Shortcut Tree Routing in ZigBee Networks

Taehong Kim, Daeyoung Kim, Noseong Park*, Seong-eun Yoo, Tomás Sánchez López
Information and Communications University, Electronics and Telecommunications Research Institute*
{damiano, kimd, seyoo, tomas}@icu.ac.kr, behack@etri.re.kr*

***Abstract*—ZigBee is the emerging industrial standard for ad hoc networks based on IEEE 802.15.4. Due to characteristics such as low data rate, low price, and low power consumption, ZigBee is expected to be used in wireless sensor networks for remote monitoring, home control, and industrial automation. Since one of the most important goals is to reduce the installation and running cost, ZigBee stack is embedded in small and cheap micro-controller units. Since tree routing does not require any routing tables to send the packet to the destination, it can be used in ZigBee end devices that have limited resources. However, tree routing has the problem that the packets follow the tree topology to the destination even if the destination is located nearby. We propose the shortcut tree routing protocol to reduce the routing cost of ZigBee tree routing by using the neighbor table that is originally defined in the ZigBee standard. While following the ZigBee tree routing algorithm, we suggest forwarding the packet to the neighbor node if it can reduce the routing cost to the destination. Simulation results show that the shortcut tree routing algorithm saves more than 30 percent of the hop count compared with ZigBee tree routing.**

***Index Terms*—ZigBee, Tree routing, Neighbor Table**

I. INTRODUCTION

Zigbee is an emerging worldwide standard for wireless personal area network. Under the main goal to provide low-power, cost-effective, flexible, reliable, and scalable wireless products, ZigBee Alliance has been developing and standardizing the ZigBee network. On December 2004, they released the ZigBee Specification version 1.0 [1] to the public. Based on IEEE 802.15.4, ZigBee Specification defines a network layer, application framework as well as security services. Since ZigBee devices are designed for low cost and low data rates, it is expected their use in home and building automation with significantly small costs. Moreover, ZigBee networks support star and mesh topology, self-forming and self-healing as well as more than 65000 address spaces; thus, network can be easily extended in terms of size and coverage area.

Among many useful functions in ZigBee network layer, the tree routing algorithm supports simple but reliable routing for any destination address. In ZigBee, network addresses are assigned using a distributed addressing scheme that is designed to provide every potential parent with a finite sub-block of network addresses. Due to such addressing scheme, the network constructs a tree topology; each device can manage the address space of its descendant. If the destination address is in the address space that a node is managing, the node forwards the packet to one of its child nodes. Otherwise, it forwards the packet to its parent node. The parent or child node which receives the packet selects the next hop node according to the destination address in the same manner.

Tree routing algorithm is thus able to find the next hop node for a given destination address without routing tables. However, a sender can not know if the destination is located nearby or if it's not in the sub-tree which the sender is contained in, since tree routing concerns only about the parent and descendants of the sender node. Although the tree routing is efficient in the view point of memory usage, the routing cost is sometimes inefficient. This paper proposes the shortcut tree routing algorithm to archive both memory efficiency and routing efficiency.

The scheme proposed in this paper improves the ZigBee routing algorithm by employing neighbor tables, which are already part of the existing ZigBee network specification. To overcome the overhead of routing along the tree, we suggest nodes to check their neighbor tables before sending the data to its parent or children. If the table contains a neighbor node that enables reducing the routing cost to the destination, it can be the next hop node for the given destination, instead of the parent or a child node.

This paper is organized as follows. Section II briefly introduces the tree routing scheme and neighbor table in ZigBee, the problem of tree routing is described in Section III. With the proposed algorithm in Section IV, we evaluate the performance and conclude in Section V and VI.

II. ZIGBEE AND IEEE 802.15.4

The ZigBee Alliance is an association of companies working together to enable reliable, cost-effective, low-power, wirelessly networked, monitoring, and control products based on open standards. It is composed of about 200 member companies including 14 promoters such as Motorola, Freescale, Philips, and Samsung. Since their release of the ZigBee Specification version 1.0 on December 2004, a new version was announced on September 2006 including multicast, end device mobility and routing mobility.

A. IEEE 802.15.4

The ZigBee protocol stack is described in Fig. 1. As we can see, the IEEE 802.15.4 and the ZigBee network are tightly coupled to provide the consumer standardization for low-power and low-rate wireless communication devices. IEEE 802.15.4 PHY layer provides 16 channels for ISM 2.4 GHz, 10 channels for ISM 900 MHz, and 1 channel for 868 MHz IEEE 802.15.4 PHY provides LQI (Link Quality Indicator) in order to characterize the quality of links between nodes, as well as data transmission and reception.

IEEE 802.15.4 MAC uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism for accessing the channel, like other wireless networks such as IEEE 802.11 and IEEE 802.15.3. There are two variations: Beacon Enabled Network which uses the Slotted CSMA-CA and Non Beacon Enabled Network which uses the Unslotted CSMA-CA. Moreover, it provides the GTS (Guaranteed Time Slots) allocation method in order to provide real time data communication.

The device types supported by IEEE 802.15.4 and ZigBee are FFD (Full Function Device) and RFD (Reduced Function Device). FFD can communicate with both FFD and RFD, and it can be the PAN Coordinator, Router, and End Device. RFD can only communicate with FFD, so it can be only End Device. Therefore, RFD requires relatively small resources including memory size.

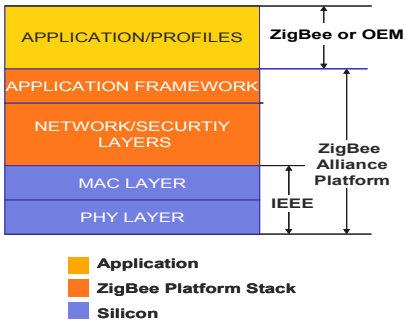

Fig. 1. ZigBee protocol stack

B. *ZigBee network*

Based on IEEE 802.15.4 PHY/MAC, the ZigBee network layer provides functionality such as dynamic network formation, addressing, routing, and discovering 1 hop neighbors. The size of the network address is 16 bits, so ZigBee is capable to accept about 65535 devices in a network, and the network address is assigned in a hierarchical tree structure. ZigBee provides not only star topology, but also mesh topology. Since any device can communicate with other devices except the PAN Coordinator, the network has high scalability and flexibility. Besides, the self-formation and self-healing features makes ZigBee more attractive. The deployed ZigBee devices automatically construct the network, and then changes such as joining/leaving of devices are automatically reflected in the network configuration.

The routing protocols that ZigBee provides are tree routing and table-driven routing. Tree routing is based on the block address allocation mechanism, called Cskip, so each device has an address spaces to distribute to their children. When a device has no capability of routing table and route discovery table, it simply follows the hierarchical tree by comparing the destination address. The most significant benefit of tree routing is its simplicity and limited use of resources. Therefore, any device with low resources can participate in any ZigBee compliant network. On the other hand, table-driven routing is basically similar to the Ad hoc On-demand Distance Vector (AODV) routing protocol for general multi-hop ad hoc network. Whereas tree routing is very simple and inefficient, the table-driven routing provides optimal routes to the destination.

1) Tree routing algorithm

Every potential parent is provided with a finite sub-block of the address space, which is used to assign network addresses to its children. Given nwkmaxChildren (*Cm*), nwkcMaxDepth (*Lm*), and nwkmaxRouters (*Rm*), we can compute the function *Cskip(d)* as the size of the address sub-block distributed by each parent at depth d as follows:

$$Cskip(d) = \frac{1 + Cm - Rm - Cm \cdot Rm^{Lm-d-1}}{1 - Rm}$$

For example, the k[th] router and n[th] end device shall be assigned the network address by their parent at depth *d* as in the following equation.

$$A_k = A_{parent} + Cskip(d) \cdot (k-1) + 1 \ (1 \le k \le Rm)$$

$$A_n = A_{parent} + Cskip(d) \cdot Rm + n \ (1 \le n \le Cm - Rm)$$

A k[th] router that has positive *Cskip(d)* can distribute address spaces to its child nodes. Since every device in the network is a descendant of the ZigBee coordinator and no device in the network is the descendant of any ZigBee end device, any device with address A at depth d has the destination device with address D if the following equation is satisfied.

$$A < D < A + Cskip(d-1)$$

In tree routing, if the destination is a descendant, the device sends the data to one of its children; otherwise, it sends to its parent.

2) Neighbor table

Each device in ZigBee maintains a neighbor table which has all the neighbor information in the 1-hop transmission range. If users limit the size of the neighbor table, the selected numbers of neighbor entries are stored in the table. The contents for a neighbor entry are the network's PAN identifier, node's extended address, network address, device type and relationship. Optionally, additional information such as beacon order, depth or permit joining can be included.

Entries in the table are created when the node joins to an existing network. When a joining node requests a NLME-NETWORK-DISCOVERY, it receives response beacons from already joined nodes. The newly joined node stores neighbors' information from the information contained in beacon packets. Conversely, the neighbor entry is removed when the neighbor node leaves the network. Nodes can know this fact by receiving NLME-LEAVE.indication messages. Since the information on the neighbor table is updated every time a device receives any frame from the some neighbor node, the information of the neighbor table can be said to be up-to-date all the time.

III. Problem Description

The tree routing protocol uses only parent and child relationship for routing, ignoring neighbor nodes. As a result, packets may be routed through several hops towards the destination even if this is within sender's 1-hop transmission range. Fig. 2 shows an example of the described problem.

In Fig. 2, the packet from the source node goes up to the root node following the parent node, and goes back to the destination. In this way, 4 hops are required to reach the destination. However, if the source node sends the packet directly to the destination, it needs only 1 hop routing cost.

In many cases, the routing overhead of tree routing algorithm can not be avoided if only parent-child relationships are considered in the routing. In order to overcome such problem, each node should consider its neighbor nodes as next hop nodes.

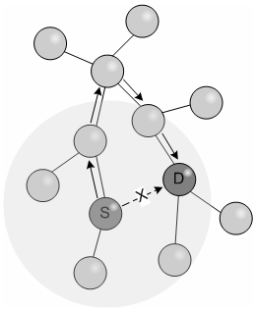

Fig. 2. Problem of Tree Routing

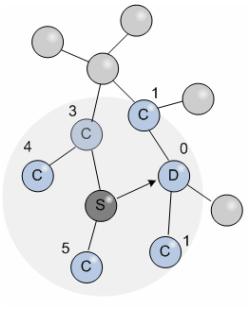

Fig. 3. Candidates for Next Hop

IV. SHORTCUT TREE ROUTING ALGORITHM

We propose the *shortcut tree routing* algorithm that improves existing ZigBee tree routing by using the neighbor table. In other words, the proposed algorithm basically follows ZigBee tree routing algorithm, but chooses neighbor nodes as next hop nodes if the routing cost to the destination can be reduced. The neighbor table that we use in the proposed algorithm is defined in the ZigBee specification, so we don't need to make an effort to search neighbor list. In order to choose the next hop node that can reduce the routing cost, the remaining hop count from the next hop node to the destination is computed for all the neighbor nodes including parent and children nodes. As Fig. 3 shows, the remaining hops to the destination for each neighbor can be computed assuming that the route from the neighbor to the destination goes along the tree. In the above Fig. 3, the route cost can be minimized if the sender transmits the data directly to the destination.

Find_NextHopAddr() function described on table 1 is the algorithm for an intermediate or source node to select the next hop node which has the minimum remaining hop count for the given destination. Because the proposed algorithm follows fundamentally the ZigBee tree routing, the parent or child node is selected as the next hop node in lines 2-3. In addition, the remaining routing cost when we follow ZigBee tree routing is stored into *minNHRouteCost*.

In line 4-13, intermediate or source nodes check the remaining routing cost *myRouteCost* when selecting a neighbor node as the next hop node. The remaining routing cost is calculated based on the remaining hop count to the destination assuming that the packet goes along the ZigBee tree routing. In order to calculate the remaining hop count, the hierarchical address structure is used.

TABLE 1 ALGORITHM TO CHOOSE NEXT HOP NODE FOR THE GIVEN DSTADDR

Find_NextHopAddr(dstAddr)

```
Input: dstAddr
Output: NHDstAddr
begin
1. depth_dstAddr = Find_AddrRange(dstAddr, 0, 0)
2. Assign the next hop node of tree routing to NHDstAddr
3. Assign the remaining hop count when selecting NHDstAddr
   to minNHRouteCost
4. for each (neighbor n_k in neighbor table)
5.    i = 0
6.    while (n_k is in AddrRange[i+1] && i < depth_dstAddr)
7.       ++i
8.    myRouteCost = (depth_dstAddr – i) + (depth(n_k) – i)
9.    if (minNHRouteCost > myRouteCost)
10.      NHDstAddr = n_k
11.      minNHRouteCost = myRouteCost
12.   end if
13.end for each
End
```

By comparing whether the address of a neighbor node is contained in the address space that contains the destination address in each level (*AddrRange[]*), we can find the root of the common sub-tree that contains both the neighbor node and the destination in line 5-7. Among several common sub-trees, the root of the highest level common sub-tree can be the reference point for the calculations as in Fig. 4. The dotted node is the root of the highest level common sub-tree, and the number besides it indicates its tree level. Based on this reference level, we can calculate the remaining hop count using the equation *(level of source node – highest level of on sub-tree) + (level of destination node – highest level of on sub-tree)*. Since the route path goes up to the parent which contains the destination and goes down to the destination in the tree routing, the proposed algorithm computes the route cost in the same way the tree routing does.

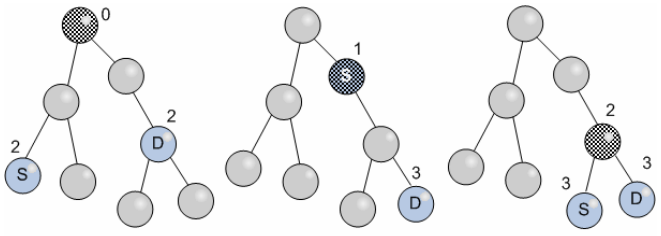

Fig. 4. Calculation of Route Cost when following ZigBee tree routing

If myRouteCost is less than the existing *minNHRouteCost*, the next hop node *NHDstAddr* is replaced with that neighbor node and *minNHRouteCost* is also changed to *myRouteCost*. Therefore, we can find the next hop node that has the minimum remaining routing cost among all the neighbors, including parent and children nodes. If there is no neighbor node that has smaller remaining hop count than the parent or some child node, the next hop node is determined by the regular ZigBee tree routing.

TABLE 2 ALGORITHM TO FIND ADDRESS RANGE OF DESTINATION

Find_AddrRange(dstAddr, startAddr, curDepth)

```
Input: dstAddr, startAddr, curDepth
Output: depth_dstAddr, AddrRange[depth_dstAddr]
begin
 1.if (dstAddr = startAddr)
 2.   return curDepth
 3.else
 4.   for i = 1 to Rm
 5.     if (dstAddr is in the address space of i^th router)
 6.       store address space of i^th router to AddrRange[curDepth+1]
 7.       return Find_AddrRange(dstAddr, i^th router, curDepth+1)
 8.     end if
 9.   end for
10.   if (Cm-Rm > 0)
11.     if (dstAddr is the end device of startAddr)
12.       store dstAddr to AddrRange[curDepth+1]
13.       return curDepth+1
14.     end if
15.   end if
16.end if
end
```

In order to calculate the remaining hops in table 1, we need to compute the address space in which the destination address is contained at each level of the tree together with the depth of the destination. The *Find_AddrRange (dstAddr, startAddr, curDepth)* function in table 2 is the algorithm to get *AddrRange[]* and depth_dstAddr. The address space of destination *AddrRange[]* can be obtained by finding its ancestor nodes in each level and calculating the address space according to the ZigBee's address assignment scheme.

The *Find_AddrRange()* is a recursive function that has the arguments *startAddr*, *curDepth*, and *dstAddr*. A *startAddr* is the address of the ancestor node at *curDepth* for the given destination *dstAddr*. This function is started with *startAddr 0* and *curDepth 0* by calling from the *Find_NextHopAddr()* function, and returns the address space in which the destination address is contained at each depth, *AddrRange[]*, and the depth of the destination, *depth_dstAddr*.

Although the next hop is selected based on the local minimum in the shortcut tree routing algorithm, loops never occur because the remaining hops are computed based on the tree routing. For instance, the route to the destination from the parent or child of a node that received a packet from a certain node *v* has always smaller remaining hops than from the node *v*.

V. PERFORMANCE EVALUATION

A. Time Complexity

Theorem1. The proposed algorithm can be solved in polynomial time.

Let *n* be the number of neighbor nodes of a certain intermediate or source node. The time complexity of the proposed algorithm can be found out by invoking the *Find_NextHopAddr()* function. This function calls *Find_AddrRange()* once to find the address space in which the destination address is contained, and calls it *n* times to find the highest level common sub-tree for each neighbor node (lines 5-7).

Since *Find_AddrRange()* function is recursively called at most *Lm* times and it takes a maximum of *Rm* times within a function to find the address space at each level, the total number of calls is at most $Lm \cdot Rm$ times. The time complexity for finding *AddrRange[]* can be ignored since Lm and Rm are constant. For calculating the remaining hop count for each node, the while loop is repeated at most *Lm* times. This procedure is performed n times for all the neighbor nodes of an intermediate or source node; Thus, the total number of calls are at most $Lm \cdot n$. Therefore, the time complexity of the shortcut tree routing algorithm is $O(Lm \cdot n)$.

B. Upper Boundary of the Routing Cost

Theorem2. The routing cost to the destination does not exceed that of the tree routing.

Every intermediate or source node *v* basically selects the next hop node following ZigBee tree routing, and selects the neighbor node as the next hop node only if the routing cost is smaller than that of the ZigBee tree routing.

In the ZigBee tree routing, the parent node *p* is selected as the next hop node if the destination address is not included in the address space of node *v*. Conversely, when the destination address is in the address space of node *v*, the next hop node is determined as the child node *c* since the destination address is one of the descendants of node *v*.

Since the proposed algorithm chooses the next hop node according to ZigBee tree routing, the selected next hop node has always smaller remaining routing cost than any other parent or children of node *v*. In order to replace the next hop node, a neighbor node should not be either its parent or child and should have a smaller routing cost to the destination than the originally selected by ZigBee tree routing. Note that even if the size of neighbor table is limited, the proposed algorithm will never select a node with higher routing cost than the one selected by ZigBee tree routing. Therefore, the proposed algorithm can not exceed the routing cost of ZigBee routing

C. Simulation Result

We evaluate the performance improvement of our modification by comparing the routing cost of ZigBee tree routing and the proposed *shortcut tree routing*. In the simulation environment, we set the network size as 100x100 and the transmission range as 20 meters. Every node has identical transmission range and they are randomly deployed. Thus, the network topology in the simulation is always variable, and it may occur for several nodes not to be able to join the network if there's no neighbor node within the transmission range. In our simulations, we only consider scenarios where more than 80 percent of the total number of nodes is able to join the network. In order to keep ZigBee's network formation and discovery procedures, we limit the number of children and maximum depth of the tree by setting *Cm*=4, *Rm*=4, and *Lm*=5.

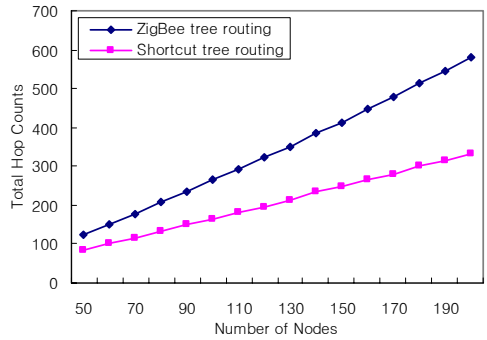

Fig.5. Average total hop count of all nodes when the destination is coordinator and the number of *MaxNeighbor* is 5

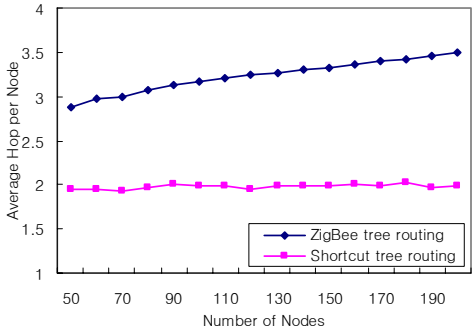

Fig. 6. Average total hop count per a node when the destination is coordinator and the number of *MaxNeighbor* is 5

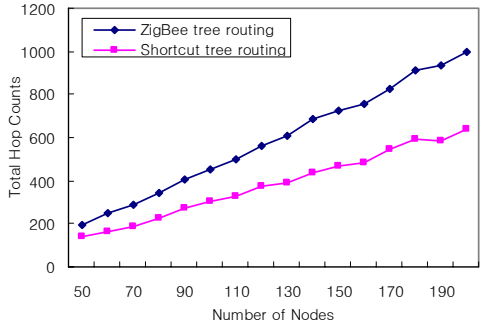

Fig. 7. Average total hop count of all nodes when the destination is random and the number of *MaxNeighbor* is 5

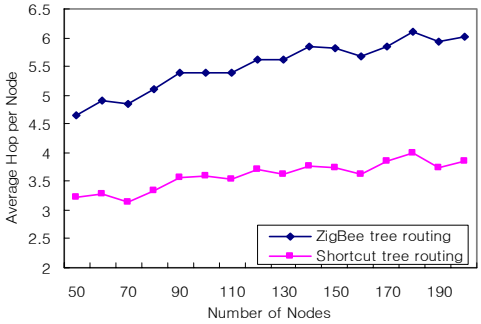

Fig. 8. Average total hop count per a node when the destination is random and the number of *MaxNeighbor* is 5

In the ZigBee standard, it is allowed to restrict the size of the neighbor table. However, the management policy for neighbor tables is delegated to developers. In this paper, we use the policy that, when the maximum number of neighbors is reached, a neighbor entry is updated whenever a neighbor node in a smaller depth is discovered. Moreover, for a finite size of the neighbor table, we only store pure neighbor node's entries (i.e. excluding parent and children nodes)

Fig. 5 and 6 compares the routing cost of ZigBee tree routing and our shortcut tree routing under the condition that the destination of all nodes is the ZigBee coordinator and that the maximum number of neighbor nodes *MaxNeighbor* is 5. The performance evaluation of both algorithms is performed in the same network topology. Fig. 5 measures the total hop count when all the nodes send the data to the coordinator as the number of nodes grows. As the number of nodes increase, the total hop count also increases. However, the total hop count of the shortcut tree routing is much smaller than that of ZigBee tree routing. As Fig. 6 shows, the average hop count per node in the shortcut tree routing is about 2 hops, whereas the average hop count per node in the ZigBee tree routing is about 3-4 hops. Moreover, the average hop count per node in the ZigBee tree routing increases as the number of nodes in the network increases; the shortcut tree routing, however, is not affected by the number of nodes in the network. This is because both, nodes in higher tree levels and neighbor nodes in lower tree levels than that of the parent increase as the number of nodes in the network increase. If the routing follows the regular ZigBee tree hierarchy routing, the hop count to the coordinator in level 0 increases as the node's tree level becomes higher. In our algorithm, however, if there is a neighbor node that has smaller tree level than the parent node, the source or intermediate node selects the neighbor node instead of its parent as the next hop node. Thus, the average hop count per node in the shortcut tree routing is almost constant even if the number of nodes in the network increases.

In Fig. 7 and 8, we set the *MaxNeighbor* to 5 and randomly select the destination of all nodes in each test. Because the destination is different in each test, we measure the average total hop count and the average hop count per node after repeating the test over 50 times. Usually, the routing cost to the random destination is higher than when the destination is the coordinator, because the packet goes up to the node that has the destination as child node and goes down to the destination in the tree routing. Thus, the total hop count when the destination is random (Fig. 7) is as much as twice as when the destination is the coordinator (Fig. 5). The average hop count per node using the regular ZigBee tree routing (Fig. 8) also increases from 4.5 hops to 6 hops as the number of nodes increases. However, in the shortcut tree routing, it only increases from 3 to 3.5 hops.

The performance improvement of the shortcut tree routing is better when the destination is random. The difference on the average hops per node between the shortcut tree routing and ZigBee tree routing, when the destination is selected at random, is about 1.5-2 hops, while the difference under the condition that the coordinator is the destination is about 1-1.5 hops. The reason for this is that the reduction on the routing

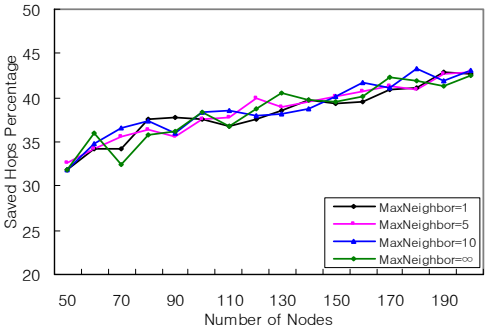

Fig. 9. Percentage of saved hops of proposed scheme according to number of *MaxNeighbor* (destination is coordinator)

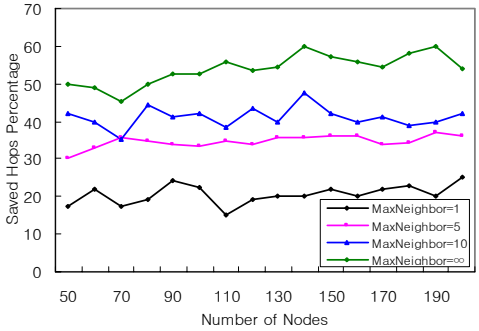

Fig. 10. Percentage of saved hops of proposed scheme according to number of *MaxNeighbor* (destination is random)

cost using the shortcut tree routing is more effective when the destination is random. For example, the routing cost for a packet to go up to some point and then go down to the destination is higher than the cost of that packet to just go up to the coordinator, that is, the root of the tree. Thus the number of hops we save when the destination is random is higher than when the destination is the coordinator.

Fig. 9 and 10 measure the percentage of saved hops when using the shortcut tree routing and varying the *MaxNeighbor* from 1, 5, 10, to infinite. Fig. 9 shows the percentage of the reduction in the hop count when the destination is the coordinator. The reason that the percentage of saved hops for every case is almost the same is the management policy for the neighbor table. In other words, the next hop node selected in the proposed shortcut tree routing is always the same in every condition. Because the next hop node should be the neighbor node that has the smallest tree level in the neighbor table for the given destination (the coordinator), the node that has the lowest tree level is always included in the neighbor table because the management policy prefers nodes in lower tree levels.

In Fig. 10, since the destination is not the coordinator but a random node, the possibility for an intermediate or source node to select a neighbor node as the next hop is quite high despite the management policy for the neighbor table. Thus, the number of neighbor node is an important factor for the performance improvement. For example, the percentage of saved hops increases from 20 percent at *MaxNeighbor* 1 to 50 percent at infinite *MaxNeighbor*. It is notable that we can save about 30-40 percent of the routing cost with only 5 to 10 neighbor nodes, compared with the 50 percent saved hops with an infinite number of neighbor nodes.

VI. Conclusion

This paper introduces the problem of ZigBee tree routing and proposes a shortcut tree routing protocol that overcomes the overhead occurred when following the tree topology. In the proposed algorithm, the neighbor table that is originally defined in the ZigBee standard is used to find the optimal next hop node that has the smallest remaining hop count to the destination. The shortcut tree routing algorithm is efficient in terms of both routing performance and time complexity: it reduces significantly the required routing costs and it can be solved within polynomial time even when the number of neighbors is not limited.

As the performance evaluation shows, the shortcut tree routing reduces more than 30 percent of the routing cost needed for the regular ZigBee tree routing. If the destination is the coordinator, the proposed algorithm shows as good performance as ZigBee's table driven routing, since intermediate or source nodes select the node that has the smallest tree level within its transmission range. For a random destination, the performance of the proposed algorithm depends on how many useful neighbor nodes are stored in the neighbor table. It can be optimized by applying a proper maximum number of nodes and management policy for the neighbor table according to the network's applications. Therefore, we expect the proposed algorithm to be utilized in many ZigBee applications requiring both small memory capacity and high routing performance.

References

[1] ZigBee Alliance, Network Layer Specification 1.0, Dec. 2004.
[2] ZigBee Alliance, www.ZigBee.org
[3] "Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE Std 802.15.4-2003, IEEE Computer Society, 01 October 2003.
[4] G. Ding, Z. Sahinoglu, P. Orlik, J. Zhang, and B. Bhargava, "Efficient and Reliable Broadcast in ZigBee Networks," IEEE Transaction on Mobile Computing, IEEE SECON'05.
[5] D. Kim, Y. Doh, S. Yoo, K. Chang, W. Park, C. Seo, "Low Rate WPAN Technologies and Standards," KISS Information and Communications Journal, Dec. 2004.
[6] J. Kim, H. Lee, D. Hwang, B. Kim, "Development Trend of Standards for Low-Rate, Low-cost, and Low-Power Wireless PAN," Vol. 18, No. 2, pp. 37, ETRI trends, 2003.
[7] E. Callaway, P. Gorday, L. Hester, J. A. Gutierrez, and M. Naeve, "Home networking with IEEE 802.15.4: A developing standard for low-rate wireless personal area networks," IEEE Communications Magazine, page 70-77, August 2002.
[8] Johan Lönn and Jonas Olsson, "Zigbee for wireless networking," Master's Thesis: LITH-ITN-ED-EX--05/015--SE, Linköping University, 2005.
[9] Andreas Andersson and Mattias Thoren, " Zigbee, A suitable base for embedded wireless development," Chalmers technology report, 2005.
[10] Nia-Chiang Liang, Ping-Chieh Chen, Tony Sun, Guang Yang, Ling-Jyh Chen, and Mario Gerla, "Impact of Node Heterogeneity in ZigBee Mesh Network Routing," 2006 IEEE International Conference on Systems, Man, and Cybernetics (SMC'06).