IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Optimizing Throughput for Distributed Systems in Wireless Sensor Networks

Nandhini.G[*1], Vinod.S[2], N.Kumar[3]

[*1,2,3]Dept of CSE, Vel Tech Multitech Dr.Rangarajan Dr.Sakunthala Engg College, India

nandhinisweety228@gmail.com

Abstract

ZigBee is one of the communication standard designed for low-rate wireless personal area network. It has low complexity, cost, power consumption, inexpensive and portable. Among the well- known ZigBee topology, ZigBee cluster-tree is suitable for low-power, low-cost wireless sensor network. Recent work has shown that ZigBee cluster-tree network cannot able to provide sufficient bandwidth if the traffic load increases. The main objective is to avoid the traffic and deliver the packets to the destination. The Pull Push Re-label algorithm is applied to measure the capacity of packets so that the delivery is performed by next region head. The data can be hacked by intruders so security is provided by encrypting the data.

Keywords: IEEE 802.15.4, ZigBee, Distributed Performance Optimization.

Introduction

RECENT advances in wireless communications and micro-electromechanical technologies have had a strong impact on the development of wireless sensor networks. The IEEE 802.15.4 protocol is a promising standard for WSN applications because it pays particular attention to energy efficiency and communication overheads [1]. Based on the physical (PHY) and medium access control (MAC) layers of IEEE 802.15.4, the upper-layer (including the network and application layers) specifications are defined by the ZigBee protocol stack. The applications supported by ZigBee include home automation schemes, remote control and monitoring systems, and health care devices [3].

Among the well-known ZigBee topologies, the cluster tree is especially suitable for low-power and low-cost WSNs because it supports power saving operations and light-weight routing. In the ZigBee cluster-tree topology, the power saving operation is managed by the IEEE 802.15.4 MAC super frame structure; and a light-weight tree routing protocol is enabled under a distributed address assignment policy configured by several system parameters. The ZigBee cluster-tree network is effective for WSNs. In a tree structure any link failure will suspend data delivery completely and the recovery operation will incur a considerable overhead. The topology also prevents the use of many potential routing paths, which means that a considerable amount of bandwidth cannot be utilized. As a result, the sampling rate of the sensor nodes deployed in the area of interest will be increased, and more traffic will be generated suddenly in the network.

Cuomo et al. [7] demonstrated the benefits of ZigBee tree routing with respect to reactive routing in typical sensor network applications. Khan and Khan [10] conducted a comprehensive performance evaluation of ZigBee cluster tree networks and provided important insights into engineering systems for developers. Koubaa et al. [13] presented an analytical model that derives worst-case end-to-end delay bounds under buffering and bandwidth requirements. Han [9] proposed an algorithm that configures cluster-tree parameters optimally and guarantees all the end-to-end deadlines of periodic real time flows deterministically. Peng et al. [19] introduced a selection strategy for ZigBee tree and mesh routing in various applications; while Kim et al. [11] developed shortcut tree routing to reduce the transmission latency. The common drawback of the existing approaches is that they do not address the poor bandwidth utilization problem in ZigBee cluster-tree networks, so it is difficult to increase the system throughput.

An adoptive-parent-based framework is proposed for a ZigBee cluster-tree network. Our objective is to provide more flexible routing and increase bandwidth utilization without violating the operating principles of the ZigBee cluster-tree protocol. The framework is well suited to networks in which there are sudden requirements for increased bandwidth to deliver additional information. Based on the existing cluster-tree topology, the framework allows a ZigBee node to request bandwidth from adjacent routers (called adoptive parents) as well as from its original parent router. To optimize the throughput in the framework, we model the process as a vertex-constraint maximum flow problem. To solve the problem, we propose a distributed algorithm that is fully compatible with the ZigBee standard. The optimality and convergence property of the algorithm are proved theoretically.

System Architecture

The IEEE standard, 802.15.4, defines the physical layer and medium access control sub-layer for low-rate wireless personal area networks (LR-WPANs) [1]. IEEE 802.15.4 defines a superframe structure that begins by transmitting a beacon issued by a PAN coordinator. The process consists of an active portion and an inactive portion. The coordinator and devices can communicate with each other during the active period and enter a low-power phase during the inactive period. The active portion with 16 time slots is comprised of three parts: a beacon, a contention access period (CAP), and a contention free period (CFP). The beacon is transmitted by the coordinator at the beginning of slot 0, and the CAP follows immediately after the transmission. During the CAP, devices can transmit non time-critical messages and MAC commands. In the CFP, the standard protocol provides a Guaranteed Time Slot (GTS) mechanism that ensures the devices can occupy the time slots exclusively for transmission. For those devices that desire guaranteed time slots in the next super frame’s CFP, they send GTS requests to the coordinator during the current super frame’s CAP. Then, the coordinator checks if there is available bandwidth in the current super frame, and determines, based on an FCFS fashion, a device list for GTS allocation in the next super frame. Finally, the GTS descriptor is included in the subsequent beacon to announce the allocation information. The coordinator provides the initialization, maintenance, and control functions for the network. The router has a forwarding capability to route sensed data to a sink node. The end device lacks such a forwarding capability. ZigBee supports three kinds of network topology, namely, star, cluster-tree, and mesh topologies. In a star network, multiple ZigBee end devices connect directly to the ZigBee coordinator. For cluster-tree and mesh networks, communications can be conducted in a multi-hop fashion through ZigBee routers. In this paper, we assume that sensed data in ZigBee cluster-tree networks is delivered by the GTS mechanism because a high-delivery ratio can be guaranteed [13].

The ZigBee cluster tree provides an effective solution for low-power and low-cost wireless sensor networking, the rigidness of the topology makes it vulnerable to link failures. To resolve such problems, we propose an adoptive-parent-based framework for a ZigBee cluster tree network. The framework provides more flexible routing and increases bandwidth utilization without violating the operating principles of the ZigBee protocol.

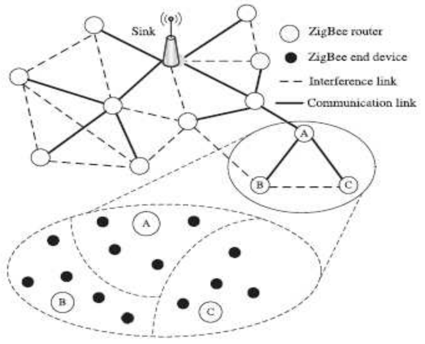


Fig.1.The ZigBee cluster-tree network

Distributed Throughput Optimization In Adoptive-Parent-Based Framework

To realize the concept of adoptive parents in ZigBee cluster-tree networks, we utilize the concept of distributed throughput optimization in an adoptive-parent-based framework. We begin by formulating the throughput maximization problem as a vertex-constraint maximum flow problem and then propose a distributed algorithm to resolve the problem. Finally, we provide a theoretical analysis of the optimality and convergence property of the distributed algorithm.

Problem Formulation

A vertex-constraint flow network can be formulated as a directed graph $G = (V, E)$, where V represents the routers in the network and E represents the possible communication links between pairs of routers. In a traditional flow network, each edge has a nonnegative capacity. By contrast, in a vertex-constraint flow network, each vertex $u \epsilon V$ is associated with a nonnegative capacity, denoted by $c(v) \geq 0$, which represents the GTS capacity of the router. Each directed edge (u, v) is associated with an implicit capacity $c(u, v) = \infty$ if $(u, v) \epsilon E$ otherwise $c(u, v) = 0$. For each flow, two vertices are distinguished in the network: a source s and a sink t, where s is the sender of the data that requires additional bandwidth and t is the data receiver.

A flow in a vertex-constraint flow network G with respect to a source s and a sink t is a real-value function $f: V \times V \rightarrow R$ that satisfies the following three properties:

- *Capacity constraint*:
  $\sum_{u\in V}\{f(u,v)|\ f(u,v)>0\}\le\hat{c}(v),\forall v\in V.$
- *Skew symmetry*:
  $f(u,v)=-f(v,u),\forall u,v\in V.$
- *Flow conservation*:
  $\sum_{u\in V}f(u,v)=0,\forall v\in V-\{s,t\}.$

The quantity f (u, v), which can be positive, zero, or negative, is called the net flow from vertex u to vertex v. The capacity constraint, which relates to a router's physical resource usage, stipulates that the net flow passing through the router must not exceed its capacity. Because of the skew symmetry property, the net flow from one vertex to another vertex is the negative of the net flow in the opposite direction; and because of flow conservation, the total net flow into a vertex, except the source or sink is equal to zero. In the vertex-constraint maximum flow problem, given a vertex-constraint flow network G with source s and sink t, the objective is to find a maximum flow f from s to t in G.

A Distributed Algorithm

As mentioned previously, we propose a distributed algorithm to resolve the vertex-constraint maximum flow problem. Specifically, we revise the push-re-label method [6], which underlies many of the asymptotically fastest algorithms used to solve traditional network-flow problems. The proposed algorithm, called the pull-push-re-label (PPR) algorithm, is designed to adapt to a ZigBee cluster-tree network of a certain scale. In the following, we present our algorithm and demonstrate its optimality and convergence. First, we define some notations and terms. Given a vertex-constraint flow network G = (V, E) with source s and sink t, let f be a flow in G, and let a vertex v ∈ V. The amount of extra flow that can be added to v before exceeding the capacity c (v) is called the residual capacity of v, and is given by

$$\hat{c}_f(v)=\hat{c}(v)-\sum_{u\in V}\{f(u,v)|f(u,v)>0\}.$$

Similarly, for any pair of vertices u, v ∈ V, the residual capacity of (u, v) is the extra flow that can be moved from u to v before exceeding the capacity c (u, v), denoted as

$$c_f(u,v)=c(u,v)-f(u,v).$$

$$E_f=\left\{(u,v)\middle|\begin{array}{ll}\hat{c}_f(v)>0, & \text{if } c(u,v)=\infty,\\ c_f(u,v)>0, & \text{Otherwise.}\end{array}\right\}$$

That is, a residual edge (u, v) is included in E_f if it can admit a positive net flow from u to v. Note that (u, v) may be a residual edge in E_f even if it is not an edge in E. A pre flow is a function f: V × V→ *R* that satisfies the capacity constraint, skew symmetry, and the following relaxation of flow conservation rule:

$$\sum_{u\in V}f(u,v)\ge 0,\forall v\in V-\{s\}.$$

We now explain the rationale behind the algorithm. Throughout the algorithm, the height of the source s is fixed at V, and the height of the sink t is fixed at 0. The height of every other vertex starts at 0 and increases over time. Initially, the vertices are lower than the source, so they tend to pull the flow1 downward from s toward t; however, any "over pulled" flow may be pushed back upward toward s eventually. During the algorithm's execution, if the excess flow of a vertex cannot be pushed or pulled in either direction, the vertex re-labels itself to increase its height. Vertices tend to push the over pulled flow back to the source by increasing their height so that they are above the fixed-height source. Once any over pulled flow has been removed from the vertices, the flow is deemed to be maximum flow.

In a PULL (u, v) operation, a lower vertex u pulls the flow of a higher vertex v downward to itself. The operation can only be implemented if all the following conditions are satisfied:

- v is overflowing, i.e., e(v) > 0;
- there is an edge from v to u in G, i.e., c(u ,v)= ∞;
- the residual capability of u is positive, i.e.,c_f (u)>0
- u is lower than v by 1, i.e., h(v)=h(u)+ 1.

Procedure 1. PULL(u,v)

```
Applicability: $e(v)>0$, $c(v,u)=\infty$, $\hat{c}_f(u)>0$, and
$h(v)=h(u)+1$
Action: $u$ pulls $\delta=\min(e(v),\hat{c}_f(u))$ units of flow from $v$
  1: $f(v,u)\leftarrow f(v,u)+\delta$
  2: $f(u,v)\leftarrow -f(v,u)$
  3: $e(v)\leftarrow e(v)-\delta$
  4: $e(u)\leftarrow e(u)+\delta$
```

```
Procedure 2. PUSH($u$,$v$)
Applicability: $e(u) > 0$, $c(u,v) \neq \infty$, $c_f(u,v) > 0$, and
$h(u) = h(v) + 1$
Action: $u$ pushes $\delta = \min(e(u), c_f(u,v))$ units of flow to $v$
  1: $f(u,v) \leftarrow f(u,v) + \delta$
  2: $f(v,u) \leftarrow -f(u,v)$
  3: $e(u) \leftarrow e(u) - \delta$
  4: $e(v) \leftarrow e(v) + \delta$
```

Note that, because there is no edge from u to v in G, v cannot pull the flow from u; therefore, u pushes the over pulled flow back to v. A RELABEL (u) operation enables a vertex u to increase its height.

```
Procedure 3. RELABEL($u$)
Applicability: $e(u) > 0$, and $(u,v) \in E_f \Rightarrow h(u) \leq h(v), \forall v \in V$
Action: $u$ increases its height to $1 + \min\{h(v)|(u,v) \in E_f\}$
  1: $h(u) \leftarrow 1 + \min\{h(v)|(u,v) \in E_f\}$
```

In other words, an overflowing vertex u needs to be re-labeled if, for every vertex v for which there is residual capacity from u to v, the flow cannot be pushed or pulled from u to v. When RELABEL (u) is applicable, u determines the minimum height. PULL-PUSH-RELABEL (u) is a compound operation in which a vertex u performs the three basic operations consecutively. Initially, u performs a pull operation to pull flows from each of its adjacent vertices if applicable. Then, if u cannot be pulled by any adjacent vertex, it performs a push operation to push any over pulled flow back to each of its adjacent vertices if applicable. After these operations, u may still be overflowing, and there may be residual capability from u to v; however, the flow cannot be pushed or pulled from u to v because of the vertices' heights.

```
Procedure 4. PULL-PUSH-RELABEL($u$)
  1: for all $v \in Adj(u)$ do
  2:   PULL($u,v$)
  3: if $u$ cannot be pulled by any other vertex then
  4:   for all $v \in Adj(u)$ do
  5:     PUSH($u,v$)
  6:   RELABEL($u$)
```

The proposed algorithm applies the compound operation so that the vertices can manipulate the initial preflow until no overflowing vertex exists. Eventually, the flow from the source to the sink will reach the maximum. An initial preflow is created by the following subroutine: INIT(u) is a subroutine whereby every vertex u ∈ V initializes itself so as to create an initial preflow in G. On completion of the initialization step, we should have an initial preflow in which every adjacent vertex of s is filled to its maximum capacity, and none of the other vertices carries any flow. The net flows into and out of s are also updated accordingly. For every other vertex u, the height h(u) and excess flow e(u) are set to 0, and the net flows into and out of the vertex are also set to 0.

The steps of the proposed pull-push-relabel algorithm are shown in Algorithm 1. After a preflow is initialized, if any vertices are overflowing, Algorithm PPR repeatedly applies a process in which all vertices perform, in any order, the PULL-PUSH-RELABEL operation sequentially. The algorithm terminates only when there are no more overflowing vertices.

```
Procedure 5. INIT($u$)
  1: if $u$ is source $s$ then
  2:     $h(u) \leftarrow |V|$
  3:     for all $v \in Adj(u)$ do
  4:         $h(v) \leftarrow 0$
  5:         $e(v) \leftarrow \hat{c}(v)$
  6:         $f(u,v) \leftarrow \hat{c}(v)$
  7:         $f(v,u) \leftarrow -\hat{c}(v)$
  8: else if $u \notin Adj(s)$ then
  9:     $h(u) \leftarrow 0$
 10:     $e(u) \leftarrow 0$
 11:     for all $v \in Adj(u)$ do
 12:         $f(u,v) \leftarrow 0$
 13:         $f(v,u) \leftarrow 0$
```

```
Algorithm 1. PPR
  1: for all $u \in V$ do
  2:     INIT($u$)
  3: while there exists any overflowing vertex do
  4:     for all $u \in V$ do
  5:         PULL-PUSH-RELABEL($u$)
```

Vertices in the same parallel subset have neither direct links nor common neighbors. Consequently, the parallel subsets execute the algorithm sequentially, but the vertices in each parallel subset operate simultaneously so that the algorithm's correctness is ensured. Since the ZigBee architecture implicitly enables the formation of parallel subsets, parallel executions can be achieved seamlessly, and all of the algorithm's operations can be complete without any extra message exchange.

The Properties of Algorithm PPR

In this section, we consider two essential properties of the PPR algorithm, namely, the optimality of throughput maximization and the convergence of the algorithm. Given a vertex-constraint flow network G = (V, E), let f be a preflow in G and let h be any height function for f.

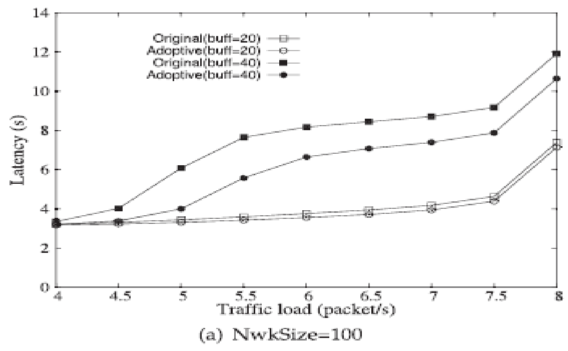
Theorem 1 (Optimality of Algorithm PPR). When Algorithm PPR terminates, the preflow f is a maximum flow from the source s to the sink t in G. To normalize the algorithm's running time, we devise a specific time unit, called a pass, which represents iteration where every vertex performs the PULL-PUSH-RELABEL operation.

Theorem 2 (Convergence of Algorithm PPR). Algorithm PPR always terminates within $2|V^2|$ passes.

Simulations and Numerical Results

In this section, we demonstrate the capability of the proposed adoptive-parent-based framework via a series of simulation experiments. The topology of the cluster tree under investigation is based on the ZigBee specification [3], i.e., it is assumed that each device associates with the parent router at the lowest depth. The tree construction parameters, R_m and C_m, are both set at 5, and Lm is set at 10. The beacon scheduling methodology proposed in [21], which is designed for low latency, is adopted in the experiments. Based on the ZigBee specification, the super frame parameters, BO and SO, are set at 8 and 3, respectively. In the experiments, 100 and 400 ZigBee routers are distributed in a 100 × 100 m^2 area and a 200 × 200 m^2 area, respectively. The ZigBee coordinator is placed in the center of the square and serves as the data sink to collect the sensed information in the network. The transmission power and receiving power threshold are set to achieve a transmission range of 20 m. The region of interest has a traffic load with a mean that ranges from 4-8 packets per second, while the remaining areas have a light average traffic load of 0.01 packets per second.

The following experiments demonstrate the performance improvement achieved by the adoptive-parent-based framework. Specifically, Algorithm PPR is executed in both the original and adoptive-parent approaches to maximize their throughputs. Fig. 2 shows the effect of the traffic load of the cluster of interest on the latency with two different buffer sizes, 20 packets and 40 packets, at the routers. Fig. 2a shows that the proposed framework's latency is lower than that of the original approach especially when the buffer size is 40 packets. This result also indicates that a large buffer causes longer latency, since more packets are delayed. When the network is larger, the performance improvement is much more significant, as shown in Fig. 2b. The performance improvement is as high as 400 percent in this experimental setting. When the traffic load is less than 4.5packets/s, the buffer space is sufficient to hold arriving packets. In this case, the latency increases linearly as the traffic load increases due to the queuing delay.



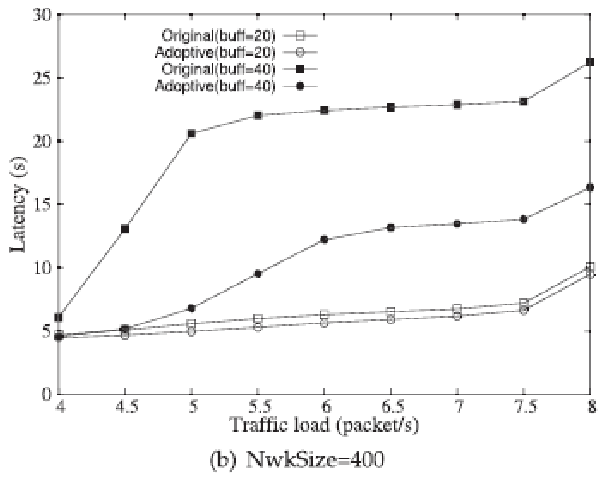
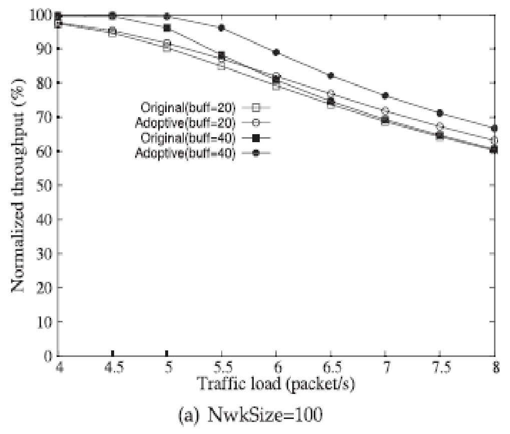


Fig.2. The latency versus various traffic loads: (a) a network of 100 routers; (b) a network of 400 routers

Fig. 3 shows the effect of the traffic load on the throughput for the network settings in Fig. 2. The normalized throughput decreases as the traffic load increases as shown in Fig. 3a. Notice that the increase of the traffic load indeed results in the increase of the achieved throughput and ideal throughput. However, due to queuing packet drops, the gap between the achieved throughput and the ideal throughput increases as the traffic load increases. The improvement achieved by the proposed approach is not significant when the network size is small; however, it is significant when the network is large, as shown in Fig. 3b. For a buffer size of 40 packets, the normalized throughput can reach almost 100 percent under the proposed adoptive parent framework. The proposed framework performs better in a larger network because additional paths are established by more adoptive parents.



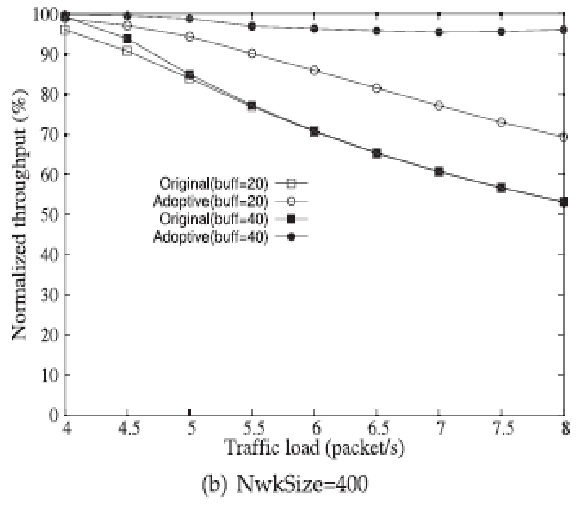


Fig.3. The throughput versus various traffic loads: (a) a network of 100 routers; (b) a network of 400 routers

Conclusion

In a constructed WSN, information about an area of interest may be required for further investigation, which means that more traffic can be generated. However, the restricted routing and poor bandwidth utilization in a ZigBee cluster tree network cannot provide sufficient bandwidth for the increased traffic load, so the additional information cannot be delivered successfully. An adoptive-parent-based framework for a ZigBee cluster tree network is used to increase the bandwidth utilization. Under the framework, a throughput maximization problem, called the vertex-constraint maximum flow problem, is formulated, and a distributed algorithm that is fully compatible with the ZigBee standard is proposed. In this paper, to enhance security while forwarding the packets from one node to another node, the data is encrypted.

References

[1] 802.15.4-2003 IEEE Standard for Information Technology-Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2003.

[2] I.F. Akyildiz, Y. Sankarasubramaniam, W. Su, and E. Cayirci, “A Survey on Sensor Networks,” IEEE Comm. Magazine, vol. 40, no. 8, pp. 102-114, Aug. 2002.

[3] Z. Alliance, “ZigBee Specifications,” 2006.

[4] A. Bhatia and P. Kaushik, “A Cluster Based Minimum Battery Cost AODV Routing Using Multipath Route for ZigBee,” Proc. IEEE Int’l Conf. Networks (ICON), Dec. 2008.

[5] R. Burda and C. Wietfeld, “A Distributed and Autonomous Beacon Scheduling Algorithm for IEEE 802.15.4/ZigBee Networks,” Proc. IEEE Int’l Conf. Mobile Adhoc and Sensor Systems (MASS), Oct. 2007.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms. The MIT Press, 2001.

[7] F. Cuomo, S. Della Luna, U. Monaco, and F. Melodia, “Routing in ZigBee: Benefits from Exploiting the IEEE 802.15.4 Association Tree,” Proc. IEEE Int’l Conf. Comm. (ICC), June 2007.

[8] J. Han, “Global Optimization of ZigBee Parameters for End-to- End Deadline Guarantee of Real-Time Data,” IEEE Sensor J., vol. 9, no. 5, pp. 512-514, May 2009.

[9] S.A. Khan and F.A. Khan, “Performance Analysis of a ZigBee Beacon Enable Cluster Tree Network,” Proc. Int’l Conf. Electrical Eng. (ICEE), Apr. 2009.

[10] T. Kim, D. Kim, N. Park, S. Yoo, and T.S. Lopez, “Shortcut Tree Routing in ZigBee Networks,” Proc. Int’l Symp. Wireless Pervasive Computing (ISWPC), Feb. 2007.

[11] Y.D. Kim and I.Y. Moon, “Improved AODV Routing Protocol for Wireless Sensor Network Based on ZigBee,” Proc. Int’l Conf. Advanced Comm. Technology (ICACT), Feb. 2009.

[12] K.K. Lee, S.H. Kim, and H.S. Park, “Cluster Label-Based ZigBee Routing Protocol with High Scalability,” Proc. Int’l Conf. Systems and Networks Comm. (ICSNC), Aug. 2007.

[13] R. Peng, M. Sun, and Y. Zou, “ZigBee Routing Selection Strategy Based on Data Services and Energy-Balanced ZigBee Routing,” Proc. IEEE Asia-Pacific Conf. Services Computing (APSCC), Dec. 2006.

[14] A. Saeyoung, J. Cho, and S. An, “Slotted Beacon Scheduling Using ZigBee Cskip Mechanism,” Proc. Int’l Conf. Sensor Technologies and Applications, Aug. 2008.