논문 99-24-6B-13 한국통신학회논문지 '99-6 Vol.24 No.6B

이질형 멀티미디어 멀티캐스트를 위한 효과적인 스트림 분배 알고리즘

정회원 김 승 훈*

Efficient Stream Distributions Algorithms for Heterogeneous Multimedia Multicast

Seung-Hoon Kim* *Regular Member*

요 약

멀티미디어 응용에서 송신자는 일반적으로 다중 스트림을 생성하게 된다. 이질형 멀티미디어 멀티 캐스트에서는 송신자가 생성한 다중 스트림을 수신자가 모두 수신할 필요 없이 일부만을 수신할 수 있다. 수신자는 수신하기를 원하는 스트림에 대하여 입찰하고 연결이 성립되면 동일한 양을 송신자는 이익으로 갖게 된다. 송신자의 이익을 최대로 하는 이질형 멀티 캐스트를 위한 스트림 분배 문제는 *NP*-complete로 알려진 0-1 정수 문제로 유도된다. 본 논문에서는 멀티캐스트 트리상의 임의 링크의 용량이 그 자손 링크의 용량보다 작지 않다는 제약조건을 가지는 제약된 모델과 제약되지 않은 모델을 모두 고려하여, 기존에 제시된 알고리즘에 비하여 시간 복잡도와 공간 복잡도 면에서 우수한 알고리즘들을 제안한다. 이 알고리즘은 또한 분산화 되어 구현되기 쉬우며, 이는 대규모 네트워크에서 매우 중요한 장점이 된다.

ABSTRACT

In multimedia applications, a source usually generates multiple streams. By heterogeneous multimedia multicast, we mean a recipient can receive some of them, not necessarily all of them. A recipient bids for what it wants to receive and the source gains the same amount when a connection is established. The problem of distributing streams for heterogeneous multicast to maximize the source's gain, can be solved using a 0-1 integer programming, known as *NP*-complete. In this paper, we propose efficient stream distribution algorithms in two different types of multicast models. The first restricted model assumes that the capacity for a link in the multicast tree is greater than or equal to the capacities of its descendant links. In the second unrestricted model, we drop out the restriction in the restricted model. Proposed algorithms have better time and space complexities compared with any existing one. In addition, distributed implementations are straightforward, which is very useful for large networks.

I. 서 론

원격 강의, 주문형 비디오 등의 멀티미디어 서비스에서는 송신자가 다중 수신자에게 정보를 송신하는 멀티캐스트(multicast)가 더욱 중요해진다. 기존 서비스와 달리, 멀티미디어 서비스에서 송신자는 일반적으로 다중 멀티미디어 스트림(multiple multimedia streams) 정보를 생성하게 되며 각 스트림은 다른 매체(medium)를 나타내게 된다. 멀티캐스트 전송문제는 송신자를 루트 노드(root node)로 하고 수신자를 잎 노드(leaf nodes)로 하는 멀티캐스트 트리(multicast tree)로 모델링 된다. 멀티미디어 스

* 상지대학교 전자계산공학과(edina@chiak.sangji.ac.kr)
논문번호 : 98328-0803, 접수일자 : 1998년 8월 3일

트림을 여러 개의 멀티캐스트 트리 혹은 다른 구조에서 분배할 수 있으나, 이러한 방식에서는 더욱 많은 자원을 필요로 하고 또한 각 스트림이 서로 다른 경로를 취할 수 있으므로 미디어간 동기화가 더욱 어려워진다는 문제가 있다. 따라서 본 논문에서는 하나의 멀티캐스트 트리 상에서의 다중 멀티미디어 스트림을 전송하는 멀티캐스트를 고려한다. 멀티미디어 스트림은 시간축에서 다중화(multiplex)되며 각 스트림은 링크 대역폭의 일부를 사용하게 된다. 기존의 많은 멀티캐스트 프로토콜에서 모든 수신자는 송신자가 생성한 동일한 정보를 수신한다고 가정 한다[3,4,7]. 그러나 이질형 네트워크(heterogeneous networks)에서는 모든 수신자가 동일한 능력을 갖추지도 않으며 또한 동일한 요구를 하지도 않는다[13]. 즉 수신자들은 자신의 능력과 요구되는 QoS(Quality of Service)에 따라, 멀티캐스트 그룹에 보내지는 스트림 중에서 일부를 수신하기 원할 수 있다. 어떤 매체를 다양한 수준으로 지원하기 위하여 계층적 암호화 기법이 필요하다[9]. 또한 네트워크에서 링크는 서로 다른 용량을 갖는 것이 일반적이다. 이런 이질형 멀티미디어 멀티캐스트(heterogeneous multimedia multicast)에서는 멀티캐스트 트리가 주어졌을때 어떻게 스트림을 트리상에 분배할 것인가를 결정하는 것이 중요하다. 고려되는 스트림 분배 문제에서 수신자들은 요청되는 서비스에 따라 필요한 스트림을 미리 입찰(bid)하고, 획득된 스트림은 지불(pay)한다. 이제부터 본 논문에서 멀티캐스트는 이질형 멀티미디어 멀티캐스트를 의미한다.

스트림 분배 문제(stream distribution problem)에서는 수신자들의 입찰과 멀티캐스트 트리상에서 링크의 사용 가능한 용량을 고려하면서 송신자가 얻을 수 있는 이익(gain)을 최대로 한다. 스트림 분배 문제는 *NP*-complete 문제[5]로 알려진 0-1 정수 프로그래밍 문제(0-1 integer programming problem)로 바꾸어진다. Shacham[9]이 제안된 알고리즘은 두 패스로 구성되어 있다. 첫 패스는 최대 대역폭 트리(maximum bandwidth tree)상에서 송신자로부터 수신자로 하향식으로 진행된다. 패스의 진행 과정에서 흐름의 가능성(feasibility)을 유지하기 위한 확장된 트리(expanded tree)를 구성한다. 이 트리를 구성하는 목적은 실행 불가능한(infeasible) 흐름 혹은 이미 발견한 흐름보다 더 좋지 않은 흐름을 열거하여 더 이상 트리에서 고려하지 않도록 하기 위함이다. 결과적으로, 확장된 트리는 최대 대역폭 트리 상에서 링크의 대역폭 제약조건을 만족하는 모든 후보 스트림 분배를 포함하게 된다. 두 번째 상향식 패스에서는 이들 후보 스트림 분배 중에서 최대 이익을 내는 분배를 결정하는 계산을 수행한다. 이 알고리즘은 확장된 트리를 유지하기 위한 상당히 많은 기억 공간과 두 패스를 수행하기 위한 많은 시간 복잡도를 요구한다.

본 저자가 이전에 제안했던 논문[6]에서는 어떤 링크의 용량이 자손 링크의 용량보다 작지 않다는 제약조건을 가지는 멀티캐스트 트리상에서 스트림 분배를 위한 새로운 효과적인 알고리즘을 제안하였다. 본 논문에서는 그 논문[6]을 확장하여 이러한 제약된 모델(restricted model)과 제약되지 않은 모델(unrestricted model)을 대상으로 하여 스트림 분배 문제를 다루었다. 또한 논문[6]에서는 송신자가 제어하는 알고리즘만을 제시하였으나, 본 논문에서는 수신자가 제어하는 알고리즘도 같이 제시한다. 본 논문에서는 먼저 제약된 모델과 제약되지 않은 모델 각각에 대하여 최적 스트림 분배 문제에 대한 일반적 해(generic solutions)를 제시한다. 이 일반적 해는 잎 노드에서 루트 노드로의 상향식 단일 패스만으로 구성되어 있으며 멀티캐스트 트리상의 링크 용량 제약을 만족하면서 최대 이익을 구한다. 모델의 일반해로부터 각기 두 개의 실제적인 알고리즘을 유도할 것이다. 제시된 스트림 분배 알고리즘들은 확장된 트리를 유지하기 위한 기억 공간을 유지할 필요가 없고 단일 패스이므로 시간 복잡도를 상당히 감소하였다. 또한 제안한 알고리즘은 수월하게 분산화되어 구현된다. 이 측면은 대규모 네트워크에서 중요한 고려 요소라 하겠다. 일단 스트림 분배가 결정되면, 송신자는 쉽게 멀티캐스트 연결(multicast connection)을 수립할 수 있다.

본 논문은 다음과 같이 구성되어 있다. Ⅱ절은 네트워크 모델과 제안될 알고리즘을 나타내는데 유용한 벡터 연산을 정의한다. 또한 스트림 분배 문제를 정식으로 정의한다. Ⅲ절은 제약된 모델에서의 일반해를 제시하며 Ⅳ절에서는 주어진 일반해에 근거하여 두 알고리즘을 제안한다. 그리고 일반해의 타당성을 증명하고 시간 복잡도를 제시한다. 또한 비교를 위하여 발견적 알고리즘(heuristic algorithm)과 그 시간 복잡도를 제시한다. 이 알고리즘은 최적해를 구하지는 않는다. Ⅴ절에서는 제약되지 않은 모델에서의 알고리즘을 제안하고, 마지막으로 Ⅵ절에서는 결론을 유도한다.

Ⅱ. 모델 및 스트림 분배 문제 유도

1. 네트워크 모델

고려중인 네트워크는 스위치 혹은 호스트에 해당되는 일련의 노드와 이들을 임의의 위상(topology)으로 연결한 링크로 구성된다. 스위치 노드간의 링크는 일대일 접속(point-to-point)으로 간주한다. 그러나 호스트 노드와 스위치 노드간의 링크는 다중분기(multidrop)도 허용한다. 이러한 네트워크는 그래프 $G=(V,E)$로 모델링 된다. 여기서 V는 노드를 나타내는 vertex의 집합이고, E는 링크를 나타내는 edge의 집합이다. (논문의 설명을 간단하게 하기 위하여 앞으로 vertex를 노드, edge를 링크와 혼용하여 사용한다.) 각 링크의 용량을 나타내는 함수 $C:E\rightarrow Z_0^+$가 있다고 가정한다. (Z_0^+는 0를 포함하는 양의 정수를 나타낸다.) 또한 임의의 멀티캐스트 라우팅 알고리즘[7,9]에 의하여 그래프 G에서 멀티캐스트 트리가 미리 주어졌다고 가정한다. 이 트리는 $T=(V',E')$로 나타내며, 여기서 $V'\subseteq V$ 그리고 $E'\subseteq E$ 이다. 그래프 상에서 이러한 트리의 예가 그림 1에 주어졌다. 트리상의 링크는 굵은 선으로 나타난다. 링크 $e_j\in E'$는 용량 C_j를 가진다. 여기서 $j=1,\ldots,L$ 그리고 $L=|E'|$. e_k가 e_j의 후손 일때 $C_j\geq C_k$를 만족하면 이 네트워크는 제약된 모델에 속하며, 만족하지 않으면 제약되지 않은 모델에 속하게 된다. 링크의 용량은 사용 가능한 링크 용량과 요구되는 서비스 질을 위한 대역폭의 최소 값으로 결정되게 된다. 트리상의 링크는 ATM 네트워크에서의 가상 채널(Virtual Channels) 혹은 가상 경로(Virtual Paths)와 같은 가상 링크로 생각할 수 있다. 만일 요구되는 서비스의 질이 멀티캐스트 트리에서 하향식으로 내려가면서 더 높아지지 않는다면 제약된 모델이 적합한 모델이 된다. 송신자는 N 매체 스트림을 생성하며, 스트림 s는 대역폭 w_s를 미리 요구한다.

트리 T에서, 루트 노드는 송신자이며 잎 노드는 수신자가 된다. 본 논문에서는 일반적인 알고리즘과는 다소 다르게, 트리 T의 링크에 중점을 두어 알고리즘이 전개되어 간다. 따라서 논문의 편의상 다음과 같은 용어를 비슷하게 정의하여 사용한다. 잎 노드로 들어가는 링크를 잎 링크(leaf link)라 하고, 잎 링크가 아닌 링크를 트리 링크(tree link)라 한다. 즉 모든 수신자는 잎 링크에만 연결되게 된다. 편의상 루트 노드는 하나의 링크인 루트 링크(*root_link*)만을 가진다 하자. 만일 루트 노드가 하나 이상의 링크를 가진다면 루트 노드와 그 자식 노드사이에 가상의 노드를 하나 첨가할 수 있다. 링크의 레벨(level)은 노드의 레벨에 1을 더한 값으로 정의된다. 가장 낮은 잎 링크는 레벨 1이다.

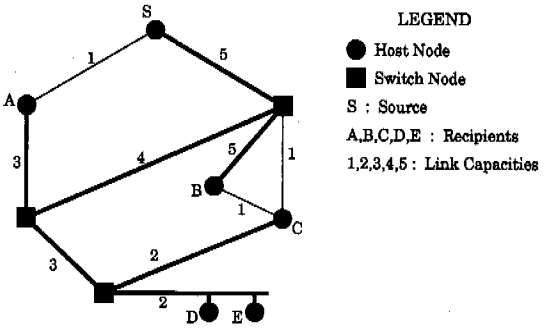

그림 1. 임의 그래프에서 멀티캐스트 트리의 예.

2. 벡터 연산 정의

제안할 알고리즘을 표기하는데 유용한 벡터 연산의 정의와 표기를 소개한다. 벡터 $\mathbf{x}$는 N-tuple의 수 $(x_1,\ldots,x_N)$를 나타낸다. 두 벡터 $\mathbf{x}=(x_1,\ldots,x_N)$와 $\mathbf{y}=(y_1,\ldots,y_N)$를 다음과 같이 비교할 수 있다:

$(\forall i)(x_i\leq y_i)\Leftrightarrow \mathbf{x}\leq\mathbf{y}$,

$(\mathbf{x}\leq\mathbf{y})\wedge(\exists i)(x_i<y_i)\Leftrightarrow \mathbf{x}<\mathbf{y}$, 그리고

$(\exists i,j)(x_i<y_i\wedge x_j>y_j)\Leftrightarrow \mathbf{x}\parallel\mathbf{y}$ (혹은, $\mathbf{x}\not\leq\mathbf{y}$ 그리고 $\mathbf{x}\not\geq\mathbf{y}$).

$\mathbf{x}\odot\mathbf{y}$로 표기되는 벡터 $\mathbf{x}$와 $\mathbf{y}$의 성분곱(component-wise product)은 $\mathbf{x}$와 $\mathbf{y}$의 각기 해당 성분의 곱으로 얻어지는 벡터로 정의 된다. 즉,

$$\mathbf{x}\odot\mathbf{y}=(x_1y_1,\ldots,x_Ny_N)$$

그리고 $\mathbf{x}=(x_1,\ldots,x_N)$의 1-norm은 $\|\mathbf{x}\|_1$로 표기되며 다음과 같이 정의된다

$$\|\mathbf{x}\|_1=\sum_{i=1}^{N}|x_i|.$$

두 벡터 $\mathbf{x}$와 $\mathbf{y}$의 성분 x_i와 y_i가 모두 0보다 크거나 같다면, 여기서 $i=1,\ldots,N$, 두 벡터의 내적(inner product)은 다음과 같이 표현된다:

$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + \ldots + x_N y_N = \| \mathbf{x} \odot \mathbf{y} \|_1.$$

3. 스트림 분배 문제 유도

스트림을 트리 T에 분배하는 문제는 링크 $e_j \in E'$에 어떤 스트림을 할당하는가 하는 문제가 된다. 링크 e_j로의 스트림 할당(stream assignment)은 N-tuple 벡터 $\mathbf{z}_{i_j} = (z_{i_{j1}}, \ldots, z_{i_{jN}})$로 표현된다. 여기서

$$z_{i_{js}} = \begin{cases} 1 & \text{만일 스트림 } s\text{가 링크 } e_j\text{에할당되면} \\ 0 & \text{그렇지 않으면} \end{cases}$$

첨자 i는 가능한 많은 할당중의 임의 할당을 나타내기 위하여 사용한다. 링크 e_j로의 실행 가능한 스트림 할당(feasible stream assignment)은 다음 제약조건을 만족하는 할당 $\mathbf{z}_{i_j}$이다[10]:

C1 연속성(Continuity).

모든 흐름은 루트 노드에서 출발하며 트리 노드에서는 새로운 스트림이 생성되지 않는다. 즉 e_k가 e_j의 후손일 때 $(\forall e_j, e_k \in E')(\ \mathbf{z}_{i_j} \geq \mathbf{z}_{i_k})$.

C2 링크 대역폭(Link bandwidth).

링크 대역폭이 지원 가능할 경우만 링크 e_j에 스트림을 할당할 수 있다. 즉 N-스트림을 위한 대역폭 요구가 $\mathbf{w} = (w_1, \ldots, w_N)$이면,

$$\sum_s z_{i_{js}} w_s = \mathbf{z}_{i_j} \cdot \mathbf{w} \leq C_j.$$

동일한 링크에 대한 임의의 두 실행 가능한 할당 $\mathbf{z}_{i_j}$와 $\mathbf{z}_{i'_j}$에 대하여, 만일 $\mathbf{z}_{i_j} < \mathbf{z}_{i'_j}$이면 $\mathbf{z}_{i_j}$가 $\mathbf{z}_{i'_j}$에 의하여 지배된다(dominated)고 한다. 이는 $\mathbf{z}_{i_j}$가 $\mathbf{z}_{i'_j}$의 순전한 부분할당(proper sub-assignment)이고, $\mathbf{z}_{i'_j}$로부터 어떤 스트림을 삭제함으로써 $\mathbf{z}_{i_j}$를 얻을 수 있다는 것을 의미한다. $\mathbf{z}_{i_j}$를 지배하는 실행 가능한 할당이 없다면, $\mathbf{z}_{i_j}$는 최대 할당이고 지배당하지 않는 할당(nondominated assignment)이라고 한다. 지배당하지 않는 할당에 추가로 스트림을 할당하면 반드시 **C2** 제약조건을 위반하게 된다. 링크 e_j는 많은 지배당하지 않는 할당을 가질 수 있으므로 그들을 모두 Pareto 집합 A_j에 원소로 유지하는 것이 편리하다.

$$A_j = \{\ \mathbf{z}_{i_j} |\ \mathbf{z}_{i_j}\text{는 링크 } e_j\text{로의 지배당하지 않는 할당}\}$$

Pareto 집합 A_j와 A_k에 대하여, 다음 조건을 만족하면 A_k가 A_j에 의하여 지배된다고 한다:

$$(\forall\ \mathbf{z}_{i_k} \in A_k)(\exists\ \mathbf{z}_{i_j} \in A_j)(\ \mathbf{z}_{i_j} \geq \mathbf{z}_{i_k}).$$

지배당하지 않는 할당과 Pareto 집합의 정의에 의하여, $C_j \geq C_k$이면 A_k가 A_j에 의하여 지배된다는 것을 추론할 수 있다. 따라서, 다음과 같은 관찰이 가능하다.

O1. 제약된 모델에서, 멀티캐스트 트리 상의 어떤 링크의 Pareto 집합은 그 선조 링크(ancestor links)의 Pareto 집합에 의하여 지배된다.

입찰(bid)은 각 수신자에 의하여 각 스트림에 할당된 음수가 아닌 값이다. 송신자는 그 스트림을 위한 연결이 수신자에 설립되었을 때 동일한 크기의 이익(gain)을 갖는다. N-벡터를 사용하여 N 스트림을 위한 이익과 입찰 정보를 나타낼 수 있다. 수신자 k의 모든 스트림을 위한 입찰은 입찰 벡터(bid vector) $\mathbf{b}_k = (b_{k1}, \ldots, b_{kN})$로 표현된다. 여기서 b_{ks}는 스트림 s를 위한 입찰을 나타낸다. 만일 잎 링크가 다중분기(multidrop)일 경우, 여기에 연결된 수신자들을 하나의 가상 수신자로 대치하고 수신자들의 입찰 벡터들의 벡터합으로 가상 수신자의 입찰 벡터를 사용한다. 따라서, 잎 링크는 오직 한 수신자만을 가지는 트리를 고려하여도 충분하다. 링크 e_j의 한 지배당하지 않는 할당 $\mathbf{z}_{i_j} \in A_j$와 관련된 이익 벡터 $\mathbf{g}_{i_j}$는 만일 e_j가 잎 링크일 경우는 수신자의 입찰 벡터를 이용하여 얻어지며, 그렇지 않을 경우는 그 자식 링크들의 이익 벡터를 이용하여 얻어진다. 만일 $z_{i_{js}} = 0$이면, $g_{i_{js}} = 0$이 된다. 이제 스트림 분배 문제는 다음과 같이 공식화된다:

스트림분배문제(StreamDistributionProblem): 송신자는 대역폭 요구 $\mathbf{w} = (w_1, \ldots, w_N)$를 가지는 N-스트림을 생성한다. 트리 T상에서 용량 C_j를 가지는 각 링크 $e_j \in E'$에 대하여, 다음과 같은 지배당하지 않는 할당 $\mathbf{z}_{i_j} = (z_{i_{j1}}, \ldots, z_{i_{jN}}) \in A_j$을 찾아라

$$\textit{maximize}\quad \mathbf{z}_{i_j} \cdot \mathbf{g}_{i_j}\ \textit{subject to}\quad \mathbf{z}_{i_j} \cdot \mathbf{w} \leq C_j$$

그리고 $\mathbf{z}_{i_j} \geq \mathbf{z}_{i_k}$ 만일 e_k가 e_j의 후손이면. 이 문제는 Shacham[10]에서의 문제와 동일하다. 이 문제를 해결하기 위하여, E'의 모든 링크의 Pareto 집합내의 모든 할당과 연관된 이익 벡터를 구해야만 한다. 이러한 이익 벡터들은 잎 링크로부터 루트 링크로 레벨 단위로 계산할 수 있다.

Ⅲ. 제약된 모델에서 문제의 최적 부구조

본 절에서는 제약된 모델에서 스트림 분배 문제의 최적 부구조(Optimal Substructure)를 제시한다. 어떤 문제에 대한 어떤 최적해가 그 자체 내에서 부문제(subproblems)에 대한 최적해를 포함하는 특징을 가질 때, 이 문제는 최적 부구조 특성을 보인다고 한다[2]. 지배당하지 않는 할당과 이익 벡터 사이에는 일대일 관계(one-to-one correspondence)가 존재한다. 따라서 링크 e_j는 이익 집합(gain set) G_j로 특징지어질 수 있다.

$G_j=\{\ \mathbf{g}_{i_j}|\ \mathbf{g}_{i_j}$는 할당 $\mathbf{z}_{i_j}\in A_j$와 연관된 이익 벡터$\}$

G1. 만일 링크 e_j가 수신자 k와 입찰 벡터 $\mathbf{b}_k$를 갖는 잎 링크이면,

$$G_j=\{\ \mathbf{g}_{i_j}|\ \mathbf{z}_{i_j}\in A_j,\ \mathbf{g}_{i_j}=\mathbf{z}_{i_j}\odot \mathbf{b}_k\}.$$

G2. 만일 링크 e_j가 m개의 자식 링크 e_{k_n}를 갖는 트리 링크이면, 여기서

$n=1,\ldots,m$,

$$G_j=\left\{\ \mathbf{g}_{i_j}|\ \mathbf{z}_{i_j}\in A_j,\ \mathbf{g}_{i_j}=\sum_{n=1}^{m}\left(\mathbf{z}_{i_j}\odot \mathbf{g}'_{k_n}\right)\right\},$$

$$\mathbf{g}'_{k_n}=\left\{\ \mathbf{g}_{i_{k_n}}|\max_{\mathbf{z}_{i_{k_n}}\in A_{k_n}}\left\{\mathbf{z}_{i_j}\cdot \mathbf{g}_{i_{k_n}}\right\}\right\}.$$

즉, $\mathbf{g}'_{k_n}$는 e_{k_n}과 연관된 이익 벡터들 중에서 $\|\mathbf{z}_{i_j}\odot \mathbf{g}_{i_{k_n}}\|_1=\mathbf{z}_{i_j}\cdot \mathbf{g}_{i_{k_n}}$의 최대값을 갖는 이익 벡터이다. 여기서 $\mathbf{g}_{i_{k_n}}\in G_{k_n}$는 $\mathbf{z}_{i_{k_n}}\in A_{k_n}$과 연관되어 있다.

두 법칙 **G1**과 **G2**를 사용하여, 트리 T상에서의 각 링크에 대한 이익을 계산할 수 있다. 또한 이 법칙은 최적 부구조 특성을 가진다. 다음절에서는 이러한 일반해를 근거로 하여 두 개의 실제적인 알고리즘을 유도한다.

Ⅳ. 제약된 모델에서의 스트림 분배 알고리즘 제안

제안될 스트림 분배 알고리즘들은 집중화(centralized) 혹은 분산화(distributed)되어 구현이 가능하다. 집중화 구현의 경우, 스트림의 분배를 담당할 프로세서는 멀티캐스트 트리, 송신자의 대역폭 요구 및 모든 수신자의 입찰을 모두 알아야 한다. 분산화 구현의 경우, 각 링크에 한 프로세서를 할당하는 것이 편리하다. 이제, 링크(실제로는 링크에 할당된 프로세서)는 멀티캐스트 트리의 일부, 즉 링크의 용량만을 알면 된다. 또한, 링크는 송신자의 대역폭 요구와 잎 링크이면 수신자의 입찰을 알아야 한다. 알고리즘이 진행되면서, 링크는 자식링크로부터 이익 정보를 수신하고 부모 링크로 이익 정보를 송신한다. 대규모 네트워크에 사용되기 위하여 알고리즘의 효율적인 분산화 구현이 요청된다.

1. 링크에 대한 모든 지배당하지 않는 할당을 열거하는 프로시저

링크에 대한 모든 지배당하지 않는 할당을 열거하는 프로시저는 논문[10]에 있는 모든 가능한 흐름 Pareto 집합을 열거하는 알고리즘과 거의 비슷하다. 프로시저 **nondominated**를 간단히 설명한다. 링크 $e_j\in E$에 대한 모든 지배당하지 않는 할당을 열거하기 위하여 링크 용량 C_j와 대역폭 요청 벡터 $\mathbf{w}=(w_1,\ldots,w_N)$가 필요하다. 프로시저 **nondominated**는 Pareto 집합 A_j를 생성하며 그 원소 $\mathbf{z}_{i_j}$는 **C2** 링크 대역폭 제약조건에 맞는 지배당하지 않는 할당을 나타내는 N-벡터이다

먼저 **nondominated**는 모든 N 스트림을 포함하는 할당이 **C2**를 위반하는지 검토한다. 만일 위반하지 않으면, 이 할당이 집합 A_j의 유일한 원소가 되고 프로시저는 종료한다. 그렇지 않으면 $(N-1)$ 스트림을 포함하는 N 할당들을 검토한다. 만일 어떤 할당이 제약조건을 위반하지 않으면 Pareto 집합에 첨가되고 그 할당에 의해 지배되는 모든 부할당(sub-assignments)은 지배되는 것으로 표기(marked)된다. 그 다음은 $(N-2)$ 스트림을 포함하는 $\binom{N}{2}$ 할당을 검토하고, 이러한 방식을 계속한다. 일반적으로, N 스트림중에서 M 스트림을 포함한 $\binom{N}{M}$ 할당을 검토한다. 여기서 $M=N,\ldots,1$. 프로시저의 진행 도중 오직 표기되지 않은 할당만을 검토하며, 이미 발견된 지배당하지 않는 할당의 자식들인 표기된 지배된 할당들은 더 이상 검토하지 않는다. 프로시저는 같은 수의 스트림을 포함한 모든 할당들이 표기되거나 Pareto 집합에 포함되면 종료된다.

2. 알고리즘 1: Dynamic_Programming

트리 링크 e_j에 대한 Pareto 집합 A_j과 연관된 이익 집합 G_j를 구하기 위하여, 링크가 m자식 링크를 가지며 자식 링크 e_{k_n}에 대한 Pareto 집합을 A_{k_n}이

라고 가정하자, 여기서 $n=1,\ldots,m$. 임의의 할당 $\mathbf{z}_{i_j} \in A_j$에 대한 이익 벡터 를 계산하기 위하여, 모든 자식 링크 e_{k_n}에 대한 Pareto 집합 A_{k_n}과 연관된 이익 집합 G_{k_n}을 구해야 한다. 이 이익 집합들은 다른 할당 $\mathbf{z}_{i'_j} \in A_j$에 대한 이익 벡터를 계산하기 위하여도 또 다시 구해야 한다. 따라서 A_j의 첫 번째 할당 $\mathbf{z}_{i_j}$에 대한 이익 벡터를 계산할 때 구해진 자식 링크에 대한 이익 집합을 구한 후 이를 테이블에 저장하여 필요할 때마다 사용한다면 매번 자손 링크에 대한 이익 집합을 다시 구하는 것에 비하여 상당히 많은 시간을 절약할 수 있다. 이러한 측면에서 단순한 재귀적 알고리즘(recursive algorithm)은 적당하다고 할 수 없다.

```
Algorithm 1 (Dynamic_Programming)
begin
  for all level_l, l = 1,...,top do
    for all e_j of level_l do
    begin
      G_{l,j} = ∅
      for all z_{i_j} ∈ nondominated(e_j) do
      begin
        if (e_j = leaf)
        then
          g_{i_j} = z_{i_j} ⊙ b_k
        else
        begin
          g_{i_j} = 0
          for all e_{k_n} = child(e_j) do
          begin
            ĝ_{k_n} = {g_{i_{k_n}} | max_{g_{i_{k_n}} ∈ G_{l-1,k_n}} {z_{i_j} · g_{i_{k_n}}} }
            g_{i_j} = g_{i_j} + (z_{i_j} ⊙ ĝ_{k_n})
          end
        end
        G_{l,j} = G_{l,j} ∪ {g_{i_j}}
      end
      store G_{l,j} into table
    end
  root_gain = max_{g_{i_j} ∈ G_{top,root_link}} (|| g_{i_j} ||_1)
end
```

동적 프로그램(dynamic-programming) 알고리즘에서는 공통된 부문제(common subproblem)를 상향식 방식으로 한번만 계산한 후 그 결과를 테이블에 저장함으로써, 부문제를 만날 때마다 매번 다시 계산하는 노력을 피할 수 있다. 재계산을 피하는 대신 테이블을 위한 많은 기억 공간이 필요하다.

Algorithm 1 (Dynamic_Programming)은 송신자의 이익 *root_gain*을 계산한다. 테이블의 i번째 행과 j번째 열의 요소(entry)는 $level_l$의 링크 e_j에 대한 이익 집합 $G_{l,j}$을 포함한다. 레벨 l의 링크를 위한 테이블 요소를 계산하기 위하여 **Algorithm 1**은 레벨 $l-1$ 링크들의 테이블 요소만을 필요로 한다. 결과적으로 어떤 임의의 순간에 알고리즘은 단지 인접한 두 행을 위한 기억 공간만을 필요로 하며 실제 필요로 하는 저장 공간을 상당히 줄일 수 있다.

```
Algorithm 2 (Tree_Traversal)
begin
  G_j = posrorder(root_link)
  root_gain = max_{g_{i_j} ∈ G_j} (|| g_{i_j} ||_1)
end

postorder(e_j)
begin
  for all e_{k_n} = child(e_j) do
    G_{k_n} = postorder(e_{k_n})
  G_j = ∅
  for all z_{i_j} ∈ nondominated(e_j) do
  begin
    if (e_j = leaf)
    then
      g_{i_j} = z_{i_j} ⊙ b_k
    else
    begin
      g_{i_j} = 0
      for all e_{k_n} do
      begin
        ĝ_{k_n} = {g_{i_{k_n}} | max_{g_{i_{k_n}} ∈ G_{k_n}} {z_{i_j} · g_{i_{k_n}}} }
        g_{i_j} = g_{i_j} + (z_{i_j} ⊙ ĝ_{k_n})
      end
    end
    G_j = G_j ∪ {g_{i_j}}
  begin
  return(G_j)
end
```

3. 알고리즘 2: Tree_Traversal

Algorithm 1 Dynamic_Programming에서 레벨 l링크를 계산하기 위하여 레벨 $l-1$의 모든 테이블 요소를 필요로 하지 않는다. 사실 임의 링크의 이익 집합을 구하기 위하여 단지 그 링크의 자식 링크의 이익 집합만을 필요로 한다. 따라서 같은 레벨의 모든 요소를 동시에 계산, 저장할 필요는 없다. 이러한 측면에서 후순서 트리 방문(postorder tree traversal) 알고리즘이 동적 프로그래밍 알고리즘보다 우수하다. 후순서 트리 방문 알고리즘은 트리상의 각 노드를 정확히 한번 방문한다. 또한 임의의 노드를 방문하기 전에 그 자식 노드들을 먼저 방문한다.

Algorithm 2 Tree_Traversal은 루트 링크 *root_link*을 입력으로 하고, 이 링크의 이익 *root_gain*을 출력으로 한다. 알고리즘은 재귀적 프로시저 **postorder**를 가진다. 이 프로시저는 링크 e_j를 입력으로 하여 이익 집합 G_j을 출력한다.

4. 동작 예

표 1. nondominated를 수행한 후 링크 e_j를 위한 Pareto 집합 A_j.

e_j	C_j	Pareto 집합 A_j
e_1	8	{(1,1,1,0),(1,1,0,1)}
e_2	3	{(1,1,0,0)}
e_3	7	{(1,1,1,0),(1,0,0,1),(0,1,0,1)}
e_4	3	{(1,1,0,0)}
e_5	1	{(1,0,0,0)}
e_6	6	{(1,1,0,0),(1,0,1,0),(1,0,0,1), (0,1,1,0)}
e_7	4	{(1,1,0,0),(0,0,1,0)}

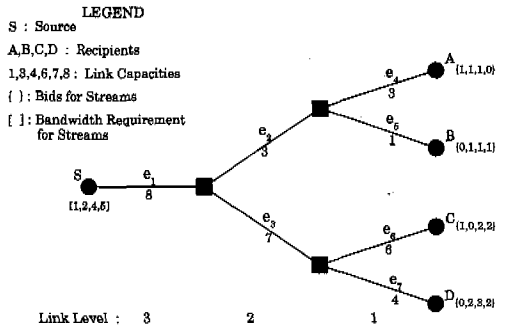

그림 2. 이질형 멀티캐스트 트리의 예.

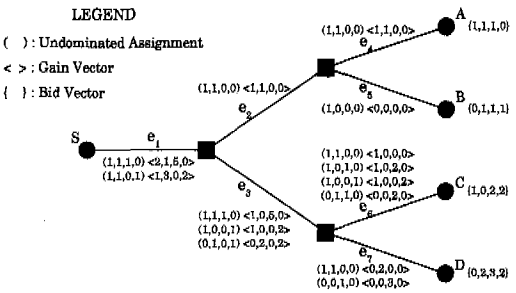

그림 3. 지배당하지 않는 할당 $\mathbf{z}_{i_j}$과 그의 이익 벡터 $\mathbf{g}_{i_j}$.

다음 예는 일반해의 동작을 설명한다. 그림 2는 이질형 멀티캐스트 트리의 예를 보인다. 대역폭 요구, 링크 용량 및 각 수신자의 스트림에 대한 입찰이 그림에 나타나 있다. 트리에서 링크 용량 C_j를 갖는 각 링크 e_j에 대하여, **nondominated**를 수행한 후의 Pareto 집합 A_j가 표 1에 주어져 있다. 알고리즘을 수행한 후의 결과가 그림 3에 있다. 모든 링크 $e_j \in E$에 대하여, 지배당하지 않는 할당 $\mathbf{z}_{i_j} \in A_j$과 그의 이익 벡터 $\mathbf{g}_{i_j}$가 그림에 나타나 있다. 링크 e_4, e_5, e_6 및 e_7는 잎 링크이므로 그들의 이익 집합은 $\mathbf{z}_{i_j}$와 입찰 벡터 $\mathbf{b}_k$의 성분곱으로부터 얻어진다. 트리 링크 e_3의 이익 벡터를 구하기 위한 자세한 계산은 표 2에 주어져 있다.

표 2. 트리 링크 e_3을 위한 이익 벡터 계산.

$\mathbf{z}_{i_3}$	계산	$\mathbf{g}_{i_3}$
(1,1,1,0)	$\max\{\|\|(1,0,0,0)\|\|_1, \|\|(1,0,2,0)\|\|_1, \|\|(1,0,0,0)\|\|_1, \|\|(0,0,2,0)\|\|_1\} + \max\{\|\|(0,2,0,0)\|\|_1, \|\|(0,0,3,0)\|\|_1\}$	(1,0,5,0)
(1,0,0,1)	$\max\{\|\|(1,0,0,0)\|\|_1, \|\|(1,0,0,0)\|\|_1 \|\|(1,0,0,2)\|\|_1, \|\|(0,0,0,0)\|\|_1\} + \max\{\|\|(0,0,0,0)\|\|_1, \|\|(0,0,0,0)\|\|_1\}$	(1,0,0,2)
(0,1,0,1)	$\max\{\|\|(0,0,0,0)\|\|_1, \|\|(0,0,0,0)\|\|_1, \|\|(0,0,0,2)\|\|_1, \|\|(0,0,0,0)\|\|_1\} + \max\{\|\|(0,2,0,0)\|\|_1, \|\|(0,0,0,0)\|\|_1\}$	(0,2,0,2)

5. 알고리즘의 타당성 증명

제안된 두 알고리즘은 일반해, 즉 법칙 **G1**과 **G2**에 근거하여 고안되었으므로, 일반해의 타당성을 증명함으로써 그들의 타당성은 충분히 증명된다. 일반해에 의하여 얻어진 스트림 분배가 최적임을 증명하기 위하여, 최대 이익을 가지는 분배가 선택되었으며 그보다 큰 이익을 가지는 실행 가능한 할당이 존재하지 않음을 보이면 된다. 먼저 다음과 같은 보조정리를 보인다.

보조정리 1.

E의 링크에 대한 임의의 지배당하지 않는 할당이 주어졌을 때, 일반해는 최대 이익 벡터를 생성한다.

증명.

증명은 트리 T의 레벨에 대한 귀납법에 의한다. 먼저, 귀납법 기초로서(induction base), 레벨 1의 링크는 항상 잎 링크이고 **G1**에 의하여 각 할당에 대하여 오직 하나의 이익 벡터를 가진다. 따라서 그 이익 벡터는 최대 이익 벡터이다. 귀납법 가설(inductive hypothesis)로 각 레벨 i의 링크에 대한 지배당하지 않는 할당에 대하여, 일반해가 항상 최

대 이익 벡터를 계산한다고 가정하자. 귀납법 단계로(inductive step), 레벨이 $l+1$일 경우, 링크는 트리 링크 혹은 잎 링크가 된다. 트리 링크의 경우, 일반해는 **G2**를 적용하고 최대 이익 벡터를 생성한다. 잎 링크일 경우, 일반해는 **G1**을 적용하게 된다. 따라서 일반해는 링크에 대한 주어진 할당에 대한 최대 이익 벡터를 생성한다. □

정리 1.

제안된 알고리즘에 의하여 얻어진 스트림 분배는 최적이다.

증명.

O1에 의하여, 제약된 모델의 경우 링크의 Pareto 집합은 그의 선조 링크들에 의하여 지배되는 것을 알았다. 따라서, 지배당하지 않는 할당과 이익 벡터 간의 일대일 대응관 계가 존재하므로, 전체 이익 정보가 잎 링크로부터 루트 링크까지 어떤 이익 정보의 손실 없이 고려된다. **보조정리 1**에 의하여, 일반해는 최대 이익 벡터를 구한다. 따라서 제안 된 알고리즘에 의하여 얻어진 분배는 스트림 분배 문제에 대하여 최적이다. □

6. 시간 복잡도

이미 언급한 바와 같이, **Tree_Traversal**이 시간 복잡도와 요구되는 기억 공간 측면에서 더욱 효과적이며 이에 대한 시간 복잡도만을 제시한다. 제안된 알고리즘의 시간 복잡도는 벡터 연산, 예를 들어 성분곱과 같은 연산에 의하여 측정할 수 있다. 그러나, 벡터 합과 1-norm 연산은 이들이 비교적 간단하기 때문에 복잡도 계산에서 제외한다. 먼저, 제안된 알고리즘에서 사용된 프로시저 **nondominated**의 시간 복잡도를 측정해야 한다. 복잡도는 **보조정리 2**에 주어져 있으며 사실 논문[10]에 제시된 복잡도와 동일하다. 따라서 그 증명은 생략한다.

보조정리 2.

만일 N 스트림 집합이 **nondominated**에 의하여 고려되면, Pareto 집합의 최대 cardinality는 $\binom{N}{\lfloor N/2 \rfloor}$과 같다.

정리 2.

만일 어떤 트리 링크가 최대 K 자식 링크를 가지면, **Tree_Traversal**는 최악의 경우 다음과 같은 벡터 연산에 의해 측정된 시간 복잡도를 가진다.

$$L\binom{N}{\lfloor N/2 \rfloor}\left(1+K\binom{N}{\lfloor N/2 \rfloor}\right)$$

증명.

후순서 트리 방문 알고리즘은 트리상의 각 노드를 정확히 한번 방문한다. 따라서 n노드의 트리가 주어질 때 그 시간 복잡도는 $O(n)$이다[1,2]. 제안된 알고리즘에서는 각 링크를 정확히 한번 방문하며 시간 복잡도의 첫 번째 항 L은 이를 나타낸다. 이익 벡터와 할당간에는 일대일의 대응관계가 존재하고, Pareto 집합의 모든 할당에 대한 이익 벡터를 계산하여야 하므로 **보조정리 2**에 의하여 두 번째 항은 이를 나타낸다. 법칙 **G1**과 **G2**를 이용하여, 한 할당에 대한 이익 집합을 계산할 수 있다. 잎 링크의 경우, 오직 한 성분곱 연산이 **G1**에 의하여 필요하다. 트리 링크는 최대 K 자식 링크를 가지므로, 최악의 경우 성분곱 연산의 수는 **G2**에 의하여 $K\binom{N}{\lfloor N/2 \rfloor}$이다. □

제안된 알고리즘들은 논문[10]의 알고리즘 혹은 모든 링크에 대한 모든 할당을 검토 하는 알고리즘(이 경우 시간 복잡도는 $O(2^{NL})$)에 비하여 낮은 시간 복잡도를 가진다. 그러나 이론적으로 아직 다소 높은 시간 복잡도를 요구하게 되며 이는 주로 Pareto 집합의 최대 cardinality 때문이다. 그러나 만일 N이 적다면, Pareto 집합의 최대 cardinality 또한 적을 것이다. 일반적으로 멀티미디어 멀티캐스트의 경우 스트림의 수는 5를 넘지 않을 것이며 이 경우 최대 cardinality는 10이 된다. 다시 말하여, 링크당 최대 10개의 할당만을 검토하면 된다. 실제적으로 제안된 알고리즘은 오늘날 구현하지 못할 정도로 복잡하지 않다.

두 가지 다른 종류의 발견적 알고리즘(heuristic algorithm)을 생각할 수 있다. 첫 번째 종류는 이익 집합의 최대 cardinality를 제한하는 것에 근거를 둔다. 예를 들어, 프로시저 **postorder**를 호출한 후 얻어진 이익 집합에 1-norm 연산을 적용하고 그 값이 큰 순서대로 M개의 원소를 선택함에 의하여 제안된 알고리즘의 시간 복잡도를 줄일 수 있다. 두 번째 종류의 알고리즘은 Pareto 집합의 최대 cardinality를 제한하는 것에 근거를 둔다. 예를 들어, 지배당하지 않는 할당과 대역폭 요청 벡터와의 내적 연산을 수행한 후 그 값이 큰 순서대로 M개의 할당만을 검토한다. 이 경우, 이 경험적 알고리즘은 다항식 시간 복잡도 (polynomial time complexity)인 $O(LM(1+KM))=O(LKM^2)$를 가진다. 물론, 이 알고리즘은 최적 분배를 생성하지 못할 수 있다.

V. 제약되지 않은 모델에서 스트림 분배 알고리즘

1. 제약되지 않은 모델에서 문제에 대한 최적 부구조

본 절에서는 링크의 용량이 자손 링크의 용량보다 크거나 같다는 제약을 생략한 제약되지 않은 모델에서 스트림 분배 문제에 대한 알고리즘을 개발한다. 제약된 모델에서 존재했던 이익 벡터와 할당간의 일대일 대응관계가 제약되지 않은 모델에서는 더 이상 존재하지 않는다. 따라서 이 모델에서는 동일한 링크에 대한 많은 이익 벡터 중에서 다른 이익 벡터에 의하여 지배당하지 않는 이익 벡터(non-dominated gain vector) $\mathbf{g}_{i_j}$에 관심이 있다. 즉 동일한 링크에 대한 이익 벡터 $\mathbf{g}_{i_j}$와 $\mathbf{g}_{i'_j}$에 대하여, $\mathbf{g} < \mathbf{g}_{i'_j}$인 이익 벡터 $\mathbf{g}_{i'_j}$가 존재하지 않는 최대 이익 벡터 $\mathbf{g}_{i_j}$를 지배당하지 않는 이익 벡터라 한다. 이제 이 모델에서 Pareto 집합 A_j을 가지는 링크 e_j에 대한 이익 집합 G_j을 다음과 같이 정의한다:

$G_j = \{ \mathbf{g}_{i_j} |$ $\mathbf{g}_{i_j}$는 $\mathbf{z}_{i_j} \in A_j$를 위한 지배 당하지 않는 이익 벡터$\}$

법칙 **G1**과 **G2**을 수정한 새로운 두 법칙에 의하여 이 모델의 이익 집합도 재귀적으로 계산할 수 있다.

G1' 만일 e_j가 수신자 k를 가지는 잎 링크이면,

$G_j = \{ \mathbf{g}_{i_j} | \mathbf{z}_{i_j} \in A_j,\ \mathbf{g}_{i_j} = \mathbf{z}_{i_j} \odot \mathbf{b}_k \}$.

G2' 만일 e_j가 m자식 링크 e_{k_n}을 가지는 트리 링크이면, 여기서 $n = 1, \ldots, m$이고 e_{k_n}은 이익 집합 G_{k_n}을 가진다,

$$G_j = \{ \mathbf{g}_{i_j} |\ \mathbf{z}_{i_j} \in A_j,\ \mathbf{g}_{i_{k_n}} \in G_{k_n},$$

$$\mathbf{g}_{i_j} = \sum_{n=1}^{m} (\mathbf{z}_{i_j} \odot \mathbf{g}_{i_{k_n}}),$$

$\mathbf{g}_{i_j}$은 지배당하지 않는 이익 벡터$\}$.

2. 알고리즘 3: Dynamic_Programming

Algorithm 3 (Dynamic_Programming)은 전 절의 **Algorithm 1**과 비슷하며 또한 송신자 이익 *root_gain*을 생성한다. 테이블의 i번째 행과 j번째 열의 요소는 $level_l$의 링크 e_j를 위한 이익 집합 $G_{l,j}$를 포함한다. **Algorithm 1**이 임의의 순간에 오직 인접한 두 행만을 필요로 한다는 사실은 **Algorithm 3**에도 또한 적용된다.

3. 알고리즘 4: Tree_Traversal

Algorithm 4 (Tree_Traversal)도 역시 전 절의 **Algorithm 2**와 비슷하며, 링크 e_j를 취하여 이익 집합 G_j를 생성하는 재귀적 프로시저 **postorder**를 가진다. 그리고 이 알고리즘은 시간 복잡도와 필요한 기억 공간 측면에서 **Algorithm 3**에 비하여 우수하다.

```
Algorithm 3 (Dynamic_Programming)
begin
  for all level_l, l = 1, ..., top do
    for all e_j of level_l do
    begin
      G_{l,j} = ∅
      if (e_j = leaf)
      then
        for all z_{i_j} ∈ nondominated(e_j) do
        begin
          g_{i_j} = z_{i_j} ⊙ b_k
          G_{l,j} = G_{l,j} ∪ {g_{i_j}}
        end
      else
      begin
        for all z_{i_j} ∈ nondominated(e_j),
            all e_{k_n} = child(e_j), all g_{i_{k_n}} ∈ G_{l-1,k_n} do
        begin
          g_{i_j} = Σ_{n=1}^{m} (z_{i_j} ⊙ g_{i_{k_n}})
          G_{l,j} = G_{l,j} ∪ {g_{i_j}}
          reconstruct G_{l,j}
        end
      end
      store G_{l,j} into table
    end
  root_gain = max_{g_{i_j} ∈ G_{top,root_link}} (|| g_{i_j} ||_1)
end
```

```
Algorithm 4 (Tree_Traversal)
begin
  G_j = posrorder(root_link)
  root_gain = max_{g_{i_j} ∈ G_j} (|| g_{i_j} ||_1)
end
```

```
postorder(e_j)
begin
  for all e_{k_n} = child(e_j) do
    G_{k_n} = postorder(e_{k_n})
  G_j = ∅
  if (e_j = leaf)
  then
    for all z_{i_j} ∈ nondominated(e_j) do
      begin
        g_{i_j} = z_{i_j} ⊙ b_k
        G_j = G_j ∪ {g_{i_j}}
      end
```

Ⅵ. 결 론

본 논문에서는 스트림 분배 문제를 다루었고 두 종류의 멀티캐스트 모델에 대한 효과적인 알고리즘을 제시하였다. 첫 번째 모델에서는 멀티캐스트 트리상에서 링크의 용량이 후손 링크의 용량보다 크거나 같다는 제약을 가진다. 네트워크에서 링크는 가상의 용량을 가지는 가상 링크로 생각되어질 수 있으므로 이러한 제약조건은 심각한 것은 아니나, 더욱 일반적인 멀티캐스트를 고려하기 위하여 두 번째 모델에서는 이러한 제약조건을 제거하였다. 제약된 모델과 제약되지 않은 모델 각각에 대하여, 링크에 대한 스트림 할당을 위한 이익을 계산하기 위한 두 개의 법칙을 제시하였다. 이러한 법칙에 근거하여 각기 두 개의 실제적인 알고리즘이 개발되었다. 제안된 알고리즘은 시간 복잡도와 요구되는 기억 공간 측면에서 기존의 어떠한 알고리즘에 비하여 우수하다. 시간 복잡도를 더욱 줄이기 위하여, 발견적 알고리즘이 또한 제시되었다. 이 알고리즘은 최적이 아닌 분배를 생성할 수 있다. 비록 제안된 알고리즘들이 이론적으로 다항식 시간 복잡도를 갖지는 못하지만 제한된 수의 멀티미디어 스트림을 가질 경우는 실제적으로 훌륭한 성능을 보이며, 다른 알고리즘들의 성능을 평가할 척도로 유용하게 사용될 수 있다. 또한, 제안된 알고리즘들의 분산화 구현이 용이하며, 이는 대규모 네트워크에서 사용되기 위한 필수적인 요소이다. 대부분의 멀티미디어 서비스의 경우 일반적으로 멀티미디어 스트림의 수가 적으므로, 제안된 알고리즘들이 실제 응용에 적용 될 수 있을 것으로 생각한다.

참 고 문 헌

[1] A.V. Aho, J.E. Hopcoft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, 1992.

[3] S.E. Deering and D.R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. on Computer Systems*, 8(2), pp. 85-110, May 1990.

[4] K. Bharath-Kumar and J.M. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Trans. on Comm*, 31(3), pp. 343-351, Mar. 1983.

[5] M.R. Garey and D.S. Johnson, *Computers and Intractability: Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[6] S.-H. Kim, K. Kim and C. Kim, "Efficient Stream Distribution Algorithm for Heterogeneous Multimedia Multicast with Link Capacity Constraint," *Information Processing Letters*, 63, pp. 309-315, 1997.

[7] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos, "Multicasting Routing for Multimedia Communication," *IEEE/ACM Trans. on Networking*, 1(3), pp. 286-292, June 1993.

[8] C.A. Noronha and F.A. Tobagi, "Optimum Routing of Multicast Streams," *in: Proc. IEEE INFOCOM '94*, pp. 865-873, 1994.

[9] N. Shacham, "Multipoint Communication by Hierarchically-encoded Data," *in: Proc. IEEE INFOCOM '92*, pp. 2107-2114, 1992.

[10] N. Shacham and J.S. Meditch, "An Algorithm for Optimal Multicast of Multimedia Streams," *in: Proc. IEEE INFOCOM '94*, pp. 856-864, 1994.

[11] B.M. Waxman, "Routing of Multipoint Connections," *IEEE J. Selected Areas Comm.*, 6(9), pp. 1617-1622, Dec. 1988.

[12] B.M. Waxman, "Performance Evaluation of Multipoint Routing Algorithms," *in: Proc. IEEE INFOCOM '93*, pp. 980-986, 1993.

[13] L. Zhang, S. Deering, D.Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource ReServation Protocol," *IEEE Network*, 7(5), pp. 8-18, 1993.

김 승 훈(Seung-Hoon Kim) 정회원

1985년 2월 : 인하대학교 전자계산학과 졸업
1989년 8월 : 인하대학교 전자계산학과 석사
1998년 2월 : 포항공과대학교 전자계산학과 박사
1998년 3월~현재 : 상지대학교 전자계산공학과 전임강사

<주관심분야> 컴퓨터 네트워크, 초고속통신망, 알고리즘