오버레이 멀티캐스트 기반 개선된 전송트리를 이용한 미들웨어의 연구

김충련*, 최성욱**

요약

오버레이 멀티캐스트는 IP멀티캐스트에 비해 대역폭 사용이 비효율적이고, 물리적 토플로지와 오버레이 토플로지의 불일치에 기인하는 지연(delay)은 피할 수가 없다. 따라서 그룹에 가입, 탈퇴가 동적으로 이뤄지는 상황에서는 안정성의 확보 노력이 필요하다. 이를 위해 본 연구에서는 오버레이 멀티캐스트를 기반으로 하여 클라이언트와 서버간의 통신 서비스를 관리하는 미들웨어를 제시하였다. 제안한 미들웨어는 효과적인 전송트리를 구성하였으며 서버와 클라이언트를 구조화 하여 단점을 개선하였다. 실험 내용으로는 오버레이 멀티캐스트 상에서 멀티미디어(이미지) 파일들을 동일한 조건으로 전송하였고 실험 결과로는 기존 시스템보다 제안 시스템이 약 20%정도의 성능 향상과 2%~4% 내의 패킷 전송 손실율을 감소시킬 수 있었다. 이는 전송 트리의 안정성 확보와 효과적인 스케줄링과 커넥션 등도 보장됨을 알 수 있었다.

Middleware Using Improved Transmission Tree Based on Overlay Multicast

Chung-Ryeon Kim*, Sung-Uk Choi**

ABSTRACT

Compared to IP multicast, overlay multicast is inefficient for the use of bandwidth, and the delay caused by disagreement between physical topology and overlay topology is unavoidable. Therefore, in the case of the situation in which subscription to and withdrawal from the group are dynamically realized, the effort of securing stability is required. For the sake of this, from this study, a middleware which manages communication service between client and server based on overlay multicast was presented. Proposed middleware constituted an effective transmission tree, and the weak point was improved by structuralizing server and client. As for the details of test, voice and image files were transmitted with identical condition on overlay multicast, and as for the test result, it could be confirmed that the performance of proposed middleware was improved around 20% compared to existing middleware. It could also be understood that securing stability of transmission tree, effective scheduling and connection, and so forth were also guaranteed.

Key Words : ip multicast, overlay multicast, middleware, traffic agent, topology manager

* 인천대학교 일반대학원 컴퓨터공학과(✉ kcr@hkt.co.kr)
** 인천대학교 컴퓨터공학부 교수
· 제1저자(First Author) : 김충련 · 교신저자(Correspondent Author) : 최성욱
· 접수일(2012년 12월 28일), 수정일(1차 : 2013년 1월 8일), 게재확정일(2013년 2월 18일)

Ⅰ. 서 론

컴퓨터와 인터넷의 발전에 따라 인터넷 사용자의 서비스 요구 사항들도 변하고 있다. 몇 년 전만 하더라도 인터넷을 통해 단순한 텍스트와 이미지에 대한 정보 전송 서비스의 요구가 주가 되었지만 현재에는 텍스트와 이미지뿐만이 아니라 다양한 멀티미디어 정보의 전송 서비스의 요구가 급증해오고 있다. 하지만 텍스트와 이미지 전송에 비해 고품질 멀티미디어 정보 전송 서비스는 서버에 많은 부하를 주게 되어 서비스에 제한을 가지게 되었다. 따라서 텍스트와 이미지 데이터를 고속으로 처리할 수 있는 네트워크상에서의 멀티미디어 데이터의 처리 요구가 증가하고 있다.

이러한 문제를 해결하고자 나온 기술이 IP 멀티캐스트와 CDN(Contents Delivery Network)이다. IP 멀티캐스트와 CDN은 서버에 부하를 줄이면서 고품질의 멀티미디어 정보 전송이 가능하게 해주는 기술이다. 그러나 초기 고정비용이 많이 요구된다. 이에 대한 대안으로 초기 비용을 적게 들이면서 유사한 성능을 제공하는 오버레이 멀티캐스트가 주목받았다[1][2].

그러나 오버레이 멀티캐스트는 초기 비용이 적게 드는 반면 IP멀티캐스트에 비해 대역폭 사용이 비효율적이고, 물리적 토플로지와 오버레이 토플로지의 불일치로 인해 지연(delay)이 된다. 따라서 오버레이 멀티캐스트는 그룹에 가입, 탈퇴가 동적으로 이루어질 때 지연을 방지하는 안정성이 필수적하다.

본 논문에서는 위와 같은 단점을 개선하기 위해 오버레이 멀티캐스트를 기반으로 하여 클라이언트와 서버간의 통신 서비스를 관리하는 미들웨어를 설계 구현하였다. 미들웨어의 기본 구조로는 최적의 전송트리를 위한 트래픽 에이전트와 오버레이 멀티캐스트 토플로지 매니저로 구성되어 있으며, 트래픽 에이전트와 토플로지 매니저는 네트워크의 자원 상태를 기반으로 미들웨어의 통제 하에 서로 통신하면서 서비스를 제공한다. 본 논문에서는 클라이언트와 서버간의 통신 서비스를 관리하는 새로운 미들웨어를 제시하고, 제안한 미들웨어는 효과적인 전송트리를 구성하였으며, 서버와 클라이언트를 구조화 하여 기존 미들웨어의 단점을 개선하였다. 따라서 제안한 미들웨어를 이용하면 노드 간 전송트리 개선에 따른 노드 그룹으로의 가입(JOIN), 탈퇴(LEAVE)가 동적으로 이루어질 수 있어서 안정성의 확보가 가능해진다.

본 논문 2장에서는 노드 간 전송트리의 개선 제안을 기반으로 오버레이 멀티캐스트와 미들웨어 구성 간의 관련연구를 통한 차이점을 고찰하고 3장에서 제안한 노드의 최적화 방안을 제시한다. 4장에서는 제안한 시스템에 대해 설계 구현을 했으며 5장에서는 실험과 성능 평가를 하여 유효성을 확인한다. 마지막으로 6장에서는 결론 및 향후 연구방향을 제시한다.

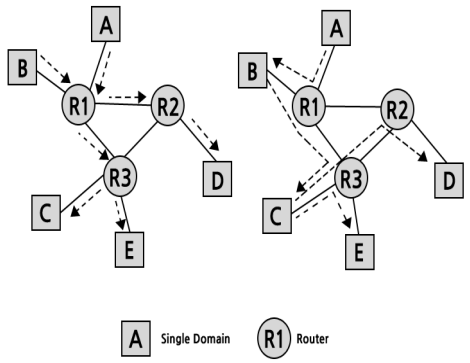
Ⅱ. 관련 연구

이 장에서는 오버레이 멀티캐스트와 IP멀티캐스트에 대해 설명하고 기존 오버레이 멀티캐스트의 노드 전송 트리와 노드 구성에 대해 기술하였다.

2.1 오버레이 멀티캐스트와 IP멀티캐스트

오버레이 멀티캐스트 네트워크는 IP 멀티캐스트의 대체 기술로, 오버레이 멀티캐스트 노드가 멀티캐스트 라우터 역할을 하여 데이터의 중복 전송을 최소화하여 네트워크 효율성을 향상시킨다. 서비스는 <그림 1>과 <그림 2>와 같이 네트워크 층(멀티캐스트 라우터)의 도움 없이 멀티캐스트 서비스를

실현하는 종단 호스트 기반의 라우팅 기술이다. 즉, 패킷의 복제 및 재전송, 라우팅, 그룹 멤버십의 관리, 그룹 통신에 관한 모든 기능들을 멀티캐스트 라우터가 아니라 종단 시스템에서 수행하는 구조이다[3][4][5].



(a) IP 멀티캐스트 (b) 오버레이 멀티캐스트

그림 1. IP계층 멀티캐스트와 오버레이 멀티캐스트
Fig. 1. IP layer multicast and overlay multicast

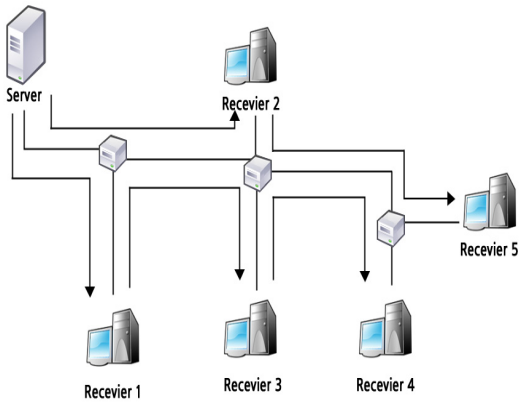


그림 2. 오버레이 멀티캐스트의 가상 토플로지
Fig. 2. Virtual topology of overlay multicast

2.2 오버레이 멀티캐스트 노드 전송 트리 조건

IP 멀티캐스트는 구현을 위해 라우터의 개선이 필요한 것에 반해 오버레이 멀티캐스트는 각 사용자의 응용 프로그램만 갖추어지면 구현될 수 있다는 장점이 있다. 반면 오버레이 멀티캐스트는 IP 멀티캐스트에 비해 지연시간이나 대역 폭 사용의 측면에 있어서 비효율적이라는 단점이 있다[6][8].

오버레이 멀티캐스트에서는 최대한 이런 비효율성을 줄이는 멀티캐스트 트리를 구성하는 것이 중요한 목표이다.

멀티미디어 데이터 중심의 응용 시스템을 위한 오버레이 멀티캐스트 트리가 충족시켜야 하는 조건은 크게 두 가지이다[7][8]. 첫 번째로, 사용자, 즉 트리 상의 노드의 차수가 적절한 상한 이하여야 한다. 트리 상의 특정 노드의 차수가 지나치게 높다는 것은 그 사용자가 패킷을 받았을 때 복제해서 보내 주어야 할 이웃 사용자가 그 만큼 많다는 것을 의미한다. 따라서 그 사용자가 패킷을 복사하고 처리하는 프로세싱의 과정에서 오버헤드가 커지고 또한 그 사용자가 접속하고 있는 네트워크 상의 링크로 동시에 많은 데이터를 보내 주어야하기 때문에 대역폭의 한계로 인해 각각의 이웃 사용자에 대한 전송 속도가 떨어지게 된다.

두 번째 조건은 멀티캐스트 트리의 지름(diameter), 즉 트리 상의 경로 거리로 볼 때 가장 먼 두 사용자 간의 거리가 작아야 한다는 것이다. 두 사용자 간의 트리 상의 경로 거리가 크면 두 사용자 간에 데이터를 보낼 때의 지연 시간이 커지기 때문에, 화상 회의와 같은 응용 시스템에 적합하지 않게 된다.

2.3 노드 구성 관련 연구

기존 미들웨어의 노드 구성은 다음과 같은 알고리즘으로 구현되어 있으나 여러 가지 문제점을 드러내고 있다.

먼저, 각 노드의 차수가 상한을 가지고 있을 때 지름이 작은 트리를 구성하기 위해 minimum

spanning tree 알고리즘을 변형한 알고리즘 등이 있으나 이는 최적 값을 보장하지 못하며, 사용자가 가입하고 탈퇴하는 동적인 상황에 대처하기가 어렵다[3][9].

또한, 지름이 일정한 상한 이하가 되면서 각 노드의 차수가 가능한 한 고르게 분포하는 트리를 구성하는 알고리즘이 있으나 이 알고리즘 역시 계산 복잡도가 크고 동적인 환경에 대응하지 못하는 단점이 있다[4][10].

이 외에도, 한 곳에서만 데이터를 전송하는 경우에 트리를 최적화하는 동적인 알고리즘 등도 많이 있기는 하지만 이 경우에는 트리의 지름이 아닌, 전송자로부터의 최대(평균)거리를 최소화하는 방법을 제시하고 있다[5][11].

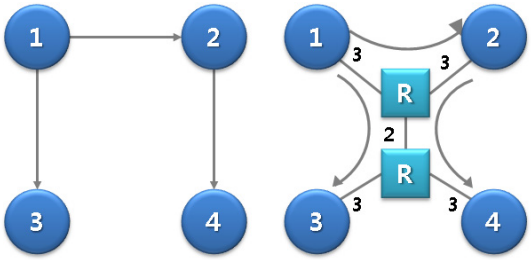
본 논문에서 사용자가 패킷을 복사하고 처리하는 프로세싱의 과정에서 오버헤드가 커지고, 동시에 많은 데이터를 처리하기 위해 각각의 이웃 사용자에 대한 전송 속도가 떨어지는 단점을 보완하여 서버와 클라이언트를 구조화함으로써 효과적인 전송트리를 구성한 미들웨어를 제안한다.

III. 제안한 노드 최적화 방안

3.1 노드 트리 모델

n명의 멀티캐스트 사용자가 있을 때 i번째 사용자를 U_i라고 정의한다. E_{ij}는 U_i와 U_j간을 연결하는 응용계층상의 선을, D_{ij}는 U_i와 U_j간의 네트워크상의 경로 거리(혹은 지연 시간)를 의미 한다. $\{U_i\}$를 점의 집합으로 $\{E_{ij}\}$를 변의 집합으로 갖는 트리 T를 오버레이 멀티캐스트 트리라고 정의한다. OD_{ij}는 T상에서의 U_i와 U_j간의 거리를 의미한다. T상에서의 U_i의 차수는 DGR_i로 나타내고, 차수의 상한은 $DGRbound_i$ 로 정의한다. <그림 3>은 한 네트워크 모델을 나타내고 있다. (a)는 U_1이 전송할 때, 응용계층상에서의 멀티캐스트 트리를 나타내고, (b)는 실제 네트워크상으로 패킷이 전송되는 경로를 나타낸다. 여기서 $OD_{13} = D_{13} = 3 + 2 + 3 = 8$ 이다. $D_{14} = 3 + 2 + 3 = 8$ 이고, $OD_{14} = D_{12} + D_{24} = 6 + 8 = 14$이다. $DGR_1 = DGR_2 = 2$이고, $DGR_3 = DGR_4 = 1$이다.

따라서, <그림 3>은 오버레이 멀티캐스트 트리의 한 예와 패킷의 전송 경로를 보여 주고 있으며, 사용자 2는 사용자 1에게서 받은 패킷을 복제하여 사용자 4 에게 전송한다. 여기서 (b)의 R 은 라우터를 의미하며 모든 U_i가 본 라우터를 통해 통신한다.



(a)멀티캐스트 트리 (b) 패킷 전송 경로
그림 3. 오버레이 멀티캐스트 트리
Fig. 3. Tree of overlay multicast

3.2 각 사용자의 경로 길이 계산

사용자 U_i는 오버레이 멀티캐스트 트리 T상에서 이웃한 사용자들을 알고 있다. 이웃 사용자 U_j에 대해서 경로 거리 D_{ij}를 알고 있고(이 경우 $D_{ij} = OD_{ij}$임), U_i에서 시작하며 E_{ij}를 포함하는 T상의 경로들 중에서 가장 긴 경로의 길이를 알고 있다. 이를 $MaxOD_{ij}$로 나타낸다. <그림 4>에서 $MaxOD_{31} = 9$, $MaxOD_{32} = 7$, $MaxOD_{43} = OD_{41} = 15$ 이다.

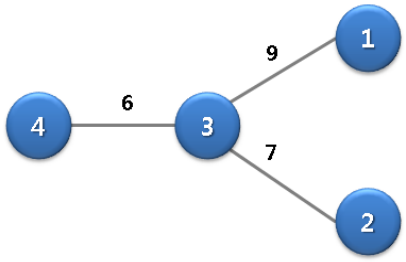


그림 4. 각 사용자의 경로 길이
Fig. 4. Path length of each user

3.3 $MaxOD_{ij}$ 계산

$MaxOD_{ij}$는 T상의 리프 노드(차수가 1인 노드)들로부터의 플러딩(flooding)을 통해 계산된다. <그림 5>에서 U_5가 U_4에게 자신이 리프 노드임을 알려주면 U_4는 이를 통해 $MaxOD_{45} = D_{45} = 7$ 임을 알게 되고, 이 값을 U3에게 알려준다. U3은 이를 통해 $MaxOD_{34} = MaxOD_{45} + D_{34} = 7 + 9 = 16$ 임을 알게 된다. 역시 U_3은 이 값을 U_1, U_2에게 보내주게 되고, 이를 통해 U_1은 $MaxOD_{13} = 22$ 임을, U_2는 $MaxOD_{23} = 23$ 임을 계산하고 역시 다른 이웃 사용자에게 값을 전달해 주게 된다. 이런 식으로, 어떤 사용자가 처음으로 MaxOD값을 전달받거나,

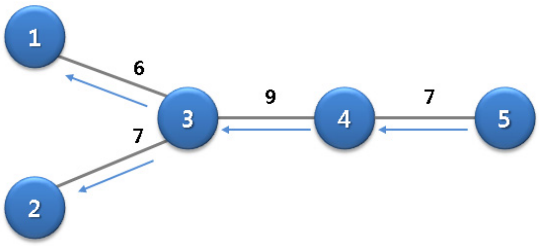


그림 5. 플러딩을 통한 MaxOD 계산 과정
Fig. 5. Calculation process of MaxOD with flooding

현재 가지고 있는 값보다 큰 MaxOD 값을 받게 되면 자신의 MaxOD값을 바꾸게 된다. 모든 플러딩이 완료된 후에는 모든 사용자가 자신의 모든 이웃에 대한 MaxOD 값을 정확히 알게 된다.

3.4 동적 트리 최적화의 과정

동적 트리 최적화는 세 사용자 사이에서 이루어지게 된 다. U_i는 자신의 MaxOD값들 중에 가장 큰 두 값을 선택하게 된다. 이것을 $MaxOD_{ij}$, $MaxOD_{ik}$라고 하자. U_i는 U_j, U_k에게 트리 최적화 과정을 시작할 것을 알리게 되고, U_j, U_k 는 각자의 MaxOD 값과 측정을 통해 알 수 있는 D_{jk}를 U_i에게 알려준다. 이를 기초로 U_i는 세 사용자 간의 변을 교환했을 때 U_i, U_j 혹은 U_k를 통과하는 트리 상의 최대 경로의 길이가 얼마인지 계산하게 된다. 세 사용자의 차수가 상한을 넘지 않는 범위 내에서 최대 경로의 길이를 최소화 하는 방향으로 변을 교환하게 된다.

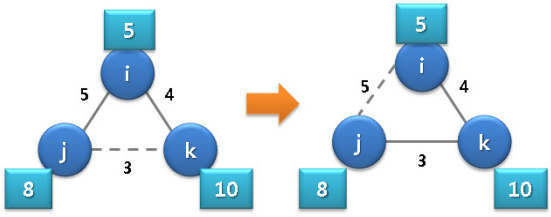


그림 6. 변 교환의 예
Fig. 6. Example of the side exchanged

위 <그림 6>은 변 교환의 예를 보여 주고 있으며 $D_{ij} = 5$, $D_{jk} = 3$, $D_{ki} = 4$, $MaxOD_{i?} = 5$, $MaxOD_{j?} = 8$, $MaxOD_{k?} = 10$임을 알 수 있다.

위의 변 교환을 좀 더 자세히 살펴보면, 처음 $D_{jk} = 3$ 인 상태에서 $D_{ij} = 5$ 로 아래의 식과 같은 과정을 통해서 변경됨을 알 수 있다.

Max {
$MaxOD_{j?} + D_{ji} + D_{ik} + MaxOD_{k?}$,
$MaxOD_{i?} + D_{ij} + MaxOD_{j?}$,

$MaxOD_{i?} + D_{ik} + MaxOD_{k?}$
$\} = 8 + 5 + 4 + 10 = 27$ (1)

변 교환 후,

Max {
$MaxOD_{j?} + D_{jk} + D_{ki} + MaxOD_{i?}$,
$MaxOD_{i?} + D_{ik} + MaxOD_{k?}$,
$MaxOD_{j?} + D_{jk} + MaxOD_{k?}$
$\} = 8 + 5 + 10 = 23$ (2)

변을 교환하기 전에 그래프 상에 사이클이 존재하지 않았다면 변을 교환해도 이 그래프에 사이클이 생기지 않는다. <그림 6> 에서 변 교환 후에 사이클이 생길 수 있는 가능성은 E_{jk}를 포함하는 사이클의 경우 밖에 없다. 이는 E_{jk} 외에도 U_j와 U_k를 연결하는 다른 경로가 있다는 것을 의미하는데, E_{ij}가 트리에서 제외되었으므로 이는 불가능 하다.

변을 교환한 후 에는 변화된 상황을 기초로 자신의 정보를 변경하게 된다. <그림 6>의 경우에는 U_j를 이웃에서 제외시키고, $MaxOD_{ik}$를 기존 값인 10 에서 $D_{kj} + MaxOD_{j?} = 11$로 변경하게 된다. 이 MaxOD 정보를 U_i에서 시작하여 플러딩을 통해 사용자 전체가 변 교환에 의해 바뀐 MaxOD정보를 갖게 된다.

3.5 제안한 동적 트리 최적화의 특징

트리 상의 모든 사용자가 자신을 중심으로 한 동적 변 교환을 시도하게 된다. 만일 함께 동적 변 교환을 수행하고자 하는 사용자가 이미 다른 변 교환에 참여 중이라면, 잠시 대기하였다가 다시 변 교환을 시도하게 된다.

이런 트리 최적화 과정은 동시에 여러 곳에서 이루어지게 된다. 트리의 지름에 해당하는 경로 상에서 일어나는 변 교환은 모두 트리의 지름을 감소시키는 방향으로 변을 교환하게 되고, 변 교환 후에도 이전의 지름보다 길이가 긴 새로운 경로는 생기지 않는다. 트리의 지름에 해당하는 경로 밖에서 일어나는 변 교환은 트리의 지름에 직접 적으로 영향을 미치지는 않게 된다. 그러므로 동시에 여러 곳에서 변 교환이 일어나더라도 트리의 지름은 증가하지 않는다. 모든 변 교환과 MaxOD의 플러딩이 완료된 후에는 모든 사용자가 올바른 MaxOD 값을 갖게 된다.

이 동적 변 교환을 통해서 DGR가 DGRbound를 넘지 않는 한도 내에서 트리의 지름이 감소하게 된다. DGR가 지나치게 커지지 않기 때문에 각 사용자의 프로세싱 오버헤드가 지나치게 커지지 않고, 또 사용자 주변 링크에 대한 대역폭 낭비가 생기지 않게 된다. 또한 트리의 지름이 감소하기 때문에 두 사용자 간의 전송의 최대 지연 시간이 감소하게 되어 실시간 응용에 적절한 환경을 만들게 된다.

이 방법은 중앙 집중적인 제어가 필요하지 않고. 짧은 시간 내에 수행할 수 있기 때문에, 사용자가 가입, 탈퇴하고, 네트워크상의 지연시간, 대역폭이 변화하는 동적인 환경에 잘 대처할 수 있으므로 이를 도입하여 다음 장에서 미들웨어의 설계에 적용한다.

IV. 미들웨어 시스템 설계

4.1 미들웨어 클라이언트

다음은 클라이언트의 동작으로써 서버에 접속(Join), 미디어 요청, 송신, 수신, 서버를 떠남(Leave), 재요청 동작 등을 이벤트 발생에 대해 살펴본다.

4.1.1 Join

서버를 실행(Start)하고 난 후 서버는 미디어를 요청하는 클라이언트의 접속을 기다리게 된다. 클라이언트가 해당 서버로 접속을 하면 Join 상태가 된다. 미디어 요청 송신과 수신 동작을 위한 서버 접속이 이루어 진 후 클라이언트는 자신의 정보를 보내며 동시에 자신이 미디어를 가지고 있는지의 유무를 보낸다.

클라이언트가 미디어를 요청할 때 어떤 미디어가 있는지 확인하기 위한 메소드로 GetNewMedia를 이용하여 메시지를 보낸다. 서버에 접속한 클라이언트 중에 해당 미디어를 가진 멤버가 있다면 그 정보들을 미디어를 요청한 클라이언트에게 보내게 된다.

그 후 전송받은 미디어들 중에 하나를 선택하여 Request 명령을 실행하면 선택한 미디어의 정보를 보기 위한 메시지인 JoinReq를 서버로 보내게 된다. 서버에서는 이 메시지를 받아 미디어를 전송할 수 있는 클라이언트를 찾아 그 클라이언트 정보를 보내게 된다. 정보를 받은 클라이언트 받은 정보를 가지고 미디어를 가지고 있는 클라이언트에게 JoinReq를 보내게 된다. 메시지를 받은 클라이언트는 메시지 정보를 저장하고 자신의 클라이언트 정보를 JoinRep 메시지로 보내게 된다.

JoinReq 메세지를 보낸 클라이언트로부터 받은 메시지를 가지고 멤버에 저장하고 MediaReq를 보내게 된다. MediaReq 메시지를 받은 클라이언트는 미디어를 송신을 처리하고 MediaRep를 보내게 된다. MediaRep 메시지를 받은 클라이언트는 미디어를 수신 처리한다.

4.1.2 Leave

Leave를 하는 경우 가장 먼저 서버에게 Leave 메시지를 보낸다. 그리고 난 후 자신이 미디어를 보내고 있는 클라이언트에게 Leave 메시지를 보낸다.

메시지를 보낸 후 전송하고 있는 미디어를 중지시킨다. Leave 메시지를 받은 클라이언트는 해당 클라이언트의 정보를 삭제하고 서버에게 미디어를 재요청하기 위해 JoinReq 메시지를 보낸다.

서버에서는 요청한 미디어가 있는지 없는지 판단 후 미디어를 보낼 수 있는 클라이언트의 정보를 보낸다. 즉, Leave하는 클라이언트는 미디어를 보내주고 있는 클라이언트에게 Leave 메시지를 보낸 후 처리하고 미디어를 받고 있던 클라이언트에서는 미디어 전송중지를 한다.

Leave하는 클라이언트는 정보를 삭제하고 스레드를 종료 시키면 전송 종료가 된다. 미디어를 재요청한 클라이언트는 서버에게 받은 정보로 미디어 전송 가능한 클라이언트에게 JoinReq 메시지를 다시 보낸다. 미디어 전송 가능한 클라이언트는 JoinReq 메시지를 받고 정보를 저장하고 포트번호와 기타 정보를 JoinRep 메시지로 보내게 된다.

메시지를 받은 클라이언트는 받은 정보로 미디어를 요청하는 MediaReq 메시지를 보내고 그 메시지를 받은 클라이언트는 미디어를 전송한 후 MediaRep 메시지를 보낸다. MediaRep 메시지를 받아 미디어 수신을 처리한다.

4.2 미들웨어 서버

미들웨어 서버의 동작에서 우선 서버의 시작은 시작 메소드를 통해서 이루어진다. 소켓 및 스레드를 생성하여 클라이언트의 접속을 기다리게 되며 클라이언트의 접속이 이루어지면 메시지를 매개로 메시지를 주고받아 처리한다.

클라이언트의 최초 접근 시 클라이언트의 정보(아이피, 아이디 등)를 얻어내고 해쉬 테이블에 저

장하기 위해 고유 키 값을 얻게 된다. 그리고는 얻어낸 정보를 멤버리스트에 저장한다. 이러한 모든 단계까지의 프로세스가 Join시의 처리 흐름이다.

Join을 마무리하게 되면 다음으로 받게 되는 메시지는 미디어를 소유하고 있는지 아닌지에 대한 메시지인데 만약 미디어를 가지고 있는 상황이면 미디어 정보를 전달 받게 된다. 이 과정은 미디어가 존재할 때 만 이루어지며 정보는 각각 분석되어 저장된다.

클라이언트가 미디어를 받기를 원하게 되면 제일 먼저 참여하고 있는 멤버의 정보를 요구한다. 이 메시지를 받게 되면 멤버 리스트를 분석한 후 미디어를 가지고 있는 멤버의 정보를 추출하여 넘겨주게 된다. 이제 클라이언트가 미디어에 접근을 요구하게 되면 해쉬테이블에 저장된 소스의 정보를 분석하여 접속해야 하는 부모 클라이언트의 정보를 전송해주게 된다. 탈퇴 시에는 탈퇴의 결과 메시지를 반환해 주게 된다.

4.3 트래픽 에이전트

트래픽 에이전트는 클라이언트간의 패킷 대역폭을 체크하여 토플로지 상의 최적화된 상위 클라이언트를 찾는 것이며 하위노드 컨트롤 모니터링을 통해서 자신에게 접속되어 있는 클라이언트의 정보를 확인하는 기능을 가진다. 다음으로는 세션 컨트롤러로 모든 클라이언트의 활성화 여부를 확인하는 기능을 담당한다. 마지막으로 노드 컨트롤러로 클라이언트간의 연결을 담당한다. 데이터베이스에 저장되는 값으로는 각 상위 IP와 현재 기준 IP 그리고 하위 IP의 각각의 속도 정보, 그리고 누적된 노드수가 있다.

트래픽 에이전트의 동작을 보면 아래 <그림 7>과 같은 플로우로 처리가 되는데 처음 클라이언트가 접속해서 자신이 최상위 클라이언트일 경우에는 메인 서버에 접속되며 그 이후에 접속하는 클라이언트는 에이전트의 통제 하에 각각 상위 클라이언트와 자신과의 속도를 체크한다. 즉 제어 파라미터 값과 비교하여 서비스 가능 여부를 확인하게 된다. 이 때 식 1에서 제어 파라미터 값인 pContBand 는 컨텐츠의 크기를 상위 클라이언트가 현재 서비스해주고 있는 노드수로 나눈 것이다.

pContBand = bit rate / node 수 (3)

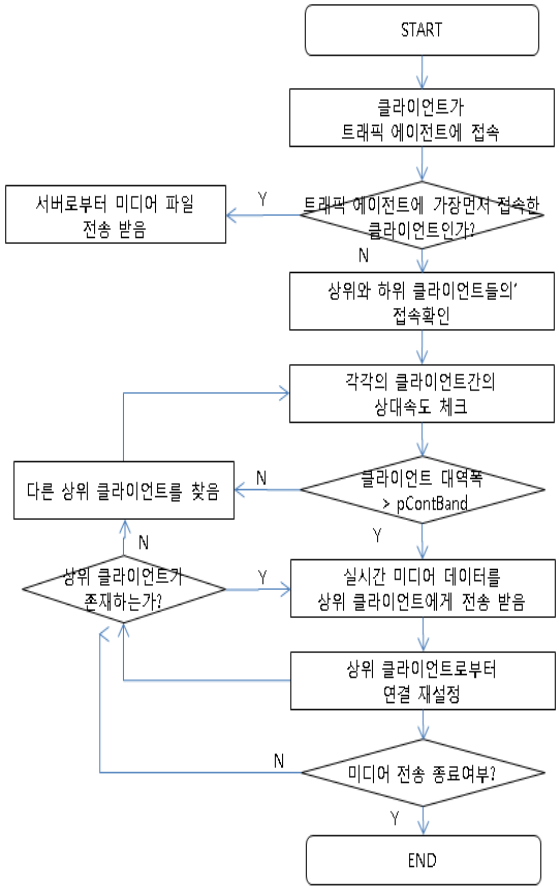


그림 7. 트래픽 에이전트 프로세스
Fig. 7. Traffic agent process

Ⅴ. 실험 및 성능 평가

5.1 시뮬레이션 환경

제안한 미들웨어 시스템의 실험은 오버레이 트래픽 에이전트들을 도입하여 클라이언트와 서버를 구동할 수 있는 미들웨어를 구현하여 실험하였다. 미들웨어는 서버와 클라이언트의 기본 기능으로써 서버에 접속(Join), 미디어 요청, 송신, 수신 동작, Leave, 재요청 동작 등을 수행하였다.

오버레이 네트워크 구축은 PC급으로 전용서버를 구축하여 오버레이 멀티캐스트 프록시(Proxy)를 구축하였다.

시스템 구현은 Java를 이용하였으며 멀티미디어 및 데이터의 정보 전송 기능은 RTP(Real Time Protocol) 를 이용하여 socket방식으로 구현하였다.

5대의 호스트가 네트워크에 연결되어 있고 장비는 모두 Windows XP환경에서 이미지 파일, 음성파일의 송수신에 대해서 실험하였다. 음성 데이터는 25Khz stereo 모드 PCM 형식의 디지털 음성파일이고, 이미지 데이터는 1024*768 16bit 컬러 이미지를 전송하였다. 데이터는 매 case 별로 1분간 전송하였다.

미들웨어 서버를 실행시키고 Start 하면 미들웨어 서버가 시작된다. 그리고 클라이언트의 접속을 기다리게 된다. 클라이언트를 실행시키고 Join 한 후 서버의 IP주소와 자신의 아이디 그리고 미디어의 유뮤를 체크한다. 체크 후 미디어를 보고자 하는 경우 GetNewVideo 메소드를 통해 원하는 미디어를 선택하면 미디어를 전송 받을 수 있다.

5.2 실험 및 성능 평가

제안한 미들웨어 시스템의 성능 평가를 위해서 minimum spanning tree 알고리즘을 변형해서 사용한 기존 미들웨어 시스템과 비교 분석하였다. 시뮬레이션 환경은<[표 1>과 같은 조건으로 하였다.

<표 1>은 네트워크에 연결되어 있는 5대의 클라이언트에서 약 60초간 음성 파일과 이미지 파일을 전송한 결과에 대해 각 클라이언트에서 평균한 값을 보여주고 있다. #1에서 #5까지 순차적으로 증가시키면서 접속을 하여 실험을 하였고 전송패킷수, 재전송패킷수, 재전송비율, 전송속도, 수신패킷수, 손실패킷수, 손실율을 구하였고 이 중 아래 세 항목에 대해서만 표로 나타내었다.

표 1. 60초간 전송 결과
Table 1. 60 seconds the transmission results

#	전송구분	전송 패킷수	전송속도 (Mbps)	손실율
#1	음성파일전송	1,000	0.114	0.0000%
#2	음성파일전송	4,000	0.357	0.0000%
#3	음성파일전송	7,000	0.990	0.0012%
#4	음성파일전송	10,000	1.253	0.0029%
#5	음성파일전송	13,000	1.697	0.0061%

(a) 음성 파일 전송 결과

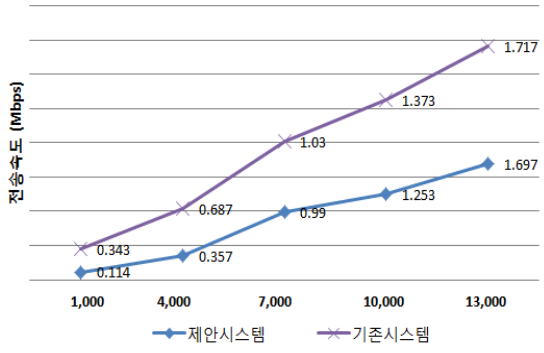
#	전송구분	전송 패킷수	전송속도 (Mbps)	손실율
#1	이미지파일전송	157,494	19.174	0.0006%
#2	이미지파일전송	295,734	37.457	0.0011%
#3	이미지파일전송	433,974	55.739	0.0029%
#4	이미지파일전송	572,168	73.668	0.0045%
#5	이미지파일전송	712,646	91.454	0.0052%

(b) 이미지 파일 전송 결과

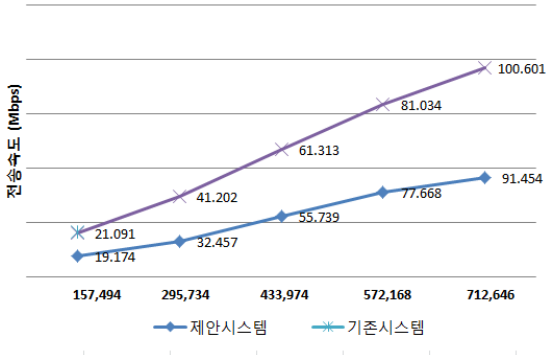
위 오버레이 멀티캐스트 상에서 미들웨어를 적용한 후의 실험 결과를 비교해보면 제안한 미들웨어가 기존 미들웨어 보다 약 20% 정도 성능이 개선되었음을 알 수 있다. 위 향상된 결과에는 오버레이 멀티캐스트 환경이라는 인프라 측면도 작용을 하였음을 배제할 수 없지만 노드 간 전송트리

개선에 따른 노드 그룹으로의 가입(JOIN), 탈퇴(LEAVE)가 동적으로 이루어질 수 있는 안정성의 확보가 가능해 졌기 때문인 것으로 분석된다. 또한, 스케줄링과 커넥션관리에서는 안정성과 성능 향상을 다소 더 보장할 수 있다.

<그림 8>은 표 1의 샘플데이터에 대해 오버레이 멀티캐스트 상에서의 미들웨어 유무의 실험결과를 비교한 그래프이다. 클라이언트 수가 늘어나기 전인 초기 단계에서는 전송속도가 비슷하다가 전송 패킷수가 증가 할수록 전송속도 차이가 많이 남을 알 수 있다.



(a) 음성 파일 전송 결과

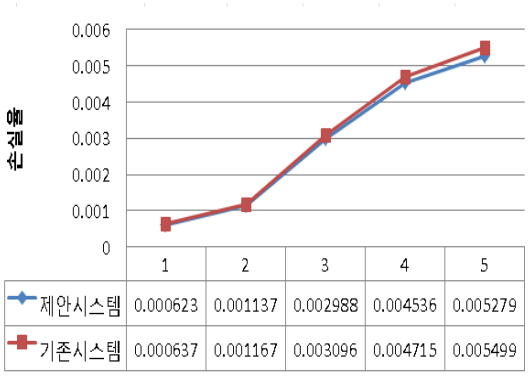


(b) 이미지 파일 전송 결과

그림 8. 미들웨어에 적용한 후 성능 비교
Fig. 8. The performance comparison after middleware is applied

<그림 9>는 표 1에서 나타난 항목 중 손실율을 비교한 그림이다. 손실율은 약 2% ~ 4% 수준에서 제안 시스템의 손실율이 기존 시스템에 비해 감소함을 알 수 있었다.

이는 대체로 LAN 환경의 서비스에서는 패킷 손실률이 2%이상 4%이하가 되어야 한다고 가정했을 때의 기준이며 이 기준은 서비스 종류나 네트워크 환경에 따라 차이가 날 수 있다.



	1	2	3	4	5
제안시스템	0.000623	0.001137	0.002988	0.004536	0.005279
기존시스템	0.000637	0.001167	0.003096	0.004715	0.005499

그림 9. 제안시스템과 기존시스템 손실율 비교
Fig. 9. Loss Rate Comparison (new system vs existing system)

Ⅵ. 결 론

본 논문은 오버레이 멀티캐스트 기반 하에 효과적인 전송트리를 구성하여 노드 간 전송트리 개선에 따른 노드 그룹으로의 가입, 탈퇴가 동적으로 이루어질 수 있는 안정성이 확보된 미들웨어를 제안하였다. 노드트리 최적화 방법을 위해 관련연구를 통해 먼저, 노드트리 모델을 구성하고 각 사용자의 경로 길이를 계산한 후, max 값을 계산하여 이를 기준으로 동적 트리 최적화가 가능하도록 하였다.

제안한 미들웨어 시스템의 성능 평가를 위해서 minimum spanning tree 알고리즘을 변형해서 사용한 기존 미들웨어 시스템과 비교 분석하였다.

실험은 클라이언트를 점차적으로 5단계로 증가시키면서 접속을 하는 동안 음성 파일과 이미지 파일을 이용하여 매 case 별로 1분간 전송하면서 전송율의 변화를 살펴보았다. 실험 결과로는 미들웨어 상에서 제안한 미들웨어가 약 20% 정도 성능이 향상됨을 알 수 있었다. 또한 미들웨어를 이용하여 클라이언트와 서버를 관리하게 함으로써 스케줄링과 커넥션관리 측면에서도 안정성과 성능 향상을 보장할 수 있게 되었다.

하지만 몇 가지 개선사항이 남아 있는데 자바가 갖고 있는 기본적인 입출력 속도문제의 개선과, 미들웨어 서버에서의 미디어 파일 전송 트리 구성의 불편함과 그에 따른 오류, 다양한 미디어 파일 전송 테스트 등이 있다. 이와 같은 점을 보완하는 추가 연구가 이루어지면 개인방송, 온라인 강의 등 다양한 곳에 응용이 가능할 것이다. 아울러 본 연구를 통하여 오버레이 멀티캐스트 기반 미들웨어 기술이 광범위하게 응용될 것으로 기대한다.

참고문헌

[1] 강미영, 양현종, 남지승. “다중 사용자의 멀티미디어 요구 서비스를 위한 오버레이 멀티캐스트 트리의 구성과 복구 방안“. 한국통신학회논문지 ’08-12 Vol. 33 No. 12

[2] 박은용, 유정, 한선영, 김진철, 강상욱. “대역폭 적응형 분산 스트리밍 기법을 이용한 IPTV 서비스용 오버레이 멀티캐스트 네트워크“. 정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제12호(2010.12)

[3] Lambert M. Surhone, Mariam T. Tennoe, and Susan F. Henssonow, “Overlay Multicast“ Betascript Publishing. 2010

[4] Sasu Tarkoma, “Overlay Networks Toward Information Networking“. Auerbach Publications. 2010.02

[5] Dorian Cecil Arnold, “Reliable, scalable tree-based overlay networks.“. Proquest, Umi Dissertation Publishing. 2008

[6] 김문화, 황준. “멀티미디어 데이터 통신의 신뢰성 보장을 위한 서비스 제공자 중심의 멀티캐스트 미들웨어 설계 및 구현“. 한국 인터넷 정보학회(3권 4호), 2002

[7] 이춘성, 한선영, 송정욱, 최병욱. “QoS와 신뢰성을 제공하는 확장성 있는 오버레이 멀티캐스트“, 정보과학회논문지 제 13-C권 제6호 (2006.10)

[8] 이보영, 조승철, 한선영. “차세대네트워크 환경에서 서비스 등급 및 사용자 환경에 따른 차별화된 QoS를 지원하는 오버레이 멀티캐스트“. 정보처리학회눈문지 제15-C권 제6호(2008.12)

[9] Liu, Xiaomei. “Building Reliable Applications in Overlay Networks.“. Proquest, Umi Dissertation Publishing. 2008

[10] Onus, Melih. “Overlay Network Construction in Highly Decentralized Networks.“. Proquest, Umi Dissertation Publishing. 2009

[11] Niveditha Sundaram,. “Distributed multirate streaming in overlay networks.“. Proquest, Umi Dissertation Publishing. 2008

저자소개

김충련(Chung-Ryeon Kim)

1991년 경희대학교 (공학석사))
2013년 인천대학교 (박사과정)

1998년~현재 HK텔레콤(주) 대표이사
※ 관심분야 : 소프트웨어 공학, 정보통신, 인터넷 스트리밍 서비스

최성욱(Sung-Uk Choi)

1983년 광운대학교(이학사)
1987년 경희대학교(공학석사)
2001년 아주대학교(공학박사)

2010~현재:인천대학교 컴퓨터공학부 교수
※ 관심분야 :SoftWare Archtecture, Mobile Network, Web Services