International Journal of Fuzzy Logic and Intelligent Systems, vol. 8, no. 2, June 2008 pp. 132-139

Scatternet Formation Algorithm based on Relative Neighborhood Graph

Chungho Cho*, Dongcheul Son and Chang Suk Kim*****

***Department of Information Communication Engineering**
Gwangju University, 592-1 Jinwol-Dong, Nam-Gu, Gwangju, Korea, 503-703
****Division of Information & Communication Engineering**
Baekseok University, 115 Anseo-Dong, Chonan, Korea, 330-704
*****Department of Computer Education**
Kongju National University, 182 Shingwan-Dong, Gongju, Korea, 314-701
Email: chcho@gwangju.ac.kr, dcson@bu.ac.kr, csk@kongju.ac.kr

Abstract

This paper proposes a scatternet topology formation, self-healing, and self-routing path optimization algorithm based on Relative Neighborhood Graph. The performance of the algorithm using ns-2 and extensible Bluetooth simulator called blueware shows that even though RNG-FHR does not have superior performance, it is simpler and easier to implement in deploying the Ad-Hoc network in the distributed dynamic environments due to the exchange of fewer messages and the only dependency on local information. We realize that our proposed algorithm is more practicable in a reasonable size network than in a large scale.

Key words : Bluetooth, Piconet, Scatternet, RNG, Blueware

1. Introduction

Bluetooth is a standard for short range, low-power wireless communication. It's MAC protocol has the function of constructing an ad hoc network without manual configuration or wired infrastructure. Bluetooth communication is based on TDD(Time Division Duplex) master-slave mechanism. A piconet is a group of nodes in which a master node controls the transmission of other slave nodes, which has the constraint that a piconet consists of one master and up to seven slaves. The basic piconet communication is done by alternating transmission slots, each odd slot of which being used by the slave, and each previous even slot of the odd slot is used by the master. Frequency hopping is used for multiple concurrent communication within radio range of each different piconet without performance degradation due to interference, which makes it possible high densities of communication devices to co-exit and independently communicate with other piconets, resulting in making the possibility of interworking multiple piconets. Therefore, the frequency hopping principle could also create the new network concept, called scatternet, which is an ad-hoc network consisting of overlapping piconets. Self-healing in scatternet is to guarantee the join of new nodes and the removal of existing nodes due to mobility or failure/deactivation of nodes in dynamic environment. Routing path optimization is optimizing routing path to meet the requirements of the performance metrics such as overall network capacity.

In this paper, we present a practical scatternet formation, self-healing and optimized routing algorithm based on RNG algorithm, called RNG-FHR, which is much simpler, less message exchanging and more practical for implementation. Then, we evaluated the performance to show that it is practically possible for deploying in small scale distributed dynamic environments.

2. Preliminary Studies for Bluetooth Scatternet Formation Algorithms

We can describe some characteristic classification criteria for Bluetooth Scatternet Formation Algorithms[13]. The first one is a connectivity guarantee which means that each node is aware of all its neighbors within a communication range. The second one is a node degree which means the algorithms will be divided into those that guarantee the degree limitation for each created piconet or not. The third one is hop based criteria. The algorithms can be divided into those that are designed for the single-hop range and multi-hop range. In single-hop range each device is within communication range of any other device in the network, and it operates only when nodes are in radio visibility, that is, each device is within the radio transmission range of any other device by one hop, e.g., electronic devices in a laboratory, or laptops in a conference room[3][8]. However, in multi-hop range, some devices are not within transmission range of each other instead of being connected via

Manuscript received Apr. 1, 2008; revised Jun. 5, 2008.
This Study was conducted by research funds from Gwangju University in 2008.

other devices, and they can operate even though some pairs of nodescan't directly communicate with each other. The communication between them is done by passing the messages through the intermediate nodes[7][11][12].

The last one is knowledge degree of neighbor devices. The algorithms can be divided into those that each device is required to learn all its neighbors and learn some of its neighbors.

The link formation process in Bluetooth Scatternet consists of two processes: Inquiry and Page. The goal of the Inquiry process is for a master node to discover the existence of neighboring devices and to collect low level information which is primarily related to their native clocks. The goal of the Page process is to use the gathered information in Inquiry process to establish a bi-directional frequency hopping communication channel[3].

It produces connected scatternets by exploiting clustering schemes for ad hoc networks, which is primarily aimed at home and office environments. In multihop solutions, it requires the exact position information because they can influence on the formation of scatternet topology, each being composed of short range devices[9][10].

None of the previous schemes deals with RNG based small-scale dynamic environments where nodes may arbitrarily join or leave the network. However, our approach has the following characteristics[6].

o Nodes can arrive and depart at any time
o The topology can be healed if the node/link failure occurs
o The routing path can be optimized with the minimum cost

3. RNG Algorithm Concept

The key concept of RNG is adding links as and when they are discovered. Let |AB| denote the Euclidean distance between nodes A and B. RNG adds a link between two nodes,A and B in the logical topology if and only if A and B are in each others transmission range and |AB| <= max(|BC|, |AC|) for any other node C which is in A's and B's transmission ranges. After RNG has added AB, if a node C that violates the above condition is discovered, then RNG deletes AB.

The RNG has the following assumptions and characteristics.

- Each node is assumed to know its neighbors in the physical topology graph
- A node also knows an ordering among the Euclidean distances between its neighbors from power measurements and subsequent information exchange with its neighbors.
- Addition and/or deletion of a link do not affect any other link additions or deletions, and depend only on local information.
- There is no need to broadcast any information throughout. Thus, RNG exchanges fewer messages and is simpler than the other algorithms such as distributed MST[6].

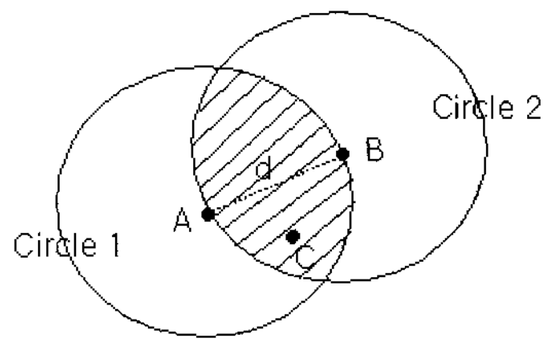


Fig 1. As RNG algorithm, link AB is added if there is no other node in the shaded area. After link AB has been added, the link is deleted if node C is in the area.

4. RNG based Scatternet Formation Algorithm

We assume that node 0 is controller which is interfaced to a bus and controls the communication between the other nodes. The other nodes(node1 to node 5) are master or slave node.

Phase 1 : Initiate each node

Piconet_No = [my_bd_addr], Node_Type = Free_Node.

If a node is controller, then Hops_to_Controller =0 else Hops_to_Controller = 1000.

Neighbor_Table(NT) = empty vector.

Format of NT is <Bd_addr, Sig_strength, Node_type, Roles, Ch_id, Piconet_no, Hops_to_controller>.

Bd_addr is the identifier of Bluetooth device.

Sig_strength is the signal strength from the the neighbor node. Role is defined as one of As_Master/As_Slave/ Not_Connected. Ch_id is assigned to each link toward neighbor nodes if a node is connected to its neighbors. Otherwise, not assigned(Not_Assigned).

Neighbor_Piconet_Table(NPT) = empty vector.

Format of NPT is <Bridge_id, Remote_piconet_no>.

Bridge_id is the identifier of the device joining more than one piconet.

Remote_piconet_no is the identifier of the neighbor combined with the local piconet .

Phase2 : Each node discovers neighbors and makes a physical link connection to one of the neighbors in baseband layer(or physical layer) according to Bluetooth specification.

a) Node A receives inquiry response from node C, meanwhile C creates a new entry for A in its NT(Neighbor_Table) based on the inquiry
b) Node A creates a new entry for C in its NT
c) Node A makes a physical link connection to C

Phase 3 : Once a physical link connection A(master node) -> C(slave node) is created in baseband layer, A and C exchange local information, and make a decision whether to create a logical connection based on the following rules. If it's rejected, disconnect the link connection

a) Master A sends RNG_REQ message to the slave C, formatted as <Bd_addr, Node_type, Piconet_no, Hops_to_controller>.
b) Upon receiving the message, the slave C makes a de-

cision whether to accept this connection request. If accepting it, C sends back RNG_RESP message as a response and updates local configuration, otherwise disconnect it.

c) Upon receiving the responding RNG_RESP, the master A updates local information

Phase 4 : A and C exchange NT and make a decision whether to accept the new physical link based on two-hop neighbors information

a) A and C send their NT(defined as Neighbor_MSG) to each other

b) If (Both of A and C have connected E) then { Select the weakest link within the triangle; If(the weakest link is A->E or C->E) then Disconnect the link, update local information and NT else Disconnect A->C, update local information and NT }

Phase5 : For A and C, if the Hops_to_controller is changed, A or C informs the neighbors, asking them to update Hops_to_controller. The neighbor also informs the change to their neighbors. In this way, all the nodes update the Hops_to_controller if they can find a shorter path.

a) If(local Hops_to_controller for A and C is changed) then Broadcast Hop_MSG to their connected neighbors, the format is <Bd_addr, Node_type, Piconet_no, Hops_to_controller>

b) On receiving Hop_MSG for all nodes If(local Hops_to_controller > Hops_to_controller in the Hop_MSG + 1) then {Update NT based on the received Hop_MSG; Set local Hops_to_controller = Hops_to_controller in the Hop_MSG + 1; Generate and broadcast Hop_MSG to all the neighbors; }

Phase 6 : Make a decision whether to stop discovery procedure

If all the node has (Hops_to_controller < 1000) then Stop discovery

5. RNG based Scatternet Formation Scenarios

We show subsequent actions(A.1, A.2, ...), and the values of messages(RNG_REQ/RNG_RESP/Hop_MSG/Neighbor_MSG) or tables(NT/NPT) if needed in each phase according to our proposed algorithm.

Stage 1 : Node 0 is a controller and node (1~5) have any role of Master/Slave/Slave-slave bridge/Master-slave bridge depending on how to deploy the discovery. Assume that at first node 1 discovers node 2(see Fig. 2 and Fig. 3).

Node 0 : Controller

Node 1~Node 5 : Initially, they all have Free_Node as node type, Not_Assigned as role, Not_Assigned as Ch_id, and 1000 as Hops_to_Controller. They have any role of Master/Slave/Slave-slave bridge/Master-slave bridge

HTC(Hop_to_Controller) of Node 1 to Node 5 is 1000

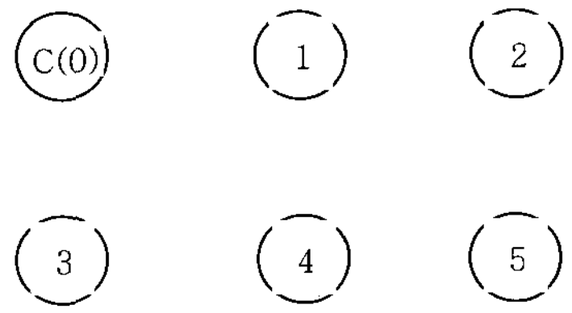


Fig. 2. Initial state

The followings are the detailed actions(A.1~A.5).

A.1 : Node 1 creates an entry for node 2 in its NT andthe signal strength from node 2 is 3. Then, the entry in node 1's NT is <002, 3, Free_Node, Not_Connected, Not_Assigned, [1], 1000>.

A.2 : Node 1 requests a connection to node 2(1 -> 2), sending RNG_REQ to node 2 as <001, Free_Node, [1], 1000>.

A.3 : Because node 2 is free which means the type of node 2 is Free_Node, it accepts the requested connection from node 1, setting accept=YES, updating the entry for node 1 in its NT as <001, 3, Master_Node, As_Master, Not_Assigned, [1], 1000>, sending back RNG_RESP message as <002, Slave_Node, 1, 1000>

A.4 : Receiving the RNG_RESP from node 2, node 1 updates its NT as <002, 3, Slave_Node, 0, [1], 1000> and local information as local Node_type = Master_Node and local Piconet_no = 1.

A.5 : Node 1 and 2 exchange Neighbor_MSG to each other. RNG condition is not satisfied. Node 1 and 2 receive Neighbor_MSG and do nothing, and node 1 and 2 don't broadcast Hop_MSG because Hops_to_controller is not changed. None of the nodes have Hops_to_controller < 1000, so continue discovery procedure.

C: Controller, M : Master, S : Slave

HTC of Node 1 to Node 5 is 1000

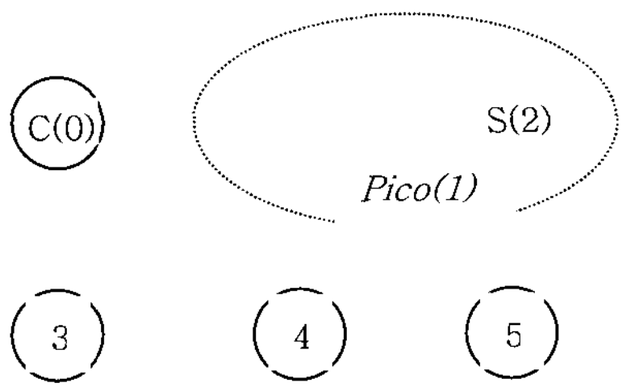


Fig 3. Node 1 discovers node 2

Stage 2 : Assume that node 4 discovers node 2(see Fig. 4).

SS : Slave-Slave bridge

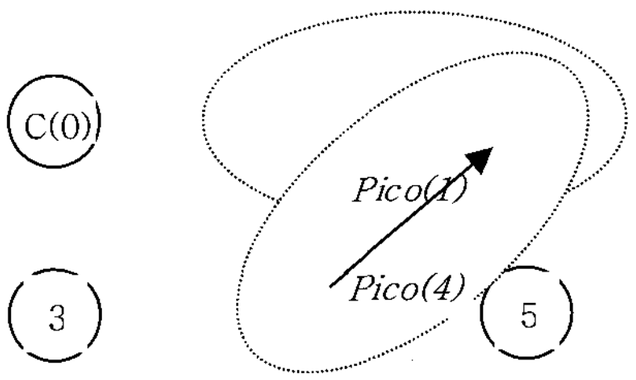


Fig 4. Node 4 discovers node 2

The followings are the detailed actions(A.1～ A.7).

A.1 : Node 4 creates an entry for node 2 in its NT,assuming that the signal strength from node 2 is 1. Then, the entry in node 4's NT is <002, 1.5, Free_Node, Not_Connected, Not_Assigned, [4], 1000>

A.2 : Node 4 requests a connection to node 2(4 -> 2), sending RNG_REQ to node 2 as <004, Free_Node, [4], 1000>

A.3 : Node 2 updates the entry for node 4 in its NT as <004, 1.5, Free_Node, Not_Connected, Not_Assigned, [4], 1000>, accepting the request by setting accept= YES because its local Piconet_no(=1) is not equal to 4, updating the entry for node 4 in NT as <004, 1.5, Master_Node, As_Master, 1, [4], 1000> , registering node 2 as S-S_Bridge, reporting the new connection(4->2) to node 1, sending back RNG_RESP message to node 4 as <002, S-S_Bridge, [1,4], 1000>..

A.4 : Node 1 update NPT by adding the entry <002, 4>

A.5 : Receiving the RNG_RESP from node 2, node 4 updates its NT as <002, 1.5, S-S_Bridge, As_Slave, 1, [1,4], 1000> and local information as local Node_type = Master_Node and local Piconet_no = 4.

/* ch_id field should be added in RNG_RESP message */

A.6 : Node 4 updates NPT as < 002, 1> because the received RNG_RESP is from S-S bridge.

A.7 : Node 2 and 4 exchange Neighbor_MSG to each other. RNG condition is not satisfied because node 2 and node 4 don't have common neighbors. Therefore, node 2 and 4 receive Neighbor_MSG and do nothing, andnode 2 and 4 don't broadcast Hop_MSG because Hops_to_controller is not changed. All the nodes don't have Hops_to_controller < 1000, so continue discovery procedure

Stage 3 : Assume that node 0 discovers node 3(see Fig. 5).
CM : Controller with Master node type
HTC of node 1 to node 5 except node 3 is 1000

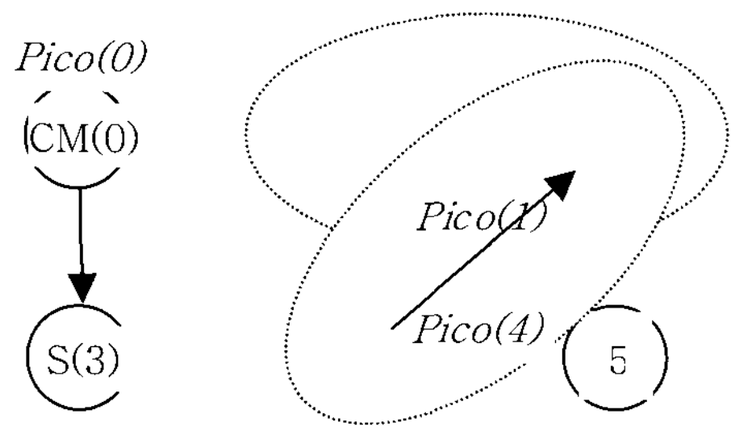


Fig 5. Node 0 discovers node 3

The followings are the detailed actions(A.1～A.5).

A.1 : Node 0 creates an entry for node 3 in its NT, assuming that the signal strength from node 3 is 2. Then, the entry in node 0's NT is <000, 2, Free_Node, Not_Connected, Not_Assigned, [0], 1000>

A.2 : Node 0 requests a connection to node 3(0 -> 3), sending RNG_REQ to node 3 as <000, Free_Node, [0], 1000>

A.3 : Node 3 updates the entry for node 0 in its NT as <000, 2, Master_Node, As_Master, Not_Assigned, [0], 0>, accepting the request by setting accept=YES, updating Hops_to_Controller=1, sending back RNG_RESP message to node 0 as <003, Slave_Node, [0], 1>

A.4 : Receiving the RNG_RESP from node 3, node 0 updates its NT as <003, 2, Slave_Node, As_Slave, Not_Assigned, [0], 1> and local information as local Node_type = Master_Node and local Piconet_no = 0.

/* ch_id field should be added in RNG_RESP message */

A.5 : Node 0 and 3exchange Neighbor_MSG to each other. RNG condition is not met because node 0 and node 3 don't have common neighbors. Therefore, node 0 and 3 receive Neighbor_MSG and do nothing, and node 0 and 3 don't broadcast Hop_MSG because Hops_to_controller is not changed. All the nodes don't have Hops_to_controller < 1000, so continue discovery procedure

Stage 4 : Assume that node 0 discovers node 4(see Fig. 6).
MS : Master-Slave bridge
HTC of node 3 and node 4 is 1
HTC of node 1, node 2 and node 5 is 1000

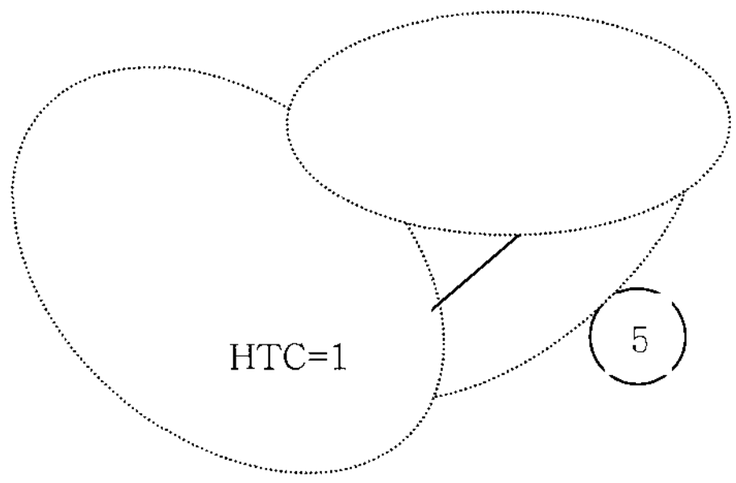


Fig 6. Node 0 discovers node 4

The followings are the detailed actions(A.1～ A.14).

A.1 : Node 0 creates a newentry for node 4 in its NT,assuming that the signal strength from node 2 is 1.5. Then, the entry in node 0's NT is <004, 1.5, Free_Node, Not_Connected, Not_Assigned, [0], 1000>

A.2 : Node 0 requests a connection to node 4(0 -> 4), sending RNG_REQ to node 4 as <004, Master_Node, [0], 1000 >

A.3 : Node 4 updates the entry for node 0 in its NT as <000, 1.5, Master_Node, Not_Connected, Not_Assigned, [0], 1000>, accepting the request by setting accept= YES because node 4 is Master_Node, updating its local information as node_type=MS_Bridge and pico_no=[0,4], updating the entry for node 0 in NT as <000, 1.5, Master_Node, As_Master, 1, [0], 1000>, registering node 4 as M-S_Bridge, , sending back RNG_RESP message to node 0 as <004, M-S_Bridge, [0,4], 1000>.

A.4 : Node 0 update local Hops_to_Controller=0 /* Not necessary because initially node 0 knows its HPC */

A.5 : Node 0 update NPT by adding the entry <004, 4>

A.6 : Receiving the RNG_RESP from node 4, node 0 updates its NT as <004, 1.5, M-S_Bridge, As_Slave, 1,

[0,4], 1> and local information as local Node_type = Master_Node and local Piconet_no = 0.

/* ch_id field should be added in RNG_RESP message */

A.7 : Node 0 updates NPT as < 004, 4> because the received RNG_RESP is from M-S bridge.

A.8 : Node 0 and 4 exchange Neighbor_MSG(not NT) to each other. RNG condition is not met because node 0 and node 4 don't have common neighbors. Therefore, node 0 and 4 receive Neighbor_MSG and do nothing

A.9 : As Hops_to_Controller of node 4 has been changed from 1000 to 1, node 4 broadcasts Hop_MSG to node 0 and node 2 as <004, MS_Bridge, [0,4],1>.

A.10 : Receiving Hop_MSG from node 4, node 2 updates local HPC=2, then broadcast Hop_MSG to its neighbors as <002, SS_Bridge, [0,4], 2>

A.11 : Receiving Hop_MSG from node 2, node 1 updates local HPC=3

A.12 : Node 1 generates and broadcasts Hop_MSG to neighbors

A 13 : When node 0 receives Hop_MSG from node 4, and node 4 receives Hop_MSG fromnode 2, and node 2 receives Hop_MSG from node 1, they do nothing(that is, don't update each local HPC) because local HTC <= HTC in the Hop_MSG + 1

A.14 : All the nodes don't have Hops_to_controller < 1000, that is, HPC of node 5 is 1000, so continue discovery procedure

Stage 5 : Assume that node 3 discovers node 4(see Fig. 7)

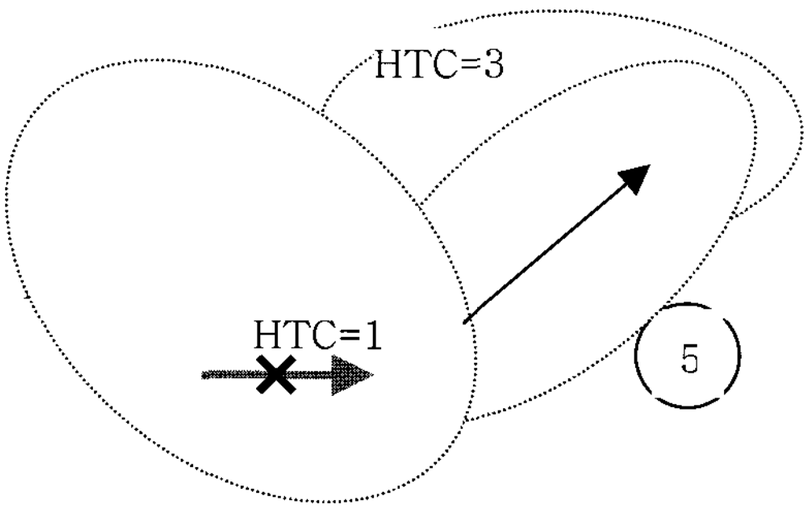


Fig 7. Node 3 discovers node 4

The followings are the detailed actions(A.1 ~ A.4).

A.1 : Node 3 creates a new entry for node 4 in its NT assuming that the signal strength from node 4 is 3. Then, the entry is <004, 3, Free_Node, Not_Connected, Not_Assigned, [3/Not_Assigned], 1000>.

A.2 : Node 3 makes physical connection to node 4, sending RNG_REQ to node 3 as <003, Slave_Node, [0], 1>.

A.3 : Node 4 updates the entry for node 3 in its NT as <003, 3, Slave_Node, Not_Connected, 1, [0], 1000>

A.4 : As node 4 is MS_Bridge and its local piconet_no(=0) is equal to piconet_no(=0) in the received RNG_REQ, it rejects the new linkand disconnects the physical link connection?(3 -> 4)

Stage 6 : Assume that node 4 discovers node 1. RNG algorithms are applied to triangle (see Fig. 8).

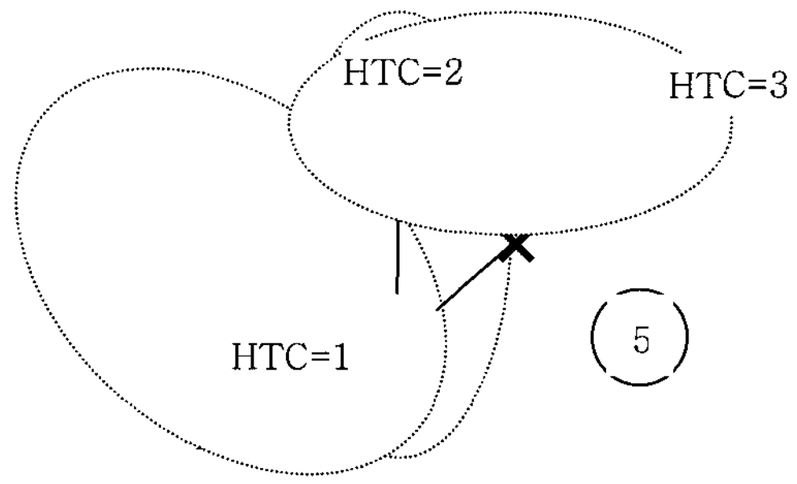


Fig 8. Node 4 discovers node 1

The followings are the detailed actions(A.1 ~ A.13).

A.1 : Node 4 creates a new entry for node 1 in its NT assuming that the signal strength from node 1 is 2. Then, the entry is <001, 2, Free_Node, Not_Connected, Not_Assigned, [4]/Not_Assigned, 1000>.

A.2 : Node 4 makes physical connection to node 1, sending RNG_REQ to node 1 as <004, MS_Bridge, [0,4], 1>.

A.3 : Node 1 updates the entry for node 4 in its NT as <004, 2, MS_Bridge, As_Master, 1, [0,4], 1>

A.4 : As node 1 is Master, it accepts the new local link connection request(accept=YES) and updates local info as type=MS_Bridge, piconet_no=[1,4]

A.5 : As local HTC of node 1 is 3, and HTC in the received RNG_REQ is 1, node 1 updates local HTC since 1+1 <= 3.

A.6 : Node 1 updates NPT by adding new entry <001, 4>, sending RNG_RESP to node 4 as <001, MS_Bridge, [1,4], 2>

A.7 : Node 4 updates NT based on the received RNG_RESP as <001, 2, MS_Bridge, As_Slave, 2, [2,4], 2>, updating local info as node_type=MS_Bridge, piconet_no=[0,4], updating NPT by adding new entry <001, 1> since the RNG is received from MS_Bridge.

A.8 : Node 4 reports the new connection to node 0 to register node 4 as bridge as it has been changed into MS_Bridge

A.9 : Node 1 and node 4 send Neighbor_MSG to each other. They have the common neighbor, node 2, selecting the weakest link (4->2). Node 4 is responsible for disconnecting the link(4->2), updating NT(node 4 and node 2 register each other as Not_Connected in each NT) and local info(node 2 become Slave, piconet_no=[1], HTC=1000)

A.10 : As HTC of node 1 has been changed, it broadcasts Hop_MSG to node 4 and node 2. Node 2 updates NT as <001, 3, MS_Bridge, As_Master, 0, [1], 2> based on received Hop_MSG, updating local HTC=3 because local HTC is 1000 and HTC in received Hop_MSG is 2

A.11 : Node 4 receives the Hop_MSG from node 1, but nothing changed because its NT is already up to date.

A.12 : Node 2 generates and broadcasts Hop_MSG to

node 1, then node 1 receives the Hop_MSG from node 2, but nothing is changed

A.13 : As HPC of node 5 is 1000, continue discovering procedure.

The routing optimization and RNG-based rule are applied on the shadow area of triangle. The link from node 4 toward node 2 is weakest in that area, therefore,the link is disconnected.

Stage 7 : Assume that node 2 discovers node 5) (see Fig. 9).

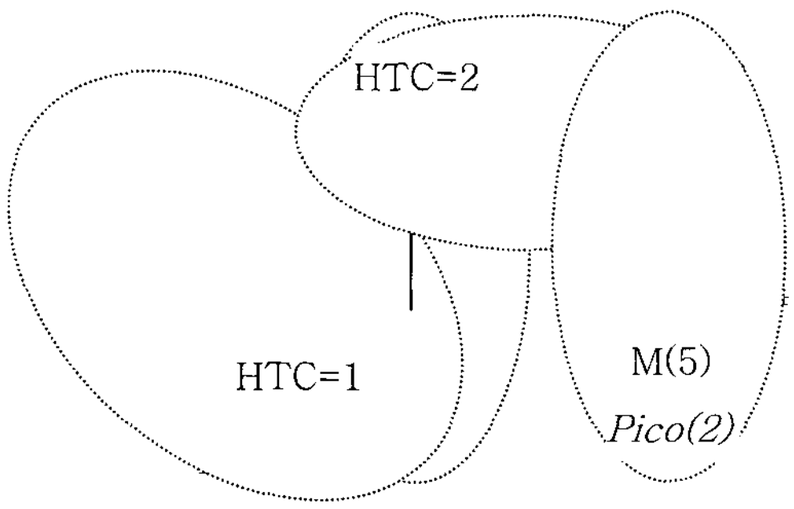


Fig 9. Node 2 discovers node 5

The followings are the detailed actions(A.1～A.11).

A.1 : Node 2 creates a new entry for node 5 in its NT assuming that the signal strength from node 5 is 2. Then, the entry is <001, 2, Free_Node, Not_Connected, Not_Assigned, [2]/Not_Assigned, 1000>.

A.2 : Node 2 makes physical connection to node 5, sending RNG_REQ to node 1 as <002, Slave, [1], 3>.

A.3 : Node 5 adds new entry in NT as <002, 2, Slave, Not_Assigned, [1], 3>

A.4 : Node 5 accepts the request(accept=YES) as it is free, updating local info as node_type=Slave, piconet_no=2, updating NT by adding new entry as <002, 2, MS_Bridge, Master, 0, [1,2],3>

A.5 : As local HTC of node 5 is 1000 and HTC in received RNG_REQ is 3, node 5 updates local HTC=4.

A.6 : Node 5 sends RNG_RESP to node 2 as <005, Slave, [2], 4>

A.7 : Receiving the RNG_RESP, node 2 updates NT by adding as <005, 2, Slave, As_Slave, 1, [2],4>, updating local info as node_type=MS_Bridge, piconet_no=[1,2].

A.8 : Node 2 reports to node 1 to register node 2 as bridge as node 2 is MS_Bridge. Node 1 updates NPT by adding as <002,2>

A.9 : Node 2 and node 5 send Neighbor_MSG to each other. RNG condition is not met, so do nothing.

A.10 : As local HTC is changed, node 5 broadcasts Hop_MSG to node 2. Receiving the Hop_MSG, node 2 does nothing because NT is already up to date.

A.11 : As all the nodes have a Hop_to_Controller < 1000, stop the discovering procedure

5. Simulation and Evaluations

In RNG-FHR, the delays to form the scatternet for n nodes grow until 30 nodes, but go down if the number of nodes are more than 30(Fig. 10). It's because the algorithm takes more messages to come to form the connected topology from the initial state, After the connected topology has been formed, the addition and deletion of a node or a link in RNG-FHR does not influence any other node/link and depends only on local information, thus, it exchanges fewer messages and simpler. However, the coveraging time to be in a stable-state topology grows logarithmically as the size of the scatternet is large(Fig. 11).

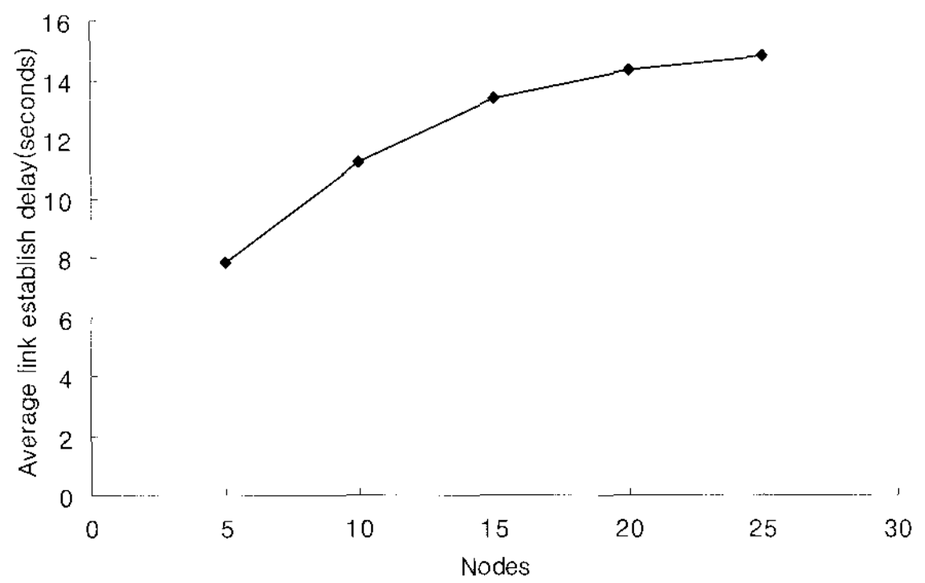


Fig 10. Average link establish delay

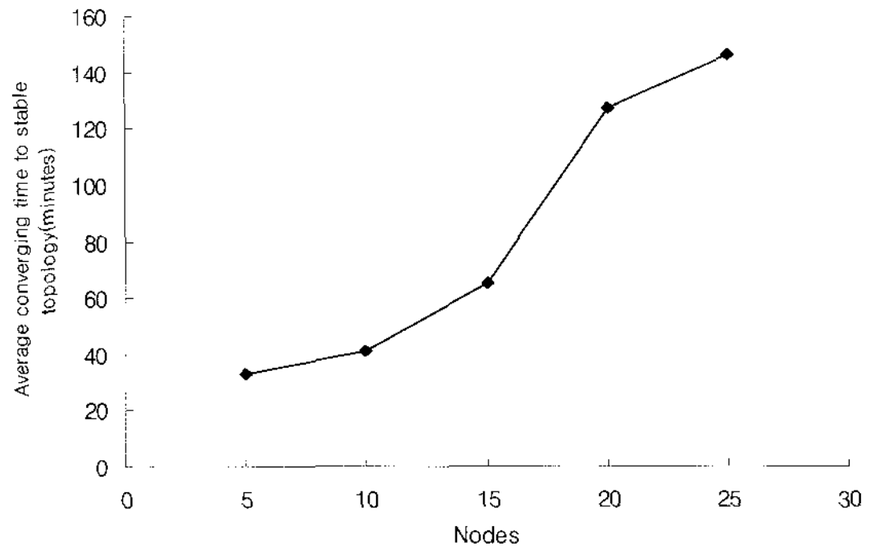


Fig 11. Average converging time to stable topology

We evaluate the delay for healing network partitions when any node leaves(Fig. 12). If any node leaves, the scatternet will be partitioned into smaller piconets. In this case, coordinating nodes try to connect each piconet during the healing the scatternet. The delay grows logarithmically with the number of network partitions. The delay for each algorithm are growing analogously for small number of partitions, but the growing rate of delay gets smaller for each algorithms as the number of partitions gets larger and larger because both of algorithm depend on only local information that each node maintains only about adjacent nodes.

Therefore, RNG-FHR is able to heal all partitions when all the nodes are within radio proximity because it reduces the tasks of discovering and merging partitions to coor-

dinators and roots respectively.

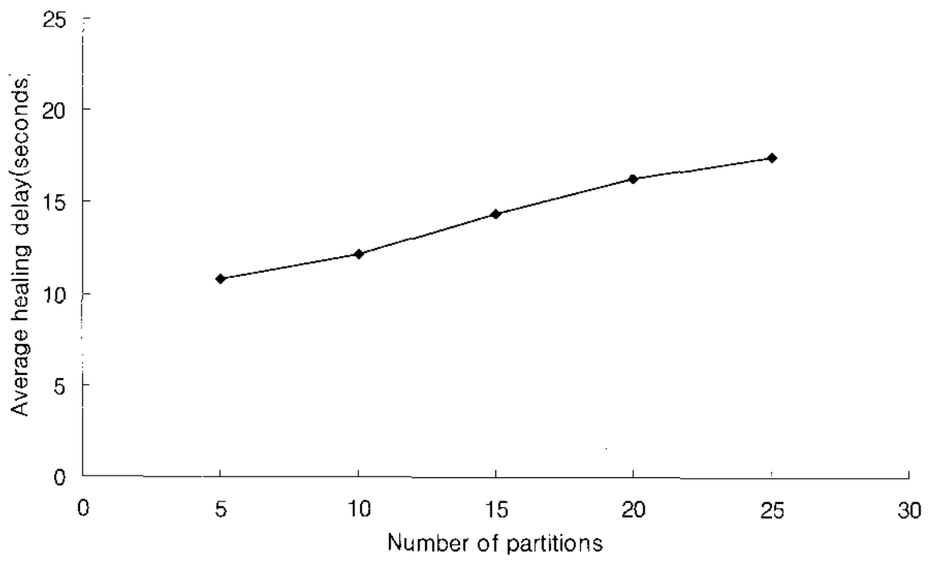


Fig 12. Average healing delay

6. Conclusion

RNG-FHR is RNG based scatternet formation, self-healing, and self-routing path optimization which are done depending only on local information throughout the graph, so is more simpler than any other algorithm. It allows nodes to be added and/or deleted at any time, healing partitions and routing whenever they happen. Furthermore, it allows an added node to begin communication with neighbor nodes.

The algorithm is simpler and more practical than any other distributed algorithms from a point of view in deploying the distributed and dynamic small scale environment. Using ns-2 and extensible Bluetooth simulator called blueware, we simulated and evaluated to show that the RNG-FHR has analogous latencies in scatternet formation and healing due to the exchange of fewer messages and the only dependency on local information. However, link establishment in RNG-FHR has large latency. As a result, we notice that even though RNG-FHR does not superior performance, it is more practical in implementing because of simplicity and local control.

As a result, RNG-FHR scatternet is unlikely that it will be deployed in large-scale ad hoc networks because it has less performance metrics and more complexity as the network size increases. However, it surely possible that the scatternet will be practically applied in deploying small-scale ad hoc networks.

References

[1] Blueware: Bluetooth simulator for ns. http://nms.lcs.mit.edu/projects/blueware/, http://nms.csail.mit.edu/projects/blueware/software

[2] ns-2 Network Simulator. http://www.isi.edu/vint/nsnam

[3] G. Tan, A. Miu, H. Balakrishnan, and J. Guttag. An Efficient Scatternet Formation Algorithm for Dynamic Environments. *IASTED International Conference on Communications and Computer Network(CCN02)*, Cambridge, MA, Nov. 2002.

[4] G. Tan and J. Guttag. A Locally Coordinated Scatternet Scheduling Algorithm. *The 27thAnnual IEEE Conference on Local Computer Networks(LCN)*, Tampa, FL, Nov. 2002.`

[5] G.Tan. Blueware: Bluetooth Simulator for ns. MIT Technical Report, MIT-LCS-TR-866, Cambridge, MA, October, 2002.

[6] Evangelos Vergetis, Roch Guerin, Saswati Sarkar, and Jacob Rank. Can Bluetooth Succeed as a Large-Scale Ad Hoc Networking Technology?, *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 3, March 2005.

[7] Xiang-Yang Li, Yu Wang, Peng-Jun Wan, Wen-Zhan Song, and Ophir Frieder. Localized Low-Weight Graph and Its Applications in Wireless Ad Hoc Networks. *IEEE INFOCOM 2004*.

[8] Theodoros Salonidis, Pravin Bhagwat, Leandros Tassiulas, and Richard LaMaire. Distributed Topology Construction of Bluetooth Personal Area Networks. *IEEE INFOCOM 2001*.

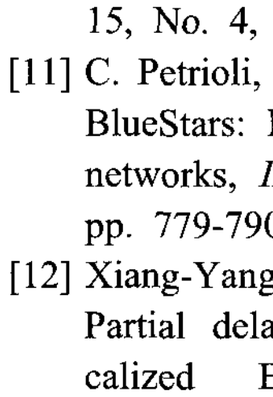
[9] Chiara Petrioli, Stefano Basagni, and Imrich Chlamtac. Configuring BlueStars : Multihop Scatternet Formation for Bluetooth Networks. *IEEE Transactions on Computers*, Vol. 52, No. 6, June 2003.

[10] Xiang-Yang Li, Ivan Stojmenivic, and Yu Wang. Partial Delaunay Triangulation and Degree Limited Localized Bluetooth Scatternet Formation. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No. 4, April 2004.

[11] C. Petrioli, S. Basagni, and M. Chlamtac, Configuring BlueStars: Multihop scatternet formation for Bluetooth networks, *IEEE Transactions on Computers.*, vol. 52, pp. 779-790, June 2003

[12] Xiang-Yang Li, Ivan Stojmenovic, and Yu Wang, Partial delaunay triangulation and degree limited localized Bluetooth scatternet formation, *IEEE Transactions on Parallel Distributed System*, vol. 15, pp. 350-361, Apr. 2004

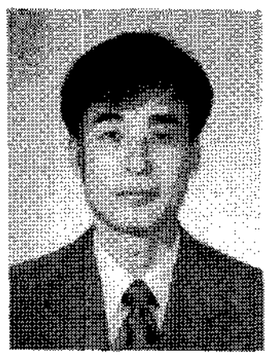
[13] Ivan Stojmenovic and Nejib Zaguia, *Bluetooth scatternet formation in ad hoc wireless networks*, University of Ottawa, Sep. 2004



Chung Ho Cho

He received the B.S., M.S., and Ph.D. degrees in Computer Science from Chonnam National University, Gwangju, Korea, in 1984, 1987, and 1996 respectively. From 1988 to 1997, he worked for ETRI. He has been a visiting researcher at the Department of Electrical and Computer Engineering, University of Arizona from 2005 to 2006. At present, he is an Associate Professor at the Department of Information and Communication Eng., Gwangju University, since 1997. His research interests include fuzzy theory, graph theory based networking algorithm and network QoS.

Phone : +82-62-670-2468
E-mail : chcho@gwangju.ac.kr



Dong Cheul Son
He received the B.S., M.S. degrees in Computer Engineering from Kyungpook National University, Daegu, Korea, in 1983 and 1985, respectively. and Ph.D. degrees in Electrinic Egineering from Chungbuk National University, Chungju, Korea, in 2001. From 1983 to 1998, he worked for ETRI. At present, he is an Associate Professor at the Department of Information and Communication Eng., Cheonan, University, since 2002. His research interests include AI, fuzzy theory and XML based information integration, Intenet engineering.
Phone : +82-41-620-9536
E-mail: dcson@cheonan.ac.kr

Chang Suk Kim
He received the B.S., M.S. and Ph. D. degrees in Computer Engineering from Kyungpook National University, Daegu, Korea, in 1983, 1990 and 1994, respectively. He was a post-doctoral researcher at University of California, San Diego. He worked for ETRI from 1983 to 1994. At present, he is an Associate Professor at the Department of Computer Education, Kongju National University, since 1998. His research interests include intelligent databases, fuzzy, XML based information integration and mobile communication.
Phone : 041-850-8822
E-mail: csk@kongju.ac.kr