액티브 네트워크 구조상에서 공동작업 응용을 위한 성능 향상 기법

정회원 이 종 화*, 고 석 주*

New Design Approach for Improving the Performance of Collaborative Applications using Active Networks

Jong-Hwa Yi*, Seok-Joo Koh* *Regular Members*

요 약

대부분의 공동작업 응용들은 클라이언트-서버 패러다임을 이용하고 있어 모든 클라이언트의 요청이 일단 서버에게 전달되고, 서버에 의해 재전송되는 서버 중심의 기능과 정보 전송 기법을 사용하고 있다. 이것은 항상 서버와 클라이언트간의 양단간 통신을 발생시킴으로써 응용의 전반적인 성능을 저하시키는 요인이 되고 있다. 본 논문에서는 이러한 공동작업 응용의 비효율성을 지양하고, 응용의 요구사항에 부합되는 기능과 동작 방법을 액티브 네트워크 개념을 도입하여 제안하고자 한다. 공동작업에 참여하는 그룹의 세션정보를 기반으로 신속한 멀티캐스트를 수행하는 ASM (Active Session Multicast) 서비스, 멀티캐스트 트리 구성 서비스인 AMTC (Active Multicast Tree Constructor) 와 트리 구성 알고리즘을 제안한다. 제안하는 서비스들은 액티브 네트워크 구조상에서 공동작업 응용의 요구에 따라 네트워크가 일부 함께 동작하는 방식으로 결과적으로는 응용의 성능을 높일 수 있는 새로운 네트워크 구조 개념을 도입한 것이다.

ABSTRACT

Most of collaborative applications use the Server-Client paradigm where all requests of clients will be delivered first to server and that server distributes again the requests to the rest of clients. This server-oriented operation and data transmission mechanism always presents end-to-end communications between the server and clients that can notably decrease the overall performance of applications. In this paper, we propose a new design approach to inhibit the inefficiency of applications using active networking concepts. We propose the ASM service that offers an application-specific in-network multicast functions, the AMTC service and its tree algorithm that builds a non-core based shared tree for multicast. These proposed services locate in active nodes performing its services as part of the overall system to improve the performance of collaborative applications.

I. 서 론

여러 사용자간에 공동작업을 지원하는 응용들의 공통적인 특성은 대부분의 사용자는 지리적으로 분산된 위치에 있으며, 이들간에 공동으로 사용할 수 있는 가상의 작업 공간 (virtual working space) 을 통해 그룹 활동을 수행하게 된다. 이러한 그룹 활동을 하기 위해 사용자들은 여러 형태의 상호동작 방식 (예 : 일-대-일, 일-대-그룹, 일-대-서브 그룹, 그룹-대-그룹 등) 을 이용하게 되며, 연결된 통신망을 통해 상호 메시지를 주고 받는다. 또한, 사용자들은 능동적인 방법으로 그룹 활동에 참여할 수 있는데, 이는 임의의 그룹에 참여하거나 탈퇴하고, 동시에

* 한국전자통신연구원 표준연구센터
논문번호 : 99387-0920, 접수일자 : 1999년 9월 20일

하나 이상의 그룹 활동을 할 수 있는 등을 의미한다[1].

대부분의 공동작업 응용들은 클라이언트-서버 패러다임을 이용하고 있다. 응용 서버가 사용자간의 그룹 활동을 지원하는 데 필요한 서비스를 제공하고, 클라이언트에 해당하는 각 사용자는 서버가 제공하는 서비스를 이용하여 다른 사용자들과 상호동작 하게 된다. 이러한 패러다임을 이용하는 공동작업 응용들은 클라이언트의 모든 요청이 일단 서버에게 전달되고 서버가 해당 동작을 지시하게 된다. 일반적으로 서버 중심의 서비스 제공과 메시지 전송 방식은 서버가 동시에 관리할 수 있는 클라이언트 수가 제한되어 있기 때문에 자신이 허용하는 클라이언트 수가 넘거나 일시적으로 증가하는 경우 서비스 질이나 성능이 급격히 감소하는 현상을 나타낸다.

공동작업 응용들의 서버 중심의 기능을 지양하여 응용의 성능을 향상시키기 위한 방법중에 하나는 서버의 직접적인 제어가 필요한 경우와 그렇지 않은 경우를 구분하는 것이다. 즉, 가상 작업 공간 생성 및 삭제, 그룹에 참가 및 탈퇴 등 그룹 활동에 관련되는 클라이언트 요청은 응용 서버에게 직접적으로 전달되어 서버의 제어하에 이루어지게 하고, 그렇지 않은 경우에는 예를 들면, 한 멤버가 다른 일부 혹은 전체에게 메시지를 전달하는 경우 서버를 거치지 않고 직접 전달되게 하는 방법이다.

다른 한편으로 응용의 성능은 얼마만큼 자신이 필요한 서비스를 플랫폼을 포함한 미들웨어, 통신망, 프로토콜 등에 의해 제공 받는냐에도 달려있다. 응용은 필요한 서비스를 요구하는 입장이고, 네트워크를 포함한 하위 계층에서는 이러한 서비스를 제공해주어야 하는 관계이지만, 현실적으로는 응용과 네트워크는 매우 독립적인 방법으로 동작한다. 이것은 반대로 만약 네트워크가 동일한 서비스를 제공하는 방법에 있어 좀 더 응용의 필요성에 부합되는 방법으로 동작된다면 전체적인 응용의 성능은 매우 향상될 수 있을 것이다[2]. 이러한 맥락에서 현재 미국을 중심으로 진행되는 연구가 있는데, 기존의 네트워크 구조의 비효율성을 해결하고 새로운 네트워크 구조 및 개념을 정립하며, 네트워크가 응용의 목적에 부합되게 동작함으로써, 최종적으로는 응용과 네트워크 효율성을 동시에 향상시키고자 하는 액티브 네트워크에 관한 연구가 한창 진행 중이다. 액티브 네트워킹의 중요한 개념 중에 하나는 응용의 서버나 클라이언트 기능을 네트워크 노드에 상주시켜 동작하게 하여 네트워크도 응용의 기능 일부를 실행함으로써 전반적인 성능 향상을 높일 수 있는 구조를 제시함에 있다[3].

본 논문에서는 액티브 네트워크 개념을 이용하여 공동작업 응용들이 서버 중심의 서비스 제공과 메시지 전송 방식에 따른 비효율성을 해결하여, 공동작업 응용들의 전반적인 성능을 향상 시킬 수 있는 기법을 제안한다.

Ⅱ. 성능 향상을 위한 접근 방식

본 절에서는 공동작업 응용들이 안고 있는 서버 중심의 서비스 제공과 메시지 전송에 따른 비효율성을 해결할 수 있는 세부적인 기법을 기술한다.

공동작업 응용에서 일반적으로 세션이라는 용어를 사용하는 데, 세션이란 여러 사용자들이 그룹을 이루어 특정 목적을 갖고 상호동작하는 공동의 가상 작업 공간을 의미한다. 그룹 활동을 시작하기 위해 세션을 생성하고, 생성된 세션에 여러 참가자가 참여하게 된다. 각 참가자는 원하는 세션에 참가하여 그룹 활동에 참여하고, 세션에서 탈퇴하고, 다른 세션에 동시에 참여하는 등 매우 다양한 활동이 가능하다. 한 그룹에 속한 모든 참가자들은 공동의 가상 작업 공간을 통해 서로의 메시지를 교환하고 동시에 그룹 활동의 결과와 진행 상황을 볼 수 있게 된다. 이러한 이유에서 공동작업 응용에게는 참가자간의 신속하고 신뢰성이 보장되는 멀티캐스트 기능이 지원되어야 한다.

대부분의 공동작업 응용들은 참가자간의 메시지 교환을 서버 중심의 전송 방식을 사용하고 있다. 이것은 한 참가자가 다른 참가자에게 메시지를 전송하고자 할 때, 일단 서버에게 전달되고 서버는 해당 메시지를 모든 참가자에게 멀티캐스트 한다. 이 방식은 서버와 클라이언트간에 양단 (end-to-end) 통신이 항상 일어나게 하여 정보 전송 시간을 지연시킴으로써 전반적인 성능을 저하시킨다. 서버의 직접적인 제어가 필요 없는 이러한 경우에는 양단 전송 방식을 최대한 지양하여, 빠른 메시지 전송을 지원할 필요가 있다.

일반적으로 응용들의 개발과 동작을 위해서는 응용들이 필요로 하는 기능을 제공하는 플랫폼과 통신망 서비스가 요구된다. 그러나, 많은 경우 응용의 입장에서는 자신들의 요구사항에 적합한 서비스를 제공 받지 못하고, 단순히 플랫폼이 지원하는 서비스를 이용하는 것으로 대처한다. 따라서, 플랫폼과

통신망이 응용들의 요구사항에 적합한 기능과 서비스를 제공하고 응용 동작에 부분적으로 참여함으로써 응용들의 성능을 향상시킬 수 있다. 이러한 맥락에서, 본 논문에서는 공동작업 응용들의 빠르고 신뢰성이 보장되는 메시지 전송을 위해 응용뿐만 아니라 네트워크가 부분적으로 응용 동작에 함께 참여하여 전반적인 성능 향상을 유도하고자 한다[4].

대다수의 공동작업 응용들이 신뢰성이 보장되는 멀티캐스트를 요구하며, 이러한 신뢰성 보장형 멀티캐스트는 호스트나 네트워크의 성능 제한과 이미 잘 알려진 NACK 메시지 폭주에 따른 기술 문제를 갖고 있다. 이를 해결하기 위하여 많은 연구가 수행되었고, 그 중에 액티브 네트워크 구조상에서 이를 해결하는 흥미 있는 연구가 진행되었다. [5]에서는 NACK메시지 폭주를 방지하기 위해 효율적인 재전송 방식을 제안하는 ARM (Active Reliable Multicast) 을 개발하였다. 이 방식의 주요 내용은 각 액티브 노드 (스위치나 라우터)가 데이터를 전달받으면, 자신의 저장소에 저장한 후 다음의 연결된 노드로 전송한다. 액티브 노드가 재전송을 요청하는 NACK 메시지를 전달 받으면 해당하는 메시지가 자신이 저장하고 있는지를 확인한 후, 저장하고 있는 경우에는 직접 메시지를 재전송하고, 그렇지 않은 경우에는 자신과 연결된 노드에게 단순히 포워딩하게 되며, 최악의 경우에는 응용 서버에게 까지 전달되게 된다. 이것은 재전송 요청시 매번 응용 서버에게 전달되어 재전송이 이루어지던 기존의 방식과 비교하면 임의의 액티브 노드가 재전송을 담당할 수 있어 신속한 재전송이 이루어 진다.

또한, 공동작업에 참가하는 참가자의 동적인 행동을 고려하고 멀티캐스트 메시지 폭주를 없애기 위하여 세션에 참가하고 있는 참가자에게만 전송하는 방식을 고려하였다.

먼저, 본 논문에서는 ASM (Active Session Multicast) 서비스와 프로토콜을 제안한다. ASM은 각 액티브 노드에 상주하여 세션 관리, 참가자 정보 관리 및 참가자 정보를 바탕으로 멀티캐스트 서비스를 제공한다. 그룹이 구성되고 세션이 생성되면 각 그룹 멤버는 세션에 참가하기 시작하여 그룹 활동을 진행하게 된다. 그룹 멤버들은 서로간에 정보 교환을 위해 ASM에게 멀티캐스트 서비스를 요청하고, ASM은 멀티캐스트 라우팅 알고리즘을 이용하여 멤버가 위치한 노드에게 전달한다. 멀티캐스트 라우팅 알고리즘은 그룹 멤버들의 노드들로 구성되는 트리 구성 알고리즘을 사용하는데, 얼마만큼 효율적인 트리를 구성하느냐가 멀티캐스트 성능을 좌우하게 된다. 현재 사용되는 트리 구성 알고리즘은 크게 SBT (Source based Trees) 와 CBT (Center Based Trees) 방식으로, SBT는 데이터 전송시 지연이 적으나 하나 이상의 송신자를 고려하는 응용인 경우 라우팅 테이블과 자원 예약 관리에 따른 오버헤드가 많은 단점이 있다. CBT는 이러한 측면에서는 관리나 구현이 손쉽게 가능하나 멀티캐스트 패스 (path) 상의 코아 노드 위치를 미리 구현해야 하며, 트리 구성시 드는 비용과 지연 시간에 대한 고려를 하지 않는 단점이 있다[6][7].

본 논문에서는 ASM이 멀티캐스트 서비스 수행시 사용할 수 있는 트리 구성 서비스 (Active Multicast Tree Constructor: AMTC) 와 알고리즘 (Shared Multicast Tree: SMT) 을 제안한다. SMT는 non-core based shared tree 구성 방식의 하나로 멀티캐스트 트리 구성시 드는 비용과 전송 지연 시간을 고려하여 멀티캐스트 트리 패스를 구성한다.

다음 절에서는 위에서 언급한 접근 방식을 이용하여 공동작업 응용들의 성능 향상을 위해 제안하는 ASM과 AMTC 서비스를 자세하게 기술한다.

Ⅲ. Active Session Multicast (ASM)

본 절에서는 ASM이 제공하는 서비스와 프로토콜을 기술한다.

1. 서비스

IDL (Interface Description Language) 로 표기한 ASM의 서비스 인터페이스는 <표 1>과 같으며, 다음과 같은 서비스를 제공한다[8].

- **세션 관리** : 권리가 있는 한 사용자 요구에 따라 세션을 생성하고 삭제할 수 있다.
- **그룹 관리** : 생성된 세션에 멤버의 참여, 탈퇴, 하나 이상의 세션에 가담 등 멤버들의 전반적인 멤버 쉽을 관리한다.
- **세션 정보 관리** : 세션에 참가하는 멤버 정보는 응용 서버에 의해 관리되고, 이 정보는 각 액티브 노드에 전달되어 모든 노드들이 멀티캐스트 시 세션 정보를 참고하여 해당 멤버들의 노드에만 전송하게 된다.
- **멀티캐스트** : 임의의 멤버가 멀티캐스트를 요청하면 가장 근접하게 연결된 액티브 노드는 자신이 관리하는 세션 정보를 참고하여 해당 노드에

게 멀티캐스트하고, 이를 받은 다른 노드들도 동일한 방법으로 세션정보를 검색하여 다시 멀테캐스트함으로써 최종적으로 멤버들에게 전달된다.

재전송 : 임의의 액티브 노드가 재전송 요청을 받으면 재전송이 요청된 데이터를 자신이 저장하고 있는지를 확인하여 갖고 있는 경우 직접 전달하고, 그렇지 않은 경우에는 다음 연결된 액티브 노드에게 재전송 요청을 전달한다. 이 방식은 항상 재전송 요청시 응용서버에게까지 전달되던 기존 방식에 비해 재전송 시간이 단축된다. 재전송 기법에 대한 개념과 메커니즘은 ARM에 제안하는 방식을 이용한다[5].

2. 프로토콜

ASM 프로토콜은 액티브 네트워킹의 프로그램 로딩방식에 따라 각 액티브 노드에 상주하며, 앞서 정의한 서비스를 수행하게 된다. 본 논문에서는 어떠한 프로그램 로딩 방식을 적용하는지 액티브 노드 수에 관한 문제는 다루지 않는다. 이것은 각 노드가 액티브인 경우 본 프로토콜을 수행할 것이며, 반대인 경우에는 기존의 라우팅 기능을 수행함을 의미한다.

ASM은 ANEP 패킷을 이용하고[9][10], 패킷의 payload 부분에 위치하며, 각 액티브 노드가 갖는 프로토콜 스택은 TCP, UDP 등의 트랜스포트 프로토콜과 IP 네트워크 프로토콜을 사용한다 (<그림 1> 참조).

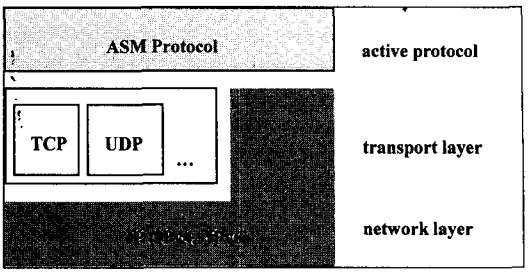


그림 1. ASM 프로토콜 스택

ASM 패킷 포맷과 각 필드의 용도는 다음과 같다.

Object _id	Session _id	Channel _id	Operation _Type	Payload

- **Object_id**: 패킷을 이용하는 서비스 식별자로 이 경우에는 ASM 서비스를 의미함.
- **Session_id**: 해당하는 세션 식별자를 의미함.
- **Channel_id**: 하나의 세션에 대해 데이터 채널과 제어 채널이 할당되며, 멤버간의 멀티캐스트는 데이터 채널로, 세션 생성 요청, 세션 정보 전달, 멤버 참여 요청 등은 제어 채널을 사용함.
- **Operation _Type**: ASM이 제공하는 서비스 유형에 따른 서비스 프리미티브를 나타냄.
- **Payload**: 멀티캐스트 할 정보에 해당함.

세션에 관한 정보는 응용 서버에 의해 직접적으

표. 1. ASM 서비스 인터페이스

```
struct SessionTable {
    long    session_id;     // identifier of a session created
    string  manager_id;             // Manager who creates a session
    string  member_list:  //    list of members who are participating in a session
};
Interface  Active_Session_Multicast_Service {
            void Create_Session (in string source, out long  session_id);
            void Join_to_Session (in string source, in long  session_id, out  boolean result);
            void Leave_to_Session (in string source, in long  session_id);
            void Delete_Session (in long  session_id);
            void Update_Session_Information (in string source, in  SessionTable table,out boolean result);
            void Multicast_Data (in string source, in string data);
            void Retransmission (in string source, in long  data_id);
};
```

로 관리된다. 하나의 세션이 생성되고, 멤버들의 참가나 탈퇴에 따라 세션 테이블 정보가 갱신되며 이 정보는 모든 액티브 노드에게 전달된다. 한 세션의 멤버는 응용 서버가 있는 단말을 비롯하여 멤버의 단말, 응용 서버와 모든 멤버간에 위치한 액티브 노드로 구성된다. <그림 2>에서는 10개의 액티브 노드, 응용 서버와 9 멤버로 구성된 경우를 나타내며, 액티브 노드간에 이어진 굵은 선은 본 논문에서 제안하는 SMT 알고리즘에 따른 멀티캐스트 트리에 해당한다. 세션 정보는 세션 식별자, 세션 생성자 그리고 각 액티브 노드에 대한 멀티캐스트 멤버 정보로 구성되며, <그림 2>에 해당하는 세션 테이블은 <표 2>와 같다. 예를 들면, 액티브 노드 A의 멀티캐스트 멤버는 (멤버1의 단말, 멤버9의 단말, 노드 H)에 해당하므로, A가 멀티캐스트 요청을 받은 경우 위의 세 멤버에게 멀티캐스트 하고, 노드 H는 자신의 멤버인 A, C, G, B에게 멀티캐스트하여 이러한 방식의 연속으로 모든 멤버들이 데이터를 전송 받는다.

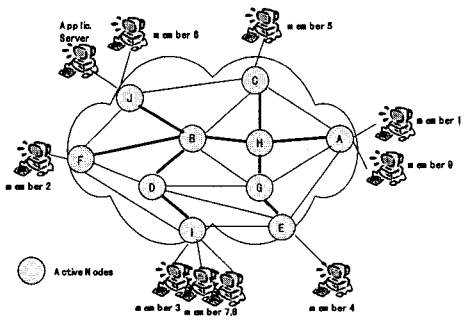


그림. 2 액티브 네트워크상에서 ASM 서비스

표 2. 세션 테이블

Session _id	session 1
Session creator	member 1
Members	A(member1, member9, H) H(A,C,G,B) C(H, member5) G(H,E) E(member5) B(J,D,H,F) J(B,application server, member6) D(B,I) F(B,member2) I(D,member3,6,7)

Ⅳ. Active Multicast Tree Constructor (AMTC)

공동작업 응용에서의 멤버 참가와 탈퇴가 계속적으로 일어나는 상황에서 AMTC는 이미 구성된 멀티캐스트 트리에 새로운 멤버 노드를 추가하거나 삭제할 수 있는 서비스를 제공한다. <표 3>에서 AMTC 서비스 인터페이스를 보여 주고 있으며, 멀티캐스트 트리를 만들기 위해서는 먼저 네트워크 환경 구성요소 (network configuration) 에 대한 정보가 필요하기 때문에 네트워크 관리자는 Set_Network_Topology_Information (....)을 이용하여 각 노드의 위치 정보, 멀티캐스트 능력, 설치 링크 비용에 대한 정보를 입력할 수 있다. Request_Tree_Information(...)은 새로운 멤버가 참가하면 기존의 멤버들로 구성되어 있는 멀티캐스트 트리에 새로운 멤버 노드가 포함될 수 있는 가장 짧은 거리를 계산하여 알려준다. 많은 멤버가 추가되거나 삭제된 상황에서는 기존의 멀티캐스트 트리가 더 이상 효율적이지 않거나 비용이 적게 드는 트리가 아닌 경우 Reconfiguration_Tree(...)를 이용하여 재구성할 수 있다.

표 3. AMTS 서비스 인터페이스

```
Interface Active_Multicast_Tree_Constructor {
        void  Set_Network_Topology_Information
        (in string node_id, in string location,
              in int multicast_capacity, in int
              link_ cost);
        Request_Tree_Information (in long
        session_ id, in long node_id,
              out string shortest_path);
        Reconfiguration_Tree (out string new_
        shortest_tree);
};
```

ASM과 AMTC 서비스가 함께 동작하는 방법을 간략하게 기술하면 다음과 같다.

- **세션 생성 경우** : 권리가 있는 한 그룹 멤버가 세션 생성을 응용 서버에게 요청하여 응용 서버가 세션을 생성하면, AMTC는 응용 서버와 세션 생성을 요청한 멤버 노드간에 가장 짧은 멀티캐스트 트리를 구성한다. <그림 2>의 경우 멤버 1과 응용 서버간의 트리는 노드 A, H, B와 J로

구성되며, 세션 테이블이 관리된다.

- **멤버의 참가와 탈퇴 경우** : 새로운 멤버가 참가 요청을 하면 AMTC는 기존의 트리를 기준으로 새로운 노드가 추가될 수 있는 가장 짧은 거리를 계산하여 알려준다. <그림 2>에서 멤버 2가 추가되는 경우 노드 F는 노드 J에 연결되거나 혹은 B로 연결될 수 있으나, 본 논문에서 제안하는 SMT 트리 구성 알고리즘에 따라 더 효율적인 B로 연결된다.
- **멀티캐스트 경우** : 각 액티브 노드에는 이미 ASM과 AMTC가 수행될 수 있는 환경이 구축되어 있고, 각 노드가 멀티캐스트 요청을 받은 경우 자신의 저장소에 데이터를 저장하고, 세션 테이블을 검색하여 자신에게 데이터를 보낸 노드를 제외한 다른 멀티캐스트 노드에게 전달한다. 저장한 데이터는 재전송 요청시 사용할 것이며, 각 노드의 저장소 용량과 필요에 따라 시간 T만큼 저장한다.
- **재전송 경우** : 각 노드가 재전송을 요청하는 패킷을 받으면 해당 데이터가 자신이 저장하고 있는지를 확인한 후 저장하고 있는 경우에는 해당 데이터를 요청한 노드에게 전달하여 최종적으로 재전송을 요청한 멤버에게까지 전달되게 한다. 이 기법은 ARM[5] 이 제안하는 재전송 방식을 따른다.

1. SMT 알고리즘

SMT (Shared Multicast Tree with delay and cost minimization) 는 전송 지연시간 (delay time)과 비용(cost)을 고려해서 멀티캐스트 트리를 구성하고, 코아 노드를 미리 설정하지 않는 트리 구성 알고리즘이다[11][12].

(1) 현재 그룹 멤버 노드에 대한 멀티캐스트 트리를 ***T***라고 하고,

(2) 새로운 노드가 가담하면 SMT는 새로운 노드가 ***T*** 에 연결될 수 있는 멀티캐스트 노드를 구한다.

(3) 새로운 노드와 멀티캐스트 노드간에 연결 가능한 모든 패스를 구하고, 그 중 전송 지연 시간과 비용을 더한 값이 가장 저렴한 패스를 결정하여 새로운 노드가 멀티캐스트 노드에 연결되어 ***T*** 에 포함되게 한다.

멀티캐스트 노드를 결정하는 방식은 다음과 같다.

$$\text{Minimize} \quad \sum_{i \in T} \{ k\, f_i(x_i) + (1-k)\, g_i(x_i) \} \tag{1}$$

$$\text{Subject to} \quad \sum_{i \in T} x_i = 1 \tag{2}$$

$$x_i = \{0, 1\} \quad \forall i \in T \tag{3}$$

Fi(xi) = 그룹 멤버 노드와 멀티캐스트 노드간에 가장 저렴한 패스

gi(xi) = 가장 저렴한 패스가 ***T*** 에 연결된 후 멀티캐스트 트리에서의 전송 지연시간의 최대치

k = 0과 1 사이로 표시되는 트리 구성 비용과 최대 지연시간을 나타내는 수치 (예를 들면, 지연시간을 중요시 하는 응용은 k=0, 트리 구성 비용이 중요시되는 응용인 경우에는 k=1을 이용)

(2)와 (3)은 하나의 멀티캐스트 노드만이 ***T***안에서 선택되도록 하기 위해 설정하였으며, 결론적으로 (1)의 조건을 만족하는 하나의 멀티캐스트 노드 i가 결정되도록 하였다.

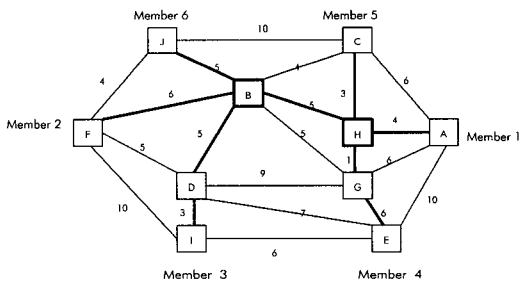


그림 3. SMT 기반 멀티캐스트 트리

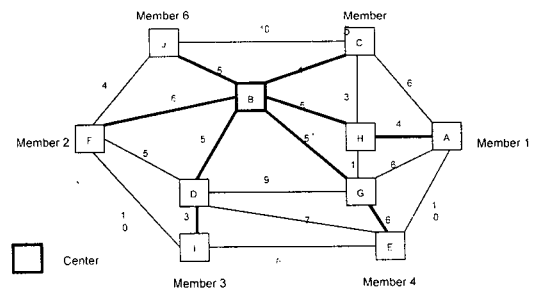


그림 4. CBT기반 멀티캐스트 트리

<그림 2>의 경우 SMT를 적용하여 구성된 멀티캐스트 트리는 <그림 3>에서 일반 CBT를 적용한 트리는 <그림 4>에서 보여준다.

2. 시뮬레이션

CBT와 SMT 알고리즘의 효율성을 비교 분석하기 위하여 간단한 시뮬레이션을 수행하였다. 시뮬레이션은 32개의 라우터와 80개의 링크로 구성된 Mbone 네트워크를 고려한 환경에서 테스트 되었으며, 그룹 멤버들은 임의의 순서로 그룹에 참가하기 위하여 네트워크에 접속하는 것으로 간주하고, 링크 비용 (link cost)은 1에서 10의 정수 (integer) 값으로 할당하였다. CBT 알고리즘에서의 코아 노드는 테스트용 네트워크상에서 임의의 한 노드를 선택하였다.

위에서 언급한 시뮬레이션 환경에서 CBT와 SMT 알고리즘의 성능비교는 트리 구성 비용 (tree cost)과 최대 양단 지연 시간 (maximum end-to-end delay)의 두 요소에 대한 결과를 측정하는 것으로 수행되었다. 정확한 결과 측정을 위하여 100번 정도의 실행을 반복하였으며, <그림 5>와 <그림 6>에서 보여주고 있는 두 알고리즘의 테스트한 결과는 평균값을 나타내고 있다. CBT 알고리즘은 미리 코아 노드를 선정하여 임의의 멤버 노드가 접속을 원할 때 코아 노드에 새로운 멤버 노드를 추가하여 트리를 구성하는 방법인 반면에, SBT 알고리즘은 임의의 멤버 노드간에 트리 구성의 비용이 저렴한 경로를 찾아 트리를 구성하는 방법이다. 이러한 특성을 갖는 두 알고리즘을 트리 구성 비용과 최대 양단 지연 시간 측면에서 비교하기 위하여 본 논문에서는 동일한 네트워크 환경에서 시뮬레이션을 수행하였기 때문에 두 알고리즘이 각 요소에 대해 나타낼 수 있는 최악의 값 (worst-case results) - 즉, 각 알고리즘의 비효율성이 나타나는 경우 - 을 측정하기는 매우 어려운 작업이었다. 따라서, 본 논문의 <그림 5>와 <그림 6>은 반복하여 수행한 테스트의 평균치를 나타내고 있다.

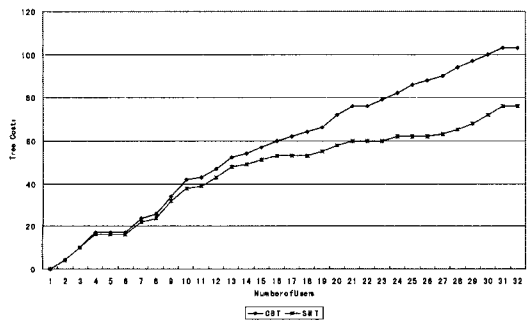


그림 5. Mbone상에서의 CBT/SMT 트리 구성 비용 비교

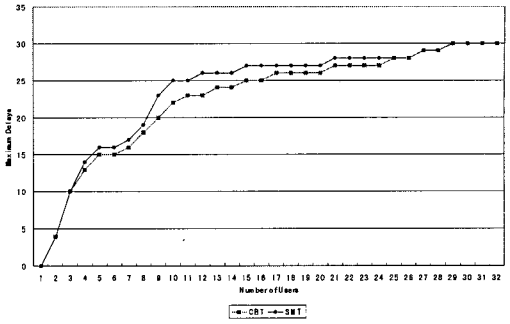


그림 6. Mbone상에서의 CBT/SMT 최대 지연시간 비교

<그림 5>에서 보여주는 바와 같이 트리 구성 비용면에서는 SMT가 전반적으로 CBT에 비해 나은 성능을 보여 주었으며, 멤버 수가 증가할 수록 SMT와 CBT간의 성능 격차가 크게 나타났다. <그림 6>의 지연시간 측면에서는 멤버 수가 4에서 24인 경우 CBT가 더 나은 성능을 보였으나, 멤버 수가 큰 경우는 두 알고리즘이 거의 비슷한 성능을 보여주었다. 따라서, 결론적으로는 본 논문에서 제안하는 SMT 알고리즘은 트리 구성 비용과 최대 양단 지연 시간면에서 멤버 수가 큰 그룹을 다룰 때 상대적으로 효율적임을 나타내었다. <표 4>는 시뮬레이션 결과를 상대적인 수치로 나타내고 있다. 트리 구성 비용에 따른 CBT의 성능을 100이라 하면 SMT는 73을 나타냄으로써 약 30%의 트리 구성 비용을 절감할 수 있음을 보여주고, 최대 지연시간에 따른 CBT의 성능을 100이라 하면 SMT도 거의 100을 나타내었다.

표 4. Mbone상에서의 CBT/SMT 알고리즘의 상대적 성능 비교

요소 \ 알고리즘	CBT	SMT
트리 구성 비용	100	73
최대 지연 시간	100	100

V. 결론

공동작업 응용들의 일반 특성과 동작 기법을 분석한 결과 대부분의 응용들이 서버 중심의 기능과 정보 전송 기법을 사용하고 있고, 이것은 서버가 동시에 관리할 수 있는 클라이언트 수가 제한되어 있기 때문에 자신이 허용하는 클라이언트 수가 넘거

나 일시적으로 증가하는 경우 서비스 질이나 성능이 급격히 감소하는 현상을 나타냄이 분석되었다. 이러한 공동작업 응용의 비효율성을 감소시키고, 응용 동작시 전반적인 성능을 높이기 위하여, 본 논문에서는 응용의 요구사항을 만족시키는 기능과 서비스를 제안함에 있어 네트워크가 일부 참여하여 최종적으로는 응용의 성능을 향상시킬 수 있는 새로운 기법을 제안하였다. 이것은 기존의 네트워크가 갖고 있는 비효율성을 없애고 새로운 네트워크 구조 및 개념을 정립하 기 위해 현재 진행 중인 액티브 네트워크 개념을 도입하였다.

그룹의 공동작업을 위한 세션 관리, 그룹 멤버 관리, 세션 정보 기본의 멀티캐스트 기능을 제공하는 ASM 서비스, 신속한 멀티캐스트를 수행하기 위해 멀티캐스트 트리 구성에 필요한 기능을 제공하는 AMTC와 트리 구성 알고리즘인 SMT를 제안하였다. 제안한 서비스들은 본 논문에서 고려한 공동작업 응용의 요구사항을 만족시키기 위해 플랫폼과 네트워크상에서 제공되어지며, 네트워크도 응용의 개발과 운용을 지원하는 시스템의 일부로서 필요한 기능을 유연하게 제공하여 응용의 전반적인 성능 향상을 유도할 수 있음을 확인해 주는 새로운 디자인 기법이라 할 수 있다.

참 고 문 헌

[1] 이종화, "개방형 분산 처리 환경에서의 다자간 대화형 응용 서비스 및 다중 세션 관리 서비스 설계", 한국정보처리학회 논문지, 제5권 제5호, 1998.6.

[2] 이종화, 함진호, "멀티미디어 서비스 프레임워크에서의 Active Networking 기술 적용 방안", 한국정보통신학회 제2권 제3호, 1998.8.

[3] Ulana Legedza, David Wetherall and John Guttag, "Improving the Performance of Distributed Applications Using Active Networks", *IEEE INFORCOM'98,* March 1998.

[4] David L. Tennehouse, Jonathan M. Smith, W. David Sincoskie, Gary J. Minden, "A Survey of Active Network Research", *IEEE Communications Magazine*, January 1997.

[5] Li-wei H. Lehman, Stephen J. Garland and David L. Tennenhouse, "Active Reliable Multicast", *IEEE INFORCOM'98,* March 1998.

[6] A. J. Ballardie, "Core Based Trees Multicast Routing Architecture", RFC2201, September, 1997.

[7] A. J. Ballardie, B. Cain and Z. Zhang, "Core Based Trees (CBT version 3) Multicast Routing: Protocol Specification", Internet-Draft: draft-ietf-idmr-cbt-spec-v3-01.txt, August, 1998.

[8] 이종화, 함진호, "Active Network 구조 및 관련 기술", 정보처리학회지 특별기고, Vol.5 No.6, 1998, 9.

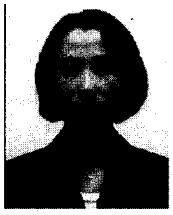
[9] David J. Wetherall, John V. Guttag and David L. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols", *IEEE OPENARCH'98,* San Francisco, CA, April 1998.

[10] Http://www.cis.upenn.edu/~switchware/ANEP/docs/ANEP.txt, "ANEP", 1997.

[11] Seok J. Koh, Myung K. Shin, Jong H. Yi, Jin H. Hahm and Chee H. Park, "Non-core based shared tree architecture for IP multicasting", *Electronics Letters,* 27th Vol.35 No.11, May 1999.

[12] S. Casner, "Major MBONE routers and links", Available from ftp.isi.edu:mbone/mbone.topology.ps, 1994.

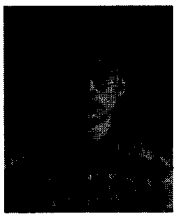
이 종 화(Jong-Hwa Yi) 정회원



1987년 8월 University of Santiago, Chile 전산학과 졸업(학사)
1990년 2월 한양대학교 대학원 전자공학과 졸업(석사)
1990년 2월~현재 한국전자통신연구원 표준연구센터 선임연구원
1996년 11월 Technical University of Madrid, Spain 통신공학과 졸업 (공학박사)

<주관심 분야> 분산 응용 구조, 개방형 분산처리 시스템 (ODP), 멀티미디어 통신 서비스, 사이버교육시스템, 객체지향 설계 기법.

고 석 주(Seok-Joo Koh) 정회원



1992.2. KAIST 경영과학과 공학사
1994.2. KAIST 경영과학과 공학석사
1998.8. KAIST 산업공학과 공학박사

1998.8.~현재 : ETRI 표준연구센터 선임연구원
<주관심 분야> 차세대 인터넷 기술, 초고속 망관리