Extended Prefix Hash Trees for a Distributed Phone Book Application

Fabian Stäber, Siemens Corporate Technology, Germany
Gerald Kunzmann, Technische Universität München, Germany
Jörg P. Müller, Clausthal University of Technology, Germany

ABSTRACT

Besides file sharing, IP telephony has become the most widely used peer-to-peer-based application. Software and hardware phones with built-in peer-to-peer stacks are used to enable IP telephony in closed networks on large company sites as well as in the Internet. As part of the PeerThings project, Siemens developed a decentralized communication platform supporting video communication, voice communication, instant messaging, and so on. While decentralized communication has already been implemented in related work, providing a scalable peer-to-peer-based distributed directory for searching user entries is still an unsolved challenge. In this article we present the Extended Prefix Hash Tree algorithm that can be used to implement an indexing infrastructure supporting range queries on top of DHTs.

Keywords: Distributed Search Index, Peer-to-Peer, Voice-Over-IP

INTRODUCTION

The PeerThings communication platform uses a Chord-like DHT (Stoica, et al, 2001) as the underlying decentralized infrastructure. Each user publishes its current IP address and port number using a unique user identifier as the keyword. In order to establish a communication channel to a user, the user’s identifier must be looked up in order to learn the TCP/IP connection data.

However, users do not always know the unique identifier of the person to be contacted. Therefore, it must be possible to look up the identifier in a phone-book-like user directory. When looking up an identifier, the user might not know all data necessary to start an exact query. For example, the user might know the last name of the person to be searched, but not its first name or address. Moreover, people often are not willing to fill out all data fields, e.g. the address of the person to be called. Therefore, the phone book is required to support range queries, like queries for all people with a certain last name.
The main challenges arise from the non-uniform distribution of peoples names. Figure 1 shows the frequency of last names in the city of Munich, Germany. The last names are Pareto-distributed, or Zipf-distributed, i.e. there are a few last names that are very common, while most last names are very rare.

In this article, we present Extended Prefix Hash Trees (EPHTs) as a scalable indexing infrastructure to support range queries on top of Distributed Hash Tables. The EPHT is evaluated with real-world phone book data, and it shows that our approach enables efficient distributed phone book applications in a reliable way, without the need for centralized index servers. A comparison with related work shows that this has not been possible using techniques introduced before.

In the next section, we review related work and highlight the problems with current approaches. Then, we present the EPHT algorithm. In the following section, we present a comparison between the EPHT and the PHT algorithm. Then, we evaluate its performance. Finally, we summarize our results and show our conclusions.

**RELATED WORK**

When entries are stored in a Distributed Hash Table, the location of an entry is defined by the hash value of its identifier. A common way to achieve a uniform distribution of the entries among the peers in the DHT is to require the hash function used to calculate the hash value to operate in the Random Oracle Model (Bellare, et al, 1993), i.e. even if two identifiers differ only in a single Bit, the hash values of these identifiers are two independent uniformly distributed random variables.

While this hash function allows for good balancing of the data load in a DHT, it makes range queries very costly. Iterating among a range of identifiers that are lexicographically next to each other means addressing nodes in a random order in the peer-to-peer network. A way to accelerate range queries is to abandon the Random Oracle Model, and to store the entries in lexicographical order. In this section, we discuss three approaches relying on this idea: Skip Graphs (Aspnes, et al, 2003), Squid (Schmidt, et al, 2004), and Mercury (Bharambe, et al, 2004). We point out the difficulties arising with these approaches in scenarios like a distributed phone book.

![Figure 1. Frequency of last names in Munich, Germany](image)