Making Chord go mobile

Stefan Zöls[1], Rüdiger Schollmeier[1], Wolfgang Kellerer[2]

[1] Lehrstuhl für Kommunikationsnetze
Technische Universität München
Arcisstr. 21
D-80333 München
{stefan.zoels, ruediger.schollmeier}@tum.de

[2] Future Networking Lab
DoCoMo Communications Laboratories
Landsberger Str. 312
D-80687 München
kellerer@docomolab-euro.com

Abstract: The Chord protocol is a structured Peer-to-Peer (P2P) protocol based on distributed hash tables (DHT). By using hash keys to identify the nodes in the network and also the shared objects, it can significantly reduce the signaling overhead in a P2P overlay network, as flooding of query messages can be avoided. However, when nodes join or leave the Chord network, object references have to be rearranged in order to maintain the hash key mapping rules. This leads to high maintenance traffic, especially when nodes stay in the Chord ring only for a short time. In mobile environments, the resources and data rates of mobile devices are limited, so the maintenance traffic generated by shifting object references may cause problems when using Chord in a mobile scenario. In this work, we present a solution to the problem of frequent joins and leaves of nodes. By distinguishing static nodes and temporary nodes, we can decrease the maintenance traffic generated by shifting object references significantly.

1 Introduction

A very promising approach to solve the core problem of P2P systems – how to find a node sharing the required object and thereby to generate as little traffic load as possible – is the concept of structured P2P networks based on Distributed Hash Tables (DHT). One realization of such systems is Chord [1]. It is a scalable, distributed lookup protocol that uses a hash function to determine a node's m-bit identifier from its IP address. The same hash function is used to produce an identifier for every shared object. The identifier of a shared object is called its key k. The Chord nodes are ordered in an identifier circle modulo 2^m. Every key k is assigned to the first node in the ring whose identifier is equal to or follows k. This node stores references to all shared objects in the network whose key precedes the identifier of the node. Thus flooding of query messages can be avoided, as the query can be routed directly to the responsible node.

Whenever a node joins the Chord ring, it has to receive all object references according to the keys it is responsible for from its succeeding node. Again when the node leaves the network, all object references have to be transferred to the succeeding node. This can result in a high traffic volume, especially when there are many shared objects (and therefore object references) in the network and when nodes participate in the overlay network only for a short period of time. In a mobile scenario, a high rate of joining and leaving nodes (= churn rate) is common, e.g. because of high costs for mobile data transfer. Besides high churn rates, also the relatively low transmission data rates and the limited resources of mobile devices demand for a low maintenance overhead.
In this paper, we propose a modification to the Chord protocol that aims at the problems in mobile, wireless scenarios. We use two different node classes – static nodes and temporary nodes – to reduce significantly the traffic load that is caused by shifting object references.

2 System architecture

Chord considers all nodes in the network to be equal. However, this equality between nodes does not exist in realistic scenarios. Thus we modify the conventional Chord protocol by subdividing all participating nodes into two classes: static nodes and temporary nodes:

- Static nodes are highly-available nodes that form a quasi-permanent part of the Chord ring. They usually remain in the network for a long time and have high data rate connections to other static nodes as well as a large storage capacity. All object references in the network are stored at static nodes.
- Temporary nodes are all nodes that do not form a quasi-permanent part of the Chord network. In most cases, temporary nodes join the network only for a short period of time to find some particular pieces of information. Temporary nodes do not store object references. When a temporary node joins the network, all object references according to the keys it is responsible for remain with its closest static successor, in order to avoid shifting object references. If temporary nodes receive a request for an assigned key, they will forward the request to their closest static successor that stores all object references for this key.

As we distinguish static and temporary nodes, we ensure that only static, quasi-permanent nodes store object references. Thus nodes that join the network only for a short time do not cause traffic load due to shifting object references, beyond transferring their own object references to static nodes. Furthermore, temporary nodes with limited storage capacity are prevented from storing a lot of object references.
As static nodes store all object references available in the network, a high volume of data has to be transferred when a static node joins or leaves the network. However, this usually does not cause any problems as such joins or leaves are assumed to occur infrequently and the data is transmitted over more stable and usually broadband links between the static nodes.

3 Election of static nodes

Static nodes could either be set up manually (infrastructure static node) or be elected from temporary nodes when certain conditions are fulfilled. In the following, we focus on the latter option. When a node joins the (modified) Chord network, it first is classified as a temporary node. To be promoted to a static node, a temporary node must fulfill the following three criteria:

- The node must have enough hardware resources, i.e. processing power, random access memory and hard disk storage capacity, in order to handle the increased load on a static node.
- The node must have a broadband date rate connection to the internet, as the traffic load for static nodes is significantly higher than the traffic load for temporary nodes.
- Based on the analysis of Maymounkov and Mazieres [2], we state that the longer a node is actively participating in an overlay network, the higher is the probability that the node remains in the overlay network. For this reason the uptime history of a node is the third criteria when deciding if it is elected as a static node.

An algorithm regularly checking the fulfillment of the above requirements is part of the software running on each peer, which performs a self-election if the conditions are met.

4 Analytical evaluation

With our analytical considerations we prove the ability of our Chord modification to reduce significantly the maintenance traffic that is generated by the new assignment of object references when nodes join or leave the overlay network. In the first part of this section we calculate the average traffic load per node in the conventional Chord protocol. Afterwards, we compare the result with the traffic load that is generated in our modified Chord network.

Assume a conventional Chord network with the following parameters: The average number of object references per node is R_n, and the average session length of all nodes is T. As object references have to be shifted when a node joins the ring and also when it leaves again, the transfer rate per node amounts to $2/T$. Assuming that b is the average size of an object reference, the average traffic load for a single node λ_n is given by

$$\lambda_n = \left(2 \cdot R_n \cdot b\right)/T$$

From this formula, we can conclude the advantage of our modification in comparison to the conventional Chord protocol. According to our system architecture, the traffic load that is caused by temporary nodes is zero, as temporary nodes do not store object references. Additionally, static nodes – which store all object references – usually have a high session length T_{Static}. Thus the average traffic load per node can be decreased significantly, as shown below.

Assume a modified Chord network whereof $x \cdot 100\%$ of all nodes are static and $(1-x) \cdot 100\%$ of the nodes are temporary, with $0 < x < 1$. The static nodes have an average session length which is α times the average session length of a conventional Chord node:

$$T_{Static} = \alpha \cdot T$$

The session length of the temporary nodes can be negligible, because temporary nodes do not store object references and therefore do not generate traffic load by shifting them. In summary, every static node stores R_n / x object references that have to be shifted when a static node joins and leaves the overlay network. Thus the transfer rate of a static node is given by $2/T_{Static} = 2 / (\alpha \cdot T)$. This results in an average traffic load per static node of

$$\lambda_{Static} = (2 \cdot R_n \cdot b) / (x \cdot \alpha \cdot T) = (1/(x \cdot \alpha)) \cdot \lambda_n$$

As stated above, the traffic load of temporary nodes $\lambda_{Temporary}$ is zero. Resulting, we have an average traffic load for all nodes in the modified Chord network of

$$\lambda_{n,modified} = x \cdot \lambda_{Static} + (1 - x) \cdot \lambda_{Temporary} = x \cdot \lambda_{Static} = (1/\alpha) \cdot \lambda_n$$

Thus we can state that the average traffic load per node that is generated by shifting object references can be decreased by a factor of $1/\alpha$ in comparison to a conventional Chord network, with $\alpha = T_{Static} / T$. This means that the traffic load reduction is direct proportional to the fraction of the length of an average session to the average session length of a static node.

5 Conclusion and Future Work

In this work we proposed a modified architecture of Chord that shows a good performance especially in unstable, e.g. mobile environments where high churn rates of the participating nodes must be expected. The architecture extends Chord by two different node classes, namely static nodes and temporary nodes. Therefore we can reduce the maintenance traffic that is generated by shifting object references significantly, as proven by our analytical evaluation.
Currently we are working on the implementation of our Chord modification in ns-2 [3]. Thus we want to verify our analytical results, determine the static node election parameters and be able to evaluate the performance of our modification in a simulated mobile environment.

References

[1] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications" presented at ACM SIG-COMM Conference, 2001.

[2] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric" presented at International Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.

[3] ns-2, "The Network Simulator ns-2 Homepage", http://www.isi.edu/nsnam/ns/