Target-Aware, Transmission Power-Adaptive, and Collision-Free Data Dissemination in Wireless Sensor Networks

Xiaorui Zhu, Xianping Tao, *Member, IEEE*, Tao Gu, *Senior Member, IEEE*, and Jian Lu

***Abstract*—Software update in wireless sensor networks requires the ability of disseminating bulk data to specified sensors of a network with low latency in an energy efficient manner. This paper proposes a target-aware, transmission power-adaptive, and collision-free data dissemination protocol to fulfill these requirements. This protocol disseminates data to sensors of a network by first constructing a connected dominating set (CDS) in the network. We propose a target-aware CDS construction to exclude many unnecessary non-target sensors from the data dissemination process. By allowing some dominators of the CDS to increase their transmission power to disseminate data to more dominatees, the protocol efficiently reduces the total energy consumption. In addition, we propose a collision-based channel assignment strategy to eliminate communication collisions among dominators so as to reduce latency. We have implemented the protocol and evaluated it through simulations and a real application scenario. Our experimental results show that the proposed protocol at most reduces non-target sensors by 75.3%, total energy consumption by 57.1%, and latency by 39.8% compared to existing dissemination protocols.**

***Index Terms*—Wireless sensor network, dissemination protocol, adaptive transmission power, multi-channel.**

I. Introduction

In recent years, wireless sensor networks (WSNs) have been successfully deployed in a variety of applications including volcano monitoring [1], wild animals tracking [2], tunnel lighting control [3], logistics management [4], clinical monitoring [5], etc. On one hand, the software running on WSNs often needs to be updated due to changes in application requirements, adoptions of new algorithms, and bug-fixing [6]. On the other hand, sensors in these applications are often hard-to-reach. Therefore, it is necessary to disseminate software updates in WSNs over-the-air via a bulk data dissemination protocol.

Despite its importance, designing such a dissemination protocol bears the following challenges. The first challenge rises from the need of *target-specified dissemination*: in many application scenarios, data only needs to be disseminated to part of the sensors in a network. For example, since sensors tend to have different functions in recently emerging applications [1], [4], [7], [8], only part of the sensors need the patch to update their software at a time. Additionally, this target-specified dissemination is also needed in WSAN applications [3], [9] where messages could be sent to a selected group of actuators. The second challenge is *energy efficiency*: sensors in WSN applications are powered by batteries, excessive energy consumption of sensors for disseminating data will shorten network lifetime. The third challenge lies in *latency*: data dissemination needs to stop applications completely to make sensors dedicate their resources for disseminating data. Data dissemination with high latency will reduce the availability of WSN applications, while high availability is expected by many WSN applications [1], [3], [5].

There are some recent efforts to address these three challenges. 1) For target-specified data dissemination, MDeluge [10] uses the shortest path algorithm to construct a dissemination tree which contains all target sensors (i.e., sensors which need the disseminated data) and part of non-target sensors (i.e., sensors which do not need to the disseminated data). However, the number of involved non-target sensors can be further reduced by considering the distribution of target sensors when constructing the dissemination tree. 2) To reduce energy consumption, existing protocols [11], [12] typically limit the number of source sensors (i.e., sensors that are required to disseminate data to other sensors) to diminish the energy consumed by transmitting redundant packets. In addition, as non-source sensors can turn off their radios for much longer time than source sensors during the data dissemination, less source sensors can further reduce the total energy consumption. However, these protocols consider no information about battery power of sensors. In most WSN applications, sensors will have different battery power after running for a period of time. If we increase the transmission power (TX power for short) of some source sensors with more battery power to let them cover larger area, the number of source sensors can be further reduced.[1] Besides receiving and transmitting packets, source sensors also

Manuscript received February 10, 2015; revised May 29, 2015; accepted July 16, 2015. Date of publication July 30, 2015; date of current version December 8, 2015. This work was supported by the National Natural Science Foundation of China (Nos. 61373011, 91318301, and 61321491). The associate editor coordinating the review of this paper and approving it for publication was Q. Li.

X. Zhu, X. Tao, and J. Lu are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China (e-mail: zxr@smail.nju.edu.cn; txp@nju.edu.cn; lj@nju.edu.cn).

T. Gu is with the School of Computer Science and Information Technology, RMIT University, Melbourne, Vic. 3001, Australia (e-mail: tao.gu@rmit.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2015.2462827

[1]To cover the same area, energy consumption of one sensor with high TX power is theoretically equal to that of multiple sensors with low TX power. The proof is given in Appendix A of the supplementary file.

need to turn on their radios to collect the packet loss information. Therefore, less source sensors can reduce the significant energy consumed for collecting the packet loss information and further reduce the total energy consumption. 3) To reduce latency, McTorrent [13] reduces the time for retransmitting lost packets by introducing the multi-channel communication which can efficiently reduce communication collisions. Before sending data packets, a sensor in McTorrent will choose an idle channel according to the channel occupation information contained in packets overheard from its neighbors. However, this gossip-based channel assignment cannot eliminate communication collisions since two sensors may choose the same idle channel to disseminate data. If we have the complete information of communication collisions among source sensors, we can eliminate all these collisions by assigning different channels to any pair of source sensors if there exit collision.

Although these ideas look promising, there are still some challenges. 1) It is difficult to directly describe and use the distribution information of target sensors because no sensor knows the locations of target sensors. 2) Increasing TX power of sensors overly may shorten the lifetime of WSN applications. In addition, it may potentially incur more communication collisions. 3) It is difficult to collect complete communication collisions among source sensors without incurring much overhead. Meanwhile, limited available channels make channel assignment be a NP-Complete problem. In addition, available channels may not be sufficient for eliminating all communication collisions.

To address the aforementioned three challenges, in this paper we propose a target-aware, TX power-adaptive, and collision-free dissemination protocol (T^2C), aimed at improving the efficiency of bulk data dissemination in multi-hop, multi-channel sensor networks. 1) Before disseminating data, T^2C first constructs a dissemination tree (a CDS [14] more specifically) in the target network. The constructing process keeps counting target sensors of each path and allows the paths with more target sensors to grow first. This strategy can efficiently use the distribution information of target sensors to converge them into fewer paths so that non-target sensors in the tree can be further reduced. 2) To increase TX power, we use the battery power of source sensors which are one-hop neighbors of the sink as the bases. If a source sensor has more battery power than one of the bases, the extra battery power is used to increase its TX power during data dissemination. This strategy can prevent shortening the lifetime of a WSN as the sensor with the least battery power will not increase its TX power.[2] In addition, multi-channel communication is introduced to resolve the problem of communication collisions. 3) Sensors generate communication collision information based on received messages which are used for constructing the dissemination tree. No other special message is required. With collision information, we propose a heuristic algorithm to resolve the channel assignment problem. If channels are not sufficient, the target network will be partitioned into several sub-networks to guarantee that channels are sufficient for any of these sub-networks.

[2]Lifetime of a WSN is generally defined by the lifetime of the sensor which first exhausts its battery power.

In summary, this paper makes the following contributions.

- We propose a target-aware CDS construction method to reduce the number of non-target sensors in target-specified data dissemination. Our experimental results show that with this method in randomly generated networks, our protocol reduces the number of non-target sensors by at least 31.7% compared to MDeluge [10].
- We propose a TX power increasing strategy to efficiently utilize excessive battery power of sensors. Our experimental results show that with this strategy, our protocol reduces total energy consumption by 15.5%–58.3% and 12.8%–57.1% compared to CORD [12] and McTorrent [13], respectively.
- We propose a collision-based channel assignment strategy to eliminate communication collisions among source sensors. Our experimental results show that with this strategy, T^2C reduces the latency by at most 39.8% compared to McTorrent [13], an advanced multi-channel data dissemination protocol.

The rest of paper is organized as follows. Section II reviews the related work. Section III gives an overview of T^2C. Section IV presents the detailed design of the protocol. Section V evaluates the performance. Section VI concludes the paper and discusses our future work.

II. Related Work

We discuss related work from the following three perspectives: target-specified dissemination support, source sensors limit, and data propagation method.

A. Target-Specified Dissemination Support

Most data dissemination protocols (e.g., Deluge [15], CORD [12], McTorrent [13]) aim at disseminating bulk data (code in most cases) to all sensors of a network. However, sensors of recent WSN applications tend to have different functions and play different roles. Normally, only part of sensors needs to update their software. In this case, it is not efficient if all sensors are involved in the data dissemination process. To address this problem, MDeluge [10] uses the shortest path method to construct a dissemination tree before disseminating data. The tree contains all target sensors and some necessary non-target sensors. T^2C also supports target-specified data dissemination. Different from MDeluge [10], T^2C takes the distribution of target sensors into consideration when constructing a dissemination tree. This target-aware method can further reduce non-target sensors in the dissemination tree by converging target sensors into fewer paths. Illustrative examples are shown in Fig. 1. In the shortest path method, the sensor n_8 will choose $n5$ as its parent to build a shortest path to n_1. This choice makes the dissemination tree contain three non-target sensors. However, the target-aware method will make n_8 choose n_9 as its parent to converge target sensors into one path. As a result, the number of non-target sensors is reduced.

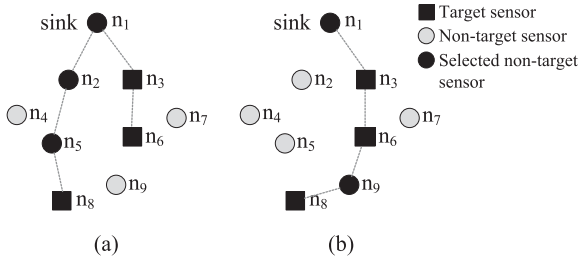

Fig. 1. Example of constructing a dissemination tree with different methods. (a) Shortest path method. (b) Target-aware method.

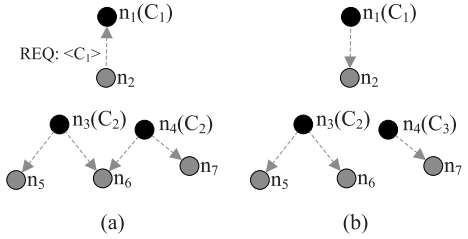

Fig. 2. Illustration of different channel assignment methods. (a) Gossip-based channel assignment. (b) Collision-based channel assignment.

B. Source Sensors Limit

Deluge [15] and MNP [11] adopt the three-phase handshaking of ADV-REQ-DATA to limit the number of source sensors. In these protocols, sensors are allowed to broadcast advertisements if they have received less than k same advertisements during a random waiting period. In addition, sensors which have broadcasted advertisements can become source sensors only if they receive at least one request. Although this mechanism efficiently limits the number of source sensors, it incurs many control messages. To address this problem, Sprinkler [16] and CORD [12] choose to fix source sensors before disseminating data. These protocols will first construct a CDS in the target network. Dominators in the CDS are taken as source sensors. T^2C is also a CDS-based data dissemination protocol. Different from Sprinkler and CORD, T^2C allows the dominators with more battery power to increase their TX power to cover more dominatees. As a result, the number of dominators can be further reduced.

C. Data Propagation Method

In most data dissemination protocols (e.g., Deluge [15], CORD [12], MDeluge [10]), data is propagated using only one channel. A practical model [17] is proposed to accurately analyze the performance of these protocols and optimize dissemination latency. However, single-channel data propagation essentially suffers from communication collisions and low propagating speed [13]. McTorrent [13] introduces multi-channel communication to address these problems. However, the gossip-based channel assignment of McTorrent may assign the same channel to source sensors which have communication collisions. For example, sensors n_3 and n_4 in Fig. 2(a) may both want to disseminate data. Since they only know that the channel C_1 has been occupied by n_1 according to the overheard REQ message, they may choose the same idle channel (e.g., C_2) to disseminate data so that n_6 becomes a hidden-terminal. In addition, this gossip-based strategy requires all sensors to turn on their radios during the whole data disseminating process. Different from McTorrent, channel assignment in T^2C is based on global communication collision information. Therefore, it can eliminate communication collisions among all source sensors as illustrated in Fig. 2(b). In this example, since we have known that each of the three sensors (n_1, n_3, and n_4) has communication collisions with the other two sensors, we can assign different channels to the three sensors to eliminate these communication collisions. Meanwhile, as channels of source sensors are assigned before disseminating data, data propagation in T^2C uses coordinated data propagation scheduling which allows non-source sensors (dominatees) to turn off their radios periodically.

The reliability of data propagation is important for dissemination protocols. Some practical methods have been proposed such as adaptive congestion control in ESRT [18]. However, since the reliability is not our main concern in this paper, we just adopt the hop-by-hop recovery method introduced by PFSQ [19] to achieve the reliability.

III. Overview

T^2C is a CDS-based bulk data dissemination protocol which targets multi-hop, multi-channel sensor networks with stationary sensors. It aims at reducing the number of non-target sensors in target-specified dissemination, and reducing the total energy consumption and the latency of the dissemination process. To achieve these targets, the protocol operates in three phases: CDS construction, channel assignment, and data dissemination.

In the first phase, T^2C takes a distinctive way to construct a CDS in the target network. Specifically, it first requires every sensor to choose the neighbor which is on the path with the most target sensors to be its parent when joining a CDS. This strategy converges target sensors into fewer paths to reduce the number of non-target sensors in the CDS. Then, it increases TX power of newly selected dominators based on their battery power. As a result, T^2C reduces the number of dominators which strongly affects the total energy consumption. In addition, it makes sensors generate communication collision information based on pre-defined rules (see Section IV-A4) for details) and messages used for constructing the CDS. The information will be aggregated to the server to help to assign channels.

In the second phase, T^2C assigns channels to dominators in the CDS based on the communication collision information. Specifically, it first computes channels of dominators to guarantee that any pair of dominators will have different channels if there exist communication collisions. It then disseminates the channel assignment results to dominators before disseminating data. This channel assignment strategy helps to reduce the latency since it can eliminate all communication collisions among dominators during the data dissemination process. Additionally, it incurs much less overhead compared with the gossip-based strategy [13] which requires source sensors to negotiate channels during the data disseminating process.

The third phase involves two steps. In the first step, data is disseminated through dominators in the CDS. Dominatees in this step will passively receive packets as many as possible. T^{2}C introduces a coordinated scheduling in this step to disseminate data. The scheduling enables dominators to work fully to speed up the data dissemination process, and adopts the DATA-NACK mode to reduce the number of control messages. Whenever a dominator has received the complete data, it starts the second step in which its dominatees will ask for packets lost in the first step.

A. Design Choice

1) CDS-Based Approach: In the design of data dissemination protocols, source sensors can be chosen dynamically during the disseminating process [15] or statically before disseminating data [12]. Although this dynamic approach is more robust to sensor failure and link fading, we choose the static CDS-based approach for the following three reasons. 1) In many WSN applications, neighboring sensors need to process data collaboratively before sending it to the base station. Protocols with the dynamic approach can still finish data dissemination and restart the application even if a sensor failure occurs during the disseminating process. It is dangerous because the sensor failure may incur unpredictable errors in the collaborative data processing. In contrast, protocols with the static approach can help to find sensor failure before restarting the application. 2) Experimental results show that link quality will not change dramatically in a short period of time [12]. Therefore, link quality fading will not seriously affect protocols with the static approach. 3) To prevent overly increasing TX power, source sensors need to be able to estimate their energy consumption in the data disseminating process. The static approach facilitates estimation as the behaviors of source sensors are predictable.

2) Static Channel Assignment: There are basically two approaches to assign channels. The dynamic approach requires every source sensor to choose a channel which has not been used by its two-hop neighbors whenever it wants to disseminate data. However, actively negotiating an available channel with two-hop neighbors is inefficient,[3] while the gossip-based implementation [13] may assign the same channel to source sensors which have communication collisions. In addition, the dynamic approach essentially needs all sensors to turn on their radios continuously during the data disseminating process.

In contrast, the static approach assigns channels to source sensors based on global information of communication collisions. Therefore, it can eliminate all these collisions with much less efforts. In addition, as channels are assigned before disseminating data, the static approach facilitates to adopt a coordinated data propagation scheduling which allows sensors to turn off their radio periodically. However, the key issue of the static approach is the overhead of collecting communication collision information. As T^{2}C can collect the information efficiently during the CDS constructing process, we choose the static channel assignment approach.

[3]Suppose two-hop neighbors of a source sensor have occupied n channels, in the worst case the source sensor needs $n + 1$ rounds of negotiation to get an available channel.

IV. Protocol Description

In this section, we describe the key techniques we propose in T^{2}C in details.

A. CDS Construction

The basic CDS construction algorithm is similar to which of CORD [12]. The main difference is that our algorithm involves three important tasks—converging target sensors, increasing TX power of dominators, and collecting communication collisions.

While converging target sensors into fewer paths, a challenge is to limit the depth of the CDS (i.e., the maximum length of paths from leaf dominators to the sink) as it affects the latency of data dissemination. This limitation is achieved by setting a threshold of the depth of a sensor (i.e., the number of hops from the sensor to the sink) when it joins the CDS. The challenge of increasing TX power is that there is no facility to measure battery power of dominators directly. Instead, we propose an approximation method to estimate battery power of sensors by analyzing their energy consumption in different states. One main challenge of collecting communication collisions is to prevent the collection from incurring much overhead. Therefore, sensors In T^{2}C generate communication collision information based on messages used for constructing the CDS. No other special message is required.

1) Basic Construction Algorithm: Before constructing a CDS in a network, a READY message is first disseminated in the network. Sensors which receive the message will stop the running application and dedicate their resources for the coming data dissemination process.

The basic CDS construction algorithm in T^{2}C is a degree-aware algorithm which selects dominators sequentially at each level. Basic steps of the algorithm are illustrated in Fig. 3. a) In the beginning, the sink marks itself as a *dominator* and broadcasts a DOMINATOR message. b) All *unsettled* sensors that receive the message will mark themselves as *candidates*. Each *candidate* will then broadcast a CANDIDATE message and c) compute its degree based on received COUNT messages from its *unsettled* neighbors. d) After that, *candidates* will broadcast COMPETE messages which contain their degrees. e) Each *unsettled* sensor will response with a SUPPORT message to the *candidate* with the maximum degree (the sensor ID is used to break the tie). f) *Candidates* which have received at least one SUPPORT message will mark them *dominators*, while others mark them *dominatees*. g) New *dominators* will then broadcast their DOMINATOR messages.

These steps will end if a dominator receives no DOMINATOR message which should be broadcasted by its child dominators during the waiting period of the dominator. This dominator will send a FINISH message to its parent dominator. If a dominator has received FINISH messages from all its child dominators, it will send its FINISH message to its

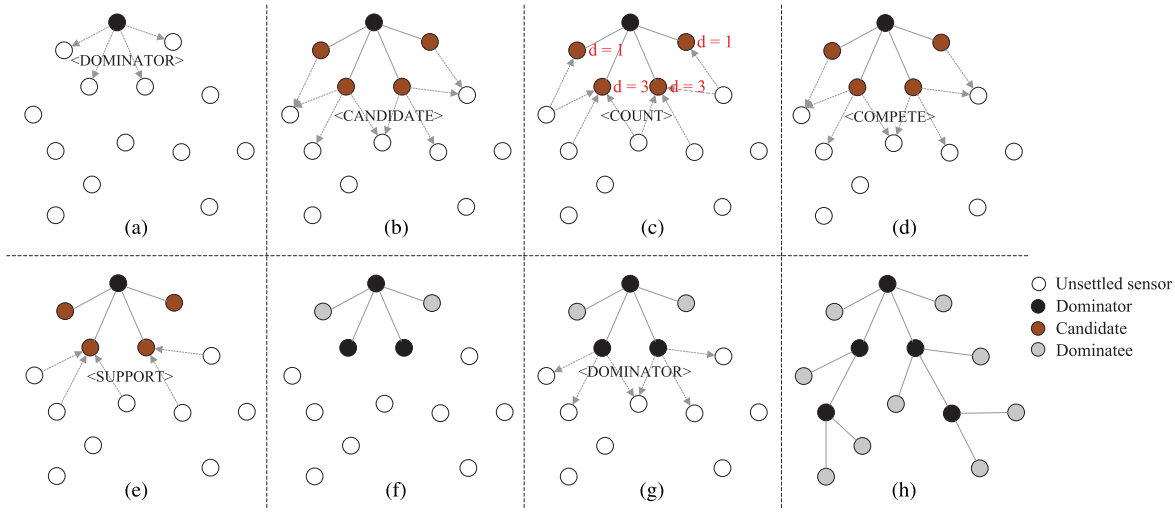

Fig. 3. Basic steps of constructing a CDS in T^{2}C.

parent dominator. When the sink dominator sends its FINISH message to the server, the CDS constructing process is done. Fig. 3(h) shows the complete CDS.

2) Target Sensor Converging: We improve the basic construction algorithm to converge target sensors. We first use a bitmap to indicate roles of sensors. The value of the i_{th} bit (1 or 0) of the bitmap indicates the role of the sensor (target or non-target) with the ID of i. We then put the bitmap into the READY message and broadcast the message to the network. Each dominator which has received the message will broadcast it after a certain delay (determined by the dominator's unique ID) to avoid communication collisions.

We then add a field which records the number of target sensors (N_{TS}) of a path to the DOMINATOR message. When an unsettled sensor receives a DOMINATOR message from its dominator, it get the N_{TS} of the path which the dominator belongs to. After the unsettled sensor becomes a candidate, it computes the new N_{TS} value based on its role and put the value into its COMPETE message. When an unsettled sensor receives COMPETE messages from candidates, it will support the candidate which has broadcasted the COMPETE message with the largest N_{TS} value. Only if the N_{TS} values are the same could be the degree values used to make a decision. As the SUPPORT message contains the role of an unsettled sensor, a dominator will easily know whether it has a child (dominator or dominatee) of target sensor. This information will be included in the dominator's FINISH message.

To further converge the target sensors, T^{2}C allows a candidate c_1 to re-choose a neighboring candidate c_n as its new dominator if the following conditions are satisfied. 1) c_1 and c_n have different dominators. 2) c_n has the largest N_{TS} value among all neighboring candidates of c_1. 3) Given N_{TS}^1 of c_1 and N_{TS}^n of c_n, N_{TS}^n is larger than $N_{TS}^1 + 1$. In this case, c_1 will send a SUPPORT message to c_n and make c_n as its new dominator. Because the CDS is constructed in a breath-first manner, and the dominator re-choosing can only select sensors with the same depth, the depth of CDS can be controlled so as not to increase the latency seriously.

3) Transmission Power Increment: To increase TX power, we first choose dominators which are one-hop neighbors of the sink as *benchmark* dominators. These benchmark dominators will keep their TX power the same as which used in the application. We choose these sensors as benchmark dominators for two reasons. First, these sensors usually have least battery power. Taking battery power of these sensors as bases will not impact the effectiveness of TX power increment heavily. Second, finding dominator with the least battery power will incur much overhead. Benchmark dominators will broadcast DOMINATOR messages containing arguments for recipient sensors to calculate the current battery power of their benchmark dominators.

The current battery power of a sensor can be calculated with the following formula.

$$P_{cnt} = P_{ini} - \sum I_{state} \cdot t_{state} \tag{1}$$

where P_{cnt} and P_{ini} are the current and the initial battery power, respectively. I_{state} and t_{state} represent the current consumption and the time of sensors in different states, respectively. According to the data sheets of some popular sensors, we find that the dominating energy consuming states are *RX* and *TX* states of radio, and *write* and *read* states of EEPROM. Therefore, Formula (1) can be transformed as follows.

$$P_{cnt} = P_{ini} - \left(\sum I'_{state} \cdot t_{state}^{byte} \cdot n_{state} + I_{RX} \cdot \left(t_{on} - t_{TX}^{byte} \cdot n_{TX} \right) \right) \tag{2}$$

where I'_{state} represents the current consumption in three states of Tx, read, and write, t_{state}^{byte} represents the time for handling a byte of data in each of the three states, and n_{state} represents the total number of bytes handled in each of the three states before data dissemination. For the time of radio in RX state, we use the working time of radio (t_{on}) subtracts the time of radio in TX state to calculate it. Sensors of WSN applications which use the T^{2}C protocol are required to record n_{state} for the three states and t_{on} (t_{on} of a sensor can be calculated according to the duty-cycle of the radio and the working time of the sensor).

These four types of data of benchmark dominators are propagated through DOMINATOR messages. When unsettled sensors receive DOMINATOR messages, they will first increase their TX power based on arguments contained in the messages before broadcasting CANDIDATE messages. To prevent overly increasing TX power, only extra battery power of candidates can be used to increase their TX power during data dissemination as shown in the following formula.

$$P_{cnt}^{candidate} - P_{cnt}^{benchmark} = E_{dis}^{candidate} - E_{dis}^{benchmark} \quad (3)$$

where P_{cnt} represents the battery power of a sensor before data dissemination, and E_{dis} represents the energy consumed by a sensor during data dissemination.

In T^{2}C, the behaviors of dominators during data disseminating stay almost the same. That means the difference of energy consumption between two dominators comes from the energy consumed by transmitting packets in different TX power. Therefore, we use the following formula to estimate the current consumption of the new TX power based on Formulae (2) and (3).

$$I_{TX}^{candidate} <= \frac{P_{cnt}^{candidate} - P_{cnt}^{benchmark}}{t_{TX}^{byte} \cdot (1+\alpha) \cdot n_{data}} + I_{TX}^{benchmark} \quad (4)$$

where n_{data} is the number of data packets, and α is a coefficient representing the estimation of the number of control packets and retransmitted data packets. Based on the experimental results, we suggest that the value of α ranges from 0.01 to 0.05.

If $P_{cnt}^{candidate} - P_{cnt}^{benchmark} <= 0$, the candidate keeps the current TX power unchanged. Otherwise, $I_{Tx}^{candidate}$ gives the upper bound of current consumption of the new TX power. The candidate can choose the TX power of which the current consumption falls between $[I_{Tx}^{cnt}, I_{Tx}^{candidate})$. This range provides flexibility of the TX power increasing. Note that the TX power of which the current consumption equals to $I_{Tx}^{candidate}$ cannot be selected because we want to prevent the exorbitant increasing of TX power caused by estimation errors in formulae (2) and (4).

4) Communication Collision Collection: T^{2}C adopts a centralized approach to assign channels based on global communication collision information collected in the CDS constructing process. According to Fig. 7, for any two dominators d_1 and d_2, if they both have odd (or even) depths (i.e., $|depth_{d_1} - depth_{d_2}| = 2n, n \in N$), they will be in the same state of parent or child. Otherwise (i.e., $|depth_{d_1} - depth_{d_2}| = 2n+1, n \in N$), they will be in different states. That means if depths of two dominators (d_1 and d_2) satisfy the condition of $|depth_{d_1} - depth_{d_2}| = 2n (n \in N)$, they will send packets (in the parent state) simultaneously and may interfere with each other. In addition, if d_1 has a child d_c (obviously $|depth_{d_2} - depth_{d_c}| = 2n+1, n \in N$), d_c will be a hidden-terminal if it can receive messages from both d_1 and d_2 in its child state. Based on this feature, we use the following three rules to collect communication collision information among dominators (the proof of the correctness of these rules is given in Appendix B of the supplementary file).

Rule 1: Suppose a dominator d_1 receives a DOMINATOR message from a new dominator d_2, if the depths of these two dominators satisfy the condition of

1) $|depth_{d_2} - depth_{d_1}| = 2n (n \in N)$, d_1 will mark that d_2 interferes with itself;
2) $|depth_{d_2} - depth_{d_1}| = 2n+1 (n \in N)$, and d_1 is not the sink dominator, d_1 will send a CONFLICT message to inform its parent d_p that d_2 and d_p have a hidden terminal of d_1.

Rule 2: Suppose a dominatee *de*'s dominator is d_1, if *de* receives a DOMINATOR message from a new dominator d_2, and the depths of *de* and d_2 satisfy the condition of $|depth_{de} - depth_{d_2}| = 2n+1 (n \in N)$, *de* will send a CONFLICT message to inform d_1 that d_2 and d_1 have a hidden terminal of *de*.

Rule 3: Suppose an unsettled node u has sent a SUPPORT message to a candidate c, if u receives a DOMINATOR message from a new dominator d_1, and the depths of u and d_1 satisfy the condition of $|depth_u - depth_{d_1}| = 2n+1 (n \in N)$, u will send a CONFLICT message to inform c that d_1 and c have a hidden terminal of u.

Information of communication collisions will be aggregated to the server along with FINISH messages. If a dominator which has sent its FINISH message receives new CONFLICT messages, it will send them to the server immediately.

B. Channel Assignment

In this phase, the server will prune the CDS to remove all unnecessary non-target dominators, and assign channels to dominators of the pruned CDS based on the communication collisions among them. There are two challenges in this phase. The first challenge rises from the processing of the communication collisions. To address this challenge, we propose a *collision graph* which uses the communication collisions to construct a well-known *graph-coloring* problem. Since the problem is proven to be NP-Complete [20], a heuristic algorithm is proposed to solve this problem. The second challenge is that channels may not be sufficient for eliminating all communication collisions as available channels are limited. To address this challenge, we partition a network into sub-networks and guarantee that channels are sufficient for every sub-network.

After the channel assignment is done on the server side, the results will be disseminated to the sensors in the target network. The key task here is to reduce control packets needed to disseminate the results. Therefore, we design four special types of packet to encapsulate the results.

1) CDS Pruning: To prune the CDS, we first construct a graph G_{CDS} based on the communication collision information collected in the first phase. The G_{CDS} contains dominators with two different colors. Dominators which are target sensors or non-target sensors with at least one child of the target sensor are colored black, while other dominators are colored white. The pruning process is to find the least white dominators which can make black dominators connected. To find these white dominators, we propose a dominator coloring algorithm as shown in Alg. 1.

Algorithm 1: Dominator_Coloring

Input: G_{CDS}
Output: G_{CDS}
1 $enqueue(root, q)$;
2 **while** *q is not empty* **do**
3 $v = dequeue(q)$;
4 **if** *v has no child* **then**
5 $enqueue(v, ql)$;
6 **else**
7 **for** *each child* c_i *of* v **do**
8 $enqueue(c_i, q)$;
9 **while** *ql is not empty* **do**
10 $v = dequeue(ql)$;
11 **while** v != $root$ **do**
12 **if** $color_v == black$ **then**
13 **break**;
14 $v = parent_v$;
15 **while** v != $root$ **do**
16 $color_v = black$;
17 $v = parent_v$;

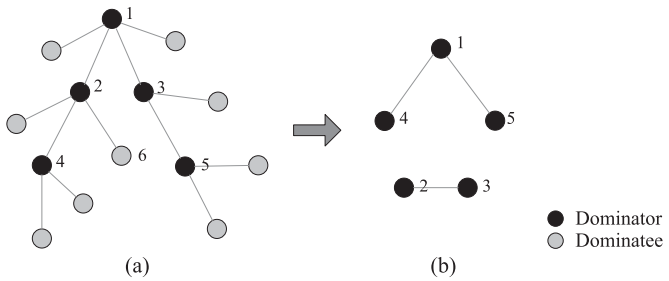

Fig. 4. Example of a collision graph derived from a CDS. (a) CDS. (b) Collosion graph.

The algorithm first gets all leaf dominators in G_{CDS} (lines 1–8). After that, it travels all paths from each leaf dominator to the root (the sink). When traveling in each path, it looks for the first black dominator (lines 10–14) and colors all white dominators between the black dominator and the root of the path black (lines 15–17). When the dominator coloring process is done, we can prune the CDS by removing all white dominators in G_{CDS}.

2) Collision Graph: Given a CDS G_{CDS}, the server will first derive a collision graph from it according to communication collision information.

Definition 1: A collision graph G_{col} is an undirected graph which contains *no* isolated vertex. Any adjacent vertexes in the graph indicate dominators which have communication collisions.

Fig. 4 shows a collision graph derived from a CDS. In the graph, dominators 2 and 3 interfere with each other and have a hidden-terminal of dominatee 6. Dominators 1 and 4 have a hidden terminal of dominator 2, while dominators 1 and 5 have a hidden terminal of dominator 3.

3) Heuristic Channel Assignment: Given k available channels and a collision graph G_{col}, the channel assignment problem is essentially equivalent to the *graph coloring* problem (here a "color" refers to a channel). We propose a heuristic algorithm as shown in Alg. 2.

Algorithm 2: Heuristic_Channel_Assignment

Input: G_{col}, k
Output: G_{fail}
1 $G'_{col} = G_{col}$; $G_{fail} = \emptyset$;
2 **while** $\exists v \in G_{col}.degree_v < k$ **do**
3 $G_{col} = G_{col} - v$;
4 $put(v, S)$;
5 **if** $|G_{col}| > 0$ **then**
6 $G_{temp} = G_{col}$;
7 **for** *each node* $v \in G_{col}$ **do**
8 **if** $min(degree_v) == true$ **then**
9 $G_{col} = G_{col} - v$;
10 $put(v, S)$;
11 **while** S *is not empty* **do**
12 $v = top(S)$; $C_{occ} = \emptyset$;
13 Add v to G_{col} according to G'_{col};
14 **for** *each neighbor* v_i *of* v *in* G_{col} **do**
15 $C_{occ} = C_{occ} \cup \{c_{v_i}\}$;
16 **if** $|C_{occ}| < k$ **then**
17 $c_v = c \in (C_{ava} - C_{occ})$;
18 **else**
19 $G_{fail} = G_{temp}$;
20 **break**;
21 $pop(S)$;
22 **return** G_{fail};

Given the number of available channels k, for each node v in the collision graph G_{col}, if the degree of v is less than k, we put v into the stack S and delete v from G_{col} (lines 2–4). The idea is that if there is a channel assignment solution for $G_{col} - v$, there must be a solution for G_{col}. This is because v's neighbors in G_{col} use at most $(k-1)$ channels, v always has an available channel.

After that, if there still exist nodes in G_{col}, degrees of these nodes (called *bottleneck nodes*) are all more than $(k-1)$. In this case, k channels *may* not be enough for G_{col}. The heuristic strategy here is to keep on putting the node with the minimum degree into stack S and removing it from G_{col} (lines 7–10). This strategy guarantees that when trying to assign channels for these bottleneck nodes, nodes with large degrees will be assigned first. The reason is that nodes with small degrees have higher possibility to get available channels as they have less neighbors being already assigned.

When all nodes in G_{col} are in stack S, we repeatedly choose the top node in S and put it back to G_{col} according to the original topology until S is empty (lines 11–13). The newly added node will be assigned a channel which is different from all its neighbors (lines 14–17). If we can put all nodes back into G_{col}, we have assigned channels for all dominators. Otherwise, the channel assignment fails and returns all bottleneck nodes in G_{fail} (lines 18–22).

4) Network Partition: If the channel assignment succeeds, we move on to the next step. However, channels may not be sufficient in some extreme cases. We propose a solution to partition the network into sub-networks. Each sub-network

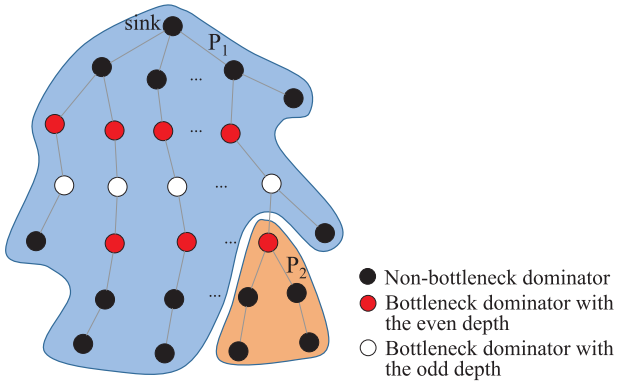

Fig. 5. Example of the CDS partition.

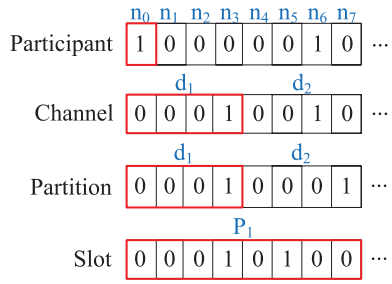

Fig. 6. Formats of configuration messages.

contains at most k bottleneck nodes with even depths and k bottleneck nodes with odd depths. As dominators with even depths and odd depths will not send data simultaneously, k channels are sufficient for any sub-network (the proof is given in Appendix C of the supplementary material).

Obviously, partitioning a network essentially equals to partitioning the CDS of the network. Fig. 5 shows an example of the CDS partition. Suppose a CDS has n ($n > k$) bottleneck dominators with even depths (red nodes in the figure), we need to partition the CDS into sub-CDSs so that each sub-CDS contains no more than k red nodes. The partitioning process first chooses the sink as the root of a sub-CDS. It then repeatedly processes all child nodes of each new added node of the sub-CDS. If a node is a bottleneck dominator with the even depth, we add it to the sub-CDS only if the sub-CDS has less than k bottleneck dominators with even depths. Similar approach is used to process bottleneck dominators with odd depths. If a dominator is not a bottleneck dominator, we add it to the sub-CDS directly. The sub-CDS is built when no node in the CDS can be added to it. After that, we build a new sub-CDS by choosing a unprocessed bottleneck dominator with the smallest depth as the root and repeat the building process.

When all nodes in the CDS have been processed, the CDS partitioning process is done. Note that except the last one, other sub-CDSs all contain k bottleneck dominators with even depths and (or) k bottleneck dominators with odd depths. This property helps to reduce the number of sub-networks. We will then reuse the channel assignment algorithm to process each of the sub-networks.

5) Configuration Dissemination: We design four types of message to encapsulate configuration of all dominators as shown in Fig. 6. In a PARTICIPANT message, the i_{th} bit is used to represent the participant information of dominator with the ID of i. Value 1 represents dominators which are involved in data dissemination. With the PARTICIPANT message, each participant dominator knows its sequence number among all participant dominators. The CHANNEL message includes channels of all participant dominators arranged with the sequence in the PARTICIPANT message. The decimal number expressed by the i_{th} four bits represents the channel of the i_{th} dominator.

If a network has been partitioned, we need to disseminate information of sub-networks to participant dominators. Similar to the CHANNEL message, the PARTITION message uses the i_{th} four bits to represent index of the sub-network to which the i_{th} participant dominator belongs. Except the first sub-network, dominators in other sub-networks need not to turn on their radios at the beginning of data dissemination. Therefore, the SLOT message uses the i_{th} eight bits to represent the number of slots required by sensors of the i_{th} sub-network to turn off their radios.

After configuration messages are prepared, the sink starts continuous slots and broadcasts these messages after a random delay in the first slot. If a dominator (a dominatee) receives configuration messages from its parent dominator (its dominator), it will start continuous slots, mark the current slot as *child* slot, and synchronize the slots to which of its parent dominator (its dominator) based on the random delay recorded in messages. The dominator will then broadcast configuration messages in the next slot after a random delay and mark that slot as *parent* slot, while the dominatee will turn off its radio in the next slot and mark that slot as *sleep* slot.

Dominators which have broadcasted configuration messages and dominatees which have received configuration messages from their own dominators will configure themselves. Nodes which are not in the CDS will quit the data dissemination and wait for the RESTART message.

C. Data Dissemination

We use coordinated data propagation scheduling to disseminate data. As communication collisions among dominators have been eliminated, data transmission among dominators can adopt the DATA-NACK mode to further reduce the number of control packets. Meanwhile, dominatees can turn off their radios periodically to save energy.

1) Two-Step Data Dissemination: T^2C takes a two-step data dissemination approach. In the first step, data is propagated among dominators according to coordinated scheduling as shown in Fig. 7. In this scheduling, dominators are in interleaved slots of *parent* and *child*. Dominators in *parent* slots will broadcast data in their own channels, while dominators in *child* slots will receive data in their parents' channels. Dominatees are in interleaved slots of *child* and *sleep*. Dominatees in *child* slots will receive data in their dominators' channels, while dominatees in *sleep* slots will turn off their radios to save energy.

Data transmission among dominators adopts the DATA-NACK mode. After having received data packets of a page in a child slot, if a dominator finds that it misses some packets, it will send a NACK message which contains a bitmap to inform

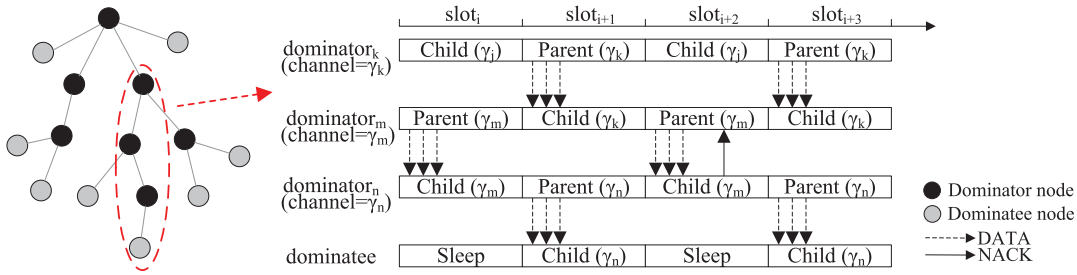

Fig. 7. Coordinated scheduling of multi-channel data propagation in T^{2}C.

its parent the lost packets after a random delay. The parent dominator will merge bitmaps contained in NACK messages and re-broadcast lost packets in the next parent slot. Unless a dominator has received the complete packets of a page, it cannot disseminate the data of the page.

If a dominator finds that all its child dominators have received the complete data, it will broadcast a RECOVERY message to inform its dominatees to start the second step. In each slot of this step, dominatees will first send a QUERY message to request lost packets after a random delay. The dominator will then merge all queries and broadcast required packets.

V. Evaluation

We first evaluate the performance of T^{2}C in various scenarios through simulations. As many common WSN simulators cannot support both of TX power adjustment and multi-channel communication, we use Castalia [21] which provides realistic wireless channels and radio models as well as realistic node behaviors such as clock drift. To further evaluate T^{2}C, we also implement T^{2}C using nesC on the TinyOS platform, and evaluate the performances through a real application running on a sensor network which consists of TelosB sensors.

A. Evaluation Metrics and Methodology

In our simulation, we use three types of metric—CDS features, energy efficiency, and dissemination latency. For CDS features, we first measure the number of non-target sensors as well as the depth of CDS in target-specified data dissemination. We also measure the number of dominators and the number of channels of CDS. For energy efficiency, we measure the total energy consumption as well as the number of data packets and control packets. For dissemination latency, we measure the time of the whole disseminating process. In the real application scenario, we use the completion time of sensors, the latency, and the number of lost packets to evaluate the proposed protocol.

We choose three protocols, MDeluge [10], CORD [12], and McTorrent [13] for benchmarking in our evaluation. To evaluate the features of CDS, we first compare T^{2}C with MDeluge [10] as they both support the target-specified data dissemination. We then choose CORD [12] for comparison as it is also a CDS-based protocol. To evaluate energy efficiency and dissemination latency, we choose CORD [12] and McTorrent [13] as they introduce the techniques of CDS and multi-channel communication, respectively.

The experiments consider various factors such as network scale, topology, network constituent, and data size. For scale and topology, we first use a set of grids (i.e., 5×10, 5×20, and 5×30) with sensor 1 as the sink. The distance between a sensor and its closest neighbor is 9 meters in these grids. These grid settings are also used in CORD [12]. To further evaluate T^{2}C, we use networks with 200, 300, and 400 sensors which are uniform-randomly distributed in an area of 100 m $\times$ 100 m.[4] For network constituent, we use networks in which 20%, 50%, or 80% of sensors are chosen randomly as target sensors. The data needed to be disseminated are divided into a number of pages. Each page can be transmitted using 128 packets, and in each packet the size of payload is 26 bytes. The data size we use in our experiments are set to 5, 10, and 15 pages. Given sizes of the program flash for popular sensor platforms (e.g., 48 KB for TelosB), these sizes of 16.25 KB, 32.5 KB, and 48.75 KB are practical.

We suppose sensors in our experiments are all equipped with CC2420 chips. The CC2420 chip has been widely used in WSN applications. TinyOS provides interfaces with 31 adjustable TX power levels and 16 available channels for users to program this chip. The initial TX power level is set to 9 in simulations. To evaluate the impact of TX power increasing, we set a dominator's TX power level based on its depth. Suppose the initial TX power level is set to l_0, if the depth of a dominator is d, its TX power level is $(l_0 + step \times d)$. The value of *step* is set to 1 and 2 in our experiments. We name T^{2}C$_n$ to represent T^{2}C with the *step* value of n. Except the experiments relating to target-specified data dissemination, we suppose networks in the evaluations contain target sensors only.

To better show the experimental results, we provide confidence intervals for each of the average values. All these confidence intervals are shown with 90% confidence level. This confidence level is also used in CORD [12] and McTorrent [13].

B. Simulation Results

1) CDS Features: Fig. 8 shows the average number of non-target sensors of CDS in networks with 200, 300, and 400 uniform-randomly deployed sensors. We repeat the experiments for 1000 times and the results show that compared to MDeluge, T^{2}C can reduce the number of non-target sensors by 31.7%–60.6%, 45.2%–69.6%, and 54.3%–75.3% in networks

[4]We only choose the networks in which all sensors are connected under the default TX power. The sink is also placed randomly in the area.

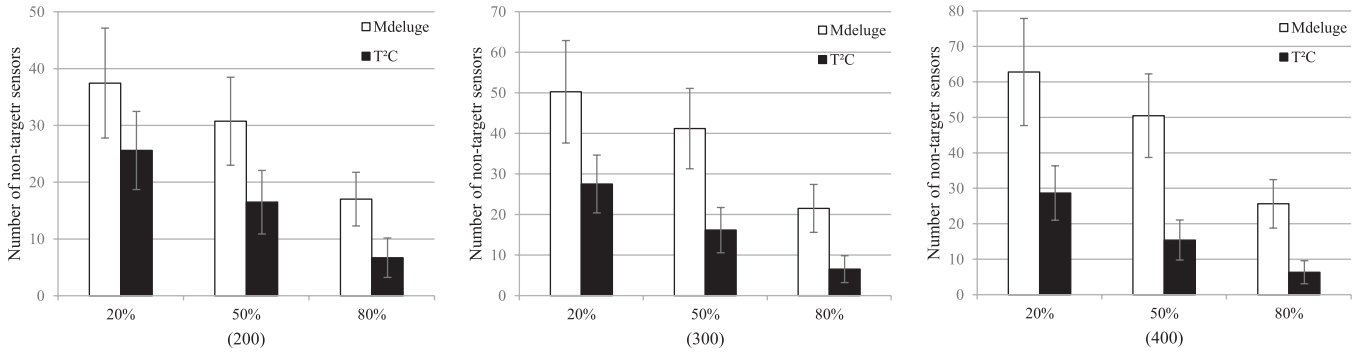

Fig. 8. Number of non-target sensors of CDS in randomly generated topologies.

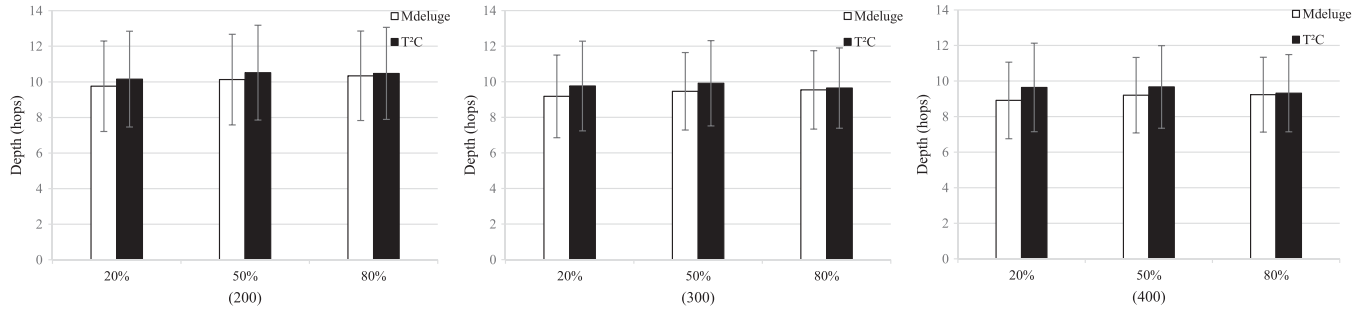

Fig. 9. Depth of CDS in randomly generated topologies.

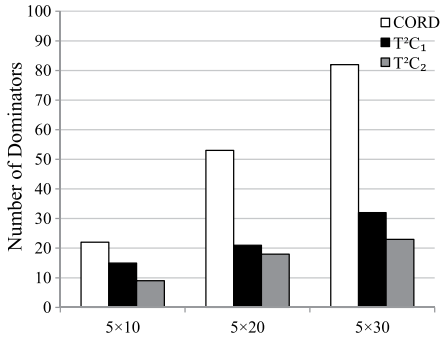

Fig. 10. Number of dominators in grid topologies.

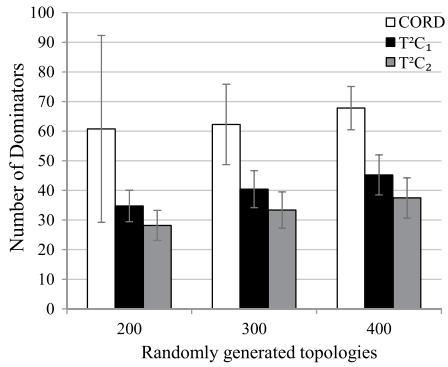

Fig. 11. Number of dominators in randomly generated topologies.

with 200, 300, and 400 sensors, respectively. The results show the effectiveness of T^2C on reducing non-target sensors of CDS, especially in large scale networks. This is because T^2C can converge target sensors into fewer paths when constructing a CDS, and sensors in large networks have more chances to choose the path with less non-target sensors. A concern of the target-aware CDS construction is that it may increase the depth of the CDS. Fig. 9 shows the average depth of CDS under the same network configuration. The results show that the difference of depths between T^2C and MDeluge is less than one hop. This is because 1) T^2C also constructs the CDS in the breath-first manner, and 2) the parent re-choosing operation in the CDS construction process can only select sensors with the same depth.

Fig. 10 shows the number of dominators of CORD and T^2C in grid topologies, respectively. Compared to CORD, T^2C_1 reduces the number of dominators by 31.8%, 60.4% and 61% in the three grids, while T^2C_2 reduces the number of dominators by 59.1%, 66% and 72%. Fig. 11 shows the average number of dominators in a set of networks with 200, 300, and 400 uniform-randomly deployed nodes. We repeat the experiments for 1000 times and the results show that T^2C_1 reduces the number of dominators by 33.28%–42.87% while T^2C_2 reduces the number of dominators by 44.74–53.61% compared to CORD. The results show that T^2C has significant achievement in reducing the number of dominators. The reason is that T^2C increases TX power of some dominators to let them cover larger area during the CDS construction process. Additionally, T^2C performs better in grid topologies because sink nodes in these grids are in the corner and dominators far from sinks can adjust their TX power higher. An interesting phenomenon is that for the average number of dominators of CORD in topologies of 200 nodes, the width of the confidence interval of the result is obviously larger. This is because some of the topologies contain much more nodes which have only one neighbor node. These topologies need much more dominators when constructing the CDS.

Fig. 12 shows the number of channels needed by T^2C_1 and T^2C_2 in the three grids, respectively. The results show that

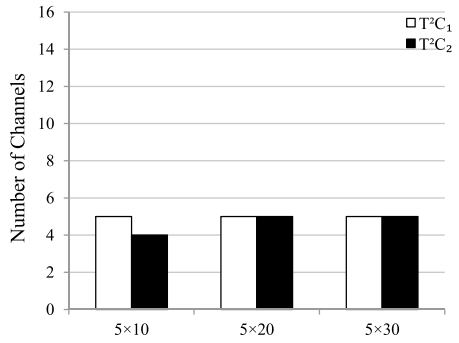

Fig. 12. Number of channels for grid topologies.

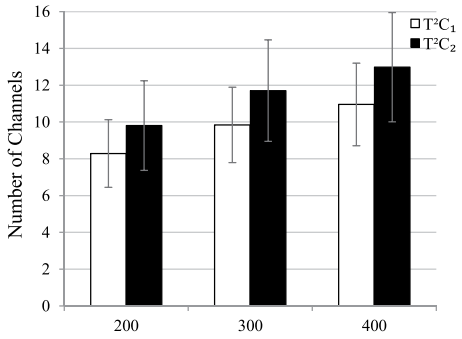

Fig. 13. Number of channels for randomly generated topologies.

channels needed in regular topologies are small and stable. For random topologies, Fig. 13 shows the average number of channels in the networks of different scales. We repeat the experiments on these networks for 1000 times. From these results, we observe that T^2C_2 in the network of 200 nodes requires over 16 channels (18 channels) for only once, and in the network of 400 nodes requires over 16 channels (up to 19 channels) for 26 times. We analyze networks which require more than 16 channels and find that sensors in these networks all have high density in some areas. However, we observe that WSNs in real deployed applications often have regular topologies [3], [5] and (or) are often deployed sparsely [1]–[3], [5] to avoid serious communication collisions. Therefore, we believe that 16 channels are sufficient for most WSN applications.

2) Energy Consumption: Total energy consumption is a fair metric for different types of data dissemination protocol. Fig. 14 shows the total energy consumption of CORD, McTorrent, and T^2C, respectively, for various networks with the data size of 5 pages. For CORD, the total energy consumption consists of the energy used for constructing CDS and disseminating data. For T^2C, the total energy consumption also includes the energy used for disseminating channel assignment information. Compared to CORD, T^2C_1 and T^2C_2 reduce the total energy consumption by 15.5%–43.0% and 33.0%–54.2%, respectively. Compared to McTorrent, T^2C_1 and T^2C_2 reduce the total energy consumption by 12.8%–45.8% and 30.9%–56.4%, respectively. An interesting phenomenon is that when of network scale grows, T^2C saves more energy compared to both CORD and McTorrent. This is because 1) T^2C needs less dominators especially in large scale networks, more sensors (dominatees) can turn off their radios periodically during the data disseminating process. 2) T^2C adopts the multi-channel communication to speed up data dissemination and reduce lost packets, less slots are needed to finish data dissemination. These two factors make the energy consumed by listening and receiving operations of sensors in T^2C much less than which of sensors in CORD and McTorrent.

To evaluate the impact of data size on energy consumption, Fig. 15 shows the total energy consumption of the three protocols with the data size of 15 pages.[5] Compared to CORD, T^2C_1 and T^2C_2 reduce energy consumption by 17.1%–43.5% and 36.9%–58.3%, respectively. Compared to McTorrent, T^2C_1 and T^2C_2 reduce energy consumption by 23.7%–41.9% and 42%–57.1%, respectively. As the results are similar to that of 5 pages, it suggests that T^2C works well on reducing energy consumption with various sizes of data.

To further evaluate energy consumption, we analyze the effective utilization of energy. Fig. 16 shows the total number of different packets with the data size of 5 pages. For CORD and McTorrent, the percentages of data packets range from 83.0% to 89.7% and 70.6% to 72.9%, respectively. For T^2C_1 and T^2C_2, the percentages are 93.9%–95.4% and 95.3%–96.5%, respectively. The reduction of control packets shows effective use of energy of T^2C. T^2C uses less control packets for two main reasons: 1) T^2C has less source sensors (dominators) to send control messages, and 2) the DATA-NACK mode saves lots of control messages compared to the ADV-REQ-DATA mode of CORD and the ADV-REQ-CHN-DATA mode of McTorrent. With the growth of data size, T^2C still uses much less control messages as shown in Fig. 17.

3) Dissemination Latency: For CORD, the dissemination latency consists of time used for constructing CDS and disseminating data. For T^2C, the latency also includes time used for computing and disseminating channel assignment information. The lengths of slot in CORD and T^2C are both set to 6*s*. Fig. 18 shows the results of the three protocols for various networks with the data size of 5 pages. Compared to CORD, T^2C_1 and T^2C_2 reduce latency by 42.1%–63.2% and 52.7%–79.7%, respectively. Compared to McTorrent, T^2C_1 performs worse when the network scale is small (i.e., 50 sensors). This is because small networks cannot fully exploit T^2C's advantages of shortening disseminating paths and eliminating communicating collisions. However, with the growth of network scale, T^2C_1 reduces latency by 10.2%–24.9% compared to McTorrent. For T^2C_2, it reduces latency by 1.1%–39.8% compared to McTorrent. To evaluate the impact of data size on latency, Fig. 19 shows the dissemination latencies of the three protocols with the data size of 15 pages. The results are similar to that of 5 pages. Therefore, we conclude that T^2C also works well on reducing dissemination latency with various sizes of data.

C. Application Description and Results

We are also interested in how T^2C performs in real deployed sensor networks. Specifically, realistic wireless communication with unpredictable link quality fading may lead to poor performance (especially the dissemination latency) of T^2C. Therefore, we further evaluate T^2C using a real deployed sensor network running an intelligent lab application, iLab.

[5]We omit all results of 10 pages due to the space limitation.

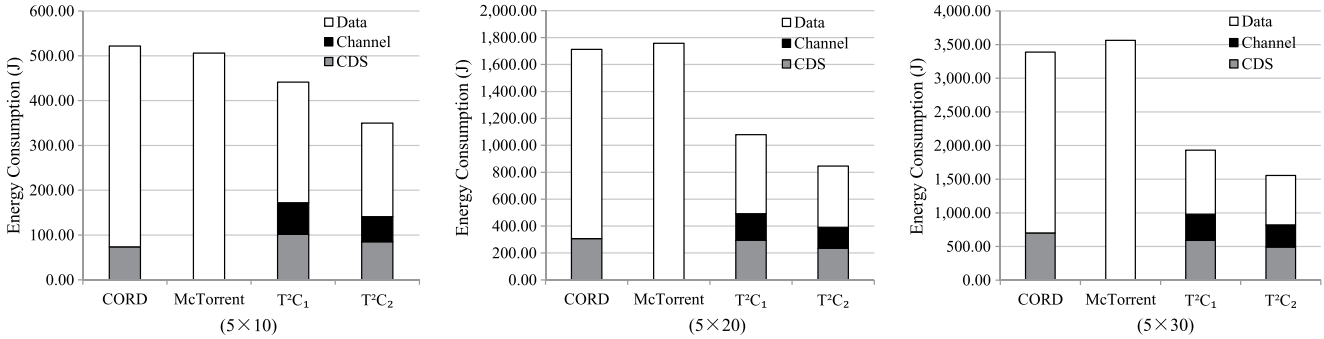

Fig. 14. Total energy consumption in various networks with the data size of 5 pages.

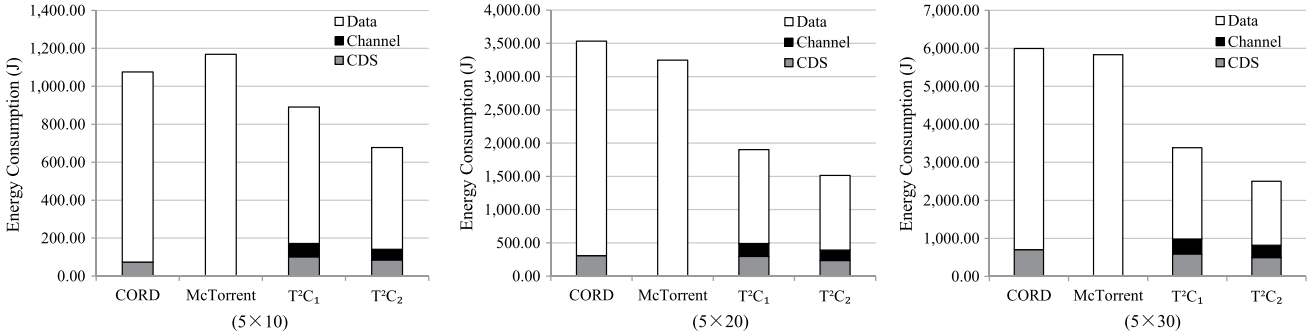

Fig. 15. Total energy consumption in various networks with the data size of 15 pages.

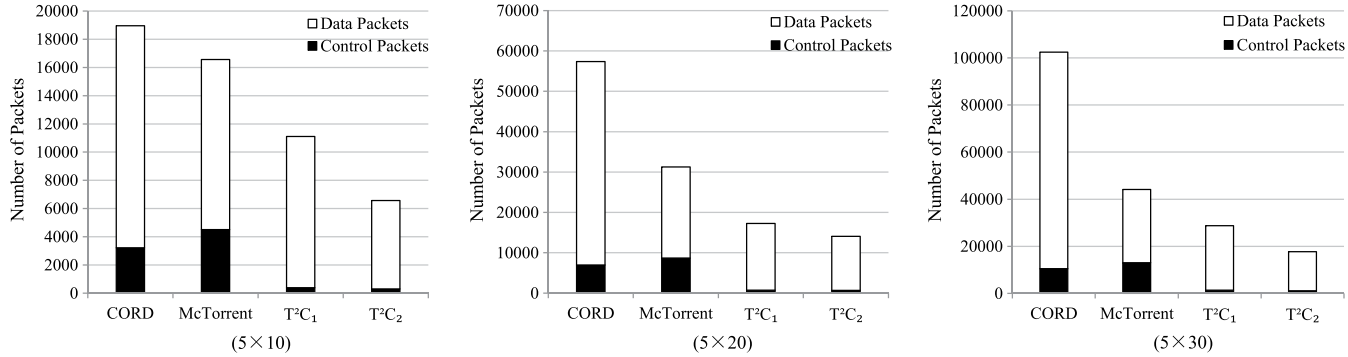

Fig. 16. Number of data packets and control packets in various networks with the data size of 5 pages.

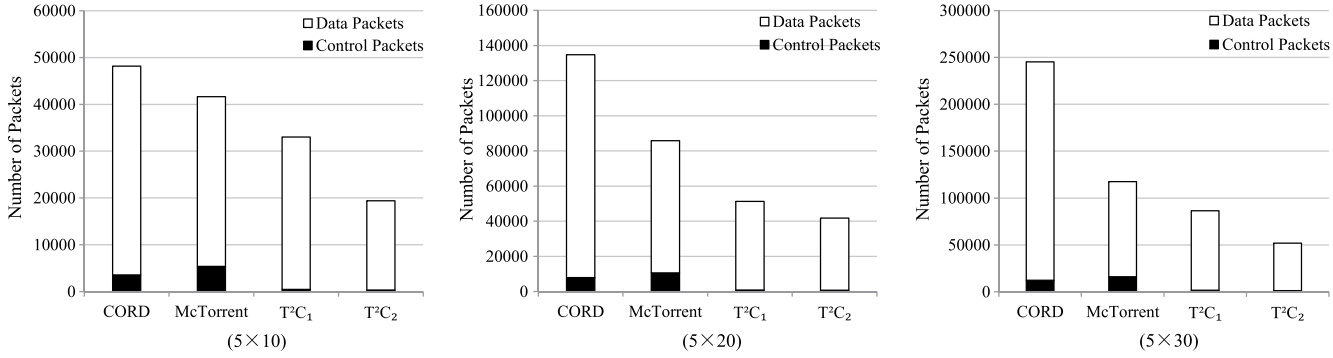

Fig. 17. Number of data packets and control packets in various networks with the data size of 15 pages.

The iLab application aims at automatically creating and maintaining a comfortable and green working environment. It detects environment (e.g., brightness, temperature) of seats which are being used and adjust the environment to fulfill default or customized requirements in the most green way (e.g., pulling curtains back rather turning on lights). Therefore, one key component of the application is the sensor network which is responsible for collecting environment information of the

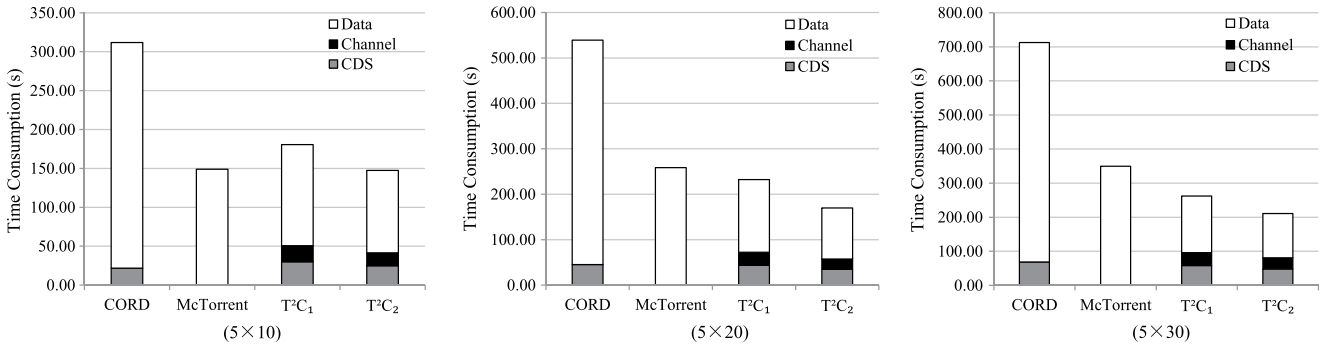

Fig. 18. Dissemination latency in various networks with the data size of 5 pages.

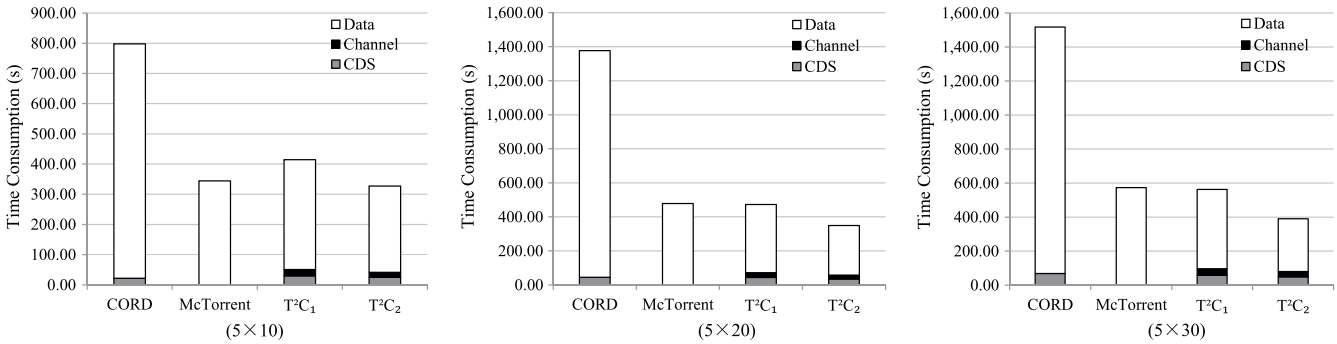

Fig. 19. Dissemination latency in various networks with the data size of 15 pages.

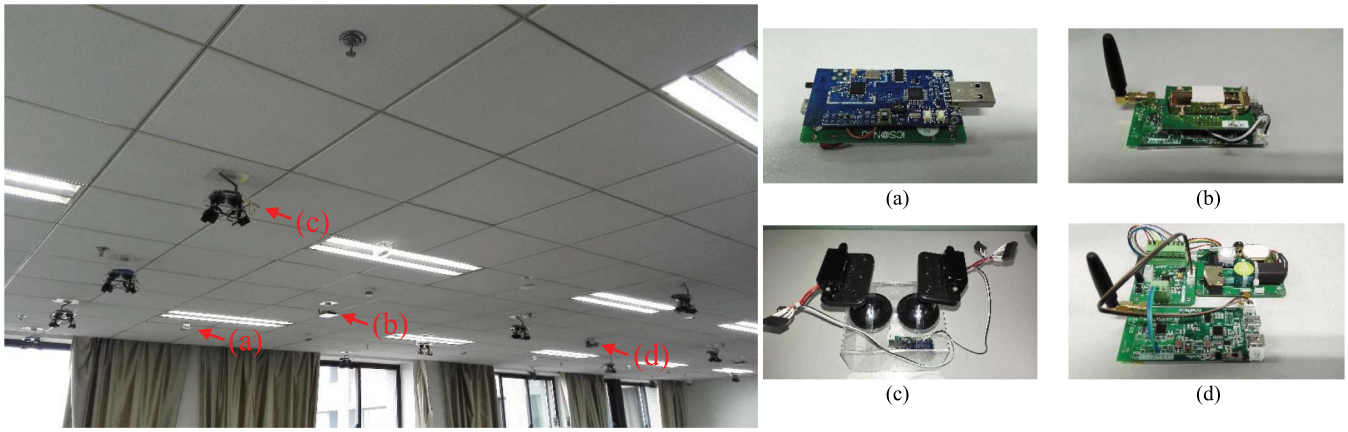

Fig. 20. Sensor network in the iLab application. (a) Temperature and light intensity sensor. (b) CO_2 sensor. (c) Active infrared sensor. (d) HCHO and PM2.5 sensor.

lab. The network consists of four types of sensor as shown in Fig. 20. 16 active infrared sensors detect whether seats are being used. 16 temperature and light intensity sensors provide temperature and brightness values around seats. Two CO_2 sensors as well as a HCHO and PM2.5 sensor provide indoor air quality.

The first version of iLab requires all sensors to send data to the base station periodically. However, temperature and brightness values around seats which are not being used are useless. To improve the efficiency of data collection, active infrared sensors in the new version of iLab are required to inform temperature and light intensity sensors the states of seats. Temperature and light intensity sensors will send data only if corresponding seats are being used. Therefore, we need to update the application code of active infrared sensors.

The size of the new code is 38.5 KB. We divide the code into 12 pages and disseminate them with T^2C_1. Fig. 21 shows the CDS constructed by T^2C_1 in the network with the sensor 1 as the sink. Except the target sensors (active infrared sensors), only three other sensors are selected to help to accomplish data dissemination. Meanwhile, with the ability of increasing TX power, the depth of the CDS is only 5 hops.

Fig. 22 shows the individual dissemination latency of all active infrared sensors in T^2C. According to the results, we find that the dissemination latency of sensors with the same depth is almost the same. In addition, differences of dissemination latency of sensors with adjacent depths are almost one slot (6s). The results indicate that T^2C can guarantee a comparatively same and low latency for sensors in the same depth.

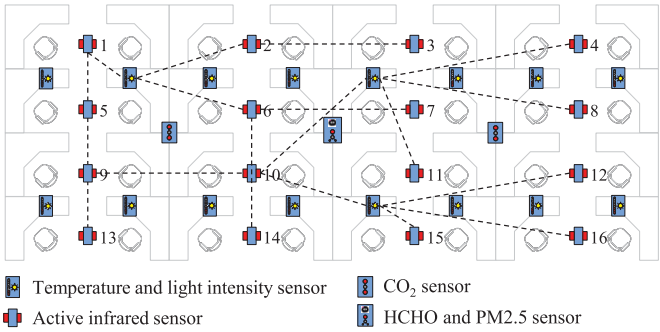

Fig. 21. CDS constructed by T^2C_1 in the network of iLab.

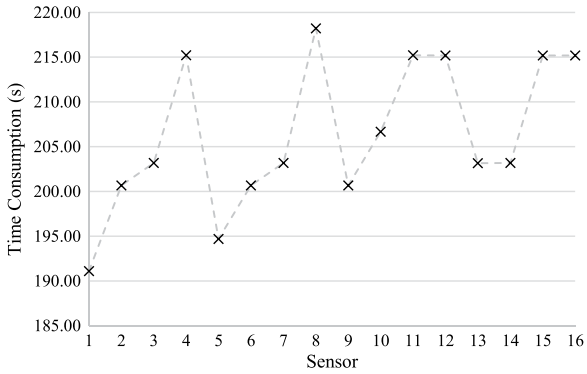

Fig. 22. Individual dissemination latency of active infrared sensors.

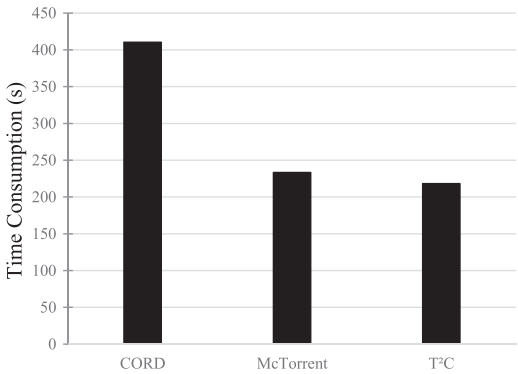

Fig. 23. Dissemination latency in the network of iLab.

Fig. 23 shows the dissemination latencies of the three protocols in the network. Compared to CORD and McTorrent, T^2C reduces the latency by 46.8% and 6.5%, respectively. The reasons are as follows: 1) T^2C introduces the multi-channel scheduling to speed up the data dissemination process. 2) The channel assignment of T^2C eliminates all communication collisions among dominators so that the time for recovering lost packets is reduced. 3) T^2C supports the target-specified dissemination and needs only to disseminate data to target sensors.

Fig. 24 shows the number of lost packets of the three protocols during the data dissemination process. Compared to CORD and McTorrent, T^2C has much less lost packets. This is because T^2C eliminates all communication collisions among dominators when disseminating data. Packet loss is mainly due to the occasional link quality fading.

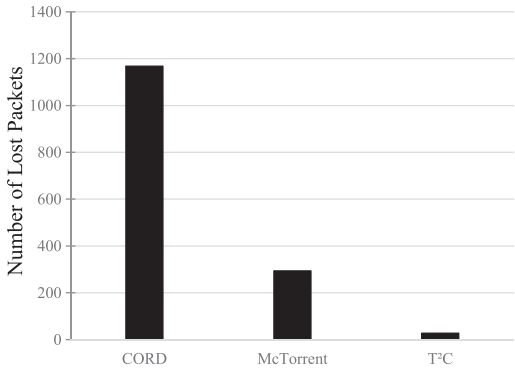

Fig. 24. Number of lost packets in the network of iLab.

These results suggest that T^2C works well in realistic sensor networks.

VI. Conclusion

This paper presents T^2C, a target-aware, TX power-adaptive, and collision-free bulk data dissemination protocol for multi-hop, multi-channel WSNs. By adopting the target-aware CDS construction, T^2C supports target-specified data dissemination with less non-target sensors. By reasonably increasing TX power of dominators, T^2C significantly reduces the total energy consumption of the data dissemination. To the best of our knowledge, it is the first work which focuses on utilizing excess battery power of sensors to improve energy efficiency during data dissemination. In addition, we introduce the collision-based channel assignment strategy which eliminates communication collisions among all dominators. With this strategy, T^2C further reduces the dissemination latency. We have implemented T^2C and evaluated it through simulation studies and a real application scenario. The results show that with the growth of network scale, T^2C saves more energy as well as reduces more latency when disseminating data compared to the-state-of-the-art approaches.

For our future work, we attempt to make the increment of TX power more intelligent. In the current version of T^2C, only the extra battery power of sensors before data dissemination can be used to increase their TX power. If we can predict the battery power usage of sensors, we can "overdraw" battery power of some sensors to aggressively increase the TX power to further improve data dissemination efficiency. The battery power overdrawn will be paid back after running application code for a period of time. Another problem of T^2C is the limitation of available channels. Although we can partition a network when the channels are not sufficient, this approach will increase dissemination latency. In the future version of T^2C, we want the TX power increment to be supervised by the prediction of channel assignment to avoid this problem. Apart from these issues, we will also take the failure recovery of the CDS as an important part of our future work. We plan to tackle this problem by introducing the partial CDS reconstruction. That means when a dominator fails, all its descendant nodes will reconstruct a sub-CDS with a root linking to the original one.

REFERENCES

[1] G. Liu *et al.*, "Volcanic earthquake timing using wireless sensor networks," in *Proc. IPSN*, Apr. 2013, pp. 91–102.

[2] V. Dyo *et al.*, "Evolution and sustainability of a wildlife monitoring sensor network," in *Proc. ACM SenSys*, 2010, pp. 127–140.

[3] M. Ceriotti *et al.*, "Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels," in *Proc. IPSN*, Apr. 2011, pp. 187–198.

[4] D. J. A. Bijwaard, W. A. P. van Kleunen, P. J. M. Havinga, L. Kleiboer, and M. J. J. Bijl, "Industry: Using dynamic wsns in smart logistics for fruits and pharmacy," in *Proc. SenSys*, 2011, pp. 218–231.

[5] O. Chipara, C. Lu, T. C. Bailey, and G.-C. Roman, "Reliable clinical monitoring using wireless sensor networks: Experiences in a step-down hospital unit," in *Proc. SenSys*, 2010, pp. 155–168.

[6] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming wireless sensor networks: Challenges and approaches," *IEEE Netw.*, vol. 20, no. 3, pp. 48–55, May 2006.

[7] R. Bajwa, R. Rajagopal, E. Coleri, P. Varaiya, and C. Flores, "In-pavement wireless weigh-in-motion," in *Proc. IPSN*, Apr. 2013, pp. 103–114.

[8] R. Fontugne *et al.*, "Strip, bind, and search: A method for identifying abnormal energy consumption in buildings," in *Proc. IPSN*, Apr. 2013, pp. 129–140.

[9] P. Sikka *et al.*, "Wireless adhoc sensor and actuator networks on the farm," in *Proc. IPSN*, 2006, pp. 492–499.

[10] X. Zheng and B. Sarikaya, "Task dissemination with multicast deluge in sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 5, pp. 2726–2734, May 2009.

[11] S. S. Kulkarni and L. Wang, "MNP: Multihop network reprogramming service for sensor networks," in *Proc. IEEE ICDCS*, 2005, pp. 7–16.

[12] L. Huang and S. Setia, "Cord: Energy-efficient reliable bulk data dissemination in sensor networks," in *Proc. IEEE INFOCOM*, 2008, pp. 1247–1255.

[13] L. Huang, S. Setia, and R. Simon, "McTorrent: Using multiple communication channels for efficient bulk data dissemination in wireless sensor networks," *J. Syst. Softw.*, vol. 83, no. 1, pp. 108–120, Jan. 2010.

[14] D.-Z. Du, *Connected Dominating Set: Theory and Applications*, vol. 77. New York, NY, USA: Springer-Verlag, 2012.

[15] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. SenSys*, 2004, pp. 81–94.

[16] V. Naik, A. Arora, P. Sinha, and H. Zhang, "Sprinkler: A reliable and energy efficient data dissemination service for wireless embedded devices," in *Proc. IEEE RTSS*, 2005, pp. 1–10.

[17] W. Dong *et al.*, "Bulk data dissemination in wireless sensor networks: Modeling and analysis," *Comput. Netw.*, vol. 56, no. 11, pp. 2664–2676, Jul. 2012.

[18] Y. Sankarasubramaniam, Ö. B. Akan, and I. F. Akyildiz, "ESRT: Event-to-sink reliable transport in wireless sensor networks," in *Proc. MobiHoc*, 2003, pp. 177–188.

[19] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: A reliable transport protocol for wireless sensor networks," in *Proc. WSNA*, 2002, pp. 1–11.

[20] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman, 1979.

[21] A. Boulis, "Castalia: Revealing pitfalls in designing distributed algorithms in WSN," in *Proc. SenSys*, 2007, pp. 407–408.

Xiaorui Zhu received the B.Sc. and M.Sc. degrees in computer science from HoHai University in 2006 and 2009, respectively. He is currently pursuing the Ph.D. degree in the Department of Computer Science at Nanjing University. His research interests include programming languages and reprogramming systems for wireless sensor network applications.

Xianping Tao (M'08) received the M.Sc. and Ph.D. degrees in computer science from Nanjing University in 1994 and 2001, respectively. He is currently a Professor in the Department of Computer Science at Nanjing University. His research interests include software agents, middleware systems, Internetware methodology, and pervasive computing. He is a member of CCF.

Tao Gu (S'03–M'07–SM'14) received the B.Sc. degree from Huazhong University of Science and Technology, the M.Sc. degree from Nanyang Technological University, Singapore, and the Ph.D. degree in computer science from National University of Singapore. He is currently an Associate Professor in the School of Computer Science and Information Technology at RMIT University. His research interests include mobile and pervasive computing, wireless sensor networks, distributed network systems, sensor data analytics, cyber physical system, Internet of Things, and online social networks. He is a Senior Member of the IEEE and a member of the ACM.

Jian Lu received the B.Sc., M.Sc. and Ph.D. degrees in computer science from Nanjing University, China. He is currently a Professor in the Department of Computer Science and Technology and the Director of the State Key Laboratory for Novel Software Technology at Nanjing University. He serves on the Board of the International Institute for Software Technology of the United Nations University (UNU-IIST). He also serves as the Director of the Software Engineering Technical Committee of the China Computer Federation. His research interests include software methodologies, software automation, software agents, and middleware systems.