제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

순수 P2P 환경에서의 효율적 자원 검색 기법

김인숙*, 강용혁*, 엄영익*
*성균관대학교 정보통신공학부
e-mail:{easy, yhkang1, yieom}@ece.skku.ac.kr

An Efficient Resource Discovery Mechanism for Pure P2P Environments

In-suk Kim*, Yong-hyeog Kang*, Young Ik Eom*
*School of Information and Communication Engineering, Sungkyunkwan University

요 약

최근 인터넷의 급속한 성장과 초고속 통신망의 구축으로 인하여 다양한 멀티미디어 서비스들이 제공되고 있다. 그러나 현재 대부분의 멀티미디어 서비스들은 클라이언트/서버 모델을 기반으로 구축되어 있기 때문에, 중앙 서버로의 과도한 부하가 집중되는 문제점을 가지고 있다. 본 논문에서는 순수 P2P 환경으로 구축된 멀티미디어 서비스 환경에서의 자원 검색 기법을 제안한다. 제안 기법은 고정 에이전트를 기반으로 자원 검색을 수행하여 기존의 순수 P2P 환경에서의 검색 기법으로 인한 문제점을 해결한다. 또한, 시나리오를 통하여 본 논문에서 제안하는 알고리즘의 구체적인 동작과정을 제시한다.

1. 서론

현재 인터넷의 급속한 성장과 초고속 정보통신망의 구축으로 인하여 다양한 멀티미디어 서비스들이 제공되고 있으며, 이러한 멀티미디어 서비스 환경은 대부분이 몇 개의 대용량 서버를 구축하여 서비스를 제공하는 클라이언트/서버 모델로 구성되고 있다. 그러나 기존 클라이언트/서버 모델은 서버 집중식으로 트래픽 집중의 한계 때문에 어려움을 겪고 있다[1].

최근, 이를 해결하기 위해 P2P(peer-to-peer) 네트워크 환경에 대한 연구가 활발히 진행 중이다. P2P 네트워크 환경이란 클라이언트 상호간 분산 및 협력이라는 새로운 개념의 네트워크라 할 수 있다[1,2].

본 논문은 멀티미디어 서비스를 제공하는 호스트들이 순수 P2P 네트워크로 구성되는 환경에서 고정 에이전트를 이용하여 특정 호스트에 의해 요청받는 자원을 효율적으로 찾는 검색 메커니즘을 제안한다. 이 제안기법을 사용할 경우, 중앙 서버를 이용한 다른 검색방식에서 나타나는 부하의 집중 현상을 제거할 수 있다[3]. 또한, 검색 과정의 경로와 무관하게 결과를 반환하므로 자원 검색 정보의 유실을 방지한다. 더불어 각 멀티미디어 서비스를 제공하는 호스트들 상에 보유한 자원과 동일 자원을 가지고 있는 다른 호스트의 위치 정보를 테이블로 유지하므로, 복수 자원의 신속한 검색을 지원한다.

본 논문의 2장에서는 P2P 네트워크의 개념과 P2P 네트워크에서의 기존 자원 검색 기법들에 대해 소개한다. 3장에서는 본 논문에서 제안하는 자원 검색 기법을 설명하고, 시나리오를 통해 제안 기법의 구체적인 동작 과정에 대해 알아본다. 마지막으로, 4장에서는 본 논문의 결론에 대해서 기술한다.

2. 관련연구

P2P 네트워크란 각 호스트들이 프로세싱 파워, 저장 공간, 컨텐츠 등의 하드웨어 자원들의 일부를 공유하는 네트워크를 말한다. P2P 네트워크에서는 임의의 호스트가 필요에 따라 서버 또는 클라이언트로 동작할 수 있다. P2P 네트워크는 중앙 서버를 갖는 혼합형 P2P(hybrid P2P) 네트워크와 중앙 서버를 갖지 않는 순수 P2P(pure P2P) 네트워크로 구분될 수 있다. 혼합형 P2P 네트워크는 인덱스 서버가 검색에 개입하는 반면, 순수 P2P 네트워크는 호스트 간에 직접 검색 과정을 수행한다[1,4].

순수 P2P 네트워크에서 효율적인 검색을 위해 제안되고 있는 검색 기법들은 일반적으로 브로드캐스트 요청 모델(flooded requests model), 도큐먼트 라우팅 모델(document routing model)로 분류될 수 있다[5,6].

브로드캐스트 모델에서 임의의 호스트가 컨텐츠를 검색하고자 할 때에는 자신과 직접 연결된 피어들에게 검색 요청 메시지를 브로드캐스트하는데, 이 메시지는 해당 컨텐츠를 가진 호스트에게 도달하게 될 때까지 전달된다. 대표적인 시스템으로는 Gnutella가 있다[5,6].

도큐먼트 라우팅 모델은 순수 P2P 네트워크에서 사용되는 검색 모델로, 공유되는 컨텐츠의 ID를 기반으로 자원을 검색하는 모델이다. 각 호스트는 컨텐츠 ID에 가장 유사한 ID를 가진 호스트 쪽으로 컨텐츠를 라우팅한다. 컨텐츠를 검색할 때 컨텐츠는 컨텐츠 ID에 기반해 요청을 발생한 피어로 전송되고 라우팅에 참가한 각 호스트는 로컬에 복사본을 갖게 된다. 대표적인 시스템으로는 FreeNet이 있다[5].

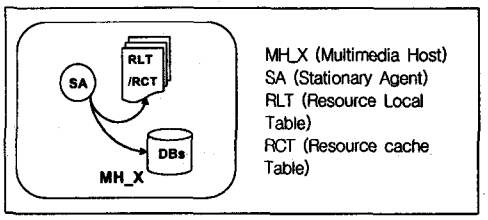
3. 순수 P2P 기반 자원 검색 기법

본 절은 논문에서 제안하는 시스템의 구성과 각각의 자료 구조, 알고리즘, 동작 시나리오에 대해 살펴보겠다.

3.1 시스템 구성

본 논문에서는 멀티미디어 호스트들로 구성된 순수 P2P 네트워크 환경을 가정하며, 방화벽(firewall)과 NAT에 대한 시스템 고려는 배제하였고 멀티미디어 호스트간의 자원 검색을 위해 에이전트를 사용하는 기법을 제안한다.

본 논문에서 제안하는 기법의 시스템 구성은 그림 1에서 보인다.



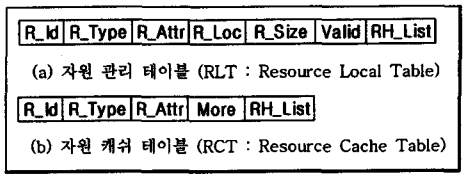
(그림 1) 멀티미디어 호스트 구성

그림 6에서와 같이 멀티미디어 호스트(MH)는 자신이 소유한 컨텐츠 정보를 유지하는 DB를 가지며, DB 및 자원 테이블(RLT, RCT)을 관리하는 고정 에이전트(SA)를 가진다. RLT는 자신이 소유한 자원들에 대한 정보와 자신이 소유한 자원과 동일 자원을 소유한 다른 MH들에 대한 정보를 유지한다.

RCT는 자신이 소유하지 않지만 최근에 검색 요청이 들어온 자원을 소유한 다른 MH들에 대한 정보를 유지한다. SA는 사용자의 UI로부터 수신한 검색 요청에 근거하여 자원 질의 메시지(QM:Query Message)를 생성하고, 자원 질의 메시지를 통해 해당 자원을 요청한다. 자원 질의 메시지에는 메시지의 경로(M_Path:Message Path)가 쌓여서 메시지가 루프에 빠지는 것을 방지하고 자원을 신속하게 찾을 수 있도록 돕는다.

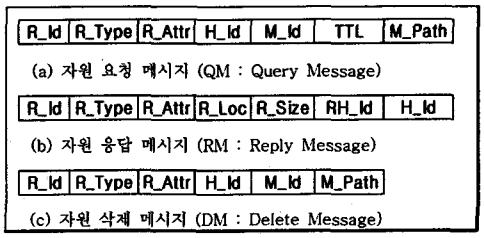
3.2 자료 구조

본 논문의 제안 기법에서 사용되는 자원 테이블들과 메시지의 구조는 그림 2와 3에서 보인다.



(그림 2) 테이블의 구조

R_Id(Resource Identifier), R_Type(Resource Type), R_Attr(Resource Attribute)는 RLT, RCT에서 사용되며, 자원을 구분하는 키워드를 구성하여 사용자가 자원 검색을 요청했을 때 사용하게 된다. 그림 2-(a)에서 필드 R_Loc(Resource Location)에는 자원의 물리적인 주소가 저장되고, 필드 Valid에는 자원 정보의 생성/삭제를 나타내는 값이 들어간다. 필드 RH_List (Resource Host List)는 해당 자원과 동일 자원을 보유한 호스트의 Id 목록이다. 그림 2-(b)의 필드 More에는 RH_List에 MH들이 추가 여부를 나타내는 값이 들어간다.



(그림 3) 메시지 구조

그림 3-(a)는 자원을 검색할 때 사용되는 요청 메시지인데, H_Id (Home Identifier)에는 사용자에게 검색 결과를 돌려주기 위해 검색을 발생시킨 홈의 Id가 저장되고, M_Id(Message Identifier)에는 각 메시지를 구분해주는 Id가 들어간다. 필드 TTL(Time To Live)과 M_Path(Message Path)는 자원 검색 질의가 P2P 네트워크 상에서 루프가 생성되거나 무한히 반복되는 것을 방지하도록 한다. 그림 3-(b)는 자원을 발견했을 때 home에게 검색 결과를 돌려주는 응답 메시지이며 RH_Id에는 자원이 발견된 MH의 Id가 들어간다. 그림 3-(c)는 자원이 삭제됐을 때 다른 호스트들에게 해당 자원이 삭제되었음을 알리는 메시지이다.

3.3 알고리즘

본 절에서는 자원의 검색과 자원의 관리를 위한 SA의 알고리즘들을 소개한다.

```
find an entry E in RLT with the specified keywords;
if (∃ E) {
    send E to UI;
    send QM to each host in RH_List;
}
else {
    create and insert a new entry E′ into RLT with the specified keywords;
    find an entry E in RCT with the specified keywords;
    if (∃ E) {
        send QM to each host in RH_List;
        E′ in RLT = E in RCT ;
    }
    else
        send QM to next neighbor;
}
wait for search result;
when (the search result is arrived)
    update the RH_List information in RLT;
```

(알고리즘 1) Home에서 자원 요청을 받았을 때 SA의 작업 절차

알고리즘 1은 SA에게 사용자가 UI를 통해 자원을 요청한 후부터 검색된 자원 위치 정보가 Home으로 전달되어 RLT에 반영될 때까지 Home에 위치한 SA의 동작 과정을 보인다. 사용자가 검색된 자원의 다운로드를 완료하더라도 SA의 작업은 종료되지 않는다. 이는 SA가 돌아오는 검색 결과에 대해 자원 위치 정보인 RH_List를 작성하여 해당 자원의 다음 번 검색 속도를 향상시키기 위함이다.

```
receive QM;
if (M_Id of QM has already been registered in current host)
    discard the QM;
else {
    find an entry E in RLT with the specified keywords;
    if (∃ E) {
        send E to home;
        send QM to each host in RH_List;
        if (RH_List is not fully occupied)
            insert H_Id into RH_List;
    }
    else {
        find an entry E in RCT with the specified keywords;
        if (∃ E) {
            send QM to each host in RH_List;
            if (RH_List not fully occupied)
                insert H_Id into RH_List;
        }
        else {
            send QM to next neighbor;
            create a new entry E′ into RCT with the specified keywords;
            insert H_Id into  of E′.RH_List;
            E′.More in RCT = 1;
        }
    }
}
```

(알고리즘 2) Home이 아닌 다른 MH 상에서의 SA 작업 절차

알고리즘 2는 Home이 아닌 다른 MH 상에 위치하는 SA들이 자원의 검색에 참여할 때의 동작 과정을 보인다. Home에서 다른 MH 상으로 보내진 검색 메시지들은 각 MH 상의 SA에 의해서 처리가 된다. SA는 도착한 QM의 M_Id를 이용해 이전에 해당 QM이 현 MH에 전달되었었는지 여부를 검사한다. 이것은 불필요한 QM이 계속적으로 발생하는 것을 막기 위함이다. SA는 해당 QM이 중복되어 전달되었으면 폐기하고, 현 MH에 처음으로 전달되었으면 다음 과정을 진행한다. SA는 해당 자원의 위치 정보가 RLT, RCT에 있는지 검사하고 없으면 가까이 이웃한 MH로 검색 메시지를 보낸다. 만약 현 MH의 RLT나 RCT에 자원의 위치 정보가 있으면 위치 정보가 발견된 테이블의 RH_List에 명시된 호스트들로 QM을 발생시킨다. 전자의 경우에는 차후 해당 자원이 Home에 추가가 되면 이에 대한 자원 위치 정보를 유지하기 위해 미리 RCT를 작성하고 RH_List에 홈의 Id만을 적어놓는다.

RLT나 RCT의 RH_List 상 위치로 자원 검색 메시지를 전송할 때에는 각 MH마다 하나씩 검색 메시지를 보내는데 이때 이 검색 메시지에는 메시지가 지나간 경로 정보가 포함된다.

```
monitor RLT and RCT;
if (E.Valid in RLT == 1) // resource is added
    send QM to local MH;
else if (E.Valid in RLT == 2) { // resource is deleted
    send DM to each host in RH_List;
    delete E;
    }
if (E.More in RCT == 1) // more RH_List is needed
    send QM to each host in RH_List;
```

(알고리즘 3) 자원의 생성/삭제에 대한 SA의 작업 절차

알고리즘 3은 사용자가 직접 MH에 새로운 자원을 추가하는 경우와 자원을 삭제하는 경우에 이를 반영하기 위한 SA의 동작 과정을 보인다.

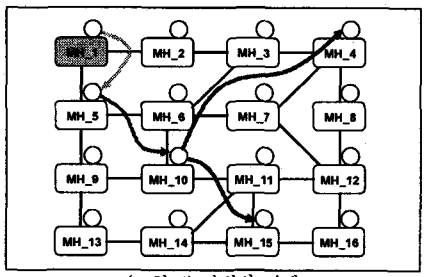
사용자가 다른 MH로부터 자원을 다운로드하는 경우 또한 새로운 자원이 MH의 DB에 추가가 되지만 자원 검색 과정에서 이에 대한 자원 테이블 상의 추가 작업은 처리되므로 이 알고리즘에서 다루지 않는다.

SA는 알고리즘 2에서 다른 MH로부터 QM을 받았으나 해당 자원의 위치 정보가 RLT나 RCT에 없었을 경우, 차후 해당 자원이 Home에 추가될 것을 감안하여 해당 자원에 대해 RCT를 작성하고 RH_List에 H_Id를 기록했다. 이 자원에 대해 RH_List의 채워지지 않은 부분에 대한 처리는 알고리즘 3에서 보인다. SA가 주기적으로 RLT와 RCT를 모니터링하면서 이를 발견하였을 때 RCT의 RH_List에 기록된 Home으로 QM 메시지를 보낸 후, 돌아오는 결과로 나머지 RH_List를 작성한다.

3.4 동작 시나리오

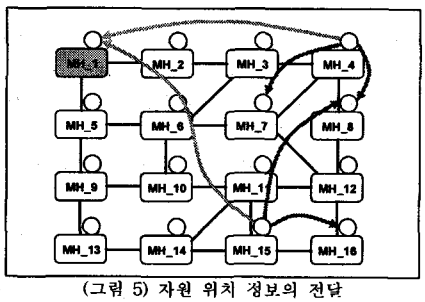
다음 시나리오는 한 MH_1이 요청하는 자원이 MH_4, MH_7, MH_8, MH_15, MH_16에 있으며 이 중 MH_15에서 해당 자원을 다운로드하는 경우를 보인다.

그림 4와 같이 MH_1에서 SA는 자신의 RLT, RCT를 검색했으나 해당 자원에 대한 위치 정보를 얻지 못해 이웃한 MH_5로 자원 검색 요청을 보낸다. MH_5 또한 자원에 대한 정보가 없으므로 MH_5 상의 SA는 다시 이웃한 MH_10으로 자원 검색 요청을 보내는데, MH_10의 RCT에서 해당 자원에 대한 위치 정보가 발견된다. MH_10의 SA는 RCT의 RH_List 상 MH들(MH_4, MH_15)에게 해당 자원의 검색 요청을 보낸다.



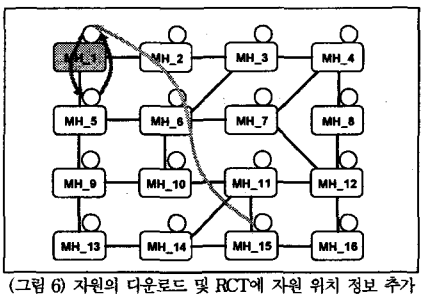
(그림 4) 자원의 검색

MH_4와 MH_15 상의 SA는 자원 검색 요청 메시지를 받고 이 메시지의 Home인 MH_1에게 RM을 통해 각각 자신의 RH_Id를 보낸다. MH_4와 MH_15는 RLT 상 RH_List의 MH들(MH_7, MH_8, MH_16)에게 자원 검색 요청을 보낸다. 이 때 MH_8은 MH_4와 MH_15의 RLT의 RH_List에 의해서 중복된다. 그러나 SA가 중복 메시지에 대해 검사를 하므로 늦게 도착하는 메시지가 무시된다. 이 과정을 그림 5에서 보인다.



(그림 5) 자원 위치 정보의 전달

MH_1은 검색된 MH의 목록 중 하나(MH_15)를 선택해서 파일을 다운받을 수 있다. 각 MH 상의 SA들은 일정 기간마다 RLT와 RCT를 점검해 RH_List 값을 갱신한다. MH_5의 경우 그림 4의 과정에서 해당 파일에 대한 정보가 전혀 없었으므로 RCT 상 RH_List에 MH_1이 추가되어 있다. 이를 MH_5의 SA가 발견하여 MH_1에게 RH_List 정보를 요구하고 이 정보를 MH_1이 제공하게 된다. 이 과정을 그림 6에서 보인다.



(그림 6) 자원의 다운로드 및 RCT에 자원 위치 정보 추가

4. 결론

본 논문에서는 순수 P2P로 구성된 멀티미디어 서비스 환경에서의 검색기법을 제안하고 알고리즘과 간단한 동작 시나리오를 통해 이를 살펴보았다. 자원의 빠른 검색을 위해서 동일 자원들의 위치 정보를 테이블로 유지하도록 하였으며 보유하지 않았으나 자원 요청이 들어온 자원에 대해서 자원들의 위치 정보를 테이블로 유지하였다. 또한 자원 검색을 위한 메시지의 호스트 재방문 여부를 확인하여 자원의 검색을 위해 발생하는 메시지가 불필요하게 증가하지 못하도록 하였는데, 이는 동일 자원에 대한 검색이 증가할 수 록 검색을 위해 발생하는 메시지의 양을 감소시킨다.

참고문헌

[1] White Paper: Intel Corp., "Peer to Peer Computing: P2P File-Sharing at work in the Enterprise," http://www.intel.com/ebusiness/pdf/prod/peertopeer/p2p, 2001.

[2] A. Oram, "Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology," O'Reilly, 2001.

[3] Duncan Aitken et al, "Peer-to-Peer Technologies and Protocols," http://matrix.netsoc.tcd.ie/~neo/4ba2/p2p/index.html

[4] B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks," In 22nd International Conference on Distributed Computing Systems (ICDCS 2002), 2002.

[5] D. Milojicic et al., "Peer-to-Peer Computing," HP Labs Technical Report HPL-2002-57, 2002.

[6] M. Jovanovic, F. Annexstein, and K. Berman, "Scalability Issues in Large Peer-to-Peer Networks - A Case Study of Gnutella," University of Cincinnati Technical Report, 2001.