韓國컴퓨터情報學會 論文誌
第14卷 第10號, 2009. 10.

2009-14-10-3

개선된 다방향 프로세싱 기반 P2P 검색 알고리즘

김 분 희*

Peer to Peer Search Algorithm based on Advanced Multidirectional Processing

Boon-Hee Kim*

요 약

분산 컴퓨팅 분야에서 P2P 기술은 네트워크로 연결된 피어들 간에 자원을 공유하기 위한 다양한 방법을 제시하고 있다. 이는 소수의 서버에 의해 자원을 제공하는 중앙 집중적 네트워크 이용 피어에 비해 자원 활용 측면에서 더욱 유용하다. 그러나 이러한 P2P 시스템을 구성하는 피어들은 항상 온라인 상태가 아님으로 시스템 이용자에게 높은 신뢰성을 제공하기 어렵다. 본 논문의 이전 연구에서는 새로운 자원 제공 피어의 선택을 위한 로딩 부담률을 낮추는데 기여하고 있으나 자원을 다운로드 하기위한 작업을 분담하는 피어들을 선택하는 방법이 가장 낮은 작업량의 피어를 선택하는 방법으로 매우 단순하다. 본 연구에서는 피어들의 평균 성공률 평가치를 기반으로 예측에 의한 피어의 오프라인 발생 빈도를 낮추어 높은 신뢰성을 제공하고자 한다.

Abstract

A P2P technology in distributed computing fields is presented various methods to share resources between network connected peers. This is very efficient that a degree of resources to good use as compared with peers by using centralized network by a few servers. However peers to compose P2P system is not always online status, therefore it is difficult to support high reliability to user. In our previous work of this paper, it is contributing to reduce the loading rates to select of new resource support peer but a selection method the peers to share works to download resources is very simple that it is just selected about peer to have lowest job. In this paper, we reduced frequency offline peers by estimate based on a average value of success rates for peers.

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

• 제1저자 : 김분희
• 투고일 : 2009. 09. 21, 심사일 : 2009. 10. 17, 게재확정일 : 2009. 10. 22.
* 동명대학교 미디어공학과 전임강사

Ⅰ. 서 론

네트워크로 연결된 컴퓨팅 환경이 일반화 되면서 개개인이 소유한 컴퓨터에서 보관하고 있는 자원을 공유하는 역할의 P2P 시스템은 자원을 제공하는 피어와 해당 피어로부터 자원을 제공받는 피어로 구성되며 자원제공 피어의 유휴 자원을 활용하는 측면에서 다양한 형태의 연구가 진행되고 있다.[1][2][3][4] P2P 시스템은 자원 제공 피어가 온라인 상태인 동안에 자원을 공유하지만 자원을 제공받는 피어가 원하는 파일을 모두 다운로드 받지 않은 상태여도 자원 제공 피어의 오프라인 상태로의 변환에 의해 영향을 받게 된다.

P2P 시스템은 네트워크로 연결된 피어와 P2P 시스템 참여자 사이에 많은 자원을 공유하는데, 이는 특정 서버에 의해 운영되는 중앙 집중적 연결에 의존하는 경우보다 네트워크로 연결된 피어에 더욱 유용하다. P2P 시스템은 네트워크로 연결된 환경에서 보유하고 있는 자원의 공유 수단으로써 그 특성상 모든 피어들은 항상 온라인 상태를 유지하고 있지 않다. 이는 P2P 시스템의 가장 큰 특징으로 네트워크로 연결된 피어들 간의 연결 상태를 보장할 수 없음을 의미한다. P2P 시스템으로부터 자원을 제공 받는 피어 입장에서는 자원 제공 피어와의 네트워크 연결 상태를 확인하고 자원 공유 작업의 선택 과정이 시작되었지만, P2P 시스템의 큰 목표인 해당 컴퓨팅 자원의 유휴 시간동안에 자원을 공유하고자 하는 것이지 자원 공유의 목적으로 피어가 컴퓨팅 환경을 온라인 상태로 유지하는 것이 아니므로 종종 자원 제공 피어의 오프라인 상태로의 변화는 불가피하다. 이러한 상황이 빈번 할 경우 P2P 시스템 사용자로써는 신뢰성 떨어지는 서비스로 인식될 것이다.

자원을 제공받는 피어의 입장에서는 이러한 상황에서 새롭게 동일한 자원을 제공하는 피어를 다시금 검색하여 해당 작업을 반복하는 절차가 진행되어야 하는데, 이로 인한 시간적 손실이 초래된다. 이로 인해 불확실한 연결성 문제를 보완할 방법에 관한 연구가 활발히 진행되고 있다.[1][2][5] 이와 같은 연구에서는 일반적으로 미완료된 작업에 대해 즉시적인 대체가 가능한 대안 피어의 선택 방법을 핵심으로 다운로드 작업을 대체하게 되는 피어를 선택하는 과정에서 발생할 수 있는 오버로드를 최소화하기 위해 다양한 대안이 제시되고 있다.

이전 연구에서 P2P 시스템을 이용한 자원의 다운로드 작업을 수행하는 가운데 발생할 수 있는 자원 제공 피어의 오프라인 상태로의 전환에 의한 미완료된 다운로드 작업을 연결성 있게 마무리하기 위해 원래의 자원을 대체하여 제공할 피어를 선택하는 과정에서 오버로드를 최소화하기 위한 다방향 프로세싱 알고리즘을 제안하였다.[5] 이는 하나의 피어와의 연결을 통한 자원 제공 방법에서 발생할 수 있는 부담률을 줄이기 위해 여러 피어로부터 자원 제공 부담을 분산하는 방법으로 히트율이 높은 특정 피어에 집중된 부담을 분산하는 방법이다. 그러나 이는 자원을 다운로드 하기 위해 해당 작업을 분담하는 피어들을 선택하는 방법에서 그 선택 기준이 해당 시점에서 가장 낮은 작업량의 피어를 선택하는 방법으로 그 선택 기준이 정밀하지 않다. 본 연구에서는 대체 피어들의 평균 성공률 평가치를 기반으로 성공 가능성이 높은 피어를 예측에 의해 결정함으로써 오프라인 발생 빈도를 낮추고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 3장에서는 개선된 다방향 프로세싱 알고리즘을 4장에서는 본 연구에 대한 평가를 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

본 논문의 기반 연구인 다방향 프로세싱 P2P 알고리즘은 기존 시스템에서 히트율이 높은 피어의 로드 가중성이 발생하는데 이를 해결하기 위해 피어 간의 비연결성 문제에 대해 해당 피어의 오버로드를 최소화하는 방법을 제안하고 있다. "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 하는 것이다. 다음은 다방향 프로세싱 P2P 시스템의 전체적인 구성 요소이다. 다방향 프로세싱 시스템은 [그림 1]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)/(자원 제공 피어의 수)). 그리고 이전 연구 TO알고리즘[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.

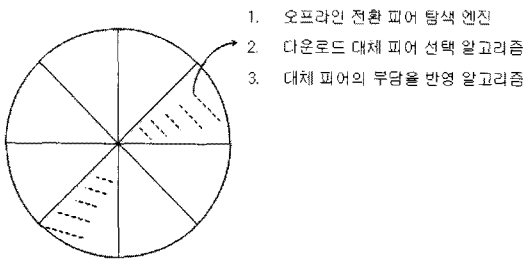


그림 1. 시스템 구성 (5)
Fig 1. Construction of System (5)

(그림 1)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, 다방향 프로세싱 시스템은 단순하게 다운로드 남은 작업이 가장 적은 피어 순으로 오프라인 피어 작업을 맡는다. 오프라인 된 피어 하나 당 하나의 대체 피어에 해당 다운로드 작업량을 부여하게 됨으로 (그림 1)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 적용 하에서 작동된다.

```
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 2. 다방향 프로세싱 알고리즘 (5)
Fig 2. Multidirectional Processing Algorithm (5)

다음 (그림 2)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. 다방향 프로세싱이 이뤄지기 위해 TO 알고리즘과 같이 기존의 피어 성공률을 기반으로 한 자원 제공 피어의 선택 과정을 따른다. 이에 따라 우선순위에 따라 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 있다."[5] 이러한 다방향 프로세싱 기반 P2P 검색 알고리즘에서 다운로드 대체 피어를 선택하는 방법의 단순함으로 초래될 수 있는 의외 상황의 시간 복잡성과 단순성을 해결할 필요가 있겠다.

III. 제안 알고리즘

본 논문에서 제안하는 알고리즘은 관련 연구에서도 살펴본 바와 같이 다방향 프로세싱 기반 P2P 시스템에서 발생할 수 있는 대체 피어 선택 방법을 개선한 다방향 프로세싱 기반 P2P 알고리즘을 제안한다. 개선된 다방향 프로세싱 기반 P2P 검색 알고리즘은 기존의 다방향 프로세싱 알고리즘에서 제안하고 있는 자원 제공 피어의 오프라인 전환 문제의 해결안을 따른다. 자원 제공 피어가 오프라인으로 전환되었을 때 하나의 피어를 이용한 연결에서 올 수 있는 피어의 자원 제공 부담률을 여러 피어로 자원 제공 부담을 나누는 방법이다. 그러나 이 방법은 자원 제공 피어의 오프라인 전환 후 역할 분담을 위한 피어 선택에 있어서 현재 시점에서 가장 낮은 작업량을 나타내는 피어를 기준으로 하고 있다. 이와 같은 자원 제공 대체 피어 선택 방법의 단순성으로 인해 오히려 자원 재검색 피어의 발생 빈도를 높임에도 불과하고 이로 인해 기존 방법에 비해 부가된 복잡도 또한 안고 가야하는 문제를 야기할 수 있다.

$$AvgSuccess = \frac{1}{n}\sum_{i=1}^{n} PastSuccess[i] \quad \cdots\cdots\cdots \text{식1)}$$

$$PastSuccess = \frac{1}{MaxRate} * \text{Process } Transmit$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \text{식2)}$$

$$SelectionCriteria = Search(Max[P]) \leq AvgSuccess \quad \cdots\cdots\cdots \text{식3)}$$

$$AgainSelection = Search(DownList[p]) \leq \text{Present}(AvgSuccess[p])$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \text{식4)}$$

본 연구에서는 기존 연구의 문제점을 해결하기 위해 자원 대체 피어들의 평균 성공률 기반 평가치를 바탕으로 성공 가능성이 높은 피어를 예측에 의해 결정함으로써 오프라인 발생 빈도를 낮추고자 한다. 식1)의 AvgSuccess는 하나의 피어가 해당 자원을 제공하면서 기존의 성공률 값을 기반으로 평균 성공률을 계산하고 있다. 이는 식2)의 기존 성공률 값을 기반으로 하고 있는데, 서로 다른 사이즈의 파일에 대한 공통의 연산을 위해 해당 파일의 사이즈를 최대 사이즈로 하여 실제 전송이 진행된 파일의 량을 측정하여 계산하며 그 전송 횟수에 따라 히스토리 구조로 보관된다. 이때 실제 재전송 요구가 발생했을 때 식3)과 같이 한 피어의 평균 성공률과 피어들의 검색 결과 나타난 평균 성공률을 비교하여 우위를 점하고 있는 피어를 선택한다.

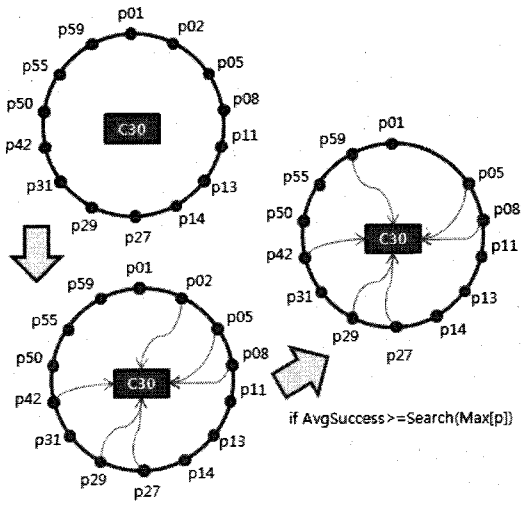


그림 3. 피어 재검색
Fig 3. Peer search again

[그림 3]는 자원을 검색하여 다운로드 받으려 하는 C30 피어가 현재 해당 P2P 시스템에 연결되어 있는 피어들 가운데 해당 자원을 보유하고 있는 피어들을 검색하고, 하나의 피어에 치중되지 않도록 작업량을 분배하는 알고리즘과 평균 성공률 기반한 피어 선택 기준을 적용하는 알고리즘을 바탕으로 자원을 제공 받는다. 다운로드 과정에서 P02 피어의 오프라인 상황으로 상태가 변화되어 기존의 피어 가운데에서 다시 평균 성공률을 기반한 피어 선택과정을 거친다. [그림 3]에서는 P59를 선택하였는데, 이때 다운로드 과정에 있지 않는 피어 가운데 가장 높은 평균 성공률의 피어를 선택하게 되는데, 기존의 다운로드 과정에 있는 피어 가운데, 해당 작업을 추가

하였을 때의 예상 성공률과 비교 하여 그 값이 기존의 다운로드 과정에 있는 피어가 더 높을 경우 식4)와 같이 현재 다운로드 과정에 피어 가운데에서 선택된 피어에게 해당 과업을 부과한다. [그림 4]에서는 [그림 3]에서와 같이 검색 결과 가장 높은 평균 성공률을 나타내는 P59를 선택하여 기존의 소실된 P02 연결을 대체하고 있는데, 기존의 자원 제공 과정에 있는 피어들 가운데 P29에 피어의 예상 성공률이 비교 결과 우수할 경우에 해당 작업이 부여되는 과정을 보여 주고 있다.

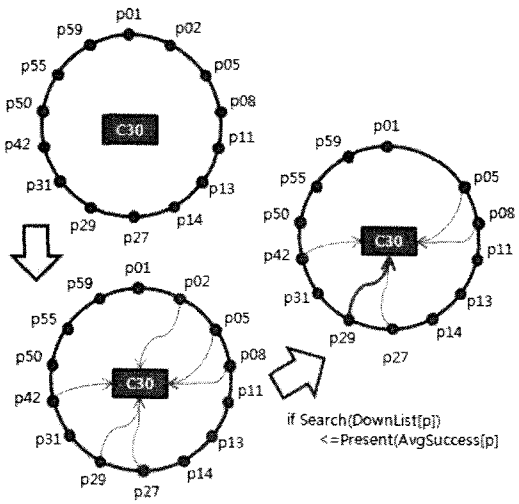


그림 4. 피어 재검색 2
Fig 4. Peer search again 2

[그림 5]은 본 논문에서 제안하는 다방향 프로세싱 기반 개선된 P2P 검색 알고리즘이다. 본 알고리즘의 초기화 내용은 기존 논문의 TO 알고리즘에 기반하며 다방향 프로세싱의 기본 틀을 바탕으로 진행된다. 오프라인 피어가 발생하면 피어들의 평균 성공률을 기반으로 한 5라인에서 12라인까지의 계산 과정을 거쳐 자원 제공 피어 선택 과정을 위한 15 라인의 함수로 이동된다. 자원 제공 피어의 선택과정은 여러 피어에게서 동시에 일어날 수 있는 일로써 라인16에서 표현한 동기화 구역으로 설정하고, 식3)과 식4)에서의 피어 선택 방법을 적용하여 오프라인 상태로 전환된 피어와 남은 작업을 맡게 될 피어를 선택하는 과정을 32번 라인까지 표현하고 있다. 이러한 대체 피어 선택의 과정을 마친 이후에 기존의 다운로드 미완료 작업의 나머지 부분에 대한 작업을 34번 라인에서 42번 라인에서와 같이 파일의 길이를 기반으로 하여 현재 진행되고 있는 전송량을 바탕으로 새로이 발생할 수 있는 대체 피어 선택 과정에 적용될 관련 정보들을 갱신하면서 병행적인 작업을 마치게 된다.

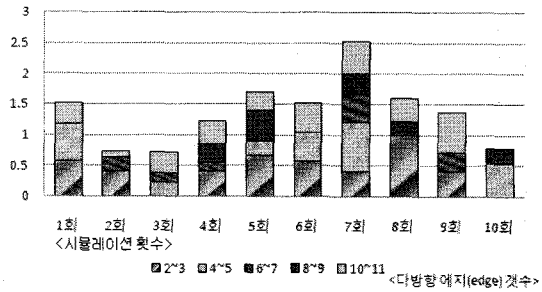
```
   PeerSelect_Processing( ) {
   Initialization : pure P2P, random graph model
1  Pre_Processor          :          counting_offline_peer(),
2  SelectPeerNum=3
3  Start. Section from counting_offline_peer
4     Vector.add(files[i]);
5     File.ListPeer();
6     int code = Avg_Success.readInt(files[i]);
7     Switch (code) {
8      (a) Select_ListPeer(0); break;
9      (b) Select_ListPeer(1); break;
10     (c) Select_ListPeer(2); break;
11    }
12    Jump SendFile()
13 End Section.
14
   Start Select_ListPeer
15    Synchronized
16    Vector.get();
17       If ( Original_Peer is Dead ) {
18        If( (Search(Max(p)) (= AvgSuccess)
19         ) (Search(DownList(p)                    (=
20    Present(AvgSuccess(p))) {
21             Changed_Peer = Peer(AvgSuccess);
22             t =  Original_Peer;
23             Original_Peer = Changed_Peer;
24             Changed_Peer = t;
25          } else {
26             Changed_Peer                          =
27    Peer(Present(AvgSuccess));
28             t =  Original_Peer;
29             Original_Peer = Changed_Peer;
30             Changed_Peer = t;
31          }
32       }
33    End Synchronized
   End Select_ListPeer Section.
34
35 Start SendFile
36    Synchronized
37     while((len = dis.read(buf)) != -1) {
38       FileInputStream.read();
39       sum += File.length;
40       remainAmount.setValue();
41     }
42    End Synchronized
43 End SendFile Section.
   }
```

그림 5. 다방향 프로세싱 기반 개선된 P2P 검색 알고리즘
Fig 5. Advanced P2P Search Algorithm based on Multidicrectional Processing

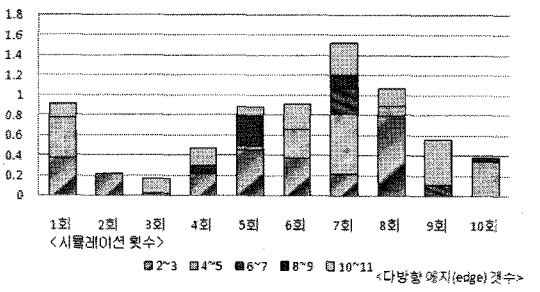
다방향 프로세싱 기반 개선된 P2P 검색 알고리즘은 문제의 크기를 나타내는 파라메터 n을 검색의 과정에서 거치게 되는 노드의 수로 정했을 때, n에 관해 가장 빨리 증가하는 항만을 남기고 다른 항은 무시해도 되는 점근적인 복잡도 평가에 의해 알고리즘을 평가해 보면으로 시간 복잡도 측면에서 긍정적인 결과를 보이고 있다.

IV. 시뮬레이션 결과

이 장에서는 본 논문이 제안한 다방향 프로세싱 기반 개선된 P2P 검색 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.



(a) 미완료율(A)

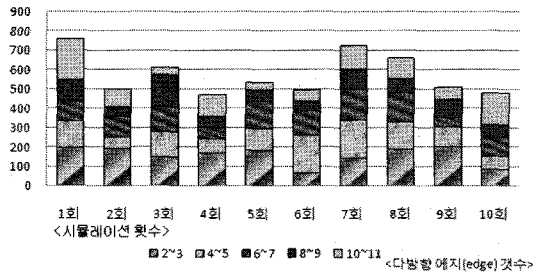


(b) 미완료율(B)

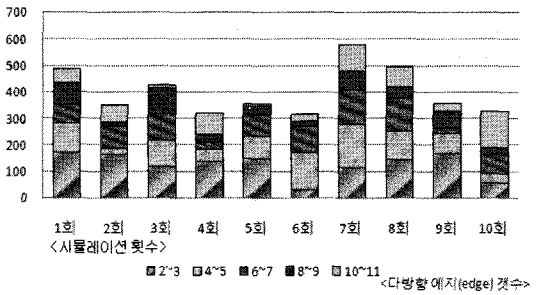
그림 6. 선택피어 미완료율
Fig. 6. Rate when download work is'nt completed at selected peer

위 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 설정은 TO 알고리즘에서의 설정 사항을 따른다. 성능 비교의 기준이 되는 시뮬레이션 횟수는 각 항목에 대해 기존의 경우와 마찬가지로 10회로 하였다. 또한 시뮬레이터는 Peersim을 실험 환경에서 이용할 자원의 사이즈는 500M를 다방향 에지의 최대수는 11로 한정한다. 다방향 에지 수 및 시뮬레이션 횟수는 이전 연구에서 나타낸 바와 같다.

본 알고리즘에 대해 자원 제공 피어를 알고리즘에 의해 선택하였는데, 해당 피어의 자원 제공 미완료율을 비교한 결과는 [그림 6]과 같다. 그림의 A는 이전의 다방향 프로세싱 기반 P2P 알고리즘의 결과를 의미하고, B는 현재의 개선된 다방향 프로세싱 기반 P2P 알고리즘의 결과이다. [그림 6]의 선택 피어 미완료율은 다방향 에지(edge)의 수에 따른 데이터로 A, B 알고리즘 모두 공통적으로 다방향 에지의 수가 2, 3일 경우 선택피어 미완료율이 제일 높았고, 6, 7일 때 가장 낮았다. 또한 A, B 알고리즘 모두 에지수가 6, 7일 때와 8-9일때 미완료율이 0인 경우가 가장 많았다. 0.1대의 미완료율을 나타내는 항목은 A 알고리즘의 경우 에지수가 6, 7일 때와 10, 11일 때로 나타났으며, B 알고리즘의 경우 4 ,5일 때와 8 ,9일 때와 10 ,11일 때로 나타났다. 전반적으로 A, B 알고리즘 모두 에지수 6, 7일 때 가장 낮은 선택피어 미완료율을 나타내고 있다. 그러나 A 알고리즘에 비해 B 알고리즘에서 전반적으로 미완료율이 0.48% 떨어짐으로써 미완료율 감소 상태를 나타내고 있다.



(a) 다운로드 완료 시간(A, 단위 : 초(seconds))



(b) 다운로드 완료 시간(B, 단위 : 초(seconds))

그림 7. 다운로드 시간
Fig. 7. Download time

[그림 7]은 다방향 에지 개수에 따른 총 다운로드 완료 시간을 초(seconds)단위로 나타냈다. 다운로드 평균 시간이 가장 짧은 경우를 측정한 결과 A, B 알고리즘 모두에서 에지수가 8, 9일 때이고, 가장 긴 경우는 에지수가 2, 3일 경우로 나타났다. 가장 짧은 다운로드 시간을 측정한 결과 A 알고리즘의 경우 에지수 10, 11일때로 38초를 나타냈고, B 알고리즘의 경우 8 ,9일 때 12초를 나타냈다. 40초대 이하의 값을 측정한 결과는 A 알고리즘의 경우 4건이 발생하였고, B 알고리즘의 경우 11건이 발생하였다. 가장 긴 다운로드 시간대로는 A 알고리즘의 경우 에지수 10, 11일 때 213초로 나타났고, B 알고리즘의 경우 에지수 2, 3일 때 175임을 보여주고 있다. 전반적으로 A 알고리즘보다 다운로드 시간이 B 알고리즘이 0.29배 감소되어 B 알고리즘의 효용성을 나타내고 있다.

V. 결론 및 향후 연구

네트워크로 연결된 많은 유휴 자원들을 공유하는 목적으로 유용하게 쓰이는 P2P 시스템은 개발 시점의 목표에서 드러나듯이 자원을 제공하는 피어와 자원을 제공 받는 피어 사이의 연결성이 항상 보장 되지 않는다.[1][2][7] 연결성 미보장 문제는 시스템의 신뢰성을 떨어트리는 요인이 됨으로 그 해결 방법에 대한 연구가 활발히 진행되고 있다. 본 논문에서는 P2P 시스템을 이용한 자원의 다운로드 작업을 수행하는 가운데 발생할 수 있는 미완료된 다운로드 작업을 연결성 있게 마무리하였다. 이를 위해 원래의 자원을 대체하여 제공할 피어를 선택하는 과정에서 오버로드를 최소화하기 위한 다방향 프로세싱 방법에서 발생할 수 있는 다운로드 대체 피어들을 선택하는 방법을 효율화 하여 다운로드 성공 가능성을 높이고자 하였다.

향후 연구로는 자원 제공 피어의 다방향성 틀을 가져가면서 최적의 성능을 제공하기 위한 개선된 오리엔티어링 기법을 적용하거나 히스토리 기반의 작업량을 기반으로 접근하여 좀 더 유용한 시스템을 제안하고자 한다.

참고문헌

[1] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," ICDCS'02, pp.103-113, 2002.

[2] Wei Xiong, Dong-Qing Xie, Zai-Hong Zhou, Jie Liu, "Decreasing system load by caching in Structured P2P Systems," HPCC Proceeding, IEEE, 2008..

[3] Silvia B., Sabina S., and so on, "Adaptive Load Balancing for DHT Lookups," CCN Proceeding, IEEE, 2006.

[4] Huifang Cheng, Zhimin Gu, "IntranetWeb: a Peer-to-Peer based Scalable Web Caching System," NCA proceeding, IEEE, 2006.

[5] 김분희, "다방향 프로세싱 기반 P2P 검색 알고리즘," 한국컴퓨터정보학회 논문지, 제 13권, 제 7호, 2008년. 12월.

[6] 김분희, "효과적인 역 추적 P2P 자원 검색 알고리즘," 한국컴퓨터정보학회 논문지, 제 12권, 제 6호, 49-58쪽, 2007년. 12월.

[7] 엄남경, 우성희, 이상호, "SVM을 이용한 플로우 기반 P2P 트래픽 식별," 한국컴퓨터정보학회논문지, 제 13권, 제 3호, 123-130쪽, 2008년. 5월.

저 자 소 개

김 분 희

2005년 2월 : 중앙대학교 컴퓨터공학과 공학박사

2005년~현재 : 동명대학교 미디어공학과 전임강사

관심분야 : 분산 시스템, P2P 검색 기법, HCI(Human Computer Interaction)