The "Umbrella" Distributed Hash Table Protocol for Content Distribution

Athanasios-Dimitrios Sotiriou
National Technical University of Athens, Greece

Panagiotis Kalliaras
National Technical University of Athens, Greece

INTRODUCTION

During the past few decades, the Internet has blossomed due to the immerse growth of the telecommunication backbone, making it one of the key players in a wide area of fields. Even traditional players such as television or radio are now being challenged by the new entertainment media, the home computer. The increase of share communities such as Weblogs (Drezner & Farrell, 2004) or MySpace (www.myspace.com) and content-sharing software proves that people want to share their content with their global Web community. The need for such content-sharing software is therefore undisputable. Such attempts have been introduced in many ways during the past, with perhaps the most common example being Napster (www.napster.com) and Gnutella (http://gnutella.wego.com).

The technical and ethical issues of these systems proved to be their weak point. Systems that have no central point of control and distribute functions among all users seem better fit for sharing and distributing content. A solution has been proposed in the form of *distributed hash-tables (DHTs)*. This article proposes an alternative architecture for content distribution based on a new DHT routing scheme. The proposed architecture is well structured and self-organized in such a way as to be fault-tolerant and highly efficient. It provides users with content distribution and discovery capabilities on top of an overlay network. The novelty of our proposed architecture lies in its routing table which is maintained by each node and is of constant size, as opposed to other algorithms that are proportional to the network's size (usually O(logN)). All operations in our architecture are of $O(\log_b N)$ steps (entry, publishing, and lookups) and degrade gracefully as up-to-date information of the routing table decreases due to numerous node failures.

BACKGROUND

The firsts to introduce routing algorithms that could be applied to DHT systems were Plaxton, Rajaraman, and Richa (1997). The algorithm was not developed for P2P systems, and thus every node had a neighborhood of O(logN) and inquires resulted in O(logN) steps. It was based on the ground rule of comparing one byte at a time until all bytes of the identifier (or best compromise) were met. Our scheme meets the logarithmic growth of inquiries introduced by Plaxton et al. (1997), even though nodes are not placed within constant distance from each other.

A variation of the Plaxton algorithm was developed by Tapestry (Zhao et al., 2004), properly adjusted for P2P systems (where overall state is not available). The algorithm once again tackles one digit at a time, and through a routing table of $\beta*\log_\beta N$ neighbors routes to the appropriate node, resulting in a search of $\log_\beta N$ maximum steps.

Pastry (Rowstron, 2001) is similar to Tapestry, but added a leaf set of neighbors that the node first checks before referring to the routing table. Also a different neighbor set is maintained for tolerability issues. Each node maintains a neighborhood of $\log_2 bN$ rows with (2^b-1) elements in each row and requires a maximum of $O(\log_2 bN)$ steps for enquires. Proper routing is maintained as long as (L/2) nodes are available in the neighborhood of each node. Once again, the variable size of each node's table (which must be maintained up-to-date) limits the algorithm's scalability.

In Chord (Stoica, Karger, Kaashoek, & Balakrishnan, 2001) a different approach was applied, placing nodes in a circular space and maintaining information only for a number of successor and predecessor nodes through a finger table. Routing is established through forwarding queries to the correct successor based on the identifier. Even though the basic Chord mechanism only requires the knowledge of one successor, modifications where needed in order for the system to be applicable to a robust environment, introducing a finger table of O(logN) size.

Finally, Kademlia (Maymounkov & Mazieres, 2002) bases nodes in a binary-tree through identifiers. Each node of the tree retains information concerning one node from each leaf, other than the one in which it resides. It also differentiates by applying an XOR comparison on identifiers instead of the casual comparison of each bit, adopted by all other algorithms.

SYSTEM ARCHITECTURE

In this section we will give an overview of the Umbrella architecture. The main functions consist of the insertion of nodes, the assignment of keys to corresponding nodes, and the routing mechanisms for three principle operations, namely the insertion of nodes, publication of content, and lookup of keys. Our architecture is based on an overlay network, and thus we assume that node connectivity is both symmetric and transitive.

Hashing Function

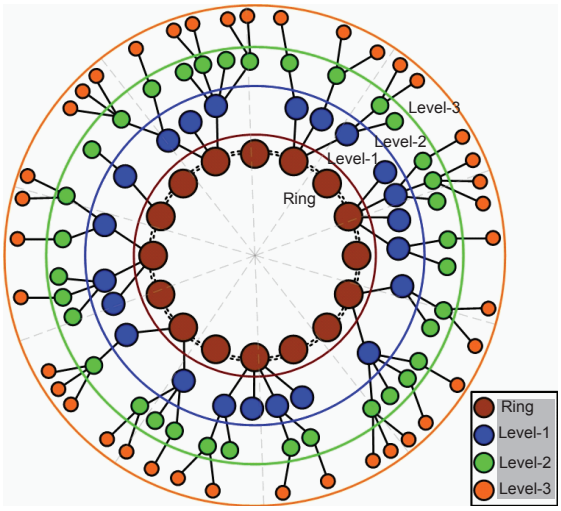
The proposed architecture is based on the creation of an overlay network, where all inserting nodes are identified by a unique code, asserted by applying the SHA-1 (NIST, 1995) hash-function on the combination of IP and computer name, which returns a 160-bit identifier. This hash-function has been proven to distribute keys uniformly in the 160-bit space and thus provides the desired load balancing for both the user space and the content space, as the same function is applied to each content destined for distribution in the system.

Structure Overview

The main objective of the architecture is to insert and retain nodes in a simple and well-structured manner, thus querying and fetching of content is both efficient and fault-tolerant. In addition, each node will need only to retain up-to-date information of a limited, constant number of neighboring nodes, allowing the system to escalate in population of both users and content. Each node is inserted into the system through an existing node, which announces the new entrance. When this procedure has ended successfully, the new node can, having acquired and informed all neighboring nodes, continue to publish all of its content. The publishing procedure is similar to the insertion mechanism, as content is characterized by a number of keys, which after being hashed can be forwarded in the same manner. All keys are published in an existing node whose identifier is the closest match to the key identifier. In a similar fashion, querying is performed by routing the request to the node with the identifier closest to the desired key. If no such node exists, it is assumed that the desired content is not available.

The overlay network is constructed in the form of a loose B-Tree, where each node is placed in a hierarchy tree with a parent node and b child nodes, which in our initial architecture is of the value 16 (in order to classify the 160-bit identifiers to a maximum B-Tree of height ≤ 40). All nodes are placed along the tree structure, without being required to fulfill pre-defined ranges as in a proper B-tree structure, and are responsible for updating their connections with

Figure 1. The Umbrella architecture



neighboring nodes that reside on either the parent, sibling, or child level. Along with obvious connections (parent, child, and sibling level of each node), further links to a limited number of nodes in the near vicinity are kept in record for fault-tolerant operations. Figure 1 illustrates the structure of this loose B-Tree. Routing in the umbrella protocol is simple and constitutes the forwarding of messages to either a parent or child node until the appropriate node is reached. In the rest of the article, with the term-appropriate node we will refer to either the exact or closest match alike.

Key Mapping

Each level n of the structure is capable of withholding b^{n+1}nodes. Each node has a unique parent node, which is always one level higher, and a maximum of b children at a lower level. The Umbrella overlay network is configured with the following simple rule. The relation between a parent node at level n and a child node (which must by default reside on level n+1) is defined as such and only such that:

- The n+1 first (from left to right) digits of the parent's identifier are equal to the corresponding digits of the child's identifier.
- The n+2 digit of the child's identifier determines the child's position in the parent's child list. Thus all children of the same parent share the first n+1 digits and all differ in the n+2 digit.

The above simple rule is obeyed by all nodes entering the Umbrella overlay network, with only the exception of the first node that actually initiates the network and is considered to be positioned at level -1. As already stated, the SHA-1 hash