ISBN 978-981-11-7861-0
Proceedings of 2018 the 8th International Workshop on Computer Science and Engineering (WCSE 2018)
Bangkok, 28-30 June, 2018, pp. 476-482
doi: 10.18178/wcse.2018.06.082

Re-design the BitTorrent Protocol in Next Generation Expressive Internet Architecture

Ziqian Meng[1+], Zhong Chen[1] and Zhi Guan[2 3]

[1] School of Electronics Engineering and Computer Science, Peking University, Beijing, China
[2] National Engineering Research Center for Software Engineering, Peking University, Beijing China
[3] Laboratory of High Confidence Software Technologies (Peking University), Beijing, China

Abstract. As a popular peer-to-peer protocol, BitTorrent is one of the most important methods for file sharing and distributing on today's Internet. However, the BitTorrent protocol is built entirely at the application level. Without the support of network layer, peers in BitTorrent protocol have to exchange data content through TCP connections and BitTorrent applications has to handle every procedure in the whole file sharing process. As peer-to-peer has become an essential way for people to share and distribute files across the Internet, we believe it should be natively supported mechanism in the future Internet Architecture. The eXpressive Internet Architecture (XIA) is one of three FIA (Future Internet Architecture) projects funded by US NSF. As a clean-slate network architecture, XIA has some novel features such as evolvability, flexible routing, and in-network cache for content delivery. In this paper, we propose a practical design of BitTorrent protocol implementation on XIA to explore and rethink the peer-to-peer file sharing mechanism in the future Internet architecture.

Keywords: Future Internet Architecture, eXpressive Internet Architecture, BitTorrent, P2P File Sharing.

1. Introduction

Peer-to-peer (P2P) file sharing is one of the most popular ways users acquire and distribute data on today's Internet. Since introduced by Napster in 1999, P2P file sharing mechanism has gained huge popularity among Internet users around the world. With more than 25% of today's total Internet traffic is P2P-related [1], it has fundamentally changed the way people share and distribute data. BitTorrent [2] is one of the most popular P2P file sharing applications with more than 150 million active users. Since first introduced in 2003 by Bram Cohen, BitTorrent protocol and applications have been widely used in sharing and distribute content like movies, music and games. The success of BitTorrent is not a coincidence, as it is well designed to provide an effective mechanism to maximize the transfer bandwidth and tackle the free-riding problems. Despite the legal issues, BitTorrent protocol and other P2P file-sharing applications have proved to be an efficient and popular way to share and acquire data.

However, the BitTorrent protocol is built entirely at the application level. Without the native support of network layer, peers in BitTorrent protocol have to discover other peers' IP address and exchange data through TCP connections. Typically, the BitTorrent application can only handle no more than ten TCP connections with other peers at the same time, which limits the potential of achieving maximum download bandwidth. Furthermore, we argue that the TCP connection might not be the best choice for BitTorrent protocol since the data integrity is checked at the application layer and unordered or lost data packets are also not the primary concern in the BitTorrent protocol.

+ Corresponding author.
E-mail address: markmzq@gmail.com

Since BitTorrent and other alike P2P file-sharing protocols contribute a large portion of today's Internet traffic, we believe the design of future Internet architecture should take into account a certain level of native support of the P2P file-sharing mechanism.

XIA (eXpressive Internet Architecture) [3] [4] [5] is one of three FIA (Future Internet Architecture) projects funded by US NSF. As a clean-slate network architecture, XIA has some novel features such as evolvability, flexible routing, and the support of in-network caching for content delivery. XIA supports multiple types of communication and protocol as the a primary entity of the network. Content chunks can be identified by their hash value and treated as primary entity of the network. It can be routed directly at the network level and cached by the on-path routers. The concept perfectly matches the demand and specification of BitTorrent protocol in many ways. Based on this observation, we design native support implementation for BitTorrent protocol in XIA to explore the possible solution for future Internet architecture to support the P2P file sharing.

The rest of this paper is organized as follows. Section 2 introduces the technical background of BitTorrent protocol and XIA. Section 3 presents the design of BitTorrent over XIA and describes the challenges and potential advantages of the implementation. Section 4 discusses the problems to be solved in the future.

2. Technical Background

2.1. BitTorrent Protocol

In the BitTorrent system, a host runs BitTorrent protocol and participates in the file sharing process is called peer. A peer who publishes the file or has the complete set of shared data pieces is called seeder. A peer who has incomplete dataset and downloads data from other peers is called leecher. The collection of seeders and leechers who are participating in the same BitTorrent file sharing process is called a swarm. Each shared file is divided into equal-sized data chunks called pieces. Pieces are the smallest data units in the BitTorrent protocol. It will be further divided into packets in the transport layer and network layer [6].

To start a P2P file sharing process. A seeder has first to create a .torrent file, which contains essential metadata of the shared files and a list of network address of trackers. Trackers keep track of seeders in the same swarm. The metadata in the .torrent file contains the file structure of shared files and the cryptographic hash values (usually SHA-1) of each piece for the integrity verification purpose upon its arrival to the leechers. After the creation of .torrent file, the seeder publishes the file to the Internet for other peers to start downloading process.

To start a downloading process, a peer should first acquire the .torrent file from the Internet and connects to trackers listed in the torrent file to register itself as a leecher. Tracker replies the leecher with a list of peers who are currently sharing the files specified in the .torrent file. The leecher establishes the TCP connection with other peers to transmit data pieces. After receiving the data piece, the leecher verifies the integrity of the piece by its hash value stored in the .torrent file. To achieve faster dissemination and wider distribution of the data pieces, BitTorrent protocol implements an algorithm called "Rarest First." A leecher constantly looks for the rarest data piece that is available among the swarm and request the piece from other peers. The Rarest First strategy has been proven to be more effective than the pure random piece selection strategy [7], [8].

2.2. BitTorrent Design Challenge

Since the BitTorrent protocol is an application layer protocol and the currently TCP/IP architecture is host-centric. A leecher has to discover the IP address of other peers before it starts downloading file pieces. In the early days of BitTorrent, trackers monitor every peer in the swarm. However, the concept of trackers breaks the decentralization philosophy of BitTorrent protocol and can cause single point failures. In the later extension of the protocol, distributed hash table (DHT) [9] and peer exchange protocol (PEX) [10] are introduced as decentralized ways for peers to discover others. However, the peer still needs to acquire the IP address of bootstrap DHT node in the .torrent files and maintain a list of IP address of other peers.

After the peer discovery process, the leecher needs to establish TCP connections with other peers to acquire data pieces. Since it costs system resource to maintain TCP connections, operating systems usually

limit the maximum number of TCP connections the BitTorrent application can have at the same time to no more than ten. This restriction could limit the BitTorrent application's potential capability to achieve higher download speed. Moreover, when the data piece arrives at the leecher, the BitTorrent application has to verify the integrity of the piece by calculating the hash value of the received piece and compare it with the value stored in .torrent file. Due to the lack of native support from network and TCP protocol, BitTorrent protocol has to handle nearly every step of the P2P file sharing process. This heavy responsibility can be inefficient since the application layer has to pay extra effort to obtain and maintain the knowledge of network status.

2.3. Expressive Internet Architecture

XIA is a novel future Internet architecture with the support of multiple communication styles and the ability to introduce new communication entities to achieve network architecture evolution. As the common view in the designs of future Internet architecture, XIA separates identifier and locator to improve scalability and simplify mobility. In terms of network identifier, XIA supports multiple types of network identifier. Different types of communication entities, such as network domain, host, service and data chunk, are coded as different network identifiers in the XIA. Each type of entity is coded as equal-length 160-bit XIA identifier (XID). Network domain (NID), host (HID) and service (SID) are coded according to their public keys for intrinsic security purpose. Data chunks are coded by the hash value of the content for integrity verification purpose. Different XIDs are processed by corresponding protocols on the on-path routers. To mitigate the flat network routing problem brought by hash-style identifier, XIA uses NID for global addressing. After sent to the specific network domain, the packets are forwarded to the host by the HID, which specifies the recipient of the packets. SID is used to achieve anycast-style communications when the originators send packets to a set of servers, instead of one single server, to request a particular service. CID can be used to retrieve content directly from network, regardless of the actual location of content storage. Since content can be directed expressed and recognized by the routers, XIA can natively support the in-network cache by the on-path routers of the network packets.

Regarding network locator, XIA uses directed acyclic graph (DAG) representation of multiple XIDs as the network address. The DAG structure of XIA address is high flexible, which allows the address constructor to express its intent of how the network packets with certain address should be processed along the forwarding path. The simplest form of a DAG address can be constructed by only a dummy source • and an HID node as shown in Fig. 1(a). The dummy source represents the conceptual source with no specific meaning. In order to be forwarded through multiple network domains in real-world Internet environment, currently the DAG address is constructed as shown in Fig. 1(b). Routers check the NID node in DAG and route the packets accordingly before they reach the destination network domain.

Another feature of XIA address is "fallback" path, which allows alternative routing paths expressed in one DAG address. Fig. 1(c) shows a DAG with fallback path, which is represented as a dotted line pointing to node N. If the direct routing path expressed in DAG is unreachable, routers would try to forward the packet to node N by the fallback path. Supposedly node N knows the routing path to the next node in DAG. The solid lines always have higher priority than dotted ones (fallback path) and the fallback path is optional and by definition.

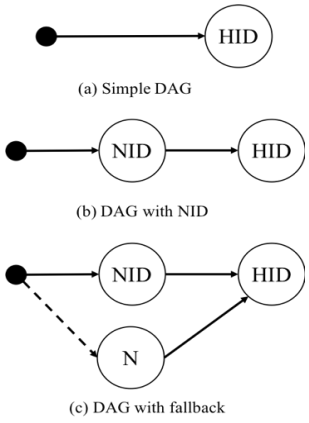

Fig. 1: DAG address structure in XIA.

Field	Description
Announce	Tracker1:AD_1→HID_1 Tracker2:AD_2→AD_3→HID_2 Tracker3:AD_4→HID_3 Tracker4:AD_5→HID_4
Size	xxx bytes
File Names	File1 File2 File3 ……
Piece Length	256KB
CID List	CID_1:E4A2CFA462AD60E53AE5D284A103AA5EA673CD66 CID_2:DB09964C8A70B57110755ED928E8B51962CBFFB2 CID_3:00987C989B0FEC84799FFA62DEBEAECDD9A66DE5 CID_4:7468237B26FE9A86D5F02023916EFAEA44EEC8F8 ……

Fig. 2: File structure of .xtorrent file.

3. BitTorrent over XIA

3.1. Challenges and Benefit of Implementing BitTorrent over XIA

Since the currently BitTorrent protocol and applications are entirely built on the application level. The protocol and applications have to be largely modified to fully utilize the novel features of XIA.

Firstly, the structure of network address in XIA is totally different from the network structure currently used on the Internet, which means BitTorrent applications have to be totally modified to adopt the DAG addresses in XIA. In current BitTorrent protocol, a peer has to send requests to specific seeders to acquire certain data piece. While in XIA, a host can just send a CID request to the network to fetch the data piece from the network without the knowledge of actual storage location.

Despite the challenges in implementing BitTorrent over XIA, some potential improvements in performance can be achieved by leveraging the novel features of XIA. Based on the observation, we notice the fact that hash values of data pieces stored in .torrent file meets the concept of content identifier (CID) in XIA perfectly. The CIDs of data pieces can be directly stored in the .torrent file and used as destination in DAG address to fetch content from seeders and potentially anywhere in the network.

In XIA, all the content packets with CID can be cached by the on-path routers. This feature can also be leveraged by BitTorrent. With the native support of in-network cache, BitTorrent and other P2P redundant traffic can be automatically eliminated. Moreover, with the native support of content transfer in XIA, the integrity of received data pieces can be verified at the network level instead of application level, which leads to a more light-weighted BitTorrent application. The flexible routing feature allows multiple routing paths and destinations to be embedded into one DAG address, which means packets can have multiple forwarding paths as backup routing choices when the primary destination is unavailable. With the support of flexible routing, a peer can send request for a specific data piece to multiple seeders to gain a better chance of successfully fetching the data piece and increase the system robustness.

3.2. Design of BitTorrent over XIA

We resign the .torrent file to meet the network architecture and novel features of XIA. Fig. 2 shows the structure of .xtorrent file, which plays the same role as .torrent file in BitTorrent protocol.

As illustrated in Fig. 2, the general structure of .xtorrent file is similar to .torrent file. Trackers' URLs in "Announce" field are replaced with trackers' DAG addresses. The "CID List" field contains the content identifiers (CIDs) of all the data pieces to be shared by .xtorrent file.

Before sharing with other peers, a publisher creates the .xtorrent file for to-be-shared files. Generally, the publisher needs to divide shared files into equal-length data pieces, calculate each data piece's hash value and fill it into CID list field. After the creation of .xtorrent file, it will be published on the Internet.

For peers who want to participate in the specific file sharing process, the protocol procedure of BitTorrent over XIA is illustrated in Fig. 3.

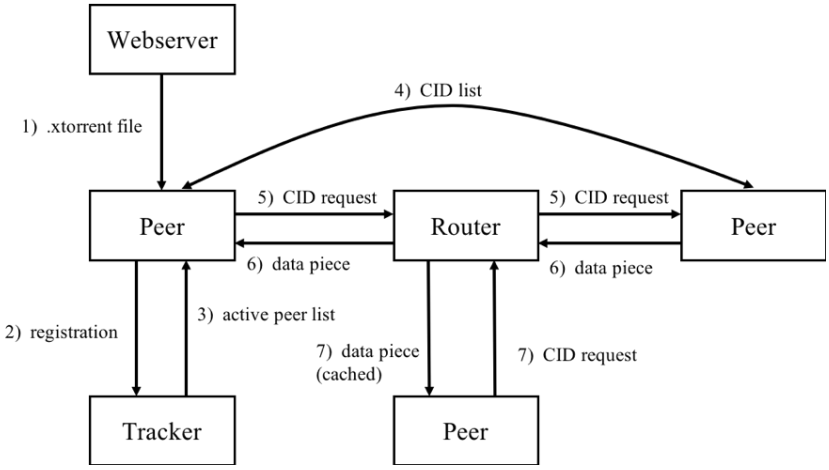

Fig. 3: Procedures of BitTorrent over XIA.

1) Firstly, the peer needs to get possession of the .xtorrent file, which can be acquire from a Webserver or other peers.

2) The peer sends handshake message and registers itself to trackers according to the DAG addresses listed in the "Announce" field of .xtorrent file.

3) Tracker replies the peer with a list of active seeders.

4) Peer sends handshake message to seeders in the list. The seeder replies with a list of data pieces it currently has.

5) The peer sends CID request packets to retrieve data pieces from seeders. Before forwarding the packets, on-path routers check if the requested data piece is in their cache, if so, the router would directly reply the CID request with the packets of data piece. The detail of DAG address will be discussed later.

6) After receiving the CID request, seeder replies the request with packets of data pieces. The on-path routers of the packets choose to cache the data piece according to their own caching policies.

7) Other peers send the CID request to retrieve the same data piece.

8) On-path routers check the CID in their cache and directly reply the CID request with cached content.

The possible DAG addresses of CID request packet are shown in Fig. 4. Since the DAG structure provides huge flexibility in terms of network addressing and routing. A peer has several options when constructing the destination address of the CID request packets.

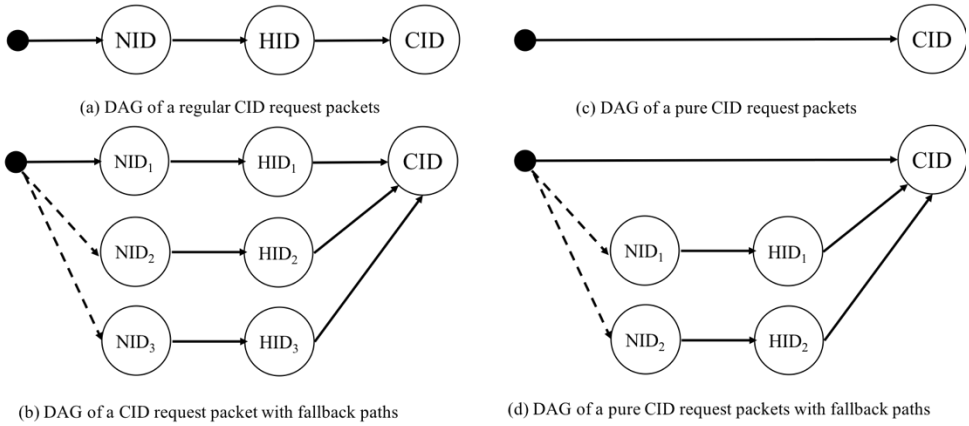

Fig. 4: DAG options for CID request packets.

The DAG address in Fig. 4(a) is the most straight-forward option for leechers. The CID request packets with this DAG will be routed to the seeder at NID→HID for the content of CID. The address shown in Fig. 4(b) is embedded with multiple fallback paths in case the primary seeder is absent. When the seeder at NID1→HID1 is not reachable, routers will try to forward the packet to seeders at NID2→HID2 or NID3→HID3 for the content identified by CID.

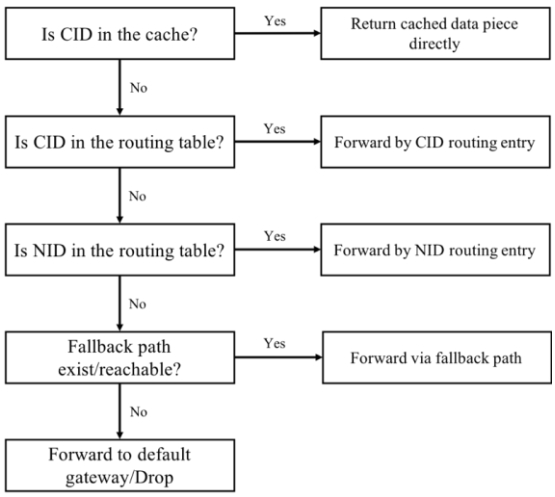

Fig. 5: Standard process flow of a DAG address with CID.

In the network domain where CID routing table is fully implemented on the routers. The leecher can choose to fetch data pieces with DAG shown in Fig. 4(c). The DAG address expresses the intent that a leecher wants to fetch the content from anywhere of the network regardless of its storage location. In the network environments where CID routing table is partially supported, a leecher can use the DAG shown in Fig. 4(d). When the CID in the DAG cannot be found in the routing table or CID routing is not supported by the router, the packets would be forwarded via fallback paths. Fig. 5 shows the standard process flow within a router when forwarding a CID request packet.

4. Discussion & Future Work

Since P2P file sharing is one of the essential ways of transferring content across the Internet, it is beyond doubt that it should be natively supported by future Internet architecture to achieve better performance and less network overhead. In this paper, we propose a practical design of BitTorrent protocol on the XIA, a novel network architecture, as an exploration of possible ways to natively support P2P file sharing mechanism on future Internet architecture.

There are still several design problems to be solved in the implementation of BitTorrent over XIA. For instance, XIA uses LRU as its in-network cache replacement algorithm which has a limited efficiency in some scenarios. We believe that a cache replacement algorithm specifically optimized for P2P is needed to unleash the power of BitTorrent in XIA and other future Internet architectures. Moreover, the design proposed in the paper have only been deployed at testbed and simulation environment. We have planned several field tests to evaluate the performance of the design in the real world. Meanwhile, figuring out an appropriate comparison method to evaluate the performance network protocol in both current IP architecture and future Internet architecture is also one of our future works.

5. Acknowledgements

This research is funded by National Natural Science Foundation of China, Project# 61672060.

6. References

[1] Price, David. "An estimate of infringing use of the internet." Disponıvel em, http://documents. envisional. com/docs/Envisional-Internet-Usage-Jan2011. pdf (2011).

[2] Cohen, Bram. "Incentives build robustness in BitTorrent." Workshop on Economics of Peer-to-Peer systems. Vol. 6. 2003.

[3] Anand, Ashok, et al. "XIA: An architecture for an evolvable and trustworthy Internet." Proceedings of the 10th ACM Workshop on Hot Topics in Networks. ACM, 2011.

[4] Han, Dongsu, et al. "XIA: Efficient Support for Evolvable Internetworking." NSDI. Vol. 12. 2012.

[5] Naylor, David, et al. "XIA: architecting a more trustworthy and evolvable internet." ACM SIGCOMM Computer Communication Review 44.3 (2014): 50-57.

[6] Bharambe, Ashwin R., Cormac Herley, and Venkata N. Padmanabhan. "Analyzing and improving bittorrent performance." Microsoft Research, Microsoft Corporation One Microsoft Way Redmond, WA 98052 (2005): 2005-03.

[7] Felber, P., and Ernst W. Biersack. "Self-scaling networks for content distribution." Proc. International Workshop on Self-* Properties in Complex Information Systems. 2004.

[8] Loewenstern, Andrew, and Arvid Norberg. "BEP 5: DHT protocol." last modified on Feb 28 (2008).

[9] Wu, Di, et al. "Understanding peer exchange in bittorrent systems." Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on. IEEE, 2010.