Journal of The Korea Society of Computer and Information
Vol. 17 No. 6 , June 2012

2012-17-6-3-5

공유 네트워크에서 공유대역폭 트리 구성을 위한 선형 시간 알고리즘

정 균 락*

A Linear Time Algorithm for Constructing a Sharable-Bandwidth Tree in Public-shared Network

Kyun-Rak Chong *

요 약

본 논문에서는 공유 네트워크에서 최소 공유대역폭 트리 구축 문제의 근사해를 구하기 위한 선형 시간 알고리즘을 제안한다. 공유 네트워크는 자신이 소유한 AP의 일부 대역폭을 다른 사람들과 공유하는 사용자가 생성하는 통신 기반 구조로 사용자는 이러한 공유 AP를 통해 어디서나 인터넷을 사용하고 데이터를 전송할 수 있다. 최근에 공유 네트워크에서 SVC 기술을 사용하는 비디오 스트리밍 전송 시스템을 구축하는 방안이 제안되었는데 서버로부터 모든 클라이언트에게 비디오 스트림을 보내기 위해서는 트리 구조를 만든다. 클라이언트의 비디오 스트림 요구를 전부 만족시키는 트리 구조를 생성하는데 있어, 사용되는 공유 AP의 공유대역폭의 합이 최소가 되는 것이 바람직한데 최소 공유대역폭 트리 구축 문제는 NP-하드임이 증명되어 있다. 이 문제를 해결하기 위해 기존에 발표된 알고리즘들은 해를 잘 찾지 못하거나 효율적이지 못한 단점을 가지고 있다. 실험 결과를 보면 제안 알고리즘이 기존의 알고리즘보다 해를 찾는 성공률이나 해의 결과에서 모두 우수하였다.

▸ Keyword : 대역폭 공유, 공유 네트워크, 스케일러블 비디오 코딩, 비디오 스트리밍

Abstract

In this paper we have proposed a linear time algorithm for solving the minimum sharable-bandwidth tree construction problem. The public-shared network is a user generated infrastructure on which a user can access the Internet and transfer data from any place via access points with sharable bandwidth. Recently, the idea of constructing the SVC video streaming

• 제1저자 : 정균락 • 교신저자 : 정균락
• 투고일 : 2012. 05. 03, 심사일 : 2012. 05. 09, 게재확정일 : 2012. 05. 17.
* 홍익대학교 컴퓨터공학과 (Dept. of Computer Engineering, HongIk University)
※ 이 논문은 2010학년도 홍익대학교 학술연구진흥비에 의하여 지원되었음

delivery system on public-shared network has been proposed. To send video stream from the stream server to clients on public-shared network, a tree structure is constructed. The problem of constructing a tree structure to serve the video streaming requests by using minimum amount of sharable bandwidth has been shown to be NP-hard. The previously published algorithms for solving this problem are either unable to find solutions frequently or less efficient. The experimental results showed that our algorithm is excellent both in the success rate of finding solutions and in the quality of solutions.

▸ Keyword : bandwidth sharing, public-shared network, scalable video coding(SVC), video streaming

I. 서 론

최근에 공유 네트워크(public-shared network)를 구축하여 사용자가 자신이 소유한 AP(access point)의 일부 대역폭을 다른 사람과 공유하는 방법이 대두되었는데, 공유대역폭을 제공하는 공유 AP를 통해 어디서나 인터넷에 접근할 수 있고 데이터를 전송할 수 있다. 공유네트워크 예로는 FON[1]이 있는데 FON은 현재 가장 큰 Wi-Fi 커뮤니티로 회원들은 인터넷에 연결되는 공유 AP(FON 라우터)를 소유하고 있으며, 자신의 대역폭의 일부를 다른 사람들과 나누어 사용한다. 그러므로 회원들은 어디에서든지 인터넷에 무료로 접속할 수 있고, 파일을 업로드하거나 다운로드할 수 있으며, 인터넷에 연결된 프린터 같은 장비에 Wi-Fi 접속이 가능하다 [2, 3].

SVC(scalable video coding)는 영상 컨텐츠를 다양한 공간적 해상도와 화질, 다양한 프레임율을 갖는 하나의 비트스트림을 구성하여 여러 가지 단말기에서 자기 자신의 성능에 맞도록 비트스트림을 받아 복원할 수 있게 하는 비디오 코딩 기술이다 [4, 5, 6, 7]. SVC 기술은 여러 네트워크 플랫폼에서 비디오 스트림을 전송하는데 응용되고 있는데 WiMAX 네트워크[8], 애드혹 네트워크[9], 모바일 TV[10], 위성 방송[11], P2P[12] 등이 있다. P2P 스트리밍 시스템 구조가 [13]에서 연구되었고, 연결 지향 네트워크에서 대량의 스트림을 전송하는 방안이 [14]에서 연구되었다.

최근에 공유 네트워크에서 응용 애플리케이션으로 SVC 방식으로 코드화된 비디오 스트림을 전송하는 방안이 제시되었다 [15, 16]. 제안된 비디오 스트림 전송 시스템들의 구조는 비디오 스트림을 제공하는 스트리밍 소스, 관리 서버, 서버로부터 비디오 스트림을 클라이언트에 전달하는 공유 AP들로 이루어져 있다. 서버로부터 모든 클라이언트에게 비디오 스트림을 보내기 위해 루트가 서버인 트리 구조를 만드는데, 내부노드는 공유 AP이며 리프(leaf)는 클라이언트가 된다. 그림 1에 그 예가 나타나 있다. 공유 네트워크에서 공유 AP들은 서로 다른 ISP에 의해 제어될 수 있으므로 제안된 시스템들은 OSI 응용 계층에서 구현되었고 인터넷 토폴로지를 바꾸지 않으면서 공유AP를 중앙에서 제어할 수 있다고 가정한다.

클라이언트들이 인터넷상에 널리 퍼져있으면 비디오 스트림이 다수의 링크를 따라 보내질 수 있다. 이 경우 네트워크 대역폭을 더 많이 점유하게 되고 전송시간도 길어져서 클라이언트들이 비디오 스트림을 제시간에 전송받아 복원하지 못하게 된다. 이 문제를 해결하기 위해 유사한 추적 경로를 갖는 공유 AP와 클라이언트를 같은 그룹에 할당하여 트리를 구축하고 이러한 그룹들을 더 큰 그룹으로 만들어 계층적 트리를 구축하는 방법이 [16]에서 제시되었다.

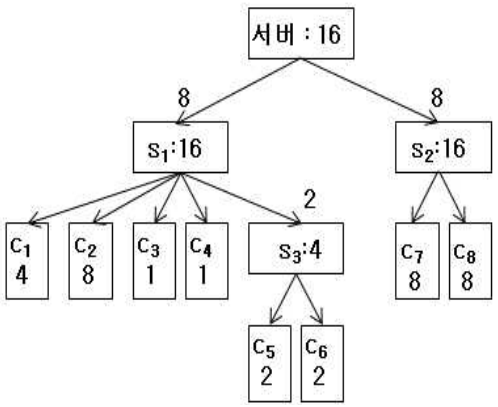

그림 1. 공유대역폭 트리의 예 (1단위=64kbs)
Fig.1. An example of sharable-bandwidth tree (1 unit=64kbs)

[15]에서 제안된 비디오 전송 시스템에서 공유 AP는 비디오 스트림 증폭기 역할을 한다. 하향 링크를 따라 공유 AP

로 전달된 비디오 스트림은, 여기서 다중 복사되어 다른 공유 AP들로 전달된다. 이와 같이 하면 비디오 스트리밍 소스는 작은 양의 대역폭으로 비디오 스트림을 시스템에 전달할 수 있게 되고, 클라이언트들은 비디오 스트림을 동시에 받아 볼 수 있다.

클라이언트가 요구하는 비디오 스트림을 성공적으로 보내기 위해서는 트리를 만드는데 사용된 공유대역폭의 총용량을 최소로 하는 것이 바람직하다. 최소 공유대역폭 트리 구축 문제는 NP-하드임이 증명되어 있으며 이 문제의 근사해를 구하기 위한 알고리즘(이하 D 알고리즘)도 [15]에서 제안되었다. D 알고리즘은 트리를 구축할 때, 공유대역폭이 큰 공유 AP부터 요구대역폭이 큰 클라이언트를 먼저 할당하기 때문에 공유대역폭이 큰 공유 AP가 일찍 소모되어 해(solution)가 존재하는데도 해를 찾지 못하는 비율이 상당히 높은 단점을 가지고 있다. 공유AP의 수를 n, 클라이언트의 수를 m이라 할 때, D 알고리즘의 시간 복잡도는 O(n+m)이다.

D 알고리즘보다 해를 찾는 성공률이 높은 알고리즘(이하 I 알고리즘)이 [17]에서 제안되었는데 I 알고리즘은 공유대역폭이 작은 공유 AP부터 요구대역폭이 작은 클라이언트를 먼저 할당한다. 이 방법은 공유대역폭이 큰 공유 AP를 나중에 사용하므로 해를 찾는 성공률은 높으나 구축된 공유대역폭 트리가 사용하는 전체 공유대역폭이 다소 커지는 단점을 가지고 있으며 시간 복잡도는 $O(n^2+m)$이다.

본 연구에서는 최소 공유대역폭 트리 구축 문제의 근사해를 구하기 위한 선형 시간 알고리즘을 개발하고 기존 알고리즘들과 실험을 통해 비교하였다. 실험 결과에 의하면 제안 알고리즘은 기존의 알고리즘들에 비해 해를 찾는 성공률과 해의 결과 모두 우수한 것으로 나타났다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 자세히 기술하고 3장에서는 최소 공유대역폭 트리 구축 문제에 대해 정의하고 제안된 알고리즘에 대해 기술한다. 4장에서는 제안 알고리즘과 기존 알고리즘들을 실험을 통해 비교 분석하고, 5장에서 결론을 맺는다.

II. 관련 연구

이 장에서는 최소 공유대역폭 트리 구축 문제를 해결하기 위해 이미 발표된 D 알고리즘[15]과 I 알고리즘[17]에 대해 구체적으로 살펴보고 문제점을 기술한다.

1. D 알고리즘

D 알고리즘은 기본적으로 빈 패킹(bin packing)과 유사한 개념을 사용하고 있는데 공유 AP를 빈으로 생각하고 클라이언트를 물체로 생각하여 클라이언트를 공유 AP에 할당한다. 여기서 빈은 크기는 동일하지 않다. 공유 AP를 공유대역폭의 크기에 따라 내림차순으로 정렬하고, 클라이언트도 요구대역폭의 크기에 따라 내림차순으로 정렬한 뒤 요구대역폭이 큰 클라이언트부터 공유대역폭이 큰 공유 AP에 할당하는데, 가장 먼저 할당 가능한 공유 AP에 할당한다. 이 방법은 빈 패킹 알고리즘 중 First Fit Decreasing과 유사하다 [18]. 클라이언트가 할당된 공유 AP는 클라이언트처럼 간주되고 다시 다음 공유 AP에 할당하는 것을 반복함으로써 계층적으로 공유대역폭 트리를 구축하고 있다. 그러므로 공유대역폭 트리를 성공적으로 구축하기 위해서는 부모노드의 공유대역폭의 크기가 자식노드들의 요구대역폭의 크기의 합보다 크거나 같아야하기 때문에 루트에 가까이 있는 공유 AP일수록 공유대역폭의 크기가 커야한다. 그런데 D 알고리즘은 공유대역폭 트리를 구축할 때 공유대역폭이 큰 공유 AP들을 낮은 레벨부터 사용하기 때문에 루트에 도달하기 전에 공유대역폭이 큰 공유 AP들이 모두 소모되는 단점이 있다. 따라서 공유대역폭 트리를 구축할 수 있는 경우에도 해를 찾지 못하는 경우가 자주 발생하게 된다.

2. I 알고리즘

I 알고리즘은 공유 AP와 클라이언트를 각각 공유대역폭과 요구대역폭에 관한 오름차순으로 정렬한 뒤, 요구대역폭이 작은 클라이언트부터 공유 AP에 할당하는데, 가장 먼저 할당 가능한 공유 AP에 할당한다. 이 방법은 해를 찾는 경우는 많아지나 공유대역폭 트리가 사용하는 총 공유대역폭은 증가하는 단점을 가지고 있다. 이 단점을 개선하기 위해 I 알고리즘은 공유노드(공유 AP) 추가와 공유노드 환원이라는 연산을 수행한다.

공유노드 추가는 클라이언트들의 요구대역폭의 합이 공유노드의 공유대역폭이 넘지 않을 때까지 최대한 클라이언트들을 할당한 뒤에도 공유노드의 공유대역폭이 남아있을 경우, 앞에서 클라이언트 할당이 완료된 공유노드를 남아있는 공유대역폭이 소진될 때까지 현재 공유노드에 추가 할당한다.

공유노드 추가를 수행했는데도 공유대역폭이 남아있으면 공유노드 환원을 수행한다. 공유노드 환원은 공유노드에 할당된 클라이언트들을 부모노드에 직접 할당하고 공유노드는 공유대역폭 트리에서 제거하는 것을 말한다. 즉, 공유노드 s가 공유노드 p의 자식이라 할 때 공유노드 s가 사용하고 있는 공유대역폭이 공유노드 s의 요구대역폭과 공유노드 p의 미사용

공유대역폭의 합을 넘지 않을 경우 공유노드 s의 자식들을 공유노드 p의 자식들로 재할당하는 것을 의미한다. 환원되는 공유노드는 공유대역폭 트리에서 제외되므로 환원되는 공유노드의 공유대역폭만큼 공유대역폭 트리가 사용하는 총 공유대역폭이 줄어들게 된다.

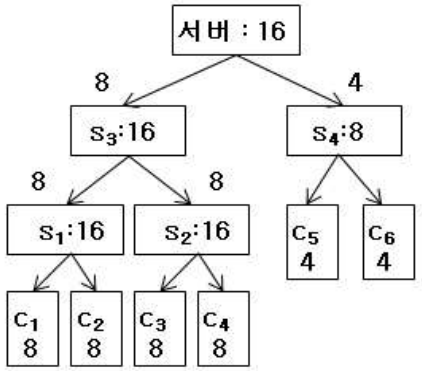

그림 2. 제안 알고리즘에 의해 구축된 공유대역폭 트리
Fig. 2. Sharable-bandwidth tree constructed by our proposed algorithm

I 알고리즘은 공유노드 추가와 공유노드 환원이라는 연산을 수행하여 해를 찾는 성공률을 높이고 원래보다 해의 질을 개선시키지만, 아직도 공유대역폭 트리가 사용하는 총 공유대역폭이 다소 큰 단점을 가지고 있다.

III. 문제 정의와 제안 알고리즘

1. 최소 공유대역폭 트리 구축 문제

이 장에서는 최소 공유대역폭 트리 구축 문제에 대해 기술하고자 한다 [15, 17]. 공유대역폭 트리의 루트는 서버이고, 내부노드는 공유AP이고, 리프는 클라이언트이다. 먼저 서버, 공유 AP와 클라이언트가 갖는 특성을 보면 다음과 같다.

서버의 자식은 클라이언트나 공유AP가 될 수 있는데 자식들의 요구대역폭의 합은 주어진 서버의 용량을 초과하지 못한다. 각 클라이언트 c_i는 요구대역폭 cb_i를 가지며, 클라이언트는 자식은 없고 부모는 공유AP이거나 서버이다.

공유AP s_i의 자식은 클라이언트나 다른 공유AP이며, 자식들에게 제공할 수 있는 공유대역폭 sb_i를 가지는데 자식들의 요구대역폭의 합은 sb_i를 초과하지 못한다. 자식들이 할당된 공유AP의 요구대역폭은 자식들의 요구대역폭 중 최대 요구대역폭으로, 자식들의 대역폭 요구들을 만족시키기 위해 부모로부터 비디오 스트림을 받는데 필요한 대역폭이다.

공유대역폭 트리의 공유대역폭은 트리를 구성하고 있는 모든 공유AP들의 공유대역폭의 전체 합이다.

최소 공유대역폭 트리 구축 문제는 공유AP들의 집합 S, 클라이언트의 집합 C와 서버의 용량이 주어졌을 때, 클라이언트들의 모든 요구를 만족시킬 수 있는 공유대역폭이 최소인 공유대역폭 트리를 구하는 문제이다.

예를 들어, 공유AP의 공유대역폭이 (sb_1, sb_2, sb_3) = (16, 16, 4)이고, 클라이언트의 요구대역폭이 (cb_1, cb_2, cb_3, cb_4, cb_5, cb_6, cb_7, cb_8) = (4, 8, 1, 1, 2, 2, 8, 8)이며, 서버 용량 L = 16일 때 구축 가능한 공유대역폭 트리 중 하나가 그림 1에 나타나 있다. 편의상 64kbs를 1단위로 사용하였고, 따라서 대역폭이 16이면 1024kbs를 의미한다. 그림 1에서 클라이언트(리프) 안에 있는 숫자는 클라이언트의 요구대역폭을 나타내고, 공유AP(내부노드) 안에 있는 숫자는 공유대역폭을, 공유AP 위에 있는 숫자는 공유AP의 요구대역폭을 나타낸다. 이 공유대역폭 트리에 사용된 공유AP는 s_1, s_2, s_3이고, 따라서 이 트리의 공유대역폭은 36이다.

2. 제안 알고리즘

본 논문에서 제안하는 알고리즘의 개요는 다음과 같다. 먼저 공유AP와 클라이언트를 각각 공유대역폭과 요구대역폭에 관한 내림차순으로 정렬한다, 공유AP 할당은 현재 사용가능한 공유AP부터 클라이언트의 요구대역폭의 합이 공유대역폭을 넘지 않을 때까지 클라이언트를 순서대로 공유AP에 할당한다. 클라이언트의 할당이 완료된 공유AP는 클라이언트 리스트의 마지막에 추가되어 클라이언트처럼 취급된다.

제안 알고리즘은 공유대역폭이 큰 공유AP들이 낮은 레벨에서 모두 소모되는 단점을 막기 위해 공유대역폭 트리를 구축하는 과정에서 일정 비율의 공유AP를 다음 레벨에서 사용할 수 있도록 비축해 놓는다. 비율을 계산하는 방법은 {sb_1, sb_2, ..., sb_n}을 공유대역폭들의 집합이라 하고, 공유대역폭에 관해 내림차순으로 정렬되어 있다고 가정하자. 공유AP s_1에 자식노드들이 할당되었고 자식노드들 중 가장 큰 요구대역폭을 $RB(s_1)$이라 하고, 공유AP s_2에 자식노드들이 할당되었고 자식노드들 중 가장 큰 요구대역폭이 $RB(s_2)$이라 하고, ... , 공유AP s_i에 자식노드들이 할당되었고 자식노드들 중 가장 큰 요구대역폭이 $RB(s_i)$이라 할 때, i가 $RB(s_1)$ + $RB(s_2)$ + ... + $RB(s_i)$ ≥ sb_{i+1}을 만족하는 가장 작은 자연수이면 공유AP s_{i+1}을 다음 레벨에서 사용하기 위해 비축해 둔다. 같은 방식으로 모든 클라이언트들이 공유AP들에

할당되어 공유대역폭 트리가 구축될 때까지 반복한다.

예를 들어, 서버 용량이 16이고 (sb_1, sb_2, sb_3, sb_4) = (16, 16, 16, 8), (cb_1, cb_2, cb_3, cb_4, cb_5, cb_6) = (8, 8, 8, 8, 4, 4)이라고 하자. 제안된 알고리즘을 적용해 보면 먼저 s_1에 c_1과 c_2를 할당하는데 $RB(s_1)$= 8이 된다. 그 다음 s_2에 c_3과 c_4를 할당하게 되는 데 $RB(s_2)$= 8이 되고, $RB(s_1) + RB(s_2) \geq sb_3$이므로 s_3를 다음 레벨에서 사용하게 비축해둔다. 그 다음 s_4에 c_5과 c_6를 할당하고, 비축해둔 s_3에 s_1과 s_2를 할당하게 된다. $RB(s_3)$= 8이고 $RB(s_4)$= 4이므로 그 합이 서버 용량 16을 넘지 않으므로 s_3와 s_4가 서버의 자식들이 되고, 고객의 요구를 모두 만족시키는 공유대역폭 트리를 구축할 수 있게 된다. 그 결과는 그림 2와 같고, 이 트리의 공유대역폭은 56이 된다.

제안된 알고리즘이 그림 3에 나타나 있는데 queueS와 queueC는 각각 공유AP와 클라이언트의 큐(queue)로 초기에는 공유대역폭과 요구대역폭의 내림차순으로 정렬되어 있다.

deleteC(S)와 insertC(S)는 각각 queueC(S)에서 노드를 삭제하고 큐에 노드를 삽입하는 함수이다. find_max_rb(s)는 공유AP s에 할당된 자식들 중에서 최대 요구대역폭을 찾아 공유AP s의 요구대역폭으로 할당하는 함수이다. sum_rb는 queueC에 남아있는 요구대역폭의 총합으로, 이 값이 서버의 용량보다 작거나 같으면 서버가 이 노드들을 직접 서비스하게 된다. sum_rb의 초기값은 모든 클라이언트의 요구대역폭의 합이 된다. Tree는 구축된 공유대역폭 트리의 에지들의 집합이고, TSB가 공유대역폭 트리가 사용한 총 공유대역폭이다.

```
Proposed Algorithm
Tree = ∅;
TSB = 0;
initialize queueS() and queueC();
sum_rb = sum of the requested bandwidth of
         customer nodes in C;
total = 0;
ci = deleteC();
sj = deleteS();
while (sum_rb > server capacity) {
   sum = 0;
   while (sum + cbi ≦ sbj) {
      sum = sum + cbi;
      Tree = Tree ∪ {<sj, ci>};
      sum_rb = sum_rb − cbi;
      ci = deleteC();
   }
   TSB = TSB + sbj;
   max_rb = find_max_rb(sj);
   sum_rb = sum_rb + max_rb;
   insertC(sj);
   total = total + max_rb;
   sj = deleteS();
   if (total ≧ sbj) {
      insertS(sj);
      sj = deleteS();
      total = 0;
   }
Tree = Tree∪{<server, ci>}∪
    {<server, ci>|∀ cbi in queueC};
return Tree, TSB
```

그림 3. 제안 알고리즘
Fig. 3. Proposed algorithm

알고리즘이 실행되는 과정을 보면 다음과 같다. 현재 사용가능한 공유AP s_j를 queueS에서 꺼내서, queueC에 있는 노드들을 순서대로 공유AP s_j의 공유대역폭을 넘지 않을 때까지 최대한 할당한다. 그 다음 공유AP s_j에 할당된 자식 중에서 가장 큰 요구대역폭(max_rb)을 구하는데, 이 값이 공유AP s_j의 요구대역폭이 된다. 공유AP s_j는 나중에 다른 공유AP나 서버의 자식이 되는 데, 이를 위해 queueC에 삽입된다. 자식이 할당된 공유AP들의 요구대역폭의 합(total)을 구하여 이 값이 queueS에 있는 다음 공유AP의 공유대역폭보다 크거나 같으면 다음 공유AP를 상위 레벨의 트리 구축에 사용하기 위해 비축해둔다. 이 과정을 반복하게 되는데 queueC에 남아 있는 요구대역폭의 합(sum_rb)이 서버의 용량을 넘지 않으면, 해당 노드들이 서버의 자식이 되고 공유대역폭 트리의 구축이 완료된다.

초기에 공유AP의 집합 S와 클라이언트의 집합 C가 내림차순으로 정렬되어 있다고 가정하고, 공유AP의 수를 n이라 하고 클라이언트의 수를 m이라 하면 제안된 알고리즘의 시간 복잡도는 O(n + m)이 된다.

IV. 성능 평가

제안된 알고리즘은 C 언어를 사용해서 Intel(R) Core i5 2 CPU를 가진 PC에서 구현되고 실험되었다. [17]에서 D 알고리즘과 I 알고리즘의 실험 결과가 비교되었는데 ,제안 알고리즘과 기존의 두 알고리즘과의 공정한 비교를 위해 실험은 [17]에서와 같은 조건으로 수행되었다. 즉, 서버 용량, 클라이언트의 수와 요구대역폭의 경우의 수, 공유AP의 수와 공유대역폭의 경우의 수를 고려하여 다양한 데이터를 생성하였다. 서버 용량은 그 크기에 따라 3가지 경우를 고려하였는데 타입

1은 서버 용량으로 가장 큰 공유대역폭의 2배를 사용하였다. 클라이언트 요구대역폭의 합을 x라 하였을 때, 타입 2는 서버 용량으로 x/4를, 타입 3는 서버 용량으로 x/2를 사용하였다. 그러므로 서버 용량이 클수록 클라이언트의 요구를 모두 만족시키는 트리를 구축할 확률이 높아지게 된다. 대역폭의 경우의 수는 4부터 10까지 7가지를 사용하였고, 같은 대역폭을 갖는 노드 수의 최대값은 7가지 경우(10, 20, 30, 50, 100, 200, 300)를 사용하였는데 같은 대역폭을 갖는 노드 수는 최대값 안에서 임의로 생성되었다. 또 두 조합 49가지 경우에 대해 각각 100개의 데이터를 생성하여 각 타입에 대해 4,900개의 데이터를 생성하였다.

1. 제안 알고리즘과 D 알고리즘의 비교

먼저 해를 찾은 성공률로 두 알고리즘을 비교하였는데 그 실험 결과가 표 1에 나타나 있다. 데이터가 임의로 생성되었기 때문에 해가 없는 데이터도 존재할 수 있으므로, 성공률은 제안 알고리즘 또는 D 알고리즘이 해를 찾은 데이터의 수를 기준으로 계산하였다. 그러므로 표 1에서 데이터의 수는 제안 알고리즘 또는 D 알고리즘이 해를 찾은 데이터의 총수이다.

표 1. 제안 알고리즘과 D 알고리즘의 성공률 비교
Table 1. Comparison of success rate between the proposed algorithm and D algorithm

타입	데이터의 수	제안 알고리즘	D 알고리즘
1	2,070	2,054	1,487
2	2,479	2,453	1,951
3	3,008	2,992	2,506
합 계	7,557	7,499	5,944
성공률		99.2%	78.7%

표 2 제안 알고리즘과 D 알고리즘의 승패 비교
Table 2. Number of wins, ties, and loses of the proposed algorithm over D algorithm

타입	데이터의 수	승	무승	패
1	2,070	640	1,028	402
2	2,479	716	1,339	424
3	3,008	806	1,728	474
합 계	7,557	2,162	4,095	1,300
비 율		28.6%	54.2%	17.2%

표 3. 제안 알고리즘과 I 알고리즘의 성공률 비교
Table 3. Comparison of success rate between the proposed algorithm and D algorithm

타입	데이터의 수	제안 알고리즘	I 알고리즘
1	2,234	2,054	1,991
2	2,594	2,453	2,343
3	3,098	2,992	2,792
합 계	7,926	7,499	7,126
성공률		94.6%	89.9%

표 4 제안 알고리즘과 I 알고리즘의 승패 비교
Table 4. Numbers of wins, ties, and loses of the proposed algorithm over D algorithm

타입	데이터의 수	승	무승	패
1	2,234	1,641	32	561
2	2,594	2,025	29	540
3	3,098	2,217	168	713
합 계	7,926	5,883	229	1,814
비 율		74.2%	2.9%	22.9%

제안 알고리즘은 총 7,557개 데이터 중 7,499개의 데이터에서 해를 찾아 99.2%의 성공률을 보였고, D 알고리즘은 5,944개의 데이터에서 해를 찾아 78.7%의 성공률을 보였다. 그러므로 D 알고리즘은 21.3%의 실패율을 보인 반면, 제안 알고리즘은 0.8%의 실패율을 보여 그 결과가 매우 우수함을 알 수 있다. 또 제안한 알고리즘은 모든 데이터 타입에 대해 대략 99%의 성공률을 보이고 있어 안정적이고, D 알고리즘은 서버의 용량이 작을 때(타입1) 성공률이 72%로 매우 낮고 서버의 용량이 커지면서(타입3) 83%로 타입에 따라 차이가 있었다.

그 다음에는 각 데이터에 대해 어느 알고리즘이 좋은 결과를 얻었는지를 비교하였다. 표 2에 제안 알고리즘을 기준으로 승패 결과가 나타나있다. 제안 알고리즘은 총 7,557개의 데이터 중 2,162(28.6%)개 데이터에서 D 알고리즘보다 더 좋은 결과를 보였고, 1,300(17.2%)개 데이터에서 더 나쁜 결과를 보였으며, 4,095(54.2%)개 데이터에서는 해가 같았다. 제안 알고리즘 과 D 알고리즘은 해의 결과가 같은 경우가 많았는데 이것은 D 알고리즘이 해를 찾는 성공률은 좋지 않지만 해의 결과는 제안 알고리즘과 비슷함을 나타내고 있다.

2. 제안 알고리즘과 I 알고리즘의 비교

제안 알고리즘과 I 알고리즘의 성공률 비교 결과가 표 3에 나타나 있다. 앞에서와 마찬가지로 7,926은 제안 알고리즘 또는 I 알고리즘이 해를 찾은 데이터의 총수이다. 제안 알고리즘은 총 7,926개 데이터 중 7,499개의 데이터에서 해를 찾아 94.6%의 성공률을 보였고, I 알고리즘은 7,126개의 데이터에서 해를 찾아 89.9%의 성공률을 보였다. 그러므로 I 알고리즘은 10.1%의 실패율을 보인 반면 본 논문에서 제안된 알고리즘은 5.4%의 실패율을 보여 그 결과가 우수함을 알 수 있다. 데이터 타입별로 성공률을 보면 제안 알고리즘은 ±2%이었고, I 알고리즘은 ±1%로 큰 차이가 없었다.

표 4에 제안 알고리즘을 기준으로 승패 결과가 나타나있다. 제안 알고리즘은 7,926개의 데이터 중 5,883(74.2%)개 데이터에서 I 알고리즘보다 더 좋은 결과를 보였고, 1,814(22.9%)개 데이터에서 더 나쁜 결과를 보였으며, 229(2.9%)개 데이터에서는 해가 같았다. 이 결과에서 보면 I 알고리즘이 해를 찾는 성공률에 비해 해의 결과가 좋지 않음을 알 수 있다.

결론적으로 제안 알고리즘이 D와 I 알고리즘에 비해 해를 찾는 성공률이나 해의 결과가 모두 우수하였다.

V. 결 론

최근에 공유 네트워크를 구축하여 사용자가 자신이 소유한 AP의 일부 대역폭을 다른 사람과 공유하는 방법이 대두되었는데, 이러한 공유 AP를 통해 어디서나 인터넷에 접근할 수 있고 데이터를 전송할 수 있어 그 효용성이 증가하는 추세이다. 또 공유 네트워크에서 응용 애플리케이션으로 SVC 기술을 사용하는 비디오 스트리밍 전송 시스템을 구축하는 방안이 제안되었다. 이 시스템은 서버로부터 모든 클라이언트에게 비디오 스트림을 보내기 위해서는 서버, 공유 AP, 클라이언트로 구성된 트리 구조를 만드는데 트리에서 사용되는 공유대역폭의 총용량을 최소화 하는 것이 바람직하다.

본 연구에서는 최소 공유대역폭 트리 구축 문제의 근사해를 구하기 위한 선형 알고리즘을 개발하고 기존에 발표된 알고리즘들과 비교하였는데, 실험결과에 의하면 제안된 알고리즘이 해를 찾는 성공률과 해의 결과 모두 우수하였다. 본 연구의 결과는 FON과 같은 공유 네트워크에서 SVC 코딩을 사용하는 비디오 스트리밍 전송에 유용하게 사용될 수 있다.

참고문헌

[1] FON official website: http://www.fon.com

[2] C. Su, Y. Hwang, C. Yeh, "A Study on the Willingness of Using FON in the Domain of Wireless Communication," 4th Intl. Conf. on Network Computing and Advanced Information Management, pp. 159-164, 2008

[3] A. Asheralieva, T. Erke, and K. Kilkki, "Traffic Characterization and Service Performance in FON Network," 1st Intl. Conf. on Future Information Networks, 2009

[4] H. Schwarz, D. Marpe, and T. Wieg, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," IEEE Trans. Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103-1120, 2007

[5] Q. Zhang, Q. Guo, Q. Ni, W. Zhu, and Y. Zhang, "Sender-adaptive and Receiver-driven Layered Multicast for Scalable Video over the Internet," IEEE Trans. Circuits and Systems for Video Technology, vol. 15, no. 4, pp. 482-495, 2005

[6] T. Kim and M. H. Ammar, "A Comparison of Heterogeneous Video Multicast Schemes: Layered Encoding or Stream Replication," IEEE Trans. Multimedia, vol. 7, no. 6, pp. 1123-1130, 2005

[7] G. Auwera and M. Reisslein, "Implication of Smoothing Multiplexing of H.264/AVC and SVC Video Streams," IEEE Trans. on Broadcasting, vol. 55 no. 3, Sep. 2009

[8] S. Sharangi, R. Krishnamurti M. ,Hefeeda, "Energy-Efficient Multicasting of Scalable Video Streams Over WiMAX Networks," IEEE Transactions on Multimedia, vol. 13, no. 1, pp. 102-115, 2011

[9] Sha Hua, Yang Guo, Yong Liu, Hang Liu and S. Panwar, "Scalable Video Multicast in Hybrid 3G/Ad hoc Networks," IEEE Trans. on Multimedia, vol. 13, issue 2, pp. 402-413, 2011

[10] D. Gomez-Barquero, K. Nybom, D. Vukobratovic and V. Stankovic, " Scalble Video Coding for

Mobile Broadcasting DVB Systems," ICME, pp. 510-515, 2010

[11] Won Sup Chi, Kwang-deok Seo, In Ki Lee, Dae-Ig Chang, "Joint source-channel coding scheme for SVC-based DVB-S2 satellite broadcasting system," 2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE), pp. 77-78, 2010

[12] C. Li, C. Yuan and Y, Zhong, "Robust and Flexible Scalable Video Multicast with Network Coding over P2p Network," 2nd Intl. Congress on Image and Signal Processing, pp. 1-5, 2009

[13] T. Kim and E. Kim, "A TTL-based Peer grouping Scheme for P2P Streaming Systems," Journal of the Korea Society of Computer Information, vol. 17, no. 1, pp. 151-159, 2012

[14] K. Chong, "An Efficient Algorithm for Finding the Earliest Available Interval on Connection Oriented Networks," Journal of the Korea Society of Computer Information, vol. 15, no. 3, pp. 73-80, 2010

[15] N.F. Huang, H.Y. Chang, Y.W. Lin, K.S. Hsu, and H.C. Liu, "On the Complexity of the Bandwidth Management Problem for scalable Coding Video Streaming on a Public-Shared Network," IEEE Communication Letters, vol. 13, no. 1, pp. 61-63, 2009

[16] N.F. Huang, H.Y. Chang, T.C. Wang, Y.S. Lin, Y.W. Lin, S.Y. Cheng, and J.J. Lin, "An Efficient and Locality-aware Resource Management Scheme for SVC-based Video Streaming System on a Public-Shared Network," Asia-Pacific Conf. on Communication, pp. 682-685, 2009

[17] K. Chong, "An Efficient Bandwidth Management Algorithm for SVC Video Streaming on Public-shared Networks," Journal of KIISE : Information Netwoking, vol. 38, no. 3, pp.243-247, June 2011

[18] E. Horowitz, S. Sahni, S. Rajasekaran, "Computer Algorithms C++," p. 769, W. H. Freeman and Company, New York, 1997

저 자 소 개

정 균 락

1980년 2월 : 한국과학기술원 전자계산학 석사

1991년 2월 : 미네소타대학교 컴퓨터공학 박사

1991년 ~ 현재 : 홍익대학교 컴퓨터공학과 교수

관심분야 : 네트워크 알고리즘, 이동통신, 무선 센서 네트워크, VLSI 알고리즘

Email : chong@cs.hongik.ac.kr