Introducing Peer Copy - A Fully Decentralized Peer-to-Peer File Transfer Tool

1st Dennis Trautwein
Data & Knowledge Engineering
University of Wuppertal
Wuppertal, Germany
trautwein@gipplab.org

2nd Moritz Schubotz
Leibniz Institute for Information Infrastructure
FIZ-Karlsruhe
Berlin, Germany
moritz.schubotz@fiz-karlsruhe.de

3rd Bela Gipp
Data & Knowledge Engineering
University of Wuppertal
Wuppertal, Germany
gipp@uni-wuppertal.de

***Abstract*—Peer Copy is a decentralized, peer-to-peer file transfer tool based on `libp2p`. It allows any two parties that are either both on the same network or connected via the internet to transfer the contents of a file based on a particular sequence of words. Peer discovery happens via multicast DNS if both peers are on the same network or via entries in the distributed hash table (DHT) of the InterPlanetary File-System (IPFS) if both peers are connected across network boundaries. As soon as a connection is established, the word sequence is used as the input for a password-authenticated key exchange (PAKE) to derive a strong session key. This session key authenticates the peers and encrypts any subsequent communication. It is found that the decentralized approach to peer-to-peer file transfer can keep up with established centralized tools while eliminating the reliance on centralized service providers.**

***Index Terms*—libp2p, ipfs, peer-to-peer, p2p, file transfer, DHT**

I. Introduction

Individual file transfer seems to be a solved problem with processes ranging from physically transporting thumb drives to protocols like `ftp`[1] or `smtp`[2], to commercial services like Dropbox[3]. Yet all of these tools require an inconvenient setup procedure. You need to be in physical possession of the thumb drive, you need to have the destination server properly configured to accept `ftp` or `smtp` requests, or both peers need to have an account at the same file hosting service. A set of tools, most notably `croc`[4] and `magic-wormhole`[5], solve this problem by only requiring the user to transmit a short passphrase to the receiving peer to initiate a file transfer. They allow the transmission of data without special knowledge about the technical infrastructure employed by the peers. These tools, however, rely on a small set of servers which are usually operated by the maintainers of the open-source projects to orchestrate peer discovery and data relaying [1]. This model of operation poses centralization concerns and puts the service's sustainable operation in question as a recent issue in the `croc` repository shows[6]. The small set of private servers constitute single points of failure and an attack target to disrupt the service. Further, the service operators have the power over whom to serve and can gather extensive knowledge about communication patterns.

In this paper, we present *Peer Copy* (`pcp`) – a decentralized, peer-to-peer file transfer tool based on `libp2p`[7]. Many concepts like the command-line user interface and user experience, as well as the Password-Authenticated Key Exchange (PAKE) [2] and the concept of channels (explained later in section II) were adapted or reused from `croc` and `magic-wormhole`. The novelties of this tool are the extensive architectural differences in the peer discovery and data relaying mechanisms that render centralized server infrastructure obsolete. The main contribution of this paper is **a decentralized peer discovery mechanism based on low entropy passphrases**.

During usual operation, the `pcp` process lifecycle can be separated into the stages of peer discovery, peer authentication, and file transfer. The novelty of `pcp` lies in the decentralized peer discovery mechanism, which employs multicast DNS (mDNS)[8] and the Distributed Hash Table (DHT) from the Interplanetary File-System (IPFS) [3]. Peer authentication is done via PAKE, where four random words from the 39th Bitcoin Improvement Proposal (BIP39)[9] serve as a passphrase. File transfer can either be direct or transitive through a `libp2p` relay node.

The following section II describes the functionality and explains how each aforementioned concept relates to and facilitates the file transfer capabilities of `pcp`. Section III gives an outlook and future improvement opportunities.

II. Functionality

`pcp` provides two top-level commands `pcp send` and `pcp receive`. When running `pcp send`, the user is presented with four words that need to be transferred, e.g., spoken or digitally, to a peer who will then use these words to run `pcp receive the-four-random-words`. Both instances of the `pcp` process initiate the discovery procedure and try to find each other. After successful connection, authentication, and manual confirmation, the file gets transferred.

[1] https://www.rfc-editor.org/rfc/rfc959.txt
[2] https://www.rfc-editor.org/rfc/rfc5321.txt
[3] https://www.dropbox.com/
[4] https://github.com/schollz/croc
[5] https://github.com/magic-wormhole/magic-wormhole
[6] https://github.com/schollz/croc/issues/289
[7] https://libp2p.io/
[8] https://www.rfc-editor.org/rfc/rfc6762.txt
[9] https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

Peer discovery, among relay services, is one of the mechanisms that poses centralization concerns in established tools like `croc` and `magic-wormhole`. `pcp` employs mDNS and a novel DHT-based approach to enable the discovery of the desired peer. In the following section, we focus on the DHT-based approach and later extend the discussion to mDNS.

a) Discovery: Peer discovery works by devising an identifier that can be constructed by both parties solely based on shared information and information that can be derived independently. For the former, `pcp` uses the aforementioned word sequence, and for the latter, the current system time. This identifier is then used as a rendez-vous point in the DHT.

When the user states the intention to initiate a file transfer by running `pcp send FILE`, four words from the BIP39 English wordlist are chosen at random. The words are claimed to be easily memorable, typeable, and pronounceable. The first word is interpreted as a numeric channel identifier (ID) by taking its index in the wordlist in the range of 0 to 2047. `pcp` uses the channel ID and the current unix system timestamp to generate the discovery ID: `/pcp/{timestamp}/{channel-identifier}`. The timestamp is used to limit collisions of the discovery ID between multiple simultaneously and independently operating peer pairs. Further, it is truncated to the most recent 5-minutes time slot to adjust for clocks that are out-of-sync. `pcp` then puts the Content Identifier (CID)[10] of that discovery ID as a "provider record" in the DHT, indicating that it possesses the associated data and is willing to provide it (Fig. 1a). Note that provider records usually contain the CIDs of the underlying data, which has integrity validation advantages. This is different in the case of `pcp` because the string is not based on the content to be transmitted. It is solely used for discovery purposes.

On the receiving side, the user runs `pcp receive the-four-random-words`, constructs the identical discovery ID, and queries the DHT for providers of the associated CID (Fig. 1a). To further mitigate clock synchronization issues and time slot boundary problems, the peer also queries the DHT in parallel for the previous 5 minute time slot. As soon as the peer has found the DHT provider record `libp2p` handles the resolution to an associated publicly reachable IP-address. It uses the "peer records" in the DHT that were automatically populated by `libp2p` on the sending side. These peer records may contain `libp2p` nodes that have advertised relaying capabilities[11]. This addresses the second centralization concern elaborated above.

When users want to transmit files on the local network `pcp` employs mDNS for peer discovery. This is completely transparent to the user as this discovery is started in parallel to the DHT-based mechanism. `pcp` uses mDNS for advertising the discovery ID from above in the current network. The receiving peer issues mDNS queries for the same discovery ID to find the peer on the local network. If the query resolves, the receiving peer immediately establishes a connection and continues with the authentication steps.

[10]https://docs.ipfs.io/concepts/content-addressing/

[11]https://docs.libp2p.io/concepts/circuit-relay/

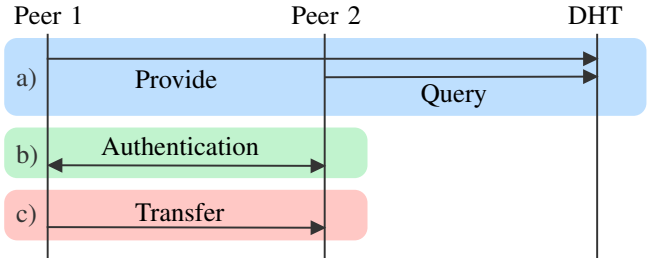

Fig. 1. Different stages of the `pcp` file transfer protocol. In the depicted scenario *Peer 1* wants to transfer a file to *Peer 2*.

b) Authentication: It is still possible that two or more pairs of peers are simultaneously trying to transmit a file with the same channel ID. This means that their discovery IDs collided and therefore may establish connections to wrong peers. `pcp` ensures with an authentication step, based on the remaining three words, that the connection is established between the correct peers.

This authentication step is adapted from `croc` and `magic-wormhole`. The peers use the complete list of words as the input passphrase to the PAKE protocol to derive a strong session key. The peers use this session key to send each other challenges to ensure that both parties arrived at the same key (Fig. 1b). If the challenge failed, the connection is dropped; otherwise, the peers proceed to the file transfer stage. Every subsequent communication is now encrypted with the strong session key.

c) Transfer: In the file transfer stage, the sending peer first transmits information about the file to be sent to the receiving peer. The user is prompted these information like filename and file size to confirm the file transfer. Upon confirmation, the transfer starts (Fig. 1c).

III. Conclusion & Future Work

We found that the discovery mechanism of `pcp` presents a viable alternative to established centralized methods. Yet, DHT query resolution can lie in the order of minutes while the alternatives operate in the seconds or even sub-second range. This will likely improve as `pcp` is built on `libp2p` and will therefore benefit from its upcoming enhancements. In the future, we plan to decrease the reliance on bootstrap nodes by, e.g., by employing a locally installed IPFS node. Further, the `libp2p` reference implementation in Javascript makes it possible to construct interoperability between a browser-based and the command-line version of `pcp`.

References

[1] B. W. P. 2016. Brian warner - magic wormhole- simple secure file transfer - pycon 2016.mp4. Youtube. [Online]. Available: https://www.youtube.com/watch?v=oFrTqQw0_3c

[2] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptography*, 0th ed. Stanford University, 1 2020.

[3] J. Benet, "IPFS - content addressed, versioned, P2P file system," *CoRR*, vol. abs/1407.3561, 2014. [Online]. Available: http://arxiv.org/abs/1407.3561