오버레이 멀티캐스팅에서 트리의 스위칭을 고려한 빠른 멤버 가입 방안에 관한 연구

조 성 연[†] · 노 경 택[††] · 박 명 순[†††]

요 약

인터넷 멀티캐스팅 기술은 지난 10여년간의 기술 발전에도 불구하고 아직 본격적인 서비스 보급이 이루어지지 못하고 있다. 주된 이유는 멀티캐스트 라우팅에서의 트래픽 제어, 글로벌 인터넷에서의 멀티캐스트 주소 할당, 멀티캐스트 신뢰전송 기법 등의 문제가 아직 해결되지 않았기 때문이다. 최근 인터넷 방송, 실시간 증권정보 서비스 등의 멀티캐스트 응용 서비스에 대한 요구가 급격히 증가함에 따라, 새로운 인터넷 멀티캐스팅 기술로써 오버레이 멀티캐스팅이 개발되고 있다. 본 논문은 오버레이 멀티캐스팅 기술을 살펴보고, 새로운 멤버가 그룹에 가입하는데 걸리는 시간을 단축하는 방안을 제안한다. 기존의 방식은 잠재적인 부모 노드를 발견하기 위해서 한번에 트리의 한 레벨씩을 검색해 내려갔으며, 이로 인하여 긴 가입 지연 시간이 야기되었다. 또한, 트리의 매 레벨에서 자신과 가까운 노드를 잠재적인 부모 노드로 지적함으로써, 최적의 부모 노드를 선택하려고 노력하였지만 실제로 노드의 제한 차수로 인하여 자신과 가장 가까운 잠재적인 부모 노드를 선택하지 못하였으며, 이로 인해 트리의 효율성이 떨어졌다. 본 논문에서는 이러한 가입 지연 시간을 감소시키고, 생성된 트리의 효율성을 높이기 위해서, 트리의 두 레벨씩 검색하는 방안을 제안한다. 이 방식은 가입 요청 메시지를 자신의 자식 노드에게 전달하는 방식으로, 평상시에 트리 유지를 위한 추가적인 오버헤드 없이, 가입 요청이 왔을 때 검색 메시지의 수를 증가시켜서, 빠르게 가입을 완료한다. 또한, 더 많은 노드를 검색함으로써 좀 더 효율적인 트리를 생성하게 도와준다. 제안하는 방안의 성능을 평가하기 위해서, 그룹의 멤버 수와 제한 차수를 기준으로 검색 시간 및 검색한 노드의 수 그리고 트리 스위칭 횟수를 측정하였다. 시뮬레이션 결과에서 제안한 기법이 기존의 방식에 비해서 가입 지연 시간을 단축시켰으며, 좀 더 효율적인 트리를 생성하는 것을 볼 수 있었다.

Fast Join Mechanism that considers the switching of the tree in Overlay Multicast

Sungyean Cho[†] · Kyungtaeg Rho[††] · Myong-Soon Park[†††]

ABSTRACT

More than a decade after its initial proposal, deployment of IP Multicast has been limited due to the problem of traffic control in multicast routing, multicast address allocation in global internet, reliable multicast transport techniques etc. Lately, according to increase of multicast application service such as internet broadcast, real time security information service etc., overlay multicast is developed as a new internet multicast technology. In this paper, we describe an overlay multicast protocol and propose fast join mechanism that considers switching of the tree. To find a potential parent, an existing search algorithm descends the tree from the root by one level at a time, and it causes long joining latency. Also, it is try to select the nearest node as a potential parent. However, it can't select the nearest node by the degree limit of the node. As a result, the generated tree has low efficiency. To reduce long joining latency and improve the efficiency of the tree, we propose searching two levels of the tree at a time. This method forwards joining request message to own children node. So, at ordinary times, there is no overhead to keep the tree. But the joining request came, the increasing number of searching messages will reduce a long joining latency. Also searching more nodes will be helpful to construct more efficient trees. In order to evaluate the performance of our fast join mechanism, we measure the metrics such as the search latency and the number of searched node and the number of switching by the number of members and degree limit. The simulation results show that the performance of our mechanism is superior to that of the existing mechanism.

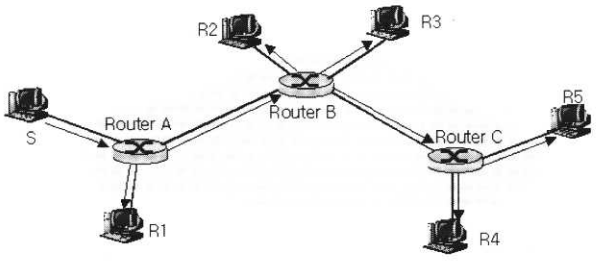
키워드 : 데이터 전송(Data Transfer), P2P, 분산된 프로토콜(Distributed Protocol), 종단 호스트 멀티캐스트(End System Multicast), 그룹 어플리케이션(Group Applications), IP 멀티캐스트(IP Multicast), 논리적인 오버레이 네트워크(Logical Overlay Network), 멀티캐스트 통신(Multicast Communication)

1. 서 론

많은 네트워크 응용 프로그램들과 P2P[1] 시스템들은 다수의 목적지에 데이터를 전송하는 능력을 요구한다. 예를 들어, 비디오 회의 어플리케이션에서는 하나의 호스트가 다른 호스트들에게 비디오 및 오디오 데이터를 보낸다. P2P 파일 공유 프로그램에서는 하나의 호스트가 다른 호스트들에게 검색 질의를 보낸다. 멀티 유저 게임에서는 참여자들의 위치 정보를 전송하며, 참가자는 다른 참가자들 사이에

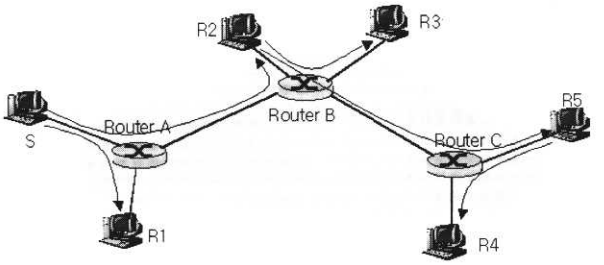
† 준 회 원 : 고려대학교 대학원 컴퓨터학과
†† 정 회 원 : 서울보건대학교 인터넷정보과 교수
††† 정 회 원 : 고려대학교 컴퓨터학과 교수
논문접수 : 2003년 6월 9일, 심사완료 : 2003년 8월 19일

서 동작을 취하게 된다. 이러한 종류의 응용 프로그램들이 점점 더 대중화됨에 따라서 다수의 목적지에 데이터를 효과적으로 전송하는 능력이 중요시되고 있다.



(그림 1) IP 멀티캐스팅

멀티캐스팅은 하나의 참가자가 그룹의 다른 참가자들에게 데이터를 효과적으로 전송하기 위한 네트워크 기술이다. IP 멀티캐스팅은 Ipv4에서 실행되는 멀티캐스팅이다[2]. IP 멀티캐스팅에서 패킷이 송신자에 의해서 한번 전송되면, 물리적인 네트워크상에 불필요한 중복 없이 모든 호스트에게 전달된다. (그림 1)은 IP 멀티캐스팅의 예를 보여준다. 노드 S가 데이터를 한번 전송하면, 이것은 Router A, Router B, Router C에서 복사되어서 R1, R2, R3, R4, R5에게 전송된다. IP 멀티캐스팅은 송신자가 데이터를 전송하고 수신자가 전송된 데이터를 받기 위해서 그룹에 가입하기 위한 특별한 주소를 사용한다[3]. 라우터들은 그룹의 상태를 유지하며, 필요한 라우팅을 실행한다. 그러나, 기술적인 문제와 관리적인 문제로 인하여, IP 멀티캐스팅은 인터넷에 널리 보급되지 못하였다[4].



(그림 2) 오버레이 멀티캐스팅

멀티캐스팅을 위한 다른 접근 방식으로 오버레이 멀티캐스팅이 있다. 이 방식은 데이터 전달을 위해서 오버레이 멀티캐스트 트리를 생성하고 유지한다. 이들의 주요 아이디어는 멀티캐스트 그룹의 참가자들이 가상 네트워크를 형성하기 위해서 서로 연결하는 것이다. 그 멤버들은 가상의 네트워크상에서 스스로 멀티캐스트 데이터를 라우팅하고, 이 데이터는 모든 멤버에게 전달된다. 멀티캐스팅의 기능이 라우터에서 종단 호스트로 이동한 것이다. 즉, 기능이 네트워크 계층에서 전송 또는 응용 계층으로 이동한 것이다[5]. 오버레이 멀티캐스팅은 일반적인 유니캐스트 전달 기능을 제외하고 라우터로부터 어떠한 지원도 요구하지 않는다. 오버레이 멀티캐스팅에는 Overcast[6], Narada[7, 8], ALMI[9], Yoid [10, 11], HMTP[5], Hypercast[12], NICE[13], CAN[14] 등이 있다.

(그림 2)는 오버레이 멀티캐스팅의 예를 보여준다. 노드 R2에서의 동작을 보면, 노드 R2가 노드 S로부터 데이터를 받으면, 이것은 노드 R3와 R5에게 데이터를 유니캐스트로 보낸다. 만약 노드 R2가 데이터를 전송한다면, 노드 S와 R3, R5에게 이 데이터를 유니캐스트로 보낸다.

오버레이 멀티캐스팅은 몇 가지 단점을 지니고 있다. 이는 IP 멀티캐스팅보다 네트워크 자원을 더 많이 소모한다는 것이다. (그림 1)과 (그림 2)를 보면, 오버레이 멀티캐스팅에서는 멀티캐스트 데이터가 Router B를 3번 지나가는 반면, IP 멀티캐스팅에서는 오직 한번만 지나간다. 또한, 오버레이 멀티캐스팅은 수 백 호스트 이상으로 확장되기 어렵다. 그러나, 이 방식은 종단 호스트에서 실행됨에 따라서 쉽고 빠르게 보급될 수 있다. 오버레이 멀티캐스팅은 원격 강의, 화상 회의, 멀티 사용자 게임, P2P 공유 프로그램과 같이 작은 멀티 캐스트 그룹을 요구하는 많은 응용들에 적합하다.

오버레이 멀티캐스팅의 확장성과 효율성은 이들의 분산된 트리의 질에 의해 영향을 받는다. 그러므로, 그룹의 멤버들간에 효과적인 트리를 구성하는 것은 중요하다. 이때, 하나의 호스트가 동시에 10개 이상의 연결을 지원하기 위한 대역폭을 지니지 못하기 때문에 트리를 구성할 때는 노드의 제한된 차수를 고려해야 한다[15].

노드의 제한된 차수를 고려한 기존 연구로써 HMTP[5]가 있다. HMTP는 트리 기반의 종단 호스트 멀티캐스팅 프로토콜이다. HMTP는 트리의 총 비용을 줄이면서 모든 호스트마다 제한된 차수를 최대한 유지하기 위해서 가까운 멤버들끼리 밀집한 구조를 형성하기 위해 노력한다. HMTP의 단점은 그룹의 규모가 클 때, 새로운 멤버가 그룹에 가입하는데 시간이 오래 걸리며, 또한 최적의 부모 노드를 선택하려고 노력하지만 실제로 자신과 가장 가까운 잠재적인 부모 노드를 선택하는 것이 어렵다는 것이다. 이러한 이유는 HMTP가 잠재적인 부모 노드를 선택하기 위해서 한번에 트리를 한 레벨씩 검색하기 때문이다[16].

본 논문의 구성은 다음과 같다. 2장에서 노드의 차수를 고려한 HMTP의 동작 방식 및 문제점을 보여줄 것이다. 3장에서는 HMTP 방식의 문제점을 해결하기 위한 알고리즘을 제안할 것이다. 본 논문에서 이러한 알고리즘을 FJM이라고 칭한다. FJM은 검색 지연시간을 줄이고 좀 더 정확한 부모 노드를 선택하기 위해서, 트리를 동시에 두 레벨씩 검색해 내려간다. 이를 위해서, 가입 요청 메시지를 자식 노드에게 전달하는 방식을 사용한다. 4장에서 시뮬레이션 환경 및 결과를 보여 줄 것이다. 시뮬레이션 결과에서 FJM이

기존의 방식에 비해서 검색속도가 빠르고 좀 더 정확한 부모 노드를 선택하는 것을 볼 수 있다. 마지막으로 5장에서는 결론과 향후 연구 방향에 대해 기술한다.

2. 이전 연구

오버레이 멀티캐스팅 노드는 네트워크상에서 호스트에 존재한다. 노드의 차수는 트리에서 한 호스트에 가상으로 연결된 다른 호스트들의 개수를 의미한다. 각 호스트는 적어도 하나 이상의 인터페이스를 가지며, 이는 네트워크상에서 자신과 연결된 다른 호스트들 사이의 물리적인 링크를 의미한다. 차수는 링크의 스트레스에 영향을 주며, 가상 링크의 개수는 인터페이스와 맵핑된다. 만약 링크 스트레스가 높으면, 그 인터페이스는 회선 쟁탈이 높으며, 네트워크 혼잡과 패킷 손실이 더 자주 발생할 것이다. 그러므로, 링크의 스트레스를 최소화하기 위해서, 각 오버레이 멀티캐스트 노드가 가질 수 있는 자수를 제한해야 한다.

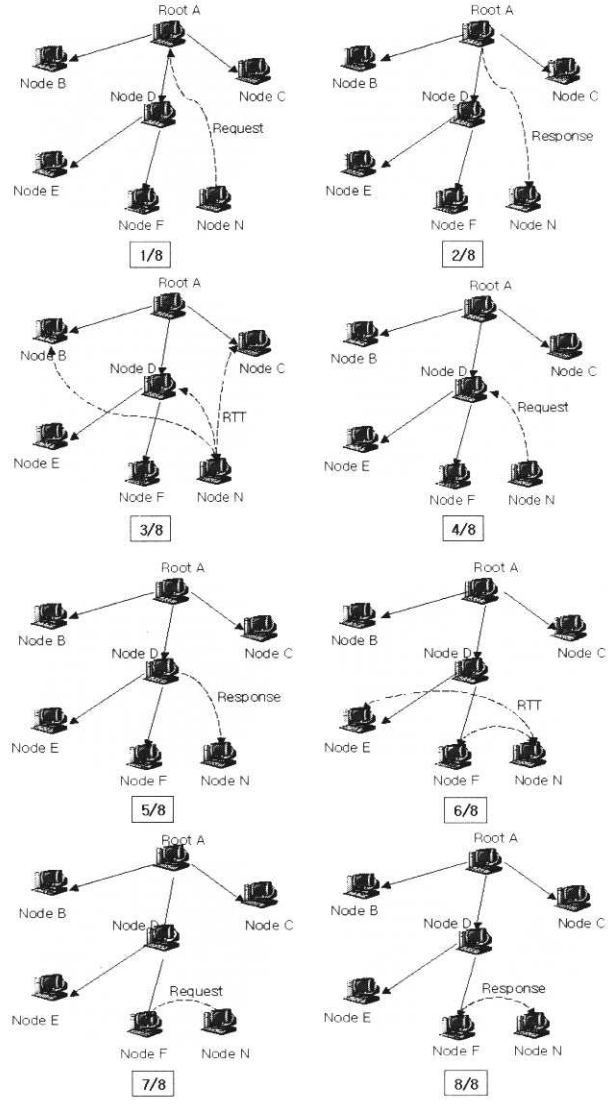
2.1 HMTP(Host Multicast Tree Protocol)[5]

HMTP는 트리의 총 비용을 줄이고, 각 호스트마다 최대 차수를 제한하기 위해서 가까운 멤버들 끼리 군집하려고 노력한다. 이를 위해, HMTP는 기존 맴버들의 부분 목록을 생성하는 법칙을 가지고 있다. 이 법칙은 새로운 멤버가 이 목록으로부터 자신과 가장 가까운 부모 노드를 선택하는 것을 도와 준다. 즉, HMTP에서 새로운 맴버는 트리의 작은 부분을 검색함으로써 좋은 부모 노드를 찾기 위해 노력한다. 이러한 검색은 리프 노드에 도착하거나, 혹은 모든 이웃 중에서 가까운 노드에 도착했을 때 멈추게 된다. 이러한 방식은 모든 멤버들 중에서 가장 가까운 노드를 부모 노드로써 선택하는 것을 보장하지는 않는다. 하지만, 모든 멤버들이 이와 같은 법칙에 따른다면, 거리 정보가 트리 구성에 영향을 끼칠 것이며, 이 정보는 매번 새로 가입하려는 노드를 도울 것이다.

2.2 HMTP의 가입 프로시저(Join Procedure)

HMTP에서 멤버들은 공유된 트리로부터 그들의 부모 노드를 찾는데 책임을 지니고 있다. 새로운 멤버 N은 다음의 과정으로 잠재적인 부모 노드를 검색한다.

① RP(Rendezvous Point)에 질의를 보내서 공유된 트리의 루트 노드를 알아낸다.
② 루트 노드부터 시작해서 트리의 각 레벨에서 노드 N은 자신과 가까운 노드 Z를 찾는다.
③ 만약 노드 Z의 자식 노드 수가 이들의 차수 제한보다 작다면, 노드 N은 Z의 자식 노드로써 그룹에 가입한다.
④ 만약 그렇지 않다면, 다음 레벨에서 위와 같은 절차를 반복하며, 노드 Z의 자식 노드들 중에서 잠재적인 부모 노드를 찾기 위해 노력한다.
⑤ 차수가 허용되는 노드를 발견하거나 혹은 리프 노드에 도착할 때까지 위의 과정을 반복한다.



(그림 3) HMTP의 멤버 가입 동작 방식

(그림 3)에서 HMTP의 멤버 가입 동작 방식의 예를 볼 수 있다. 새로운 노드 N이 루트 노드 A에게 request를 보낸다. 노드 A에 허용된 차수가 없다면, 노드 A의 자식 노드들 중에서 자신과 가까운 노드 D를 발견하기 위해서 request를 보낸다. 그러나, 노드 D에도 허용된 차수가 없다면, 노드 N은 다음 레벨에서 노드 D의 자식 노드들 중에서 가장 가까운 노드 F를 발견한다. 노드 F에 허용 가능한 차수가 있으면, 노드 N은 노드 F의 자식 노드로써 그룹에 가입하게 된다.

2.3 HMTP의 문제점

HMTP는 다음과 같은 문제점을 지니고 있다.

① HMTP는 트리의 매 레벨에서 한 번씩 검색을 수행하기 때문에 그룹의 멤버가 많을 때 가입 지연 시간이 길다. 이러한 긴 가입 지연 시간은 그룹의 멤버가 빠르게 변할 때 데이터 전송에 영향을 끼치게 된다. 예를 들어, HMTP를 이용하여 멀티미디어를 시청하는 사용자가 있다고 하자. 그 사용자의 호스트는 멀티캐스트 트리상에서 자신의 부모 노드로부터 데이터를 전송 받고 있을 것이다. 만약, 그의 부모 노드에 갑작스러운 정전이 발생하면, 그 멤버는 멀티미디어 데이터를 받지 못할 것이다. 멀티미디어 데이터를 계속 받기를 위해서 맴버는 기존 멀티캐스트 그룹에 다시 가입을 시도하게 된다. 만약 이때 가입 시간이 오래 걸린다면, 사용자는 멀티미디어를 보는데 오랜 끊어짐을 경험하게 될 것이다. 그러므로 이러한 방해를 줄이기 위해서, 그룹 가입 시간은 가능한 짧을수록 좋다.

② HMTP에서 새로운 노드는 최적의 부모 노드를 선택하려고 노력하지만, 자신과 가장 가까운 부모 노드를 선택하는 것은 어렵다. 이유는 트리의 각 레벨을 검색하여 자신과 가까운 노드중에서 차수가 허용되는 노드에게 가입을 수행하기 때문이다. 이는 새로운 노드가 그룹에 가입한 이후에 트리의 최적화를 위해서 트리 스위칭[15]을 발생시킨다. 트리 스위칭은 네트워크 트래픽 상태를 변화시키며, 스위칭 동안에 패킷 손실 및 중복을 야기 할 수 있기 때문에 최소화해야 하며, 이를 위해서는 새로운 노드는 좀 더 최적의 부모 노드에 선택하여 그룹에 가입을 해야 한다.

다음 장에서, 이러한 HMTP의 문제점을 개선하기 위한 실험적인 알고리즘을 소개하겠다.

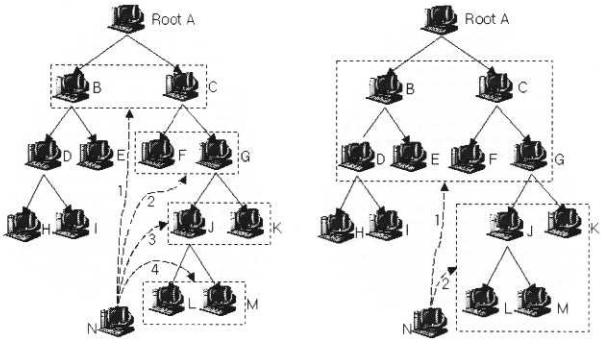
3. 제안 방안

본 장에서는 기존의 방식보다 빠르고 정확도가 높은 잠재적인 부모 노드를 찾기 위한 빠른 가입 기법을 제안한다. 이를 위해서, 다음과 같은 가정을 한다.

- 모든 종단 호스트는 성능이 매우 좋다고 가정한다. 그러므로, 네트워크상의 전송 지연만을 고려하며, 종단 호스트상의 처리 지연은 고려하지 않는다.
- 모든 트리의 멤버는 물리적인 네트워크에 대해 어떠한 정보도 가지고 있지 않다고 가정한다. 그러므로 멤버간의 거리 벡터로써 RTT(Round Trip Time)를 이용한다. 논문[17]에 설명된 기술에 기초하여, RTT는 TCP Ping 유틸리티[18]를 이용하여 측정 가능하다.

3.1 FJM(Fast Join Mechanism)

본 장에서는 단순한 방식과 랜덤 방식 및 한 레벨 방식을 간단히 설명한 후에 본 논문에서 제안하는 두 레벨 방식을 소개한다.



(그림 4) 한 레벨 검색 방식(왼쪽)과
두 레벨 검색 방식(오른쪽)의 비교

① 단순한 방식 : 잠재적인 부모 노드를 찾는 가장 단순한 방식은 누군가가 그룹의 모든 멤버에 대한 목록을 가지고 있으며, 이 목록을 이용하여 새로운 맴버는 기존의 모든 맴버들 중 가장 가까운 노드를 선택하는 방식이다. 이러한 단순한 방식은 잠재적인 부모 노드를 찾는 시간이 매우 빠르며, 가까운 멤버들끼리 군집할 것이다. 하지만, 이는 누군가가 모든 그룹 멤버의 목록을 유지해야만 한다. 만약 그룹의 멤버 수가 많아진다면, 이러한 목록을 유지하기 위한 어려움이 증가할 것이다.

② 랜덤 방식 : 만약 새로운 맴버가 기존 맴버의 랜덤한 부분에서 가장 가까운 부모 노드를 찾는다면, 이러한 목록은 필요하지 않을 것이다. 이 랜덤 방식은 모든 그룹 멤버에 대한 목록을 유지할 필요가 없으므로, 그룹 가입하는 시간은 매우 빨라 질것이다. 하지만, 이러한 방식으로 생성된 트리는 매우 랜덤한 구조를 가지게 될 것이며, 이후에 많은 트리의 스위칭을 야기할 것이다.

③ 한 레벨 방식 : 한 레벨 방식으로 HMTP[5]가 있다. (그림 4)의 한 레벨 방식은 기존 맴버의 부분 목록으로부터 가장 가까운 맴버를 선택함으로써 그룹에 가입한다. 그 결과 생성된 트리는 전체적으로 랜덤하지 않을 것이다. 또한, 모든 그룹 멤버의 목록을 유지 할 필요가 없다. 하지만, 이러한 방식은 그룹 맴버의 수가 많을 때 그룹에 가입하는 시간이 길어지는 단점을 지니고 있다.

④ 이러한 기존 방식들의 단점을 극복하기 위해서 본 논문에서는 (그림 4)의 두 레벨 검색 방식인 빠른 가입 기법(FJM)을 제안한다. FJM은 잠재적인 부모 노드를 찾기 위해서 트리의 두 레벨씩 검색해 내려간다. 주요한 아이디어는 평소에 그룹을 유지하기 위한 메시지 교환 회수는 그대로 유지하면서, 새로운 맴버가 그룹

에 가입하려고 할 때만 메시지의 수를 증가시켜서 빠르게 가입하는 것이다. 또한, 메시지의 수가 증가한 만큼 더 많은 노드들을 검색함으로써 좀 더 가까운 부모 노드를 선택할 기회를 부여하고, 그 결과 새로운 노드가 그룹에 가입한 후에 트리의 스위칭 횟수를 줄인다. 이를 위해서, 기존의 멤버가 Request 메시지를 그들의 자식 노드에게 전달하는 방식을 사용한다.

3.2 FJM의 메시지 타입

FJM에서 사용하는 메시지에 대한 설명 및 동작 방식을 <표 1>에서 볼 수 있다. 새로운 멤버는 oCLR 메시지를 전송하여 가입이 가능한지 여부를 알아본다. oCLR 메시지를 받은 멤버에게 허용된 차수가 없으면, fCLR 메시지로 변환하여 자신의 자식 노드에게 메시지를 전달한다. 응답으로써 nLRS 메시지를 받은 새로운 멤버는 받은 노드의 목록들 중에서 자신과 가장 가까운 노드를 검색한다. 만약 새로운 멤버가 yLRS 메시지를 받는다면 이것은 잠재적인 부모 노드를 발견했다는 것을 의미하고, 가입을 시도한다.

3.3 FJM의 가입 프로시저(Join Procedure)

〈표 1〉 요청 메시지 및 응답 메시지

이 름	의 미	메시지 수신 후 동작
oCLR	최초의 요 청	유효 차수 있는 경우 : yLRS를 새로운 멤버에게 전송. 유효 차수 없는 경우 : fCLR을 자식 노드들에게 전송.
fCLR	전달된 요 청	유효 차수 있는 경우 : yLRS를 새 멤버에게 전송. 유효 차수 없는 경우 : nLRS를 새로운 멤버에게 전송.
nLRS	가 입 불 능	목록의 노드의 RTT를 측정하여 가장 가까운 노드에게 CLR0을 전송.
yLRS	가 입 가 능	그 노드에게 가입함.

FJM에서 멤버들은 공유된 트리로부터 그들의 부모 노드를 찾는데 책임을 지니고 있다. 새로운 멤버 N은 다음의 과정으로 잠재적인 부모 노드를 검색한다.

① RP에게 질의를 보내서 공유된 트리의 루트 노드를 알아낸다.

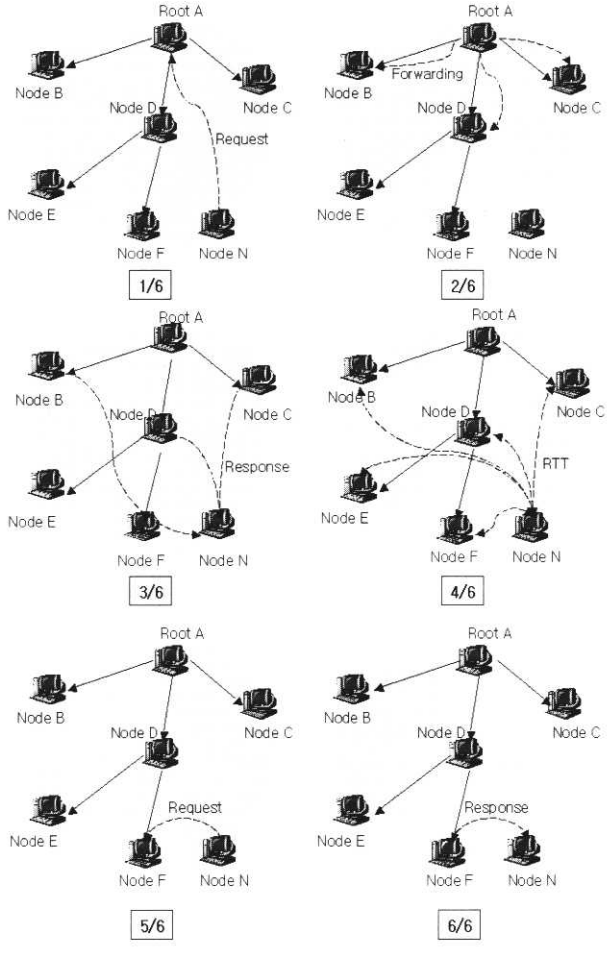
② 노드 N은 루트 노드를 잠재적인 부모 노드로 지정하고, oCLR 메시지를 루트 노드에게 전송한다.

③ oCLR 메시지를 받은 노드에게 허용된 차수가 없다면, fCLR로 메시지를 변환해서 자신의 자식 노드들에게 전달한다.

④ fCLR 메시지를 받은 노드는 자신과 자식 노드들의 주소를 nLRS 메시지에 넣어서 노드 N에게 보낸다.

⑤ 노드 N이 nLRS 메시지를 받으면, 받은 목록의 노드에게 probe를 보내서 자신과 가장 가까운 노드를 알아내고, 그 노드에게 oCLR 메시지를 전송한다.

⑥ 만약, oCLR 메시지를 받은 노드에게 허용된 차수가 있다면, yLRS 메시지를 노드 N에게 전송하고, 노드 N은 그 노드의 자식 노드로써 그룹에 가입하게 된다.

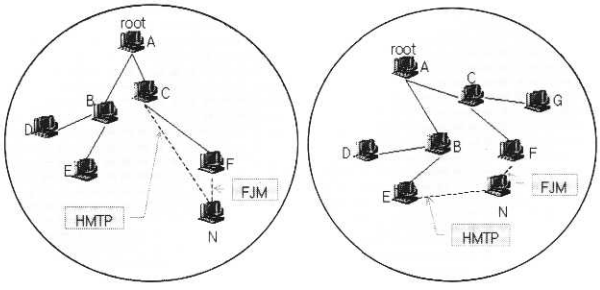


(그림 5) FJM의 멤버 가입 동작 방식

(그림 5)에서 FJM의 멤버 가입 동작 방식의 예를 볼 수 있다. 새로운 노드 N은 RP에게 질의를 보내서 루트 노드 A의 주소를 알아낸다. 노드 N은 루트 노드 A에게 oCLR 메시지를 전송하고, oCLR 메시지를 받은 노드 A는 자신에게 허용되는 차수가 없을 경우, oCLR 메시지를 fCLR 메시지로 변환해서 자신의 자식 노드인 B와 C와 D에게 전달한다. fCLR 메시지를 받은 노드 B와 C와 D는 자신과 자식 노드 주소를 nLRS 메시지에 넣어서 노드 N에게 보낸다. nLRS를 받은 노드 N은 노드 B, C, D, E, F에게 probe 메시지를 보내서 RTT를 알아낸다. 이들 중에서 가장 짧은 RTT를 가지는 노드 F를 발견하고, 노드 F에게 oCLR 메시지를 전송한다. oCLR 메시지를 받은 노드 F가 허용된 차수가 있으면, 노드 N에게 yLRS 메시지를 전송한다. yLRS 메시지를 받은 노드 N은 노드 F의 자식 노드로써 그룹에 가입하게 된다. 위의 자세한 알고리즘을 (그림 6)과 (그림 7)에서 볼 수 있다.

3.4 FJM의 기대 효과

FJM은 다음과 같은 기대 효과를 지니고 있다.



(그림 6) HMTP와 FJM의 가입 결과 비교

FJM은 그룹의 모든 멤버에 대한 목록을 유지할 필요가 없다. HMTP처럼, FJM에서 모든 노드는 자신의 부모 노드와 자식들 노드 그리고 루트 노드만 알고 있으면 된다. 그러므로, FJM은 동적으로 변하는 그룹의 멤버 관리가 용이하며, 평상시에는 그룹 유지를 위해서 HMTP 방식과 동일한 수의 메시지를 교환한다.

FJM은 랜덤한 트리 구조를 생성하지 않는다. FJM은 루트부터 시작하여 트리의 두 레벨씩 검색을 수행하면서, 자신과 가장 가까운 노드를 찾으려고 노력할 것이다. 이것은 한 레벨 검색을 수행하는 HMTP보다 좀 더 좋은 부모 노드를 선택할 수 있는 기회를 줄 것이다. (그림 6)을 보면 FJM이 HMTP보다 좀 더 자신과 가까운 노드를 선택 할 수 있음을 볼 수 있다. 이것은 멤버 가입 후에 트리의 스위칭 횟수를 줄여줄 것이다.

FJM은 새로운 맴버가 그룹에 가입할 때, 메시지의 수를 증가시킨다. 즉, FJM은 순간적으로 메시지의 수를 증가시켜서 검색 속도를 향상 시킨다. 빨라진 검색 속도는 데이터 전송의 끊어짐을 줄여줄 것이다.

FJM은 HMTP보다 검색 단계가 줄었다. 예를 들어, (그림 3)에서 HMTP는 트리의 두 레벨을 검색하기 위해서 총 8 회가 필요하다. 반면에, (그림 5)에서 FJM는 총 6회가 필요하다.

다음 장에서는 FJM 방식이 HMTP 방식보다 새로운 멤버가 잠재적인 부모 노드를 빠르게 선택하며, 안정된 트리를 생성하는 것을 볼 수 있다.

4. 시뮬레이션

이 장에서는 FJM을 HMTP와 비교하여 성능을 평가하였다. 본 논문에서 제안하는 두 레벨 검색 방식을 FJM으로 표기하였으며, 한 레벨 검색 방식을 HMTP로 표기하였다. 시뮬레이션을 통해서 검색 속도, 검색에 사용되는 메시지의 수 및 검색한 노드의 수, 트리의 효율성 및 트리 스위칭의 횟수를 비교하였다. 실험 결과에서 FJM이 HMTP보다 빠르게 부모 노드를 검색하며 좀 더 안정된 트리를 생성하는 것을 볼 수 있었다.

4.1 시뮬레이션 환경

Linux kernel 2.4.18 환경에서 인터넷 토폴로지 생성기인 Inet 3.0[19]을 이용하여 네트워크상에 호스트를 6000개 만들었다. Inet으로 발생시킨 가상의 인터넷 위에 실험을 위하여 임의의 노드들을 선택하여 다양한 멤버 수를 가진 트리를 생성하였다. 데이터 전송을 위한 트리를 유지하기 위해서, 모든 멤버는 자신과 연결된 부모 노드 및 자식 노드 그리고 루트 노드의 주소를 유지하며, ping 메시지를 이용하여 주기적으로 부모 노드 및 자식 노드의 존재여부를 확인한다. 만약 부모 노드의 존재가 확인되지 않으면, 루트노드에게 메시지를 보내서 그룹에 다시 가입을 한다. 이러한 그룹의 멤버 가입 방식은 (그림 7)과 (그림 8)의 알고리즘을 이용하여 잠재적인 부모 노드를 검색하였다.

```
When a new member tries to join {
  Send Query ( RP ) ;   // To know Root's IP address.
  If RP responds {
    oCLR = New_Member_IPAddress ;
       // Put new member IP address to oCLR.
    Send oCLR ( Root_IPAddress ) ;
       // Send oCLR to root.
  }
}

When a new member receives LRS {
  If ( yLRS ) {
    Potential_Parent_IPAddress = yLRS ;
       // Get potential parent's IP address in yLRS.
    Join ( Potential_Parent_IPAddress ) ;
       // Join to node that send yLRS.
  }
 If ( nLRS ) {
    Send Ping ( nLRS_All_Lists ) ;
       // Send ping to all lists.
    The_Nearest_Node_IPAddress
     = SmallRTT ( nLRS_All_Lists ) ;
       // Find the nearest node among the lists.
    oCLR = New_Member_IPAddress ;
       // Put new member IP address to oCLR.
    Send oCLR ( The_Nearest_Node_IPAddress ) ;
       // Send oCLR to the nearest node.
    }
}
```

(그림 7) 새로운 맴버측 알고리즘

```
When an existing member receives CLR{
  If ( oCLR ) {
    If ( Available_Degree == 0 ) {
         // No available degree.
       fCLR = oCLR ;
         // Put a new member's IP address to fCLR.
      Send fCLR ( All_Children_IPAddress ) ;
         // Send fCLR to its all children.
       }
     else {
         // Available degree.
```

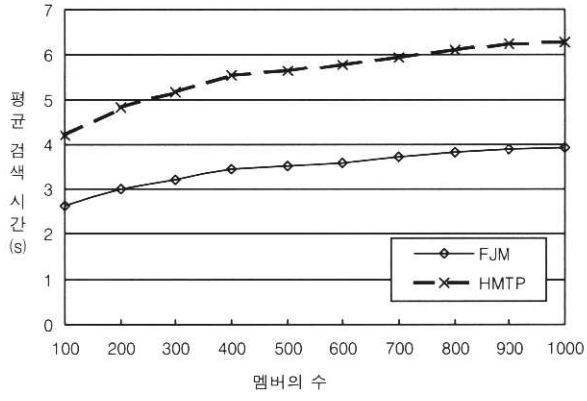
```
        New_Member_IPAddress = oCLR ;
          // Get new member's IP address in oCLR.
        yLRS = Own_IPAddress ;
          // Put own IP address to yLRS.
        Send yLRS ( New_Member_IPAddress ) ;
          // Send yLRS to a new member.
      }
    }
If ( fCLR ) {
        New_Member_IPAddress = fCLR ;
        // Get a new member's IP address in fCLR.
        nLRS = Own_IPAddress
                + All_Children_IPAddress ;
          // Put its own IP address and
          // all children IP address to nLRS.
        Send nLRS ( New_Member_IPAddress ) ;
          // Send nLRS to a new member.
    }
}
```

(그림 8) 기존 맴버측 알고리즘

시뮬레이션은 두 가지 방식으로 실행되었다. 첫 번째, 노드들의 차수를 2로 제한한 후에 그룹의 멤버 수를 100에서부터 1000까지 증가시키면서 실험하였다. 두 번째, 그룹의 멤버 수를 500으로 고정시킨 후에 노드들의 차수를 1에서 10까지 증가시키면서 실험하였다. 각 시뮬레이션에서, 그룹의 멤버와 루트 노드를 임의로 선택하였으며, 매 시뮬레이션마다 30번씩 멤버 가입을 시도하였다.

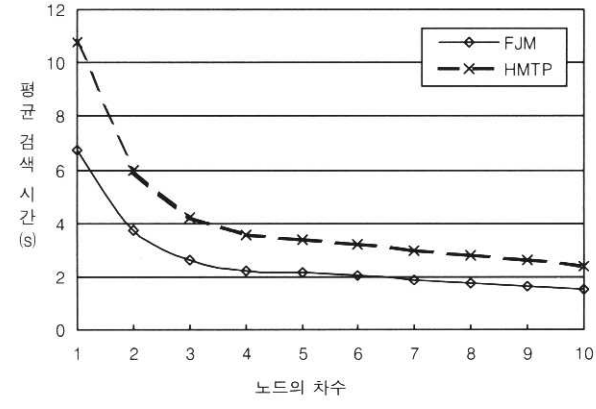
4.2 시뮬레이션 결과 및 분석

FJM과 HMTP를 그룹의 멤버 수와 노드의 차수에 따라서 비교하였다. 첫 번째 실험 결과는 평균 검색 시간을 보여준다. 두 번째 실험 결과는 검색을 위한 메시지의 수와 검색한 노드의 수를 보여준다. 세 번째 실험 결과는 멤버 가입이 이루어진 후에 트리의 안정성을 보여준다. 네 번째 실험 결과는 트리의 스위칭 횟수를 보여준다.



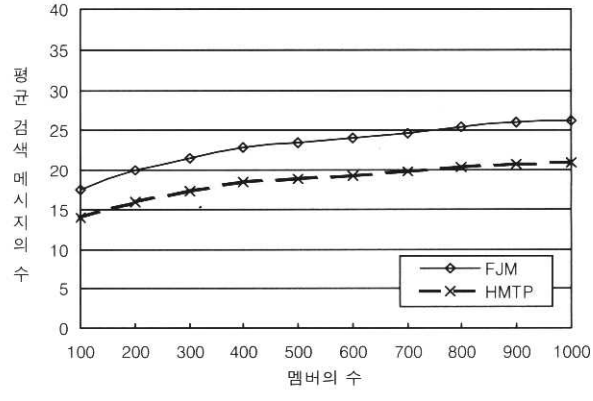
(그림 9) 그룹의 멤버 수에 따른 평균 검색 시간

(그림 9)는 그룹의 멤버 수가 증가 할 때 평균 검색 시간을 나타낸다. 그룹의 멤버 수가 증가하면, 잠재적인 부모 노드를 찾는데 걸리는 시간이 증가한다. 멤버수가 증가할 때 트리의 높이가 높아지게 되는데. 이때 트리의 한 레벨에서 포함하는 멤버수가 증가함으로 검색 시간은 지수적으로 증가하지는 않는다. 그림에서 FJM이 HMTP보다 검색 속도가 빠른 것을 볼 수 있다. 이것은 FJM이 트리의 두 레벨을 검색하기 위해서 Request 메시지 1회, Forwarding 메시지 1회, Response 메시지 1회, Ping 1회를 수행하는 반면, HMTP는 Request 2회, Response 2회, Ping 2회가 필요하기 때문이다.



(그림 10) 노드의 차수에 따른 평균 검색 시간

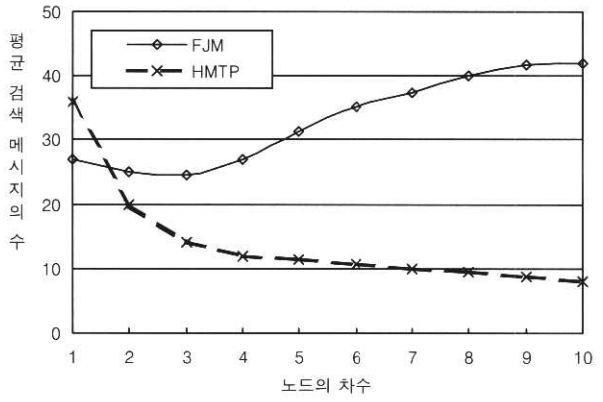
(그림 10)은 노드의 차수가 증가할 때 평균 검색 시간을 보여준다. 그림에서 노드의 차수가 증가 할수록 검색 시간이 빨라지는 것을 볼 수 있다. 이는 멤버의 수가 고정된 상태에서 노드의 차수 높아짐에 따라서 전체 트리의 높이가 낮아져서 검색 단계가 줄어들었기 때문이다. 그림에서 FJM이 HMTP보다 평균 1.6배 빠른 것을 볼 수 있다.



(그림 11) 그룹의 멤버 수에 따른 평균 검색 메시지의 수

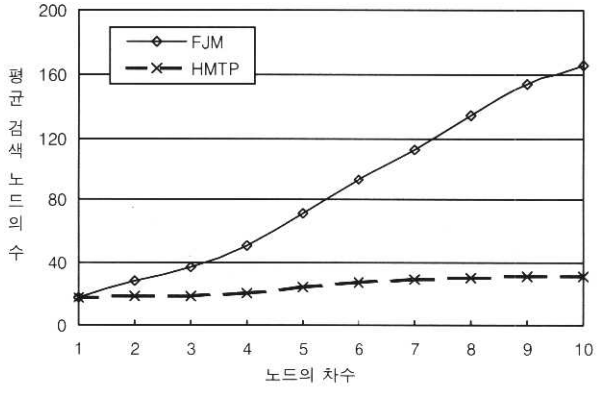
(그림 11)은 그룹의 멤버 수가 증가할 때 평균 검색 메시지의 수를 보여준다. 그룹의 멤버 수가 증가함에 따라서 검색을 위해서 사용하는 메시지의 수는 조금씩 증가하는 것을 볼 수 있다. 이러한 증가는 멤버 수가 증가함에 따라서 트리의 높이가 조금씩 높아지기 때문이다. 그림에서 FJM이

HMTP보다 검색을 위한 메시지가 1.25배 정도 더 많은 것을 볼 수 있다.



(그림 12) 노드의 차수에 따른 평균 검색 메시지의 수

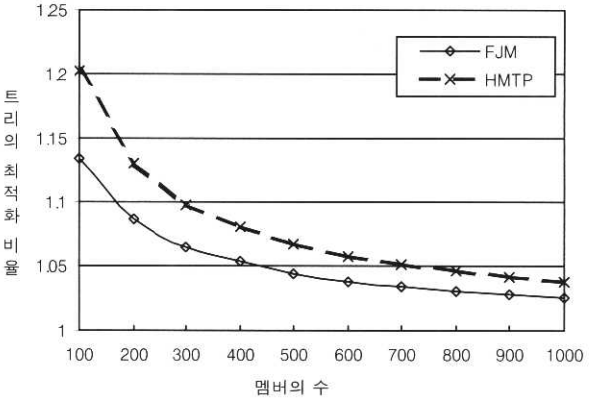
(그림 12)는 노드의 차수에 따른 검색 메시지의 수를 보여 준다. 노드의 차수가 증가함에 따라서 HMTP는 검색 메시지의 수가 줄어든 반면에, FJM은 매우 많이 증가한 것을 볼 수 있다. HMTP는 노드의 차수에 영향을 받지 않으며, 단지 멤버의 수가 고정된 상태에서 노드의 차수가 증가함에 따라서, 트리의 높이가 낮아진 것에 영향을 받은 것이다. 반면 FJM은 노드의 차수에 영향을 받기 때문에, 노드의 차수가 증가함에 따라서 검색 메시지의 수가 매우 증가한 것을 볼 수 있다. 이것은 메시지의 수가 증가한 만큼 검색한 노드의 수가 증가했음을 (그림 13)에서 볼 수 있다.



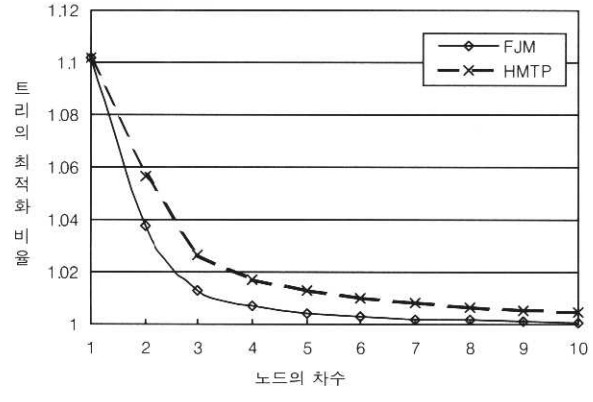
(그림 13) 노드의 차수에 따른 검색한 노드의 수

(그림 13)은 노드의 차수가 증가 할 때, 평균적으로 검색한 노드의 수를 보여 주고 있다. HMTP는 노드의 차수가 증가함에 따라서 검색한 노드의 수가 조금 증가한 반면에, FJM은 노드의 차수가 증가함에 따라서 검색한 노드의 수가 매우 큰 폭으로 증가한 것을 볼 수 있다. 이것은 FJM이 짧은 시간에 더 많은 노드를 검색함에 따라서 좀 더 좋은 부모 노드를 선택할 수 있는 기회를 제공하는 것으로 볼 수 있다.

(그림 14)는 그룹의 멤버 수에 따른 트리의 최적화 비율을 보여주고 있다. 여기서 트리의 최적화 비율이란 HMTP와 FJM으로 만든 트리를 최소 경로 트리 알고리즘으로 만든 트리와 비교 한 결과를 의미한다. 즉, 동일한 멤버를 이용하여 생성된 트리가 얼마나 최소 경로 트리에 가까운지를 알아본 것이다. Y축의 값이 1에 가까울수록 최소 경로 트리에 가까운 것을 의미한다. 그림에서 멤버의 수가 증가할수록 1에 가까워지는 이유는 100에서 1000까지 멤버 수를 증가시키면서 매회 30개의 노드를 가입시켰기 때문이다. 예를 들어, 100개의 노드에 새로운 30개의 노드를 가입시킬 경우 전체 트리의 23%에 영향을 주는 반면, 1000개의 노드에 새로운 30개의 노드를 가입 시킨 것은 전체 트리의 3%에 영향을 미치기 때문이다. 그림에서 FJM이 HMTP보다 1에 가까운 것을 볼 수 있다. 이는 FJM이 HMTP보다 좀 더 많은 노드를 검색함으로써 최적의 부모 노드를 선택할 기회를 좀 더 가지고 있기 때문이다. 예를 들어, 노드의 제한 차수가 2일 때, 트리의 두 레벨을 검색할 경우 FJM은 6개의 노드를 검색하는 반면에 HMTP는 4개의 노드를 검색한다.



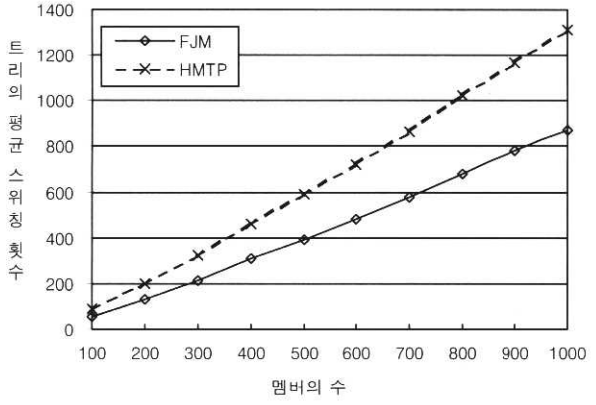
(그림 14) 그룹의 멤버 수에 따른 트리의 최적화 비율



(그림 15) 그룹의 멤버 수에 따른 트리의 최적화 비율

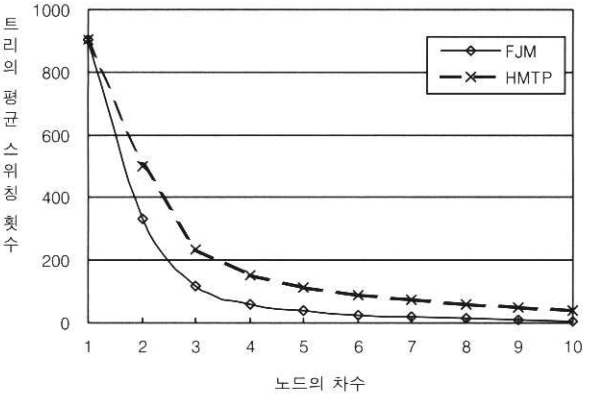
(그림 15)는 노드의 차수가 증가할 때 트리의 최적화 비율을 보여 준다. 노드의 차수가 1일 때, 두 방식의 최적화

비율이 같은 것은 동일한 수의 노드를 검색하여 가입한 결과의 트리가 동일하기 때문이다. 노드의 차수가 높아질수록 최적화 비율이 1에 가까워지는 것을 볼 수 있다. 이것은 노드의 차수가 높아짐에 따라서 많은 노드를 검색하기 때문이다. 그림에서 FJM은 HMTP보다 1에 가까운 것을 볼 수 있다. 이는 FJM의 경우 차수가 높아질수록 HMTP보다 훨씬 더 많은 노드를 검색하기 때문이다. 예를 들어 노드의 제한 차수가 5일 때, 트리의 2 레벨을 검색할 경우 FJM은 30개의 노드를 검색하는 반면에 HMTP는 5개의 노드를 검색하게 된다.



(그림 16) 그룹의 멤버 수에 따른 평균 스위칭 횟수

(그림 16)은 그룹의 멤버 수에 따른 트리의 평균 스위칭 횟수를 보여준다. 트리의 스위칭 횟수는 논문[15]에서의 switch-any방식을 이용하여 계산하였다. 멤버의 수가 증가함에 따라서 스위칭 횟수가 증가한 것을 볼 수 있다. 이것은 멤버가 정확한 위치에 가입되지 못하였기 때문에, 트리의 최적화를 위해서 많은 노드에게 영향을 끼치기 때문이다. 그림에서 FJM이 HMTP보다 트리의 스위칭 횟수가 낮은 것을 볼 수 있다.



(그림 17) 그룹의 멤버 수에 따른 평균 스위칭 횟수

(그림 17)은 노드의 차수가 증가할 때, 트리의 스위칭 횟수를 보여준다. 노드의 차수가 증가함에 따라서 트리의 스위칭 횟수가 매우 감소한 것을 볼 수 있다. 이는 노드의 차수에 따라서, 검색하는 노드의 수가 증가함으로 좀 더 정확한 부모 노드를 선택할 기회를 제공하였기 때문이다. 그림에서 FJM는 차수가 증가함에 따라서 HMTP보다 2배에서 5배까지 트리의 스위칭 횟수가 적은 것을 볼 수 있다. 이는 FJM이 차수가 증가함에 따라서 HMTP보다 더 많은 노드를 검색하여 더 안정된 트리를 생성하기 때문이다.

시뮬레이션 결과에서 FJM은 잠재적인 부모 노드를 검색하는데 걸리는 시간을 매우 단축한 것을 볼 수 있었다. 이것은 그룹의 멤버가 다이나믹하게 변화하는 환경에서 빠르게 대처할 수 있게 도와준다. 또한, FJM은 노드의 제한 차수가 증가함에 따라서 검색을 위한 메시지의 수가 증가한 것을 볼 수 있다. 이것은 짧은 시간에 더 많은 노드를 검색함으로써 좀 더 좋은 부모 노드를 선택할 수 있는 기회를 제공한다. 결과적으로 생성된 트리의 안정성이 높아진 것을 볼 수 있었다. 즉, 트리의 질을 향상시킴으로써, 트리 스위칭으로 인한 패킷 손실 및 중복 등을 줄일 수 있다. 또한, FJM은 HMTP처럼 각 노드는 자신의 부모 노드와 자식 노드들만을 기억하면 되므로, 평상시에 트리를 유지하기 위한 추가적인 오버헤드가 없다는 장점을 지녔다.

5. 결 론

본 논문은 오버레이 멀티캐스팅에서 새로운 멤버가 그룹에 가입하는데 걸리는 시간을 단축하고 안정된 트리를 생성하기 위한 빠른 가입 기법을 제안하였다. 이 FJM은 가입 요청 메시지를 자식 노드에게 전달하는 방식을 사용하여 트리의 두 레벨을 동시에 검색하였다. 시뮬레이션 결과에서 FJM이 가입 지연 시간을 단축 시킨 것을 볼 수 있었다. 또한, 노드의 차수에 따라서 메시지 수를 증가시킴으로써 더 많은 노드를 검색하여 좀 더 좋은 부모 노드를 선택할 수 있는 기회를 제공하였고, 결과적으로 생성된 트리의 질을 향상시켰다.

향후의 연구 과제는 노드의 제한 차수에 따른 가입의 문제점을 해결하는 것이다. 예를 들어, 제안된 방안은 새로운 멤버가 자신과 가장 가까운 노드를 발견하더라도 허용 가능한 차수가 없다면, 다른 노드에 연결되고 이후에 트리의 스위칭을 야기하게 된다. 이를 위해서는 제한 차수를 고려하여 최적의 위치에 가입을 시키면서, 이로 인한 패킷의 손실이 발생하지 않도록 하기 위한 방안을 연구해야 한다.

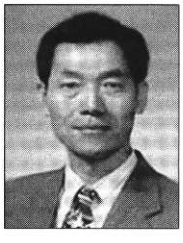
참 고 문 헌

[1] Schoder, D., Fischbach, K., "Peer-to-peer (P2P) computing," Proceedings of the 36th Annual Hawaii International Conference on, 6-9, p.217, Jan., 2003.

[2] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," ACM Transactions on Computer Systems, 8(2), 85110, May, 1990.
[3] D. Thaler, etc., "The Internet Multicast Address Allocation Architecture," IETF RFC 2908, Sep., 2000.
[4] C. Diot et al., "Deployment Issues for the IP Multicast Service and Architecture," IEEE Networks Magazine's Special Issue on Multicast, Jan., 2000.
[5] B. Zhang, S. Jamin and L. Zhang, "Host multicast : A framework for delivering multicast to end users," In Proceedings of IEEE Infocom, June, 2002.
[6] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek and James W. O'Toole, Jr., "Overcast : Reliable multicasting with an overlay network," Proceedings of the Fourth Symposium on Operating Systems Design and Implementation, pp.197-212, October, 2000.
[7] Yang-hua Chu, Sanjay G. Rao and Hui Zhang, "A case for EndSystem multicast," Proc. of ACM IGMETRICS '00, p. 112, June, 2000.
[8] Y.-H. Chu, S. G. Rao, S. Seshan and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture," In Proceedings of ACM SIGCOMM, August, 2001.
[9] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma and Marcel Waldvogel, "ALMI : an application level multicast infrastructure," 3rd USENIX Symposium on Internet Technologies and Systems, March, 2001.
[10] Paul Francis, "Yoid : extending the internet multicast architecture," Technical report, NTT, April, 2000.
[11] P. Francis, "Yoid : Extending the Multicast Internet Architecture," White paper http://www.aciri.org/yoid/, 1999
[12] Tyler K. Beam, J. Liebeherr, "A Protocol for Maintaining Multicast Group Members in a Logical Hypercube Topology," Proc. First International Workshop on Networked Group Communication (NGC '99), pp.72-89, 1999.
[13] S. Banerjee, B. Bhattacharjee and C. Kommareddy, "Scalable application layer multicast," In Proceedings of ACM Sigcomm, August, 2002.
[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A scalable content-addressable network," In Proceedings of ACM Sigcomm, August, 2001.
[15] D. Helder and S. Jamin, "End-host multicast communication using switchtrees protocols," Global and Peer-to-Peer Computing on Large Scale Distributed Systems, 2002.
[16] Sungyean Cho and Myong-Soon Park, "FJM : Fast Join Mechanism For Overlay Multicast," 2003 IEEE Conference On Control Applications (CCA 2003), Istanbul, TURKEY, June, 2003.
[17] Amgad Zeitoun, TCP Ping homepage, http://www.eecs.umich.edu/azeitoun/tools.html, November, 2001.
[18] Martin Horneffer, "Assessing Internet Performance Metrics Using Large-Scale TCP-SYN Based Measurement," Passive and Active Measurement Workshop, 2000.
[19] C. Jin, Q. Chen and S. Jamin, "Inet : Internet Topology Generator," Technical Report CSE-TR-433-00, EECS Department, University of Michigan, 2000.

조 성 연
e-mail : pastco@ilab.korea.ac.kr
2001년 서울여자대학교 컴퓨터학과(이학사)
2001년~현재 고려대학교 컴퓨터학과 (이학석사)
관심분야 : 멀티캐스트, 유비쿼터스, 이동 컴퓨팅, 스토리지

노 경 택
e-mail : rho@shjc.ac.kr
1986년 중앙대학교 전자계산학과(이학사)
1989년 New Jersey 공과대학 컴퓨터정보학과(이학석사)
1997년~현재 고려대학교 컴퓨터학과 박사수료
1993년~현재 서울보건대학교 인터넷정보과 교수
관심분야 : 멀티캐스팅, 모바일컴퓨팅, 웹서버시스템



박 명 순
e-mail : myongsp@cslab1.korea.ac.kr
1975년 서울대학교 공과대학 전자공학과 (공학사)
1982년 Utah 대학교 공과대학 전기공학과 (공학석사)
1985년 Iowa 대학교 공과대학 전기 및 컴퓨터공학과(공학박사)
1988년~현재 고려대학교 컴퓨터학과 교수
관심분야 : 모바일 컴퓨팅, 웹서버시스템, 스토리지