韓國 컴퓨터情報學會 論文誌
第13卷 第7號, 2008. 12.

2008-13-7-2-4

다방향 프로세싱을 위한 P2P 검색 알고리즘 제안

김 분 희*

A Suggestion of P2P Search Algorithm for Multidirectional Processing

Boon-Hee Kim*

요 약

P2P 분산 시스템에서 각 피어들은 항상 온라인 상태를 유지하지 않는다. P2P 시스템은 각 피어들은 하나의 시스템의 구성 요소로써 유기적으로 작동되는 일반적인 시스템과 다른 것이다. 이런 경우 P2P 시스템을 이용하는 사용자에게는 신뢰성이 떨어지는 시스템이 될 수 있는 것이다. 이에 많은 P2P 연구에서 이러한 문제를 해결하고자 시도하고 있다. 본 연구에서는 다운로드 되고 있는 자원에 대해 신뢰성 있는 연결을 제공하기 위해 다방향 프로세싱 P2P 검색 알고리즘을 제안한다. 기존 연구에서는 새로운 자원 제공 피어의 선택을 위한 로딩 부담률이 높았는데 본 연구에서는 이를 해결하고자 한다.

Abstract

In the P2P distributed system, such peers don't always keep on-line status. P2P systems are different from the ordinary systems that such peers are organically operated as the elements of system. In this case, users using P2P system are able to loss the reliability. At this point, many P2P studies are tried to resolve these problems. In this paper, we suggest a P2P search algorithm for multidirectional processing about the resources to be downloading. In existing studies, the loading rate is very high to select new resources supported peer, so we suggest new solution in this study.

▸ Keyword : 다방향(Multi-Direction), 재전송(Re-transmission), P2P(Peer-to-Peer), 신뢰성(Reliability)

• 제1저자 : 김분희
• 접수일 : 2008. 10. 6, 심사일 : 2008. 11. 12, 심사완료일 : 2008. 12. 24.
* 동명대학교 멀티미디어공학과 전임강사

Ⅰ. 서 론

P2P 분산 시스템은 인터넷이 활성화되고 네트워크에 연결된 컴퓨터들이 일반화 되면서 유휴 자원을 활용하는 측면과 데이터 공유의 측면에서의 장점으로 다양한 형태로 발전되고 있다. 네트워크 속도 및 컴퓨터의 성능 또한 예전의 그것과는 확연히 달라져 고성능을 자랑하고 있다. 이에 P2P 시스템의 활용도를 더욱 높일 수 있도록 성능 향상 측면에서의 연구가 활발히 진행되고 있다.[1][2]

P2P 시스템의 가장 큰 특징은 네트워크로 연결된 피어들 간의 연결 상태를 보장할 수 없다는 것이다. 네트워크 간 연결 가능한 상태임을 확인하고 P2P 자원 공유 작업이 이뤄졌다 하더라도 어느 순간 자원 제공 피어가 오프라인으로 변경될 수 있는 것이다. 이와 같은 상황이 되면 새로운 자원 공유 피어를 찾는 작업부터 시작하여야 하고 이로 인한 시간적인 손실이 초래되는 것이다. 해당 시스템의 신뢰성 측면에서 이러한 불확실한 연결성은 불리한 점으로 작용할 수 있는 것이다. 따라서 P2P 시스템의 특징이기도 한 불확실한 연결성을 보완할 수 있는 방법들에 관한 연구가 활발히 진행되고 있는 것이다.[3][4][5] 이러한 연구는 미완료된 다운로드 작업을 즉시 자동 대체하여 P2P 시스템 이용자 입장에서는 순조롭게 진행되고 있는 상태로 비춰지게 하는 것이다. 이때 다운로드 작업을 대체하게 되는 피어를 선택하는 과정에서 오버로드를 최소화 하는 것이 관건이다.

본 연구에서는 P2P 시스템을 이용하여 자원 다운로드 작업을 수행하는 가운데 발생할 수 있는 비연결성 문제에 대해 미완료된 다운로드 작업을 자동으로 마칠 수 있도록 대체 자원을 제공할 피어를 선택하는 과정에서의 오버로드를 최소화 하기위한 다방향 프로세싱 알고리즘을 제안한다. 이전 연구에서 미완료된 다운로드 작업을 대체할 피어를 선택하는데 있어 자원 제공 시작 시점에서 미리 정해진 알고리즘에 의해 대체 피어들을 지정해 놓고 비연결 시점에 바로 우선순위가 높은 대체 피어로의 작업 이행이 이뤄질 수 있는 방법을 제시한 바 있다. 그러나 이러한 방법은 하나의 적중률 높은 피어에 로드가 집중되어 효율적인 네트워크 자원 활용 면에서 취약하다 할 수 있다. 본 연구에서 제안하는 다방향 프로세싱은 대체 피어를 선택하는 과정에서의 부담률 뿐만 아니라 자원 제공 피어 자체의 로드 부담률을 줄이기 위해 여러 피어로부터 적절한 할당량으로 나뉜 만큼의 자원 제공 부담을 져 특정 피어에 집중된 로드를 분산하고자 한다. 본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 3장에서는 다방향 프로세싱 알고리즘을 4장에서는 본 연구에 대한 평가를 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

기존의 P2P 시스템에서 피어들 간의 자원 다운로드 작업이 이뤄지는 과정에서 자원 제공 피어의 오프라인 전환이 나타나 다운로드 작업이 미완료 될 수 있는데, 이러한 상황에서 다운로드 작업을 완료할 수 있도록 새로운 피어를 선택하는 다양한 알고리즘들이 제시되고 있다. 새로운 피어를 선택하는 방법 가운데 대표적으로 TO(The Orienteering) 알고리즘 기반의 방법은 본 연구의 연장선상에 있다. 이는 자원 제공 피어의 오프라인으로 전환시 발생되는 자원 재전송의 상황에서 기존의 TO 알고리즘을 기반으로 한 자원 역 추적 검색 알고리즘이다. "이 알고리즘의 동료피어 선정 기준은 이전의 동료피어 동반 검색 알고리즘[6]과 같이 다음의 식을 기반으로 이루어진다.

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

$$S = Ran(\left(\frac{\sum_{i=1}^{n} Hi(hit)}{n}\right) > H(1)) \quad \cdots\cdots (2)$$

$$NRlist[i] = Rank(t + R(\frac{TT}{HR})) \quad \cdots\cdots (3)$$
$$, 0 \leq HR \leq 1, t(j \leq D \leq k)$$

$$Regu(D) = Selection(j -> pt : k -> pr) \quad \cdots\cdots (4)$$
$$, pt \leq j \leq half(time) \leq k \leq pr$$

식 1)에서 Size(Grid)는 실험 공간의 전체 네트워크 사이즈이다. H는 현재 피어로부터 떨어진 거리를 나타내는 홉 수이다. 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.

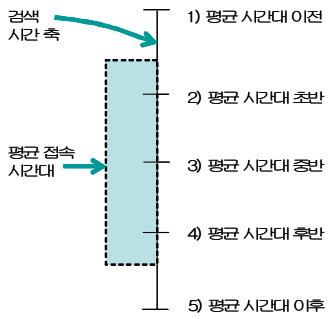

그림 1. 평균 접속 시간대에 따른 검색 시간대 추정[6]
Fig 1. Estimate of a search time zone according to the average connected time zone

```
Reverse_Scheduling_Search( ) {
Initialization : pure P2P, random graph model
Definition : Qjnode, Rjnode, i, Niindex, K=1
Pre_Processor : S from an expression 1) or an
expression 2)
Start Parallel_Section(S).
   For (i = 0 ; i <= K; i++) {
    If (Niindex == null) {
      For ( i = 0 ; i <= neighbor; i++ ) {
        If ( Probabilistic.value[i] > Max ) {
          Max = Probabilistic.value[i] ;
      } }
     Neighbor_Select(Max);
   }
   else {
     Top_List(all_of_Niindex);
     Select_Max(Top_List.Probabilistic.value);
     Neighbor_Select;
   }
   If (Qjnode == Rjnode) {
          Backward_Forwarding(update_index);
          For (All neighbors of the hit node) {
            TO_hit_ratio_update
            constantV+recentTOHitRatio(0..1) }
   }
    When Broadcast_Flooding(Kwalker_success);
End Parallel_Section.
}
```

그림 2. TO 기반 동료 피어 동반 검색 알고리즘[6]
Fig 2. Peer search algorithm with partner based on TO

그림 2는 해당 피어의 이전 기록에 따른 평균 접속 시간에 대해 현재 검색 시점의 시간의 경우를 검색 시간 축에 표현된 바와 같이 5가지의 경우로 표현하였다. 기존의 기록에 의해 계산된 평균 접속 시간대 보다 현재 검색 시점이 앞선 경우 평균 시간대 이전으로 본다. 이는 평균 접속 시간대의 연장선상에서 평균 시간대 초반의 경우와 마찬가지로 우수한 가중치 값을 가질 수 있는 조건으로 판단된다. 평균 시간대 중반에 위치한다면 전체 평균 접속 시간대의 너비에 영향을 받겠지만 비교적 안정적인 가중치값을 가질 수 있겠다. 평균시간대 후반과 그 이후의 경우 곧 접속이 끊어질 수 있음을 염두하여 가중치값을 부여하지 않는 방향으로 적용할 수 있다. 그러나 평균시간대 후반의 경우 평균 접속 시간대의 너비 변화가 큰 피어의 경우를 고려한다면 낮은 가중치 부여의 기회를 부여할 수도 있겠다.

$$NRlist[i] = Sq(time) \quad \cdots\cdots (5)$$

식 5)는 찾아진 시간 time이 이른 순서대로 i가 결정되어 해당 재전송 리스트의 상위 순위를 결정하게 되는 경우로 순위 결정 계산이 불필요하여 해당 연산과 관련된 비용을 줄일 수 있다. 이는 과거 자료에 대한 예측치가 개입되지 않았으나 현재의 검색 시행 노드로부터의 거리가 가깝거나 네트워크 속도가 높은 순위로 자연스럽게 반영되어 처리되는 효과가 있다. [그림 2]는 동료 피어 동반 검색 알고리즘의 전체이다. 기본 입력값, 인덱스, 자원 보유 노드의 수에 따른 네트워크 구조 설정, 노드의 평균 차수 등에 대한 내용은 TO(The Orienteering)를 따른다. Walker는 질의 발생 노드의 K 수만큼 질의가 발생되어 K 수에 따라 전체 메시지 발생량에 큰 영향을 줌으로 적절한 수의 K를 결정하는데 있어 주요한 역할을 하고, 이웃노드에 관한 연관성 확률값 보유 인덱스인 Ni 인덱스의 값이 초기값 null을 가지고 있는 경우 이웃 노드를 선택하는 기준은 오로지 온라인 상태에 대한 확률값이다. 동료 피어는 기준식에 의해 임의의 값 S로 결정되고, 해당 TO의 검색 작업이 원래의 자원 검색 노드와 S 노드가 동시에 병렬로 동작되게 된다. 이러한 과정에서 원하는 자원을 발견하였을 시 질의 진행 방향과 역방향으로 연관성 확률 값을 갱신하면서 질의 발생 피어에 도달하게 된다. 즉 질의를 발생한 노드 Qj 노드와 부합하는 자원을 지닌 노드 Rj 노드를 찾은 경우 역방향으로 연관성 확률 값을 갱신하면서 질의 발생 피어에 도달하게 되는 것이다."[6] P2P 시스템에서 다운로드 미완료 작업에 대한 해결안으로써 새로운 피어를 선택에 대한 연구는 이외에도 다수 존재한다.[3][4][5]

III. 제안 알고리즘

본 논문에서 제안하는 알고리즘은 관련 연구에서도 살펴본 바와 같이 P2P 시스템을 이용하여 자원 다운로드 작업을 수

행하는 가운데 발생할 수 있는 비연결성 문제에 대해 미완료된 다운로드 작업을 자동으로 마칠 수 있도록 대체 자원을 제공할 피어를 선택하는데 있어서의 오버로드가 존재할 수 있는데, 이러한 과정에서의 오버로드를 최소화 하기위한 다방향 프로세싱 알고리즘을 제안한다. 다방향 프로세싱이란 원래의 작업을 대체할 다운로드 제공 피어를 선택하는 과정에서의 부담률과 자원 제공 피어 자체의 로드 부담률을 줄이기 위해 하나의 피어가 아닌 여러 피어로부터 자원 제공 부담을 분산하여 특정 피어에 집중된 로드를 분산하고자 한다. 다음은 제안하는 시스템의 전체적인 구성 요소이다.

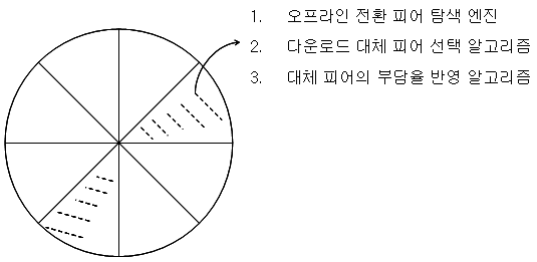

그림 3. 시스템 구성
Fig 3. Construction of System

다방향 프로세싱 시스템은 [그림 3]과 같이 오프라인 전환 피어 탐색 엔진, 다운로드 대체 피어 선택 알고리즘, 대체피어의 부담률을 반영한 알고리즘으로 구성된다. 이러한 구성요소의 기본 설정 사항은 사용자의 요구에 따라 한번에 다운로드 받을 자원 제공 피어의 수이다. 해당 자원의 크기를 자원 제공 피어의 수로 나누어 각 피어의 다운로드 양을 할당한다(자원 할당량 =(자원의 크기)/(자원제공 피어의 수)). 그리고 이전 연구 TO 알고리즘[6]에서처럼 자원 제공 피어의 수 만큼 K가 결정되며, K개의 검색 루트를 통해 자원 제공 피어를 탐색하고 [그림 3]과 같은 자원 제공 피어를 구성하게 된다. [그림 3]은 8개의 자원 제공 피어를 선택했을 경우에 해당되는데, 빗금친 피어는 다운로드 과정에서 오프라인으로 전환된 피어를 나타낸다. 이렇듯 오프라인 피어가 발생했을 때 해당 작업을 대체해야 하는데, 본 시스템은 단순하게 다운로드 남은 작업이 가장 적은 피어 순으로 오프라인 피어 작업을 맡는다. 오프라인 된 피어 하나 당 하나의 대체피어에 해당 다운로드 작업량을 부여하게 됨으로 [그림 3]과 같이 2개의 오프라인 피어가 발생한 경우 해당 작업을 부여받을 피어 또한 2개를 선택하는 작업이 필요하다. 이는 대체 피어의 개수에 따라 제한을 두게 되는데, 8개의 자원 제공 피어 가운데 7개가 오프라인이라면 하나의 피어가 모든 작업을 할당해야 하는 것이다. 즉, 오프라인 피어의 개수를 최대값으로 대체 피어를 결정해야 한다. 다방향 프로세싱 시스템은 이러한 기본 알고리즘을 바탕으로 하며, 다운로드 과정에서 발생할 수 있는 오프라인 전환 피어를 탐색하기 위해 해당 시스템이 작동중인 동안 데몬과 같이 계속해서 감시체제를 가동한다. 이는 사용자의 다운로드 요구가 들어온 다음부터 해당 작업이 마무리 될 때 까지 지속된다. 다음으로 오프라인 전환 피어가 발생한 다음에는 해당 피어를 탐지하고, 대체피어를 선택하는 다운로드 대체 피어 선택 알고리즘이 가동된다. 이는 다운로드 대체 피어가 될 피어를 선택하는데 있어 해당 피어의 부담률을 반영하여 결정하도록 대체 피어의 부담률을 반영한 알고리즘의 적용하에서 작동된다.

다음 [그림 4]는 다방향 프로세싱 알고리즘의 다운로드 대체 피어의 부담률을 반영하고, 이로써 해당 작업을 대체할 피어를 선택하는 부분이다. 해당 시스템의 초기화 내용은 TO 알고리즘과 동일하며, 오프라인 전환 피어의 개수에 따라 알고리즘의 진행이 결정되므로 미리 처리되어야 할 사항이다. max는 전체 온라인 피어의 개수, Amount_Remain은 각 피어의 남아있는 다운로드 량을 나타낸다. 각 피어의 남아있는 다운로드 량을 비교하여 전체를 정렬하여 순위를 정하게 되고, 다운로드 량이 적은 최하위 순으로 대체되어야할 피어를 결정한다. 이렇게 선택된 피어들은 현재의 다운로드 량에 추가적으로 대체하여 수행할 다운로드 량이 부가된다. 이렇게 다운로드 대체 피어를 선택한 이후에는 원래의 다운로드 처리를 각 피어에서 수행하게 된다.

```
Multidirectional_Processing( ) {
Initialization : pure P2P, random graph model
Pre_Processor : counting_offline_peer()
Start. Section from counting_offline_peer
   For (i = 0 ; i <= max; i++) {
      If ( Amount_Remain[i-1] > Amount_Remain[i] )
   {
            t = Amount_Remain[i-1];
            Amount_Remain[i-1] = Amount_Remain[i];
            Amount_Remain[i] = t;
      }
   }
   For (i = 0; i < count(Alternated_Peer) ; i++) {
     While (Alternated_Peer == true) {
        Amount(Alternated_Peer)                    +=
   Amount_Remain[i-1];
     }
   }
   Jump Download_Process.
End Section.
}
```

그림 4. 다방향 프로세싱 알고리즘
Fig 4. Multidirectional Processing Algorithm

〔그림 5〕와 같이 다방향 프로세싱이 이뤄지기 위해 TO 알고리즘과 같이 기존의 피어 성공률을 기반으로 한 자원 제공 피어의 선택 과정을 따르고 있다. 이에 따라 우선순위를 정하여 자원의 나누어진 분량만큼을 순서대로 배당하게 된다. 이는 다운로드 완료 후 하나의 완성된 자원으로 배치하는 기준이 된다. 이러한 상황에서 다운로드 작업이 진행되는데, 다방향 프로세싱 알고리즘이 발생하는 상황 즉 오프라인으로 전환된 피어가 발생한 상황에서는 이에 적합한 다운로드 재설정 작업이 이뤄지고, 다시 재설정된 상황에서 다운로드 작업이 진행된다. 다방향 프로세싱 알고리즘이 발생하는 상황에 대해서는 무한루프 하에서 정해진 시간마다 재검사하게 되어 있으며, 이러한 시간 간격의 정함 또한 사용자의 초기 설정값에 기인한다. 시간 간격의 정함에 따라 오프라인으로 전환된 피어의 발견이 늦어져 전체 수행 결과 다운로드 속도 면에서 불이익이 초래될 수도 있고, 잦은 재검사 작업으로 인한 불이익이 발생될 수도 있겠다.

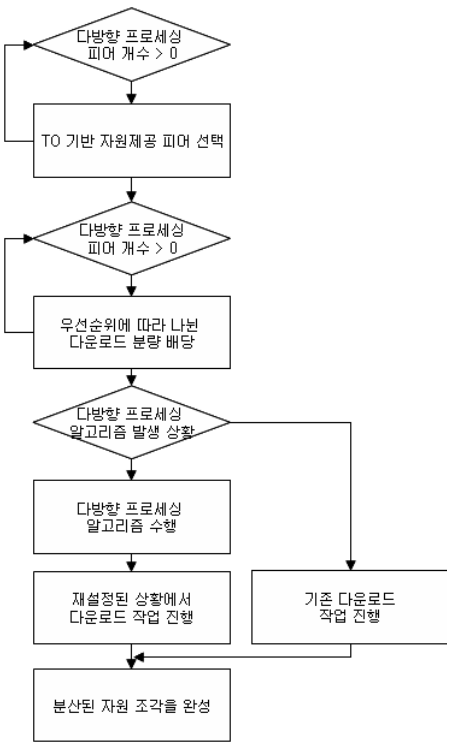

그림 5. 다방향 프로세싱 순서도
Fig 5. Flowchart for multidirectional processing

제안한 알고리즘은 다음과 같은 선택피어 미완료율, 다운로드 속도를 이용하여 그 효율성의 측정기준으로 한다.

선택피어 미완료율
= (미완료 리스트)/(미완료 리스트 + 다운로드 리스트)
다운로드 시간 = (다운로드 끝나는 시점의 시간)
- (다운로드 시작 시점의 시간)

선택피어 미완료율은 0과 1사이의 값을 갖는데, 미완료 리스트와 다운로드 완료된 리스트를 합한 값을 분모로 하고 미완료 리스트를 분자로 하여 나눈 값이다. 1에 가까운 값일수록 선택피어 미완료율이 높아지고 반대로 0에 가까운 값일수록 선택피어 미완료율이 낮다. 다운로드 시간은 말 그대로 다운로드 시작시점과 끝시점의 시간을 뺀 간극이 되겠다.오프라인 전환율((오프라인 전환 피어 수)/(전체 피어 수)) 또한 측정해 볼 수 있겠는데 전체 피어의 수를 분모로 하고 분자로 오프라인 전환 피어 수로써 전체 피어의 수 보다 오프라인 전환 피어 수가 더 많은 경우가 발생할 수 있어서 1 이상의 값이 나올 수 있다. 이는 오프라인 전환 피어가 다시 온라인 상태로 돌아오는 경우도 있으며, 1 이상의 값이 나올 경우 오프라인 전환율 자체가 성능에 미치는 영향이 클 가능성이 높다.

IV. 시뮬레이션 결과

이 장에서는 본 논문이 제안한 알고리즘에 대해 실험을 통해 평가하고자 한다. 실험환경은 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 TO 알고리즘의 선택을 따른다. 성능 비교의 기준이 되는 시뮬레이션 횟수는 각 항목에 대해 10회로 한정하였다. 또한 시뮬레이터는 Peersim을 실험 환경에서 이용할 자원의 사이즈는 500M를 다방향 에지의 최대수는 11로 한정한다. 본 알고리즘에 대해 여러 가지 성능 비교 결과는 다음과 같다. 우선 선택피어 미완료율을 측정하였는데, 다방향 에지(edge)의 수에 따라 〔그림 6〕과 같은 데이터를 얻었다. 다방향 에지의 수가 2, 3일 경우 선택피어 미완료율이 제일 높았고, 6, 7일 때 가장 낮았다. 또한 다음 결과에 대해 각 각 정렬을 통해 분석해 보면, 에지수가 6, 7일 때와

8-9일때 미완료율이 0인 경우가 가장 많았으며 0.1대의 미완료율을 나타내는 항목은 에지수 6, 7일 때와 10, 11일 때 나타났다. 전반적으로 에지수 6, 7일 때 가장 낮은 선택피어 미완료율을 나타내고 있다.

	다방향 에지(edge) 개수				
시뮬레이션 횟수	2-3	4-5	6-7	8-9	10-11
1회	0.58	0.60	0.00	0.00	0.34
2회	0.42	0.00	0.23	0.00	0.10
3회	0.00	0.23	0.15	0.00	0.34
4회	0.42	0.00	0.15	0.28	0.38
5회	0.67	0.23	0.00	0.51	0.29
6회	0.58	0.48	0.00	0.00	0.46
7회	0.42	0.80	0.45	0.34	0.52
8회	1.00	0.00	0.00	0.24	0.37
9회	0.42	0.00	0.31	0.00	0.65
10회	0.00	0.55	0.00	0.24	0.00
평균	0.39	0.23	0.13	0.16	0.31

그림 6. 선택피어 미완료율
Fig. 6. Rate when download work is'nt completed at selected peer

〔그림 7〕는 다방향 에지 개수에 따른 총 다운로드 완료 시간을 나타냈다.

	다방향 에지(edge) 개수				
시뮬레이션 횟수	2-3	4-5	6-7	8-9	10-11
1회	198	138	101	112	213
2회	195	55	111	48	93
3회	151	129	128	167	41
4회	169	74	58	61	110
5회	182	113	141	63	38
6회	64	198	128	49	58
7회	145	195	161	102	126
8회	190	140	153	74	104
9회	201	103	49	95	62
10회	88	64	102	60	166
평균(초)	158	121	113	83	101

그림 7. 다운로드 시간
Fig. 7. Download time

다운로드 평균 시간이 가장 짧은 경우는 에지수가 8, 9일 때이고, 가장 긴 경우는 에지수가 2, 3일 경우로 나타났다. 가장 짧은 다운로드 시간을 나타낸 경우는 에지수 10, 11일 때로 38초를 나타냈고, 40초대로는 에지수 6, 7, 8, 9일 때 나타났다. 가장 긴 다운로드 시간대로 213초는 에지수 10, 11일때 나타났다. 100초 이하의 값이 가장 많이 나온 에지수는 다운로드 평균 시간이 가장 짧았던 경우과 동일하게 에지수 8, 9일때 나타났다. 선택피어 미완료율과 다운로드 완료 시간을 측정한 실험 결과에서 6~9의 에지수를 가졌을 때 가장 좋은 성능을 보여줌을 알 수 있었다. 기존의 TO 알고리즘과 선택피어 미완료율에 대해 비교해 본 결과 시뮬레이션 10회 평균값 0.24에 못미치는 0.33로써 본 알고리즘의 성능이 크게 향상 되었음을 알 수 있었다.

V. 결론 및 향후 연구

P2P 시스템은 네트워크로 연결된 피어들 간의 자원 공유의 목적으로 이용되는데, 해당 피어들 간의 연결성이 보장되지 않는다는 것이 가장 큰 특징이다. 이로써 자원 다운로드 과정에서의 완결성을 보장하지 못하게 되어 신뢰성 있는 시스템으로써의 보완 방법이 요구된다. 이에 본 논문에서는 자원 제공 피어의 오프라인 상태로의 변환 과정에서 미완료된 다운로드 작업에 대해 자동으로 완결될 수 있도록 다운로드 작업 대체 피어를 선택하는 과정이 필요하게 되고, 이러한 선택의 과정에서의 오버로드를 최소화 하기위해 다방향 프로세싱 알고리즘을 제안하였다. 결과적으로 다방향 에지수가 6~9인 상태에서의 성능이 가장 우수했으며, 사용자의 다방향 에지수 선택의 상황에서 제안될 수 있겠다.

향후 연구로는 다방향성의 틀을 가져가면서 최적의 성능을 제공하기 위한 다양한 요인들을 분석하고 제안하여 그 유용성을 입증하고자 한다. 또한 교육 시스템에서의 P2P 관련 연구〔7〕들이 활발하게 진행되고 있는데 관심 영역을 확대하는 기틀로써 적용하고자 한다.

참고문헌

〔1〕 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.

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

〔3〕 Daniel A. Menasce and Lavanya Kanchanapalli, "Probabilistic Scalable P2P Resource Location Services," ACM SIGCOMM, pp.48-58, 2002.

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

〔5〕 엄남경,우성희,이상호, "SVM을 이용한 플로우 기반 P2P 트래픽 식별", 한국컴퓨터정보학회논문지, v.13, no.3, pp.123-130, 2008.5.

〔6〕 김분희, "효과적인 역 추적 P2P 자원 검색 알고리즘", 한국컴퓨터정보학회 논문지, 12권 6호, pp.49-58, 2007. 12. 31.

〔7〕 김진일, 황윤철, "사이버 교육을 위한 P2P 기반 콘텐츠 전송시스템에서 효율적인 부하 분산 정책", 한국컴퓨터정보학회논문지, v.12, no.6, pp.31-39, 2007.12.

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)