웜홀 라우팅 망에서의 효율적인 다중 멀티캐스트 알고리즘

(Efficient Multiple Multicast Algorithms in Wormhole-Routed Networks)

김 시 관 † 조 정 완 ††
(Si-Gwan Kim) (Jung Wan Cho)

요 약 웜홀망의 성능에 영향을 미치는 가장 중요한 요소는 개시지연시간인데 이는 메시지가 생성되어 망에 투입되기까지의 시간으로 정의된다. 본 논문에서는 다중 멀티캐스트 메시지의 개시지연시간을 단축하기 위하여 노드간의 메시지 충돌을 최소화하는 효율적인 멀티캐스트 전송 알고리즘을 제안한다. 제안되는 3가지의 알고리즘은 사용 가능한 채널들을 되도록 균등하게 사용함으로써 기존 제안된 알고리즘보다 우수한 성능을 보인다. 제안된 알고리즘들이 교착상태가 없음을 증명하고 2차원 메쉬망에서의 여러 가지 조건하에서 제안된 알고리즘의 우수함을 시뮬레이션을 통하여 증명한다. 제안 알고리즘의 전반적인 성능은 기존 알고리즘보다 20% 정도 우수함을 알 수 있다. 제안된 2차원 메쉬 다중 멀티캐스트 알고리즘은 3차원 메쉬망으로 확장이 용이하다.

Abstract The most important metric in wormhole-routed networks is the start-up latency. In this paper, we present new multicast algorithms that reduce node contention so that multiple multicast messages can be implemented with reduced latency. By exploiting available channels evenly as much as possible, these new algorithms show better performance than the existing multicast algorithms for wormhole 2D systems when multiple multicasts are involved. All algorithms presented are proven to be deadlock-free. A simulation study has been conducted that compares the performance of these multicast algorithms under various situations in a 2D mesh. We show that the overall performance of ours are up to 20% better than the previous studies. We observe that reducing the number of the generated multidestination messages closely related to shorter message latency. These proposed algorithms can be easily extended to 3D mesh systems.

1. 서 론

고성능 컴퓨터를 설계할 때 구성하는 프로세서의 수를 늘임으로써 전체 시스템의 성능을 점진적으로 향상시킬 수 있는 확장성이 있는 병렬 컴퓨터(scalable computer)에 대한 연구가 활발히 이루어지고 있다. 이러한 병렬 컴퓨터 중에서, 다중컴퓨터(multicomputer)는 분산 메모리를 갖는 다중처리기로서 여러 개의 노드 컴퓨터로 이루어져 있으며, 이 노드 컴퓨터는 기억 장치가 물리적으로 분산되어 있기 때문에 상호연결망을 통해서 메시지를 주고 받음으로써 다른 노드 컴퓨터와 통신을 한다. 각 노드 컴퓨터는 프로세서와 지역 메모리 그리고 통신 장치인 라우터로 구성되어 있다. 다중컴퓨터의 성능은 노드 컴퓨터를 연결하는 상호연결망의 성능에 의해 큰 영향을 받으므로, 상호연결망의 성능을 향상시키는 효율적인 통신 방법이 필요하다[1].

다중컴퓨터에서 많이 사용되는 상호연결망의 형태는 *k-ary n-큐브*인데 이는 각 *n*차원마다 *k*개의 노드들로 구성되어 있다. 메쉬, 토러스 및 하이퍼큐브는 *k-ary n-큐브*의 일종이라 할 수 있다. 예를들어, 2차원 메쉬는 Intel Paragon, Touchstone DELTA과 Caltech

† 비 회 원 : 한국과학기술원 전산학과
sgkim@camars.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
jwcho@camars.kaist.ac.kr
논문접수 : 1999년 4월 27일
심사완료 : 2000년 3월 8일

Mosaic C에 채용되었다. 3차원 메쉬는 J-Machine에서, 3차원 토러스는 CRAY T3D에서 그리고 하이퍼큐브는 nCUBE-2에 채용되었다.

일반적으로 다중 컴퓨터의 통신 방법은 단일전송(unicast), 멀티캐스트(multicast), 방송(broadcast) 기법이 있다. 단일전송과 방송은 목적 노드가 하나이거나 모두인 경우의 멀티캐스트의 특수한 경우이다. 멀티캐스트는 병렬시스템에서 널리 사용되는 집합체 통신[12, 15, 16]의 일종이다.

최근 다중 전송에 대한 연구가 활발히 이루어지고 있다 [5, 6, 11, 13]. 기존의 일반적인 멀티캐스트 알고리즘들은 단일 멀티캐스트(single multicast)만을 고려한 것이다. 그러나, 분산 시스템에서의 캐쉬 무효화(cache-invalidation), 과학용 계산에서의 다중 멀티캐스트, 동시 경계선 동기화(barrier synchronization) 연산에서의 다중 방송 및 멀티캐스트에서의 연산은 2개 이상의 멀티캐스트가 동시에 이루어져야 한다. 이러한 연산들을 다중 멀티캐스트(multiple multicast)라고 부르며 여러 개의 멀티캐스트는 목적지 노드가 서로 중복될 수 있기 때문에 노드들간에 충돌이 발생할 수 있게 된다. 효율적인 멀티캐스트 시스템에서는 개시 지연 시간(start-up latency)을 최소화하는 것이 필요하다[13].

기존에 제시된 멀티캐스트 알고리즘은 다중 멀티캐스트를 구현하기 위하여 각 원천지 노드들에서 단일 멀티캐스트 알고리즘을 그대로 사용한다. 목적지 노드들이 중복되면 전반적인 지연시간에 많은 영향을 미치게 된다. 여러 멀티캐스트의 원천지와 목적지의 전반적인 분포를 안다면 지연 시간을 줄일 수가 있으나 이러한 지식을 모든 노드들이 안다는 것은 현실적이지 못한 문제점을 가지고 있다. 본 논문에서는 지연시간을 줄일 수 있는 2차원 메쉬망에서의 효율적인 멀티캐스트 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문에서 사용되는 시스템 모델에 대하여 설명하고 3절에서는 기존의 멀티캐스트 알고리즘에 대해서 기술하며, 4절에서는 메쉬 구조를 갖는 네트워크에서의 효율적인 다중 멀티캐스트 알고리즘을 제안한다. 5절에서 제안한 네트워크의 멀티캐스트 알고리즘을 시뮬레이션을 통해서 성능을 비교하고, 6절에서 결론을 맺는다.

2. 시스템 모델(System Model)

메시지 개시지연(startup latency)과 네트워크 지연(network latency), 그리고 불통 시간(blocking time)의 합으로 이루어지는 메시지 통신지연은 다중컴퓨터 시스템의 성능을 측정하는 척도이다[15]. 메시지 개시지연은 네트워크 시스템이 출발노드와 목적 노드 두 곳에서 메시지를 처리하는데 소요되는 시간이고, 네트워크 지연은 메시지의 시작 플릿이 출발 노드에서 네트워크에 들어간 시각부터 메시지의 끝 플릿이 목적 노드에 들어올 시각까지 소요된 시간이다. 불통 시간은 메시지의 전송 기간 동안에 일어나는 가능한 모든 지연을 포함하고,

예를 들어, 망에서 통신 자원을 할당받기 위해서 경쟁하는데 소요되는 지연이 있다.

멀티캐스트의 성능을 측정하기 위한 멀티캐스트 지연(multicast latency)는 출발 노드가 메시지의 첫 복사본을 보내는 시각부터 마지막의 목적 노드가 메시지를 받는 시각까지 소요된 시간이다. 멀티캐스트 지연은 시스템 구조의 특성에 많은 영향을 받으며, 본 논문에서는 네가지의 특성으로 구분지어지는 시스템 구조에서 효율적인 멀티캐스트 알고리즘을 구현하는 방법을 제안한다.

다중컴퓨터 시스템 구조는 메시지를 전달하는 스위칭 방식에 의해 특징지워진다. 대부분의 다중컴퓨터는 메시지를 여러 개의 고정된 크기의 플릿(flit)으로 분할하여 각각을 네트워크를 통해서 파이프라인 형태로 목적 노드까지 전달하는 웜홀(wormhole) 스위칭 방식이 널리 쓰이고 있다[4].

메시지의 길이가 길 경우에, 웜홀 라우팅의 파이프라인 효과로 인하여 메시지가 가야할 거리는 네트워크 지연에 큰 영향을 주지 않는다[15]. 메시지의 길이가 짧을 경우에, 메시지 개시지연은 단일전송 메시지에 대한 지연에 큰 영향을 미친다. 그러므로, 네트워크 자원에 대한 메시지들간의 경쟁(contention)이 없을 경우에는 웜홀 스위칭 방식에서 메시지 통신지연은 거의 거리에 무관하게 되며 메시지 개시지연 시간에 상당히 의존하게 된다. 웜홀 스위칭 방식의 가장 큰 문제점은 메시지들이 여러 채널을 점유하게 되므로 교착 상태(deadlock)가 발생할 가능성이 크다는 점이다.

다중컴퓨터 시스템 구조를 특징짓는 두 번째 요소는 네트워크의 구조이다. 본 논문에서 다루는 다중컴퓨터 시스템의 네트워크 구조인 메쉬(mesh)는 다음과 같이 정의된다. n 차원 메쉬일 경우, 각 차원을 i, $0 \le i \le n-1$이라 하고 각 차원 i에서 $k_i(\ge 2)$개의 노드를 기반으로 이루어진다면, 다차원 메쉬 구조는 $k_0 \times k_1 \times \cdots \times k_{n-2} \times k_{n-1}$개의 노드로 구성되어 있다. 임의의 노드 x는, 각 차원 i, $0 \le i \le n-1$에 대해서 $0 \le \sigma_i(x) \le k_i-1$라 하면,

$\sigma_0(x)\sigma_1(x)\ldots\sigma_{n-1}(x)$의 n개의 좌표로 정의된다. 두 노드 x와 y에 대해서, 하나의 차원 j에 대해서만 $\sigma_j(y) = \sigma$

$j(x) \pm 1$이고 j를 제외한 나머지 모든 차원 i에 대해서 $\sigma_i(y) = \sigma_i(x)$이기만 하면, x와 y는 연결되어 있다. 다중컴퓨터 시스템의 대부분의 네트워크 구조는 2차원 메쉬, 3차원 메쉬 그리고 하이퍼큐브를 포함하는 다차원 메쉬 구조의 특별한 경우이다.

다중컴퓨터 구조를 특징짓는 요소인 라우터(router)는 노드 프로세서에 개별적으로 구성되어 있으며, 노드 프로세서간의 통신을 처리한다. 그림 1의 노드 구조와 같이, 노드의 라우터가 여러 개의 외부 채널(external channels)들을 통하여 이웃의 라우터와 연결되는데 이러한 외부 채널이 연결되는 방식은 다중컴퓨터 네트워크의 구조에 의해 결정된다. 라우터로 들어오는 메시지들이 서로 다른 채널을 통해서 나가려고 한다면, 라우터는 여러 개의 메시지를 동시에 전달할 수 있다. 그와 더불어, 노드의 라우터와 이웃하는 라우터 사이에서 서로 반대의 방향으로 두개의 메시지를 동시에 전송하는 것도 가능하다. 라우터는 노드 프로세서와 여러 개의 내부 채널(internal channels)들로 연결되어 있다. 내부 채널은 노드 프로세서에서 들어오는 입력 채널과 노드 프로세서로 들어가는 출력 채널로 구분된다.

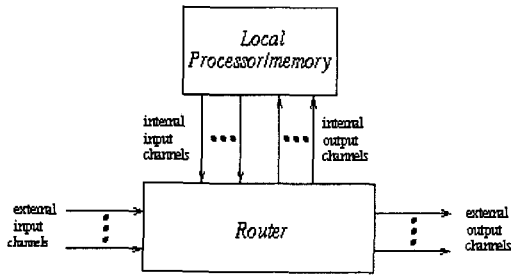

그림 1 일반적인 노드의 구조

다중컴퓨터 구조를 특징짓는 네 번째 요소로 라우터가 동시에 복사할 수 있는 메시지의 갯수와 라우터로 노드 프로세서가 동시에 보낼 수 있는 메시지의 개수인 통신 출구의 개수이다. 단출구 통신은 메시지를 보내는 단계마다 최대 두 배만큼의 노드 프로세서가 메시지를 받을 수 있기 때문에 m개의 멀티캐스트의 목적 노드로 메시지를 보내는 경우에 최소 $\lceil \log_2(m+1) \rceil$ 만큼의 단계가 메시지 전송 단계에 필요하다. 대부분의 멀티캐스트 알고리즘은 단출구 통신만을 지원하는 라우터를 사용하며 본 논문에서도 이를 가정한다.

노드 프로세서와 라우터가 동시에 여러 개의 메시지를 전송하기 위해서는 다음과 같은 기능이 필요하다. 노드 프로세서는 그림 1에서와 같이 라우터에서 노드 프로세서로 출력되는 메시지가 사용하는 출력 채널이 출력 버퍼로 향하는 채널과 입력 채널로 향하는 채널 두 가지로 구분된다. 라우터는 출력 채널로 들어오는 메시지를 여러 개 복사할 수 있는 기능과 출력 버퍼로 향하는 채널로 하나를 보내고 나머지를 입력 버퍼로 향하는 채널로 보내는 기능이 필요하다. 이러한 기능을 갖는 라우터로써 동시에 여러 개의 메시지를 전송할 수 있는 다출구 통신을 이용한 멀티캐스트를 구현할 수 있다.

3. 기존 연구

멀티캐스트를 구현하기 위한 가장 간단한 방법[2]은 목적지 노드의 갯수만큼 단일전송 메시지를 생성하여 전송하는 방법이다. 그러나, 이 방법은 너무 많은 메시지가 생성되어 성능이 떨어지는 문제점을 가지고 있다. 메시지가 전송되는 중간 노드에서 메시지를 복사하여 메시지가 트리 형태로 생성되는 방법인 Steiner Tree 기법[10]과 multicast tree 기법[9]은 교착상태(*deadlock*)가 발생하는 문제점이 있다. *U-mesh* 알고리즘[13]은 e-cube 라우팅을 이용하여 차원 순서대로 메시지를 전달하는 기법이다. 교착상태 문제를 해결하기 위해 네트워크를 2개 혹은 4개로 분할한 뒤 메시지가 해밀톤 경로를 따라서 전달되는 Dual-Path 알고리즘과 Multi-Path 알고리즘이 제안되었다[11].

또, *HL* 기법[17]을 기반으로한 다중 목적지 메시지 전달 기법(*multidestination message passing*)을 사용한 멀티캐스트 기법도 소개되었다. 그러나, 제안된 알고리즘들은 단일 멀티캐스트를 위주로 고안되었기 때문에 다중 멀티캐스트에서는 노드 충돌(node contention) 문제가 발생할 수 있다.

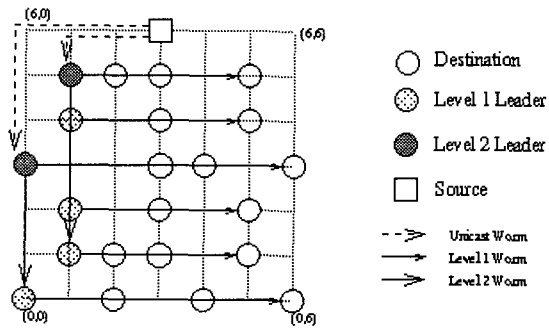

그림 2 메쉬에서 *HL* 방법을 이용한 멀티캐스트의 예

HL 기법에서의 노드충돌 문제를 그림 2에서 원천지 노드가 (6,3)인 7 x 7 메쉬의 예를 통하여 살펴본다. 첫 번째 단계에서 차원 0를 따라서 레벨 1인 노드를 선택한다. 레벨 1 노드는 (5,1), (4,1), (3,0), (2,1), (1,1), (0,0) 이다. 다음 단계에서 차원 1을 따라서 레벨 1 노드들 중에서 레벨 2 노드들을 결정한다. 레벨 2 노드는 (5,1)과 (3,0) 이다. 이 예에서는 6개의 레벨 1 노드와 2개의 레벨 2 노드로 구성되어 있다. 다음 단계에서는 실제적인 다중 전송이 일어난다. 즉, 전송 1단계에서는 원천지 노드에서 2개의 레벨 2 노드에게 *U-mesh* 알고리즘[13]을 사용하여 메시지를 내보낸다. 그 다음 단계에서 이 레벨 2 노드들은 6개의 레벨 1 노드에게 다중 목적지 메세지를 내 보낸다. 마지막으로 모든 레벨 1 노드들은 차원 0을 따라서 남아있는 그 이외의 목적지로 메시지를 내보낸다.

이러한 방법의 문제점은 다중 멀티캐스트의 경우 노드간의 충돌인데 이 문제점을 해결한 것이 *SQHL* 혹은 *SCHL* [7, 8] 알고리즘이다. *HL* 알고리즘에서는 항상 한 방향으로만 메시지가 생성되어 다중 멀티캐스트의 경우 나쁜 성능을 나타내지만 *SQHL*에서는 원천지의 위치에 따라 메시지의 생성을 다르게 하여 노드 충돌을 줄이는 시도를 하였다. *SCHL*은 원천지의 위치에 따라 4가지 방향으로 메시지를 생성하여 *SQHL*보다 더 적은 노드 충돌을 일으키도록 시도한 알고리즘이다.

그러나, 기존의 연구[7, 8]에서는 대부분의 레벨 1 리더 이외 대부분의 메시지들이 차원 1을 따라서 생성됨을 알 수 있다. 만약 대부분의 목적지 노드들이 차원 1(그리고, 0)을 따라서 위치해 있다면 많은 수의 메시지들이 차원 0(그리고, 1)을 따라서 생성이 될 것이다. 이는 차원 0의 채널을 과다 사용을 초래하게 되며 따라서 지연 시간이 길어지게 된다. 이러한 문제를 해결하기 위하여 가능한 한 모든 차원의 채널을 균일하게 사용할 수 있는 정책이 필요하다. 기존의 연구에서는 차원 0에서부터 시작하여 레벨 1 노드들이 생성되었지만 본 논문에서는 레벨 1 노드들이 차원 *(n-1)*에서 시작하는 것이 가능하다. 이러한 정책은 가능하면 채널을 균일하게 사용할 수가 있게 한다.

4. 다중 멀티캐스트 알고리즘

본 절에서는 필요한 가정과 정의에 대하여 설명한 뒤 가용 채널을 최대한 사용할 수 있는 다중 멀티캐스트를 위한 3가지 알고리즘을 제시한다.

4.1 기본 사항

목적지 노드가 하나인 유니캐스트(unicast) 메시지는 헤더 플릿, 하나 이상의 데이타 플릿과 테일 플릿으로 구성되어 있다. 목적지가 2개 이상인 다중 목적지 메시지는 헤더 플릿이 목적지 노드 번호들로 구성되어 있는 점을 제외하고는 유니캐스트 메시지의 형태와 같다. 목적지 노드를 헤더 플릿에 인코딩하는 방법에 대해서 많은 연구[3]가 진행되고 있는데 본 논문에서는 목적지 노드의 갯수만큼 플릿이 형성되는 all-destination encoding 방법을 사용한다. 본 논문에서 추가로 필요한 가정은 다음과 같다.

- 웜홀 라우팅 방식이 사용된다.
- 네트워크의 구조는 메쉬이다.
- 노드에서 한번에 하나의 메시지만을 주고 받을 수 있는 단일 출구(one-port) 통신을 가정한다.
- 메시지의 형태는 다중 목적지 메시지 전달 기법(multidestination message passing)을 사용한다.

정의 1 **순방향 리더(*BHL*)**는 [17]에서 정의된 리더와 동일한 의미이다. **역방향 리더(*RHL*)**는 차원 순서가 순방향 리더와는 반대 방향으로 이루어진 것이다. 예를 들어, 2차원 시스템에서는 레벨 1 *RHL*은 차원 1부터 그루핑이 이루어진 뒤 차원 0을 따라서 레벨 0 *RHL*이 정의된다.

정의 2 *BHL*(혹은 *RHL*)을 사용하여 생성되는 메시지의 수를 각각 N_{BHL}(혹은 N_{RHL})으로 표시한다.

정의 3 원천지 노드 (s_x, s_y)와 목적지 노드들 $G = \{d_1, d_2, ..., d_w\}$가 주워졌을 때 이 목적지 노드들은 다음의 규칙에 따라서 그룹을 결정짓게 된다.

$G^1_s = \{d_i \mid (s_x+1, s_y+1) \le d_i \le (N-1,N-1)\}$,
$G^2_s = \{d_i \mid (s_x+1, 0) \le d_i \le (N-1,s_y)\}$,
$G^3s = \{d_i \mid (0, 0) \le d_i \le (s_x,s_y)\}$,
$G^4_s = \{d_i \mid (0, s_y+1) \le d_i \le (s_x,N-1)\}$, 단, $i = 1,2, ..., w$

BHL_i와 RHL_i가 각 그룹별(G^i_s, $0 \le i \le 4$) 로 정의된다.

레벨 1과 2는 다음과 같이 정하여 진다. 편의상 *BHL*을 기준으로 설명한다. 목적지 노드 중에서 각 행(차원 0)별로 제일 왼쪽에 위치한 목적지 노드가 바로 레벨 1 노드가 된다. 이 레벨 1 노드들 중에서 각 열(차원 1)별로 제일 위쪽에 위치한 목적지 노드가 레벨 2 노드로 결정된다. 그림 2에서 (5, *) 노드들 중에서 (5,1), (4, *)중에서 (4,1), (3, *)중에서 (3,0), (2, *) 노

드들 중에서 (2,1), (1,*)중에서 (1, 1), (0,*)중에서 (0,0)이 레벨 1 노드가 되며(점으로 채워진 노드) 이 레벨 1 노드들 중 (3,0)과 (5,1)이 레벨 2 노드로(검정으로 채워진 노드) 결정됨을 볼 수 있다. 한편, *RHL*인 경우는 차원을 서로 바꾸어 고려하면 된다. 레벨 1 노드는 그 행에 속한 모든 노드들에 대한 메시지 전송을 담당하기 때문에 목적지 노드 수가 많아 지더라도 추가적인 메시지 생성이 없이 메시지가 전달될 수 있다.

그림 3은 7 x 7의 메쉬에서 *RHL*의 예를 나타낸다. 이 예에서는 목적지의 노드 수가 차원 0을 따라서 적을 때는 *BHL*이 적당한 선택이며 차원 1을 따라 숫자가 많으면 *RHL*이 좋은 선택임을 알 수 있다. 그러므로, *BHL* 혹은 *RHL*을 적절히 선택하면 다중 목적지 웜의 수를 줄일 수 있음을 알 수 있다.

그림 3에서 보듯이 모든 채널들을 골고루 사용하기 위하여 *RHL*가 사용됨을 알 수 있다. 즉, 차원 0을 따라서 목적지 노드들이 조밀하게 배치가 되었다면 *RHL*을 사용하는 것이 더 적절한 선택임을 알 수 있고 차원 1을 따라서 목적지 노드들이 조밀하게 배치가 되었다면 *BHL*을 사용하는 것이 더 적절한 선택임을 알 수 있다. *BHL*과 *RHL* 중에서 어떤 것을 선택하는 것은 정책에 따라서 달라질 수 있다.

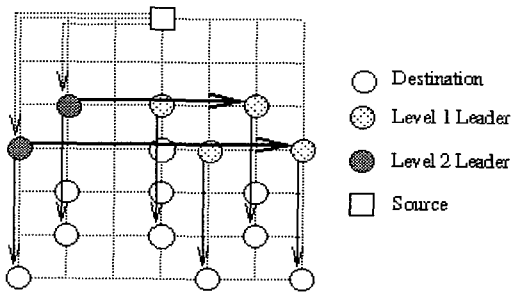

그림 3 2차원 메쉬에서의 역방향 리더의 예

4.2 알고리즘 A1

원천지 노드가 s이고 목적지 노드의 집합이 D인 멀티캐스트를 가정한다. 이 알고리즘에서는 원천지 노드의 위치에 따라 목적지 노드가 4개 그룹으로 분류가 된다. 각각의 그룹에서 *BHL*과 *RHL*를 결정한다. 그런 다음 N_{BHL}와 N_{RHL}을 비교하여 값이 작은 *HL*을 선택한다. 각 4개의 그룹에서 *HL*이 정해지면 다중 전송은 다음과 같이 진행된다. 원천지 노드는 레벨 2 노드들로 구성된 목적지들에게 *U-mesh* 라우팅 알고리즘[13]을 사용하여 다중 웜을 내보낸다. 다음 페이즈에서는 이 레벨 2 노드들이 *BHL* 혹은 *RHL*에 의해서 결정된 차원을 따라서 레벨 1 노드들에게 다중 웜을 전송하게 된다. 마지막 페이즈에서는 레벨 2 노드들을 포함한 레벨 1 노드들은 나머지 목적지 노드들에게 이전 페이즈에서 사용치 않은 다른 차원을 따라서 다중 웜을 전송한다. 알고리즘은 그림 4에 나타나 있다.

Input: 원천지 노드 $s = (s_x, s_y)$, n개의 목적지 노드 집합 $G = \{d_1, d_2, ..., d_n\}$

Output: 모든 목적지 노드가 메시지를 수신.

Procedure:

1. G를 4개의 그룹으로 분할한다. ($G^i_{x,y}(1 \le i \le 4)$)
2. 각 그룹별로 BHL_i과 RHL_i를 각각 결정한다.
3. 각 그룹별로 예상 생성 메시지 수 N_{HBL}과 N_{RHL}를 계산한다.
4. 각 그룹별로 적은 수를 가진 *HL*을 선택한다.
5. 멀티캐스트를 다음과 같이 수행한다.
   - 5.1 노드 $s = (s_x, s_y)$는 레벨 2노드로 *U-mesh* 알고리즘을 이용하여 다중목적지 웜을 전송한다.
   - 5.2 레벨 2 리더는 레벨 1 노드로 다중목적지 웜을 전송한다.
   - 5.3 모든 레벨 1, 2 리더는 나머지 노드로 다중목적지 웜을 전송한다.

그림 4 알고리즘 *A1*

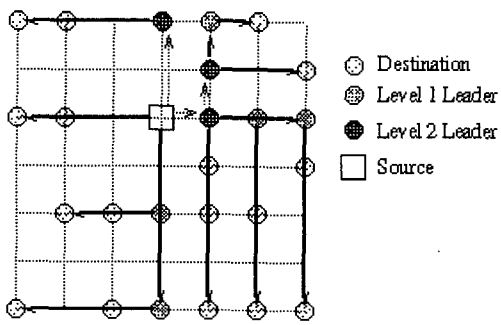

그림 5 2차원 메쉬에서의 다중 전송 예

그림 5는 알고리즘 A1에 의한 멀티캐스트 예를 보여주고 있다. G_1, G_2 및 G_3에 대해서는 *BHL*와 *RHL*에 대한 예상 생성 메시지 수가 같기 때문에 *HL*로서 *BHL*을 선택한다. 그리고, G_4에 대해서는 *BHL*에 비해서 예상 생성 메시지 수가 적은 *RHL*을 *HL*로 선택한다.

정리 1 알고리즘 A1은 교착 상태가 발생하지 않는다.

증명 모든 레벨 2 노드들은 교착 상태가 없는 *U-mesh* 알고리즘을 사용하여 메시지를 수신한다. 레벨 2 노드들이 메시지를 수신한 후 나머지 목적지 노드들은 모두 채널이 서로 독립적(disjoint)임을 알 수 있다. 각 4개의 그룹 중 $G_{X,Y}^{I}$인 경우만 고려해 보면 나머지 그룹들은 마찬가지 방법으로 증명할 수 있다. 레벨 2 리더가 레벨 1 리더에게 메시지를 전송하는 동안 메시지들은 차원을 바꾸지 않는다. 또한, 레벨 1 노드들이 리더 이외의 노드에게 메시지를 전송하는 동안 채널의 방향을 변경하지 않는다. 그러므로, 알고리즘 A1은 교착 상태가 발생하지 않는다.

알고리즘 A1은 각 그룹별로 예상되는 메시지의 생성 수가 적은 *HL*을 선택하여 멀티캐스트 메시지를 생성하기 때문에 기존의 *SCHL*보다 적은 수의 메시지를 생성한다. 결과적으로 모든 차원의 채널이 균일하게 사용됨으로 자원을 효율적으로 사용하게 되어 전반적인 멀티캐스트 지연시간이 감소하게 된다.

4.3 알고리즘 A2와 A3

알고리즘 A2는 알고리즘 A1에서와 같이 4개의 그룹으로 목적지를 분류한다. 그러나, 알고리즘 A1와는 달리 *HL*을 선택하는 방법은 모든 그룹이 *BHL*과 *RHL*을 동시에 선택된다는 점이다. *BHL* 혹은 *RHL*을 선택하는 방법은 알고리즘 A2에서는 총 예상 생성 메시지 수가 적은 쪽을 선택하고, 알고리즘 A3에서는 *BHL* 혹은 *RHL*을 랜덤하게 선택한다. 이 후의 다중 전송 알고리즘은 알고리즘 A1과 동일하다. 알고리즘 A2와 A3가 그림 6과 8에 각각 설명되어 있다

Input: 원천지 노드 $s = (s_x, s_y)$, n개의 목적지 노드 집합 $G = \{d_1, d_2, ..., d_n\}$

Output: 모든 목적지 노드가 메시지를 수신.

Procedure:

1. G를 4개의 그룹으로 분할한다. ($G^i_{x,y}(1 \le i \le 4)$)
2. 각 그룹별로 BHL_i과 RHL_i를 각각 결정한다.
3. 각 그룹별로 예상 생성 메시지 수 N_{HBL}과 N_{RHL}를 계산한다.\\
4. 모든 그룹에 대해서 적은 수를 가진 *HL*을 선택한다.
5. 멀티캐스트를 알고리즘 *A1*과 같이 수행한다.

그림 6 알고리즘 *A2*

그림 7은 알고리즘 A2를 사용한 다중 전송 예를 보여준다. 그림 7(a)에서는 *BHL*을 사용한 *SCHL* 알고리즘의 예이고 그림 7(b)에서는 *RHL*을 사용한 알고리즘 A2의 예이다. 이 예에서는 *RHL*을 사용한 방법이 적은 수의 다중 웜을 생성하기 때문에 *SCHL*을 사용한 방법보다 더 짧은 다중 지연 시간을 갖게 된다. 또한, 알고리즘 A2와 A3은 모든 채널을 가능한 한 균일하게 사용하게 된다.

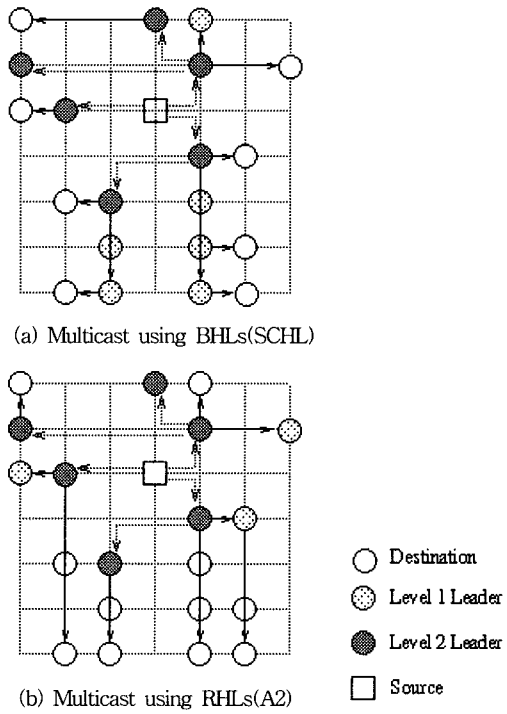

그림 7 SCHL과 알고리즘 A2를 사용한 멀티캐스트의 예

정리 2 알고리즘 A2와 A3는 교착 상태가 발생하지 않는다.

증명 정리 1과 동일한 방법으로 증명이 된다.

Input: 원천지 노드 $s = (s_x, s_y)$,
n개의 목적지 노드 집합 $G = \{d_1, d_2, ..., d_n\}$.

Output: 모든 목적지 노드가 메시지를 수신.

Procedure:

1. G를 4개의 그룹으로 분할한다. ($G^i_{x,y}(1 \le i \le 4)$)
2. 각 그룹별로 BHL_i과 RHL_i를 각각 결정한다.
3. 각 그룹별로 예상 생성 메시지 수를 계산한다.

4. 모든 그룹에 대해서 *HL*을 랜덤하게 선택한다.
5. 멀티캐스트를 알고리즘 *A1*과 같이 수행한다.

그림 8 알고리즘 A3

5. 시뮬레이션

본 절에서는 제시한 멀티캐스트 알고리즘에 대한 성능 분석을 위하여 기존의 여러 멀티캐스트 알고리즘을 시뮬레이션하고 비교하였다. 다중컴퓨터 시스템의 성능을 측정하는 척도[15]인 멀티캐스트 지연(multicast latency)은 출발 노드가 메시지의 첫 복사본을 보내는 시각부터 마지막의 목적 노드가 메시지를 받는 시각까지 소요된 시간을 의미한다.

실험에 사용된 값들은 다음과 같다. 통신 개시 시간(communication start-up time) t_s = 5 사이클, 링크 지연 시간 = 0, 노드에서의 라우팅 지연 시간 = 0, 망에 메시지를 주입하는 시간 = 50 flits/cycle, 망에서 들어오는 메시지를 소비시키는 시간 = 50 flits/cycle. 메시지의 길이 = 50 플릿 및 16 x 16 메쉬를 가정한다. 각 실험은 Dual Path 알고리즘[11](*DP*로 표시), *SCHL*[17], A1, A2, A3 알고리즘에 대해 각 30번 수행 후 평균치를 산정한 후 비교하였다.

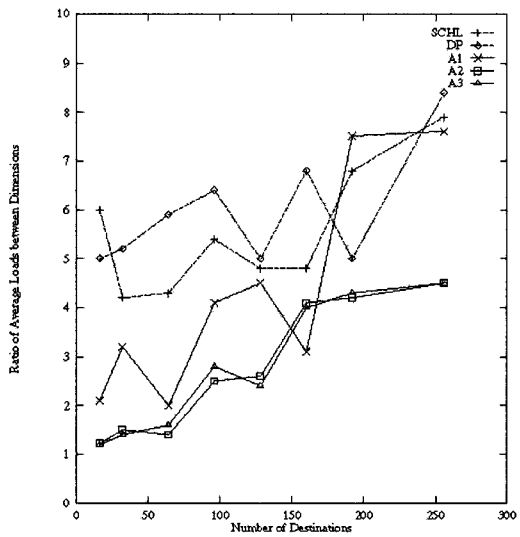

그림 9 16x16 메쉬에서의 채널 트래픽 부하 비율

그림 9는 16 x 16 메쉬에서 각 차원간의 트래픽 비율을 나타낸 것이다. 예상했던대로 *DP*와 *SCHL*의 경우가 *A2, A3*과 비교해 2배 이상의 부하 불균형이 측정되었다. 이는 *SCHL*이 차원 0 혹은 1의 채널 방향으로 메시지가 생성되는 반면 *A2*와 *A3*은 양 차원 채널 모두 골고루 사용하기 때문이다. *DP*의 경우도 메시지들이 주로 채널 0 방향으로 많이 생성되기 때문에 채널의 불균형적으로 사용된다.

그림 10은 16 x 16 메쉬에서 원천지 노드가 하나인 멀티캐스트 지연 시간을 보여준다. 전반적인 지연시간은 알고리즘 *A1, A2, A3*이 *SCHL*과 *DP*보다 짧은 것을 알 수 있다. 이는 3개의 알고리즘이 *DP*와 *SCHL*보다 채널을 더 효율적으로 사용하기 때문이다. 한가지 주목할 사항은 목적지의 노드 수가 증가하면 증가할수록 지연시간이 줄어드는데 이는 하나의 메시지가 담당하는 목적지의 수가 늘어나게 되어 총 생성되는 메시지의 수가 감소하기 때문이다.

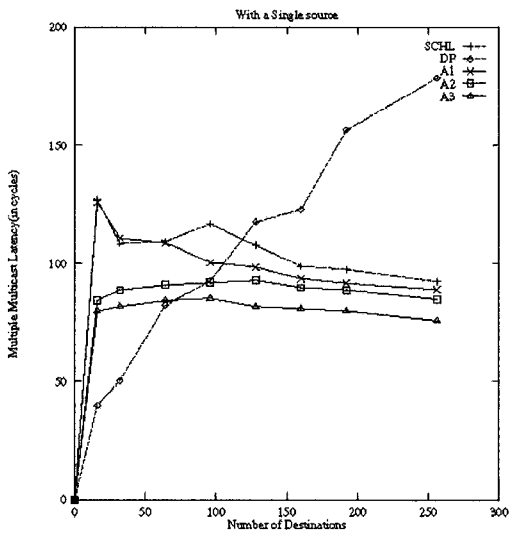

그림 10 멀티캐스트의 지연 시간(원천지가 한 개, 목적지는 여러 개)

그림 11은 고정된 128개의 원천지 노드와 다양한 갯수의 목적지 노드가 있는 다중 멀티캐스트의 경우이다. 목적지 노드의 수는 16, 32, 64, 96, 128, 160, 192, 256이다. 이 결과의 형태는 그림 10과 매우 유사한 것을 알 수 있다. 목적지의 수가 32개 이상인 경우 *A1*의 성능이 *SCHL*보다 약간 우수함을 볼 수 있다. *DP*의 경우는

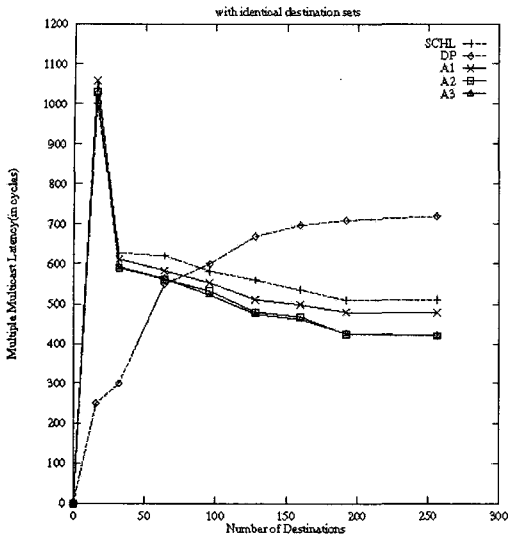

그림 11 다중멀티캐스트의 지연시간(여러 고정 개수의 목적지, 128개의 원천지)

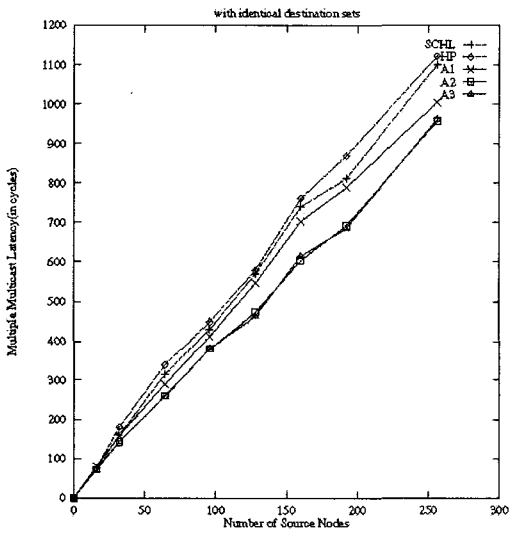

그림 12 다중 멀티캐스트의 지연시간(동일 목적지 128개, 여러 고정 개수의 원천지)

메시지의 수들이 많아짐에 따라 주로 채널 0 방향으로 메시지의 수가 많아지기 때문에 성능이 점점 나빠짐을 볼 수 있다. 그리고, *A2*와 *A3*은 *A1* 보다 약 20% 정도 더 우수한 것도 나타나 있다. 이는 *A1*은 단지 국부적으로 사용할 수 있는 채널만 고려하기 때문이다.

그림 12는 여러 개의 원천지 노드와 동일한 갯수로 이루어진 여러 종류의 목적지 노드에 대한 다중 멀티캐스트 지연 시간을 나타낸다. 목적지 노드의 수는 128개이며 원천지 노드의 수는 16, 32, 64, 96, 128, 160, 192, 256 이다. 이는 원천지 노드의 수가 증가할수록 지연 시간도 증가됨을 알 수 있다. 그림 11에 나타난 효과가 여기에서는 나타나지 않는다. *A1, A2*와 *A3*의 지연 시간은 전반적으로 *SCHL*보다 약 15% 더 좋은 성능을 보여준다.

다음은 그림 11과 12는 달리 목적지 노드들을 랜덤하게 선택하였을 때의 성능을 나타낸다.

그림 13은 128개로 이루어진 원천지 노드와 랜덤하게 이루어진 여러 개의 목적지 노드들의 시뮬레이션 결과이다. 목적지의 수가 점점 증가할수록(96개 이상) *A1, A2*와 *A3*의 지연 시간은 전반적으로 *SCHL*보다 약 20% 더 좋은 성능을 보여준다. 그림 14는 여러 개로 이루어진 원천지 노드와 랜덤하게 이루어진 여러 개의 목적지 노드들의 시뮬레이션 결과이다. 목적지 노드의 수는 128개이다.

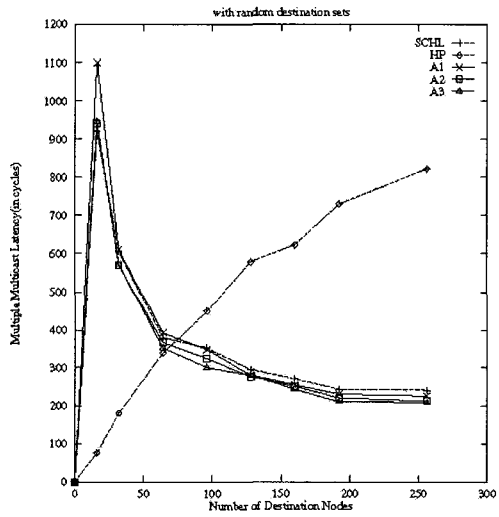

그림 13 다중 멀티캐스트의 지연 시간(128개 랜덤 목적지)

이러한 시뮬레이션의 결과에서 볼 때 *A1, A2, A3* 알고리즘은 *SCHL*보다 전반적으로 20%정도 우수하며,

*DP*의 경우는 목적지의 수가 점점 증가할수록 최고 4배 정도 우수하고, *A2*의 전반적인 성능은 *A3*과 유사함을 알 수 있다. 이는 생성되는 메시지의 수를 최소화하는 것이 점유하는 채널 수를 줄이게 되고 따라서 지연 시간이 줄어들었기 때문으로 추측된다. 이와 달리 *A1*은 *A2*와 *A3*보다 낮은 성능을 보이는데 이는 채널의 사용을 국부적으로만 고려하였기 때문이다.

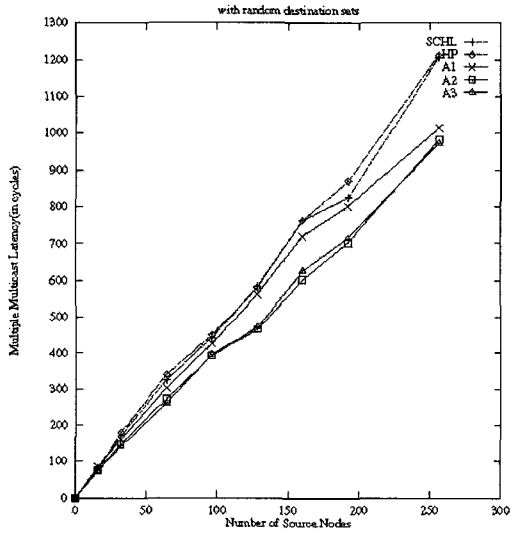

그림 14 다중 멀티캐스트의 지연 시간(가변, 랜덤 목적지)

6. 결 론

본 논문에서는 다중 멀티캐스트에서 지연 시간을 줄일 수 있는 3개의 새로운 알고리즘을 제안하였다. 다중 멀티캐스트는 목적지 노드가 서로 중복될 수 있기 때문에 노드들간에 충돌이 발생하기 때문에 메시지의 지연 시간이 길어지게 된다. 기존의 알고리즘은 하나의 멀티캐스트만을 고려한 알고리즘이기 때문에 효율적인 다중 멀티캐스트 시스템에서는 개시 지연 시간(start-up latency)을 최소화하는 것이 필요하다. 기존의 방법은 채널 사용이 한 방향(차원)으로만 주로 형성되어 노드간의 충돌을 일으켜 결과적으로 개시 지연 시간이 길어지는 문제점을 가지고 있다. 제안 알고리즘은 사용 가능한 채널들을 가능한 한 여러 방향으로 골고루 사용하기 때문에 개시 지연 시간을 줄일 수가 있다. 제안한 알고리즘들이 교착 상태가 발생하지 않는다는 것도 증명하였다. 이 알고리즘의 우수성을 보이기 위해 여러 가지 경우를 시뮬레이션을 통하여 증명하였다. 새로운 알고리즘은 성능이 기존 방법보다 약 20%정도 향상됨을 볼 수 있다.

참 고 문 헌

[1] W. Athas and C.L. Seitz, "Multicomputers: Message-Passing Concurrent Computers," *IEEE Computers*, Vol. 21, No. 8, pp.9-24, Aug. 1988.

[2] G. Byrd, N. Saraiya, and B. Delagi, "Multicast Communication in Multiprocessor Systems," *Proceedings of the 1989 International Conference on Parallel Processing*, pp. I-196-I-200, 1989.

[3] C.M. Chiang and L.M. Ni, "Multi-Address Encoding for Multicast," *Proc. Parallel Computer Routing and Comm. Workshop*, pp. 146 160, May 1994.

[4] W. J. Dally and C. L. Seitz. "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, pages 547--553, May 1987.

[5] J. Duato. "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320--1331, 1993.

[6] S. L. Johnsson and C.-T. Ho. "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Transactions on Computers*, pages 1249--1268, September 1989.

[7] Ram Kesavan and D. K. Panda. "Multiple Multicast with Minimized Node Contention on Wormhole k-ary n-cube Networks," *IEEE Transactions on Parallel and Distributed Systems*, In Press

[8] Ram Kesavan and D. K. Panda. "Minimizing Node Contention in Multiple Multicast on Wormhole k-ary n-cube Networks," *Proceedings of the International Conference on Parallel Processing*, Chicago, IL, Aug 1996.

[9] Y. Lan, A. Esfahanian, and L. Ni, "Distributed Multi-Destination Routing in Hypercube Multiprocessors," *Proceedings of the Third Conference on Hypercube Computers and Concurrent Applications*, pp. 631-639, Jan. 1988.

[10] X. Lin, and L. Ni. "Multicast Communication in Multicomputer Networks," Technical Report, Michigan State University, Dept. of Computer Science, MSU-CPS-ACS-19, Dec. 1989.

[11] X. Lin and L. M. Ni. "Deadlock-free Multicast Wormhole Routing in Multicomputer Networks," *Proceedings of the International Symposium on Computer Architecture*, pages 116--124, 1991.

[12] P. K. McKinley and D. F. Robinson. "Collective Communication in Wormhole-Routed Massively

Parallel Computers," *IEEE Computer*, pages 39-50, Dec 1995.

[13] P. K. McKinley, H. Xu, A.-H. Esfahanian, and L. M. Ni. "Unicast-based Multicast Communication in Wormhole-routed Networks," *IEEE Transactions on Parallel and Distributed Systems*, 5(12):1252-1265, Dec 1994.

[14] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, Mar 1994.

[15] L. Ni and P. K. McKinley. "A Survey of Wormhole Routing Techniques in Direct Networks," *IEEE Computer*, pages 62--76, Feb. 1993.

[16] D. K. Panda. "Issues in Designing Efficient and Practical Algorithms for Collective Communication in Wormhole-Routed Systems," *1995 Workshop on Challenges for Parallel Processing*, pages 8--15, 1995.

[17] D. K. Panda, S. Singal, and P. Prabhakaran. "Multi-destination Message Passing Mechanism Conforming to Base Wormhole Routing Scheme," *IEEE Transactions on Parallel and Distributed Systems*, In Press.

김 시 관

1982년 경북대학교 전자공학과 졸업. 1984년 한국과학기술원 전산학과 석사학위 취득. 1984년 ~ 1995년 삼성전자, LG정보통신 근무. 1994년 ~ 현재 한국과학기술원 전산학과 박사과정. 관심분야는 컴퓨터구조, 병렬처리, Wireless ATM등

조 정 완

1964년 서울대학교 공과대학 전자공학과 졸업. 1968년 와이오밍 주립대학 전기공학과 석사학위 취득. 1973년 노스웨스턴 대학 전산학과 박사학위 취득. 1968년 ~1973년 미국 IBM 연구원. 1982년 ~ 1984년 삼성전자 고문. 1985년 ~ 1988년 금성소프트웨어 대표, 고문. 1973년 ~ 현재 한국과학기술원 전산학과 교수. 관심분야는 computer architecture, parallel processing, knowledge systems, man-machine interfaces임