Optimal Real-Time Sampling Rate Assignment for Wireless Sensor Networks

Xue Liu, Qixin Wang, Wenbo He, Marco Caccamo and Lui Sha
University of Illinois at Urbana-Champaign

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, a distributed solution is needed to achieve scalability. This paper presents solutions to both of the new challenges. The first solution to the optimal rate 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 sensor networks using a pricing scheme. It is capable of incremental adjustment when utility functions change. This paper also presents a new sensor device/network backbone architecture – Real-time Independent CHannels (RICH), which can easily realize multi-hop Real-Time Wireless Sensor Networks.

Categories and Subject Descriptors: C.2.2 [**Computer-communication Networks**]: Network Protocols—*Applications*; C.3 [**Special-purpose and Application-based Systems**]: Real-time and Embedded Systems—*Applications*

General Terms: Algorithms, Design, Performance, Experimentation

Additional Key Words and Phrases: Sensor Network, Real-Time Wireless Sensor Network, Optimization, Pricing, Distributed Algorithms

1. INTRODUCTION AND RELATED WORK

Real-Time Wireless Sensor Network (RTWSN) is expected to carry out various applications such as remote control or video/audio monitoring in *ad hoc* environments. Instead of using conservative (lowest allowed) sampling/actuating rates (since sampling and actuating rate allocation are similar, unless explicitly denoted, “sampling rate” is used instead of “sampling/actuating rate” in the following), a sampling rate allocation that maximizes global utility while maintaining real-time

---

Author’s address: Xue Liu, Qixin Wang, Wenbo He, Marco Caccamo and Lui Sha, Department of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, IL 61801. Email: {xueliu, qwang4, wenbohe, mcaccamo, lrs }@cs.uiuc.edu.

This research is supported by (listed in alphabetical order) MURI N00014-01-0576, NSF ANI 02-21357, NSF CCR 02-09202, and ONR N00014-02-1-0102. The first two authors are also supported by Saburo Muroga Fellowship and Vodafone Fellowship respectively.

Part of this work has been published at 24th IEEE Real-Time Systems Symposium [Liu et al. 2003].

schedulability is wanted.

Resource allocation has been studied in general for computing systems [Kurose and Simha 1989; Waldspurger and Weihl 1994; Bolot et al. 1994]. Recently, the problem of resource allocation and congestion control in network has been studied together by Kelly and Low et al. [Kelly et al. 1998; Kelly 1997; Low and Lapsley 1999]. However, these works do not consider real-time constraints, and therefore cannot be directly applied to real-time systems.

Stoica *et al.* [1996] have studied a proportional share resource allocation algorithm for real-time, time-shared system. The scheme is for single processor scheduling and is more focused on fairness of the scheduling algorithms rather than system optimality. Finding the optimal control rates subject to schedulability constraints was first studied by Seto *et al.* [1996] and by Sha *et al.* [2000] for analog and digital controllers respectively. An offline solution method is given based on Kuhn-Tucker condition. However, the schedulability analysis is still for single processor. Rajkumar *et al.* [1997] investigated QoS-based Resource Allocation Model (Q-RAM), which is capable of handling complex multiple quality dimensions. But the solution can only be used with single constraint scenario. In the following works, Lee [1999] and Ghosh [2003] *et al.* studied the scenario under multiple constraints. However, the problem they studied is an integer programming problem, which is different from the model we will discuss in this paper. In [Lee et al. 1999], the integer programming problem is proven to be NP-Hard. Several sub-optimal algorithms are proposed. According to [Ghosh et al. 2003], 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 multi-hop RTWSN.

RTWSN presents new challenges for real-time resource allocation. Routes in a RTWSN may interleave with each other. The sampling rate optimization must take into consideration the traffic contention at each router. This makes the optimization problem harder than those studied before. We present an *Interior Point Method* (IPM) based solution to show how the optimal rates can be found efficiently. On the other hand, though a centralized method is usually efficient for small and moderately large RTWSNs, it may not scale well for very large RTWSNs, because control messages for optimization converge at the central node and create a bottleneck. To dynamically find the global optimum in a very large network, a distributed solution is needed to generate balanced optimization control traffic that avoids bottlenecks. In addition, the solution shall be incremental, so that when the utility functions at a few nodes change, updated optimum can be found with small cost.

To the best of our knowledge, works by Caccamo *et al.* [2002] are the first to provide real-time support for multi-hop RTWSN. In [Caccamo et al. 2002], a cellular base station layout is deployed as the backbone for the whole RTWSN, as shown in Fig. 1. The base station network uses seven non-overlapping *Radio Frequency* (RF) bands. At the center of each cell, there is one base station, which also functions as a router. A router in a cell labeled i has a single transmitter that always transmits at RF band i, and a single receiver that receives from one neighbor at a time (i.e. listens to one of the six neighbors' RF bands at a time). All RF broadcasts are one-hop. The specific geographical layout makes each base station and its six neighbors transmits with distinct RF bands, and any two base stations sending with

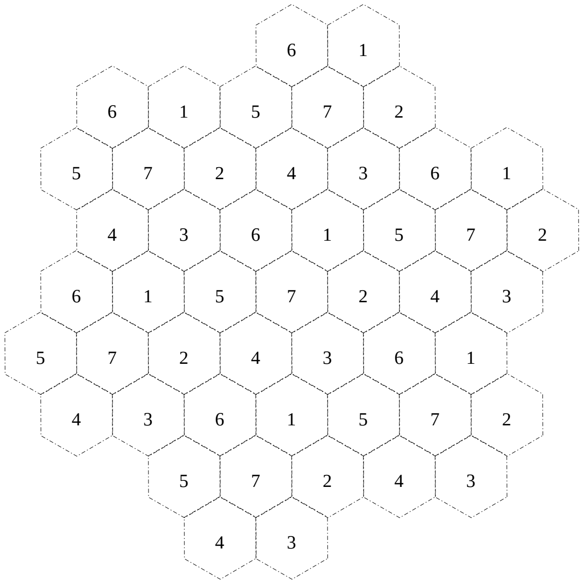


Fig. 1. A mixed FDMA-TDMA base station backbone layout

the same RF band are at least two cells apart. The inter-cell communication uses a globally synchronized TDMA scheme. Specifically, all base stations' receivers listen to their northeast neighbors at time slot 1, listen to east neighbors at time slot 2, so on and so forth. Therefore, the inter-base station communication is a mixed FDMA-TDMA scheme. More recently, based on the mixed FDMA-TDMA scheme, Giannecchini *et al.* [2004] provide an online *suboptimal* approximation algorithm (CoRAl) to dynamically reconfigure sensing rates of RTWSN. CoRAl runs fast but only applies to exponential performance loss function. It is worth mentioning that inside of each cell, there can be randomly distributed wireless slave sensors with more constrained capabilities, which does the actual sensing and communicates with the cell's base station at other RF bands (which do not interfere with inter-base station communications). *The intra-cell communication is not the focus of this paper*, and therefore intra-cell sensors are not plotted in this paper's figures.

We adopt Caccamo *et al.* [2002]'s cellular base station backbone layout. In this paper, *we focus on the scenario that data flows among backbone base stations are unicast flows.* More sophisticated routing topologies such as multicast and convergecast are beyond the scope of this paper[1]. Also, we assume routes are already decided by given algorithms, for example SPIN [Heinzelman et al. 1999], GPSR [Karp and Kung 2000], GEAR [Xu et al. 2001], SPEED [He et al. 2003] or Rumor Routing [Braginsky and Estrin 2002] etc.[2]; and the total data bandwidth of each one-hop wireless link is adjusted so that the wireless medium is reliable enough for real-time communication (according to information theory and CDMA thoery [Viterbi 1995]

---

[1] Intra-cell communications between slave sensors and the base station are more often convergecast, but is not the focus of this paper.

[2] A good survey of routing protocols in sensor networks is given in [Akkaya and Younis 2005]

(also see Appendix A), if the worst case wireless medium condition is given, there is an upper bound on data bit throughput so that the data bit error rate can be maintained below an upper bound). How to incorporate routing and tolerate device failures and message drops into our sampling rate optimization are future research issues beyond the scope of this paper. According to information theory, the inter-base station bandwidth available is determined by wireless medium quality and reception quality requirements, and is fundamentally irrelevant with specific multiple access schemes, such as FDMA, TDMA or CDMA. The mixed FDMA-TDMA multiple access scheme proposed in Caccamo *et al.* [2002] is more complicated for real-time schedulability analysis, mainly because TDMA provides poorer isolation between wireless links. Therefore, in this paper, we propose a mixed FDMA and *Direct Sequence Spread Spectrum CDMA* (DSSS-CDMA)[3] scheme called *Real-time Independent CHannels* (RICH), which provides better isolation between wireless links and is easier to analyze. A brief tutorial on DSSS-CDMA is provided in Appendix A.

The major contributions of this paper are:

- Study and model the optimal rate assignment problem in RICH multi-hop RTWSN using real-time schedulability analysis and non-linear optimization.
- Using the state-of-art methods in optimization, two solutions to the optimal rate assignment problem are given. One is in a centralized fashion using Interior Point Method, the other is in a distributed fashion based on pricing scheme.
- Compare the trade-offs between the centralized and distributed algorithms under different situations.

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 example of RICH RTWSN is presented to show the practicability of RICH. In Section 3, we formulate the optimal QoS sampling rate assignment problem into a non-linear programming problem. Section 4 and 5 give centralized and distributed solution for optimal QoS sampling rate assignment 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 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.

2. SUPPORTING MULTI-HOP RTWSN

2.1 RICH Architecture

We assume our wireless base stations (the so-called RICH base stations) have the internal architecture illustrated by Fig. 2. Each RICH base station has seven DSSS-CDMA modulation/demodulation co-processors (CoPUs), each operates with a distinct DSSS-CDMA *Pseudo Noise* (PN) sequence at a distinct FDMA RF band. Among which, six of the DSSS-CDMA CoPUs are receivers, and the other one is the only transmitter of the base station. We allow the data bit bandwidths of

---

[3] Nowadays, the term CDMA usually refers to DSSS-CDMA.

each DSSS-CDMA CoPU to be distinct. For the time being, suppose the singular transmitter sends packets according to *Earliest Deadline First* (EDF) scheduling algorithm. A dedicated EDF scheduling queue is attached to it to buffer/schedule the outgoing packets. In addition, a RICH base station also includes a sensing CoPU, or an intra-cell communication CoPU that gathers data from intra-cell slave sensors. The CPU interacts with each CoPU by periodical polling.

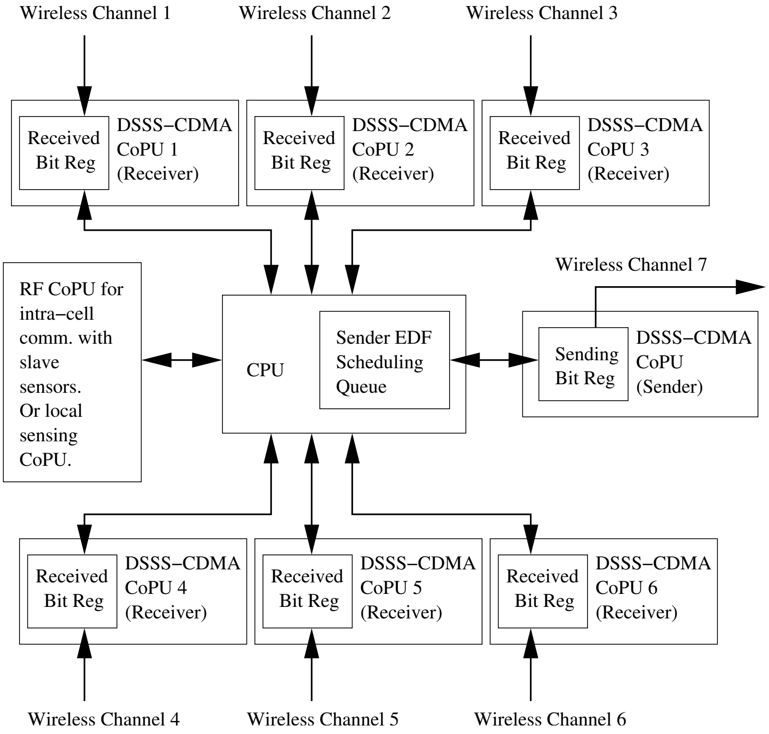


Fig. 2. Internal Architecture of a RICH Base Station

We adopt the cellular base station layout of Caccamo *et al.* [2002]'s (see Fig. 3), and maintain the seven RF band coloring of the cells. Meanwhile, we deploy forty-nine DSSS-CDMA PN sequences, denoted as $A1, \ldots, A7$, $B1, \ldots, B7$, $C1, \ldots, C7$, $D1 \ldots, D7$, $E1, \ldots, E7$, $F1, \ldots, F7$, $G1, \ldots,$ and $G7$ respectively. In a cell labeled XY, the RICH base station transmitter deploys the XYth PN sequence for DSSS-CDMA modulation, and transmits at the Yth RF band. For example, the base station in a cell labeled $G7$ transmits with the $G7$th DSSS-CDMA PN sequence at the 7th RF band. The transmission range of every transmitter in our RICH RTWSN is one-hop. Each of the six receivers on a RICH base station listens to one of its one-hop neighbor's transmission. Take the RICH base station at a cell labeled $A5$ for example, its six receivers listens to the $6, 7, 4, 1, 3, 2$th RF band respectively, and demodulates with DSSS-CDMA PN sequence $A6, A7, A4, G1, F3, F2$ respectively.

Under such design, the broadcast of a base station is simultaneously received by its six one-hop neighbors. The effective receiver is designated by the broadcast

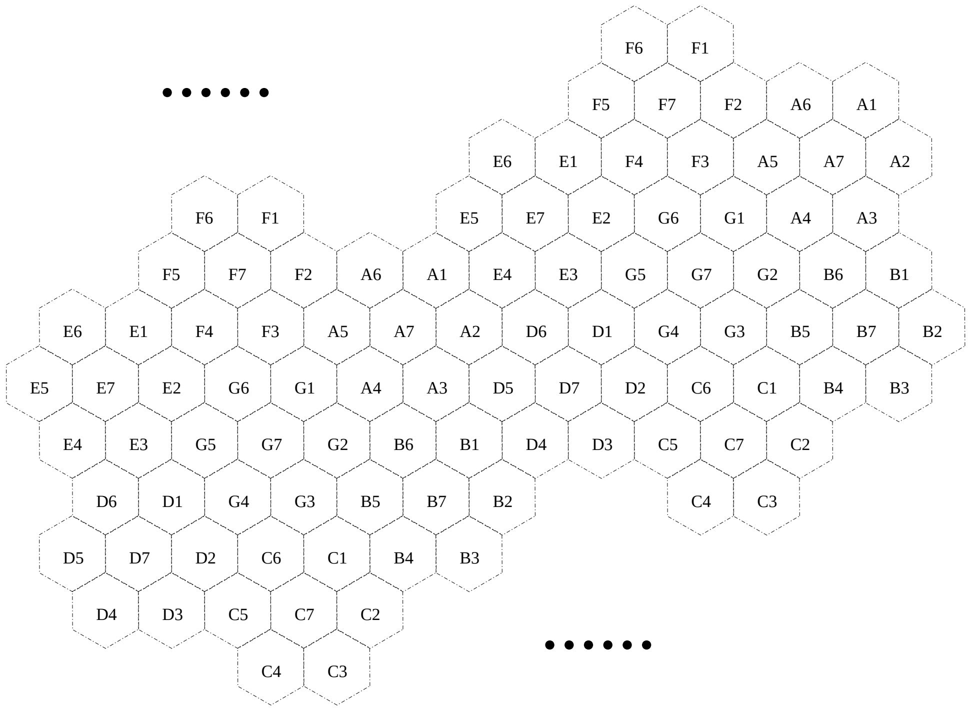


Fig. 3. The mixed FDMA-CDMA base station backbone layout

packet's "destination" data segment. More importantly, because of the deployment of DSSS-CDMA, any transmission can be carried out independently. For example, in Fig. 3, an $F2$ base station and an $A6$ base station can both send packets to an $A5$ base station (their common one-hop neighbor) at anytime. Furthermore, the schedule can be independently adjusted to be per base station specific. For example, the $F2$ base station can dedicate 100% of its time sending to the $A5$ base station; at the same time, the $A6$ base station can dedicate $\frac{1}{3}$ of its time to the $A5$ base station. There is no synchronization requirements between any pair of transmissions. Under TDMA, however, this is impossible. For example, $F2$'s sending schedule must not overlap with $A6$'s. Such mutual exclusive relationship propagates throughout the network, finally all base stations' broadcast schedules are inter-locked, which complicates analysis and reconfiguration.

Also, the layout guarantees two base stations transmiting with same DSSS-CDMA PN sequence (and therefore at the same RF band) are at least eight hops away. This implies that *a base station does not have to be at the center of its cell.*

2.2 Schedulability Analysis of RICH RTWSN

The broadcast of a RICH base station is overheard by all its six neighbor base stations. Usually, the wireless medium to the six neighbors are irregular [Zhou et al. 2004; Zhao and Govindan 2003]. According to DSSS-CDMA theory (see Appendix A), given RF band, worst case wireless medium conditions and maximum acceptable bit error rate, the upper bound of data bit bandwidth is determined, which we call *affordable bandwidth.* Suppose for a RICH base station X, because of the irregularity of wireless medium, the affordable bandwidths to its six neighboring RICH base stations are B_1, B_2, …, B_6. We set the transmission data bit bandwidth of X to be $B = \min\{B_1, B_2, \ldots, B_6\}$. Therefore the broadcast of X is reliably received by all its six neighbors, *i.e.* the bit error rate of one-hop transmission is always below the maximum acceptable bound. In another word, B models factors such as the impact of radio irregularity on the wireless medium.

For a RICH base station, the real-time scheduling is carried out in the EDF scheduling queue attached to its singular transmitter. Let $\mathcal{T}$ be the set of all routes that go through it. For a route $\tau \in \mathcal{T}$, suppose it has a sampling/reporting rate of f_τ, and each report is a packet of length l_τ. The corresponding transmission time of the packet is therefore $c_\tau = l_\tau/B$. When there are multiple contending routes through one RICH base station, the transmitter should be regarded as a *non-preemptive* resource. Because once a packet starts transmitting, it cannot be preempted until completely transmitted. Hereby, the real-time scheduler of a RICH base station's EDF queue should be *non-preemptive EDF scheduler* to ensure both the EDF behavior and non-preemptive usage of the broadcast link. Under such scheduling, a packet (job) can be blocked by at the most one other packet. Therefore we can apply the well-known schedulability bound for the non-preemptive EDF scheduler as follows [Buttazzo 1997]:

$$\sum_{\tau \in \mathcal{T}} c_\tau f_\tau + C_j f_j \le 1, \text{ for each } j \in \mathcal{T}, \tag{1}$$

where C_j is the maximum blocking time for sending a packet of route j.

$$\begin{aligned} C_j &= \max_{\{\tau \in \mathcal{T} \text{ and } \tau \neq j\}} \{c_\tau\} \\ &= \max_{\{\tau \in \mathcal{T} \text{ and } \tau \neq j\}} \left\{\frac{l_\tau}{B}\right\} \\ &= \max_{\{\tau \in \mathcal{T} \text{ and } \tau \neq j\}} \frac{\{l_\tau\}}{B}. \end{aligned}$$

Therefore (1) is transformed into:

$$\sum_{\tau \in \mathcal{T}} \frac{l_\tau}{B} f_\tau + \frac{L_j}{B} f_j \le 1, \text{ for each } j \in \mathcal{T}, \tag{2}$$

where $L_j = \max_{\{\tau \in \mathcal{T} \text{ and } \tau \neq j\}} \{l_\tau\}$.

Multiply both sides of Eq. (2) with broadcast link bandwidth B, we get

$$\sum_{\tau \in \mathcal{T}} l_\tau f_\tau + L_j f_j \le B, \text{ for each } j \in \mathcal{T}. \tag{3}$$

Besides of being a router, a RICH base station can also simultaneously function as the source end of a route. The data are either from the base station's local sensing CoPU or by gathering intra-cell slave sensors' readings. Either way, the base station can be regarded as the virtual singular source end sensor for the route, and its sampling rate is upper bounded, creating the following constraint (for the time being, we assume each base station can be the source end of at the most *one* route):

$$f_j \le f_j^{max}, \tag{4}$$

where f_j is the sampling rate of route j, and f_j^{max} is the maximum affordable sampling rate at the route's source end base station. [4] Hereby, by analyzing each base station of the RICH RTWSN according to inequality set (3) and each route according to inequality (4), we can derive a set of linear inequalities, which is the sufficient real-time schedulability constraints for RICH RTWSN sampling rate assignment. We can summarize them into the following form:

$$\begin{cases} \mathbf{A}f \le W \\ \quad f \le f^{max}, \end{cases} \tag{5}$$

where $f = (f_1, f_2, \ldots, f_N)^\mathsf{T}$ is the vector of sampling rates assigned to each of the N routes. $f^{max} = (f_1^{max}, f_2^{max}, \ldots, f_N^{max})^\mathsf{T}$ is the maximum sampling rate for the N end point sensors (see (4)). Matrix $\mathbf{A}$ and vector W are obtained by base station-wise analysis according to (3), which reflect the specific routing topology of the RICH RTWSN. Suppose these schedulability analysis 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 rate requirements. These extra requirements can be written as:

$$f \ge f^{min} = (f_1^{min}, \ldots, f_N^{min})^\mathsf{T}. \tag{6}$$

Inequality set (5) and (6) constitute a complete set of real-time schedulability constraints for RICH RTWSN sampling rate allocation. An example is given as follows:

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

Let $B_i (i = 1, \ldots, 10)$ *be the transmitting bandwidth of base station* i*. Suppose the sampling rate 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 data packet size, minimum and maximum sampling rate constraints for Route* j*.*

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

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

---

[4] In this paper, we assume the capacity of CPU, internal bus, sensing CoPU and RF CoPUs are big enough, so that as long as (4) is satisfied, we only need to concern about the wireless network bandwidth schedulability.

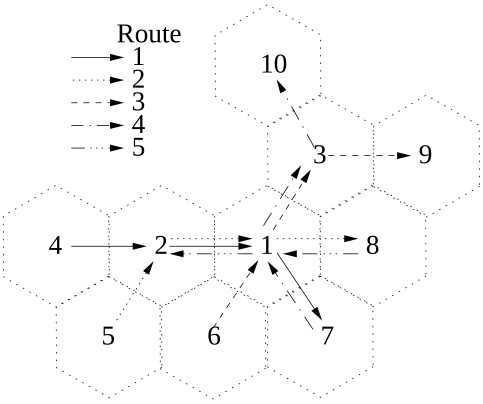


Fig. 4. A RICH RTWSN Schedulability Analysis Example

$$
\begin{aligned}
(l_1f_1 + l_2f_2 + l_3f_3 + l_4f_4 + l_5f_5) + \max\{l_2, l_3, l_4, l_5\}f_1 &\leq B_1 \\
(l_1f_1 + l_2f_2 + l_3f_3 + l_4f_4 + l_5f_5) + \max\{l_1, l_3, l_4, l_5\}f_2 &\leq B_1 \\
(l_1f_1 + l_2f_2 + l_3f_3 + l_4f_4 + l_5f_5) + \max\{l_1, l_2, l_4, l_5\}f_3 &\leq B_1 \\
(l_1f_1 + l_2f_2 + l_3f_3 + l_4f_4 + l_5f_5) + \max\{l_1, l_2, l_3, l_5\}f_4 &\leq B_1 \\
(l_1f_1 + l_2f_2 + l_3f_3 + l_4f_4 + l_5f_5) + \max\{l_1, l_2, l_3, l_4\}f_5 &\leq B_1
\end{aligned}
$$

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

$$
\begin{aligned}
(l_1f_1 + l_2f_2) + l_2f_1 &\leq B_2 \\
(l_1f_1 + l_2f_2) + l_1f_2 &\leq B_2
\end{aligned}
$$

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

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

$$
\begin{aligned}
(l_3f_3 + l_4f_4) + l_4f_3 &\leq B_3 \\
(l_3f_3 + l_4f_4) + l_3f_4 &\leq B_3
\end{aligned}
$$

Node 4: As a base station along Route 1, we have $l_1f_1 \leq B_4$.

Node 5: As a base station along Route 2, we have $l_2f_2 \leq B_5$.

Node 6: As a base station along Route 3, we have $l_3f_3 \leq B_6$.

Node 7: As a base station along Route 4, we have $l_4f_4 \leq B_7$. *As the destination end for Route 1, there are no constraints (the corresponding constraints are analyzed at base station 1, Route 1's last sending hop).*

Node 8: As a base station along Route 5, we have $l_5f_5 \leq B_8$. *As the destination end for Route 2, there are no constraints (the corresponding constraints are analyzed at base station 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).

$$
\left\{
\begin{array}{l}
\mathbf{A} = \begin{bmatrix}
l_1 + \max\{l_2, l_3, l_4, l_5\} & l_2 & l_3 & l_4 & l_5 \\
l_1 & l_2 + \max\{l_1, l_3, l_4, l_5\} & l_3 & l_4 & l_5 \\
l_1 & l_2 & l_3 + \max\{l_1, l_2, l_4, l_5\} & l_4 & l_5 \\
l_1 & l_2 & l_3 & l_4 + \max\{l_1, l_2, l_3, l_5\} & l_5 \\
l_1 & l_2 & l_3 & l_4 & l_5 + \max\{l_1, l_2, l_3, l_4\} \\
l_1 + l_2 & l_2 & 0 & 0 & 0 \\
l_1 & l_1 + l_2 & 0 & 0 & 0 \\
0 & 0 & l_3 + l_4 & l_4 & 0 \\
0 & 0 & l_3 & l_3 + l_4 & 0 \\
l_1 & 0 & 0 & 0 & 0 \\
0 & l_2 & 0 & 0 & 0 \\
0 & 0 & l_3 & 0 & 0 \\
0 & 0 & 0 & l_4 & 0 \\
0 & 0 & 0 & 0 & l_5
\end{bmatrix} \\
f = (f_1, f_2, f_3, f_4, f_5)^{\mathsf{T}} \\
W = (B_1, B_1, B_1, B_1, B_1, B_2, B_2, B_3, B_3, B_4, B_5, B_6, B_7, B_8)^{\mathsf{T}}
\end{array}
\right.
\tag{7}
$$

In addition to all above, because of the minimum sampling rate constraints, we have: $f_j \geq f_j^{min}$*, where* $(j = 1, \ldots, 5)$*. The complete rate 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 (7).*

Suppose the numerical values of parameters are as shown in Table I, then the complete rate allocation constraints are transformed into (8).

Table I. Parameter Values for Example 1

Node	Bandwidth(B_i Mbps)	Max Sampling Capability (f^{max} Hz)
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} Hz)	Affordable Max Freq. (f_j^{max} Hz)	Report Packet Size (l_j Mbit)
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

3. OPTIMIZING QOS IN WSN WITH REAL-TIME CONSTRAINTS — MATH MODELING

In this section, we model the optimal sampling rate allocation problem as an nonlinear convex optimization problem, using constraints set (5)(6) from the previous

$$\left\{\begin{array}{l} \mathbf{A}f = \begin{bmatrix} .04 & .015 & .02 & .025 & .03 \\ .01 & .045 & .02 & .025 & .03 \\ .01 & .015 & .05 & .025 & .03 \\ .01 & .015 & .02 & .055 & .03 \\ .01 & .015 & .02 & .025 & .055 \\ .025 & .015 & 0 & 0 & 0 \\ .01 & .025 & 0 & 0 & 0 \\ 0 & 0 & .045 & .025 & 0 \\ 0 & 0 & .02 & .045 & 0 \\ .01 & 0 & 0 & 0 & 0 \\ 0 & .015 & 0 & 0 & 0 \\ 0 & 0 & .02 & 0 & 0 \\ 0 & 0 & 0 & .025 & 0 \\ 0 & 0 & 0 & 0 & .03 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} \leq W = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ .6 \\ .6 \\ .4 \\ .4 \\ .25 \\ .25 \\ .25 \\ .2 \\ .15 \end{bmatrix}, \\ f \leq f^{max} = (30, 25, 30, 40, 30)^{\mathsf{T}}, \text{ and } f \geq f^{min} = (11, 2.5, 5, 1, 2)^{\mathsf{T}}. \end{array}\right. \tag{8}$$

section.

3.1 Utility Loss Index

The base station at the source end of a route periodically samples and reports sensor readings. Let the sampling/reporting rate (or "frequency") for the jth route be f_j. For most applications, the higher the sampling/reporting rate f_j, the higher is the QoS. For example, for control applications, the faster the sampling rate, the better the control performance [Seto et al. 1996]. Ideally, the best performance is achieved if the sampling rate is approaching ∞, i.e. continuous sampling. In practice, this is not achievable, so we use *Utility Loss Index* (ULI) function to capture the performance *loss* at a discrete sampling rate f compared to continuous sampling. For control applications, Seto *et al.* [1996] shows the ULI is in the following form:

$$U_j(f_j) = \omega_j \alpha_j e^{-\beta_j f_j}, \tag{9}$$

where f_j is the sampling rate for task j (*i.e.* route j), ω_j, α_j and β_j are application specific constraints. The values of ω_j, α_j and β_j can be determined through data fitting using real-world measurements. In this paper, the form of ULI function is generalized to be any strictly decreasing differentiable convex function with regard to rate f.

3.2 Mathematical Formulation

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

$$\min_{(f_1,\dots,f_N)} \sum_{j=1}^{N} U_j(f_j) \tag{10}$$

$$\text{s.t.: } \mathbf{A}f \leq W \tag{11}$$

$$f \leq f^{max} \tag{12}$$

$$f \geq f^{min} \tag{13}$$

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.* [1996]'s *Single Constraint Optimization Problem* (SCOP), and we denote the former as $MCOP(U, \mathbf{A}, W)$.[5]

The feasible set of MCOP is compact and convex and $U_j(f_j)$ is differentiable and convex, therefore MCOP has optimal solutions [Bertsekas 1995]. Furthermore, if $U_j(f_j)$ is strictly convex, the optimal solution is unique [Bertsekas 1995].

When $M = 1$ and there is no constraint set (12), then the ULIs are in the negative exponential form, and $MCOP(U, \mathbf{A}, W)$ becomes SCOP as follows [Seto et al. 1996]:

$$\min_{(f_1,\dots,f_N)} \textstyle\sum_{j=1}^{N} U_j(f_j) = \displaystyle\sum_{j=1}^{N} \omega_j \alpha_j e^{-\beta_j f_j} \tag{14}$$

$$\text{s.t.: } Af \leq W \tag{15}$$

$$f \geq f^{min} = (f_1^{min}, \dots, f_N^{min})^{\mathsf{T}} \tag{16}$$

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

Based on Kuhn-Tucker condition, Seto *et al.* [1996] provides an algorithm to solve SCOP problem analytically. MCOP is a generalization of SCOP. We shall show 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 (11) and (12) becomes equality constraint and show why it is not viable to tackle the MCOP in an analytical fashion similar to [Seto et al. 1996]. In $MCOP(U, \mathbf{A}, W)$, constraint set (11) and (12) can be combined as:

$$\begin{aligned} &\mathbf{A}'f \leq W', \\ &\text{where } \mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \mathbf{I} \end{bmatrix}_{(M+N)\times N}, \; W' = \begin{bmatrix} W \\ f^{max} \end{bmatrix}_{(M+N)\times 1} \end{aligned} \tag{17}$$

THEOREM 3.1. *MCOP's optimal solution f^* must ensure that at least one of the $(M+N)$ constraints $A'_i f^* \leq W'_i, i = 1, \dots, (M+N)$ reaches equality. i.e., $\exists i \in \{1, \dots, M+N\}$ such that $A'_i f^* = W'_i$. Here A'_i is the ith row of $\mathbf{A}'$.*

PROOF. Please refer to Appendix B for the proof. ☐

Though we know for the optimal rate assignment f^* there is at least one i such that $A'_i f^* = W'_i$, we do not know explicitly which constraints reach equality. This makes the Kuhn-Tucker based solution method not applicable. In contrast, the problem discussed in [Seto et al. 1996], which is a SCOP ($M = 1$), 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. In addition to [Seto et al. 1996], Rajkumar

---

[5] Notation used in [Kelly et al. 1998].

et al. [1997] proposed a numerical solution. But that solution is also for single constraint scenario. In their more recent works, Lee [1999] and Ghosh [2003] *et al.* studied the scenario under multiple constraints. However, the problem they studied is an integer programming problem, which is different from the model we will discuss in this paper. In [Lee et al. 1999], the integer programming problem is proven to be NP-Hard. Several sub-optimal algorithms are proposed. According to [Ghosh et al. 2003], 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 multi-hop RTWSN (see Section 6.5). Fortunately, as will become clear later, MCOP can be solved with the state-of-art Interior Point Methods [Nesterov and Nemirovsky 1994; Ye 1997a] and Internet pricing schemes [Low and Lapsley 1999; Kelly et al. 1998; Kelly 1997].

4. CENTRALIZED SOLUTION METHOD FOR MCOP

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

Definition 4.1. 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 $h : \mathbb{R}^n \rightarrow \mathbb{R}^p; g : \mathbb{R}^n \rightarrow \mathbb{R}^q$.

Definition 4.2. 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 [Nesterov and Nemirovsky 1994; Ye 1997a] 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 [Luenberger 1984] 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 [Seto et al. 1996], but can be general strictly decreasing differentiable convex functions.

To solve the MCOP problem, we use optimization library COPL_LC [Ye 1997b]. It is easy to transform our MCOP problem to the form used by COPL_LC. The transformation method can be found in Appendix C. 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 RATE 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 6.5). 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 rates. The algorithm is based on the recent researches of Internet pricing schemes [Low and Lapsley 1999; Kelly et al. 1998; Kelly 1997], especially [Low and Lapsley 1999].

The main idea is to impose a price on each constraint in (11)∼(13). 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 sampling rate for the route (to simplify, we call it *rate proposal* in the following). The rate proposal is then delivered to each of the route's routers, where each of the route's relevant constraint updates (imagine each constraint as an active agent) its price (called *constraint price*) accordingly. This procedure works in an iterative manner until it converges.

The distributed algorithms has two main attributes:

(1) It converges to the optimal rates of MCOP (Theorem 5.1).

(2) Each route's computation is only based on local information.

Notations used in the Distributed 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 rate proposal vector for each route $j, j \in \{1, \ldots, N\}$ at iteration step s. $f(s) = (f_1(s), \ldots, f_N(s))^{\mathsf{T}}$.

The Distributed Algorithm:

The distributed algorithm is made up of iterations. Each iteration consists of two consecutive steps: the *Constraint Algorithm*, and the *Route Algorithm*.

At the very beginning of the distributed algorithm, set $f(0) = f^{min}$, and $p(0) \geq 0$.

1) Constraint Algorithm

During iteration $s = 1, 2, \ldots$, for each constraint $i (i = 1, \ldots, M)$:

C1. Receives rate proposal $f_j(s)$ from each relevant route j. Route j and constraint i are relevant if $\mathbf{A}_{ij} \neq 0$.

C2. Computes a new constraint price for itself using the following price update equation:

$$p_i(s+1) = [p_i(s) + \gamma(f^i(s) - W_i)]^+ \qquad (18)$$

Here $f^i(s) = A_i f(s)$, and A_i is the ith row of $\mathbf{A}$. Function $[\bullet]^+$ is defined as $[x]^+ = \max\{x, 0\}$, where x is a real number.

C3. 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 $j (j = 1, \ldots, N)$:

R1. Receives from the network the sum of all the constraints' prices $p^j(s)$ imposed on this route:

$$p^j(t) = \sum_{i=1}^{M} p_i(s) \mathbf{A}_{ij} \tag{19}$$

R2. Update the route's rate proposal $f_j(s+1)$ for the next iteration according to local optimization of:

$$\begin{aligned} \min \quad & U_j(f_j) + f_j p^j(s) \\ \text{s.t.:} \quad & f_j^{min} \le f_j \le f_j^{max} \end{aligned}$$

$$i.e. \;\; f_j(s+1) = \arg \min_{f_j^{min} \le f_j \le f_j^{max}} (U_j(f_j) + f_j p^j(s)) \tag{20}$$

The iteration of Constraint and Route Algorithms stops until the predefined convergence criterion is reached. For example, when both of the following two criteria are met.

$$\|f(s) - f(s-1)\|_n \le \varepsilon_f, \tag{21}$$

$$\|q(s) - q(s-1)\|_n \le \varepsilon_q, \tag{22}$$

where $f(s) = (f_1(s), f_2(s), \ldots, f_N(s))^{\mathsf{T}}$, $q(s) = (p^1(s), p^2(s), \ldots, p^N(s))^{\mathsf{T}}$. $\varepsilon_f > 0$ and $\varepsilon_p > 0$ are sufficiently small real numbers. $\|v\|_n$ denotes the nth-norm of vector $v = (v_1, \ldots, v_k)$. If $n = 1$, $\|v\|_1 = \max(v_i), i \in \{1, \ldots, k\}$. If $n \in \mathbb{Z}^+$, then $\|v\|_n = (\sum_{i=1}^{k} v_i^n)^{\frac{1}{n}}$.

Now we prove the convergence and correctness of the above iterative algorithm. This is summarized in *Theorem 5.1.* First, we give the assumptions and notations to be used.

Assumptions:

A1. The feasibility condition holds for each constraint $i (i = 1, \ldots, M)$, such that $\sum_{j=1}^{N} \mathbf{A}_{ij} f_j^{min} \le W_i$.

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. $U_j''(f_j) \ge \frac{1}{\bar{\alpha}_j} > 0$, for all $f_j \in I_j$.

Notations Used in Theorem 5.1:

· $L(j) = \sum_{i=1}^{M} \mathbf{A}_{ij}$. It is the column sum of $\mathbf{A}$;

· $\bar{L} = \max_{j=1,\ldots,N}\{|L(j)|\}$, which is the maximum absolute value of column sum of $\mathbf{A}$;

· $S(i) = \sum_{j=1}^{N} \mathbf{A}_{ij}$. It is the row sum of $\mathbf{A}$;

· $\bar{S} = \max_{i=1,\ldots,M}\{|S(i)|\}$, which is the maximum absolute value of row sum of $\mathbf{A}$;

· $\bar{\alpha} = \max_{j=1,\ldots,N}\{\bar{\alpha}_j\}$. *i.e.* $\bar{\alpha}$ is the upper bound on $\frac{1}{U_j''(f_j)}, j = 1, \ldots, N$.

THEOREM 5.1. *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 rates $f^{min} \leq f(0) \leq f^{max}$ 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 Appendix D. □

6. EVALUATION

In this section, we shall present the simulation results and discuss the tradeoffs between centralized algorithm and distributed algorithm, showing which is more appropriate in what situations. We also show that distributed algorithm has the desirable incremental adjustment property.

6.1 Implementation

First let us look at the real-world feasibility of RICH architecture. Real-world DSSS Chip-Sets consists of multiple parallel independent transmitters and receivers are already available. For example, the QualComm CSM2000 chipset [Qualcomm 2004a], originally designed for low-cost lightweight cellular base stations, supports eight parallel users, which is enough for the seven-transceiver RICH architecture. Higher performance chipsets can be QualComm CSM5000 [Qualcomm 2004b], CSM5500 [Qualcomm 2004c] *etc.*, which can be easily reconfigured to build RICH base stations, providing no less than 1.8Mbps data bandwidth for each of the seven transceivers. The sizes and power consumption of these chip sets are also satisfactorilly small. For example, a CSM5500 chip complies with BGA560 packaging, which is $35 \times 35 \times 2.5$mm in dimension; and is of $3 \sim 3.6$volt I/O voltage and 1.8volt core voltage.

Based on the above real-world parameters, we carry out simulation using J-Sim [DRCL 2004]. The centralized algorithm is straightforward. To simulate the distributed algorithm, we need to devise a network protocol that matches the algorithm described in Section 5, which is as following:

Network Protocol for Distributed Algorithm:

The protocol is carried out in iterations, each iteration s consists of two steps:

Step 1 Each constraint's price is updated by the router that creates that constraint based on (18). Next, each of these updated prices must be propagated to all the relevant routes. To do that, each route's source end sends an empty packet toward 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.

Step 2 After Step 1, each route's destination end carries out the route algorithm (20) to update the sampling rate proposal. Then the destination end sends another packet towards the source end, to notify every router along this route about the updated sampling rate proposal. This packet's payload is also just one floating-point number (*i.e.* 4 bytes), which is enough to carry the updated sampling rate proposal.

If the payload of the control traffic is piggybacked to the data traffic, it will add a 4 bytes overload. If the control traffic is sent separately from the data traffic, it can be encoded into a 16 byte packet. Within this 16 byte packet, 4 bytes are the control payload, 4 bytes are for source address, and 4 bytes are for destination address, and the remaining 4 bytes are for other purposes such as checksum etc. In the following discussions, we discuss the separate control message scheme, that is, the distributed algorithm incurs a 16 byte packet in Step 1 and Step 2 respectively for each route.

For distributed algorithm, we also assume all the involved routes of the RTWSN are coarse-grain synchronized in the sense that in each iteration, all the routes finish Step 1 and then move on to Step 2; and when all the routes finish Step 2, they move on to the next iteration. This can be achieved, *e.g.*, by synchronizing all the nodes and start each step at time kT_{step}, where $k \in \mathbb{Z}$ and T_{step} is the empirical upper bound of end-to-end packet travel time along the network diameter, assuming there is a specified upper bound on network diameter. The GPS System [Getting 1993] can already provide global time synchronization with an accuracy of within 0.25msec [Exit Consulting 2004], which is enough for our application. For example, in our simulation setup, a synchronization granularity of 2msec is enough for the testbed.

6.2 Numerical Example of Centralized and Distributed Algorithm

First both the centralized and distributed algorithm are applied to the scenario discussed in Example 1 of Section 2.2. The setup involves 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 II. These parameters are taken from those reported in [Sha et al. 2000]. Constraints are listed in equation (8).

Table II. Parameters for ULI in Example 1

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

Using the centralized algorithm and the COPL_LC package [Ye 1997b], by 14 iterations, the optimal solution is derived: $f^*_{central} = (12.35, 6.58, 5.70, 5.73, 5.00)^\mathsf{T}$, with an optimal value of 0.916.[6]

For the distributed algorithm, we choose the initial values to be $f(0) = f^{min}$, $p(0) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ and $\gamma = 2 \times 10^{-7}$. The network parameter settings follow Table I. The convergence criterions are described by equation (21) and (22), where we pick $\varepsilon_f = 1 \times 10^{-9}$ and $\varepsilon_q = 1 \times 10^{-9}$. The rate proposal update trace is shown in Fig. 5. The trace shows the algorithm converges in short time (no more than 9.48sec). The converged rate proposal value is: $f^*_{distributed} = (12.35, 6.58, 5.70, 5.73, 5.00)^\mathsf{T}$ which matches the results got from centralized algorithm.

It is worth noting that, for many RTWSN applications, it is not necessary to derive the exact optimum sampling rate. Instead, getting a quasi-optimum with relatively shorter time is often more preferrable. In Table III, the convergence time for each route with certain error bound is listed. We see if coarser error bound is allowed, the convergence time is even shorter.

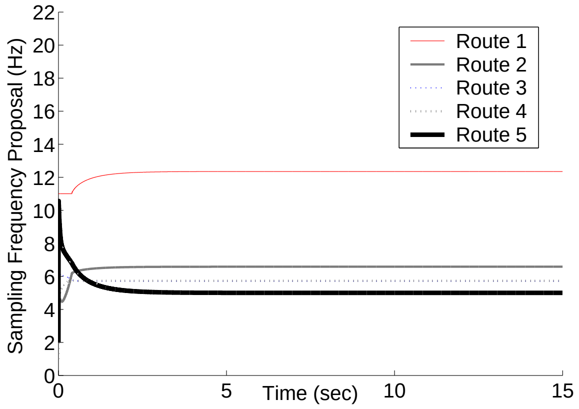


Fig. 5. Rate Proposal Update Trace

Table III. Convergence Time

Route	1	2	3	4	5
Convergence Time (CT) (sec)	6.58	5.80	5.59	5.66	9.48
CT when ±1% error is allowed (sec)	1.79	1.39	0.45	0.27	2.73
CT when ±5% error is allowed (sec)	0.74	0.46	0.23	0.13	1.6

---

[6] The primal objective values reported in the 14 iterations are {1.649, 1.339, 1.237, 1.072, 0.834, 0.504, 0.649, 0.683, 0.792, 0.899, 0.915, 0.916, 0.916, 0.916}.

6.3 Monte Carlo Simulation on Convergence Speed

In order to give a feeling of how fast the distributed algorithm converges, the following Monte Carlo simulation is carried out:

Rate error of route j at the sth iteration of distributed algorithm is defined as: $e\_f_j(s) = |f_j(s) - f_j^*|$, where $f_j(s)$ is the rate proposal for route j at the sth iteration; f_j^* is the optimal sampling rate for this route.

We still use the testbed depicted by Example 1. But in each trial of the Monte Carlo, a different ULI function for each route is picked by setting the coefficients of ω_j, α_j and β_j randomly. Then the distributed algorithm is carried out. The rate error for iteration $s = 1, \ldots, 500$ is traced. Eight hundred trials are run. For each route, the eight hundred rate error traces are averaged and plotted in Fig. 6.

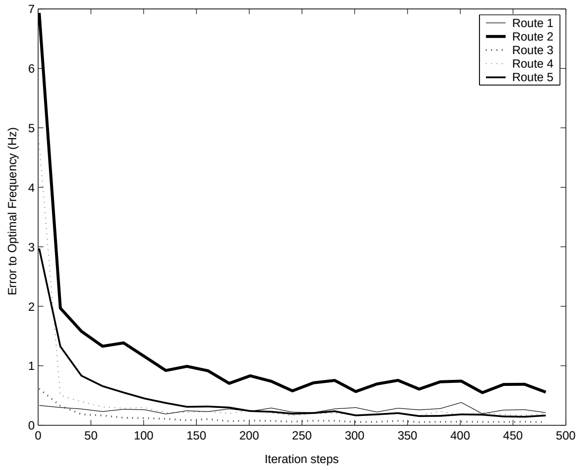


Fig. 6. Trace of error between proposed and optimal rates. Empirically, in 100 iterations, the proposed rates converge to a satisfactory range around the optimum.

According to Fig. 6, the rate error converges in a negative exponential form. Empirically, in 100 iterations the distributed algorithm reaches a satisfactory quasi-optimum.

It is worth noting that by exploiting the "incremental adjustment" property discussed in Section 6.4, the distributed algorithm can converge even faster.

6.4 Incremental Adjustment Property of Distributed Algorithm

In real-world, there are often times that ULI functions and constraint set change dynamically. These changes transform the original optimization problem $MCOP(U, \mathbf{A}, W)$ into a new optimization problem $MCOP(U', \mathbf{A}', W')$, hence the optimal sampling rate f^* has to be re-calculated. If distributed algorithm is used, new iterations can be carried out from the existing optimum (f^*, p^*), so as to reach the new optimum (f'^*, p'^*) faster. We call this "incremental adjustment property". An example is given as follows:

Continue with the $MCOP(U, \mathbf{A}, W)$ simulation example in Section 6.2. Suppose at time 15sec, the ULI coefficients switch from old value set (see Table II) to the new value set depicted in Table IV. Fig. 7 and Table V shows the comparison between incremental and non-incremental adjustment schemes: the incremental adjustment scheme starts with $MCOP(U, \mathbf{A}, W)$'s optimum sampling rate and its corresponding price vector (f^*, p^*); the non-incremental adjustment scheme starts with a constant tuple (f^0, p^0), where $f^0 = f^{min}$ and $p^0 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$. All other settings of the testbed are the same as those for Section 6.2, Under incremental adjustment scheme, in about 10.30sec, the rate proposal converges to the new optimum f'^*. In contrast, if start from (f^0, p^0), the convergence to new optimum f'^* is much slower, taking 12.67sec.

Furthermore, if quasi-optimum is allowed, the new solutions can be derived faster. The convergence time costs are listed in Table V. The incremental adjustment scheme still runs faster than the non-incremental version.

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

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 V. Convergence Time of $MCOP(U', \mathbf{A}', W')$

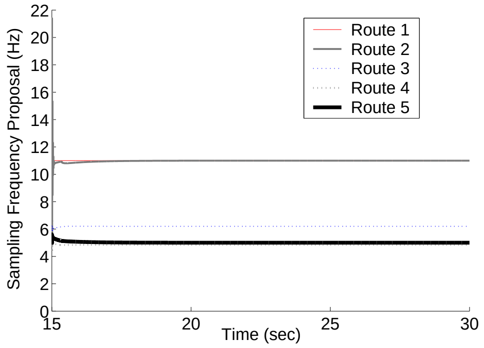
	Route	1	2	3	4	5
starts with (f^*, p^*)	Convergence Time (CT) (sec)	8.67	7.68	6.98	7.32	10.30
	CT when ±1% error is allowed (sec)	0.01	1.13	0.25	0.07	1.24
	CT when ±5% error is allowed (sec)	0.01	0.05	0.03	0.02	0.15
starts with (f^0, p^0)	Convergence Time (CT) (sec)	11.04	10.05	9.35	9.69	12.67
	CT when ±1% error is allowed (sec)	0.002	3.51	1.25	1.61	3.61
	CT when ±5% error is allowed (sec)	0.002	1.93	0.03	0.39	2.01

6.5 Control Traffic and Scalability Analysis for Distributed and Centralized Algorithm

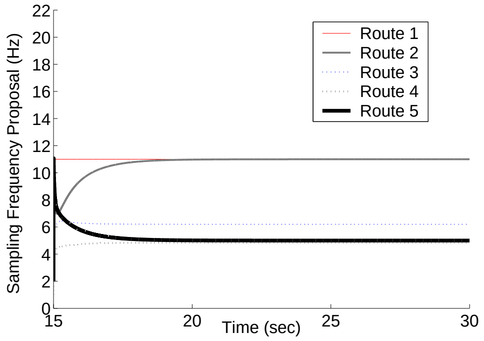
In this section, the control traffic for both distributed and centralized algorithms are analyzed. The centralized algorithm is efficient even when the network is moderately large. However, when the network continues to scale up, the centralized algorithm would finally reach its bottleneck. In contrast, under certain assumptions, the distributed algorithm provides better scalability, though it may be inefficient for smaller networks.

Control Traffic Analysis for Distributed Algorithm:

Let $\mathcal{N}$ be the set of all base stations in a RICH RTWSN. Let ϕ_i^{dis} be the accumulated control traffic (in bytes) passing through base station $i (i \in \mathcal{N})$ under the distributed algorithm. *Let* Φ^{dis} *be the maximum accumulated control traffic (in bytes) passing through any of the base stations, i.e.* $\Phi^{dis} = \max_{i \in \mathcal{N}} \{\phi_i^{dis}\}$.



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



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

Fig. 7. Illustration of Incremental Adjustment Property

Let $\mathcal{R}$ be the set of all routers (i.e. RICH base stations that serve as 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 maximum number of routes pass through any routers, *i.e.* $D = \max_{r\in\mathcal{R}}\{d_r\}$.

During each iteration, in Step 1, totally d_r control packets pass through router r. In Step 2, totally d_r packets pass through router r. Without loss of generality, we assume all control packets are of 12 bytes of headers. As mentioned previously, each packet's payload length is 4 bytes. Therefore the total control traffic passing through each router r during each iteration is $32d_r$ bytes, we therefore have the following proposition:

PROPOSITION 6.1. *For any base station $i \in \mathcal{N}$, during each distributed algorithm iteration, the number of control packet passing through it is no more than $32D$ bytes.*

According to our simulation results in Section 6.3, the distributed algorithm usually converges or reaches very good approximation in $K \leq 100$ steps (exploiting incremental adjustment property discussed in Section 6.4, or allowing quasi-optimum, the number of iterations may be even less). Hence after all the iterations, the accumulated control traffic passing through any router is no more than $32KD$. That is, for any node $i \in \mathcal{N}$, $\phi_i^{dis} \leq 32KD \leq 32 \times 100D = 3200D$ (byte). Therefore we have:

$$\Phi^{dis} \leq 3200D \tag{23}$$

$$i.e.\ \Phi^{dis} = O(D) \tag{24}$$

Traffic Load Analysis for Centralized Algorithm:

Let $\mathcal{N}$ be the set of all base stations in a RICH RTWSN. Let ϕ_i^{cen} be the accumulated control traffic passing through base station $i (i \in \mathcal{N})$ under the centralized algorithm. *Let* Φ^{cen} *be the maximum accumulated control traffic passing through any of the base stations, 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 **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). To be consistant, we still assume the control packet header is of 12 bytes. Thus the accumulated control traffic payload at node **C** is $\phi_{\mathbf{C}}^{cen} \geq 32\Gamma^{total}$, *i.e.* $\phi_{\mathbf{C}}^{cen} = \Omega(\Gamma^{total})$. Because $\Phi^{cen} \geq \phi_{\mathbf{C}}^{cen}$, we hereby have:

$$\Phi^{cen} = \Omega(\Gamma^{total}) \tag{25}$$

One may argue that the routes in an RICH RTWSN may not be all directly or indirectly connected, but rather be partitioned into several disjoint *maximal sub-graphs* (routes within each maximal sub-graph are directly or indirectly connected). So that there need not to be *ONE* central computing node. Each maximal sub-graph can elect its own central computing node, which takes charge of the optimal sampling rate planning for and only for the routes within that maximal sub-graph. In this case, let $\mathcal{G}$ be the set of all maximal sub-graphs, for a specific maximal 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 (25), we have $\phi_{\mathbf{C}_g}^{cen} \geq 32\Gamma_g$. By the definition of Φ^{cen}, we still have $\Phi^{cen} \geq \phi_{\mathbf{C}_g}^{cen} \geq 32\Gamma_g$. That is: $\forall g \in \mathcal{G}, \Phi^{cen} \geq 32\Gamma_g$, which implies $\Phi^{cen} \geq \max_{g \in G}\{32\Gamma_g\}$. Let $\Gamma = \max_{g \in G}\{\Gamma_g\}$, *i.e.* the maximum number of directly or indirectly connected routes of the whole RTWSN, then we have:

$$\Phi^{cen} \geq 32\Gamma$$

$$i.e.\ \Phi^{cen} = \Omega(\Gamma) \tag{26}$$

In the following simulation for wide area monitoring, we shall show $\Gamma \approx \Gamma^{total}$, i.e. (26) is empirically equivalent to (25). We shall also show Φ^{dis} is empirically insensitive to the scale of the RICH RTWSN while Φ^{cen} 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.

Comparison of Distributed and Centralized Algorithm:

In many cases, even for moderately large networks, the centralized algorithm is efficient enough. But there are certain cases where the distributed algorithm is more scalable than the centralized algorithm. An example scenario is as follows:

Suppose all routes are desseminated in a square area of $l \times l\text{km}^2$, where l is the square edge length. The square area deploys a RICH RTWSN cellular division with a hexagon cell edge length of 0.1km. All routes between base stations are unicast, and we assume the network diameter is upper bounded by a fixed constant, which, without loss of generality, is set to 10. Specifically, the source end base stations of all routes are uniformly distributed across the square area with density $\rho = 10/\text{km}^2$; and the destination end is also uniformly distributed within 10 hops from the source end. This also implies the total number of routes is ρl^2. The route is determined by the source/destination end and a simple geographical routing protocol that always forwards the packet closer to the destination in each hop.

For each RICH RTWSN scale ($l = 5, \ldots, 100(\text{km})$), thirty trials are carried out. In each trial, ρl^2 routes are generated according to the previous description; then the maximum number of routes passing through any router (*i.e.* D), and the maximum number of directly or indirectly connected routes (*i.e.* Γ) are counted. The results are shown in Table VI. For ease of comparision, we plot the same data in Fig. 8. We can see from Fig. 8, *D is bounded by a relatively small constant, insensitive to the network scale l, while Γ rises in a roughly l^2 speed.*[7]

According to (24) and (26), $\Phi^{dis} = O(D)$ and $\Phi^{cen} = \Omega(\Gamma)$. Therefore for distributed algorithm, the maximum per-base station accumulated control traffic (*i.e.* Φ^{dis}) is upper bounded by D and D is insensitive to the scale of the network. In contrast, for centralized algorithm, the maximum per-base station accumulated control traffic (*i.e.* Φ^{cen}) is lower bounded by Γ and Γ is growing quadratically with the network scale l. The underlying reason is that for centralized algorithm, the control traffic is bottlenecked at the central computing node, while for distributed algorithm, the control traffic is evenly distributed among all the nodes. Hence, the distributed algorithm shows better scalability.

According to Section 6.2, centralized algorithm works efficiently when the network scale is small or even moderately large. For distributed algorithm, from the above analysis, because of better scalability, when the network continues scaling up, there will be a point where the distributed algorithm starts to outperform the centralized algorithm. It is useful to set up a threshold to determine when to switch from centralized to distributed algorithm. According to Fig. 8 and Table VI, D is always less than 15; whereas Γ is never less than 1500 when $l \geq 15\text{km}$. According to (23), $\Phi^{dis} \leq 3200D = 48000$ (byte). According to (26), $\Phi^{cen} \geq 32\Gamma \geq 32 \times 1500 = 48000$ (byte), when $l \geq 15\text{km}$. That is, when l is bigger than 15km, there is $\Phi^{dis} \leq \Phi^{cen}$, which means distributed algorithm is more desirable.

In the end, it's worth mentioning 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 rate proposal. Ideally, control messages are sent in background using spare bandwidth. We call this "best-effort" approach since the bandwidth assigned for background traffic is not guaranteed. According to Theorem 3.1, when the optimal rate $f^* =$

---

[7] In other scenarios, the sensitivity analysis of D to network scale is left for future research.

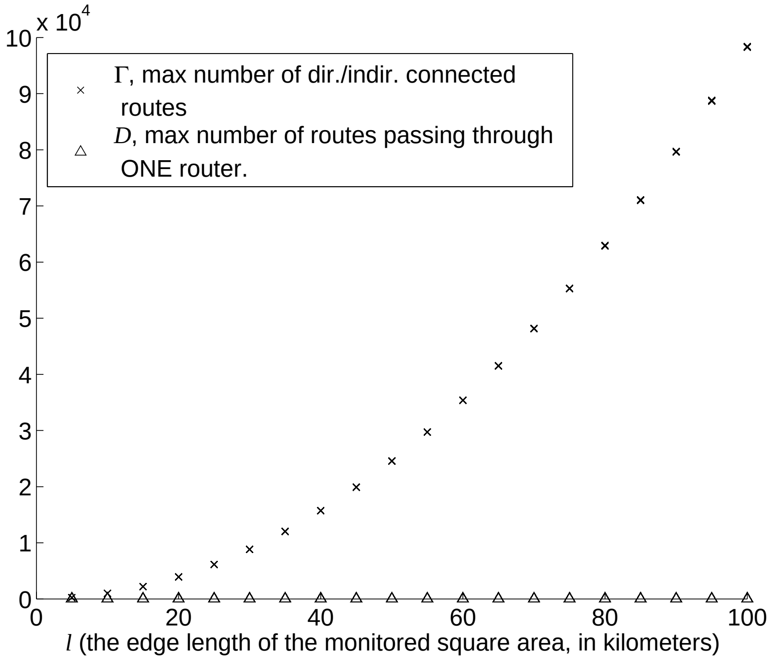


Fig. 8. Scalability Comparison. l(km) is the edgelength of the square area. The route density $\rho = 10/\text{km}^2$. Γ is the maximum number of directly or indirectly connected routes. The centralized algorithm incurs a time cost $\Phi^{cen} = \Omega(\Gamma)$. D is the maximum number of routes passing through any router. The distributed algorithm incurs a time cost $\Phi^{dis} = O(D)$. Thirty trials are carried out for each l. The results shows the distributed algorithm has better scalability than the centralized algorithm.

$(f_1^*, \ldots, f_N^*)^\mathsf{T}$ 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 data 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 6.1, the maximum amount of control message payload bytes passes through each RICH RTWSN node during each iteration is no more than $32D$, where D is the maximum number of routes passing through any router in the whole RTWSN. Fig. 8 and Table VI show when the maximum route length and density of routes in the RTWSN are fixed, and the end points of routes are uniformly distributed, empirically, D is bounded by a constant, which can be estimated via simulation. Therefore, the bandwidth to be reserved for control message exchange can be planned accordingly. For example, according to Fig. 8 and Table VI, D is empirically bounded by 15. Therefore each iteration causes a control traffic of $32 \times 15 = 480$ bytes. If 100 iterations is needed to get the result, and the distributed algorithm is supposed to finish

Table VI. Scalabillity comparison

l	Γ			D		
(km)	mean	min	max	mean	min	max
5	242	236	247	7	6	10
10	977	965	988	8	7	10
15	2203	2185	2220	8	7	11
20	3920	3894	3935	8	7	10
25	6126	6100	6145	9	8	11
30	8835	8810	8860	9	8	11
35	12020	11975	12057	9	8	11
40	15713	15681	15760	9	8	11
45	19889	19843	19951	9	8	11
50	24564	24510	24611	10	9	11
55	29726	29675	29766	10	9	12
60	35380	35317	35420	10	9	11
65	41506	41446	41551	10	9	11
70	48161	48103	48208	10	9	11
75	55294	55224	55362	10	9	11
80	62908	62831	62983	10	9	12
85	71029	70943	71106	10	9	11
90	79647	79559	79720	10	9	11
95	88722	88619	88835	10	9	11
100	98315	98214	98432	10	9	13

l(km) is the edgelength of the square area. The route density $\rho = 10/\text{km}^2$.
Γ is the maximum number of directly or indirectly connected routes. The centralized algorithm incurs a time cost $\Phi^{cen} = \Omega(\Gamma)$.
D is the maximum number of routes passing through a node. The distributed algorithm incurs a time cost $\Phi^{dis} = O(D)$.
Thirty trials are carried out for each l. The data shows the distributed algorithm has better scalability than the centralized algorithm.

in 4sec, then the control traffic bandwidth should be $480 \times 8 \times 100/4 = 96$ (kbps). Note according to Section 6.1, a RICH base station can achieve 1.8Mbps transmitting bandwidth with lightweight hardware. Also note the above bound is based on worst case analysis. In practical applications, if more detailed network information, such as the number of routes passing through each router is available, then different router nodes can reserve different bandwidth based on this information.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we study the optimal sampling rate assignment in RTWSN, and formalize it into a non-linear optimization problem. 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 and even moderately large RTWSNs, 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. Note that using the GPS for synchronization in the distributed algorithm is not an obligation, asynchronous algorithm can be designed, for example, similar to the asynchronous flow control algorithm in [Low and Lapsley 1999]. However, asynchronous algorithm usually converges slower.

Our on-going research topics include: (1) Integrating QoS optimization and error modeling for WSN; (2) Theoretical analysis of distributed algorithm's convergence rate for 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 providing and giving helpful advice on the COPL_LC package. We would also like to thank Professor Steven Low at CalTech for the discussion on the distributed algorithm.

REFERENCES

Akkaya, K. and Younis, M. 2005. A survey of routing protocols in wireless sensor networks. *Elsevier Ad Hoc Network Journal 3,* 3.

Bertsekas, D. 1995. *Nonlinear Programming.* Athena Scientific, Belmont, MA.

Bertsekas, D. and Tsitsiklis, J. 1989. *Parallel and Distributed Computation.* Prentice Hall.

Bolot, J.-C., Turletti, T., and Wakeman, I. 1994. Scalable feedback control for multicast video distribution in the internet. In *SIGCOMM.* 58–67.

Braginsky, D. and Estrin, D. 2002. Rumor routing algorithm for sensor networks. In *International Conference on Distributed Computing Systems.*

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

Caccamo, M., Zhang, L., Sha, L., and Buttazzo, G. 2002. An implicit prioritized access protocol for wireless sensor networks. In *Proc. of IEEE Real-Time Systems Symposium'02.*

DRCL. 2004. Drcl j-sim [online]. Available at: http://www.j-sim.org.

Exit Consulting. 2004. Gps synchronization clock – model 200. Available at: http://www.gpsclock.com/specs.html.

Getting, I. A. 1993. Perspective/navigation-the global positioning system. *Spectrum, IEEE 30,* 36–38, 43–47.

Ghosh, S., Rajkumar, R., Hansen, J., and Lehoczky, J. 2003. Scalable resource allocation for multi-processor qos optimization. In *Proc. of the 23rd IEEE Intl. Conf. on Distributed Computing Systems (ICDCS 2003).* Providence, RI.

Giannecchini, S., Caccamo, M., and Shih, C. 2004. Collaborative resource allocation in wireless sensor networks. In *IEEE Euromicro Conference on Real-Time Systems.* Catania, Italy.

He, T., Stankovic, J. A., Lu, C., and Abdelzaher, T. 2003. Speed: A stateless protocol for real-time communication in sensor networks. In *Intl. Conf. on Distributed Computing Systems (ICDCS 2003).* Providence, RI.

Heinzelman, W., Kulik, J., and Balakrishnan, H. 1999. Adaptive protocols for information dissemination in wireless sensor networks. In *5th ACM/IEEE Mobicom Conference.*

Karp, B. and Kung, H. 2000. Greedy perimeter stateless routing for wireless networks. In *Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000).* Boston, MA.

Kelly, F. 1997. Charging and rate control for elastic traffic. *European Tran. on Telecommunications 8.*

Kelly, F., Maulloo, A., and Tan, D. 1998. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society 49*, 237–252.

Kurose, J. F. and Simha, R. 1989. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Trans. Comput. 38,* 5, 705–717.

Lee, C., Lehoczky, J., Siewiorek, D., Rajkumar, R., and Hansen, J. 1999. A scalable solution to the multi-resource qos problem. In *Proc. of the IEEE Real-Time Systems Symposium.*

Liu, X., Wang, Q., Sha, L., and He, W. 2003. Optimal qos sampling frequency assignment for real-time wireless sensor networks. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium.* IEEE Computer Society, 308.

Low, S. H. and Lapsley, D. E. 1999. Optimization flow control, i: Basic algorithm and convergence. *IEEE/ACM Tran. on Networking 7,* 6 (Dec.), 861–75.

Luenberger, D. 1984. *Linear and Nonlinear Programming.* Addison-Wesley, Reading, Massachusetts.

Muqattash, A. and Krunz, M. 2003. Cdma-based mac protocol for wireless ad hoc networks. In *Proceedings of the 4th ACM Intl' Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003).* Annapolis, Maryland, USA, 153–164.

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

Price, R. and Jr., P. E. G. 1958. A communication technique for multipath channels. *Proceedings of the IRE 46*, 555–570.

Qualcomm. 2004a. Csm2000 cell site modem [online]. Available at:
http://www.cdmatech.com/solutions/products/csm2000.jsp.

Qualcomm. 2004b. Csm5000 cell site modem [online]. Available at:
http://www.cdmatech.com/solutions/products/csm5000.jsp.

Qualcomm. 2004c. Csm5500 cell site modem [online]. Available at:
http://www.cdmatech.com/solutions/products/csm5500.jsp.

Rajkumar, R., Lee, C., Lehoczky, J., and Siewiorek, D. 1997. A resource allocation model for qos management. In *Proc. of IEEE Real-Time Systems Symposium.*

Rudin, W. 1976. *Principles of Mathematical Analysis.* McGraw-Hill Inc.

Seto, D., Lehoczky, J. P., Sha, L., and Shin, K. G. 1996. On task schedulability in real-time control systems. In *Proc. of IEEE Real-Time Systems Symposium'96.*

Sha, L., Liu, X., Caccamo, M., and Buttazzo, G. 2000. Online control optimization using load driven scheduling. In *Proc. of the 39th IEEE Conf. on Decision and Control.*

Stoica, I., Abdel-Wahab, H., Jeffay, K., Baruah, S., Gehrke, J., and Plaxton, C. G. 1996. A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems. In *IEEE Real-Time Systems Symposium.*

Viterbi, A. J. 1995. *CDMA: Principles of Spread Spectrum Communication.* Prentice Hall.

Waldspurger, C. A. and Weihl, W. E. 1994. Lottery scheduling: Flexible proportional-share resource management. In *Operating Systems Design and Implementation.* 1–11.

Xu, Y., Heidemann, J., and Estrin, D. 2001. Geography-informed energy conservation for ad hoc routing. In *International Conference on Mobile Computing and Networking.* Rome, Italy.

Ye, Y. 1997a. *Interior Point Algorithms: Theory and Analysis.* Wiley.

Ye, Y. 1997b. User's guide of *copl_lc*, computational optimization program library: Linearly constrained convex programming. Available at:
http://www.stanford.edu/~yyye/Col.html.

Zhao, J. and Govindan, R. 2003. Understanding packet delivery performance in dense wireless sensor networks. In *First international conference on Embedded networked sensor systems.* LA, CA.

Zhou, G., He, T., Krishnamurthy, S., and Stankovic, J. A. 2004. Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services (MobiSys '04).* ACM Press, 125–138. Boston, MA, USA.

A. A BRIEF TUTORIAL ON DSSS-CDMA

DSSS is a physical layer baseband modulation/demodulation scheme for digital communication. Without loss of generality, we assume digital "1" and "0"s are represented by $+1$ and -1 (volt) rectangular pulses. Unlike conventional baseband modulation schemes, where each bit is represented with a *single* $+1$ or -1 pulse, DSSS multiplies a *Pseudo Noise* (PN) sequence onto the stream of user data bits, as shown in Fig. 9.

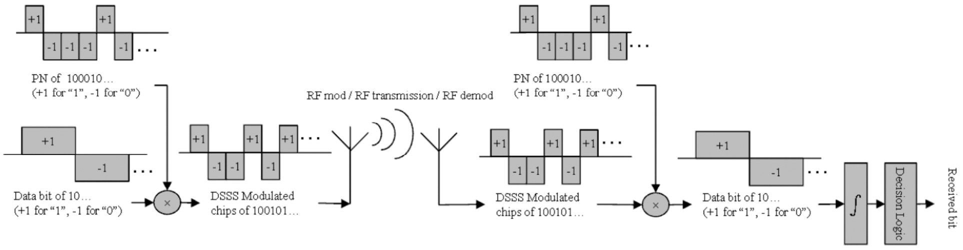


Fig. 9. DSSS modulation/demodulation process: a simplified view

The PN sequence is also a sequence of ± 1 rectangular pulses, with a $+1$ pulse representing digit "1" and a -1 pulse for digit "0". Each digit of a PN sequence is called a *chip*. The number of PN chips generated per second is called *chip rate*, represented by r_c, chip duration T_c is defined as $T_c \overset{def}{=} \frac{1}{r_c}$. Correspondingly, the number of data bits generated per second is called *bit rate*, represented by r_b, and bit duration $T_b \overset{def}{=} \frac{1}{r_b}$. Usually r_c is a positive integer multiple of r_b, the ratio is called *processing gain*, denoted as $g \overset{def}{=} \frac{r_c}{r_b}$. According to the DSSS modulation scheme, each modulated bit consists of g chips (see Fig. 9). The process of DSSS modulation, i.e. multiplying PN sequence chips onto user data bits is often called *scrambling.*

At the demodulator, if the same PN sequence with 0 phase shift (i.e., synchronized, or say, *coherent*) is again multiplied to the scrambled data chip stream, the original user data bit stream (the ± 1 sequence at bit rate r_b) recovers. If the received data stream is scrambled with another PN sequences, or the phase shift is more than a chip, the data stream can not be recovered, instead, it looks like a stream of random ± 1s generated by independently flipping a fair coin at chip rate r_c. The multiplication carried out at the demodulator is also called *descrambling.* Next, by integrating over T_b units of time, a decision logic can decide whether a wanted user data bit is received or not.

Therefore, a specific PN sequence decides a unique DSSS data transmission *channel.* Different data streams scrambled with different PN sequences are allowed to occupy the same RF spectrum. In the time domain, data streams of different DSSS channels may be sent out in parallel without TDMA (Time Division Multiple Access). The matching PN sequence at the receiver can filter out the wanted user data signal from the shared spectrum. Such parallel multiple access scheme is called *Code Division Multiple Access* (CDMA). Note DSSS is a physical layer concept, while CDMA is a MAC layer concept. Other MAC layer schemes, such as TDMA can also be deployed on top of DSSS physical layer.

Quantitatively, a number of important features of DSSS communication is captured by its *Bit Error Rate* (BER) upper bound (27), which assumes QPSK RF modulation, and per connection pilot tone [Viterbi 1995][Muqattash and Krunz 2003] (different implementation alternatives may affect details of the formula, but will not cause fundamental differences):

$$\mathcal{P}_{ber} \leq \exp\left(-\frac{gP_u}{J + \sum_{i=1, i\neq u}^{\Xi} P_i + \sum_{h=1}^{H} A_h + P_u}\right) \tag{27}$$

where $\mathcal{P}_{ber}$ is the BER; g is processing gain; J is the received power of *External RF Interference* (EI), which specifically refers to EMI, thermal noise and the RF interference from RF devices that are turned on accidentally or maliciously. P_i $(i = 1 \ldots \Xi)$ is the received power of CDMA channel i, Ξ is the total number of received CDMA channels. u is the intended channel, whose corresponding received power is P_u. Each transmitting node may send out several CDMA channels in parallel. To ease the reception, the node may also transmit an additional pilot tone. In (27) the pilot tone of transmitting node h $(h = 1, \ldots, H)$ is of power A_h. $\sum_{i=1, i\neq u}^{\Xi} P_i + \sum_{h=1}^{H} A_h$ is therefore the upper bound of total *Multiple Access Interference* (MAI), i.e. the interference caused by other CDMA channels and pilot tones received in parallel with the intended channel. Note P_u also appears in the denominator, adding up to the total interference power. This is to provide a pessimistic estimation on *Inter Symbol Interference* (ISI), which is usually a result of multipath fading. To simplify, we can merge $\sum_{i=1, i\neq u}^{\Xi} P_i$ and P_u together to be denoted as $\sum_i P_i$. The $gP_u/(J + \sum_i P_i + \sum_h A_h)$ part shows the effective SNR for the intended channel, where $J + \sum_i P_i + \sum_h A_h$ represents the upper bound of noise power and gP_u represents effective signal power. The bigger the SNR, the smaller the probability of bit error $\mathcal{P}_{ber}$. When $\mathcal{P}_{ber}$ is below a certain threshold Θ_{ber}, the wireless communication is acceptable for real-time communication. Therefore, to maintain a real-time DSSS-CDMA channel in fact means to maintain the SNR of the channel from dropping below an acceptable threshold Θ_{snr}.

(27) implies that SNR of the intended channel can be raised by increasing the processing gain g. Meanwhile, g is defined as the ratio of chip rate and bit rate: $g \overset{def}{=} r_c/r_b$. Usually, chip rate r_c is fixed by hardware because of multipath effect and hardware cost constraints [Price and Jr. 1958][Viterbi 1995], therefore raising processing gain means slowing down user data bit rate r_b. DSSS hereby provides a mechanism to leverage between SNR and data bit rate.

B. PROOF OF THEOREM 3.1

Using Lagrangian multipliers $\lambda_j, j = 1, \ldots, N$, and $\mu_i, i = 1, \ldots, M + N$, we can write the Kuhn-Tucker condition of $MCOP(U, \mathbf{A}, W)$ as:

$$\frac{dU_j(f_j^*)}{df_j} + \mu_1\mathbf{A}'_{1j} + \ldots + \mu_{(M+N)}\mathbf{A}'_{(M+N)j} - \lambda_j = 0, \tag{28}$$

$$\text{where } (j = 1, \ldots, N)$$

$$f_j^{min} - f_j^* \leq 0 \tag{29}$$

$$\lambda_j(f_j^{min} - f_j^*) = 0 \tag{30}$$

$$\sum_{j=1}^{N} \mathbf{A}'_{ij} f_j^* \leq W'_i, (i = 1, \ldots, M+N) \tag{31}$$

$$\mu_i(\sum_{j=1}^{N} \mathbf{A}'_{ij} f_j^* - W'_i) = 0 \tag{32}$$

$$\mu_i \geq 0, \lambda_j \geq 0, (i = 1, \ldots, M+N, j = 1, \ldots, N) \tag{33}$$

Suppose for all $i = 1, \ldots, M+N$, $\sum_{j=1}^{N} \mathbf{A}'_{ij} f_j^* < W'_i$, then from (32), we know $\mu_i = 0$. Then $\frac{dU_j(f_j^*)}{df_j} + \mu_1\mathbf{A}'_{1j} + \ldots + \mu_{(M+N)}\mathbf{A}'_{(M+N)j} - \lambda_j = \frac{dU_j(f_j^*)}{df_j} - \lambda_j < 0$, since $\frac{dU_j(f_j^*)}{df_j} < 0$. This contradicts with equation (28). So we know Theorem 3.1 holds.

C. CONVERSION OF MCOP TO COPL_LC

The original COPL_LC package is used to solve the problem of equality constraints as follows:

$$\begin{aligned} \min\ g &\ (x) \\ \text{s.t.: } Tx &= b, x \geq 0 \\ \text{where } \mathbf{A} &\in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m \end{aligned}$$

This is NOT the form of our problem formulation of MCOP. We have to do some transformations to transform MCOP into the framework of COPL_LC. Here, we use the combined constraints set (17) of MCOP, *i.e.*

$$\min_{(f_1,\ldots,f_N)} \sum_{j=1}^{N} U_j(f_j) \qquad (10)$$

$$\text{s.t.: } \mathbf{A}'f \leq W' \qquad (17)$$

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

$$f \geq f^{min} \qquad (13)$$

First, we let $\tilde{f}_j = f_j - f_j^{min}$, so that the constraints $f_j \geq f_j^{min} \Leftrightarrow \tilde{f}_j \geq 0$.

We also add a slack variable $y = (y_1, \ldots, y_{M+N})^\mathsf{T}$, so that

$$\begin{aligned}\mathbf{A}'f \le W' &\Leftrightarrow \mathbf{A}'f + y = W', y \ge 0\\ &\Leftrightarrow \mathbf{A}'\tilde{f} + y = W' - \mathbf{A}'f^{min}, y \ge 0\end{aligned}$$

MCOP is hereby transformed to the following form:

$$\min_{(\tilde{f}_1,\ldots,\tilde{f}_N)} \sum_{j=1}^{N} U_j(\tilde{f}_j)$$

$$\begin{aligned}\text{s.t.: } \tilde{f}_j &\ge 0 (j = 1, \ldots, N)\\ y &\ge 0\\ \sum_{j=1}^{N} \mathbf{A}'_{ij}\tilde{f}_j + y_i &= W'_i - \sum_{j=1}^{N} \mathbf{A}'_{ij} f_j^{min},\\ (\text{where } i &= 1, \ldots, M+N).\end{aligned}$$

This is in the form of COPL_LC. To see this, simply let $x = [\tilde{f}_1, \ldots, \tilde{f}_N, y_1, \ldots, y_{M+N}]^{\mathsf{T}}$, $T = [\mathbf{A}'_{(M+N)\times N} | \mathbf{I}_{(M+N)\times(M+N)}]$, where $\mathbf{I}_{(M+N)\times(M+N)}$ is the $(M+N)\times(M+N)$ identity matrix and $b = W' - \mathbf{A}'f^{min}$.

D. PROOF OF THEOREM 5.1

First by defining $I_j = [f_j^{min}, f_j^{max}]$ and $V(\bullet) = -U(\bullet)$, we can rewrite the $MCOP(U, \mathbf{A}, W)$ as:

$$\textbf{Primary: } \max_{f_j \in I_j} \textstyle\sum_{j=1}^{N} V_j(f_j) \tag{34}$$

$$\text{st: } \textstyle\sum_{j=1}^{N} \mathbf{A}_{ij} f_j \le W_i \qquad (i = 1, \ldots, M) \tag{35}$$

We call the above constraint optimization problem as Primary problem. The following proof sketch follows Low and Lapsley's result in [Low and Lapsley 1999] with modifications for our problem.

Let's first convert the Primary problem to its dual form.

Define the Lagrangian with multipliers vector p for the Primary problem as:

$$\begin{aligned}L(f,p) &= \sum_{j=1}^{N} V_j(f_j) - \sum_{i=1}^{M} p_i(\sum_{j=1}^{N} \mathbf{A}_{ij} f_j - W_i)\\ &= \sum_{j=1}^{N}(V_j(f_j) - f_j \sum_{i=1}^{M} \mathbf{A}_{ij} p_i) + \sum_{i=1}^{M} p_i W_i.\end{aligned}$$

Notice the first term is separable in f_j, and hence:

$$\max_{f_j \in I_j} \sum_{j=1}^{N}(V_j(f_j) - f_j \sum_{i=1}^{M} \mathbf{A}_{ij} p_i) = \sum_{j=1}^{N} \max_{f_j \in I_j}(V_j(f_j) - f_j \sum_{i=1}^{M} \mathbf{A}_{ij} p_i). \tag{36}$$

Then, by defining the objective function:

$$\begin{aligned} D(p) &= \max_{f_j \in I_j} L(f,p) \\ &= \sum_{j=1}^{N} B_j(p^j) + \sum_{i=1}^{M} p_i W_i, \end{aligned}$$

where

$$B_j(p^j) = \max_{f_j \in I_j} (V_j(f_j) - f_j p^j), \tag{37}$$

$$p^j = \sum_{i=1}^{M} A_{ij} p_i. \tag{38}$$

The dual problem is:

$$\textbf{Dual:} \ \min_{p \geq 0} D(p). \tag{39}$$

We call the unique optimizer of (37) as $f_j(p^j)$. From Kuhn-Tucker theorem, it is easy to see:

$$f_j(p^j) = [V_j'^{-1}(p^j)]_{f_j^{min}}^{f_j^{max}}, \tag{40}$$

where $[z]_a^b = \min\{\max\{z, a\}, b\}$, $V_j'^{-1}(\bullet)$ represents the inverse of derivative function $V_j'(\bullet)$ (with respect to f_j).

The three Lemmas given below are used in the proof of Theorem 5.1.

LEMMA D.1. *Under assumptions A1, A2, the dual objective function $D(p)$ is convex, lower bounded, and continuously differentiable.*

PROOF. Directly follow assumption A1, A2. □

For any price vector p, define $\beta_j(p)$ by

$$\beta_j(p) = \begin{cases} \frac{1}{-V_j''(f_j(p))} & \text{if } V_j'(f_j^{max}) \leq p^j \leq V_j'(f_j^{min}) \\ 0 & \text{otherwise.} \end{cases} \tag{41}$$

Here, $f_j(p)$ is the unique maximizer of (37), which is defined in equation (40). Let $\mathbf{B}(p) = Diag(\beta_j(p)), (j = 1, \ldots, N)$ be the $N \times N$ diagonal matrix. Note from assumption **A3**, we know for $\forall p \geq 0$, $\frac{1}{-V_j''(f_j(p))} = \frac{1}{U_j''(f_j(p))} \leq \overline{\alpha_j}$, so

$$0 \leq \beta_j(p) \leq \overline{\alpha_j} \leq \infty. \tag{42}$$

LEMMA D.2. *Under assumption A1, A2, the Hessian matrix of D is given by $\nabla^2 D(p) = \mathbf{A}\mathbf{B}(p)\mathbf{A}^{\mathsf{T}}$, where it exists.*

PROOF. Let $\frac{\partial f}{\partial p}(p)$ denote the $N \times M$ Jacobian matrix whose (j, i) element is $\frac{\partial f_j}{\partial p_i}(p)$. When it exists,

$$\frac{\partial f_j}{\partial p_i}(p) = \begin{cases} \frac{\mathbf{A}_{ij}}{V_j''(f_j(p))} & \text{if } V_j'(f_j^{max}) \le p^j \le V_j'(f_j^{min}) \\ 0 & \text{otherwise} \end{cases}.$$

using (42), we have:

$$\left[\frac{\partial f_j}{\partial p_i}\right] = -\mathbf{B}(p)\mathbf{A}^{\mathsf{T}}. \tag{43}$$

We know $\frac{\partial D}{\partial p_i}(p) = W_i - \sum_{j=1}^{N} \mathbf{A}_{ij} f_j(p)$, *i.e.* $\nabla D(p) = W - \mathbf{A}f(p)$, hence

$$\nabla^2 D(p) = -\mathbf{A}\left[\frac{\partial f}{\partial p}(p)\right], \tag{44}$$

substitute equation (43) into (44) yields the results. □

LEMMA D.3. *Under conditions A1 $\sim$ A3, ∇D is Lipschitz with $\|\nabla D(q) - \nabla D(p)\|_2 \le \overline{\alpha}\overline{L}\overline{S}$, for $\forall p, q \ge 0$.*

PROOF. Using Lemma D.2, we will show that $\nabla^2 D(p) = \|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_2 \le \overline{\alpha}\overline{L}\overline{S}$. The lemma then follows from [Rudin 1976]. We know:

$$\|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_2^2 \le \|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_\infty \|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_1$$

i.e. $\|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_2$ is upper bounded by the product of the maximum row sum and the maximum column sum of the $M \times M$ matrix $\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}$. Since $\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}$ is symmetric, $\|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_1 = \|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_\infty$, and hence:

$$\begin{aligned} & \|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_2 \le \|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_\infty \\ = & \max_l \sum_{l'} [\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}]_{ll'} \\ = & \max_l \sum_{l'} \sum_j \beta_j(\omega)\mathbf{A}_{lj}\mathbf{A}_{l'j} \\ = & \max_l \sum_j \beta_j(\omega)\mathbf{A}_{ij}|L(j)|. \end{aligned}$$

By definition $|L(j)| \le \overline{L}$, $\beta_j(\omega) \le \overline{\alpha}$, and hence $\|\mathbf{A}\mathbf{B}(\omega)\mathbf{A}^{\mathsf{T}}\|_2 \le \overline{\alpha}\overline{L} \max_l \sum_j |\mathbf{A}_{lj}| \le \overline{\alpha}\overline{L}\overline{S}$. □

Proof of Theorem 5.1: The dual objective function D is lower bounded and ∇D is Lipschitz from Lemma D.1 and D.3. Then any accumulation point p^* of the sequence $\{p(s)\}$ generated by he gradient projection algorithm (*i.e.*, the distributed algorithm) for the dual problem is dual optima [Bertsekas and Tsitsiklis 1989].

Let $\{p(s)\}, s = 1, 2, \ldots$ be a subsequence converging to p^*. At least one exists since the level set $\{p \ge 0 | D(p) \le D(p(0))\}$ of D is compact and that the sequence $\{D(p(s))\}$ is decreasing in s and hence in the level set, provided $0 < \gamma < 2/(\overline{\alpha}\overline{L}\overline{S})$. To show that the subsequence $\{f(s) = f(p(s))\}, s = 1, 2, \ldots$ converges to the primal optimal node rate $f^* = f(p^*)$, note that $V_j'(f_j)$ is defined on a compact set I_j.

Moreover it is continuous and one-to-one, and hence its inverse is continuous on $[f_j^{min}, f_j^{max}]$ [Rudin 1976]. From (40), $f(p)$ is continuous. Therefore $\lim_{s\to\infty} f(s) = f(p^*)$. Because our Primal problem is the same as $MCOP(U, \mathbf{A}, W)$, so Theorem 5.1 holds.