Optimal QoS Sampling Frequency Assignment for Real-Time Wireless Sensor Networks

Xue Liu, Qixin Wang, Lui Sha and Wenbo He
Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, IL 61801
Email: {xueliu, qwang4, lrs, wenbohe}@cs.uiuc.edu

Abstract

How to allocate computing and communication resources in a way that maximizes the effectiveness of control and signal processing has been an important area of research. The characteristic of a multi-hop Real-Time Wireless Sensor Network raises new challenges. First, the constraints are more complicated and a new solution method is needed. Second, we need a distributed solution to achieve scalability. This paper presents solutions to both of the new challenges. The first solution to the optimal frequency allocation is a centralized solution that can handle the more general form of constraints as compared with prior research. The second solution is a distributed version for large networks using a pricing scheme. It is capable of incremental adjustment when utility functions change.

1. Introduction

Real-Time Wireless Sensor Network (RTWSN) is expected to monitor dynamic phenomena in remote areas such as pollutants, animal habitats and seismic activities. Phenomena of interest will change at different places at different times. So do the set of utility functions associated with them. Instead of using worst case sampling frequencies that will work for all the phenomena, we would like to dynamically find the set of globally optimal sampling frequencies. Since the transmission rate is a function of sampling frequency, finding the optimal sampling frequencies provides a foundation for the optimal use of the wireless bandwidth and battery life of sensor networks.

Finding the frequencies that optimize QoS at the application level has been studied before. Finding the optimal control frequencies subject to schedulability constraint was first studied by Seto *et al.* [1] and by Sha *et al.* [2] under analog and digital controllers respectively. Rajkumar *et al.* [3] investigated QoS-based Resource Allocation Model (Q-RAM), which is capable of handling complex multiple quality dimensions.

However, RTWSN presents new challenges. Routes in a RTWSN may interleave with each other. The optimal sampling problem must take the traffic contention problems at each route into considerations. This makes the optimization problem have a mathematical structure that is harder than those studied before. We present an Interior Point Method based solution to show how the optimal frequencies can be found efficiently. However, a centralized method can only be used in a small network. In a large network, the control messages needed for optimization from all sensors will converge towards the central computing node. This creates bottleneck and makes centralized solution inappropriate for large RTWSN.

To dynamically find globally optimal sampling frequencies in a large network, we need a distributed solution that will generate balanced optimization traffic loads so as to avoid the creation of bottlenecks. In addition, the solution must be incremental, so that when the utility function at a few nodes change, near optimal global solutions can be found with smaller cost.

To focus on the optimization problem, we shall assume that routes are determined by some given algorithms such as AODV [4] or SPEED [5]. We also assume that certain reliable communication channels exist. How to tolerate device failures and message drops in the context of distributed optimization are future research issues beyond the scope of this paper.

To the best of our knowledge, works by Caccamo *et al.* [6] are the first to deal with multi-hop RTWSN. Architecturally, we adopt their cellular structure as illustrated in Figure 1. This network uses seven distinct radio frequencies. At the center of each cell, there is one wireless sensor device that also functions as a router. The router has one sender sending at the radio frequency type same as its cell's label. The specific geographical layout makes any two senders sending with the same radio frequency to be at least two cells apart; and each cell and its six neighbors are of distinct radio frequencies. Each router also has a receiver that

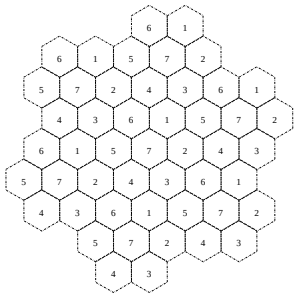


Figure 1. Cellular RTWSN

can receive any of the other 6 radio frequencies. The inter-cell communication uses a globally synchronized TDMA scheme.

However, it is very complicated to adjust bandwidth allocation among neighbors under TDMA scheme. What is more, TDMA needs global synchronization. On the other hand, we find CDMA [7, 8] can be deployed to provide better flexibility for the adjustment of inter-cell bandwidth. CDMA also provides benefits such as better security, better throughput etc. Therefore, in this paper, we replace the TDMA scheme with CDMA . Specifically, we use asynchronous CDMA[1] because it allows temporally and spatially overlapping real-time wireless channels to work independently. We name the proposed multi-hop real-time wireless sensor device/network architecture as Real-time Independent CHannels (RICH) architecture. Because of the independence between real-time channels, RICH RTWSN doesn't need global time synchronization; real-time schedulability constraints are very easy to analyze, and are only related to local information. The bandwidth differentiation among neighbors can be carried out at application level in a very flexible way. We will further describe the RICH RTWSN scheme in Section 2.

The rest of the paper is organized as follows. Section 2 describes our proposed RICH architecture, which can easily support multi-hop real-time networking. We also give the real-time schedulability constraints analysis in this section. An illustrated example of RICH RTWSN is presented to show the practicability of RICH. In Section 3, we formulate the optimal QoS sampling frequency assignment problem into a non-linear programming problem. Section 4 and 5 give centralized and distributed algorithm for the optimal QoS sampling frequency assignment problem respectively. In Section 6, we first compare the numerical solutions based on the example discussed in Section 2. We discuss in details the trade-offs between the centralized solution and distributed solution. Then we analyze the possible problems and solutions associated with both methods. Finally, conclusions and future work are discussed in Section 7.

1 Asynchronous CDMA is already the standard approach deployed in nowadays CDMA technology.

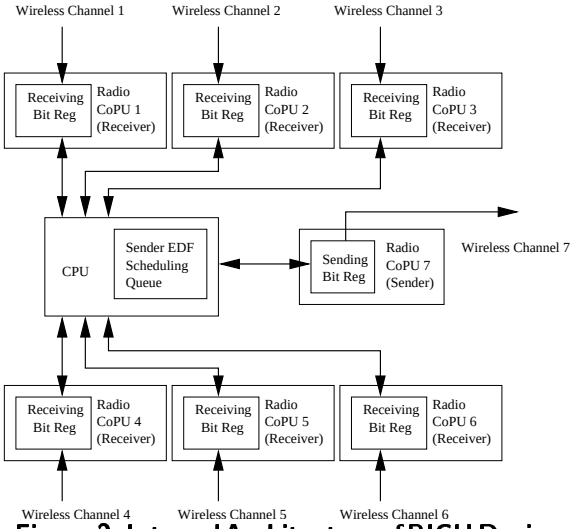


Figure 2. Internal Architecture of RICH Device

2. Supporting Multi-hop RTWSN

2.1. RICH Architecture

We assume our wireless devices (nodes) have the internal architecture illustrated by Figure 2. Each wireless node has seven radio co-processors (CoPUs), each operates at a distinct asynchronous CDMA channel (encoding sequences). Among which, six of the radio CoPUs are dedicated as receivers, and the remaining one is dedicated as the sender of the node. We allow the bandwidths of each receiver or sender to be distinct from each other. For the time being, let's suppose the singular sender sends its packets according to EDF real-time scheduling algorithm. A dedicated EDF scheduling queue is attached to it to buffer/schedule the outgoing packets. The CPU communicates with each receiver and sender co-processor by clock driven periodical polling.

Based on the internal architecture of RICH wireless devices, we can design the wireless network, which we call RICH RTWSN. We use the cellular geographical layout of Caccamo *et al.* [6]'s scheme, but instead of using seven frequencies, we deploy seven CDMA encoding sequences throughout the network. Each sender of a cell sends at the cell's labeled CDMA channel (encoding sequence). That is, sender of the node located in cell labeled "i" sends with the type i CDMA encoding sequence. The transmission range of every sender in our RICH RTWSN is one hop. The effective receiver is identified by the broadcast packet's "destination"

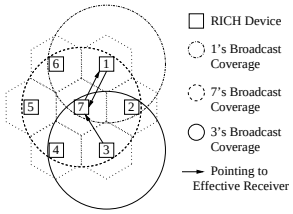


Figure 3. Concurrent Transmission in RICH RTWSN

data segment. Besides the sender, the six receivers on a node are dedicated to listening to the six other CDMA channels respectively. That is, each receiver is dedicated to listening to one of the cell's six neighbors. Because of asynchronous CDMA, inter-cell communications can now be carried out independently, i.e. parallelly and asynchronously. For example, as shown in Figure 3, node in cell 7 can broadcast to its 6 neighbors (with cell 1 as the *effective receiver*) while *at the same time* listening (receiving) from neighbor 1 and 3.

2.2. Schedulability Analysis of RICH RTWSN

Without loss of generality, consider the scenario depicted in Figure 3, where the dashed circle around node 7 represents a channel whose sender is the singular sender of node 7, and the channel also involves the six receivers that are listening to node 7 on nodes 1,2,3,4,5,6 respectively. Let the bandwidth of the sender and the six receivers to be $B_s, B_{r1}, B_{r2}, \ldots, B_{r6}$. For a specific channel *chnl*, we define the *bandwidth of the channel*, represented by B_{chnl}, to be $B_{chnl} = \min\{B_s, B_{r1}, B_{r2}, \ldots, B_{r6}\}$. Note under RICH RTWSN scheme the mapping between channel and its sending node is a one to one mapping. For ease of expression, *we also say the (channel's sending) node has a bandwidth of* B_{chnl}. Note the bandwidth of the *sender* of the sending node of the channel is B_s.

For an RICH node, the real-time scheduling is carried out in the EDF scheduling queue attached to its singular sender. Let $\mathcal{T}$ be the set of all routes that need to go through it. For a route $\tau \in \mathcal{T}$, suppose it has a sampling/reporting frequency of f_τ (reports per second), and each report is a packet of length l_τ. When there is no contention from other routes, each report should be delivered to the receiving hop within time l_τ / B_{chnl}. When there are multiple contending routes in one channel, the physical media of the channel should be regarded as a *non-preemptive* resource. Because if one packet starts transmitting, it cannot be preempted by any other packet until it is completely transmitted. Hereby, the real-time scheduler of RICH node's EDF scheduling queues should be *non-preemptive EDF scheduler* to ensure both the EDF behavior and non-preemptive usage of the channel medium. A sufficient fine-grain schedulability bound for the non-preemptive EDF scheduler is described as follows [9][2]:

$$\sum_{\tau \in \mathcal{T}} l_\tau f_\tau + L_j f_j \leq B_{chnl}, \text{ for each } j \in \mathcal{T} \qquad (1)$$

Where B_{chnl} is the channel bandwidth, and $L_j = \max_{\{\tau \in \mathcal{T} \text{ and } \tau \neq j\}} \{l_\tau\}$.

An RICH node can also simultaneously function as an end point sensor by multiplexing CPU time. We furthermore assume that each end point sensor can *initiate* at most *one* route. Hence the schedulability criterion for the sampling co-processor is:

$$f_j \leq f_j^{max} \qquad (2)$$

Where f_j is the sampling/reporting frequency of route j, and f_j^{max} is the maximal sampling frequency of the route's end point sensor node specified by sensor device manufacturer. As long as (2) is satisfied, the sampling co-processor can be encapsulated, and be analyzed as a receiver. Hereby the whole node can still be analyzed the same way as an intermediate router. Thus, by analyzing each node of the RICH RTWSN according to inequality set (1) and inequality (2), we can derive a set of linear inequalities, which is the fine-grain sufficient real-time schedulability constraint set for RICH RTWSN sampling frequency assignment. We can summarize them into the following form:

$$\begin{cases} \mathbf{A} f \leq W \\ f \leq f^{max} \end{cases} \qquad (3)$$

Where $f = (f_1, f_2, \ldots, f_N)^\mathsf{T}$ is the vector of sampling/reporting frequencies assigned to each of the N routes. $f^{max} = (f_1^{max}, f_2^{max}, \ldots, f_N^{max})^\mathsf{T}$ is the maximal sampling frequency for the N end point sensors (see (2)). Matrix $\mathbf{A}$ and vector W are obtained by node-wise analysis according to (1), which reflect the specific routing topology of the RICH RTWSN. Suppose these schedulability analyses generate M inequalities in total, then $\mathbf{A} \in \mathbb{R}^{M \times N}$, $W \in \mathbb{R}^{M \times 1}$.

Besides the constraints from real-time schedulability, there are often application specific minimum sampling/reporting frequency requirements. These extra requirements can be written as:

$$f \geq f^{min} = (f_1^{min}, \ldots, f_N^{min})^\mathsf{T} \qquad (4)$$

2 According to [9], the bound can be even more fine-grained

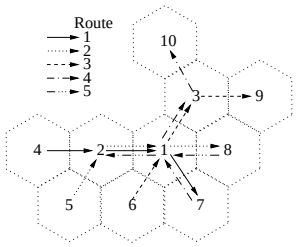


Figure 4. A RICH RTWSN Schedulability Analysis Example

Inequality set (3) and (4) constitute a complete set of real-time schedulability constraints for RICH RTWSN sampling frequency allocation. An example is given as follows:

Example 1 *As depicted by Figure 4, a RICH RTWSN consists of 10 nodes (as labeled* $1, 2, \ldots, 10$*) and 5 routes (identified with different arrow styles): Route 1:* $4 \rightarrow 2 \rightarrow 1 \rightarrow 7$*; Route 2:* $5 \rightarrow 2 \rightarrow 1 \rightarrow 8$*; Route 3:* $6 \rightarrow 1 \rightarrow 3 \rightarrow 9$*; Route 4:* $7 \rightarrow 1 \rightarrow 3 \rightarrow 10$*; Route 5:* $8 \rightarrow 1 \rightarrow 2$*.*

Let $B_i (i = 1, \ldots, 10)$ *be the channel bandwidth of node i (note, this is not the bandwidth of the sender co-processor of node i, but the bandwidth of the corresponding channel which includes the singular sender of node i and six neighboring nodes' receivers). Suppose the sampling frequency assigned to Route j to be* $f_j (j = 1, \ldots, 5)$*. Let* l_j*,* f_j^{min} *and* $f_j^{max} (j = 1, \ldots, 5)$ *be the reporting packet size, minimal frequency and maximal frequency requirements for Route j.*

According to the topology in Figure 4, we have the following real-time schedulability constraints:

Node 1: Route 1, 2, 3, 4, 5 are passing through it, hence:

$$\begin{array}{rcl}
(l_1 f_1 + l_2 f_2 + l_3 f_3 + l_4 f_4 + l_5 f_5) + \max\{l_2, l_3, l_4, l_5\} f_1 & \leq & B_1 \\
(l_1 f_1 + l_2 f_2 + l_3 f_3 + l_4 f_4 + l_5 f_5) + \max\{l_1, l_3, l_4, l_5\} f_2 & \leq & B_1 \\
(l_1 f_1 + l_2 f_2 + l_3 f_3 + l_4 f_4 + l_5 f_5) + \max\{l_1, l_2, l_4, l_5\} f_3 & \leq & B_1 \\
(l_1 f_1 + l_2 f_2 + l_3 f_3 + l_4 f_4 + l_5 f_5) + \max\{l_1, l_2, l_3, l_5\} f_4 & \leq & B_1 \\
(l_1 f_1 + l_2 f_2 + l_3 f_3 + l_4 f_4 + l_5 f_5) + \max\{l_1, l_2, l_3, l_4\} f_5 & \leq & B_1
\end{array}$$

Node 2: Route 1, 2 are passing through it, hence:

$$\begin{array}{rcl}
(l_1 f_1 + l_2 f_2) + l_2 f_1 & \leq & B_2 \\
(l_1 f_1 + l_2 f_2) + l_1 f_2 & \leq & B_2
\end{array}$$

As the destination end for Route 5, there are no constraints (the corresponding constraints are analyzed at node 1, Route 5's last sending hop).

Node 3: Route 3,4 are passing through it, hence:

$$\begin{array}{rcl}
(l_3 f_3 + l_4 f_4) + l_4 f_3 & \leq & B_3 \\
(l_3 f_3 + l_4 f_4) + l_3 f_4 & \leq & B_3
\end{array}$$

Node 4: As a node along Route 1, we have $l_1 f_1 \leq B_4$*.*

Node 5: As a node along Route 2, we have $l_2 f_2 \leq B_5$*.*

Node	Bandwidth(B_i)	Max Sampling Capability (f^{max})
1	1.0	-
2	0.6	-
3	0.4	-
4	0.25	($f_1^{max} =$)30
5	0.25	($f_2^{max} =$)25
6	0.25	($f_3^{max} =$)30
7	0.2	($f_4^{max} =$)40
8	0.15	($f_5^{max} =$)30

Route	Required Min Freq. (f_j^{min})	Affordable Max Freq. (f_j^{max})	Report Packet Size (l_j)
1	11	30	0.01
2	2.5	25	0.015
3	5	30	0.02
4	1	40	0.025
5	2	30	0.03

Table 1. Parameter Values for Example 1

Node 6: As a node along Route 3, we have $l_3 f_3 \leq B_6$*.*

Node 7: As a node along Route 4, we have $l_4 f_4 \leq B_7$*. As the destination end for Route 1, there are no constraints (the corresponding constraints are analyzed at node 1, Route 1's last sending hop).*

Node 8: As a node along Route 5, we have $l_5 f_5 \leq B_8$*. As the destination end for Route 2, there are no constraints (the corresponding constraints are analyzed at node 1, Route 2's last sending hop).*

Node 9 and Node 10: As purely destination end for routes, there are no constraints (corresponding constraints are analyzed at the corresponding routes' last sending hops).

In addition to all above, because of minimal sampling requirements, we have: $f_j \geq f_j^{min}$*, where* $(j = 1, \ldots, 5)$*. The complete frequency allocation constraints are hereby:* $\mathbf{A}f \leq W$*,* $f \leq (f_1^{max}, \ldots, f_5^{max})^{\mathsf{T}}$ *and* $f \geq (f_1^{min}, \ldots, f_5^{min})^{\mathsf{T}}$*, which are detailed by (5).*

Suppose the numerical values of parameters are as shown in Table 1, then the complete frequency allocation constraints are as shown in (6).

3. Optimizing QoS in WSN with Real-time Constraints — Math Modeling

In this section, we model the optimal sensor frequency allocation problem as an optimization problem, using constraints set (3)(4) from the previous section.

3.1. Utility Loss Index

Assume each end point sensor is the source of a route that periodically reports sampling results. Let the sampling/reporting frequency for the jth route be f_j. For most applications, the higher the sampling/reporting frequency f_j, the higher is the QoS. This is captured

$$
\begin{cases}
\mathbf{A} = \begin{bmatrix}
l_1 + \max\{l_2, l_3, l_4, l_5\} & l_2 & l_3 & l_4 & l_5 \\
l_1 & l_2 + \max\{l_1, l_3, l_4, l_5\} & l_3 & l_4 & l_5 \\
l_1 & l_2 & l_3 + \max\{l_1, l_2, l_4, l_5\} & l_4 & l_5 \\
l_1 & l_2 & l_3 & l_4 + \max\{l_1, l_2, l_3, l_5\} & l_5 \\
l_1 & l_2 & l_3 & l_4 & l_5 + \max\{l_1, l_2, l_3, l_4\} \\
l_1 + l_2 & l_2 & 0 & 0 & 0 \\
l_1 & l_1 + l_2 & 0 & 0 & 0 \\
0 & 0 & l_3 + l_4 & l_4 & 0 \\
0 & 0 & l_3 & l_3 + l_4 & 0 \\
l_1 & 0 & 0 & 0 & 0 \\
0 & l_2 & 0 & 0 & 0 \\
0 & 0 & l_3 & 0 & 0 \\
0 & 0 & 0 & l_4 & 0 \\
0 & 0 & 0 & 0 & l_5
\end{bmatrix} \\
f = (f_1, f_2, f_3, f_4, f_5)^{\mathsf{T}} \\
W = (B_1, B_1, B_1, B_1, B_1, B_2, B_2, B_3, B_3, B_4, B_5, B_6, B_7, B_8)^{\mathsf{T}}
\end{cases}
\tag{5}
$$

$$
\begin{cases}
\mathbf{A}f = \begin{bmatrix}
.04 & .015 & .02 & .025 & .03 \\
.01 & .045 & .02 & .025 & .03 \\
.01 & .015 & .05 & .025 & .03 \\
.01 & .015 & .02 & .055 & .03 \\
.01 & .015 & .02 & .025 & .055 \\
.025 & .015 & 0 & 0 & 0 \\
.01 & .025 & 0 & 0 & 0 \\
0 & 0 & .045 & .025 & 0 \\
0 & 0 & .02 & .045 & 0 \\
.01 & 0 & 0 & 0 & 0 \\
0 & .015 & 0 & 0 & 0 \\
0 & 0 & .02 & 0 & 0 \\
0 & 0 & 0 & .025 & 0 \\
0 & 0 & 0 & 0 & .03
\end{bmatrix}
\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix}
\leq W = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ .6 \\ .6 \\ .4 \\ .4 \\ .25 \\ .25 \\ .25 \\ .2 \\ .15 \end{bmatrix}, \\
f \leq f^{max} = (30, 25, 30, 40, 30)^{\mathsf{T}}, \text{ and } f \geq f^{min} = (11, 2.5, 5, 1, 2)^{\mathsf{T}}
\end{cases}
\tag{6}
$$

by a *Utility Loss Index* (ULI) function by Seto *et al.* [1] in the following form:

$$U_j(f_j) = \omega_j \alpha_j e^{-\beta_j f_j} \tag{7}$$

Where f_j is the sampling frequency for task j (route j), ω_j, α_j and β_j are application specific constants. In this paper, the form of ULI function is further generalized to be any strictly decreasing differentiable convex function with regard to frequency f.

3.2. Mathematical Formulation

We assume each individual ULI function $U_j(f_j)$ is strictly decreasing differentiable and convex. Assume ULIs are additive, hence the system's overall ULI is the sum of the ULIs of individual routes: $\sum_{j=1}^{N} U_j(f_j)$. Our performance optimization problem becomes:

$$\min_{(f_1,\ldots,f_N)} \quad \sum_{j=1}^{N} U_j(f_j) \tag{8}$$

$$\text{s.t.: } \mathbf{A}f \leq W \tag{9}$$

$$f \leq f^{max} \tag{10}$$

$$f \geq f^{min} \tag{11}$$

Where $\mathbf{A}$ is a constraint matrix with dimension $M \times N$. M is dependent upon the routing topology of the RTWSN, and N is the number of total routes. We call this problem *Multiple Constraints Optimization Problem* (MCOP) in contrast to Seto *et al.* [1]'s *Single Constraint Optimization Problem* (SCOP), and we denote the former as $MCOP(U, \mathbf{A}, W)$.[3]

The feasible set of MCOP is compact and convex and $U_j(f_j)$ is differentiable and convex, then MCOP has optimal solutions [11]. Furthermore, if $U_j(f_j)$ is strictly convex, the optimal solution is unique [11].

When $M = 1$, the ULIs are in the negative exponential form, and there is no constraint set (10), $MCOP(U, \mathbf{A}, W)$ becomes SCOP as follows [1]:

$$\min_{(f_1,\ldots,f_N)} \quad \sum_{j=1}^{N} U_j(f_j) = \sum_{j=1}^{N} \omega_j \alpha_j e^{-\beta_j f_j} \tag{12}$$

$$\text{s.t.: } Af \leq W \tag{13}$$

$$f \geq f^{min} = (f_1^{min}, \ldots, f_N^{min})^{\mathsf{T}} \tag{14}$$

where W is bandwidth (utilization) constraint bound, and $A \in \mathbb{R}^{1\times N}, f \in \mathbb{R}^N, W \in \mathbb{R}^N$.

Based on Kuhn-Tucker condition, Seto *et al.* [1] provides an algorithm to solve SCOP problem analytically. MCOP is a generalization of SCOP. We will show

3 Notation used in [10].

that the approach for deriving an analytical solution to SCOP is not viable for solving MCOP. To this end, we first prove that the optimal solution f^* of MCOP will make at least one of the constraints in constraint sets (9) and (10) becomes equality constraint and show why it is not viable to tackle the MCOP in an analytical fashion similar to [1]. In $MCOP(U, \mathbf{A}, W)$, constraint set (9) and (10) can be combined as:

$$\mathbf{A}' f \leq W',$$
$$\text{where } \mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \mathbf{I} \end{bmatrix}_{(M+N)\times N}, W' = \begin{bmatrix} W \\ f^{max} \end{bmatrix}_{(M+N)\times 1} \quad (15)$$

Theorem 1 *MCOP's optimal solution f^* must ensure that at least one of the $(M+N)$ constraints $A'_i f^* \leq W'_i, i = 1, \ldots, (M+N)$ reaches equality. i.e., $\exists i \in \{1, \ldots, M+N\}$ such that $A'_i f^* = W'_i$. Here A'_i is the ith row of $\mathbf{A}'$.*

Proof: Due to the space limit, we omit the proof here. Readers interested in the proof can see [12]. ■

Though we know for the optimal frequency assignment f^* there is at least one i such that $A'_i f^* = W'_i$, we do not know how many constraint inequalities reach equality and which i(s) make the equality. This makes the Kuhn-Tucker based solution method not applicable. Compared with the SCOP case where $M = 1$, paper [1]'s problem is much easier because there is only one non-trivial constraint $A_1 f^* \leq W_1$, and it is exactly this constraint that should reach equality. Besides [1], Rajkumar *et al.* [3] proposed a numerical solution. But that solution is also for single constraint scenario. In their more recent works [13, 14], Rajkumar *et al.* studied the scenario of multiple constraints. However, the problem they study is an integer programming problem, which is different from ours. In [14], the integer programming problem is proven to be NP-Hard. Several semi-optimal algorithms are proposed. According to [13], the one that scales well is Hierarchical Q-RAM. However, that algorithm requires the division of multiple constraints into independent groups, which is impractical for our multi-hop RTWSN scenario (see Section 6.3). Fortunately, as will become clear, MCOP can be solved with the state-of-art Interior Point Methods [15, 16] and Internet Pricing Schemes [17, 10, 18].

4. Centralized Solution Method for MCOP

In this section, we apply *Interior Point Method* (IPM) to solve the MCOP problem for RTWSN.

Definition 1 *A* Constrained Optimization Problem *is expressed as* $\min\{f(x) : x \in Q \subseteq \mathbb{R}^n\}$, *where the constraint set Q is defined by multiple equalities and inequalities: $Q = \{x : h(x) = 0, g(x) \leq 0\}$, where $h : \mathbb{R}^n \to \mathbb{R}^p; g : \mathbb{R}^n \to \mathbb{R}^q$.*

Definition 2 *A* Convex Optimization *(CO) problem is a constrained optimization problem whose objective function $f(x)$ is continuous and convex, and whose constraint set Q is compact (i.e. closed and bounded) and convex.*

It's easy to see that an MCOP is a constrained convex optimization problem with linear constraints. For solving a convex optimization problem, the major difficulties come from the multiple inequality constraints. Closed form solutions are generally unavailable. However, IPMs [15, 16] can solve linear constraint convex optimization problems numerically. IPM is a numerical method that iterates in the interior of the solution space defined by constraint set Q to find the optimal solution. IPMs can be further divided into two sub-categories: *primal methods* and *primal-dual methods*. Primal-dual methods try to solve the primal and dual optimization problems [19] together. In practice, the primal-dual methods are more efficient. The advantages of interior-point method based numerical solution includes: (1) Efficient. IPMs give the correct solution very fast; (2) Multi-hop application scenarios: The objective function needs not to be confined as exponential form as in paper [1], but can be general strictly decreasing differentiable convex functions.

To solve the MCOP problem, we use optimization library COPL_LC [20]. It is easy to transform our MCOP problem to the form used by COPL_LC. The transformation method can be found in the extended version of this paper [12]. To implement the IPM based centralized solution, the whole RICH RTWSN elects a central computing node **C**, which gathers ULI and constraints information from all the network, carries out the optimization algorithm, and returns the final results.

5. Distributed Algorithms for Optimal Frequency Assignment

However, a direct application of IPM results in a centralized solution that requires collecting data from each node. This will create a traffic bottleneck around the central computing node (detailed discussion is in Section 6.3). To overcome the bottleneck problem, we give a distributed algorithm for solving the MCOP. The distributed algorithm let routers and routes' end point nodes collaborate to find the optimal frequencies. The algorithm is based on the recent researches of Internet pricing schemes [17, 10, 18], especially [17].

The main idea is to impose a price on each constraint in (9)$\sim$(11). Each route will accumulate its relevant constraints' prices and solve a local optimization problem based on its own ULI function. The result is the next proposed route frequency (to simplify, we call it "route frequency" or "frequency" in the following context). The route frequency is then delivered to each of the route's routers, where the route's relevant constraints will (imagine each constraint as an agent) update their prices accordingly. This procedure works in an iterative manner until it converges. The distributed algorithms has two main attributes:

1. It converges to the optimal frequencies of MCOP (Theorem 2).
2. Each route's computation is only based on local information.

Notations used in the iterative algorithm:
s is the algorithm's iteration step, $s = 0, 1, \ldots$.
$p(s)$ is the updated constraint price vector for each constraint $i, i \in \{1, \ldots, M\}$ at iteration step s. $p(s) = (p_1(s), \ldots, p_M(s))$.
$f(s)$ is the updated route frequency vector for each route $j, j \in \{1, \ldots, N\}$ at iteration step s. $f(s) = (f_1(s), \ldots, f_N(s))^{\mathsf{T}}$.

This algorithm stops until the predefined convergence criterion is reached. For example, we can use one of the following criteria or the combination of them.

1. $\|f(s) - f(s-1)\|_n \leq \epsilon$
2. $|\sum_{j=1}^{N} U_j(f_j(s)) - \sum_{j=1}^{N} U_j(f_j(s-1))| \leq \epsilon$

where $\epsilon > 0$ is a sufficiently small real number. $\|v\|_n$ denotes the nth-norm of vector $v = (v_1, \ldots, v_k)$. If $n = 1$, $\|v\|_1 \equiv \max(v_i), i \in \{1, \ldots, k\}$. If $n \in \mathbb{Z}^+$, then $\|v\|_n \equiv (\sum_{i=1}^{k} v_i^n)^{\frac{1}{n}}$.
Distributed Algorithm: The distributed algorithm has two parts. One is the *constraint algorithm*, and the other is *route algorithm*. Both algorithms are run in each iteration. At the beginning of the algorithm, set $f(0) = f^{min}$, and $p(0) > 0$.
(1) Constraint algorithm During iteration $s = 1, 2, \ldots$, for each constraint $i (i = 1, , M)$:

1. Receives route frequencies $f_j(s)$ from all relevant route j. Route j and constraint i are relevant if $\mathbf{A}_{ij} \neq 0$.
2. Computes a new constraint price for itself using the following price update equation:

$$p_i(s+1) = [p_i(s) + \gamma(f^i(s) - W_i)]^+ \tag{16}$$

   Here $f^i(s) = A_i f(s)$, and A_i is the ith row of $\mathbf{A}$. Function $[\bullet]^+$ is defined as $[x]^+ \equiv \max\{x, 0\}$, where x is a real number.
3. Delivers new price $p_i(s+1)$ to all routes which are relevant to constraint i.

(2) Route algorithm
During iteration $s = 1, 2, \ldots$, for each route $j (j = 1, \ldots, N)$

1. Receives from the network the weighted sum of all the constraints' prices $p^j(s)$ imposed on this route:

$$p^j(t) = \sum_{i=1}^{M} p_i(s) \mathbf{A}_{ij} \tag{17}$$

2. Update the route frequency $f_j(s+1)$ for the next iteration according to local optimization of:

$$\begin{array}{ll} \min & U_j(f_j) + f_j p^j(s) \\ \text{s.t.:} & f_j^{min} \leq f_j \leq f_j^{max} \end{array}$$

$$i.e.\ f_j(s+1) = \arg \min_{f_j^{min} \leq f_j \leq f_j^{max}} (U_j(f_j) + f_j p^j(s)) \tag{18}$$

Now we prove the convergence and correctness of the above iterative algorithm. The proof follows Low's result in [17].
Assumptions:
A1: The feasibility condition holds for each constraint $i (i = 1, \ldots, M)$, such that $\sum_{j=1}^{N} \mathbf{A}_{ij} f_j^{min} \leq W_i$.
A2: For each route j, on the interval $I_j = [f_j^{min}, f_j^{max}]$, the utility function U_j is *strictly decreasing, strictly convex, and twice continuously differentiable.*
A3: The curvatures of U_j for each route satisfies the following condition on $I_j = [f_j^{min}, f_j^{max}]$, $\exists \bar{\alpha}_j$, s.t. $U_j''(f_j) \geq \frac{1}{\bar{\alpha}_j} > 0$, for all $f_j \in I_j$.
Notations in Theorem 2:
Define $L(j) = \sum_{i=1}^{M} \mathbf{A}_{ij}$. It is the column sum of $\mathbf{A}$.
Define $\bar{L} = \max_{j=1,\ldots,N}\{|L(j)|\}$, which is the maximum absolute value of column sum of $\mathbf{A}$.
Define $S(i) = \sum_{j=1}^{N} \mathbf{A}_{ij}$. It is the row sum of $\mathbf{A}$.
Define $\bar{S} = \max_{i=1,\ldots,M}\{|S(i)|\}$, which is the maximum absolute value of row sum of $\mathbf{A}$.
We also define $\bar{\alpha} = \max_{j=1,\ldots,N}\{\bar{\alpha}_j\}$. *i.e.* $\bar{\alpha}$ is the upper bound on $\frac{1}{U_j''(f_j)}, j = 1, \ldots, N$.

Theorem 2 *Suppose assumptions **A1** $\sim$ **A3** hold and the step size γ satisfy $0 < \gamma < 2/(\bar{\alpha}\bar{L}\bar{S})$. Then starting from any initial frequencies $f^{min} \leq f(0) \leq f^{max}$ and prices $p(0) \geq 0$, the sequence $\{(f(s), p(s))\}$ generated by the above distributed algorithm will converge to a accumulation point (f^*, p^*), and f^* is the solution of $MCOP(U, \mathbf{A}, W)$.*

Proof: See [12]. ■

Route	α_j	β_j	ω_j
1	0.66	0.3	1
2	0.66	1.0	2
3	0.66	0.5	3
4	0.66	0.7	4
5	0.66	0.3	5

Table 2. Parameters for ULI in Example 1

6. Comparison of Centralized and Distributed Algorithm

In this section, we shall discuss the tradeoffs between centralized algorithm and distributed algorithm, and show which is more appropriate in different situations. Specifically, we identify the bottleneck problem of centralized algorithm and the potential live lock problem of distributed algorithm and propose solutions to these problems. We also show that distributed algorithm is scalable and have the desirable incremental adjustment property.

6.1. Numerical Example of Centralized and Distributed Algorithm

We now apply the centralized and distributed algorithms to the scenario discussed in Example 1 of Section 2.2. In this example, we have 5 routes. The ULI function for each route j is in the form of $\omega_j \alpha_j e^{-\beta_j f_j}$, so the $MCOP(U, \mathbf{A}, W)$'s total ULI (i.e. the objective function) is: $\sum_{j=1}^{5} U_j(f_j) = \sum_{j=1}^{5} \omega_j \alpha_j e^{-\beta_j f_j}$, with parameters shown in Table 2. Constraint sets are shown in equation (6).

Using the centralized algorithm and the COPL_LC package [20], we see by 14 iterations, we can get the optimal solution of: $f^*_{central} = (12.35, 6.58, 5.70, 5.73, 5.00)^{\mathsf{T}}$, with an optimal value of 0.917. Using the distributed algorithm, we choose the initial iteration value of $f(0) = f^{min}, p(0) = (1,1,1,1,1,1,1,1,1,1,1,1,1,1)$ and $\gamma = 0.8$. We run the simulation for 500 steps, the simulation result shows the algorithm converges. The route frequency vector update process is shown in Figure 5. The converged frequency value is: $(12.34, 6.58, 5.70, 5.73, 5.01)^{\mathsf{T}}$ which conforms to the results got from centralized IPM based method.

In order to give a feeling of how fast the distributed algorithm converges, we did the following Monte Carlo simulation. We define the "frequency error" at each route j with regard to iteration s as: $e\_f_j(s) = |f_j(s) - f_j^*|$, where $f_j(s)$ is the frequency for route j at iteration s; f_j^* is the optimal frequency for this route. Consider the following form of $MCOP(U, \mathbf{A}, W)$ problem.

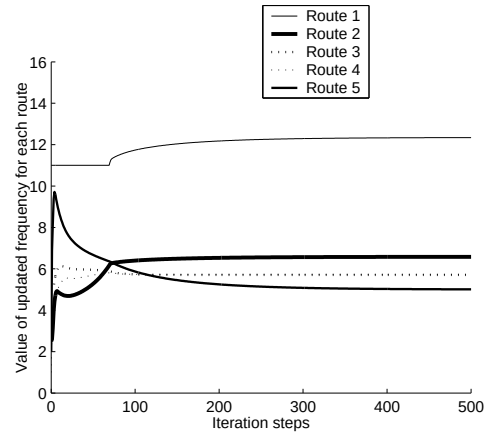


Figure 5. Updated Freq. v.s. Iteration Steps

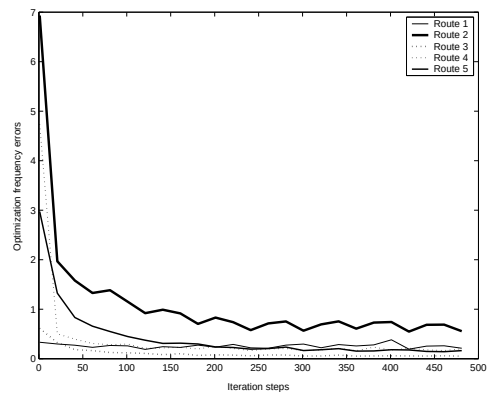


Figure 6. Rate of Frequency Error Reduction

$$\begin{aligned} \min_{(f_1,\ldots,f_N)} \sum_{j=1}^{N} U_j(f_j) &= \sum_{j=1}^{N} \omega_j \alpha_j e^{-\beta_j f_j} \\ \text{s.t.: } \mathbf{A} f &\leq W \\ f &\leq f^{max} \\ f &\geq f^{min} \end{aligned}$$

We show by Monte Carlo simulation that the distributed algorithm converges very fast in the sense that "frequency error" approaches to 0 in a negative exponential form. We pick $N = 5$ and the value of $\mathbf{A}$ is given in Example 1. In each run of the sample of simulation, we first choose a new ULI function by changing the coefficients of ω_j, α_j and β_j randomly, we then run the distributed algorithm long enough to get the optimal frequency vector f^*. Next, we collect the frequency error for step $s = 1, \ldots, 500$. Suppose we run L such random simulations and each simulation's ULI is selected randomly. We average all the L simulations' frequency error $e\_f_j(s)$ at each step $s = 1, \ldots, 500$. Figure 6 shows a result of the simulation for $L = 800$.

We can see from Figure 6 that the frequency error

Route	α_j	β_j	ω_j
1	0.33	0.3	4
2	0.22	0.2	3
3	1.32	0.5	2
4	1.98	0.7	1
5	0.66	0.3	6

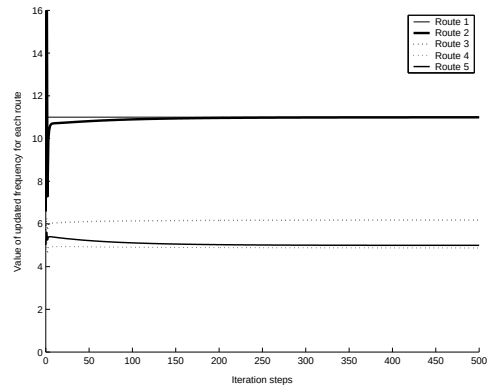
Table 3. New $MCOP(U', \mathbf{A}', W')$ Parameters

converges very fast, usually within 500 steps, and the frequency error convergence conforms to a negative exponential form. It is worth noting that in real application scenario, by using the "incremental adjustment" property discussed in Section 6.2, the distributed algorithm usually converges even faster.

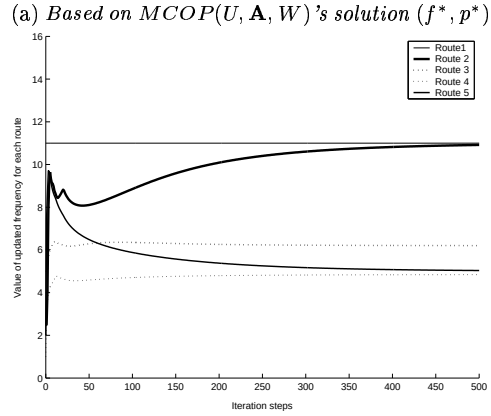
6.2. Incremental Adjustment Property of Distributed Algorithm

In real-world applications, there are often times that ULI functions and constraint set parameters would change. For example, in target tracking application, the speeds, importance or number of monitored targets may change, causing the ULIs to change with them. These changes will transform the original optimization problem $MCOP(U, \mathbf{A}, W)$ into a new optimization problem with new parameters $MCOP(U', \mathbf{A}', W')$, hence the optimal frequency f^* has to be re-calculated. If we use the distributed algorithm, we already have the solution pair (f^*, p^*) for the original problem, we can now continue the iteration based on (f^*, p^*) to get the new optimal solution f'^* quickly. We call this property "incremental adjustment property". Below we give an example to show how effective it is to use "incremental adjustment".

Continue with the example in Section 6.1. The new ULI parameters' changes are shown in Table 3. Figure 7 shows the comparison between using the $MCOP(U, \mathbf{A}, W)$'s optimal solution pair (f^*, p^*) and a constant value pair (f^0, p^0) as the initial state for the solution of the new optimization problem $MCOP(U', \mathbf{A}', W')$. Here $f^0 = f^{min}$ and $p^0 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$. We see in Figure 7(a) if we carry out the new computation based on the previous solution (f^*, p^*), after around 150 iterations, the frequency converges to the new optimal frequency vector f'^*. On the other hand, if we still use the (f^0, p^0) to carry out the new computation in the distributed algorithm, the convergence to new optimal frequency vector f'^* is much slower, and requires around 400 iterations as shown in Figure 7(b).



(a) *Based on $MCOP(U, \mathbf{A}, W)$'s solution (f^*, p^*)*



(b) *Based on constant initial value (f^0, p^0)*

Figure 7. Illustration of Incremental Adjustment Property

6.3. Control Traffic and Scalability Analysis for Distributed and Centralized Algorithm

In this section, we shall give the (optimization) control traffic analysis for both distributed and centralized algorithm. We show centralized algorithm has inferior scalability because it suffers the bottleneck problem when the network is large while the distributed algorithm does not. This is shown by analyzing the maximal number of bytes of control payloads passed through each node in the RTWSN in both algorithms.

In order to show this, we shall first look at how the distributed algorithm should be implemented on a real-world RTWSN.

Distributed Algorithm Implementation Steps:

Step 1: Each constraint's price is updated by the router that creates that constraint. During each iteration s, in the constraint algorithm stage, each router will calculate the price for its created constraints based on (16). Next, each of these updated prices must be propagated to all the relevant routes. To do that, each route's end point sensor sends an empty packet to-

ward the destination end. The packet's payload is just one floating-point number (*i.e.* 4 bytes), dedicated to carry the total price $p^j(s)$, where j refers to the jth route. As this packet travels toward the destination along the route, on each hop, it will accumulate onto $p^j(s)$ all relevant constraints' prices maintained by the local router. When the packet reaches the destination end, the total price $p^j(s)$ is obtained. This packet is then returned along the same route to the source end sensor.

Step 2: After Step 1, each source end sensor carries out the route algorithm (18) to update the route frequency. Then the source end sensor sends another packet towards the destination end to notify every router along this route the updated route frequency. This packet's payload is also only one floating-point number (*i.e.* 4 bytes), which is enough to carry the updated route frequency value.

Traffic Load Analysis for Distributed Algorithm:
Let $\mathcal{N}$ be the set of all nodes in an RICH RTWSN. Let ϕ_i^{dis} be the accumulated control traffic in terms of number of bytes of payloads passing through node $i(i \in \mathcal{N})$ under the distributed algorithm. *Let Φ^{dis} be the maximal accumulated control traffic payload bytes passing through any of the nodes, i.e.* $\Phi^{dis} = \max_{i\in\mathcal{N}}\{\phi_i^{dis}\}$.

Let $\mathcal{R}$ be the set of all *routers* in the same RICH RTWSN. Let d_r be the number of routes passes through router $r(r \in \mathcal{R})$, *i.e.* the out-degree of router r. Let D be the maximal number of routes passes through any of the routers, *i.e.* $D = \max_{r\in\mathcal{R}}\{d_r\}$.

During each iteration's Step 1, totally $2d_r$ packets pass through each router r, and 2 packets pass through each end point sensor node (one send and one receive). During Step 2, totally d_r packets pass through each router r, and 1 packet for each end point sensor node. Since each packet's payload length is 4 bytes. Hereby the total number of bytes of payloads pass through each router r during each iteration is $12d_r$, and for an end point sensor node it is 12. Since a node in the RICH RTWSN must be a router, or end point sensor, or idle node, we have the following proposition:

Proposition 1 *For any node $i \in \mathcal{N}$, during each distributed algorithm iteration, the number of control packet payload bytes passing through it is no more than* $12D$.

According to our simulation results, the distributed algorithm usually converges or reaches very good approximation in $K \leq 500$ steps (If we use the incremental adjustment property discussed in Section 6.2, usually the number of iterations required is even less). Hence after all the iterations, the accumulated control traffic payload passing through any node is no more than $12KD$. That is, for any node $i \in \mathcal{N}$, $\phi_i^{dis} \leq 12KD \leq 12 \times 500D = 6000D$. Therefore we have:

$$\Phi^{dis} \leq 6000D \tag{19}$$

$$i.e.\ \Phi^{dis} = O(D) \tag{20}$$

Traffic Load Analysis for Centralized Algorithm:
Let $\mathcal{N}$ be the set of all nodes in an RICH RTWSN. Let ϕ_i^{cen} be the accumulated control traffic in terms of bytes of payloads passing through node $i(i \in \mathcal{N})$ under the centralized algorithm. *Let Φ^{cen} be the maximal accumulated control traffic payload bytes passing through any of the nodes, i.e.* $\Phi^{cen} = \max_{i\in\mathcal{N}}\{\phi_i^{cen}\}$.

Suppose the total number of *routes* in an RICH RTWSN is Γ^{total}. Under centralized algorithm, each route at least needs to send the central computing node $\mathbf{C}$ its ULI information together with at least one constraint. Without loss of generality, we suppose each ULI function is expressed by 3 floating point numbers (*i.e.* 12 bytes) and each constraint is at least represented by 2 floating point numbers (*i.e.* 8 bytes). Thus the accumulated control traffic payload at node $\mathbf{C}$ is $\phi_{\mathbf{C}}^{cen} \geq 20\Gamma^{total}$, *i.e.* $\phi_{\mathbf{C}}^{cen} = \Omega(\Gamma^{total})$. Because $\Phi^{cen} \geq \phi_{\mathbf{C}}^{cen}$, we hereby have:

$$\Phi^{cen} = \Omega(\Gamma^{total}) \tag{21}$$

One may argue that the routes in an RICH RTWSN may not be all directly or indirectly connected, but rather can be partitioned into several disjoint "maximal connected sub-graphs". Under that case, there need not to be *ONE* central computing node. Each maximal connected sub-graph of the routing topology can elect its own central computing node, which takes charge of the optimal sampling frequency planning for and only for the routes within that specific maximal connected sub-graph. In this case, let $\mathcal{G}$ be the set of all maximal connected sub-graphs, for a specific maximal connected sub-graph $g \in \mathcal{G}$, let Γ_g be the number of routes in g. Let $\mathbf{C}_g$ be the elected central computing node for g. Because of the same reason how we derived (21), we have $\phi_{\mathbf{C}_g}^{cen} \geq 20\Gamma_g$. By the definition of Φ^{cen}, we still have $\Phi^{cen} \geq \phi_{\mathbf{C}_g}^{cen} \geq 20\Gamma_g$. That is: $\forall g \in \mathcal{G}, \Phi^{cen} \geq 20\Gamma_g$, which implies $\Phi^{cen} \geq \max_{g\in G}\{20\Gamma_g\}$. Let $\Gamma = \max_{g\in G}\{\Gamma_g\}$, which is the maximal number of directly or indirectly connected routes, then we have:

$$\Phi^{cen} \geq 20\Gamma$$

$$i.e.\ \Phi^{cen} = \Omega(\Gamma) \tag{22}$$

In the following simulation for wide area monitoring, we shall show $\Gamma \approx \Gamma^{total}$, i.e. (22) is empirically equivalent to (21)[4]. We shall also show Φ^{dis} is empirically insensitive to the scale of the RICH RTWSN while Φ^{cen}

4 The formal proof is left for our future research.

increases at least quadratically with the scale of the RICH RTWSN. *This means the centralized algorithm's central computing node is a control message exchanging bottleneck.* Hence centralized algorithm does not scale up well while the distributed algorithm does.

Scalability Comparison of Distributed and Centralized Algorithm in Wide Area Monitoring Scenario:

Suppose we are monitoring a square shaped wide area. The area has an edge length of $l(km)$. The monitored area deploys an RICH RTWSN cellular division with a hexagon cell edge length of $0.1km$. Suppose the monitoring observers are uniformly distributed across the square area and their density is $\rho = 10/km^2$. Each observer can randomly pick a cell to monitor. Therefore, there are totally $l^2\rho$ routes between observers and their picked cells. In addition, we reasonably require every route in the network to have a total number of hops (excluding the destination node) of no more than 10, *i.e.*, no observer can observe a cell more than 10 cells (hops) away. We deploy a simple geographical routing protocol of randomly picking the next hop, under the requirement that the packet is always getting closer to the destination in distance.

We carry out the following simulation. For each RICH RTWSN scale ($l = 5, \ldots, 100(km)$) we run 30 trials. In each trial, we randomly generate $l^2\rho$ routes as described above. After that, we count the maximal number of routes passing through any node (*i.e.* D), and the maximal number of directly or indirectly connected routes (*i.e.* Γ). As we can see from Figure 8, D *is bounded by a relatively small constant, insensitive to how large* l *is, while* Γ *soars up in a roughly* l^2 *speed.*

According to (20) and (22), $\Phi^{dis} = O(D)$ and $Phi^{cen} = \Omega(\Gamma)$. This means if we deploy distributed algorithm, the maximal accumulated control traffic passing through any of the nodes (*i.e.* Φ^{dis}) is upper bounded by D and D is insensitive to the scale of the network. However, if we deploy centralized algorithm, the maximal accumulated control traffic passing through any of the nodes (*i.e.* Φ^{cen}) is lower bounded by Γ and Γ is growing quadratically with the network scale l. The underlying reason is that in centralized algorithm, the control traffic is bottlenecked at the central computing node, while in distributed algorithm, the control traffic is roughly evenly distributed among all the nodes. Hence, the centralized algorithm is not scalable to large-scale networks.

We see from Section 6.1 that centralized algorithm works efficiently when the network scale is small, while from the above analysis, distributed algorithm is more appropriate when the network scale is large. It is useful to set up a threshold to determine when to switch from centralized to distributed algorithm in real-world

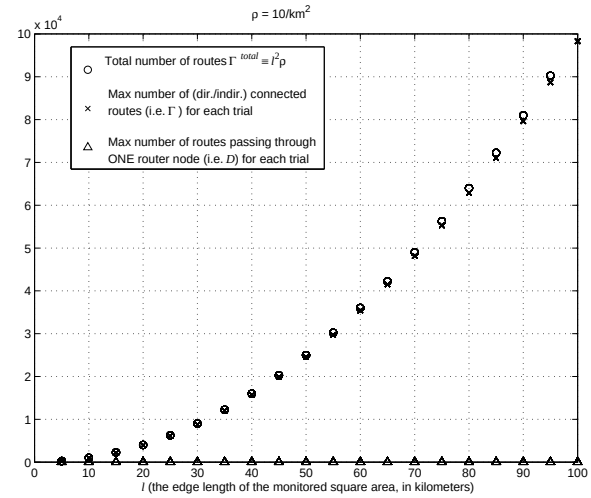


Figure 8. Scalability Example

deployment. Take the above application for example, as shown by Figure 8, D is always no more than 15 while Γ is never less than 6000 when l is bigger than $25km$. According to (19) $\Phi^{dis} \leq 6000D = 90000$. According to (22) $\Phi^{cen} \geq 20\Gamma = 120000$. That is, after l is bigger than $25km$, we will have $\Phi^{dis} << \Phi^{cen}$, which means distributed algorithm is more desirable.

However, we should note that there is a possible "live lock" problem for the distributed algorithm. In the distributed algorithm, each route needs to communicate with all its routers to exchange the updated constraint price and route frequency. Ideally, we want to send control messages in background using the spare bandwidth. We call this approach the "best-effort" approach since the bandwidth assigned for background traffic is not guaranteed.

As we proved in Theorem 1, when the optimal frequency $f^* = (f_1^*, \ldots, f_N^*)^\mathsf{T}$ is reached, at least one of the constraints will reach equality. This means at the router where that constraint is created, all the bandwidth is used up by the sampling/reporting traffic, and *no control messages can be sent through that router any more*! This causes a "live lock" problem since if there is future need for exchanging control messages, the saturated router can no longer participate.

A solution to the "live lock" problem is to preserve a small amount of dedicated bandwidth for exchanging the control messages. According to Proposition 1, the maximal amount of control message payload bytes passes through each RICH RTWSN node during each iteration is no more than $12D$, where D is the maximal number of routes passing through any router in the whole RTWSN. Figure 8 shows when the maximal route length and density of routes in the RTWSN are fixed, and the end points of routes are uniformly

distributed, D can be empirically regarded as being bounded by a constant, which can be estimated via simulation. Therefore, the bandwidth to be reserved for control message exchange can be planned accordingly.

7. Conclusions and Future Work

In this paper, we first propose a sensor device/network architecture – Real-time Independent CHannels (RICH), which can easily realize multi-hop Real-Time Wireless Sensor Networks (RTWSN). More importantly, we study the optimal QoS in RICH RTWSN, and formalize it into a non-linear optimization problem with total Utility Loss Index as the objective function and end point sensor sampling/reporting frequencies as the maneuverable variables. By using the state-of-art methods in optimization, two solutions are given. One is in a centralized fashion, the other is in a distributed fashion. Our solutions can handle multi-hop routing scenarios, which is not covered by previous research. We compare the trade-offs between the centralized and distributed algorithms under different situations. Specifically, we quantitatively analyze the node-wise control traffic under both algorithms. We show that though centralized algorithm works efficiently with small-scale RTWSN, it has a bottleneck problem which limits its scalability. On the other hand, distributed algorithm is a better choice for large-scale RTWSN and has the desirable incremental adjustment property. Also, the convergence of the distributed algorithm is guaranteed and empirically shown to be fast.

Our on-going research topics include: (1) Integrating QoS optimization and error modeling for WSN; (2) Theoretical analysis of distributed algorithm's convergence rate under specific WSN applications; (3) ULI function formulation based on stochastic models.

Acknowledgement

The authors would like to thank Jiawei Zhang and Professor Yinyu Ye at Stanford University for the help of providing and using the COPL_LC package. We would also like to thank Professor Steven Low at Cal Tech for the discussion on distributed algorithm.

References

[1] D. Seto *et al.*, "On task schedulability in real-time control systems," in *Proc. of IEEE Real-Time Systems Symposium'96*, 1996.

[2] L. Sha *et al.*, "Online control optimization using load driven scheduling," in *Proc. of the 39th IEEE Conf. on Decision and Control*, Dec. 2000.

[3] R. Rajkumar *et al.*, "A resource allocation model for qos management," in *Proc. of IEEE Real-Time Systems Symposium*, 1997.

[4] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, Feb. 1999.

[5] T. He *et al.*, "Speed: A stateless protocol for real-time communication in sensor networks," in *Intl. Conf. on Distributed Computing Systems (ICDCS 2003)*, Providence, RI, May 2003.

[6] M. Caccamo *et al.*, "An implicit prioritized access protocol for wireless sensor networks," in *Proc. of IEEE Real-Time Systems Symposium'02*, Dec. 2002.

[7] D. Gerakoulis and E. Geraniotis, *CDMA: Access and Switching.* John Wiley & Sons, 2001.

[8] K. Onodera. Low power techniques for high-speed wireless systems (draft 1.1 ph.d. dissertation) — chp 3 ds-cdma system overview. U. C. Berkeley. [Online]. Available: http://kabuki.eecs.berkeley.edu/~keitho/

[9] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications.* Kluwer Academic Publishers, 1997.

[10] F. Kelly *et al.*, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, Nov. 1998.

[11] D. Bertsekas, *Nonlinear Programming.* Belmont, MA: Athena Scientific, 1995.

[12] X. Liu, Q. Wang, L. Sha, and W. He. (2003) Optimal qos sampling frequency assignment for real-time wireless sensor networks (extended version). Dept. of Computer Sci., UIUC. [Online]. Available: http://www-rtsl.cs.uiuc.edu/papers/rtss03optimal_extended.ps

[13] S. Ghosh *et al.*, "Scalable resource allocation for multi-processor qos optimization," in *Proc. of the 23rd IEEE Intl. Conf. on Distributed Computing Systems (ICDCS 2003)*, Providence, RI, May 2003.

[14] C. Lee *et al.*, "A scalable solution to the multi-resource qos problem," in *Proc. of the IEEE Real-Time Systems Symposium*, Dec. 1999.

[15] Y. Nesterov and A. Nemirovsky, *Interior Point Polynomial Methods in Convex Programming.* SIAM, 1994.

[16] Y. Ye, *Interior Point Algorithms: Theory and Analysis.* Wiley, 1997.

[17] S. H. Low and D. E. Lapsley, "Optimization flow control, i: Basic algorithm and convergence," *IEEE/ACM Tran. on Networking*, vol. 7, no. 6, pp. 861–75, Dec. 1999.

[18] F. Kelly, "Charging and rate control for elastic traffic," *European Tran. on Telecommunications*, vol. 8, 1997.

[19] D. Luenberger, *Linear and Nonlinear Programming.* Reading, Massachusetts: Addison-Wesley, 1984.

[20] Y. Ye. (1997, Sept.) User's guide of *copl_lc*, computational optimization program library: Linearly constrained convex programming. [Online]. Available: http://www.stanford.edu/~yyye/Col.html