A Multi-Channel Bulk Data Collection for Structural Health Monitoring using Wireless Sensor Networks

Takuto Kuroiwa*, Makoto Suzuki*, Yasutaka Yamashita*,
Shunsuke Saruwatari†, Tomonori Nagayama‡ and Hiroyuki Morikawa*
*Research Center for Advanced Science and Technology, The University of Tokyo
{kuroiwa, makoto, yasu, mori}@mlab.t.u-tokyo.ac.jp
†Faculty of Informatics/Graduate School of Informatics, Shizuoka University
saru@inf.shizuoka.ac.jp
‡Department of Civil Engineering, The University of Tokyo
nagayama@bridge.t.u-tokyo.ac.jp

***Abstract*—Data-intensive wireless sensor network applications, such as structural health monitoring and earthquake monitoring, require high throughput bulk data collection. Based on the fact that each node stores the same amount of sensor data, we propose a Maximum-Subtree-First Collection Protocol (MSFCP), which adopts Maximum-Subtree-First scheduling on top of multi-channel block transfer to maximize overall throughput. We present the theoretical analysis that MSFCP can achieve optimal throughput in the ideal propagation environment, and we achieve overall throughput of 135 kbps on the IRIS Mote platform.**

***Index Terms*—Wireless sensor networks, bulk transfer, multi-channel, scheduling**

I. Introduction

Wireless sensor networks are increasingly used in data-intensive applications such as structural health monitoring[1], [2], earthquake monitoring[3], and volcano monitoring[4]. These applications require bulk transfer of large amounts of sensed data to a common sink, typically over a tree-based routing topology. For this class of applications, it is critical to maximize throughput.

The nature of wireless communications brings several challenges for efficient and reliable collection in multi-hop settings; these include inter-path and intra-path interference [5]. Inter-path interference occurs between multiple flows, while intra-path interference occurs between multiple hops of the same flow, where a flow is defined as a set of nodes between the sink and a source that is actively transmitting packets to the sink.

Existing bulk transfer protocols such as Flush [6] and PIP [7] consider only one active flow at a time to eliminate the issue of inter-path interference. A typical sensor node is equipped with a single half-duplex transceiver, which can either transmit or receive only one packet at any time. Therefore, in Flush and PIP, the sink is idle over half the time since the child of the sink can be continuously active by either transmitting or receiving, whereas the sink only receives, as shown in Fig. 1.

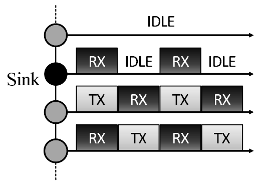
We propose here the *Maximum-Subtree First Collection Protocol* (MSFCP), which collects data from multiple flows to maximize the overall throughput. MSFCP utilizes multiple channels to eliminate intra-path interference. This protocol also uses CSMA-based block transfer to mitigate overhead including channel switching and ACKs and adopts Maximum-Subtree-First (MSF) scheduling to reduce inter-path interference and utilize the sink capability.

We present a theoretical analysis in which MSF scheduling achieves the optimal throughput under an ideal communication environment, and we describe our evaluation of overall throughput on various routing trees using the IRIS Mote platform. In our laboratory experiments, achieved throughput is up to 135 kbps.

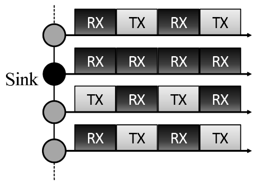
The rest of this paper is organized as follows. In section II, we discuss prior work on data collection. In section III, we describe the design of MSFCP. Section IV presents an analysis of collection time under an ideal communication environment. Section V discusses the experimental evaluation on the IRIS Mote platform. Finally, in section VI, we conclude the paper and discuss our future work.

II. Related Work

Traffic patterns of data collection in wireless sensor networks are mainly classified into two types, as indicated in TABLE I: low-power data collection and high-throughput bulk data collection. Applications such as environmental monitoring, habitat monitoring, and data center monitoring periodically collect data at a low data-rate over a long period. To save energy and improve the network lifetime, sensor data are typically aggregated to eliminate redundancy and to minimize the number of transmissions [8]. In contrast, our target applications such as structural health monitoring and volcano monitoring attempt to acquire a large amount of data at a high data-rate, typically over 100 Hz. For signal analysis or other purposes, these applications collect complete data from nodes without aggregation. Because it is not feasible to



Collection from only one flow at a time



Collection from multiple flows at the same time

Fig. 1. Data collection from *one flow/multiple flows*

TABLE I
COLLECTION PROTOCOLS IN WIRELESS SENSOR NETWORKS

Traffic pattern	Applications	Protocols
Low-power data collection	Environmental monitoring Wildlife monitoring Data center monitoring	CTP [9] BCP [10]
High-throughput bulk data collection	Structural health monitoring Earthquake monitoring Volcano monitoring	Fetch [4] Flush [6] PIP [7] SSMH [2]

collect such a huge amount of data in real time, nodes store sensor data in flash memory while sensing and then deliver the data to a common sink.

Several protocols exist for low-power data collection, for example, Collection Tree Protocol (CTP) [9] and Backpressure Collection Protocol (BCP) [10]. CTP is widely used in wireless sensor networks, and is a routing approach where a minimum cost tree with respect to Expected Transmission Count (ETX) [11] is dynamically constructed based on a link quality estimation. BCP is an integrated routing and scheduling approach based on backlogged queue information.

These periodical data collection protocols are designed for long term collection and mainly focus on handling topological changes to maintain appropriate routes. These protocols were developed for relatively low traffic rates, and achievable throughput is limited to a little more than 10 kbps[12].

Several other protocols exist for high-throughput bulk data collection, including Fetch [4], Flush [6], and PIP [7]. Fetch was the first bulk transfer protocol in wireless sensor networks; it achieves a throughput of only 1 kbps because of intra-path and inter-path interference. Flush [6] and PIP [7] mitigate these interferences. Flush reduces intra-path interference by the end-to-end rate control and achieves a throughput of about 10 kbps. PIP uses multiple channels statically assigned by hop count to reduce intra-path interference and achieves a throughput of about 60 kbps.

Flush and PIP collect data sequentially from nodes, i.e., only one flow at a time, to eliminate inter-path interference. These sequentially collecting protocols achieve only half the maximum throughput because the sink becomes idle when its active child node is receiving.

In our previous study, we proposed a single-sink multi-hop communication protocol called SSMH, which employs link-by-link block data transfer using multiple RF channels for collecting large amounts of data [2]. SSMH adopts a sender-initiated greedy scheduling strategy and does not ensure maximum achievable throughput. Unlike SSMH, MSFCP adopts a receiver-initiated scheduling strategy to achieve close to the optimal throughput for an arbitrary topology. Note that every node stores the same amount of data in our target applications, and MSFCP takes advantage of this application-specific characteristic.

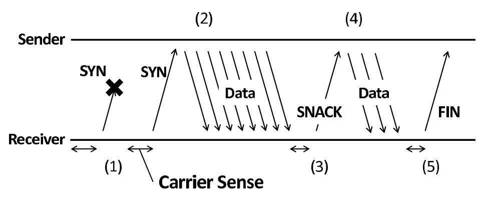


Fig. 2. Block transfer

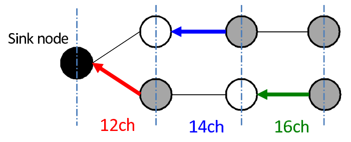


Fig. 3. Channel allocation and transmission scheduling

III. PROTOCOL DESIGN

We present a design of Maximum-Subtree-First Collection Protocol (MSFCP) here. MSFCP uses multi-channel block transfer and adopts Maximum-Subtree-First (MSF) scheduling to reduce interference and enhance overall throughput.

A. Channel Assignment

IEEE standard 802.15.4 specifies 16 different non-overlapping channels in the 2.4-GHz ISM band. If we utilize available multiple channels effectively, we can exploit parallel transmission to enhance throughput. For high throughput, it is essential to assign channels to links to minimize interference.

There are two ways to assign multiple channels: dynamic channel assignment and static channel assignment. Although dynamic channel assignment schemes can reduce interference to some degree, frequent negotiations result in large overhead. In contrast, static channel assignment schemes work with less overhead, but they are not suitable for rapidly changeable propagation environment [13].

In practical deployment, interference occurs between nodes within a few hops [6], [7], and hence, we assign a communication channel statically based on the hop count to eliminate intra-path interference. This hop count based channel assignment enables not only packet pipelining as in PIP, but also reduces inter-path interference to some extent because it occurs within nodes at the same hop count. Thus, to avoid inter-path interference, multiple nodes at the same hop count should not send data at the same time.

B. Block Transfer

We use a *block transfer* where a sender transfers a block of multiple packets. This block transfer technique mitigates overhead such as channel switching and control packets including ACKs. Although transferring a block requires more memory than transmitting a packet, it reduces the uncertain time delay, and the transmission delay over each link becomes almost equal.

The sequence of the block transfer is shown in Fig. 2. (1)The receiver initiates the block transfer by sending SYN packets to the sender repeatedly until receiving a data packet from the sender. (2) After receiving a SYN packet, the sender starts to send multiple data packets (a block) without performing clear channel assessment (CCA) or random backoff. This mechanism shortens the uncertain transmission delay. (3) After receiving the last packet in a block, if there are any missing packets, the receiver transmits a Selective Negative ACK (SNACK) that indicates which packet to retransmit. When the last packet in a block is lost, the receiver uses a timer to send a SNACK packet. The sender that receives a SNACK packet transmits packets from the block that are missing at the receiver. If the receiver has received all packets correctly, it transmits a FIN packet to complete the block transfer.

C. Maximum-Subtree-First Scheduling

MSFCP adopts MSF scheduling, which is unsynchronized distributed scheduling based on its own transmission buffer information.

Each node stores the same amount of data in our target application. Because of this, and also due to the characteristics of block transfer, we can reduce interference and enhance the overall throughput by using a simple strategy.

We describe in the following the MSF scheduling algorithm, which is run locally by each node. The key idea is to schedule transmission in parallel along multiple branches of the tree, and to keep the sink as busy receiving as possible. We assume that all the nodes are aware of the number of nodes in each subtree. This can be easily obtained by gathering the routing information. The subtree of a node is defined as a tree that has the child of the node as its root, as shown in Fig. 4.

1) All source nodes wait for their parent's send request if their buffer is full.
2) All nodes choose the subtree with the largest number of total remaining blocks, i.e., *maximum subtree*, and they request the subtree root to send data if their buffer is empty.

The sink always follows rule 2 (the sink's buffer is treated as always empty). The sink chooses subtrees whose roots have a full buffer, that is, subtrees other than the subtree whose root previously sent a block to the sink. This strategy can keep the sink as busy as possible and enhance the overall throughput.

In addition, this MSF scheduling ideally realizes one flow at each hop at a time. If only node N_1 is transmitting at hop h, only N_1 is receiving at the next hop $h+1$ after completing the transmission.

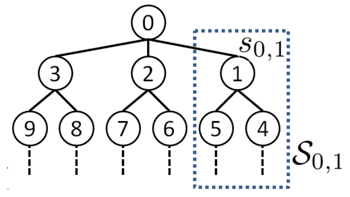


Fig. 4. Subtrees

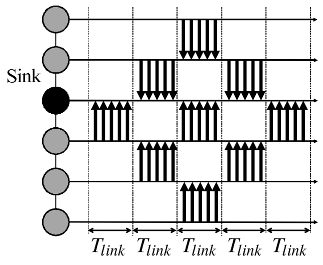


Fig. 5. Slotted behavior of nodes with duration T_{link}.

IV. Mathematical Analysis of Collection Time

A. Lower Bound on Collection Time

In this section, we consider a lower bound on the collection time for a given topology. For simplicity, overhead including channel switching is ignored, all packets can be received, and topology is fixed during the collection. We assume that every node has a similar amount of data, and we denote the transmission time of the block transfer for all links as T_{link}. This assumption leads to a slotted transmission behavior of nodes, as depicted in Fig. 5. We define a *slot* as the duration of T_{link} for transmitting a block of data.

Because of the half-duplex transceivers, when the root $s_{0,j}$ of $\mathcal{S}_{0,j}$ is fully busy in either receiving or transmitting, $s_{0,j}$ receives $n_j - 1$ packets from the children and transmits n_j packets to the sink. Hence, the minimum time of collection from a subtree $T_{LB,j}$ is described as follows:

$$T_{LB,j} = (2n_j - 1)T_{link}. \tag{1}$$

This leads to the following theorem.

Theorem 1 (Lower bound of collection time) *The lower bound of overall collection time T_{LB} is described by*

$$T_{LB} = \max(N, 2n_{max} - 1) \cdot T_{link}, \tag{2}$$

where N is the number of sources in the network and n_{max} is the maximum number of nodes in any subtree of the sink.

Proof *Since the nodes cannot receive multiple packets simultaneously, NT_{link} is a trivial lower bound to receive all packets. However, as in the above discussion, we need at least $(2n_{max} - 1)T_{link}$ for the largest subtrees.* □

B. Collection Time of MSF Scheduling

In the following, we prove that MSF scheduling can achieve exactly the minimal collection time presented in Theorem 1. We start to show that all nodes except the sink can always receive soon after their buffer becomes empty; that is, nodes can always follow rule 2 of MSF scheduling. Then, we describe the recurrence formula for the remaining blocks in each subtree at each 2 k slots. Finally, we derive the recurrence formula between the total number of nodes and the total time of collection, and prove that MSF scheduling needs exactly the minimum time of Theorem 1 to complete collection. We define a subtree that has remaining blocks to transfer as an *active* subtree.

Lemma 1 *By applying MSF scheduling, a node with an empty buffer at slot k can receive all its children which are the root nodes of the active subtrees.*

Proof *We prove this by induction of the slot.* (i) *For all nodes with an empty buffer at some slot* $k \geq 1$*, suppose that they can receive from all children that are the roots of the active subtrees. Then, any node i with an empty buffer receives from a child* $s_{i,j}(= i')$*, and the buffer node* i' *is empty at the beginning of the next slot* $k+1$*. We now consider child* $s_{i',j'}$ *of node* i'*. The buffer of* $s_{i',j'}$ *is empty or full at slot k. If node* $s_{i',j'}$ *has an empty buffer at slot k, by the above assumption, it can receive from one of its children that are roots of its active subtrees. If node* $s_{i',j'}$ *has a full buffer at slot k, it is idle at the slot because its parent* i' *is communicating with node i. Hence, at slot* $k+1$*, node* i' *can receive all the active children. This holds for all nodes with empty buffers at slot* $k+1$*.*

(ii) *At the beginning of collection, i.e., when* $k = 1$*, all nodes have a block in their buffers, and a sink's child* $s_{0,j}$ *begins a block transfer. Then, at slot 2, the* $s_{0,j}$*'s buffer is empty, and* $s_{0,j}$ *is obviously able to receive from all its children.*

(i) *and* (ii) *indicate that any node with an empty buffer at the beginning of a slot can receive from all the active children.* □

Then, we obtain the following corollary directly from Lemma 1.

Colorally 1 *By applying MSF Scheduling, any node with an empty buffer can be ready for transmission after a slot. In other words, any node can potentially transmit once every two slots.*

Proof *Through MSF scheduling, when a node completes a block transfer at a certain slot, its buffer becomes empty. From Lemma* 1*, it can receive from its child node and is ready to send.* □

Note that Lemma 1 and Corollary 1 are derived from the fact that a node with an empty buffer immediately can receive from its children, and the actual order of the subtrees does not matter.

Next, we introduce the recurrence formula for the remaining blocks in each subtree at each 2 k slots. We define a *cycle* as a couple of slots here. Let $\mathcal{R}^k = \{r_j^k | r_1^k \geq r_2^k \geq \cdots \geq r_{S_0}^k\}$ be a set of the number of remaining blocks in S_0 subtrees of the sink and sorted in nonincreasing order.

Lemma 2 *Through MSF scheduling, for* $\mathcal{R}^k = \{r_1^k, r_2^k, r_3^k, \ldots, r_{S_0}^k\}$ *at an arbitrary cycle k,* $\mathcal{R}^{k+1}$ *is described by*

$$(i) r_1^k \geq r_2^k > 0,$$
$$\mathcal{R}^{k+1} = sort\{r_1^k - 1, r_2^k - 1, r_3^k, \ldots, r_{S_0}^k\} \quad (3)$$
$$(ii) r_1^k > r_2^k = r_3^k = \cdots = r_{S_0}^k = 0,$$
$$\mathcal{R}^{k+1} = \{r_1^k - 1, 0, \ldots, 0\} \quad (4)$$

where $sort$ *represents a descending sort.*

Proof *Equation* (3) *gives the number of remaining blocks in each subtree after the sink receives from the root of the largest subtree and then receives the root of the second largest subtree during cycle k. Equation* (4) *gives the number of remaining blocks in each subtree after the sink receives from the root of the largest subtree at the first slot of cycle* k *and is idle at the second slot.*

Now, we start to show that equation (3) *holds for any cycle by induction. Suppose equation* (3) *holds for cycle k. Then,* $\mathcal{R}^{k+1}$ *consists of* $\{r_1^k - 1, r_2^k - 1, r_3^k, \ldots, r_{S_0}^k\}$*. Since* $r_1^k \geq r_2^k$*, at the first slot of cycle* $k+1$*, i.e., slot* $2k+1$*, the sink should be able to receive from the child* $s_{0,i}$ *where* $r_i^{k+1} = \max\{r_1^k - 1, r_3^k, \ldots, r_{S_0}^k\}$*. Corollary* 1 *ensures that the sink can receive from an arbitrary node of its children at slot* $2k+1$*. Similarly, at the second slot of cycle* $k+1$*, i.e., slot* $2k+2$*, the sink can receive from all the nodes except for those that transmitted at the previous slot. In addition, at the beginning of cycle* 1*, the sink can receive from all the children trivially. Hence, equation* (3) *holds for arbitrary cycle k by induction.*

Equation (4) *is proved in a similar way.* □

V. Implementation and Evaluation

We implemented a prototype of MSFCP using the IRIS Mote platform, which is based on 802.15.4-compliant RF230 radio. We used the TinyOS software platform. We set the packet length at 117 bytes including 100 bytes of sensed data corresponding to 16 samples of a 3-axis accelerometer. We achieved a throughput of 184 kbps without CCA in these settings.

We evaluated the impact of block size and topologies on the throughput, and we present our findings here.

First, we show the relationship between block size (number of packets contained in one block) and link throughput in Fig. 6. For larger block sizes, overhead is reduced, but more memory is required. Although typical sensor nodes have memory limitations (e.q. 8 k bytes on IRIS Mote and 10 k bytes on Tmote Sky), 2–3 k bytes of RAM is adequate for efficient block transfer, as shown in Fig. 6. Thus, we set the number of packets in a block at 20.

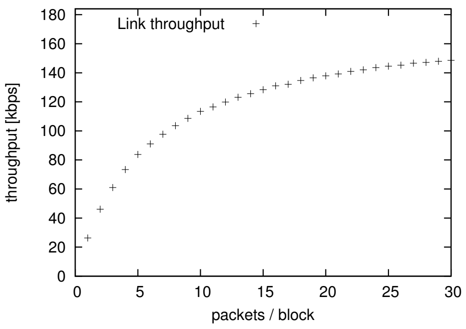


Fig. 6. Efficiency of block transfer

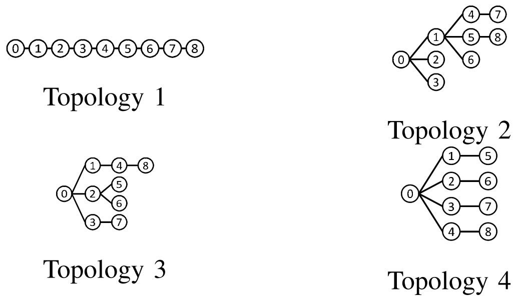


Fig. 7. 9-node routing trees

We conducted a test on four 9-node routing trees on the testbed. The trees had the topologies shown in Fig. 7. In these settings, $N = 8$ and $n_{max} = 8,\ 6,\ 3,\ 2$, respectively. The ideal throughput is the overall throughput without any protocol overhead, which is calculated based on the link throughput of a block size of 20 packets. Fig. 8 shows that 80-95 percent of the ideal throughput wasis achieved. This throughput degradation is caused by channel switching and packet losses.

VI. Conclusion and Future Work

We proposed the Maximum-Subtree-First Collection Protocol for high throughput bulk transfer in wireless sensor networks. When combined with hop-count-based channel assignment and hop-by-hop block transfer, Maximum-Subtree-First scheduling can mitigate both intra-path and inter-path interference. We presented the lower bound on collection time and showed the optimality of MSF scheduling in ideal settings. We implemented a prototype of MSFCP on the IRIS Mote platform and evaluated the efficiency of block transfer and overall throughput of MSFCP.

In the future, we will focus on the construction of a balanced topology. Theorem 1 indicates that the routing tree should be constructed in a balanced manner so that $n_{max} \leq (N+1)/2$. Constructing a tree where $n_{max} \leq (N+1)/2$ on an arbitrary graph G is a kind of *Capacitated Minimal Spanning Tree Problem*, and is proven to be NP-complete[14]. For MSFCP, each node needs information on its subtrees; thus, a balanced routing tree construction integrated with a function to gather subtree information is required.

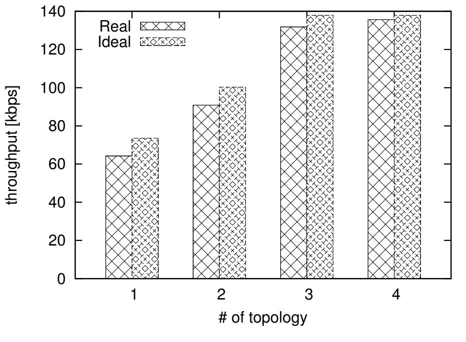


Fig. 8. Impact of topology on throughput

References

[1] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proc. 6th International Conf. on Information Processing in Sensor Networks*, 2007, pp. 254–263.

[2] T. Nagayama, P. Moinzadeh, K. Mechitov, M. Ushita, N. Makihata, M. Leiri, G. Agha, B. Spencer Jr, Y. Fujino, and J. Seo, "Reliable multi-hop communication for structural health monitoring," *Smart Structures and Systems*, vol. 6, no. 5-6, pp. 481–504, 2010.

[3] M. Suzuki, S. Saruwatari, N. Kurata, and H. Morikawa, "A high-density earthquake monitoring system using wireless sensor networks," in *Proc. 5th International Conf. on Embedded Networked Sensor Systems*, 2007, pp. 373–374.

[4] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proc. 7th Symp. on Operating systems design and implementation*, 2006, pp. 381–396.

[5] A. K. Vyas and F. A. Tobagi, "Impact of interference on the throughput of a multihop path in a wireless network," in *Proc. 3rd International Conf. on Broadband Communications, Networks, and Systems*, 2006, pp. 1–10.

[6] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: A reliable bulk transport protocol for multihop wireless networks," in *Proc. 5th International Conf. on Embedded Networked Sensor Systems*, 2007, pp. 351–365.

[7] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale, "Pip: A connection-oriented, multi-hop, multi-channel tdma-based mac for high throughput bulk transfer," in *Proc. 8th International Conf. on Embedded Networked Sensor Systems*, 2010, pp. 15–28.

[8] Ö. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Trans. on Mobile Computing*, vol. 11, no. 1, pp. 86–99, 2012.

[9] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. 7th International Conf. on Embedded Networked Sensor Systems*, 2009, pp. 1–14.

[10] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in *Proc. 9th International Conf. on Information Processing in Sensor Networks*, 2010, pp. 279–290.

[11] D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[12] M. H. Alizai, O. Landsiedel, J. A. B. Link, S. Götz, and K. Wehrle, "Bursty traffic over bursty links," in *Proc. 7th International Conf. on Embedded Networked Sensor Systems*, 2009, pp. 71–84.

[13] Q. Yu, J. Chen, Y. Fan, X. Shen, and Y. Sun, "Multi-channel assignment in wireless sensor networks: A game theoretic approach," in *Proc. 29th International Conf. on Computer Communications*, 2010, pp. 1127–1135.

[14] C. H. Papadimitriou, "The complexity of the capacitated tree problem," *Networks*, vol. 8, no. 3, pp. 217–230, 1978.