Implementing Distributed Hash Table for Data Advertisement and Lookup in Seattle Testbed

Xin Tong[1] Faculty Advisors: Albert Rafetseder[2] Justin Cappos[3]

1. NYU Abu Dhabi Class of 2019 2. Research Professor, Secure Systems Lab, NYU Tandon School of Engineering
3. Professor and Lab Director, Secure Systems Lab, NYU Tandon School of Engineering

Abstract

The purpose of this project is to build an interface for using Distributed Hash Table (DHT) to announce and look up information in Seattle Testbed. A DHT is a decentralized system commonly used in peer-to-peer software like BitTorrent; it provides a lookup service similar to that of a hash table; any participating nodes can retrieve the value associated with a given key efficiently from its peers. The first part of this project is to implement a DHT in Repy, Seattle's restricted subset of the Python programming language. The second step is to build a library that interfaces the DHT and directly provides Seattle users data advertisement and lookup functionalities.

Project Background

Distributed Hash Tables has been a research topic in the field of decentralized systems over the past two decades. There are multiple existing algorithms and protocols for DHT implementations, the original ones including Chord, Pastry and Tapestry. DHTs are decentralized, scalable and fault-tolerant to a certain extent; these properties make them suitable for practical uses such as peer-to-peer file sharing and content distribution.

BitTorrent is one of the most common protocol for data and file distribution over the Internet. In 2005, BitTorrent Inc. released their own "Mainline DHT" implementation based on the Kademlia DHT protocol. By 2013, BitTorrent has 15 – 27 million concurrent users[1]. In this project, we examine both the original Kademlia paper and the BitTorrent protocol to build our own DHT for Seattle.

Implementation Details

The Kademlia system assigns 160-bit opaque IDs to its nodes, and provides a lookup algorithm that locates successively closer nodes to any desired ID, converging to the lookup target with logarithmic efficiency[2]. Distances between nodes are calculated with the XOR metric, which is unidirectional, symmetrical, and implicit in the binary-tree-based design of the Kademlia routing table.

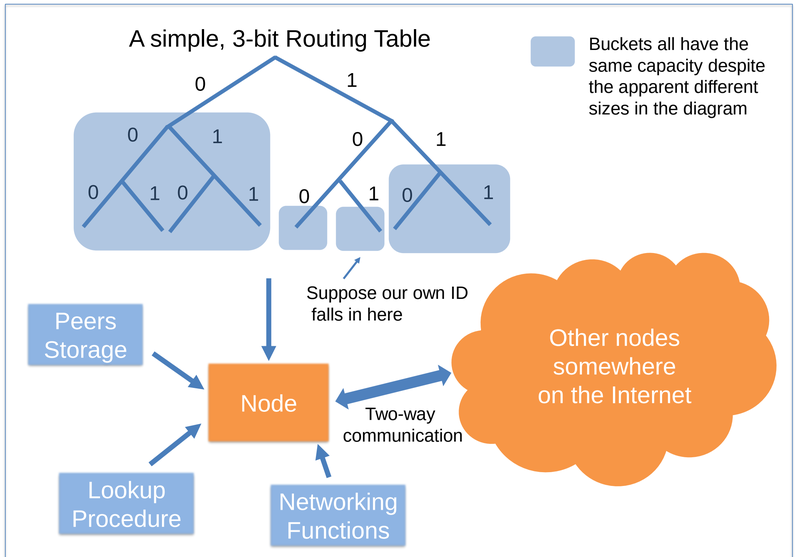

Figure 1: Kademlia DHT Node components and structure

For a DHT node to function properly, it needs several subcomponents. For instance, the routing table stores peers' information and groups them into buckets. The size of each bucket is a constant k (typically k=8 in BitTorrent). The node also needs to perform an essential lookup procedure to announce and lookup data. To communicate with nodes scattered around on the Internet, each node agrees to use bencoded RPC messages, as specified by the BitTorrent protocol[3]. In addition, the DHT system maintains itself by performing refreshes on its data and sending messages to peers periodically.

announce_peer message fields		example packet (bencoded)
"id"	"<querying node id>"	d1:ad2:id20:abcdefghij0123 4567899:info_hash20:mno pqrstuvwxyz123456e1:q9:g et_peers1:t2:ab1:y1:qe
"info_hash"	"<20-byte target infohash>"	
"port"	<port number>	
"token"	"<opaque token>"	
"transaction id"	"<typically 2-4 bytes transaction identifier>"	

Figure 2: A Table showing an example of one of the four RPC messages

Results and Conclusion

By the end of this project, we will have a functional DHT capable of sending and receiving data packets through UDP connections. When called upon, the setup function of the advertise library bootstraps the DHT into an initial network by contacting several public, known nodes through their DNS addresses. After the bootstrap step is completed, the DHT can perform data advertisement and lookup by contacting the peer addresses it has saved into its peer file and accumulated during runtime.

Future Plans

- Follow the extended BitTorrent protocol in order to store arbitrary data in the DHT [4]
- Add additional functionalities such as nodes blacklist

Works Cited

[1] Wang, Liang; Kangasharju, Jussi. (2013). "Measuring Large-Scale Distributed Systems: Case of BitTorrent Mainline DHT". *IEEE Peer-to-Peer*.
[2] Maymounkov, Peter; Mazières, David (2002). "Kademlia: A Peer-to-peer Information System Based on the XOR Metric". *Proceeding of the 1st International Workshop on Peer-to-Peer Systems,*53-65.
[3] Loewenstern, Andrew; Norberg, Arvid (2008). "DHT Protocol (BitTorrent BEP-5)".
[4] Norberg, Arvid; Siloti, Steven (2014). "Storing arbitrary data in the DHT (BitTorrent BEP-44)".