Scheduling Communication in Real-Time Sensor Applications

Huan Li, Prashant Shenoy,
Department of Computer Science,
University of Massachusetts,
Amherst, MA 01003
{lihuan,shenoy}@cs.umass.edu

Krithi Ramamritham
Dept. of Computer Science and Engineering,
Indian Institute of Technology,
Mumbai 400076, India
krithi@cse.iitb.ac.in

Abstract

The design and deployment of wireless sensor applications has received increased research attention in recent years. In this work, we consider a class of wireless sensor applications—such as mobile robotics—that impose timeliness constraints. We assume that these applications are built using commodity 802.11 wireless networks and focus on the problem of providing qualitatively-better QoS during network transmission of sensor data. Our techniques are designed to explicitly avoid network collisions and minimize the completion time to transmit a set of sensor messages. We argue that this problem is NP-complete and present several heuristics, based on edge coloring, to achieve these goals. We present detailed simulation results to evaluate our heuristics and to compare them to the optimal solution.

Keywords*: sensor networks, wireless communication, network transmission*

1 Introduction

1.1 Motivation

The design and deployment of wireless sensor applications has received increased research attention in recent years. Wireless sensors are useful in a variety of remote data monitoring and data gathering applications. Example applications include environmental monitoring in remote areas [1], monitoring of ocean temperatures [7], and search and rescue operations. A sensor application typically includes a collection of sensors that continuously monitor the surrounding environment, and a collection of sinks that aggregate, process, and react to this sensory data. Communication between the sensors and sinks requires a network; since the inherent nature of the above applications precludes the use of wired networks, wireless networks are commonly used in many sensor applications.

In this work, 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). An example of such an application is a team of robots searching for people trapped 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, weight and design considerations (e.g., 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 use it to determine where to move next, both individually and as a group. Since the path for each robot needs to be determined in real-time to ensure real-time mobility, the *transmission* and *processing* of sensory data imposes timeliness constraints.

In this paper, we focus on the former problem, namely the network transmission of data in a real-time sensor application. The latter problem of scheduling processing tasks to meet timeliness constraints in a real-time sensor application was considered in a prior work [8]. 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 network transmission technique based on CSMA/CA. Such a best-effort technique can delay the receipt of the data at receivers, which can be problematic in time-sensitive sensor applications. Consequently, we propose techniques to provide qualitatively-better QoS in such wireless networks. The reasons for considering such alternatives are manifold. First, CSMA/CA networks do not completely eliminate the possibility of a collision despite the collision avoidance techniques. Second, senders can back-off exponentially when they sense ongoing transmissions on the channel. Third, vanilla 802.11 networks suffer from the *blocking problem* as observed in [2]. In such networks, a node must explicitly request permission to transmit and must receive a "clear-to-send" (CTS) acknowledgment before sending data. Further, all nodes in the vicinity that receive these messages must in-

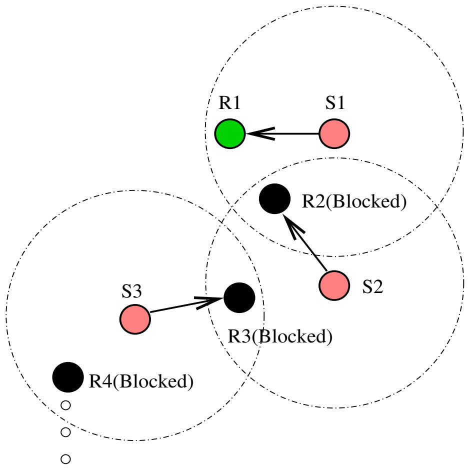


Figure 1. False Blocking Problem

habit transmission for the transmission duration, and are thus *blocked*. Such a protocol can lead to *false blocking* and *blocking propagation* as observed in [11] and illustrated in Figure 1. False blocking occurs when a node sends a "request-to-send" (RTS) message to a blocked destination node; the RTS causes other nodes to block even though no data is actually sent. As shown in the figure, node S_1 transmits data to node R_1 and node R_2 is disabled 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 even though the transmissions $S_1 \rightarrow R_1$ and $S_3 \rightarrow R_3$ can occur in parallel. This phenomenon is referred to as *false blocking*. As indicated in [11], false blocking may *propagate* through the whole network, and even lead to a *deadlock*, at least for temporary periods (until one of the nodes backs-off after a certain number of retries and the deadlock is be broken).

In this paper, we consider techniques to avoid such problems in sensor applications built using vanilla wireless networks. We exploit the specific characteristics of sensor applications, such as the robotics scenario, to devise network transmission techniques so that collisions are *explicitly* avoided and the total completion time for transmitting a set of messages is minimized (by parallelizing non-interfering transmissions to the extent possible).

1.2 Research Contributions

In this paper, we consider the problem of scheduling the transmission of sensor data over commodity wireless networks so as to explicitly avoid collisions and minimize completion time. We argue that the problem is NP-complete (proof is in the appendix) and present heuristics based on edge coloring to address this problem. We discuss modifications to our approach to incorporate timeliness constraints and also present an optimal search technique based on the A^* technique to enable comparisons with our heuristics. We present simulation results to evaluate the effectiveness of our heuristics and show that our heuristics are close to the optimal solution in small sensor networks.

The remainder of this paper is structured as follows. Section 2 presents our system model and the problem formulation. Sections 3 and 4 present the optimal solution and our edge coloring-based heuristics, respectively. Our simulation results are presented in Section 5. Section 6 presents related work and, finally, Section 7 presents our conclusions.

2 Background and Problem Formulation

In this section, we formulate the problem of communication scheduling in sensor applications and present the model assumed in our research.

2.1 System Model

Consider a wireless sensor application with N nodes. Each node can be a source or a sink or both. 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 gave different transmission ranges. All nodes are assumed to communicate in the ad-hoc mode and no base-stations are assumed in this environment. For simplicity, we assume that a sink for a data item is within transmission range of its source. All communication is assumed to be unicast in this environment. The terms source and sender as well as sink and receiver are used interchangeably in this paper.

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 there can be interference if more than one sender attempts to transmit simultaneously. However, two receivers that are *mutually* outside the other sender's range can receive data simultaneously. For the example shown in Figure 2, R_1 is covered by the transmission range of both S_1 and S_2, and interference may occur if both senders attempt to communicate with their receivers simultaneously. However, S_3 can transmit in parallel with either of 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 network transmissions. This is a reasonable assumption, since in the robotics example, robots carry GPS receivers and can additionally use localization algorithms [10] to precisely determine their locations.

Observe in Figure 2 that source S_2 sends data to two different receivers R_2 and R_3. Since each robot carries multiple sensors, such a scenario might result from the need to transmit data from different sensors on S_2 to different robots (R_2

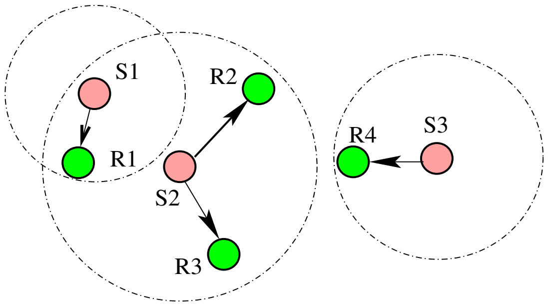


Figure 2. Sensor Communication: An Example

and R_3). Even when data from the same sensor is needed by different robots (e.g., R_2 and R_3), the unicast nature of the communication necessitates separate messages to each receiver.

We can precisely state the above assumptions as follows. Let $Range(S_i, R_x)$ denote that R_x is within transmission range of S_i, and $S_i \rightarrow R_x$ describe the state that S_i sends messages to R_x.

- *Wireless NIC Constraint:* At one instant, a node can be either a sender or a receiver, but not both.
- *Single hop constraint:* A node can receive a message only if it is in the range of the sender range, i.e., $S_i \rightarrow R_x \Longrightarrow Range(S_i, R_x)$.
- *Unicast constraint:* Each message can have only one recipient. That is $(S_i \rightarrow R_x) \wedge (S_i \rightarrow R_y) \wedge (R_x \neq R_y) \Longrightarrow (S_i \rightarrow R_x) \neq (S_i \rightarrow R_y)$
- *Interference constraint:* Two simultaneous transmissions will not interfere if and only if both receivers are mutually outside the other sender's range. That is, at time t, $\neg Interference(S_i \overset{t}{\rightarrow} R_x,\ S_j \overset{t}{\rightarrow} R_y) \iff \neg(Range(S_i, R_y) \vee Range(S_j, R_x))$. Where, $S_i \overset{t}{\rightarrow} R_x$ means at time t, S_i is sending a message to R_x.

Given a set of sources, sinks, their locations and transmission ranges, the objective of a good communication scheduler is to determine a transmission schedule such that there is no interfering communication and the total time to transmit all messages is minimized. Depending the assumptions made about individual messages, there can be different variants of this problem, which we describe next.

2.2 Problem Formulation

We discuss three possible variants of the problem of scheduling communication in real-time sensor applications.

Static Scheduling with identical ready times: In this version of the problem, the scheduler is assumed to run periodically and considers all messages that have become ready for transmission since the previous scheduling instance. The length of each message is assumed to be known and all messages are assumed to be ready for transmission at schedule time. In addition, as noted earlier, the location of each node and the range constraints are assumed to be known to the scheduler. Given these assumptions, the objective of the scheduler is to determine a schedule (i.e., a transmission time for each message) such that there is no interference between simultaneous transmissions and the total completion time for sending all messages is minimized.

Observe that this variant of the problem assumes a centralized scheduler since the scheduler must consider the range constraints of all nodes when determining a schedule. The assumption of a centralized scheduler is reasonable for our scenario, since in application such as robotics, a centralized path planner is used to determine the path for each robot for the next time period P. Consequently, the scheduler can run periodically in conjunction with the path planner to determine the transmission schedule. Observe that the schedule is static—any new messages are queued up until the next scheduling instance and do not affect the current schedule.

In a sensor application, sensors will periodically generate data and this data will need to be sent to one or more sinks. Different senors may have different periods. It is assumed that the scheduler runs periodically and considers all sensor data that has been produced since the last scheduling instance for transmission.

Static Scheduling with different ready times: This variant is a generalization of the previous scenario. In this case, messages are assumed to become ready for transmission at arbitrary times but the ready times are assumed to be known a priori at schedule time. The length of the message and the range constraints are also assumed to be known. The scheduler, also centralized in this case, must determine a schedule that avoids conflicts and minimized completion times be parallelizing non-interfering transmissions. An additional constraint is that a message may not be scheduled for transmission prior to its ready time.

Dynamic Scheduling: In this variant, the message ready times and message lengths are known only at run time. The range constraints are assumed to be known when a message is ready for transmission. When a message becomes ready for transmission, the scheduler can insert the message into the current schedule (by modifying it appropriately). In this sense, the scheduling is dynamic. Dynamic scheduling can either be centralized or distributed (in the latter case, scheduling decisions can be made locally at each node). Like in the previous two scenarios, a good dynamic scheduler should avoid conflicts while attempting to minimize completion times.

In this paper, we consider the first variant, namely static scheduling with identical ready times. Although this is the simplest of the three scenarios, as indicated above, it has practical uses in applications such as real-time robotics. Further, as we show in the next section, even the simple scenario of static scheduling with identical ready times is NP-Complete, and consequently, we will need to resort to heuristics when devising our solutions.

2.3 Graph-based Representation of the Problem

Consider a set of messages that are queued for transmission at various nodes. The scheduling problem can be formulated as a directed graph $G = (V, E)$ where each vertex in the graph represents a node in the sensor application. We refer to this graph as the *communication graph*. 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 represents the transmission cost and is a function of the message length (and the transmission rate). Given such a graph, the problem of scheduling messages to avoid interference and minimize completion time can be addressed as follows. Choose a set of edges $\mathcal{S}_i$ that do not conflict with one another (we will define how to do this in a moment). These edges are deleted from the graph and the process repeats recursively until all edges have been chosen. Since each set of edges is chosen such that the edges do not conflict, the corresponding messages can be scheduled in parallel. Thus, the above process yields a transmission schedule $\mathcal{S}_1, \mathcal{S}_2, \ldots$ that indicates which set of messages should be scheduled for transmission concurrently. In essence, this is the intuition behind our approach.

There can be two types of conflicts between edges. First, two edges that are incident on a common vertex can not belong to the same set. This is because a node can send or receive only one message at any time. Second, if the transmission of two messages interfere, then the corresponding edges have a conflict and can not belong to the same set. Edges in each set should be chosen subject to these constraints. Last, we note messages can have different lengths and the total completion time is the sum of the length of the longest message in each set. The length of messages will also need to be considered when choosing these sets.

Consider a simplified version of the problem where all messages are of equal length (edge weights are equal) and there is no interference between any of the messages. In this case, the only constraint is that all edges incident to the same vertex cannot be scheduled simultaneously. This simpler version of the problem can be mapped onto the *edge coloring* problem . An *edge coloring* of a graph is an assignment of colors to the edges such that edges incident to the same vertex receive different colors. Edges with identical color can be scheduled in parallel, since these edges will not conflict with one another. We also note that an edge coloring can be viewed as a partitioning of the edges into disjoint *matchings* (a matching is a subset of edges such that no two edges share a vertex).

It has been shown that the problem of determining the minimum number of colors is NP-Complete [5]. Consequently, the problem of minimizing the completion time is computationally intractable even for the simple case where messages are of equal length and there is no interference between messages.

3 Optimal Communication Scheduling

In this section, we present a search-based technique to compute the optimal solution to the communication scheduling problem. An optimal solution is one that minimizes the completion time for all transmissions, subject to the constraints. Since the problem is NP-Complete (see the appendix), any attempt to find an optimal solution will incur an exponential increase as the problem scales. Nevertheless, it is useful to consider the optimal solution to enable comparisons with our proposed heuristics. We first discuss the the intuition behind the search-based technique and then propose an optimal solution based on the A^* search algorithm.

3.1 A Search-based Technique

It is helpful to think of the search process as a search tree where the root of the tree is a node representing the original communication graph. The search process expands each node in the tree as follows. It finds all matchings for the node subject to the interference constraint. Each matching essentially represents a set of edges that have no conflicts and hence can be scheduled in parallel. For each such matching, the corresponding edges are deleted from the graph and the resulting graph is added as a leaf node in the search tree. The search process continues until we are left with an empty graph at each leaf. Each path from the root to a leaf is essentially a transmission schedule. The optimal solution is the path with the minimum cost, where the cost is defined to be the sum of cost of each matching along that path (the cost of a matching is the cost of the largest edge weight in that matching).

Since finding all matchings for a graph is a subset problem, the search to the optimal solution takes at least $O(2^n)$ time.

3.2 An A^*-based Search Algorithm

Rather than conduct a brute force search, we use the A^* algorithm to conduct a directed search. A^* search has been proved to be **optimally efficient** in that no other search algorithm will expand fewer nodes in the search tree to locate the optimal solution [3]. The A^* algorithm requires an evaluation function $f(n) = g(n) + h(n)$ for any node n on the search tree, where $g(n)$ 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. A^* then expands the node with the minimum f value. To guarantee that the search algorithm is complete and optimal, A^* requires that the h function should *never overestimate* the cost to reach the goal. Such an h is called an *admission heuristic*.

We define the g function as the sum of costs of all matchings from the root to the node. Let $W(e_x)$ denote the cost of edge e_x in the original communication graph. Let n_G denotes a node representing a graph G in the search tree, and let $n'_{G'}$ denote a child of this node representing graph G', then $g(n'_{G'})$ can be defined as:

$$\begin{aligned} g(n'_{G'}) &= g(n_G) + matching\_cost(M_i) \\ &= g(n_G) + max\left(W(e_x)\right),\ e_x \in M_i \end{aligned} \quad (1)$$

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:

$$\begin{aligned} h(n_G) &= max\left(W(v_i)\right) \\ &= max\left(\sum_j W(e_{ij})\right) \end{aligned} \quad (2)$$

where, e_{ij} are all edges that are incident to 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 Heuristic Communication Scheduling

In this section, we will present polynomial time heuristics for the communication scheduling problem. Our heuristics will 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 considered when generating a transmission schedule. The notation used in this section is summarized in Table 1.

Notation	Meaning
e_i	edge ID
c_i	color ID
$c(e_i)$	the color of e_i
$e_i \sim e_j$	edge e_i is adjacent to edge 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

Table 1. Notation

4.1 Edge Coloring Heuristics

The objective of the heuristic is to assign a color to each edge such that (i) no two edges incident on the same vertex have the same color, (ii) no two edges with the same color interference with one another, and (iii) the total completion time is minimized.

Given a communication graph, each edge is assigned a *palette*. Initially the palettes of all edges are identical and are assumed to contain a sufficiently large number of colors. Each color in the palette is assumed to have a weight. Initially, all colors are given a weight of zero and as the heuristic progresses, the weight of 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. Each edge incident on this vertex is assigned a different color. If the degree of this node is d, then we need d distinct colors to color these 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 share a 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). 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 adjacent to this edge.

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

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 $W(c_i)$). Hence, the total completion time of all messages is given as $\sum_k W(c_k)$, where k is the number of distinct colors that are needed to color the graph.

Figure 3 depicts the various steps in our heuristic.

4.2 Color Selection Policies

We present three heuristics to choose the color for an edge from the colors in its palette.

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 includes a *longer* message, then choosing that color will not result in any increase in the total time to transmit all messages of that color. This is the intuition behind this heuristic.

Suppose that the weight of the current 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 contain a message that is *longer* than the current message. The MWC

Heuristic:
Input: A communication graph with message lengths and all constraints.
Output: The total 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.

Figure 3. Edge Coloring Heuristic

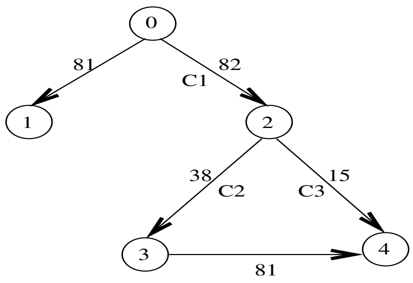


Figure 4. An Example to Illustrate the Algorithm

heuristic picks a color with the *least weight* from all colors that have a weight greater than the edge weight. Note that the interference constrain 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.

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 such that 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.

Consider the example depicted in Figure 4. Initially, suppose that each palette has 4 colors. Since vertex v_2 has the maximum degree, the heuristic begins by assigning distinct colors to all edges incident on v_2. Since now edge $v_3 \rightarrow v_4$ has the smallest palette, it is considered next. If we use the MWC heuristic, color c_1 is selected, because $W(c_1) = 82 > 81$. If we use the LUC heuristic, color c_4 is chosen and $W(c_4) \leftarrow 81$, since c_4 is the currently least used color. Finally, let us consider edge $v_0 \rightarrow v_1$. Any color except c_1 is a possible color. If we use MWC, c_2 is chosen since it is the heaviest edge (the weights of all possible colors are less than 81), and $W(c_2) \leftarrow 81$. And for LUC, since all colors have been used once, any possible color can be used. In above analysis, we assume that there is no interference in any step. Therefore, the completion time for MWC is $W(c_1)+W(c_2)+W(c_3) = 82+81+15 = 178$; for LUC, the completion time is larger than the time before assigning $v_0 \rightarrow v_1$, which is $W(c_1) + W(c_2) + W(c_3) + W(c_4) = 82 + 38 + 15 + 81 = 216$.

4.3 Incorporating Timeliness Constraints

Our heuristics thus far have focused on scheduling messages so as to minimize completion time. Since we are concerned with real-time sensor applications in this work, it is conceivable that messages will have deadlines on when they should be received at the destination node. In general, these deadlines will be determined by the deadline of the processing task at the sink that will consume the data, once received. It is possible to enhance our heuristics to take deadlines of messages into account. To do so, we assume that each edge on the communication graphs is labeled by a deadline (in addition to a weight).

One approach is to first determine a coloring of edges based on the techniques described in the previous section. Observe that while messages with identical colors can be scheduled in parallel, our heuristics leave the *ordering* of colors unspecified. For instance, if a graph is assigned two colors—red and blue—then we could schedule all red messages first, followed by the blue messages, or vice versa. We can exploit this flexibility to take deadlines into account. We define the deadline of a color to be the minimum deadline of all messages with that color. We can then simply order colors by their deadline. This ordering determines an ordering on the scheduling of messages across various colors—colors with earlier deadlines get scheduled before those with later deadlines (messages of the same color are scheduled in parallel, like before). Note that, regardless of the ordering of colors, the total completion time remains unchanged.

Another approach is to incorporate deadlines when assigning colors to edges. This will require us to modify the edge coloring heuristics outlined in the previous section. While a detailed discussion of such heuristics is beyond the scope of this paper, we present an brief outline of how such a technique might work. The technique will need to balance three factors—the weight, the deadline, and the palette of an edge,. Edges can be chosen based on the most important factor. For instance, if meeting deadlines is a primary goal and reducing completion times is a secondary goal, then edges can be chosen for coloring in order of their deadlines. If the opposite is true, the edges are chosen based on their palette sizes and they can be assigned a color with the smallest deadline. A detailed exposition of these ideas is the subject of future work.

4.4 Discussion

Recall from Section 2, that we assume a centralized scheduler that is invoked periodically, which then schedules all messages that are ready for transmission. For simplicity of exposition, we have also implicitly assumed that each sensor produces no more than one update between two consecutive invocations of the scheduler. In general, a sensor may produce an arbitrary number of updates between two scheduler invocations, all of which will need to be scheduled for transmission (with minimum queuing delays). It is easy to extend our technique to this general case. Let us assume that different sensors have different periods (i.e., frequency at which they produce updates) and let p_i denote the period of sensor i. Let p_{min} denote the sensor with the least period in the system. Let P denote the period at which the scheduler is invoked. Since the scheduler invocation frequency can be chosen by the system designer, we will choose P to be an integral multiple of p_{min}.

When invoked, the schedule partitions the next P time units into slots of duration p_{min}. At the beginning of each such time slot, it determines all sensor updates that were produced in the previous slot and schedules them for transmission using our edge coloring heuristics. As an example, consider two sensors A and B produce updates once every 1 second and 2 second, respectively. Let the scheduler be invoked once every 10 seconds. Since $p_{min} = 1s$, at each invocation, the scheduler considers each 1 second interval for the next 10 seconds. In each such slot, it schedules all updates produced in the prior 1 second time slot. At the end of the first second, sensor A produces an updates and it is scheduled for transmission. At $t = 2s$, both A and B produce updates, and both are scheduled using the edge coloring heuristics. At $t = 3s$, only the update from A needs scheduling, and so on until t=10s In this fashion, our techniques can schedule sensor updates with arbitrary periods.

5 Simulation Results

We conduct a simulation study to understand the performance of our heuristics. Our study also compares our heuristics to the optimal solution computed by the A^* search algorithm. Our simulations assume that nodes are uniformly distributed in a given area and may have different transmission ranges. We assume that the sensor network is represented by the 3-tuple (N, P, R), where N is the number of nodes, $P = \{(X_i, Y_i), 1 \le i \le N\}$ is the set of positions for each

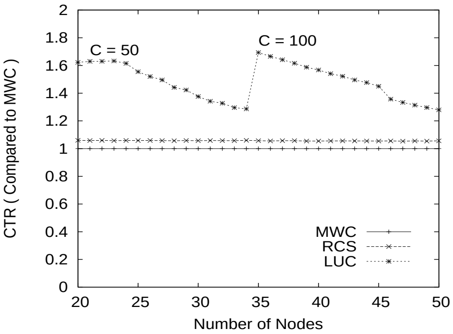


Figure 5. Effect of the Number of Nodes

of the node in the area. The position, P_i of node N_i is generated randomly in the area of $[100, 100]$ units for its X and Y coordinates. R is the transmission range. 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 the range $[10, 100]$.

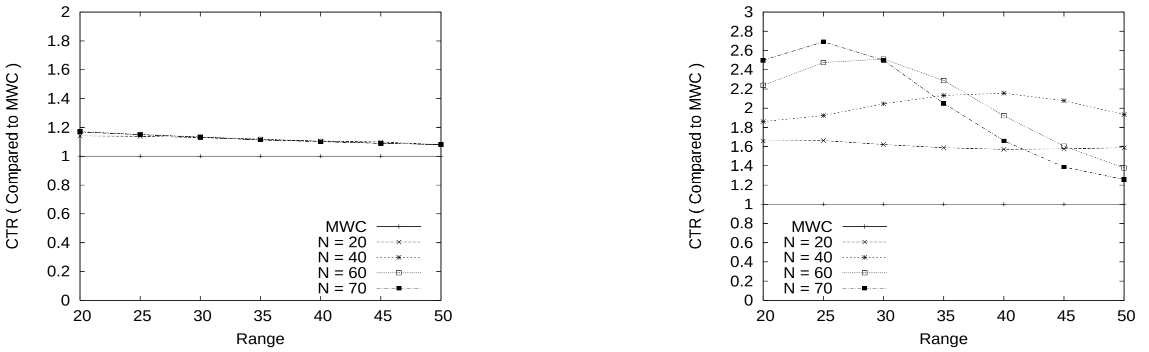
All results shown in this section are obtained as the mean of 1000 runs. In each run, one communication graph is generated with some specific settings. Since the algorithms attempt to minimize the total completion time for all transmissions in a graph, the performance is measured by comparing the completion times across different algorithms. We denote this as the *Completion Time Ratio (CTR)*.

5.1 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 range.

5.1.1 Effect of the Number of Nodes

To study the impact of the number of nodes (denoted by N), we systematically vary N from 20 to 50. For each N, we generate a thousand different communication graphs and determine the total time to schedule all messages using the three heuristics. For each graph, the transmission radius of a sender is chosen randomly over the range $[10, 100]$. Figure 5 shows the completion time ratio 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. Random color selection (RCS) yields completion times that are within 8% of MWC (and the performance is not sensitive to the value of N). The least used color (LUC) heuristic yields the worst performance. 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) RCS vs MWC

(b) LUC vs MWC

Figure 6. Effect of Range

a fixed palette size, when N increases, the CTR decreases. The closer the initial palette size the number of colors needed, the better is the LUC performance. We choose more colors at $N = 35$ because the palette has not enough colors to cover all edges. In the figure, the number of colors is represented by C.

5.1.2 Effect of the Range

In this experiment, we systematically vary the transmission range of a node from 20 to 50. For a given transmission range, we determine the completion times for the three heuristics for sensor networks containing 20, 40, 60 and 70 nodes. The initial size of the palette is set to 100 colors.

Figure 6 (a) compares the completion times of RCS and MWC, while Figure 6(b) 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 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.

5.1.3 Effect of the Number of Edges

The number of edges reflects the communication density. We vary the number of edges in the communication graph systematically and study the performance of the three heuristics. Figure 7 plots the completion time ratio for the three heuristics. Again, the MWC outperforms the other two heuristics. Random color selection is about 10-18% worse. For LCU, when is the density is small, the number of colors chosen is much larger than necessary (and depends on the initial palette). The performance of LUC improves as the number of edges increases.

Overall, our results indicate that the minimum weight

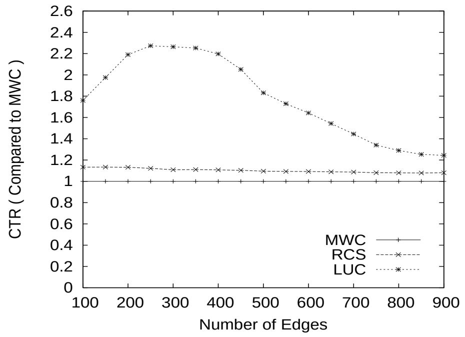


Figure 7. Effect of the Number of Edges

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.

5.2 MWC versus the Optimal Solution

MWC is a time-based heuristic that is specifically designed to minimize 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 network sizes. Consequently, we restrict the network size to no more than 20 nodes in our experiments. The number of edges is restricted to 20—this involves expanding $O(2^{20})$ nodes in the search tree (even for these small sensor networks, computing the optimal transmission schedule can take several hours on a state-of-the-art 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 time of the schedules produced by MWC and A^*. In the second experiment, each node in the network is assumed to have an identical transmission range; we vary the number of nodes and compute the completion times for the two techniques. Figure 8 and 9 plot our results. The figures plot the completion time ratio (normalized by the optimal solution) and 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% of the optimal solution for sensor networks of up to 20 nodes (and 20 edges).
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 6-8% worse for a variety of transmission ranges.
3. When the transmission range becomes larger, e.g., in $R = [10, 80]$ or $R = 30$, the performance actually becomes stable, which indicates that MWC is robust even when the complexity increases.

5.3 Summary of Results

Our experiments yield 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 time into account when generating a schedule.
- RCS is close second in terms of the completion times of its schedules. Further, RCS, like MWC, is not very sensitive to the number of nodes or the range parameters.
- LUC has worst performance and is very sensitive to the initial number of colors in the palette.
- For small networks, the results of MWC are close to the optimal solution.
- The performance of MWC is robust as the communication density (complexity) increases.

6 Related Work

Real-time issues that arise in various layers of the network stack in sensor networks are studied in [13]. 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. Our work aims to provide the explicit delay for data transmissions by avoiding collisions; at the same time, the total transmission time for sending sensor messages is minimized. Our techniques can also take timeliness constraints into account when making scheduling decisions.

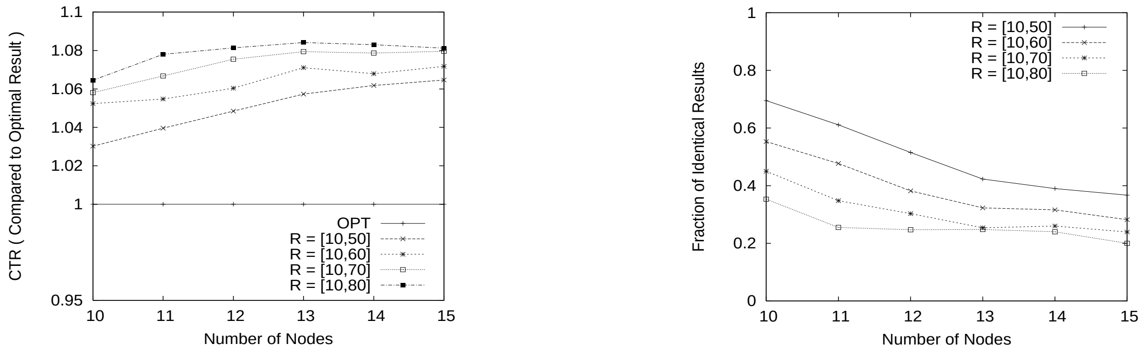
Other efforts that address real-time issues in sensor networks include the design of the communication stack and real-time routing. For example, RAP [9] is a real-time communication architecture for large-scale wireless sensor networks that proposes a novel packet scheduling policy called velocity monotonic scheduling. In this method, the requested velocity is mapped to a MAC-layer priority, which in turn reduces the deadline miss ratio. In [4], an adaptive real-time routing protocol, SPEED, uses feedback-based techniques that try to satisfy per-hop deadlines in face of unpredictable traffic.

In cellular telephones systems, a communication channel—a band of frequencies—can be used simultaneously by many callers if these callers are spatially apart and their calls do not interfere with one another. 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*. This problem is similar to our problem in terms of the ability to reuse the channel. The difference is that there is no need to consider the transmission cost. Ramanathan [12] 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 [6], the authors established a relationship between the mutual exclusion problem and the distributed dynamic channel allocation problem.

7 Conclusions

In this paper, we considered a class of wireless sensor applications—such as mobile robotics—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 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. 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 in small sensor networks.

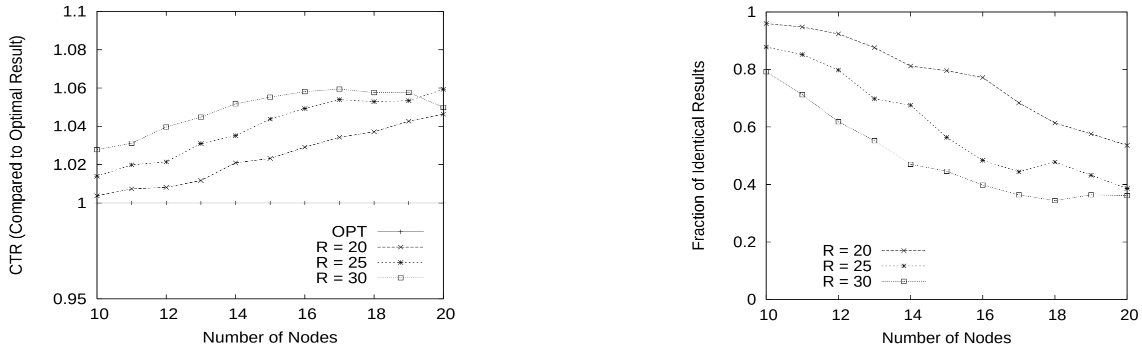
As part of future work, we plan to evaluate the effectiveness of our techniques by implementing them into a sensor testbed. To do so, we plan to design a scheduler above the MAC layer to prioritize the packet transmissions based on the transmission schedule. We also plan to examine the impact of message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



(a) Completion Time of MWC to OPT

(b) Fraction of Identical Results

Figure 8. MWC Versus OPT for Varying Transmission Ranges



(a) Completion Time of MWC to OPT

(b) Fraction of Identical Results

Figure 9. MWC Versus OPT for Varying Transmission Ranges

multi-hop sensor networks.

References

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

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

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

[4] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher. Speed: A staleless protocol for real-time communication in sensor networks. In *ICDCS*, May 2003.

[5] I. Holyer. The np-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, November 1981.

[6] J. Jiang, T.-H. Lai, and N. Soundarajan. On distributed dynamic channel allocation in mobile cellular networks. *IEEE Transactions on Paralle and Distributed Systems*, 13(10):1024–1037, October 2002.

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

[8] H. Li, J. Sweeney, K. Ramamritham, R. A. Grupen, and P. J. Shenoy. Real-time support for mobile robotics. In *IEEE Real Time Technology and Applications Symposium 2003 (RTAS)*, pages 10–18. IEEE, May 2003.

[9] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. Rap: A real-time communication architecture for large-scale wireless sensor networks. In *RTAS*, June 2002.

[10] S. Meguerdichian, S. Slijepcevic, V. Karayan, and M. Potkonjak. 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*, pages 106 – 116. ACM, 2001.

[11] S. Ray, J. B. Carruthers, and D. Starobinski. Rts/cts-induced congestion in ad hoc wireless lans. In *WCNC 2003: Wireless Communications and Networking Conferenc*, March 2003.

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

[13] J. A. Stankovic, T. F. Abdelzaher, C. Lu, L. Sha, and J. C. Hou. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7):1002–1022, 2003.

APPENDIX

Note: this appendix is provided for ease of the reviewing. It is not required to read this appendix to review the rest of the paper.

In the appendix, we are proving the **NP**-completeness of the **Optimal Parallel Communication Scheduling (OPCS) Problem**. In the OPCS problem, a graph is used to represent the communication, and there is an interference constraint set $I \subseteq E \times E$ such that, $\forall (e_i, e_j) \in I$, e_i cannot be scheduled simultaneously with e_j because of interference.

Optimal Parallel Communication Scheduling (OPCS) Problem can be formulated as follows.

Input: Graph $G = (V, E)$, weight function $w : E \rightarrow Z^+$, interference constraint set $I \subseteq E \times E$.

Question: 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. $\sum_{1 \le i \le n} \max_{e \in E_i}(w(e))$ is minimized.

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

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

To show that it is **NP**-hard, we prove that Minimum-Edge-Coloring (MEC) is polynomially reducible to OPCS problem, i.e., MEC $\le_p$ OPCS. The Minimum-Edge-Coloring below is an **NP**-complete problem [5]:

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 $1 \le i \le n$, 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.

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

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

Suppose $E_1, E_2, \ldots, E_k$ is an optimal solution of OPCS. 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 OPCS 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 OPCS, and it has a lower value: $\sum_{1 \le i \le k'} max_{e \in E_i^{'}}(w(e)) = \sum_{1 \le i \le k'} 1 = k' < k = \sum_{1 \le i \le k} max_{e \in E_i}(w(e))$. Therefore, $E_1, E_2, \ldots, E_k$ is not an optimal partition, which contradicts the assumption.

This completes the proof.