대용량 멀티캐스트 서버 시스템의 설계

함 진 호*

Design of a Large-Scale Multicast Server System

Jin-Ho Hahm*

요 약

멀티캐스팅에서는 호를 지원하는 통신용 버퍼, 프로세서의 복제능력, 전송대역폭 등의 자원을 모든 멤버들이 공유하는 까닭에 많은 멤버가 참여하게 될 때 자원의 결핍을 초래하여 서비스 품질(Quality of Service)을 보장하지 못하는 경우가 발생하게 된다. 이를 해결하기 위해서는 멀티캐스팅을 수행하는 멀티캐스터(multicaster)들을 계층화하여 연결하고 각각의 멀티캐스터들이 처리 능력 범위 내에서 동작되도록 하는 방안이 제시되었다.

본 논문에서는 이러한 방안을 실제 시스템에서 실현하기 위하여 하이퍼큐브 형태의 초병렬컴퓨터를 기반으로 멀티캐스터를 계층화하여 다지점통신 서버를 구현하는 방안을 제시하였으며, 이때 하이퍼큐브 컴퓨터의 노드간을 연결하는 링크의 전송 트래픽이 과도하게 증대되지 않고 안정적인 값 내에 수렴하는 것을 확인함으로써 하이퍼큐브 초병렬 컴퓨터를 기반으로 대용량 다지점통신 서버를 구성할 수 있음을 입증하였다.

Abstract

In the multicasting communications, the quality of service is not guaranteed occasionally, according to the shortage of resources, because all resources such as communication buffer, processing power for the packet multiplication, and bandwidth, should be shared among each members. This problem can be solved through the hierarchical multicaster construction mechanism which guarantees that all multicaster can be operated within the required performance.

This paper proposes the architecture of multipoint communication server based on hypercube type massively parallel computer. Through the simulation, it is verified that the traffic of links between each node of hypercube computer is under the some bandwidth. So, it is certified that the hypercube computer is suitable for the multicast communication servers supporting the various groups.

* 한국전자통신연구원 멀티미디어표준연구실

접수일자 : 1997년 11월 12일

제1장 서 론

그룹 통신 기반의 응용을 실현하는 방법은 데이터를 교환하는 형태에 따라 분산형과 집중형으로 나눌 수 있는데 분산형에서는 모든 단말이 데이터의 멀티캐스트 처리를 수행하는데 반하여 집중형에서는 서버가 데이터를 받아 멀티캐스트를 수행한다. 따라서 분산형 처리의 경우 각 단말의 성능이 비교적 우수하고 균등하여야 하며, 대규모의 그룹을 구성하는 것은 불가능하다. 집중형의 경우 서버가 멀티캐스트 처리를 전담하므로 단말의 기능이 비교적 낮아도 그룹에 참여할 수 있고 서버의 능력에 따라 대규모의 그룹을 구성할 수 있다.

그룹통신의 수행은 인터넷의 Mbone[1][2]과 같이 라우터에서 처리될 수도 있지만, *ITU-T SG16*에서 정의하고 있는 다지점 통신 프로토콜 (MCS : Multipoint Communication Service) [3][4] 과 같이 네트워크에 연결되어 있는 서버를 통하여 수행될 수도 있다. 본 논문에서 도출된 알고리즘들은 인터넷이나 ATM 네트워크와 같은 협의의 네트워크에서 적용될 수도 있지만 초병렬컴퓨터(MPP : Massively Parallel Processor) [5][6]와 같이 내부적으로 수많은 인터커넥션 네트워크를 갖고 있는 경우에 적용하는 것이 바람직하다.

본 논문에서는 하이퍼큐브 방식의 초병렬컴퓨터를 대상으로 본 논문에서 제안하고 있는 알고리즘을 적용하여 다양한 구성에 대하여 멀티캐스팅을 수행하기 위한 다지점 통신 서버를 설계할 때, 노드를 연결하는 링크 자원이 시스템의 한도 내에서 무리없이 제공될 수 있음을 보인다.

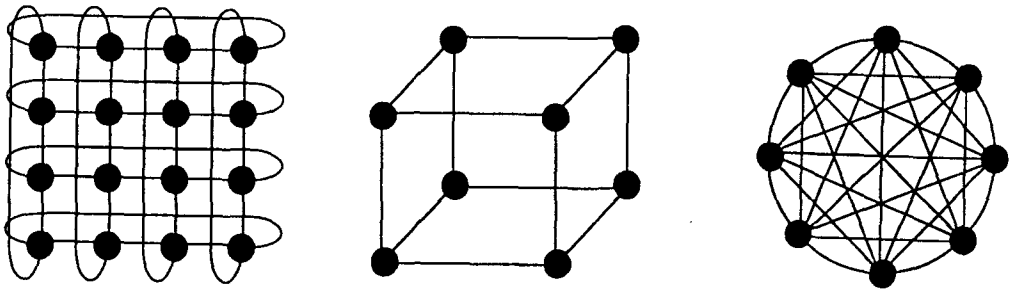
제2장 멀티캐스트 서버의 시스템 구조

2.1 개 요

초병렬컴퓨터에 대한 연구는 1980년대 초반부터 활발히 수행되었으며[5], 하이퍼큐브, 완전연결형, 메쉬형, 트리형 등 여러 구조에 대하여 스케일러빌리티, 내고장성, 복잡도, 데이터 전달의 블러킹 여부 등의 특성이 다양하게 분석되었다[7][8]. 이들 연구에서 하이퍼큐브 방식의 컴퓨터 구조는 스케일러빌리티가 있고, 노드간의 연결 경로가 다수 존재하므로 내고장성(fault-tolerant)이 높은 것으로 분석되었으며[9], 노드의 일부에 I/O기능을 두어 입출력 기능을 갖게하는 방안의 제시[10][11]와 함께 디스크 액세스와 같은 I/O 처리와 매트릭스 계산과 같은 분산된 연산작업 모두에서 상당히 효과적이라는 연구결과[12]가 있었다.

여러개의 프로세서를 병렬로 결합하여 처리 속도를 향상 시키기 위한 연결방식에는 밀결합(tightly coupled) 방식과 소결합(loosely coupled) 방식이 있는데, 프로세서가 버스 등을 통하여 메모리 등을 공유하는 밀결합 방식과는 달리 소결합방식은 프로세서와 메모리 등으로 구성된 소규모의 완전한 컴퓨터들을 네트워크를 통하여 연결한 것으로서, 각각의 컴퓨터들은 자체 운영체계를 가지면서, 전체적으로도 하나의 컴퓨터처럼 동작하는 일종의 컴퓨터 네트워크이다.

초병렬컴퓨터를 구현하는 일반적인 방법은 노드



연결메쉬형 하이퍼큐브형 완전연결형

그림 2.1 각 초병렬컴퓨터 구조

Fig. 2.1 MPP structure

들을 다양한 토폴로지의 형태로 상호연결(interconnection)하는 것인데, 완전메쉬형, 하이퍼큐브형, 완전연결형 등이 대표적인 연결 방법이고 이를 기본으로 한 다양한 변형이 있다. 이들의 구조적 성능을 평가하는 요소로는 인터커넥션 링크의 수, 링크 당 필요한 전송대역폭, 데이터 전달 시의 노드간의 평균거리, 하나의 노드에 연결되는 링크의 수 등이 있다.
그림 2.1에 각 초병렬컴퓨터의 연결 형태를 보이고 있다.

각 노드 당 처리 데이터 양을 B라고 하고, 시스템이 총 N개의 노드로 구성된다고 할 때 시스템 전체에서 처리할 수 있는 데이터 양을 B * N이다. 각 구조에서 이들의 값을 비교하면 표 2.1과 같다.

표 2.1 연결 형태에 따른 시스템 특성
Table 2.1 System characteristics for interconnection style

특 성	연결메쉬형	하이퍼큐브형	완전연결형
노드당 링크 수	4	log2N	N - 1
총 링크의 수	4*N	N*log2N	N*(N - 1)
노드간 평균 거리	N	log2N/2	1
링크당 소요대역폭	N*B/4	B/2	B/N - 1

이와 같은 수치를 바탕으로 각 구조의 특성을 비교하여 보면 시스템 전체의 처리 용량을 늘리기 위하여 노드 수를 증가시킬 때 연결메쉬형은 총 링크의 수는 비교적 적으나 노드간 거리가 증가됨으로 통신지연이 증대될 우려가 있고, 특히 링크당 처리되어야 할 트래픽의 양이 증가되므로 노드 시스템의 구성이 복잡해져야 한다. 완전연결형은 노드간이 직접 연결되므로 통신 지연은 없으나 노드당 링크 수가 급증하므로 실현 가능성이 없다.

이에 비하여 하이퍼큐브형은 링크 당 필요한 대역폭이 노드의 증가에 무관하게 일정하므로 노드 시스템의 복잡도 역시 증가하지 않으며, 노드 당 링크 수가 증가하고 있기는 하지만 로그 함수의 형태를 보이므로 노드 증가에 비하여 매우 완만하게 증가하므로 크게 문제가 되지는 않는다. 그리고 노드의 평균 거리 역시 지수적으로 증가하나 1024개의 노드를 갖는 10차원의 경우에도 평균값은 5정도 이므로 크게 문제되지는 않는다. 따라서 대용량 멀티캐스트 서버 구현에 적합한 시스템 구조로서 하이퍼큐브 방식을 선택한다.

현재 대용량 서버에 대한 연구로는 초병렬컴퓨터 구조의 내부 장치간의 연결망(interconnection network)에서의 병목현상을 해결하려는 목적으로 탠덤사의 ServerNet [13] 과 시퀀트사의 SDI (Scalable Data Interconnect)[14]에서 제시되고 있다.

ServerNet에서는 초병렬컴퓨터에서의 노드단위의 구성요소들을 6개의 입출력 포트를 갖는 라우터를 기반으로 하여 다른 라우터 또는 메모리와 프로세서를 갖는 처리장치나 디스크로 구성된 저장장치와 서로 연결하고 있다. ServerNet의 특징은 임의의 프로세서, 정보저장장치, 통신포트 등 멀티미디어 서버를 이루는 구성요소간에 임의의 연결이 가능하고 이경우 프로세서의 도움없이 여러개의 라우터를 하드웨어적으로 경유하여 최종단에 데이터를 전달하고 있고, 라우터를 육각형 벌집모양으로 계속 확장해 나감으로써 스케일러빌리티를 가지며, 듀얼포트 회로를 채택하여 신뢰성을 높이고 있다.

라우터를 통한 데이터 전송방식은 ATM과 유사한 형태의 패킷을 사용하고 있으며, 통신지연을 고려하여 64Byte의 고정길이로 되어있으며, 웜홀(wormhole) 라우팅 방식을 사용하고 있다. 라우터 연결방식에 따라 다양한 형태의 토폴로지를 구성할 수 있다. 그러나 ServerNet에 대한 적용은 정보검색을 위한 서버에만 국한되어 있으며 이러한 구조가 갖는 장점을 교환기에 적용하려는 시도는 없으며, 다양한 서버 보드를 노드에 설치함으로써 이 서버가 다양한 멀티미디어 응용을 위하여 사용되도록 하는 방안도 고려되지 않고 있다.

시퀀트사의 SDI 구조는 비교적 소용량 서버를 구성하는 방법으로서 노드단위에서 고속의 스케일러블 버스로 다수의 프로세서와 메모리, 저장장치를 연결하고, 이 노드들은 다시 640Mbyte/sec의 고속 인터커넥션 네트워크로 연결한 것이다. SDI 구조를 채택함으로써 각 응용 레벨에서 SCSI 버스로 연결된 프로세서간에 DMA 방식으로 노드간 데이터 전송을 수행함에 따라 매우 빠른 응답을 얻을 수 있다.

2.2 하이퍼큐브 초병렬컴퓨터 구조

n차원의 하이퍼큐브 구조는 N = 2n개의 노드들로 구성되어 있으며, 그림 2.2와 같은 이웃한 노드간의 연결 상태를 갖는다. 하이퍼큐브 구조에 대한 상세한 내용은 논문[15][16]을 참조바람.

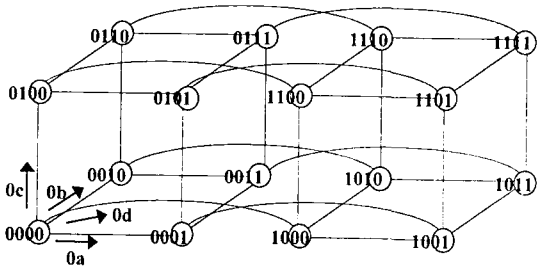


그림 2.2 하이퍼큐브 구조
Fig. 2.2 Hypercube Architecture

2.3. 하이퍼큐브 노드의 구조

아래 그림 2.3에 멀티캐스트 서버의 기능을 위한 하이퍼큐브 컴퓨터의 각 노드의 구조를 보이고 있다. 노드의 구성 요소 및 동작에 대한 상세한 설명은 논문 [15][16]을 참조 바람.

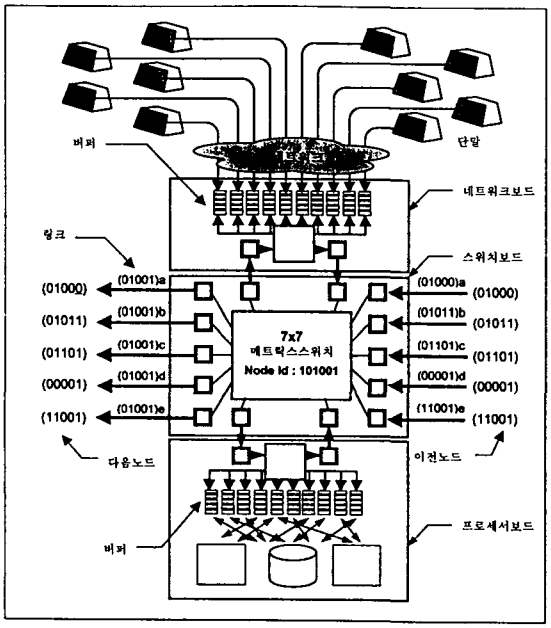


그림 2.3 멀티캐스터 서버 구성을 위한 하이퍼큐브의 노드 구조
Fig. 2.3 Node architecture of hypercube multicast server

2.4 멀티캐스팅의 구현

앞에서 멀티캐스터를 실현하기 위한 구조로서 하이퍼큐브를 선택하고 하이퍼큐브 컴퓨터가 어떻게 구성되어 있는가를 설명하였다. 본 절에서는 이를 바탕으로 멀티캐스터를 어떻게 구현할 수 있는가에 대하여 설명하고자 한다.

서버 시스템을 구성하는 5차원 하이퍼큐브 구조의 토포로지는 그림 2.4와 같으며 32 (= 25)개의 노드와 160 (= 32 * 5)개의 링크로 구성된다. 네트워크를 통하여 단말들은 하이퍼큐브의 임의 노드로 접속하며 여러개의 노드를 거쳐 멀티캐스터, 컨버터, 믹서의 기능 모듈과 연결된다. 그림 2.4는 5개의 단말이 참여하는 멀티캐스트 그룹의 연결 상태를 보이고 있다.

제3장 구현가능성 검증을 위한 시뮬레이션 환경

검증에서는 멀티캐스트 그룹을 구성하기 위하여 하이퍼큐브의 각 노드에 멀티캐스터를 생성하고 새로운 멤버들의 참여에 의해 그룹의 규모가 증대됨에 따라 노드에 생성된 멀티캐스터들이 나무 구조의 형태로 서로 연결될 때 하이퍼큐브 컴퓨터의 노드 간을 연결하는 링크가 멀티캐스터간의 연결

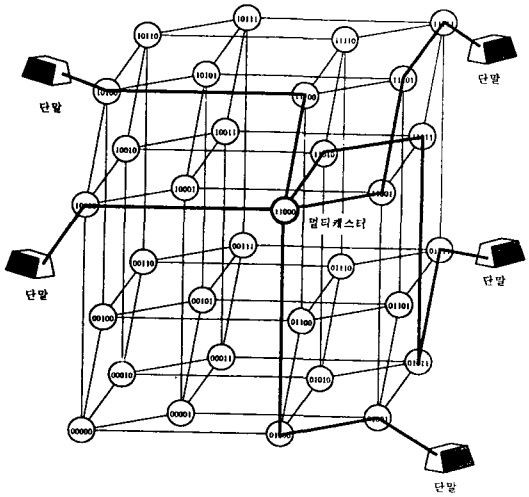


그림 2.4 5차원 하이퍼큐브 구조의 토폴로지
Fig. 2.4 Topology of 5-dimensional hypercuber architecture

을 지원할 수 있을 만큼 충분한 전송대역폭을 제공할 수 있는가를 확인한다. 이와 함께 멀티캐스터 그룹의 나무 구조의 깊이의 상황을 분석한다.

시뮬레이션 대상의 멀티캐스터 기능을 갖는 초병렬컴퓨터는 5차원의 하이퍼큐브 형태의 컴퓨터로서 25 = 32개의 노드로 구성된다. 멀티캐스터는 각각의 노드에 구성되며, 멀티캐스터 간의 데이터 전달은 하이퍼큐브의 노드를 연결하는 링크를 통하여 교환된다.

5차원 하이퍼큐브의 총 링크 갯수는 32개의 노드에 대하여 각기 이웃한 5개의 노드로 출력 링크가 연결되어 있으므로 160개가 된다. 각각의 링크가 200Mbps의 전송 대역폭을 가지고 있다고 할 때, 이 링크는 5Mbps급 MPEG 2디지틀 비디오 신호를 40개까지 전송할 수 있다.

각각의 노드에는 앞의 하이퍼큐브 노드 구조에서 보이는 바와 같이 가입자용 통신용 보드가 설치되어 있다. 시뮬레이션 환경에서 노드마다 24개의 가입자 카드가 존재한다고 가정하면 32개의 노드로 구성된 하이퍼큐브 서버에 동시에 접속할 수 있는 가입자는 총 768명이다.

본 시스템에서 접속가능한 가입자의 수는 768명이 되지만 4.5절에서 언급한 조각난 자원의 배분 문제에 따라 시스템의 효율적인 운용을 위하여 이보다 조금 적은 700 가입자에 대하여 시뮬레이션을 수행한다.

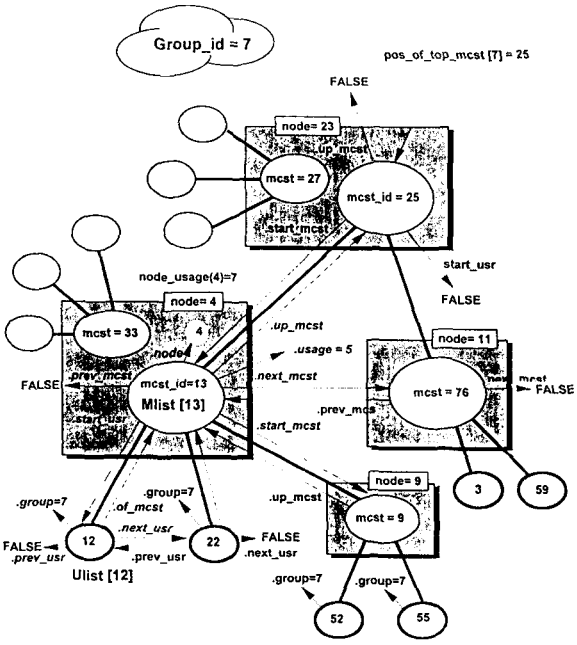


그림 3.1 시뮬레이션을 위한 데이터 구조도
Fig. 3.1 The Data Structure for the Simulation

사용자들은 네트워크를 통하여 하이퍼큐브로 구성된 멀티캐스터 서버에 접속되며, 임의의 노드에 위치하는 가입자용 보드에 무작위적으로 연결된다. 즉 멀티캐스터 서버는 접속을 요구하는 사용자들의 위치를 고려하지 않고 서버의 임의의 노드에 접속된다.

본 시뮬레이션에서는 700명의 사용자가 아래와 같이 다양한 그룹을 형성할 때 하이퍼큐브의 각 노드간 사용도를 중점적으로 분석하였다.

- 700명이 모두 하나의 그룹으로 구성되는 경우 [그림 4.2]

700명이 각각 2, 4, 8, 16, 32, 64, 128개의 그룹으로 구성되는 경우에 있어서

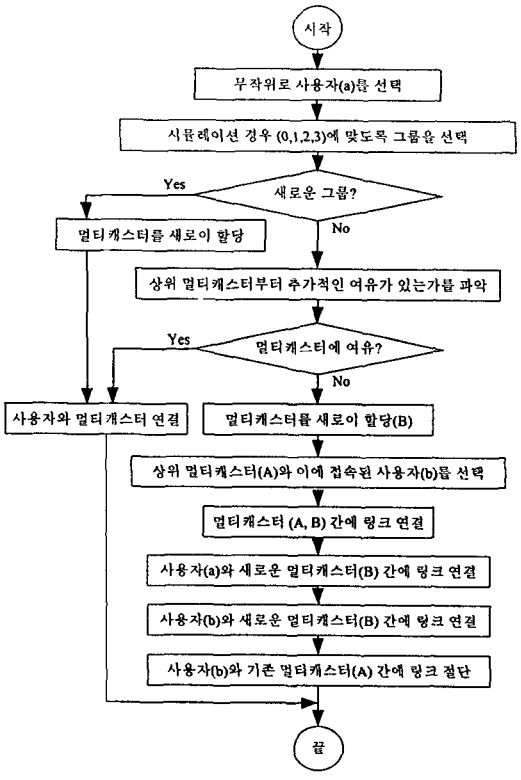


그림 3.2 시뮬레이션의 순서도
Fig. 3.2 Flow Chart of the Simulation

- 각 그룹이 모두 동일한 사용자로 나뉘는 경우 (경우1) [그림 4.3]
- 각 그룹의 사용자의 배정이 랜덤하게 결정되는 경우 (경우2): 각 그룹의 사용자 숫자 간에 약간의 불균형을 보일 수 있다. [그림 4.4]
- 사용자의 분포도가 몇몇 그룹에 편중되도록 임의로 조작한 경우(경우3) [그림 4.5]

본 시뮬레이션을 통하여 확인하고자 하는 상황은 다음과 같다.

- 알고리즘의 수행은 실현 가능한가
- 다양한 그룹의 분포에 대하여 알고리즘이 적용 가능한가
- 알고리즘을 수행하는 과정에서 어떤 링크로 부하가 과도하게 집중되지는 않는가
- 멀티캐스터의 나무 구조의 깊이가 너무 깊어지지는 않은가

시뮬레이션을 위한 데이터구조 연결도를 그림3.1에서 보였으며 시뮬레이션 순서도를 그림3.2에서 보였다.

제4장 시뮬레이션 결과의 분석

시뮬레이션을 통하여 균등, 랜덤 또는 비대칭 배열의 여러 그룹 형태와 소규모에서 대규모 멤버를 갖는 다양한 규모의 그룹을 구성하는 경우에 있어서, 멀티캐스팅을 위한 멀티캐스터가 하이퍼큐브의 노드에 생성되며, 이들간이 계층구조를 갖도록 링크로 연결함에 있어서 하이퍼큐브의 노드간을 연결하는 링크가 충분한 대역폭을 제공할 수 있음을 확인하는 것이다.

4.1 그룹의 구성

실제의 상황에 가깝도록 700명의 멤버를 다양한 그룹으로 구성하여 보았다. 이를 위해 700명을 하나의 그룹으로 구성하였으며, 700명이 각기 2, 4, 8, 16, 32, 64, 128개의 그룹으로 나뉠 때 각기 다음의 3경우와 같이 그룹 분포의 특성을 부여하였다.

- 각 그룹이 모두 동일한 사용자를 갖도록 조절한 경우
- 각 사용자의 그룹 배정을 랜덤하게 결정한 경우
- 사용자의 분포도가 몇몇 그룹에 편중되도록 임의로 조작한 경우

그림4.1은 700명의 멤버가 64개 그룹으로 나뉠 때 랜덤 배열과 편중 배열 경우에 큰 그룹부터 소팅한 각 그룹의 멤버의 숫자를 보여주고 있다. 동일한 숫자의 멤버로 각 그룹을 구성한 경우 한그룹은 11~ 12명으로 구성된다.

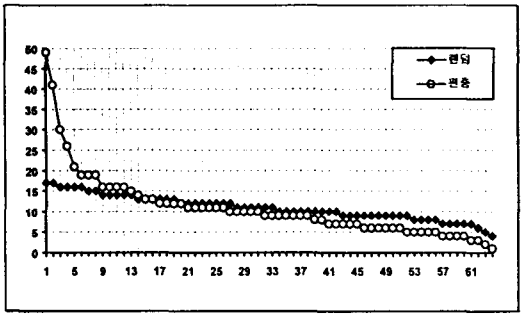


그림 4.1 64개의 그룹을 구성할 때의 각 그룹의 소팅한 멤버의 수

Fig. 4.1 The Member of each group in case of 64 groups of 700 users

4.2 각 링크의 사용도

앞에서 각 링크는 40개까지의 MPEG-2 데이터 스트림의 수용이 가능함을 언급하였다. 700명의 멤버를 하나의 멀티캐스트 그룹으로 구성하도록 시뮬레이션을 수행한 결과 각 링크를 지나는 데이터 스트림의 분포는 그림4.2와 같이 나타났다. 최대 34부터 최소12개의 데이터 스트림이 각 링크를 지나고 있으며 이 값은 설계 용량인 40을 넘지 않으므로 하이퍼큐브 멀티캐스트 서버는 무리없이 동작하리라 예상할 수 있다.

그림4.3은 700명의 멤버를 64개의 그룹으로 구성할 때, 각 그룹의 사용자수를 같도록 배정한 경우에 대한 각 링크마다의 필요한 전송대역폭을 보이고 있다.

그림4.4는 700명의 멤버를 64개의 그룹으로 구성할 때, 각기 랜덤 번호에 따른 값을 그대로 그룹 식별자로 적용한 경우이며, 그림4.5는 랜덤하게 선택된 번호에 함수를 부가함으로써 멤버의 숫자가 특정 그룹에 편중되도록 처리한 경우이다.

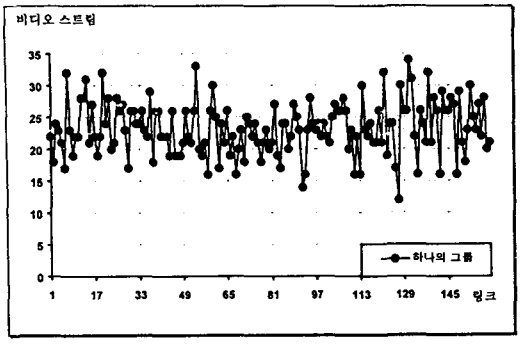


그림 4.2 700명이 하나의 그룹인 경우의 각 링크의 부하

Fig. 4.2 The traffic load of each link of hypercube, in case of single group of 700 members

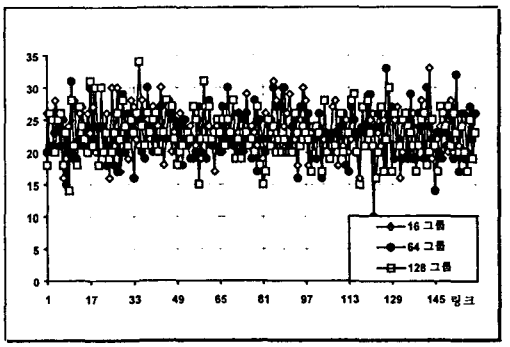


그림 4.3 700명의 참여자가 각 그룹에 균등하게 배열된 경우

Fig. 4.3 The traffic load of each link of hypercube, in case of equally-allocated groups

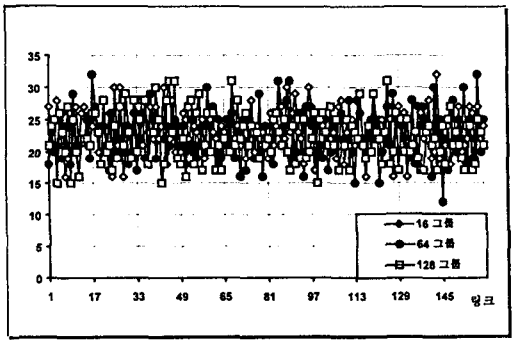


그림 4.4 700명의 참여자가 각 그룹에 랜덤하게 배열된 경우

Fig. 4.4 The traffic load of each link of hypercube, in case of randomly-allocated groups

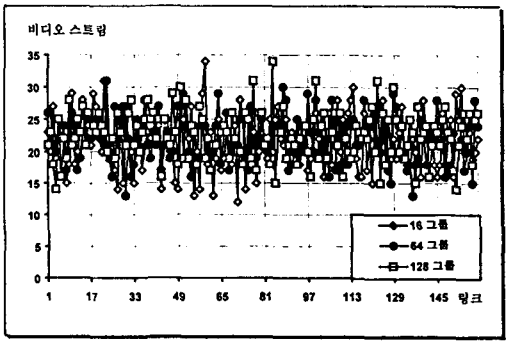


그림 4.5 700명의 참여자가 각 그룹에 비대칭적으로 배열된 경우

Fig. 4.5 The traffic load of each link of hypercube, in case of asymmetrically-allocated groups

이들 결과를 서로 비교하여 볼 때 각 링크를 지나는 데이터 스트림의 수는 서로 비슷한 분포를 나타내고 있고, 모두 35보다 작은 값을 가졌다. 이외에도 700명의 멤버를 각기 2, 4, 8, 16, 32, 128개의 그룹으로 구성하여 각 경우에 대하여 시뮬레이션을 수행하였을 때, 모두 비슷한 결과를 얻었다. 따라서 본 논문에서 제안하는 멀티캐스터 확장 메커니즘을 하이퍼큐브 구조의 컴퓨터에 적용하여 멀티캐스트 서버를 구현하는 경우에 있어 노드 간을 연결하는 링크의 전송대역폭 한도 내에서 서버가 동작됨을 확인함으로써 서버의 실현이 가능하다는 결론을 얻었다.

그림4.5의 64그룹 구성의 경우에 있어서 링크의 사용도 첫번째 값 26은 부록1에서 결과2의 00000 노드에서 00001노드로 연결된 0a 링크의 사용도를 나타내고 있으며, 두번째 값 26은 00000노드에서 00010노드로 연결된 0b링크, 세번째 값 21은 00000노드에서 00100노드로 연결된 0c 링크의 사용도를 각기 표시하고 있다.

4.3 나무구조의 깊이

나무 구조의 깊이가 깊어지면 전송지연을 초래한다. 시뮬레이션 결과를 검토한 바 700명이 하나의 그룹을 구성하는 경우를 제외하고는 나무구조

의 깊이가 최대값 2를 보였다. 나무구조의 깊이 2란 최상위 멀티캐스터 아래에 한 레벨의 멀티캐스터 만이 존재하는 경우이다. 700명이 하나의 그룹을 구성한 경우에 나무구조의 깊이는 3인 경우가 나타난다.

따라서, 나무구조의 깊이가 단대단 전송지연을 크게 유발하지 않는 범위에 머무는 것을 확인할 수 있었다.

4.4 시뮬레이션 결과

별첨한 시뮬레이션 결과에 대한 간략한 설명은 다음과 같다.

부록1의 [결과 1]은 700명의 멤버가 각 그룹에 할당되고 이에 따라 멀티캐스터가 생성되거나 기존에 멀티캐스터에 접속되는 상황에서 단말이 위치하는 소스(source) 노드부터 멀티캐스터가 위치하는 종착(destination)노드까지의 링크의 연결 과정을 보이고 있다.

서버 전체적으로는 46번째로 접속한 그룹3의 첫 번째 멤버의 경우 멤버 식별자 (user ID)는 638이며, 26번 노드를 단말노드로 사용한다. 이 멤버는 그룹 3의 최초의 참여자이므로 새로운(new) 멀티캐스터가 생성되며 멀티캐스터 식별자(multicaster ID)는 33이다. 이 멀티캐스터는 노드 1에 위치한다. 단말 노드에서 멀티캐스터 노드까지의 경로가 표시되어 있으며, 이들 노드의 거리는 4이다.

기존에 생성된 멀티캐스터에 더 이상의 멤버가 접속할 수 없어 새로운 멀티캐스터를 생성하여 해당 그룹의 기존의 멀티캐스터에 유연 절체 알고리즘을 통하여 연결하는 과정이 369식별자를 갖는 멤버에서 나타나고 있다. 이 멤버는 노드15를 단말노드로 사용하여 그룹 3에 연결하고자 하는데 기존의 멀티캐스터에 더 이상의 여유가 없으므로 새로운 멀티캐스터 65을 생성하여 이를 멀티캐스터 33에 연결한다. 이때 멀티캐스터 들이 연결될 수 있는 여유를 마련하기 위하여 멤버 599를 멀티캐스터33에서 떼어내어 멀티캐스터 65로 옮기고 멤버 599도 멀티캐스터 65에 연결한다.

[결과2]는 5차원 하이퍼큐브의 160개의 각 링크의 사용도를 보이고 있다. 노드 00000에서 노드 00001로 나가는 링크 0a의 경우 26개의 데이터 스트림이 지나고 있으며, 노드 00001에서 노드 00000으로 나가는 링크의 1a 경우 22개의 데이터 스트림이 지나고 있다.

[결과3]은 모두 64개의 멀티캐스팅 그룹에 대한 멀티캐스터의 나무 구조를 보이고 있다. 그룹0는 멤버300에서 멤버688까지의 34명의 멤버가 그룹을 이루고 있는데, 멀티캐스터 35에는 멤버 300부터 멤버 371까지 18명의 멤버와 멀티캐스터64와 멀티캐스터 77이 연결되어 있다. 멀티캐스터 64에는 멤버752부터 멤버368까지 12명의 멤버가 접속되어 있다.

4.5 조각난 자원의 배분 문제

알고리즘을 적용하여 멀티캐스터를 구성함에 있어서 시스템이 제공할 수 있는 한계 용량에 거의 접근할 경우 각 노드에는 자원의 조그만 조각들이 남게 된다. 이렇게 조각난 자원을 바탕으로 멀티캐스터로 구성할 경우 여기에 1-2 명의 사용자가 접속하게되면 포화상태에 이르게 되어 시스템의 효율이 급속도로 떨어지게 된다. 따라서 대략 시스템 용량의 90% 정도를 멀티캐스터로 사용하는 것이 바람직하며 따라서 시뮬레이션에서도 768명의 사용자에 대하여 수행하지 않고 700명에 대하여 시뮬레이션을 수행하였다.

제5장 결 론

본 논문에서는 하이퍼큐브 컴퓨터를 기반으로 하여 구축된 멀티캐스트 서버에서 멀티캐스트 확장 메커니즘이 제대로 수용될 수 있는가를 검증하기 위하여 시뮬레이션을 수행하였다. 32개의 노드로 구성되어 있고 700명 정도의 멤버를 수용할 수 있는 하이퍼큐브 시스템에서 각 멤버들이 다양한 그룹 상황을 대상으로 멀티캐스터 확장 메커니즘에 따라 그룹을 구성하도록 시뮬레이션을 수행한 결과 멤버가 접속하는 노드의 위치와 상관없이, 또 다양한 그룹 구성에 따른 멀티캐스터의 나무구조와 상관없이, 각 노드를 연결하기 위하여 필요한 전송대역폭이 일정 수준(시뮬레이션에서는 35이하) 아래로 유지됨으로써 하이퍼큐브 방식을 기반으로

멀티캐스트 서버를 실현할 수 있다는 것을 입증하였다.

시뮬레이션 상황에서는 각 노드 당 120Mbps 수준의 멀티캐스팅 능력을 갖고 있다고 가정하고, 각 노드간을 연결하는 링크의 전송대역폭이 200Mbpb 정도라고 가정하여 32개의 노드가 연합하여 약 700명을 대상으로 다양한 멀티캐스트 그룹을 구성할 수 있음을 입증하였으며 하이퍼큐브의 스케일 러빌리티 특성에 따라 노드 수를 늘림으로써 쉽게 수천, 수만명을 대상으로 멀티캐스팅 그룹을 구성하는 것이 가능하리라고 생각된다.

참 고 문 헌

[1] Kevin C. Almeroth and Mostafa H. Ammar, "Multicast Group Behavior in the Internet's Multicast Backbone (Mbone)", IEEE Communications Magazine, June 1997

[2] Hans Eriksson, "MBONE : The Multicast Backbone," Communications of the ACM, Vo.37, No.8, pp.54-60, August 1994

[3] ITU-T Recommendation T.122, Multipoint communication service for audiographics and audio-visual conferencing service definition, 1993

[4] ITU-T Recommendation T.125, Multipoint communication service protocol specification, 1996

[5] Daniel Tabak, "Multiprocessors," Prentice-Hall International, Inc., Chap.2 Multiprocessor Structure : processing and communication, pp. 11-22, 1990

[6] Howard jay Siegel, Robert J. McMillen, "The Multistate Cube : A Versatile Interconnection Network," IEEE Computer Magazine. pp.65-76, December 1981

[7] Larry D. Wittie, "Communication Structure for Large Networks of Microcomputers," IEEE Tr. on Computer, Vo.C-30, No.4 pp.264-273, April 1981

[8] Leonard S. Haynes, Richard L. Lau, Diniel P. Siewiorek, and David W. Mizell, "A Survey of Highly Parallel Computing," IEEE Computer Magazine, pp.9-24, January 1982

[9] Dhiraj K. Pradhan, and Sudhakar M. Reddy, "A Fault-Tolerant Communication Architecture for Distributed Systems," IEEE Tr. on Computers, Vol.C-31, No.9, pp.863-870, September 1982

[10] Laxmi N. Bhuyan and Dharma P. Agrawal, "Generalized Hypercube and Hyperbus Structure for a Computer Network," IEEE Tr. on Computers, Vol.C-33, No.4, pp.323-333, April 1984

[11] Kai Hwang, and Joydeep Ghosh, "Hypernet : A Communication-Efficient Architecture for Constructing Massively Parallel Computers," IEEE Tr. on Computers, Vol.C-36, No.12, pp.1450-1466, December 1987

[12] A. L. Narasimha Reddy and Prithviraj Banerjee, "Design, Analysis and Simulation of I/O Architectures for Hypercube Multiprocessors," IEEE Tr. on Parallel and Distributed Systems, Vol.1, No.2, pp.140-151, April 1990

[13] Tandem, "ServerNet : A new approach to satisfying massive data throughput requirements," Tandem, http://www.tandem. com/INFOCTR/BRFSWPS/BKTECWP/BRKTECWP.HTM, 1996

[14] Sequent, "Scalable Data Interconnect," http://www.sequent.com/public/s..chnology/sdiindex.htm#scalable, 1996

[15] 함진호, 최병욱, "주문형 비디오 서비스를 위한 하이퍼큐브 형태의 대용량 비디오 서버의 설계," 정보과학회 논문지(A), Vol.23, No.8, pp.773-785, 1996년8월

[16] 함진호, 최병욱, "멀티미디어 통신과 서비스의 효율적 처리를 위한 교환기-서버 통합구조의 제안," 한국통신학회지 논문지, Vol.21, No.11, pp.2869 - 2885, 1996년 11월

〔별 첨〕

RESULT 1 : Asymetrically Allocaton (TYPE = 3) to 64 Group. Concentration Ratio = 50 percent

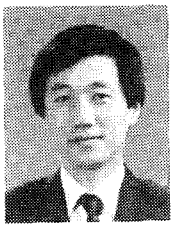
count	usr (nd) -> grp(new)	mcst(new)/(nd)	sorc node -> dest node	(?) path through links	D
46th	638 (26) -> 3 new	33 new (1)	1.1.0.1.0 -> 0.0.0.0.1	(+) 26e 10d --- 2b 0a	4
			0.0.0.0.1 -> 1.1.0.1.0	(+) 1e 17d --- 25b 27a	4
151th	730 (30) -> 3 ---	33 --- (1)	0.0.0.0.1 -> 1.1.1.1.0	(+) 1e 17d 25c 29b 31a	5
			1.1.1.1.0 -> 0.0.0.0.1	(+) 30e 14d 6c 2b 0a	5
209th	202 (8) -> 3 ---	33 --- (1)	0.0.0.0.1 -> 0.1.0.0.0	(+) --- 1d --- --- 9a	2
			0.1.0.0.0 -> 0.0.0.0.1	(+) --- 8d --- --- 0a	2
250th	45 (1) -> 3 ---	33 --- (1)	0.0.0.0.1 -> 0.0.0.0.1	(+) --- --- --- --- ---	0
			0.0.0.0.1 -> 0.0.0.0.1	(+) --- --- --- --- ---	0
351th	599 (24) -> 3 ---	33 --- (1)	0.0.0.0.1 -> 1.1.0.0.0	(+) 1e 17d --- --- 25a	3
			1.1.0.0.0 -> 0.0.0.0.1	(+) 24e 8d --- --- 0a	3
388th	369 (15) -> 3 ---	usr(nd) -> grp	0.0.0.0.1 -> 1.0.1.0.1	(+) 1e --- 17c --- ---	2
			1.0.1.0.1 -> 0.0.0.0.1	(+) 21e --- 5c --- ---	2
	upper_mcst (node) = 33	(1)	0.0.0.0.1 -> 1.1.0.0.0	(-) 1e 17d --- --- 25a	3
			1.1.0.0.0 -> 0.0.0.0.1	(-) 24e 8d --- --- 0a	3
	new_mcst (node) = 65	(21)	1.0.1.0.1 -> 0.1.1.1.1	(+) 21e 5d --- 13b ---	3
			0.1.1.1.1 -> 1.0.1.0.1	(+) 15e 31d --- 23b ---	3
	last_usr (node) = 599	(24)	1.0.1.0.1 -> 1.1.0.0.0	(+) --- 21d 29c --- 25a	3
			1.1.0.0.0 -> 1.0.1.0.1	(+) --- 24d 16c --- 20a	3
397th	435 (18) -> 3 ---	65 --- (21)	1.0.1.0.1 -> 1.0.0.1.0	(+) --- --- 21c 17b 19a	3
			1.0.0.1.0 -> 1.0.1.0.1	(+) --- --- 18c 22b 20a	3
409th	292 (12) -> 3 ---	65 --- (21)	1.0.1.0.1 -> 0.1.1.0.0	(+) 21e 5d --- --- 13a	3
			0.1.1.0.0 -> 1.0.1.0.1	(+) 12e 28d --- --- 20a	3
463th	690 (28) -> 3 ---	65 --- (21)	1.0.1.0.1 -> 1.1.1.0.0	(+) --- 21d --- --- 29a	2
			1.1.1.0.0 -> 1.0.1.0.1	(+) --- 28d --- --- 20a	2
621th	682 (28) -> 3 ---	65 --- (21)	1.0.1.0.1 -> 1.1.1.0.0	(+) --- 21d --- --- 29a	2
			1.1.1.0.0 -> 1.0.1.0.1	(+) --- 28d --- --- 20a	2
625th	453 (18) -> 3 ---	65 --- (21)	1.0.1.0.1 -> 1.0.0.1.0	(+) --- --- 21c 17b 19a	3
			1.0.0.1.0 -> 1.0.1.0.1	(+) --- --- 18c 22b 20a	3
663th	166 (6) -> 3 ---	usr(nd) -> grp	0.0.0.0.1 -> 0.1.0.1.0	(+) --- 1d --- 9b 11a	3
			0.1.0.1.0 -> 0.0.0.0.1	(+) --- 10d --- 2b 0a	3
	upper_mcst (node) = 33	(1)	0.0.0.0.1 -> 0.0.0.0.1	(-) --- --- --- --- ---	0
			0.0.0.0.1 -> 0.0.0.0.1	(-) --- --- --- --- ---	0
	new_mcst (node) = 79	(10)	0.1.0.1.0 -> 0.0.1.1.0	(+) --- 10d 2c --- ---	2
			0.0.1.1.0 -> 0.1.0.1.0	(+) --- 6d 14c --- ---	2
	last_usr (node) = 45	(1)	0.1.0.1.0 -> 0.0.0.0.1	(+) --- 10d --- 2b 0a	3
			0.0.0.0.1 -> 0.1.0.1.0	(+) --- 1d --- 9b 11a	3

```
============================================================
RESULT 2 : List-up link usage : link [0][0] ~ link [31][4]
============================================================
```

	..a	..b	..c	..d	..e
0	0a : 26	0b : 26	0c : 21	0d : 20	0e : 22
1	1a : 22	1b : 24	1c : 17	1d : 24	1e : 26
2	2a : 24	2b : 17	2c : 19	2d : 23	2e : 23
3	3a : 25	3b : 22	3c : 24	3d : 25	3e : 25
4	4a : 22	4b : 21	4c : 31	4d : 19	4e : 16
5	5a : 27	5b : 22	5c : 21	5d : 27	5e : 13
6	6a : 22	6b : 16	6c : 18	6d : 25	6e : 24
7	7a : 24	7b : 28	7c : 21	7d : 19	7e : 23
8	8a : 21	8b : 27	8c : 17	8d : 21	8e : 23
9	9a : 19	9b : 20	9c : 23	9d : 27	9e : 27
10	10a : 29	10b : 24	10c : 20	10d : 16	10e : 24
11	11a : 19	11b : 27	11c : 24	11d : 24	11e : 21
12	12a : 21	12b : 20	12c : 24	12d : 29	12e : 18
13	13a : 23	13b : 26	13c : 17	13d : 21	13e : 22
14	14a : 21	14b : 25	14c : 26	14d : 20	14e : 17
15	15a : 27	15b : 21	15c : 19	15d : 22	15e : 25
16	16a : 22	16b : 20	16c : 18	16d : 23	16e : 24
17	17a : 24	17b : 24	17c : 30	17d : 28	17e : 17
18	18a : 18	18b : 20	18c : 25	18d : 20	18e : 21
19	19a : 20	19b : 29	19c : 23	19d : 20	19e : 28
20	20a : 23	20b : 20	20c : 21	20d : 22	20e : 16
21	21a : 28	21b : 21	21c : 17	21d : 18	21e : 22
22	22a : 24	22b : 18	22c : 24	22d : 25	22e : 21
23	23a : 21	23b : 22	23c : 28	23d : 23	23e : 25
24	24a : 24	24b : 25	24c : 27	24d : 20	24e : 22
25	25a : 25	25b : 17	25c : 15	25d : 29	25e : 27
26	26a : 24	26b : 24	26c : 20	26d : 17	26e : 18
27	27a : 13	27b : 18	27c : 19	27d : 25	27e : 22
28	28a : 18	28b : 22	28c : 23	28d : 22	28e : 28
29	29a : 25	29b : 21	29c : 16	29d : 18	29e : 17
30	30a : 23	30b : 21	30c : 21	30d : 22	30e : 17
31	31a : 20	31b : 24	31c : 15	31d : 28	31e : 24

```
Total Usage of link     = 3540
Average Usage per link  = 22.12
```

```
RESULT 3 : TREE STRUCTURE OF MULTICASTERS FOR 64 GROUPS

Tree Structure Display of Group Id : 0
======================================

[ 35]---(300)./.(218)./.(591)./.( 19)./.(621)./.(325)./.(196)./.(649)./.(184)./.(610)./.(137)./.
        (704)./.(260)./.( 48)./.(402)./.( 83)./.(290)./.(371)./.

        [ 64]---(752)./.( 18)./.(103)./.(156)./.(494)./.(227)./.(491)./.(131)./.( 37)./.( 70)./.
                ( 11)./.(363)./.

        [ 77]---(339)./.(531)./.(133)./.(688)./.


Tree Structure Display of Group Id : 1
======================================

[  9]---(651)./.(467)./.(404)./.(252)./.(536)./.(472)./.(160)./.( 42)./.(552)./.(307)./.(360)./.
        (664)./.(  0)./.(763)./.


Tree Structure Display of Group Id : 2
======================================

[  7]---(281)./.(304)./.(742)./.(390)./.(128)./.(418)./.(  6)./.(412)./.(462)./.(  4)./.(377)./.
        (617)./.(583)./.(639)./.( 89)./.(421)./.(129)./.(340)./.(127)./.(736)./.(456)./.

        [ 61]---( 78)./.( 66)./.(694)./.(735)./.( 30)./.( 94)./.(121)./.( 62)./.( 41)./.(334)./.
                (761)./.(344)./.(535)./.(675)./.(507)./.

        [ 68]---(286)./.(524)./.(748)./.(625)./.(551)./.(753)./.(198)./.

        [ 73]---(670)./.(473)./.(269)./.(527)./.(662)./.(676)./.


Tree Structure Display of Group Id : 3
======================================

[ 33]---(638)./.(730)./.(202)./.

        [ 65]---(369)./.(599)./.(435)./.(292)./.(690)./.(682)./.(453)./.

        [ 79]---(166)./.( 45)./.


Tree Structure Display of Group Id : 4
======================================

[ 51]---(104)./.(483)./.(605)./.(226)./.(654)./.(513)./.(228)./.(432)./.(693)./.(381)./.

        [ 66]---(343)./.(718)./.(544)./.(478)./.(406)./.(546)./.(142)./.(268)./.(517)./.




Tree Structure Display of Group Id : 62
=======================================

[ 10]---(247)./.(  2)./.( 49)./.(627)./.(444)./.


Tree Structure Display of Group Id : 63
=======================================

[ 31]---(597)./.(305)./.(246)./.( 14)./.(216)./.
```



咸珍浩(Jin-Ho Hahm)
1982.2. 한양대학교 공과대학
전자공학과 학사
1984.2. 한양대학교 공과대학
전자통신공학과 석사
1998.2. 한양대학교 공과대학
전자통신공학과 박사
1984.3. 한국전자통신연구원 입사
1998. 현재 : 동(同)연구원 정보통신표준연구센터
멀티미디어표준연구실장
관심분야는 멀티미디어 정보통신서비스, 멀티미디어통신프로토콜, 멀티미디어서버