resource management in a resource-constrained distributed environment, and investigates fundamental problems in the analysis and design of distributed real-time embedded systems. Typically, these resource-constrained systems operate collaboratively within a dynamic information exchange environment,

adapting to changing conditions in a timely manner to meet interactive real-time needs. Some challenges for delivering sensor data with temporal constraints in such resource-constrained systems are: i) effective use of limited resources to improve system feasibility and scalability; ii) meeting the temporal requirements that are inherent in such severely resource-limited distributed systems; and iii) developing communication mechanisms that can be used to optimize the system performance.

In my dissertation, I have provided answers to these questions by solving realistic and complex distributed resource management problems. One of my contributions is the development of a resource allocation algorithm with the purpose of improving system feasibility. The algorithm not only minimizes communication cost, but also minimizes and balances the workload of each processor. We have also applied the algorithm to a real-world robotic example to achieve efficient resource allocation and scheduling.

Collaborative sensory systems must deliver, process and react to the sensor data subject to temporal constraints. The second contribution of my work is the study of temporal issues that arise in the design of distributed real-time systems. In particular, I have characterized the wireless spatial/temporal properties when transmitting sensor data with end-to-end and per-hop deadline constraints. I have formulated the problem of minimizing total transmission time and providing timeliness guarantees for multi-hop message transmission in wireless sensor networks. I have shown that the optimization problems are NP-hard. Since the complexity results apply only to the worst case scenario, I have thus developed a good heuristic to achieve near-optimal solutions. Specifically, I have developed novel spatial channel reuse techniques to i) optimize the completion time of transmitting a set of sensor messages, ii) provide timeliness guarantees for multi-hop sensor data transmission.

6.1.2 Contributions

The contributions of this dissertation can be summarized as follow.

- *Dynamic Resource Allocation for Mobile Robotics* Distributed control systems are built using sets of functionally equivalent controllers that are designed in the form of coordinated, adaptive control schemes on multi-robot systems. These controllers are distinguished by their use of resources including communication, processor or sensors. Although the control strategies are logically equivalent, some of them may not be feasible, given limited resource and time availability. Thus, dynamic feasibility checking becomes important as the coordination between robots and the tasks that need to be performed evolve with time.

  However, given limited resource and time availability, finding an optimal feasible resource allocation solution is known to be NP-hard. I have developed two simple but efficient on-line algorithms for allocating control tasks to distributed processing entities (robots). For the purpose of schedulability, both communication cost and utilization bound are considered. By minimizing the inter-robot communication overhead and workload of each processor, we can then improve system feasibility. Extensive experimental results have shown the effectiveness of our approaches even under extreme resource-constrained environments. Since schedulability is an overlooked part of large multi-robot system design, we have also demonstrated the application of our approach to real-world large robotic teams. By analyzing the structure of the processes and their tasks' timeliness constraints, we were able to calculate the upper bound on the size of feasible teams.

- *Communication Scheduling in Real-Time Sensor Networks* To ensure timely mobility for robotic teams, in addition to the problem of task allocation and processing that has been addressed in the above work, another critical problem

is the time-critical message transmission over a wireless channel to guarantee the freshness of sensor data. However, message transmission over traditional wireless networks, such as CSMA/CA-based 802.11, may suffer from unbounded delay due to collisions, backoff, and blocking.

I have studied the problem of providing qualitatively-better QoS to transmit sensor data, using commodity 802.11 wireless networks. The first problem I have solved is to minimize the completion time of transmitting a set of sensor messages. In particular, I have conducted a complexity analysis to show that this problem is NP-complete. I have exploited the specific characteristics of the wireless channel and devised novel spatial channel reuse algorithms to optimize the transmission time. By performing non-interfering transmissions in parallel, the transmission collisions are explicitly avoided and the total time for transmitting a set of messages is minimized. Further, I have proposed an A^*-based optimal algorithm to enable comparisons. The evaluation results demonstrate that our approach results in good performance: the time to complete transmissions is within a factor of 8.5 of the optimal solution, and it outperforms the random method. More importantly, it is robust to the increases in communication density.

In sensor applications where messages must traverse multiple hops across the network, the communication delay affects both the end-to-end delivery and the validity of sensor data. Another problem I have studied is scheduling messages to meet deadlines over multi-hop wireless transmission. I have shown the problem to be NP-hard even for single hop message transmission. Consequently, I have designed a technique to derive the effective deadlines and the latest start times for per-hop message transmissions from the validity intervals of the sensor data and the temporal constraints imposed by the data-consuming tasks. In order to solve the problem in an efficient manner, I have investigated the po-

tential impact of spatial reuse with the purpose of decreasing deadline misses. Then, I proposed an algorithm that carefully schedules the messages according to their per-hop temporal constraints and the wireless spatial properties. Extensive simulations have shown the effectiveness of our channel-reuse-based SLF (Smallest Latest-start-time First) technique when compared to a simple per-hop SLF approach, especially at moderate to high channel utilization, a large interference range, or a high collision rate.

6.2 Future Work

Allocating and scheduling of real-time tasks in a distributed environment is a difficult problem. In addition to task-level constraints, e.g., periods and deadlines, such systems also have system-level constraints, e.g., precedence and communication. The algorithms discussed in this paper provide a framework for allocating and scheduling periodic tasks with precedence and communication constraints in a distributed dynamic environment, such as a mobile robot system. The required properties of sets of distributed, coordinated tasks, such as in the LOS chain of a multi-robot system, are well captured by our methods. In the scheduling part of the algorithm, various temporal characteristics of tasks are taken into account at each search step. Our algorithm was applied to a real world example from mobile robotics to achieve a simple but efficient allocation and scheduling scheme for a team of robots. We believe that this approach can enable system developers to design a predictable distributed embedded system, even if there are a variety of temporal and resource constraints.

Now we discuss some of the possible extensions to the algorithm. First, if the system design does not have pre-allocated tasks, the heuristic is still applicable. In this case, the initial threshold is 0. After selecting the first pair of communicating tasks and randomly assigning them to a processor, the algorithm can continue to work on remaining tasks as discussed in the original algorithms.

Second, the algorithm can be tailored to apply to heterogeneous systems. If processors are not identical, the execution time of a task could be different if it runs on different sites. To apply our approach in such an environment, first, we can take the worst case communication cost ratio, which is calculated by the slowest processors for each pair of communicating tasks, and then we can use these values as estimates to choose the task to be considered next. Second, when we select the processor, if the task can be assigned to the processor that the producer is on, then we are done; otherwise, we need to consider the utilization and the speed of a processor the same time, e.g., compare the utilization from the fastest processors to see which processor will have the least utilization after loading the task, and choose the one with the minimum value. After assigning each task, the threshold will change in a way similar to the original algorithm.

In the work of providing qualitatively-better QoS during network transmission of sensor data, we proposed three heuristics based on edge coloring that are designed to explicitly avoid network collisions and minimize the completion time to transmit a set of sensor messages.

As part of future work, we plan to evaluate the effectiveness of our techniques by implementing them into a sensor testbed. To do so, we plan to design a scheduler above the MAC layer to prioritize the packet transmissions based on the transmission schedule. We also plan to examine the impact of message deadlines and will extend our techniques to multi-hop sensor networks.

In this dissertation, we presented two algorithms for scheduling message transmissions with validity and processing constraints in multi-hop robotic sensor networks. Our results show that *CR-SLF* outperforms *PH-SLF* since it not only takes the deadline into account, but also attempts to schedule parallel per-hop transmissions as many as possible, so that the end-to-end effective deadlines can be met. We now discuss some other issues that need further study.

In this work, we assume a leader node that is responsible for routing, scheduling and path planning decisions. This is a realistic model for small robotic teams. When team becomes large, the whole group can be split to multiple smaller groups. A hierarchical communication infrastructure can then used, and our algorithm can be used for communications at each level.

In robotic applications, the speed of a robot is usually much slower than message transmissions. Since robots can transmit messages while moving, we must ensure that these transmissions will succeed, which means that the overlay network topology must not change even though two nodes may move in opposite directions. Our current work assumes that the leader robot will recompute new routes and a new schedule before a topology change and while the nodes are moving as per the current plan. However, due to the issues of speed, signal strength and routing, this is a problem that needs further study.

We made a few assumptions in our paper that would need further studies in future work. First, the impact of radio irregularity [111] needs to be embedded in our performance evaluation studies. Second, packet loss rates typically increase with distance [2], resulting in retransmissions that is not included in our current evaluation metrics. Although we did not specifically consider transmission losses, we believe that our results will hold even in perturbed settings. We will incorporate these issues in future work.

BIBLIOGRAPHY

[1] Abdelzaher, Tarek F., and Shin, Kang G. Period-based load partitioning and assignment for large real-time applications. *IEEE Transactions on Computers 49*, 1 (January 2000), 81–87.

[2] Aguayo, D., Bicket, J., Biswas, S., J., G., and Morris, R. Link-level measurements from an 802.11b mesh network. *ACM SIGCOMM Computer Communication Review 34*, 4 (October 2004), 121–132.

[3] Ahn, G-S, Campbell, A. T., Veres, A., and Sun, L-H. Supporting service differentiation for real-time and best-effort traffic in stateless wireless ad hoc networks (swan). *IEEE Transactions on Mobile Computing 1*, 3 (July 2002), 192–207.

[4] Ahn, G.S., Campbell, A.T., Veres, A., and Sun, L.H. Swan: Service differentiation in stateless wireless ad hoc networks. In *Proceedings of IEEE INFOCOM* (June 2002).

[5] Akyildiz, I. F., and Kasimoglu, I. H. Wireless sensor and actor networks: Research challenges. *Ad Hoc Networks Journal (Elsevier) 2*, 4 (October 2004), 351–367.

[6] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. *Computer Networks 38* (2002), 393–422.

[7] A.Mainwaring, Szewczyk, R., Culler, D., and Anderson, J. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)* (2002), IEEE.

[8] Andersson, B., Baruah, S., and Jonsson, J. Static-priority scheduling on multiprocessors. In *IEEE real-time systems symposium* (December 2001), pp. 193–202.

[9] Arkin, Ronald C. Towards the unification of navigational planning and reactive control. In *Proc. AAAI Spring Symp.,AAAI, Menlo Park, Calif.* (1989).

[10] Baker, T. P. Multiprocessor edf and deadline monotonic schedulability analysis. In *Proceedings of the 24th IEEE real-time systems symposium* (2003), pp. 120–129.

[11] Baruah, S. Scheduling periodic tasks on uniform multiprocessors. *Information Processing Letters 80*, 2 (2001), 97–104.

[12] Baruah, S., and Goossens, Joel. Rate-monotonic scheduling on uniform multiprocessors. *IEEE Trans. Computers 52*, 7 (July 2003), 966–970.

[13] Baruah, S. K., Cohen, N. K., Plaxton, C. G., and Varvel, D. A. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica 15*, 2 (June 1996), 600–625.

[14] Bharghavan, Vaduvur. Performance evaluation of algorithms for wireless medium access. In *IEEE Performance and Dependability Symposium (IPDS)* (July 1998), IEEE, pp. 86–95.

[15] Bulusu, N., Estrin, D., Girod, L., and Heidemann, J. Scalable coordination for wireless sensor networks: Self-configuring localization systems. In *Proceedings of the 6th IEEE International Symposium on Communication Theory and Application* (July 2001).

[16] Caccamo, Marco, Zhang, Lynn Y., Sha, Lui, and Buttazzo, Giorgio. An implicit prioritized access protocol for wireless sensor networks. In *Proceedings of the IEEE Real-Time System Symposium (RTSS)* (December 2002), IEEE.

[17] Choi, M., and Krishna, C. M. An adaptive algorithm to ensure differential service in a token-ring network. *IEEE Transactions on Computers 39*, 1 (Jan. 1990), 19–33.

[18] Coffman, E.G., Jr., M. R. Garey, and Johnson, D. S. Approximation algorithms for bin packing: A survey. *Approximation Algorithms for NP-Hard Problems* (1996), 46–93.

[19] Culler, D., Estrin, D., and Srivastava, M. Overview of sensor networks. *IEEE Computer, Special Issue on Sensor Networks* (August 2004).

[20] Dechter, Rina, and Pearl, Judea. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM (JACM) 32*, 3 (July 1985), 505–536.

[21] Estrin, D., Culler, D., Pister, K., and Sukhatme, G. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing 1*, 1 (2002), 59–69.

[22] Estrin, Deborah, Girod, Lewis, Pottie, Greg, and Srivastava, Mani. Instrumenting the world with wireless sensor networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001), Salt Lake City, Utah* (May 2001).

[23] Estrin, Deborah, Govindan, Ramesh, Heidemann, John, and Kumar, Satish. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99), Seattle, Washington.* (August 1999).

[24] Funk, S., Goossens, J., and Baruah, S. On-line scheduling on uniform multiprocessors. In *IEEE real-time systems symposium* (December 2001), pp. 183–192.

[25] Gamal, A. E., Mammen, J., Prabhakar, B., and Shah, D. Throughput-delay trade-off in wireless networks. In *IEEE INFOCOM04* (Mar 2004).

[26] Garey, M. R., and Johnson, D. S. Strong np-completeness results: Motivation, examples, and implications. *ACM 25*, 3 (July 1978), 499–508.

[27] Garey, Michael R., and Johnson, David S. *Computers And Intractability: A Guide to The Theory of NP-Completeness.* W.H.Freeman, New York, 1979.

[28] Gastpar, M., and Vetterli, M. On the capacity of wireless networks: the relay case. In *IEEE INFOCOM'02* (Jun 2002), pp. 1577–1856.

[29] Gerber, R., Hong, S., and Saksena, M. Guaranteeing real-time requirements with resource-based calibration of periodic processes. *IEEE Transactions on Software Engineering 21*, 7 (July 1995), 579–592.

[30] Golestani, S. A framing strategy for congestion management. *IEEE Journal on Selected Areas in Communications 9*, 7 (Sept. 1991), 1064–1077.

[31] Goossens, J., Funk, S., and Baruah, S. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-Time Systems 25*, 2/3 (2003), 187–205.

[32] Goossens, J., Funk, S., and Baruah, Sanjoy. Edf scheduling on multiprocessor platforms: some(perhaps)counterintuitive observations. In *Real-Time Computing Systems and Applications Symposium* (March 2002).

[33] Goossens, Joel, Funk, Shelby, and Baruah, Sanjoy. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-Time Systems 25*, 2-3 (September-October 2003), 187–205.

[34] Grnkvist, J. Comparison between graph-based and interference-based stdma scheduling. In *MobiHoc 2001* (2001), pp. 255–258.

[35] Gronkvist, J. Traffic controlled spatial reuse tdma in multi-hop radio networks. In *The Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* (Sept. 1998), pp. 1203 – 1207.

[36] Gronkvist, J. Assignment methods for spatial reuse tdma. In *Proceedings of IEEE/ACM Workshop on Mobile and Ad Hoc Networking and Computing (MobiHOC)* (August 2000).

[37] He, T., Vicaire, P. A., Yan, T., Luo, L., Gu, L., Zhou, G., Stoleru, R., Cao, Q., Stankovic, J. A., and Abdelzaher, T. Achieving real-time target tracking using wireless sensor networks. In *Proceedings of the 12th IEEE Real-Time Embedded Technology and Applications Symposium (RTAS'06), San Jose, California* (April 2006).

[38] He, Tian, Stankovic, John A., Lu, Chenyang, and Abdelzaher, Tarek F. Speed: A staleless protocol for real-time communication in sensor networks. In *International Conference on Distributed Computing Systems (ICDCS)* (May 2003).

[39] He, Tian, Stankovic, John A., Lu, Chenyang, and Abdelzaher, Tarek F. A spatiotemporal communication protocol for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 16*, 10 (October 2005), 995–1006.

[40] Holyer, Ian. The NP-completeness of edge-coloring. *SIAM Journal on Computing 10*, 4 (November 1981), 718–720.

[41] Jain, Kamal, Padhye, Jitendra, Padmanabhan, Venkata N., and Qiu, Lili. Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th annual international conference on Mobile computing and networking* (San Diego, CA, USA, 2003), pp. 66–80.

[42] Jiang, Jianping, Lai, Ten-Hwang, and Soundarajan, Neelam. On distributed dynamic channel allocation in mobile cellular networks. *IEEE Transactions on Parallel and Distributed Systems 13*, 10 (October 2002), 1024–1037.

[43] Johnson, D. B., and Maltz, D. A. Dynamic source routing in ad hoc wireless networks. *Mobile Computing* (1996), 153–181.

[44] Jung, E., and Vaidya, N. A power control mac protocol for ad hoc networks. In *Proc. of the 8th annual international conference on Mobile computing and networking, Atlanta, Georgia, USA* (2002).

[45] Kanodia, V., Li, C., Sabharwal, A., Sadeghi, B., and Knightly, E. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proceedings of the 7th annual international conference on Mobile computing and networking* (July 2001), pp. 200–209.

[46] Kim, T., Lee, J., Shin, H., and Chang, N. Best case response time analysis for improved schedulability analysis of distributed real-time tasks. In *Proceedings of ICDCS workshops on Distributed Real-Time systems* (April 2000).

[47] Krishna, C. M., and Shin, K. G. *Real-Time Systems.* McGraw-Hill, New York, 1997.

[48] Lazaridis, Iosif, and Mehrotra, Sharad. Capturing sensor-generated time series with quality guarantees. In *IEEE International Conference on Data Engineering (ICDE)* (2003), IEEE.

[49] Li, H., Shenoy, P., and Ramamritham, K. Scheduling messages with deadlines in multi-hop real-time sensor networks. In *Proceedings of the 11th IEEE Real-Time Embedded Technology and Applications Symposium (RTAS'05), San Francisco, California* (March 2005).

[50] Li, H., Shenoy, Prashant, and Ramamritham, K. Scheduling communication in real-time sensor application. Tech. Rep. TR-04-05, University of Massachusetts at Amherst, January 2004.

[51] Li, H., Shenoy, Prashant, and Ramamritham, K. Scheduling communication in real-time sensor applications. In *Proceedings of the Tenth IEEE Real-Time Embedded Technology and Applications Symposium (RTAS'04), Toronto, Canada* (May 2004).

[52] Li, H., Sweeney, J., Ramamritham, K., Grupen, R., and Shenoy, Prashant. Real-time support for mobile robotics. In *Proceedings of the Ninth IEEE Real/Time Embedded Technology and Applications Symposium (RTAS'03), Washington DC, USA* (May 2003).

[53] Li, Jinyang, Blake, Charles, De Couto, Douglas S. J., Lee, Hu Imm, and Morris, Robert. Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking* (Rome, Italy, July 2001), pp. 61–69.

[54] Li, V. O. K. Fair spatial tdma channel access protocols for multihop radio networks. In *Proc. of IEEE INFOCOM* (April 1991), pp. 1064–1073.

[55] Liu, C. L., and Layland, J. W. Scheduling algorithms for multiprogramming in a hard-real-time environment. *ACM 20*, 1 (1973), 46–61.

[56] Liu, J. W. S. *Real-Time Systems.* Prentice Hall, New Jersey, 2000.

[57] Lopez, J. M., Diaz, J. L., and Garcia, D. F. Minimum and maximum utilization bounds for multiprocessor rm scheduling. In *Proc. of the EuroMicro Conference on Real-Time Systems* (June 2001), pp. 67–75.

[58] Lu, C., Xing, G., Chipara, O., Fok, C.-L., and Bhattacharya, S. A spatiotemporal query service for mobile users in sensor networks. In *Proceedings of International Conference on Distributed Computing Systems (ICDCS'05), Columbus, OH* (June 2005).

[59] Lu, Chenyang, Blum, Brian M., Abdelzaher, Tarek F., Stankovic, John A., and He, Tian. Rap: A real-time communication architecture for large-scale wireless sensor networks. In *IEEE Real Time Technology and Applications Symposium (RTAS)* (September 2002), IEEE.

[60] Luo, H., Cheng, J., and Lu, S. Self-coordinating localized fair queueing in wireless ad-hoc networks. *IEEE Transactions on Mobile Computing 3*, 1 (January-March 2004), 86–98.

[61] Luo, H., Lu, S., and Bharghavan, V. A new model for packet scheduling in multihop wireless networks. In *Mobile Computing and Networking* (August 2000), pp. 76–86.

[62] Luo, H., Lu, S., Bharghavan, V., Cheng, J., and Zhong, G. A packet scheduling approach to qos support in multihop wireless networks. *ACM Journal of Mobile Networks and Applications (MONET), Special Issue on QoS in Heterogeneous Wireless Networks 9*, 3 (June 2004), 193–206.

[63] Meguerdichian, Seapahn, Slijepcevic, Sasa, Karayan, Vahag, and Potkonjak, Miodrag. Localized algorithms in wireless ad-hoc networks: location discovery and sensor exposure. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing* (2001), ACM, pp. 106 – 116.

[64] Moraes, R., Sadjadpour, H., and Garcia-Luna-Aceves, J. J. Reducing delay while maintaining capacity in mobile ad-hoc networks using multiple random routes. In *Asilomar Conference on Signals, Systems, and Computers* (Pacific Grove, CA, Nov 2004).

[65] Neely, Michael, and Modiano, Eytan. Capacity and delay tradeoffs in mobile ad hoc networks. In *(Invited Paper)IEEE BroadNets* (San Jose, CA, October 2004).

[66] Nelson, R., and Kleinrock, L. Spatial tdma, a collision-free multihop channel access protol. *IEEE Transactions on Communications COM-33*, 9 (Sept. 1985), 934–944.

[67] Oh, Dong-Ik, and Baker, T.P. Utilization bounds for n-processor rate monotone scheduling with static processor assignment. *Real-Time Systems 15* (1998), 183–192.

[68] Palencia, J. C., and Harbour, M. G. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings of the 19th IEEE Real-Time Systems Symposium* (December 1998).

[69] Palencia, J. C., and Harbour, M. G. Exploiting preceding relations in the schedulability analysis of distributed real-time systems. In *Proceedings of the 20th IEEE Real-Time Systems Symposium* (December 1999).

[70] Peng, D., Shin, K. G., and Abdelzaher, T. F. Assignment and scheduling communicating periodic tasks in distributed real-time systems. *IEEE Transactions on Software Engineering 23*, 12 (December 1997).

[71] Perkins, C. E., and Royer, E. M. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications* (Feb. 1999), pp. 90–100.

[72] Phillips, C. A., Stein, C., Torng, E., and Wein, J. Optimal time-critical scheduling via resource augmentation. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (1997), pp. 140–149.

[73] Ramamritham, K., Stankovic, J. A., and Shiah, P.-F. Efficient scheduling algorithms for realtime multiprocessor systems. *IEEE Trans. on Parallel and Distributed Systems 1*, 2 (April 1990), 184–194.

[74] Ramamritham, Krithi. Allocation and scheduling of precedence-related periodic tasks. *IEEE Transactions on Parallel and Distributed Systems 6*, 4 (November 1995).

[75] Ramamritham, Krithi. Where do time constraints come from and where do they go? *International Journal of Database Management 7*, 2 (November 1996), 4–10.

[76] Ray, Saikat, Carruthers, Jeffrey B., and Starobinski, David. RTS/CTS-induced congestion in ad hoc wireless LANs. In *Wireless Communications and Networking Conference (WCNC)* (March 2003).

[77] Ryu, M., and Hong, S. A period assignment algorithm for real-time system design. In *Proceedings of 1999 Conference on Tools and Algorithms for the Construction and Analysis of System* (1999).

[78] Saksena, M. Real-time system design: A temporal perspective. In *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering* (May 1998), pp. 405–408.

[79] Saksena, M., and Hong, S. Resource conscious design of distributed real-time systems an end-to-end approach. In *Proceedings of 1999 IEEE International Conference on Engineering of Complex Computer Systems* (October 1996), pp. 306–313.

[80] Sathaye, S. S., and Strosnider, J. K. Conventional and early token release scheduling models for the ieee 802.5 token ring. *Journal of RealTime Systems* (August 1993).

[81] Seto, D., Lehoczky, J. P., and Sha, L. Task period selection and schedulability in real-time systems. In *IEEE real-time systems symposium* (December 1998), pp. 188–198.

[82] Seto, D., Lehoczky, J. P., Sha, L., and Shin, K. G. On task schedulability in real-time control system. In *IEEE real-time systems symposium* (December 1996), pp. 13–21.

[83] Sha, L., Abdelzaher, T. F., rzn, K-E., Cervin, A, Baker, T. P., Burns, A., Buttazzo, G. C., Caccamo, M., Lehoczky, J. P., and Mok, A. K. Real time scheduling theory: A historical perspective. *Real-Time Systems 28*, 2-3 (Nov.-Dec. 2004), 101–155.

[84] Sha, L., Rajkumar, R., and Sathaye, S. S. Generalized rate monotonic scheduling theory: A framework for developing real-time systems. *Proceedings of the IEEE 82*, 1 (January 1994), 68–82.

[85] Sharma, G., and Mazumdar, R. On achievable delay/capacity trade-offs in mobile ad hoc networks. In *Conference on Wireless Networks and Optimization (WiOpt)* (Cambridge, UK, March 2004).

[86] Shih, E., Cho, S., Ickes, N., Min, R., Sinha, A., Wang, A., and Chandrakasan, A. Physical layer driven protocol and algorithm design for energy-efficient wireless

sensor networks. In *Proceedings of the Seventh Annual ACM International Conference on Mobile Computing and Networking* (July 2001), pp. 272–286.

[87] Sibley, G. T., Rahimi, M. H., and Sukhatme, G. S. Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks. In *Proceedings of International Conference on Robotics and Automation, Washington DC.* (Sept. 2002), pp. 1143– 1148.

[88] Snchez, M., Zander, J., and Giles, T. Combined routing scheduling for spatial tdma in multihop ad hoc networks. In *The 5th International Symposium on Wireless Personal Multimedia Communications, 2002.* (Oct. 2002), pp. 781–785.

[89] Sohrabi, K., Gao, J., Ailawadhi, V., and Pottie, G. J. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications 7*, 5 (2000), 16–27.

[90] Sohrabi, K., and Pottie, G. Performance of a novel self-organization protocol for wireless ad hoc sensor networks. In *Proc. of the IEEE 50th Vehicular Technology Conference. Amsterdam* (1999), pp. 1222–1226.

[91] Somarriba, O., and Giles, T. Transmission control for spatial tdma in wireless radio networks. In *IEEE* (2002).

[92] S.Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks 5*, 2 (March 1999), 81–94.

[93] Srinivasan, Anand, and Baruah, S. K. Deadline-based scheduling of periodic task systems on multiprocessors. *Information Processing Letters 84*, 2 (November 2002), 93–98.

[94] Stankovic, John A. Research challenges for wireless sensor networks. *SIGBED Review: Special Issue on Embedded Sensor Networks and Wireless Computing 1*, 2 (July 2004).

[95] Stankovic, John A., Abdelzaher, Tarek F., Lu, Chenyang, Sha, Lui, and Hou, Jennifer C. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE 91*, 7 (2003), 1002–1022.

[96] Stevens, D. S., and Ammar, M. H. Evaluation of slot allocation strategies for tdma protocols in packet radio networks. In *Proc. of IEEE MILCOM* (1990), pp. 835–839.

[97] Sub-kilogram intelligent tele-robots. http://www-robotics.usc.edu/ behar/SKIT.html.

[98] Sweeney, J., Brunette, T.J., Yang, Y., and Grupen, R. Coordinated teams of reactive mobile platforms. In *Proceedings of the 2002 IEEE Conference on Robotics and Automation, Washington, D.C.* (May 2002).

[99] Sweeney, J., Li, H., Grupen, R., and Ramamritham, K. Scalability and schedulability in large, coordinated, distributed robot systems. In *Proceedings of the 2003 IEEE Conference on Robotics and Automation, Taipei, Taiwan* (May 2003).

[100] Tay, Y., Jamieson, K., and Balakrishnan, H. Collision-minimizing csma and its applications to wireless sensor networks. *IEEE Journal on Selected Areas in Communications 22*, 6 (2004), 1048–1057.

[101] Tindell, Burns, and Wellings. Allocating hard real-time tasks: An NP-hard problem made easy. *RTSYSTS: Real-Time Systems 4* (1992).

[102] Umass: Networks of distributed sensory and motor services. http://www-robotics.cs.umass.edu/Research/Distributed_Control/uBot/uBot.html.

[103] Wang, S., and Farber, G. On the schedulability analysis for distributed real-time systems. In *Joint IFAC—IFIP WRTP'99 & ARTDB-99* (May 1999).

[104] Woo, A., Tong, T., and Culler, D. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. of the ACM SenSys* (November 2003).

[105] Woo, Alec, and Culler, David E. A transmission control scheme for media access in sensor networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking, Rome, Italy* (July 2001), pp. 221–235.

[106] Wrege, D., and Liebeherr, J. A near-optimal packet scheduler for qos networks. In *Proceedings of IEEE INFOCOM '97* (1997).

[107] Xu, N., Rangwala, S., Chintalapudi, K., Ganesan, D., Broad, A., Govindan, R., and Estrin, D. A wireless sensor network for structural monitoring. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems,Baltimore, MD* (November 2004).

[108] Ye, W., Heidemann, J., and Estrin, D. An energy-efficient mac protocol for wireless sensor network. In *Proc. of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY* (June 2002).

[109] Zhao, W., and Ramamritham, K. Virtual time csma protocols for hard real-time communication. *IEEE Transactions on Software Engineering 13*, 8 (1987), 938–952.

[110] Zhao, W., Stankovic, J.A., and Ramamritham, K. A window protocol for transmission of time-constrained messages. *IEEE Trans. Computers 39*, 9 (Sept. 1990), 1186–1203.

[111] Zhou, G., He, T., Krishnamurthy, S., and Stankovic, J. Impact of radio irregularity on wireless sensor networks. In *ACM Conference on Mobile Systems, Applications, and Services (Mobisys)* (June 2004), pp. 125–138.