2015 IEEE 35th International Conference on Distributed Computing Systems

TeleAdjusting: Using Path Coding and Opportunistic Forwarding for Remote Control in WSNs

Daibo Liu[1], Zhichao Cao[2], Xiaopei Wu[2], Yuan He[2], Xiaoyu Ji[3], Mengshu Hou[1]
dbliu.sky@gmail.com, {caozc, wuxp, he}@greenorbs.com, xji@cse.ust.hk, mshou@uestc.edu.cn
[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China
[2]School of Software, TNLIST, Tsinghua University
[3]Department of Computer Science and Engineering, Hong Kong University of Science and Technology

***Abstract*—On-air access of individual sensor node (called remote control) is an indispensable function in operational wireless sensor networks, for purposes like network management and real-time information delivery. To realize reliable and efficient remote control in a wireless sensor network (WSN), however, is extremely challenging, due to the stringent resource constraints and intrinsically unrealizable wireless communication. In this paper, we propose *TeleAdjusting*, a ready-to-use protocol to remotely control any individual node in a WSN. We develop a coding scheme for addressing on the cost-optimal reverse routing tree. In the address of each node, all its upstream relaying nodes are implicitly encoded. Then through a distributed prefix-matching process between the local address and the destination address, a packet used for remote control is forwarded along a cost-optimal path. Moreover, *TeleAdjusting* incorporates opportunistic forwarding into the addressing process, so as to improve the network performance in terms of reliability and energy efficiency. We implement *TeleAdjusting* with TinyOS and evaluate its performance through extensive simulations and experiments. The results demonstrate that compared to the existing protocols, *TeleAdjusting* can provide high performance of remote control, which is as reliable as network-wide flooding and much more efficient than remote control through a pre-determined path.**

I. Introduction

Wireless sensor networks have been widely applied in many scenarios, such as environmental monitoring [1][2], disaster forecast [3], agricultural surveillance [4] and microclimate control [5], etc. In most sensor network deployments, it has been reported [6] that the predefined configurations often hardly adapt to the diversity and dynamics of environments [7] [8]. The optimal network configuration is usually achieved by lots of tuning and online adjustment. Due to the unattended locations of sensor nodes, remote control, namely delivering control packets from the sink to a certain node via multi-hop relay, is deemed as a key technique for network management.

Some efforts have been made to achieve remote control in the past few years. They can be classified into two main categories, i.e., unstructured and structured. In unstructured approaches [10][11][16], nodes flood control packets to their neighbors and the neighboring nodes likewise broadcast the message until the destination node (or target node) receives the control packets. Unstructured approaches guarantee the reliability by periodical advertisement and on-demand queries. Although flooding is efficient for network-wide dissemination, it consumes excessive energy and bandwidth for each control message on adjusting individual nodes. Therefore, unstructured approaches have poor applicability in practice. In contrast, structured approaches deliver control packets to the destination node along predefined paths to meet different design criteria. The paths are constructed based on some topology structures like reverse routing tree [17][18] or connected dominating set [19]. However, due to the resource-constraint on sensor nodes, it is generally inefficient to construct end-to-end paths for all nodes. Moreover, the topology structure is frequently changing due to the intrinsically unrealizable wireless links [21][23]. Therefore, the reliability of network tuning is hard to guarantee.

In this paper, we propose a novel way for remote control aiming to guarantee both the efficiency and reliability properties. To guarantee this end, several challenges should be taken into account. First, the structured path with the optimal cost of packet delivery should be constructed in a distributed manner. Second, the convergence speed, communication cost, computation overhead, and scalability of the path construction process should adapt to the diversity and dynamics of a network. Finally, to accommodate network dynamics, other nodes around the optimal relaying path should be involved to deal with lossy links and pursue opportunities of further improving packet delivery performance.

To address the above challenges, we propose *TeleAdjusting*, a ready-to-use remote control protocol for large-scale wireless sensor networks. First, according to the cost-optimal reverse routing tree [17] obtained by data collection protocol (e.g., CTP [20]), *TeleAdjusting* develops a coding based addressing scheme. In the address of each node, all the upstream relaying nodes from the node to the sink are implicitly encoded. Each node's address is assigned by its parent node. Then through a distributed prefix-matching process between the local address and the destination address, control packets are forwarded along a cost-optimal path. Moreover, TeleAdjusting incorporates opportunistic forwarding into the addressing process. The earlier wake-up nodes, which are much closer to the destination, around the optimal relaying path are exploited to relay the control packet, so as to improve the network performance in terms of reliability and energy efficiency.

The contributions of this work are as follows:

•We propose a ready-to-use remote control protocol, *TeleAdjusting*, to reliably and efficiently deliver control packets from the sink to any individual node in large-scale wireless sensor networks. *TeleAdjusting* can be easily extended to application scenarios of one-to-all or one-to-many packet dissemination.

1063-6927/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDCS.2015.78

•*TeleAdjusting* develops a novel distributed coding based addressing scheme that inherits the advantages of both structured and unstructured approaches. By incorporating opportunistic forwarding into our design, *TeleAdjusting* can simultaneously reduce the total transmission count and end-to-end latency.

•We implement *TeleAdjusting* with TinyOS [33] and conduct extensive simulations and experiments on real testbeds. As the experimental results show, *TeleAdjusting* can provide high performance of remote control, which is as reliable as network-wide flooding and much more efficient than remote control through a pre-determined path.

The rest of this paper is organized as follows. The background and motivation of this work is given in the next section. Section III introduces the detail design of *TeleAdjusting*. Sections IV presents the simulation and evaluation result. Section V discusses the related works. Section VI concludes this paper.

II. Background and Motivation

A. Background

During the last several years, learning from the real experiences of large-scale wireless sensor networks deployments, i.e., GreenOrbs [2] and CitySee [12], we become increasingly aware of the importance of remote control as an approach to achieve the optimal configuration of whole network. Specifically, with the change of network (topology, energy and environment, etc), the predefined protocol parameters may not be adaptive to current status for some nodes, such as the observed network anomaly in previous works [13][14][15]. The difference between the current status and the expected status might incur performance degradation.

In the projects of GreenOrbs and CitySee, nodes are usually deployed on the trunk of tall trees, with on average 180 centimeters above the ground. It is impractical and laborintensive to manually adjust the operation mode of these nodes. Hence, it urgently needs an efficient and reliable remote control protocol to automatically adjust the network configuration.

B. Motivation

As we have known, there are several approaches that can be used to deliver control packets from sink to an appointed destination. Nevertheless, there exists two main drawbacks for current solutions. First, it is energy inefficient to disseminate control packets to individual nodes by network-wide flooding. Second, it is vulnerable for remote control to deliver control packets along a predefined path which is constructed and maintained by sink. These methods just do a tradeoff between energy efficiency and reliability, which motivates us to build an new remote control protocol with these two properties simultaneously.

III. Design of TeleAdjusting

TeleAdjusting targets on developing a ready-to-use remote control scheme for various scales of sensor networks by integrating the structured path with gossip communication. In this section, we first give an overview of *TeleAdjusting*, and then detailedly introduce the construction of path code and the adopting of opportunistic forwarding for reliable and efficient control packet forwarding.

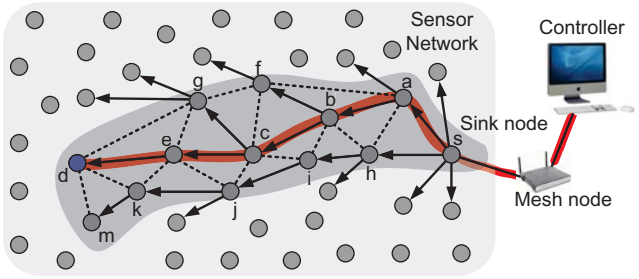


Fig. 1. Overview of *TeleAdjuting* for remote control from sink to an individual node.

A. Overview

The operation overview of *TeleAdjusting* is shown as Figure 1. Sensor nodes are scattered in monitoring area. All of them forward the collected data packets toward sink along a cost-optimal path. After receiving a data packet, the sink will further forward it to the remote data center (controller in the figure) through mesh node. On the controller side, network manager will monitor the abnormal situation by real-time data analysis. Once detecting an anomaly, the manager utilizes network diagnostic methods to confirm the root cause.

Each node has a path code (or reverse path code mentioned in following sections) where all its upstream relaying nodes are implicitly encoded. And such code will be reported to the remote controller. By obtaining the root cause of an abnormal behavior, the manager needs to send a control packet to adjust its corresponding operation mode. Here, the control packet that contains the path code of destination and related control parameters generated by the controller is delivered to the sink by the mesh. Then, the sink forwards the control packet downwards along the encoded path, which is indicated as red bold line in Figure 1. We will next elaborately discuss the forwarding strategies. Specifically, the control packet is opportunistically forwarded in the area around the encoded path toward the destination, denoted as shadow region.

In Figure 1, assume that sink **s** has a control packet with **d** as its destination. Here, the line with arrow means parent-child relationship. And the neighbor relationship is denoted as dotted line. The encoded forwarding path, generated by the attached path code, goes through **s**, **a**, **b**, **c**, **e**, and **d**. Practically, the control packet can be passed through different paths, such as path **s**, **a**, **f**, **g**, or path **s**, **h**, **i**, **c**, **e**. It is because *TeleAdjusting* takes full advantage of forwarding opportunities from temporally available neighbors, who can assist to improve the reliability of remote control and reduce the end-to-end latency.

To achieve this goal, several problems should be solved. The first is how to efficiently encode the reverse path information as nodes' path code address. The second problem is to design a distributed and light-weight algorithm for each node to identify whether it is on the reverse path given by sink. The last problem is to adopt opportunistic forwarding against dynamic wireless links. We introduce the details next.

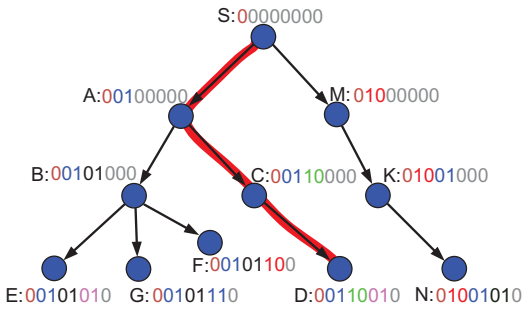


Fig. 2. Concept of path code construction. The parent node's valid path code is the prefix of any child node's path code. Parent node allocates unique position to each child node.

TABLE I. CHILD NODE TABLE, MAINTAINING CODE INFORMATION FOR SON NODES.

Child	Position	New code	Old code	Flag
A	1	00001000	00001000	1
B	2	00010000	00011000	1
⋮	⋮	⋮	⋮	⋮

B. Path Information

In this section, we introduce the process that *TeleAdjusting* generates reverse path code, by which the path from sink to an individual node is indicated as a short binary bit string and can be identifiable to any node in the network.

1) Concept of reverse path code: In this paper, we will use the concept of reverse path code to construct the forwarding path from sink to any specific node, rather than the fixed routing information. The reverse path code means an array of 0-1 string that contains certain potential relationship with other nodes. Any node in the filed, once overhearing a control packet from its neighbors, it should use its reverse path code to determine whether it would be the encoded relay node or a better one than the encoded relay node toward the appointed destination. Figure 2 shows the procedure of constructing such code. *TeleAdjusting* iteratively generates a unique path code for each node starting from sink, which initially sets its code to **0**0000000 with only one valid bit (path code length is 1). Note that each node should select an upstream node as its parent according to link quality and the distance between the node and sink, e.g., A and M select S as their parent node. Through local coordination, S knows A and M select itself as their upstream node in the reverse path, so S provides a two bits space (two bits space can accommodate up to 4 positions and is enough for the discovered two children nodes and the potential hidden children nodes) and allocates different positions to them, which are **01** and **10**, respectively. Hence, by appending the prefix code of parent node, the path code of A and M are **001**00000 and **010**00000, respectively, with 3 valid bits. Iteratively, after setting the path code, A and M allocate different positions to their downstream (children) nodes. Eventually, each node is allocated with a unique path code with specific valid bits, such as B's **00101**000 with 5 valid bits, E's **0010101**0 with 7 valid bits, and D's **0011001**0 with 7 valid bits etc.

When a node overhears a control packet, by checking whether its valid path code or a neighbor's valid path code is the prefix of the destination's code, and whether it is much closer (with a longer matched prefix of code of the destination) to the destination than the transmitter and the attached expected relay, the node decides to forward the control packet or not. Take Figure 2 as an example, assuming M is the neighbor of S and C, if M overhears a control packet transmitted by S attaching the content: the identification of the destination D, the path code of D (**0011001**0), the expected relay A, and the valid path code length of A (3). M knows it can assist to forward the control packet to C, no matter M knows A's path code or not, because A's valid path code length is less than C's path code length, and C is also on the given path from sink to the destination D. By using this mechanism, *TeleAdjusting* can successfully decode node's implicit path and achieve opportunistic forwarding to improve control reliability, reduce single hop latency and total transmission hops.

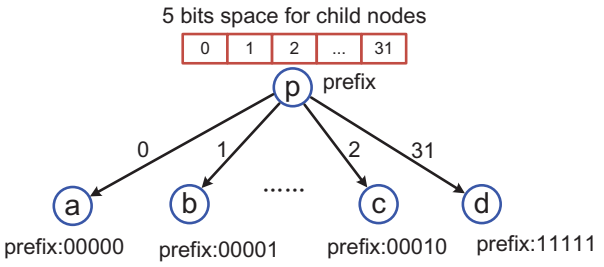


Fig. 3. One hop position allocation. p provides 5 bits space for child nodes a, b, c, and d, etc. prefix is p's path code.

2) Initial construction: The reverse path code is constructed starting from the sink and sequentially generated according to the hop count from sink to each node. Due to the serious network dynamics at the stage of network initialization, the operation of generating path code should not be started during this stage to reduce control overhead. As mentioned above, a node starts to generate its path code when the selected parent node (which is the same to the parent node in data collection protocol at the beginning of building a network) has published its path code and provided the appropriate bit space. In the construction of network topology, if there is no further finding of new child node (sometimes also called son node) for ten rounds of routing beacons (the duration is 10×wake-up interval) after the trigger of *parent found event*, *TeleAdjusting* estimates the appropriate size of bit space (π) considering the number of child nodes (N), shown in Line 1-6 of Algorithm 1. And then, as a parent node, it deterministically allocates different positions to each child node by coordination.

As Figure 3 shows, **a**, **b**, **c**, **d** et al. are with the same parent node **p**. The path code of **p** is a binary string denoted as *prefix*. By providing 5 bits space, which can support up to 32 child nodes, each child node is allocated a sequence number (position) followed by *prefix* as its path code. For example, **c** selects the third position of the space, hence its path code is *prefix:00010*. The space size is determined by the actual number of child nodes. The detail allocation strategy is shown as follows, containing the deterministic position allocation, position request, and allocation acknowledgement.

Algorithm 1 Initial Position Allocation Algorithm

Input: meet the condition of initial construction, approximate number of children nodes, N, in children nodes set **S** shown as Table I;
Output: initial position allocation.

Computing the needed number χ of positions for the discovered and potential hidden child nodes: $\chi = N + \lfloor 10, \frac{N}{2} \rfloor$;
$i = 1$;
while $\chi \leq 2^i$ **do**
  i++;
end while
$\pi \leftarrow i$; // π is the space size
for $i = 1$; $i < N$; i++ **do**
  $\mathbf{S}_i \leftarrow$ *unique*(*position*); // allocate unique position in $[0 : 2^\pi]$
  Flag($\mathbf{S}_i$)=0; // *unconfirmed* flag;
  Consecutively broadcast two *TeleAdjusting beacon* attaching all <child, position, flag> information;
end for

3) Deterministic position allocation: Once knowing the overall of children nodes, each node first determines the size of bit space. As a parent node, it allocates a unique space position to each known child node in deterministic way. The deterministic allocation is broadcasted as a *TeleAdjusting beacon*, containing the entries which consist of the identification (ID) of children nodes, the corresponding allocated positions, and the corresponding flags, shown as the first, the second, and the last columns of child node table (Table I). The process of initial position allocation is given in Algorithm 1. And then, the flag of each entry is set to *unconfirmed* status (0). The status will be changed to *confirmed* (1) if it is confirmed by a routing beacon sent by the child node containing the same allocated position. The information of all child nodes is maintained by *TeleAdjusting* in Table I. When a node overhears a *TeleAdjusting beacon* from its parent containing an allocated position for it, it generates its path code according to the parent's code and the allocated position according to the rules mentioned in Section III-B1. Furthermore, it replies a *confirmation frame* indicating the successful receipt.

Algorithm 2 Parent Node's Interaction Algorithm

Input: receiving a *routing beacon* or *position request frame* from child node ID; p is the allocated position of ID; <**S**, **P**, **F**> is the maintained child node table in Table I, where **S** denotes the child node set, **P** denotes the corresponding position set, and **F** denotes the corresponding flag set, respectively, the size of these sets is M;
Output: consistency of position allocation.

for $i = 0$; $i < M$; i++ **do**
  if ID==$\mathbf{S}_i$ && p==$\mathbf{P}_i$ && $\mathbf{F}_i$==0 **then**
    $\mathbf{F}_i \leftarrow 1$; // *confirmed* flag
  else if ID==$\mathbf{S}_i$ && $p \neq \mathbf{P}_i$ **then**
    $\mathbf{F}_i \leftarrow 0$; reallocate a position to ID;
    Send an *allocation acknowledge frame*;
  else if $ID \notin$**S** **then**
    if no free position **then**
      Extend bit space;
    end if
    Allocate a free position k to ID;
    $\mathbf{S}_i \leftarrow ID$, $\mathbf{P}_i \leftarrow k$, $\mathbf{F}_i \leftarrow 0$;
    send an *allocation acknowledge frame*;
  end if
end for

4) Position request and allocation acknowledgement: If a node is not allocated a position by its parent, or did not overhear its parent's *TeleAdjusting beacon*, by overhearing a beacon (e.g., a beacon sent by its parent node or a neighbor with the same parent) which indicates its parent has allocated positions to children nodes, it sends a *position request frame* to its parent requesting a position in deterministic way. Once receiving this type of frame, as a parent node, it selects a free position and allocates it to the requester by feeding back an *allocation acknowledge frame*, and then sets the related flag to *unconfirmed* status. If there is no free position, it first extends the bit space without changing all children nodes' positions discussed later, and then allocates a free position to the requester and notifies the change to all children nodes by broadcasting a *TeleAdjusting beacon*. The space extension will trigger all children nodes to update their path codes, and we discuss it detailedly in Section III-B6.

5) Position maintenance: Wireless link burstiness might result in packet loss, so as to cause inconsistent confirmation between parent and children nodes. To guarantee the consistency, each child node will notify its position attaching to routing beacon. Once overhearing a routing beacon sent by child node, the parent checks whether the attached position has been allocated for the child node. If the position has been allocated for the child node, the parent sets the corresponding flag to *confirmed* in Table I, otherwise the parent immediately deterministically reallocates a free position to the child node.

6) Space extension: Although *TeleAdjusting* has provided enough space to against the joining of new children nodes, like the computation of space size in Algorithm 1, the sudden increase of children nodes caused by network dynamics can result in insufficient space. Then, *TeleAdjusting* extends one bit based on previous bit space once it knows the shortage of the bit space. After the extension, the previous allocated positions for children nodes remain unchanged, but *TeleAdjusting* notifies the expansion of space to all children nodes and neighbors by broadcasting a *TeleAdjusting beacon*. Once a child node overhears the notification, it updates its path code and iteratively notifies the change to downstream nodes.

Algorithm 3 Children Nodes' Interaction Algorithm

Input: receiving a *TeleAdjusting beacon* from parent; extracting child information sets <**S**, **P**, **F**> from the beacon with size M; *ID* is this node's identification and its allocated position is p;
Output: consistency of position allocation.

for $i = 0$; $i < M$; i++ **do**
  if $\mathbf{S}_i$==ID **then**
    if $\mathbf{P}_i$==p && $\mathbf{F}_i$==0 **then**
      Send a confirmation frame; generate path code;
    else if $\mathbf{P}_i \neq p$ **then**
      $p \leftarrow P_i$; send a confirmation frame; generate path code;
    else if Space extension **then**
      Update path code; notify its change to child nodes by broadcasting a *TeleAdjusting beacon*;
    end if
  end if
end for
if $ID \notin$**S** **then**
  Send a *position request frame*;
end if

The main processes of the maintenance of position consistency between parent node and all children nodes is given by Algorithm 2 and Algorithm 3. The consistency is maintained by sending a broadcast beacon or a unicast frame without resetting the default routing beacon frequency of original protocol stack.

By understanding how to construct path code, in the

following section we give the detailed specification of exploiting path code to opportunistically forward remote control packets. In addition to child node table, each node also maintains its own path code and records all neighbors' path codes in a *neighbor code table* with entries of form $(neighbor, new\ code, old\ code)$. The old code for each neighbor will be remained for a period of time to guarantee reliable control against code change caused by network dynamics. The form of $new\ code$ and $old\ code$ just like the third and forth columns in Table I, respectively.

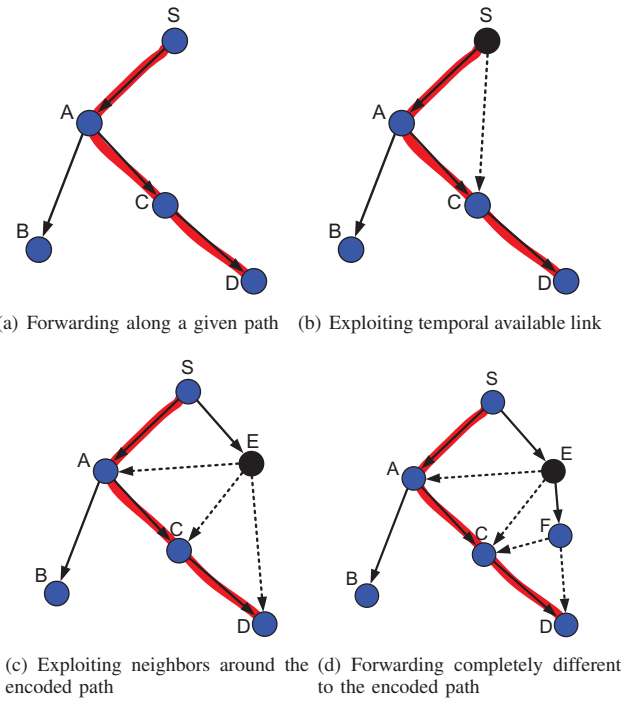
C. Forwarding Strategy

To forward downwards a control packet, the current relay attaches an expected relay, which means the practical next relay in the encoded path towards the destination should not be farther than the expected one. Once overhearing a control packet attaching the ID of an appointed destination, the path code of the destination, an expected relay node, and the valid path code length of the expected relay, a node should receive and relay the control packet if at least one of the three conditions is satisfied: (1) it is the expected relay; (2) it is a relay in the encoded path, and much closer to the destination than the expected relay, or (3) one of its neighbor is belong to the second case. Through prefix-match queries, the node can effectively check whether it satisfies one of the three conditions. With this forwarding strategy, *TeleAdjusting* can forward a control packet downwards from sink to any individual node through a path around the given one implicitly encoded in the path code of the appointed destination. The detailed forwarding strategies are given as follows.

1) Forwarding along the encoded path: As Figure 4(a) shows, the red bold line denotes the encoded path from sink (S) to the destination (D), going through A and C. For each downwards forwarding, the practical relay is just happen to be the expected relay. Hence, the control packet is actually forwarded downwards along the given path implicitly encoded in the destination's path code.

2) Opportunistically exploiting available relays: However, for a duty-cycled wireless sensor network, nodes' wake-up schedule is likely different, especially in an asynchronous network. Hence, when a node forwards a control packet, the currently available relays might change over time. To reduce transmission count and end-to-end latency from sink to the destination, if possible, *TeleAdjusting* opportunistically utilizes temporally available relays without strictly residing in the encoded path. Specifically, if there is an awake node which can forward the control packet toward destination providing more progress than the expected relay, it acknowledges the transmitter and assists to relay it. Otherwise, when the expected relay wakes up, it receives the control packet and continuously forwards downwards.

As Figure 4(b) shows, S intends to forward downwards a control packet to D, by setting A as the expected relay. C overhears the packet before A. By matching the path code between C and D, C confirms that it is in the path from S to D (the path code of C is the prefix of D's path code), and it can provide much more routing progress than A toward D because the prefix size of C is larger than that of the expected relay (A). C immediately acknowledges the control packet and forwards it by setting D as its expected relay. This case shows the utilization of temporally available relays to reduce transmission count and latency. In low power wireless networks, asynchronized wake-up schedule and link burstiness are two significant feature. Hence, the utilization of earlier wake-up and temporally available relays can improve the performance of *TeleAdjusting*.

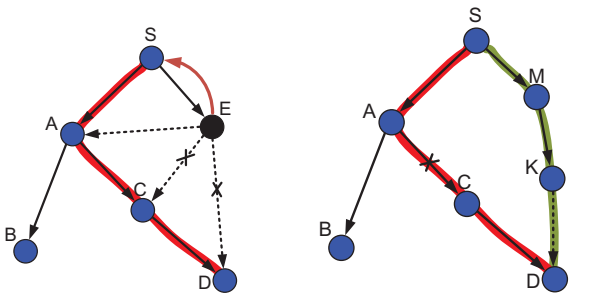


(a) Forwarding along a given path (b) Exploiting temporal available link

(c) Exploiting neighbors around the encoded path (d) Forwarding completely different to the encoded path

Fig. 4. Forwarding strategies. The solid lines with arrow denote the parent-children relationship, the red bold line denotes the encoded path from sink to D, going through S, A, and C, the dotted lines denote neighbourhood, and the black node holds the control packet now.

The other cases of exploiting opportunistic forwarding are shown as Figure 4(c) and 4(d). In Figure 4(c), although E is not in the encoded path, by querying and matching neighbors' path codes with the destination's path code, it knows that forwarding to a neighbor C or D is better than to A (no matter A is E's neighbor or not). Then, the control packet will not be strictly forwarded downwards along the encoded path, but forwarded downwards around the path by exploiting any possible relay superior than the expected relay. To make full use of all available relays, E sets C rather than D to be its expected relay (superior than A) in the following forwarding. In the case of Figure 4(d), F is not in the encoded path, locating between E and D. By overhearing the control packet from E, it can relay the packet to D without the participation of C. This case indicates the practical forwarding path may be different to the encoded path to a great extent. However, the difference can improve the performance of *TeleAdjusting*, such as reduced end-to-end latency and less transmission count. It is noteworthy that the opportunistic forwarding strategy can guarantee the control packet won't be far from the encoded path, and finally will be forwarded to the destination.



(a) Backtracking forwarding (b) The destination unreachable problem and the countermeasure

Fig. 5. Backtracking forwarding and the mechanism against unreachable problem, where the crosses on lines denote the failed links, and the curve-line from E to S in (a) denotes the backtracking forwarding.

3) Backtrack: As mentioned above, to forward a control packet, each neighbor of the current relay will try to assist it by replying an acknowledgement as long as it can forward the packet much closer to the destination. Otherwise, if no one could overhear the control packet due to link burstiness, or the destination is unreachable through any neighbor due to network dynamics or node failure, the transmission will not be acknowledged. By recognizing this case after several transmission fails, the relay infers that it is temporally unreachable to the destination downwards through itself. Hence, the packet should backtrack to the previous upwards relay.

Take Figure 5(a) as an example, when E overhears a control packet and acknowledges it, the links from E to C and D are temporally failed. By setting C as the expected relay, E repeatedly forwards the control packet (e.g., more than 5 times). However, E is not acknowledged. Hence, E deems it unreachable to further forward toward the destination, and returns the control packet to S using a *feedback packet*, also it temporally sets an unreachable flag in the entries of the related neighbors (such as C and D) in neighbor table until it hears the corresponding routing beacon from them again. If A overhears the feedback packet from E to S, A will continue to forward the packet downwards because it knows the accessibility from A to D, going through C. Then, A acknowledges E to stop the transmission of E's *feedback packet*.

If S finally overhears the feedback packet of E, S will try to forward downwards the control packet again, setting A as the expected relay. Due to the infeasible path through E to D, E won't relay the packet, and then it is forwarded by A.

4) Against destination unreachable problem: In the worst case, if sink node deems to unable to successfully forward control packet to the destination along the path implicitly encoded in the destination's path code, sink node returns this case to controller, requesting a neighbor node of the destination with different path code to the greatest extent. And then the sink forwards the packet to the destination's neighbor, instructing it to directly deliver the control packet to the destination. Note that as a controller of a deployed sensor network, the local topology information of each node is necessary and likely known. For example, in Figure 5(b), the controller knows that K is a neighbor of D with a different path code and the link quality between them is high. Hence, once node D is unaccessible through A and C, S will forward the control packet to K along the path through M denoted as green line, and then K directly delivers it to D by unicast forwarding.

5) End-to-end acknowledgement: Once receiving a new control packet, the destination will send an acknowledgement packet to sink to notify the reception. For simplicity, *TeleAdjusting* transmits the acknowledgement as a data packet. Note that if the destination receives the control packet in broadcasting way, which implies the packet is forwarded downwards along the path indicated by the destination's path code, it sends the acknowledgement to its parent in deterministic way. However, if the destination receives the control packet from a neighbor in deterministic way, denoting the packet is forwarded downwards from another path contrary to the destination's path code and the path from its parent to sink is possible blocked, hence, the destination forwards the acknowledgement back to the neighbor. And then, the acknowledgement will be forwarded upwards along another path to sink.

IV. Evaluation

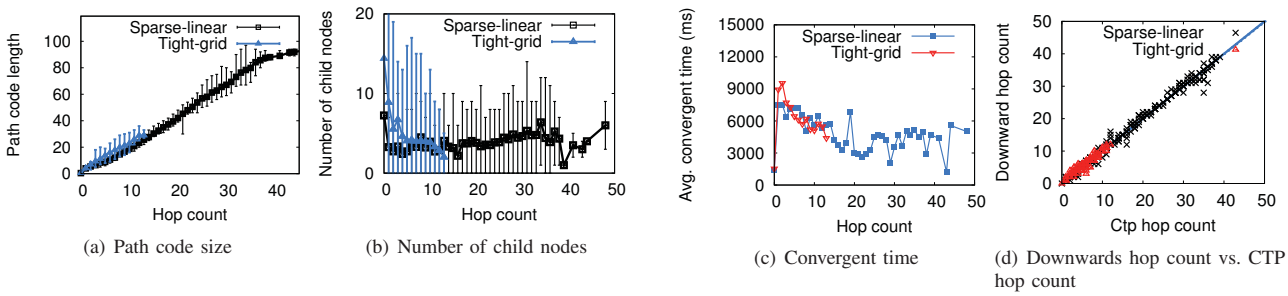
We implemented *TeleAdjusting* in TinyOS 2.1.1 [33]. The RAM and ROM consumption of the program are 896 bytes and 4516 bytes, respectively. To verify the feasibility and scalability of *TeleAdjusting*, we test the path code length and convergent rate of each node by simulation in 225 nodes networks and evaluation in indoor testbed with 40 Telosb nodes. Then, we test the performance of *TeleAdjusting*.

A. Simulation

In order to understand the performance of *TeleAdjusting* in generating path code under numerous network settings, in this section, we conducted our experiments in a low-level TinyOS simulator TOSSIM [27].

1) Simulation Setup: TOSSIM requires the user to supply the gain of the links used in the simulated topologies. We compute these gains using the Log Distance Path Loss model with a path exponent of four, to approximate challenging signal propagation environments. Furthermore, we model noise using the CPM model [26] adopted by TOSSIM. All simulations use the *meyer-heavy.txt* noise trace from [26]. We select radio model parameters in the simulations strictly according to the CC2420 radio hardware specification [28]. These parameters accurately reflect the performance of MicaZ motes in that they have the same modulation method, encoding method, frame length and path loss exponent.

In the simulation, we set node wake-up interval to 512ms. Network topology is constructed by CTP [20] with Trickle algorithm [29]. We deployed 225 sensor nodes randomly in a 200m$\times$200m square field divided into 15$\times$15 with high gain and 60m$\times$600m square field divided into 5$\times$45 grids with low gain, which are marked as *Tight-grid* and *Sparse-linear*, respectively. Sink was positioned in the center of the deployment field of *Tight-grid* and one endpoint of the field of *Sparse-linear*, and each sensor node connects to the sink over multiple hops. Then, we test the path code length of each node with different hop count to sink. Further more, we also generate path code in an indoor tested, deployed as 2$\times$11 grid networks providing up to 6 hops by setting the transmission power of CC2420 to 2.



(a) Path code size (b) Number of child nodes (c) Convergent time (d) Downwards hop count vs. CTP hop count

Fig. 6. Simulation results on path code length, number of child nodes, convergent rate, hop count in different networks (*Tight-grid* and *Sparse-linear*) with 225 nodes. *Tight-grid* denotes a network with 225 sensor nodes randomly deployed in a 200m×200m square field divided into 15×15 with high gain, and *Sparse-linear* is a network deployed in 60m×600m square field divided into 5×45 grids with low gain

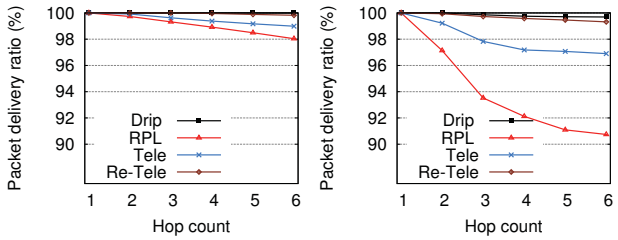
TABLE II. NODES' CODE LENGTH OF INDOOR EXPERIMENT.

Hop count	1 hop	2 hops	3 hops	4 hops	5 hops	6 hops
Avg. Code len	4.23	7.06	9.41	11.28	13.83	15.8
Min. Code len	3	4	5	7	8	12
Max. Code len	5	9	18	16	17	20

2) Path Code Length: As Figure 6(a) and Table II show, path code length almost increases linearly with hop count, no matter in simulated *Tight-grid* and *Sparse-linear* deployment field, or in the testbed scenarios. In the network with 15×15 grids (*Tight-grid*), 5 bytes (40 bits) buffer space is enough to identify a node's path code. With the increasing of hop count and decreasing of network density, such as the 5×45 grids (*Sparse-linear*), the expected path code length is larger than that of a tight network due to the waste of bit space in each hop supporting for the joining of new children nodes. Figure 6(b) shows the distribution of the number of children nodes in each hop. Although some nodes can solicit multiple children nodes in a tight network increasing the single hop bits space, it will reduce the total hop count of the entire network, so as to reduce the maximum path code length of the network. The simulation results of *Tight-grid* and *Sparse-linear* can greatly support this conclusion. In our indoor testbed experiment, a 6 hop network costs no more than 20 bits to support the maximum path code length given in Table II. From the simulation and evaluation results on path code length, we can conclude that *TeleAdjusting* can be efficiently adopted by existing protocols to provide explicit path code, without strictly considering network scale or network topology.

3) Convergent rate: As mentioned above, *TeleAdjusting* allocates position, requests a position, and acknowledges the allocation by attaching related information in routing beacon. By hearing a routing beacon carrying with code information, a node immediately maintains or updates the path information. Note that *TeleAdjusting* is triggered after the occurring of routing found event. As Figure 6(c) shows, after the trigger of *TeleAdjusting*, nodes can quickly generate its own path code and associate different positions for children nodes, without exceeding 20 beacons (each 512ms) time no matter in *Tight-grid* or *Sparse-linear* deployment field, and most of the nodes completed it less than 10 beacons time. The convergent time of *TeleAdjusting* is so little that it brings negligible control overhead to construct and maintain path information.

4) Reverse hop count: Other than the dynamically changed routing metric, the reverse path information is marked by a string of bits code, so there is no need to dynamically update the path information to avoid loops. Path information will be changed after parent's department or sharp degradation of link quality. Hence, a routing change event possibly does not trigger the change of path information. In a network, the reverse path from sink to an individual node may be different to the routing path from the node to sink. Figure 6(d) plots the reverse hop count (downward hop count in the figure) of each node versus the routing hop count which is constructed by CTP in our simulations. As the figure shows, the reverse hop count is very close to routing hop count. The ratio of average reverse hop count on the average ctp routing hop count is only 1.08.



(a) Without interference (b) With interference

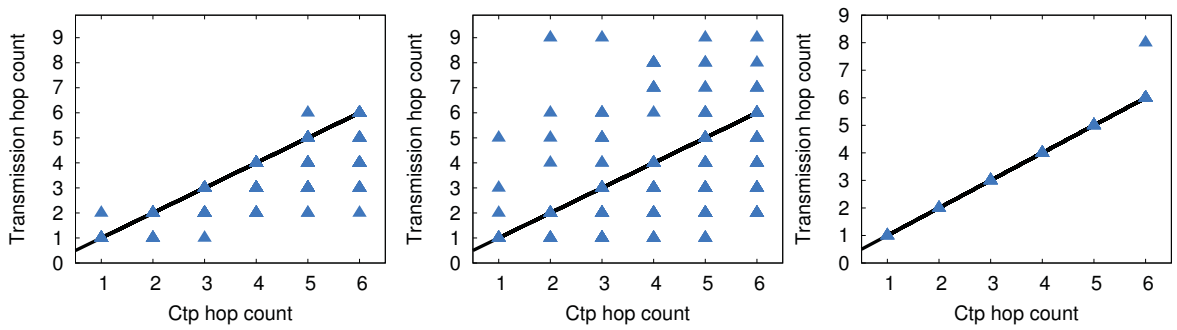
Fig. 7. The packet delivery ratio from sink to individual nodes (Drip, RPL, Tele, and Re-Tele) of indoor testbed experiments with two scenarios: (a) without being interfered by WIFI; and (b) interfered by WIFI.

From the simulation results, we conclude that the path code is adaptive to large-scale networks and diverse network topologies. *TeleAdjusting* can quickly converge after the routing found event, and induce limited control overhead to maintain and update path information. Although the reverse path is not always consistent with the data collection routing path, its hop count is very close to the path hop count of CTP.

B. Testbed Based Evaluation

In this section, we evaluate *TeleAdjusting* through indoor testbed experiments. We compare the network reliability, downwards forwarding efficiency, and the robustness of *TeleAdjusting* against network dynamics. To clearly show the performance of downwards forwarding using *TeleAdjusting*, we compare its performance with two protocols: RPL[31] and Drip[32].

RPL [31] is a routing protocol that provides any-to-any routing in low-power IPv6 networks, standardized by the IETF



(a) TeleAdjusting (b) Drip (c) RPL

Fig. 8. Accumulated transmission hop count of received packets in (a) *TeleAdjusting*, (b) Drip, and (3) RPL, vs. the corresponding CTP hop count of the destination.

TABLE III. AVERAGE NETWORK-WIDE TRANSMISSION COUNT FOR DELIVERING A CONTROL PACKET.

Protocol	TeleAdjusting	Drip	RPL
Transmission count (26^{th} channel)	4.43	109.35	5.17
Transmission count (19^{th} channel)	4.59	116.35	5.52

in March 2012. Its design is largely based on CTP [20], the reference data collection protocol for sensor networks. The RPL topology is a DODAG (Destination-Oriented Directed Acyclic Graph) built in direction of the sink. Any-to-any traffic is routed first upwards, i.e., toward the sink, until a common ancestor of destination and source is found, and then downwards, following the nodes' routing table. In our experiments, we only use the downward part of RPL protocol. Drip [32] is a reliable dissemination (flooding) protocol for wireless sensor networks. Although its efficiency is argued and improved by a lot of research work, it has a very high performance of the reliability of dissemination and convergence rate. In our experiments, we compare the reliability and end-to-end delay of *TeleAdjusting* to that of Drip.

We focus on three key metrics: (1) the Packet Delivery Ratio (PDR) of control packets in end-to-end communication; (2) the end-to-end latency of control packet; and (3) the energy efficiency. We use transmission count and radio duty cycle (using a packet collection protocol with the same IPI) as a metric for energy efficiency. Unless otherwise mentioned, all our experiments run for a period of raging from 3 hours to 9 hours. The results are averaged over at least 5 runs.

1) Evaluation Setup: We evaluate the performance of protocols on indoor testbed with 40 TelosB nodes (22 nodes on the testbed board and 18 nodes scattered around the testbed). In the experiments, we set the transmission power of CC2420 as 2 to ensure multi-hop communication (the maximum hop count is 6). The network topology is constructed by CTP [20]. Nodes' wakeup interval is set to 512ms and inter-packet interval is set to 10 minutes. Sink node randomly selects a destination, and sends a control packet to it every one minute. Once receives the control packet, the destination immediately sends an end-to-end acknowledge packet to sink. Each node records the count of received control packets, and periodically sends these counters to the controller through serial port of each node. In addition, we use a node sitting near the testbed to broadcast (using the highest power level 31) its time to all testbed nodes to provide global time synchronization.

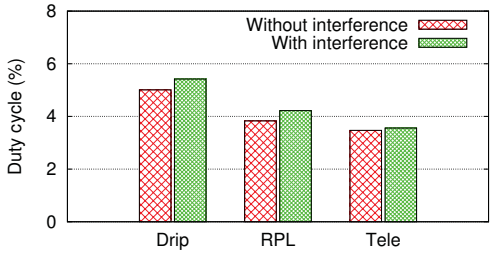
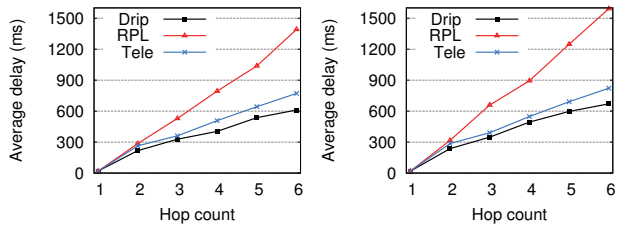


Fig. 9. The average radio duty cycle of each protocol of indoor testbed experiments with nodes interfered by WIFI or not.

2) Reliability: We conduct experiments in the indoor testbed with two scenarios: nodes interfered by WIFI (using the 19^{th} channel of ZigBee) and without being interfered by WIFI (using the 26^{th} channel). By integrating Drip, RPL, and $TeleAdjusting$ into the same protocol stack (CTP built upon LPL), we compute the average control packet delivery ratio (PDR) of each hop nodes and plot the PDRs in Figure 7. In the scenario of without being interfered by WIFI shown in Figure 7(a), Drip provides the most reliable remote control (PDR is almost 100% in the figure denoted as Drip) because the mechanism of Drip can guarantee the control packet will reach the destination after enough time. RPL's PDR (denoted as RPL) is decreased from 100% to 98% when hop count increases from 1 to 6. We infer that RPL is susceptible to network dynamics. Once occurring topology change (e.g., link degradation), the existing path information maintained by sink or each relay may be not consistent with the practical topology, but RPL uses deterministic forwarding strategy according to the maintained routing table, resulting in packet drop. *TeleAdjusting* without using the *against destination unreachable problem* mechanism (denoted as $Tele$ in Figure 7) discussed in Section III-C4 can forward 98.9% control packets to individual nodes when hop count is 6, and by using the mechanism (denoted as $Re\text{-}Tele$), it forwards more than 99.8% control packet to the fixed destination although hop count is 6.

By suffering the interference of WIFI, network dynamics increases. In this case, the drawback of deterministic forwarding strategy is much more obvious than the experiment using the 26^{th} channel. The PDR of RPL is sharply decreased



(a) Without interference (b) With interference

Fig. 10. The average end-to-end delay of control packet from sink to individual nodes verse hop count of indoor testbed experiments with two scenarios: (a) without being interfered by WIFI; and (b) interfered by WIFI.

from 100% to 90.1% when hop count increases from 1 to 6. Interfered by WIFI, the PDR of $TeleAdjusting$ is slightly decreased from 100% to 96.9% along with the change of hop count, compared with the change of RPL. By adopting the mechanism against unreachable destination mentioned in Section III-C4, the PDR of $TeleAdjusting$ is increased to 99.3% when hop count is 6, which is a little lower than that of Drip (99.7%).

From the experimental results, we can conclude that *TeleAdjusting* can provide a high control packet deliver ratio, which is very close to that of Drip and far better than that of RPL especially when network is dynamic.

3) Efficiency: In each of the experiments mentioned above, we also record the average number of in-network packet transmissions. Table III lists the average transmission count of all control packets in the indoor network (40 nodes). Although Drip is the most reliable protocol for remote control, the network-wide flooding causes a large number of redundant transmissions. Compared with RPL's 5.17 (5.52 in the 19^{th} channel) transmissions and *TeleAdjusting*'s 4.43 (4.59) transmissions under the experiments with 26^{th} channel, the total transmission count of Drip is 109.35 (115.35). Adopting Drip to achieve remote control on individual nodes is energy waste, sometimes the network-wide flooding may negatively affect data collection. *TeleAdjusting*'s opportunistic forwarding along a encoded path can further reduce the transmission count. Compared with RPL, the average transmission count (including duplicate transmissions) is reduced by more than 14.3% (16.8%). We also plot the accumulated transmission hop count (ATHX) of each received control packet (no matter the destination is set to the receiver or not) verse the corresponding CTP hop count from sink to the node in Figure 8. Overall, the ATHXs of incoming control packets of *TeleAdjusting* plotted in Figure 8(a) is likely less than the corresponding CTP hop count due to the opportunistic forwarding adopted by *TeleAdjusting*. As Figure 8(b) shows, Drip adopts network-wide flooding to guarantee reliability. However, it can not suppress inefficient duplicate transmissions. The ATHXs of control packets in RPL is likely equal to the CTP hop count because each relay forwards a control packet strictly according to its routing table. The exception in the top right of Figure 8(c) is possible caused by network loop.

At same time, the average duty cycle of 40 nodes is plotted in Figure 9. Since the network flooding consumes a great portion of energy in our experiments (with very limited data packets), the average duty cycle of Drip is 5.01% when the work channel is set to 26, and the duty cycle is 5.42% suffering the interference of WIFI. The duty cycle of RPL is 3.83% and 4.22% when the work channel is set to 26 and 19, respectively, which is far less than that of Drip. Compared with Drip and RPL, TeleAdjusting consumes the least energy due to the least control packet transmissions. Generally, we can conclude that *TeleAdjusting* can forward control packets to individual nodes not only reliably but also efficiently.

4) Latency: We also compute the end-to-end delay of each control packet forwarding from sink to each individual node, and plot it in Figure 10. As it shows, no matter the network is interfered by WIFI or not, by adopting network-wide flooding, Drip can rapidly forward a control packet to the appointed destination. Compared with Drip, the average end-to-end delay of RPL is proportional to wakeup interval and hop count. Between Drip and RPL, the end-to-end delay of forwarding a control packets by *TeleAdjusting* is far less than RPL's due to its opportunistic forwarding strategy, but a little larger than Drip's, because Drip exploits all possible nodes to forward a control packet, which makes full advantage of link burstiness and the earlier wake-up neighbors. Consider both efficiency and latency, *TeleAdjusting* can provide both efficient delivery of control packet and low end-to-end latency.

V. Related Works

Some forwarding protocols have been proposed for remote control in past few years. Most of them could be classified into two categories, structured and unstructured. In unstructured approaches [10][16], nodes flood the control packets to their neighbors until the destination receives these packets. [16] introduces a reliable bulk data dissemination protocol base on advertisement, requirement and data. [10] proposes a forwarder chosen scheme by considering the link quality to improve the energy efficiency. Moreover, some works [24][25] further improve the energy efficiency of unstructured flooding by noticing link correlation and utilizing network coding. Instead of whole network-wide flooding, *TeleAdjusting* adopts structured based path for remote control.

On the other hand, the control packets are forwarded to the destination through predefined path to improve the energy efficiency in structured approaches. The path is constructed on some topology structures like reverse routing tree [17][18][22] or connected dominating set [19]. [17] explores opportunistic routing to reduce the end-to-end delay in low duty-cycle wireless sensor network. [18] explores constructive interference to fulfil fast network-wide flooding. [19] is based on the off-line trained dissemination structure. All above methods are vulnerable to lossy links and topology change. ORPL [22] is an opportunistic routing protocol that supports any-to-any, on-demand traffic by using bitmaps and bloom filters to represent and propagate sub-tree in a space-efficient way, but the inherent false positive of bloom filter can incur multiple rounds of ineffectual transmissions, especially in the large-scale networks.

Additionally, opportunistic forwarding was applied early on geographic routing protocols [35][36] [37] to achieve any-to-any communication. In that case, the forwarding decision is based on physical node locations. This approach is often used analytically or in simulation, but is sometimes complex to put

in practice, because real nodes do not always have location information and because there is no direct mapping between distance and radio connectivity.

Different from existing approaches, *TeleAdjusting* exploits the adaptive coding based address scheme and opportunistic forwarding strategy to achieve both reliability and efficiency of remote control in wireless sensor networks.

VI. Conclusions

To realize reliable and efficient remote control protocol is urgent for large-scale wireless sensor networks. In this paper, we propose *TeleAdjusting*, a ready-to-use forwarding protocol to simultaneously improve both network efficiency and reliability for remote control. Based on the adaptive coding based address scheme, *TeleAdjusting* mainly constructs path code where all its upstream relaying nodes are implicitly encoded. Moreover, *TeleAdjusting* incorporates opportunistic forwarding into the addressing process, so as to improve the network performance in terms of reliability and energy efficiency, avoiding the bad influence incurred by lossy links. We implement *TeleAdjusting* in TinyOS-2.1.1. The simulation tests in TOSSIM and evaluation results in an indoor testbed show that *TeleAdjusting* can reliably and efficiently control individual nodes.

Acknowledgment

This study is supported in part by NSFC No. 61472067, National Key Technology R&D Program No. 2013BAH33F02, National Basic Research Program (973 program) No. 2014CB347800, Sichuan Province National Key Technology R&D Program No. 2012GZX0090, NSFC No. 61202444, NSFC No. 61472217, NSFC No. 61373146, NSFC No. 61170213, National Science Fund for Excellent Young Scientist No. 61422207, and Guangxi Key Lab of Wireless Wideband Communication & Signal Processing 2012 Open Fund.

References

[1] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, et al., A macroscope in the redwoods, *in Proc. of SenSys*, 2005.

[2] L.F. Mo, Y. He, Y.H. Liu, J.Z. Zhao, et al., Canopy closure estimates with greenorbs: Sustainable sensing in the forest, *in Proc. of SenSys*, 2009.

[3] C. Hartung, R. Han, C. Seielstad, S. Holbrook, FireWxNet: A multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments, *in Proc. of SenSys*, 2006.

[4] X. Wu, M.Y. Liu, Y. Wu, In-situ soil moisture sensing: Optimal sensor placement and field estimation, *ACM Trans. Sen. Netw. vol.8(4), pp.33:1-33:30*, 2012.

[5] L. Feng, T.Y. Xiang, Z.C. Chi, J. Luo, et al., Powering indoor sensing with airflows: a trinity of energy harvesting, synchronous duty-cycling, and sensing, *in Proc. of SenSys*, 2013

[6] M. Zimmerling, F. Ferrari, L. Mottola, T. Voigt, L. Thiele, pTunes: Runtime parameter adaptation for low-power MAC protocols, *in Proc. of SenSys*, 2012.

[7] X.L. Zheng, Z.C. Cao, J.L. Wang, Y. He, Y.H. Liu, ZiSense: Towards Interference Resilient Duty Cycling in Wireless Sensor Networks, *in Proc. of SenSys*, 2014

[8] D.B. Liu, Z.C. Cao, J.L. Wang, M.S. Hou, Y.J. Li, RxLayer: adaptive retransmission layer for low power wireless, *In Proc. of ACM MobiHoc*, 2014.

[9] J. Wang, Z.C. Cao, X.F. Mao, Y.H. Liu, Sleep in the Dins: insomnia therapy for duty-cycled sensor networks, *in Proc. of INFOCOM*, 2014

[10] D. Wei, Y.H. Liu, C. Wang, X. Liu, Link quality aware code dissemination in wireless sensor networks, *in Proc. of ICNP*, 2011.

[11] W. Dong, Y.H. Liu, Z.W. Zhao, X. Liu, et al., Link Quality Aware Code Dissemination in Wireless Sensor Networks, *IEEE Trans. Parallel Distrib. Syst. vol.25(7), pp.1776-1786*, 2014.

[12] X.F. Mao, X. Miao, Y. He, X.Y. Li, Y.H. Liu, Citysee: Urban CO_2 monitoring with sensors, *in Proc. of INFOCOM*, 2012.

[13] Q. Ma, K.B. Liu, T. Zhu, W. Gong, Y.H. Liu, BOND: exploring hidden bottleneck nodes in large-scale wireless sensor networks, *in Proc. of ICDCS*, 2014.

[14] X. Miao, K.B. Liu, Y. He, D. Papadias, et al., Agnostic diagnosis: discovering silent failures in wireless sensor networks, *in Proc. of INFOCOM*, 2011.

[15] Y.H. Liu, K.B. Liu, M. Li, Passive Diagnosis for Wireless Sensor Networks, *IEEE/ACM Trans. Network, vol.18(4), pp.1132-1144*, 2010.

[16] J.W. Hui, D. Culler, The dynamic behavior of a data dissemination protocol for network programming at scale, *in Proc. of SenSys*, 2004

[17] G. Shuo, Y. Gu, B. Jiang, T. He, Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links, *in Proc. of MobiCom*, 2009.

[18] M. Doddavenkatappa, M.C. Chan, B. Leong, Splash: Fast data dissemination with constructive interference in wireless sensor networks, *in Proc. of NSDI*, 2013.

[19] L. Huang, S. Setia, CORD: energy-efficient reliable bulk data dissemination in sensor networks, *in Proc. of INFOCOM*, 2008.

[20] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, Collection tree protocol, *in Proc. of SenSys*, 2009.

[21] K. Srinivasan, M.A. Kazandjieva, S. Agarwal, P. Levis, The β-factor: measuring wireless link burstiness, *in Proc. of SenSys*, 2008.

[22] S. Duquennoy, O. Landsiedel, T. Voigt, Let the tree bloom: scalable opportunistic routing with ORPL, *in Proc. of SenSys*, 2013.

[23] D. Aguayo, J. Bicket, S. Biswas, G. Judd, R. Morris, Link-level measurements from an 802.11b mesh network, *in Proc. of SIGCOMM*, 2004

[24] T. Zhu, Z.G. Zhong, T. He, Z.L. Zhang, Exploring link correlation for efficient flooding in wireless sensor networks, *in Proc. of NSDI*, 2010.

[25] S. Wang, S.M. Kim, Y.H. Liu, G. Tan, T. He, CorLayer: a rransparent link correlation layer for energy efficient broadcast, *in Proc. of MobiCom*, 2013.

[26] H. Lee, A. Cerpa, P. Levis, Improving wireless simulation through noise modeling, *in Proc. of IPSN*, 2007.

[27] P. Levis, N. Lee, M. Welshd, D. Culler, TOSSIM: accurate and scalable simulation of entire TinyOS applications, *in Proc. of SenSys*, 2003.

[28] Chipcon. CC2420 Product Information and Data Sheet. Avaiable at http://www.chipcon.com/.

[29] P. Levis, N. Patel, D. Culler, S. Shenker, Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks, *in Proc. of NSDI*, 2004.

[30] O. Landsiedel, E. Ghadimi, S. Duquennoy, M. Johansson, Low power, low delay: opportunistic routing meets duty cycling, *in Proc. of IPSN*, 2012.

[31] T. Winter (Ed.), P. Thubert (Ed.), and RPL Author Team, RPL: IPv6 routing protocol for low power and lossy networks, *RFC 6550*, 2012.

[32] G. Tolle, D. Culler, Design of an application-cooperative management system for wireless se nsor networks, *in Proc. of EWSN*, 2005.

[33] TinyOS: An Operating System for Sensor Networks, http://www.tinyos.net/.

[34] Contiki: The Open Source Operating System for the Internet of Thing, http://www.contiki-os.org/.

[35] S. Liu, K.W. Fan, P. Sinha, CMAC: an energy-efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks, *ACM Trans. Sen. Netw. vol.5*, 2009.

[36] M. Zorzi, R.R. Rao, Geographic random forwarding (GeRaF) for Ad Hoc and sensor networks: energy and latency performance, *IEEE Trans. on Mobile Computing, vol.2(4)*, 2003.

[37] D.B. Liu, Z.C. Cao, J.L. Wang, Y. He, et al., DOF: duplicate detectable opportunistic forwarding in duty cycled wireless sensor networks, *in Proc. of ICNP*, 2013.