노드 형태에 따른 블루투스 스캐터넷 재형성 알고리즘

(A Bluetooth Scatternet Reformation Algorithm based on Node Types)

이 한 욱 † 고 상 근 ††
(Han Wook Lee) (S. Ken Kauh)

요 약 블루투스(Bluetooth)는 휴대폰을 중심으로 하여 다양한 디지털 기기 사이의 무선 인터페이스를 제공할 수 있는 기술로 가능성을 인정받아 왔다. 특히 블루투스에서 지원되는 스캐터넷(Scatternet)이라는 네트워크의 형태는 PAN(Personal Area Network)과 같은 동적인 Ad-hoc 네트워크 환경에서 블루투스가 지원되기 위해서는 필수적인 요소이다. 하지만 기존의 블루투스 스캐터넷 관련 연구들은 노드들이 수시로 추가 혹은 이탈하는 동적인 네트워크 환경을 거의 고려하지 않고 있다. 본 논문에서는 동적인 네트워크 상황을 고려하여 블루투스 스캐터넷 내부에서 노드가 이탈하였을 경우 이를 재형성하기 위한 알고리즘을 제시하였다. 본 알고리즘은 다양한 스캐터넷 형태와 무관하게 적용될 수 있는 범용 알고리즘이며, Inquiry 과정이 생략된 채 Page 과정만으로 빠르게 네트워크가 복구되는 특성을 지닌다. 또 노드의 형태에 따라 복구 마스터/슬레이브를 지정하여 작성되는 복구 노드 벡터(Recovery Node Vector)에 따라 동작한다. 또 본 논문에 제시한 알고리즘은 상용 하드웨어에 적용이 가능한 구체적인 알고리즘으로, 실제 상용 하드웨어 실험을 통해 그 성능을 평가하고 발생되는 문제점을 수정하였다. 실험을 통해 제안한 알고리즘의 재형성 지연 시간은 Inquiry 과정이 포함된 경우의 23~60% 정도로 단축된 결과를 얻었고, 97% 이상의 복구 성공률을 보였다.

키워드 : 블루투스, 스캐터넷, 브릿지 노드, 재형성 알고리즘, 복구 노드 벡터 알고리즘, 노드 행렬, 노드 비중치

Abstract Bluetooth has been reputed as a wireless networking technology supplying ad-hoc networks between digital devices. In particular, bluetooth scatternet is a most essential part for dynamic ad-hoc networks. But past researches on bluetooth scatternet has hardly treated dynamic scatternet environment. In this paper, we proposed a scatternet reformation algorithm for the case that some nodes escape from the scatternet. The proposed algorithm is a general algorithm which can be applied to many types of bluetooth scatternet regardless of the topology. The proposed algorithm has short reformation time delay because the process has only page process (not including inquiry process). The algorithm is operated based on Recovery Node Vector which is composed of Recovery Master and Recovery Slave. In this paper, we performed the real hardware experiments for evaluating the performanec of the proposed algorithm. In that experiments, we measured the reformation time and reformation probability. In comparison with the case including inquiry process, the proposed algorithm had the improvement in reformation time delay and we obtained high success rate over 97%.

Key words : Bluetooth, Scatternet, Bridge Node, Reformation Algorithm, Recovery Node Vector, Node Matrix, Node Weight

1. 서 론

블루투스(Bluetooth)는 휴대폰을 중심으로 하여 다양한 디지털 기기 사이의 무선 인터페이스를 제공할 수 있는 기술로 가능성을 인정받아 왔다. 블루투스는 기본적으로 피코넷(Piconet)과 스캐터넷(Scatternet)이라는

† 학생회원 : 서울대학교 기계항공공학부
equinox2@snu.ac.kr
†† 정 회 원 : 서울대학교 기계항공공학부 교수
kauh@snu.ac.kr
논문접수 : 2004년 8월 3일
심사완료 : 2004년 9월 22일

두 가지 형태의 네트워크를 지원한다. 이중 스캐터넷은 분산적인 ad-hoc 네트워크를 제공할 수 있으며, 이를 이용하면 다수의 디지털 기기 사이의 탄력적이고 확장성 있는 네트워크를 제공할 수 있다. 하지만 블루투스 규격에서는 아직 스캐터넷 관련 형성이나 스케쥴링 등이 구체적으로 명시되지 못한 상태이며[1,2], 이와 관련 다양한 형태의 연구가 진행되어 왔다.

이중 스캐터넷 형성과 관련되어 기존의 연구들은 대부분 노드들이 정적으로 존재하는 상황을 가정하여 출발하는 경우가 많다. 또 노드들이 스캐터넷에 추가되거나 이탈하는 동적인 상황은 고려하지 않거나 재형성 알고리즘을 언급하지 않는 경우가 대부분이다[3-9]. 그러나 블루투스가 스캐터넷을 기반으로 PAN(Personal Area Network) 환경 등에 적용되기 위해서는 동적인 상황에서의 스캐터넷 재형성 알고리즘은 필수적이다.

본 논문에서는 기존에 연구되었던 다양한 스캐터넷의 형태(Topology)와 무관하게 범용으로 적용될 수 있는 스캐터넷 재형성 알고리즘을 제안하였다. 이 재형성 알고리즘은 모든 노드들이 통신 범위 내에 존재하는 스캐터넷 내부에서 하나 이상의 노드가 네트워크를 이탈하였을 경우 적용되는 것으로 다양한 형태의 스캐터넷에 적용을 할 수 있고, 재형성 이후에도 스캐터넷 형태의 일관성을 유지할 수 있다. 또 이 알고리즘의 핵심은 노드의 형태에 따라 복구 마스터/슬레이브를 사전 정보 교환을 통해 지정하고 이를 바탕으로 복구 노드 벡터(Recovery Node Vector)를 작성하여 동작한다는 점이다. 따라서 이 알고리즘을 복구 노드 벡터 알고리즘(Recovery Node Vector Algorithm)이라 명명하였다. 또 Inquiry 과정이 생략되고, Page 과정만으로 이루어지므로 그 지연 시간이 최소화되는데 중점을 두었다. 마지막으로 제안된 알고리즘은 상용 블루투스 하드웨어로 구현이 가능한 구체적 알고리즘으로, 실제 하드웨어 구현을 통한 실험을 수행하고 성능 평가와 관련된 결과를 도출하였다. 본 알고리즘은 바이너리 CDMA와 같이 블루투스와 유사한 Ad-hoc 네트워크 분야에도 응용될 수 있을 것으로 예상된다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 관련 연구를 설명하고, 3장에서 스캐터넷에 관련된 인자를 정형화하고 각 노드의 형태를 분류한다. 4장에서는 각 노드 형태에 따른 복구 노드 벡터 알고리즘에 대해 설명하고, 5장에서는 알고리즘을 기반으로 실제 하드웨어 구현을 통한 실험 결과를 도출하고 성능 평가를 한다. 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

스캐터넷 형성과 관련되어서 다양한 연구가 진행되어 왔다. 스캐터넷은 그 형태에 따라 크게 몇 가지로 분류할 수 있으며, 트리형[5,10-12], 스타형[7], 링형[6,13,14], 메쉬형[8,9] 등이 대표적이다. 스캐터넷의 각 형태는 각자의 장단점을 지니고 있으며, 형태에 따른 성능 평가를 하는 연구도 진행되었다[15]. 하지만 기존의 연구들은 노드의 출입이 없는 정적 네트워크 환경을 가정한 상태에서 출발하거나, 노드의 출입 시 재형성 알고리즘에 대해서는 언급이 없는 경우가 대부분이다[3-9].

또 동적인 노드 상황에서의 스캐터넷 재형성 알고리즘을 연구한 경우에도 대부분 가능성 정도만을 간단하게 언급하는데 그치고 있다[14]. [10,11]에서는 마스터(Master), 슬레이브(Slave), 브릿지(Bridge) 등의 노드의 형태에 따라 노드 이상 시 복구 알고리즘을 간략하게 언급하고 있다. [12]에서는 프로토콜 자체적인 Healing 기능을 구현하여, 동적인 노드 상황에서도 적용할 수 있는 가능성을 제시하고 있다.

[13]에서는 특정 노드의 스캐터넷 이탈시 이를 복구하기 위한 재형성 알고리즘을 비교적 구체적으로 제시한 바 있다. 이 알고리즘은 DIAC(Dedicated Inquiry Access Code)를 이용한 Inquiry 과정을 기반으로 재형성이 이루어지며, 비교적 간단한 알고리즘으로 구현이 가능하다는 장점이 있다. 하지만 재형성 과정에 Inquiry 과정이 포함되므로 재형성 지연 시간이 비교적 길다는 단점이 있다. 또 두 개 이상의 노드가 동시에 제거되었을 경우 네트워크의 형태가 이상적으로 복구될 수 없다는 문제점도 지니고 있다[16].

3. 스캐터넷의 노드 형태 분류

본 논문에서 제안한 복구 알고리즘은 노드의 형태에 따라 적용이 된다. 따라서 다양한 스캐터넷의 형태에 적용되고 있는 노드의 형태를 분류하였다. 또 각 노드의 속성을 정량적으로 분류하기 위하여 노드 행렬(Node Matrix)과 노드 비중치(Node Weight)를 도입하였다.

3.1 스캐터넷의 기본 인자

블루투스의 스캐터넷은 피코넷의 집합으로 이루어진다. 피코넷은 하나의 마스터 노드 당 7개까지의 슬레이브 노드가 연결될 수 있는 소규모 네트워크의 단위이다. 하나의 피코넷 내부에서 마스터 노드는 TDD(Time Division Duplex) 기반의 시간 슬롯 및 주파수 호핑(Frequency Hopping) 채널 등을 슬레이브 노드들과 동기화하고, 링크 및 노드의 상태 등을 관리한다[1,2]. 노드 X가 마스터인 피코넷을 $Pico(X)$로 나타내고, 이것은 그 피코넷 내부에 소속된 슬레이브 노드의 집합을 의미한다. 그림 1에는 3개의 피코넷이 존재하고, 아래와 같이 나타낼 수 있다.

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

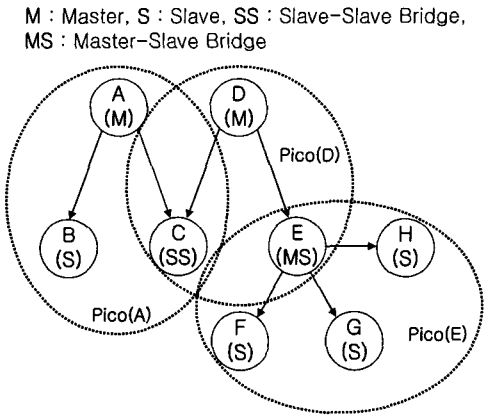

그림 1 피코넷과 스캐터넷

또 피코넷 내부의 슬레이브 노드의 개수를 $|Pico(X)|$라고 할 수 있으므로 아래와 같은 식이 성립한다.

$$|Pico(X)| \le s_{max}$$

여기서 s_{max}는 하나의 피코넷에 포함될 수 있는 슬레이브 노드 개수의 최대값을 의미한다. 블루투스 규격상 s_{max}는 최대 7까지 될 수 있다. 본 논문에서는 s_{max}는 6으로 한다.[1)]

또 마스터 노드가 $M_1, M_2, \cdots, M_n$인 n개의 피코넷으로 구성된 스캐터넷을 아래와 같이 나타내기로 하자.

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

이제 그림 1의 스캐터넷은 다음과 같이 나타낼 수 있다.

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

또 d_{scat}을 스캐터넷의 차원(Dimension)이라고 하면

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

으로 나타낼 수 있다. 즉 d_{scat}는 스캐터넷에 포함된 피코넷의 개수에 해당한다.

다수의 피코넷이 모여 스캐터넷을 구성하기 위해서는 피코넷 간을 연결하는 노드가 필요한데, 그 노드를 브릿지(Bridge)라고 하다. 즉 브릿지 노드는 보통 2개의 피코넷에 동시에 소속되어 각각에 대해 마스터와 슬레이브로 동시에 동작하거나, 두 피코넷에 대해 모두 슬레이브로 동작할 수 있다. 전자를 마스터-슬레이브 브릿지, 후자를 슬레이브-슬레이브 브릿지라고 한다.[2)]

두 개의 피코넷 $Pico(X)$와 $Pico(Y)$ 사이의 브릿지 노드를 $Bridge(X, Y)$라고 하면

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

와 같이 나타낼 수 있다. 그림 1의 스캐터넷에 대한 브릿지 노드는 아래와 같이 나타낸다.

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

또 SS-브릿지와 MS-브릿지로 구분할 수 있는 기준은 다음과 같이 정리할 수 있다.

if $Bridge(X, Y)$ is X or Y, then Master-Slave Bridge.

else Slave-Slave Bridge

표 1 노드의 속성의 결정하기 위한 인자들

Measure	Symbol	Details
Degree of Node	d	하나의 노드가 속한 피코넷의 개수
Role of Node	r	마스터 노드의 경우 1, 슬레이브 노드의 경우 0
Number of Link	l	하나의 노드에 연결된 링크의 개수

표 2 노드의 종류에 따른 노드 행렬 및 노드 비중치

Node Type	$N(X)$	$w(X)$
Master	$[11l]^T$	$\sqrt{2+l^2}\ (l \ge 1)$
Slave	$[101]^T$	$\sqrt{2}$
SS-Bridge	$[202]^T$	$\sqrt{8}$
MS-Bridge	$[21l]^T$	$\sqrt{5+l^2}\ (l \ge 2)$

3.2 노드의 형태 분류

스캐터넷 내부를 구성하는 노드의 형태는 마스터, 슬레이브, MS-브릿지, SS-브릿지의 4가지이다. 일반적으로 SS-브릿지와 MS-브릿지가 모두 사용되는 것이 대부분이지만, [4,13,14]의 경우 SS-브릿지만 사용되며, [6,15]의 경우 MS-브릿지만 사용된다.

본 논문에서는 스캐터넷을 구성하는 노드의 형태를 정량적으로 구분하기 위한 인자를 표 1과 같이 정하였다. 여기서 하나의 노드에 연결되는 링크의 개수는 아래 식을 만족한다.

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

즉 하나의 노드는 최대 $r(s_{max} - 1) + d$개까지의 링크를 만들 수 있다. 또 본 논문에서는 표 1의 세 가지 인자를 이용하여 아래와 같은 노드 행렬(Node Matrix)을 정의하였다.

$$N(X) = \begin{bmatrix} d \\ r \\ l \end{bmatrix}$$

1) s_{max}가 7이면 복구 노드 벡터 알고리즘의 일부가 정상적으로 동작을 못 할 수 있다.

2) 마스터-마스터 브릿지의 경우 하나의 노드에 걸리는 부하가 크고, 실제 하드웨어 구현의 어려움이 많아 거의 사용되지 않고 있다. 편의를 위해 마스터-슬레이브 브릿지를 MS-브릿지, 슬레이브-슬레이브 브릿지를 SS-브릿지로 나타내기로 한다.

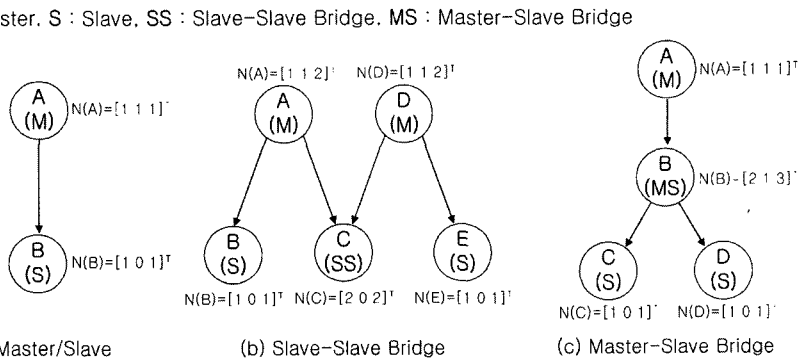

그림 2 스캐터넷 내의 노드의 형태 분류 및 노드 행렬

그림 2에는 각 노드에 대한 노드 행렬을 나타내었다. 또 노드의 비중치(Node Weight)를 $w(X)$라고 하면 아래와 같이 정의하였다.

$$w(X)=|N(X)|$$

w는 각 노드가 소속된 피코넷의 개수, 노드의 역할, 연결된 링크의 개수 등으로 노드에 걸리는 부하를 나타내는 값이며, w값이 큰 노드일수록 그 노드가 스캐터넷을 이탈했을 경우 전체 피코넷에 큰 영향을 주게 된다. 즉 마스터 혹은 브릿지일수록, 노드와 연결된 링크의 개수가 많을수록 노드 비중치가 크게 된다.

일반적으로 d값이 큰 노드는 그 노드에 걸리는 부하가 커지게 되므로 그 노드에서 병목현상이 발생할 수 있다[11]. 따라서 대부분의 기존 연구들에서는 d값을 2 이하로 제한하고 있다[6,10,11,13,14]. 본 논문에서도 d값이 2이하인 경우만 고려하기로 한다. 노드의 종류에 따른 노드 행렬 및 노드 비중치는 표 2에 나타내었다.

4. 노드의 형태에 따른 복구 노드 벡터 알고리즘

스캐터넷과 같이 분산적 네트워크 구조에서는 하나 이상의 노드가 스캐터넷을 이탈하게 되면 전체 네트워크 구조에 이상이 발생하게 된다. 이러한 이상 정도는 노드 비중치가 클수록 더해진다. 특히 스캐터넷 내에서 노드의 출입이 잦은 동적인 환경의 경우 노드 이탈 시 네트워크 구조를 빠른 시간에 복구할 수 있는 여부가 중요한 문제가 된다. 또 PAN과 같은 환경에서는 사용자의 조작 혹은 배터리 문제 등으로 전원 공급이 차단되어 노드 자체가 비정상적으로 종료되어 네트워크 구조에 이상이 발생할 수 있다.

본 논문에서는 이러한 모든 경우에 적용되어 스캐터넷 구조를 복구하고 재형성 하기 위한 복구 노드 벡터(Recovery Node Vector) 알고리즘을 제안하였다. 4.1에는 복구 노드 벡터 알고리즘의 개요를 설명하고, 4.2에서는 노드의 형태별로 적용 알고리즘을 설명하였다. 또 4.3에서는 실제 알고리즘을 구현하기 위한 사항을 정리하였다.

4.1 복구 노드 벡터 알고리즘의 개요

본 논문에서는 스캐터넷 내에서 하나 이상의 노드가 스캐터넷을 이탈 시, 스캐터넷을 재형성하기 위한 복구 노드 벡터 알고리즘(Recovery Node Vector Algorithm)을 제안하였다. 본 알고리즘은 아래와 같은 특징을 지닌다.

(1) 모든 노드들이 블루투스의 통신 범위(Radio Coverage)[3] 내에 존재하는 스캐터넷에 적용된다.

(2) 트리형, 스타형, 메쉬형, 링형 등 스캐터넷의 형태와 상관없이 노드 개별적으로 적용될 수 있는 범용 알고리즘이다. 또 재형성 후에도 기존의 스캐터넷 형태를 일관성 있게 유지할 수 있다.

(3) 스캐터넷 내부의 하나 이상의 노드가 정상적 혹은 비정상적 이탈의 경우에 모두 적용될 수 있다.

(4) 마스터, 슬레이브, MS 브릿지, SS 브릿지의 네 가지 노드 형태를 기반으로 적용된다.

(5) 재형성 시간을 최소화하기 위하여 Inquiry 과정 없이 Page 과정만으로 빠른 시간 재형성이 가능하다.

(6) 두 개 이상의 노드가 동시에 스캐터넷을 이탈한 경우에도 적용 가능하다.

그림 3에는 본 논문에서 제안하는 복구 노드 벡터 알고리즘의 개략적인 설명을 나타내었다. 그림 3의 (a)에는 3개의 노드 A, B, C가 존재하며, 각 노드들은 순서대로 마스터, MS-브릿지, 슬레이브 노드이다. 3개의 노드는 각각 링크 형성 과정에서 서로의 노드 정보를 교환한다. 이때 교환되는 노드 정보는 BD_ADDR, 클럭 오프셋(Clock Offset), 노드 비중치로 이루어진다. 여기

3) 블루투스의 통신 범위는 블루투스 모듈의 파워 클래스에 따라 달라진다[1,2].

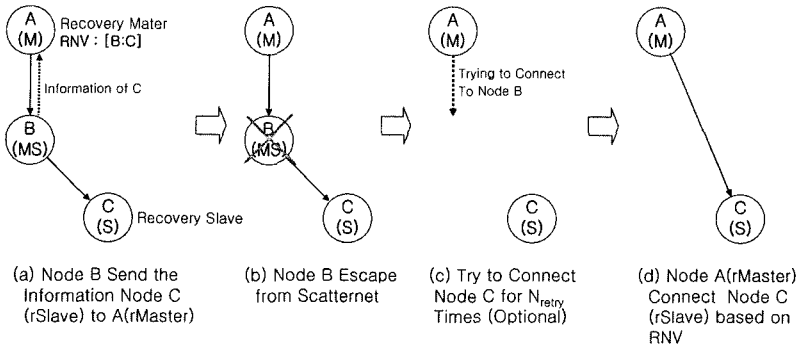

그림 3 복구 노드 벡터 알고리즘의 개요

서 타 노드에 대한 BD_ADDR과 클럭 오프셋을 얻으면 Inquiry 과정 없이 바로 Page 과정을 통한 링크 형성이 가능하다.[4)]

링크 형성 후 각 노드들은 자신이 스캐터넷 이탈 시 재형성을 위해 이웃하는 노드 중에서 복구 마스터(Recovery Master)와 복구 슬레이브(Recovery Slave)를 지정한다. 특정 노드가 스캐터넷을 이탈하였을 경우 그 노드가 사전에 지정한 복구 마스터와 복구 슬레이브 사이에 링크가 형성되어 스캐터넷을 재형성한다. 이러한 복구 마스터와 복구 슬레이브의 선정은 사전에 교환된 노드 비중치를 기반으로 이루어진다.

노드 X의 스캐터넷 이탈 시 스캐터넷 재형성을 위해 지정된 복구 마스터와 복구 슬레이브를 각각 $rMaster(X)$와 $rSlave(X)$라고 하면 그림 3의 (a)의 경우는 아래와 같이 나타낼 수 있다.

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

복구 마스터/슬레이브의 선정을 마치면 복구 슬레이브의 정보를 복구 마스터에게 전달한다. 이때 복구 마스터는 전달받은 정보를 바탕으로 복구 노드 벡터(Recovery Node Vector)를 작성한다. 만약 이웃하는 노드 X로부터 복구 마스터로 지정된 노드가 복구 슬레이브 Y에 대한 정보를 전달받았다면 복구 노드 벡터는 아래와 같이 나타내기로 한다. 아래에서 복구 슬레이브는 두 개 이상이 존재할 수 있다.

Recovery Slave Y from Node X : $[X:\{Y\}]$

그림 3의 (a)에서 노드 B는 자신의 스캐터넷 이탈 시 스캐터넷 재형성을 위한 복구 마스터와 복구 슬레이브를 각각 A와 C로 지정한다. 이후 노드 A는 노드 B로부터 노드 C의 정보를 전달받고, 복구 노드 벡터를 $[B:\{C\}]$로 작성한다.

(b)와 같이 노드 B가 스캐터넷을 이탈하게 되면 노드 A는 복구 노드 벡터에 의해 노드 C와 연결하여 스캐터넷을 재형성한다. 이 과정에서 만약 노드 B가 비정상적으로 스캐터넷을 이탈한 경우에는 (c)와 같이 B에 대한 재연결 시도를 N_{retry}번 시도한 후 B와의 연결이 복구되지 않으면, C와의 연결을 시도하는 알고리즘 적용도 가능하다.

결국 복구 노드 벡터 알고리즘이란 각 노드들이 자신이 스캐터넷을 이탈하였을 경우 네트워크를 재형성하기 위한 복구 마스터/슬레이브 노드를 주변의 노드 중에서 사전에 지정하여 실제 노드가 스캐터넷을 이탈하면 네트워크를 빠르게 복구하고 재형성하는 알고리즘이다. 따라서 각 노드들은 이웃하는 노드들이 스캐터넷을 이탈하였을 경우 연결하기 위한 노드들의 정보를 벡터 형식으로 저장하게 된다. 특히 그 재형성 과정이 Inquiry 과정이 없이 Page 과정만으로 그 지연 시간을 매우 단축할 수 있다.

4.2 노드의 형태에 따른 복구 노드 벡터 알고리즘

본 논문에서 제안한 복구 노드 벡터 알고리즘은 표 2와 같이 네 가지 노드의 형태에 따라 적용된다. 기본 알고리즘은 4.1에서 소개한 내용과 같지만 각 노드에 따라 복구 마스터와 복구 슬레이브를 예약하는 방식은 차이가 나게 된다.

슬레이브 노드($N(X) = [101]^T$)의 경우 스캐터넷을 이탈하더라도 전체 스캐터넷 구조에 이상을 일으키지 않는다. 그러므로 이 경우는 논의 대상에서 제외하기로 한다.

4.2.1 마스터 노드의 복구 노드 벡터 알고리즘

마스터 노드가 스캐터넷을 이탈하게 되면 그 마스터가 관리하고 있는 하나의 피코넷 전체가 해체되므로 네트워크에 큰 이상을 초래한다. 마스터 노드가 스캐터넷을 이탈하게 되면 그 마스터와 연결되어 있던 모든 슬레이브를 네트워크에 복구해야 하므로, 2개 이상의 복구

4) 블루투스 규격상의 HCI 명령 중 Page를 수행하기 위한 명령어인 Create_Connection이라는 명령어의 입력인자로 BD_ADDR과 클럭 오프셋이 필요하다.

슬레이브가 발생하게 될 수 있다.

마스터 노드가 복구 마스터와 복구 슬레이브를 지정하는 알고리즘은 그림 4에 나타내었다. 복구 슬레이브는 복구 마스터가 지정이 되면 그 노드를 제외한 나머지 피코넷 내부의 노드가 된다.

복구 마스터 지정 알고리즘은 피코넷 내부의 슬레이브 노드 존재 여부에 의해서 결정된다. 피코넷 내부에 슬레이브 노드가 한 개 이상 존재하면, 그 노드를 복구 마스터로 지정한다(그림 4 Line 4-5). 이 알고리즘은 그림 5의 (a)에 나타내었다.

만약 피코넷 내부에 슬레이브가 존재하지 않고, 오직 브릿지 노드만 존재할 경우에는 MS-브릿지의 존재 여

Node M is a Master Node, Node B is one of Slaves 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 Slaves are Bridges, $w(X)\geq\sqrt{8}$),
7 if there is a **Node B** which satisfies $w(B)\geq\sqrt{9}$ (**B** is Master-Slave Bridges),
8 Designate **Node B** as $rMaster(M)$
9 else (All Slaves are Slave-Slave Bridges)
10 Designate one of Nodes in $Pico(M)$ as $rMaster(M)$
11 if $rMaster(M)$ was designated
12 $\{rSlave(M)\}=Pico(M)-\{rMaster(M)\}$ ($|Pico(M)|\geq 2$)

그림 4 마스터 노드에서의 복구 마스터/슬레이브 지정 알고리즘

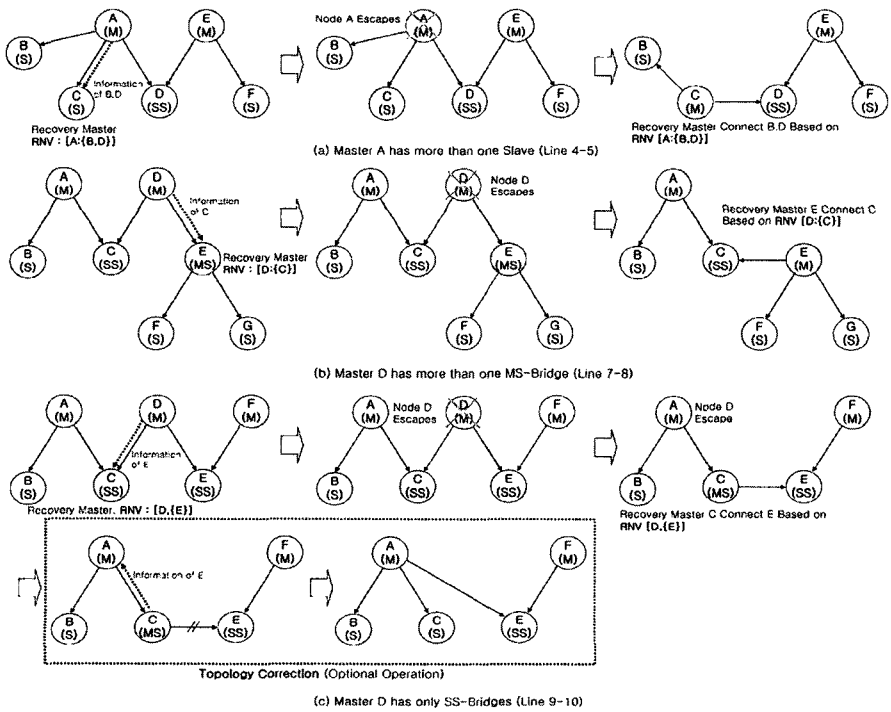

그림 5 마스터 노드 이탈 시 복구 노드 벡터 알고리즘의 적용 예

부에 의해 복구 마스터를 결정한다. MS-브릿지가 존재할 경우에는 그 노드를 우선적으로 복구 마스터로 지정을 한다(그림 4 Line 7-8). 이 알고리즘은 그림 5의 (b)에 나타내었다. *Pico*(*D*)에는 두 개의 브릿지 C, E가 존재하고, 이중 E가 MS-브릿지이다. 그러므로 마스터 노드 D는 E를 복구 마스터로 지정한다.

반면 브릿지 노드가 모두 SS-브릿지일 경우에는 SS-브릿지 중 하나를 복구 마스터로 선정한다(그림 4 Line 9-10). 이 과정은 그림 5의 (c)에 나타내었다. *Pico*(*D*)에는 오직 SS-브릿지인 C, E만 존재하고, 마스터 노드 D는 SS-브릿지 중 하나인 C를 복구 마스터로 지정한다.

다만 이 경우 스캐터넷이 재형성되면 C는 MS-브릿지로 된다. 그림 5의 (c)에서 MS-브릿지는 기존의 스캐터넷에서는 존재하지 않는 노드의 형태이고, 재형성 과정에서 새롭게 생성되었다. 스캐터넷의 형태에 따라 MS-브릿지를 사용하지 않는 경우들이 있어[4,13,14], 재형성 후 MS-브릿지의 생성은 스캐터넷 형태를 변형시킬 수 있다. 그러므로 MS-브릿지를 사용하지 않는 스캐터넷의 경우 재형성 후에 스캐터넷 형태의 일관성을 유지하기 위한 형태 보정(Topology Correction) 작업이 필요하다. 그림 5의 (c)와 같이 재형성을 통해 발생한 MS-브릿지인 노드 C는 슬레이브 노드로 연결된 E와 링크를 종료하고, 마스터 노드로 연결된 A에게 노드 E의 정보를 전달한다. 이후 노드 A는 E와 연결을 하면 MS-브릿지가 사라지고, SS-브릿지만 존재하게 된다. 이렇게 간단한 형태 보정 작업을 통해 스캐터넷 재형성 후에도 그 형태의 일관성을 유지할 수 있다.

4.2.2 SS-브릿지 노드의 복구 노드 벡터 알고리즘

SS-브릿지 노드에서 복구 마스터/슬레이브의 지정 알고리즘은 그림 6에 나타내었다. 한 SS-브릿지가 *Bridge*(M_1, M_2)라면 그 SS-브릿지는 두 마스터 M_1과 M_2, 그리고 *Pico*(M_1)과 *Pico*(M_2)의 정보를 지니고 있다. SS-브릿지는 이 정보를 바탕으로 복구 마스터와 복구 슬레이브를 결정한다.

SS-브릿지의 복구 마스터와 복구 슬레이브의 선정은 슬레이브의 존재 여부에 의해 결정된다. 만약 두 개의 피코넷 *Pico*(M_1)과 *Pico*(M_2) 내부에 슬레이브 노드가 모두 존재한다면 마스터 노드의 비중치가 높은 피코넷의 슬레이브를 복구 슬레이브로, 나머지 피코넷의 마스터를 복구 마스터로 선정한다(그림 6 Line 2-8). 이렇게 노드 비중치가 낮은 마스터를 복구 마스터로 지정하는 이유는 각 피코넷 크기($|Pico(X)|$)의 형평성을 맞추기 위함이다.

반면 *Pico*(M_1)과 *Pico*(M_2) 중 어느 한 피코넷에만

Node A is *Bridge*(M_1, M_2)**, Node S1 is one of Slaves in** *Pico*(M_1) **if exist. Node S2 is one of Slaves in** *Pico*(M_2) **if exist.**

```
1  if there is a Slave in Pico(M1)∪Pico(M2)
2    if there is a Slave 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 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)
10   else
20     Designate Node M1 as rMaster(A)
21     Designate Node M2 as rSlave(A)
```

그림 6 SS-브릿지에서의 복구 마스터/슬레이브 지정 알고리즘

슬레이브가 존재한다면 그 슬레이브를 복구 슬레이브로, 반대편 피코넷의 마스터를 복구 마스터로 선정한다(그림 6 Line 9-14). 그림 7의 (a)에서 노드 C는 *Bridge*(A, D)이며, 슬레이브 노드는 *Pico*(A)에 유일하게 B가 존재한다. 그러므로 노드 B는 복구 슬레이브로 지정되고, 마스터 노드 D가 복구 마스터가 된다.

만약 양쪽 피코넷에 모두 슬레이브가 존재하지 않고, 브릿지 노드만 존재할 경우 두 마스터 M_1과 M_2 중 비중치가 작은 마스터를 복구 마스터로, 나머지 마스터를 복구 슬레이브로 지정한다(그림 6 Line 15-21). 그림 7의 (b)는 *Pico*(A)와 *Pico*(D)에 모두 브릿지 노드만 존재하는 경우이다. 이 경우 두 마스터 노드 A와 D의 비중치를 비교한 후 하나의 마스터 노드 A는 복구 슬레이브로, 반대편 피코넷의 마스터 D는 복구 마스터로 지정된다.

여기서 마스터 노드 이탈 시 재형성 과정과 마찬가지로 재형성 후 노드 A는 기존에 존재하지 않던 새로운 노드 형태인 MS-브릿지가 되었다. MS-브릿지가 허용되지 않는 피코넷의 경우 그림 7의 (b)에 나타난 형태 보정 작업을 통해 MS-브릿지를 제거할 수 있다. 또 두

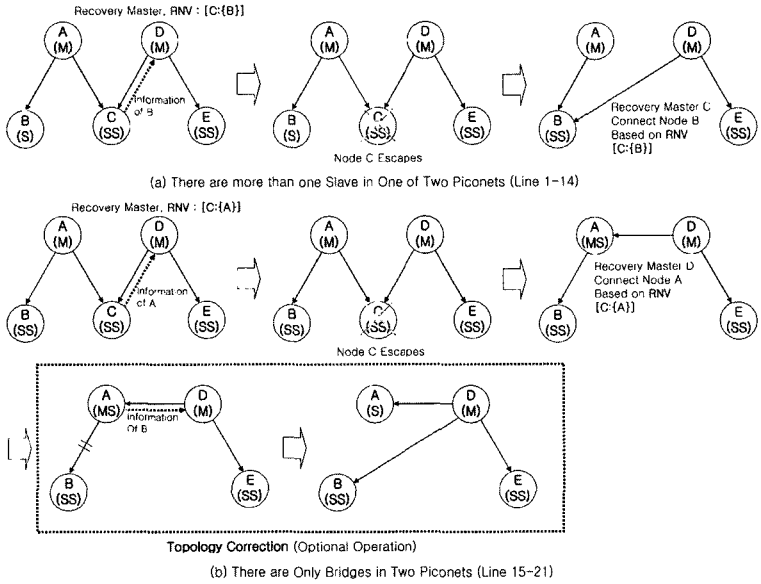

그림 7 SS-브릿지의 스캐터넷 이탈 시 복구 노드 벡터 알고리즘의 적용 예

피코넷의 마스터 노드 중 하나 이상이 MS-브릿지인 경우에는 재형성 과정 후 노드 차원(Degree of Node)이 3인 경우가 발생할 수 있다. 이 경우에도 형태 보정 작업을 수행하면 그 노드의 노드 차원을 2로 낮출 수 있다.

4.2.3 MS-브릿지 노드 이상 시 복구 노드 벡터 알고리즘

그림 8에는 MS-브릿지 노드에서의 복구 마스터와 복구 슬레이브의 지정 알고리즘을 나타내었다. 알고리즘은 A가 *Bridge*(*M*,*A*)인 MS-브릿지일 경우 *Pico*(*A*)에 슬레이브 혹은 MS 브릿지가 존재하는지와 SS-브릿지만 존재하는 경우로 나뉘게 된다. 만약 *Pico*(*A*)에 SS-브릿지만 존재하는 경우에는 A의 마스터인 M이 복구 마스터가 되고, *Pico*(*A*)의 모든 노드들이 복구 슬레이브가 된다(그림 8의 Line 1-2). 그림 9의 (a)에서 노드 D는 MS-브릿지이고, *Pico*(*D*)에는 오직 SS-브릿지인 C, E만 존재한다. 이 경우 노드 D는 자신의 마스터인 노드 G를 복구 마스터로 지정하고, *Pico*(*D*) 전체를 복구 슬레이브로 지정하여 G에게 정보를 전달한다.

그림 8의 알고리즘에서 *Pico*(*A*)에 슬레이브 노드 혹은 MS-브릿지가 존재할 경우는 그림 4의 알고리즘과 유사하게 그 노드를 복구 마스터로 지정하고, 그 노드를 제외한 *Pico*(*A*) 전체의 노드를 복구 슬레이브로 지정한다. 동시에 자신의 마스터인 노드 *M*을 또 다른 복구 마스터로 지정하고, 앞에서 복구 마스터로 지정된 노드를 M에 대한 복구 슬레이브로 지정하여 그 정보를 전달

Node A is *Bridge*(*M*,*A*), ***M* is a Master Node of A, Node B is a Slave in** *Pico*(*A*)

```
1  if there are only SS-Bridges in Pico(A),
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)=√2 (B is Slave),
5      Designate Node B as rMaster(A)
6      {rSlave(A)} = Pico(A) − {rMaster(A)}
7      Designate Node M as rMaster(A) and rSlave(A) = B
8    else there is a Node B which satisfies w(B)≥√9 (B is MS-Bridge),
9      Designate Node B as rMaster(A)
10     {rSlave(A)} = Pico(A) − {rMaster(A)}
11     Designate Node M as rMaster(A) and rSlave(A) = B
```

그림 8 MS 브릿지에서의 복구 마스터/슬레이브 지정 알고리즘

한다(그림 8의 Line 4-7). 그림 9의 (b)에는 *Bridge* (A, B)인 MS-브릿지 B가 존재하고, *Pico*(B)에는 슬레이브 노드인 D가 존재한다. 그러므로 노드 D는 복구 마스터로 지정되고, D를 제외한 *Pico*(B)의 전체 노드가 복구 슬레이브로 지정되어, 그 정보가 D에게 전달된다. 여기서 복구 마스터로 지정된 노드 D는 MS-브릿지의 스캐터넷 이탈 시 *Pico*(B)를 복구하는 역할을 수행한다. 동시에 노드 B는 자신의 마스터인 A를 복구 마스터로 지정하고, 전단계에서 복구 마스터로 지정된 노드 D를 복구 슬레이브로 지정하여 그 정보를 A에게 전달한다. 이후 노드 B가 스캐터넷을 이탈하면 노드 A는 노드 D와 연결하고노드 D는 MS-브릿지가 되는 재형성 과정이 이루어진다.

그림 9의 (c)는 (b)와 유사하지만 *Pico*(C)에 슬레이브가 존재하지 않고, MS-브릿지인 노드 E가 존재하는 경우이다(그림 8의 Line 8-11). 슬레이브 노드 대신에 MS-브릿지인 노드 E가 복구 마스터로 지정된다는 점을 제외하면 (b)와 유사하다.

4.3 알고리즘의 구현

위 알고리즘을 실제로 구현하기 위해서는 노드에 따른 구현 기능을 부여해야 한다. 마스터, SS-브릿지, MS-브릿지, 복구 마스터 등 4.2에서 제시된 노드의 형태에 따른 알고리즘이 적용되기 위해서는 노드별로 구현 되어야 할 사항을 정리하면 다음과 같다.

1. 마스터

(1) 노드가 마스터가 되면 자신의 피코넷 내부의 노드 중 그림 4의 알고리즘에 의거하여, 복구 마스터를 지정한 후 그 노드에게 피코넷 내부에서 자신을 제외한 모든 노드의 정보를 전달한다. 이 작업은 자신의 피코넷 구성이 변경될 때마다 수행한다.

(2) 자신의 피코넷 내부에 SS-브릿지가 존재하면, 그 노드에게 자신의 정보를 포함한 피코넷 내의 모든 노드 정보를 전달한다.

(3) 자신과 연결된 브릿지 노드로부터 복구 슬레이브 정보가 전달되면 복구 노드 벡터를 작성한다. 또 자신과 연결된 브릿지 노드가 스캐터넷을 이탈한 경우, 그 노드에 대한 복구 노드 벡터가 존재하면, 그 벡터에 저장된 복구 슬레이브에 연결한다.

2. SS-브릿지

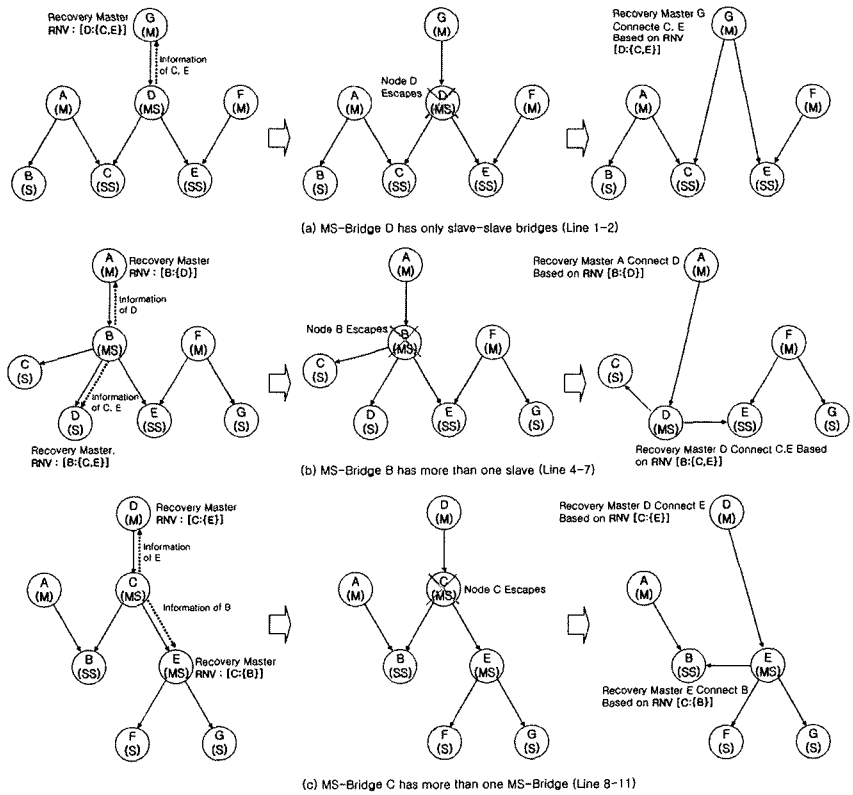

그림 9 MS-브릿지의 스캐터넷 이탈 시 복구 노드 벡터 알고리즘의 적용 예

(1) 자신과 연결된 두 개의 마스터로부터 사전에 각 피코넷의 정보를 전달받는다. 이 정보를 바탕으로 그림 6에서 제시한 알고리즘에 의거하여 복구 마스터와 복구 슬레이브를 지정한다. 지정된 복구 마스터로는 복구 슬레이브의 정보를 전달한다.

3. MS-브릿지

(1) 마스터의 (2),(3) 동작을 동일하게 수행한다.

(2) 자신의 피코넷 내부에서 그림 8의 알고리즘에 의거하여 복구 마스터를 지정하고, 그 노드에게 정보를 전달한다.

4. 복구 마스터

(1) 자신과 연결된 이웃한 노드의 스캐터넷 이탈 시 사전에 그 노드에 대한 복구 노드 벡터가 존재하면, 그 벡터에 포함된 노드로 연결 한다.

위와 같은 각 노드별 작업은 이웃한 노드가 변경될 경우마다 반복된다.

5. 알고리즘의 하드웨어 구현 및 성능 평가 실험

본 논문에서는 4장에서 제안한 재형성 알고리즘의 성능 평가를 위해 직접 하드웨어 구현을 통한 실험을 하였다. 또 실제 하드웨어 구현을 통해 동작 과정에서 발생하는 문제점을 해결하였다. 본 논문에서는 알고리즘 성능 평가를 위한 인자로 재형성 지연 시간, 재형성 확률, 메시지 수의 세 가지를 설정하였다. 이러한 인자들은 기존의 연구들에서 스캐터넷 알고리즘의 성능 평가를 위해 채택되고 있다[4,11]. 이 중 재형성 지연 시간과 재형성 확률은 5.3에서 실험적 결과를 얻었고, 메시지의 수는 5.2에서 간단히 언급하였다.

5.1 실험 장치 및 방법

본 논문에서 하드웨어 구현을 위해 사용된 시스템의 형태는 그림 10과 같다. 시스템은 PC를 호스트로 구성되어 있으며, PC와 연결하기 위한 PC 인터페이스 보드를 제작하였다. PC 인터페이스 보드의 구조는 그림 10과 같고, PC와 인터페이스 보드는 RS-232통신을 통해 115200bps로 연결되었다. 보드 내부에는 1.1버전 블루투스 규격에 상응하는 삼성전기의 BlueSEM-CII Class2 상용 모듈이 사용되었다.

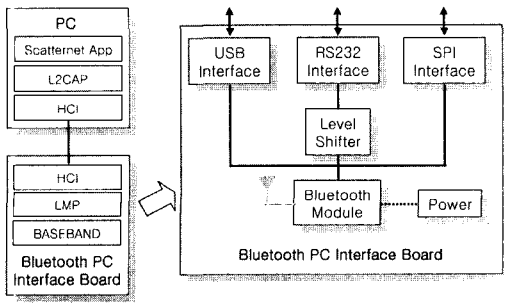

그림 10 하드웨어 시스템 및 프로토콜의 구조

또 그림 10에서 보는 바와 같이 블루투스 모듈 내부에는 베이스밴드(Baseband)와 링크 매니져(LMP)의 프로토콜만 내장되어 있으며, 그 이상의 상위 프로토콜은 PC 내부에 구현하였다. 이렇게 나누어진 상하위 프로토콜은 HCI(Host Controller Interface)로 연결이 된다. 이 HCI를 통해 PC에서 블루투스 모듈의 모든 동작을 제어하고, 데이타를 교환할 수 있다. 또 모든 노드들은 2m×2m의 공간 내부에 설치하여 실험이 이루어 졌다. 이 공간에서는 Class 2 모듈 사용 시 모든 노드가 통신 범위 내에 존재할 수 있다.

5.2 노드 사이의 정보 교환

노드 사이에 연결 후 링크가 형성이 되면 BD_ADDR, 클럭 오프셋, 노드 비중치의 세 가지 정보를 교환해야만 한다. 이 정보 교환 방식은 두 노드 사이의 연결 시 마스터 혹은 슬레이브에 따라 달라진다. 마스터 노드의 경우 슬레이브 노드의 BD_ADDR과 클럭 오프셋의 정보를 기존에 저장하고 있으므로 노드 비중치의 정보만 필요하다. 반면 슬레이브는 마스터 노드에 대한 사전 정보가 없는 경우가 대부분이다. 블루투스는 노드 사이에 ACL 링크가 형성이 되면 마스터와 슬레이브 양쪽 노드에서 Connection_Complete라는 HCI 이벤트가 발생한다. 이 이벤트 내부에는 원격으로 연결된 노드의 BD_ADDR 정보가 포함되어 있으므로 슬레이브 노드는 이 이벤트에서 마스터 노드의 BD_ADDR을 획득할 수 있다. 나머지 정보들은 ACL 링크 형성 후 데이타 교환을 통해 이루어진다. 마스터에서 슬레이브로는 마스터의 클럭 오프셋과 노드 비중치 정보를 포함한 패킷을 전달하고, 슬레이브는 마스터에게 자신의 노드 비중치 정보만 전달하면 된다. 이러한 과정을 통해 각 노드들은 이웃하는 노드들의 정보를 획득할 수 있다.

따라서 노드 사이 링크가 연결될 때마다 2개의 메시지 교환이 이루어지고, 마스터 노드는 자신의 피코넷이 변경될 때마다 복구 마스터와 SS-브릿지에게 정보를 업데이트 해주기 위한 메시지를 한 번씩 보내게 된다. 따라서 재형성을 위해 필요한 메시지 개수는 노드의 개수와 SS-브릿지의 유무에 의해 달라지는데, 6개의 슬레이브를 지닌 하나의 피코넷에서 재형성 알고리즘을 위해 교환되는 메시지의 개수는 약 20개 내외이다.

5.3 실험 결과

그림 11에는 마스터 노드의 스캐터넷 이탈 시 재형성 지연 시간을 나타내었다. 그림 4의 알고리즘에 의하면 마스터 노드는 현재 연결되어 있는 슬레이브 혹은 브릿지 노드의 개수에 따라 재형성 지연 시간이 달라진다. 그림 11에서 보듯이 마스터 노드와 연결된 노드의 개수

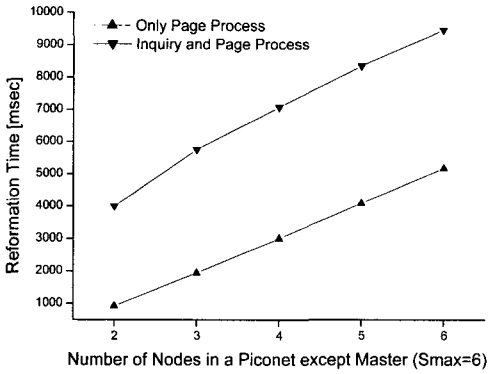

그림 11 마스터 노드 이탈 시 재형성 지연 시간

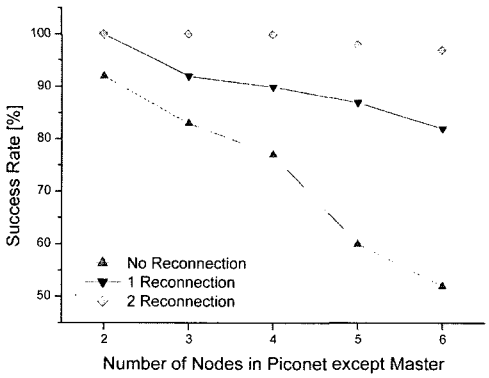

그림 12 마스터 노드 이탈 시 재형성 성공 확률

가 증가할수록 재형성 지연 시간을 길어진다.

그림 11에는 Inquiry 과정과 Page 과정을 동시에 수행할 경우와 Page 과정만 수행할 경우의 결과를 비교하였다. 본 논문에서 제안한 재형성 알고리즘은 사전에 복구할 노드의 정보를 미리 저장하고 있다가 Inquiry과정이 없이 Page 과정만으로 빠른 복구를 수행할 수 있다. 그림 11의 결과를 보면 Page 과정만 수행할 경우 Inquiry 과정이 포함되었을 경우 지연 시간의 23~60% 정도로 단축되는 결과를 얻었다.

그림 12에는 재형성 성공 확률을 나타내었다. 그림 12의 결과에 나타낸 것처럼 실제 하드웨어로 알고리즘을 동작 시키면 마스터 노드가 관리하는 피코넷 내부의 노드 개수가 증가할수록 연결 확률이 낮아지는 결과를 나타내었다. 실제 블루투스 하드웨어는 Page 과정에서 타임 아웃 에러[5]가 발생하는 경우가 있는데, 연결해야할 노드의 개수가 많아질수록 이러한 에러에 의한 연결 확률이 낮아지는 문제점을 나타내었다. 특히 마스터 노드와 연결된 노드의 개수가 6개가 되면 50% 정도로 연결 확률이 낮아지는 결과를 얻었다.

본 논문에서는 이러한 문제점을 해결하기 위하여 재연결 방식을 추가하였다. 즉 Page 과정에서 타임 아웃 에러가 발생하면 다시 연결을 시도하는 방식이다. 그림 12에는 재연결 횟수를 1회, 2회까지 허용할 경우의 연결 확률을 나타내었다. 재연결을 1회 시도할 경우 연결 확률이 상당 부분 높아지는 결과를 얻었으며, 재연결을 2회까지 허용할 경우 피코넷 내부 노드의 개수가 6개가 되더라도 연결확률을 97%이상 결과를 나타내었다. 그런데 재연결을 시도할 경우 그만큼 재형성 지연 시간이 길어지게 된다. 이 결과는 그림 13에 나타내었다. 재연

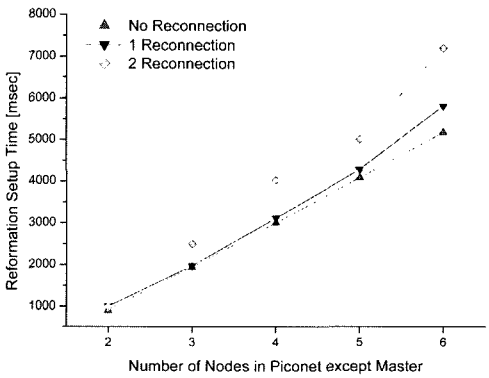

그림 13 재연결 적용 시 재형성 지연 시간

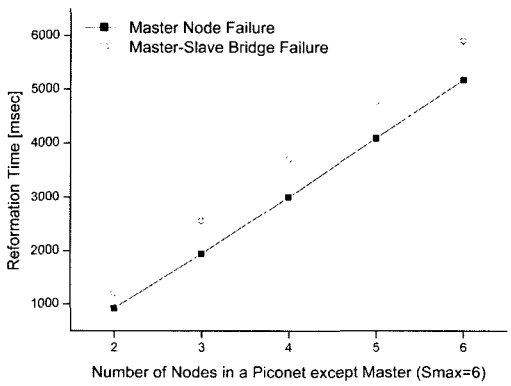

그림 14 MS-브릿지 노드 이탈 시 재형성 지연 시간

결을 허용한 경우는 재연결 허용하지 않을 경우에 비해 최대 2초 정도의 시간 지연이 길어진다. 하지만 이 결과도 그림 11의 Inquiry 과정 지연 시간의 25~76%이다.

그림 14에는 MS-브릿지 노드의 이상 발생 시 재형성 지연 시간을 나타내었다. 4.2.3에서 설명했듯이 MS-브릿지 노드는 스캐터넷 이탈 시 최소 2개 이상의 링크

5) 블루투스 규격 상에는 Page Timeout(에러코드 0x04)이라 하여 Page 과정 시 일정 시간 내에 Page 과정이 이루어지지 못하면 발생하는 에러이다. 실제 상용 블루투스 하드웨어를 통한 Page 과정에서 Page Timeout 에러가 종종 발생한다.

가 복구되어야 한다. 즉 MS 브릿지가 마스터로 관리했던 슬레이브 노드들 사이의 링크를 모두 복구한 후 원래 MS 브릿지의 마스터 노드와 연결이 되어야 한다. 그림 14의 결과를 보면 자신이 관리하는 피코넷 내부에 소속된 슬레이브 노드의 개수가 동일하더라도, MS 브릿지의 마스터와 연결이 되는 시간이 추가가 되므로, 마스터 노드 이상 시 복구 시간보다 평균적으로 600ms의 시간이 더 소요된다. 그런데 그림 15의 결과를 보면 재형성 성공 확률은 마스터 노드일 경우보다 현저하게 떨어지는 결과를 얻었다. 그림 9의 (b)의 경우를 예로 들기로 하자. MS 브릿지인 B의 이탈 시 D는 복구 마스터로 복구 슬레이브 C, E와 링크를 형성해야 한다. 동시에 A는 D를 복구 슬레이브로 링크를 형성한다. 만약 D가 C, E에 대한 Page 작업 중에 A가 D에 대한 Page 작업을 시도하면 충돌이 발생하여 에러가 발생하게 된다. 이러한 에러는 C, E와 같은 복구 슬레이브 개수가 증가할수록 발생할 가능성이 높다. 그러므로 그림 15와 같이 재형성 성공확률이 현저히 떨어지는 결과를 얻었다.

실제 하드웨어 구현 과정에서 발생한 이러한 문제점을 해결하기 위해 MS 브릿지에 대해서는 간단한 링크 연결 순서를 정하였다. 그림 9의 (b)의 경우를 예로 들면 먼저 노드 A가 D와 연결을 한다. D는 자신의 마스터가 된 A가 연결을 완료할 때까지 자신의 복구 슬레이브와의 연결을 시도하지 않다가 자신의 마스터와 연결이 완료되면 그때부터 복구 슬레이브들과의 연결을 시도한다. 이렇게 하면 앞에서와 같은 충돌 현상을 막을 수 있다. 이렇게 복구 순서를 적용하여 실험한 결과도 그림 15에 나타내었다. 그림 15에 의하면 마스터 노드일 경우와 거의 유사한 결과를 나타내었다. 그림 15의 결과에는 그림 13에 나타낸 재연결 방법을 적용하지 않았다.

SS 브릿지의 경우 어떠한 스캐터넷 형태에서도 SS-브릿지의 이탈 시 한 개의 복구 마스터와 복구 슬레이브 사이에 하나의 링크만 연결되면 되므로 본 논문에서 자세한 언급은 생략하기로 한다.

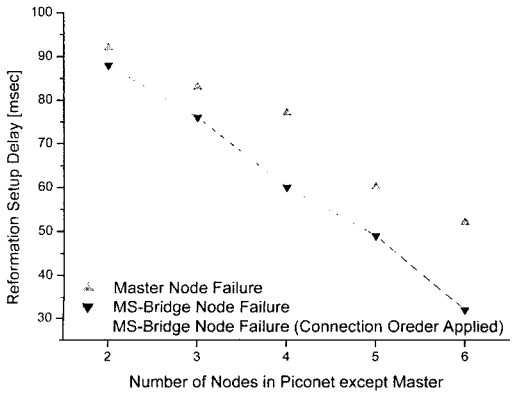

그림 15 MS 브릿지 노드 이탈 시 재형성 성공 확률

6. 결 론

본 논문에서는 기존의 블루투스 스캐터넷 관련 연구에서 거의 다루어지지 않았던 스캐터넷 재형성 알고리즘을 제안하고, 복구 노드 벡터(Recovery Node Vector) 알고리즘이라 명명하였다. 이 알고리즘은 스캐터넷 내부에서 하나 이상의 노드가 네트워크를 이탈하였을 경우 적용되는 것으로 다음과 같은 몇 가지 특징을 지닌다. 첫째, 노드의 형태에 따라 개별적으로 적용되므로 전체 스캐터넷의 형태와 무관하게 적용될 수 있는 범용 알고리즘이다. 또 재형성 이후에도 기존 스캐터넷 형태의 일관성을 유지할 수 있다. 둘째, 이웃하는 노드들이 스캐터넷을 이탈하였을 경우 다른 노드와 링크 연결을 하기 위한 정보를 복구 노드 벡터로 저장하고 있다. 이렇게 하기 위해서는 사전에 정보 교환을 통해 복구 마스터/슬레이브를 선정한다. 셋째, 재형성 과정에서 Inquiry 과정이 없이 Page 과정으로만 진행되므로 재형성 지연 시간이 상당 부분 단축된다. 이러한 것이 가능한 이유는 복구 노드 벡터를 작성하는 과정에서 연결에 필요한 정보를 사전에 획득하기 때문이다.

특히 복구 노드 벡터 알고리즘에서는 복구 노드 벡터를 작성하기 위한 전단계로 복구 마스터/슬레이브의 선정 과정이 중요한데, 이 선정 알고리즘은 노드의 형태에 따라 이루어진다. 본 논문에서는 노드 형태를 정형화하기 위한 인자로 노드 행렬과 노드 비중치를 도입하고, 이 인자들을 바탕으로 마스터, SS-브릿지, MS 브릿지 등의 노드 형태에 따른 복구 마스터/슬레이브 선정 알고리즘을 제안하였다.

이상에서 제안된 알고리즘을 실제 상용 하드웨어 구현을 하였고, 이를 통한 성능 평가 실험을 수행하였다. 또 실제 하드웨어 구현을 통해 동작 과정에서 발생하는 문제점을 해결하였다. 이러한 과정을 통해 제안된 복구 노드 벡터 알고리즘은 Inquiry 작업을 수행할 때 지연 시간의 23~60% 정도의 짧은 지연 시간과 97% 이상의 높은 복구 성공 확률 결과를 얻었다.

이러한 결과를 통해 블루투스 스캐터넷이 PAN과 같이 노드들이 동적으로 변화하는 상황에서도 적용될 수 있는 가능성을 제시하였다. 본 논문에서 제안한 재형성 알고리즘은 기존에 연구되어온 다양한 형태의 스캐터넷에 일부분을 추가하면 기존 스캐터넷의 형태를 유지하면서 적용될 수 있다. 또 실제 상용 하드웨어에 구현이 되고, 그 동작 및 성능을 검증한 실제적이고 구체적인 알고리즘이다. 또 본 논문에서 제안한 알고리즘은 바이

너리 CDMA와 같이 블루투스와 유사한 Ad-hoc 네트워크 기술에도 응용이 될 수 있을 것으로 예상된다.

본 논문에서는 노드가 네트워크에서 이탈하는 경우만을 고려하였으며, 추가되는 상황은 다루지 않았다. 하지만 노드가 추가되는 경우는 스캐터넷 형태를 고려해야 하고, 노드가 제거되는 경우와 비교하면 상대적으로 간단한 문제로 예상된다. 추후 노드가 추가되는 경우를 고려한 통합적인 재형성 알고리즘을 제안할 계획이다.

참 고 문 헌

[1] Bluetooth SIG, "Specification of the Bluetooth System Ver1.1", In http://www.csr.com, 2001.

[2] Bluetooth SIG, "Specification of the Bluetooth System Ver1.2", In http://www.csr.com, 2003.

[3] L. Ramachandran, M. Kapoor, A. Sarkar and A. Aggarwal, "Clustering Algorithms for Wireless Ad Hoc Networks," In Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications, pp. 54-63, 2000.

[4] T. Salonidis, P. Bhagwat, L. Tassiulas and R. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks" In Proceedings of the IEEE Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM 2001), Vol.3, pp. 1577-1586, 2001.

[5] G. Zaruba and S. Basagni, "Bluetress-Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks," In Proceedings of the IEEE International Conference on Communicatins (ICC 2001), Vol. 1, pp. 273-277, 2001.

[6] C.C. Foo and K.C. Chua, "Bluering-Bluetooth Scatternets with Ring Structures," In Proceedings of the IASTED International Conference on Wireless and Optical Communication(WOC 2002), 2002.

[7] C. Petrioli, S. Basagni and I. Chlamtac, "Configuring BlueStars: Multihop Scatternet Formation for Bluetooth Networks," In Proceedings of the IEEE Transactions on Computers, Vol. 52, Issue 6, pp. 779-790, 2003.

[8] Z. Wang, R.J. Thomas and Z.J. Haas, "Bluenet-a New Scatternet Formation Scheme," In Proceedings of the 35th Hawaii International Conference on System Science (HICSS-35), 2002.

[9] C. Petrioli, S. Basagni, and I. Chlamtac, "BlueMesh: Degree-constrained multihop scatternet formation for Bluetooth networks," In Journal of the Mobile Networking and Applications(MONET), Vol. 9, pp. 33-47, 2004.

[10] C. Lay and K.Y. Siu, "A Bluetooth Scatternet Formation Algorithm," In Proceedings of the Global Telecommunications Conference (GLOBECOM '01), Vol. 5, pp. 2864-2869, 2001.

[11] C. Law, A.K. Mehta and K.Y. Siu, "Performance of a New Bluetooth Scatternet Formation Protocol," In Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing, pp. 182-192, 2001.

[12] G. Tan, A. Miu, J. Guttag and H. Balakrishnan, "Forming Scatternets from Bluetooth Personal Area Networks," In MIT Technical Report MIT_LCS-TR-826, 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 Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS '03), pp. 313-322, 2003.

[14] H. Zhang, J.C. Hou and L. Sha, "A Bluetooth Loop Scatternet Formation Algorithm," In Proceedings of the IEEE International Conference on Communications (ICC '03), Vol. 2, pp. 1174-1180, 2003.

[15] S. Basagni, R. Bruno and C. Petrioli, "A Performance Comparison of Scatternet Formation Protocols for Networks of Bluetooth Devices," In Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003), pp. 341-350, 2003.

[16] 이한욱, 고상근, "하드웨어 구현을 기반으로 한 블루투스 스캐터넷 형성 알고리즘", In 한국정보과학회 논문지 I, Vol. 31, No. 3, pp. 314-326, 2004.

이 한 욱

1995년 3월~1999년 2월 서울대학교 기계항공공학부 학사. 1999년 3월~2001년 2월 서울대학교 기계항공공학부 석사. 2001년 3월~현재 서울대학교 기계항공공학부 박사과정 재학중. 관심분야는 무선 계측 시스템, 회전체 계측 시스템, 블루투스 스캐터넷 프로토콜

고 상 근

1973년 3월~1978년 2월 서울대학교 기계공학 학사. 1968년 3월~1980년 2월 서울대학교 기계공학 석사. 1984년 3월~1987년 8월 서울대학교 기계공학 박사. 1988년 2월~1989년 2월 스탠포드 대학교 방문 선임 연구원. 1997년 12월~1999년 1월 UCLA 방문 교수. 1989년 3월~현재 서울대학교 기계항공공학부 교수. 관심분야는 메카트로닉스 기반 시스템, 회전체 계측 시스템, 무선 계측 시스템, 카트로닉스(Cartronics)