

Cost-Based Analysis of Hierarchical DHT Design

Stefan Zoels¹, Zoran Despotovic², Wolfgang Kellerer²

¹*Institute of Communication Networks, Munich University of Technology, Germany*

²*DoCoMo Communications Laboratories Europe, Munich, Germany*

stefan.zoels@tum.de, {despotovic, kellerer}@docomolab-euro.com

Abstract

Flat DHT architectures have been the main focus of the research on DHT design so far. However, there have been also a number of works proposing hierarchical DHT organizations and pointing their advantages. They mostly rely on the intuitive understanding that hierarchy is desirable in any complex system. In this paper we formalize this intuition within a general cost-based framework. We provide a cost model of a specific hierarchical DHT organization composed of superpeers and leafnodes, and show that the costs of running the network are not necessarily minimized for flat DHT organization, providing thus a formal motivation for hierarchical DHTs. We further hint on what distributed algorithms can be applied in practice to reach optimal operating point of the network.

1. Introduction

Most of current research on Distributed Hash Table (DHT) design concentrates on so-called flat DHT designs, where all participating peers are considered equal in functionality. Chord [1], Pastry [2], Kademlia [3] or P-Grid [4] are some examples of flat DHT designs. Recently, however, there have been a number of works presenting advantages of hierarchical DHT designs. [5] points out better fault isolation, more effective bandwidth utilization, and better adaptation to the underlying physical network as the main reasons for choosing hierarchical systems. [6] demonstrates that hierarchical systems offer a reduction of the lookup path length. Further, in our earlier paper [7] we have shown the benefits of a hierarchical DHT design for mobile environments, characterized by high churn rates, high failure probabilities and resource-constraint mobile peers.

In this paper we take a more general view by considering the Peer-to-Peer (P2P) network as a whole having its own costs of running. Based on the cost

model of [8], we calculate the costs in one specific type of hierarchical systems in which so-called superpeers run a DHT protocol (Chord in this case) and at the same time serve as proxies for so-called leafnodes. The flat DHT design can be viewed as a special case where the fraction of leafnodes equals zero.

Our analysis shows that, depending on the properties of the constituent peers, costs can be minimized for a non-zero number of leafnodes. In such cases flat designs are not an optimal DHT design solution. Instead, there are one or more optimal points corresponding to true hierarchical designs with non-zero leafnodes and superpeers.

Our motivation to consider this type of hierarchical systems was driven by our wish to provide a P2P solution for highly heterogeneous environments, i.e. with large variations among the performances of the involved peers. However, we emphasize that the analysis below holds for homogeneous settings too, where peers have similar capabilities.

Note also that we do not claim that the specific hierarchical design we consider in this paper is optimal across all possible hierarchical P2P systems. The purpose of the paper and our cost-based analysis is rather to formally investigate whether and when hierarchical designs are better than flat designs. In this way we want to provide strong justifications as well as an analytical framework for investigating hierarchical P2P systems in general, and determining optimal values of the involved parameters in particular. Generalizations of the work presented in this paper targeting any general hierarchical system are an important item of our future work.

The rest of the paper is structured as follows. Section 2 introduces the hierarchical system design that underlies the analysis in this work. In Section 3 we derive expressions for all costs generated in the analyzed overlay network and show that total network costs increase with decentralization. Section 4 evaluates the costs of centralization, provides a methodology

to determining optimal superpeer ratios and gives an illustrative example. In Section 5 we describe how these optima can be discovered and maintained in a distributed fashion. Section 6 concludes and gives an outlook for future work.

2. System Architecture

The system architecture we analyze in this paper defines two different classes of peers: superpeers and leafnodes. Every superpeer acts as a proxy for its leafnodes. The leafnodes communicate, apart from uploads and downloads, only with their superpeer. In contrast, superpeers establish a structured DHT-based overlay in form of a Chord ring.¹ Figure 1 depicts the architecture of the analyzed overlay network.

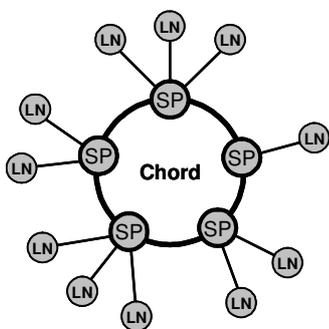


Figure 1: Hierarchical P2P overlay network with superpeers forming a Chord ring and leafnodes attached to them

Our wish to provide a P2P solution for heterogeneous environments led us to select this form of hierarchical systems. Heterogeneous environments are normally characterized by high differences among performances of the involved peers, e.g. highly mobile unreliable peers versus static and reliable ones. Intuitively, the above architecture would fit better these settings than a flat DHT design if reliable high performance peers take the role of superpeers while low performance peers operate as leafnodes. But, as we will see below, it does make sense to propose the above hierarchical architecture even in homogeneous settings, where the differences among peers are negligible.

Tasks Performed by Peers. The tasks performed by the two groups of peers can be briefly described as follows.

¹ Although we use Chord here as DHT between the superpeers, any other structured P2P protocol could also be used. We believe this would result in only minor changes for the analysis in Section 3.

Leafnodes maintain only an overlay connection to their superpeer. To be able to recognize and react to a superpeer failure, they periodically run a simple PING/PONG algorithm. Moreover, they store a list containing other available superpeers in the system, in order to be able to rejoin the overlay network after a superpeer failure.

In contrast, superpeers perform multiple other tasks. We assume that on joining the network a leafnode transfers its list of pointers to the objects it shares to its corresponding superpeer. The superpeer then inserts these references into the overlay network and acts as their owner. When a leafnode performs a lookup, e.g. to insert or query for an object, the superpeer it is connected to resolves the lookup by using the search functionality of the Chord overlay, determines the responsible superpeer (based on the object's key), and forwards the result to the leafnode. Additionally, since superpeers establish a conventional Chord ring, they periodically run Chord's STABILIZE and FIXFINGER algorithms for overlay maintenance. We analyze here the original STABILIZE algorithm that is proposed in [1] and do not consider extensions of [9]. We use a slight modification of the FIXFINGER algorithm that we will explain in Section 3.2. Finally, the superpeers refresh periodically all references they maintain in order to keep them up-to-date.

3. Cost-Based Analysis of the Superpeer Ratio

An important design parameter for the architecture proposed in Section 2 is the ratio between superpeers and the total number of peers; we denote it α and call it simply the superpeer ratio. In this section, we analyze how the total network traffic costs depend on this ratio. To start, we make a number of definitions, introduce our notation and explain our assumptions:

- Number of Peers: N
- Number of superpeers: $N_{SP} = \alpha \cdot N$
- Number of leafnodes: $N_{LN} = (1 - \alpha) \cdot N$
- Costs for peer k for sending a message: c_k^s
- Costs for peer k for receiving a message: c_k^r
- Lookup rate of peer k : $r_{LKP, k} = 1 / T_{LKP, k}$
- Number of shared objects provided by peer k : f_k

We assume that every message sent/received by peer k creates same costs $c_k^{s/r}$, regardless of the message size. We further assume that the system is in a steady state, i.e. no churn occurs, that every superpeer knows the shared objects of its leafnodes at any time, and that the superpeers' Chord ring is fully populated.

We begin our analysis with the total traffic costs for a hierarchical overlay network with the above parameters. The total traffic costs C consist of costs for lookup traffic (see Section 3.1) and of costs for maintenance traffic (see Section 3.2):

$$C = C_{LKP} + C_{MAINT}$$

3.1. Lookup Traffic Costs

When regarding lookups, we differentiate between lookup costs for superpeers and lookup costs for leafnodes. The total costs for lookup traffic C_{LKP} then consist of lookup costs for all leafnodes $C_{LKP, LN}$ and of lookup costs for all superpeers $C_{LKP, SP}$:

$$C_{LKP} = C_{LKP, LN} + C_{LKP, SP}$$

Lookup Costs for Leafnodes. When a leafnode performs a lookup, it sends a QUERY message to its superpeer. The superpeer resolves the lookup in the superpeers' Chord overlay and returns the result to the leafnode. Therefore, the number of sent and received messages m^s and m^r for any leafnode j is given by

$$m_{LNj}^s = m_{LNj}^r = r_{LKP, LNj}$$

To obtain the lookup costs for leafnode j we multiply the number of messages with the respective costs:

$$\begin{aligned} C_{LKP, LNj} &= c_{LNj}^s \cdot m_{LNj}^s + c_{LNj}^r \cdot m_{LNj}^r = \\ &= (c_{LNj}^s + c_{LNj}^r) \cdot r_{LKP, LNj} \end{aligned}$$

The total lookup costs for leafnodes are thus given by

$$C_{LKP, LN} = \sum_{j=1}^{N_{LN}} C_{LKP, LNj} = \sum_{j=1}^{N_{LN}} (c_{LNj}^s + c_{LNj}^r) \cdot r_{LKP, LNj}$$

Lookup Costs for Superpeers. To calculate lookup costs for superpeers, we separate costs generated by superpeer lookups and costs generated by leafnode lookups. Lookups performed by any superpeer i generate a mean amount of

$$m_{SPi} = r_{LKP, SPi} \cdot \log_2 N_{SP} = r_{LKP, SPi} \cdot \log_2 \alpha N$$

messages in the superpeers' Chord ring. (Note that the average number of hops to resolve a lookup in a fully populated ring is $\frac{1}{2} \log_2 N_{SP}$, and that two messages are needed per hop.) Lookups performed by any leafnode j generate a mean amount of

$$m_{LNj} = r_{LKP, LNj} \cdot [\log_2 N_{SP} + 1] = r_{LKP, LNj} \cdot [\log_2 \alpha N + 1]$$

messages sent and received by superpeers, because the leafnode's superpeer receives the lookup request (one additional message received) and, after resolving the lookup in the superpeers' Chord overlay, sends the result back to the leafnode (one additional message sent). Thus, the total number M of messages that are

sent and received by all superpeers due to lookups accumulates to

$$\begin{aligned} M &= \sum_{i=1}^{N_{SP}} m_{SPi} + \sum_{j=1}^{N_{LN}} m_{LNj} = \\ &= \sum_{i=1}^{N_{SP}} r_{LKP, SPi} \cdot \log_2 \alpha N + \sum_{j=1}^{N_{LN}} r_{LKP, LNj} \cdot [\log_2 \alpha N + 1] = \\ &= \overline{r_{LKP, SP}} \cdot N_{SP} \cdot \log_2 \alpha N + \overline{r_{LKP, LN}} \cdot N_{LN} \cdot [\log_2 \alpha N + 1] = \\ &= N \cdot \left[\overline{r_{LKP, SP}} \cdot \alpha \cdot \log_2 \alpha N + \overline{r_{LKP, LN}} \cdot (1 - \alpha) \cdot [\log_2 \alpha N + 1] \right] \end{aligned}$$

with $\overline{r_{LKP, SP}}$ and $\overline{r_{LKP, LN}}$ being the mean lookup rates of all superpeers and leafnodes, respectively.

We assume that every superpeer manages an equal number of leafnodes² and that the ID space is fully populated. Hence every superpeer processes an equal proportion of M and the number of sent and received messages m^s and m^r for any superpeer i is given by

$$\begin{aligned} m_{SPi}^s &= m_{SPi}^r = \frac{M}{N_{SP}} = \frac{M}{\alpha \cdot N} = \\ &= \overline{r_{LKP, SP}} \cdot \log_2 \alpha N + \overline{r_{LKP, LN}} \cdot \frac{1 - \alpha}{\alpha} \cdot [\log_2 \alpha N + 1] \end{aligned}$$

To obtain the lookup costs for superpeer i we multiply the number of messages with the respective costs:

$$\begin{aligned} C_{LKP, SPi} &= c_{SPi}^s \cdot m_{SPi}^s + c_{SPi}^r \cdot m_{SPi}^r = \\ &= (c_{SPi}^s + c_{SPi}^r) \cdot \left(\overline{r_{LKP, SP}} \cdot \log_2 \alpha N + \overline{r_{LKP, LN}} \cdot \frac{1 - \alpha}{\alpha} \cdot [\log_2 \alpha N + 1] \right) \end{aligned}$$

The total lookup costs for superpeers are thus given by

$$\begin{aligned} C_{LKP, SP} &= \sum_{i=1}^{N_{SP}} C_{LKP, SPi} = \\ &= \left(\overline{r_{LKP, SP}} \cdot \log_2 \alpha N + \overline{r_{LKP, LN}} \cdot \frac{1 - \alpha}{\alpha} \cdot [\log_2 \alpha N + 1] \right) \cdot \sum_{i=1}^{N_{SP}} (c_{SPi}^s + c_{SPi}^r) \end{aligned}$$

By summing up the lookup costs for leafnodes and the lookup costs for superpeers, we get the total costs for lookup traffic C_{LKP} .

3.2. Maintenance Traffic Costs

Maintenance traffic is generated by the PING/PONG algorithm between leafnodes and their superpeer and

² We believe this can be achieved by using an appropriate load balancing algorithm.

by the STABILIZE, FIXFINGER and REPUBLISH algorithms in the superpeers' Chord overlay:

$$C_{MAINT} = C_{PING} + C_{STAB} + C_{FIX} + C_{REP}$$

PING Costs. Every leafnode runs the PING/PONG algorithm periodically every T_{PING} seconds. It sends a PING message to its superpeer, and the superpeer answers with a PONG message. Therefore, the number of sent and received PING/PONG messages m^s and m^r for any leafnode j is given by

$$m^s_{LNj} = m^r_{LNj} = 1 / T_{PING}$$

To obtain the PING costs for leafnode j we multiply the number of messages with the respective costs:

$$C_{PING, LNj} = (c^s_{LNj} + c^r_{LNj}) / T_{PING}$$

As in the previous section, we assume an appropriate load balancing algorithm that spreads all leafnodes uniformly over all superpeers. Thus, the number of sent and received PING/PONG messages m^s and m^r for any superpeer i is given by

$$m^s_{SPi} = m^r_{SPi} = \frac{1}{T_{PING}} \cdot \frac{N_{LN}}{N_{SP}} = \frac{1}{T_{PING}} \cdot \frac{1-\alpha}{\alpha}$$

To obtain the PING costs for superpeer i we multiply the number of messages with the respective costs:

$$C_{PING, SPi} = (c^s_{SPi} + c^r_{SPi}) \cdot \frac{1}{T_{PING}} \cdot \frac{1-\alpha}{\alpha}$$

The total PING costs for the overlay network are thus given by

$$\begin{aligned} C_{PING} &= \sum_{j=1}^{N_{LN}} C_{PING, LNj} + \sum_{i=1}^{N_{SP}} C_{PING, SPi} = \\ &= \frac{1}{T_{PING}} \cdot \left[\sum_{j=1}^{N_{LN}} (c^s_{LNj} + c^r_{LNj}) + \frac{1-\alpha}{\alpha} \cdot \sum_{i=1}^{N_{SP}} (c^s_{SPi} + c^r_{SPi}) \right] \end{aligned}$$

STABILIZE Costs. Every superpeer runs the STABILIZE algorithm periodically every T_{STAB} seconds. STABILIZE requires three messages: The initiating superpeer sends a REQUESTPREDECESSOR message to its successor, the successor responds with a RESPONSEPREDECESSOR message and finally the initiating peer sends a NOTIFY message. Therefore, the number of sent and received STABILIZE messages m^s and m^r for any superpeer i is given by

$$m^s_{SPi} = m^r_{SPi} = 3 / T_{STAB}$$

To obtain the STABILIZE costs for superpeer i we multiply the number of messages with the respective costs:

$$C_{STAB, SPi} = (c^s_{SPi} + c^r_{SPi}) \cdot 3 / T_{STAB}$$

The total STABILIZE costs for the overlay network are thus given by

$$C_{STAB} = \sum_{i=1}^{N_{SP}} C_{STAB, SPi} = \frac{3}{T_{STAB}} \cdot \sum_{i=1}^{N_{SP}} (c^s_{SPi} + c^r_{SPi})$$

FIXFINGER Costs. Every superpeer runs the FIXFINGER algorithm periodically every T_{FIX} seconds for each of its $\log_2 \alpha N$ fingers (assuming a fully populated ID space). Fixing a finger usually corresponds to a Chord lookup of the finger's ID. However, we use here an improved FIXFINGER algorithm that sends a PING message to a finger peer, and initiates a finger lookup only when no PONG message is received or when the finger peer indicates a new peer being responsible for this finger ID. Resulting, finger lookups can be avoided when the system is in a steady state, and the number of sent and received FIXFINGER messages m^s and m^r for any superpeer i is given by

$$m^s_{SPi} = m^r_{SPi} = \log_2 \alpha N \cdot 2 / T_{FIX}$$

Here we assume that every superpeer receives the same number of FIXFINGER PINGS from other superpeers as it sends to them. (This is the case for a fully populated ID space.) To obtain the FIXFINGER costs for superpeer i we multiply the number of messages with the respective costs:

$$C_{FIX, SPi} = (c^s_{SPi} + c^r_{SPi}) \cdot \log_2 \alpha N \cdot 2 / T_{FIX}$$

The total FIXFINGER costs for the overlay network are thus given by

$$C_{FIX} = \frac{\log_2 \alpha N \cdot 2}{T_{FIX}} \cdot \sum_{i=1}^{N_{SP}} (c^s_{SPi} + c^r_{SPi})$$

REPUBLISH Costs. Every superpeer runs the REPUBLISH algorithm periodically every T_{REP} seconds for every shared object of the superpeer itself and its leafnodes. Republishing a shared object corresponds to a Chord lookup of the object's ID. Therefore, it generates on average $\log_2 N_{SP}$ messages. For the analysis of REPUBLISH costs, we assume that every superpeer manages the same number of objects shared by the superpeer itself and its leafnodes, i.e.

$$\text{Shared objects managed by one superpeer} = \frac{1}{N_{SP}} \cdot \sum_{k=1}^N f_k$$

Thus, republishing shared objects generates a total number of REPUBLISH messages M sent and received by all superpeers of

$$M = N_{SP} \cdot \frac{1}{T_{REP}} \cdot \frac{\sum_{k=1}^N f_k}{N_{SP}} \cdot \log_2 N_{SP} = \frac{\sum_{k=1}^N f_k}{T_{REP}} \cdot \log_2 \alpha N$$

As we assume a fully populated Chord ring, every superpeer processes an equal proportion of M . There-

fore, the number of sent and received REPLICATE messages m^s and m^r for any superpeer i is given by

$$m^s = m^r = \frac{M}{N_{SP}} = \frac{\sum_{k=1}^N f_k}{T_{REP} \cdot \alpha \cdot N} \cdot \log_2 \alpha N$$

To obtain the REPLICATE costs for superpeer i we multiply the number of messages with the respective costs:

$$C_{REP, SPi} = (c_{SPi}^s + c_{SPi}^r) \cdot \frac{\sum_{k=1}^N f_k}{T_{REP} \cdot \alpha \cdot N} \cdot \log_2 \alpha N$$

The total REPLICATE costs for the overlay network are thus given by

$$C_{REP} = \frac{\sum_{k=1}^N f_k}{T_{REP} \cdot \alpha \cdot N} \cdot \log_2 \alpha N \cdot \sum_{i=1}^{N_{SP}} (c_{SPi}^s + c_{SPi}^r)$$

3.3. Example: Homogeneous Peers

With the expressions developed in the previous sections we can compute the total network costs for the proposed system architecture with N_{SP} superpeers and N_{LN} leafnodes, and their specific values for message costs, lookup rate and number of shared objects.

The following example considers a small overlay network with 100 homogeneous peers $p_0 \dots p_{99}$. Every peer p_k ($k \in \{0 \dots 99\}$) has the same message costs $c_k^s = c_k^r = 1$, same number of shared objects $f_k = 20$ and same lookup rates $r_{LKP, k} = 1/30[s]$. Further we define system-wide timer values $T_{PING} = 10[s]$, $T_{STAB} = 10[s]$, $T_{FIX} = 60[s]$ and $T_{REP} = 300[s]$. Therefore, the total network costs are given by

$$C = C_{LKP, LN} + C_{LKP, SP} + C_{PING} + C_{STAB} + C_{FIX} + C_{REP}$$

with

$$C_{LKP, LN} = 20/3 \cdot (1 - \alpha)$$

$$C_{LKP, SP} = 20/3 \cdot [\log_2 100 \alpha + 1 - \alpha]$$

$$C_{PING} = 40 \cdot (1 - \alpha),$$

$$C_{STAB} = 60 \alpha$$

$$C_{FIX} = 20/3 \cdot \alpha \log_2 100 \alpha$$

$$C_{REP} = 40/3 \cdot \log_2 100 \alpha$$

Figure 2 shows the total network costs as a function of the superpeer ratio $\alpha = N_{SP} / N$.

Obviously, a centralized overlay network with only one superpeer (an index server) generates the lowest network traffic costs, because only lookup and PING/PONG messages are exchanged between the superpeer and its leafnodes. Moreover, we notice increased network traffic costs if the number of superpeers increases, mostly caused by maintenance

traffic in the superpeers' Chord ring. The highest network costs can be observed if all peers are superpeers and are thus forming a conventional Chord overlay. This is the price for a completely flat overlay topology.

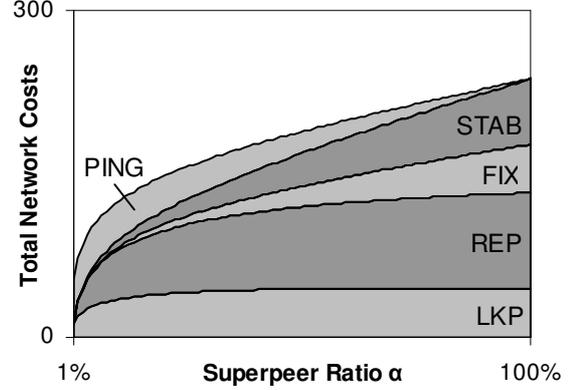


Figure 2: Total network costs against superpeer ratio α

4. Costs of Centralization

Based on the conclusions made in the previous paragraph it is obvious that, with regard to the total network traffic costs, a centralized overlay topology is the optimal solution and that a small number of superpeers is preferable. However, such centralization imposes certain costs on individual peers. Every peer may have a maximum value of costs, i.e. a cost limit it is poised to spend for participation. In our analyzed system, most of the arising costs account for superpeers, and the fewer superpeers there are the higher the costs every superpeer has to bear. As a result, superpeers may be overloaded, i.e. bear higher costs than their cost limits, and thus we face the risk of breaking system stability. From this point of view a high number of superpeers is preferable, as shown below.

4.1. Individual Costs for Superpeers

The costs for any superpeer i consist of all costs determined in Section 3, i.e. lookup costs, PING costs, STABILIZE costs, FIXFINGERS costs, and REPLICATE costs:

$$C_{SPi} = C_{LKP, SPi} + C_{PING, SPi} + C_{STAB, SPi} + C_{FIX, SPi} + C_{REP, SPi}$$

If we apply this formula to our homogeneous example scenario of Section 3.3, the costs per superpeer are given by

$$C_{SPi} = \frac{1}{15\alpha} \cdot [(\alpha + 3) \cdot \log_2 100 \alpha + 4 + 5\alpha]$$

Figure 3 depicts the costs per superpeer for different superpeer ratios $\alpha = N_{SP} / N$. As expected, the costs per superpeer decrease as the number of superpeers increases.

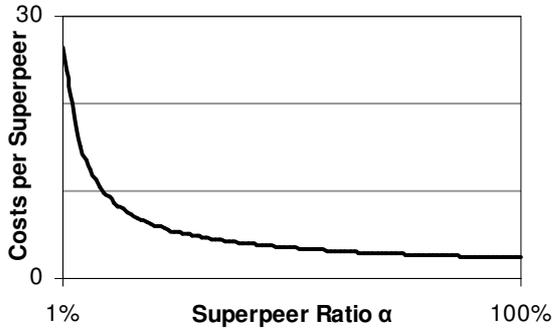


Figure 3: Costs per superpeer against superpeer ratio α

4.2. Load Factor

To determine an optimal value for the number of superpeers taking into account their individual costs, we define a load factor LF for every participating peer. The load factor of peer k specifies the ratio between the costs C_k for peer k in a given scenario and the maximum value of costs peer k is willing to accept (i.e. k 's cost limit $C_{k, max}$):

$$LF_k = \frac{C_k}{C_{k, max}}$$

For evaluating a given scenario we focus on the highest load factor HLF that can be observed across all peers:

$$HLF = \max(LF_k) \forall k$$

Let us again have a look at our small example scenario. If we define a cost limit for every peer $C_{k, max} = 10, k \in \{0..99\}$, we will get the dependence of the HLF on the superpeer ratio $\alpha = N_{SP} / N$ as shown in Figure 4. Because we have a fully homogeneous setting in this example the shown curve is smooth and basically coincides with the one from Figure 3.

Nevertheless, Figure 4 shows three interesting facts. First of all, if the ratio of superpeers is below 10%, we will obtain $HLFs$ of more than 100%. Consequently, one or more peers are overloaded, because they bear higher costs than they accept. In general, $HLFs$ higher than 100% should be avoided in order to ensure system stability.

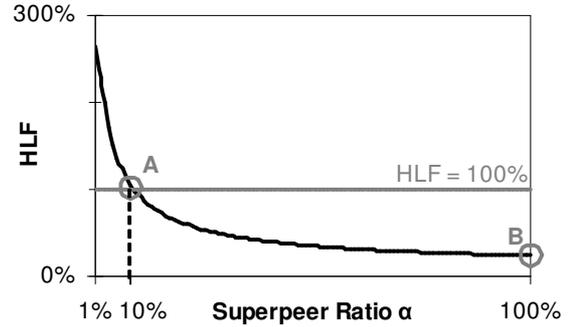


Figure 4: Highest Load Factor against superpeer ratio α

Secondly, if we want to minimize total network costs without overloading any peer in the system, a superpeer ratio of 10% (point A) should be chosen. Point A can be seen as the optimal value for α from the network's point of view.

Thirdly, we can also focus on minimizing the HLF (point B), i.e. costs for the highest loaded peer are minimized in relation to the peer's cost limit. In our small example scenario, the HLF is minimized if all peers are superpeers and build a conventional Chord overlay. We regard point B as an optimal value for α from the peers' point of view.

We believe that the HLF can play an important role in determining optimal operating points of a network in a more general sense. As a simple example, assume that the system designer has to strike a balance between minimizing the total network costs and minimizing the HLF in the system. For this purpose, he can specify a weighted average

$$c = \omega_1 \cdot \text{Total Network Costs} + \omega_2 \cdot HLF,$$

and define an optimal ratio $\alpha_{opt} = \text{argmin}(c)$. As mentioned above, setting $\omega_1 = 1$ and $\omega_2 = 0$ results in choosing point A in the example from Figure 4 as the optimal value, while setting $\omega_1 = 0$ and $\omega_2 = 1$ yields point B as the optimum.

Further, the load factor of a peer can be viewed as an indicator of the probability that the peer will fail. Consequently, distributions of load factors across the peers may be indicative of what we could generally call the quality of the overlay network. We do not attempt here to define the concept of overlay network quality, but we firmly believe that this can be done by extending the analysis of Gummadi et al. [10] by the above ideas. A serious difficulty in applying this would be the necessity to determine the distributions of load factors across the peers. However, we believe that in practice the situation is not so complicated, as the example in the following section illustrates.

4.3. Example: Heterogeneous Peers

Consider an overlay network with 90,000 peers in total, of which 30,000 peers are DSL subscribers, 30,000 peers are UMTS-connected PDAs, and 30,000 peers are built of GPRS-connected cell phones. All peers are modeled according to the parameters given in Table 1.

Table 1: Modeling of peers

	DSL	UMTS	GPRS
Lookup rate	1 / 60s	1 / 30s	1 / 30s
Shared objects	500	100	50
Upstream [kbit/s]	256	92	50

To define a cost limit for every peer, we here focus on the upstream bit rate of every participating peer. We set $C_{k,max}$ to 10% of the upstream bit rate of peer k . Consequently, DSL peers provide 25.6 kbit/s for P2P network participation, UMTS peers 9.2 kbit/s, and GPRS peers 5.0 kbit/s. In addition, we set message costs $c_k^s = 1$ and $c_k^r = 0$ for all k , thus taking only sent messages into account.³ Further system parameters are defined according to Table 2.

Table 2: System parameters

Parameter	Value
T_{PING}	10s
T_{STAB}	10s
T_{FIX}	60s
T_{REP}	600s
Mean message size	1000 Bits

When increasing the number of superpeers in the following analysis, we first promote DSL peers to superpeers. For scenarios with more than 30,000 superpeers we also take UMTS peers. Finally, if we have more than 60,000 superpeers, even GPRS peers will act as superpeers. Figures 5 and 6 show the total network costs and the highest load factor for the given overlay network against the number of superpeers in the system.

As expected, we again notice increasing total network costs for an increasing number of superpeers. This corresponds directly to our argumentation in Section 3.3.

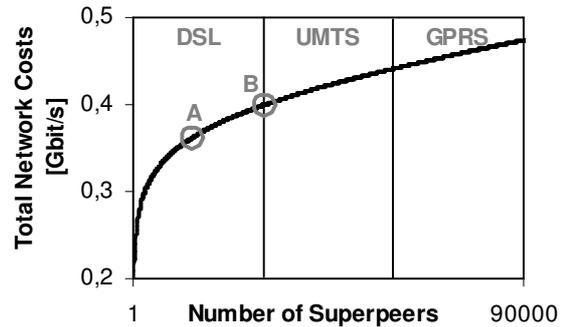


Figure 5: Total network costs

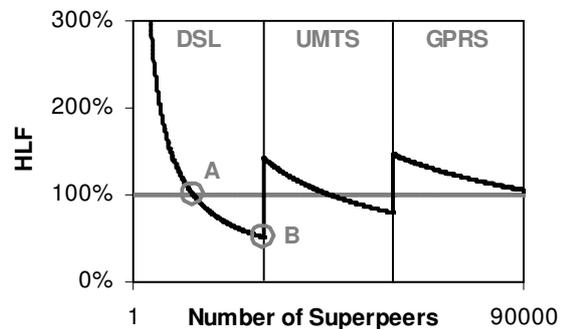


Figure 6: Highest Load Factor

With regard to the *HLF* diagram, we can draw the following conclusions: When the number of superpeers is less than 13,775 (number of superpeers in point A), we observe *HLFs* of more than 100%, i.e. overloaded DSL superpeers. This is also the case for a number of superpeers between 30,001 and 45,291 (UMTS superpeers overloaded) and whenever GPRS peers act as superpeers. As stated above these superpeers ratios should be avoided to ensure system stability. Interestingly, this also implies that neither a centralized system (due to overloaded index server) nor a flat Chord overlay (due to overloaded GPRS peers) is an applicable solution for the given system, thus only a hierarchical architecture fulfils the system requirements.

The optimal number of superpeers for the given system lies between 13,775 (point A) and 30,000 (point B). As we can see from figures 5 and 6, at point A the total network costs are minimized without overloading any participating peer in the system. At point B the *HLF* is minimized. In this case all DSL peers act as superpeers with a load factor of 51%, while all UMTS and GPRS peers are leafnodes and therefore have much lower load factors.

³ We focus on the upstream bit rate of peers here because this is mainly the bottleneck in today's overlay networks, e.g. due to an asymmetric down- and upstream in DSL.

5. On Distributed Algorithms to Determine Optimal Superpeer Ratios

There are two instances of the problem of distributed determining optimal operating points of a network. How can an optimal point be reached from any operating state and how can it be maintained once reached? Answering these questions in detail is an important item of our future work, here we just hint on a possible answer to the second question in order to back our belief that distributed implementations of the presented ideas are feasible.

Under the assumption of sequential peer arrivals an optimal superpeer ratio can be maintained roughly as follows. When a new peer joins the network it can communicate its cost limit to the superpeer through which it is about to join. The superpeer is responsible for deciding which role the arriving peer should take and whether it should replace one or more superpeers that are already in the network. To be able to do this, all superpeers need to maintain approximations of the current state of the network. In particular, good approximations of the sizes of the superpeer population and the entire network as well as an estimate of the load distribution across peers are necessary. We believe that the DHT size estimation from [11] can be extended and applied to our problem. On the other hand, it seems to be hard to discover an accurate estimate of the load distribution. We wonder whether it suffices to maintain samples made up of routing neighbors only or larger sets are necessary. In case they are needed, these sets can be spread among neighbors, either by broadcasting them periodically or piggybacking them to other exchanged messages.

6. Conclusion and Future Work

We showed in this paper that hierarchical DHT organizations provide a plausible approach to building P2P overlay networks. We presented an analytical framework to analyze a specific type of hierarchical systems, where superpeers build a conventional Chord overlay and leafnodes use them as proxies. We evaluated the costs for running the whole network as well as the costs for every single participant, in order to determine an optimal superpeer ratio for a given system. We found that total network costs decrease with centralization; while on the other hand, centralization may overload peers and therefore endanger system stability. As our main results, we showed that hierarchical DHT design is better than flat design and that there is a trade-off between minimizing total network costs and

minimizing the costs for the highest loaded peer in the system.

In our future work we plan to extend our analytical framework to other hierarchical DHT designs, in order to compare them to the system architecture proposed in this paper. We also intend to evaluate the impact of churn on our analysis. Further, we are currently working on distributed algorithms to balance leafnodes and shared objects uniformly over superpeers, and to maintain optimal superpeer ratios.

References

- [1] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", ACM SIGCOMM Conference, 2001.
- [2] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems", IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001.
- [3] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric", International Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.
- [4] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Puceva, and R. Schmidt, "P-Grid: A Self-organizing Structured P2P System", SIGMOD Record, vol. 32, 2003.
- [5] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in G Major: Designing DHTs with Hierarchical Structure", International Conference on Distributed Computing Systems (ICDCS 2004), 2004.
- [6] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller, "Hierarchical P2P Systems", ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par), 2003.
- [7] S. Zoels, S. Schubert, W. Kellerer, and Z. Despotovic, "Hybrid DHT Design for Mobile Environments", International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2006), 2006.
- [8] N. Christin and J. Chuang, "A Cost-Based Analysis of Overlay Routing Geometries", IEEE INFOCOM'05, 2005.
- [9] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowcz, "Handling Churn in a DHT", USENIX 2004 Annual Technical Conference, 2004.
- [10] P. K. Gummadi, R. Gummadi, S. D. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The Impact of DHT Routing Geometry on Resilience and Proximity", ACM SIGCOMM Conference, 2003.
- [11] G. S. Manku, "Routing Networks for Distributed Hash Tables", 22nd ACM Symposium on Principles of Distributed Computing (PODC), 2003.