웜홀 네트웍을 위한 새로운 교착상태 발견 기법

(A New Deadlock Detection Mechanism in Wormhole Networks)

이 수 정 †

(Su-Jung Lee)

요 약 웜홀 네트웍에서 교착상태의 복구를 기반으로 하는 라우팅 알고리즘은 간단한 하드웨어 구조와 높은 라우팅 적응성으로 인해 관심을 이끌었다. 진보적인 교착상태 복구 방안들은 교착상태에 속한 패킷들을 삭제하는 대신에 소수 전용 리소스들을 통해 전송한다. 교착상태에 속한 패킷은 타임아웃에 의해 선정하는데 다양한 트래픽 형태 및 패킷 길이를 고려하여 가장 바람직한 성능을 가져오는 제한 시간 값을 결정하기는 매우 어려운 일이다. 본질적으로, 타임아웃을 사용하는 현재의 방법들은 네트웍 부하가 심하거나 메시지 길이가 긴 경우에 교착상태의 존재를 잘못 판단할 가능성이 크다. 또한 교착상태가 발생할 경우, 하나 이상의 메시지가 교착상태로 판단되어 복구를 위해 마련된 자원을 과포화시킬 수 있다. 본 논문에서는 타임아웃을 사용하지 않고 보다 정확히 교착상태를 발견하는 방안을 제시한다. 제안한 방안은 교착상태를 잘못 판단하는 확률을 현저히 낮출 수 있다. 또한 순환 구조를 이루는 대기 상태의 메시지들 중에서 하나만을 교착상태라고 선언함으로써 복구에 따른 부담을 감소시킨다.

키워드 : 웜홀 네트웍, 인터커넥션 네트웍, 교착상태, 웜홀 라우팅

Abstract Deadlock recovery-based routing algorithms in wormhole networks have gained attraction due to low hardware complexity and high routing adaptability. Progressive deadlock recovery techniques require a few dedicated resources to transmit deadlocked packets rather than killing them. Selection of deadlocked packets is primarily based on time-out value which should be carefully determined considering various traffic patterns or packet length. By its nature, current techniques using time-out accompany unignorable number of false deadlock detections especially in a heavy-loaded network or with long packet size. Moreover, when a deadlock occurs, more than one packet may be marked as deadlocked, which saturate the resources allocated for recovery. This paper proposes more accurate deadlock detection scheme which does not make use of time-out to declare deadlock. The proposed scheme reduces the probability to detect false deadlocks considerably. Furthermore, a single message is selected as deadlocked for each cycle of blocked messages, thereby eliminating recovery overheads.

Key words : wormhole network, interconnection network, deadlock, wormhole routing

1. 서 론

웜홀 라우팅은 네트웍 자원을 사용하기 위한 메시지간의 경쟁이나 패킷 버퍼의 필요성을 없애기 때문에 매우 평판 좋은 방법이다 [1,2]. 이 방식에서는 패킷을 전송하기 위해서 여러 플릿(flit)으로 나눈다. 선두 플릿은 전송 경로를 주도하고 나머지 플릿은 이를 따라가는 방식을 취한다. 그러나 웜홀 라우팅은 교착상태에 걸릴 확률이 높은데 이 때 이에 속한 패킷들은 영원히 대기상태에 남게 된다. 교착상태는 이에 속한 각 패킷이 또다른 패킷이 점유하고 있는 자원을 서로간에 순환적으로 요청함으로써 비롯된다. 웜홀 네트웍에서 교착상태와 관련된 자원은 채널이다.

· 본 연구는 한국과학재단 목적기초연구(R05-2002-000-01126-0)지원으로 수행되었음.

† 정 회 원 : 인천교육대학교 컴퓨터교육과 교수
sjlee@gin.ac.kr

논문접수 : 2002년 10월 23일
심사완료 : 2003년 3월 4일

전통적인 교착상태 회피(deadlock avoidance) 방안에서는 채널간에 순환적인 의존관계가 발생하지 않도록 라우팅을 제한한다 [3,4]. 예를 들어, turn model은 라우팅 방식에 있어서 순환적 대기 상태를 초래할 가능성이 있는 방향 전환을 금지한다 [5,6]. 이러한 라우팅 알고리즘들의 특징은 적응성(adaptivity)이 낮다는 것이며 이에 따라 전송시간이 지연된다는 것이다. 교착상태를 회피하면서 보다 높은 처리률(throughput)을 성취하기 위한 방법 중 하나는 [7]에서 개발한 가상 채널(virtual channel)을 이용하는 것이다. 여러 가상 채널들이 하나의 물리적 채널을 공유함으로써 가상 네트웍을 구성하며 라우팅 알고리즘의 적응성을 향상시킨다. [8]에서는 가상 채널들을 두 분류로 나누었는데 하나는 교착상태가 발생하지 않는 확정된 라우팅(deterministic routing)에 이용되며 또다른 가상 채널은 완전한 적응성을 가진 최단거리 라우팅(fully adaptive minimal routing)에 사용된다. 이 방식은 보다 유연성이 있지만 부분적인 적응성을 갖는다는 단점이 있다.

[9,10]에서는 완전 적응성을 가진 라우팅 알고리즘은 교착상태의 발생 확률이 매우 낮다는 사실을 보고했다. 따라서 [11]에서 지적한 대로, 드물게 발생하는 교착상태 때문에 라우팅 알고리즘을 고안하는데 있어서 적응성을 제한한다는 것은 낭비라고 할 수 있다. 이는 교착상태를 다루는 새로운 방식을 유발시켰는데, 교착상태 발견 및 복구(deadlock detection and recovery) 방법이다. [12]에서는 교착상태 복구를 위한 특별한 플릿 버퍼인 교착상태 버퍼(deadlock buffer)를 각 라우터마다 마련했다. 이 버퍼들은 교착상태가 방지되는 경로를 제공한다. 교착상태에 속한 것으로 판단된 패킷은 이 경로를 사용하기 위해 독점적으로 사용허가를 받아야 한다. 이를 위해 토큰은 모든 라우터들을 운행한다. 이러한 순차적인 복구 방안 대신에 [13]에서는 교착상태를 동시적으로 복구하는 방안을 제시하였다. 즉, 교착상태로 의심되는 패킷은 언제든지 교착상태 버퍼를 사용할 수 있다. 이 방안에서는 복구를 위한 경로 상에 순환적 의존관계를 발생시키지 않기 위해 교착상태 버퍼들간에 Hamiltonian 궤도에 의한 순서를 부여하여 정해진 순서에 따라 패킷이 전송되도록 하였다.

[13]에서 제시한 방안의 성능은 교착상태의 발생빈도에 의해 좌우된다. 네트웍 상에 부하가 발생하면 교착상태로 추정되는 패킷들은 이의 복구를 위해 마련된 자원들을 포화시킴으로 인해 심각한 성능 저하를 가져온다. 그러므로, 실제 교착상태에 속한 패킷들만이 복구를 위한 자원을 사용하도록 함이 요구된다. 그러나, [13]의 방안은 제한시간을 초과하여 대기 상태에 있는 패킷과, 실제 교착상태에 속한 패킷을 구분하지 못하는 약점이 있다. 또한, 교착상태를 해결하기 위하여 단 하나의 패킷만을 선택하면 충분함에도 불구하고, 이 방안은 교착상태에 속한 모든 패킷들이 복구를 위한 궤도를 사용하도록 한다. 또 다른 문제점으로는 교착상태 복구를 위한 궤도가 최단거리가 아니기 때문에 신속한 복구를 이루는데 방해가 된다는 것이다.

교착상태의 판단 오류를 줄이기 위한 여러 가지 방안들이 개발되었다. [14]에서는 임의의 메시지가 요청한 모든 채널들이 이 채널들을 점유하고 있는 대기상태의 메시지들로 인해 일정 시간 동안 비활동중이라면 이 메시지를 교착상태로 판단한다. 이 방법은 [13]의 방법의 문제점들을 그대로 갖고 있다. [15]에서는 이를 개선하기 위하여 대기상태의 메시지들이 트리구조를 갖는 것으로 인식하고, 대부분의 경우에 이 트리의 근노드만을 교착상태로 결론짓는 방안을 제시하였다. 따라서 교착상태에 속하는 것으로 간주되는 메시지 수 및 복구에 따른 부담을 현저히 줄이게 되었다. 그러나 이 방안의 성능도 역시 제한시간에 의존한다. 문제는 [16]에서 시뮬레이션 결과를 통해 보인 것처럼, 모든 트래픽 형태를 감안하여 네트웍 포화상태를 방지하는 단 하나의 제한시간을 결정하는 일이 어렵다는 것이다. 더군다나 [15]에서처럼 다수의 메시지가 교착상태에 속한 것으로 간주되고 또한 잘못 판단되는 경우도 발생한다. 이 방안의 상세한 설명은 다음 절에 제공된다.

앞에서 언급한 대로 완전한 적응성을 가진 라우팅 알고리즘의 효율성은 교착상태 발견 알고리즘이 좌우함을 알 수 있다. 교착상태의 발견 방안은 대기상태의 메시지와 교착상태의 메시지를 구분해야 한다. 또한 복구를 위해 마련된 자원을 과부하시키지 않기 위해 단 하나의 패킷만을 선택해야 한다. 본 논문은 이러한 특성들을 지닌 새로운 교착상태 발견 기법을 제시한다. 이 기법에서는 probe라고 불리는 특별한 제어 패킷(control packet)을 사용한다. 대기상태의 메시지가 필요한 모든 채널들이 비활동적일 때 probe를 발생시킨다. Probe는 모든 비활동적인 채널을 거쳐서 인접하는 노드에 계속 전달된다. 만약 probe가 이를 발생시킨 시작노드에 도착하면 이는 대기중인 패킷들이 네트웍 상에서 순환구조를 이룰 가능성이 있으므로 교착상태로 판단한다. 제안된 방안에서는 probe를 통한 교착상태의 발견 작업을 독점적으로 수행하기 위하여 토큰을 이용한다.

기존의 방식들과는 달리, 본 논문의 기법은 제한시간을 토대로 하는 경험적인 방식에 의존하지 않는다. 결과

적으로 네트웍 성능은 네트웍의 부하정도나 패킷 길이와는 거의 무관하다. 더욱이 대기상태의 메시지들이 순환구조를 이룰 가능성이 있을 경우만 교착상태를 선언하므로 교착상태에 속한 것으로 판단하는 메시지 수를 크게 줄일 수 있다. 또한 대기 상태의 메시지들 중 하나의 메시지만을 교착상태에 속한 것으로 지정하기 때문에 복구 자원을 과부하시킬 가능성을 배제한다. 본 기법의 성능은 시뮬레이션을 통하여 제시하였다.

본 논문의 나머지 구성은 다음과 같다. 2절에서 네트웍 교착상태에 대한 기초 지식 및 기존 알고리즘에 대해 소개하였다. 3절에서는 새로운 기법에 대해 상세히 기술하였다. 4절에서 제시한 기법의 성능을 시뮬레이션 결과를 통하여 나타내었고 마지막 절에서 결론을 맺는다.

2. 배경

2.1 교착상태

웜홀 라우팅에서 대기상태의 패킷은 채널을 점유한 상태로 네트웍 상에 남아 있을 수 있다. 이는 패킷 전송이 무한정 지연되는 교착상태를 초래할 수 있다. 그림 1(i)은 네 개의 채널과 네 개의 라우터가 연관된 채널 교착상태의 예를 보여준다. A, B, C, D의 네 메시지가 각기 출력 채널을 기다리면서 플릿 버퍼에 대기 중에 있다. 각 채널 ci는 각기 다른 메시지가 점유하고 있다. 예를 들어, c1은 D 메시지가 사용 중이다. 각 라우터에서는 메시지가 채널의 사용이 허락되기를 대기하는 중이다. 그림에서 나타나듯이 메시지들이 순환 구조를 이루며 대기상태에 있기 때문에 더 이상의 진행이 불가하다. 이와는 반대로, 그림 1(ii)는 교착상태가 아닌 네트웍 상황을 보여준다. 그림 1(i)과의 차이점은 E 메시지가 c5와 c6 채널을 횡단한다는 점이다. 또한 메시지 A

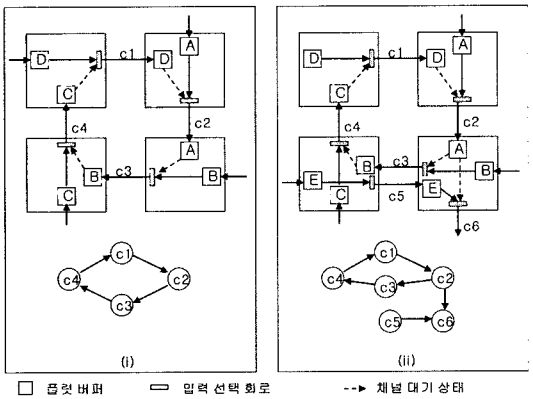

그림 1 (i) 교착상태 (ii) 비교착상태

가 c3과 c6 채널을 모두 요청하고 있다. E 메시지가 대기 중이 아니라고 가정할 때 A 메시지는 c6 채널을 언젠가는 획득할 수 있고 따라서 진행상태가 될 것이다. 마찬가지로 다른 메시지들도 순차적으로 진행상태가 될 수 있다. 그러므로 주어진 상황은 교착상태가 아님을 알 수 있다.

채널의존그래프(Channel Dependency Graph, CDG)는 네트웍 상에서 라우팅 알고리즘을 모델화하는데 사용된다 [4]. 이 그래프에서 노드는 네트웍의 단방향 채널에 해당하고 간선(edge)은 라우팅이 허락된 (ci, cj)의 채널쌍을 의미한다. 본 논문에서는 네트웍의 현재 상태를 반영하기 위해 CDG의 부분집합을 사용하기로 하고 이를 CCDG(Channel Current Dependency Graph)이라고 부른다. CCDG의 간선 (ci, cj)은 두 가지를 표현한다. 첫째는 ci를 점유한 메시지가 cj를 요청한 후 대기중일때이며 둘째는 채널의 점유 순서를 의미한다. 간선이 나타내는 바와 상관없이, cj를 ci의 후계자라고 부르기로 한다. 또한 ci의 모든 후계자 집합을 succ(ci)로 표기한다. 그림 1(i)과 (ii)에서 각 네트웍에 해당하는 CCDG가 하단에 제시되었다. 그림 1(ii)의 CCDG에서 볼 수 있듯이 완전 적응 라우팅 알고리즘(fully adaptive routing algorithm)에서 순환적 대기상태의 존재는 교착상태 형성의 필요조건에 지나지 않음을 알 수 있다 [17]. 교착상태는 CCDG가 knot를 포함하여야만 발생 가능하다. 3절에 제시된 방안은 간편성을 위해서 knot보다는 순환구조의 존재 가능성을 찾고자 한다. 그럼에도 불구하고, 이는 단지 제한시간에 의해 교착상태를 알아내는 것보다 정확한 판단을 가능하게 한다.

2.2 기존 방안

본 절에서는 [15]에서 제시한 교착상태 발견 방법에 대해 논의한다. 이 방법은 교착상태에 속한 메시지 중 하나만을 선택하고자 하는 데에 목적을 두었다. 일련의 대기상태의 메시지들이 트리를 형성하고 이 트리의 근 노드는 진행 상태의 메시지라는 사실에 초점을 맞추었다. 예를 들어, 그림 2에서 B, C, D 메시지는 대기상태

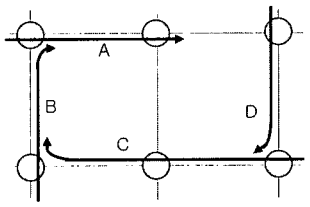

그림 2 A 메시지를 근노드로 하는 대기 중인 메시지들의 트리 예

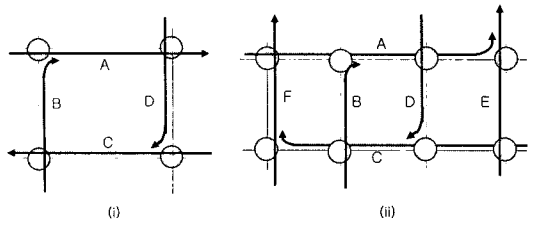

그림 3 (i) 비교착상태. A와 C 메시지가 근노드. (ii) 비교착상태. B와 D 메시지가 교착상태로 표시됨

이고 A 메시지는 진행상태이다. 이들 메시지는 A 메시지를 근노드로 하는 트리를 형성한다. A 메시지가 이후에 대기상태로 전환되면 이 방안에서는 B 메시지를 교착상태로 표시한다. 즉, 근노드로 인해서 대기상태가 된 메시지만 복구 자격을 준다. 그러므로, 교착상태에 놓인 메시지들 중 하나만이 교착상태로 표시될 수가 있다.

그러나 [15]의 방안에 따르면 여러 메시지가 교착상태로 표시될 수 있거나 또는 교착상태를 잘못 판단할 수 있다. 그림 3은 이러한 예를 보여준다. 그림 3(i)에서는 A와 C 메시지가 각기 근노드가 된다. 이후에 두 근노드가 그림 3(ii)에서와 같이 대기상태가 될 때 교착상태가 발생하지 않았음에도 불구하고 B와 D 메시지는 교착상태로 표시된다. 한편 만약 이들 메시지가 실제로 교착상태를 이룬다면 B와 D 메시지는 각기 교착상태로 표시된다. 이러한 단점 외에 [15]의 방안은 기존과 마찬가지로 제한 시간에 의존하는 문제가 있다. 만약 제한시간 값이 크다면, 교착상태의 발견은 지연될 것이고 그 값이 작다면 교착상태의 복구를 위해 준비된 자원이 과포화될 것이다.

3. 새로운 교착상태 발견 기법

본 절에서 제시하는 교착상태 발견 기법은 판단의 오류를 상당히 줄이고 단 하나의 메시지를 선택하여 복구한다. 이 기법은 메시지가 단순히 대기상태에 있는지 또는 순환구조를 이루는 대기 메시지들 중 하나의 가능성이 있는지를 구분한다. 더욱이 교착상태의 발견을 위해서 제한시간을 사용하지 않기 때문에 네트웍의 성능은 트래픽 형태나 패킷 길이에 따라 변동되지 않는다.

제시한 기법의 기본적 개념은 다음과 같다. Probe라고 불리는 특별한 제어 패킷이 비활동적인 채널 상으로 전송된다. Probe가 자신을 발생시킨 라우터에게 되돌아오면, 이는 메시지들이 순환구조 형태를 이루며 대기상태에 있을 가능성을 이름이며 따라서 교착상태가 존재하는 것으로 추정한다. Probe 작업을 독점적으로 수행하기 위하여 토큰을 사용한다. 즉, 토큰은 probe를 시작하는 임의의 라우터가 이를 획득하기 전까지 네트웍 상에 표류한다. 임의의 probe 작업은 토큰 번호로 식별하고 이 작업을 시작할 때 토큰 번호는 증가하며 probe에 의해 전송된다. 이러한 방식으로 단 하나의 대기 중인 메시지만이 교착상태인 것으로 표시된다. 이후의 논의에서는 제안하는 기법을 편의상 Probe 기법이라 일컫기로 한다.

Probe를 통한 교착상태 발견 작업은 다음의 두 조건이 모두 만족되면 시작된다. 즉, (i) 임의의 라우터에서 패킷 헤더가 요청한 모든 채널이 그 채널들을 점유한 대기상태의 메시지들로 인해 비활동적인 경우와 (ii) 그 라우터가 토큰을 갖고 있는 경우이다. 논문 기술의 편리함을 위하여 다음의 용어를 정의한다:

- InitR : probe 수행 작업이 시작된 라우터.
- InitM : InitR에서 대기중이며 probe 작업을 유발한 메시지.
- InitC : InitM의 헤더가 마지막으로 점유했던 채널.

그림 4와 같은 네트웍 형상을 살펴보자. 토큰은 R3에 있다고 가정하자. A 메시지가 R3에 도착했을 때 A 메시지가 요청한 채널은 이미 D 메시지가 사용하고 있다. D 메시지는 또한 R6에서 대기상태에 있다. 따라서 R3에서 R6에 이르는 채널은 비활동적이다. 그러므로, R3은 토큰을 획득한 후 A 메시지를 위하여 probe 작업을 시작한다. 위 용어 정의에 따르면 InitR=R3, IniM=A 이며 InitC는 R2에서 R3에 이르는 채널을 가리킨다. Probe는 채널을 따라 R6으로 전달되며 R6에서는 R3과 비슷한 상황이 발생한다. 따라서 probe는 R5로 전달된다. 그림에서 알 수 있듯이 probe는 비활동적인 채널 상으로 전파되어 마침내 시작노드인 R3에 도착하게 된다. Probe의 전송 순서는 굵은 화살표 상의 숫자로 나타내었다. 시작노드인 R3이 probe를 수신하면 A 메시

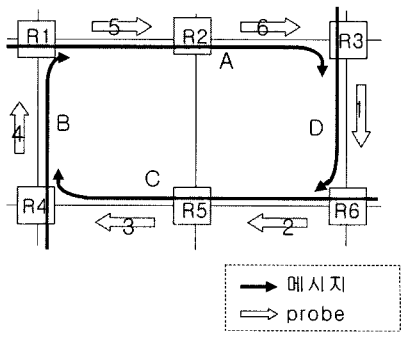

그림 4 교착상태. R3이 probe 작업을 시작함

지는 교착상태로 표시되고 R3은 토큰 번호를 증가시킨 후 토큰을 재생시킨다.

위의 예에서 간편성을 위하여 각 메시지는 단 하나의 출력 채널을 요구하는 것으로 가정하였다. 그러나 실제로는 완전 적응성을 지닌 라우팅에서 허용된 출력 채널이 하나 이상일 경우가 있다. 물론 임의의 메시지가 대기상태라면 메시지가 요청하는 모든 채널 상으로 probe를 전달해야 한다. 따라서 라우터는 같은 토큰 번호의 probe를 한 번 이상 수신할 경우가 있으며 이 때 probe가 무조건 전달된다면 네트워 상에서 영원히 전송될 위험이 있다. 이를 예방하는 간단한 방법으로 각 라우터는 probe를 최초로 수신하면 라우터 내의 버퍼에 수신된 probe를 저장하고 같은 토큰 번호를 가진 이후의 probe는 모두 폐기한다. 이는 probe 작업의 종료를 보증한다.

Probe들은 비활동적 채널을 통하여 전송되기 때문에 정상적인 메시지의 처리 작업에 지장을 주지 않는다. Probe 전송을 구현하는 한 가지 방법으로 [12,13]에서 제의한대로 각 라우터의 중심부에 위치한 특별한 플릿 버퍼를 사용할 수 있다. 이들 버퍼들은 교착상태가 방지되는 복구 궤도를 형성한다. 본 논문에서는 이 버퍼를 ProbeBuf라고 명명하기로 하며 토큰번호를 저장할 만한 크기를 가진 것으로 가정한다. Probe 기법은 그림 5에 요약되어 있다. probe.tokenNum는 probe가 전달하는 토큰번호를 지칭하며 ProbeBuf.tokenNum는 ProbeBuf에 저장된 토큰번호를 의미한다.

이제 토큰의 재생 시간에 대해 논의하자. 이 시간은 probe 작업을 시작할 때 InitM의 상태와 관련되어 있다. InitM의 상태는 결국 진행상태가 되거나 혹은 교착상태로 인하여 영원히 대기상태가 되는 두 가지 경우가 있다. 후자의 경우에 InitM은 교착상태에 직접 속하거나 또는 교착상태의 외부에서 이의 해결을 기다림으로 해서 간접적으로 관련될 수 있다. 그림 6은 이들 세 가지 경우를 보여준다. 첫 번째 경우에 InitM은 probe 작업이 시작된 지 얼마 후 진행상태로 전환된다. 두 번째 경우, Probe 기법에 의하면 *InitR*은 순환구조를 발견하게 된다. 이 두 경우에 있어서 토큰은 InitM이 진행상태가 되는 시점 또는 InitR이 교착상태를 발견하는 시점에 재생된다. 세 번째 경우에는 InitM이 향후 진행상태로 변화하지도 않고 InitR이 순환구조를 발견하지도 않는다. 즉, probe가 InitR로 되돌아오지 않을 수 있다. 따라서 네트워은 토큰을 다시 가질 수 없게 되는데 이는 InitR이 probe 작업 시작 시점에 토큰을 네트워에서 제거했기 때문이다.

그러나 probe 작업이 교착상태를 발견하지 못할지라

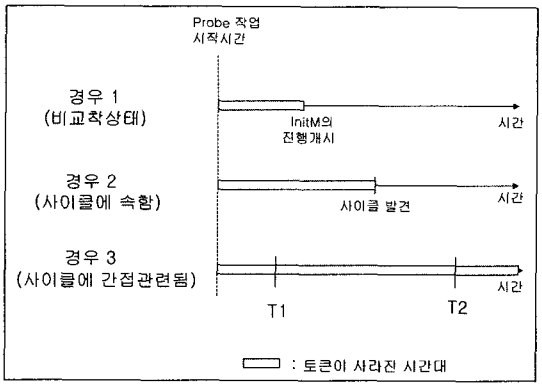

그림 6 InitM 상태의 가능한 경우들

```
Probing Initiation Condition:
(1) A router has the token and
(2) A message is blocked and all channels requested by the message are inactive.

InitR upon initiation of probing:
Store probe(TokenNum) into ProbeBuf;
Send probe along each inactive channel in succ(InitC);

A router(≠InitR) upon receiving a probe through channel c:
If ProbeBuf is empty or ProbeBuf.TokenNum < probe.TokenNum
        Store the probe into ProbeBuf;
        Send probe along each inactive channel in succ(c);

InitR upon receiving a probe:
If InitM is still blocked and probe.TokenNum=ProbeBuf.TokenNum
        Mark InitM as deadlocked;          /* Cycle found */
```

그림 5 Probe 기법

도 토큰은 재생되어야만 한다. 그러므로 InitM이 경우 1이나 경우 2에 속하지 않으면 probe 작업 시작 후 일정 시간이 지난 후 토큰은 재생되어야 한다. 이 때 probe 작업에 영향을 주지 않도록 재생 시간을 주의 깊게 결정해야 한다. 예를 들어, 그림 6은 T1과 T2의 두 토큰 재생 시간을 보여주고 있다. 만약 재생시간을 T2로 결정하면, 그 시간까지 InitM이 경우 1에 속하는지 경우 2에 속하는지 판가름 나기 때문에 오류 없는 결정이라 할 수 있다. 즉, T2 시간에 InitR은 토큰을 아직 재생하지 않았다면 이를 수행한다. 그러나 만약 T1을 토큰 재생시간으로 결정하면, InitM의 상태는 지나치게 일찍 결정되어 판단착오를 일으킨다. 예를 들어, 실제로 InitM의 상태가 경우 2에 속한다고 가정하자. 그렇다면 Probe 기법에 의해서 순환구조는 발견되지 않을 수 있다. 그림 7은 이의 예이다. R1이 'tokenNum1'으로 식별되는 probe 작업을 시작했다고 가정하자. 또한 probe가 전달되는 동안, R1이 'tokenNum1'+1의 번호를 지닌 토큰을 T1 시각에 재생했다고 하자. 그림에서 보듯이 네트웍 내의 임의의 노드가 이 토큰을 획득하고 'tokenNum1'+1의 식별자를 지닌 또다른 probe 작업을 시작했다. 그리고 후자의 probe 작업이 전송한 probe가 전자의 probe보다 빨리 R5에 도달하였다. 따라서 R5는 'tokenNum1'+1의 probe를 R6으로 전달하고 R4로부터 수신된 probe는 폐기한다. 그러므로 그림에 나타난 순환구조는 R1의 probe 작업에 의해 발견되지 못한다.

위의 설명에서 알 수 있듯이 경우 3에서 토큰 재생 시간은 충분히 크게 정해야 한다. Probe가 전체 네트웍을 모두 통과하는 시간이 한 예이다. 그러한 시간을 PTime이라 하자. 그림 8은 토큰 재생에 관한 logic을 정리해 놓은 것이다.

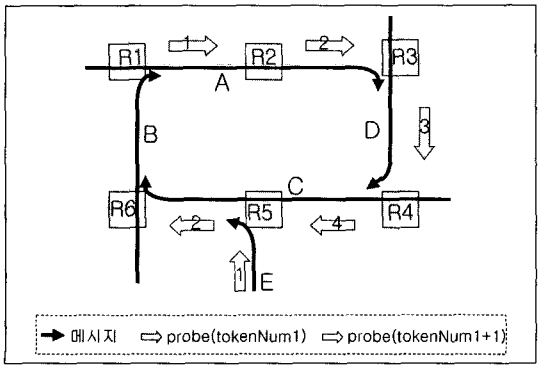

그림 7 다수의 probe 작업이 수행될 때 발견되지 못하는 교착상태의 예

```
InitR upon initiation of probing:
        Remove the token(TokenNum);
        elapsed_time := 0;

At InitR:
If InitM becomes unblocked or marked as deadlocked
        Regenerate the token(TokenNum+1);
        exit; /* terminate Probe mechanism */
else
        elapsed_time := elapsed_time + 1;
If elapsed_time >= PTime
        Regenerate the token(TokenNum+1);
        exit;       /* terminate Probe mechanism */
```

그림 8 토큰 재생 로직

교착상태를 발견하는 시간은 토큰 획득 시간에 크게 좌우된다. 토큰을 구현하는 초보적인 방법은 토큰과 라우터의 클락(clock) 사이클을 동기화하는 것이다. 이러한 방법은 N을 네트웍의 노드 수라 할 때 평균 토큰 획득 시간이 N/2가 되게 한다. 그러나 펄스를 전달하는 방식을 사용하면 펄스 전달에 따른 지연시간을 라우터 클락의 1/5이라고 가정할 때, 토큰 획득시간은 대략 N/10으로 축소시킬 수 있다 [12]. 토큰 구현에 관한 주제는 이 논문의 범위에서 벗어나며 자세한 내용은 [12]에 제시되었다.

Probe 기법의 정확성은 시스템 내 토큰의 존재 여부에 의존한다. 따라서 다음 명제는 이를 증명한다.

명제 1. 네트웍에서 토큰 손실은 발생하지 않는다.

증명. 토큰이 손실되는 유일한 경우는 임의의 라우터가 Probe 기법을 시작함으로 인해 토큰을 획득한 후 재생시키지 않을 때이다. 토큰 재생 로직에 따르면 InitM이 진행 상태가 되거나 교착상태를 발견하게 되면 토큰을 재생한다. 이러한 경우가 성립되지 않을 때는 probe를 발생시킨 후 PTime이 경과하면 토큰을 재생한다. 그러므로 명제는 성립한다. □

명제 2. 네트웍에서 임의의 시각에 토큰은 둘 이상 존재하지 않는다.

증명. 토큰이 둘 이상 존재한다고 가정하자. 토큰 개수는 Probe 기법과 관련있으므로 두 가지 가능성이 있다. 첫째는 시스템 내에 Probe 기법을 진행 중인 라우터가 둘 이상인 경우이며 둘째는 Probe 기법을 시작한 라우터가 하나이며 그 라우터에서 토큰을 두 번 이상 재생시킬 경우이다. 첫 번째 가능성부터 고려하자. 두 개의 라우터를 각각 A와 B라고 하고 A 라우터가 B 라우터보다 먼저 Probe 기법을 시작했다고 하자. Probe 기법을 시작하기 위해서 B 라우터는 토큰을 획득해야 한다. 토큰은 A 라우터가 이미 획득하였으므로 B 라우

터가 Probe 기법을 시작할 수 있었다는 것은 A 라우터가 토큰을 재생하였다는 것을 의미한다. 그러나 A 라우터는 토큰을 재생하면 Probe 기법을 종료한다. 따라서 둘 이상의 라우터가 Probe 기법을 동시 진행할 수 없다. 마찬가지로 두 번째 가능성도 성립하지 않음을 알 수 있다. □

위의 설명에 따르면 Probe 기법은 교착상태를 잘못 판단할 가능성이 있다. 그러나 네트웍 상에 교착상태가 실제로 발생하면 반드시 발견하게 된다. 다음 정리는 이를 증명한다.

정리. 교착상태가 발생하면 이에 속한 메시지들 중 적어도 하나는 교착상태를 발견한다.

증명. 교착상태를 발견하는 라우터가 존재하지 않음을 가정하자. 명제 1에 의해서 교착상태에 속한 임의의 라우터 A는 토큰을 획득할 수 있다. 또한 이 때 A 라우터는 교착상태에 속하므로 대기 상태인 메시지, InitM을 포함한다. 따라서 Probe 기법을 시작하는 조건을 만족하여 probe를 발생시킨다. InitM은 교착상태에 속하므로 계속 대기상태를 유지한다. 그러므로 Probe 기법의 교착상태 발견 조건에 따라서 A 라우터가 교착상태를 발견하려면 자신이 발생시킨 probe를 수신하면 된다. 그러나 증명 서두의 가정에 의해 A 라우터는 probe를 수신하지 못함을 알 수 있다. 여기엔 두 가지 원인을 고려해 볼 수 있다. 첫째는 probe가 되돌아올 네트웍 경로상의 조건이 성립하지 않기 때문이고, 둘째는 probe가 중도에 손실되었기 때문이다. 그러나 첫째의 경우, A 라우터가 교착상태이므로 네트웍상에 이를 포함한 knot가 존재하며 따라서 순환구조 속에 있게 된다. Probe 기법에 따르면 probe는 모든 후계자에게 전송되므로 A 라우터는 자신의 probe를 수신하게 된다. 두 번째 원인의 경우는 A 라우터의 probe를 수신한 임의의 라우터 B가 존재하여 B 라우터가 A 라우터의 probe를 두번째로 수신하게 되면 이를 Probe 기법에 의하여 폐기하는 경우이다. 즉, B 라우터가 전달하는 첫 번째 probe는 A 라우터에서의 교착상태 발견에 도움을 주지 못하는 경우이다. 이러한 경우에 A 라우터는 PTime이 경과한 후 토큰을 재생하게 되며 결국 B 라우터는 Probe 기법을 시작할 수 있다. 앞에서 B 라우터는 같은 probe를 두 번 수신한다고 하였으므로 교착상태 발견조건을 만족하여 이를 발견할 수 있다. □

4. 성능 연구

4.1 시뮬레이션 모델

본 절에서는 제안한 방법과 [15]에서 제시한 기존의 방법을 비교하는 시뮬레이션 결과를 제시한다. [15]의 방법은 타임아웃에 근거한 전통적 방법보다 우세하다고 보고되었으므로 본 논문에서는 전통적 방법의 시뮬레이션은 생략한다. 시뮬레이션은 16×16 메쉬 네트웍을 기본으로 하였다. 양방향 채널과 두 개의 플릿을 저장할 수 있는 버퍼가 각 가상 채널에 마련되었음을 가정하였다. 다양한 메시지 길이와 트래픽 형태에 대해 100000 클락 사이클 동안 시뮬레이션을 행하였고 처음 10000 사이클 동안에 수집된 결과는 무시하였다. 메시지는 각기 다른 주기로 발생되었다. 만약 노드 내에 새로운 메시지를 위한 공간이 부족하다면 그 노드는 새로운 메시지를 발생시키지 않도록 하였다. 완전한 적응성을 가진 최단거리 라우팅을 시뮬레이션하였고 성능평가의 기준은 교착상태에 속한 것으로 선언되는 메시지 수로 정하였다. 앞 절에서 설명한 바와 같이 교착상태 복구 기반의 라우팅 기법에서는 메시지의 대기시간을 측정한 후 일정시간이 경과하면 교착상태에 속한 것으로 판단한다. 따라서 임의의 메시지가 교착상태에 속하면 이러한 상태가 회복되지 않는 한 영원히 대기상태에 머물게 되므로 이 메시지는 교착상태에 속한 것으로 판단될 수밖에 없다. 그러므로 문제는 교착상태가 아님에도 불구하고 이를 잘못 판단하게 되는 데 있다. 따라서 교착상태로 선언되는 메시지 수가 적으면 교착상태를 오류 판단하는 확률이 낮아지기 때문에 해당 방법의 성능이 우수한 것으로 판단한다. 메시지가 교착상태인 것으로 추정되면 노드에서 삭제되고 그 노드는 또다른 새로운 메시지를 발생시킨다. 즉, 교착상태에 속한 메시지들을 위한 복구 경로는 예비되지 않았다.

4.2 교착상태 발견 주기

다양한 메시지 발생률에 따라 교착상태로 선언되는 메시지 비율을 측정하였다. 메시지의 도착점이 균등분포되어 있는 경우와 perfect-shuffle인 두 가지 형태의 트래픽에 대해 시뮬레이션하였다. [15]의 방안에 대해서는 타임아웃을 변화시켰다. 메시지 길이에 따른 영향을 파악하기 위해 16 플릿, 32 플릿, 그리고 64 플릿의 메시지를 사용하였다.

표 1에서 메시지의 도착점이 균등분포일 때 16 플릿, 32 플릿, 그리고 64 플릿의 메시지 길이에 대해 교착상태로 선언된 메시지의 백분율을 나타냈다. [15]의 결과는 'Lopez'라고 표시하였으며 16 사이클, 32 사이클, 128 사이클(메시지 길이가 64플릿인 경우)의 타임아웃에 대해 시뮬레이션되었다. 16 플릿과 32 플릿의 메시지 길이에 대한 두 결과는 대략 비슷한 양상을 나타낸다. 기대한 바와 같이 교착상태 발견 비율은 메시지 발생률

에 비례한다. [15]의 방법은 타임아웃이 짧을 때 네트웍이 혼잡해짐에 따라 보다 높은 발견 비율을 보인다. 또한 메시지 길이의 영향을 보다 크게 받는 것으로 나타났다. 동일한 타임아웃에 대해 메시지 길이가 길수록 교착상태 발견 비율이 높아지는 경향이 있는 것을 알 수 있다. 이는 메시지 길이가 길수록 네트웍 상에 머무는 시간이 길어지고 따라서 대기 시간도 증가하기 때문이다. 표 1에서 동일한 타임아웃(Th=32)에 대해 메시지 길이가 64 플릿인 경우에 교착상태로 간주될 확률이 가장 높은 것으로 나타났다. 따라서 이 경우에는 메시지 길이를 반영하여 타임아웃을 128 클락까지 시험하였다. Probe 기법은 전반적으로 메시지 길이의 영향을 보다 적게 받는 것을 알 수 있다.

두 가지 방안의 성능을 알아보기 위하여 메시지 도착점이 perfect-shuffle 분포를 이루는 트래픽에 대해서도 시뮬레이션하였다. 표 2는 16 플릿과 32 플릿의 메시지에 대한 결과이다. 균등분포의 경우와 유사한 양상을 보이는데 특히 32 플릿의 메시지에 대해 네트웍 부하가 적을 때 두 방법의 성능은 비슷하다. 그러나, 대부분의 경우 Probe 기법의 성능이 보다 우수함을 알 수 있다.

전반적으로 [15]의 방법은 타임아웃에 따라 교착상태 발견 비율이 변동한다. 타임아웃이 짧으면 네트웍 부하가 증가됨에 따라 더욱 영향을 받게 된다. 그러므로 다양한 트래픽 형태와 메시지 길이를 만족하는 적절한 타임아웃 값을 결정하기는 쉽지 않다. 반면에, Probe 기법에 따르면 교착상태를 잘못 판단할 확률은 보다 낮은 것으로 나타났다. 또한 트래픽 형태나 메시지 길이의 영향을 보다 적게 받는다. 그러나 표에서와 같은 발견 비율은 교착상태가 네트웍상에 반드시 존재함을 의미하는 것은 아니다. 실제로 시뮬레이션 결과 교착상태를 잘못 판단하는 비율은 가상 채널 수가 둘인 경우에 99%에 달하는 것으로 나타났다. 이는 [10]에서 언급하였듯이 네트웍이 과부하되지 않는 한 교착상태가 발생하는 경우가 미미하다는 것을 말해준다. 실제로 네트웍상에 교착상태가 존재하지 않음에도 이를 잘못 판단하는 비율을 알아보기 위하여 가상채널이 존재하지 않음을 가정하고 실험한 결과를 표 3에 나타내었다. 표의 숫자는 교착상태로 판단된 메시지 수 중에서 오류로 판단된 메시지 수가 차지하는 백분율이다. [10]의 결과와 마찬가지로 가상채널 수를 줄임에 따라, 교착상태를 오류 판단하는 확률이 99%(가상채널수=2)에서 표 3의 비율로 감소했음을 알 수 있다. 이는 가상채널 수가 적을수록 메시지가 대기 상태로 지속되는 시간이 길어지고 따라서 교착상태가 실재할 확률은 높아지며 이에 따라 각 기법이

표 1 메시지 도착점이 균등분포일 때 교착상태 발견 비율 (%)

메시지 크기	메시지 발생률 / 방안	0.01	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4
16플릿	Probe 기법	0	0	0.003	0.004	0.032	0.036	0.055	0.201	0.255
	Lopez(Th=16)	0.011	0.013	0.112	0.163	0.23	0.321	0.354	0.711	6.291
	Lopez(Th=32)	0	0	0.003	0.068	0.076	0.088	0.113	0.149	1.155
32플릿	Probe 기법	0	0	0.002	0.029	0.049	0.081	0.084	0.111	0.597
	Lopez(Th=16)	0.78	0.805	0.851	0.954	1.188	2.19	2.191	4.505	10.478
	Lopez(Th=32)	0.264	0.274	0.311	0.368	0.383	0.535	0.898	0.958	1.131
64플릿	Probe 기법	0	0.012	0.036	0.062	0.072	0.16	0.27	0.574	0.613
	Lopez(Th=32)	3.506	3.575	5.205	6.014	6.299	6.796	8.276	9.141	11.471
	Lopez(Th=128)	0.063	0.068	0.118	0.227	0.417	0.698	0.953	1.372	1.923

표 2 메시지 도착점이 perfect-shuffle일 때 교착상태 발견 비율 (%)

메시지크기	메시지 발생률 / 방안	0.01	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4
16플릿	Probe 기법	0	0	0	0	0	0.005	0.009	0.011	0.023
	Lopez(Th=16)	0	0	0.028	0.129	0.141	0.152	0.505	0.604	0.822
	Lopez(Th=32)	0	0	0.003	0.022	0.036	0.037	0.078	0.091	0.186
32플릿	Probe 기법	0	0	0	0	0	0	0.005	0.007	0.015
	Lopez(Th=16)	0.83	0.83	1.083	1.254	1.298	1.363	2.04	2.334	2.63
	Lopez(Th=32)	0	0	0	0	0.014	0.021	0.03	0.39	0.462

교착상태를 오류 판단하는 확률은 낮아지게 되기 때문이다. 또한 네트워 부하가 클수록, 그리고 타임아웃의 길이가 짧을수록 오류 판단 비율은 증가하는 것을 알 수 있다. 특히 Probe 기법은 오류 판단의 확률이 상대적으로 매우 낮은 것으로 나타났는데 이에 따라 교착상태 복구를 위한 자원에 과부하를 유발할 가능성이 적음을 알 수 있다.

표 4는 Probe 기법에 따른 평균 probe 발생 수를 나타낸다. 각 트래픽 형태와 메시지 크기 및 메시지 발생률에 크게 좌우되지 않고 일반적으로 수 개의 probe가 발생되는 것을 알 수 있다. Probe는 비활동적인 채널을 통하여 전달되기 때문에 정상적인 메시지 소통에는 크게 영향을 미치지 않는다. 또한 채널이 비활동적으로 지속되는 시간을 측정하여 이 시간이 제한값에 이르렀을 때 probe를 전달하도록 하고 제한값을 늘리는 방식으로 probe 수를 보다 감소시킬 수 있다.

5. 결 론

본 논문에서는 웜홀 네트웍에서 교착상태를 발견하는 보다 향상된 성능의 기법을 제시하였다. 기존의 방법들과는 달리, 제안한 방법은 타임아웃에 의존하지 않는다. 교착상태에 속하였는지를 알아내기 위하여 대기상태인 메시지는 제어(control) 메시지를 발생시키고 제어 메시지가 되돌아오면 교착상태에 속한 것으로 간주한다. 제어 메시지는 비활동적 채널을 통해서만 전송되기 때문에 정상적 메시지 통신을 방해하지 않는다. 이러한 특성은 이전 방안들과 비교할 때 교착상태를 오류 판단하는 경우를 상당히 줄이는 결과를 가져온다. 더욱이 교착상태가 판단되었을 때 하나의 메시지만을 그러한 상태에 속한 것으로 판단하므로 복구에 따른 부담을 경감시킬 수 있다. 제안한 방법의 성능은 시뮬레이션 결과에서 보여지듯이 트래픽 형태나 메시지 길이의 영향을 보다 적게 받는 것으로 나타났다.

제안한 방법은 토큰을 사용함으로써 토큰 관리가 이루어지는 시스템에서 구현이 가능하다. 교착상태를 발견하기 위한 프로세스를 시작하기 전에 라우터는 토큰을 획득하여야 한다. 따라서, 토큰 전송 시간이 신속한 교착상태의 발견에 중요한 요소로 등장한다. 향후 과제는 토큰을 사용하지 않고 본 논문에서 보고한 성능과 유사한 결과를 가져오는 방안을 개발하는 것이다.

참 고 문 헌

[1] K. M. Al-Tawil, M. Abd-El-Barr, and F. Ashraf, "A survey and comparison of wormhole routing techniques in a mesh networks", IEEE Network, Vol. 11, No. 2, pp. 38-45, 1997.

[2] L. M. Ni and P. K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks", IEEE Computer, Vol. 26, No. 2, pp. 62-76, 1993.

[3] H. Park and D. P. Agrawal, "Generic Methodologies for Deadlock-free Routing", Proc. of the 10th Int'l Parallel Processing Symp., pp. 638-643, 1996.

[4] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", IEEE Trans. Computers, Vol. C-36, No. 5, pp. 547-553, 1987.

[5] G.-M. Chiu, "The Odd-even Turn model for Adaptive Routing", IEEE Trans. Parallel and Distributed Systems, Vol. 11, No. 7, Jul. 2000.

[6] C. J. Glass and L. Ni, "The Turn Model for Adaptive Routing in Multicomputer Networks", Int'l Symp. on Computer Architecture, pp. 278-287, May 1992.

[7] W. J. Dally, "Virtual Chanel Flow Control", IEEE Trans. Parallel and Distributed Systems, Vol. 3, No. 2, pp. 194-205, Mar. 1992.

[8] J. Duato, "A New Theory of Deadlock-free Adaptive Routing in Wormhole Networks", IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 12, pp. 1320-1331, Dec. 1993.

[9] T. M. Pinkston and S. Warnakulasuriya, "On Deadlocks in Interconnection Networks", Proc. of Int'l Symp. on Computer Architecture, pp. 38-49, 1997.

[10] S. Warnakulasuriya and T. M. Pinkston, "Charac-

표 3 Probe 기법이 발생시키는 평균 probe 수

트래픽 형태	메시지 발생률 / 메시지크기	0.01	0.1	0.05	0.15	0.2	0.25	0.3	0.35	0.4
균등분포	16 플릿	2.202	2.076	2.181	2.213	2.327	3.306	4.056	4.167	4.298
	32 플릿	2.391	2.146	2.033	2.359	4.507	2.526	2.442	3.292	3.839
	64 플릿	2.564	2.432	2.372	3.75	3.722	2.866	2.836	2.508	2.727
perfect-shuffle	16 플릿	1.947	1.95	1.932	1.977	1.972	1.995	2.032	2.086	2.041
	32 플릿	1.973	2.181	1.976	1.711	2.021	2.011	1.879	1.98	2.509

terization of deadlocks in interconnection networks", Proc. of the 11th Int. Parallel Processing Symposium, Apr. 1997.
[11] T. M. Pinkston, "Flexible and Efficient Routing Based on Progressive Deadlock Recovery", IEEE Transactions on Computers, Jul. 1999.
[12] Anjan K. V. and T. M. Pinkston, "DISHA: A Deadlock Recovery Scheme for Fully Adaptive Routing", The 9th Int'l Parallel Processing Symp., pp. 537-543, Apr. 1995.
[13] Anjan K. V., T. M. Pinkston, and J. Duato, "Generalized Theory for Deadlock-free Adaptive Wormhole Routing and its Application to Disha Concurrent", The 10th Int'l Parallel Processing Symp., pp. 815-821, 1996.
[14] J. M. Martinez, P. Lopez, J. Duato, and T. M. Pinkston, "Software-Based Deadlock Recovery Technique for True Fully Adaptive Routing in Wormhole Networks", Int'l Conf. on Parallel Processing, 1997.
[15] P. Lopez, J. M. Martinez, and J. Duato, "A Very Efficient Distributed Deadlock Detection Mechanism for Wormhole Networks", Proc. High Performance Computer Architecture Workshop, Feb. 1998.
[16] M. Thottethodi, A. R. Lebeck, and S. S. Mukherjee, "Self-Tuned Congestion Control for Multiprocessor Networks", Proc. the 7th Int'l Symp. on High-Performance Computer Architecture, 2001.
[17] S. Warnakulasuriya and T. M. Pinkston, "A Formal Model of Message Blocking and Deadlock Resolution in Interconnection Networks", IEEE Trans. Parallel and Distributed Systems, Vol. 11, No. 3, pp. 212-229, Mar. 2000.

이 수 정

1985년 이화여자대학교 수학교육과 졸업. 1985년~1987년 삼성생명 정보시스템실 근무. 1988년~1994년 미국 Texas A&M 대학교 컴퓨터과학과 졸업(M.S, Ph.D). 1994년~1998년 삼성전자 통신개발연구소 근무. 1998년~현재 인천교육대학교 컴퓨터교육과 조교수. 관심분야는 분산시스템, 교착상태 알고리즘, 라우팅 알고리즘 등