Journal of The Korea Society of Computer and Information
Vol. 18, No. 4, April 2013
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2013.18.4.065

한정된 자원 환경에서의 주문형 비디오 스트리밍 서비스를 위한 효율적인 인센티브 메커니즘

신 규 용*, 이 종 덕*, 신 진 희*, 박 찬 진*

An Incentive mechanism for VOD Streaming Under Insufficient System Resources

Kyuyong Shin*, Jongdeog Lee*, Jinhee Shin*, Chanjin Park*

요 약

과거에는 단순한 파일 공유 어플리케이션들이 인터넷 트래픽의 가장 큰 비중을 차지하고 있었지만 현재는 실시간 스트리밍을 포함한 비디오 어플리케이션들이 그 자리를 대체하고 있다. 이에 따라 협조적 분산 환경을 지원하는 피투피 (P2P) 어플리케이션들의 패러다임도 비디오 스트리밍을 지원하는 방향으로 변화하고 있다. 현재 비디오 스트리밍을 지원하는 피투피 방식들은 대부분 구현이 쉽고 스트리밍 어플리케이션으로의 전환이 용이한 비토렌트 (BitTorrent)를 기반하고 있다. 하지만 비토렌트 기반의 비디오 스트리밍 방식들은 무임승차에 취약한 비토렌트의 특성을 그대로 반영하기 때문에 무임승차에 의해 시스템 자원이 부족한 상황에서는 제대로 동작하지 않는 단점이 있다. 따라서 본 논문에서는 비토렌트 기반의 비디오 스트리밍 방식들의 무임승차에 대한 취약성을 분석하고, 기존 방식들에 대한 대안으로 (협조적 분산 환경 하에서 협동을 강제하기 위해 개발되었던) T-Chain의 적용 가능성에 대해 검토한다. 시뮬레이션 결과 T-Chain을 기반으로 한 비디오 스트리밍 방식인 S-TChain의 경우 부족한 자원 환경에서 비토렌트 기반의 다른 방법들에 비해 평균 60% 이상의 향상된 성능을 보였다.

▸ Keywords : 비디오 스트리밍, 주문형 비디오, 피투피, 무임승차, 인센티브

Abstract

Recently the ratio of the Internet traffic generated by video streaming applications including video-on-demand (VOD) is getting higher and higher, while P2P-based naive content distribution has been the main source of the Internet traffic in the past. As a result, the paradigm of cooperatively distributed systems (e.g., P2P) is changing to support streaming applications. Most P2P assisted approaches for video streaming today are based on

•제1저자 : 신규용 •교신저자 : 신규용
•투고일 : 2012. 11. 19, 심사일 : 2012. 12. 9, 게재확정일 : 2013. 1. 20.
* 육군사관학교 전자정보학과(Dept. of Electrical Engineering and Computer Science, Korea Military Academy)
※ 본 논문은 육군사관학교 화랑대연구소 2013년도 연구 활동비를 지원받아 연구되었음

BitTorrent thanks to its simplicity of implementation and easy adaptability. They, however, have immanent vulnerability to free-riding inherited from BitTorrent, which inevitably hurts their performance under limited system resources with free-riding. This paper studies the weakness to free-riding of existing BitTorrent-based video streaming applications and investigates the adaptability of T-Chain (which was originally designed to prevent free-riding in cooperatively distributed systems) to video streaming applications. Our experiment results show that the video streaming approach based on T-Chain outperforms most existing BitTorrent-based ones by 60% on average under limited system resources with free-riding.

▸ Keywords : Video Streaming, Video-on-Demand (VOD), Peer-to-Peer (P2P), Free-riding, Incentive

I. 서 론

과거 인터넷 트래픽 (traffic)의 대부분을 차지하고 있던 피투피 (Peer-to-Peer, P2P) 기반의 단순 콘텐트 분배 어플리케이션들이 이제는 실시간 스트리밍을 포함하는 비디오 스트리밍 어플리케이션들로 빠르게 대체되고 있다. 최근 시스코에서 실시한 연구조사 결과[1]에 의하면 현재 인터넷에서 차지하는 비디오 트래픽은 전체 소비자 트래픽의 약 51%에 달하고 있으며, 이 비율은 점차 증가해 2016년에는 대략 86%에 이를 것으로 전망하고 있다.

주문형 비디오 (video-on-demand, VOD)를 포함하는 비디오 스트리밍의 인기가 높아짐에 따라 이를 지원할 수 있는 다양한 피투피 기반의 어플리케이션들이 제안되고 있다. 피투피 기반의 비디오 스트리밍 어플리케이션들은 시스템에 참여하는 구성원들이 공통의 목표를 달성하기 위해 자신들의 자원을 자발적으로 공유하므로 기존의 클라이언트 서버 (client-server) 방식에 비해 확장성이 뛰어나다는 장점이 있다. 특히 대표적인 피투피 프로토콜인 비토렌트 (BitTorrent[2,3])의 경우 구현이 간편하고 비디오 스트리밍에 대한 적용성이 높이 때문에 현재 대부분의 피투피 기반 비디오 스트리밍 어플리케이션들에 활용되고 있다. 그 대표적인 어플리케이션으로 BiTos[4]와 Give-to-Get (G2G[5])이 있는데, 이들 방식은 기존의 비토렌트에 대한 최소한의 수정만으로 손쉽게 피투피 기반의 주문형 비디오 스트리밍 어플리케이션을 구현하고 있다는 특징이 있다.

하지만 비토렌트의 경우 이타주의 공략 (exploiting altruism[6]), 부정행위 (cheating[7]), 큰 시야 공략 (large view exploit[7-9]), 신분세탁 (whitewashing[10, 11]), 시빌 공격 (Sybil attack[12, 13]), 그리고 공모 (collusion[14]) 등 다양한 무임승차 (free-riding) 기법에 취약하다는 단점이 있다[7]. 따라서 비토렌트를 기반으로 하고 있는 BiTos와 G2G도 별도의 안전장치를 강구하지 않는 한 무임승차에 취약할 수밖에 없다. 주지하는 바와 같이 피투피와 같은 협조적 분산시스템 환경에서 무임승차의 증가는 아무도 시스템으로부터 서비스를 받지 못하는 공유지의 비극 (tragedy of common[15])을 초래할 수 있다. 이런 점들에 비추어 볼 때 BiTos와 G2G는 무임승차들이 증가하거나 (단기간에 많은 사용자가 폭주하는) 플래시 크라우드 (flash crowd)와 같이 시스템 자원이 부족한 환경에서는 만족할 만한 결과를 기대하기 어렵다.

협조적 분산 시스템 환경에서 협동을 강제하고 한정된 자원을 최대로 활용할 수 있도록 개발된 T-Chain[16]은 무임승차에 취약한 비토렌트와 달리 무임승차에 대한 면역력이 좋을 뿐 아니라 그 구현도 비토렌트만큼 간단하다. T-Chain은 기존의 비토렌트 메커니즘에 간단한 대칭키 알고리즘을 추가해 시스템에 공헌 없는 참여자에 대한 서비스를 제한함으로써 효과적으로 무임승차를 방지한다. 또한 직접호혜 (direct reciprocity)만을 중시하는 비토렌트와 달리 T-Chain은 전방 보상기법[17]을 통해 효과적으로 간접호혜 (indirect-reciprocity)를 유도할 수 있기 때문에 자원 활용을 극대화 할 수 있다. 따라서 T-Chain을 기반으로 하는 새로운 주문형 비디오 스트리밍은 기존의 비토렌트 기반의 비디오 스트리밍에 대한 좋은 대안이 될 수 있다.

본 논문은 T-Chain을 주문형 비디오 스트리밍 어플리케이션에 적합하도록 변형시킨 프로토콜인 스트리밍 T-Chain (S-TChain)을 제안하고 평가한다. 주문형 비디오 스트리밍 어플리케이션의 경우 비디오 재생 제한시간 (paly-back deadline), 즉 각 비디오 프레임 (video frame)이 재생되기 이전에 도착해야 한다는 제한사항이 존재한다. 따라서 S-TChain은 T-Chain의 피스 선택 알고리즘인 희소우선 (rarest first) 알고리즘을 비디오 스트리밍을 지원할 수 있도록 순차적인 피스 선택이 가능한 알고리즘으로 수정하였다. 이때 비디오 재생 제한시간이라는 제한사항 때문에 순차적인

피스 선택만을 강조하면 (모든 참여자들이 피스를 순서대로 받기 때문에) 희소 우선선택 알고리즘이 가지는 장점 (즉, 참여자들 간의 상호 관심 유도)이 사라져 전체적인 시스템 성능이 낮아질 수 있다. 따라서 이 둘 사이의 적절한 균형점을 찾는 것이 중요하다. 본 논문은 순차적인 피스 선택 알고리즘과 희소우선 알고리즘 사이의 적절한 균형점을 찾아 효과적인 주문형 비디오 스트리밍 서비스를 구현하는데 그 초점을 맞추었다. 시뮬레이션 결과 S-TChain은 주문형 비디오 스트리밍에 매우 적합하고, 기존의 비토렌트 기반의 방식들에 비해 효과적으로 무임승차를 제한하며, 부족한 자원 환경에서도 잘 동작하는 것을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 먼저 2장 배경지식 및 관련연구에서는 비토렌트, BiTos, G2G, 그리고 T-Chain 에 대해 핵심적인 메커니즘 위주로 간략히 소개한다. 3장에서는 T-Chain을 기반으로 구현된 주문형 비디오 스트리밍 프로토콜인 S-TChain에 대해 설명하고, 4장에서 그 실험결과에 대해 분석하고 토의한다. 마지막으로 5장에서는 결론 및 향후 연구방향에 대해 기술한다.

II. 배경 지식 및 관련 연구

본 장에서는 비토렌트, BiTos, G2G, 그리고 T-Chain 프로토콜의 동작원리에 대해 가장 핵심적인 메커니즘 위주로 간략히 소개한다.

1. 비토렌트 (BitTorrent)

비토렌트[2, 3]는 대규모 파일에 대한 공유 및 분배를 촉진하기 위한 프로토콜이다. 비토렌트에서는 동일한 파일을 다운로드하고자 하는 참여자들이 하나의 스웜 (swarm)을 형성하여 서로 자원을 공유하기 때문에 기존의 클라이언트 서버 구조의 파일 분배 시스템에 비해 확장성이 뛰어나다. 이때 스웜에 참여하는 사용자들은 각자의 역할에 따라 시더 (seeder), 리처 (leecher), 그리고 트래커 (tracker)로 구분된다.

먼저 시더는 다른 참여자들을 위해 파일을 제공하는 참여자이다. 빠른 파일 분배를 위해 시더는 공유하고자 하는 파일을 여러 개의 피스 (piece)로 나누고, 그 파일 및 스웜에 대한 정보를 동호회 사이트와 같은 공공장소에 메타데이터 (meta-data)형태로 공시한다. 이때 공시되는 메타데이터를 **.torrent**라 부르며 하나의 **.torrent**에는 공유되는 파일 피스들의 SHA-1 해시 값 및 트래커에 대한 정보 등이 포함된다. 비토렌트에서 시더는 다른 참여자들에게 피스를 무료로 제공할 뿐 자신이 제공하는 서비스에 대해 어떤 보상도 기대하지 않는 이타적인 참여자이다.

리처는 비토렌트 프로토콜을 이용해 파일을 다운로드하고자 하는 참여자이다. 파일 다운로드를 위해 리처는 먼저 원하는 파일에 대한 **.torrent**를 홈페이지 등을 통해 다운로드한다. 이때 **.torrent**에는 해당 스웜에 대한 트래커 정보가 포함되어 있으므로 그 트래커에 접속해 스웜 안에 있는 다른 참여자들의 정보를 얻는다. 리처는 트래커로부터 얻어진 정보를 이용해 다른 참여자들에게 TCP 연결을 시도하고, 연결이 성공하면 그 두 참여자들은 서로에게 이웃 (neighbor)이 된다. 이웃이 된 참여자들은 자신들이 가지고 있는 피스들을 서로 교환함으로써 필요한 피스들을 다운로드할 수 있다.

비토렌트는 참여자들 사이의 빠른 피스 교환을 위해 Tit-for-Tat (TFT)과 희소우선 (Rarest First) 메커니즘을 구현하고 있다. 이때 TFT는 참여자가 자신이 가지고 있는 피스를 이웃들에게 업로드할 때 지난 10초 동안 자신에게 가장 많은 피스를 업로드한 이웃들을 먼저 선택하는 알고리즘이다. 따라서 이웃들에게 공헌이 많은 참여자는 그 이웃들로부터 보다 많은 피스들을 다운로드할 수 있다. 이런 점에서 TFT는 직접적인 교류가 있는 리처들 사이에서 상호간의 협동을 기반으로 동작하는 직접호혜 (direct reciprocity) 방식의 인센티브 메커니즘이라고 할 수 있다. 희소우선 메커니즘은 리처가 이웃으로부터 다운로드할 피스를 결정할 때 자신의 이웃들이 가지고 있는 피스들 중에서 가장 희소한 피스를 선택하는 알고리즘이다. 이와 같이 희소한 피스를 먼저 다운로드하면 리처들이 서로 다른 피스들을 가질 확률이 높아지므로 상호관심이 증가하고 결과적으로 피스 교환이 용이해진다.

트래커는 스웜에 참여하고 있는 시더 및 리처들에 대한 정보를 가지고 있는 사용자이다. 따라서 파일 다운로드를 위해 스웜에 참여는 리처들은 제일 먼저 트래커에 접속해서 해당 스웜에 참여하고 있는 다른 시더 및 리처들에 대한 정보를 얻는다.

앞서 설명했듯이 비토렌트는 직접호혜 방식의 인센티브인 TFT를 이용해 참여자들 사이의 협동을 유도한다. 하지만 TFT만을 강조하는 경우 스웜에 처음 참여하는 사용자는 파일 피스를 하나도 가지고 있지 않기 때문에 자신의 의지와는 무관하게 시스템을 위해 공헌할 수 없다. 따라서 TFT만을 강조하면 신규 참여자는 파일 피스를 전혀 다운로드할 수가 없다. 이런 상황을 방지하기 위해 비토렌트에서 각 참여자는 자신의 자원 중 약 80% 정도를 TFT로 할당하고, 나머지 20% 정도는 공헌도와 무관하게 임의로 선택된 이웃에게 할당하는데, 이를 Optimistic Unchoking이라 한다. 비토렌트의 Optimistic Unchoking은 신규 참여자 유도를 위해 필수적인 메커니즘이지만 또한 무임승차를 원하는 참여자들에게는 취약점으로 작용한다.

2. BitTos

BiTos[4]는 비토렌트를 기반으로 하는 주문형 비디오 스트리밍 프로토콜이다. BiTos는 비토렌트 프로토콜의 피스 선택 알고리즘인 희소우선 메커니즘을 수정해 주문형 비디오 스트리밍을 구현하였다. 즉, 각 피스의 비디오 재생 제한시간을 고려해 재생이 임박한 피스들은 우선 세트로, 그렇지 않은 피스들은 나머지 세트로 구분하고, 피스를 선택할 때 우선 세트에 있는 피스들에게 보다 높은 우선순위를 두고 선택함으로써 비디오 재생 제한시간을 맞출 수 있도록 하였다. BiTos의 경우 피스 선택을 제외한 나머지 메커니즘은 비토렌트와 동일하며 비토렌트와 마찬가지로 TFT를 통해 리처들 간의 협동을 유도하는 직접호혜 방식의 인센티브를 사용한다.

그림 1. BiTos의 피스 세트 구분
Fig. 1. BitTos overview

그림 1에서 보는 바와 같이 BitTos는 전체 비디오 파일 피스를 비디오 재생지점을 기준으로 우선 세트와 나머지 세트로 구분한다. 이때 우선 세트의 크기는 시스템 파라미터로 통상 전체 비디오 파일 크기의 8% 내외가 적당한 것으로 알려져 있다. BiTos에서는 리처가 자신의 이웃으로부터 피스를 다운로드할 수 있는 권한을 획득했을 때 확률 **p**로 우선 세트에서, 그리고 확률 **1-p**로 나머지 세트에서 피스를 선택한다. 이때 확률 변수 값 **p**에 의해 피스를 선택할 세트가 결정되고 나면, 해당 세트 안에서는 비토렌트의 희소우선 알고리즘이 그대로 적용된다. 단, 최대 희소 피스가 여러 개일 경우는 주문형 비디오 스트리밍인 점을 감안해 재생 제한시간이 가장 가까운 피스를 먼저 선택한다.

BiTos에서 확률 **p**는 비디오 재생 제한시간을 맞추기 위한 순차적인 피스 다운로드와 참여자들 간의 관심 유도를 위한 희소우선 선택간의 균형을 맞추는 매우 중요한 시스템 파라미터가 된다. 저자들의 실험 결과에 의하면 BiTos는 확률 변수 **p**가 0.8 내외일 때 성능이 가장 좋은 것으로 나타났다.

3. Give-to-Get (G2G)

G2G[5]도 BiTos와 마찬가지로 비토렌트를 기반으로 하는 주문형 비디오 스트리밍 프로토콜이다. 하지만 G2G는 비토렌트의 희소우선 메커니즘만 수정한 BiTos와는 달리 상호협동을 유도하기 위한 비토렌트의 인센티브 메커니즘인 TFT에 대한 수정이 추가되었다. 즉, 비토렌트가 자신에게 많은 피스를 업로드한 이웃을 먼저 선택해 피스를 업로드하는 TFT 메커니즘을 사용하는데 비해 G2G는 자신이 업로드한 피스를 다른 이웃들에게 보다 많이 전달한 이웃을 선호한다. 이를 위해 G2G는 자신의 각 이웃 **Pi**에 대해 일정 기간 동안 자신으로부터 받은 피스들을 다른 리처들에게 얼마나 많이 전달했는지 나타내는 지표인 발송순위 (forwarding rank)를 기록해 관리한다. 이 발송순위는 추후 어떤 이웃에게 자신이 가지고 있는 피스를 우선적으로 업로드할 지 판단하는 근거로 사용된다. 비토렌트와 BiTos가 직접호혜를 중시하는 프로토콜이라면 G2G는 간접호혜를 중시하는 프로토콜인 것이다.

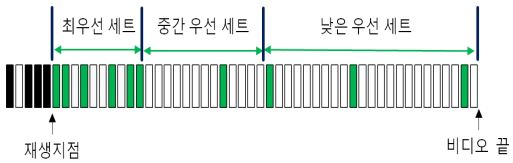


그림 2. G2G의 피스 세트 구분
Fig. 2. G2G overview

희소우선 선택 알고리즘에 대한 수정도 BiTos와는 매우 다르다. 그림 2에서 보는 바와 같이 G2G는 전체 비디오 파일 피스를 최우선 세트, 중간 우선 세트, 그리고 낮은 우선 세트 등 세 그룹으로 나눈다. G2G 논문의 저자들의 경우 10초 정도의 비디오 재생 분량을 최우선 세트로 지정하고, 약 40 초 정도의 재생 분량을 중간 우선 세트로 지정하고 있다. 이때 다운로드할 피스는 다음과 같은 방식으로 결정한다. 먼저 최우선 세트에 다운로드가 가능한 피스가 있으면 최우선적으로 선택한다. 이때 다운로드할 수 있는 피스가 여러 개인 경우, 비디오 재생이 시작된 이후에는 비디오 재생 제한시간이 가장 빠른 피스를 선택하고, 그렇지 않다면 최우선 세트 안의 피스들을 대상으로 희소우선 선택을 실시한다. 만일 최우선 세트에 다운로드할 수 있는 피스가 없으면 중간 우선 세트의 피스들을 대상으로 희소우선 선택을 실시한다. 마지막으로 최우선 세트와 중간우선 세트에 다운로드할 수 있는 피스가 없으면 낮은 우선 세트에서 희소우선 선택을 실시한다. 위와 같이 각 그룹을 순차적으로 점검하는 G2G의 피스 선택 알고리즘은 BiTos의 피스 선택 알고리즘에 비해 순차적인 피스 선택을 하는 성향이 강하다.

4. T-Chain

비토렌트를 기반으로 구현된 T-Chain[16]은 무임승차에 취약한 비토렌트의 단점을 보완하기 위해 대칭키 알고리즘을 추가해 무임승차에 대한 면역력을 향상시켰으며, 직접호혜 방식의 인센티브인 비토렌트의 TFT에 대해 전방 보상기법[17]을 도입해 간접호혜도 가능케 함으로써 시스템 자원 활용을 극대화한 프로토콜이다. T-Chain은 그림 3과 같이 동작한다.

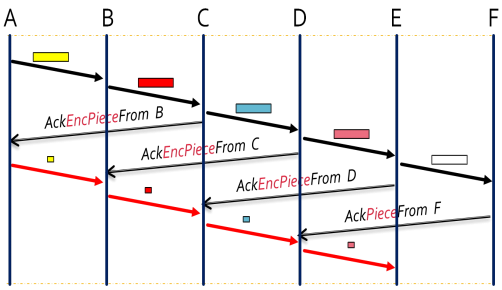


그림 3 T-Chain의 개요
Fig. 3. T-Chain overview

먼저 시더 A는 자신의 이웃들 중 하나인 B를 임의로 선택해 B가 원하는 피스를 보낸다. 이때 T-Chain의 피스 선택 알고리즘은 비토렌트의 피스선택 알고리즘인 희소우선과 동일하다. 즉, B가 A로부터 받을 피스를 선택할 때 자신의 이웃들이 가지고 있는 피스들 중에서 가장 희소한 피스를 우선적으로 선택한다. 피스를 전송할 때 A는 무임승차를 방지하기 위해 대칭키 알고리즘을 이용해 해당 피스를 암호화한다. 암호화된 피스를 보내면서 A는 자신의 이웃들 중 하나인 C를 임의로 선택해 B로 하여금 C에게 동일한 일을 하도록 요구한다. 이때 B가 A의 요구대로 C에게 하나의 (암호화한) 피스를 업로드하면 C는 A에게 수신 확인 메시지를 보내고, 이 수신 확인 메시지를 받은 후 A는 B에게 복호키를 보낸다. 따라서 B는 A의 요구에 따라 다른 참여자 C에게 A가 자신에게 제공했던 서비스와 동일한 서비스를 제공해야만 복호키를 받아 트랜잭션을 완료할 수 있다.

동일한 방법으로 **B**는 **C**에게 (암호화된) 피스를 업로드하면서 (**A**가 그랬던 것처럼) **C**로 하여금 **B** 자신의 이웃들 중 임의로 선택된 **D**에게 동일한 서비스를 제공하도록 요구한다. 이때 만일 **C**에게 **B**가 원하는 피스가 있다면 **D**는 **B** 자신이 될 수 있다. 즉, T-Chain의 각 리처들은 업로드한 피스의 수신자가 자신이 원하는 피스를 가지고 있는 경우에는 자기 자신을 수혜자로 지정함으로써 직접호혜를 구현하고, 그렇지 않은 경우에는 자신의 이웃들 중 하나를 수혜자로 지칭함으로써 간접호혜를 실현한다. 이때 직접호혜는 비토렌트의 TFT 메커니즘과 동일한 역할을 하며 간접호혜는 전방 보상기법[17]과 동일한 효과를 가진다.

위와 같은 일련의 트랜잭션들은 보답사슬 (reciprocation chain)을 형성하면서 정해진 한계 없이 계속 진행된다. 하지만 경우에 따라 그림 3에서 보는 바와 같이 **E**가 **D**의 요구에 따라 **F**에게 (암호화한) 피스를 보내면서 수혜자를 결정할 때 자기 자신을 포함해 자신의 이웃 들 중 아무도 **F**로부터 원하는 피스가 없는 경우가 발생한다. 이런 경우 **E**는 **F**에게 암호화되지 않은 피스를 보냄으로써 해당 보답사슬을 종료한다.

T-Chain에서 시더는 리처들이 보다 많은 보답사슬에 참여할 수 있도록 여건이 허용하는 범위에서 많은 보답사슬들을 만들고, 경우에 따라 유휴 자원을 가지고 있는 리처들로 하여금 시더와 동일한 방식으로 보답사슬들을 만들도록 유도함으로써 시스템 자원 활용을 극대화한다. T-Chain의 보답사슬에 참여하는 각 시더 및 리처들에 대한 인센티브, 신규 참여자 유도 방식, 그리고 무임승차 방지 성능에 대해서는 원 논문[16]을 참조할 수 있다.

III. 스트리밍 T-Chain (S-TChain)

본 장에서는 앞서 Ⅱ-4장에서 설명된 T-Chain에 대한 수정을 통해 주문형 비디오 스트리밍 서비스를 지원하도록 디자인된 스트리밍 T-Chain (S-TChain)에 대해 설명한다.

기본적으로 S-TChain은 T-Chain과 동일하게 동작한다. 단, 주문형 비디오 스트리밍에서 가장 중요한 요소인 비디오 재생 제한시간을 고려해 피스 선택 알고리즘에 대한 수정이 이루어졌다. 즉, 가능한 많은 비디오 피스들이 비디오 재생 제한시간 안에 도착하도록 하기위해 단순히 희소성만을 고려하는 비토렌트의 희소우선 알고리즘을 비디오 재생 제한시간을 고려할 수 있는 새로운 알고리즘으로 대체하였다.

1. S-TChain의 피스 세트 구분

우선 주문형 비디오 스트리밍을 위한 S-TChain에서 각 리처 **P**는 자신이 다운로드할 전체 비디오 파일 피스들을 그림 4와 같이 4개의 세트로 구분해 관리한다. 이때 각 세트의 구분기준은 현재의 비디오 재생지점이 된다. 즉, 현재 비디오 재생지점 이후의 일정 범위를 최우선 세트로, 그 다음 일정 범위를 중간 우선 세트로, 그리고 나머지 부분을 낮은 우선 세트로 지정하되, 현재 비디오 재생지점 이전에 받지 못한 피

스들이 있는 경우에는 그 피스들을 나머지 세트로 구분해서 관리한다.

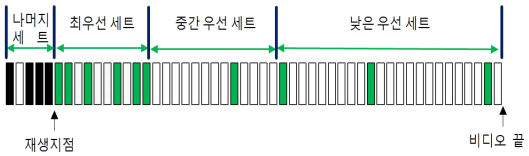


그림 4. S-TChain의 피스 세트 구분
Fig. 4. S-TChain overview

S-TChain에서 각 피스 세트의 역할과 각 피스 세트 안에서 피스를 선택하는 기준은 다음과 같다.

- 최우선 세트 (S^P_{High}) : 최우선 세트는 아직 다운로드 되지 않는 피스들 중에서 리처 **P**의 현재 비디오 재생지점을 기준으로 비디오 재생 제한시간이 거의 임박한 피스들이 포함되어 있는 세트이다. 이 세트가 선택되는 경우, 만일 리처 **P**의 비디오 재생이 시작되었으면 비디오 재생 제한시간이 가장 빠른 피스를 우선적으로 선택하고, 그렇지 않은 경우에는 비토렌트의 희소우선 알고리즘에 따라 피스를 선택한다.
- 중간 우선 세트 (S^P_{Mid}) : 이 세트에 포함되어 있는 피스들은 최우선 세트에 포함되어 있는 피스들만큼 비디오 재생 제한시간이 임박하지는 않았지만 비교적 가까운 미래에 재생될 피스들이다. 이 세트가 선택되는 경우 리처 P는 비디오 재생 시작과 관계없이 비토렌트의 희소우선 알고리즘에 의해 피스를 선택한다.
- 낮은 우선 세트 (S^P_{Low}) : 이 세트에 포함되어 있는 피스들은 비디오 재생 제한시간에 관한한 상대적으로 중요도가 떨어지는 피스들이지만 리처들 간의 상호관심을 유도하기 위해 다운로드할 필요가 있는 피스들이다. 즉, 최우선 세트와 중간 우선 세트가 끊김 없는 비디오 재생을 위한 피스들의 집합이라면 낮은 우선 세트는 향후 참여자들 간의 활발한 피스 교환을 보장하기 위한 피스들의 집합이다.
- 나머지 세트 ($S^P_{Remaining}$) : 이 세트에 포함되어 있는 피스들은 비디오 재생 제한시간이 이미 지난 피스들 중에서 아직 다운로드가 되지 않은 피스들이다. 주문형 비디오 스트리밍의 특성상 비디오 재생 제한시간이 지난 피스들은 (현재의 스트리밍 서비스를 위해서는) 필요하지 않지만 향후 해당 비디오를 다시 보고 싶을 때를 대비해 다운로드할 필요가 있다. 하지만 현재 스트리밍 서비스에서는 중요도가 가장 떨어지므로 앞선 3개 세트에서 받을 수 있는 피스가 없는 경우에만 다운로드 되는 피스들의 집합이다.

S-TChain에서 비디오 피스들의 세트를 구분하는 방식과 크기는 나머지 세트를 제외하고는 G2G와 유사하다. 즉, S-TChain은 약 10초 재생 분량을 최우선 세트로, 40초 정도의 재생 분량을 중간 우선 세트로, 그리고 나머지 부분을 낮은 우선 세트로 지정하되, G2G와 달리 비디오 재생 제한시간을 지난 피스들도 별도의 세트로 지정해 다운로드한다. 이와 같이 나머지 세트를 지정하는 것은 두 가지 장점을 가진다. 첫째, 각 리처들이 비디오 재생 제한시간이 지난 피스들을 다운로드함으로써 다음에 참여하는 리처들을 보다 원활하게 서비스할 수 있고, 둘째, 향후 다운받은 비디오를 재생 하고자 할 때 처음부터 끝까지 끊김 없이 재생할 수 있다.

2. S-TChain의 피스 선택 알고리즘

S-TChain과 G2G는 피스 세트를 구분하는 측면에서는 유사점이 있지만 피스를 선택하는 알고리즘은 서로 다르다. 즉, 각 세트를 순차적으로 선택하는 G2G와 달리 T-Chain은 BiTos처럼 비디오 재생 제한시간과 희소우선 선택간의 균형을 맞추기 위해 일정한 확률을 가지고 각 세트를 선택한다. 이를 위해 S-TChain은 세트 선택 확률 변수인 **p**와 확률 범위 변수인 α 및 β ($\rangle$ α)를 정의하고, 그 변수들의 값에 따라 세트를 선택하는 순서를 달리한다. 알고리즘 1은 S-TChain 프로토콜 하에서 각 리처 P가 피스를 선택하는 방법을 보여준다.

알고리즘 1에서 보듯이 리처 **P**가 다른 리처로부터 피스를 다운로드할 권한을 획득하면 확률 **p**에 따라 자기 자신의 피스 세트들 중에서 어떤 세트에서 우선적으로 피스를 다운로드할 지를 결정한다. 이때 중요한 역할을 하는 것이 확률 범위 변수인 α와 β이다. 즉, α가 크다면 리처 **P**는 높은 확률로 최우선 세트에서 피스를 먼저 다운로드하려고 할 것이며, 그 다음으로 중간 우선 세트, 낮은 우선 세트, 그리고 나머지 세트를 순차적으로 검색해 다운로드 할 수 있는 피스가 있는 지 검색하려 할 것이다. 마찬가지로 또 다른 확률 범위 변수인 β가 클수록 리처 **P**는 높은 확률로 중간 우선 세트에서 피스를 먼저 다운로드 하려고 할 것이며, 그 다음으로 최우선 세트, 낮은 우선 세트, 그리고 나머지 세트를 순차적으로 검색해 다운로드 할 수 있는 피스가 있는 지 검색하려 할 것이다. 따라서 확률범위 변수 α와 β는 시스템 성능을 조율하는 파라미터가 된다[1]. 또한 알고리즘 1에서 보듯이 리처 **P**는 최

우선 세트, 중간 우선 세트, 그리고 낮은 우선 세트에 선택할 수 있는 피스가 없는 경우에 한해서 나머지 세트에서 비디오 재생 제한시간이 이미 지난 피스를 다운로드하려고 시도한다.

Input: $S^{\mathbb{P}}_{High}, \ldots S^{\mathbb{P}}_{Remaining}, \alpha, \beta$
Output: Selected piece i

$p \leftarrow \frac{rand()}{RAND-MAX}$;
if $p \leq \alpha$ **then**
  if $i \in S^{\mathbb{P}}_{High}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{Mid}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{Low}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{Remaining}$ **then return** i
else if $p \leq \beta$ **then**
  if $i \in S^{\mathbb{P}}_{Mid}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{High}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{Low}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{Remaining}$ **then return** i
else
  if $i \in S^{\mathbb{P}}_{Low}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{High}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{Mid}$ **then return** i
  else if $i \in S^{\mathbb{P}}_{Remaining}$ **then return** i
return *notFound*

알고리즘 1. T-Chain의 피스 선택
Algorithm 1. Piece selection of S-TChain

Ⅳ. 성능분석

이번 장에서는 실험을 위한 설정들 (가정 사항, 시뮬레이션 시나리오, 성능분석 척도)에 대해 설명하고, 실험결과에 대한 분석 및 토의를 실시한다. 이때 S-TChain에 대한 성능 평가는 비토렌트[2, 3], BiTos[4], 그리고 G2G[5]와의 비교를 통해 이루어졌으며, 각 프로토콜에 대한 자세한 사항은 Ⅱ장 및 저자들의 원 논문을 참고할 수 있다.

1. 실험 설정 (simulation setup)

실험을 위해 우리는 TBeT[7]에서 사용되었던 이벤트 기반 비토렌트 시뮬레이터를 BiTos와 G2G에 맞게 수정하였다. 또한 S-TChain을 위해서는 기존의 T-Chain[16] 시뮬레이터에서 Ⅲ장에서 언급한 피스 선택 알고리즘 부분만을 수정해 비디오 스트리밍 프로토콜 시뮬레이터로 변환하였다.

1.1 가정 사항

각 실험에서 참여자들은 다음과 같이 행동한다. 먼저 주문형 비디오 서비스 공급자인 시더는 실험 기간 내내 스웜 안에 남아서 다른 리처들을 서비스한다. 리처들은 스웜에 참여함과 동시에 피스들을 다운로드 함으로써 비디오 스트리밍 서비스에 참여한다. 이때 리처들은 원활한 비디오 스트리밍을 위해 $\mathbf{T_w}$만큼의 선행 버퍼링 (pre-buffering) 시간동안 기다린 뒤 비디오 재생을 시작한다.

리처들이 스웜에 도착하고 떠나는 절차는 각각 두 가지 방식으로 모델링 되었다. 먼저 리처들의 도착과 관련해서는 (1) 시스템에 참여하는 모든 리처들이 10초 이내에 도착하는 플래시 크라우드 (flash crowd) 상황과 (2) 일반적인 비토렌트 스웜에서 리처들이 도착하는 패턴을 이용한 실제 트레이스 (real trace) 상황을 고려하였다. 이때 플래시 크라우드는 해당 주문형 비디오 스트리밍 서비스에 대한 관심이 집중되어 비디오 스트리밍 서비스를 시작하자마자 동시에 참여자들이 폭주하는 상황을 가정한 것이며, 실제 트레이스의 경우 참여자들의 꾸준한 관심이 있는 일반적인 비디오 스트리밍 서비스를 가정한 것이다. 시스템 측면에서는 (2)의 경우보다 (1)의 경우가 훨씬 많은 시스템 자원을 필요로 한다. 리얼 트레이스를 묘사하기 위해 우리는 RedHat 9 Torrent Tracker Trace[18] 분석을 통해 해당 비토렌트 스웜에 참여했던 리처들의 도착 패턴을 획득했다. 리처들이 스웜을 떠나는 상황에 대해서는 (1) 리처들이 스트리밍의 종료와 상관없이 모든 비디오 피스들을 다운로드한 이후 바로 스웜을 떠나는 상황과 (2) 리처들이 모든 비디오 피스들을 다운로드를 완료한 이후에도 자신들의 스트리밍이 끝날 때까지 남아서 계속 서비스를 제공하는 두 가지 상황을 고려하였다. 시스템 측면에서는 (1)의 경우보다 (2)의 경우에 보다 많은 자원을 보유할 수 있기 때문에 부담이 적다.

무임승차가 이루어지는 경우 별도의 언급이 없는 한 그 비율은 전체 리처의 20%[4, 5]이며, 무임승차자들은 자신이 참여하는 시스템에 효과적인 모든 무임승차 기법들을 알고 적용할 수 있는 것으로 가정하였다. 따라서 비토렌트와 BiTos에서는 이타주의 공략, 부정행위, 큰 시야 공략 등이, G2G에서는 이타주의 공략, 큰 시야 공략, 시빌 공격 및 공모 등이, 그리고 S-TChain에서는 시빌 공격 및 공모 등이 가능한 것으로 가정하였다[7].

시더의 업로드 속도는 2,000Kbps이고, 리처들의 업로드

1) 4장 시뮬레이션 결과에 의하면 α와 β의 값이 각각 0.5와 0.8일 때 대부분의 시나리오에서 S-TChain의 성능이 가장 좋게 나타났다.

속도는 500~1,000Kbps 사이에서 균일한 분포로 임의로 선택하였으며, 리처들의 다운로드 대역폭에는 제한이 없다고 가정하였다. 스웜에서 공유되는 파일은 500Kbps로 인코딩된 10분 분량의 비디오 파일로서 총 2,400개의 피스로 분할되었으며, 기본 선행 버퍼링 시간 T_w는 60초로 설정하였다[4, 5]. 성능분석 시 플래시 크라우드의 경우에는 10초 이내에 참여하는 400 리처들의 평균 성능을 고려하였으며, 실제 트레이스의 경우는 성공적으로 비디오 스트리밍을 완료한 1,000명의 리처들에 대해 데이터를 수집한 뒤 (시스템이 안정화된 이후의 결과를 반영하기 위해) 마지막 500 리처들의 평균 성능을 고려하였다. 별도의 언급이 없는 경우 결과상의 모든 데이터는 서로 다른 난수 시드를 바탕으로 수행된 5번의 시뮬레이션 결과에 대한 평균값을 의미한다.

1.2 시뮬레이션 시나리오

성능분석을 위한 시뮬레이션 시나리오는 앞서 Ⅳ-1.1장에서 설명한 것처럼 리처들이 시스템에 도착하고 떠나는 방식에 따라 3가지로 구성되었다. 시나리오 구성에 있어 가장 중점적으로 고려된 사항은 시스템 전체의 자원 총량이다.

- CASE 1 (풍부한 자원 환경) : 이 환경은 실제 트레이스 모델에서 각 리처들이 공유되는 모든 비디오 피스들을 다운로드한 이후에 자신의 비디오 재생이 끝날 때까지 시스템에 남아 다른 리처들을 위해 계속 서비스를 제공하는 경우이다. 리처들의 업로드 속도가 500~1,000Kbps로 스트리밍 속도인 500Kbps보다 빠르다는 것을 감안할 때 대부분의 리처들은 다운로드를 완료한 이후에도 상당시간 시스템에 남아 시더로서의 역할을 수행할 것이다. 따라서 이런 환경에서는 시간이 지날수록 시스템 자원을 풍부해질 것이다. 하지만 다운로드를 완료한 리처들의 경우 시스템을 떠나서도 비디오 재생이 가능하기 때문에 그들을 시스템에 머물도록 강제할 수단이 적어 이런 환경은 현실적으로 기대하기 힘들다.

- CASE 2 (보통의 자원 환경) : 이 환경은 실제 트레이스 모델에서 각 리처들이 공유되는 모든 비디오 피스들을 다운로드하고 나면 즉시 시스템을 떠나는 경우이다. 즉, 각 리처들은 자신들이 다운로드를 받고 있는 동안에만 시스템에 공헌하고, 그 이후에는 더 이상 시스템을 위해 공헌하지 않는다. 대부분의 스웜에 한꺼번에 많은 사용자가 시스템에 참여하기보다는 일정한 수의 사용자가 꾸준히 참여하고 있다는 점과 그들 대부분이 이기적 (selfish)이라는 점을 감안하다면 이런 환경이 가장 현실적인 환경이라 볼 수 있다.

- CASE 3 (극한의 자원 환경) : 이 환경은 플래시 크라우드 환경에서 각 리처들이 공유되는 모든 비디오 피스들을 다운로드한 이후에 바로 시스템을 떠나는 경우이다. 한꺼번에 많은 리처들이 몰리는 사용자 폭주상황을 가정한 플래시 크라우드의 특성상 순간적으로 많은 시스템 자원이 필요한데 다운로드를 완료한 리처들이 더 이상 시스템에 공헌하지 않고 떠나버리므로 자원 부족현상이 심해질 수밖에 없다. 무임승차가 추가되는 경우를 제외한다면 이런 환경은 대부분의 주문형 비디오 스트리밍 어플리케이션들이 감당해야 하는 최악의 상황이라 볼 수 있다.

1.3 성능분석 척도

본 논문에서는 각 시스템의 상대적인 성능을 평가하기 위한 척도로서 지속성 지수 (Continuity Index)와 업로드 대역폭 사용률 (Upload Bandwidth Utilization)을 고려하였다.

먼저 (1) 지속성 지수는 전체 비디오 피스 개수에 대해 비디오 재생 제한시간 이전에 도착한 피스 개수가 차지하는 비율로 정의된다. 주문형 스트리밍 비디오 서비스의 특성상 비디오 재생 제한시간 이후에 도착한 피스들은 재생될 수 없으므로 비디오 재생 제한시간 이전에 다운로드 되는 피스의 개수가 많을수록, 즉 지속성 지수가 클수록 스트리밍 품질이 좋다는 것을 의미한다. 따라서 지속성 지수가 높을수록 효과적인 시스템이라 할 수 있다. 다음으로 (2) 업로드 대역폭 사용률은 (시스템 순응적인) 각 리처의 최대 업로드 대역폭 대비 그 리처가 시스템에 도착한 후부터 다운로드를 완료하는 시점까지 실제로 사용한 평균 업로드 대역폭의 비율로 정의된다. 어떤 시스템에서 리처들의 업로드 대역폭 사용률이 높다는 것은 그 시스템이 리처들로 하여금 서로 협동할 수 있는 여건을 잘 만들어 주고 있다는 의미이다. 따라서 리처들의 업로드 대역폭 사용률이 높을수록 효과적인 시스템이라 할 수 있다.

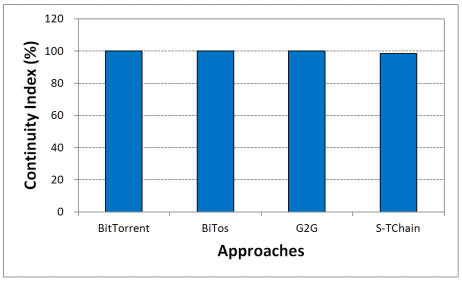
2. 실험결과 분석 및 토의

본 논문에서는 앞서 Ⅳ-1.2장에서 제시한 3가지 시나리오에 대해 각각 무임승차가 이루어지는 경우와 그렇지 않은 경우로 구분해 비토렌트, BiTos, G2G, 그리고 S-TChain의 성능을 비교 및 분석한다.

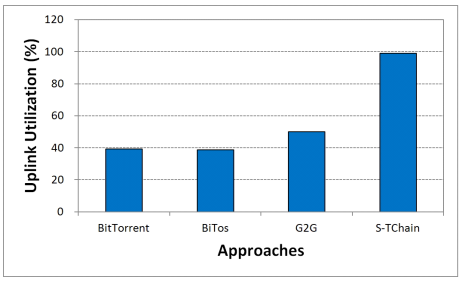
2.1 CASE 1 (시스템 자원이 충분할 경우)

이번 실험에서는 시스템 자원이 가장 충분한 경우 (즉, 실제 트레이스 모델에서 리처들이 다운로드를 완료한 이후에도 자신의 스트리밍이 끝날 때까지 시스템에 남아 다른 리처들을

서비스 하는 경우)에 대해 비토렌트, BiTos, G2G, 그리고 S-TChain 내 시스템 순응적인 리처들의 평균 지속성 지수 및 평균 업로드 대역폭 사용률에 대해 알아본다. 이번 실험에서 무임승차는 고려하지 않았다.



(a) 평균 지속성 지수



(b) 평균 업로드 대역폭 사용률

그림 5. 실제 트레이스에서 (무임승차가 없이) 리처들이 다운로드를 완료한 이후 시스템에 남아 계속 서비스를 제공할 때 시스템 순응적인 리처들의 평균 지속성 지수 (a)와 평균 업로드 대역폭 사용률 (b)
Fig. 5. The average continuity index and upload bandwidth utilization of compliant leechers when leechers stay in the system after their download completion.

그림 5(a)는 시스템 내에 자원이 충분할 때 비토렌트, BiTos, G2G, 그리고 S-TChain 시스템 내의 시스템 순응적인 리처의 평균 지속성 지수를 보여준다. 결과에서 보듯이 시스템 자원이 충분한 경우에는 모든 시스템 하에서 리처들의 지속성 지수가 거의 100%에 가깝다. 특이할만한 점은 비디오 스트리밍에 대한 고려가 전혀 없는 비토렌트도 거의 100%에 가까운 성능을 보이고 있다는 점이다. 이는 비토렌트 기반 주문형 비디오 스트리밍 서비스의 활용 가능성을 보여주는 결과이다.

그림 5(b)는 동일한 환경에서 각 시스템 안의 리처들의 평균 업로드 대역폭 사용률을 보여준다. 결과에서 보듯이 비토렌트, BiTos, 그리고 G2G 시스템 내에서는 리처들의 평균 업로드 대역폭 사용률이 50% 이하로 매우 낮음을 볼 수 있다. 이는 비토렌트, BiTos, 그리고 G2G 안에서는 리처들 사이의 상호협동에 의해 서비스가 이루어지기 보다는 다운로드를 마친 후 스트리밍이 끝날 때까지 시스템에 남아서 서비스를 제공하는 이타적인 리처들에 의해서 서비스가 이루어지고 있음을 의미한다. 하지만 결과에서 보듯이 S-TChain의 경우는 리처들의 평균 업로드 대역폭이 거의 100%에 가깝다. 이는 S-TChain 안에서는 대부분의 서비스가 (이타적인 리처들에 의해서가 아니라) 리처들 상호간의 협동에 의해 이루어지고 있음을 의미한다. 결론적으로 시스템 자원이 충분한 경우 대부분의 비토렌트 기반 비디오 스트리밍 어플리케이션들이 효과적으로 동작하지만 특히 S-TChain 하에서 참여자들의 자원공유가 가장 활발하다.

그림 6은 그림 5와 동일한 환경에서 전체 리처의 20%가 무임승차자일 때의 시스템 순응적인 리처들과 무임승차자들의 평균 지속성 지수를 보여준다. 결과에서 보듯이 시스템 순응적인 리처들이 다운로드를 완료하고 남아서 계속 서비스를 하는 경우 20%정도의 무임승차자들이 존재하더라도 모든 시스템들이 매우 만족할 만한 성능을 보이고 있음을 알 수 있다. 하지만 결과에서 주목해야 할 점은 비토렌트, BiTos, 그리고 G2G 하에서는 무임승차자들의 성능도 시스템 순응적인 리처들의 성능만큼이나 좋다는 점이다. 이런 상황에서는 무임승차자의 비율이 점점 높아질 수밖에 없다. 왜냐하면 시스템에 대한 공헌이 전혀 없어도 공헌이 많은 다른 리처들과 거의 동일한 서비스를 받을 수 있다면 이기적인 리처들은 무임승차를 선호할 것이기 때문이다. 이에 반해 S-TChain 하에서 무임승차자들의 성능은 시스템 순응적인 사용자들의 성능의 10%에 불과하다. 따라서 S-TChain은 무임승차에 대한 욕구를 억제할 수 있을 것이다.

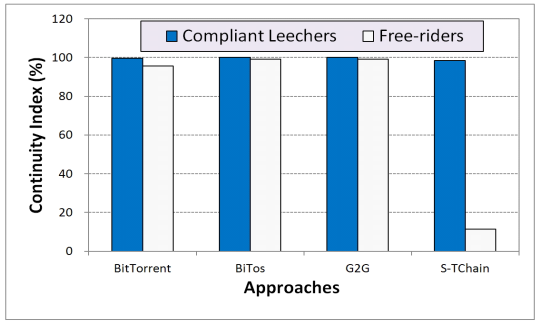


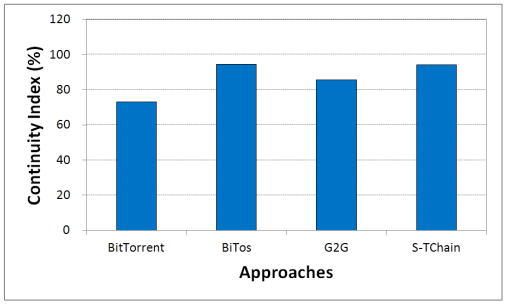
그림 6. 그림 5와 동일한 환경에서 전체 리처의 20%가 무임승차자인 경우에 대한 시스템 순응적인 리처들과 무임승차자들의 평균 지속성 지수
Fig. 6. The average continuity index of compliant leechers and free-riders with 20% free-riding under the same condition as in fig. 5.

2.2 CASE 2 (시스템 자원이 보통일 경우)

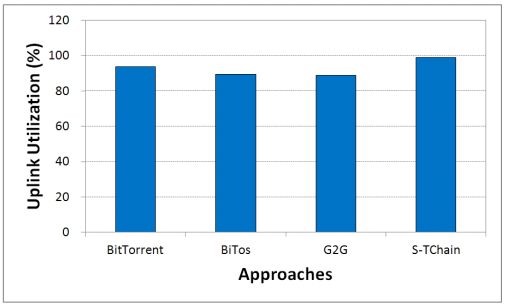
이번 실험에서는 실제 트레이스에서 리처들이 모든 비디

오 피스들을 다운로드한 이후 (스트리밍이 끝나기 전에) 바로 시스템을 떠나는 경우에 대한 결과에 대해 알아본다. 피투피 시스템을 포함한 대부분의 협조적 분산 환경에 참여하는 사용자들이 이기적이라는 점을 감안한다면 이러한 케이스가 가장 일반적인 환경이라 볼 수 있다.

그림 7(a)는 비토렌트, BiTos, G2G, 그리고 S-TChain 하에서 리처들이 다운로드를 완료한 이후 바로 시스템을 떠날 때 시스템 순응적인 리처들의 평균 지속성 지수를 보여주고 있다. 결과에서 보듯이 리처들이 다운로드를 완료하고 시스템을 떠나는 경우에는 그렇지 않은 경우 (그림 5(a))에 비해 시스템 순응적인 리처들의 평균 지속성 지수가 많이 낮아졌음을 볼 수 있다. 이는 리처들이 다운로드를 완료하자마자 시스템을 떠나기 때문에 그림 5의 경우보다 시스템 자원이 상대적으로 부족해졌기 때문이다. 특히 비토렌트의 성능이 가장 좋지 못함을 알 수 있는데 이는 비토렌트가 스트리밍을 고려해 디자인된 시스템이 아니기 때문이다.



(a) 평균 지속성 지수



(b) 평균 업로드 대역폭 사용률

그림 7. 실제 트레이스에서 (무임승차 없이) 리처들이 다운로드를 완료한 이후 바로 시스템을 떠날 때 시스템 순응적인 리처들의 평균 지속성 지수 (a)와 평균 업로드 대역폭 사용률 (b)
Fig. 7. The average continuity index and upload bandwidth utilization of compliant leechers when compliant leechers leave the system upon their download completion.

그림 7(b)는 각 시스템에서 시스템 순응적인 리처들의 평균 대역폭 사용률을 보여주고 있는데, 비토렌트, BiTos, G2G 시스템에서 시스템 순응적인 리처들의 평균 업로드 대역폭 사용률이 그림 5(b)에 비해 크게 증가하고 있음을 알 수 있다. 이는 다운로드를 완료한 리처들이 시스템을 떠남으로써 부족해진 시스템 자원을 보충하기 위해 남아있는 리처들이 보다 많은 자원을 공유하기 때문이다. 하지만 다른 시스템들과는 달리 S-TChain의 경우 평균 지속성 지수와 평균 업로드 대역폭 사용률 측면에서 그림 5의 결과와 별로 차이가 없다. 그 이유는 S-TChain의 경우 시스템의 잉여 자원이 아니라 리처들 간의 적극적인 자원공유에 의해 동작하는 방식이기 때문이다.

그림 8은 그림 7과 동일한 환경에서 전체 리처들 중 20%가 무임승차자라고 가정하였을 경우 시스템 순응적인 리처들과 무임승차자들의 평균 지속성 지수를 보여준다. 결과에서 보듯이 비토렌트, BiTos, G2G의 경우 무임승차가 발생하면 그렇지 않은 경우에 비해 시스템에 순응적인 리처들의 성능이 평균 10% 가까이 떨어짐을 볼 수 있다. 그러나 S-TChain의 경우는 무임승차자의 참여가 시스템 순응적인 사용자들의 성능에 미치는 영향이 거의 없다. 이는 S-TChain의 경우 대칭키 알고리즘을 통해 효과적으로 무임승차를 제한하는 T-Chain의 특성을 그대로 이어받았기 때문이다. 또한 결과에서 보듯이 비토렌트, BiTos, G2G 시스템 하에서는 무임승차자들의 성능이 시스템 순응적인 리처들의 성능의 약 80%에 육박하지만 S-TChain 하에서는 무임승차자들은 비디오 스트리밍이 사실상 거의 불가능함을 볼 수 있다. 따라서 무임승차에 대한 효과적인 제한과 비디오 스트리밍 품질 측면에서 S-TChain이 다른 시스템들보다 훨씬 우수하다고 결론지을 수 있다.

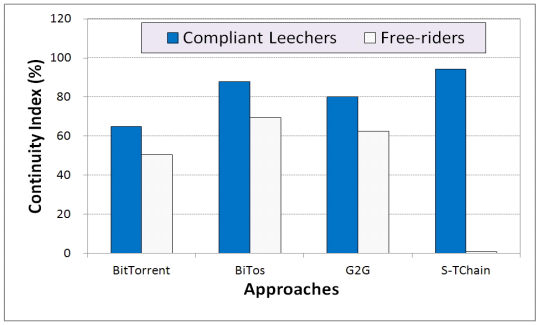


그림 8. 그림 7과 동일한 환경에서 전체 리처의 20%가 무임승차자인 경우에 대한 시스템 순응적인 리처들과 무임승차자들의 평균 지속성 지수
Fig. 8. The average continuity index of compliant leechers and free-riders with 20% free-riding under the same condition as in fig. 7.

2.3 CASE 3 (시스템 자원이 극히 제한되는 경우)

이번 실험에서는 한꺼번에 많은 사용자가 시스템에 참여하

는 사용자 폭주 상황을 가정한 플래시 크라우드 환경에서 리처들이 모든 비디오 피스를 다운로드한 후 바로 시스템을 떠나는 환경에서 각 시스템의 성능을 살펴본다. 플래시 크라우드 상황에서는 제한된 자원을 보유하고 있는 시더가 많은 리처들에게 서비스를 제공해야 하기 때문에 리처들 간의 적극적인 자원공유가 이루어지지 않으면 제대로 동작할 수 없다. 이런 점에서 이런 환경은 주문형 비디오 스트리밍 어플리케이션들이 다루어야 하는 최악의 상황이라고 볼 수 있으며 이런 환경에서 잘 동작하는 시스템이 좋은 시스템이라 할 수 있다. 따라서 플래시 크라우드 환경에서 각 시스템의 성능을 알아보기 위한 실험이 실시되었다.

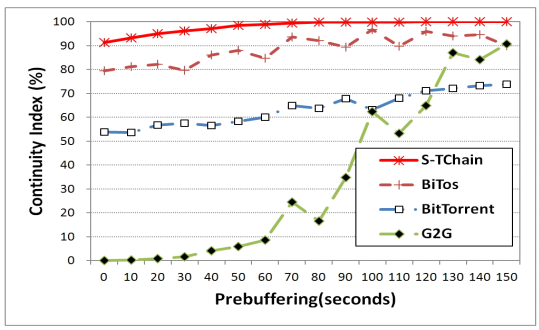
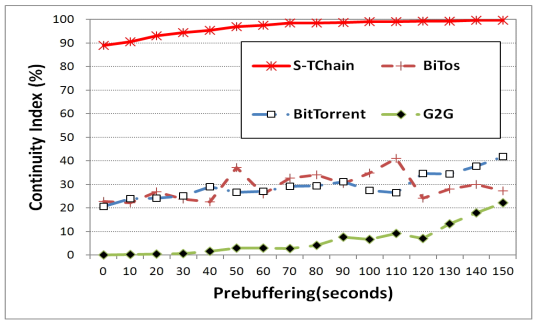
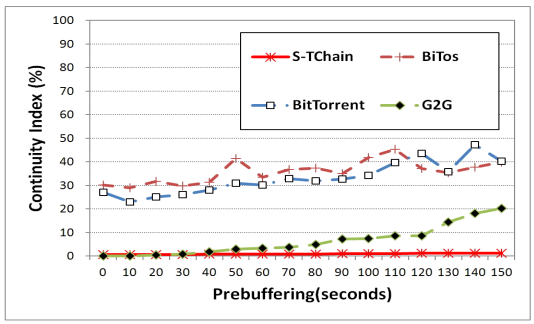


그림 9. 플래시 크라우드 환경에서 선행 버퍼링 시간(Tw)에 따른 비토렌트, BiTos, G2G, S-TChain 내의 시스템 순응적인 리처들의 평균 연속성 지수
Fig. 9. The average continuity index of compliant leechers in BitTorrent, BiTos, G2G, and S-TChain as a funtion of prebuffering time under in a flash crowd.

그림 9는 무임승차가 없는 경우 선행 버퍼링의 크기에 따른 시스템 순응적인 리처들의 평균 연속성 지수를 보여준다. 결과에서 보듯이 비토렌트와 BiTos의 경우 선행 버퍼링 시간에 비례해서 연속성 지수가 완만하게 증가하고 있으며, 150초의 선행 버퍼링을 기준으로 봤을 때 최대 성능은 각각 74%와 96%였다. G2G의 경우는 다른 시스템보다 선행 버퍼링의 시간에 민감한 것으로 나타났는데, 이는 순차적인 피스 선택을 선호하는 G2G의 특성에서 기인한 것이다. 즉, 비토렌트와 BiTos의 경우는 희소우선에 기인한 피스선택을 하기 때문에 리처들 사이에서 피스의 다양성이 증가해 시스템 자원이 부족한 환경에서도 리처들 간의 원만한 협조를 이끌어내 짧은 선행 버퍼링으로도 어느 정도의 성능을 보이는 반면, 순차적인 피스 선택을 선호하는 G2G는 시스템 자원이 부족한 환경에서는 리처들 사이에서 피스의 다양성이 부족해 리처들 간의 협조를 유도하는데 상대적으로 많은 시간이 걸리기 때문이다. 이에 비해 S-TChain은 다른 시스템들에 비해 짧은 선행 버퍼링만으로도 거의 100%의 성능을 내고 있다. 그 이유는 두 가지로 볼 수 있다. 첫 번째로 알고리즘 1에서 보듯이 S-TChain은 비디오 재생 제한시간을 보장하기 위한 순차적인 피스선택과 리처들 간의 피스 다양성을 보장하는 희소우선 사이의 균형을 맞추고 있다. 두 번째로 S-TChain은 대칭키 알고리즘에 기반을 둔 전방 보상기법을 통해 직접호혜와 간접호혜를 동시에 구현할 수 있기 때문에 리처들 간의 활발한 자원공유를 유도할 수 있다. 이런 두 가지 측면에서 얻어지는 시너지 효과는 짧은 선행버퍼링(약 40초)만으로도 최대성능을 발휘할 수 있도록 한다. 결론적으로 이번 실험을 통해 우리는 시스템 자원이 부족한 환경에서 S-TChain이 다른 시스템들에 비해 훨씬 좋은 성능을 나타내고 있음을 확인할 수 있었다.



(a) 시스템 순응적인 리처들의 연속성 지수



(b) 무임승차자들의 연속성 지수

그림 10. 그림 9와 동일한 환경에서 전체 리처의 20%를 무임승차자라 가정했을 때 선행 버퍼링 시간에 따른 시스템 순응적인 리처들의 평균 연속성 지수 (a)와 무임승차자들의 평균 연속성 지수 (b)
Fig. 10. The average continuity index of compliant leechers (a) and free-riders (b) in each system with 20% free-riding under the same condition as in fig. 9.

그림 10은 그림 9와 동일한 환경에서 전체 리처의 20%를 무임승차자로 가정하였을 때 선행 버퍼링 시간에 따른 시스템 순응적인 리처들과 무임승차자들의 평균 연속성 지수를 보여준다. 결과에서 보듯이 무임승차에 상대적으로 취약한 비토렌트와 BiTos, G2G의 경우 무임승차가 발생했을 때 (그렇지 않은 경우에 비해) 평균 지속성 지수가 40% 이하로 급격히 저하

되고 있음을 볼 수 있다. 이는 무임승차자들이 시스템 순응적인 리처들의 자원을 소모함으로써 시스템 자원을 더욱 부족하게 만들기 때문이다. 하지만 무임승차에 대한 면역력이 높은 T-Chain을 기반으로 한 S-TChain의 경우는 무임승차에 거의 영향을 받지 않고 있다. 따라서 시스템 자원이 부족할수록 S-TChain이 다른 시스템들에 비해 훨씬 좋은 성능을 보인다고 결론지을 수 있다.

V. 결론 및 향후 연구방향

본 논문은 자원이 부족하거나 충분하지 않은 환경에서 주문형 비디오 스트리밍 서비스를 지원하기 위한 프로토콜인 S-TChain을 제안하고 평가했다. S-TChain은 협조적 분산 환경에서 상호협동을 강제하기 위해 개발된 T-Chain을 기반으로 구현된 스트리밍 프로토콜로 기존의 비토렌트 기반의 주문형 비디오 스트리밍 프로토콜인 BiTos와 G2G에 비해 다음과 같은 장점을 갖는다.

- S-TChain은 대칭키 알고리즘을 바탕으로 무임승차를 효과적으로 방지하는 T-Chain의 특성을 그대로 이어받아 비토렌트 기반의 BiTos나 G2G에 비해 무임승차에 대한 내구성이 높다. 실험결과 BiTos나 G2G에서와는 달리 S-TChain 내에서는 무임승차를 통한 스트리밍이 거의 불가능했다.

- 전방 보상기법을 통해 직접 및 간접호혜를 동시에 구현하는 S-TChain은 TFT를 통해 직접호혜만을 구현하는 BiTos나 발송순위 (forwarding rank)를 통해 간접호혜만을 구현하는 G2G에 비해 자원 활용도가 높기 때문에 시스템 자원이 부족한 환경에서 BiTos나 G2G에 비해 훨씬 좋은 성능을 보인다. 시뮬레이션 결과 무임승차를 동반한 플래시 크라우드 환경과 같이 시스템 자원이 극히 부족한 경우 S-TChain은 BiTos나 G2G에 비해 60% 이상의 향상된 성능을 보였다.

하지만 T-Chain을 기반으로 한 S-TChain의 경우 하나의 단점이 있다. 그것은 리처들의 상호협동을 최대로 끌어올리고 무임승차 기회를 최소화하기 위해 시스템 자원이 남는 경우에도 무료로 분배하지 않는 (그래서 일부 자원이 낭비되는) T-Chain의 특성을 그대로 이어받은 S-TChain은 리처들이 공헌하는 것 이상으로는 서비스를 제공하지 않는다는 점이다. 이러한 S-TChain의 특성 때문에 시스템 자원이 풍부한 경우에는 오히려 다른 시스템에 비해 성능이 약간 떨어질 수 있다. 따라서 무임승차 기회를 최소화하면서 리처들에게 잉여 시스템자원을 공평하게 분배할 수 있는 방안들에 대한 연구가 뒤따라야 하겠다.

참고문헌

[1] Cisco visual networking index : Forecast and methodology, http://www.cisco.com/

[2] The bittorrent protocol specification. http://www.bittorrent.org/

[3] B. Cohen, Incentives build robustness in bittorrent, In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, June 2003.

[4] A. Vlavianos, M. Iliofotou, M. Faloutsos, Bitos: enhancing bittorrent for supporting streaming applications, IEEE International Conference on Computer Communications, pp. 1-6, April 2006

[5] J. J. D. Mol, J. A. Pouwelse, M. Meulpolder, D. H. J. Epema, H. J. Sips, Give-to-get: free-riding resilient video-on-demand in p2p systems, In Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 6818, January 2008,

[6] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, A. Venkataramani, Do incentives build robustness in bittorrent?, USENIX NSDI, April 2007.

[7] K. Shin, D. S. Reeves, I. Rhee, Treat-before- trick : Free-riding prevention for bittorrent-like peer-to-peer networks, IEEE International Symposium on Parallel & Distributed Processing, pp. 1-12, May 2009.

[8] T. Locher, P. Moor, S. Schmid, R. Wattenhofer, Free riding in bittorrent is cheap, In Fifth Workshop on Hot Topics in Networks, November 2006.

[9] M. Sirivianos, J. H. Park, R. Chen, X. Yang, Free-riding in bittorrent networks with the large view exploit, International workshop on Peer-To-Peer Systems, February 2007.

[10] M. Feldman, K. Lai, I. Stoica, J. Chuang, Robust incentive techniques for peer-to-peer networks, Proceedings of the 5th ACM conference on Electronic commerce, pp. 102-111, May 2004.

[11] S. D. Kamvar, M. T. Schlosser, H. Garcia -molina, The

eigentrust algorithm for reputation management in p2p networks, Proceedings of the 12th international conference on World Wide Web, pp. 640-651, May 2003.
〔12〕 J. R. Douceur, The sybil attack, International workshop on Peer-To-Peer Systems, March 2002.
〔13〕 R. Landa, D. Griffin, R. G. Clegg, E. Mykoniati, M. Rio, A sybilproof indirect reciprocity mechanism for peer-to-peer networks, IEEE Conference on Computer Communications, pp. 343 - 351, April 2009.
〔14〕 M. Feldman, J. Chuang, Overcoming free-riding behavior in peer-to-peer systems, ACM Sigecom Exchanges, Vol. 5, July 2005.
〔15〕 G. Hardin, Tragedy of the commons, Science 162.
〔16〕 K. Shin, Preventing misbehavior in cooperative distributed systems, Ph.D. thesis, North Carolina State University, December 2009.
〔17〕 L. Jian, J. K. MacKie-Mason, Why share in peer-to-peer networks?, International Conference on Electronic Commerce, August 2008.
〔18〕 Redhat 9 torrent tracker trace. http://mikel.tlm.unavarra.es/~mikel/bt_pam2004/

저 자 소 개

신 규 용

1996년 3월 육군사관학교 전산학 학사
2000년 2월 한국과학기술원 전산학 석사 (ATM 네트워크).
2009년 12월 노스캐롤라이나 주립대학 전산학 박사 (분산 시스템 보안)
현재 : 육군사관학교 전자정보학과 정보과학 부교수
관심분야 : 컴퓨터 네트워크, 분산 시스템, 인센티브, 네트워크 보안
Email : yessss@gmail.com

이 종 덕

2005년 3월 육군사관학교 전산학 학사
2009년 5월 버지니아 주립대 컴퓨터과학석사 (무선 센서 네트워크 전공)
현재 : 육군사관학교 전자정보학과 정보과학 조교수
관심분야 : 무선 센서 네트워크, 보안
Email : jdlee@kma.ac.kr

신 진 희

2001년 2월 금오공과대학교 컴퓨터공학 학사
2006년 2월 국방대학교 전산정보 석사 (엔터프라이즈 아키텍처)
현재 : 육군사관학교 전자정보학과 정보과학 전임강사
관심분야 : 상호운용성, NCW, 엔터프라이즈 아키텍처
Email : suhacci@naver.com

박 찬 진

2010년 3월 성균관대학교 컴퓨터교육 학사
2012년 5월 성균관대학교 컴퓨터공학 석사 (네트워크 전공)
현재 : 육군사관학교 전자정보학과 강사
관심분야 : 캐리어 이더넷, 네트워크 포렌식
Email : pcj0722@kma.ac.kr