완전 연결된 네트워크에서 MDST와 MST 목적을 갖는 오버레이 멀티캐스트 트리구현 알고리즘

조명래 (전남도립대학 U-정보산업과 교수)*

국 문 요 약

인터넷 기반 멀티캐스트는 일대다 또는 다대다 통신을 위한 차세대 중요한 서비스로 주목받고 있다. 멀티캐스트는 네트워크 또는 애플리케이션 레벨에서 서비스할 수 있다. IP 멀티캐스트는 소스노드에서 라우터로 데이터그램을 보내면 라우터가 이를 복제하여 수신노드들에게 전달해 주는 네트워크레벨 서비스로 네트워크 자원을 효율적 사용할 수 있다. 그러나 네트워크에 IP 멀티캐스트 라우터가 설치되어야 하는 등 여러 문제로 인해 널리 사용되지 못하고 있다. 따라서 대안으로 애플리케이션 레벨에서의 오버레이 멀티캐스트가 주목 받고 있다. 오버레이 멀티캐스트는 종단 호스트가 라우터 처럼 동작하는 것으로 비록 IP 멀티캐스트에 비해서 링크 사용율과 지연값이 높아질 수 있지만, IP 멀티캐스트의 현실적인 적용의 어려움을 해결할 수 있는 장점을 가지고 있다.

본 연구는 완전 연결된 네트워크에서 탐욕 알고리즘을 이용하여 MDST(소스 노드에서 다른 모든 각각의 노드까지 최단 경로를 갖는 신장트리)와 MST(네트워크 상의 모든 링크에 주어진 가중치의 합이 최소가 되는 신장트리)를 목적으로 하는 오베레이 멀티캐스트 트리를 구현하는 알고리즘을 제시하고, 실험에 의해 MDST 와 MST 를 비교 분석 하고자 한다.

핵심주제어: 오버레이 멀티캐스트, 최소지름신장트리, 최소신장트리, 탐욕 알고리즘

Ⅰ. 서 론

최근 서로 다른 물리적 네트워크를 상호 연결하여 하나의 통합된 단위로 동작 시킬 수 있는 인터네트워킹 기술의 발전은 개방형상호연결시스템 기반에서 인터넷 기술의 진보를 가져다 주고 있다. 특히 일반적으로 TCP/IP 라고 불리는 인터넷 프로토콜군(TCP/IP Internet Protocol Suite)기술은 상호 연결된 어떤 종류의 네트워크 간 정보교환을 가능하게 하였으며, 다양한 종류의 애플리케이션 및 네트워크 레벨 인터넷 서비스를 가능하게 하였다. 이러한 인터넷 기술의 진전과 함께 HD 급의 H.264,

* 제1저자, 교신저자, 전남도립대학 U-정보산업과 교수, mrcho@dorip.ac.kr

MPEG2, AAC, AC-3 와 같은 영상 및 음성 압축기술 및 MPEG2-TS 다중화 방식 등의 멀티미디어 가공 기술의 진보 그리고 FTTH 또는 HFC 와 같은 가입자 망의 광대역화 구성은 AON, PON, Home RUN 등의 전송 기술과 접목하여 100Mbps 급 이상의 멀티미디어 서비스를 가능하게 하였으며, 이를 통해 20Mbps 이상의 높은 대역폭을 요구하는 IPTV 등의 서비스를 가능하게 하였다.

높은 대역폭을 요구하는 대표적인 인터넷 기반 서비스인 IPTV 는 인터넷이라는 통신의 영역과 TV 라는 방송의 영역이 공존하는 분야로서 기존의 통신과 방송에 대한 수직적 규제의 틀을 적용하기 어려운 융합서비스이나, 관련 규제 정책과 표준화가 활발히 논의되고 있는 차세대 유망한 멀티미디어 서비스로 주목 받고 있다. IPTV 는 IP(비신뢰적, 비연결형, 최선전달 전송 매커니즘) + TV 의 합성어로, '일정 수준의 QoS, QoE, 보안, 양방향성 및 신뢰성을 제공하도록 관리된 IP 기반의 네트워크에서 전송되는 텔레비전, 비디오, 택스트, 그래픽스, 데이터와 같은 멀티미디어 서비스 [ITU-T IPTV Focus Group, 2006/11]' 라 할 수 있으며, 제공되는 서비스는 유니캐스트를 통한 일대일(one-to-one) VOD 서비스와 멀티캐스트를 통한 일대다(one-to-many) 또는 다대다(many to many) 방송 서비스로 크게 나눌 수 있다.

지금까지 대부분의 인터넷 응용서비스들은 일대일 유니캐스트 전송을 기반으로 하고 있으나, IPTV 를 비롯한 비디오 회의 그리고 원격 교육 서비스와 같은 일대다 또는 다대다 서비스들은 특정한 그룹을 대상으로 통신을 지원해야 하기 때문에 IP 멀티캐스트가 효율적인 전송 메커니즘으로 제안되고 있다.[1, 2]

IP 멀티캐스트는 호스트 컴퓨터의 부분집합(멀티캐스트 그룹)에 정보를 전송하는 인터넷 추상으로, 인터넷에 걸쳐있는 임의의 물리적 네트워크에 흩어져 있는 부분집합을 허용하도록 개념을 일반화 하였다. IP 멀티캐스트의 특징은 (1)각 멀티캐스트 그룹은 유일한 D 클래스 주소(224.0.0.0 ~ 239.255.255.255)를 갖는다. (2)호스트는 아무 때나 IP 멀티캐스트 그룹에 가입하거나 탈퇴할 수 있다. 더구나 호스트는 몇 개의 멀티캐스트 그룹이든 구성원이 될 수 있다. (3) IP 멀티캐스트 그룹의 구성원들이 여러 물리적 네트워크에 연결되어 있을 수 있기 때문에, IP 멀티캐스트를 포워딩하기 위한 특별한 라우터(IP 멀티캐스트 라우터)가 필요하다. (4) IP 멀티캐스트는 다른 IP 데이터그램 전달과 같이 최선의 전달 체계(best-effort delivery semantics)를 사용한다. (5)임의의 호스트는 어떤 멀티캐스트 그룹이든 데이터그램을 전송할 수 있으며, 그룹 가입상태는 호스트가 그 그룹으로 보내진 데이터그램을 수신할지 여부만을 결정한다. 이러한 특징을 갖는 IP 멀티캐스 h 트는 멀티캐스트 라우터와 호스트 간의 그룹 가입 상태 정보를 교환하기 위해 IGMP(Internet Group Management Protocol)을 이용하고 있다. IGMP 는 TCP/IP 의 표준이며, IP 멀티캐스트를 수신해야 하는 모든 머신에서 필수적으로 필요하다. 그러나 IP 멀티캐스트는 지금까지 여러 가지 IP 멀티캐스트 포워딩 방법(ex. RPF : Reverse Path Forwarding, TRPF : Truncated Reverse Path Forwarding, TRPB : Truncated Reverse Path Broadcasting)과 라우팅 정보를 전파하기 위한 프로토콜(ex. DVMRP : Distance Vector Multicast Routing Protocol, CBT : Core Based Trees, PIM : Protocol Independent Multicast, MOSPF : Multicast extension to OSPF)이 제안되었지만 단일 표준은 아직 합의되지

못하고 있다.

IP 멀티캐스팅 전송방식은 서비스 제공 노드가 네트워크에 위치한 IP 멀티캐스트 라우터로 데이터그램을 보내면, 라우터가 데이터그램을 분석하고, 복제하여 동일 멀티캐스트 그룹내 수신 노드들의 주소로 데이터그램을 전송하고, 다시 주변의 IP 멀티캐스트 라우터로 재전송하는 형태로 확대해 나가 수신자 그룹에게 데이터그램을 제공하는 전송 방식이라 할 수 있겠다. 따라서 IP 멀티캐스트는 하나의 송신 노드에서 전송된 데이터그램이 네트워크의 각 링크에서 한번씩만 전송됨으로써 효율적인 망 자원의 사용을 가능하게 한다. 그러나 (1) DVMRP 와 같은 멀티캐스트 프로토콜을 지원하는 IP 멀티캐스트 라우터가 모든 네트워크에 설치되어야 한다.(2008 당시 IP 멀티캐스트 라우터는 30% 이하에 불과함.[19]) (2) IP 멀티캐스트는 네트워크 계층에서 이루어지므로 에러제어, 흐름제어, 혼잡제어, 멤버 관리 그리고 보안과 같은 상위계층에 속한 기능을 지원하기 어렵다. (3)라우터에서 데이터그램이 복제되므로 이에 대한 서비스 제공자의 과금 정책 모델 구성이 힘들다. 등의 문제로 IP 멀티캐스트는 많은 연구자와 네트워크 연구단체가 관심을 가지고 있지만 아직 인터넷에 널리 사용되지 못하고 있다.[3]

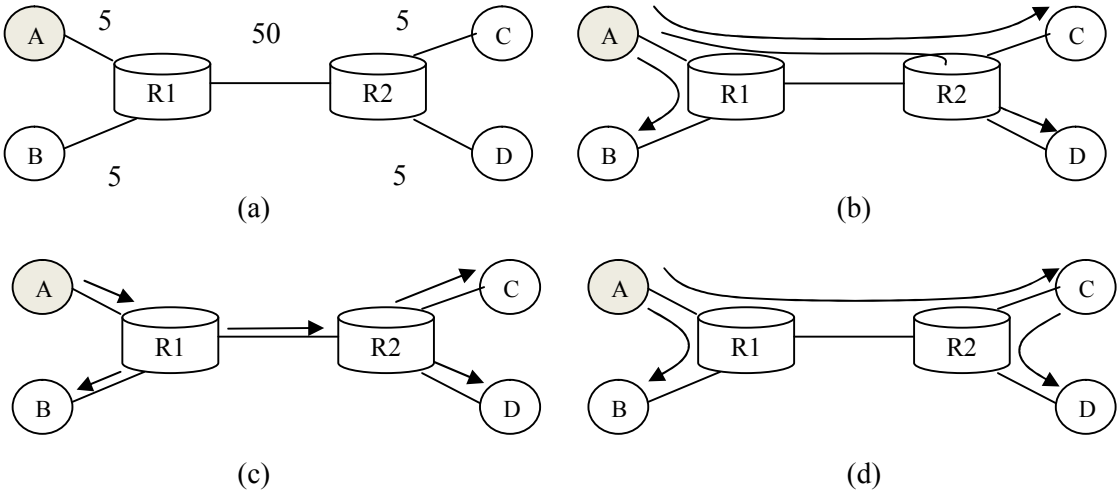
그 결과 최근 IPTV, NVOD(Near VOD), 또는 비디오 회의 같은 멀티캐스트 기반의 애플리케이션을 지원하기 위한 방법으로 오버레이 네트워크 상에서의 종단 시스템에 의한 멀티캐스트 즉, 오버레이 멀티캐스트가 대안으로 관심 받고 있다.[4, 5, 6, 7]

오버레이 멀티캐스트 전송방식은 IPTV 영역에서 멀티캐스트 표준화를 위해 ETRI를 중심으로 우리나라가 제안하였으며, ITU-T FG IPTV의 WG4에 반영되고 있다. 현재 ITU-T Q/1/17 및 ISO/IEC JTC1/SC6 WG7의 국제표준으로 채택되어 있으며, 비표준 오버레이 멀티캐스트 기술이 일부 기업(ex. MCast Box)을 통해 사용되고 있다.[20]

여기서 오버레이 네트워크란 물리적인 네트워크 위에 구현된 논리적인 네트워크로 생각할 수 있다. 오버레이 멀티캐스트는 종단 시스템 멀티캐스트(end system multicast)라고도 불리며, 특정 멀티캐스트 기반의 애플리케이션에 참여하려는 종단 호스트들이 스스로 멀티캐스트 트리를 만든다는 개념에 바탕을 두고 있다. 따라서 오버레이 멀티캐스트는 애플리케이션 레벨 인터넷 서비스로, 네트워크 레벨 인터넷 서비스를 하는 IP 멀티캐스트와 크게 구별할 있다. 오버레이 멀티캐스트의 동작 원리는 라우터가 아닌 호스트가 오버레이에 참여하여 라우터 처럼 동작하는 것이다. 또한 이러한 호스트들은 애플리케이션 프로그램에서 구현하기 쉽도록 IP 터널보다는 UDP 터널을 사용한다. 따라서 이러한 방식으로 접근하면, 인터넷 상의 특정 호스트는 다른 모든 호스트에 메시지를 보낼 수 있으므로, 기반 네트워크는 완전히 연결된 그래프로 간주할 수 있다. 따라서 오버레이 멀티캐스트는 인터넷을 완전히 연결된 그래프로 가정하고, 소스 노드에서 모든 구성원들에게 도달 가능한 효율적인 오버레이 멀티캐스트 트리를 찾아 내는 것이 매우 중요하다.

<그림 1>은 물리적인 토폴로지 위에서 여러 가지 멀티캐스트 트리를 구현하는 개념도를 보여주고 있다. <그림 1>에서 (a)는 물리적인 토폴로지 구조로 R1, R2 은 라우터를 의미하며, A 는 소스 노드 그리고 B, C, D 는 각각 종단 호스트들이다. 각 노드

사이의 링크 위의 숫자는 왕복 지연 시간 등의 가중치를 의미한다. (b)는 소스 노드 A 가 멀티캐스트 메시지를 다른 호스트들에게 유니캐스트로 메시지로 보내는 경우를 보여주고 있다. 이때 동일한 메시지가 A-R1 간에는 세 개, R1-R2 간에는 두 개 전달되고 있음으로, 좋지 않는 방법임을 쉽게 알 수 있다. (c)는 IP 멀티캐스트 트리를 보여 주고 있다. 이때 만약 라우터 R1 과 R2 가 IP 멀티캐스트 라우터라면, 단 한번의 데이터가 전송되어 링크 사용이 효율적임을 알 수 있다. 그러나 앞에서 살펴본 바와 같이 현실적으로 적용하는데 한계를 가지고 있다. (d)는 라우터에 추가 기능을 구현하지 않고 종단 시스템에서 멀티 캐스트하는 오버레이 멀티캐스트 트리를 보여주고 있다. 그림에서 오버레이 멀티캐스트 방식은 유니캐스트에 비해 종단 노드를 거치면서 정보가 전송되므로 지연값이 높아질 수 있지만 링크 사용률은 좋으며, IP 멀티캐스트에 비해서 링크 사용율과 지연값이 높아질 수 있지만, IP 멀티캐스트의 현실적인 적용의 어려움을 해결할 수 있는 장점을 가지고 있다.



〈그림 1〉
(a) 물리적인 토폴로지, (b) 단순히 유니캐스트 메시지를 보내는 경우,
(c)라우터의 네트워크 계층에서 멀티캐스트 트리를 구성한 경우,
(d)종단 호스트의 애플리캐이션 계층에서 오버레이 멀티캐스트 트리를 구성한 경우

오버레이 멀티케스트 트리는 애플리케이션의 효율적인 서비스를 위해서는 소스 노드에서 다른 모든 각각의 노드까지 소요되는 서비스의 지연시간을 최소화 하는 목적[8, 9, 10, 11] 또는 네트워크 상의 모든 링크에 소요되는 서비스 지연시간의 합을 최소화하는 목적[7, 12] 등으로 구현되어야 할 것이다. 이러한 목적은 텍스트 기반 데이터 통신의 경우 QoS 설계 시 셀 손실을 최소화 하는 것이 매우 중요한 기준이 되는 반면, 영상 기반 멀티미디어 통신 특히 실시간 멀티캐스트 통신의 경우 데이터그램 지연의 최소화가 신뢰성 확보의 중요한 기준이 되기 때문이다. 특히 오버레이 멀티캐스트 방식은 유니캐스트와 IP 멀티캐스트에 비해 지연값이 상대적으로 높기 때문

에 서비스 지연시간을 최소화 하는 목적으로 망을 설계해야 할 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 다루는 문제를 설정하고, 3장에서는 문제를 최소지름신장트리(MDST : Minimum Diameter Spanning Tree)와 최소신장트리(MST : Minimum Spanning Tree)를 목적으로 모형화 하고 그 해결방법을 제시하며, 서로 비교 한다. 4장에서는 문제 해결의 결과를 실험을 통해 보여 주며, 5장은 결론으로 한다.

Ⅱ. 문제설정

본 연구에서는 물리적인 인터넷 네트워크에서 멀티캐스팅을 위한 효율적인 오버레이 멀티캐스트 트리를 구성하는 문제를 각각 두 가지 목적에 따라 해결 방법을 제안한다. 첫째는 멀티케스트 서비스를 제공하는 소스 노드에서 서비스를 제공 받는 각각의 모든 노드까지 소요되는 지연시간을 최소화 시키는 목적으로 MDST를 구하는 방법과, 둘째는 오버레이 멀티캐스트상의 모든 링크에 소요되는 서비스 지연시간의 합을 최소화 시키는 목적으로 MST를 구하는 방법을 다룬다.

오버레이 멀티캐스트 트리를 만드는 개념적인 절차는 첫 번째 단계로 서비스 대상이 되는 모든 호스트들을 인터넷 상에서 완전 연결된 메시 네트워크로 만든다.[13]

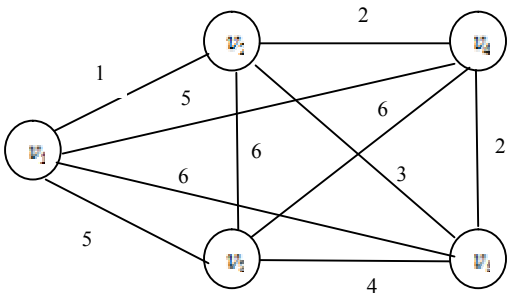
이때, 이 메시 네트워크에서 라우터 정보는 고려하지 않으며, 단지 호스트들 만이 정보의 대상이 된다. 호스트들은 각각 직접 연결된 이웃 호스트에 대한 왕복지연시간(roundtrip latency)을 측정하여 서로 교환한다. 이웃 호스트들로부터 상태 정보를 받으면, 자신이 가지고 있는 정보를 업데이트해서 다른 이웃 호스트로 전달하게 된다. 결국은 거리벡터 라우팅 프로토콜(DVMRP)처럼 메시 구조 전체를 통해 정보가 전파되어 완전 연결된 네트워크가 된다. 두 번째 단계는 이 메시 구조 내에서 해당 애플리케이션을 지원할 수 있는 효율적인 오버레이 멀티캐스트 트리를 목적에 맞게 신장트리로 구성하는 것이다.

따라서 본 연구에서 다루는 네트워크는 멀티캐스트 대상이 되는 소스 노드를 포함한 모든 노드의 쌍이 인접해 있는 완전 그래프(complete graph)로 두고, 그래프 상의 모든 링크에는 왕복지연시간이 주어져 있다고 본다. 여기서 가중치는 왕복지연시간이며, 편의상 거리로 부르고자 한다.

본 논문은 문제 해결을 위해 그래프 알고리즘을 적용하였으며, 문제 해결을 위해 <그림 2>를 예제 네트워크로 두었으며, 사용되는 기호는 다음과 같이 정의한다.

A	네트워크상의 모든 링크의 집합
$D = \| d_i \|$	d_i 행렬
$D^* = \| d^*_{ij} \|$	d^*_{ij} 행렬
d_{ij}	v_i와 v_j사이에 연결된 링크의 거리
d^*_{ij}	v_i에서 v_j까지의 최단 경로가 알려졌을 때 최단 거리

d_i	v_1에서 v_i까지의 최단 경로가 알려졌을 때 최단 거리
$G = (V, A)$	집합 A와 V로 구성된 네트워크 또는 그래프
$st(A)$	신장트리에서의 모든 링크의 집합. $st(A) \in A$
V	네트워크상의 모든 노드의 집합
v_1	시작 또는 소스 노드
v_i	i번째 노드 $i = 1, 2, ..., k$
v_k	마지막 노드
(i, j)	노드 v_i와 v_j에 연결된 링크



〈그림 2〉 완전 연결된 예제 네트워크

Ⅲ. 모형화

3.1 MDST

MDST는 소스 노드에서 다른 모든 각각의 노드까지 최단 경로를 갖는 신장 트리를 구현하는 것으로 목적은 MIN $\sum_{j=2}^{k} d_{1j}$ 가 된다.

모든 링크의 가중치가 음이 아닌 경우에 가중 방향 그래프 $G = (V, A)$ 에서 단일 출발점 최단 경로 문제는 1959년 다익스트라(*Dijkstra*)에 의해 제안된 알고리즘이 벨만-포드 알고리즘 또는 가보우의 스케일링 알고리즘 등 보다 효율적인 방법으로 알려져 있다. 다익스트라 알고리즘은 비록 우선순위 큐를 언급하고 있지 않지만, 최단 경로 문제를 해결하는데 많이 응용되고 있다. 따라서 본 연구에서는 다익스트라 기법을 응용한 탐욕 알고리즘(greedy algorithm)을 적용하고자 한다. 다익스트라 알고리즘은 네크워트 $G = (V, A)$가 주어졌을 때, 출발점 v_1으로부터 최종 최단 경로 가중치가 이미 결정된 노드들의 집합 D를 유지 관리 한다. 그리고 $V - D$에서 가장 가까운 노드를 선택하여 D에 더하기 때문에 탐욕 알고리즘이라 할 수 있다. 탐욕 알고리즘은 각 단계에서 결정이 필요한 순간 마다, 그 순간에 최상으로 보이는 것을 선택하는 발견적 기법(heuristic method)으로, 이 선택이 마지막 단계에서는 최적의 해답을 유도할 것이라는 기대를 가지지만 최적해를 찾는 것을 보장해주지는 못한다. 그러나

MDST나 MST 문제 등은 특정한 탐욕 기법을 사용하면 효율적인 해를 구할 수 있다.

완전 연결된 $G = (V, A)$가 주어졌을 때, 노드 v_i와 v_j 사이의 거리를 d_{ij} 라 하고, 노드 v_i까지의 최단 경로가 알려졌을 때 최단 거리를 d_i 라고 둔다. 이때, 노드 v_j까지의 최단 경로를 찾으려면 노드 v_j를 도착점으로 하는 모든 링크 (i, j)를 찾아 다음 식으로 구할 수 있다.

$$d_j = \min_i\{d_i + d_{ij}\} \qquad (1)$$

이때, 소스 노드 v_1을 시작점으로 하여 다른 모든 노드 ($v_j \in V$, $j=2,3,\cdots,k$) 까지의 최단 경로는 식(1)을 이용하여 다음과 같은 절차로 구할 수 있다.

단계 1] 초기화

$d_1 \leftarrow 0$, $D^* \leftarrow \emptyset$, $D \leftarrow \emptyset$ 로 둔다

단계 2] while ($j <= k$)

$j \leftarrow j + 1$ (단, j는 v_1에서 거리가 가까운 노드 순으로 갱신한다)

for ($i=1; i<= j-1 ; i++$) {

$d_j \leftarrow \min_i\{d_i + d_{ij}\}$

단계 2-1] 부분 최적해 유지 관리

v_1에서 v_j까지의 최단 거리 d_j와 이를 만족하는 d_{ij}를 d^*_{ij}로 두어 최종해를 구할 때까지 이 값을 유지 관리한다.

$D^* \leftarrow D^* \cup d^*_{ij}$, $D \leftarrow D \cup d_i$(단, $d_i \leftarrow d_j$)

단계 3] 최종해

$d_k \leftarrow \min_i\{d_i + d_{ik}\}$ 까지 구해지면, 후방으로 추적하여 최단 경로를 찾는다.

단계 2-1] 에서의 D^*은 소스 노드 v_1 에 관한 기저 상에서 v_i 와 v_j 사이의 최단 거리 d^*_{ij} 행렬이며, D는 소스 노드 v_1 에서 v_i까지의 최단거리 d_i 행렬을 나타낸다.

[그림 2]를 예제로 위의 절차에 의해 다음과 같이 MDST를 만들 수 있다.

1 $d_1 = 0$

2 $d_2 = min_i\{d_i + d_{i2}\}$: v_1에서 가장 가까운 v_2까지의 최단 거리

$i \leftarrow 1$: $d_1 + d_{12} = 0 + 1 = 1$

3 $d_3 = min_i\{d_i + d_{i3}\}$

$i \leftarrow 1$: $d_1 + d_{13} = 0 + 5 = 5$

$i \leftarrow 2$: $d_2 + d_{23} = 1 + 6 = 7$

4 $d_4 = min_i\{d_i + d_{i4}\}$

$i \leftarrow 1$: $d_1 + d_{14} = 0 + 5 = 5$

$i \leftarrow 2$: $d_2 + d_{24} = 1 + 2 = 3$

$i \leftarrow 3$: $d_3 + d_{34} = 5 + 6 = 11$

5 $d_5 = min_i\{d_i + d_{i5}\}$

$i \leftarrow 1$: $d_1 + d_{15} = 0 + 6 = 6$

$i \leftarrow 2$: $d_2 + d_{25} = 1 + 3 = 4$

$i \leftarrow 3$: $d_3 + d_{35} = 5 + 4 = 9$

$i \leftarrow 4$: $d_4 + d_{45} = 3 + 2 = 5$

6 set of links in minimal path ← (2,5) (2, 4) (1, 3) (1, 2)

따라서 주어진 예제 그래프의 MDST는 (그림 3)의 (a)와 같으며, 소스 노드 v_1에서 다른 모든 각각의 노드들까지의 최단경로와 거리는 다음과 같다.

v_1 에서 v_2 까지의 최단 경로는 $v_1 \rightarrow v_2$ 이며, 거리는 1 이다.

v_1 에서 v_3 까지의 최단 경로는 $v_1 \rightarrow v_3$ 이며, 거리는 5 이다.

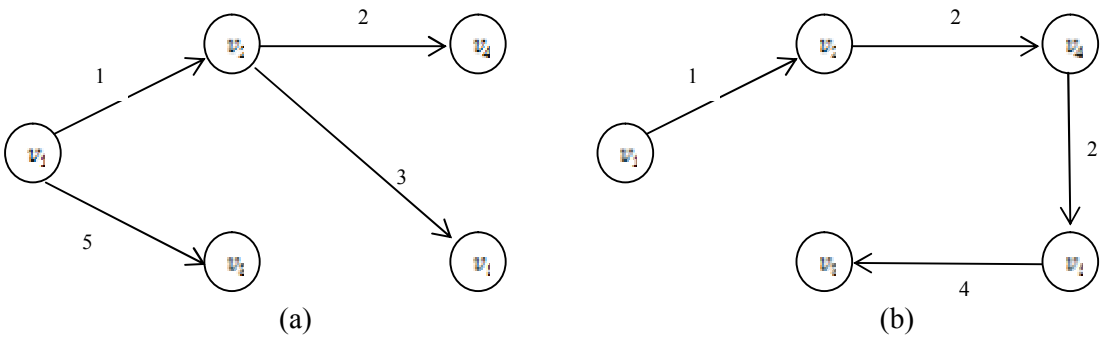
v_1 에서 v_4 까지의 최단 경로는 $v_1 \rightarrow v_2 \rightarrow v_4$이며, 거리는 3 이다.

v_1 에서 v_5 까지의 최단 경로는 $v_1 \rightarrow v_2 \rightarrow v_5$이며, 거리는 4 이다.

이때, 각 최단 경로의 합 $\sum_j d_j$는 13 이 되며, 구해진 신장트리의 링크 가중치 합 $\sum_{(i,j)\in est(A)} d_{ij}$ 는 11임을 알 수 있다. 다음 <표 1>은 소스 노드 v_1 에 관한 기저 상의 $D*$ 와 D 행렬을 보여주는 MDST 테이블을 보여주고 있다.

<표 1> 예제 네트워크의 MDST 테이블

	v_1	v_2	v_3	v_4	v_5
v_1	0	d_2 =1	d_3 =5	d_4 =3	d_5 =4
v_2		0		d^*_{24}=2	d^*_{25}=3
v_3			0		
v_4				0	
v_5					0



<그림 3> 예제 네트워크의 (a) MDST (b) MST

3.2 MST

MST는 네트워크 상의 모든 링크에 주어진 가중치의 합이 최소가 되는 신장트리를 구하는 것으로 목적은 $\mathbf{MIN}\sum_{(i,j)\in st(A)} d_{ij}$ 가 된다.

그래프 *G = (V, A)* 에서 링크에 주어진 가중치가 주어질 때, MST를 구하는 방법으로는 *Prim* 알고리즘과 *Kruskal* 알고리즘 등이 알려져 있으나 본 연구에서는 논문의 3.1에서 제시한 MDST를 구현하는 절차를 확대하여 구하고자 한다. 즉, <표 1>의 행렬에 비어있는 d^*_{ij}를 구하여 네트워크상의 모든 각 두 노드 간의 최단 경로를 찾아내는 것이다. 이때, 방향이 없는 네트워크에서 두 노드간의 링크는 오는 화살과 가는 화살의 합으로 되어있다라고 보면 되기 때문에 이 네트워크의 거리 행렬은 대칭이 된다. 이러한 네트워크에서 모든 두 노드 간의 최단 경로를 구한다는 것은 MST를 구하는 것과 같다. MST를 만드는 절차는 다음과 같다.

단계 1] 소스노드 v_1을 시발점으로 다른 모든 각각의 노드까지의 최단경로를 식(1)을 이용하여 구한다. 이 단계는 MDST를 구하는 절차와 같으며, 이를 통해 d^*_{ij}를 구하여 행렬을 만든다.

단계 2] 단계1에서 구해진 행렬 중 빈 공간이 가장 많이 남은 행이나 열을 선택하여 이 노드를 다시 시발점으로 선택하여 단계1의 과정을 거친다.

단계 3] 단계2의 과정을 행렬이 완성될 때까지 계속한다.

단계 4] 완성된 행렬에서 전방에서부터 탐욕 알고리즘에 의해 다음의 절차에 따라 MST를 구한다.

절차 1] 단계3]에서 구해진 MST테이블을 준비한다. 이때의MST는 노드가 없는 상태로 초기화 한다.

절차 2] 소스 노드 v_1을 MST 루트노드에 삽입한다.

절차 3] MST에 삽입되어 있는 노드들과 인접한 모든 노드들 사이의 경로를 조사하여 가장 작은 경로를 갖는 노드를 MST에 삽입한다. 단, 새로 삽입된 노드는 MST에 삽입된 기존의 노드와 사이클을 형성해서는 안된다.

절차 4] 절차 3]의 과정을 반복하다가 MST가 그래프 상의 모든 노드가 신장트리로연결되면 종료한다.

<표 2>는 <그림 2>를 예제로 단계 1]에서 단계 3]에 의해 구해진 네트워크상의 모든 각 두 노드 간의 최단거리 *D**행렬로 MST 테이블을 보여주고 있다.

<표 2> 예제 네트워크의 MST 테이블

	v_1	v_2	v_3	v_4	v_5
v_1	0	$\mathbf{d^*_{12}=1}$	$d^*_{13}=5$	$d^*_{14}=3$	$d^*_{15}=4$
v_2	$d^*_{21}=1$	0	$d^*_{23}=6$	$\mathbf{d^*_{24}=2}$	$d^*_{25}=3$
v_3	$d^*_{31}=5$	$d^*_{32}=6$	0	$d^*_{34}=6$	$d^*_{35}=4$
v_4	$d^*_{41}=3$	$d^*_{42}=2$	$d^*_{43}=6$	0	$\mathbf{d^*_{45}=2}$
v_5	$d^*_{51}=4$	$d^*_{52}=3$	$\mathbf{d^*_{53}=4}$	$d^*_{54}=2$	0

단계 4]의 절차는 다음과 같다

절차 1] <표 2>와 같은 MST테이블을 준비한다.

절차 2] v_1을 먼저 MST에 삽입한다.

절차 3] v_1과 최단경로를 갖는 v_2를 MST에 삽입한다. ← d^*_{12}=1

절차 4] 반복 및 종료

- v_1 또는 v_2와 최단 경로를 갖는 v_4를 MST에 삽입한다. ← d^*_{24}=2
- v_1 또는 v_2 또는 v_4 와 최단 경로를 갖는 v_5를 MST에 삽입한다. ← d^*_{45}=2
- v_1 또는 v_2 또는 v_4 또는 v_5와 최단 경로를 갖는 v_3을 MST에 삽입한다. ← d^*_{53}=4
- set of links in minimal path ← (1, 2) (2, 4) (4, 5) (5, 3)

MST 테이블의 모든 노드가 신장트리로 연결되었으므로 종료한다.

따라서 주어진 예제 그래프의 MST는 (그림 3)의 (b)와 같으며, 소스 노드 v_1에서 다른 모든 각각의 노드들까지의 경로와 거리는 다음과 같다.

v_1 에서 v_2 까지의 경로는 v_2 ← v_1 이며, 거리는 1 이다.

v_1 에서 v_4 까지의 경로는 v_4 ← v_2 ← v_1이며, 거리는 3 이다.

v_1 에서 v_5 까지의 경로는 v_5 → v_4 → v_2 ← v_1 이며, 거리는 5 이다.

v_1 에서 v_3 까지의 경로는 v_3 ← v_5 ← v_4 ← v_2 ← v_1 이며, 거리는 9 이다.

이때, 링크 상의 가중치의 합 $\sum_{(i,j)\in st(A)} d_{ij}$ 는 9가 되며, 소스노드에서 모든 각각의 노드들까지의 경로 가중치 합 $\sum_j d_j$ 는 18임을 알 수 있다.

다음 <표 3>은 주어진 예제 그래프에서 MDST 와 MST 를 비교하고 있다.

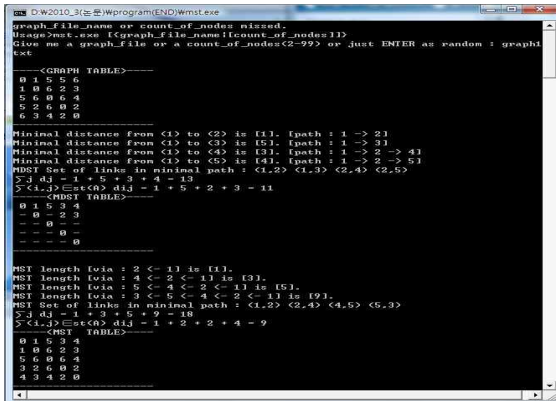
<표 3> 예제 네트워크에서 MDST와 MST 비교

	MDST	MST
$\sum_j d_j$	1+3+4+5 = 13	1+3+5+9 = 18
$\sum_{(i,j)\in st(A)} d_{ij}$	1+2+3+5 = 11	1+2+2+4 = 9
Spanning tree	(그림 3)의 (a)	(그림 3)의 (b)

Ⅳ. 실 험

본 장에서는 3장에서 제시한 완전 연결된 네트워크에서 MDST와 MST를 목적으로 하는 오버레이 멀티캐스트 트리 구현 알고리즘을 실험하여 결과를 보여주고 분석 하고자 한다. 네트워크는 소스노드를 포함한 모든 노드가 완전 연결된 메시구조를 가지며, 링크 상의 가중치는 이웃 노드에 대한 왕복지연시간으로 둔다. 실험은 주어진 네트워크에서 *MIN* $\sum_{j=2}^{k} d_{1j}$를 목적으로 하는 MDST와 $MIN \sum_{(i,j)\in st(A)} d_{ij}$ 를 목적으로 하는 MST를 각각 구하여, 두 신장 트리에서의 $\sum_j d_j$ 와 $\sum_{(i,j)\in st(A)} d_{ij}$ 값을 비교 분석

한다. 또한 구해진 신장트리에 대한 차수(degree) 평활화 정도를 분석하고자 한다. 문제 해결을 위해 C++로 프로그램 하였으며, <그림 4>는 예제 네트워크 <그림 2>에 대한 프로그램 결과를 보여주고 있다.



(그림 4) 예제 네트워크에 대한 실험결과

실행 파일을 시작하면 다음과 같은 질문으로 입력을 기다린다.

```
graph_file_name or count_of_nodes missed.
Usage>mst.exe [{graph_file_name|[count_of_nodes]]}
Give me a graph_file or a count_of_nodes(2-99) or just ENTER as random :
```

먼저 실험을 위해 네트워크를 그래프 테이블로 만드는데, 그 생성 방법은 1)주어진 네트워크에 가중치가 정해지면 그래프 테이블을 파일로 만들어 입력한다. 2)임의의 노드 수 *n*을 입력하면, *n−1*개의 링크에 1에서 9까지 랜덤 가중치를 갖는 그래프 테이블이 생성된다. 3)파일명이나 노드 수를 입력하지 않고 바로 엔터를 쳤을 때는 노드 수와 가중치를 모두 랜덤하게 할당하는 그래프 테이블이 생성된다.

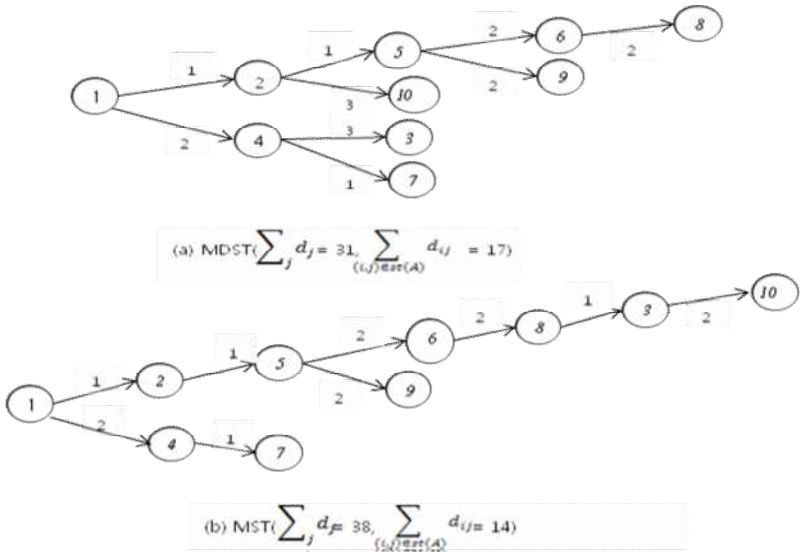
다음은 노드의 수를 10으로 입력했을 때, 랜덤 가중치를 갖는 그래프 테이블이 생성되면서, MDST와 MST를 구하는 실험 결과를 보여주고 있다.

```
Give me a graph_file or a count_of_nodes(2-99) or just ENTER as random : 10
Using RANDOM graph table with 10 nodes.
----<GRAPH TABLE>----
 0 1 9 2 8 7 4 7 8 6
 1 0 6 6 1 6 3 9 4 3
 9 6 0 3 3 8 4 1 3 2
 2 6 3 0 7 6 1 9 5 6
 8 1 3 7 0 2 6 6 2 8
 7 6 8 6 2 0 8 2 5 8
 4 3 4 1 6 8 0 6 3 3
 7 9 1 9 6 2 6 0 7 4
 8 4 3 5 2 5 3 7 0 3
 6 3 2 6 8 8 3 4 3 0
---------------------
```

```
Minimal distance from (1) to (2) is [1]. [path : 1 -> 2]
Minimal distance from (1) to (3) is [5]. [path : 1 -> 4 -> 3]
Minimal distance from (1) to (4) is [2]. [path : 1 -> 4]
Minimal distance from (1) to (5) is [2]. [path : 1 -> 2 -> 5]
Minimal distance from (1) to (6) is [4]. [path : 1 -> 2 -> 5 -> 6]
Minimal distance from (1) to (7) is [3]. [path : 1 -> 4 -> 7]
Minimal distance from (1) to (8) is [6]. [path : 1 -> 2 -> 5 -> 6 -> 8]
Minimal distance from (1) to (9) is [4]. [path : 1 -> 2 -> 5 -> 9]
Minimal distance from (1) to (10) is [4]. [path : 1 -> 2 -> 10]
MDST Set of links in minimal path : (1,2) (1,4) (2,5) (2,10) (4,3) (4,7) (5,6) (5,9) (6,8)
∑j dj = 1 + 5 + 2 + 2 + 4 + 3 + 6 + 4 + 4 = 31
∑(i,j)∈st(A) dij = 1 + 2 + 1 + 3 + 3 + 1 + 2 + 2 + 2 = 17
-----<MDST TABLE>----
 0 1 5 2 2 4 3 6 4 4
 - 0 - - 1 - - - - 3
 - - 0 - - - - - - -
 - - 3 0 - - 1 - - -
 - - - - 0 2 - - 2 -
 - - - - - 0 - 2 - -
 - - - - - - 0 - - -
 - - - - - - - 0 - -
 - - - - - - - - 0 -
 - - - - - - - - - 0
---------------------
MST length [via : 2 <- 1] is [1].
MST length [via : 5 <- 2 <- 1] is [2].
MST length [via : 4 <- 1] is [2].
MST length [via : 7 <- 4 <- 1] is [3].
MST length [via : 6 <- 5 <- 2 <- 1] is [4].
MST length [via : 9 <- 5 <- 2 <- 1] is [4].
MST length [via : 8 <- 6 <- 5 <- 2 <- 1] is [6].
MST length [via : 3 <- 8 <- 6 <- 5 <- 2 <- 1] is [7].
MST length [via : 10 <- 3 <- 8 <- 6 <- 5 <- 2 <- 1] is [9].
MST Set of links in minimal path : (1,2) (2,5) (1,4) (4,7) (5,6) (5,9) (6,8) (8,3) (3,10)
∑j dj = 1 + 2 + 2 + 3 + 4 + 4 + 6 + 7 + 9 = 38
∑(i,j)∈st(A) dij = 1 + 1 + 2 + 1 + 2 + 2 + 2 + 1 + 2 = 14
-----<MST  TABLE>----
 0 1 5 2 2 4 3 6 4 4
 1 0 4 3 1 3 3 5 3 3
 5 4 0 3 3 3 4 1 3 2
 2 3 3 0 4 6 1 4 4 4
 2 1 3 4 0 2 4 4 2 4
 4 3 3 6 2 0 6 2 4 5
 3 3 4 1 4 6 0 5 3 3
 6 5 1 4 4 2 5 0 4 3
 4 3 3 4 2 4 3 4 0 3
 4 3 2 4 4 5 3 3 3 0
---------------------
```

실험 결과 10개의 노드를 갖는 랜덤 그래프 테이블이 제시되었으며, MDST 및 MST 테이블이 생성되어 각각의 목적에 맞는 신장트리와, 각각의 신장트리에 대한 $\sum_j d_j$ 그리고 $\sum_{(i,j)\in st(A)} d_{ij}$ 값을 구할 수 있었다.

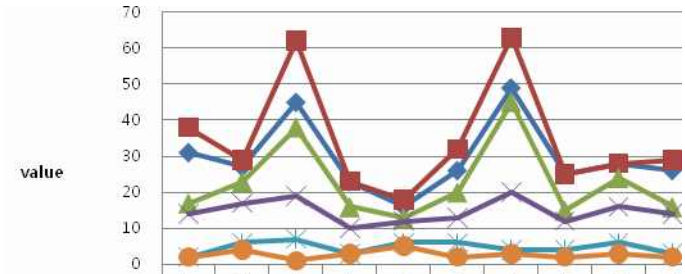
다음 <그림 5>는 실험 네트워크의 결과로 생성된 MDST와 MST를 보여주고 있다.



<그림 5> 실험 네트워크의 MDST와 MST

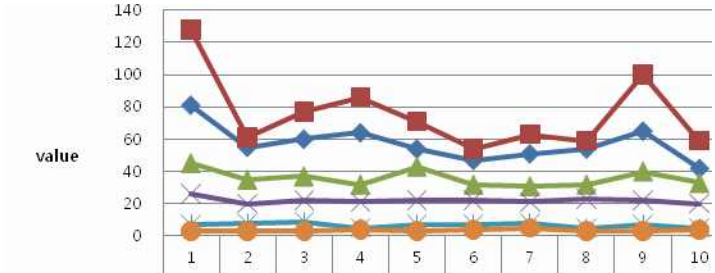
<그림 5>의 MDST와 MST에서, 각각 최대 차수를 갖는 노드는 그 차수값이 2이므로, 네트워크의트래픽이 특정 노드에 집중되지 않고, 신장트리 내의 모든 노드들에 비교적 평활화 되었음을 보여주고 있다.

다음 <그림 6>, <그림 7>, <그림 8> 그리고 <그림 9>는 노드 수를 각각 10, 20, 30, 그리고 50개로 두고 각각 10회 그래프 테이블을 랜덤 발생 시켜 생성한 MDST와 MST를 비교하고 있다. 여기서 Max(degree)는 구해진 신장트리에서 최대 차수값을 나타내며, 이는 트래픽의 평활화 척도로 삼고자 한다.



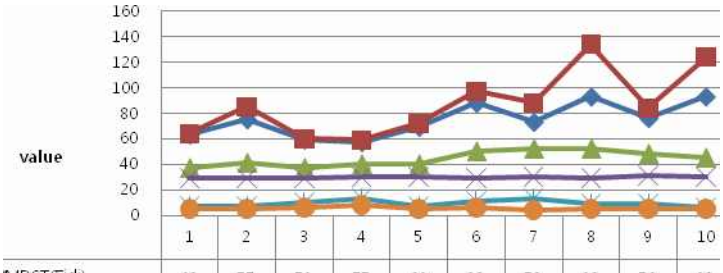
	1	2	3	4	5	6	7	8	9	10
MDST(Σjdj)	31	27	45	23	16	26	49	25	28	26
MST(Σjdj)	38	29	62	23	18	32	63	25	28	29
MDST(Σ(i,j)∈(A)dij)	17	23	38	16	13	20	45	15	24	16
MST(Σ(i,j)∈(A)dij)	14	17	19	10	12	13	20	12	16	14
MDST_max(degree)	2	6	7	3	6	6	4	4	6	3
MST_max(degree)	2	4	1	3	5	2	3	2	3	2

<그림 6> 10개 노드의 MDST와 MST 비교



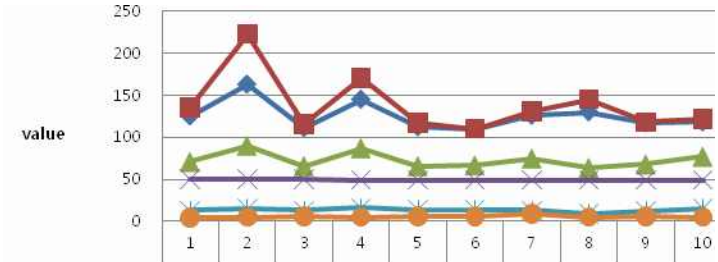
	1	2	3	4	5	6	7	8	9	10
'MDST(Σjdj)	81	55	60	64	54	47	51	54	65	42
'MST(Σjdj)	128	61	77	86	71	54	63	59	100	59
'MDST(Σ(i,j)∈(A)dij)	45	35	37	32	43	32	31	32	40	33
'MST(Σ(i,j)∈(A)dij)	26	20	22	21	22	22	21	23	22	20
'MDST_max(degree)	7	8	9	5	7	7	8	5	7	5
'MST_max(degree)	3	3	3	4	3	4	5	3	3	4

<그림 7> 20개 노드의 MDST와 MST 비교



	1	2	3	4	5	6	7	8	9	10
'MDST(Σjdj)	63	75	59	57	69	88	73	93	76	93
'MST(Σjdj)	64	85	60	59	72	97	88	134	84	124
'MDST(Σ(i,j)∈(A)dij)	37	41	37	40	40	50	52	52	48	45
'MST(Σ(i,j)∈(A)dij)	29	29	29	30	30	29	30	29	31	30
'MDST_max(degree)	7	7	10	13	7	11	13	9	9	6
'MST_max(degree)	5	5	6	8	5	6	4	5	5	5

<그림 8> 30개 노드의 MDST와 MST 비교



	1	2	3	4	5	6	7	8	9	10
'MDST(Σdj)	125	163	111	145	112	110	126	130	117	118
'MST(Σdj)	135	223	116	170	117	110	131	145	118	122
'MDST(Σ(i,j)∈(A)dij)	71	90	66	87	66	67	75	64	68	77
'MST(Σ(i,j)∈(A)dij)	50	50	50	49	49	49	49	49	49	49
'MDST_max(degree)	13	15	13	16	14	14	13	9	12	15
'MST_max(degree)	4	5	6	5	6	6	9	5	6	5

〈그림 9〉 50개 노드의 MDST와 MST 비교

실험 결과 MIN $\sum_{j=2}^{k} d_{1j}$ 를 목적으로 하는 MDST와 $\mathbf{MIN}\sum_{(i,j)\in st(A)} d_{ij}$ 를 목적으로 하는 MST에서, 모든 경우 MDST는 $\sum_j d_j$값이 그리고 MST는 $\sum_{(i,j)\in st(A)} d_{ij}$ 값이 상대적으로 작은 값을 보였다. 신장트리에서 트래픽 평활화 지수로 삼고 있는 최대 차수는 MDST의 경우 소스 노드에서의 차수가 대부분 최대 차수임을 보였고, MST보다 항상 큰 값을 나타냈다. MST에서의 $\sum_{(i,j)\in st(A)} d_{ij}$ 값은 노드 수가 증가 할 때, 완전 연결된 그래프에서 가중치를 1에서 9까지 랜덤하게 발생 시킨 이유로 모든 링크가 1에 근사한 값으로 연결됨을 볼 수 있었다. 따라서 MDST는 MST에 비해 서비스 지연시간이 QoS에 민감한 멀티캐스트에 적합함을 보였다. 특히, 실험 결과에 의해 생성된 MST 테이블은 그래프 상의 모든 노드들 간 각각의 최단 거리 행렬이며, 이를 통해 전방으로부터 탐욕알고리즘을 적용하여 MST를 생성하기 때문에 차수 제약, 대역폭 제약 또는 링크의 고장에 의한 제약 등에 따른 대체 경로 설정에 빠르게 적용할 수 있는 기회를 보여주었다. 그러나, 본 연구에서는 제약 조건에 의한 대체 경로 설정 문제는 다루지 않으며, 차후 연구에서 논하기로 하겠다.

Ⅴ. 결 론

본 연구에서는 완전 연결된 네트워크에서 MDST와 MST를 목적으로 하는 오버레이 멀티캐스트 트리를 구현하는 문제를 다루었다. 이를 위해 다익스트라 기법을 응용한 탐욕 알고리즘으로 MDST와 MST를 동시에 구하였다. MDST는 서비스 지연시간이 QoS에 민감한 멀티케스트에 적합함을 보였고, MST 테이블은 다양한 제약에 따른 대체 경로 설정 문제 해결을 기대할 수 있게 하였다. 오버레이 멀티캐스트는 IP 멀티캐스트의 분명한 장점에도 불구하고, 현실적인 적용의 어려움을 해결하는 대안임을 보였다.

본 연구는 IPTV등 다양한 멀티캐스트 서비스를 제공하기 위한 오버레이 멀티캐스트 망 구현에 관한 연구에 활용될 수 있을 것으로 기대된다.

참 고 문 헌

S. Deering and D. Cheriton(1990), "Multicast Routing in Datagram Internetworks and Extended LANS," ACM Trans. Comp.Syst., Vol.8, No.2, pp.85-111.

S. E. Deering(December 1991), 'Multicast routing in a datagram internetwork,' PhD thesis, Stanford University.

C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen(January 2000), "Deployment issues for the IP multicast service and architecture," *IEEE Network*, Vol.14, No.1, pp.78-88.

Y. Chawathe(Dec. 2000), 'Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service.' PhD thesis, University of California, Berkeley, USA.

Y. Chu, S. G. Rao, S. Seshan, and H. Zhang(Oct. 2002), "A case for end system multicast," *IEEE Journal on Selected Areas in Communications*,

Vol.20, No.8, pp.1456-1471.

J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. J. O' Toole(Oct 2000), 'Overcast: Reliable multicasting with an overlay network,' 4th Symposium on Operating System Design and Implementation (OSDI).
D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel(2001), 'Almi : An application level multicast infrastructure,' 3rd Usenix Symposium on Internet Technologies and Systems, USITS 2001, March 26-28, San Francisco, California.
P. Francis(1999), 'Yoid: Extending the Multicast Internet Architecture,' available: http://www.icir.org/yoid/docs/ycHtmlL/htmlRoot.html.
H. Deshpande, M. Bawa, and H. Garcia-Molina(2001), "Streaming Live Media over a Peer-to-Peer Network," Tech. Rep. version 2001-31, available: http://dbpubs.stanford.edu/pub/2001-30.
M. Kwon and S. Fahmy(May 2002), "Topology-Aware Overlay Networks for Group Communication," ACM NOSSDAV, Miami, FL, pp. 127- 36.
V. Roca and A. El-Sayed(July 2001), 'A Host-Based Multicast (HBM) Solution FOR Group Communications,' IEEE Int'l. Conf. Net.
PAN Yun, YU Zhenwei and WANG Licheng(2003), 'A Genetic Algorithm for the Overlay Multicast Routing Problem,' ICCNMC'03.
Zongming Fei, and Mengkun Yang(FEBRUARY 2007), "A Proactive Tree Recovery Mechanism for Resilient Overlay Multicast," IEEE/ACM TRANSACTIONS ON NETWORKING, Vol.15, No.1, pp.173-186.
Shan Jin, Yanyan Zhuang, Linfeng Liu, and Jiagao Wu(2007), 'An Efficient Overlay Multicast Routing Algorithm for Real-Time Multimedia Applications,' APWeb/WAIM 2007, LNCS 4505, pp. 829-836.
Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller(2003), 'Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications,' IEEE INFOCOM , pp.1521-1531.
Bernard M. Waxman(DECEMBER 1988), " Routing of Multipoint Connections," *IEEE Journal on Selected Areas in Communications*, Vol.6, No.9, pp.1617-1622.
Sherlia Y. Shi and Jonathan S. Turner(2001), 'Routing in Overlay Multicast Networks, ' available:http://www.arl.wustl.edu/Publications/2000-04/wucs0119.pdf .
박승권(2008-05). 'TPS와 대응전략 및 사례', 2008 IPTV 및 관련종목 세미나, 산학교육연구소.
'IPTV 표준화 동향(2007-7)', 2007 VOIP 사업과 IPTV+BcN 컨버전스 세미나, 한국정보통신기술협회.

Overlay Multicast Tree Building Algorithm for MDST and MST in Complete Network

Cho, Myeong Rai *

Abstract

It is strongly believed that multicast will become one of the most promising services on internet for the next generation. Multicast service can be deployed either on network-layer or application-layer. IP multicast (network-layer

* Professor Ph.D., Dept. of U-Information Industry, Jeonnam Provincial College, mrcho@dorip.ac.kr

multicast) is implemented by network nodes (i.e., routers) and avoids multiple copies of the same datagram on the same link. Despite the conceptual simplicity of IP multicast and its obvious benefits, it has not been widely deployed since there remain many unresolved issues. As an alternative to IP multicast, overlay multicast (application-layer multicast) implements the multicast functionality at end hosts rather than routers. This may require more overall bandwidth than IP multicast because duplicate packets travel the same physical links multiple times, but it provides an inexpensive, deployable method of providing point-to-multipoint group communication.

In this paper we develop an efficient method applied greedy algorithm for solving two models of overlay multicast tree building problem that is aimed to construct MDST (Minimum Diameter Spanning Tree : minimum cost path from a source node to all its receivers) and MST (Minimum Spanning Tree : minimum total cost spanning all the members). We also simulate and analyze MDST and MST.

Keywords: Overlay Multicast, MDST : Minimum Diameter Spanning Tree, MST : Minimum Spanning Tree, Greedy Algorithm