A Bluetooth Scatternet Reformation Algorithm

Han-Wook Lee and Sang Ken Kauh

Abstract: **Bluetooth is reputed as a wireless networking technology supplying ad-hoc networks between digital devices. In particular, Bluetooth scatternet is an essential part of dynamic ad-hoc networks. Yet, there have not been sufficient researches performed on scatternet environment. This paper proposes a scatternet reformation algorithm for ad-hoc networks for instances where some nodes enter or leave the scatternet. The proposed algorithm is a general algorithm that can be applied to many types of Bluetooth scatternet regardless of the topology. The proposed algorithm is made for two reformation cases, i.e., nodes leaving and nodes entering. For the reformation when nodes leave a scatternet, the recovery node vector (RNV) algorithm is proposed. It has short reformation setup delay because the process involves a single page process (not including inquiry process). For the reformation when nodes enter a scatternet, the entry node algorithm is proposed. This is a simple and easily implementable algorithm. In this paper, real hardware experiments are carried out to evaluate the algorithm's performance where the reformation setup delay, the reformation setup probability and the data transfer rate are measured. The proposed algorithm has shown improvement in the reformation setup delay and probability.**

Index Terms: **Ad-hoc network, Bluetooth, bridge, entry node, node matrix, node weight, recovery node vector, reformation, scatternet.**

I. INTRODUCTION

Bluetooth has been reputed as a promising wireless networking technology supplying ad-hoc networks between digital devices such as cellular phone, PDA, notebook, desktop, etc. Bluetooth supports two types of networks, namely piconet and scatternet. A scatternet can form fully distributed ad-hoc networks and can be applied to scalable and flexible networks between digital devices. However, a scatternet has not been clearly described in Bluetooth specification [1], [2]. This topic has been a major topic of discussion among researches in relation to the formation algorithm, scheduling scheme, etc.

In past researches on scatternet formation, only static environment where nodes neither enter nor leave the network was considered [3]–[9]. In order for Bluetooth to be applied to dynamic networks such as PAN, a reformation algorithm is needed to deal with instances where some nodes enter or leave the network.

In this paper, a general Bluetooth scatternet reformation algorithm which can be applied to most types of scatternet regardless of the topology is proposed. The algorithm can be applied for network reformation after some nodes leave or enter the scatternet, and can retain the scatternet topology even after the reformation process. The proposed reformation algorithm is made for two reformation cases, i.e., nodes leaving and entering.

The reformation algorithm when a node leaves a scatternet is based on recovery node vector (RNV) which is composed of recovery master (RM) and recovery slave (RS), and hence given the name recovery node vector algorithm. The RM and RS are normally designated based on the neighboring node information which is exchanged after a connection process. As the reformation process is composed of only page process, the reformation setup delay can be minimized.

The reformation algorithm when a node enters a scatternet is called entry node algorithm. An entry node plays the role of offering new nodes the entrance to the scatternet. This means that a new node can only enter the scatternet via this entry node. This algorithm is simple and can be easily implemented in a Bluetooth system.

The proposed reformation algorithm, which is implemented in commercial Bluetooth hardware, has many practical uses. To evaluate the performance of the proposed algorithm, we conducted real hardware experiments. In these experiments, the reformation setup delay, the reformation setup probability and the data transfer rate were measured.

In this paper, we present some related works in Section II and then, present the basic parameters related to scatternet and classify the node types in a scatternet in Section III. Next, we will describe the details of the proposed reformation algorithm in Section IV and analyze the performance of the algorithm based on the results of real hardware experiments in Section V. Finally, we will present our conclusions in Section VI.

II. RELATED WORKS

Scatternet topology can be classified into the following types such as tree [5], [10]–[12], star [7], ring [6], [13], [14], and mesh [8], [9]. Each type of scatternet topology has its pros and cons. The performance comparison of various scatternet topology has been reported in past researches [15], but most of these researches started their research from the assumption that no node enters or leaves the scatternet, or did not consider the aspect of the reformation process after these nodes enter or leave the scatternet [3]–[9].

Only a few researches have been reported dealing with the reformation process. Among these researches, only the possibility of reformation algorithm was shown in [14]. In [10] and [11], a reformation algorithm based on node types such as master, slave and bridge was referred to briefly. In [12], the possibility that healing function in the protocol can be applied to dynamic networks was shown. In [13], the reformation algorithm in the case where some nodes in the scatternet leave was described in more detail. This algorithm was operated by inquiry process using DIAC [1], [2]. It has an advantage in that it is easy to im-

Manuscript received October 25, 2004; approved for publication by Ekram Hossain, Division II Editor, June 29, 2005.

The authors are with the School of Mechanical and Aerospace Engineering, Seoul National University, Seoul, Korea, email: ezuinos@paran.com, kauh@snu.ac.kr.

plement. However, the algorithm faces a long reformation setup delay due to the inquiry duration. In addition, it is possible that if two nodes in the scatternet leave the networks simultaneously, the reformation could be conducted incorrectly.

III. NODE CLASSIFICATION IN THE SCATTERNET

The proposed algorithm is operated based on node types. In this paper, node matrix and node weight are introduced for the quantitative classification.

A. Basic Parameters of Scatternet

Bluetooth scatternet is composed of a set of piconets. A piconet is a small size network which has up to 7 slave nodes per 1 master node. Master synchronizes TDD time slots and frequency hopping channels with slave nodes, and manages the status of the nodes and links [1], [2]. If a node X is master in a piconet, the piconet is represented as $Pico(X)$, which is a set of slave nodes in the piconet. In Fig. 1, there are three piconets and each piconet is represented as follows.

$$Pico(A)=\{B,C\}, Pico(D)=\{C,E\}, Pico(E)=\{F,G,H\}.$$

If the number of slave nodes in a piconet can be represented as $|Pico(X)|$, the following relation can be obtained. $|Pico(X)| \leq s_{\max}$ where $s_{\max}$ is the maximum number of slave nodes in a piconet and which can go up to 7 [1], [2]. For this research, $s_{\max}$ was set to 6.

If a scatternet is composed of n piconets whose master nodes are $M_1, M_2, \cdots, M_n$, the scatternet can be represented as follows.

$$Scat(M_1, M_2, \cdots, M_n) = \{Pico(M_1), Pico(M_2), \cdots, Pico(M_n)\}.$$

The scatternet in Fig. 1 is described in the following manner.

$$Scat(A, D, E) = \{Pico(A), Pico(D), Pico(E)\}.$$

We introduce scatternet dimension which means the number of piconet in a scatternet, and express it as d_{scat}.

$$d_{scat} = |Scat(M_1, M_2, \cdots, M_n)| = n.$$

In order for a few piconets to try to form a scatternet, some interface nodes between the piconets are needed. The interface nodes are called bridges. In general, a bridge node belongs to two piconets, and has two simultaneous roles. If a bridge is operated as master in one piconet and as slave in the other piconet at the same time, the bridge is named master-slave bridge (MS-bridge). Similarly, if a bridge has two slave roles for two piconets, the bridge is named slave-slave bridge (SS-bridge).

If a bridge node between $Pico(X)$ and $Pico(Y)$ is represented as $Bridge(X, Y)$, the following relation is obtained.

$$Bridge(X, Y) = (\{X\} \cup Pico(X)) \cap (\{Y\} \cup Pico(Y)).$$

For Fig. 1, the bridge nodes are represented as follows.

$$Bridge(A, D) = C, Bridge(D, E) = E.$$

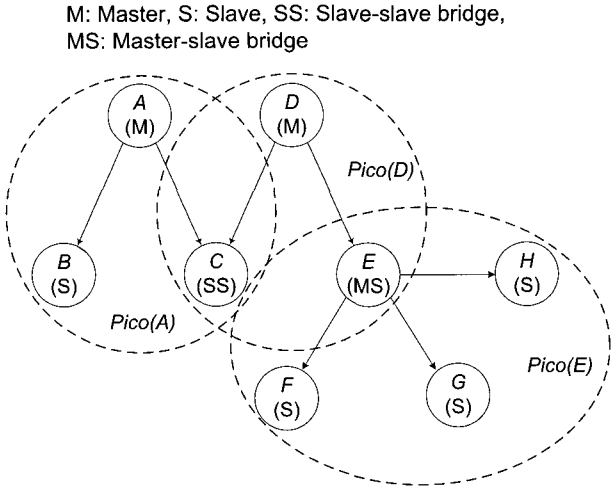

Fig. 1. Piconets and scatternets.

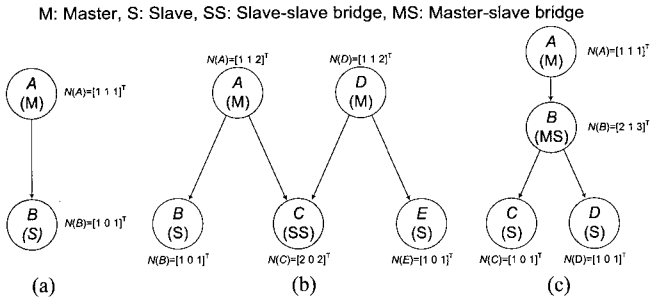

Fig. 2. Node classification and node matrix: (a) Master/slave, (b) slave-slave bridge, (c) master-slave bridge.

Table 1. Node parameters.

Parameter	Symbol	Details
Degree of node	d	Number of piconets which one node belongs to
Role of node	r	For master/MS-bridge, $r = 1$ for slave/SS-bridge, $r = 0$
Number of links	l	Number of links connected to a node

B. Node Classification

The nodes in a scatternet can be classified into four types: Master, slave, MS-bridge, and SS-bridge. In general, MS-bridge and SS-bridge may be used together in one scatternet topology. However, only SS-bridge is used in [4], [13], [14], and only MS-bridge is used in [6].

For the quantitative classification of nodes, the parameters which determine node properties are introduced and listed in Table 1. In particular, the number of links l which are connected to one node satisfies the following relation.

$$l \leq r(s_{\max} - 1) + d.$$

In other words, the maximum number of links which one node can make is $r(s_{\max} - 1) + d$.

Based on the parameters in Table 1, a node matrix is intro-

Table 2. Node matrix and node weight according to node types.

Node type	$N(X)$	$W(X)$
Master	$[1\ 1\ 1]^T$	$\sqrt{2+l^2}\ (l \geq 1)$
Slave	$[1\ 0\ 1]^T$	$\sqrt{2}$
SS-bridge	$[2\ 0\ 2]^T$	$\sqrt{8}$
MS-bridge	$[2\ 1\ l]^T$	$\sqrt{5+l^2}\ (l \geq 2)$

duced and defined as follows.

$$N(X) = [d\ r\ l]^T.$$

The examples of node matrices according to node types are shown in Fig. 2. In addition, the node weight is defined as the follows.

$$W(X) = |N(X)|.$$

The node weight refers to the amount of loads which a node can endure. The load which a node endures is proportional to the number of links connected to the node, and a master node has a larger load than a slave node. In addition, a bridge node which has degree of nodes d over 2 endures a larger load than a non-bridge node. The larger the node weight is, the more severe the scatternet failure will be when the node leaves the scatternet.

In particular, the node which has very large degree of node may pose as an obstacle in the networks [11]. Therefore, the degree of node is limited to 2 [6], [10], [11], [13], [14]. Node matrix and node weight according to the node types is listed in Table 2.

IV. BLUETOOTH SCATTERNET REFORMATION ALGORITHM

In this paper, the proposed reformation algorithm is made for two reformation cases, i.e., nodes leaving and entering. For the reformation when a node leaves a scatternet, a RNV algorithm is proposed. For the reformation case where a node enters a scatternet, an entry node algorithm is proposed. The RNV algorithms will be presented in Section IV-A below. Next, the entry node algorithm will be described in Section IV-B.

A. Recovery Node Vector Algorithm

In the distributed topology networks such as a scatternet, the failure of one node may materially affect the entire network. The amount of total network failure is determined in terms of the node weight of the failed node. It is particularly important how fast networks are recovered in the case where nodes leave the network frequently. For example, some devices such as PDA, cellular phone, notebook, etc., in wireless PAN may enter or leave the network frequently depending on the intended purpose of the user and may be terminated abruptly because of user's mistake or battery exhaustion. In these cases, it is important that the network is recovered and the reformation process should be undertaken as quickly as possible.

The recovery node vector algorithm was developed so that the scatternet may be recovered and reformed when the above situation occurs. The RNV algorithm has the following properties.

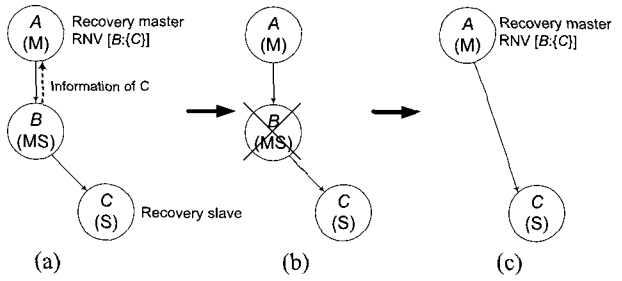

Fig. 3. Overview of recovery node vector algorithm: (a) Node B sends the information of node C ($rSlave$) to node A ($rMaster$), (b) node B leaves the scatternet, (c) node A ($rMaster$) connects to node C ($rSlave$) based on RNV.

(1) It is a general algorithm which can be applied to most scatternet topology such as tree, star, mesh, and ring regardless of the topology. In addition, the scatternet topology can be maintained after the reformation process.
(2) In can be applied to the scatternet which does not have all nodes in the radio coverage. However, all nodes in two neighboring piconets are in the same radio coverage.
(3) It can be applied for both normal and abnormal node termination.
(4) It is operated based on four types of node namely master, slave, MS-bridge, and SS-bridge.
(5) The network recovery and reformation include only the page process. As there is no inquiry duration, the reformation setup delay can be reduced.
(6) It can be applied to cases where more than two nodes leave scatternet simultaneously.

Fig. 3 shows the schematic of the operation process of an RNV algorithm is shown. There are three nodes (node A, B, and C) as shown in Fig. 3(a), the roles of each node are master, MS-bridge, and slave, respectively. These nodes exchange information after connection is achieved. The exchanged information is composed of BD_ADDR, clock offset, and node weight.

In particular, the information of BD_ADDR and clock offset enables a node to make connection without having to undergo any inquiry process. They are the essential parameters of a HCI command, HCI_Create_Connection, in Bluetooth specification. This command is used for activating baseband to perform a page process. In general, a node gathers information during the inquiry process which normally takes 2 to 8 seconds hence resulting in network formation delay. However, with this algorithm, a node gathers information through the exchange of information with neighboring nodes after the connection process which takes less than 1 second, and hence decreasing time delay.

After the connection process between two nodes is completed, the nodes designate a RM and RS among neighboring nodes. If a node leaves a scatternet, the RM designated by the node tries to connect to the RS designated by the node. In this way, the scatternet is recovered and reformed. Therefore, the designation of RM and RS is a core algorithm and is performed using node weight exchanged between nodes.

The RM and RS designated by node X can be represented as $rMaster(X)$ and $rSlave(X)$, respectively. For example, in

Fig. 3(a), the RM and RS are expressed as

$$rMaster(B) = A,\ rSlave(B) = C.$$

After a node designates the RM and RS, the node transfers the information of the RS to the RM. Then, the RM makes a RNV based on the information of the RS. If a RM receives the information of the RS Y from the node X, the RNV is expressed as follows (more than one RS can be designated).

RS Y from the node X : $[X : \{Y\}]$.

In Fig. 3(a), node B designates nodes A and C as the RM and RS, respectively, and RM A receives the information of RS C from node B. Therefore, the RNV which RM A makes is $[B : \{C\}]$.

After node B leaves the scatternet as shown in Fig. 3(b), RM A connects to RS C according to the RNV. In this way, the scatternet is reformed.

Therefore, an important aspect of the RNV algorithm is that the RM and RS of each node are designated in advance. If a node leaves the scatternet, the recovery and reformation process is activated immediately by the connection between the RM and the RS. Therefore, these nodes have to reserve the RNV for neighboring nodes.

The RNV algorithm is operated based on four types of nodes as shown in Table 2. In the case of slave node ($N(X) = [1\ 0\ 1]^T$), no network failure occurs after a slave node leaves the scatternet. Therefore, the reformation algorithm for the failure of a slave node need not be considered.

A.1 Algorithm for the Case of a Master Node Failure

If a master node leaves a scatternet, the piconet managed by the master node is broken entirely. This can result in severe network failure throughout the scatternet structure. In the case of a master node failure, the number of the RS may be more than one because all members of the piconet managed by the failed master are isolated from the scatternet.

The algorithm of designating the RM and RS in master node is shown in Table 3. If a RM node is designated, the rest of the piconet members which the failed master managed become the RS nodes.

The algorithm of designating the RM is decided depending on the type of members in the piconet. A slave node has priority over a bridge node in RM designation. If there are more than one slave node in the piconet, one of the slave nodes is designated as RM (Table 3 lines 4–5). An example of this case is shown in Fig. 4(a).

If there is no slave node and only bridge nodes exist in the piconet, an MS-bridge node has priority over an SS-bridge node in RM designation (Table 3 lines 7–8). This case is shown in Fig. 4(b). In Fig. 4(b), there are two bridge nodes C and E in $Pico(D)$, and node E is designated as the RM because it is MS-bridge.

However, there are only SS-bridge nodes in the piconet, one of SS-bridge nodes is designated as the RM (Table 3 lines 9–10). This case is shown in Fig. 4(c). In Fig. 4(c), there are only SS-bridge nodes C and E in $Pico(D)$ and one of SS-bridge nodes, C is designated as the RM by a master node D.

Table 3. Algorithm for designating RM/RS in master node.

Node M is a master node, node B is one of slave nodes in $Pico(M)$
1 if $W(M) = \sqrt{3}$ ($\|Pico(M)\| = 1$)
2 Do nothing
3 else if $W(M) > \sqrt{3}$ ($\|Pico(M)\| \geq 2$)
4 if there is a node B which satisfies $W(B) = \sqrt{2}$ (B is slave)
5 Designate node B as $rMaster(M)$
6 else (All slave nodes are bridges, $W(B) \geq \sqrt{8}$)
7 if there is a node B which satisfies $W(B) \geq \sqrt{9}$ (MS-bridge)
8 Designate node B as $rMaster(M)$
9 else (All slaves are SS-bridges)
10 Designate one of nodes in $Pico(M)$ as $rMaster(M)$
11 if $rMaster(M)$ was designated
12 $\{rSlave(M)\} = Pico(M) - \{rMaster(M)\}$ ($\|Pico(M)\| \geq 2$)

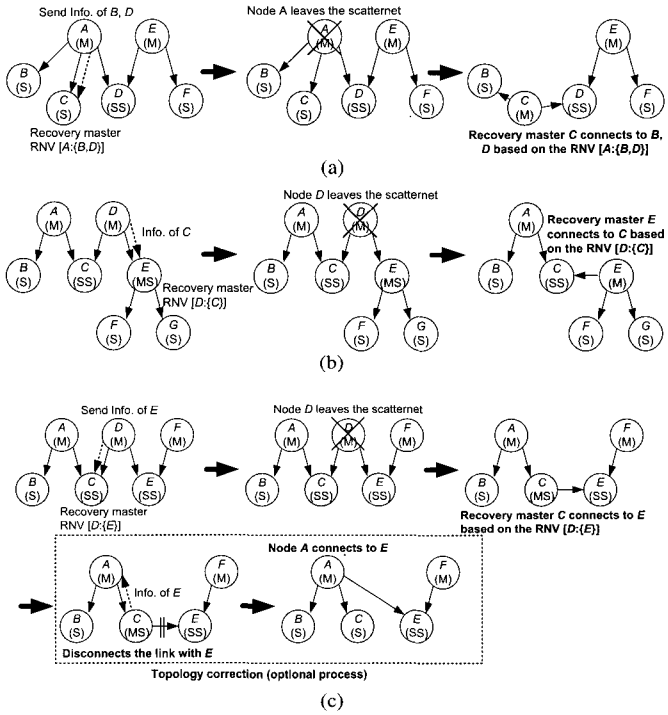

Fig. 4. Example of RNV algorithm in the case of a master node failure: (a) Master A has more than one slave (lines 4–5), (b) master D has more than one MS-bridge (lines 7–8), (c) master D has only SS-bridge (lines 9–10).

In this process, the role of node C is changed into MS-bridge after the reformation. However, MS-bridge may be a new type of node in Fig. 4(c), so that it may distort the existing scatternet topology. Moreover, MS-bridge is not allowed in a few types of topology [4], [13], [14]. In this case, a scatternet topology correction process is necessary after the reformation process in order to maintain the topology.

In Fig. 4(c), the process of the topology correction is shown. In Fig. 4(c), MS-bridge C which is formed after the reformation

Table 4. Algorithm for designating RM/RS in SS-bridge node.

Node A is $Bridge(M_1, M_2)$. node S_1 is one of slave nodes in $Pico(M_1)$ if exists. Node S_2 is one of slave nodes in $Pico(M_2)$ if exists.

```
1  if there is a slave node in Pico(M1) ⋃ Pico(M2)
2    if there is a slave node in both Pico(M1) and Pico(M2)
3      if W(M1) ≥ W(M2)
4        Designate node M2 as rMaster(A)
5        Designate node S1 as rSlave(A)
6      else
7        Designate node M1 as rMaster(A)
8        Designate node S2 as rSlave(A)
9    else if there is a slave node in only Pico(M1)
10     Designate node M2 as rMaster(A)
11     Designate node S1 as rSlave(A)
12   else
13     Designate node M1 as rMaster(A)
14     Designate node S2 as rSlave(A)
15 else
16   if W(M1) ≥ W(M2)
17     Designate node M2 as rMaster(A)
18     Designate node M1 as rSlave(A)
19   else
20     Designate node M1 as rMaster(A)
21     Designate node M2 as rSlave(A)
```

breaks the link with node E and transfers the information of node E to master node A. Then, master node A connects to node E by means of the information from node C. Therefore, after the topology correction, only SS-bridge node E exists and no more MS-bridge node is in the scatternet. The SS-bridge is a node type that is permissible in the scatternet. In this way, the scatternet topology can be maintained.

A.2 Algorithm for the Case of an SS-bridge Node Failure

The algorithm for designating the RM and RS in an SS-bridge node is shown in Table 4. If an SS-bridge node is $Bridge(M_1, M_2)$, the node can receive all the information of two piconets $Pico(M_1)$ and $Pico(M_2)$ from the two master nodes M_1 and M_2. When the information is received, the SS-bridge designates the RM and RS.

In contrast to the case of master node algorithm, a slave node has priority over a bridge node in the RS designation. If there are slave nodes in both piconets $Pico(M_1)$ and $Pico(M_2)$, one of the slave nodes in the piconet which has larger node weight is designated as the RS and the master node in the other piconet as the RM (Table 4 lines 2–8). The purpose for making this kind of designation is to ensure that the node weight, if possible, of each master node is fairly maintained.

In the case where there are slave nodes in only one piconet of $Pico(M_1)$ and $Pico(M_2)$, one of slave nodes is designated as the RS, and the master node in the other piconet which has no slave node is designated as the RM (Table 4 lines 9–14).

In Fig 5, some examples are shown. In Fig. 5(a), node C is an SS-bridge node $Bridge(A, D)$, and there is only one slave

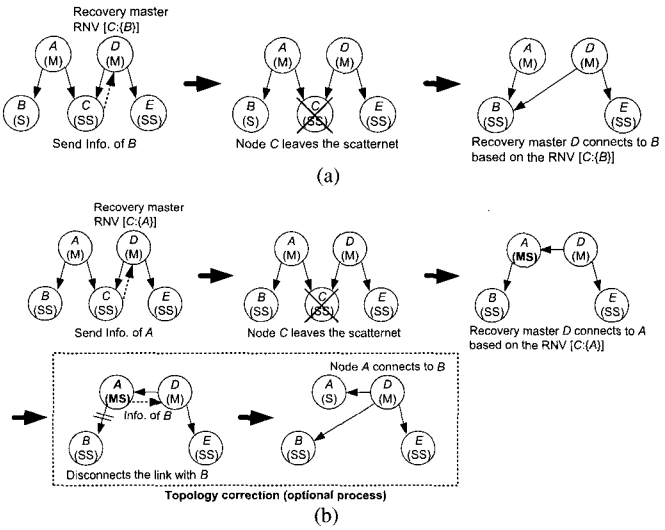

Fig. 5. Example of RNV algorithm in the case of an SS-bridge node failure: (a) There are more than one slave in one of two piconets (lines 1–14), (b) there are only bridges in two piconets (lines 15–21).

node B in $Pico(A)$. Therefore, node B is designated as the RS, and the master node D becomes the RM. Next, node D receives the information of RS B from node C and makes the RNV.

If there is no slave node in both piconets, the master of the piconet which has larger node weight is designated as the RS, and the master of the other piconet as RM (Table 4 lines 15–21) for the purpose of the fairness with respect to node weight of each master node. In Fig. 5(b), there are only bridge nodes in both $Pico(A)$ and $Pico(D)$. After comparing the node weight of both A and D, master A is designated as the RS and master D as the RM.

Here, MS-bridge A is newly born after the reformation. Like in the case of master node, the topology correction may be applied if deemed necessary.

A.3 Algorithm for the Case of an MS-bridge Node Failure

The algorithm for designating the RM and RS in an MS-bridge node is shown in Table 5. If node A is an MS-bridge node $Bridge(M, A)$, node M becomes the master of node A, and node A is the master node in $Pico(A)$.

The first case is that there are only SS-bridge nodes in $Pico(A)$. In this case, the master node of the MS-bridge node A is designated as the RM and all members in $Pico(A)$ as the RS (Table 5 lines 1–2). Some examples are shown in Fig. 6(a). In Fig. 6(a), there is a MS-bridge node D and there are only SS-bridge nodes C and E in $Pico(D)$. The MS-bridge D designates node G which is the master node of itself to the RM, and all the members in $Pico(D)$ as the RS. Therefore, the information of C and E is transferred to the RM G.

The second case is that there are slave or MS-bridge nodes in $Pico(A)$. In this case, two pairs between the RM and RS are needed. The first RM node is always the master of MS-bridge node A. The first RS node which is connected by the first RM is designated among slave or bridge nodes in $Pico(A)$. Here, the first RS node plays the role of the second RM node as well. Therefore, the node is operated as MS-bridge which has two roles of both the first RS and the second RM after the

Table 5. Algorithm for designating RM/RS in MS-bridge node.

Node A is $Bridge(M, A)$, node M is master of node A, node B is a slave node in $Pioc(A)$
1 if all nodes in $Pico(A)$ satisfies $W(B) = \sqrt{8}$ (SS-bridge)
2 Designate node M as $rMaster(A)$ and $\{rSlave(A)\} = Pico(A)$
3 else
4 if there is a node B which satisfies $W(B) = \sqrt{2}$ (slave)
5 iDesignate node B as second $rMaster(A)$
6 Second $\{rSlave(A)\} = Pico(A) - \{rMaster(A)\}$
7 Designate node M as first $rMaster(A)$ and first $rSlave(A) = B$
8 else there is a node B which satisfies $W(B) \geq \sqrt{9}$ (MS-bridge)
9 Designate node B as second $rMaster(A)$
10 Second $\{rSlave(A)\} = Pico(A) - \{rMaster(A)\}$
11 Designate node M as first $rMaster(A)$ and first $rSlave(A) = B$

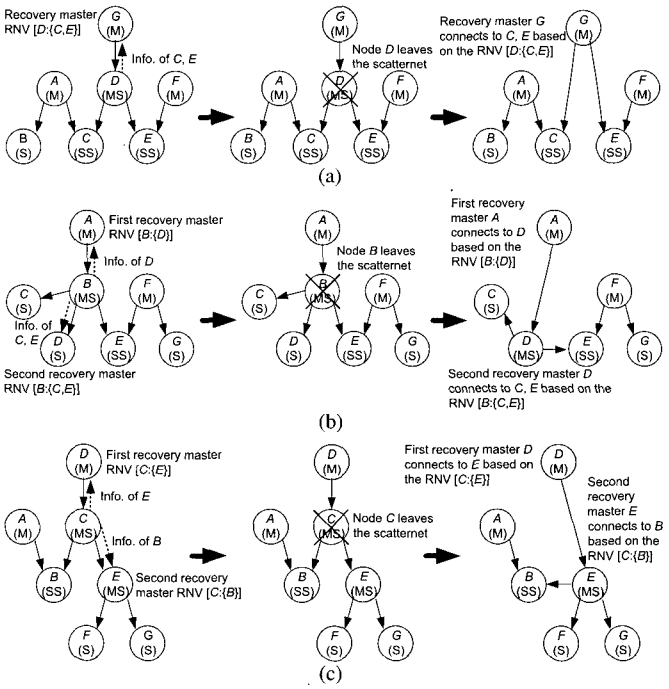

Fig. 6. Example of RNV algorithm in the case of an MS-bridge node failure: (a) MS-bridge D has only SS-bridges (lines 1–2), (b) MS-bridge B has more than one slave (lines 4–7), (c) MS-bridge C has more than one MS-bridge (lines 8–11).

reformation. The rest of members in $Pico(A)$ except the second RM node become the second RS nodes which are connected to by the second RM.

For the second RM designation, a slave node has priority over an MS-bridge node as in the case of master node in Table 3. If there are more than one slave node in $Pico(A)$, one of the slave nodes is designated as the second RM, and the rest in $Pico(A)$, except the second RM, are designated as the second RS (Table 5 lines 4–7). In Fig. 6(b), node B is an MS-bridge node $Bridge(A, B)$. Node A, which is the master node of the MS-bridge, is designated as the first RM. In $Pico(B)$, there is a slave node D and this slave node D is designated as the first RS and as the second RM simultaneously. The rest of $Pico(B)$ (i.e., nodes C and E) is designated as the second RS.

If there are only MS-bridge nodes in $Pico(A)$, one of the MS-bridge nodes is designated as the first RS and the second RM. The remaining process is the same as the case where more than one slave node exists (Table 5 lines 8–11). An example is shown in Fig. 6(c). In Fig. 6(c), there is an MS-bridge node E in $Pico(C)$. Therefore, the MS-bridge node E is designated as the second RM, and for the rest of $Pico(C)$ (i.e., node B) is designated as the second RS. The remaining process is the same as shown in Fig.6(b).

A.4 Node Functions for Implementing RNV Algorithm

For implementing the RNV algorithm which was described in Sections IV-A.1, A.2, and A.3, there are some functions that are needed according to the node types. Here, the functions which must be implemented according to node types such as master, SS-bridge, MS-bridge, and RM are summarized below.

1. Master

(1) When a node becomes master, the node designates the RM and RS based on the algorithm in Table 3, and transfers the information of the RS to the RM. This process is repeated whenever the piconet, managed by the master node, is changed.

(2) If there is an SS-bridge node connected to the master node, the master node transfers the information of the total piconet members to the SS-bridge node.

(3) If the information of the RS is received from a bridge node connected to the master node, the master node makes the RNV for the bridge node. When one of bridge nodes connected to the master node leaves the networks, the master node checks the RNV. If the RNV for the bridge node is found, the master node connects to the RS saved in the RNV.

2. SS-bridge

(1) When a node becomes SS-bridge, the SS-bridge node can receive the information of the two piconets from the two master nodes connected to it. From this information, the RM and RS are designated based on the algorithm in Table 4. Then, the SS-bridge transfers the information of the RS to the RM.

3. MS-bridge

(1) When a node becomes MS-bridge, the node designates the RM and RS based on the algorithm in Table 5, and transfers the information of the RS to the RM. This process is repeated whenever the piconet, managed by the MS-bridge node, is changed.

(2) Perform the functions (2) and (3) of master.

4. Recovery master

(1) Whenever a neighboring node leaves the networks, a RM node checks the RNV. If the RNV for the failed node is found, the RM connects to the RS saved in the RNV.

B. Entry Node Algorithm

For the reformation when a node enters a scatternet, an entry node algorithm is proposed. The reformation algorithm for

Table 6. Algorithm for designating an entry node.

Node Y is one of nodes in $Pico(X)$
1 if there is a node Y which satisfies $W(Y) = \sqrt{2}$ (slave),
2 node Y is designated as an entry node
3 else (There are only bridge nodes in $Pico(X)$)
4 if $\|Pico(X)\| = 1$,
5 node X is designated as an entry node

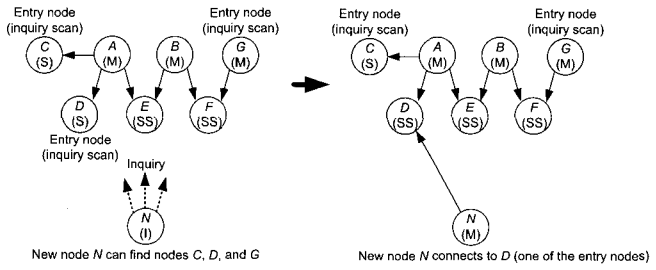

Fig. 7. Example of nodes entering the scatternet.

nodes entering a scatternet is relatively simple compared with the reformation algorithm for nodes leaving a scatternet. In this reformation algorithm, some nodes in the scatternet are designated as entry nodes. Then, the entry nodes are set into inquiry scan states. When a new node conducts inquiry process in order to enter the scatternet, the node will find only entry nodes. Therefore, the new node will connect to one of the entry nodes. As shown above, the entry node algorithm is very simple and can be applied to most types of the scatternet regardless of the topology. In addition, if a new node selects an entry node properly in the connection process, the scatternet topology can be also maintained.

B.1 Algorithm for Designating Entry Nodes

Entry nodes are designated based on the algorithm shown in Table 6. As one of the entry nodes is connected to as slave by a new node, the degree of entry nodes increases by one after the new node enters the scatternet. The degree of bridges nodes may be 3 if the bridges nodes are designated as the entry nodes. Therefore, bridge nodes are not designated as entry nodes, and they are designated among slave or master nodes.

If $Pico(X)$ has at least one slave node, the slave node is designated as the entry node (Table 6 lines 1–2). If $Pico(X)$ has no slave node and only one bridge node, node X which is a master of $Pico(X)$ is designated as the entry node (Table 6 lines 3–5). If all piconets in a scatternet have no slave node, and only bridge nodes, there will be at least one piconet which has only one bridge node.

More than one entry node can be designated in one scatternet. All entry nodes are set into inquiry scan states so that the new node can find the entry nodes.

B.2 Connection Establishment Scheme

A new node that wishes to enter the scatternet finds the entry nodes in the inquiry process. As more than one node can be designated as the entry node, the new node can find more than one node.

However, the new node cannot connect to all of the nodes found in an inquiry process. The new node should check if the nodes found support the scatternet reformation, because it is possible that the new node may connect to other node which is not in the scatternet. This checking process is conducted using service discovery protocol (SDP) [1], [2].

In the SDP process, the node weights of the entry nodes can be sent to the new node. Using the node weights of the entry nodes, the new node can select one of entry nodes with the lowest node weight. Here, a new node may select only slave nodes or only master nodes in order to retain the scatternet topology after the reformation process. After that, the new node connects to the selected node, and become a member of the scatternet.

Fig. 7 shows an example of the entry node algorithm. There are two slave nodes C and D in $Pico(A)$, and a master node G whose $|Pico(G)|$ is 1. From the algorithm shown in Table 6, nodes C, D, and G are designated as entry nodes, and set into inquiry scan states. A new node N can find only entry nodes. Therefore, node selects one of entry nodes with the lowest node weight, namely node D (or node C), and connects to it.

V. HARDWARE EXPERIMENTS AND RESULTS

Hardware experiments were performed using a real commercial Bluetooth module to evaluate the performance of the proposed reformation algorithm, which was described in Section IV. In addition, some problems in the hardware operation were solved. Four parameters: reformation setup delay, reformation setup probability, number of messages and data transfer rate, were considered to evaluate the performance of the algorithm. These parameters have been generally used for evaluating the performance in past researches [4], [11]. The reformation setup delay, the reformation probability and the data transfer rate were taken from the experiments as discussed in Sections V-C and V-D. The number of messages will be briefly described in Section V-B.

A. Experimental Systems and Methods

The system used for the experiments is shown in Fig. 8. The system is composed of PC and PC interface board. The PC is the host which contains higher layer Bluetooth protocol such as L2CAP and SDP in the software. The PC interface board includes the commercial Bluetooth module which contains lower layer Bluetooth protocols such as baseband and link manager in the firmware. The commercial Bluetooth module used in the PC interface board is made by Samsung Electromechanics whose power level is class 2 and compatible to version 1.1 Bluetooth specification.

For the interface between PC and PC interface board, host controller interface (HCI) is used via USB. All controls and data exchanges can be performed by means of HCI [1], [2]. As shown in Fig. 8, there are only lower Bluetooth protocols in the commercial Bluetooth module. Therefore, a higher layer protocol stack such as L2CAP, SDP and scatternet application was implemented in the PC software. The OS in the PC is Windows XP Professional. All nodes exist in the room whose size is 3

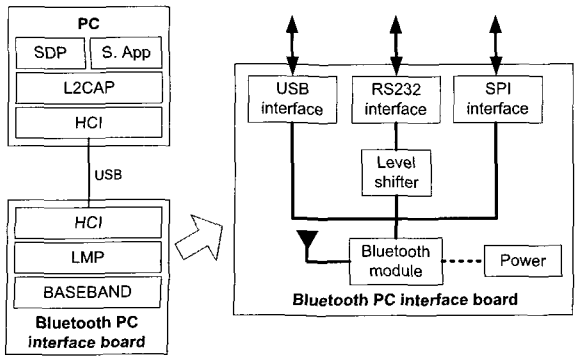

Fig. 8. Structure of hardware and protocol.

m×3 m. Therefore, all nodes exist in the transfer range of class 2 module.

B. Methods of Exchanging Information between Nodes

In order for the RNV algorithm, some pieces of information are exchanged between two nodes. The details of the information are BD_ADDR, clock offset and node weight. The method of exchanging these kinds of information varies according to the node types.

A master node saves BD_ADDR and clock offset of the slave node prior to connection. Therefore, the information of only node weight is needed. However, a slave node does not have any information on a master node prior to connection. After the connection between two nodes, the HCI event named Connection_Complete is generated to both nodes. Because the HCI event contains the BD_ADDR of the remote node, the slave node can obtain the BD_ADDR of the master node from the event.

The other information should be exchanged via ACL link after the connection. The master node transfers the packet which contains its clock offset and its node weight to the slave node. In the opposite direction, the slave node transfers the packet which contains only its node weight. Basically, all nodes can gather the information of neighboring nodes connected to it. The information which is gathered in advance is used for the algorithm for designating RM and RS.

Therefore, two messages are exchanged whenever two nodes are connected. In addition, a master node transfers one message to each of the RM and SS-bridge node whenever the members of the piconet, managed by the master, are changed. The total number of messages varies according to the number of the piconet members and the SS-bridge nodes. On average, the number of the messages which are used for the RNV algorithm in a piconet which has 6 slave nodes is about 20.

C. Experimental Results and Performance Evaluation of the RNV Algorithm

First, the reformation in the case of a master node failure is treated. In this paper, the maximum number of slave nodes in a piconet $s_{\max}$ is set to be 6. In addition, the reformation process in the case of a master failure is not needed when only one node except the master exists in the piconet. Therefore, the reformation setup delay was measured by varying the number of piconet nodes not including the master node from 2 to 6.

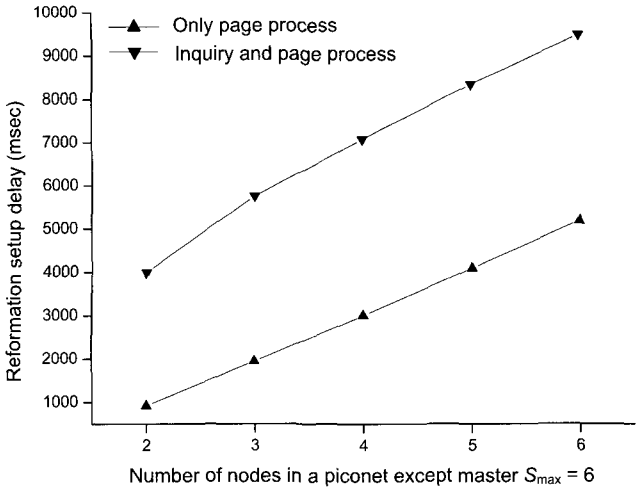

Fig. 9. Reformation setup delay in the case of master node failure.

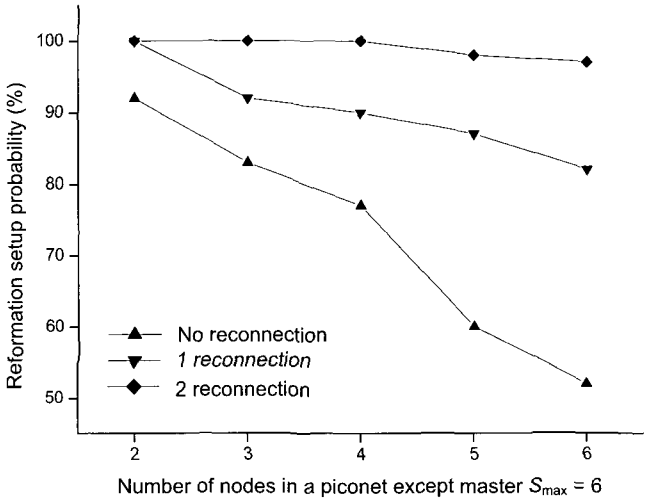

Fig. 10. Reformation setup probability in the case of master node failure.

In Fig. 9, the reformation setup delay is shown for the case of a master node failure. The delay increases linearly as the number of nodes in the piconet managed by the master node increases. If there are N nodes in the piconet, $N-2$ nodes are designated as the RS by the master because the master and the RM nodes are excluded from the RS designation. In the case of the master failure, the RM must repeat the page process $N-2$ times. Therefore, the reformation setup delay has the linear relation to the number of piconet nodes.

In Fig. 9, the results of reformation including both inquiry and page process are shown. The RNV algorithm performed only page process in the reformation process in contrast with other reformation algorithm such as [13] that includes both inquiry and page process. In Fig. 9, the results of the reformation including only page process which is designed in the RNV algorithm have shorter delay, which is 23~60% of the results including both inquiry and page process.

In Fig. 10, the reformation setup probability is shown for the case of master node failure. In contrast to the delay, the refor-

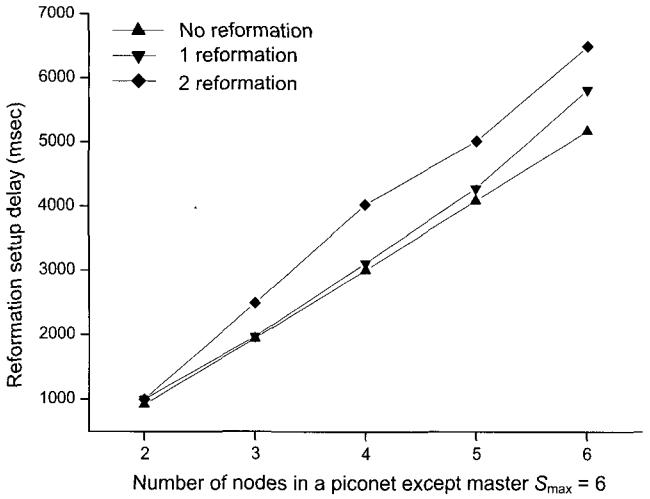

Fig. 11. Reformation setup delay using reconnection algorithm in the case of master node failure.

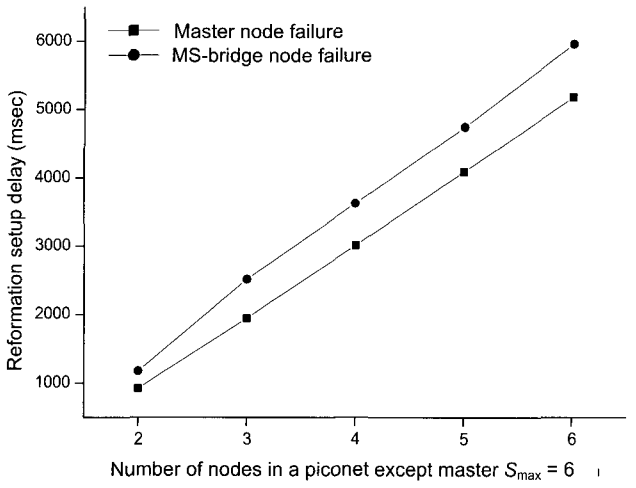

Fig. 12. Reformation setup delay in the case of MS-bridge node failure.

mation setup probability decreases according to the number of nodes in the piconet, managed by the master node, increases. When a Bluetooth node repeats the page process in the real hardware, the timeout error which is called the page timeout error occurs whenever a page process is not performed properly within the established time. The error code is assigned as 0x04 in Bluetooth specification [1], [2]. The frequency of the error occurrence may vary according to Bluetooth hardware, but this generally occurs for most Bluetooth hardware. This timeout error occurs more frequently as the number of page process increases, which in turn decreases the reformation setup probability. In the case of 6 nodes in the piconet, the reformation setup probability is taken as 50%.

For solving this problem, the reconnection algorithm is introduced. Through this reconnection algorithm, a node repeats the page process whenever the page timeout occurs. In Fig. 10, the results of using the reconnection algorithm are shown. Even through a reconnection, the reformation setup probability can be increased remarkably. When the reconnection is allowed up to two times, the reformation setup probability increases to over 97% for 6 piconet nodes.

However, this reconnection process tends to increase the reformation setup delay. So, the reformation setup delay in the cases was measured for two instances: One is the case when one time reconnection is allowed and the other case is when up to two time reconnections are allowed. The results of those cases are shown in Fig. 11. The reformation setup delay is seen to have increased because of reconnection algorithm. However, the results are still 25~76% of the reformation results including both inquiry and page process.

Second, the reformation in the case of MS-bridge failure was also investigated. As mentioned above, at least two links should be recovered after an MS-bridge node failed. One link between first RM and first RS should be made, and more than one link between second RM and second RS should be made. In Fig. 12, the reformation setup delay is longer than the result of master node failure. The reason is the creation of an additional link if formed between first RM and first RS. The delay difference

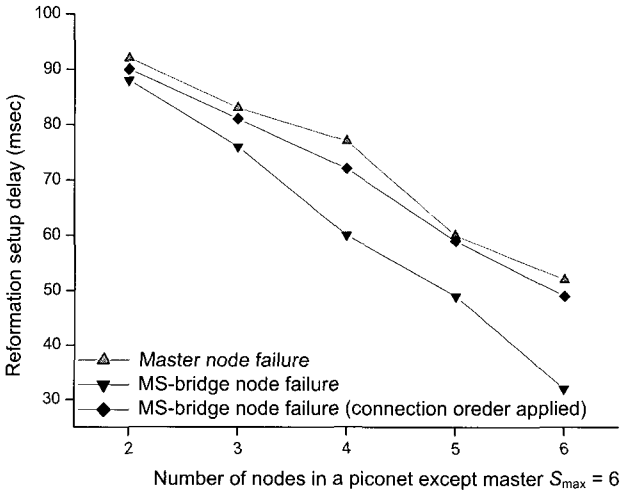

Fig. 13. Reformation setup probability in the case of MS-bridge node failure.

between master and MS-bridge failure is about 600 ms. The difference in the time consumed is due to the additional link formed between first RM and first RS.

However, the reformation setup probability decreases remarkably as the number of nodes in the piconet, managed by the MS-bridge node, increases in comparison with the results of master node failure. The results are shown in Fig. 13. The reason for this is presumably due to the collision in page process.

An example of this case is given in Fig. 6(b). Second, RM D designated by the MS-bridge B must connect to the second RS C and E after the failure of MS-bridge B. At the same time, the second RM D is connected to by the first RM A which is the master of MS-bridge B. If node A tries to connect to the second RM D while the second RM D tries to connect to the second RS C and E, this collision occurs. This collision may occur more frequently as the number of the second RS like nodes C and E increases which in turn may result in the low reformation setup probability as shown in Fig. 13.

In order to solve this problem, the connection order for the

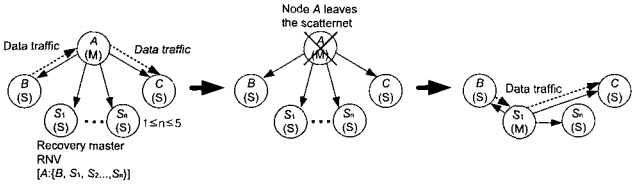

Fig. 14. Scenario of data traffic experiments.

MS-bridge failure is set. In Fig. 6(b), the first RM A connects to the second RM D in advance. The second RM D does not try to connect to the second RS until the connection with first RM A is terminated. After this connection, the second RM D tries to connect to the second RS. Through this simple connection order, the above-mentioned collision can be avoided. The results are also shown in Fig. 13. The reformation setup probability of the case in which the connection order is applied increases remarkably in comparison with the results of the case in which no connection order is applied. The results are almost similar to those of the master node failure. In the results of Fig. 13, no reconnection algorithm as shown in Fig. 10 is applied.

In the case of SS-bridge failure, only one link between the RM and RS which are designated by the failed SS-bridge node must be recovered. This is a simple process and will not be discussed in further detail herein.

In addition, an experiment when there is data traffic was performed. The experimental scenario is shown in Fig. 14. There is a master node A, and slave nodes B, C, and S_n. While nodes B and C are fixed, the number of node S_n varies from 1 to 5. Node S_1, one of nodes S_n, was designated as a recovery master.

Here, node B sends traffic data to node C via master node A. The packet type of traffic data is DM5, and single packet includes user data of 200 bytes. The transfer rate of user data which is sent from node B to node C is set to 93 kbps.

In the middle of sending traffic data from node B to node C, master node A is suddenly powered off, causing data transmission to stop. Then, recovery master S_1 connects to all the remaining nodes after which data transmission is reactivated. Here, the traffic data are sent via node S_1.

As the number of nodes in $Pico(A)$ varies from 3 to 7, the average transfer rate was measured. The total amount of the data traffic was also varied from 100 kbytes to 5 Mbytes. The experimental results are shown in Fig. 15. As $|Pico(A)|$ increases, the average data transfer rate decreases, because the reformation setup delay increases. However, as the total amount of the data traffic increases, the influence of the reformation setup delay is weakened. Therefore, the average transfer rate approaches 93 kbps as the amount of total data traffic increases.

D. Experimental Results and Performance Evaluation of the Entry Node Algorithm

The reformation setup delay when a node enters a scatternet was also measured. The entry node algorithm shown in Table 6, includes an inquiry process. In addition, some data are exchanged for SDP process. Therefore, the delay time for nodes entering a scatternet can be expressed as follows.

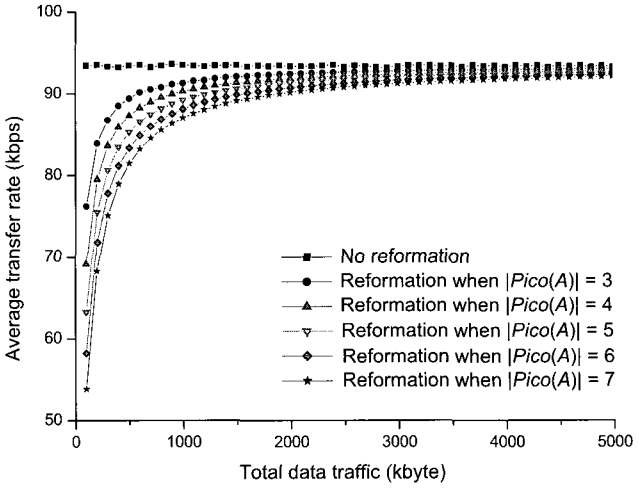

Fig. 15. Average data transfer rate when a master node leaves a scatternet.

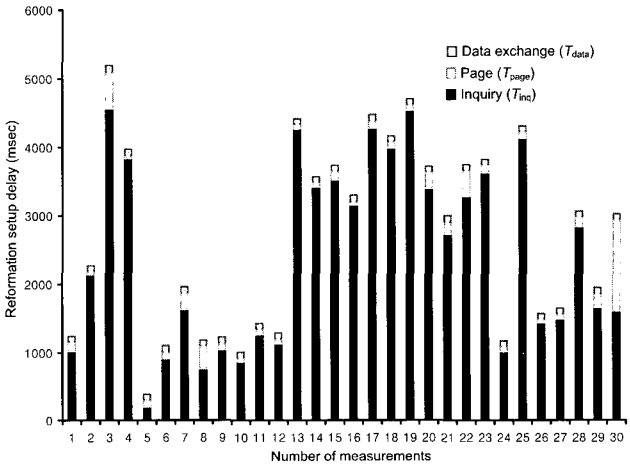

Fig. 16. Reformation setup delay in the case of nodes entering a scatternet.

$$T_{\text{total}} = T_{\text{inq}} + T_{\text{page}} + T_{\text{data}}.$$

In above relation, T_{inq}, T_{page} and T_{data} are the times consumed in an inquiry, a page and a SDP data exchange process, respectively. The experimental results are shown in Fig. 16. In that experiment, the delay time which is consumed for one new node entering a scatternet was measured. The data shown in Fig. 16 are the results of 30 time repetitions of the measurements. These real measurements were taken again 100 times. As shown in Fig. 16, T_{inq} and T_{page} varies considerably. The average value of T_{inq} is 2705 msec, but the standard deviation of T_{inq} is 1311 msec. Therefore, the extent of variation is very large. The average value of T_{page} is 309 msec and the standard deviation is 185 msec. Although the variation of T_{page} is smaller than that of T_{inq}, the extent of variation is still large. However, T_{data} remains almost constant. The average value of T_{data} is 89 msec and the standard deviation is 9 msec.

Because the inquiry process is included when a node enters a scatternet, the delay time becomes longer than that of the RNV

algorithm and the variation of the delay time is large. However, this inquiry process is an essential process for instances where a new node enters a scatternet.

VI. CONCLUSION

In this paper, we proposed a scatternet reformation algorithm which has not been extensively researched in the past. This algorithm is a reformation process that can be applied to the case where more than one node enter or leave the scatternet and are divided into two algorithms, i.e., RNV algorithm and entry node algorithm. While the RNV algorithm is applied to the scatternet reformation when a node leaves a scatternet, the entry node algorithm is applied to the scatternet reformation when a node enters a scatternet.

The RNV algorithm has the following properties. First, this algorithm is operated based on each node. Therefore, it is a general algorithm which can be applied to most types of the scatternet regardless of the topology. In addition, the scatternet topology is retained after the reformation process. Second, most nodes in the scatternet produce RNV which are used for recovering links and reforming networks when one of the neighboring nodes leaves the scatternet. Third, the reformation algorithm includes only a page process. As inquiry process is not required, the reformation setup delay can be reduced remarkably.

The entry node algorithm has the following properties. First, some nodes in the scatternet are designated as entry nodes. These entry nodes are set into inquiry scan states. Therefore, a new node that wishes to enter the scatternet can find the entry nodes and connect to them. Second, the entry node algorithm is very simple and can be applied to most types of the scatternet regardless of the topology. Third, if a new node can select an entry node properly in the connection process, the scatternet topology can also be maintained.

The proposed algorithm was implemented in a real commercial Bluetooth hardware and the experiments were performed to evaluate its performance. In addition, we presented a few solutions to solve certain problems which may occur in hardware implementation. In the experiments, the reformation setup delay, the reformation setup probability and the data transfer rate were measured. In comparison with the case where the inquiry process is undertaken, the RNV algorithm has shown to reduce reformation setup delay and increase reformation setup probability of over 97%. The data transfer rate was hardly influenced even when the total amount of the data traffic increased.

From the above results, we could safely conclude that the Bluetooth scatternet can be applied to dynamic ad-hoc networks like WPAN. Furthermore, the proposed algorithm can be easily used in the existing scatternet formation algorithm by adding some functions, and the concrete and practical performance can be verified by the real commercial hardware implementation and experiments. In addition, the proposed algorithm can also be applied to other ad-hoc networks similar to the Bluetooth.

The applicability of WPAN using the proposed reformation algorithm is currently being tested. This test will evaluate the practical usage of the proposed reformation algorithm more concretely.

ACKNOWLEDGMENTS

Authors gratefully acknowledge the financial support of Brain Korea 21 and ERC of Korean Science and Engineering Foundation.

REFERENCES

[1] Bluetooth SIG, "Specification of the Bluetooth system ver1.1," available at http://www.bluetooth.org, 2001.
[2] Bluetooth SIG, "Specification of the Bluetooth system ver1.2." available at http://www.bluetooth.org, 2003.
[3] L. Ramachandran, M. Kapoor, A. Sarkar, and A. Aggarwal, "Clustering algorithms for wireless ad-hoc networks," in *Proc. 4th DIALM 2000*, Boston, USA, Aug. 2000, pp. 54–63.
[4] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed topology construction of Bluetooth personal area networks," in *Proc. IEEE INFOCOM 2001*, Anchorage, USA, Apr. 2001, pp. 1577–1586.
[5] G. Zaruba and S. Basagni, "Bluetress-scatternet formation to enable Bluetooth-based ad-hoc networks," in *Proc. IEEE ICC 2001*, Helsinki, Finland, June 2001, pp. 273–277.
[6] C. C. Foo and K. C. Chua, "Bluering-Bluetooth scatternets with ring structures," in *Proc. 2nd WOC 2002*, Banff, Canada, July 2002.
[7] C. Petrioli, S. Basagni, and I. Chlamtac, "Configuring blueStars: Multihop scatternet formation for Bluetooth networks," *IEEE Computer*, vol. 52, pp. 779–790, June 2003.
[8] Z. Wang, R. J. Thomas, and Z. J. Haas, "Bluenet-a new scatternet formation scheme," in *Proc. 35th HICSS 2002*, Hawaii, USA, Jan. 2002.
[9] C. Petrioli, S. Basagni, and I. Chlamtac, "BlueMesh: Degree-constrained multihop scatternet formation for Bluetooth networks," *Mobile Networking and Appl.*, vol. 9, pp. 33–47, Feb. 2004.
[10] C. Lay and K. Y. Siu, "A Bluetooth scatternet formation algorithm," in *Proc. 44th GLOBECOM 2001*, San Antonio, USA, Nov. 2001, pp. 2864–2869.
[11] C. Law, A. K. Mehta, and K. Y. Siu, "Performance of a new Bluetooth scatternet formation protocol," in *Proc. 2nd MobiHoc 2001*, Long Beach, USA, Nov. 2001, pp. 182–192.
[12] G. Tan, A. Miu, J. Guttag, and H. Balakrishnan, "Forming scatternets from Bluetooth personal area networks," MIT Technical Report MIT_LCS-TR-826, Oct. 2001.
[13] T. Y. Lin, Y. C. Tseng, K. M. Chang, and C.L Tu, "Formation, routing, and maintenance protocols for the blueRing scatternet of Bluetooths," in *Proc. 36th HICSS 2003*, Hawaii, USA, Jan. 2003, pp. 313–322.
[14] H. Zhang, J. C. Hou, and L. Sha, "A Bluetooth loop scatternet formation algorithm," in *Proc. IEEE ICC 2003*, Anchorage, USA, May. 2003, pp. 1174–1180.
[15] S. Basagni, R. Bruno, and C. Petrioli, "A performance comparison of scatternet formation protocols for networks of Bluetooth devices," in *Proc. 1st PerCom 2003*, Fort Worth, USA, Mar. 2003, pp. 341–350.

Han-Wook Lee was born in Seoul, Republic of Korea, on September 16, 1975. He received the B.Sc. and the M.Sc. in Mechanical & Aerospace Engineering from the Seoul National University, Republic of Korea, in 1999 and 2001, respectively. Now, he is in Ph.D course in Seoul National University. His major interests are ad-hoc network, Bluetooth scatternet, wireless sensor networking, and telemetry system.

S. Ken Kauh was born in Republic of Korea, on April 1, 1956. He received the B.Sc., the M.Sc., and the D.Sc degrees in Mechanical Engineering from the Seoul National University, Seoul, in 1978, 1980, and 1987, respectively. From February 1988 to February 1989, he was a senior researcher in Stanford University, USA. From December 1997 to 1999, he was a guest-professor in UCLA, USA. From March 1989, he has been a professor of department of Mechanical & Aerospace Engineering in Seoul National University. His major interests are telemetry system, wireless sensor networking, and temperature and torque measurement.