포워드캐스트(ForwardCast) : P2P에서의 새로운 VoD 스트리밍 방법

윤 수 미[†] · 김 상 철[††] · 김 중 환[††]

요 약

최근 P2P(Peer-to-Peer) 네트워크를 VOD(Video On Demand) 스트리밍에 적용하는 연구들이 활발히 발표되고 있다. VOD 스트리밍에 관한 기존 연구를 살펴보면, 클라이언트의 새로운 서비스 요청시, 초기 지연을 없애는 전송과 서비스 기각률을 최소화하는 것을 가장 중요한 목표로 삼고 있다. 본 논문은 이들 목표를 달성하기 위해서 P2P 기반의 멀티캐스팅 트리를 이용해서 효과적으로 비디오 스트리밍(streaming)을 제공하는 새로운 방법, 일명 포워드캐스트(ForwardCast)를 제안한다. 제안된 방법은, 새로운 클라이언트는 초기 지연없이(zero-delay) 비디오를 전달받고, 경우에 따라서는 데이터의 앞,뒤 부분을 동시에 전송받기도 한다. 실험 결과, 포워드캐스트는 기존 연구에 비해서 서버 부하를 증가시키지 않으면서 기각률을 낮출 수 있었다.

키워드 : 포워드캐스트, 피어 대 피어, 스트리밍, VOD

ForwardCast : A New VOD Streaming Method in P2P

Soo-Mi Yoon[†] · Sang-Chul Kim[††] · Joong-Hwan Kim[††]

ABSTRACT

Recently researches that apply P2P networks to VOD streaming have been actively published. In the previous works on VOD streaming, they aimed at achieving two major goals, which are zero-delay transmission and minimization of service rejection ratio. This paper proposes a new method, called ForwardCast, for VOD streaming based on a P2P-based multicasting tree in order to achieve theses two goals. In this method, basically a new client selects one of the preceding clients and starts receiving a whole video from the selected one without any delay. In some situation, two preceding clients are selected to transfer the ending part of the video and its the remaining part simultaneously. In our experiment, ForwardCast can reduce the rejection ratio compared to previous works without increasing server stress.

Key Words : ForwardCast, P2P, Streaming, VOD

1. 서 론

네트워크의 발전으로 아날로그에서 디지털 세계로의 변화는 디지털 콘텐츠 산업의 성장으로 이어지게 되었다. 디지털 콘텐츠를 네트워크에서 제공하는 서비스 중에서 특히 인터넷을 통한 스트리밍(streaming) 서비스에 대한 요구가 폭발적으로 증가하고 있다[1]. 비디오 스트리밍 서비스의 형태는 실시간 전송과 VOD(Video on Demand) 전송으로 나눌 수 있는데, 이 중 후자가 보다 일반적인 서비스 형태이다[2]. VOD 서비스는 높은 대역폭의 데이터를 비교적 오랜 시간 제공하여야 하는 속성 때문에 비디오 서버와 전송 네트워크에 부담을 준다. 특히 중앙 집중 방식인 경우, 비디오 서버의 처리 및 전송 능력이, 동시에 서비스 가능한 클라이언트의 수를 결정하는 한계가 있다. 이와 같은 이유에서, 최근 VOD 스트리밍에 관한 연구는 비디오 데이터를 여러 서버에 분산 배치하는 방법과 네트워크 대역폭(network bandwidth)을 효과적으로 사용하기 위한 멀티캐스팅(multicasting) 분야에서 활발히 이루어지고 있다[1, 2, 3].

우리의 조사에 의하면, VOD 서비스에 대한 기존 연구들이 달성하려는 목표들 중에 중요한 것이 초기 지연 없는 전송과 서비스 요청 기각률(rejection ratio)의 최소화로 볼 수 있다. 초기 지연 없는(zero-delay) 비디오 데이터 전송이란 클라이언트가 서버에 자료를 요청하면 일정한 시간을 기다려야 한다는 조건 없이 즉시 스트리밍이 시작되는 것을 말한다. 서비스 요청에 대한 기각은 비디오 서버에 과부하(overload)가 발생하거나 네트워크 대역폭의 여유분이 없으

※ 이 연구는 2005학년도 한국외국어대학교 교내학술연구비의 지원에 의하여 이루어진 것임.
† 준 회 원 : 장안대학 컴퓨터정보계열 겸임교수
†† 정 회 원 : 한국외국어대학교 컴퓨터공학과 교수
논문접수 : 2005년 7월 19일, 심사완료 : 2005년 11월 2일

면 발생하게 된다.

본 논문은 위에서 언급한 목표들을 달성하고자 P2P(Peer-to-Peer) 기반의 멀티캐스팅 트리를 이용해서 효과적으로 비디오를 스트리밍하는 방법, 일명 포워드캐스트(ForwardCast)를 제안한다. P2P 기반의 멀티캐스팅이란 비디오 시청을 요구하는 피어들(클라이언트들 또는 노드들)이 멀티캐스트 트리(multicast tree)를 형성하는 것을 말한다. 멀티캐스트 트리의 루트(root) 노드는 비디오 서버가 되며 각 클라이언트는 비디오 스트림을 부모 노드로부터 전송 받아서 재생(playback)함과 동시에 자식 노드에게 전달하는 역할을 수행한다.

멀티캐스팅에 관한 연구는 주로 IP 기반 네트워크상의 멀티캐스팅[5, 6, 16, 17]에서 이루어 졌다. IP 기반 멀티캐스트 네트워크가 아직 일반화되어 있지 않고 클라이언트들의 처리 능력이 증대됨에 따라 최근 비디오 스트리밍 분야에서는 P2P 기반의 멀티캐스팅에 대한 연구가 활발히 진행되고 있다. 참고로, P2P 기반 멀티캐스팅은 응용수준(application-level)의 멀티캐스팅이라고 불린다. 지금까지 P2P 개념을 VOD 스트리밍 서비스에 적용시킨 기존 연구[3, 8, 10, 11, 13, 14]가 발표되었지만 대부분 문제점을 가지고 있다. [10]은 P2P 개념을 VOD 서비스에 도입한 초기 연구로서, 초기 지연시간이 있으며 클라이언트들 간의 네트워크 대역폭(bandwidth)에 대한 고려를 하지 않았다. [3]은 중앙 집중 방식으로 클라이언트들과 비디오 서버들을 관리하여 P2P의 특징을 충분히 활용하지 못하고 있고, 여러 클라이언트들이 한 클라이언트에게 동시에 비디오를 전송하는 [8]이나 비디오의 복수 코딩(multiple coding)을 사용하는 [13]은 그 동작 원리가 복잡한 단점이 있다. 또한, [14]는 비디오 초기 부분을 클라이언트들에게 캐싱(caching)하는데 걸리는 초기 지연이 존재한다.

본 논문에서 제안하는 포워드캐스트 방법의 동작원리를 간단히 설명하면 다음과 같다: 현재 스트리밍이 진행 중인 상황에서, 뒤에 도착한 클라이언트는 선행 클라이언트들(자신 보다 앞에 도착한 클라이언트) 중에 하나를 선택해서 전체 비디오를 전송받는다. 만약 네트워크 대역폭의 제약 때문에 이런 선행 클라이언트를 검색할 수 없다면, 두 선행 클라이언트를 선택해서 비디오 데이터의 앞부분과 뒷부분을 동시에 전송받을 수 있다. 기존 연구 중에서 포워드캐스트와 비교할 연구는 [11]로 볼 수 있다. 비디오서버로부터 스트림을 전송받는 클라이언트들은 멀티캐스팅 트리를 형성한다. 대부분의 경우, 새 클라이언트는 멀티캐스팅 트리에 참여하여 자신의 부모노트로부터 이미 진행 중인 스트림을 전송받기 시작한다. 즉, 전체 비디오의 처음부터가 아니라 참여 시점부터 전송받기 때문에 처음부터 참여시점까지의 비디오 데이터는 다른 노드로부터 별도로 전송받아야 하는데, 이런 비디오 데이터를 패치(patch)라 부른다. 따라서 전체 비디오 데이터를 패치 스트림과 나머지 스트림으로 나누어 동시에 전송받는다. [11]에서는 비디오 데이터를 패치스트림과 나머지 스트림으로 나누어 동시에 전송받는다.

본 실험에 의하면, 포워드 캐스트는 기존 연구[11]과 비교할 때 서버 부하 면에서는 큰 차이를 보이지 않지만 서비스 요청의 기각률을 낮출 수 있음을 볼 수 있었다. 본 논문은 다음과 같이 구성되어 있다. 2장에서는 문제 정의와 본 연구에서 제안하는 전송방법과 알고리즘에 대하여 설명한다. 3장에서는 전송트리의 재구성과 장애에 관하여 기술하며, 4장에서는 실험을 통하여 제안된 연구의 성능을 확인한다.

2. 포워드캐스트의 개요

2.1 용어 정의

본 논문에서 설명의 편의를 위해서 다음과 같은 기호를 정의한다:

$T_0(C_i)$ = 노드 C_i에 서비스를 시작한 후 경과된 시간

$B(C_i)$ = 노드 C_i의 버퍼 사이즈(단위는 시간임)

$T_s(C_i)$ = 노드 C_i의 서비스 시작 시점(실시간임) 또는 클라이언트 C_i의 서비스 요청시간. 단, C_i의 요청이 수락된 후 서비스의 시작까지 발생하는 지연시간은 무시할 정도라고 본다.

V_L = 비디오 V의 길이 (단위는 시간임)

$V[T_1, T_2]$ = 비디오 V의 시점 T_1에서 T_2까지의 부분. 전체 비디오는 $V[0, V_L]$로 표현.

$T_A(C_i, C_j) = T_s(C_j) - T_s(C_i)$ C_i의 서비스 시작시점과 C_j 서비스 시작시점의 차이(gap), C_i가 부모노드로부터 전달받은 스트림을 C_j에게 전송할 때, C_i가 C_j에게 자신의 버퍼에 있는 데이터를 전송하는 데 걸리는 시간을 나타냄. 노드 C_i의 C_j에 대한 잔여시간.

2.2 포워딩 서버(Forwarding Server)의 조건

VOD 서비스를 받고 있는 클라이언트들은 멀티캐스팅 트리를 형성하게 된다. 그 트리의 루트 노드는 비디오 서버(S)가 되고, 루트가 아닌 각 노드는 부모 노드로부터 비디오 스트림을 전송받으면서 동시에 자식 노드들에게 자신의 버퍼에 캐싱된 비디오 데이터를 스트리밍하는 일을 수행한다. [11]에서와 같이 각 노드는 자신이 수신했던 비디오 데이터를 FIFO 방식으로 버퍼링한다.

새롭게 도착한 클라이언트는 멀티캐스팅 트리상의 한 노드를 부모 노드로 선택하여 그 노드로부터 비디오를 처음부터 전송받게 된다. 부모 노드가 비디오 서버가 아닌 경우에는, 자신이 이미 전송받았던 비디오를 다시 자식노드에게 전달하기 때문에 그런 부모 노드를 자식 노드의 포워딩 서버라고도 부른다. 각 노드의 부모 노드가 비디오 서버일 수도 있지만, 대부분 자신보다 앞서 도착한 클라이언트들 중의 하나이다.

어떤 클라이언트 C_{new}에 대해서, 전에 도착했던 노드 C_p가 C_{new}의 부모 노드가 되려면 다음 세 가지 조건을 만족하여야 한다.

[조건 1] C_p와 C_{new} 사이에 충분한 네트워크 대역폭이 존재함.

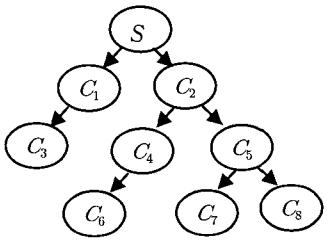
[조건 2] $T_0(C_p) \leq B(C_p)$

[조건 2]가 만족되지 못한다면 노드 C_p의 버퍼가 비디오의 시작 부분을 갖지 못하기 때문이다. 노드 C_p가 노드 C_{new}에게 비디오 전송을 끝내는 시점은 '$T_s(C_{new}) + V_L$'이 되고, 그 시점까지 멀티캐스팅 트리 상에 남아 전송을 책임져야 한다. 만약 노드 C_p가 자식노드를 갖지 않는다면, C_p는 자신에게 전달되는 스트림이 끝나는 시점인 '$T_s(C_p) + V_L$'까지만 멀티캐스팅 트리 상에 살아있으면 된다. 그러나 C_p가 C_{new}에게 스트림 전송을 완성하려면, $T_A(C_p, C_{new})$만큼 더 연결을 유지하여야 하는데, 이런 시간을 노드 C_p의 C_{new}에 대한 잔여 시간이라 부른다. 따라서, 클라이언트 C_p가 C_{new}의 부모 노드가 되려면, [조건 3]이 추가 되어야한다:

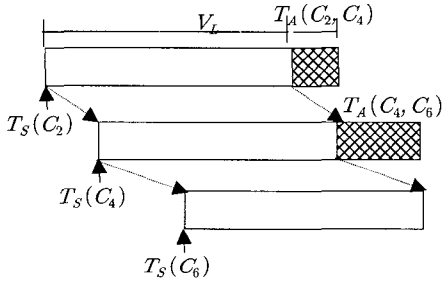
[조건 3] $T_A(C_p, C_{new}) \leq \Delta FT$

여기서, ΔFT는 $B(C_p)$보다는 작은 값으로서 노드 C_p의 파라미터이다. ΔFT는 C_p의 사용자가 비디오 재생이 끝난 후 비디오 스트리밍에 추가로 참여할 수 있는 최대 시간을 나타낸다. 파라미터 ΔFT는 각 노드의 사용자마다 별도의 값을 설정할 수 있겠지만, 본 논문에서는 모든 노드가 동일한 값을 갖는다고 가정한다. 참고로, 경우에 따라서는 부모 노드로 선택되기 위해서 [조건 1]과 [조건 2]만을 만족하면 되는데, 이런 경우는 2.3절에서 설명한다.

(그림 1)은 멀티캐스팅 트리의 예를 보여 준다. 클라이언트 C_2은 비디오 서버 S로부터 스트림을 전송받으면서, C_4와 C_5의 포워딩 서버로서 스트림을 전송하고 있다. C_6는 자신의 포워딩 서버인 C_4로부터 스트림을 받는다. (그림 2)는 C_2, C_4 및 C_6의 스트림 시작 시간과 스트림의 길이, 잔여시간 등을 보여 주는데, 이들은 각각 $T_s(C_2)$, $T_s(C_4)$ 및 $T_s(C_6)$시점



(그림 1) 멀티캐스팅 트리의 예



(그림 2) 스트림 전송

에서 비디오의 처음부터 전체 비디오를 전송받고 스트림을 전송한다.

2.3 포워딩 서버(Forwarding Server)의 선택

새로운 스트리밍 서비스를 받기 위해서 클라이언트는 멀티캐스팅 트리에서 포워딩 서버가 가능한 부모 노드를 정해야 한다. 이를 위해서 새로운 클라이언트는 먼저 비디오 서버에 접속한다. 비디오 서버는 다음과 같은 과정을 거쳐서 부모 노드를 선택하도록 한다.

Step 1 : 2.2절에서 언급한 세 가지 조건을 만족하는 기존 노드들 중에서 버퍼의 사용되지 않는 여유 공간이 가장 작은 노드를 부모 노드로 선택한다. 만약 위의 3가지 조건을 만족하는 노드가 없다면 Step 2를 실행한다.
Step 2 : 비디오 서버와 새로운 클라이언트간의 충분한 대역폭이 존재하면 비디오 서버를 부모 노드로 선택한다. 그렇지 않으면 Step 3을 수행한다.
Step 3 : [조건 1]과 [조건 2]를 만족하지만 [조건 3]을 만족하지 못하는 기존 노드들 중에서 버퍼의 사용되지 않는 여유 공간이 가장 작은 노드를 부모 노드로 선택한다. 또한, 그 부모 노드가 전송하지 못하는 비디오의 뒷부분(suffix)을 전송할 수 있는 다른 노드도 선택한다. 만약 해당되는 두 노드들을 찾지 못하면 그 클라이언트의 스트림 서비스 요구는 기각된다.

(그림 3) 부모 노드 선택 알고리즘

Step 1에서 [조건 2]를 만족하는 노드의 버퍼는 이제까지 전송된 비디오 데이터를 저장하고 있는 캐싱 공간과 사용되지 않고 있는 여유 공간으로 나누어진다. 클라이언트 C_i에 대해서, C_i의 사용되지 않고 있는 여유 공간의 크기 $U(C_i)$는 다음과 같이 구할 수 있다:

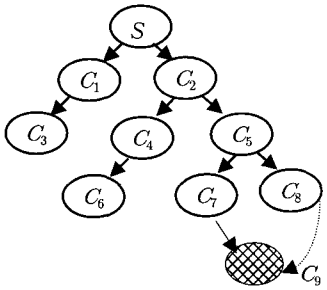
$$U(C_i) = B(C_i) - T_0(C_i)$$

다른 노드들에 비해서, $U(C_i)$가 작은 노드는 가장 가까운 미래에 [조건 2]를 만족하지 못하게 된다. 이런 노드를 부모 노드(즉, 포워딩 서버)로 선택해서 네트워크 대역폭을 먼저 할당하면, 다른 노드들이 뒤에 도착할 클라이언트들의 부모로 선택될 가능성을 높여준다.

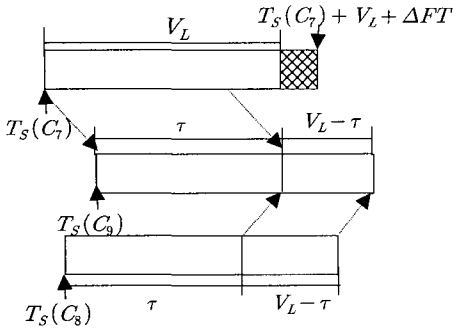
Step 3에서 선택되는 부모 노드 C_p는 [조건 3]을 만족하지 않기 때문에 새 클라이언트 C_{new}에게 '$T_s(C_p) + V_L - \Delta FT$'시점까지만 비디오를 전송한다. 그러므로 C_p는 $V[0, \tau]$를 새 클라이언트에게 전송할 수 있다. 여기서, '$\tau = V_L + \Delta FT - T_A(C_p, C_{new})$'이다. 비디오의 뒷 부분인 $V[\tau, V_L]$은 또 다른 노드를 통해서 전송받아야 하는데, 본 연구에서는 이런 노드를 서픽싱(suffixing) 서버라고 새롭게 정의한다. 부모 노드와 서픽싱 서버는 동시에 새 클라이언트에게 비디오를 스트리밍 서비스하게 된다.

클라이언트 C_{new}에 대해서 기존의 노드들 중 어떤 노드가 서픽싱 서버 C_q가 되려면 [조건 1]과 [조건 4]를 만족하여야 한다:

[조건 4] $\tau \leq T_0(C_q) \leq \tau + \Delta FT$



(그림 4) C_9의 포워딩 서버와 서픽싱 서버



(그림 5) 스트림 전송 관계

이 조건은 서픽싱 서버 C_q가 현재 자신의 버퍼에 $V[\tau, V_L]$의 전부 또는 앞 부분만을 가지고 있으면서, C_{new}에게 $V[\tau, V_L]$을 전송해 줄 수 있음을 나타낸다.

(그림 4)와 (그림 5)는 새로운 클라이언트 C_9가 멀티캐스팅 트리에 참여하여 C_7을 포워딩 서버로 정하고, 서픽싱 서버를 C_8로 선택한 후 스트림을 전송받는 내용이다.

3. 포워드캐스트의 세부 알고리즘

3.1 분산처리 방식의 부모노드 선택

2장의 Step 1은 비디오 서버가 혼자 수행하지 않고 분산처리 방식으로 수행한다. 새롭게 도착한 클라이언트 C_{new}에 대해서, 비디오 서버는 자식노드들에게 GET_NODE라는 메시지를 보낸다. (그림 6)은 부모노드로부터 GET_NODE 메시지를 받은 노드 C_p의 역할을 기술하고 있다.

Step 1.0 : 집합 $S = \{\ \}$
Step 1.1 : 노드 C_{new}에 대해서, C_p가 [조건 1],[조건 2] 및 [조건 3]을 만족하면, $S = S \cup \{[C_p,\ U(C_p)]\}$.
C_p가 한 개 이상의 자식노드를 가지고 있다면 Step 1.2를 수행하고, 그렇지 않다면 Step 1.3을 수행한다.
Step 1.2 : C_p의 자식노드는 D_i, $1 \le i \le N$ 이다. N는 C_p의 자식노드 수이다.
각 D_i에게 GET_NODE 메시지를 보내서 결과값 R_i를 받은 후, $S = S \cup R_i$
Step 1.3 : S가 공집합이면, 공집합을 C_p의 부모노드에게 결과값으로 반환한다. 그렇지 않으면, S의 원소들 중 U 값이 가장 작은 원소를 C_p의 부모노드에게 결과값으로 반환한다.

(그림 6) GET_NODE 메시지의 처리 과정

GET_NODE 메시지를 보낸 후, 비디오 서버가 모든 자식노드들로부터 받은 결과값이 공집합이면, 2.2절의 3가지 조건을 만족하는 노드가 현재 멀티캐스팅 트리 상에는 없다는 뜻이다. 그렇지 않다면, U값이 가장 작은 결과값에 명시된 노드가 C_{new}의 부모 노드가 된다. 예를 들면, 비디오 서버가 자식들로부터 받은 결과값 중에서 $[C_5, 12]$가 U값(이 경우에는 12)이 가장 작은 경우라면, C_5가 새 클라이언트 C_{new}의 부모가 된다.

Step 1.3에서 U값이 같은 경우, 새 클라이언트와 네트워크상 거리가 가까운 노드가 선택된다. 예를 들면, S의 원소들 중 U값이 가장 작은 원소가 $[C_1, 15]$와 $[C_2, 15]$인 경우. C_1와 C_2 중에서 C_{new}와 가까운 노드가 선택된다. 이때, 두 노드간의 네트워크상 거리는 홉(hop) 수로 계산한다. 홉은 인터넷 라우팅 프로토콜에 의해 사용되는 측정 단위로서 최종목적지까지 거쳐가게 되는 경로상의 네트워크(또는 라우터)를 의미한다[18].

3.2 부모노드와 서픽싱 서버의 선택

본 절은 (그림 3)의 Step 3에 대한 추가적인 설명이다. Step 3에서 부모노드를 선택하는 방법은 3.1절에서 언급한 것과 같은 분산처리 방법을 사용한다. 차이점은 Step 1.1에서 [조건 1]과 [조건 2]만을 검사하는 것이다. 서픽싱 서버의 선택 방법도 부모노드 선택과 같이 3.1절에서 언급한 것과 같은 분산처리 방법을 사용한다. 차이점은 Step 1.1에서 [조건 1]과 [조건 4]를 검사하는 것이다.

3.3 장애(Fault)의 처리

스트림 전송과정에서 멀티캐스팅 트리상의 어떤 노드가 갑작스런 장애로 네트워크를 벗어나는 경우, 그 장애 노드의 모든 자손노드(descendant node)는 영향을 받는다. 영향을 최소화하려면 장애 발생 즉시, 장애 노드의 모든 자식노드들은 새로운 부모노드를 찾아서 스트리밍이 계속 진행되도록 해야 한다. 멀티캐스팅 트리상의 노드는 해당 시간 내에 스트림이 도착하지 않을 경우에 자신의 부모 노드가 장애라는 사실을 알아낼 수 있다. 장애에 대한 처리는 기본 연구 [11]의 패치스트림 회복(patch stream recovery)과 유사한 개념으로 장애 노드의 각 자식노드들만 새로운 부모를 선택한다. 이들은 먼저 비디오 서버에 접촉해서 새로운 부모 노드를 선택하는 작업을 시작한다. 이 작업은 새로운 클라이언트가 부모노드 선택을 하는 분산처리과정과 유사하다. 단지 어떤 노드가 장애노드의 새 부모노드가 되려면, [조건 2]대신에 장애 발생 이후의 비디오 데이터 부분을 현재 버퍼에 가지고 있는 조건을 만족하여야 한다.

4. 실험과 분석

4.1 시뮬레이션 환경

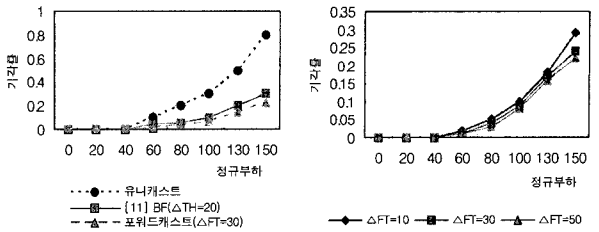
본 시뮬레이션 환경은 기존 연구 [11]과 같이 구성하였다. 실험에 사용될 네트워크는 GT-ITM[15]를 사용해서 생성하

였는데, 3단계 구조를 가진다. 중심부는 4개 노드로 구성되는 트랜짓(transit) 네트워크와 12개로 구성되는 스텁(stub) 도메인이 있고, 트랜짓 네트워크와 스텁 도메인의 노드들은 120 개로 하였다. 두 노드간의 채널은 네트워크 대역폭이 허락하는 경로들 중에 가장 작은 홉 수의 것으로 정하였다. 또한, 서버는 임의의 노드가 될 수 있다. 본 연구의 시뮬레이션에서는 기존 연구에서와 같이 한 개의 비디오 서버가 존재하는 것으로 제한하고, 비디오 스트림의 재생률은 CBR (Constant Bit Rate)로 가정한다. 서버의 위치는 트랜짓 도메인 내로 설정하였다. 한편, 비디오 서비스를 요청하는 클라이언트들의 빈도는 포아슨(Poisson)분포를 따른다. 비디오 스트림의 재생 길이는 100(분)으로 설정한다.

4.2 실험 결과

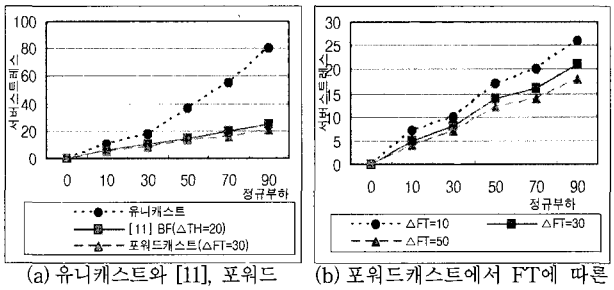
(그림 7)과 (그림 8)은 본 연구에서 제시한 방법, 유니캐스트(unicast) 및 기존 연구 [11]를 기각률과 서버 스트레스 면에서 각각 비교한 결과이다. 먼저, 유니캐스트란 각 클라이언트의 요구에 대해서 비디오 서버가 개별 스트림을 1:1로 전송하는 방식을 말한다. 기존 연구 [11]은 멀티캐스트를 이용한 스트림 전송방법으로서 서비스를 요청하는 대부분의 클라이언트는 초기에 두 개의 스트림 즉 패칭 스트림과 공유 스트림을 동시에 전송받아야 한다. 포워드캐스트는 기본적으로 한개의 포워드 전송으로 모든 스트림을 전송받지만, 필요시 서픽싱 서버로부터 나머지 뒷부분의 스트림을 전송받을 수 있다. 이들 그림에서 정규부하(normalized workload)란 시뮬레이션 실행동안 스트리밍 서비스를 요구하는 전체 클라이언트의 수를 의미하며, 평균 도착률 * 시뮬레이션 실행시간 으로 계산할 수 있다. 서버 스트레스(sever stress)란 비디오 서버가 전송하는 스트림의 평균 개수를 의미한다. 실험에서 각 클라이언트의 버퍼 크기는 0과 ΔFT 사이의 임의의 값을 갖는다. 실험 결과를 요약하면, 서버 스트레스는 기존 연구[11]와 큰 차이가 없지만 기각률은 유리하게 나타남을 알 수 있다. 그리고, 포워드캐스트에서는, 파라미터 ΔFT이 커짐에 따라서 기각률과 서버 스트레스가 조금씩 낮아짐을 알 수 있었다. 그 이유는 그 파라미터가 커짐에 따라서 [조건 3]이 보다 쉽게 만족되기 때문이다.

포워드캐스트가 기존 연구[11]보다 낮은 기각률을 보장하는 주된 이유는 다음과 같이 설명할 수 있다. 대부분 경우에 포워드캐스트는 새 클라이언트를 위해서 한 개의 스트림 채널만 할당하면 된다. 반면에 [11]에서는 초기에 항상 두 개의 비디오 스트림을 동시에 전송하여야 하기 때문에 두 배의 네트워크 대역폭을 새 클라이언트를 위해서 할당해야 한다. 포워드캐스트의 또 다른 장점은 버퍼링에 대한 부담이 적다는 것이다. 기존 연구에서는 새 클라이언트는 지나간 데이터 분량만큼의 기본 스트림을 저장할 버퍼가 반드시 필요하다. 반면에 포워드캐스팅에서는 이런 제약이 없다. 즉, 어떤 노드가 포워딩 서버로부터만 스트림을 전송받고 자신이 다른 클라이언트의 포워딩 서버 역할을 하지 않으면, 데이터 버퍼링이 필요없다. 결론적으로, 본 논문에서 제안한 방법은 각 노드의 저장 공간을 효율적으로 이용하면서 서버와 노드들의 네트워크 부하를 분산시킴으로서 기각률을 낮추고 기존연구와 같이 초기 지연 없는 서비스를 수행할 수 있었다.



(a) 유니캐스트와 [11], 포워드캐스트의 기각률 비교 (b) 포워드캐스트에서 FT에 따른 기각률의 비교

(그림 7) 기각률의 비교



(a) 유니캐스트와 [11], 포워드캐스트의 서버스트레스비교 (b) 포워드캐스트에서 FT에 따른 서버스트레스의 비교

(그림 8) 서버스트레스의 비교

5. 결 론

클라이언트의 VOD 서비스 요청시, 가장 중요한 목표로 삼고 있는 것이 초기 지연 없는 전송과 서비스 기각률의 최소화이다. 본 연구에서는 이런 요구사항을 지원하기 위해서 P2P 기반의 멀티캐스팅 트리를 이용하는 새로운 비디오 스트리밍 방법으로서 포워드캐스트를 제안하였다. 실용화된 IP 멀티캐스팅 네트워크의 수가 제한되어 있고 클라이언트 컴퓨터 성능의 향상으로 인해서, P2P 기반의 멀티캐스팅 네트워크를 VOD 스트리밍에 적용하는 시도가 최근 활발히 진행되고 있지만, 본 연구에 의하면, 기존 P2P 기반 멀티캐스팅에 관한 스트리밍 방법은 앞에서 언급한 요구사항들을 효과적으로 지원하지 못한다. 포워드캐스트에서는 새로 도착한 클라이언트가 대부분의 경우, 네트워크 대역폭의 여유가 있는 선행 클라이언트들 중 하나를 포워딩 서버로 선택해서 비디오 전송을 전달받는다. 경우에 따라서는 비디오의 뒷부분을 서픽싱 서버로부터 데이터를 동시에 받는다. 기존 연구와 비교해 보면, 포워드캐스트는 초기 지연시간이 없는 장점은 살리면서 네트워크 대역폭을 효과적으로 분산시킴으로써 서버 부담과 서비스 요청 기각률을 줄일 수 있었다.

참 고 문 헌

[1] Due A. Tran, Kien A. Hua, Tai T. Do, "A Peer-to-Peer

Architecture for Media Streaming," in IEEE Journal on Selected Areas in Communication, Special Issue on Advances in Overlay Network, 2003.
[2] San Jose, "A measurement study of peer-to-peer file sharing systems," in Proc. of ACM/SPIE on Multimedia Computing and Networking(MMCN'02), CA, USA, January, 2002.
[3] Yang Guo, Kyounwon Suh, Jim Kurose, and Don Towsely, "A Peer-to-Peer On-Demand Streaming Service and Its Performance Evaluation," in Proc. of IEEE Int. Conf. on Multimedia Expo(ICME'03), 2003.
[4] H. Despande, M. Bawan and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," in Work at CS-Stanford. Submitted for publication, 2002.
[5] Y. Guo, L. Gao, D. Towsley, and S. Sen, "Seamless workload adaptive broadcast," in Proc. of International Packet Video Workshop, April, 2002.
[6] D. Eager, M. Vernon, and J. Zahorjan, "Bandwidth skimming : A technique for cost-effective VOD," in Proc. SPIE/ACM Conference on Multimedia Computing and Networking, January, 2000.
[7] "Cost Effective and Scalable Video Streaming Techniques," in Borko Fuht, Oge Marques(eds.): Handbook of Video Database(to appear200/2003) CRC press, 2002.
[8] "Dongyan XU, Mohamed Hefeeda, Susanne Hambrusch, and Bharat Bhargava, On peer-to-peer media streaming," in Proc. of IEEE Conference on Distribured Computing and System, pp.363-371, July, 2002.
[9] J. Jannotti,D. Glifford, K. Johnson, M. Kaasho, and J. O'Toole, "OverCast: Reliable multicasting with an overlay network," in Proc. of the Fourth Symposition on Perating Sytems Design and Implementation, pp.197-212, 2000.
[10] Ottaw, Ontario, "Chaining: A Generalized Batching Technique for VOD System," in Proc. of the IEEE Int'l Conf. On Multimedia Computing and System, Canada, pp.110-117, June, 1997.
[11] Yang Guo, Kyoungwon Suh, Jim Kurose, "P2Cast: Peer-to-Peer Pathing Scheme for VOD Service," in ACM WWW 2003.
[12] Sridhar Ramesh, Injong Rhee, Katherine Guo, "Multicast with Cache(Mcache): An Adaptive Zero-delay Video-on-Demand Service," in IEEE INFOCOM, Vol.1, pp.22-26, Apr., 2001.
[13] M. Castro, P. Drushel, A-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth content distribution in a coorperative environment," in Proc. IPTPS'03, February, 2003.
[14] Tai T. Do, Kien A. Hua, Mounir A. Tantaoui, "P2VOD: Providing Fault Tolerant VOD Streaming in Peer-to-peer Environment," in tech. rep. 2003, SEECS. UCF, http://www.cs.ucf.edu/tdo/
[15] Ellen W. Zegura, Ken Calvent, and S. Bhattacharjee, "How to model an Internet," in IEEE INFOCOM, Sanfrancisco, CA, 1996.
[16] K. Hua, Y. Cai and S. Sheu, "Patching: A multicast technique for true Video-on-Demand services," in Proc. ACM Multimedia, September, 1998.
[17] L.Gao and D. Towsley, "A Multicast technique for true video-on-demand services," in Proc. of the sixth ACM international conf. on Multimedia, pp.191-200, September, 13-16, Bristol, United Kingdom, 1998.
[18] Comer, D. E. [2000], Internet Book, 3rd edition, Prentice-Hall, Upper Saddle River, NJ.

윤 수 미

e-mail : yoonsm98@hanmail.net
1992년 한국외국어대학교 정보산업공과대학 컴퓨터공학전공(학사)
1995년 홍익대학교 교육대학원 전산교육 전공(석사)
2001년 한국외국어대학교 컴퓨터공학과 박사과정 수료
2002년~현재 장안대학 컴퓨터정보계열 겸임교수
관심분야 : 멀티미디어 네트워크, P2P 네트워크, 네트워크 게임

김 상 철

e-mail : kimsa@hufs.ac.kr
1983년 서울대학교 컴퓨터공학과(학사)
1994년 미시간 주립대학교 컴퓨터공학과 (박사)
1983년~1994년 한국전자통신연구원 연구원
1994년~현재 한국외국어대학교 컴퓨터공학과 교수
관심분야 : 멀티미디어 시스템, 네트워크 게임, 정보검색

김 중 환

e-mail : jhkim@hufs.ac.kr
1970년 육군사관학교
1974년 서울대학교 공과대학 응용수학과 (학사)
1977년 고려대학교 대학원 산업공학과 (석사)
1984년 고려대학교 산업공학과(박사)
1974년~1984년 육군사관학교 수학과 부교수
1985년~현재 한국외국어대학교 컴퓨터공학과 교수
관심분야 : 소프트웨어공학, 성능평가, 영상처리