자원 예약 방식을 사용한 오버레이 멀티캐스트 트리의 구성과 복구 방안

준회원 허 권*, 정회원 손 승 철*, 김 경 훈*, 송 호 영* 종신회원 남 지 승*

A Proactive Approach to Reconstructing Overlay Multicast Trees using Resource Reservation

Kwon Heo* ***Associate Member***, Suung-Chul Son*, Kyung-Hoon Kim* ***Regular Members***, Ho-Young Song* ***Regular Member***, Ji-Seung Nam* ***Lifelong Member***

요 약

오버레이 멀티캐스트는 하드웨어적인 인프라의 구축 없이도 시스템의 자원과 네트워크 대역폭을 효율적으로 사용할 수 있는 기법이지만 IP 멀티캐스트와는 달리 중간 노드의 이탈이 발생하게 되어 멀티캐스트 트리의 복구가 필요하다. 본 논문에서는 이러한 결점을 보완하기 위해 각각의 노드가 가지는 Out Degree 자원을 사전에 예약하여 트리의 복구가 필요로 할 때 자원을 예약한 노드에게 바로 서비스를 요청하는 모델을 제안한다. 제안된 모델은 백업 노드의 도움으로 빠른 복구가 가능하고 복구된 노드들에게는 새로운 경로에서 발생한 지연시간의 영향을 최소화 했다. 시뮬레이션 결과를 통해 제안된 모델이 기존의 기법들 보다 적은 복구 시간이 소요되고 부모 노드의 이탈로 인해 많은 수의 노드가 영향을 받는 상황일수록 더욱 효과적인 방안임을 보여주고 있다.

Key Words : Overlay Multicast, Application Layer Multicast, Proactive Route Maintenance, Backup, Resource Reservation

ABSTRACT

Overlay Multicast is an effective method for efficient utilization of system resources and network bandwidth without using hardware customization. Unlike in IP multicast, multicast tree reconstruction is required when non-leaf node leaves or fails. In this paper, we propose a proactive approach to solve this defect by using a resource reservation of the out degrees. This allows children of non-leaf node to connect to its new parent node immediately when its parent node leaves or fails. In our proposal, a proactive route maintenance gives a fast recovery time and reduces a delay effect in the new route. The simulation results show that our proposal takes shorter period of time than the other algorithms to reconstruct a similar tree and that it is a more effective way to deal with a lot of nodes that have lost their parent nodes.

Ⅰ. 서 론

오버레이 멀티캐스트(Overlay Multicast)란 멀티캐스트의 기능을 종단 호스트(end-host)에 구현한 것이다. 기존의 IP 멀티캐스트가 실제로 사용되기 위해서는 각각의 라우터에서 IP 멀티캐스트 패킷을 처리 할 수 있도록 구현되어야 한다는 점과는 다르게 사용되고 있는 네트워크 구조의 변경 없이 단지 종단 호스트가 자신이 받은 패킷을 자신의 다음 호스트에게 전달해주는 방법을 사용한 어플리케이션

* 전남대학교 공과대학 컴퓨터공학과 멀티미디어 데이터통신 연구실 (rusifery@gmail.com)
논문번호 : KICS2006-11-479, 접수일자 : 2006년 11월 7일, 최종논문접수일자 : 2006년 12월 4일

소프트웨어의 설치만으로 멀티캐스트를 사용할 수 있는 기법이다.

그러나 오버레이 멀티캐스트는 트리를 구성하고 있는 중간 노드가 멀티캐스트 그룹을 이탈을 하였을 경우 트리를 재구성 해야만 된다는 문제점을 가지고 있다. IP 멀티캐스트는 트리를 구성하고 있는 중간 노드들이 라우터이기 때문에 이와 같은 문제를 고려할 필요가 없지만 오버레이 멀티캐스트는 중간 노드를 포함한 모든 노드들이 언제든지 멀티캐스트 그룹을 자유롭게 이탈할 수 있는 종단 호스트로 구성이 된다는 점에서 새롭게 생겨난 문제점인 것이다.

기존의 오버레이 멀티캐스트에 관한 연구는 효율적인 멀티캐스트 트리 구성을 위해 대부분 라우팅 프로토콜의 설계 분야에서 이루어져왔다.[1-10] 반면 멀티캐스트 트리를 재구성 하는데 중점을 둔 연구는 상대적으로 많이 이루어 지지 않았는데 이는 멀티캐스트 환경에서 하고자 하는 서비스의 종류에 따라 그 비중이 달라지는데도 한 원인이 있다. 멀티캐스트를 이용하여 일반 데이터를 전송하는 경우 최적의 경로를 통해 모든 데이터가 전송되는 것이 일차적인 목표이지만 실시간 방송 서비스와 같은 스트리밍 서비스를 하게 되는 경우에는 사용자에게 중간 노드들의 이탈로 인한 트리 구성의 변화에도 종단 호스트들이 최대한 영향을 받지 않도록 QoS (Quality of Service)를 보장 하는 것이 최우선이기 때문에 어느 정도 패킷 손실이 있을 지라도 빠른 시간 안에 트리를 복구해야만 한다.

이와 같이 멀티캐스트 트리를 복구하기 위한 기법으로는 크게 노드가 트리를 이탈한 후에 복구를 시작하는 Reactive approach 와 사전에 복구 계획을 세워두어 노드가 이탈하게 되면 즉시 트리를 재구성하게 되는 Proactive approach 로 나누어 볼 수 있다. 본 논문에서는 최소 RTT(Round Trip Time) 값을 사용하여 멀티캐스트 트리를 구성한 후 트리의 빠른 복구를 위해 Proactive approach 를 사용하여 실시간 방송과 같은 서비스에 적합한 오버레이 멀티캐스트 알고리즘을 제안 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 기술하고 3장에서는 제안된 모델의 개요 및 전체적인 동작 절차를 설명한다. 4장에서는 시뮬레이션을 통하여 기존 방안과의 비교 및 성능을 평가하고 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

Reactive approach 에서는 중간 노드가 이탈하였을 경우 일반적으로 해당 노드의 Grandfather 나 Root 노드에 Rejoin 시키는 방법으로 트리를 복구하는 반면에 Proactive approach 에서는 각각의 호스트가 백업 루트를 가지고 있음으로서 부모 노드의 이탈에 대비하게 된다.

Probabilistic Resilient Multicast(PRM)[11]는 각각의 호스트가 무작위로 선별한 일정한 수의 다른 호스트들에게 적은 확률로 데이터를 재 전달하는 방법을 사용한다. 이 기법은 트리의 연결이 끊어 졌을 때 무작위로 선택된 경로를 따라서 트리를 재구성하여 패킷 전송률을 증가 시킬 수 있다는 이점이 있지만 항상 추가적인 데이터를 다른 노드들에게 전송해야만 한다는 점에서 대역폭에 민감한 서비스의 경우에는 적합하지 않다는 단점을 가지고 있다.

Okada의 제안 기법[12]에서는 우선 최소 RTT 값으로 멀티캐스트 트리를 구성한 후에 두 번째 최소 RTT 값을 가지는 노드를 후보 부모 노드로 저장하여 백업 경로를 지정해 두는 방법을 사용한다. 이 알고리즘의 특징은 최소한의 작업으로 백업 노드를 선정 하고 자원의 선점 또한 없지만 백업 연결 요청이 이루어 졌을 때 Out Degree 값의 여유가 없다면 요청을 받아들일 수 없는 문제가 발생하게 된다.

Yang의 제안 기법[13]에서는 기본적으로 중간노드가 자신이 가지고 있는 자식 노드들을 위한 백업 경로를 사전에 선별하는 방법을 사용한다.

백업 경로의 설정을 담당하게 된 노드는 먼저 자식 노드들의 여유 Out Degree 값을 조사한다. 최대 Out Degree 값을 *n* 이라고 가정을 하고 자식 노드들의 여유 Out Degree 값의 합이 *n-1* 보다 적지 않다면 부모 노드와 자식노드들의 연결만으로 백업 경로를 설정할 수 있다. 하지만 그림 1에서와 같이 자식 노드들이 가지고 있는 여유 Out Degree 값의 합이 더 적다면 부모 노드와 자식노드들 사이의 RTT 값을 계산한 후 이 값이 가장 적은 노드가 부모노드

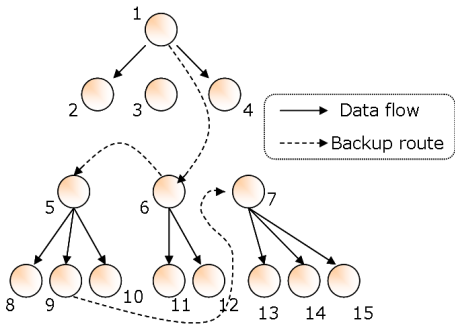

그림 1. Finding a backup route in Yang's approach

와의 백업 경로를 가지게 된다. 그 후, 아직 백업 경로를 갖지 못한 5,7번 노드의 서브 트리는 6번 노드와 그 자식 노드들인 11, 12번 노드와의 RTT 값을 계산하게 되고 이 중 가장 적은 값을 가지는 5번 노드가 6번 노드와 연결이 되며 나머지 7번 서브 트리는 이전 작업과 같이 여유 Out Degree를 가지고 있는 8,9,10 번 노드와의 RTT 값을 구해 자신의 새로운 부모 노드를 찾아 연결이 된다. 이 기법은 자원 선점 방식이 아닌 백업 경로만을 사전에 설정해 두는 방식이기 때문에 각 노드들의 자원을 효율적으로 사용하면서 빠른 트리 복구를 기대할 수 있다는 장점이 있지만 트리를 구성하고 있는 노드의 수가 많은 경우 복구 과정 시 여유 Out Degree가 있는 노드를 찾기 위한 작업의 반복으로 인해 복구 시간이 오히려 더 길어 질 수 있다는 단점도 가지고 있다.

Kusumoto의 제안 기법[14]은 모든 노드들이 항상 1 이상의 여유 Out Degree 값을 가지면서 자식 노드와 조부모 노드 사이의 연결 작업만으로 백업 경로를 설정한다.

그림 2에서 멀티캐스트 트리에 노드 8이 Join 하면 노드 2는 새로 업데이트 된 자식 노드 리스트를 노드 1에게 전송하고 노드 1은 이 리스트 안의 노드들과의 RTT 값을 계산한다. 이때 최소 RTT 값을 가지는 노드가 노드 1의 백업 경로를 가지게 되고 아직 백업 경로를 찾지 못한 노드는 이 전 작업에서 백업 노드를 찾은 노드와의 RTT 값을 계산하여 최소 RTT 값을 가지는 노드가 이 전 작업의 노드와 백업 연결을 가지게 된다. 하지만 백업 연결이 이루어 져 모든 Out Degree를 써버린 경우 다시 여유 Out Degree를 만드는 작업을 해야만 하는데 이 작업은 백업 연결로 새롭게 연결된 노드가 이웃 노드들에게 여유 Out Degree 값을 묻는 Query를

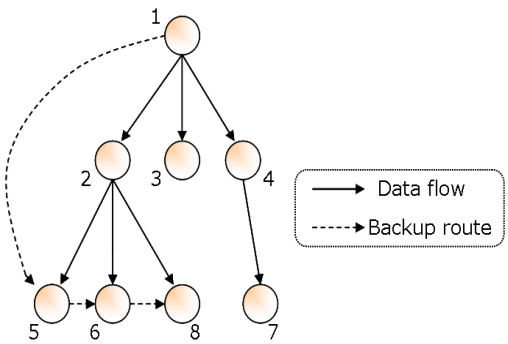

그림 2. Finding a backup route in Kusumoto's approach

보내 가장 먼저 이 값이 있음을 알리는 노드의 자식 노드가 되어 백업 연결을 마치게 된다. 이 알고리즘은 Yang의 제안 기법보다 백업 연결 설정과정을 단순화 하여 오버헤드 (Overhead)를 감소 시켰다는 장점이 있지만 여유 Out Degree를 확보하는 작업이 한 번에 끝나지 않는 경우 멀티캐스트 트리의 멤버가 많을수록 이 작업이 반복되어 복구에 긴 시간이 소요될 수 있다는 점과 무엇보다 항상 하나 이상의 Out Degree의 여유를 가지고 있음으로서 자원을 모두 사용하지 못해 트리의 깊이 (Depth)가 상대적으로 길어지는 문제점을 가지고 있다.

Ⅲ. 제안하는 Overlay Multicast 모델

3.1 개 요

제안 모델에서 서비스를 제공하는 Root 노드를 제외한 모든 노드는 Service Out Degree 외에 Backup Out Degree 를 가진다. 때문에 한 노드가 사용 할 수 있는 최대 Out Degree 값이 m 이라고 하고 Backup Out Degree 로 n 의 값이 필요 하다면 이 노드가 Service Out Degree 로 사용 할 수 있는 값은 m-n 이 되게 된다. 각 노드가 Backup Out Degree 를 별도로 가지고 있는 이유는 관련 연구에서 살펴보았던 Proactive approach의 또 다른 방안으로서 어느 한 노드의 부모 노드가 이탈 하게 되면 즉시 백업 노드에게 서비스를 요청하게 되고 Backup Out Degree 로 예약해 두었던 자원을 서비스용으로 전환 하여 빠른 시간 안에 다시 서비스를 받게 할 수 있게 하기 위해서이다. 하지만 각각의 노드들이 Backup Out Degree 값을 가지게 되어 멀티캐스트 트리를 구성하게 되면 노드의 자원을 모두 활용하지 못해 전체적인 트리의 깊이가 길어지게 되고 이 트리가 가지게 되는 평균 RTT 값이 증가되게 되는 결과로 이어진다. 때문에 최상위 노드인 Root 노드의 Out Degree 만은 모두 서비스용으로 사용함으로서 이러한 단점을 완화시키게 되었다.

멀티캐스트 트리에서 Root 노드는 새로운 호스트가 멀티캐스트 그룹에 가입하고자 할 때 일차적으로 접속하게 되는 랑데부 (rendezvous)포인트의 기능을 수행한다. 이때 Root 노드는 새로운 호스트에게 현재 그룹에 가입할 수 있는 정보를 주기위해 PPL (Portemtial Parent List)을 전송해 주는데 이 PPL의 구조는 그림 3과 같다. PPL은 서비스를 제공 해 줄 수 있는 노드의 IP 주소와 그 노드가 부모 노드를 거쳐 Root 노드까지 가지는 RTT 값으

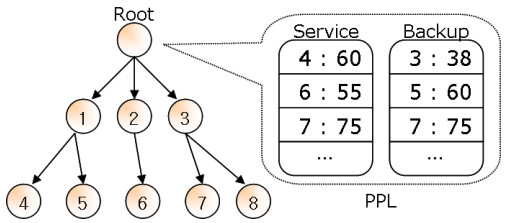

그림 3. Structure of Potential Parent List of Root node

로 구성이 된다. 일반적으로 PPL은 서비스를 제공해줄 수 있는 노드들에 대한 정보만을 가지고 있지만 본 제안 기법에서는 이 정보 외에 Backup Out Degree의 정보까지 가지고 있음으로서 노드들이 자신의 백업 노드를 선정할 수 있도록 해주고 Root 노드는 멀티캐스트 그룹에 새로운 노드가 가입하고 떠나거나 노드들이 가지고 있는 정보가 변경될 때마다 이 정보를 갱신하여 항상 최신의 트리 정보를 유지하게 된다.

3.2 클라이언트의 가입

위의 그림 4는 새로운 호스트가 멀티캐스트 그룹에 가입하는 과정의 예를 보여주고 있다. 새로운 호스트 *N*은 제일 먼저 Root 노드에게 서비스를 요청하고 Root 노드는 자신의 Out Degree 정보를 확인한다. 만약 자신이 직접 서비스를 제공할 수 있는 상태라면 바로 서비스 요청을 수락하는 Ack 메시지를 전송하여 서비스를 제공하게 되지만 그렇지 못한 경우에는 그룹 멤버들로부터 수집한 PPL을 대신 전송하게 된다. 이를 수신한 노드 N은 이 리스트 내에 있는 노드들과의 RTT 값을 체크하는 작업을 수행하게 되고 이 때 얻어진 값을 PPL 내의 각 노드의 값과 합산하여 전체적인 경로의 RTT 값을 계산한다. 이렇게 얻어진 값은 적은 순서대로 정렬이 되어 순차적으로 서비스를 요청하게 되는데 만약 하나 이상의 노드들이 동시에 그룹에 가입하는 작업을 하고 있었다면 성능이 좋은 특정 한 노드가 동시에 다수의 노드로부터 서비스 요청을 받게 되는 상황이 발생할 수 있다. 이 경우 가장 먼저 서비스를

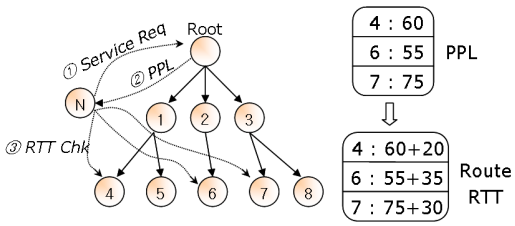

그림 4. Example of Join procedure

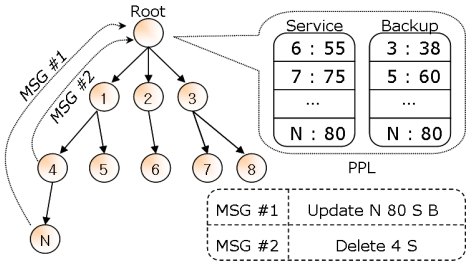

그림 5. Example of PPL update procedure

요청한 노드 순으로 서비스를 제공하고 더 이상 서비스 요청을 받아들일 수 없는 상태가 되면 Nack 메시지를 통해 서비스가 불가능함을 알린다. 이를 수신한 노드는 정렬된 PPL로부터 그 다음 노드를 선택하여 다시 서비스를 요청하는 작업을 수행하게 된다.

위의 작업을 거쳐 노드 N이 자신의 부모 노드를 찾아 서비스를 받게 되면 그림 5와 같이 새로 멀티캐스트 그룹에 가입한 노드 N 과 Out Degree 정보가 변경이 된 노드 4는 Root 노드에게 자신들의 새롭게 변경된 정보를 알려준다. 새롭게 가입한 노드 N은 자신이 가지고 있는 전체 RTT 값과 Service와 Backup Out Degree 값에 여유가 있음을 알려주는 메시지를 전송하게 되고 새로운 자식 노드를 가지게 된 노드 4는 더 이상 자식 노드를 수용할 수 없게 된 경우 PPL 로부터 자신을 삭제하라는 메시지를 전송한다. 이러한 방법으로 Root 노드는 항상 그룹 멤버들의 최신의 정보를 가지고 멀티캐스트 트리를 구성할 수 있는 준비를 할 수 있게 된다.

3.3 복구 계획

새로 가입한 호스트가 멀티캐스트 트리의 구성원이 되어 서비스를 받게 되면 자신의 부모 노드가 이탈해버리는 상황을 대비하기 위해 백업 노드를 찾는 작업을 수행 한다. 백업 노드를 찾는 작업은 기본적으로 각 노드들에게 할당해 주었던 Backup Out Degree 값을 이용하여 이루어진다. 백업 노드가 필요한 노드는 Root 노드가 가지고 있는 PPL의 Backup 정보를 받아와 자신의 부모 노드를 찾는 과정과 동일하게 최적의 경로를 가지는 노드를 검색 한 후 백업 노드 요청을 하게 되고 이를 수락한 노드는 자신의 Backup Out Degree 값을 증가 시켜 더 이상 백업 요청을 받아들일 수 없다면 Root 노드에게 자신을 Backup PPL에서 삭제하라는 메시지를 전송한다. 하지만 이러한 기법만을 사용하게 될 경우 부모 노드가 점유하고 있었던 자리는 어느 새로운 노드가 멀티캐스트 그룹에 새로 참여 할 때

까지 비어있게 되어 자원의 낭비가 발생하게 된다. 이러한 문제를 해결하기 위한 방안으로 백업 노드를 찾는 작업에 앞서 자신의 조부모 노드와의 RTT 값을 계산한다. 만약 자신이 부모 노드의 유일한 자식 노드라면 이 값에 관계없이 백업 노드는 조부모 노드로 선택이 되지만 이 연결은 Backup Out Degree 값을 선점하지 않는다. 그 이유는 이러한 백업 연결은 부모 노드의 이탈에 대비하는 것이므로 자신은 부모 노드가 점유하고 있었던 장소에서 다시 서비스를 받게 되므로 추가적인 Out Degree 자원은 필요로 하지 않기 때문이다. 부모 노드가 새로운 자식 노드를 받아들이게 되면 새로 들어온 노드 역시 조부모 노드와의 RTT 값을 계산한다. 이 값이 이미 조부모 노드를 백업 노드로 가지고 있었던 노드가 측정했었던 RTT 값 보다 크다면 위에서 설명 하였던 Backup PPL을 이용한 백업 노드를 찾는 작업을 수행 하지만 이 값이 더 작다면 새로운 노드가 조부모 노드를 백업 노드로 선택하게 되고 이전에 조부모 노드를 백업노드로 가지고 있었던 노드는 새로운 백업 노드를 찾는 작업에 들어가게 된다.

그림 6은 이와 같은 과정을 그림으로 표현한 것이다. 새롭게 그룹에 가입한 노드 N은 노드 1과 RTT 값을 계산하여 노드 9가 가지고 있었던 값과 비교를 했다. 위의 예제에서는 비교한 값의 결과가 노드 N이 더 작아 노드 N이 노드 1을 백업 노드로 선정하게 되었고 노드 9는 새로운 백업 노드인 노드 6을 선택하게 되어 Out Degree 정보는 노드 6의 Backup 정보만 바뀌게 된다. 이 작업은 노드의 가입과 이탈로 인한 자신의 형제 (Sibling)노드나 조부모 노드가 바뀌는 변화가 있을 때 마다 반복 수행되어 항상 최적의 경로를 가지는 노드가 백업 노드로 선정될 수 있게 한다.

하지만 백업 노드를 선정하는데 있어서는 몇 가지 제약 조건이 필요하다. 이는 백업 노드를 이용하여 다시 서비스를 받을 때 이 연결이 Looping을 발

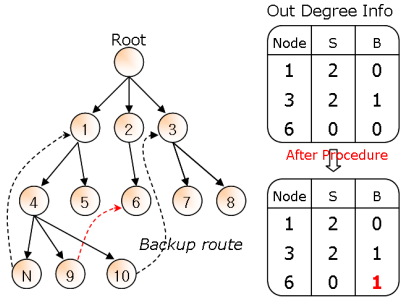

그림 6. Example of Backup node selection

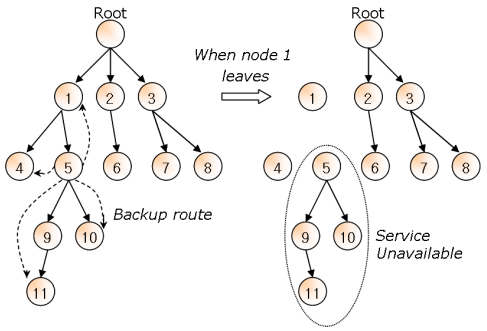

그림 7. Invalid backup route case 1

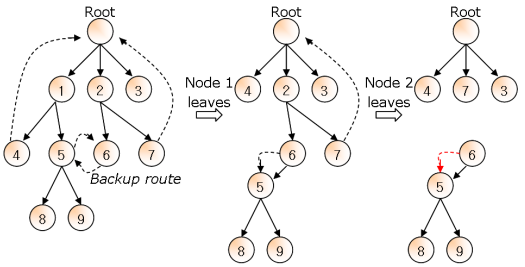

그림 8. Invalid backup route case 2

생시키는 것을 방지하고자 함인데 그림 7과 그림 8에서 그 예를 보여주고 있다.

첫 번째 제약 조건은 부모 노드와의 백업 연결이다. 노드 5가 백업 노드를 찾아야 된다면 Backup PPL로부터 그 후보 노드를 선택하게 되는데 만약 그 후보 노드가 자신의 부모 노드라면 부모 노드가 이탈한 경우 백업 노드로서의 의미가 사라지게 되므로 이 연결은 피해야 된다. 두 번째는 자신의 형제 노드들과의 연결이다. 만약 노드 4와 노드 5가 서로 맞물려서 백업 노드 연결을 맺고 있다면 부모 노드가 이탈하게 된 후 서로 서비스를 요청하는 문제가 발생한다. 세 번째로는 자신의 하위 노드들에 대한 백업 연결이다. 이러한 연결은 노드 5가 새로운 백업 노드가 필요하여 재 연결 작업을 요청하는 경우에 발생할 수 있는데 이 때 자신의 하위 노드들과 연결을 맺게 되면 부모 노드가 이탈한 후 자신의 하위 노드에게 서비스를 요청하여 서비스를 다시 받을 수 없는 문제가 발생한다. 마지막은 그림 8과 같이 부모 노드가 다른 노드 5와 노드 6이 서로 백업 요청을 하고 있는 경우이다. 이러한 상황에서는 두 노드의 부모 노드가 모두 이탈 하지만 않으면 문제가 발생하지 않지만 모두 이탈하게 된다면 결국 그림 8과 같이 서로에게 서비스를 요청하는 상황이 발생함으로 이러한 연결 또한 사전에 예방해야 한다.

본 제안 기법에서 백업 노드를 사용할 때 처음에 설정한 백업 노드는 시간이 흐르면서 부분적인 트리의 구조가 변화함에 따라 최적의 경로를 가지는 백업 노드일 확률이 감소하게 된다. 때문에 항상 최적의 경로를 가지는 멀티캐스트 트리를 구성하고자 백업 노드를 갱신하는 작업을 할 수 있다. 이 작업은 노드의 이탈이 발생했을 때 그 영향을 받는 모든 노드들이 가지고 있는 백업 연결을 갱신하면 되는데 이탈한 노드가 상위 노드일 수 록 이 작업에 부하가 크다는 단점이 있다. 이러한 이유로 상위 노드부터 순차적으로 새로운 백업 노드를 찾는 작업을 수행하여 동시에 많은 수의 노드들이 백업 노드를 찾는 부하를 줄였고 한 순간에 하나의 노드만이 작업을 하여 백업 노드를 찾는 과정 중에 부모 노드가 이탈하여 장시간 서비스를 받지 못하는 확률을 최대한 감소 시켰다.

Ⅳ. 성능 평가

4.1 시뮬레이션 환경

본 논문에서는 제안된 모델의 성능을 평가하기 위해 Matlab 7.2.0 환경 상에서 최대 50개의 노드를 생성하였고 멀티캐스트 트리 구성 시 노드들이 짧은 지연(Delay) 시간을 가지고 있는 특정 노드들에게만 몰리는 것을 방지하기 위해 각 노드들의 지연 시간을 30 ms 에서 90 ms 사이의 비교적 그 값의 차이가 크지 않은 임의의 값으로 주었다. 한 개의 노드가 가지는 최대 Out Degree 값은 100Mbps의 네트워크 환경에서 25Mbps 정도의 대역폭이 필요한 HD 급 영상을 서비스 한다고 가정하여 4로 정의 하였고 노드를 거치면서 발생 할 수 있는 작업(Processing) 지연 시간은 모든 노드가 동일한 환경 하의 시스템이라고 가정하여 환경 요소에서 제외 하였다.

제안된 알고리즘의 성능 비교를 위해 관련 연구에서 살펴보았던 Yang의 제안 기법과 Kusumoto의 제안 기법을 동일 환경 하에서 프로그래밍 후 테스트 하였다. 단 복구 알고리즘 상의 비교를 위해 멀티캐스트 트리의 구성 알고리즘은 본 논문에서 제안한 각 경로가 가지는 최소 RTT 값을 이용하여 트리를 구성하는 방법으로 통일하였다. 또한 제안된 기법에서의 Backup Out Degree 값은 1로 설정하여 멀티캐스트 트리 구성 시 발생하게 되는 다른 제안 기법과의 평균 RTT 값의 차이를 최소화 하였다.

4.2 결과

첫 번째 시뮬레이션에서는 트리를 구성하는 노드의 수를 10개부터 50개까지 증가시켜가며 이 때 구성이 된 트리의 평균 RTT 값을 측정하였고 이렇게 구성이 된 트리에서 특정 한 노드를 이탈 시켰을 때 변화된 평균 RTT 값과 트리의 복구 시간을 측정하였다.

그림 9에서는 제안된 기법과 Yang의 기법이 적은 수의 노드로 트리가 구성이 될 때에는 값의 차이가 없다가 노드의 수가 증가해 가면서 그 차이가

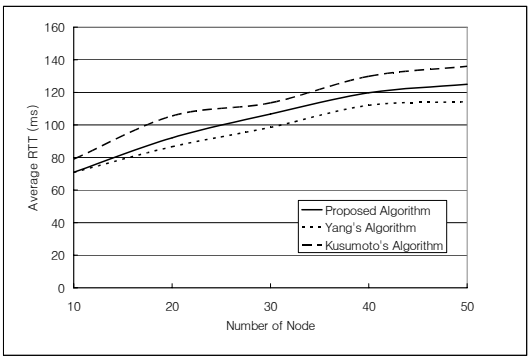

그림 9. Average RTT before node fails

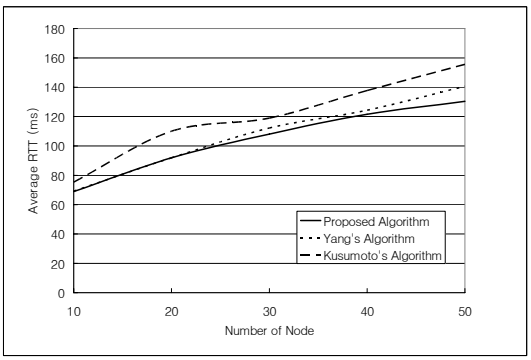

그림 10. Average RTT after tree reconstruction

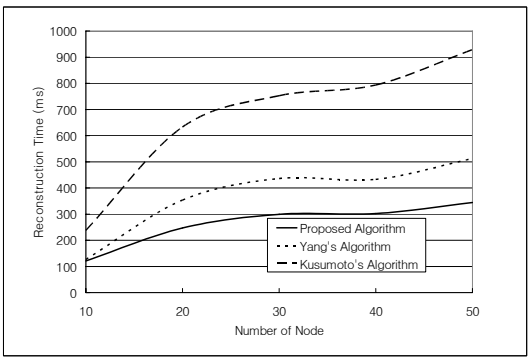

그림 11. Variation of tree reconstruction time

생기는 반면 Kusumoto의 기법은 초기부터 성능이 비교적 좋지 않음을 보여주고 있는데 이는 트리 구성 시 사용할 수 있는 Out Degree 값의 제한 때문이다. Yang의 기법에서는 한 노드가 사용할 수 있는 Out Degree 값을 모두 서비스용으로 사용하는반면 Kusumoto의 기법에서는 모든 노드가 항상 1이상의 여분의 Out Degree 값을 가지고 있어야 한다. 이러한 이유로 같은 수의 노드로 트리를 구성하게 되면 Out Degree 값을 적게 사용하는 쪽이 상대적으로 트리의 깊이가 증가하여 전체적인 RTT 값을 증가 시키는 결과로 이어진다. 반면 제안된 기법에서는 Kusumoto의 기법과 같이 백업 노드를 위해 1 이상의 여분의 값이 필요하지만 최상위 노드인 Root 노드의 Out Degree 값만은 모두 서비스용으로 사용하게 함으로서 평균 RTT 값의 차이를 줄일 수 있게 되었다.

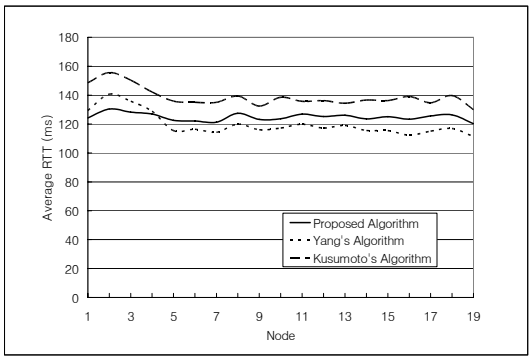

그림 12. Average RTT after tree reconstruction

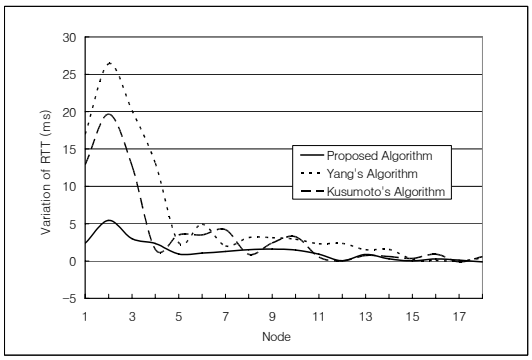

그림 13. Variation of RTT after tree reconstruction

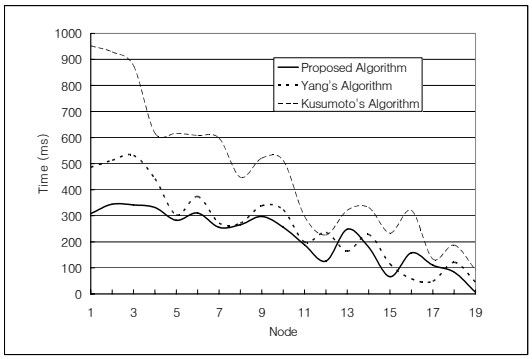

그림 14. Sum of tree reconstruction time

그림 10과 그림 11은 트리가 구성될 때 2번째로 가입한 노드를 이탈시켜 복구 후 변화된 평균 RTT 값과 복구에 소요된 시간을 측정한 것이다. 이 시뮬레이션에서는 상위 노드를 이탈시킴으로서 그 영향을 받는 노드의 수가 변화할 때 미치는 영향을 알아보고자 하였다. 적은 수의 노드로 트리가 구성이 되었을 때에는 제안된 기법과 Yang의 기법 사이에 차이는 근소하지만 노드의 수가 증가 할수록 제안된 기법의 성능이 더 좋음을 볼 수 있는데 이는 복구가 필요한 노드의 백업 경로가 제안된 기법에서는 비교적 상위 노드에 분포하고 있지만 Yang의 기법에서는 부모 노드의 자리를 차지한 한 개의 자식 노드를 제외 하면 모두 하위에 위치한 노드에게서 다시 서비스를 받기 되는 확률이 크기 때문이고 이 때 복구 작업이 하위 노드로 계속 내려가면서 발생하게 되는 시간 때문에 복구 시간 또한 길어지게 된 결과이다.

두 번째 시뮬레이션에서는 3개의 기법 모두 50개의 노드들로 트리를 구성한 후 특정 노드를 선택해 가며 이탈 시켰을 때의 변화를 측정하였다. 그림 12에서는 상위에 위치한 노드들이 이탈을 할 경우 제안된 기법이 좀 더 적은 RTT 값을 가지지만 하위에 위치해 있는 노드가 이탈 할수록 그 영향을 받는 노드의 수가 감소하기 때문에 Yang의 기법의 성능이 더 좋은 결과를 볼 수 있다. 하지만 동일한 상황에서 복구가 일어났을 때 복구 전의 평균 RTT 값과의 변화율을 측정해 본 결과 제안된 기법은 그 차이 값이 5ms 이하였던 것에 비해 Yang의 기법이 가장 심한 변화를 보여임을 그림 13에서 확인해 볼 수 있다. 이와 같은 결과 역시 상위 노드일수록 그 영향을 받는 노드의 수가 증가하기 때문인데 Kusumoto의 기법의 경우에는 Out Degree 값의 제한으로 인해 상대적으로 영향을 받는 노드의 수가 Yang의 기법 보다 적어 변화율이 작게 나타났다. 이 변화율은 위의 알고리즘들을 사용하여 서비스를 하게 되면 복구 후 값의 차이가 클수록 사용자에게 미치는 영향이 심해서 결과적으로 QoS를 감소시키는 결과로 이어지기 때문에 값의 차이가 적을수록 좋은 알고리즘이라 할 수 있다. 그림 14는 복구에

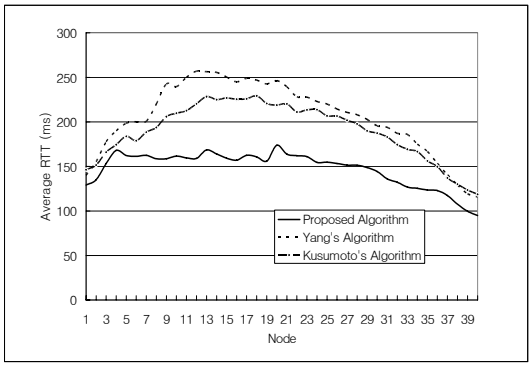

그림 15. Variation of average RTT when the node fails consecutively - Case 1

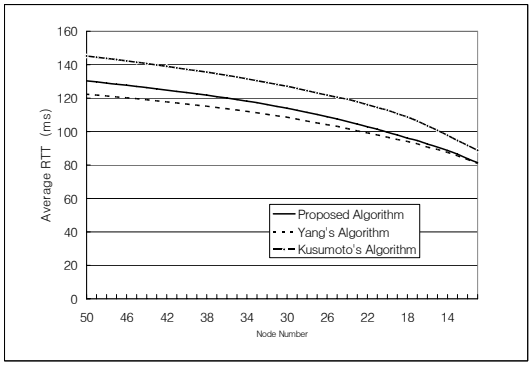

그림 16. Variation of average RTT when node fails consecutively - Case 2

소요된 시간의 합을 측정한 것인데 백업 노드를 사용하여 바로 서비스를 받는 제안된 기법이 가장 짧은 시간이 소요된 반면 Kusumoto의 기법은 상대적으로 트리의 깊이가 더 크기 때문에 상위에 위치한 노드가 이탈할수록 Out Degree 값에 여유가 있는 노드를 찾는 시간이 길어져 가장 긴 시간이 소요된 결과를 볼 수 있다.

마지막 시뮬레이션에서는 멀티캐스트 트리가 서비스를 종료하는 상황과 같이 어느 순간부터 다수의 노드가 연속적으로 이탈을 할 때 트리의 변화를 측정하였다. 먼저 그림 15에서는 상위 노드부터 연속적으로 이탈을 시켜가며 평균 RTT 값을 측정한 결과 제안된 기법은 RTT 값에 큰 변화를 보이지 않고 있다가 트리를 구성하는 노드 수의 감소로 인해 RTT 값도 감소하는 것을 볼 수 있었던 반면 Yang과 Kusumoto의 기법에서는 전체 노드의 1/3 이상이 사라졌음에도 RTT 값은 오히려 100ms 이상 증가하다가 감소하는 것을 살펴볼 수 있다. 그림 16은 위의 상황과는 반대로 하위 노드부터 이탈을 시켰는데 이 경우 트리의 복구는 전혀 일어나지 않았기 때문에 트리가 구성이 되던 상황과 대칭이 되는 결과 값이 나오게 되었다.

Ⅴ. 결 론

본 논문에서는 오버레이 멀티캐스트에서 중간 노드가 이탈하였을 때 발생하는 문제점을 해결하기 위해 Backup Out Degree를 이용하여 트리를 재구성 하는 모델을 제안하였다. 제안된 모델에서 Backup Out Degree는 어느 한 노드가 자신의 부모 노드를 잃어 버렸을 때 다시 새로운 부모 노드를 찾는 과정 없이 바로 서비스를 받을 수 있게 해주었다. 이러한 방법은 노드가 가지고 있는 Out Degree 자원을 예약하기 때문에 트리 구성적 측면에서는 성능이 떨어지게 되지만 Root 노드의 Out Degree 자원을 모두 서비스용으로 사용하여 이를 개선하였다. 또한 백업 노드 갱신작업을 통해 항상 최적의 경로를 가지는 트리를 구성하게 되었다.

모의실험을 통해서 기존의 제안 기법들과 비교한 결과 본 논문의 제안 기법이 복구 작업에서 가장 적은 시간이 소요되었고 RTT 값의 변화 또한 가장 작았음을 볼 수 있었다. 평균 RTT 값 또한 많은 수의 노드가 영향을 받을 때에는 가장 작은 값을 가짐을 확인 하였다. 따라서 제안된 모델은 멀티캐스트 그룹의 이탈 빈도가 낮은 안정적인 노드들로 구성이 된 트리보다는 이탈이 빈번한 노드들로 멀티캐스트 트리를 구성해야 되는 상황에서 효율적으로 적용될 수 있는 기법이 될 수 있음을 알 수 있다.

참 고 문 헌

〔1〕 Y. Chu, S. G. Rao, H. Zhang, "A Case for End System Multicast," in Proceedings of ACM SIGMERTICS 2000, June.

〔2〕 D. Pendarakis, S. Shi, D. Verma, M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure", 3rd USENIX Sysmposium on Internet Technologies and Systems, Mar 2001.

〔3〕 Y. Chawathe, S. McCanne, E. Brewer, "Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service" PhD Thesis, University of California, Berkeley, 2000

〔4〕 P. Francis, "Yoid: Extending the Internet Multicast Architecture", http://www.icir.org/yoid/ 1999
〔5〕 J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, J. O'Toole, "Overcast: Reliable Multicasting with an OVerlay Network", 4th Symposium on Operating Systems Design & Implementation, Oct. 2000
〔6〕 H. Deshpande, M. Bawa, H. Garcia-Molina, "Streaming Live Media over Peers", Technical Report 2002-21, Stanford University, Mar. 2002
〔7〕 S. Zhuang, B. Zhao, A. Joseph, R. Kartz, S. Shenker, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination", ACM NOSSDAV 2001, June.
〔8〕 S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level Multicast using Content-Addressable Networks", In Proceedings of NGC, 2001
〔9〕 D. Tran, K. Hua, T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming", in proceedings of IEEE INFOCOM 2003, Apr.
〔10〕 S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications", in proceedings of IEEE INFOCOM 2003, Apr.
〔11〕 S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays", in Proceeding of ACM Sigmetrics 2003, June. san Diego, CA.
〔12〕 Y. Okada, M. Oguro, J. Katto, S. Okubo, "A New Approach for the Construction of ALM Trees using Layered Video Coding", in Proceedings of ACM Multimedia 2005
〔13〕 M. Yang, Z. Fei. "A Proactive Approach to Reconstructing Overlay Multicast Trees", in proceedings of INFOCOM 2004, March
〔14〕 T. Kusumoto, Y. Kunichika, J. Katto, S. Okubo, "Proactive Route Maintenance and Overhead Reduction for Application Layer Multicast", ICAS/ICNS 2005

허 권 (Kwon Heo) 준회원

2005년 2월 전남대학교 정보통신공학과 졸업
현재 전남대학교 컴퓨터공학과 석사과정
<관심분야> 컴퓨터 네트워크, 라우팅 프로토콜

손 승 철 (Seung-Chul Son) 정회원

2002년 2월 전남대학교 컴퓨터공학과 졸업
2003년~2004년 (주)금영 음향연구소
현재 전남대학교 컴퓨터공학과 석사과정
<관심분야> 컴퓨터 네트워크, 통신 프로토콜

김 경 훈 (Kyung-Hoon Kim) 정회원
2000년~2003년 (주)포스트림 개발팀
2002년 2월 전남대 컴퓨터공학과 (박사수료)
2000년 2월 전남대 컴퓨터공학과(공학석사)
2004년 1월~현재 (주)포스트림대표
<관심분야> 실시간통신 프로토콜, 분산처리, P2P

송 호 영 (Ho-Young Song) 정회원
1983년 홍익대학교 전자계산학과 학사 졸업
2003년 충북대학교 정보통신공학과 박사 수료
1983년~현재 FTTH 서비스 팀장
2003년~현재 초고속정보통신건물인증위원회위원
2005년~현재 FTTH산업협의회 기술/표준화분과위원
2006년~현재 한국 ITU-T 연구위원회 위원
<관심분야> 통신, FTTH, 멀티미디어, 통신방송, 융합서비스

남 지 승 (Ji-Seung Nam) 종신회원
1992년 Univ. of Arizona, 전자공학과(공학박사)
1992년~1995년 한국전자통신연구소 선임연구원
1999년~2001년 전남대학교 정보통신특성화 센터장
2001년~현재 전남대학교 인터넷창업보육 센터장
1995년~현재 전남대학교 컴퓨터공학과 교수
<관심분야> 통신 프로토콜, 인터넷 실시간서비스, 라우팅