고장 감내를 위한 이중루트 CBT 기법
(Double Rooted Core based Tree for Enhancing Fault Tolerance)

오 윤 근 † 조 평 동 †† 김 남 훈 ††† 이 영 희 ††††
(YoonKeun Oh)(Pyung-Dong Cho)(Namhoon Kim)(Young Hee Lee)

요 약 정보화 시대가 열림에 따라 인터넷은 사회 전반에 걸쳐 매우 중요한 인프라구조가 되었고, 인터넷을 통한 서비스의 중요성 또한 대두되고 있다. 멀티캐스트는 차세대 인터넷 서비스의 대표적인 예로써, 매우 활발한 연구가 진행되고 있다. 본 논문에서는 멀티캐스트 서비스의 신뢰성을 높이기 위해 이중루트를 갖는 CBT기법을 고안하였다. DRCBT는 CBT방법에서 가장 중요한 코어를 백업함으로써, MCBT방법과 CBT방법에 비해 우수한 성능을 보였다. 제안한 DRCBT방법의 고장 감내도를 분석하기 위하여 고장 시 재구성해야 하는 트리의 hop 수에 관한 수식을 도출하고 각 변수의 변화에 따른 결과를 도시하였다.

Abstract The Internet has become very important infrastructure over the world and the evolution toward next generation Internet becomes a key issue for next era. The multicast is one of the key issues of next generation Internet services, and many researchers have proposed various multicast protocols. In this paper, we propose a new multicast routing protocol, DRCBT that has survivability. In the DRCBT (Double Rooted Core Based Tree) a core back up another cores routing table. By backing up another cores routing table the DRCBT shows a good performance on fault tolerance level compared with CBT and MCBT when core fault is happened. For analyzing the fault tolerance level of the DRCBT, we develop a formula that considered total hop counts of reconstruction when core fault happened.

1. 서 론

정보화 시대가 도래하면서 정보통신 분야의 발전 정도는 한 국가의 경쟁력을 의미할 정도로 매우 중요한 위치를 차지하게 되었다. 대부분의 선진국들도 정보통신 분야에 많은 자본 및 자원, 인력을 집중시키고 있다. 이러한 추세에 따라 대부분의 국가들은 각국 내에 네트워크 인프라구조를 구축하고 있으며, 국내의 구축을 넘어 여러 나라를 잇는 인프라구조 구축에도 힘쓰고 있다. 이러한 네트워크 인프라구조는 국가 전체 산업의 근간이 되므로, 네트워크 전체가 항상 안정적으로 운영되고 신뢰도를 갖추어야 한다. 왜냐하면, 국가의 기간망이 멈추거나 오 동작을 한다는 것은 매우 치명적인 일이기 때문이다.

또한, 인터넷과 전기 통신망의 발전은 네트워크 인프라의 고속화와 광대역화를 야기시켰고, 그에 따라 통신망 장애는 더욱 더 많은 데이타에 대한 손실 문제를 불러일으키게 되어 보다 장애에 대한 대책이 보다 중요한 쟁점이 되었다.

네트워크 생존성(Network Survivability)은 다양한 복구 기술을 적용하여 네트워크 장애 시 네트워크를 계속 유지하거나 네트워크의 성능을 받아들일 수 있는 레벨까지 회복할 수 있는 능력을 말하거나, 또는 예방 기술을 적용하여 잠정적인 네트워크 장애로부터 서비스 중단을 방어하거나 완화하는 것을 의미한다. 통신망에 미치는 장애는 케이블의 절단, 화재, 사람의 오류, 파괴 등에 의해 발생할 수 있다[1-2].

네트워크 인프라의 고속화, 광대역화가 네트워크 유지

† 학생회원 : 한국정보통신대학원대학교 공학부
ykoh@icu.ac.kr
†† 비 회 원 : 한국전자통신연구원 표준연구센터 연구원
pdcho@pec.etri.re.kr
††† 비 회 원 : 한국정보통신대학원대학교 공학부
nhkim@icu.ac.kr
†††† 정 회 원 : 한국정보통신대학원대학교 공학부 교수
yhlee@icu.ac.kr
논문접수 : 1999년 11월 29일
심사완료 : 2000년 5월 24일

성을 매우 중요하게 부각시킨 것과 더불어 정보 통신의 발전으로 인해 네트워크 자체가 사람들에게 생명선으로 여겨지게 되었다. 이에 세계 각국은 자국의 네트워크 인프라구조를 구축하는 데 신경을 쓰고 있는 것은 물론 통신망 자체의 관리에 주력하며, 특히 네트워크 생존성과 신뢰성에 주안점을 두고 연구하고 있다[3-4].

네트워크 생존성 문제는 주로 전기 통신망을 중심으로 연구되고 있으나 본 논문에서는 현재 가장 커다란 네트워크로 성장한 인터넷상에서의 생존성 문제를 다룬다. 특히 차세대 인터넷의 필수적인 서비스인 멀티캐스트 기술 제공하에 생존성을 고려하였다. 전기통신망에서의 생존성 문제가 주로 물리적인 장비들에 대한 복구 및 그에 대한 대책이라면, 본 논문에서 제안한 인터넷상에서의 네트워크 생존성 문제는 제공되는 멀티캐스트 서비스의 신뢰도와 안정성을 높이는 데 주 목적이 있다고 하겠다.

본 논문은 기존의 멀티캐스트 라우팅 기법인 CBT방법을 개선하여 멀티캐스트 서비스의 신뢰도 향상을 도모하였다. 기존의 CBT방법이 단일 코어를 사용하는 반면 본 논문은 이중 루트 코어를 사용하는 방법을 제안하였다. 코어들은 서로 주기적으로 자신의 생존을 알리는 구조를 가지며, 만일 하나의 코어의 고장이 감지되었을 때, 기존의 방법에서 볼 수 없는 신속한 서비스 복구가 이루어지게 된다.

본 논문이 구성은 다음과 같다. 2장에서는 네트워크 생존성과 멀티캐스트 기법 및 관련 연구와 그 동향에 대해서 살펴보고, 3장에서는 제안한 이중 코어 멀티캐스트 기법에 대하여 기술한다. 4장에서는 제안한 방법의 성능을 분석하고 , 5장에서는 결론 및 향후 연구 방향에 대하여 기술한다.

2. 관련 연구 및 동향

2.1 멀티캐스트 라우팅 프로토콜

멀티캐스트 라우팅 프로토콜은 크게 소스 기반 트리(source based tree) 구조를 갖는 것과 공유 트리(shared tree) 방식으로 나눌 수 있다. 소스 기반 트리 방식은 송신자마다 각각의 멀티캐스트 트리를 가지고 있는 방식으로 대표적인 것으로 DVMRP(Distance Vector Multicast Routing Protocol)와 MOSPF (Multicast Open Shortest Path First)가 있다[5-6]. 이에 반해 공유 트리 방식은 하나의 멀티캐스트 그룹은 하나의 트리를 공유하며 그 그룹 내에서 각 송신자는 코어 또는 RP(Rendezvous point)로 메시지를 보내고 코어 또는 RP에서 멀티캐스트를 담당하는 방식을 말한다[7]. 공유 트리 방식의 대표적인 것으로 CBT(Core Based Tree)[8]가 있으며 이 방식은 하나의 대표노드가 멀티캐스트 그룹 내에서 송신자의 역할을 하며 그룹 멤버 중 메시지를 전송하려는 노드는 다른 그룹 멤버에게 멀티캐스트 할 필요 없이 이 코어 노드로 유니캐스트로 메시지를 전송하면 코어가 알아서 메시지를 멀티캐스트 해 주는 방식이다. CBT 외에 공유 트리 구조로서 CBT 백본(backbone) 위에 여러 개의 코어가 존재하는 다중 코어(Muiti-Core) CBT 방법과 지역을 구분해 각 지역별로 코어를 갖는 DCBT(Decentralized Core Based Tree)방법[9]이 있다. 또한 공유 트리와 소스 기반 트리 방식을 혼합한 형태의 PIM(Protocol Independent Multicast)방식[10- 11]의 멀티캐스트 프로토콜이 있다.

2.2 네트워크 생존성에 관한 연구

네트워크 생존성에 관한 연구는 주로 국제 표준화 기구를 중심으로 한 연구 혹은 국책 연구 등의 형태로 이루어지고 있는 것을 볼 수 있다. 우선 T1 위원회(Committee T1)는 90년대 초부터 네트워크의 다양한 분야에 걸쳐 표준화활동을 하고 있으며, 통신망 생존성과 신뢰성 부분에 있어서도 매우 활발한 활동을 보이고 있다. T1 위원회는 화재 예방, 전력, E911, ISDN, SS7 구조(architecture), SONET과 개인 휴대 통신을 위한 무선 액세스(wireless access) 부분까지 포함해 네트워크 생존성에 대한 개발과 그에 대한 지침서와 관련 문서를 제공하려 하고 있다. 또 최근 정지 색인(outage index)을 마련하여 네트워크 생존성에 대한 기준을 제시하고 있으며, 향후 활발한 활동을 계속 보일 것으로 예상된다. ITU-T도 스타디 그룹 2, 스타디 그룹 4, 스타디 그룹 5, 스타디 그룹 11, 스타디 그룹 13, 스타디 그룹 15가 각 스타디 그룹 내의 연구 주제와 더불어 네트워크 생존성과 관련하여 연구하고 있다[1-2].

표준화 기구 외에 정부 차원에서 DARPA가 지원하는 많은 프로젝트가 진행 중이다. DARPA는 고 신뢰 네트워크(High Confidential Network)라는 부분을 지원하고 있는데, 이 분야는 주로 고의로 망에 스트레스를 주는 것을 방지하는 네트워크 보안에 관한 사항과 더불어 네트워크 생존성에 대한 많은 프로젝트를 지원하고 있다. 벨코아(Bellcore)는 큰 네트워크 망의 생존성을 위한 트래픽 관리, 네트워크 생존성을 위한 독립적인 모니터링, 소프트웨어 기술을 도입한 생존 액티브 네트워크(survivable active networking)등 다양한 분야에 걸쳐 연구를 수행하고 있다. 이외에도 델라웨어(Delaware) 대학에서는 실시간 네트워크 서비스 상에서의 생존성에

대하여 연구하고 있으며, 미져리-캔사스(Missouri-Kansas City) 대학은 큰 범위에 이종적인 성격을 지니는 정보 시스템의 생존성에 대하여 연구하고 있다. 캘리포니아(California) 대학은 안전하고, 생존성을 지닌 능동적인 인터네트워킹을 위한 프로토콜 개발에 힘쓰고 있다. DARPA의 지원 외에도 NSF(National Science Foundation)의 지원을 받고 있는 피치버그(Pittsburgh) 대학은 95년부터 98년까지 광역 패킷 네트워크의 생존성을 위한 네트워크 디자인과 트래픽 회복 절차에 대해 연구하였다. 즉, 피치버그 대학은 통신망 장애 시 특정한 QoS(Quality of Service)를 제공하기 위한 알고리즘 개발 및 네트워크 토폴로지 디자인과 네트워크 장애 후 트래픽을 회복하는 관리 알고리즘 연구에 중점을 두었다.

3. 이중 루트 CBT(Double Rooted Core Based Tree)

본 논문에서 제안하는 DRCBT는 루트가 2개인 CBT 형태를 가진다. CBT에서 코어를 택하는 기본적인 방식으로는 첫째, random method가 있고 둘째, geographic center method 셋째, weighted core방식이 있다. Random method는 그룹 중에서 임의의 노드를 선택하여 코어를 삼는 방식이고, geographic center method는 멀티캐스트 그룹 내에서 지리적으로 가장 중간에 있는 노드를 코어로 삼는 방법을 말하며, weighted 코어방식은 각 노드에 특정 알고리즘을 적용하여 가중치가 제일 높은 것을 코어로 삼는 방식이다.

위와 같은 방법들은 각 멤버에서 코어까지의 경로가 각각 최적화 되어 있지 않다는 단점이 있다. 하지만 그러한 단점을 heuristic한 방법으로 어느 정도 해결한 maximum weight method[12] 가 있으며 DRCBT에서 코어를 선택하는 방법은 기본적으로 maximum weight method를 DRCBT에 적합하게 응용한 형식을 적용한다.

3.1 코어 선택 방법

Maximum weight method[12]에서 각 멤버는 각각 모든 멀티캐스트 멤버에게 최적화 경로의 path를 잡는다. 이때 각 노드는 자신이 각 path에 포함된 횟수를 counting(weighting)하며 이때 가장 많은 path에 속하게 된 노드가 코어로 선택되게 된다. 만약 가장 높은 값이 하나 이상 나올 경우 그 중에서 임의로 하나를 선택하여 코어를 삼는다.

예를 들어, 그림 1에서 3번 노드와 5번 노드가 가장 높은 weight를 가지게 되었으며, 이 경우 임의의 한 노드인 3번 노드를 선택하여 코어로 삼은 예이다. DRCBT에서는 maximum weight값을 갖는 노드를 코어로 삼고 선택된 코어에 직접 연결된 노드 중에서 가장 weight 값이 높은 노드를 두 번째 코어로 삼는다. 그림 1의 경우에 3번 노드와 3번 노드에 연결된 것 중 가장 weight값이 높은 6번 노드가 DCBT의 코어로 선정되게 된다.

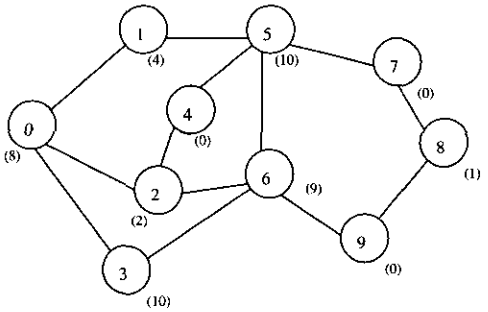

그림 1 maximum weight 방식을 사용한 코어 선택 방법

3.2 트리 구조

3.1 절에서 설명한 바와 같이 DRCBT에서는 2개의 코어를 선정하여 멀티캐스트 트리의 루트로 삼는다. 루트(코어)노드를 2개 갖는 DRCBT의 대략적인 구조는 그림2 와 같다.

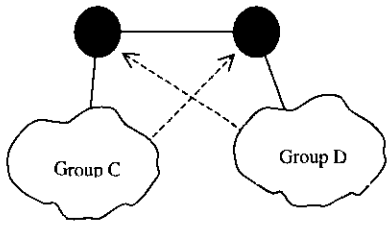

그림 2 DRCBT의 대략적인 구조도

그림 2와 같은 DRCBT의 구조에서 코어 A,B는 그룹 C와 그룹 D에 관한 멤버 정보를 공유한다. 하지만 멤버 정보는 공유하되 멀티캐스트 패킷 전송은 자신에게 속한 그룹에게만 하게 된다. 예를 들어 그룹 C쪽에서 멀티캐스트 패킷이 코어 A쪽으로 왔을 경우 코어 A는 자신이 받은 멀티캐스트 패킷을 코어 B에게 유니캐스트 해 주고 코어 A는 그룹 C에게 패킷을 멀티캐스트로 전송하고 코어 B는 그룹 D에게 패킷을 멀티캐스트로 전송한다. 또한 DRCBT의 가장 큰 특징으로서 그림 2 에서와 같이 코어 A는 그룹 C쪽으로의 멀티캐스트를 담당하지만 그룹 C는 자신의 코어를 B라고 인식하게

되는 구조를 갖는다는 점이다. 그룹 D도 마찬가지로 실제적으로 멀티캐스트 패킷은 코어 B로부터 받지만 그룹 D는 자신의 코어는 A로 인식하는 특징을 갖는다. 이와 같은 특징을 가능하게 해 주기 위하여 3.4절에서 DRCBT에서는 코어를 새롭게 바꿀 수 있는 컨트롤 메시지인 Change_core 메시지 타입을 새로 정의하였다.

3.3 코어의 고장 시 트리의 복구

단일 코어를 갖는 CBT에서는 코어 노드의 failure 시 처음부터 다시 코어를 선정해야만 한다. 따라서 코어의 failure시 전체 tree를 새로 생성해야 하는 작업이 필요하다. 하지만 DRCBT에서는 코어의 고장 시 새로운 전체 tree의 재구성을 필요로 하지 않고 트리를 복구한다. DRCBT에서 2개의 코어는 서로 그룹 멤버에 관한 정보를 공유하게 된다. 또한 코어끼리는 서로간의 주기적인 시그널 교환을 통하여 상대방의 고장을 즉시 감지할 수 있는 구조를 가지고 있다.

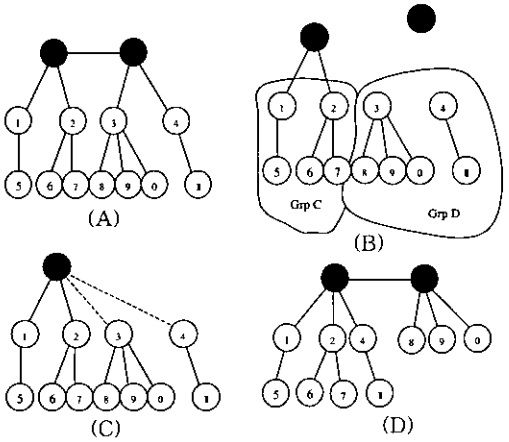

그림 3 코어 고장 시 복구 과정

그림 3의 (A)와 같은 정상적인 DRCBT구조에서 코어 중의 하나인 B노드가 고장이 났을 경우 B와 주기적으로 시그널을 교환하는 코어A는 즉시 B의 고장을 감지하게 되고 노드를 복구하게 된다. 또한 A에서 B의 고장을 감지할 동안의 짧은 지연 시간동안 일시적으로 그림 3의 (B)와 같은 상태가 되게 된다. 만약 그림 3의 (B)상태에 있는 동안 그룹C와 그룹 D에서 각각의 상황을 살펴보면 아래와 같다.

그룹 C의 입장에서 살펴보면 비록 코어 그룹C의 코어 B는 고장이 난 상태이지만 그룹 C에서 코어 B로 가기 위해서는 코어 A를 거쳐야 한다. 이때 코어 A는 노드 C와 D에서의 rejoin request를 기다리며 request를 받으면 join ack를 보내 C와 D로의 링크를 설정하게 된다. 즉 그룹 C의 입장에서는 자신의 그룹의 코어가 죽었지만 코어 A가 중간에서 코어의 역할을 대신 함으로 인해 트리 구조의 재 설정이 일어나지 않아도 된다. 즉 그림 3의 (C)와 같이 일시적으로 코어 A가 단일 코어 역할을 한다.

또한 그룹D의 입장에서 자신의 그룹의 코어인 A노드는 정상적으로 작동하지만 A로 패킷을 전송하는 과정에 거쳐야 하는 코어 노드인 B가 죽었을 경우, 먼저 노드 B의 child들은 echo_request를 통해 B의 고장을 확인하게 된다. 비록 노드 B는 코어이지만 B의 child들은 노드 B를 단지 코어 A로 가기 위한 parent로 인식하므로 코어 A를 향해 rejoin_request를 전송하게 된다. Rejoin_request를 받은 코어 A 또는 코어 A의 하위 노드인 그룹 C는 ack를 보내고 새로운 트리 링크를 형성하게 된다. 그림 3의 (C)에서는 노드 3, 4가 코어 A로 연결되었다고 가정했으나 노드 3, 4가 그룹 C에서 코어 A와에 다른 멤버의 자식이 될 수도 있다. 이렇게 노드 3, 4가 멀티캐스트 트리에 rejoin을 요청 했을 경우 만일 rejoin_request가 그룹 D의 하위 노드에 도착했다면 loop에 빠질 수 있다. 비록 한쪽 코어의 고장 시 단일 코어 상태가 되고 이 상태에서는 loop가 거의 발생 하지 않지만[8,13], 최악의 경우 만일 rejoin_request를 보낸 뒤 timeout이 발생하게 되면 노드 3, 4는 Flush메시지를 자신의 하위 노드에 보내어 그룹 D의 하위 구조는 깨어지게 된다. 하지만 이 경우에도 그룹 C의 구조는 살아 남게 된다.

결과적으로 코어 노드에 오류가 발생했을 경우 DRCBT에서는 각 그룹이 자신의 코어를 엇갈려 갖는 구조를 갖기 때문에 고장 난 B노드를 자신의 코어라고 생각하는 그룹 (C)는 중간에서 실질적으로 코어 역할을 하는 A노드로 인하여 자신의 코어 노드의 고장을 인식하지 못하기 때문에 그룹(트리 구조)의 해체가 이루어지지 않게 된다. 또한 B가 죽었을 경우 실질적으로 자신의 그룹에 코어 역할을 해 주었던 코어 노드가 고장났음에도 불구하고 그룹 (D)는 단순히 하나의 upstream링크가 고장 났다고 여기므로 그룹(트리 구조)의 해체가 이루어 지지 않게 된다.

위의 그림3에서 보듯이 코어 노드 B가 죽었을 경우 그림 3의 (C)와 같이 일시적으로 하나의 단일 코어가 멀티캐스트 트리를 유지하는 형태를 갖게 되며 이 경우 단일 코어의 자식 중에서 가장 많은 노드를 가지고 있는 자식을 새로운 코어로 삼는다. 그림 3의 (D)와 같이 새로운 노드를 코어로 삼은 후에 코어 노드 A는 자신의 하위 노드로 Change_core 메시지를 멀티캐스트 함으로

써 그룹 C의 코어를 3번 노드로 인식하게 하며 노드 3도 마찬가지로 자신의 하위 노드에 Change_core 메시지를 멀티캐스트 함으로써 코어를 A로 인식하게 만든다.

3.4 이중 루트 CBT Common Control Packet Header

그림 4에서 보듯이 DRCBT의 control packet header는 CBT와 같다. 단지 기존의 8가지 type외에 type 9로 Change_core Type을 정의 한다.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Vers	Type	Addr len	Check sum

그림 4 DRCBT common Control Packet Header

4. 성능 분석

이 장에서는 코어의 고장 시 멀티캐스트 트리의 복구 시간을 DRCBT와 CBT, MCBT 상황하에서 비교 분석하였다.

그림 5의 네트워크 구조에서 maximum weight method를 적용하였을 때 생성되는 DRCBT와 CBT의 논리적 tree 구조와 그림 5에서 (4),(2)번 노드가 코어 backbone에 연결된 코어 노드라고 가정했을 경우 생성되는 논리적인 MCBT의 구조는 그림 6과 같다.

멀티캐스트 트리를 구성하는 여러 변수의 값을 표1에서와 같이 나타내었을 때, 코어가 고장이 났을 경우 새로운 코어를 선정하여 새로 멀티캐스트 트리를 형성하는 시간은 각 노드에서 코어까지의 hop count수의 총합에 비례한다고 가정했을 경우, DRCBT, CBT,

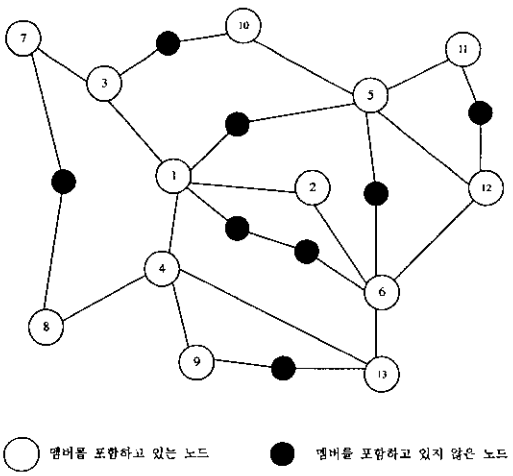

그림 5 멀티캐스트 트리가 형성되기 전의 초기 구조

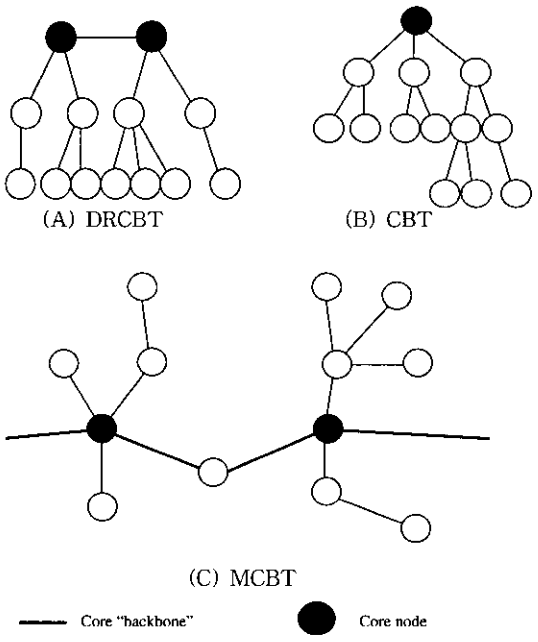

그림 6 여러 가지 코어 기반 트리

표 1 멀티캐스트 트리를 구성하는 여러 변수

Notation	Description
N	전체 노드의 수
a	하나의 부모 노드가 평균적으로 갖는 자식 노드의 수
iT	새로 복구되는 멀티캐스트 트리의 깊이 i에서 코어까지의 평균 hop count
c	MCBT에서 멀티캐스트 그룹에 연결된 코어의 수

MCBT 구조에서 코어가 고장이 났을 경우 각각의 경우에 복구하는 시간을 살펴보면 아래와 같다.

가) DRCBT의 경우

DRCBT구조에서 코어가 고장이 났을 경우 3장에서 설명되었듯이 고장이 난 노드의 자식 노드만 살아있는 코어에 연결시켜 주면 멀티캐스트 트리가 복구된다. 이때 고장이 난 노드의 자식 노드 수는 a개가 되고 살아있는 코어까지의 트리 깊이 차이가 1이므로 복구하는데 걸리는 총 시간은 복구할 노드의 총 hop count 수 이므로 DRCBT에서 코어의 고장 시 복구에 걸리는 시간 Thop(DRCBT)는 다음과 같다

Thop(DRCBT) = $a \times iT = a \times 1T = aT$

나) CBT의 경우

CBT 구조에서 코어가 고장이 났을 경우 새로운 코어가 선택되어야 하고 그 새로운 코어를 중심으로 전체 노드의 멀티캐스트 트리 구조를 재 구성해야 한다. 이 경우 전체 노드의 개수는 n이고 새로 구성되어야 할 트리의 총 깊이는 $\lfloor \log_a n \rfloor$이 된다. 새로 구성될 트리의 깊이 1에서 복구할 노드의 총 hop count 수는 aT이고 깊이 2에서 복구할 노드의 총 hop count 수는 $2a^2T$가 된다. 식으로 표현하면 복구할 노드의 총 hop count 수 Thop(CBT)는 다음과 같다.

$$\text{Thop(CBT)} = \sum_{i=1}^{\lfloor \log_a n \rfloor} ia^iT = aT\left(\frac{1-n-n\log_a n + an\log_a n)}{(1-a)^2}\right)$$

다)MCBT의 경우

MCBT 구조에서 코어 backbone에 코어가 c개가 있을 경우 각각의 코어는 노드를 $\frac{n}{c}$개만큼 담당한다고 가정했을 때 코어 중의 하나가 고장 났을 때 복구해야 할 노드의 개수는 $\frac{n}{c}$개가 되고 이때의 복구할 노드의 총 hop count 수 Thop(MCBT)는 다음과 같다.

$$\text{Thop(MCBT)} = \sum_{i=1}^{\lfloor \log_a \frac{n}{c} \rfloor} ia^iT = aT\left(\frac{1-\frac{n}{c}-\frac{n}{c}\log_a \frac{n}{c}+a\frac{n}{c}\log_a \frac{n}{c})}{(1-a)^2}\right)$$

위에서 나온 결과값을 정리하면 표 2와 같다.

표 2 여러 가지 코어 기반 트리에서의 트리 복구 시간

여러 가지 코어 기반 트리	Total hop count
DRCBT	aT
CBT	$aT\left(\frac{1-n-n\log_a n+an\log_a n)}{(1-a)^2}\right)$
MCBT	$aT\left(\frac{1-\frac{n}{c}-\frac{n}{c}\log_a \frac{n}{c}+a\frac{n}{c}\log_a \frac{n}{c})}{(1-a)^2}\right)$

표 1에서 정의한 여러 변수 값을 가지고 특정 값을 변화 시켜 보았을 때, 각각의 코어 기반 멀티캐스트 트리에서 걸리는 복구 시간을 비교 분석 해 보았다.

다른 변수 값을 고정 시켜 놓고 총 노드 수(n)를 증가 시켜 보았을 때, 각각의 코어 기반 멀티캐스트 트리에서 걸리는 복구 시간은 그림 7과 같다. 그림 7에서 보듯이 CBT와 MCBT의 경우 노드의 수가 증가 할수록 복구 시간이 증가 하였으나 DRCBT의 경우 노드의 수에 상관없이 복구 시간은 항상 일정한 값을 가졌다.

다른 변수 값을 고정 시켜 놓고 T(평균 hop count

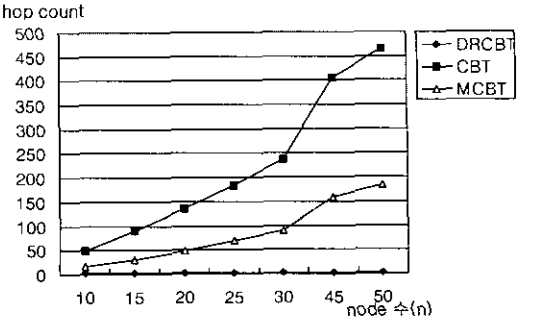

그림 7 노드의 수에 따른 트리 복구 시간

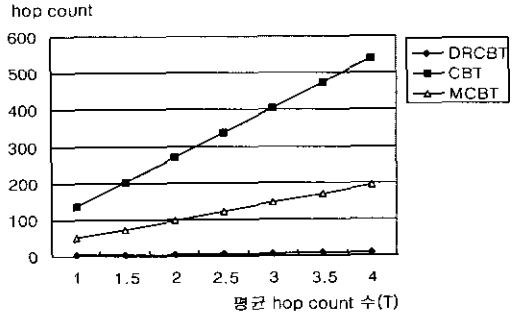

그림 8 평균 hop count수에 따른 트리 복구 시간

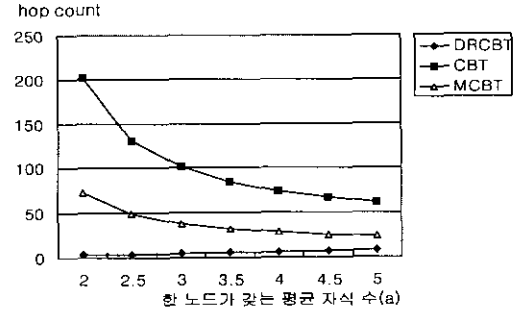

그림 9 평균 자식 수에 따른 트리 복구 시간

수)값을 증가 시켜 보았을 때의 복구 시간은 그림 8과 같다. 그림 8의 결과에서 보듯이 CBT와 MCBT의 경우에는 T값이 증가 할수록 복구 시간이 증가 하였으나 DRCBT에서의 복구 시간은 거의 증가 하지 않았다.

다른 변수 값을 고정 시켜 놓고 a(평균 자식 수)값을 증가 시켜 보았을 때의 복구 시간은 그림 9와 같다. 그림 9의 결과에서 보듯이 a값이 증가할수록 DRCBT의 복구 시간은 약간 증가 하였으며 CBT와 MCBT의 트

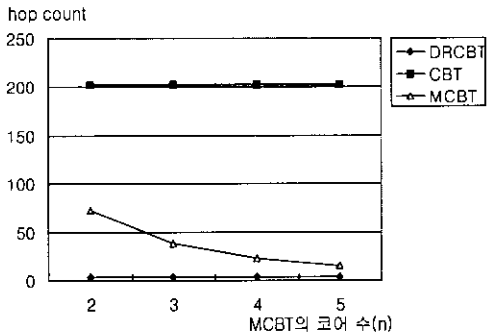

그림 10 MCBT의 코어 수에 따른 트리 복구 시간

리 복구 시간은 줄어 들었으며 점차로 DRCBT의 값에 접근 해 나가는 결과를 보였다.

MCBT에서 c(코어의 개수)를 증가 시키고 다른 변수 값을 고정 시켜 놓았을 때의 그림 10과 같으며 그림 10에서 보듯이 코어의 개수가 점차로 증가 함에 따라 MCBT에서 복구 시 걸리는 시간이 점차로 줄어들고 있음을 알 수 있다.

5. 결 론

네트워크 인프라의 발전 정도가 그 나라 국력의 지표로 여겨지고 있을 만큼 네트워크의 중요성은 크게 부각되고 있다. 네트워크 중요성의 커지는 만큼 네트워크 관리 및 생존성 문제는 필수적으로 고려되어야 할 사항이다. 본 논문에서 제시한 인터넷상에서의 멀티캐스트 서비스에 대한 안정성 문제 또한 같은 맥락으로 생각할 수 있다.

본 논문은 코어의 오류에 뛰어난 생존력을 보이는 이중 루트 코어를 갖는 DRCBT 구조를 제안하고 코어 고장 시 복구 시간을 비교하였다. CBT 하에서는 코어의 고장 시 전체적인 트리 재 설정이 이루어 지나, DRCBT는 이중 코어 루트 구조에서 하위 그룹 멤버가 서로의 코어를 엇갈리게 설정하는 특수한 구조를 갖기 때문에 코어의 고장 시에도 각 코어에 달려 있는 멀티캐스트 그룹의 트리 구조는 그대로 유지 되며, 단지 고장 난 트리의 자식 노드만 새로운 코어로 연결 해 줌으로써 새로운 멀티캐스트 트리를 생성 할 수 있었다. 따라서 DRCBT 구조는 총 노드의 수에 상관 없이 일정한 복구 시간을 가질 수 있었으며 상대적으로 CBT나 MCBT에 비해 월등히 빠른 복구 성능을 나타낼 수 있었다. DRCBT 는 코어의 고장 시 타격을 받을 수 있는 멀티캐스트 그룹 또는 코어의 생존성을 보장하고자 하는 여러 응용 어플리케이션에 사용 될 수 있을 것이다.

그러나 본 논문에서 제안한 DRCBT방법은 코어 외의 노드나 링크의 고장 시는 기존 CBT 방식과 같은 복구 방식을 사용하기 때문에 일반 노드나 링크의 고장 시는 성능 향상을 기대할 수 없다. 또한, 고장이 일어나지 않았을 때에, 코어 간의 정보를 공유하고 주기적으로 생존 여부를 확인하여야 하는 부가적인 오버헤드가 요구된다. 그러나 이러한 오버헤드는 본 논문에서 제시한 결과에서 볼 수 있듯이 코어의 오류 시 크게 트리 복구 시간을 크게 줄일 수 있다는 점에서 간과할 수 있는 정도라 하겠다.

참 고 문 헌

[1] A Technical Report on Enhanced Analysis of FCC-Reportable Service Outage Data, T1A1.2 Working Grout on Network Survivability Performance, Report No.42, August 1995.

[2] A Technical Report on Reliability and Survivability Aspects of the Interactions between the Internet and the Public Telecommunications Network, T1A1.2 Work Group on Network Survivability Performance, Report No.55 October 1998.

[3] A. Srikitja, D. Tipper and D. Medhi, "On Providing Survivable Services in the Next Generation Internet," to appear Proceedings of IEEE Military Communications Conference (Milcom '99), Atlantic, NJ, Oct., 1999.

[4] G. Rogers, D. Medhi, W.-J. Hsin, S. Muppala and D. Tipper, "Performance Analysis of Multicast and Priority-Based Routing Under a Failure in Differentiated Services Internet," to appear Proceedings of IEEE Military Communications Conference (Milcom '99) , Atlantic, NJ, Oct., 1999.

[5] Jhyda Lin, Rudy-Shiung Chang. A comparison of the Internet Multicast routing protocols. Computer Communications 22 (1999)144-155.

[6] S. Deering, C. Partridge, D. Waitzman, Distance vector multicast routing protocol RFC 1075, November 1998.

[7] B. Cain, S. Deering, A. Thyagarajan, Internet group management protocol version 3, Internet draft, August 1995

[8] A. Ballardie, Core Based Trees (CBT version 2)Multicast Routing: Protocol Specification IETF RFC2189, Sept 1997

[9] Won Tae KIM and YongJin Park. DCBT:An Efficient Multicast Architecture for Wide Scale and Large Group Multicast Communications. IEICE Trans. INF. & SYST. Vol. E82-D, NO.4 April 1999

[10] S. Deering, D. Eastrin, D. Fariancci. Protocol

independent multicast spare mode <PIM-SM>:protocol specification, RFC2117, june 1997
[11] S. Deering, D. Eastrin, D. Farinacci. Protocol Indenpent Multicast Version 2 Dense Mode Specification IETF Draft, Aug. 1998
[12] Hwa-Chun Lin and Shou-Chun Lai. Core placement for core based tree multicast routing architecture 1998 IEEE
[13] C.Shields, Ordered core bases trees, Masters thesis, University of California, Santa Cruz, California, June 1996

오 윤 근
1999년 2월 한동대학교 전산전자공학부 공학사. 1999년 2월 ~ 현재 한국 정보통신 대학원 대학교(ICU) 석사과정. 1999년 3월 ~ 2000년 3월 한국통신 제 2 연구소 연구기획실 위촉연구원

조 평 동
1980년 연세대학교 전자공학과 졸업(공학사). 1995년 충남대학원 컴퓨터과학과 졸업(이학석사). 1980년 ~ 1997년 ISDN, 지능망, 통신처리시스템 개발. 1998년 ~ 현재 한국전자통신연구원 표준연구센터 책임연구원. 관심분야는 전기통신 기술기준, 지능망, Network architecture

김 남 훈
1996년 2월 숭실대학교 전자계산학과 공학사. 1998년 2월 숭실대학교 전자계산학과 공학석사. 1998년 3월 ~ 현재 한국 정보통신 대학원 대학교(ICU) 박사과정.

이 영 희
1976년 2월 서울대학교 공과대학 공업교육학과 공학사. 1980년 2월 서울대학교 공과대학 공업교육학과 공학석사. 1981년 9월 ~ 1984년 6월 불란서 UTC (Universite de Technologie de Compiegne), 전산학 박사. 1984년 8월 ~ 1997년 12월 한국전자통신연구원, 정보통신표준연구센터 센터장. 1986년 7월 ~ 1987년 11월 IBM T. J. Watson 연구소 초빙과학자. 1998년 1월 ~ 현재 한국 정보통신 대학원 대학교(ICU) 교수.