Performance of WebRTC in the Context of a Decentralised Storage Solution

Pierre-Louis Dubouilh
University of Strathclyde
Department of Electronic
& Electrical Engineering
Glasgow, United Kingdom
gfb13141@uni.strath.ac.uk

Greig Paul
University of Strathclyde
Department of Electronic
& Electrical Engineering
Glasgow, United Kingdom
greig.paul@strath.ac.uk

James Irvine
University of Strathclyde
Department of Electronic
& Electrical Engineering
Glasgow, United Kingdom
j.m.irvine@strath.ac.uk

***Abstract*—Distributed Hash Table-based storage solutions provide a secure means for the storage and retrieval of data. While DHTs present interesting features, specifically around their inherently decentralised nature, in contrast to most Internet services, they also experience a significant reduction in performance when the latency increases between two peers. The challenge of latency is a particular concern for mobile users, and those using cellular connections to the Internet, which typically encounter higher network latencies than users on fixed-line wired connections. This paper proposes that the challenge can be partially mitigated by using the DHT only once, for peer discovery, to coordinate the initiation of data transfer directly between two peers using WebRTC. This raises potential for the deployment of such techniques in the near future, on account of the widespread availability of WebRTC technology in modern Internet browsers, both on desktop and mobile platforms.**

***Keywords*—decentralisation, DHT, Kademlia, peer-to-peer, WebRTC**

I. Introduction

A common technology used in the creation of decentralised storage and data transmission networks is the Distributed Hash Table (DHT), one of the first of which was presented in [1]. A DHT is a logical construct within which data can be stored across a large number of computers, which are themselves distributed around the world. To achieve secure storage of private data within a DHT, without a central point of failure (such as a server, or other company providing a storage service), user data must be sent to the DHT in an encrypted form. This prevents other users from being able to read the data. Additionally, in order to achieve reliability of storage, and protect against computers on the DHT going offline, the encrypted data should be split into smaller chunks, which are then replicated. This replication of chunks allows for the recovery of data when some systems on the DHT are offline or have failed. One of the key characteristics of a DHT is that, in addition to its ability to store data, it also efficiently and scalably routes requests for data, while maintaining reasonably-sized routing tables on each node. This means that every member of the network need not have direct knowledge of each other network member, since requests for any given piece of data can be routed via known nodes.

Our previous work has investigated the real-world performance limitations of DHT-based storage and service implementations [2], [3]. This highlighted the significance of latency within DHT-based networks, on account of DHT member nodes being organised based on arbitrary uniformly generated addresses, rather than based on geographic location. While this improves the resiliency of a network, since logically-adjacent nodes are likely to be uniformly distributed around the world (and thus adjacent nodes are not likely to be lost at one time), it also means that many of these logically close nodes are likely to encounter significant latencies between each other.

This paper considers an alternative approach to resolving these performance challenges, by restricting the use of the DHT to the discovery of peers, and then carrying out all subsequent data transfers directly between sender and recipient using a technology such as WebRTC, which is designed to allow for the direct communication of data between web browsers. This allows for geographical proximity (or rather, network proximity, measured by latency or another performance metric) to be used when carrying out data transfers, while ensuring that the robustness of a geographically distributed hash table continues to provide resiliency against network disruption, even on a large scale.

II. The Distributed Hash Table

A. Overview

A DHT is fundamentally a decentralised key-value pair store. Data (the value) is stored at a given address (the key), and this key is selected based upon the output of a cryptographic hash function. This prevents collisions within the hash-space, which would occur if an attempt was made to store another, different, piece of data at the same address. It is therefore possible to retrieve an arbitrarily large quantity of data from a DHT, holding only the knowledge of a single cryptographic hash output (DHT key).

Data is retrieved from the network by making a request for the corresponding value of a known key. The DHT client then carries out recursive lookup operations, querying known neighbouring nodes which are members of the DHT, enquiring as to if they know of a route to reach the address of the key. Since the nodes holding a given value are those closest to the address of the key, this operation allows a user to retrieve a given piece of data even when they do not know of a means to communicate with a node currently holding the data, provided another member of the network does.

This allows for a highly scalable decentralised network, where it is not necessary for every member of the network to know about (or receive and propagate signalling messages about) every other node. Rather than carrying out flood-based requests (which would severely impair the network's ability to grow), users of a DHT communicate with a small selection of neighbours. Every member of the DHT has a different set of neighbours, and nodes are identified by their own address (within the same hash address-space as data chunks), making it is possible to evaluate available neighbours' connectivity, to locate the desired data.

The DHT implementation used for this work is a variant of Kademlia [4]. Using a XOR based metric system, this DHT allows to find peers with a difficulty of $O(log(n))$. Compared to other DHT implementations, Kademlia is highly efficient and scalable, and is commonly used for various large DHT implementations. For example, Kademlia is at the heart of BitTorrent's mainline DHT, meaning it is used on a daily basis by millions of users throughout the world to discover peers sharing a given set of files.

B. DHT Limitations

While DHTs can be very flexible, they suffer from an efficiency issue. Various lookups are needed in order to locate a given piece of data on the network. These frequent requests require a lightweight means of data exchange between the peers, in order for them to be performed quickly. For this reason, lightweight UDP datagram packets are typically preferred over regular TCP links (which have greater overhead and setup time). While UDP packets allow for rapid, low-overhead lookup requests and responses, actual data packets have a very constrained payload - the maximum theoretical size for a datagram packet is 65KB, although as discussed in [2] this is not typically achieved. In real deployments, UDP packets of under 8KB are preferred in order to prevent the loss of packets — many networks are not designed to handle larger UDP packets (which themselves are not widely used).

While the use of a DHT presents significant benefits for decentralisation and scalability of services, its performance is highly dependent on the average latency between a node and the nodes it is required to communicate with, when locating the required required data [2]. As such, we found when a user's network latency is significant, the actual data throughput significantly decreases when the latency increases. This issue hinders the use of such a technology in real-world applications, as the latency would naturally fluctuate between the various peers of the network, causing poor performance, as we identified in our previous work. Specifically, this presents a challenge for mobile users, whose connections may encounter higher (and more variable) latencies than users with fixed-line connections. Nonetheless, the performance impact of inter-node latency was still visible when carrying out experiments between servers, each with gigabit connectivity, that were geographically distributed throughout the world. Since chunks may be uniformly located around the world, it is therefore necessary to carry out worldwide searches for the chunk.

This results in significant query latency, since the requests to locate the chunk will require attempts to be made to distant (geographically) nodes, with high latencies. This poses a limitation on the chunk lookup, and therefore blocks access to the file until it is located. If retrieval of a given piece of data depends on communicating with high-latency nodes, as a DHT typically entails, this poses a significant delay on the retrieval of each chunk. In scenarios (such as storage systems like MaidSafe [5]) where data is encrypted using chain-mode ciphers (such as the AES cipher in used in CBC mode), it is necessary for previous chunks to be retrieved prior to decryption of the next chunk. This means that access to a file is constrained by the access performance of the slowest chunk to be retrieved.

III. Channelling Directly

The solution this paper proposes is to use the DHT to establish a direct connection between the various peers, where peers are prioritised based on their relative latencies. This direct connection can be established following different various protocols. For the purpose of this paper, we used the Web RealTime Communication API (WebRTC), as a mean to exchange data directly between the peers.

WebRTC [6], is an API defined by the W3C, with the aim of extending the capacities of modern web-browsers. This recent technology created a means for direct browser-to-browser communications, allowing various applications from in-browser video chat, to peer-to-peer file sharing services. The WebRTC protocol handles and works around NAT and other connectivity challenges typically faced when attempting to directly connect to other users directly. Significantly, WebRTC is available in modern web browsers, and is itself standardised, and allows for the potential use of decentralised services through an interface which users are already familiar with — the web browser.

WebRTC presents various advantages compared to usual peer-to-peer protocols. First of all, the availability is technically very broad, as a simple updated web-browser is required to connect to the network. This reduces the barrier-to-entry of using a decentralised network, and makes it possible for other web-based software to potentially access content from the decentralised network, in a manner in which users are familiar with. WebRTC has also been developed with a "security by default" design, so all communications are encrypted. The protocol also handles the signalling process, so there is no need for every user to have the directly routable public IP address to allow data-exchange. Interactive Connectivity Establishment (ICE) [7] is used, alongside Traversal Using Relays around NAT (TURN) [8] servers to circumvent NAT limitations that peers may encounter.

In order to allow for the web browser to facilitate such networks, it would be necessary to include DHT support in the browser. This would facilitate the discovery and location of other peers, and is itself the subject of a standards track RFC [9] for Resource Location and Discovery. Having the ability to use these technologies directly within standards-complaint web browsers would also be of benefit for mobile users, where application and service developers may build their software for a single standardised platform, rather than for each different, incompatible mobile platform.

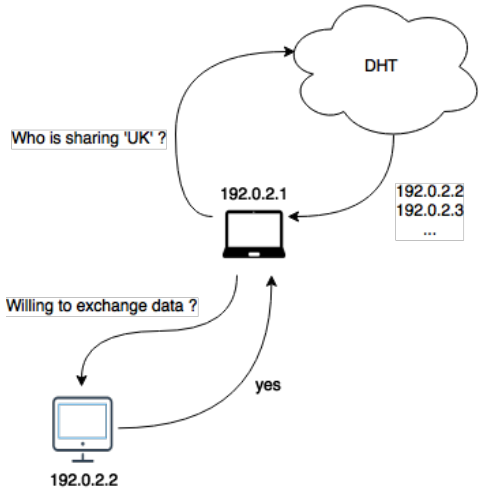

Fig. 1: Interested Peer Discovery Process

IV. Peer Discovery

Unlike the solution proposed by Maidsafe [5], which stores encrypted blobs of data within a DHT, we propose that the data be exchanged directly between the peers. The discovery of interested peers will still be performed with the DHT, in order to keep the service decentralised and prevent reliance upon centrally controlled discovery servers.

The Mainline BitTorrent DHT (Kademlia) has been used in the scope of this project, as it is very reliable, and already used by millions of users. With our solution, unlike a purely DHT-based solution, a DHT-lookup is carried out only once, to locate peers. Subsequent data transfers may take place outwith the DHT, with peers selected as having the lowest round-trip latency. By directly communicating with discovered peers, rather than locating content within the DHT, there is no need for latency-sensitive queries (such as identifying files available in a folder, when the user enters the folder) to be passed around the DHT.

For instance, an interested peer in the United Kingdom wishing to store encrypted chunks will locate another party also wishing to do the same, and they will exchange chunks. To carry this out, the users initially query the DHT, requesting information on users sharing the a DHT key of `uk` (their approximate geographical location). Users may query several locations, if they are happy to accept users from a variety of nearby countries. All users who wish to be a part of this approximate geographical region may become a member of it. The DHT returns a list of IP addresses which are holding this virtual chunk, which indicates peers that would be potentially interested exchanging data. The client is then able to select one (or more) of the IP addresses returned, and start an exchange directly with it, using WebRTC. The process is illustrated in Figure 1.

V. Related Works

While this solution can be compared to Bittorent's BTSync[10] (and to a certain extend to Bittorent), there are some fundamental differences. BTSync is a service offered by Bittorent to synchronise a user's data between their devices. BTSync advertises that it is not relying on a "cloud", as it, indeed, does not rely on one single point of storage, and the data is actually spread between a user's running instances using the Bittorent protocol. The main difference between the proposed solution and this protocol is that while the BTSync DHT is used as a means of discovering peers sharing a given file on the Bittorent network, our solution aims at using the DHT to discover peers that are willing to accept and store files from the current user. Therefore, while BTSync is designed to retrieve a user's files from their own devices wherever they may be, our protocol is designed to find a place to store a user's data, and then retrieve it later, even though this will (most likely) be on someone else's system. Therefore, while BTSync finds a device based on it holding a particular file, we propose that the DHT can instead be used to identify suitable low-latency peers, which can then be used for future storage requests.

VI. Implementation and tests results

In order to measure the efficiency of the presented solution, an implementation has been created in NodeJS. The chunk self-encryption is identical to that used in the Maidsafe network [11]. Using the hybrid solution, the peer discovery process took an average of 15 seconds to find the IP address of an interested peer. Although, using this solution, the data transfer itself is much faster; depending on the bandwidth available, we found it possible to use (and saturate) the entire connection capacity between two users, once the connection was established. Significant here is that the DHT is only used for peer discovery, meaning that while the latency remains present for this discovery process, data can be directly exchanged after this is complete. There are therefore no further latency-prone DHT requests needed in order to begin exchanging data, or indeed to query this node for data. Additionally, once peers are discovered, it is possible to re-use them for future requests (or to gather a small group of such peers, and then use them without re-locating them on the DHT). A DHT discovery request would only be needed in the event of a user wishing to identify new peers if their existing ones were going offline.

It is important to note, however, that careful consideration must be given to the selection of peers for resiliency. For example, a user should not place all their chunks on a single peer node (or indeed a small group of nodes), as this would result in the effective re-centralisation of their data on that one node. Provided that adequate replication takes place, however, across multiple nodes, users should have similar levels of protection of data compared to a purely DHT-based solution.

Figure 2 shows the difference in throughput between the two solutions. While the DHT-only solution provides satisfactory results in a low latency environment (approximately 420 kB/s on a 3.5ms latency connection), the actual throughput is drastically reduced when encountering higher latency, falling to 50kB/s on a 100ms latency connection. On the other hand, while the hybrid solution takes approximately 12 seconds to fetch the information related to the IP address of the interested peer, the throughput afterwards is saturating the available bandwidth.

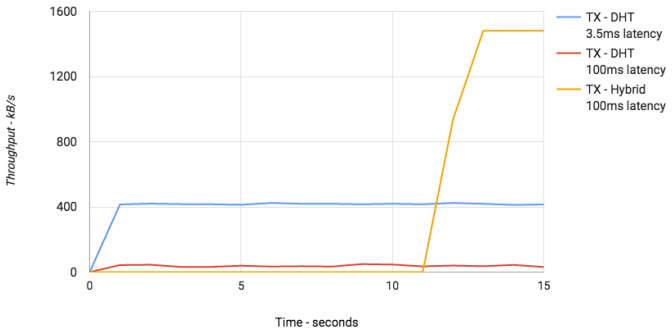

Fig. 2: Throughtput between the proposed solutions

It is worth noting that due to technical constraints on the use of the mainline DHT, the measurements performed on the purely DHT based solution were achieved on a local DHT composed of only two nodes. These conditions were therefore favourable for the DHT's performance (the benchmark for our proposal), and the resulting throughput is the maximum throughput achievable using such a solution. The results for our proposed hybrid solution were obtained using Bittorent's Mainline DHT. This public DHT is composed of million of users, therefore giving less favourable performance. It also highlights the performance benefits of the proposed hybrid solution, which we found could achieve a higher throughput than the best-case situation where the DHT was used directly for the storage of data.

VII. Conclusion

This paper presented our proposal for a solution to the challenge of DHT-latency when attempting to deploy a decentralised storage network. Our proposal uses WebRTC, which is widely available in modern web browsers, and which allows for new and interesting web-based applications to leverage decentralised storage technologies. Making decentralised services available through the web browser is a major benefit for users of mobile devices, as it offers a platform-agnostic base upon which these services can run. Rather than use a DHT for location of chunks and transfer of data, we propose the use of the DHT solely for the discovery of peers which are somewhat geographically close (i.e. within the same region), to reduce latency between peers. We found our approach yielded significant performance gains over traditional DHT-based data transfers, as carried out in our previous work. Our proposed solution was able to avoid the extensive lookup phases required for each chunk of each file, allowing for lower latency access to data, and higher throughput in general. Being less dependent to the intrinsic challenges of network latency when dealing with a worldwide network allows our proposed solution to offer high data transfer performance, while still remaining decentralised, without any single entity able to bring the network down through their failure.

Acknowledgement

This work was partly funded by EPSRC Doctoral Training Grant EP/K503174/1 and MaidSafe.Net.

References

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149–160, Aug. 2001. [Online]. Available: http://doi.acm.org/10.1145/964723.383071

[2] G. Paul, P. Dubouilh, and J. Irvine, "Performance challenges of decentralised services (in press)," in *Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd*, September 2015, pp. 1–5. [Online]. Available: https://pure.strath.ac.uk/portal/files/43605996/Paul_et_al_VTC2015_challenges_of_decentralised_services.pdf

[3] G. Paul and J. Irvine, "5G-enabled decentralised services," in *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*, May 2015, pp. 1–5.

[4] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[5] D. Irvine, J. Irvine, and S. K. Goo, "Sigmoid (x): Secure distributed network storage," in *WWRF27*, 2011. [Online]. Available: https://pure.strath.ac.uk/portal/files/8395548/wwrf_27_sigmoid_system_EPS.pdf

[6] A. Bergkvist, D. Burnett, C. Jennings, and A. Narayanan, "WebRTC 1.0: Real-time communication between browsers," *World Wide Web Consortium WD WD-webrtc-20120821*, 2012.

[7] J. Rosenberg, "Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal for offer/answer protocols," Tech. Rep., 2010.

[8] S. Perreault and J. Rosenberg, "Traversal using relays around NAT (TURN) extensions for TCP allocations," 2010.

[9] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) base protocol," Internet Requests for Comments, RFC Editor, RFC 6940, January 2014.

[10] "BTSync," https://www.getsync.com/, 2015, [Online; accessed 10-Sept-2015].

[11] G. Paul, F. Hutchison, and J. Irvine, "Security of the MaidSafe vault network," in *WWRF32*, 2014. [Online]. Available: https://pure.strath.ac.uk/portal/files/34898763/Paul_etal_wwrf32_vault_network.pdf