韓國 컴퓨터情報學會 論文誌
第12卷 第6號, 2007. 12.

2007-12-6-1-6

효과적인 역 추적 P2P 자원 검색 알고리즘

김 분 희*

An Effective Backtracking Search Algorithm for the P2P Resources

Boon-Hee Kim*

요 약

P2P 분산 시스템은 네트워크로 연결된 다양한 컴퓨팅 환경 하에 존재하는 유휴 컴퓨팅 자원을 활용함으로써 다양한 연구가 활발히 진행되고 있다. 이는 복수로 존재하는 검색 대상 파일들 가운데 다운로드 시간이 가장 짧은 피어를 대상으로 P2P 통신이 이루어지는 것이 일반적인 방법이다. 여기에 P2P 검색 알고리즘이 복수로 존재하는 검색 대상 파일들 가운데 다운로드 시간이 가장 짧은 피어를 선택하는 기준에 따라 실제 다운로드 시간을 결정하는 가장 중요한 요인이다. 그러나 네트워크 연결성이 약하기 때문에 자원 제공 피어의 오프라인 상태로 전환 될 수 있고, 이때 주로 자원 재전송의 방법을 선택하게 된다. 본 연구에서는 자원 재전송 요구 발생시 성능 개선을 위한 역 추적 자원 검색 알고리즘을 제안한다.

Abstract

The P2P distributed systems are proceeded various studies lively to use the idleness computing resources under the network connected computing environments. It's a general mean to communication from the peer of the shortest downloaded time among same target files to be searched. The P2P search algorithms are very important primary factor to decide a real downloaded time in the criteria to select the peer of a shortest downloaded time. However the peer to give resources could be changed into offline status because the P2P distributed systems have very weakness connection. In these cases, we have a choice to retransmit resources mainly. In this study, we suggested an effective backtracking search algorithm to improve the performance about the request to retransmit the resource.

▸ Keyword : P2P(Peer-to-Peer), 재전송(Re-transmission), 적중(Hit), 평균값(Average Value)

• 제1저자 : 김분희
• 접수일 : 2007.10.25, 심사일 : 2007.11.5, 심사완료일 : 2007.11.12.
* 동명대학교 멀티미디어공학과 전임강사

Ⅰ. 서 론

다양한 컴퓨터 시스템이 공존하는 가운데 소요 비용에 비해 해당 성능이 매우 우수한 분산 시스템에 대한 연구가 활발하다[1]. 이러한 분산 시스템 가운데 P2P 기술은 개개인이 보유한 자원을 효율적으로 공유하는 측면에서 활용도 높은 분야이다. PC 환경이 예전의 워크스테이션 급을 넘어서는 가운데 실제적으로 이러한 컴퓨터를 이용하는 이용자는 성능의 대부분을 이용하지 않는 비효율적인 사용성을 보이고 있다. 실제로 일반 컴퓨터 사용자가 컴퓨터 자원을 가장 많이 이용하는 예는 게임 소프트웨어를 실행하여 조작하는 경우일 것이고, 나머지는 워드프로세서나 인터넷을 이용하는 수준이다. 이 또한 많은 시간은 중요 컴퓨터 자원들이 유휴기 속에서 낭비되고 있는 상황이 대부분이다. 이에 훌륭한 성능의 컴퓨터들이 도처에 산재해 있는 상황에서 해당 네트워크 환경을 이용하는 연구의 유효성은 누구나 인정하게 된다. 이와 더불어 개인이 보유하고 있는 파일과 같은 컨텐츠 자원들을 여러 사람이 공유할 수 있다면 그 이용성 면에서 추앙될 수 있겠다. 이러한 점에서 P2P 시스템 관련 연구가 여전히 활성화 되고 있는 것이다.

효율적인 자용 운용이 가능한 P2P 시스템은 공유된 자원을 보유한 자원 제공 피어의 온라인 상태 유지 여부에 따라 해당 자원에 대한 검색 적중률에 결정적인 기여하게 된다. 그로인해 P2P 관련 연구 가운데 불완전한 네트워킹 환경을 극복 할 수 있는 분야에 대한 연구가 활발히 진행되고 있다[1][2]. 이러한 P2P 환경에서 자원 검색 피어는 원하는 자원에 대한 검색의 과정을 우선적으로 거치게 되고, 이렇게 얻어진 검색 결과를 이용하여 원하는 자원을 검색 참여 피어로부터 다운로드하게 된다. 이때 P2P 시스템에서 늘 발생할 수 있는 경우인 다운로드 완료 전 자원 제공 피어의 오프라인 상태로의 변화를 자주 접하게 된다. 이 상황에서 원하는 자원에 대한 다운로드 완료를 위해서는 주로 재전송의 방법을 이용하게 된다. 그러나 단순히 재전송하는 방법을 이용하게 되면 기본적으로 자원을 검색하는 단계부터 다시 시작되어야 악순환이 이어질 수 있다. 이는 P2P 시스템의 성능 저하의 주요 원인이 되므로 이에 대한 해결책이 필요하다.

본 연구에서는 자원 검색 피어가 해당 자원에 대한 다운로드 작업 중 재전송 요구가 발생하였을 때 해당 요구에 효과적으로 대응하기 위한 알고리즘을 제안한다. P2P 통신 기반 검색 알고리즘이 원하는 검색 대상 파일이 산재해 있는 가운데 다운로드 시간이 가장 짧은 피어를 선택하는 기준에 따라 그 유효성이 높아지는데, 이는 실제 다운로드 시간을 결정하는 가장 중요한 요인이다. 그러나 P2P 환경은 네트워크 연결성이 약하기 때문에 자원 제공 피어의 오프라인 상태로 전환 될 수 있고, 이때 주로 자원 재전송의 방법을 선택하게 되는 것이다. 이에 자원 재전송 요구 발생시 성능 개선을 위한 역 추적 자원 검색 알고리즘을 제안하여 재전송 요구시 대응성을 높이고 해당 P2P 시스템의 신뢰도 향상에 기여하고자 한다. 본 논문의 구성은 다음과 같다. 2장에서 관련연구, 3장에서는 본 논문의 핵심 부분인 제안한 검색 알고리즘을 4장에서 이에 대한 평가를, 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

이 장에서는 본 논문의 연구 분야의 P2P 환경을 기반으로 한 자원 검색 알고리즘의 기존 연구에 대해 설명하고자 한다. 본 장에서는 논의되고 있는 주요 검색 시스템을 분류해 보고, 각 검색 매커니즘의 특징을 살펴보겠다[3][4][5][6].

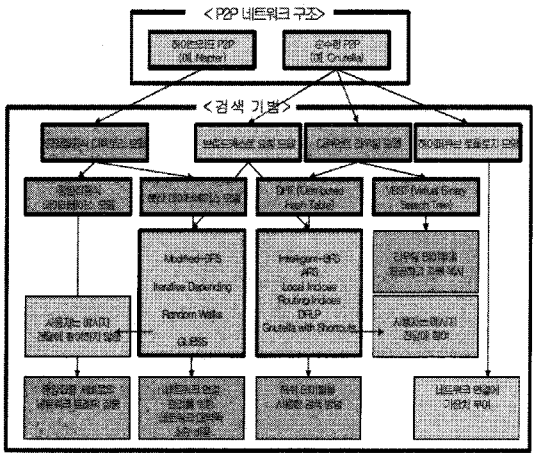

그림 1. P2P 검색 알고리즘 분류 [2]
Fig 1. Classification of P2P search algorithm [2]

P2P 시스템에서 자원의 검색 기법은 [그림 1]의 분류된 바와 같이 P2P 네트워크 구조에 따라 분류를 할 수 있다[4]. 먼저 P2P 네트워크 구조는 피어간의 네트워킹 방법과 역할에 따라 순수한 P2P, 간단한 조회 기능 서버를 가진 P2P, 조회 서버와 룩업 서버를 가진 P2P, 조회/룩업/컨텐츠 제공 기능의 서버를 가진 P2P 모델로 분류할 수 있는

데, 여기서 순수한 P2P 모델을 제외한 나머지 서버의 기능이 있는 모델을 일반적으로는 하이브리드 P2P 모델로 묶어서 분류한다. "하이브리드 P2P 구조에 속한 검색 기법은 중앙집중식 모델 (Centralized Directory Model)로써 대변되는데, 피어의 연결정보를 보유한 중앙 서버가 검색 대상 데이터의 보유 여부에 따라 중앙집중형 데이터베이스 모델(Centralized DB Model)과 분산형 데이터베이스 모델(Decentralized DB Model)로 나뉜다. 순수한 P2P 모델의 경우는 검색 기법의 특징에 따라 브로드캐스트 요청 모델(Broadcast Requests Model), 도큐먼트 라우팅 모델(Document Routing Model), 하이퍼큐브 토플로지 검색 모델(Hypercube Topology Search Model)로 나뉜다.

중앙집중식 디렉토리 모델은 하이브리드 네트워크 구조에서 사용되는 디스커버리 기법으로써 Napster[7]가 대표적인 예이다. 이는 피어의 연결정보를 보유한 중앙 서버가 검색 대상 데이터의 보유 여부에 따라 중앙집중형 데이터베이스 모델(Centralized DB Model)과 분산형 데이터베이스 모델(Decentralized DB Model)로 나뉜다. 중앙 집중형 데이터베이스 모델의 중앙 서버는 피어들의 보유 자원에 대한 메타데이터를 저장하고 관리해야 하는 역할을 하므로 중앙서버의 데이터베이스 구축을 위한 네트워크 트래픽이 많다. 이에 비해 분산형 데이터베이스 모델은 중앙 서버가 피어들의 네트워크 연결 만을 관리한다. 필요한 자원을 요청하고자 하는 피어는 중앙서버에게 질의를 하게 되고, 중앙서버는 이 메시지를 현재 연결되어있는 모든 피어들에게 전달하게 된다. 전달 받은 피어는 자신의 자원 가운데 해당되는게 있다면 해당 메타 데이터를 중앙 서버에게 전달하고, 전달 받은 중앙 서버가 자원 요청 피어에게 메타데이터를 전송하고, 이후 피어들 간의 통신이 일어난다. 따라서 네트워크 대역폭 요구가 많은 편이다.

순수한 P2P 모델에서 브로드캐스트 요청 모델은 검색 메시지를 전달하는데 있어, 피어들의 참여여부에 따라서 모델들 구분할 수 있다. 피어들이 검색 메시지를 전달할 때 정해져 있는 규칙에 의해 메시지 전달하는 방법 가운데, Gnutella는 자원을 찾기 위해 TTL 홉 내에 있는 접근 가능한 모든 피어에 대해서 BFS(Breath First Search) 기반의 브로드캐스트 기법을 사용한다. 이 알고리즘의 검색 기법은 검색의 원형으로써 간단하고 구현이 용이하다. 그러나 자원을 검색할 경우 메시지 내의 TTL을 포함시키기 때문에 네트워크 그룹 내에 참여하고 있는 모든 피어에게 메시지를 전송하지 않는다. 단, TTL 값의 범위 내에서 이 메시지를 받은 피어들은 연결되어 있는 모든 피어에게 메시지를 전송함으로써 상당한 네트워크 트래픽을 유발한다. 변형된 BFS(Modified-BFS)는 브로드캐스트 기법의 변형된 형태로써, 인접피어 중 일정 비율에 해당하는 피어들만을 선택하여 전송하는 방법이다. 인접한 피어들보다 근거리에 있는 피어들의 검색에 중점을 둔 알고리즘으로 메시지의 평균 발생률은 Gnutella에 비해 줄었지만, 메시지를 수신하는 피어 및 검색 결과의 수는 Gnutella와 유사한 특징을 보인다. 반복적 전개법(Iterative Deepening)은 TTL 값을 점점 증가시키면서 BFS 알고리즘을 이용하여 검색하는 방법이다. 사용자가 정의한 특정 검색 결과에 만족할 때까지 TTL의 수를 증가하면서 검색 메시지를 전송한다. 사용자가 정의한 검색 결과에 만족하지 않을 경우 TTL의 값을 증가시키면서 검색 메시지를 전송하기 때문에 표준 브로드캐스트 기법보다 더 많은 트래픽을 발생시킬 수 있다. RandomWalks는 검색 메시지를 전송할 때 검색 메시지의 수를 일정한 수로 제한하여 생성하고 전달한다. 검색 하고자 하는 피어는 k개의 검색 메시지를 인접피어들에 전송하고, 이 메시지를 받은 피어는 검색 메시지에 대한 작업을 처리하게 되며, 검색 메시지 내의 TTL 값이 1 이상이라면 1만큼 감소시킨 후 인접한 피어 중 임의의 피어에게 검색 메시지를 전송하게 된다. 전달되는 검색 메시지들에 의해 검색 수행이 이뤄진 피어는 메시지에 대한 처리를 한 후 검색 메시지를 최초 발생시킨 피어에게 현재 피어의 검색 결과가 만족스러운지에 대해 질의함으로써 메시지 전송 중단을 결정할 수도 있다. 결국 메시지 내에 지정된 TTL 홉 수 만큼 전달될 수 있기 때문에 최악의 경우 검색 메시지는 k*TTL 만큼 발생한다. GUESS[8][9]는 울트라 피어(ultrapeer) 개념을 기반으로 하며, 각 피어들은 해당 울트라 피어에 접속하고 울트라 피어는 자식 피어들의 프락시 역할을 하게 된다. 첫 자식 피어로부터 검색 메시지를 수신한 울트라 피어는 자신의 자식 피어에게 검색 메시지를 전송하게 되고, 검색 메시지에 해당하는 자원이 검색되지 않을 경우 인접한 다른 울트라 피어에게 접속하여 검색 메시지를 전송하게 된다. 다른 울트라 피어로부터 검색 메시지를 받은 울트라 피어는 자신의 자식 피어들에게 검색 메시지를 전송하게 되고, 검색된 자원들의 메타 데이터를 자원 검색을 요청한 피어에게 전송하게 된다. 이러한 작업은 충분한 검색 결과가 나올 때까지 진행되며, 이러한 요청에 있어서 울트라 피어의 순서는 정해져 있지 않다.

순수한 P2P 모델에서 브로드캐스트 요청 모델에서 피어들이 관련 정보를 저장하고 있어서, 이 정보를 이용하여 메시지가 전송될 피어를 결정하여 전송하는 또 다른 형태가

있다. 그 예로 지능적 BFS(Intelligent-BFS) 기법은 변형된 BFS(modified-BFS) 알고리즘을 확장하여 지식기반으로 동작하도록 변형시킨 경우이다. 검색을 하고자 하는 피어는 변형된-BFS 알고리즘을 이용하여 인접 피어에게 검색 메시지를 보내게 된다. 이후 검색 메시지를 받은 피어들 중에서 해당 자원을 가지고 있는 피어는 역경로로 결과값을 전송하게 된다. 이때 검색을 요청한 피어와 결과값을 반환하는 피어 사이에서 결과 메시지를 중재하여 주는 피어들은 결과 메시지로부터 얻게되는 자원이 검색된 피어에 대한 정보를 자신의 데이터베이스에 저장하게 된다. 그 다음 피어로 부터 검색 메시지를 수신하게 되면 해당 피어는 저장된 결과값을 이용하여 검색메시지의 전달 경로를 결정할 수 있게 된다. 이 알고리즘은 네트워크에서 발생되는 검색 메시지 양을 줄일 수는 없지만, 자원이 있을 법한 피어에게 검색 메시지를 전달함으로써 검색의 적중률을 높일 수 있다. 그러나 피어 및 데이터의 삭제와 관련된 메시지는 전송하기 않아 자원의 삭제가 자주 발생하는 환경에서는 부정확한 결과를 낳을 수 있다. APS(Adaptive Probabilistic Search)는 자원을 찾기 위해 TTL 홉 내에 있는 접근 가능한 모든 피어에 대해서 BFS 기반의 인접 피어 중 일정 비율에 해당하는 피어들만을 선택하여 전송하는데, 인접한 피어들보다 근거리에 있는 피어들의 검색에 중점을 둔 알고리즘으로 각 피어가 검색 메시지의 송신과 검색에 대한 결과 메시지의 수신이 했을 때 해당 자원에 대한 방향성을 지역 데이터베이스에 저장하게 되고, 이후 질의 발생시 검색에 대한 적중률을 높이게 한다. LI는 각 피어들이 r홉 내에 있는 모든 피어들이 보유한 자원들의 목록을 유지한다. 하나의 피어가 검색 메시지를 수신할 경우 해당 피어는 r홉 내의 모든 피어에게 보유 자원에 대한 메터데이터를 가지고 있기 때문에 인접한 피어들에게 검색 메시지를 정송한 것과 같은 결과를 얻을 수 있다. 이 방법을 이용함으로써 몇 개의 피어에게만 검색 메시지를 전송하여도 많은 피어들에게서 검색한 것과 동일한 결과값을 얻게되어 낮은 비용으로 충분한 결과물을 획득하는 이점이 있다. RI는 각 피어는 자원을 보유한 실제 피어에 대한 정보를 저장하는 것이 아니고, 그 자원이 있는 곳의 방향을 저장하는 방법이다.

DRLP는 LI 알고리즘과 유사하며, 이 기법을 이용하는 피어들은 처음부터 다른 피어들의 메타 데이터를 저장하지 않는다. 다른 피어부터 검색이 이뤄질 경우 각 피어들은 결과값 및 메터데이터도 함께 전송하여 LI 테이블을 업데이트 한다. 각 피어는 자신이 보유한 자원들을 가리키는 지역 디렉토리와 다른 피어의 지역 디렉토리를 가리키는 디렉토리 캐쉬를 가지게 된다. 만약 검색 메시지를 받은 피어에서 자원이 발견된다면 자원을 발견한 피어는 자원을 요청한 피어에게 역경로와 자원의 메타데이터가 저장된 메시지를 전송하게 된다. 그리고 검색 결과를 전송하는 피어는 해당 디렉토리 캐쉬 정보를 이용하여 역경로로 전송하기 때문에 관련 메시지를 수신하는 피어는 특정피어의 디렉토리 캐쉬 정보를 유지할 수 있게 되는 것이다. 그렇지 않고 자원이 발견되지 않을 경우는 같은 방법으로 이웃 피어에게 메시지를 전송한다. 이 알고리즘을 이용하게 되면 처음에는 자원의 위치를 찾기위해 BFS 알고리즘을 이용하여 많은 검색 메시지를 전송하지만, 그 다음의 검색 과정에서는 디렉토리 캐쉬를 보유하고 있는 피어에 의하여 메시지 전송 과정이 감소하게 된다. 그러나 네트워크 구조가 변경될 경우 BFS 방법을 사용하여 다시 구축하기 때문에 검색 실패율이 증가되며, 네트워크 트래픽 발생율이 늘어나게 된다. GS는 Gnutella 알고리즘을 기반으로 하며, 검색 과정이 이루어진 후 자원을 지닌 피어에 대해 단거리 피어로 등록하는 방법이다. 검색을 원하는 피어는 브로드캐스트 기법을 이용하여 원하는 자원을 찾게 되며, 이후 검색 결과를 받은 피어는 검색된 피어들을 단거리로 이용한다. 따라서 이후의 검색 과정에서 단거리에 등록된 피어와 직접 연결하여 검색 메시지를 전송함으로 검색 시간이 줄어들게 된다. 단거리 경로에 해당하는 피어를 저장하는 단거리 리스트를 이용하여 해당 피어에 접속하게 되고, 그 결과 원하는 자원을 찾지 못하게 되면, 인접 피어로 브로드캐스트 기법을 적용하여 검색하게 된다.

순수 P2P 모델에서 도큐먼트 라우팅 모델은 각 피어가 보유하고 있는 공유 자원의 ID를 기반으로 자원을 검색하는 기법인 DHT는 해쉬 테이블을 이용한 검색 기법으로 모든 피어들은 공유하고 있는 자원에 대한 메타데이터를 해쉬 값에 의해 지정된 피어에게 전송한다. 메타데이터를 수신한 피어는 다른 피어로부터 검색 요청이 들어올 경우 자원의 위치를 알려주는 방식으로 동작하게 된다. VBST는 P-Grid를 이용한 구조로써 피어가 공유하고 있는 자원의 사본들을 네트워크 그룹 내에 있는 여러 피어에게 분포시키고 각 피어로 하여금 필요한 자원을 검색할 수 있도록 라우팅 테이블을 제공하는 방법이다."[2] [그림 1]의 분류 방법과 연관된 P2P 네트워크 기반의 검색 기법들은 이외에도 다수 연구되고 있다[10][11][12][13]. 지금까지 P2P 모델에 따른 기존 검색 기법들과 그 특징에 대해 간략히 설명하였다.

III. 제안 알고리즘

3.1 역추적 알고리즘 요소 설계

본 논문에서는 자원 제공 피어의 오프라인으로 전환시 발생되는 자원 재전송의 상황에서 기존의 TO 알고리즘을 기반으로 한 자원 역 추적 검색 알고리즘을 제안한다. 본 알고리즘의 동료피어 선정 기준은 이전의 동료피어 동반 검색 알고리즘과 같이 다음의 식을 기반으로 이루어진다. 본 논문은 본인의 이전 연구[14]에서 아이디어만을 제시하고 미실험 된 논문을 바탕으로, 구현하고 실험하고 본래의 내용을 수정 및 확장하였다.

$$S = Ran(Size(Grid)) >= \left(\frac{\sum_{i=1}^{n} Hi(hit)}{n} \right) \quad \text{식1)}$$

$$S = Ran\left(\frac{\sum_{i=1}^{n} Hi(hit)}{n} \right) > H(1)) \quad \text{식2)}$$

$$NRlist[i] = Rank(t + R(\frac{TT}{HR})) \quad \text{식3)}$$
$$,0 \leq HR \leq 1, t(j \leq D \leq k)$$

식 1)에서 Size(Grid)는 실험 공간의 전체 네트워크 사이즈이다. H는 현재 피어로부터 떨어진 거리를 나타내는 홉수이다. Hi(hit)는 해당 피어가 이전에 원하는 피어를 찾았던 기록이다. 따라서 이전 기록을 기준으로 히트가 일어난 홉수를 합산하여 전체 횟수 n 만큼 나누어 평균값을 구한다. 식 2)는 임의의 피어를 선정하는데 있어서 최대 홉 수를 이전 히트 기록의 평균수로 하고, 최저 홉 수를 바로 이웃 피어로 하여 검색 피어와의 평균값 보다 가까운 거리의 동료 피어를 선정하는 방법이다.

이러한 동료 피어 동반 검색 알고리즘 기반하에 해당 자원 검색 피어가 다운로드 중에 재전송 요구가 발생할 경우 이 요구에 즉시 대응하기 위한 자원 보유 피어 리스트를 전역 리스트 NRlist에 보관하게 된다. 식3)은 해당 자원 보유 피어 리스트에 해당 자원 보유 피어의 리스트를 우선 순위화 하여 리스트를 만들게 되는 방법이다. TT(Turnaround Time)은 네트워크로 연결된 P2P 환경에서의 해당 자원을 찾음에 대한 결과로써 해당 시스템 내의 응답 시간에 회선 처리 시간과 단말기 처리시간이 합쳐진 응답시간이다. HR(Hit Ratio)은 0과 1사이의 값으로 큰 값이 적중률이 상대적으로 높다. TT와 HR 값을 이용하여 해당 자원 보유 피어의 신뢰도를 측정하게 되고 t을 온라인 평균 유지 시간대에 따른 가산치 적용에 이용되며, Rank는 이들을 이용하여 해당 자원 보유 피어에 정규화 과정을 거친 우선순위를 부여하여 재전송 요구에 즉시 대응할 후보 리스트의 결정 자료로써 적용되는 것이다. 식3)에서 t는 해당 접속 시간대에서의 시스템 접속 시간과 끝의 시간대인 k-j 간격의 너비 정도(D)에 따라 가중치를 부여하여 우선순위 결정의 중요 요소로 적용한다. 이러한 우선순위 기반 재전송 후보 리스트의 선정은 해당 우선순위를 정하는 기준의 적용이라는 복잡한 과정을 거쳐서 선정되므로 계산 시간에서의 추가 비용이 예상되지만 재전송 발생 이전에 이루어지는 작업이므로 재전송 시기의 시간 복잡도에 영향을 미치지 않아 유효하다. 이렇듯 NRlist를 이용하여 재전송 요구에 대응하는 것은 이전 검색 결과값들을 역 추적하여 현재의 우선순위 평가자료로 이용하게 되고, 다양한 결과노드들 간의 효과적인 순위부여 스케쥴링이 적용되는 것이다. 이러한 순위부여 스케쥴링의 핵심인 D는 식4)와 같은 정규화 과정을 거친다.

$$Regu(D) = Selection(j -> pt : k -> pr) \quad \text{식4)}$$
$$,pt \leq j \leq half(time) \leq k \leq pr$$

$$O = Opti(\overleftarrow{\frac{HistoryHit}{RouteStep}}, Random()) \quad \text{식5)}$$

D를 결정하는데 있어 일정한 규칙을 정함으로써 후보 리스트 선정 과정에 필요한 계산의 효율성을 얻어 볼 수 있겠다. D의 정규화를 의미하는 Regu(D)는 해당 시스템의 접속 시간대의 시작을 뜻하는 j와 끝을 뜻하는 k를 관련연구[1]에서 실험으로 증명되었던 평균 시간대의 초반부를 pf라는 기준점으로 정하고, 평균 시간대의 후반부를 pr라는 기준점으로 정하여 해당 범위 내에서 평균 시간대 중반부(half(time))에 근접한 값을 선택하게 된다. 동일한 값이 발생했을 때에 대한 2가지 선택 방법 중 어느 하나를 확정적으로 사용할 수 있겠고, 이 값이 향후 미치는 영향은 실험에서 살펴보도록 하겠다. 관련연구 [1]의 경우 자원보유 피어의 과거 접속 시간에 대한 평균값을 현재 검색 시점의 시간을 기준값으로 5가지 요소로 표현하여, 기존의 기록에

의해 계산된 평균 접속 시간대 보다 현재 검색 시점이 앞선 경우 평균 시간대 이전으로 계산하고, 평균 시간대 중반에 위치할 경우 전체 평균 접속 시간대에서 비교적 안정적인 가중치값을 가질 수 있으며, 평균시간대 후반과 그 이후의 경우 곧 접속이 끊어질 수 있음을 염두하여 가중치 값을 부여하지 않는 방향으로 적용함으로써 가중치 적용에 있어 복합적인 상황을 반영할 수 있고, 이에 따른 유효성을 입증한 바 있다. 식5)는 해당 피어에 기록되어 있는 히트 수와 라우팅 단계를 고려하여 동일한 값이 발생하였을 때 최적치를 구하기 위함이며, 이 결과 또한 동일한 값일 경우 임의 값 발생 방법에 의해 선택되도록 하였다.

3.2 제안한 알고리즘 설계

[그림 2]에서 제안한 알고리즘을 보여주고 있다. 이는 이전 연구의 결과물인 TO 알고리즘에서처럼 순수 P2P 모델을 기반으로 하며, P2P 오버레이 네트워크의 토플로지는 대부분의 P2P 검색 논문에서 적용한 랜덤 그래프 모델을 바탕으로 한다. 또한 질의와 자원의 분포는 Zipfian 이외의 실제 P2P 시스템에서 보여 지는 분포와 유사한 다양한 분포 법칙을 적용한다. 기본 입력값, 인덱스, 자원 보유 노드의 수에 따른 네트워크 구조 설정, 노드의 평균 차수 등에 대한 내용은 TO를 따른다. Walker는 질의 발생 노드의 K 수만큼 질의가 발생된다. 검색 과정상의 각 노드들은 이전의 기록을 기준으로 히트가 일어난 홉수를 평균 합산하여, 최대 홉수(MaxHop)로 기록되어 있는 내용과 자원 요청 피어로 부터의 현재 홉수(NowHop)을 비교하여 우선되는 노드를 선택하게 되고 이로써 검색의 과정상 한 홉이 추가되는 결과에 이른다. 이러한 노드 결정의 과정을 거친 후 요청 노드의 원하는 자원이 존재하는 노드(Rnode)인지를 확인하는 과정을 통해서 실제 자원을 다운로드하게 된다. 이때 원하는 자원이 존재하는 노드를 발견하게 되면 식 3)에서 표현된 자원 보유 피어 리스트의 첫 정보로써 기록되고 자원 다운로드 작업을 실행한다. 이와 같은 과정에서 현재의 노드 검색 작업이 끝나는 것이 아니라 해당 K의 개수만큼 병렬로 위의 과정을 수행하게 된다. 이러한 작업에서 NRlist의 다음 정보들이 누적 기록되는 것이다. 이때 해당 리스트에 포함된 노드들 간의 순위부여 스케쥴링 과정을 거쳐 해당 리스트가 주기적으로 갱신된다. 이렇게 매번 갱신된 정보는 자원 다운로드 메서드의 실행과정에서 해당 피어의 로그오프 상황에 처하였을 때 식 4)에 의해 갱신된 자원 보유 리스트를 바탕으로 우선순위가 가장 높은 다음 노드를 재빠르게 선택하여 나머지 자원 다운로드의 이어받기 작업이 순조롭게 이루어지게 된다.

```
Reverse_Scheduling_Search( ) {
Initialization : pure P2P, random graph model
Definition : Qjnode, Rjnode, i, D, K=1, count=0
Pre_Processor : S from an expression 1) or an
expression 2)
Start Parallel_Section(S).
try {
For (i = 0 ; i <= K; i++) {
  if (MaxHop : NowHop) {
            selection(node);
            addHop();
  } else {
            addHop();
  }
  If (Qjnode == Rjnode) {
      if (count == 0) {
            count++;
            average(Hi);
            NRlist[i]=Rank(priarity); // expression 3)
            execute(Rnode);
      } else {
            average(Hi);
            NRlist[i]=Rank(priarity);
      }
  } else {
    continue;
  }
} //end of repetition
} catch (executeException e) {
            errorMessage(e);
            searchRegu(D);          // expression 4)
            selectionNext(Rnode);
            execute(Rnode);
}
When Broadcast_Flooding(Kwalker_success);
End Parallel_Section.
```

그림 2. 제안한 알고리즘
Fig 2. A suggested algorithm

[그림 3]은 [그림 2] 알고리즘의 자원 다운로드 과정에서 발생할 수 있는 다운로드 중단 시 다운로드 재개 작업의 노드 선택 과정에 관한 알고리즘이다. 우선 NRlist를 만드는 과정에 대한 언급을 구체화하였는데, 관련연구 [3]의 입증된 알고리즘을 바탕으로 정규화 과정을 거친 식 4)의 내용을 바탕으로 노드 선택 과정상의 순위 부여를 위한 작업을 거친다. 이후 [그림 2] 알고리즘이 수행되는 동안 동적으로 증가된 노드 리스트를 바탕으로 순위별 이분할 정렬 과정을 거쳐 해당 리스트에 새로운 노드 정보가 추가될 때마다 즉각적으로 전체 리스트에서의 위치를 부여한다. 이러한 과정을 거쳐 정렬된 노드 리스트에서 현재의 인덱스를 바탕으로 다음 노드를 결정한다. 이때 발생될 수 있는 해당

리스트에 내용이 없는 상태에 대해서는 리스트가 채워진 상태로 변환 될 때 까지 무조건 기다리는 메서드를 수행하게 된다. 이로써 다운로드 재개 작업을 시작하게 된다.

```
Object selection_Next( node n ) {
Definition : Qinode, Rinode, index, point, a, begin,
end
Pre_Processor : P from an expression 3)
Start Realtime_Section.
if ( lim j .OR. lim k ) {
   j->pt      k->pr
  regularPriarity(P);
  For ( i = 0; i (= all.index; i++) {
        point = partition(a, begin, end);
        Sort(vector.a, begin, point-1);
        Sort(vector.a, point+1, end);
  }
}
try {
   a = now.index();
   selectionNext(a+1);
} catch (noListException e) {
   errorMessage(e);
   wait();
}
End Realtime_Section.
```

그림 3. 선택 알고리즘
Fig 3. A selection algorithm

본 논문이 제안한 알고리즘의 복잡도를 계산해 보면, 문제의 크기를 나타내는 파라메터 n에 대하여 본 알고리즘의 검색 과정에서 거치게 되는 노드의 수로 정하였을 때, K 수 20에 대하여 기존의 Gnutella는 20, 380, 7220 등과 같이 n에 관해 가장 빨리 증가하는 항만을 남기고 다른 항은 무시해도 되는 점근적인 복잡도 평가에 의해 $O(n^n)$ 의 결과를 나타내었고, 본 알고리즘의 복잡도는 $O(\log n)$ 으로 알고리즘 복잡도가 확연히 낮음을 확인할 수 있었다.

IV. 평가

이 장에서는 본 논문이 제안한 알고리즘에 대하여 확장형 PeerSim P2P 시뮬레이터를 이용해서 실험한다. 실험환경은 비교 대상인 이전 연구에서와 동일한 환경인 Window Server 2000 운영체제 하에 JDK 개발 환경을 기반으로 소프트웨어 플랫폼을 구성했고, 시스템 사양은 Intel Pentium III 871MHz, 하드디스크 40 Gb, 메인 메모리 256 Mb 환경에서 실험하였다. 해당 실험 환경 하에서 중요 시뮬레이션 인수 가운데 P2P 모델은 순수한 P2P 모델, 그래프 모델은 랜덤 그래프 모델을 기본으로 하며, 노드 수, 평균 노드의 차수(degree), TTL의 기본 설정을 따른다.

본 검색 알고리즘에 대해 Gnutella, TO와 여러 가지 성능에 대해 비교한 결과는 다음과 같다. 우선 [그림 4]는 실험 단계별로 각 알고리즘들의 노드 분산율을 측정한 결과이다. 노드 분산율은 피어들 간의 해당 질의가 전파되면서 특정 노드에 부하가 분산되는 정도를 나타낸다. 실험결과에서 볼 수 있듯이 기존의 Random Walks는 시뮬레이션 단계의 변화에 따른 분산율의 변화에 영향이 거의 없었다. TO의 경우 현재의 알고리즘에 비해 큰 차이를 보이나 실험 단계가 증가되면서 급격한 상승 추이를 보인다. 이러한 TO에 비교되는 본 알고리즘의 경우 기존 결과에 비해 그 폭의 차이가 두드러지는데, 이는 재전송과 관련한 요소가 추가되어 나타난 결과로 판단된다.

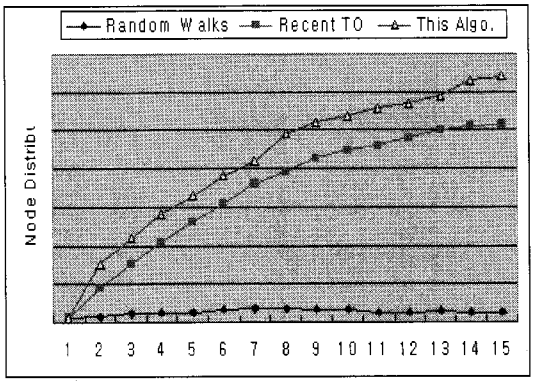

그림 4. 실험 단계에 따른 노드 분산율
Fig 4. Node distribution rates at the simulation steps

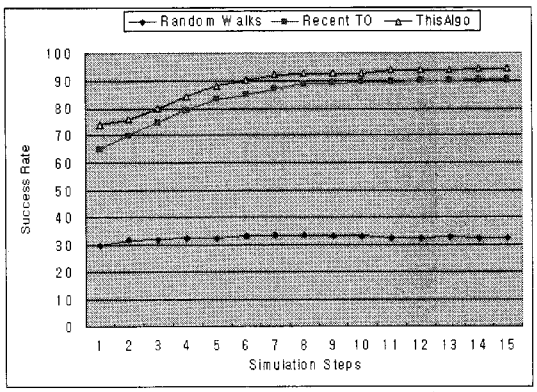

그림 5. 실험 단계에 따른 검색 성공률
Fig 5. Success rates at the simulation steps

〔그림 5〕는 해당 실험 단계에 따른 검색 성공률을 측정하여 비교한 결과이다. Random Walks의 경우 실험 초기의 검색 성공률을 실험 단계가 증가함에 따른 변화에 대해 측정한 결과 실험의 횟수에 따른 영향이 거의 없는 것으로 판단된다. 그러나 TO와 제안한 알고리즘의 경우 실험의 횟수가 증가함에 따른 영향을 파악한 결과 그림과 같이 의미있는 수치로 나타났다. 또한 제안한 알고리즘의 검색 성공률과 TO의 검색 성공률를 비교 하였을 때 전반적으로 고른 성공률 증가 추이를 보이고 있으며, 특히 실험 초기의 성공률은 평균 4.68% 높은 성공률을 보인점을 감안하였을 때 그 2배에 달하는 9%대의 차이를 나타내어 실험 초기 성공률이 두드러짐을 확인할 수 있었다. 이는 실험 단계에 따라 해당 경험치가 미치는 성공률 요소의 상관관계 측면에서 상대적으로 낮은 실험 단계를 감안하였을 때 매우 의미있는 결과라 할 수 있겠다.

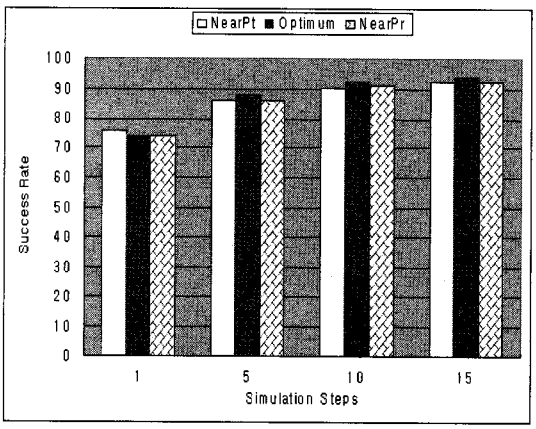

그림 6. 정규화 방법에서 실험 단계에 따른 검색 성공률
Fig 6. Success rates at the simulation steps in regular methods

〔그림 6〕은 식4)에서 언급되었던 정규화 방법에서 평균 시간대의 선택 방법에 대해 실험 단계별 검색 성공률을 측정하여 비교한 결과이다. 식5)의 방법의 경우가 〔그림 5〕에서도 밝힌 바 있는 제안한 논문의 성공률 측정의 기준이 되는 최적화(Optimum) 방법이다. 여기에 평균 시간대 전반부 요소를 적용한 방법(NearPf)과 평균 시간대 후반부 요소를 적용한 방법(NearPr)을 비교한 결과가 〔그림 6〕과 같은데, 실험 단계의 반복 초기에 NearPf방법의 성공률이 다소 우수하나 전체 실험 단계를 보았을 때 최적화 방법의 결과가 우수하였다. 그러나 최적화 방법은 식5)의 계산 과정에서 추가되는 시간 요소를 고려하였을 때 NearPf나 NearPr 방법 중 하나를 결정적으로 사용하는 방법을 선택할 수도 있겠다.

V. 결론 및 향후 연구

P2P 검색 알고리즘은 주로 해당 검색 알고리즘의 검색 성공률은 높이고 자원의 재전송률을 줄임으로써 사용자 편의성을 제공하고 자원을 효율적으로 사용할 수 있도록 하는 것이 주요 역할이다. 본 연구에서는 P2P 시스템에서 피어들 간의 자원 다운로드 작업을 진행하는 동안 재전송 요구가 발생할 수 있는데, 이때 네트워크 연결성이 약한 P2P 환경에서 자원 제공 피어의 오프라인 상태로의 전환 시 대부분 자원 재전송의 방법을 선택한다. 이에 자원 재전송 요구 발생시 성능 개선을 위한 역 추적 자원 검색 알고리즘을 제안하였다. 실험 결과에서도 확인 할 수 있듯이 검색 성공률 및 노드 분산율에서 모두 기존의 방법과 의미있는 차이를 보임으로써 그 유용성을 확인할 수 있었다.

향후 연구과제로는 본 검색 알고리즘에서 동료 피어 선정 기준에 게임 및 애니메이션 분야에서 많이 적용하고 있는 인공지능 기법을 적용하여 좀더 명확한 결과값을 제시하고자 한다.

참고문헌

〔1〕 김분희, "분산 객체의 확률적 비례 검색 기반 전송률 향상 검색 알고리즘", 한국컴퓨터정보학회 논문지, 11권 3호, pp.49-56, 2006. 7.

〔2〕 김분희, "다중 피어 분산 처리 기반 효과적 P2P 검색 알고리즘", 한국컴퓨터정보학회 논문지, 11권 제 5호, pp.29-37, 2006. 11. 30.

〔3〕 P. Ganesan, et al., "YAPPERS: A Peer-to-Peer Lookup Service over Arbitrary Topology", INFOCOM'03, pp.1250-1260, 2003.

〔4〕 B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," ICDCS'02, pp.103-113, 2002.

〔5〕 D. Tsoumakos and N. Roussopolulos, "Analysis and Comparison of P2P Search Methods," Technical Report(CT-TR-4451), University of Maryland, Dept. of Computer Science, 2003.

[6] Boon-Hee Kim, Young-Chan Kim, "Ptops Index Server for Advanced Search Performance of P2P System with a Simple Discovery Server," LNCS 3032, Springer-Verlag, pp. 285-291, 2004. 4.
[7] Napster website: http://www.napster.com
[8] S. Daswani and A. Fisk, Gnutella UDP extension for scalable searches (GUESS) v0.1, White Paper, 2002.
[9] Gnutella website: http://gnutella.wego.com
[10] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," ICDCS, pp.103-113, 2002.
[11] D. Tsoumakos and N. Roussopoulos, "A Comparison of Peer-to-Peer Search Methods," WebDB, pp.61-66, 2003.
[12] Daniel A. Menasce and Lavanya Kanchanapalli, "Probabilistic Scalable P2P Resource Location Services," ACM SIGCOMM, pp.48-58, 2002.
[13] Sean C. Rhea and John Kubiatowics, "Probabilistic Location and Routing," INFOCOM02, pp.336-346, 2002.
[14] 김분희, "자원 역 스케쥴링 기반 P2P 검색 알고리즘", 제 27회 한국정보처리학회 춘계 학술발표대회, 제14권 1호, 2007. 5.

저 자 소 개

김 분 희

2005년 2월 : 중앙대학교 컴퓨터공학과 공학박사
2005년~현재 : 동명대학교 멀티미디어공학과 전임강사
〈관심분야〉 분산 시스템, P2P 검색 기법, HCI(Human Computer Interaction)