Chapter 21

A Decentralized Directory Service for Peer-to-Peer-Based Telephony

Fabian Stäber  
Siemens Corporate Technology, Germany

Gerald Kunzmann  
Technische Universität München, Germany

Jörg P. Müller  
Clausthal University of Technology, Germany

ABSTRACT

IP telephony has long been one of the most widely used applications of the peer-to-peer paradigm. Hardware phones with built-in peer-to-peer stacks are used to enable IP telephony in closed networks at large company sites, while the wide adoption of smart phones provides the infrastructure for software applications enabling ubiquitous Internet-scale IP-telephony.

Decentralized peer-to-peer systems fit well as the underlying infrastructure for IP-telephony, as they provide the scalability for a large number of participants, and are able to handle the limited storage and bandwidth capabilities on the clients. We studied a commercial peer-to-peer-based decentralized communication platform supporting video communication, voice communication, instant messaging, et cetera. One of the requirements of the communication platform is the implementation of a user directory, allowing users to search for other participants. In this chapter, we present the Extended Prefix Hash Tree algorithm that enables the implementation of a user directory on top of the peer-to-peer communication platform in a fully decentralized way. We evaluate the performance of the algorithm with a real-world phone book. The results can be transferred to other scenarios where support for range queries is needed in combination with the decentralization, self-organization, and resilience of an underlying peer-to-peer infrastructure.

DOI: 10.4018/978-1-60960-603-9.ch021
INTRODUCTION

Structured peer-to-peer overlay protocols such as Chord (Stoica et al. 2001) are increasingly used as part of robust and scalable decentralized infrastructures for communication platforms. For instance, users connect to an overlay network to publish their 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. Registration and lookup of addresses are realized using Distributed Hashtables (DHT).

However, users in such applications 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.

A challenge arises from the non-uniform distribution of people’s names. Figure 1 shows the frequency of last names in the city of Munich, Germany. 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 propose the use of Extended Prefix Hash Trees (EPHTs) as a scalable indexing infrastructure to support range queries on top of Distributed Hash Tables. The EPHT is evaluated by using real-world phone book data; experiments show 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 following section, we review related work and highlight the problems with current approaches. Then, we present the EPHT algorithm, and compare it with the original Prefix Hash Tree (PHT) algorithm. Then, we evaluate its performance by running a series of experiments. 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.