Application of Multiple Interfaces for Balanced Tree Routing in Low-Delay Convergecast under IEEE 802.15.4e TSCH M2M Networks

Jian-Yuan Yu* and Hung-Yun Hsieh*†
*Graduate Institute of Communication Engineering
†Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan 106
Email: hungyun@ntu.edu.tw

***Abstract*—Delay time has been a key factor to consider in many real-time wireless applications, especially in industrial applications that the new TSCH mode proposed in IEEE 802.15.4e targets. Existing methods focus on minimizing the use of communication resources, yet they are unable to exploit the availability of channel multiplicity for achieving shorter average delay time in TSCH networks. In this paper, we propose a new method to construct cost-balanced routing trees that can utilize multiple interfaces at the coordinator. Our simulation results show that the proposed method is able to achieve low-delay data collection in TSCH networks.**

***Index Terms*—Cost-balanced tree, shortest-path tree, traffic-aware scheduling.**

I. Introduction

Time Slotted Channel Hopping (TSCH) proposed in IEEE 802.15.4e [1] is a new MAC-layer protocol that adapts for highly reliable transmission in industrial scenarios. It uses deterministic time slots to avoid contentions and channel hopping to mitigate the effects of multi-path fading. In the time domain, all nodes are synchronized by the superframe structure composed of multiple time slots, where a sender is expected to finish its transmission and receive the acknowledgment (ACK) from the receiver in a time slot. Every node keeps a global schedule to decide its behavior at each time slot; once there are no transmissions for it at the current time slot, a node will turn into the sleep mode to save energy, and it wakes up only when its turn comes. In the frequency domain, nodes perform channel hopping for each transmission. All nodes keep and follow a hopping pattern to avoid collisions or get stuck at deteriorated channels. Several advanced channel hopping methods have been proposed to cope with the dynamic quality of each channel [2].

To exploit the performance of TSCH, several papers have investigated the scheduling problems and proposed centralized and distributed algorithms. Centralized algorithms [3] can potentially reach the theoretical optimal performance at the cost of high computation complexity, while distributed algorithms [4] are more flexible. Of these, the traffic-aware scheduling algorithm (TASA) proposed in [5], [6] based on matching and coloring in graph theory has been the most popular for TSCH networks. In TASA, a matching set free of the duplex collisions is first obtained by a traffic-aware matching method for each time slot, where links in the matching set can be scheduled in the same time slot. Then, such a matching set is divided into several matching-coloring sets by graph coloring to avoid protocol-interference collisions, where links in different coloring sets transmit in different channels. Finally, a deterministic network free of collisions is obtained. TASA has been shown to potentially reach minimum active slots under certain conditions.

We note that TASA considers only a single interface at the sink (or the coordinator, the root of the routing tree), thus resulting in the long-tail problem in the scheduling table. Since the coordinator with a single interface cannot receive signals from multiple children simultaneously, the children can only take turns in certain order to send messages to the sink. The result is that the average delay time is not minimized, which is crucial for some industrial applications. Hence, in this paper we start by introducing multiple interfaces at the sink to enable parallel transmissions to the sink using different channels. It is expected that faster converge speed, shorter schedule length (the number of time slots needed to finish the schedule), and shorter average delay time compared to the original implementation with a single interface can be achieved. While such improvement may come at the cost of additional hardware complexity, modern technology has shown that adding one more radio interface is affordable in many hardware implementations [7].

We also address the problem of cost-balanced routing in this paper. The default routing tree generated from shortest path algorithms, which aim at minimizing the number of hops from every node to the sink, can easily result in an unbalanced tree. In this way, the subtree with the largest number of nodes could turn out to be the bottleneck during scheduling (while nodes at other branches have ended their transmissions). In addition, we have found that the matching and coloring method proposed in TASA results in low channel usage even with a large network size. (For example, TASA uses only

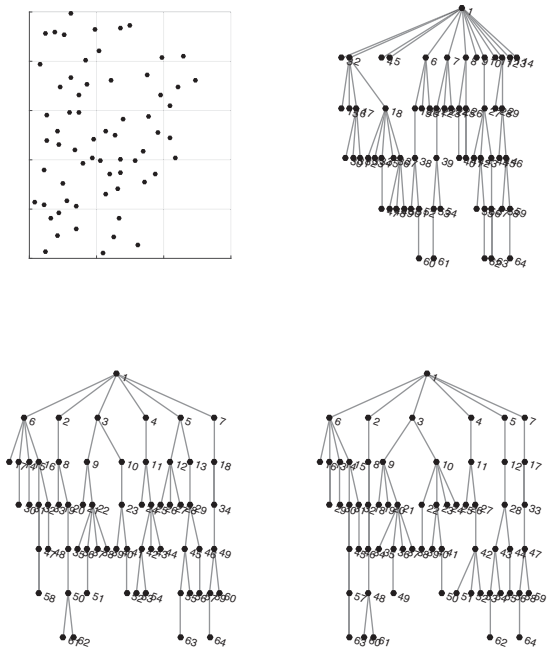


Fig. 1: Different routing trees for the same node placement

4 channels out of 16 channels provided by the standards.) To leverage multiple interfaces, we propose to divide the whole network into multiple subtrees, where nodes in each subtree use a subset of orthogonal channels so that different branches can work simultaneously to leverage spatial reuse. While several papers have proposed the concept of the balance tree [8], [9] to ease the unbalancing problem, they cannot make full use of all channels for general node topologies. The Tree-based Multi-Channel Protocol (TMCP) [10] proposes a disjoint tree to minimize intra-interference, but it suffers from inter-interference in the same subtree for raw-data collection. We propose a better method to build a cost-balanced tree that has been shown to perform better in the target scenario.

The rest of the paper is organized as follows. Section II describes the target scenario and then explains the extended TASA algorithm with support for multiple interfaces at the sink. Section III first introduces the new metric for communication costs, and then describes how we adapt the Load-Balanced Tree (LBT) [8] into the Cost-Balanced Tree (CBT). A refining algorithm to avoid the ping-pong effect during tree building is also proposed in Section III. Finally, Section IV presents our evaluation results and Section V concludes the paper.

II. Traffic Awareness with Multiple Interfaces

In each superframe, each node generates packets to be forwarded to the root (the edge router) with the least average end-to-end delay time. The procedure of collecting data from all nodes to the root is called *convergecast*. For the given node topology, a routing tree G_t can be obtained based on

Algorithm 1 Multi-interface traffic-aware scheduling

1: //G_t: connectivity graph, G_i: physical interference graph
2: //η: completed workload efficiency vector for each node
3: //**m**: vector of elements of the current matching set
4: //q: traffic of node, Q: traffic of a subtree
5: //$Schedule$: schedule table
6: **while** $\eta \leq \mathbf{1}$ **do**
7: $\mathbf{m} \leftarrow Matching(Gt)$
8: $\mathbf{m} \leftarrow \mathbf{m} \cup \{b_i | b_i \in ch(n_0), q(bi) \neq 0, ch(b_i) \notin m\}$
9: $TimeFreqSet \leftarrow Coloring(G_i, \mathbf{m})$
10: $Schedule \leftarrow Schedule \cup TimeFreqSet$
11: $Update(\theta, \mathbf{m}, G_i, q, Q)$
12: **end while**

different metrics. The key issue is to form a routing tree that can fully utilize all channels and each subtree can have similar sizes and communication loads. We consider both the *aggregated-data* and *raw-data* convergecast as proposed in [9]. For aggregated-data convergecast, packets are aggregated at each hop, thus allowing packets to be compressed or summarized with aggregation functions such as MEAN, MAX, MIN, and so on. For raw-data convergecast, data from each sensor is equally important (or the correlation between them is minimal), so packets from children are directly added to the cache of parents without any compression.

We assume that the sink is equipped with $NI \geq 1$ interfaces, and the number of subtrees K is equal to NI. The sink can receive signals from several children via different interfaces operating in different channels simultaneously. Notice that only the sink is assumed to have multiple interfaces, and the rest of the nodes are equipped with only one interface as usual. As a result, at each matching set or time slot, all the top-level nodes that are not empty can be allocated to the same matching set at each time slot, and then they should be allocated to different coloring sets or channels. For other nodes that are not the sink or top-level nodes, they are scheduled as usual. To extend TASA to multiple interfaces, as shown in Alg. 1 we first get a matching set out of the routing tree G_t with a metric of the subtree queue Q, and then expand the set with extra legal links between top-level nodes and the sink. After that, we color the set into several sets using the conflict graph G_i. Once the schedule of the current time slot is determined, nodes are added to the schedule table, and related parameters are updated. The algorithm goes on until all nodes finish their transmissions. The computation complexity is $O(Nlog(N))$, where N is the network size. Notice this algorithm can be applied for both aggregated-data and raw-data convergecast.

III. Cost-Balanced Routing Tree

A. Communication Cost

To start with, we define the communication cost C to reflect the cost to transfer the message from the source to the final destination. Let $c = p * l$, where p is the packet size (in bits) and l is the distance of the path (in hops), and let the overall cost $C = \sum_{n=1}^{N} p_n * l_n$ for all N nodes. Apparently, for nodes

Algorithm 2 Cost-balanced routing tree

Initialization:
$t \leftarrow children(sink)$
$t \leftarrow sort(n, |neighbors(t)|, cache(t))$
$b \leftarrow first\ K\ elements\ of\ t$;
for i from 1 to K **do**
  $B_i \leftarrow b(i)$
end for
$M \leftarrow \{sink, b\}$, $MC \leftarrow N \setminus \{sink, b\}$
while MC *not empty* **do**
  $B_i' \leftarrow minWeight(B)$
  $B_i \leftarrow minFreedom(B_i')$
  $Borders \leftarrow \{neighbors(B_i)\}$
  $n' \leftarrow maxWeight(Borders)$
  $n \leftarrow maxGrowSpace(n')$
  $m \leftarrow minDepth(neigbors(n) \cap M)$
  $Tree(n, m) \leftarrow 1$
  $Update(B_i, M, MC, neighbors(B_i), Weight(B_i))$
end while

with given local traffic, different topologies lead to different communication costs. Generally, a fat tree is likely to have smaller C than a thin tree, and SPT (shortest-path tree) has the least overall communication cost compared to all other routing trees since it has the shortest path length.

B. Cost-Balanced Routing Tree

We show in Alg. 2 the proposed algorithm for building the cost-balanced tree (CBT). Firstly, we set K nodes with the largest number of neighbors (major) and cache size (minor) as the top-level nodes. When the unmarked set MC is not empty, we get the branch with the minimum weight (major) and growing space (minor) as the current branch. The growing space defined in [8] is a measure of the freedom to grow the tree towards this node. Then we search for the candidate node with the maximum weight (major) and growing space (minor) out of the neighborhood of the current branch. Finally, the candidate node connects to its marked parent with the least depth in the current branch. Once a connection is established, the tree graph, branch sets, marked and unmarked sets, neighborhood and weights of branches are updated. The complexity of the algorithm can be calculated to be $O(NlogN)$. We note that the channels used by nodes in the same set are not necessarily adjacent, which is a plus for avoiding possible cross-talk in adjacent channels.

C. Refining the Cost-Balanced Tree

After the CBT is built based on Alg. 2, we propose a refining algorithm for achieving further performance improvement. The reason is that the original algorithm may suffer from the unexpected ping-pong effect, meaning that the same node could repeatedly alter its connection from one branch to another, as shown in Fig. 2. The original algorithm first finds for the largest subtree b_1 the neighbor subtree b_2 with the least weight. It then moves (lends) the common node D_1 with proper weight from b_1 to b_2. However, b_2 may return D_1 back to b_1 by the algorithm in the next run, thus causing the algorithm to get stuck in the ping-pong effect or reach

Algorithm 3 CBT refining algorithm

$Avr \leftarrow mean(Weight(B))$
while $count < max\ iteration\ count$ **do**
  $big \leftarrow \{B | Weight(B) > Avg\}$
  $small \leftarrow \{B | Weight(B) < Avg\}$
  $B_i \leftarrow maxWeight(big)$
  $B_j \leftarrow minWeight(small)$
  $n \leftarrow maxWeight\{B_i \cap B_j\}$
  if n is null **or** n is leaf **then**
    $B_j \leftarrow minWeight(small \setminus B_j)$
  else
    $m \leftarrow neigbors(n) \cap B_j$
  end if
  Start from the smallest branch set similarly
  Update the branch set, weight and the connection in the tree
end while

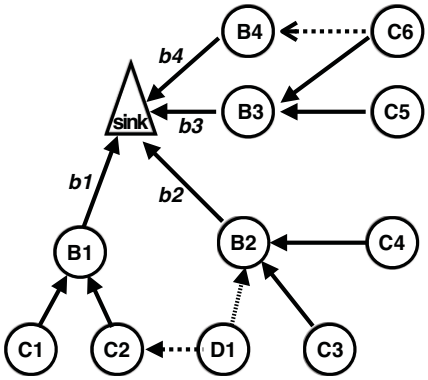


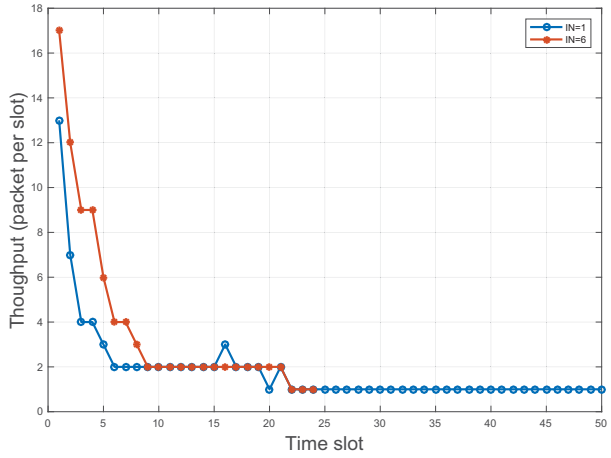
Fig. 2: Ping-pong effect

premature stop. For other branches such as b_4, it may ask for node C_6 from b_3, but the request could be ignored.

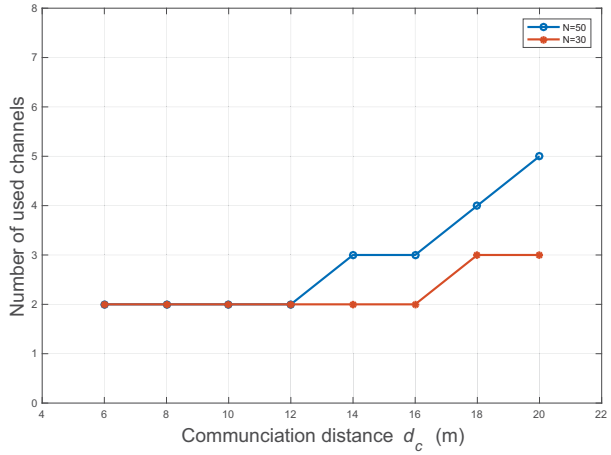
As a refinement, we perform node reconnection in a bidirectional way rather than the original way that starts from the largest branch unidirectionally. That is, we also start from the smallest branch to search for the possibility to borrow a weighted node from bigger neighbor branches. Any newly searched node needs to be compared to the last altered node recorded in a history list; if they are the same, the ping-pong effect is detected and there is no need to reconnect. As described in Alg. 3, when the algorithm has not reached the max iteration count, we get the large and small branch sets first, and then get the branch with the maximum weight and the one with minimum weight. If there is no common node that can be transferred or there is no leaf node, we abandon this branch and choose the minimum branch out of the remaining ones; otherwise, we transfer the node from the larger branch to the smaller branch. Similarly, we also can start from the smallest set to borrow nodes from the proper larger sets. The time complexity of the algorithm is $O(KlogN)$.

IV. Simulation Results

We consider transmissions without packet losses, and set the number of branches $K = 6$ for different routing trees. We randomly place the nodes and build the network topology



(a) Throughput variations



(b) Number of used channels vs. d_c

Fig. 3: Impact of multiple interfaces

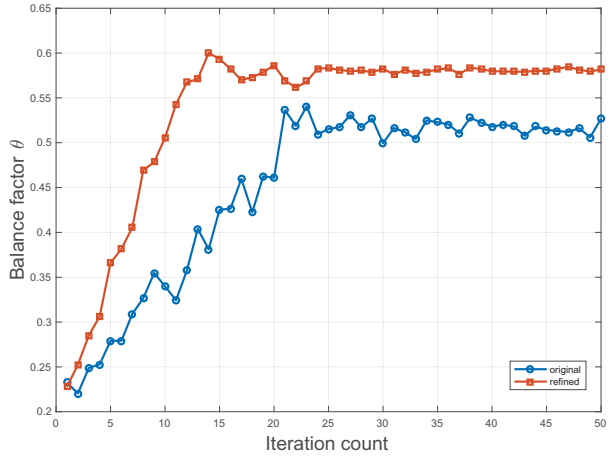


Fig. 4: Performance of the CBT refining algorithm

with two rules: (i) distance between any two nodes should be larger than *distinction distance* d_s, (ii) nodes should be within the *communication distance* d_c of at least one node. The first rule ensures that nodes should not stay too close to each other, while the second one ensures that all nodes are connected to the network and no node is isolated. Notice that d_c is directly decided by the power of the transmitter, and we assume the same transmit power and hence d_c for each node except the root. We set the communication distance d_c of the sink a bit larger than other nodes to get more candidate top-level nodes (which are children of the root) to ensure enough K branches, since the sink is better equipped and line-powered. We define the *coverage distance* $d_v = L/\sqrt{N}$, where L is the length of the square area. Generally, d_s is fixed and d_c is decided by the coverage distance d_v; a small d_c may lead to insufficient coverage and a large d_c brings stronger interference or power consumption, so we set $d_c \approx 0.62 d_v$ by heuristics. Fig. 1 shows an instance of node placement. We implement the algorithms in Matlab, and follow simulation settings from the minimal 6TiSCH configuration [11]. Each data point is the average of 200 simulation runs.

A. Impact of Multiple Interfaces

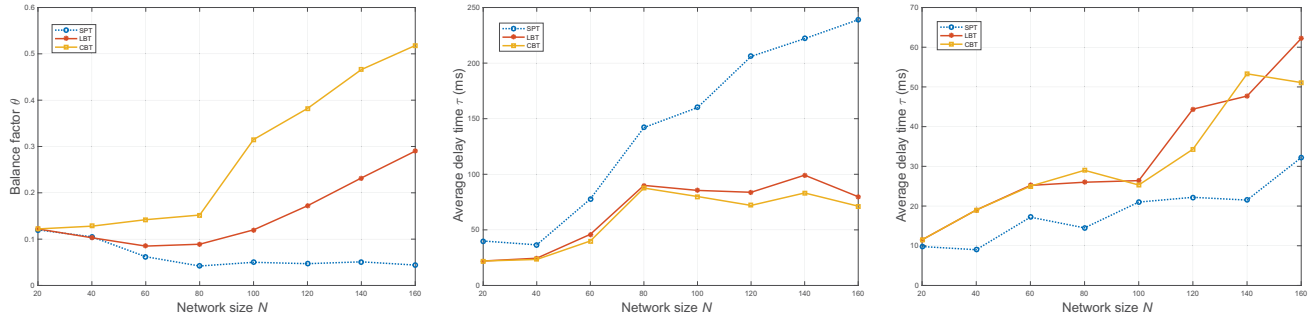
We consider a scenario with $N = 50$, $L = 50$, and $d_c = 8$, and compare the throughputs for $NI = 6$ and $NI = 1$. As we can see from Fig. 3(a), a larger value of NI can achieve higher throughput and finish transmissions before the case for $NI = 1$. The number of slots needed for $NI = 1$ is about twice of that for $NI = 6$. To see the number of used channels when multiple interfaces are used, we set $NI = 6$ and vary d_c from 6 to 20. Fig. 3(b) shows the relationship of d_c and the number of used channels. It can be observed that a larger value of d_c results in a more serious interference problem between nodes, and hence more channels are used. Additionally, it can be observed that as the node density increases, the number of channels needed also increases.

B. Performance of the CBT Refining Algorithm

The refining algorithm proposed in Alg. 3 aims to fix the ping-pong effect observed in the original algorithm. Fig. 4 shows that it can achieve much faster convergence speed and it converges to a better balanced factor than the original algorithm. The *balance factor* θ is defined in [8] to evaluate how the defined weight of each branch differs, namely $\theta = (\sum_{k=1}^{n} W_{bk})^2/(n\sum_{k=1}^{n} W_{bk}^2)$, where W_{bk} is the weight of branch k. A larger value of θ denotes better balance of the routing tree.

C. Performance with Different Routing Trees

Fig. 1 shows a simple example of different routing trees for the same node placement with 64 nodes. Nodes at the upper level of SPT (top-right subfigure) tend to have large degrees, and large branches are mixed with small branches. On the other hand, the branches of CBT and LBT are more balanced. As Fig. 5(a) shows, the difference in the balance factor becomes more significant as the network size gets larger. The reason is that when the network size grows larger, the node density increases with more connections. The benefits of the proposed cost metric are more pronounced in this case,



(a) Balancing factor (b) Average delay of raw-data convergecast (c) Average delay of aggregated-data convergecast

Fig. 5: Performance with different routing trees

resulting in much better performance for CBT compared to LBT and SPT.

1) Raw-Data Convergecast: To calculate the delay time, we set the packet size of each node to be uniformly distributed in the range [1, 10] and set the length of the time slot to be 15ms [11]. Fig. 5(b) shows the relationship of the average delay time and the network size when $L = 150$ and $d_c = 8$. It can be seen that the average delay time of SPT increases much faster than LBT and CBT as the network size grows larger. CBT performs slightly better than LBT, and the gap becomes more significant as the network size increases. On average, CBT can reduce the average delay time to approximately 52% of SPT and 11% of LBT.

2) Aggregated-Data Convergecast: We consider the MIN aggregation rule for aggregated-data convergecast. Since aggregation is introduced, the network load decreases. It can be shown in Fig. 5(c) that with low traffic load SPT can achieve lower time slots and shorter average delay time than LBT and CBT, meaning the SPT can still be beneficial for low-traffic scenarios.

Note that the proposed CBT can contribute to easing the memory burden and energy consumption for top-level nodes. We find that in raw-data convergecast, the peak memory size among all the top-level nodes in the schedule decreases about 73% off from SPT to CBT, and maximum power consumption cuts nearly 47% off in the same way.

V. Conclusions

To reduce the delay time when using the IEEE 802.15.4e TSCH mode for data collection, we address the problems of the long-tail schedule table and unbalance of routing subtrees in TASA. By leveraging multiple interfaces at the sink, we build Cost-Balanced Tree (CBT) to allow more efficient parallel transmissions, such that nodes at different branches can use different subset of channels simultaneously. We have observed that the proposed method can result in lower latency of about 52% off the baseline algorithm in raw-data convergecast.

Acknowledgment

This work was supported in part by funds from the Ministry of Science and Technology, National Taiwan University and MediaTek Inc. under Grants MOST-104-2622-8-002-002, MOST-104-2221-E-002-076-MY2, and NTU-ERP-105R8300.

References

[1] LAN/MAN Standards Committee, "Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)," *IEEE Computer Society*, 2003.

[2] P. Du and G. Roussos, "Adaptive time slotted channel hopping for wireless sensor networks," in *IEEE Computer Science and Electronic Engineering Conference (CEEC)*, pp. 29–34, 2012.

[3] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[4] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler, "Decentralized traffic aware scheduling for multi-hop low power lossy networks in the Internet of things," in *IEEE International Symposium and Workshops on aWorld of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6, 2013.

[5] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE802.15.4e networks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 327–332, 2012.

[6] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On optimal scheduling in duty-cycled industrial IoT applications using IEEE802.15.4e TSCH," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3655–3666, 2013.

[7] Texas Instruments, *SimpleLink ultra-low power wireless MCU for Bluetooth low energy*, Feb 2015. Rev. B.

[8] H. Dai and R. Han, "A node-centric load balancing algorithm for wireless sensor networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 1, pp. 548–552, 2003.

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

[10] Y. Wu, J. A. Stankovic, T. He, and S. Lin, "Realistic and efficient multi-channel communications in wireless sensor networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[11] X. Vilajosana and K. Pister, "Minimal 6TiSCH configuration," *IETF Internet Draft*, Feb 2016.