Journal of KIIT. Vol. 16, No. 1, pp. 133-143, Jan. 31, 2018. pISSN 1598-8619, eISSN 2093-7571 133
http://dx.doi.org/10.14801/jkiit.2018.16.1.133

비트 토렌트에서 건강한 피어 선택 기반의 파일 다운로드 속도 향상

박용현*, 구자환**, 김응모***

Improving File Download Speed Based on the Selection of Healthy Peers in BitTorrent

Yong-Hyeon Park*, Jahwan Koo**, and Ung-Mo Kim***

본 연구는 문화체육관광부 및 한국저작권위원회의 2017년도 저작권기술개발사업의 연구결과로 수행되었음

요 약

P2P 환경에서 파일 조각을 다운로드 받을 피어를 효과적으로 선택하면 다운로드 속도를 개선할 수 있다. 본 논문에서는 성능 개선에 영향을 줄 수 있는 성능 지표 및 메카니즘을 고려하여 파일 다운로드 속도를 향상시킬 수 있는 비트 토렌트 시스템을 설계하고 구현하였다. 최고 업로드 속도의 값이 클수록 그리고 왕복 소요시간 값이 작을수록 피어가 건강하다고 정의하였고 파일 다운로드시 건강한 피어가 선택될 수 있도록 하였다. 이러한 메카니즘이 파일 다운로드 성능 향상에 기여할 수 있음을 다양한 네트워크 환경에서 성능 평가를 통해 검증하였다. 인터넷 환경에서 본 논문의 제안사항을 적용한 토렌트는 기존 토렌트에 비해 약 10%의 성능 향상이 있었다. 하지만 내부망 환경에서는 괄목할만한 성능 향상을 보이지 못했다.

Abstract

In a P2P environment, effective selection of peers where download file piece can improve download speed. In this paper, we design and implement a bittorrent system that can improve file download speed considering the performance criterion and mechanism such as maximum upload speed and round trip time between peers that can affect performance improvement. We define larger maximum upload speed and smaller the round trip time as healthier the peers. We coded that the healthy peer is selected first when download file. We verified that this mechanism can contribute to download speed improvement through performance evaluation in various network environments. Improved torrent considering suggestion of this thesis is about 10% better than original in internet environment. However, in intranet environment, there is almost no improvement.

Keywords

P2P, peer selection, performance criteria, round trip time, maximum upload speed

* 성균관대학교 소프트웨어대학
- ORCID: http://orcid.org/0000-0002-1093-7193
** 성균관대학교 사회과학대학 연구교수(교신저자)
- ORCID: http://orcid.org/0000-0002-2844-3183
*** 성균관대학교 소프트웨어대학 교수
- ORCID: http://orcid.org/0000-0001-5464-6358

· Received: Dec. 03, 2017, Revised: Jan. 12, 2018, Accepted: Jan. 15, 2018
· Corresponding Author: Jahwan Koo
Room# 27309, Engineering bldg. 2, Sungkyunkwan University, 2066 Seobu-Ro, Jangan-gu, Suwon-si, Gyeonggi-do 16419, South Korea
Tel.: +82-31-290-7218, Email: jhkoo@skku.edu

Ⅰ. 서 론

P2P 방식을 사용하는 파일 다운로드 서비스인 토렌트는 인터넷 트래픽의 거의 절반을 사용하는 주요한 파일 공유 및 업로드 다운로드 방식이다[1][2]. 이러한 토렌트는 크게 다운로더, 피어, 트래커 등으로 구성되어 있으며, 트래커는 여러 피어들의 메타정보들을 가지고 있고, 다운로더는 그 메타정보들을 받아 다른 피어들로부터 파일의 조각을 다운로드 받는다. 다운로더가 파일 다운로드를 요청하면, 이 요청 메세지가 트래커에게 전송된다. 이 요청 메세지는 파일의 종류를 구분하는 고유값인 해쉬값을 포함한다. 요청 메세지를 받은 트래커는 자기가 관리하고 있는 스웜에서 동일 해쉬값을 가지는 피어를 뽑아 피어리스트를 만든 뒤 다운로더에게 전달한다. 다운로더는 피어리스트의 피어들을 선택 및 연결하여 그들이 가지고 있는 파일 조각을 다운로드 받아 완전체 파일을 만든다[3][4].

P2P 환경에서는 피어끼리의 파일 조각 공유를 통해 파일 다운로드가 이루어지기 때문에, 물리적으로 가까이 있고 파일 공유 속도가 빠른 피어를 선택하는 것이 매우 중요하다[5]. 하지만 현재 파일 조각을 공유할 피어 정보를 선택하는 방법은 무작위이다[6]. 즉, 트래커는 물리적으로 멀리 있으며 건강하지 않은 피어의 정보를 전달할 수도 있으며 그럴 경우 다운로더는 건강하지 못한 피어와 파일 조각 교환을 하게 될 가능성이 높아진다는 사실을 의미한다. 이는 다운로드 속도 감소로 이어진다. 이런 문제를 해결하기 위해서는 다운로더가 건강한 피어를 구별하여 그 피어에게 다운로드를 받을 수 있어야하며, 이를 구현하기 위한 다양한 연구가 진행 중이다.

알토 프로토콜(ALTO Protocol)에서는 ALTO 서버는 다운로더와 연결될 수 있는 상태인 피어를 알려주는 네트워크 지도와 네트워크 지도에 있는 피어와 연결을 시도했을 때 드는 비용을 상대적으로 나타낸 비용 지도를 제공한다. 이러한 정보를 넘겨받은 다운로더는 최적경로를 직접 계산하여 건강한 피어들과 연결될 수 있는 가능성을 높인다[6][7]. 편향 이웃 선택(Biased Neighbours Selection)에서는 동일한 ISP에 있는 피어(내부 피어)들과 통신하기 위해 트래커는 다운로더의 ISP 정보를 확인하고 이와 동일한 ISP 정보를 가진 피어를 우선적으로 선정한다[6]. 편향 초킹해제(Biased Unchocking)에서는 거리 정보를 활용해 물리적으로 가까이 있고 비용이 적게 드는 피어에게 초킹해제 될 기회를 우선적으로 부여한다[6]. P4P 프레임워크(P4P Framework)에서는 다운로드 속도 향상의 열쇠를 쥐고 있는 기존 트래커와는 다른 종류의 트래커인 iTracker를 별도로 운용한다. iTracker는 피어들의 위치 정보를 가지고 있으며 트래커는 이 정보에 기반하여 다운로더에게 가장 가깝고 비용이 적게 드는 피어들을 우선적으로 선별하여 피어리스트를 전달한다[8][9]. 이러한 방법들은 토렌트의 성능을 개선시킬 수 있는 방안이기는 하나 대부분 어떤 피어가 건강한 피어인지를 결정하는 기준을 제시하지 않았다. 따라서 성능을 결정하는 지표를 결정하고 그 지표를 활용하여 건강한 피어를 선택하는 방식을 고안할 필요가 있다.

토렌트의 원리를 분석하면 P2P 환경에서 파일 다운로드 속도에 영향을 주는 요인은 트래커의 피어리스트 선택과 이들 중 파일 다운로드를 요청할 피어들의 선택이다. 본 논문에서는 파일 다운로드를 요청할 피어들을 선택하는 방법을 수정하여 성능 개선을 유도하였다. 트래커는 다운로더에게 무작위로 피어리스트를 건네지만 다운로더는 또 다른 객체인 S-트래커에게 피어리스트를 넘기며 피어들의 첫 번째 성능 지표 값을 요청한다. S-트래커는 피어리스트 내의 모든 피어와 통신하여 첫 번째 성능 지표를 구한 뒤 다운로더에게 전달한다. 다운로더는 첫 번째 성능 지표 값이 부진하지 않은 피어들만을 대상으로 두 번째 성능 지표 값을 구하고 이 값들을 적절히 활용하여 수치화한 뒤 건강한 순서대로 피어리스트를 정렬한다. 다운로더는 정렬 후 순위가 높은 피어에게 우선적으로 다운로드를 요청한다.

합리적인 기준을 가지고 특정 피어들에게 우선적으로 다운로드 요청을 하므로 다운로드 속도가 빨라질 것이다. 2장에서는 기존 비트 토렌트의 원리를 분석하고 이를 바탕으로 3장에서는 성능 개선을 위한 새로운 개념과 방안을 제안을 할 것이다.

II. 비트 토렌트 작동 원리 및 성능 개선에 관한 기존 연구

2.1 비트 토렌트 작동 원리

그림 1은 기존 토렌트 작동 원리를 담고 있는 시퀀스 다이어그램이다[4]. 이 때 다운로드 속도에 가장 큰 영향을 끼치는 부분은 5번과 6번인데, 5번에서 트래커가 피어리스트를 구성하여 다운로더에게 넘겨줄 때 다운로드 성능이 일부 결정된다.

그림 2는 그림 1의 1번, 5번이 진행되는 과정을 나타낸 시퀀스 다이어그램이다. 현재 비트 토렌트에서는 그림 2에서 다운로더는 트래커에게 HTTP get 명령을 이용해 피어리스트를 요청한다. 그러면 트래커는 스웜 내에 있는 피어에서 무작위로 n_0개를 골라 취합하여 그 목록을 다운로더에게 전달한다. 만약 스웜 내에 n_0개 이하의 피어가 있다면 모든 피어 정보를 전달한다. 이와 같이 기존 토렌트는 무작위로 피어리스트를 만들며, 이는 다운로드 속도 저하의 요인이 될 수 있다. 이 부분을 개선하여 다운로드 속도를 개선하려는 수많은 연구가 있었다.

2.2 성능 개선에 관한 기존 연구

서론에서 제시한 기존 연구 중 유일하게 성능 지표를 제시한, 성능 지표로 물리적 위치를 사용한 P4P 프레임워크에서는 두 개의 트래커를 운용하는데 하나는 기존의 트래커와 같은 역할을 하는 pTracker이고, 나머지 하나는 스웜 내 피어들의 위치정보를 가지고 있는 iTracker이다. 위치정보는 ASID(Autonomous System Identifier), PID(Provider-defined network location identifier), LOC(Location) 세 가지 정보를 포함한다. ASID는 ISP(Internet Service Provider) 식별자이고 PID는 ASID 내부의 피어를 위치에 기반하여 묶은 그룹의 식별자이고, LOC는 PID 내부 피어를 위치에 기반하여 세부적으로 구분하는 식별자이다. 다운로더는 가장 먼저 iTracker에게 본인의 ip주소를 전달하고 iTracker로부터 ASID, PID, LOC 정보를 얻는다. 다운로더는 세 가지 위치정보와 ip주소를 함께 pTracker에게 전달하고, pTracker는 해당 다운로더의 위치정보와 동일한 위치정보를 가지는 피어를 우선적으로 선택하여 피어리스트를 구성한다. 결과적으로 다운로더는 지리적으로 근접한 피어들에게 다운로드를 요청할 수 있게 되어 속도 개선을 이루게 된다. 이는 ISP 사업자들과의 협력이 필요하다[8]. 하지만 토렌트는 네트워크에 부하를 주기 때문에 ISP 사업자들의 협력을 기대하기는 어려우며, 따라서 ISP가 개입하지 않는 개선안이 요구된다.

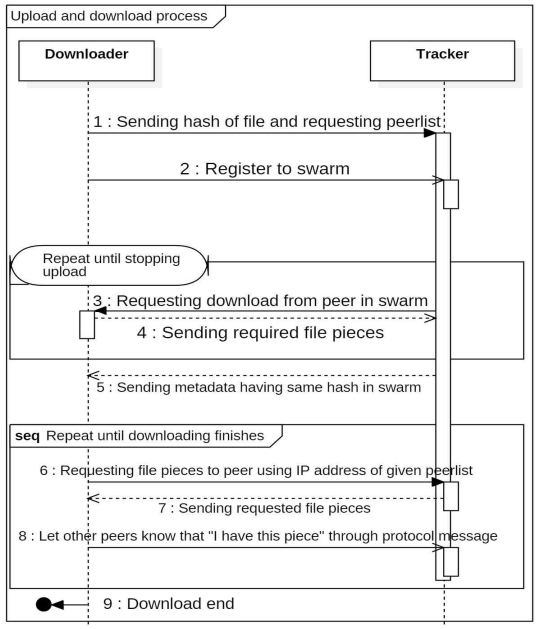

그림 1. 기존 비트 토렌트 작동 원리
Fig. 1. How current bit torrent works

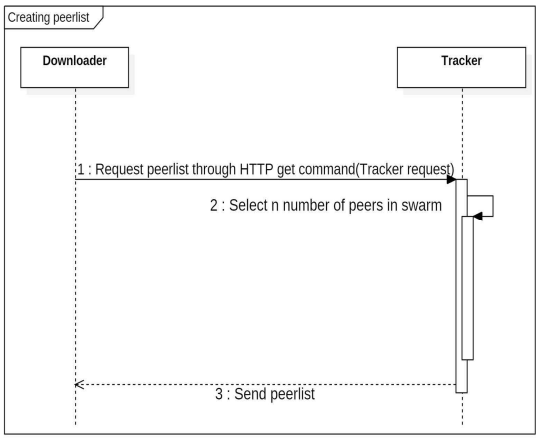

그림 2. 기존 토렌트에서 피어리스트 생성 과정
Fig. 2. Process of making peer list in current torrent

III. 건강한 피어 선택 기반의 성능 향상 연구

3.1 성능지표의 정의

3.1.1 최고 업로드 속도

최고 업로드 속도(MUS, Maximum Upload Speed)는 파일 조각 공유에 참여한 순간부터 현재까지의 업로드 속도 중 가장 빠른 값을 말한다. 토렌트는 256KB인 파일 조각을 주고받으므로 초당 몇 개의 파일조각을 주고받는지 조사하면 순간 업로드 속도를 얻을 수 있다[10]. MUS가 경신될 때마다 해당 값은 바뀐다. MUS가 빠른 피어들로부터 파일을 다운로드 받는다면 다운로드를 더 빠르게 완료할 수 있다고 가정할 수 있다.

3.1.2 왕복 소요 시간

왕복 소요 시간(RTT, Round Trip Time)은 작은 파일 조각을 다른 피어에게 전송한 뒤, 다시 돌아오기까지 걸리는 시간을 의미한다. 그림 3에서 다운로더는 P1, P2, P3, P4에 대한 RTT 값을 측정하고, 표 3처럼 저장하고 있다. 그런데 RTT를 측정하는 것은 파일 조각이 왕복하는 동안 대기해야 하는 것을 의미하므로, S-트래커가 전달한 모든 피어의 RTT를 측정하는 것은 굉장히 시간 소모가 클 수 있다. 그에 대한 해결책은 최대 RTT값을 상수 k0로 제한하는 것이다. 그림 3에서 다운로더는 P1, P2, P3, P4에게 동시에 파일 조각을 전송하여 RTT를 측정하고자 하였으나 P2, P4의 경우 2ms가 넘어서도 응답이 오지 않으므로 즉시 측정을 중단하고 피어리스트 구성에서 배제된다. 적절한 한계 값은 실험을 통해 정할 수 있을 것이다.

코드 수준에서 스웜 내부의 모든 피어를 대상으로 RTT를 구하는 것은 O(nk)의 시간 복잡도를 가지는데, 이는 파일을 다운로드하는 사람이 많아지면 급격하게 커질 수 있는 값이므로 비현실적이다. 여기서 k는 각 피어들에 대한 평균 RTT값이며, n값은 스웜 내부의 피어들의 개수이다.

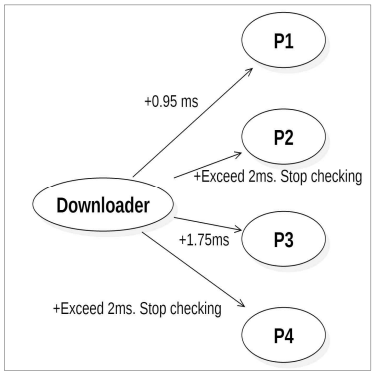

그림 3. 다운로더 P1의 RTT 측정
Fig. 3. Measuring RTT of downloader P1

예를 들어, 스웜 내부 피어들의 평균 RTT값이 1ms이고 스웜 내부의 피어가 10000개라고 가정하면, 사용자는 피어리스트를 만드는 데에만 약 10초를 할애해야한다. 이는 결코 다운로드 속도를 개선한다고 볼 수 없는 긴 시간이다. 하지만 다운로더가 전달받은 피어리스트에는 n_0개의 피어 정보만 있으며 최대 RTT값을 k_0으로 제한했으므로 최악의 경우에도 시간 복잡도는 특정 상수 값에 수렴하게 된다.

3.2 S-트래커의 운용

본 연구에서는 다운로드 속도 개선과는 별개로 가용성 보장을 위해 S-트래커라는 것을 별도로 운용한다. S-트래커는 최초 유포자가 파일을 업로드하면 피어와 동일한 방식으로 파일을 다운로드 받아 보관하며 피어리스트에 항상 포함되어 가용성을 보장한다. S-트래커는 건강한 피어를 선택하는 주체가 될 수 있으므로 기존 트래커의 기능을 변형시키지 않고 S-트래커의 구축으로 본 논문에서 제안하는 바를 구현하고자 한다. 그림 4는 S-트래커의 개입을 통해 건강한 피어를 선정하는 과정이다. 그림 5는 본 연구를 통해 개선된 토렌트로 파일을 다운로드받는 과정이다.

개선된 토렌트는 다운로더의 최종적인 피어리스트 획득에 S-트래커가 개입하는 것이 특징이다. 먼저 다운로더는 트래커에게 N개의 피어 정보를 요청한다.

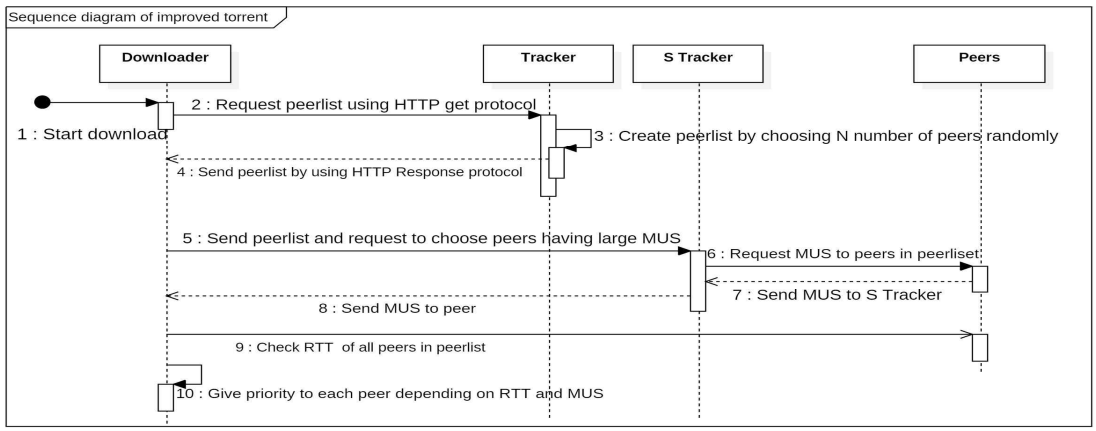

그림 4. 개선된 토렌트의 피어리스트 생성 과정
Fig. 4. Process of making peer list in developed torrent

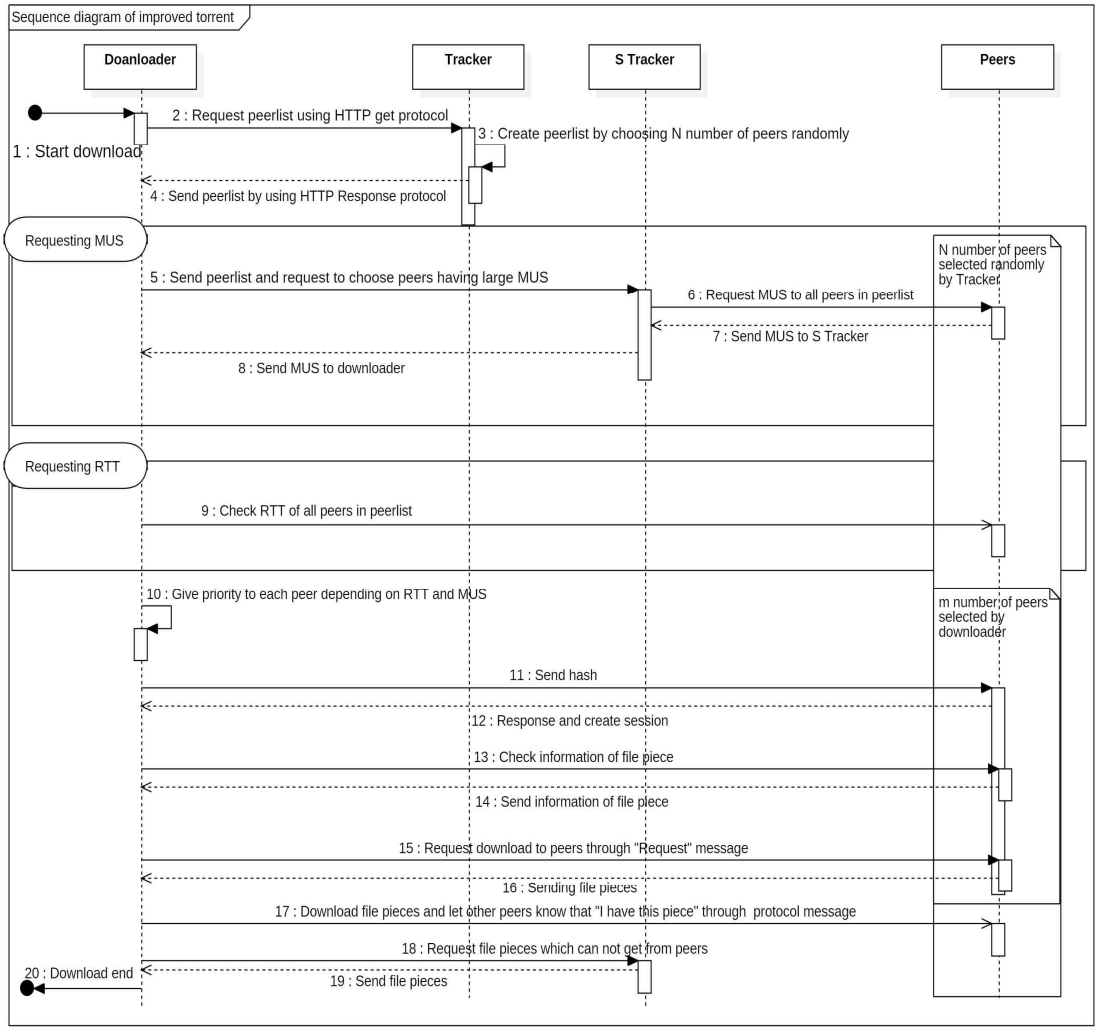

그림 5. 개선된 토렌트의 파일 다운로드 과정
Fig. 5. Process of downloading file in developed torrent

트래커는 이 요청에 응답하고, 다운로더는 건네 받은 피어리스트를 이번엔 S-트래커에게 전달하며 모든 피어들의 최고 업로드 속도 값을 요청한다. S-트래커는 피어리스트 안에 있는 메타정보의 ip주소를 이용해 피어들에게 일일이 최고 업로드 속도를 요청하고, 이 값들을 다운로더에게 전송함으로서 응답한다. 다운로더는 최고 업로드 속도를 기준으로 피어리스트를 정렬하고, 업로드 속도가 빠른 n개 (N>=n)의 피어들만을 대상으로 RTT를 측정한다. 이 두 지표를 적절히 사용하여 빠르다고 생각되는 피어들에게 우선적으로 다운로드를 요청한다. 이는 속도를 고려한 다운로드 방식이며, 가용성을 보장하기 위한 다운로드 방식이 필요하다.

토렌트에는 피어들에게 다운로드를 요청하기 전에 각 피어가 가지고 있는 파일 조각을 확인하고 어떤 피어로부터 어떤 파일조각을 다운로드받을지 결정하는 과정이 있는데, 성능지표를 고려하여 선택된 피어들이 다운로더에게 필요한 파일조각을 가지고 있지 않을 수도 있다. 이런 문제를 해결하기 위해 시드파일은 S-트래커의 주소를 항상 포함한다. 선택된 피어들로는 가용성이 보장되지 않아 다운로드를 완료할 수 없을 때 다운로더는 S-트래커로부터 필요한 파일 조각을 다운로드받아 파일을 완성시킨다. 또한, 피어들에게 파일조각을 받는 것 보다 S-트래커로부터 받는 것이 더 빠르다고 판단되면 S-트래커에게 우선적으로 다운로드를 요청한다.

3.3 피어 선택 방법

본 논문에서는 다운로드 성능을 향상시키기 위해 건강한 피어를 선택하는 구체적인 기준인 성능 지표를 정의하고 이 성능 지표에 따라 특정 피어에게 우선적으로 다운로드를 요청하는 방법을 제안하고자 한다. 본 논문에서는 성능 지표로 최고 업로드 속도와 RTT를 활용하였다. S-트래커의 경우 최고 업로드 속도를 다운로더에게 전달하며, 다운로더는 RTT 정보를 결합하여 우선적으로 다운로드를 요청할 피어들을 선별한다.

표 1은 트래커로부터 전달받은 N개 피어들의 최고 업로드 속도를 S-트래커가 다운로더에게 전달하는 것을 그림으로 나타낸 것이다. 각 피어들은 본인들의 최고 업로드 속도를 기록하고 있는데, S-트래커는 메타 정보에 포함된 ip주소를 이용하여 각각의 피어에게 최고 업로드 속도를 요청한다. 이 값을 다운로더에게 전달하면 다운로더가 정렬을 통해 업로드 속도가 빠른 n개의 피어를 추려낸다.

표 2는 RTT값을 조사하는 과정이다. 다운로더는 ICMP 방식으로 표 1에서 선택된 n개 피어들의 RTT값을 모두 조사한 후 이 정보를 최고 업로드 속도와 적절히 조합하여 피어들에게 우선순위를 부여한 뒤, 우선순위가 높은 피어에게 파일 다운로드를 우선적으로 요청한다. 다음 장에서는 본 논문에서 제안한 사항이 실제로 성능 개선에 효과가 있는지 확인하기 위해 코드 수준에서 개선된 토렌트 버전을 사용해 진행된 실험의 진행 방법과 결과를 다루고자 한다.

표 1. S-트래커로부터 전달받은 최고 업로드 속도 값을 이용하여 정렬을 수행한 다운로더 내부의 정보
Table 1. Information in the downloader after sorting by maximum upload speed taken from S-Tracker

	Maximum Upload speed	Selection
P5	20	O
P2	13	O
P1	9	O
P4	8	O
...		
P3	7	O
Pn	6	X
.....		
PN	5	X

표 2. 표 1에서 S-트래커가 전송한 피어의 RTT를 구하는 과정
Table 2. Downloader takes RTT information from

	P5	P2	P1	P4	P3	Pn	...	PN	S-Tracker
MUS	20	13	9	8	7	6	...	5	-
RTT	1	3	2	5	0	-	...	-	-

표 3. <그림 3>의 결과로 만들어진 RTT 테이블
Table 3. RTT table from Figure 3

RTT from Downloader to	P1	P2	P3	P4
RTT	0.95	-	1.75	-

Ⅳ. 실험 방법 및 실험 결과

4.1 실험 방법

이 논문에서 구현하는 것은 크게 두 가지이다. 첫 번째는 S-트래커가 피어리스트 내 피어들의 MUS를 다운로더에게 전달하는 것이고, 두 번째는 다운로더가 RTT를 측정하여 우선적으로 다운로드를 요청할 피어들을 선정하도록 돕는 것이다. 따라서 본 실험에서는 첫 번째로 MUS만을 고려하였을 때 다운로드에 소요되는 시간을 측정하고, 두 번째로 RTT만을 고려하였을 때 다운로드에 소요되는 시간을 측정하였다. 마지막으로 본 논문에서 제안한 성능 개선 방식을 사용하지 않은 토렌트를 사용하였을 때 다운로드에 소요되는 시간을 측정한 뒤 이 값을 첫 번째, 두 번째에서 소요된 시간과 비교하여 성능 개선의 정도를 비교한다.

내부 망에서는 대부분의 피어가 같은 조건을 가지기 때문에, 다운로드 속도 개선은 내부 망 실험보다는 실제 인터넷 환경에서 두드러진다. 따라서 내부 망에서 토렌트 버전별로 3회, 외부 인터넷 환경에서 토렌트 버전별로 3회 실험하여 다운로드 속도를 확인해 보았다. 외부 인터넷 환경을 조성하기 위해 트래커에게 외부 IP를 부여하였고 vpn gate를 사용해 리처들의 IP를 우회하였다.

4.1.1 최고 업로드 속도에 기반한 다운로드

MUS를 고려하여 파일을 다운로드 받는 메커니즘은 그림 6과 같다. 이를 구현 및 실험하기 위해 S-트래커를 구현할 컴퓨터 1대, 트래커를 구현할 컴퓨터 1대, 다운로더를 구현할 컴퓨터 1대, 피어들을 구성할 컴퓨터 3대, 가상 컴퓨터 5대를 사용하였다. 실험자는 다운로더를 제외한 모든 피어들을 리처로 만든다. 여기서 피어들을 리처로 만드는 이유는 시더는 리처보다 성실하게 업로드를 하기 때문에 리처보다 높은 연결 우선순위를 가지고, 따라서 MUS와 RTT값을 고려한 다운로드 메커니즘과 무관하게 항상 우선 다운로드 대상이 되기 때문이다. 이 상태에서 다운로더의 다운로드를 시작한다. 이때 S-트래커에 출력되는 정렬된 피어 목록과 MUS를 다운로더에게서도 확인할 수 있으며 MUS가 높은 피어들로부터 다운로드를 받고 있다면 실험이 정상적으로 이루어지고 있음을 확인할 수 있다. 같은 환경에서 성능 개선이 이루어지지 않은 다운로더 버전을 이용하여 다운로드를 수행하고 걸리는 시간을 비교하여 해당 방식이 유효한 성능 개선을 만드는지 확인할 수 있다.

4.1.2 RTT에 기반한 다운로드

RTT값을 고려하여 파일을 다운로드받는 메커니즘은 그림 7과 같다. 이를 구현 및 실험하기 위해 S-트래커를 구현할 컴퓨터 1대, 트래커를 구현할 컴퓨터 1대, 다운로더를 구현할 컴퓨터 1대, 피어들을 구성할 컴퓨터 3대, 가상 컴퓨터 5대를 사용하였다.

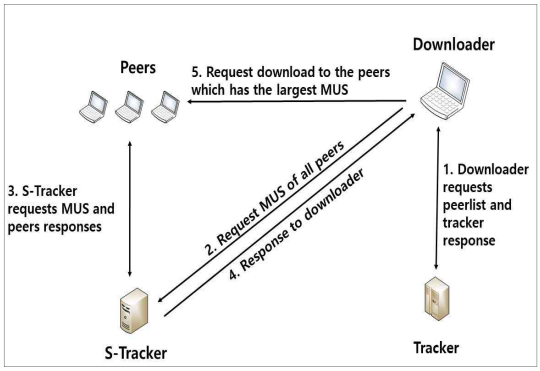

그림 6. 최고 업로드 속도에 기반한 다운로드 과정
Fig. 6. Process of downloading file in MUS torrent version

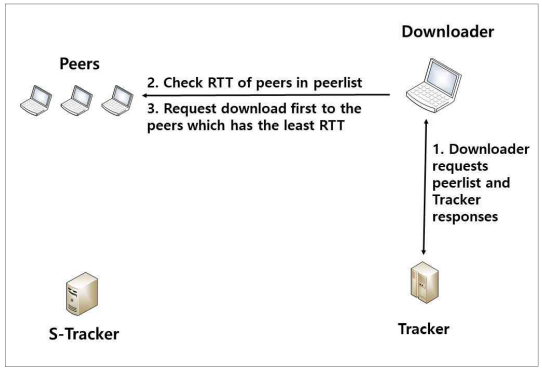

그림 7. RTT에 기반한 다운로드
Fig. 7. Process of downloading file in RTT Torrent version

4.1.1에서와 같은 이유로 실험자는 모든 피어를 리처로 만들어야 한다. 이 상태에서 다운로더의 다운로드를 시작한다. 이때 다운로더가 연결된 피어들에 대해 RTT를 올바르게 측정하고 RTT가 작은 피어로부터 다운로드를 받고 있다면 다운로더가 RTT를 고려하여 다운로드를 수행하고 있음을 알 수 있다.

같은 환경에서 성능 개선이 이루어지지 않은 다운로더 버전을 이용하여 다운로드를 수행하고 걸리는 시간을 비교하여 해당 방식이 유효한 성능 개선을 만드는지 확인할 수 있다.

Info Files Trackers Peers Speed Settings

Address	Download	Upload	MaxUploadSpeed(kbs)
192.168.0.13:6969	53kB/s	48kB/s	7059kB/s
192.168.0.6:6969	57kB/s	0kB/s	6906kB/s
192.168.0.15:6969	26kB/s	35kB/s	6684kB/s
192.168.0.11:6969	1385kB/s	1kB/s	5511kB/s
192.168.0.12:6969	1046kB/s	84kB/s	3975kB/s
192.168.0.4:6969	1650kB/s	306kB/s	3409kB/s
192.168.0.7:6969	0kB/s	0kB/s	2048kB/s
192.168.0.14:6969	0kB/s	39kB/s	-1kB/s

그림 8. MUS 토렌트 버전에서 다운로드 직후 상황
Fig. 8. Snapshot of downloader in MUS torrent version right after starting downloading

Info Files Trackers Peers Speed Settings

Address	Download	Upload	MaxUploadSpeed(kbs)
192.168.0.13:6969	0kB/s	40kB/s	7059kB/s
192.168.0.6:6969	0kB/s	0kB/s	6906kB/s
192.168.0.15:6969	0kB/s	70kB/s	6684kB/s
192.168.0.11:6969	1723kB/s	1kB/s	5511kB/s
192.168.0.12:6969	1612kB/s	112kB/s	3975kB/s
192.168.0.4:6969	1928kB/s	449kB/s	3409kB/s
192.168.0.7:6969	0kB/s	0kB/s	2048kB/s
192.168.0.14:6969	0kB/s	42kB/s	-1kB/s

그림 9. MUS 토렌트 버전에서 다운로드 상황
Fig. 9. Snapshot of downloader in MUS torrent version when almost half of downloading is done

Info Files Trackers Peers Speed Settings

Address	Download	Upload	MaxUploadSpeed(kbs)
192.168.0.13:6969	1336kB/s	440kB/s	7059kB/s
192.168.0.6:6969	0kB/s	0kB/s	6906kB/s
192.168.0.15:6969	1316kB/s	387kB/s	6684kB/s
192.168.0.11:6969	1209kB/s	17kB/s	5511kB/s
192.168.0.12:6969	1229kB/s	41kB/s	3975kB/s
192.168.0.4:6969	865kB/s	2615kB/s	3409kB/s
192.168.0.7:6969	0kB/s	0kB/s	2048kB/s
192.168.0.14:6969	1219kB/s	129kB/s	-1kB/s

그림 10. MUS 토렌트 버전에서 피어들이 시딩 상태일 때의 다운로드 상황
Fig. 10. Snapshot of downloader in MUS torrent version when other peers are seeding

4.2 실험 결과

4.2.1 최고 업로드 속도에 기반한 다운로드

최고 업로드 속도를 고려하여 파일을 다운로드 받는 과정은 그림 8, 그림 9, 그림 10에서 확인할 수 있다.

그림 8은 다운로드 직후 피어들과의 연결 상태이며, 상위 피어들에게 주로 다운로드를 시도한다.

하지만, 다운로드가 어느 정도 진행되면 그림 9와 같은 상황이 된다. 최고 업로드 속도가 높은 피어들에게 다운로드를 우선적으로 요청 하였으나 각 피어들의 환경 때문에 다운로드를 받지 못한 경우 차선 피어에게 다운로드를 요청하였다고 생각될 수 있다. 그림 10은 대다수의 피어가 시더가 되었을 때의 상황을 나타낸다.

시더는 회선의 대부분을 업로드에 사용하므로 리처에 비해 업로드 비율이 압도적으로 높다. 따라서 다운로더는 시더에게 무조건 다운로드를 시도하고, 결국 상태가 좋지 않은 일부 피어를 제외한 모든 피어로부터 다운로드를 받는다.

4.2.2 RTT에 기반한 다운로드

RTT를 고려한 파일 다운로드 모습은 그림 11, 그림 12, 그림 13에서 확인할 수 있다.

그림 11은 다운로드 직후 피어들과의 연결 상태이다.

Info Files Trackers Peers Speed Settings

Address	Download	Upload	RTT(ms)
192.168.0.7:6969	0kB/s	0kB/s	5
192.168.0.13:6969	0kB/s	30kB/s	5
192.168.0.12:6969	0kB/s	42kB/s	4
192.168.0.15:6969	0kB/s	0kB/s	0
192.168.0.6:6969	830kB/s	0kB/s	2
192.168.0.11:6969	0kB/s	0kB/s	11
192.168.0.4:6969	894kB/s	120kB/s	7
192.168.0.5:6969	1122kB/s	588kB/s	1
192.168.0.3:6969	0kB/s	0kB/s	4
192.168.0.14:6969	0kB/s	8kB/s	4

그림 11. RTT 토렌트 버전에서 다운로드 직후 상황
Fig. 11. Snapshot of downloader in RTT torrent version right after starting downloading

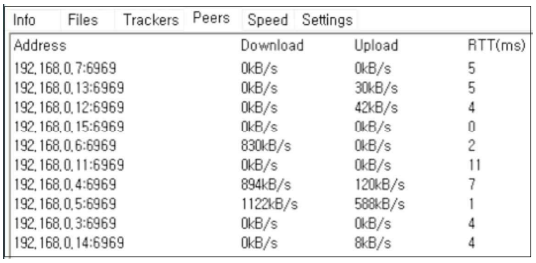

그림 12. RTT 토렌트 버전에서 다운로드 상황
Fig. 12. Snapshot of downloader in RTT torrent version when almost half of downloading is done

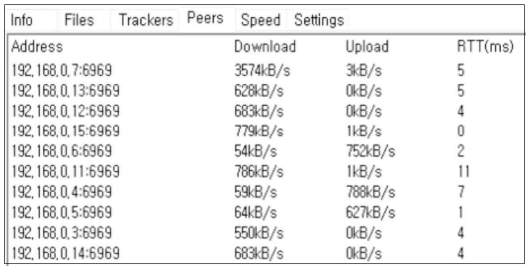

그림 13. RTT 토렌트 버전에서 피어들이 시딩 상태일 때의 다운로드 상황
Fig. 13. Snapshot of downloader in RTT torrent version when other peers are seeding

다운로드가 상당 부분 진행 되었을 경우 그림 12과 같은 상황이 된다. RTT가 작은 피어들에게 우선적으로 다운로드를 요청 하였으나 일부 피어들의 업로드 불량으로 인해 차선 피어에게 다운로드를 받고 있는 모습이 확인된다.

그림 13은 대다수의 피어가 시더가 되었을 때의 상황을 나타낸다. 시더는 업로드에 충실하므로 다른 리처보다 높은 연결 우선순위를 가지며 다운로더는 시더에게 무조건 연결을 요청한다. 결과적으로 업로드 상태가 불량한 시더를 제외한 모든 시더들에게 파일조각을 다운로드를 받게 된다.

4.2.3 결과 분석 및 기존 토렌트와 비교

표 4, 표 5는 각각 내부 망, 인터넷 환경에서 약 1GB 파일을 다운로드 받았을 때 걸리는 시간을 표로 정리한 것이다. 인터넷 환경에서 파일을 다운로드 할 때 더 많은 시간이 걸렸다는 점에서 ip우회가 성공적으로 이루어졌음을 알 수 있다.

표 4. 내부 망에서의 실험 결과
Table 4. The result from intranet environment

Unit (s)	First	Second	Third	AVE
CURRENT	158	164	196	173
RTT	169	171	186	175
MUS	152	148	195	165

표 5. 인터넷 망에서의 실험 결과
Table 5. The result from internet environment

Unit (s)	First	Second	Third	AVE
CURRENT	334	386	341	354
RTT	300	363	315	326
MUS	311	372	319	334

표 4의 결과를 보면 내부 망에서 RTT 버전 토렌트는 기존 토렌트와 크게 다르지 않으나 최대 업로드 속도 버전 토렌트는 기존 토렌트보다 빠른 것으로 확인되었다. 하지만 다운로드 순서가 뒤바뀌기도 하였으며, 두드러지는 속도 개선은 없었다. 표 5의 결과를 보면 외부 망에서 RTT버전 토렌트와 최대 업로드 속도 버전 토렌트는 기존 토렌트보다 확연히 빨랐다. 모든 실험에서 개선된 버전은 기존 버전보다 빠른 속도로 파일을 다운로드 하였으며, RTT 버전 토렌트는 최대 업로드 속도 버전 토렌트보다 조금 더 빨랐다.

그림 14는 내부 망에서 각 토렌트 버전에서의 다운로드 속도를 시간별로 나타낸 그래프이다. 다이아몬드 표식을 가진 최대 업로드 속도 버전 토렌트는 시더가 거의 없는 다운로드 초기에도 높은 다운로드 속도를 유지하며 다운로드를 이어나간다. 다시 말해, 성능 지표에 기반 하여 건강한 피어에게 우선적으로 다운로드를 요청하는 방식이 효과가 있었다. 이에 반해 정사각형 표식을 가진 RTT 버전 토렌트와 역삼각형 표식을 가진 기존 토렌트는 주변 리처들이 시더가 되고 나서야 급격한 다운로드 속도 향상을 보이고 있다. 해당 차트는 최대 업로드 속도 버전 토렌트는 타 토렌트보다 빠르고, RTT버전 토렌트는 기존 토렌트와 유사하다는 표 4의 결과를 뒷받침하고 있다.

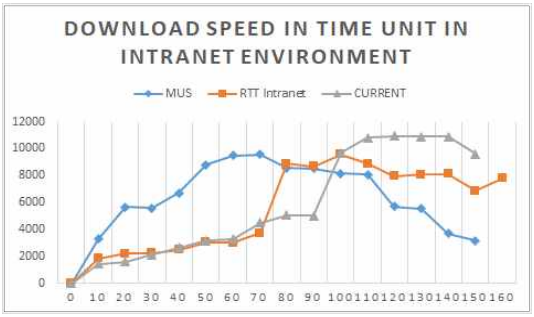

그림 14. 내부 망에서 시간대별 다운로드 속도
Fig. 14. Download speed of each torrent versions in intranet environment

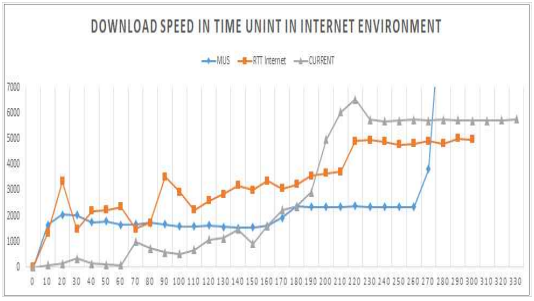

그림 15. 인터넷 망에서 시간대별 업로드 속도
Fig. 15. Download speed of each torrent versions in internet environment

그림 15는 인터넷 망에서 각 토렌트 버전에서의 다운로드 속도를 시간별로 나타낸 그래프이다. 개선된 토렌트 버전은 시더가 거의 없는 다운로드 초기에 기존 토렌트보다 높은 다운로드 속도를 유지하며 다운로드를 이어나간다. 하지만 최대 업로드 속도 버전 토렌트는 RTT 버전 토렌트보다 다운로드 속도가 약간 느리다. 기존 토렌트는 주변 피어들이 시더가 된 이후에야 급격하게 다운로드 속도가 빨라진다는 점에서 성능 개선이 이루어지지 않은 상태임을 확인할 수 있다.

V. 결 론

본 논문에서는 가장 먼저 P2P 서비스에서 다운로드 성능 향상을 목표로 하는 기존 연구와 그 문제점을 살펴보았다. 또한 성능지표를 새롭게 정의하여 건강한 피어를 선택할 수 있는 척도로 활용하였다. 본 논문에서 제시한 성능 지표는 최고 업로드 속도와 RTT이며 성능 개선이 전혀 이루어지지 않은 기존 토렌트와 각각의 성능 지표를 활용하여 개선이 이루어진 토렌트의 성능을 다운로드 시간을 기준으로 비교하였다.

내부 망 환경에서는 최고 업로드 속도 버전 토렌트가 타 버전보다 약간 우세했으나 괄목할만한 성능 향상을 이끌어내지는 못했다. 인터넷 망 환경에서는 개선된 토렌트 버전이 기존 토렌트 버전보다 약 10% 정도 빨랐다. 두 가지 성능 지표를 모두 고려한 토렌트를 개발한다면 큰 성능 향상이 있을 것으로 기대된다. 개선된 토렌트 버전의 공통점은 시더가 거의 없는 다운로드 초기에도 높은 다운로드 속도를 유지한다는 것인데, 이것은 건강한 피어들을 성공적으로 식별한 결과이다.

References

[1] BitTorrent Still King of P2P Traffic, http://torrentfreak.com/bittorrent-still-king-of-p2p-traffic-090218/ [Accessed : Nov. 30, 2017]

[2] Simon Oechsner, Frank Lehrieder, Tobias Hoßfeld, Florian Metzger, and Dirk Staehle, "Pushing the Performance of Biased Neighbor Selection through Biased Unchoking", P2P '09. IEEE Ninth International Conference on, pp. 1-10, Sep. 2009.

[3] Bram Cohen, "Incentives build robustness in BitTorrent", Workshop on Economics of Peer-to-Peer systems, pp. 1-5, May 2003.

[4] Principles of BitTorrent Protocol, https://www.netmanias.com/ko/?m=view&id=techdocs&no=10644 [Accessed : Nov. 30, 2017]

[5] Daniel S. Bernstein, Zhengzhu Feng, Brian Neil Levine, and Shlomo Zilberstein, "Adaptive Peer Selection", International Workshop on Peer-to-Peer Systems, pp. 237-246, 2003.

[6] Menno Tammens, "Comparing Peer Selection Mechanisms in P2P Systems", 15th Twente Student Conference on IT, 2011, pp. 1-8, Jun. 2011.

[7] Jan Seedorf, Sebastian Kiesel, and Martin Stiemerling, "Traffic Localization for P2P-

Applications: The ALTO Approach", P2P '09. IEEE Ninth International Conference on, pp. 1-7, Sep. 2009.

[8] Problem of P2P and appearance of P4P, https://www.netmanias.com/ko/?m=view&id=techdocs&no=10648 [Accessed : Oct. 10, 2017]

[9] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yabin Liu, and Avi Silberschatz, "P4P: Provider Portal for Applications", ACM SIGCOMM, pp. 1, Aug. 2008.

[10] In-Chul Hwang, Jahwan Koo, and Ung-Mo Kim, "S-Tracker Design and Implementation for the Guarantee of File Download Availability in BitTorrent", Journal of KIIT, Vol. 15, No. 12, pp. 29-38, Dec. 2017.

저자소개

박 용 현 (Yong-Hyeon Park)

2015년 3월 ~ 현재 : 성균관대학교 컴퓨터공학과 재학중

관심분야 : P2P, BitTorrent, BigData, Data Security, Reversing Engineering

구 자 환 (Jahwan Koo)

1995년 : 성균관대학교 정보공학과 졸업(학사)

1997년 : 성균관대학교 일반대학원 전기전자컴퓨터공학 졸업 (공학석사)

1999년 ~ 2002년 : LG CNS 정보기술연구소 연구원

2006년 : 성균관대학교 일반대학원 전기전자컴퓨터공학 졸업(공학박사)

2007년 ~ 2010년 : 미국 위스콘신대학교 컴퓨터과학과 박사후 연구원

2016년 ~ 현재 : 성균관대학교 사회과학대학 연구교수

관심분야 : Data Communication, Cloud Computing, Big Data

김 응 모 (Ung-Mo Kim)

1981년 : 성균관대학교 수학과 졸업(학사)

1986년 : 미국 Old Dominion 대학교 컴퓨터과학과 졸업 (공학석사)

1990년 : 미국 Northwestern 대학교 컴퓨터과학과 졸업 (공학박사)

1990년 ~ 현재 : 성균관대학교 소프트웨어대학 교수

관심분야 : Database, Data Mining, Big Data