Journal of KIIT. Vol. 15, No. 12, pp. 29-38, Dec. 31, 2017. pISSN 1598-8619, eISSN 2093-7571 29
http://dx.doi.org/10.14801/jkiit.2017.15.12.29

비트토렌트에서 파일 다운로드 가용성 보장을 위한 S-트래커 설계 및 구현

황인철*, 구자환**, 김응모***

S-Tracker Design and Implementation for the Guarantee of File Download Availability in BitTorrent

In-Chul Hwang*, Jahwan Koo**, and Ung-Mo Kim***

본 연구는 문화체육관광부 및 한국저작권위원회의 2017년도 저작권기술개발사업의 연구결과로 수행되었음

요 약

기존의 비트토렌트에서는 특정 조각을 배포중인 피어가 하나도 없어 다운로더가 이 조각을 다운로드 받지 못해 완전체 파일을 얻지 못하는 경우가 발생한다. 이러한 경우를 파일 다운로드 가용성이 1보다 작다고 하며 이로인해 비트토렌트의 신뢰성은 낮아지게 된다. 본 논문에서는 비트토렌트에서 파일 다운로드 가용성 보장을 위한 S-트래커를 제안한다. 제안하는 S-트래커는 시더로부터 토렌트 파일을 건네받아 실행하여 파일을 다운로드 및 보관하며, 다른 다운로더들에게 파일정보를 제공하는 역할을 한다. 이로인해 비트토렌트에서 파일 가용성이 1보다 작은 경우를 회피하게 하며, 우리는 이 S-트래커가 기존 비트토렌트의 고질적인 문제였던 파일 다운로드 가용성을 보장하지 못하는 문제점을 극복하였음을 보인다.

Abstract

In the existing BitTorrent, there is no peer distributing a particular piece, so the downloader can not download this piece and can not get the whole file. In this case, the file download availability is less than 1, which lowers the reliability of the bittorrent. In this paper, we propose an S-tracker for file download availability in bit torrent. The proposed S-Tracker handles torrent files from cedar to download and archive files, and provides file information to other downloaders. This avoids the case where the file availability is less than 1 in the bittorrent, and we show that this S-tracker overcomes the problem of not guaranteeing file download availability, which was a persistent problem of existing BitTorrent.

Keywords
P2P, bit torrent, availability, download, s-tracker

* 성균관대학교 수학과
- ORCID: http://orcid.org/0000-0002-1664-9854
** 성균관대학교 사회과학대학 연구교수(교신저자)
- ORCID: http://orcid.org/0000-0002-2844-3183
*** 성균관대학교 소프트웨어대학 교수
- ORCID: http://orcid.org/0000-0001-5464-6358

· Received: Nov. 06, 2017, Revised: Dec. 04, 2017, Accepted: Dec. 07, 2017
· 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(Peer-to-Peer) 혹은 동등 계층간 통신망은 비교적 소수의 서버에 집중하기보다는 망 구성에 참여하는 기계들의 계산과 대역폭 성능에 의존하여 구성되는 통신망이다. 순수 P2P 파일 전송 네트워크는 클라이언트나 서버라는 개념 없이, 오로지 동등한 계층 노드들이 서로 클라이언트와 서버 역할을 네트워크 위에서 동시에 하게 된다. 이 네트워크 구성 모델은 보통 중앙 서버를 통하는 형태의 클라이언트-서버 모델과는 구별이 된다[1].

토렌트는 P2P 방식을 사용하는 다운로드 서비스이다. 토렌트는 크게 클라이언트, 피어, 트래커 등으로 구성되어 있으며, 트래커는 여러 피어들의 메타정보들을 가지고 있고 클라이언트는 그 메타정보들을 받아 다른 피어들로부터 파일의 조각을 다운로드 받는다. 클라이언트가 파일 다운로드를 요청하면 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 클라이언트에게 전달한다. 클라이언트는 피어리스트의 피어들을 선택 및 연결하여 그들이 가지고 있는 파일 조각을 다운로드 받아 완전체 파일을 만든다[2].

그러나 배포를 하던 피어들이 배포를 중단하여 배포 중인 파일 조각들로 완전한 파일을 만들수 없는 경우가 발생할 수 있다. 본 논문에서는 이러한 경우를 극복하여 파일의 다운로드 가용성을 보장할 수 있는 S-트래커를 생성 및 운영하는 방법을 제안한다.

S-트래커는 토렌트파일의 가용성이 1보다 작아져 다운로드 받지 못하는 경우를 해결하기 위한 것이다. 이 S-트래커를 사용하면 지금까지 비트로렌트의 고질적인 문제였던 가용성이 보장되지 않아 파일을 다운로드 받지 못하는 문제를 해결하여 다운로더들의 원활한 다운로드를 보장 할 수 있다[3].

본 논문의 구성은 다음과 같다. 2장에서는 기존 연구를 기술하고, 3장에서는 구체적인 구현 방안을 S-트래커의 기능별로 시퀀스 다이어그램을 통해 표현하고 기술한다. 4장에서는 3장에서 설계한 기능을 구현 및 성능을 평가 한다. 마지막으로 5장에서는 결론에 대해 기술하고 본 연구가 갖는 한계점 및 향후 연구에 대하여 기술한다.

Ⅱ. 기존 연구

2.1 가용성 계산 방법

토렌트에서 가용성은 완전한 파일을 다운 받을 수 있는 조각 뭉치들을 가용성의 정수 부분으로, 그 외의 나머지 즉 완전한 파일을 다운 받을 수 없는 남은 조각들을 가용성의 소수 부분으로 하여 계산한다[4]-[7].

표 1에서 완전체 파일을 다운 받을 수 있는 조각은 피어 A의 조각 1, 2, 3과 피어 C의 조각 4, 5로 한 뭉치 피어 B의 조각 2, 3, 4와 피어 D의 조각 1, 5로 한 뭉치가 총 두 뭉치의 파일이 나온다.

표 1. 배포중인 피어들이 가지고 있는 파일조각
Table 1. Files fragmented by the peers in deployment

	piece1	piece2	piece3	piece4	piece5
peer A	o	o	o		
peer B		o	o	o	
peer C			o	o	o
peer D	o			o	o

따라서 가용성의 정수부분은 2가 되고, 그 나머지 피어 C의 조각 3과 피어 D의 조각 4, 총 조각 3, 4 두 가지가 남으므로 가용성의 소수 부분은 0.4가 되어 총 가용성은 2.4가 된다[6][7].

위의 상황에서 몇 개의 피어가 배포를 중단하여 가용성이 1보다 작아질 수 있다. 예를들어 피어 A와 피어 B가 배포를 중단한다면 배포중인 피어는 피어 C와 피어 D만 남게 된다. 표 2를 보면 이때 이 두 피어가 가지고 있는 파일 조각은 아무리 합쳐도 완전한 파일 1개가 나오지 않는다. 즉 가용성이 1보다 작은 0.8이 되어 다운로더는 완전한 파일을 다운로드 받을 수 없다.

표 2. 피어 A, B의 배포 중단 후 피어들이 가지고 있는 파일조각
Table 2. File fragments held by peers after distribution of peers A and B

	piece1	piece2	piece3	piece4	piece5
peer A					
peer B					
peer C			o	o	o
peer D	o			o	o

표 3. S-트래커가 추가된 후 배포중인 피어들이 가지고 있는 파일조각
Table 3. File fragments held by distributed peers after S-Tracker was added

	piece1	piece2	piece3	piece4	piece5
peer A					
peer B					
peer C			o	o	o
peer D	o			o	o
S-tracker	o	o	o	o	o

표 3은 최소 가용성 1을 보장하게끔 S-트래커 1개가 존재할 때 표 1의 상황에서 피어 A, B가 순차적으로 배포를 중단하고 난 뒤 각 피어들이 가지고 있는 조각들의 상태를 나타낸다. 이 경우에서는 S-트래커가 파일 정보를 보유하고 있기 때문에 가용성은 1.8이 되어 다운로더가 완전한 파일을 받을 수 있게 된다.

2.2 토렌트파일 업로드 및 다운로드

토렌트파일의 업로드는 최초 업로더 시더가 신규 토렌트 파일을 만드는 것으로부터 시작한다. 업로더 시더는 파일을 조각으로 나누는 과정을 통해 토렌트파일을 만들고 자신과 파일의 정보를 트래커에 배포를 하는데 보통 조각당 256KB로 나누게 된다. 그러면 트래커는 이 정보를 토대로 스웜을 생성하게 되고 그 후 이 파일을 웹 콘텐츠 서버에 업로드를 하게 되면 다운로더들은 웹 콘텐츠 서버에서 토렌트파일을 검색 및 다운로드 할 수 있다. 이 과정은 그림 1의 토렌트파일 업로드 및 다운로드 부분에 나와있다[8][9].

2.3 피어리스트 생성

피어리스트는 다운로더가 토렌트파일을 실행함으로써 만들어지기 시작한다. 다운로더가 토렌트파일을 실행하면 트래커에게 트래커 요청 메세지를 보낸다. 이 메세지에는 토렌트파일의 해쉬값 등 여러 정보를 포함하고 있는데 이 메세지를 받은 트래커는 자신이 받은 해쉬값과 동일한 해쉬값을 갖고 있는 스웜을 탐색한다. 이때 스웜이 존재하지 않는다면 새로운 스웜을 생성하는데 이 경우는 파일이 최초로 업로드 되는 경우이다. 동일한 해쉬값을 갖고 있는 스웜을 찾았으면 스웜에 있는 피어의 IP주소들로 피어리스트를 생성한다. 일반적으로 최대 50개까지 생성이 되며, 피어의 개수가 50개를 넘을 시에는 랜덤하게 50개를 선택한다. 이렇게 생성된 피어리스트는 트래커 응답 메세지로서 다운로더에게 전해진다. 이 과정은 그림 1의 피어리스트 생성 부분에 나와있다[8][9].

2.4 파일 다운로드

파일 다운로드 과정은 그림 1의 파일 다운로드 과정과 같다. 다운로더가 위의 피어리스트 생성과정에서 만들어진 피어리스트를 트래커 응답 메세지로 받으면 다운로더는 피어리스트에 있는 피어들의 IP주소로 파일의 해쉬값과 피어 ID를 전송하여 연결 가능 여부를 파악한다. 피어가 초크상태 일 경우 파일조각을 얻기 못하기에 연결 가능 여부를 파악하는 것이다. 그러면 해쉬값을 받은 피어중 해쉬값이 같으며 언초크 피어는 자신의 해쉬값과 피어 ID를 다운로더에게 전송하고 다운로더는 이 피어들과 세션을 연결하여 파일을 서로 주고받는다.

이때 다운로더가 한번에 다운로드 받는 파일조각은 4개인데 이 조각을 선택하는 것은 일련의 조각 선택 알고리즘을 따른다. 다운로더가 다운로드 받은 파일조각들은 다른 피어들에게 자신이 해당 파일조각을 가지고 있다는 정보를 전송하여 해당 파일조각이 없는 다른 피어들이 자신에게 파일조각을 다운로드 받게끔 유도한다. 다운로더는 다운로드가 완료 될 때 까지 위 과정을 반복 시행한다.

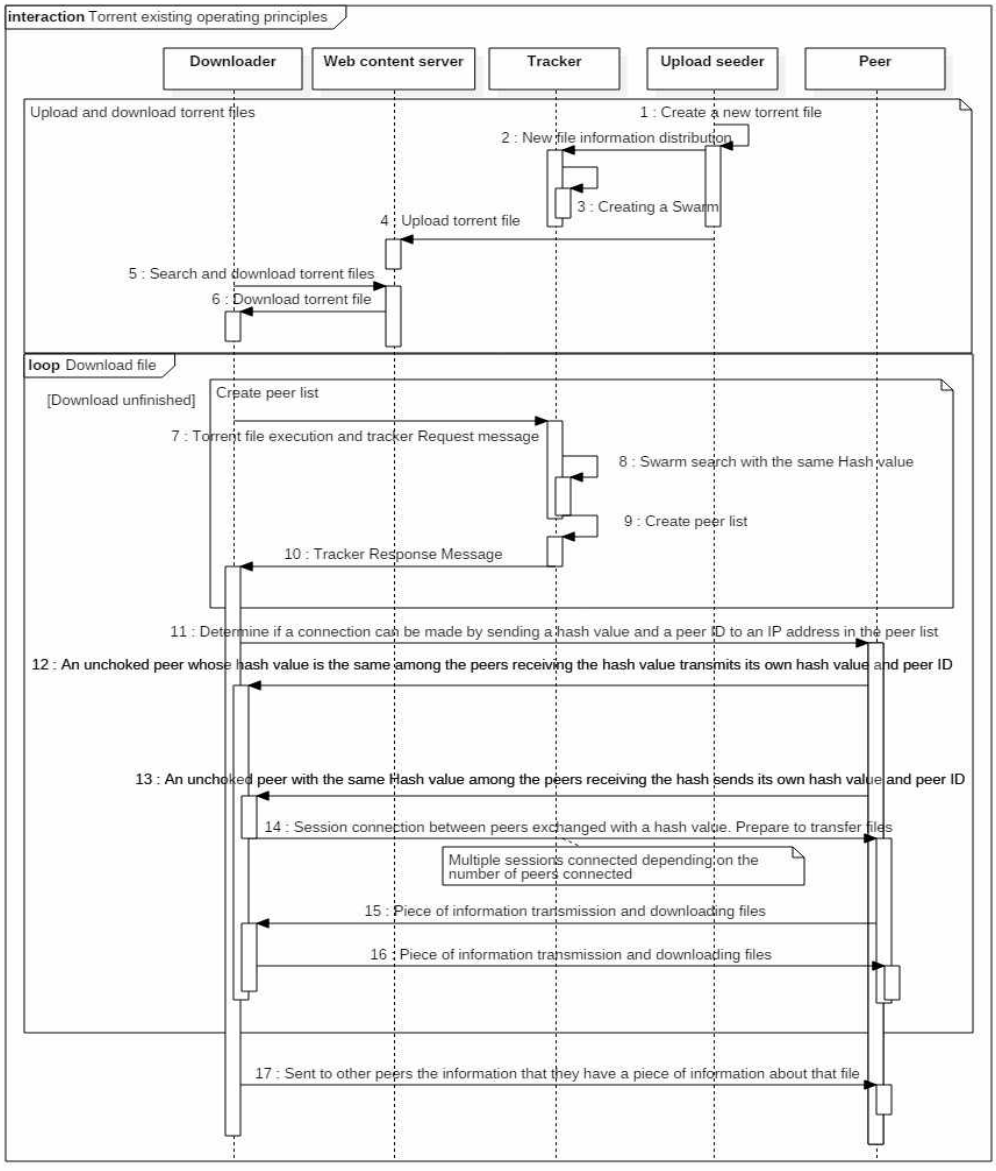

그림 1. 토렌트의 작동원리
Fig. 1. How a torrent works

III. S-트래커를 이용한 파일 다운로드 보장 방안

3.1 S-트래커의 최초파일 복제 매커니즘

그림 2의 S-트래커의 최초 파일 복제 매커니즘과 같이 최초 업로드 시더가 신규 파일정보 배포를 시작하면 트래커에 스웜을 생성시키고 그 스웜에 최초유포자의 URL 주소값과 신규 파일의 해쉬값 등이 저장된다. 이후 시더는 이 토렌트파일을 S-트래커에게 전달한다. 토렌트파일을 받은 S-트래커는 이 토렌트파일과 같은 이름의 파일이 S-트래커에 존재하는지 확인을 하고 존재하지 않을 시에 다운로드 받은 토렌트파일을 실행하여 파일을 다운로드 받는다. 이때 S-트래커는 최초로 업로드 된 파일의 정보를 갖게 된다.

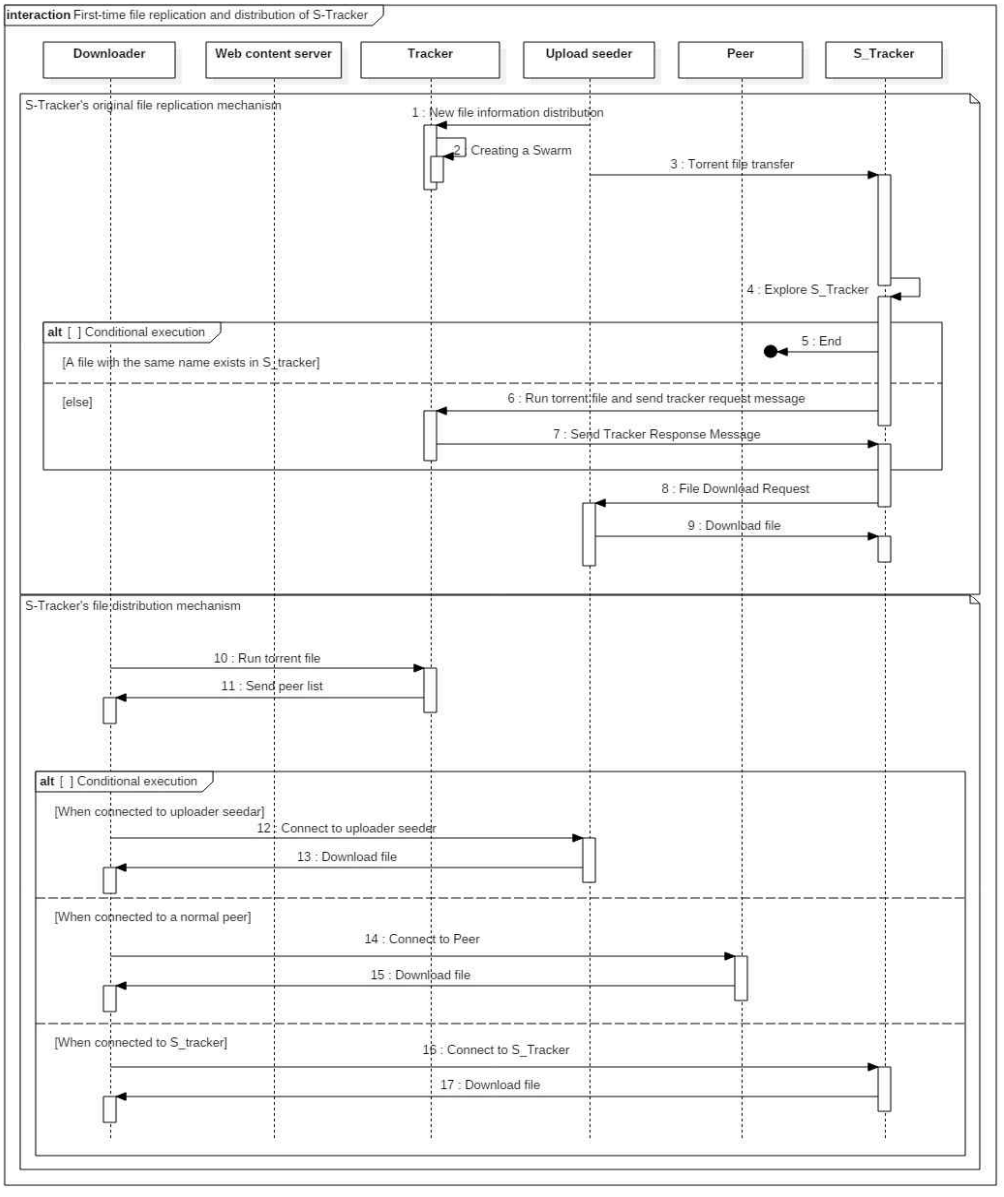

그림 2. S-트래커의 최초파일 복제 및 배포 방법
Fig. 2. How S-tracker originally replicates and distributes files

3.2 S-트래커의 파일 배포 매커니즘

다운로더가 파일을 다운로드 받는 과정은 그림 2의 S-트래커의 파일 배포 매커니즘 부분과 같다. 다운로더가 토렌트파일을 실행하면 이에 연결된 트래커는 스웜에서 피어리스트를 만들고 이것을 다운로더에게 전송한다. 그러면 다운로더는 피어리스트에 있는 피어들에 연결을 해서 파일을 다운로드 받는다. 이때 트래커의 스웜에는 S-트래커도 피어로서 포함이 되어 있다. 파일 다운로드 과정에서 S-트래커는 다른 피어들과 마찬가지로 파일 정보를 제공하는 역할을 한다.

다만 이 S-트래커는 파일 조각들을 배포중인 다른 피어들에게 없는 나머지 파일 조각들을 가지고 있다.

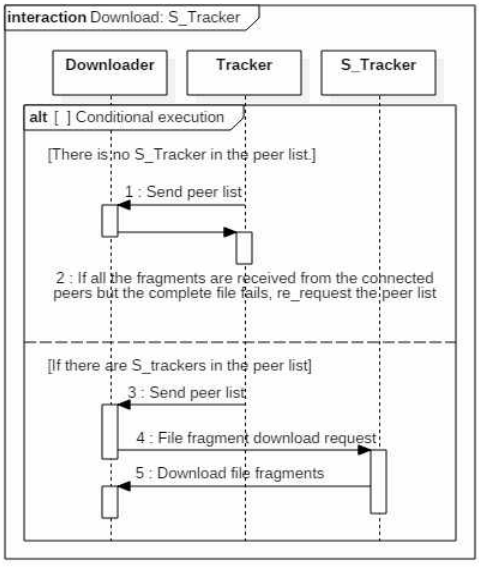

그림 3. S-트래커 유/무에 따른 파일 다운로드
Fig. 3. Downloading files according to S-tracker availability

그림 3을 보면 다운로드 과정에서 피어리스트의 모든 파일 조각을 다운로드 받아도 완전한 파일을 다운받지 못하는 경우가 생긴다. 이럴 경우 다운로더는 트래커에 새로운 피어리스트를 요청하게 된다. 최악의 경우 이 과정을 수십번 반복 하게 될 수 있다. 오직 1개의 피어(S-트래커)에만 존재하는 파일 조각을 다운받지 못한 상황이 발생한 것이다. 그러나 S-트래커도 피어이기 때문에 이 과정을 반복하게 되면 언젠가는 파일 다운로드를 완료할 수 있다. 이러한 방법으로 S-트래커는 파일 다운로드 가용성을 보장한다.

3.3 S-트래커의 파일 정보 관리 매커니즘

그림 4는 앞에서 설명한 매커니즘의 동작시 S-트래커가 파일 정보를 어떻게 보관하고 있는지를 보여준다. S-트래커 2개를 운영하는 상황에서 시더가 파일 A와 파일 B의 토렌트파일을 생성하면 2개의 S-트래커는 이 파일을 다운받는다. 이때의 파일 A, B의 가용성은 3이 된다. 그 후 시더가 파일 A, B의 배포를 중단하고 파일 C, D의 토렌트파일을 생성하면 파일 A, B의 가용성은 2가 되고 파일 C, D의 가용성은 3이 된다.

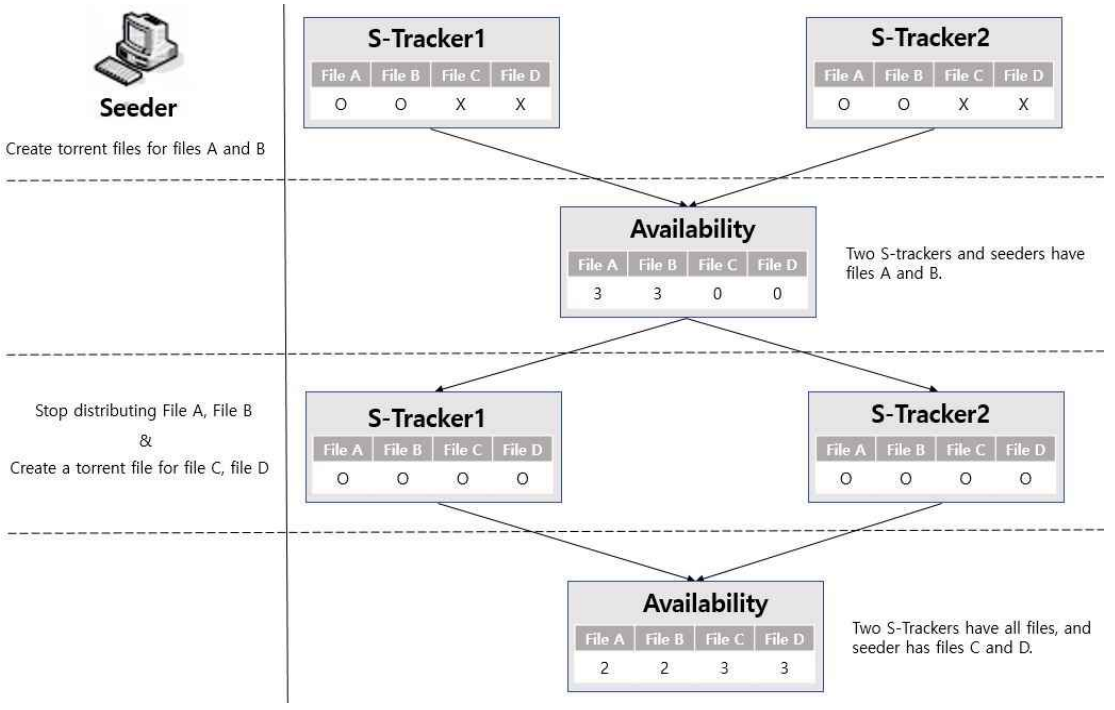

그림 4. 시더의 토렌트파일 생성 및 파일배포 중단에 따른 S-트래커의 동작
Fig. 4. S-tracker's behavior due to ceder's torrent file creation and file distribution disruption

이와 같이 S-트래커가 파일 정보를 가지고 있으면 시더나 다른 피어들이 배포를 중단해도 다운로더들은 S-트래커가 배포하고 있는 파일들에 의해 파일을 다운 받을 수 있다. S-트래커가 작동을 중단할 경우에는 그만큼의 가용성이 낮아지지만 이 문제는 다수의 S-트래커를 운용함으로써 해결할 수 있다.

Ⅳ. 실험방법 및 결과

본 논문을 통해 구현되는 S-트래커는 몇 개를 운영하느냐에 따라서 가용성이 보장되는 정도가 달라진다. 본 논문은 S-트래커 1개를 운영하여 가용성 1를 보장하는 것을 전재로 실험을 하였으며 이를 기반으로 작성하였다.

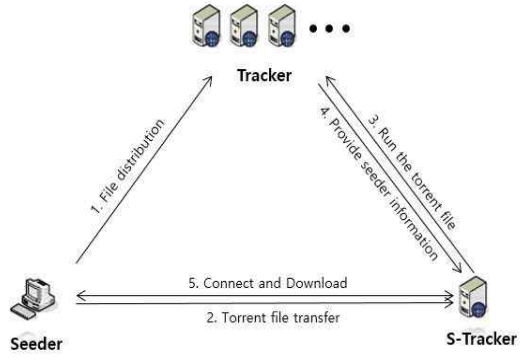

그림 5. 최초 토렌트파일 업로드
Fig. 5. Uploading the first torrent file

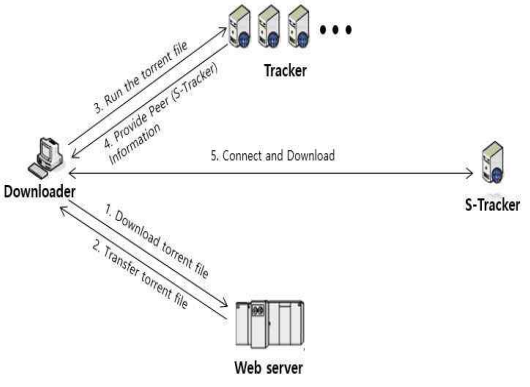

그림 6. 다운로더와 S-트래커의 1:1 다운로드
Fig. 6. 1: 1 Download of downloader and S-tracker

4.1 실험방법

이 논문에서 구현하는 것은 크게 두가지 이다. 먼저 S-트래커의 최초파일 복제 매커니즘을 구현해야 하며 그 다음으로 S-트래커의 파일 배포 매커니즘을 구현해야 한다.

4.1.1 S-트래커의 최초파일 복제 매커니즘 구현 실험방법

최초파일 복제 매커니즘은 그림 5와 같다. 이를 구현하기 위해서는 트래커의 역할을 할 컴퓨터 1대, 시더 역할을 할 컴퓨터 1대, S-트래커를 구축할 컴퓨터 1대로 총 3대의 컴퓨터와 이들을 연결 할 포트가 필요하다.

시더가 토렌트파일을 만들었을 때 S-트래커는 이 토렌트파일을 전달받아 실행하여 파일을 다운로드 받는다.

이때 S-트래커의 폴더 안에 토렌트파일과 다운로드받은 파일이 존재하는 것을 확인함으로써 S-트래커의 최초파일 복제 매커니즘이 정상적으로 실행되었음을 확인할 수 있다.

4.1.2 S-트래커의 파일 배포 매커니즘 구현 실험방법

두 번째로 구현해야 하는 것은 S-트래커의 파일 배포 매커니즘이다. 이를 구현하기 위해서는 트래커의 역할을 할 컴퓨터 1대, 시더 역할을 할 컴퓨터 1대, S-트래커 역할을 할 컴퓨터 1대, 다운로더 역할을 할 컴퓨터 1대로 총 4대의 컴퓨터와 이들을 연결 할 포트가 필요하다.

실험은 앞의 4.1.1 최초파일 복제 매커니즘 실험에 이어 실시한다. 그림 6을 보면 먼저 시더가 파일 배포를 중단한 뒤 다운로더는 시더가 생성한 토렌트파일을 받아 실행한다. 이때 다운로더가 파일 다운로드를 완료한다면 S-트래커의 파일 배포 매커니즘을 확인할 수 있다.

4.2 실험결과

실험에 사용한 트래커의 주소는 210 . 126 . 31 . 176, 시더의 주소는 192 . 168 . 0 . 14, S-트래커의

주소는 192 . 168 . 0 . 7이며 다운로더의 주소는 192 . 168 . 0 . 8 이다.

4.2.1 S-트래커의 최초파일 복제 매커니즘 구현 실험결과

먼저 시더에서 토렌트파일 생성 및 배포하는 동작은 그림 7이다. 시더는 2017-11-24 오후 4:13:59에 964.54MB의 'Torrent Experiment File.mp4' 파일을 성공적으로 만든 후 배포하는 것을 확인할 수 있다.

그 후 그림 8을 보면 S-트래커(192 . 168 . 0 . 7)는 시더(192 . 168 . 0 . 14)와 연결을 하였고, 시더의 최고 업로드 속도 7864kbps로 파일을 다운 받았음을 알 수 있다. 시더가 S-트래커에게 준 토렌트파일은 1개이므로 Recived가 1이 되고 파일 다운로드가 완료 되었기에 다운로드는 1에서 0으로, Seeding은 0에서 1로 바뀐다.

그렇다면 이 토렌트파일과 파일을 S-트래커가 가지고 있음을 직접 확인해 보자. 그림 9를 보면 'Torrents' 폴더에 '24d16h13m19s'라는 토렌트파일이 생성되었음을 볼 수 있다. 이는 이 토렌트파일이 24일 16시 13분 19초에 생성되었음을 의미한다. 이번에는 그림 10을 보자. 'Files' 폴더에 앞에서 토렌트파일로 만들었던 'Torrent Experiment File.mp4' 파일이 성공적으로 생성된 것을 확인할 수 있다. 이로써 S-트래커의 최초파일 복제 매커니즘 구현을 확인하였다.

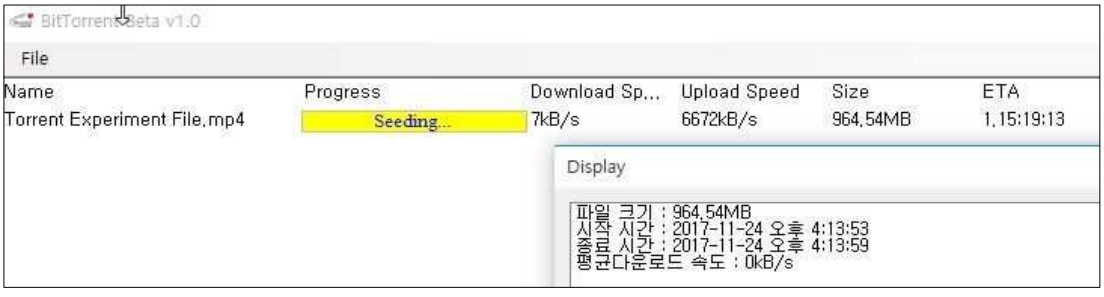

그림 7. 시더의 토렌트파일 생성 완료
Fig. 7. Seeder's torrent file creation completed

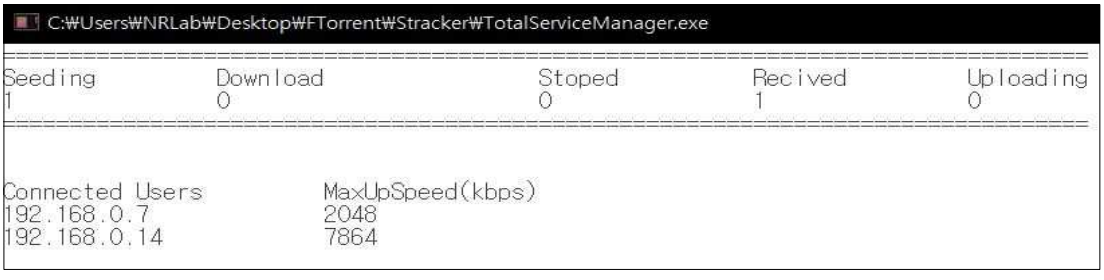

그림 8. S-트래커의 파일 다운로드 완료
Fig. 8. S-tracker file download complete

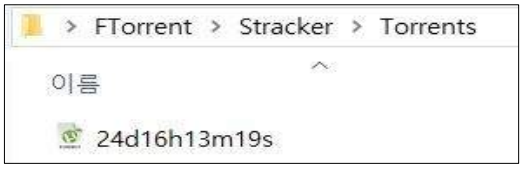

그림 9. S-트래커가 보유중인 토렌트파일
Fig. 9. Torrent file owned by S-tracker

그림 10. S-트래커가 다운로드한 파일
Fig. 10. Files downloaded by S-tracker

4.2.2 S-트래커의 파일 배포 매커니즘 구현 실험 결과

먼저 그림 11과 같이 4.2.1의 실험에 이어 피어의 배포를 중단한 뒤 실험을 하였다. 그 후 다운로더는 웹 콘텐츠 서버에서 토렌트파일을 다운로드 받아 실행하여 파일 다운로드를 진행 하였다. 다운로드를 실행한 결과 그림 12와 같이 'Torrent Experiment File.mp4'파일을 S-트래커(192 . 168 . 0 . 7)로부터 다운받고 있는 것을 볼 수 있다.

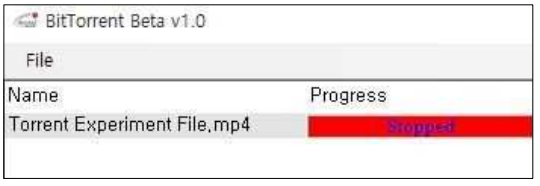

그림 11. 피어의 파일 배포 중단
Fig. 11. Stopping file distribution on the peer

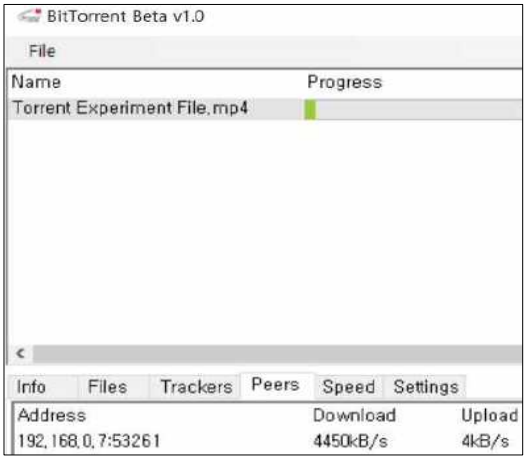

그림 12. 다운로더와 S-트래커의 1대1 다운로드 과정
Fig. 12. Downloader and S-tracker one-to-one download process

다운로더의 다운 완료 후 그림 13을 보면 S-트래커에 시더 뿐만 아니라 다운로더(192 . 168 . 0 . 9)도 연결 돼 있는 것을 확인할 수 있다. 다운로더의 파일 다운로드가 끝났기 때문에 즉, S-트래커의 파일 배포가 끝났기 때문에 업로딩은 1에서 0으로 변한 상태이다. 다운로더의 업로드속도가 낮은 것은 파일을 배포해주는 피어가 아닌, 파일을 다운로드 받는 피어이기 때문이다. 이로써 S-트래커의 파일 배포 매커니즘 구현을 확인하였다.

V. 결론 및 향후 과제

비트토렌트를 사용하는 사용자들은 자신이 다운받기를 원하는 파일의 가용성이 1보다 낮아 필요한 파일을 다운받지 못하는 불편함과 문제점을 야기했다. 이러한 문제점들을 해소하기 위하여 본 논문에서는 기존 P2P 서비스에서 파일 다운로드 가용성의 문제점을 제시하고 이를 극복하기 위해 S-트래커를 구현하였다.

본 논문에서는 기존 비트토렌트 시스템에 파일들의 가용성을 보장할 수 있게 하는 S-트래커를 제안하였다. 이를 위해 오픈소스로 공개되어 있는 F토렌트의 코드를 수정하여 클라이언트를 수정하였고, S-트래커를 새롭게 만들어 가동하였다. S-트래커가 토렌트파일의 가용성을 보장하는지 확인하기 위하여 테스트배드를 구축하여 S-트래커의 최초파일 복제 매커니즘과 파일 배포 매커니즘을 확인하여 다운로더가 파일을 다운로드 받을 수 있는 최소 가용성을 보장함을 보였다. 즉 파일을 배포하는 피어들이 하나도 없다고 하더라도 다운로더는 파일 다운로드를 완료 할 수 있다는 점에서 기존의 비트토렌트 시스템과 다르다는 것을 알 수 있다.

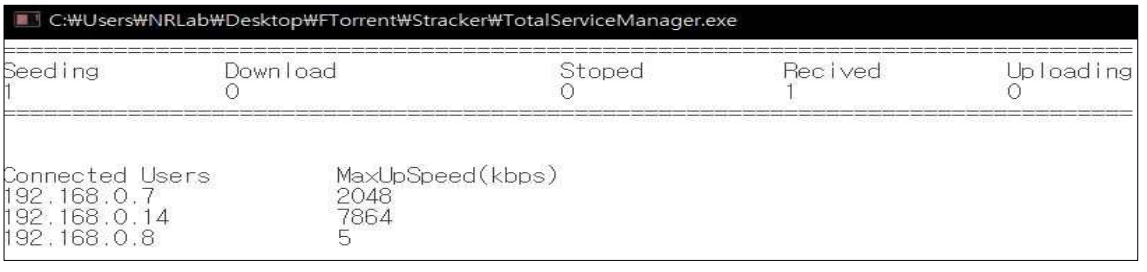

그림 13. 다운로더의 S-트래커와 1대1 다운로드시 S-트래커의 상태
Fig. 13. Status of S-Tracker on downloader's S-Tracker and 1-on-1 download

본 논문에서는 이전에 시도된 바 없던 P2P형식의 비트토렌트의 파일 다운로드 가용성을 보장할 수 있는 S-트래커를 제안 및 구현하고 실제로 가용성을 보장하는 것을 입증하였다는 점에서 연구에 의의를 찾을 수 있다. 그러나 S-트래커가 파일을 다운 받기만 하는 것은 제한된 컴퓨터의 용량에 무리를 줄 수 있다는 점에서 본 연구의 한계를 찾을 수 있다.

따라서 향후 과제로는 구현한 S-트래커의 효율적인 운용을 위하여 일정 시간마다 가용성을 체크하여 S-트래커가 스스로 파일을 다운 혹은 삭제하는 매커니즘과 더 나아가 파일단위가 아닌 조각 단위로 파일을 관리하는 기술을 개발하는 연구를 본 논문의 향후 과제로 한다.

References

[1] P2P, https://ko.wikipedia.org/wiki/P2P, [Accessed: May 08, 2017]

[2] B. Cohen, "Incentives build robustness in BitTorrent", Workshop on Economics of Peer-to-Peer systems. Vol. 6, pp. 68-72, May 2003.

[3] In Chul Hwang, Jahan Koo, and Ung Mo Kim, "A Guarantee of File Download Availability in P2P Environment", Proceedings of KIIT Summer Conference, pp. 394-396, Jun. 2017.

[4] Glossary of BitTorrent terms, https://en.wikipedia.org/wiki/Glossary_of_BitTorrent_terms#Availability, [Accessed: May 08, 2017]

[5] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher, "Availability in BitTorrent Systems", IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications, Jun. 2007.

[6] Availability, https://wiki.vuze.com/w/Availability, [Accessed: Jun. 15, 2017]

[7] How is 'availability' calculated in BitTorrent?, https://superuser.com/questions/1087085/how-is-availability-calculated-in-bittorrent, [Accessed: Jun. 15, 2017]

[8] naver bittorrent, http://blog.naver.com/PostView.nhn?blogId=manhdh&logNo=220038243469, [Accessed: Aug. 14, 2017]

[9] bittorrent protocol, https://www.netmanias.com/ko/post/techdocs/5185/p2p/understanding-of-the-bittorrent-protocol, [Accessed: Aug. 14, 2017]

저자소개

황 인 철 (In-Chul Hwang)

20011년 3월 ~ 현재 : 성균관대학교 수학과 재학중

관심분야 : P2P, BitTorrent, BigData

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