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

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 three heuristics, based on edge coloring, to achieve these goals. Our simulations results show that the minimum weight color heuristic is robust to increases in communication density and yields results that are close to the optimal solution.

1 Introduction

The design of wireless sensor applications has received increased research attention in recent years. 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 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.

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 to ensure timely mobility, the *transmission* and *processing* of sensory data posses timeliness constraints.

The problem of scheduling *processing* tasks to meet timeliness constraints in a real-time sensor application was considered in [8]. In this paper, we focus on the *transmission* of sensory data. Traditional wireless network protocols, such as the CSMA/CA-based 802.11 family of protocols, cannot be directly used in time-sensitive sensor applications due to the following reasons. First, CSMA/CA networks do not completely eliminate the possibility of collisions despite 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 *blocking problem* as observed in [1]. In such networks, a node must explicitly request permission to transmit via a "request-to-sent" (RTS) and must receive a "clear-to-send" (CTS) acknowledgment before sending data. Further, all nodes in the vicinity that receive these messages must avoid transmission for the transmission duration and are thus *blocked*. Such a protocol can lead to *false blocking* and *blocking propagation* as observed in [12] and illustrated in Figure 1. As shown in the figure, node S_1 transmits data to node R_1 and node R_2 is blocked 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 $S_1 \to R_1$ and $S_3 \to R_3$ can occur in parallel.

*This research was supported in part by NSF grants CCR-9984030, CCR-0098060, CCR-0219520, EIA-0080119 and gifts from IBM, Microsoft and Intel.

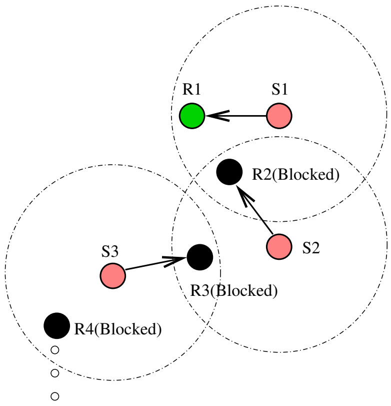


Figure 1. False blocking problem (blocking propagates from R_3 to R_3 to R_4)

In this paper, we consider techniques to avoid such problems in sensor applications. 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 time for transmitting a set of messages is minimized (by parallelizing non-interfering transmissions to the extent possible).

We argue that the problem is NP-complete and present heuristics based on edge coloring to address this problem. We discuss modifications to our approach to incorporate timeliness constraints. 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 paper is structured as follows. Section 2 presents our system model and the problem formulation. Sections 3 and 4 present our edge coloring-based heuristics and the optimal solution, respectively. Section 5 presents simulation results. Section 6 discusses how our scheme can be extended to handle deadline constraints. Section 7 presents related work and, finally, Section 8 summarizes our work.

2 Background and Problem Formulation

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

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

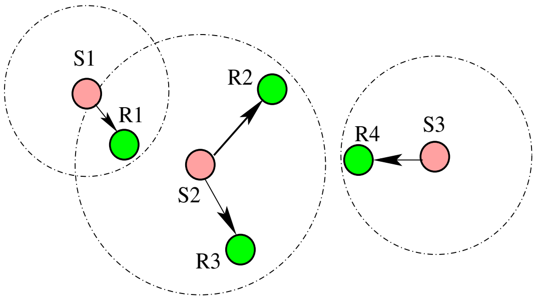


Figure 2. A communication example

- *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 such that the above constraints are satisfied and the time to complete all transmissions is minimized (by parallelizing non-interfering transmissions). We refer to this problem as the Optimal Parallel Communication Scheduling (OParCS) problem.

2.2 Communication Scheduler

In this work, we assume a centralized scheduler for scheduling message transmissions. This is a reasonable assumption for applications such as robotics since a centralized path planner is used to determine the movement for each robot as well as for the whole group. Consequently, the scheduler can run in conjunction with the planner to determine when each message should be transmitted.

The scheduler can be invoked periodically or on demand (like the scheduler studied in Spring System [11]). 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 to compute a new schedule. 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.

2.3 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.

The proof is given in [7].

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. However, determining the minimum number of colors is also NP-Complete [5].

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 notation used in this section is summarized in Table 1.

Notation	Meaning
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

Table 1. Notation

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. 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. 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. Figure 3 depicts the various steps in our heuristic.

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: $\sum_i W(c_i)$.

Figure 3. Edge coloring heuristic

3.2 Color Selection Policies

We propose three heuristics to choose a color from the palette for the selected edge.

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

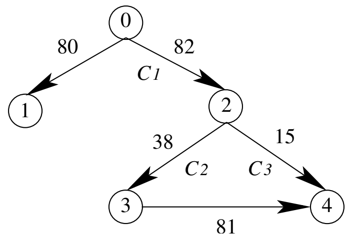


Figure 4. An example

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 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. 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: $W(c_1) + W(c_2) + W(c_3) + W(c_4) = 82 + 38 + 15 + 81 = 216$.

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 [3]. 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:

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

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:

$$\begin{aligned} h(n_G) &= max\,(W(v_i)) \\ &= max\left(\sum_j W(e^i_j)\right) \quad (2) \end{aligned}$$

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).

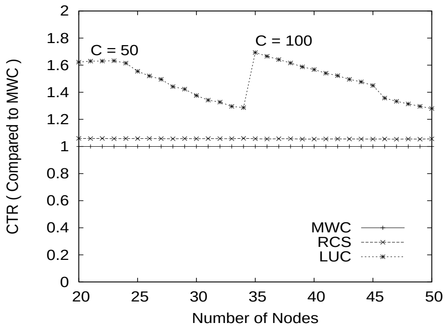


Figure 5. Effect of the number of nodes

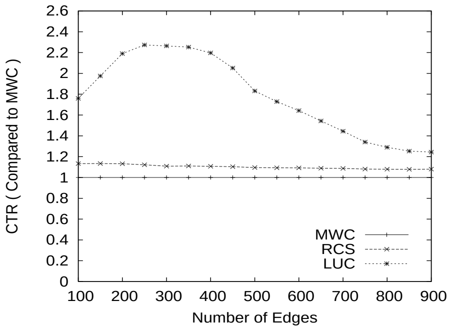


Figure 6. Effect of the number of edges

5 Simulation 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 $Comp(RCS/MWC) = Time(RCS)/Time(MWC)$, and if we use *Completion Time Ratio (CTR)* to denote the mean, then we have:

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

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.

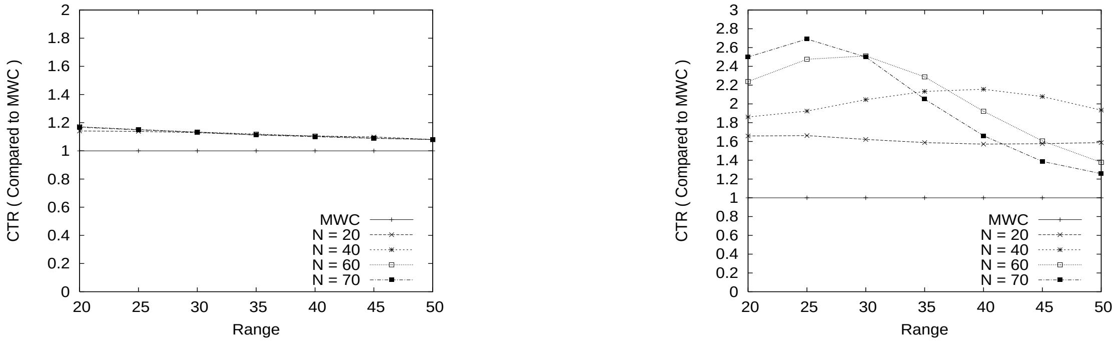
5.1.1 Effect of the Number of Nodes

To study the impact of the number of nodes (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 5 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. Random color selection (RCS) yields completion times that are within 8% of MWC, but the performance is not sensitive to the value of N. The least used color (LUC) heuristic has 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 fixed palette size, when N increases, the CTR decreases. This indicates if the initial palette size is closer to 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, $C = 50$ means that the initial number of colors in the palette is 50).

5.1.2 Effect of the Number of Edges

The number of edges reflects the communication density. Figure 6 plots the completion time ratio for the three heuristics as the number of edges varies. Again, the MWC outperforms the other two heuristics. Random color selection is about 10-18% worse and is not sensitive to different communication densities. For LUC, when 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.



(a) RCS vs MWC (b) LUC vs MWC

Figure 7. Effect of range

5.1.3 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, the initial palette is set to have 100 colors.

Figure 7 (a) compares the completion times of RCS and MWC, while Figure 7(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.

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.

5.2 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 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 [7]). Figure 8.(a) plots the CTR (normalized by the optimal solution); and Figure 8.(b) 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).

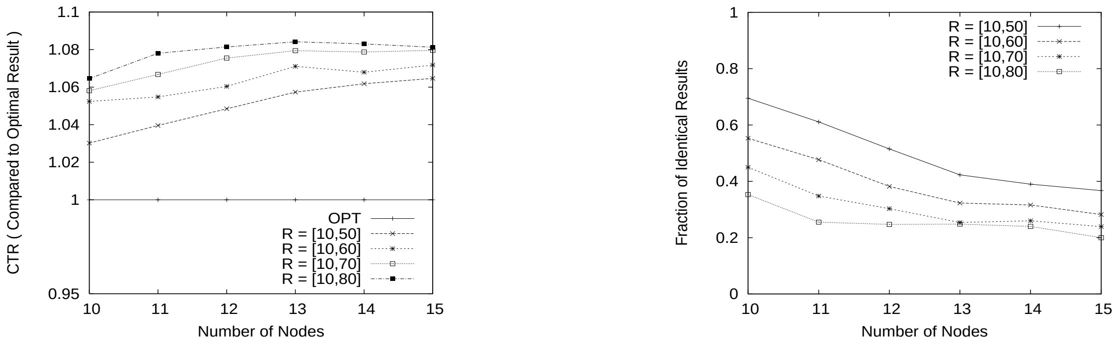
2. While the solution yielded by MWC is different from the optimal solution in a large fraction of the cases, this sub-optimal schedule is only about at most 6-8.5% worse for a variety of transmission ranges.

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.

5.3 Summary of Results

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 time into account when generating a schedule.



(a) Completion Time of MWC to OPT

(b) Fraction of Identical Results

Figure 8. MWC Versus OPT for varying transmission ranges

- 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.
- 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 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.

6 Incorporating Timeliness Constraints

Our heuristics so far have focused on scheduling messages in order 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.

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. Doing so orders 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 a brief discussion on how such a technique might work. The technique will need to balance three factors—the weight, the deadline, and the palette size 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 the order of their deadlines, such as EDF. If the opposite is true, the edges are chosen based on their palette sizes and then color is assigned on the deadlines and weights. A detailed exposition of these ideas is the subject of future work.

7 Related Work

Real-time issues that arise in various layers of the network stack in sensor networks are studied in [14]. 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.

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 includes 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. A EDF-based MAC protocol is exploited in a hexagonal cellular network architecture in [2], and the schedulability condition is given for a hybrid message set consisting of hard periodic and soft aperiodic messages.

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. 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 transmission costs and time constraints. Ramanathan [13] 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.

8 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 the sensor networks tested.

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.

9 Acknowledgment

We thank Shrikumar Hariharasubrahmanian, Zihui Ge and Zhengzhu Feng for various helpful discussions and comments. We also thank the anonymous referees for their comments.

References

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

[2] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *Proceedings of the IEEE Real-Time System Symposium (RTSS)*. IEEE, December 2002.

[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 *International Conference on Distributed Computing Systems (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 Parallel and Distributed Systems*, 13(10):1024–1037, October 2002.

[7] H. Li, P. Shenoy, and K. Ramamritham. Scheduling communication in real-time sensor application. Technical Report TR-04-05, University of Massachusetts at Amherst, January 2004.

[8] H. Li, J. Sweeney, K. Ramamritham, R. A. Grupen, and P. Shenoy. Real-time support for mobile robotics. In *IEEE Real Time Technology and Applications Symposium (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 *IEEE Real Time Technology and Applications Symposium (RTAS)*. IEEE, September 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] K. Ramamritham, J. A. Stankovic, and P. Shiah. Efficient scheduling algorithm for real-time multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, 1(2):184–194, April 1990.

[12] S. Ray, J. B. Carruthers, and D. Starobinski. RTS/CTS-induced congestion in ad hoc wireless LANs. In *Wireless Communications and Networking Conference (WCNC)*, March 2003.

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

[14] 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.